Nouvelles démarches de réduction de modèles pour le traitement des problèmes à très grand nombre de paramètres

Charles Paillet

université PARIS-SACLAY

école normale supérieure paris-saclay

NNT : 2019SACLN015

Thèse de doctorat

Nouvelles démarches de réduction de modèles pour le traitement des problèmes à très grand nombre de paramètres

Thèse de doctorat de l'Université Paris-Saclay
préparée à École Normale Supérieure Paris-Saclay

Ecole doctorale n°579 Sciences Mécaniques et Énergétiques, Matériaux et Géosciences (SMEMaG)
Spécialité de doctorat : Mécanique - Génie Mécanique - Génie Civil

Thèse présentée et soutenue à Cachan, le 24 Juin 2019, par

CHARLES PAILLET

Composition du Jury :

Francisco Chinesta Professeur, Arts et Métiers ParisTech (Procédés et Ingénierie en Mécanique et Matériaux)	Président
David Dureisseix Professeur, Institut National des Sciences Appliquées de Lyon (Laboratoire de Mécanique des Contacts et des Structures)	Rapporteur
Anthony Nouy Professeur, École Centrale de Nantes (Laboratoire de Mathématiques Jean Leray)	Rapporteur
Virginie Ehrlacher Maître de conférences, École des Ponts ParisTech (Centre d'Enseignement et de Recherche en MathématIques et Calcul Scientifique)	Examinatrice
Hachmi Ben Dhia Professeur, CentraleSupélec (Laboratoire de Mécanique des Sols, Structures et Matériaux)	Examinateur
David Néron Professeur, École Normale Supérieur Paris-Saclay (Laboratoire de Mécanique et Technologie)	Directeur de thèse
Pierre Ladevèze Professeur, École Normale Supérieur Paris-Saclay (Laboratoire de Mécanique et Technologie)	Co-directeur de thèse

3 ans déjà ! Cette thèse est passée très vite et même si je quitte volontairement le monde de la recherche, j'en garderai un souvenir très positif. Mes remerciements sont l'occasion de jeter quelques fleurs aux gens géniaux qui m'ont accompagné dans cette aventure.

Merci tout d'abord aux membres de mon jury d'avoir su se rendre disponibles en juin à cette période de l'année si chargée, en particulier à mes rapporteurs, Anthony Nouy et David Dureisseix pour leur relecture attentive de ce manuscrit et pour les questions soulevées. Merci à Hachmi Ben Dhia et à Virginie Ehrlacher pour leur participation bienveillante à mon jury ainsi qu'à Paco Chinesta qui l'a présidé avec enthousiasme et humour.

J'ai eu la chance durant cette thèse d'être magistralement encadré par Pierre Ladevèze et David Néron, merci à tous deux de m'avoir fait confiance et d'avoir cru en mes capacités. Pierre, passionné et bienveillant, a toujours été très disponible pour m'aider à décrypter ses algorithmes les plus fous, dont vous trouverez certains spécimens dans le présent manuscrit. David, encadrant chaleureux, a su guider mes premiers pas dans le monde de la recherche et j'ai eu le plaisir supplémentaire d'être son collègue auprès des étudiants de l'ENS.

Laboratoire accueillant et dynamique, le LMT a été un endroit rêvé où travailler pendant trois ans. Impossible de citer tout le monde ici, encore un coup de la « Malédiction de la dimentionnalité » ... Merci à nos chercheurs qui savent rester disponibles pour leurs jeunes et à toute l'équipe - techniciens, ingénieurs, gestionnaires, agents - aux petits soins pour que nos thèses se déroulent au mieux. Je tiens aussi à remercier les collègues du DGM avec qui j'ai eu mes premières belles expériences d'enseignement.

J'ai pu m'intégrer à un groupe de doctorants formidables. Merci en particulier à tous mes collègues de la 211 qui ont su maintenir une ambiance studieuse à coup de Nerf. On ne peut pas imaginer une meilleure promo ! Merci à mes co-bureau qui m'ont laissé le fauteuil le plus moelleux, ainsi qu'aux acheteurs de Gerbloss, organisateurs de barbecue, DJ du CDC ... qui font du LMT un lieu de vie convivial. En passant, je vais même remercier les affreux B3 du labo d'à coté.

Merci enfin à mes proches et amis qui ont su m'encourager et m'accompagner dans des projets parfois très éloignés de la PGD. Et du fond du cœur, merci à mon frère et à mes parents et qui m'ont soutenus dans mes choix ainsi qu'à Xu qui m'a supporté pendant toute la durée de cette thèse.

Saint-Maur-des-Fossés, 24 Juillet 2019

Table des matières

Symboles et notations

Seules les notations principales et répétées dans tout le manuscrit ont été retenues dans cette liste.

Notations Générales

$a_{i,(E)}^{k,(r)}$ Indices et exposants d'une fonction paramétrique :
- k : a apparait à la k-ème itération
- r : a appartient au mode r
- i : a est une fonction du paramètre μ_i
- E : a est une fonction associée au sous-domaine Ω_E

$A_E^{k,(r)}$ Indices et exposants d'une fonction locale en espace :
- k : A apparait à la k-ème itération
- r : A appartient au mode r
- E : A_E est une fonction définie spatialement sur le sous-domaine Ω_E

$A_{(E)|E'}$ Indices d'une fonction globale en espace :
- E : A est associée au sous domaine Ω_E de part son origine par exemple.
- E' : A peut être restreinte à sa valeur sur le sous-domaine $\Omega_{E'}$

$\widetilde{\bullet}$ Mode spatial associé à un champ donné

$\overline{\bullet}_E$ Grandeur associée à l'ensemble tous les $E' \in \mathbf{E}$ sauf E

Domaine Spatial

$\Omega \subset \mathbb{R}^3$ Domaine physique tridimensionnel étudié

N_p Nombre de sous-domaines (et donc de paramètres)

$\mathbf{E}$ Ensemble des index paramétriques : $\mathbf{E} = \{1 \dots N_p\}$

Ω_E Sous domaine physiques : $\Omega = \{\Omega_E\}_{E \in \mathbf{E}}$

$\mathscr{U} = [H^1(\Omega)]^3$ Espace de définition des champs de déplacement

$\mathscr{U}^{ad}$ Espace de définition des champs de déplacement admissibles

$\underline{u}$ Solution complète du problème en déplacement : $\underline{u} \in \mathscr{U}^{ad}$

$\mathscr{U}^h$ Sous-espace de dimension finie de $\mathscr{U}^{ad}$

d Nombre de degrés de liberté de $\mathscr{U}^h$

$\underline{\widetilde{X}}$ ou $\underline{u}^h$ Champ de déplacement continu de $\mathscr{U}^h$

$\underline{\widetilde{\mathrm{X}}}$ Vecteur des coefficients de $\underline{\widetilde{X}}$

$\underline{\widetilde{\mathrm{X}}}_E$ Restriction de $\underline{\widetilde{\mathrm{X}}}$ sur les degrés de liberté de Ω_E

d_E Nombre de degrés de liberté de la restriction $\underline{\widetilde{\mathrm{X}}}_E$

$\boldsymbol{\sigma}$ Solution en contrainte associée à $\underline{u}$

$\boldsymbol{\varepsilon}$ Solution en déformation associée à $\underline{u}$

Domaine Paramétrique

μ_E Paramètre associé au sous domaine Ω_E

$\underline{\mu}_E$ Vecteur paramétrique, discrétisation de μ_E

n_E Taille du vecteur $\underline{\mu}_E$

$\boldsymbol{\mu}$ Ensemble des paramètres, $\boldsymbol{\mu} = \{\mu_E\}_{E\in\mathbf{E}}$

$\boldsymbol{\mu}^{(i)}$ Un jeu particulier de paramètres

$\boldsymbol{\Sigma}_\mu$ Espace de définition de l'ensemble des paramètres

$\boldsymbol{\Sigma}_\mu$ Discrétisation de l'espace $\boldsymbol{\Sigma}_\mu$

$\mathscr{S}_\mu$ Ensemble des fonctions de $\boldsymbol{\Sigma}_\mu$ dans $\mathbb{R}$

$\boldsymbol{I}_k$ Ensemble de définition du paramètres μ_k, $\bigotimes_{k=1}^{N_p} \boldsymbol{I}_k = \boldsymbol{\Sigma}_\mu$

$\mathscr{I}_k$ Ensemble de fonctions de $\boldsymbol{I}$ dans $\mathbb{R}$

$\mathbf{I}_k$ Discrétisation de $\boldsymbol{I}_k$ sur n_k points, $\mathbf{I}_k = \mathbb{R}^{n_k}$ et $\bigotimes_{k=1}^{N_p} \mathbf{I}_k = \boldsymbol{\Sigma}_\mu$

γ_i Fonction continue du paramètre μ_i

$\underline{\boldsymbol{\gamma}}_i$ Discrétisation de γ_i

$\boldsymbol{X}(\boldsymbol{\mu})$ Fonction multiparamétrique (continue) de $\boldsymbol{\Sigma}_\mu$ dans $\mathbb{R}$

$\mathbf{X}$ Tenseur associé à la discrétisation de $\boldsymbol{X}$ sur $\boldsymbol{\Sigma}_\mu$

Domaine Produit Espace-Paramètres

$\mathscr{X}$ Ensemble des fonctions de carré intégrable de $\boldsymbol{\Sigma}_\mu$ dans $\mathscr{U}$

$\mathcal{X}$ Ensemble des fonctions de de carré intégrable de $\boldsymbol{\Sigma}_\mu$ dans $\mathbb{R}^d$

$\underline{\mathbf{X}}$ Tenseur espace+paramètres

Données du Problème

$\boldsymbol{H}$ ou $\boldsymbol{H}(\boldsymbol{\mu})$ Loi de Hooke globale

$\boldsymbol{H}_E$ ou $\boldsymbol{H}_E(\boldsymbol{\mu}_E)$ Loi de Hooke locale sur le sous-domaine Ω_E

$\boldsymbol{H}^0$ Valeur moyenne de la loi de Hooke

ϵ Coefficient de variation paramétrique de la loi $\boldsymbol{H}_E = \boldsymbol{H}^0(1+\epsilon\boldsymbol{\mu}_E)$

$\underline{f}_d$ Champ d'effort donné du problème modèle

$\partial_1\Omega$ Portion de la frontière de Ω

$\underline{u}_d$ Déplacement donné, imposé sur $\partial_1\Omega$

$\partial_2\Omega$ Portion de la frontière de Ω

$\underline{F}_d$ Effort extérieur donné et appliqué sur $\partial_2\Omega$

$\underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})$ Opérateur de rigidité après discrétisation spatiale

$\underline{\underline{\mathbf{K}}}^0$ Opérateur de rigidité associé à la valeur moyenne des paramètres

Algorithmes PGD et PM-PGD

$\widehat{\mathbf{C}}_E$ Voisinage de Ω_E associé à des fonctions micro

$\mathbf{R}$ Erreur liée au non-respect de la loi de comportement

$\mathscr{E}$ Estimateur d'erreur, normalisation de $\mathbf{R}$ par rapport à la contrainte moyenne

$\mathscr{E}_{mM}$ Estimateur d'erreur prenant en compte l'approximation liée au format multiéchelle

$\mathscr{E}_{loc}$ Estimateur d'erreur local dans l'espace paramétrique

Introduction

« Et si ... ? »

Curiosités, conjectures, hypothèses sages ou déraisonnables sont souvent à l'origine des découvertes scientifiques et bien souvent les innovations techniques sont d'abord les fruits de l'imagination de l'homme. Les inventeurs s'interrogent sur les limites du monde réel. Leur créativité leur permet de les dépasser. Et si ... ? Ce questionnement est au cœur du travail de conception des ingénieurs : « Et si on modifie cette forme, si on change ce matériau ? ». Pour répondre à ces questions, on peut passer aujourd'hui par l'étape de la simulation numérique qui permet de s'affranchir du coût et du temps nécessaire à l'expérimentation. Ce type de simulations est devenu un outil indispensable dans de multiples domaines et permet d'accélérer les phases de développement. "What if ... ?" La recherche en *computational mechanics* est très active et les méthodes proposées sont nombreuses, variées et en constante évolution. Aujourd'hui associés à des techniques d'optimisation, les outils informatiques sont utilisés pour effectuer des choix : « Quelle est la forme optimale ? Quel matériau est le plus adapté ? ». Les ambitions des inventeurs d'aujourd'hui et de demain sont et seront concrétisées par les ordinateurs.

L'accroissement des ressources informatiques au cours des dernières décennies permet d'aller dans ce sens et l'évolution des simulations numériques suit de près l'évolution des moyens de calcul. On dispose aujourd'hui de modèles de plus en plus complexes, efficaces pour décrire une large variété de phénomènes. Les résultats obtenus décrivent fidèlement les comportements observés et les erreurs effectuées peuvent être contrôlées. Cependant, en appliquant ces modèles à des problèmes toujours plus riches et détaillés, les limites des capacités informatiques sont rapidement atteintes dans un contexte industriel où on accepte traditionnellement de laisser tourner un calcul le temps d'une nuit.

Par ailleurs, certaines procédures telles que les algorithmes d'optimisation ne nécessitent pas les résultats d'une unique simulation mais d'un grand nombre de tests différents. Et si on disposait rapidement et simplement de toutes les solutions possibles dans un espace de conception donné ? On peut aujourd'hui créer des outils capables de telles résolutions grâce aux méthodes de réduction de modèle. Leur principe de construction consiste à effectuer des calculs en deux temps (voir FIG.1). Tout d'abord, une phase de calcul « offline » en amont de l'utilisation du modèle réduit permet de le mettre en place. Son élaboration peut être très coûteuse et son stockage nécessite l'utilisation de formats spécifiques. Dans un second temps, on peut utiliser ces

informations lors de la phase « online » : des opérations rapides de post-traitement ou de lecture des données permettent d'obtenir des solutions particulières du problème.

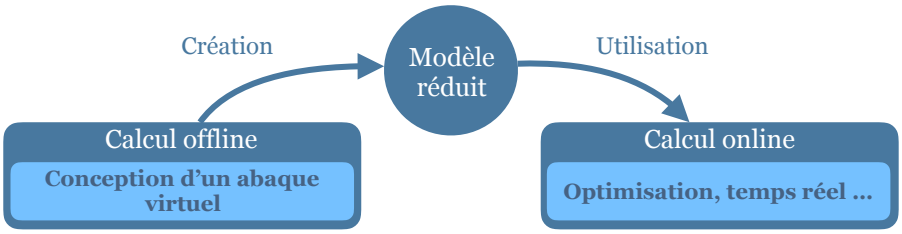


FIGURE 1 : Approche en deux temps de la réduction de modèle

Pour être utilisable, un bon modèle réduit doit posséder les caractéristiques suivantes :

— Précision : la compression des données ou la simplification de la résolution génère des erreurs qui doivent être limitées et contrôlées.
— Gain de temps online : il doit être conséquent, une utilisation en temps réel des solutions ainsi calculées peut souvent être envisagée.
— Temps de calcul offline : il doit rester raisonnable, mais peut profiter des moyens du calcul hautes performances (HPC).
— Légèreté d'utilisation : une utilisation sur des ordinateurs de bureau ou des outils mobiles doit être possible.
— Richesse : le domaine de validité du modèle réduit doit être suffisamment large pour obtenir des solutions à des problèmes variés et pertinents.

De nombreuses stratégies de réduction ont été développées, toutes basées sur une idée simple : on souhaite résoudre un grand nombre de problèmes mécaniques de même nature qui ne diffèrent que par un nombre donné de paramètres. Il y a donc des points communs d'origine physique entre toutes ces solutions. Il s'agit d'en tirer parti et de profiter notamment de la redondance des informations présentes dans chaque solution. Une première famille de méthodes telles que la POD (Proper Orthogonal Decomposition [Chatterjee, 2000]) ou les Reduced Basis (RB, [Patera *et al.*, 2007]) permettent à partir d'un ensemble de solutions particulières appelées *snapshots* de construire une base réduite adaptée à la résolution d'un problème donné (calcul offline). Cette base qui concentre la physique du problème est de petite taille : une résolution dans celle-ci sera rapide et efficace (calcul online). D'autres méthodes telles que la PGD (Proper Generalized Decomposition, [Ammar *et al.*, 2007]) par exemple permettent d'aller plus loin et de stocker toutes les solutions en incluant les dépendances paramétriques dans le modèle réduit. Celui-ci peut alors être considéré comme un abaque virtuel [Courard *et al.*, 2016] (ou *computational vademecum* [Chinesta *et al.*, 2013b]) dans lequel il suffira d'aller chercher les solutions nécessaires d'un problème donné.

Ces abaques virtuels sont de puissantes aides à la conception. Comme illustré FIG.2, des interfaces graphiques permettent en temps réel de sélectionner un jeu de paramètres et de visualiser la réponse associée. Tous les « Et si ...? » trouvent une réponse immédiate dans le domaine de validité du modèle réduit.

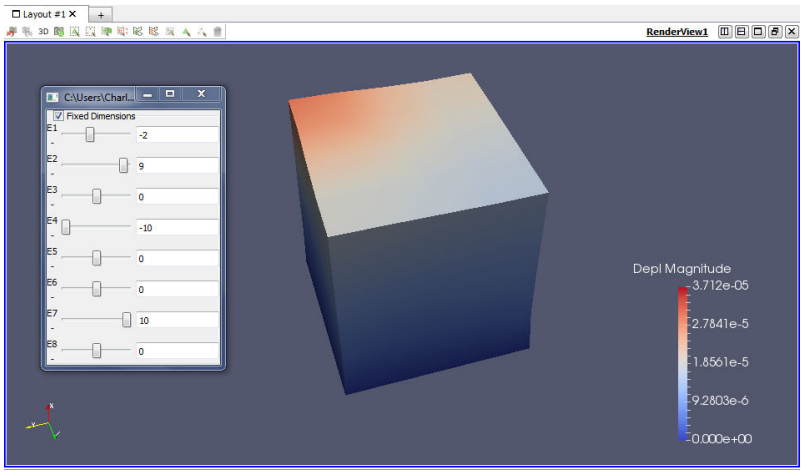


FIGURE 2 : Capture d'écran : logiciel ParaView et extension PXDMF [Bordeu, 2013] : à gauche, un ensemble de curseurs permet de modifier manuellement 8 paramètres, associés à des rigidités locales. La solution spatiale tridimensionnelle correspondante s'affiche en temps réel.

De telles « bibliothèques de solutions » peuvent être conçues à partir des bases réduites issues de la POD ou des RB en les interpolant dans un espace paramétrique. On peut aussi choisir de considérer les paramètres du problème comme des coordonnées supplémentaires [Chinesta *et al.*, 2013a] au même titre que le temps et l'espace et chercher à résoudre un problème en grande dimension. C'est la démarche des méthodes telles que la PGD qui a servi de base aux présents travaux. En notant D la dimension spatiale du problème ($D = 1$, 2 ou 3), t sa dimension temporelle ($T = 0$ (statique) ou $T = 1$ (dynamique)) et N_p le nombre de paramètres considérés, on cherche donc à résoudre un problème à $N_d = D + T + N_p$ dimensions. L'espace des solutions lui-même est alors de très grande taille et dès que le nombre de paramètres N_p est trop important, on se heurte à la « malédiction de la dimensionalité ». En supposant que chaque paramètre est discrétisé sur un nombre n_p de degrés de liberté, le nombre d'éléments de l'espace paramétrique uniquement est $n_p^{N_P}$ et augmente exponentiellement avec le nombre de paramètres. Il atteint très vite les limites des capacités de stockage classiques des ordinateurs.

Des stratégies de représentation compacte des solutions doivent donc être mises en place pour construire les abaques virtuels des problèmes paramétriques, si possible en association avec les méthodes de réduction de modèle choisies. Pour la méthode PGD par exemple, on choisit de représenter les solutions comme des sommes de

produits de fonctions à variables séparées, chacun des termes du modèle réduit étant directement généré sous cette forme lors de la phase de calcul offline.

Mathématiquement, les données discrètes en grandes dimensions sont représentées par des tenseurs. Ceux-ci ont fait l'objet de développements spécifiques et de nombreuses méthodes existent pour les manipuler et les stocker lorsque le nombre de dimensions augmente [Hackbusch, 2012]. Cependant, ces formats tensoriels sont génériques et notre connaissance mécanique des problèmes traités nous permet de choisir des formats plus pertinents. En particulier, le Principe de Saint-Venant est à la base d'un nouveau format de stockage utilisé dans cette thèse et conçu pour s'adapter aux paramètres localisés spatialement [Ladevèze *et al.*, 2018].

Historiquement, la PGD a été conçue comme un des éléments du solveur LaTIn destiné à résoudre des problèmes mécaniques non-linéaires [Ladevèze, 1985, Ladevèze, 1999]. En elle-même, elle permet de construire des modèles réduits pour de nombreux types de paramètres : chargements [Niroomandi *et al.*,], coefficients matériaux [Vitse *et al.*, 2014], géométrie du problème [Courard *et al.*, 2016] ... Ces exemples ainsi que beaucoup d'autres [Ladevèze, 2016] illustrent les qualités de la PGD pour résoudre des problèmes non-linéaires. Ces non-linéarités ne sont cependant pas l'objet de cette thèse. Elles sont généralement traitées en résolvant une succession de problèmes linéaires et on s'intéressera exclusivement à de tels problèmes. On introduira cependant une difficulté supplémentaire : le nombre de paramètres considérés.

On trouve dans la littérature des cas d'intérêt industriel prenant en compte jusqu'à une vingtaine de paramètres générés via les méthodes POD (par exemple dans [Bui-Thanh *et al.*, 2008]) ou PGD [Marchand *et al.*, 2016]. Ce nombre correspond en pratique à la limite haute de la richesse des modèles réduits calculables, le principal verrou scientifique étant la convergence des algorithmes traditionnels [Ladevèze *et al.*, 2018] qui s'effondre avec le nombre de paramètres. Le but de cette thèse est de s'attaquer à cette limite pour la faire reculer.

Objectif

Construire des modèles réduits paramétriques prenant en compte un grand nombre de paramètres

En pratique, les nouveaux algorithmes décrits dans le présent manuscrit sont capables de prendre en compte jusqu'à un millier de paramètres pour des problèmes académiques et la centaine de paramètres est aisément atteinte pour des problèmes de grande taille, plus proches des besoins de l'industrie. On s'est en particulier intéressé à une classe de problèmes qui se prêtent bien à cette approche : les problèmes à paramètres distribués spatialement, c'est à dire définis localement sur l'espace physique d'étude, leur grand nombre permettant de couvrir tout le domaine. Leur nature peut être très variée, l'exemple principalement traité dans cette thèse concerne des coefficients matériaux locaux.

Une démarche originale a été mise en œuvre pour résoudre de tels problèmes et a donné lieu à de nombreux questionnements. En particulier, deux procédures diffé-

rentes sont discutées dans ce manuscrit correspondant à deux difficultés majeures. D'une part, les formats de donnés spécifiques conçus pour nos problèmes à partir d'observations mécaniques amènent naturellement à représenter les solutions spatiales de manière discontinue. De telles discrétisations accessibles par exemple via les méthodes de Galerkin Discontinues (Discontinuous Galerkin, DG) [Arnold *et al.*, 2002] sont principalement utilisées industriellement dans le domaine de la mécanique des fluides, mais elles ont été adaptées aux problèmes solides [Rivière, 2008]. Un grand nombre de méthodes existent et cette thèse a été l'occasion d'implémenter pour la première fois une formulation discontinue riche en contenu mécanique : la méthode de Weak Trefftz Discontinuous Galerkin (WTDG) introduite théoriquement dans [Ladevèze, 2011].

D'autre part, les procédures de réduction de modèle doivent permettent de résoudre des problèmes mécaniques variés et d'intérêt industriel. Leur couplage avec des solveurs classiques a donc été étudié et des algorithmes compatibles avec les éléments finis (EF) donnent des résultats aussi bons que ceux associés aux discrétisations discontinues, permettant de résoudre des problèmes de géométries complexes.

Les travaux réalisés pendant cette thèse sont structurés en quatre parties :

- Un état de l'art de la réduction de modèle linéaire multiparamétrique est présenté dans le chapitre 1. Les méthodes de réduction classiques dans notre domaine sont décrites et leurs limites pour les grands nombres de paramètres illustrées. Une discussion autour des formats de données disponibles permet d'aboutir à la construction et justification d'une représentation adaptée aux problèmes mécaniques.
- Le chapitre 2 est consacré à un premier algorithme de réduction de modèle permettant de prendre en compte jusqu'à mille paramètres. Il est associé à une discrétisation spatiale discontinue (la méthode WTGD) que l'on introduira.
- Au chapitre 3, une adaptation de la démarche précédente est présentée pour rendre la procédure compatible avec les solveurs éléments finis. Des problèmes de géométries complexes et irrégulières sont résolus.
- L'utilisation pratique des modèles réduits construits précédemment est étudiée au chapitre 4 et leur performances quantifiées. En particulier, les abaques virtuels sont appliqués à des problèmes d'inversion.

Par ailleurs, deux annexes présentent des travaux originaux et adaptés aux problèmes de grandes dimensions mais sans lien direct avec les algorithmes principaux développés dans ce manuscrit. L'annexe A est consacrée à une version spécifique de la PGD développée à l'occasion du stage de Master préliminaire à cette thèse (l'Extended-PGD, [Paillet, 2016a]) et l'annexe B décrit une méthode d'interpolation utilisable en grande dimension.

Ce manuscrit reprend et développe l'ensemble des résultats publiés ou en cours de publication dans trois articles :

- [Ladevèze *et al.*, 2018] introduit pour la première fois le format spécifique exposé au chapitre 1 et l'applique à des problèmes monodimensionnels.
- [Paillet *et al.*, 2018] inclut tous les développements du chapitre 2 : l'introduction de la WTDG et l'algorithme de réduction associé.
- [Paillet *et al.*, 2019] *(en cours de publication)* donne la version de l'algorithme compatible avec les solveurs EF et reprend des résultats présentés aux chapitres 3 et 4.

Chapitre 1

Réduction de modèle pour les problèmes paramétriques linéaires

De nombreux formats de données permettent de surmonter la « malédiction de la dimensionnalité », les plus classiques sont introduits dans ce chapitre avant de discuter en détail de la construction d'une structure adaptée aux problèmes mécaniques à paramètres distribués. Ensuite, différents algorithmes classiques de réduction de modèle sont présentés dans le contexte multi-paramétrique. Est illustrée en particulier la PGD et ses limites pour les grands nombres de paramètres.

La représentation de champs à grand nombre de dimensions et la résolution d'équations aux dérivées partielles (EDP) à grands nombres de paramètres semblent être deux problématiques différentes. Nous montrons en préambule de ce chapitre que ces deux catégories de problèmes sont finalement très proches.

En effet, les différentes représentations de données possibles sont très liées à leur utilisation pour résoudre des EDP mécaniques grâce aux méthodes de réduction de modèle. Il est justifié de différencier le format retenu de la démarche de résolution pour des méthodes établies sur des bases réduites telles que la POD. Au contraire, la PGD par exemple ne peut se concevoir sans le format à variables séparées qui lui est associé.

Par ailleurs, les formats compacts de représentation de données en grandes dimensions ne sont généralement que des approximations de champs exacts. On a de fait deux problématiques identiques, on cherche à approcher un champ multi-

dimensionnel connu sous une forme explicite (approximation d'un champ donné) ou non (approximation d'une solution d'EDP).

Formellement, définissons un champ de grande dimension $\boldsymbol{X}(\boldsymbol{\mu})$. On cherche son approximation optimale $\boldsymbol{X}^{\mathscr{F}}(\boldsymbol{\mu})$ au sens d'une norme $|||\bullet|||$ en utilisant un format $\mathscr{F}$ plus compact grâce à la minimisation :

$$\min_{\boldsymbol{X}^{\mathscr{F}}} \left|\left|\left| \boldsymbol{X}(\boldsymbol{\mu}) - \boldsymbol{X}^{\mathscr{F}}(\boldsymbol{\mu}) \right|\right|\right| \tag{1.1}$$

Si on veut maintenant résoudre une EDP en grande dimension, ce qui correspond mathématiquement à la recherche du champ $\boldsymbol{X}$ solution de $\mathscr{M}(\boldsymbol{X}) = 0$, l'approximation $\boldsymbol{X}^{\mathscr{F}}$ de $\boldsymbol{X}$ au format $\mathscr{F}$ est donnée par :

$$\min_{\boldsymbol{X}^{\mathscr{F}}} \left|\left|\left| \mathscr{M}\left(\boldsymbol{X}^{\mathscr{F}}(\boldsymbol{\mu})\right) \right|\right|\right| \tag{1.2}$$

Formellement il s'agit simplement de minimiser une fonctionnelle potentiellement plus complexe. Résoudre une équation via un modèle réduit et représenter un champ connu de manière compacte sont donc deux problèmes de même nature. Les méthodes appliquées sont cependant différentes et l'étude particulière des formats utilisés s'appuie sur les outils de l'analyse tensorielle, vaste domaine des mathématiques appliquées qui mérite d'être discuté en détails. On séparera donc dans ce chapitre l'étude des formats de données de la présentation des méthodes de réduction de modèle.

Remarque 1 (Problèmes espace-paramètres) *Cette thèse traite de problèmes multi-paramétriques et les exemples présentés sont des problèmes de statique. Toutes les méthodes de réduction introduites dans ce chapitre sont traitées dans ce contexte bien que la majorité d'entre elles aient été à l'origine développées dans un cadre mono ou bi-dimensionnel afin notamment de résoudre des problèmes spatio-temporels.*

Remarque 2 (Grand nombre de dimensions) *On considère dans le présent manuscrit qu'un problème a un « grand nombre de dimensions » lorsqu'on atteint les limites liés à la « malédictions de la dimensionnalité » (voir partie 1.2), ce qui apparait en général entre 5 et 10 dimensions suivant la complexité des problèmes traités.*

1 « Malédiction de la dimensionnalité »

Pour concevoir des algorithmes de réduction de modèle à grands nombres de paramètres, il a fallu retenir un problème à mettre en œuvre numériquement qui permette de les tester. Deux critères principaux ont été retenus pour le choisir :

- La simplicité : le problème modèle doit être rapide à implémenter et doit permettre de mettre en avant les particularités et difficultés liées aux grands nombres de paramètres. Il doit être contrôlé et ne pas présenter de sources d'incertitudes qui masqueraient les caractéristiques que l'on veut isoler.

- La richesse : le problème modèle ne doit pas être un cas particulier dégénéré. Il doit permettre d'observer toutes les difficultés liées aux grands nombres de paramètres et bien représenter la diversité des situations qui peuvent se présenter.

C'est pour la première raison qu'on a choisi de retenir un problème linéaire de statique. Sans intégration temporelle et sans comportement non-linéaire, la seule source d'erreur est la discrétisation spatiale. On va donc considérer que la solution discrétisée est notre référence. C'est par rapport à celle-ci qu'on définira la précision des modèles réduits construits.

La recherche de problèmes suffisamment riches nous a poussé à choisir des paramètres matériaux locaux. En décomposant le domaine spatial Ω en sous-domaines Ω_E, chacun associé à un paramètre matériau $\boldsymbol{\mu}_E$, on peut augmenter le nombre de paramètres à volonté et donc la dimension de l'espace de résolution. Par ailleurs, le choix de paramètres locaux n'est pas une limitation en soi : un paramètre global peut toujours être décomposé localement sur un ensemble de paramètres distribués. C'est ce qui est fait grâce à des polynômes dans [Bachmayr *et al.*, 2017], ce qui augmente l'efficacité des méthodes de réduction étudiées. Enfin, en agissant directement sur la loi de comportement, les paramètres matériaux sont délicats à prendre en compte et imposent d'écrire des algorithmes généraux qui pourraient être adaptés facilement à des paramètres plus simples comme par exemple les conditions limites ou initiales.

1.1 Problème modèle

On s'intéresse à un milieu élastique linéaire occupant un domaine $\Omega \subset \mathbb{R}^3$ de bord $\partial\Omega = \partial_1\Omega \cup \partial_2\Omega$. Des conditions limites (CL) de Dirichlet s'appliquent sur $\partial_1\Omega$ et de Neumann sur $\partial_2\Omega$:

$$\begin{cases} \underline{u}(\boldsymbol{\mu}) \in \mathscr{X} \\ \underline{\mathrm{div}}(\boldsymbol{\sigma}) + \underline{f}_d = 0 \quad \text{sur} \quad \Omega \\ \underline{u} = \underline{u}_d \quad \text{sur} \quad \partial_1\Omega \\ \boldsymbol{\sigma}\underline{n} = \underline{F}_d \quad \text{sur} \quad \partial_2\Omega \\ \boldsymbol{\sigma} = \boldsymbol{H}(\boldsymbol{\mu})\varepsilon \\ \boldsymbol{\varepsilon} = \frac{1}{2}(\boldsymbol{\nabla}\underline{u} + \boldsymbol{\nabla}\underline{u}^T) \end{cases} \tag{1.3}$$

Les champs spatiaux sont définis dans $\mathscr{U} = [H^1(\Omega)]^3$ et $\boldsymbol{\Sigma}_\mu$ est l'espace paramétrique. On définit $\mathscr{X}$ comme l'ensemble des fonctions de $\boldsymbol{\Sigma}_\mu$ dans $\mathscr{U}$ telles que :

$$\mathscr{X} = \left\{ \underline{u} : \boldsymbol{\Sigma}_\mu \to \mathscr{U} \,\middle|\, \int_{\boldsymbol{\Sigma}_\mu} |||\underline{u}|||^2 d\boldsymbol{\mu} < \infty \right\} \tag{1.4}$$

où $|||\bullet|||$ est une norme énergétique sur $\mathscr{U}$ discutée en détails en 2.3. Le domaine spatial Ω est divisé en sous-domaines $\Omega_E, E \in \mathbf{E}$. Chaque paramètre $\boldsymbol{\mu}_E$ est associé à la rigidité d'un Ω_E ce qui permet de définir $\boldsymbol{\mu} = \{\boldsymbol{\mu}_E\}_{E\in\mathbf{E}}$. Ainsi la relation de comportement choisie dans (1.3), la loi de Hooke $\boldsymbol{H}$, dépend des paramètres $\boldsymbol{\mu}$ et est donnée localement par sa valeur $\boldsymbol{H}_E(\boldsymbol{\mu}_E)$. Un exemple d'un tel problème est présenté FIG.1.1.

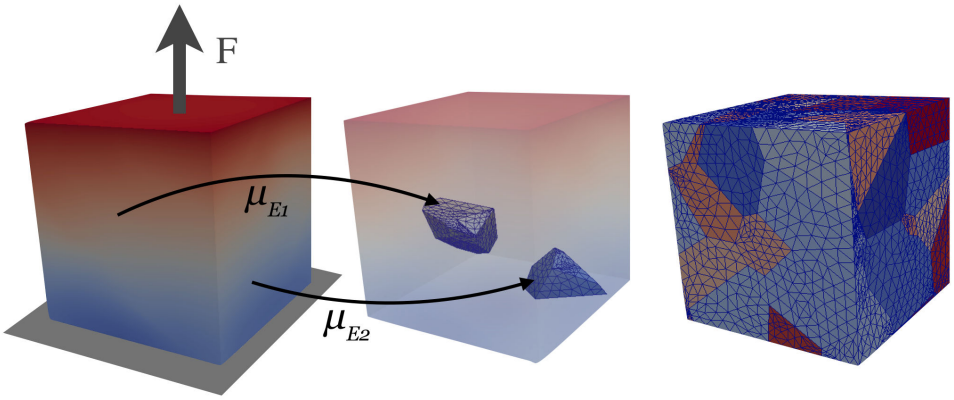


FIGURE 1.1 : Cube encastré en traction : problème modèle ; une solution particulière ; mise en valeur de deux sous-domaines Ω_{E_1} et Ω_{E_2} ; maillage et décomposition du cube.

En pratique, on choisit de n'associer à chaque loi locale qu'un seul paramètre qui peut être interprété comme une variation du module d'Young $\boldsymbol{H}_E = \boldsymbol{H}^0 f_E(\mu_E)$ où $\boldsymbol{H}^0$ est la valeur moyenne de la loi et f_E une fonction paramétrique. En particulier, on a ici un paramètre linéaire et on exprime sa valeur autour de la moyenne suivant la formulation :

$$\boldsymbol{H}_E = \boldsymbol{H}^0(1+\epsilon\mu_E) \tag{1.5}$$

où ϵ est l'autre coefficient de cette expression. Cette formulation peut aussi être interprétée comme un endommagement variable localement. On pourrait très facilement généraliser les algorithmes présentés avec plus d'un paramètre par sous-domaine.

Remarque 3 (Domaine paramétrique) *On choisira en particulier* $\mu_E = \left[-\frac{1}{2}, \frac{1}{2}\right]$. *Ce choix permet de garder un domaine de définition du paramètre de longueur 1 afin que la norme* L^2 *d'un paramètre constant égal à 1 reste égale à 1, ce qui s'avère très utile lorsque le nombre de paramètres augmente. On a donc* $\boldsymbol{\Sigma}_\mu = \left[-\frac{1}{2}, \frac{1}{2}\right]^{N_p}$ *et l'amplitude des variations autour de la moyenne est contrôlée par le paramètre* ϵ.

Remarque 4 (Variation de ϵ) *On ne choisira que des valeurs* $\epsilon \in]0,2[$ *pour rester dans un cas physique (*$\epsilon < 2$*, pas de module d'Young négatif ou nul) et non trivial (*$\epsilon \neq 0$*).*

1.2 Dimension de la solution

Numériquement, il est nécessaire de discrétiser les espaces $\mathscr{U}$ et $\boldsymbol{\Sigma}_\mu$. On note d le nombre de degrés de liberté de l'espace $\mathscr{U}^h$, sous-espace de $\mathscr{U}$ défini via une méthode d'approximation spatiale, et n_E le nombre de degrés de liberté associés à chaque dimension paramétrique. En supposant que chaque paramètre est discrétisé avec la même résolution, le nombre de coefficients de l'espace de solution complet $\mathscr{U}^h \otimes \boldsymbol{\Sigma}_\mu$ avec $\boldsymbol{\Sigma}_\mu = (\mathbb{R}^{n_E})^{N_p}$ est alors $d.n_E^{N_p}$. Il augmente exponentiellement avec le nombre de

paramètres et atteint très vite une valeur trop importante pour la résolution du problème mais aussi pour la manipulation ou le stockage de ces données. C'est cette augmentation exponentielle de la taille des espaces de solutions que l'on nomme « malédiction de la dimensionnalité ». Elle peut être aisément surmontée en stockant et en manipulant des données formatées intelligemment.

2 Formats de données en grandes dimensions

2.1 Séparation de variables et Analyse en Composantes Principales

La séparation de variables est une méthode simple pour décrire des champs de grande dimension. Par exemple, la solution du problème modèle (1.3) peut être recherchée sous la forme :

$$\underline{u}(\boldsymbol{\mu}) = \sum_{k=1}^{M} \left(\underline{\widetilde{u}}^{(k)} \prod_{E \in \mathbf{E}} \gamma_E^{(k)}(\mu_E) \right) \tag{1.6}$$

Cette solution est composée de M modes, chaque mode k étant un produit de fonctions chacune associée à une des variables du problème. Chacun d'entre eux comporte donc un mode spatial $\underline{\widetilde{u}}^{(k)}$ et une fonction paramétrique $\gamma_E^{(k)}$ par paramètre $E \in \mathbf{E}$.

Cette formulation a un intérêt si les contributions élémentaires de la solution sont peu couplées et donc facilement séparables. Certains problèmes se séparent très mal, les problèmes de transport ou de convection en dynamique par exemple. En effet, des équations telles que les équations d'ondes couplent fortement l'espace et le temps. Des changements de variables peuvent cependant rendre ces systèmes séparables [Cagniart *et al.*, 2019].

En supposant que l'on se contente d'une approximation de la solution sur M modes, le nombre de données à stocker est alors réduit à $M.(d + N_p.n_E)$. Il est non seulement bien inférieur à celui de l'ensemble complet de toutes les solutions possibles pour des nombres raisonnables de modes mais en plus il n'augmente plus que linéairement en fonction du nombre de paramètres.

Historiquement ce type de représentations a été introduit en dimension 2 grâce notamment aux outils de l'analyse matricielle qui permettent via une SVD (Singular Value Decomposition) de séparer deux dimensions [Stewart, 1993]. A plus de deux dimensions, ces outils peuvent toujours être utilisés, mais en donnant plus d'importance à une des variables. On choisit souvent l'espace qui a le plus de degrés de liberté et les variations les plus complexes. On s'intéressera dans la suite de cette partie uniquement à des champs représentés de manière discontinue.

L'analyse en composantes principales (PCA, Principal Componant Analysis) [Jolliffe, 2011] permet de trouver une base réduite optimale pour décrire un ensemble de fonctions données. On suppose qu'on connait N_s fonctions spatiales $\underline{u}^{(i)}$, $i \in [\![1, N_s]\!]$. Ces champs peuvent par exemple être des « snapshots », c'est à dire des solutions du problème pour un pas de temps donné (d'où le terme *snapshot*) ou pour un certain jeu de paramètres. La méthode consiste à extraire de ces solutions $\underline{u}^{(i)}$ des informa-

tions pertinentes et peu nombreuses qui suffisent à décrire l'ensemble de la famille $\{\underline{u}^{(i)}\}_{i\in[\![1,N_s]\!]}$ avec une précision suffisante. Une telle opération a deux applications principales. C'est d'abord un moyen de concentrer l'information contenue dans les N_s fonctions $\underline{u}^{(i)}$, ce qui permettra ensuite de les manipuler ou de les représenter aisément. D'autre part, on verra en partie 3 que c'est une base efficace pour décrire des fonctions de même nature, par exemple pour déterminer toute solution particulière $\underline{u}(\boldsymbol{\mu})$ via une méthode de réduction de modèle.

De manière discrète, on note $\underline{\underline{U}} = Col(\underline{u}^{(i)})_{i=1}^{N_s}$ la matrice composée en colonnes des vecteurs $\underline{u}^{(i)}$. Pour un problème à seulement deux dimensions, par exemple en espace et en temps, on peut construire une matrice $\underline{\underline{U}}$ représentant l'intégralité du champ à approximer.

Soit $\mathscr{U}^M$, un sous-espace de $\mathscr{U}$ de dimension M, soit $\{\boldsymbol{\Phi}\} = \{\underline{\phi}_1, \ldots, \underline{\phi}_M\}$ une base orthonormée de $\mathscr{U}^M$ et $\underline{\underline{\boldsymbol{\Phi}}} = Col(\underline{\phi}_k)_{k=1}^{M}$ la matrice associée. On cherche la meilleure base $\{\boldsymbol{\Phi}\}$ pour approximer les $\underline{u}^{(i)}$, on va pour cela minimiser la différence entre les $\underline{u}^{(i)}$ et leurs projections dans $\mathscr{U}^M$:

$$\min_{\{\boldsymbol{\Phi}\}} \frac{1}{2} \sum_{i=1}^{N_s} \left\| \underline{u}^{(i)} - \sum_{m=1}^{M} \left\langle \underline{u}^{(i)}, \underline{\phi}_m \right\rangle \underline{\phi}_m \right\|^2 \tag{1.7}$$

où $\langle \bullet, \bullet \rangle$ est un produit scalaire de $\mathscr{U}$ et $\|\bullet\|$ est la norme associée. On montre que cette minimisation est équivalente, via le principe de Rayleigh, à la résolution d'un problème aux valeurs propres. Les $\underline{\phi}_m$ sont les M premiers vecteurs propres de la matrice $\underline{\underline{U}}.\underline{\underline{U}}^T$. Pour tout $m \in [\![1, M]\!]$, on a $\underline{\underline{U}}.\underline{\underline{U}}^T\underline{\phi}_m = \lambda_m \underline{\phi}_m$ où les λ_m sont les valeurs propres associées aux $\underline{\phi}_m$.

La famille $\left\{\underline{\phi}_1, \ldots, \underline{\phi}_M\right\}$ est une base orthonormée et forme les composantes principales de $\underline{\underline{U}}$. Cette base est optimale au sens de la norme $\|\bullet\|$ pour représenter la famille des vecteurs $\{\underline{u}_i\}_{i\in[\![1,N_s]\!]}$. Une telle base peut aussi être obtenue grâce à la décomposition de Karhunen-Loève (KL, [Karhunen, 1947, Loève, 1955]) qui en dimension finie est exactement équivalente à la PCA [Liang *et al.*, 2002].

On peut rechercher directement les valeurs propres de la matrice (carrée) $\underline{\underline{U}}.\underline{\underline{U}}^T$ ou de manière équivalente calculer les valeurs singulières de $\underline{\underline{U}}$ en réalisant une SVD (Singular Value Decomposition). Cette méthode est la plus ancienne [Stewart, 1993] et sa robustesse en fait un des piliers de l'analyse matricielle. Elle consiste à décomposer $\underline{\underline{U}}$ sous la forme : $\underline{\underline{U}} = \underline{\underline{\mathbf{U}}}\,\underline{\underline{\Sigma}}\,\underline{\underline{\mathbf{V}}}$ ou $\underline{\underline{\mathbf{U}}}$ et $\underline{\underline{\mathbf{V}}}$ sont des matrices carrées orthogonales et $\underline{\underline{\Sigma}}$ une matrice diagonale non négative. On montre que dans ce cas, $\underline{\underline{\mathbf{U}}}$ est une matrice dont les colonnes sont les vecteurs propres de $\underline{\underline{U}}.\underline{\underline{U}}^T$. Les première colonnes de $\underline{\underline{\mathbf{U}}}$ permettent alors de définir la même base d'approximation que celle donnée par la PCA, par exemple sur M modes : $\underline{\underline{\mathbf{U}}}^M = Col(\underline{\phi}_k)_{k=1}^{M}$.

La construction d'une représentation sous forme de variables séparées peut être relativement coûteuse. Elle n'est rentable que si un nombre limité de modes permet de décrire la solution avec une précision suffisante. C'est en pratique le cas pour un grand nombre de solutions de problèmes physiques. On peut montrer mathématique-

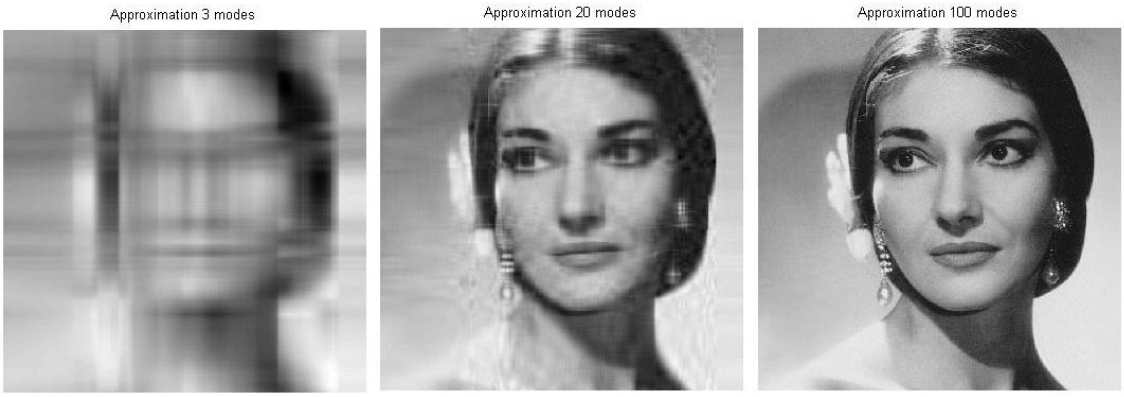


FIGURE 1.2 : Différentes troncatures de la SVD d'une image de 600 x 600 pixels

ment que grâce aux propriétés d'orthogonalité de la base $\{\mathbf{\Phi}\}$ l'erreur de projection est majorée via la relation :

$$\frac{\sum_{i=1}^{N_s} \left\| \underline{u}^{(i)} - \underline{u}_{SVD}^{(i)} \right\|^2}{\sum_{i=1}^{N_s} \left\| \underline{u}^{(i)} \right\|^2} = \frac{\sum_{m=M+1}^{N_s} \lambda_m}{\sum_{m=1}^{N_s} \lambda_m} \tag{1.8}$$

où $\underline{u}_{SVD}^{(i)}$ est l'approximation (c'est à dire la projection) de $\underline{u}^{(i)}$ dans la base $\{\mathbf{\Phi}\}$. Par conséquent, on en déduit que les approximations que l'on obtient en projetant les $\underline{u}^{(i)}$ sur la base propre sont d'autant plus précises que les valeurs singulières de $\underline{\underline{U}}$ décroissent vite. C'est effectivement le cas pour des ensembles de fonctions régulières notamment si celles-ci ont des formes proches.

Pour vérifier que des fonctions quelconques sont bien décrites par des approximations à variables séparées, on a extrait les modes propres de champs particuliers. Par exemple, les vecteurs composés par une colonne de pixels sur une image ont servi d'ensemble de fonctions $\underline{u}^{(i)}$. Après avoir extrait un certain nombre M de vecteurs propres $\underline{\phi}_k$, on projette les colonnes de pixels sur cette nouvelle base. Pour une image de 600×600 pixels, la solution obtenue est visuellement satisfaisante avec une centaine de modes environ, comme on peut l'observer FIG.1.2. Ce nombre est relativement élevé. Il s'agit cependant d'un champ particulièrement difficile à approximer, car comprenant de nombreuses discontinuités.

En mécanique des structures, la majorité des problèmes de statique ont des comportements beaucoup plus faciles à approcher même si il existe des exceptions telles que les structures endommageables (composites, bétons...). Elles ont des comportements très localisés, il faut donc potentiellement beaucoup de modes pour approximer ces solutions.

Remarque 5 *On peut choisir d'appeler la représentation obtenue une POD (Proper Or-*

thogonal Decomposition), comme dans [Liang et al.*, 2002]. Les méthodes citées précédemment (PCA, décomposition de Karhunen-Loève et SVD) sont trois moyens équivalents d'obtenir cette projection sur une base réduite.*

Cette représentation est donc une approximation beaucoup plus compacte de la famille $\{\underline{u}^{(i)}\}_{i\in[\![1,N_s]\!]}$. Il suffit de stocker la base $\{\mathbf{\Phi}\}$ et les coefficients $\underline{\alpha}^{(i)} = [\alpha_1^{(i)},\dots,\alpha_M^{(i)}]$ associés à chaque élément $\underline{u}^{(i)}$ pour pouvoir reconstruire chacun des champs initiaux :

$$\underline{u}^{(i)} \approx \underline{u}^{(i)}_{POD} = \sum_{k=1}^{M} \alpha_k^{(i)}\, \underline{\phi}_k \tag{1.9}$$

En réduction de modèle, cette base réduite va être utilisée comme base de résolution des problèmes traités avec la POD par exemple. Elle permet de concentrer l'information contenue dans $\underline{\underline{U}}$, champ bidimensionnel ou ensemble de snapshots.

A deux dimensions, ce format est un cas particulier de la formulation à variables séparées (1.6). En notant $\underline{\underline{\boldsymbol{\alpha}}} = Col(\underline{\alpha}^{(i)})_{i=1}^{N_s}$, on a $\underline{\underline{U}} \approx \underline{\underline{\mathbf{\Phi}}}\,\underline{\underline{\boldsymbol{\alpha}}} = \sum_{k=1}^{M} \underline{\phi}_k\, \underline{\alpha}_{(k)}$ où $\underline{\alpha}_{(k)}$ représente la k-ième ligne de $\underline{\underline{\boldsymbol{\alpha}}}$.

2.2 Tenseurs : formats génériques

Il existe plusieurs formats adaptés au stockage et à la manipulation des données discrétisées de dimensions supérieures à 2. Les tenseurs, généralisations à N dimensions des matrices, peuvent être considérés comme des tableaux de dimension N que l'on cherche dans cette partie à stocker de manière optimale.

L'analyse numérique des tenseurs (Numerical Tensor Analysis) est un vaste domaine des mathématiques appliquées né, dans les années 1940, de la recherche d'une généralisation en dimension supérieure à 3 du concept matriciel de SVD [Cattell, 1944, Tucker, 1966].

Les tenseurs de Tucker donnent la représentation compacte la plus proche de l'esprit d'une SVD mais ne permettent pas de s'affranchir de la « malédiction de la dimentionnalité ». Au contraire, la forme canonique d'un tenseur, basée sur l'hypothèse d'une séparation de variables, permet une forte compaction de données séparables. D'autres structures génériques plus récentes existent pour décrire des champs de grandes dimensions, elles sont brièvement décrites en partie 2.2.

Cette partie ne traite que des formats génériques, on ne s'intéresse donc qu'à des tenseurs quelconques et non plus à des solutions physiques du problème (1.3) par exemple. On considère un tenseur de dimension N potentiellement grande. Tout comme une matrice qui peut être la représentation discrétisée d'un champ à deux dimensions, le tenseur peut être interprété comme la discrétisation d'un champ multiparamétrique. Soit la fonction à N variables $\boldsymbol{X}$:

$$\boldsymbol{X} : \left\{ \begin{array}{ccc} \boldsymbol{\Sigma}_\mu & \rightarrow & \mathbb{R} \\ \boldsymbol{\mu} & \rightarrow & \boldsymbol{X}(\boldsymbol{\mu}) \end{array} \right. \tag{1.10}$$

On définit l'ensemble $\boldsymbol{\Sigma}_\mu = \boldsymbol{I}_1 \times \boldsymbol{I}_2 \times \cdots \times \boldsymbol{I}_N = \times_{k=1}^{N} \boldsymbol{I}_k$ de dimension N en notant $\boldsymbol{I}_k \subset \mathbb{R}$ l'ensemble associé à la k-ième dimension de $\boldsymbol{\Sigma}_\mu$. On définit de plus les ensembles fonctionnels associés $\mathscr{I}_k = \left\{ f : \boldsymbol{I}_k \to \mathbb{R} \;\middle|\; \int_{\boldsymbol{I}_k} f^2 d\mu_k < \infty \right\}$ et $\mathscr{S}_\mu = \left\{ \boldsymbol{X} : \boldsymbol{\Sigma}_\mu \to \mathbb{R} \;\middle|\; \int_{\boldsymbol{\Sigma}_\mu} \boldsymbol{X}^2 d\boldsymbol{\mu} < \infty \right\}$. Pour donner une représentation discrète de la fonction $\boldsymbol{X} \in \mathscr{S}_\mu$, on discrétise donc chacun des ensembles $\boldsymbol{I}_k$ sur n_k degrés de liberté : $\mathbf{I}_k = \{\mu_k(1), \ldots, \mu_k(n_k)\}$. Le tenseur $\mathbf{X}$ qui représente complètement le champ $\boldsymbol{X}(\boldsymbol{\mu})$ aux différents points de discrétisation est composé de $n_1 \times .... \times n_N$ composants qu'on notera :

$$\boldsymbol{X}\left(\mu_1(i_1), \mu_2(i_2), \ldots, \mu_N(i_N)\right) = \mathbf{X}(i_1, i_2, \ldots, i_N) = x_{i_1 i_2 \ldots i_N} \tag{1.11}$$

On s'intéresse en particulier dans cette thèse à des tenseurs de très grandes dimensions qu'il est impossible de stocker complètement sous la forme de la liste de leurs coefficients (1.11). De plus, les opérations nécessaires à la construction des approximations ne doivent pas non plus nécessiter la manipulation de tenseurs complets.

En analyse tensorielle, de nombreuses notations et opérations originales sont utilisées. Nous avons cherché dans ce manuscrit à minimiser l'introduction de ces notions spécifiques, nous utilisons cependant plusieurs fois le « produit du n-ième mode » d'un tenseur (n-mode product), noté $\times_n$. Appliqué à un vecteur $\underline{v}$ de taille n_n, il s'agit de réaliser son produit scalaire avec toutes les fibres du tenseur $\mathbf{X}$ dans la direction n. Une fibre est l'analogue tensoriel des lignes ou des colonnes dans une matrice, on les obtient en fixant tous les indices sauf un. Les fibres de direction n du tenseur $\mathbf{X}$ sont au nombre de $n_1 \times \ldots n_{n-1} \times n_{n+1} \times \ldots n_N$ et ont la forme : $\underline{f}_{i_1 i_2 \ldots i_{n-1} i_{n+1} \ldots i_N} = \mathbf{X}(i_1, i_2, \ldots i_{n-1}, [\![1, n_n]\!], i_{n+1}, \ldots, i_N)$. Le tenseur produit $\left(\mathbf{X} \times_n \underline{v}\right)$ est donc de dimension $N-1$.

On définit de la même manière le produit du n-ième mode d'un tenseur avec une matrice. Il s'agit là encore d'effectuer le produit de chacune des fibres de direction n, cette fois avec une matrice. Ainsi, pour une matrice $\underline{\underline{M}}$ de taille $n_n \times n_M$, le tenseur produit $\left(\mathbf{X} \times_n \underline{\underline{M}}\right)$ est de taille $n_1 \times \cdots \times n_{n-1} \times n_M \times n_{n+1} \cdots \times n_N$. Cette opération ne modifie pas la dimension finale du tenseur et l'ordre d'une succession de produits au n-ième mode peut être modifié sans influer le résultat. Ce n'est pas le cas du produit avec un vecteur, lors d'une telle succession d'opérations il faut donc veiller à ce que les indices restent associés aux bonnes dimensions car celles-ci changent.

2.2.1 Forme canonique, formulation à variables séparées

La représentation d'un champ multidimensionnel sous forme de produit à variables séparées est une technique simple permettant de contourner la « malédiction de la dimensionnalité ». D'un point de vue tensoriel, cette approximation est généralement nommée décomposition canonique ou décomposition CP (Canonical Polyadic). Elle peut redonner de manière exacte la valeur du tenseur initial, on nommera alors r le nombre minimal de modes nécessaires, c'est le rang du tenseur. Cependant, pour les applications mécaniques visées, seule une approximation des champs est nécessaire et

on se contentera systématiquement d'une troncature de la décomposition canonique. D'un point de vue tensoriel, on peut la noter :

$$\mathbf{X}_{CP} = \sum_{k=1}^{M} \bigotimes_{i=1}^{N} \underline{\boldsymbol{\gamma}}_i^{(k)} \tag{1.12}$$

où le tenseur $\mathbf{X}_{CP}$ est représenté par une somme de M modes composés des produits des N vecteurs $\underline{\boldsymbol{\gamma}}_i^{(k)}$, chacun de ces vecteurs étant associé à une dimension i. Le produit $\otimes$ utilisé est le produit tensoriel. Aussi nommé produit dyadique (outer product) en deux dimensions, on peut le définir par la relation $\underline{a} \otimes \underline{b} = a.b^T$ si $\underline{a}$ et $\underline{b}$ sont des vecteurs colonnes. Le produit tensoriel permet de générer un tenseur de dimension N à partir de N vecteurs. Par exemple, le premier mode de la représentation canonique (1.12) $\bigotimes_{i=1}^{N} \underline{\boldsymbol{\gamma}}_i$ est un tenseur défini en chaque point par $\mathbf{X}_1(i_1, i_2, \ldots, i_N) = \underline{\boldsymbol{\gamma}}_1(i_1) \times \underline{\boldsymbol{\gamma}}_2(i_2) \times \ldots \underline{\boldsymbol{\gamma}}_N(i_N)$.

Un tel tenseur, qui peut être écrit sous la forme d'un unique produit tensoriel, est un tenseur de rang 1. Si le rang du tenseur est bien unique, sa décomposition exacte (1.12) ne l'est pas, pas plus qu'une approximation d'ordre donné $M < r$ [Hackbusch, 2012].

Le tenseur de Kruskal (ou k-tensor) est une autre représentation de ce format, en particulier utilisé dans [Bader *et al.*, 2015]. Une pondération constante α_k est ajoutée à chaque mode pour ne manipuler que des vecteurs normalisés :

$$\mathbf{X}_K = \sum_{k=1}^{M} \alpha_k \bigotimes_{i=1}^{N} \underline{\boldsymbol{\gamma}}_i^{(k)} \tag{1.13}$$

2.2.2 Technique de calcul : PGD

La méthode PGD permet de construire une telle approximation via un algorithme glouton (greedy). On va d'abord présenter une vision continue de cet algorithme, puis on l'adaptera aux notations tensorielles en le discrétisant.

Vision continue

On cherche à approximer la fonction $\boldsymbol{X} \in \mathscr{S}_\mu$ sous une forme à variables séparées :

$$\boldsymbol{X}(\boldsymbol{\mu}) \approx \sum_{k=1}^{M} \prod_{i=1}^{N} \gamma_i^{(k)}(\mu_i) \tag{1.14}$$

On raisonne par récurrence : supposons que les $m-1$ premiers modes soient déjà connus et donnent l'approximation $\boldsymbol{X}_{PGD}^{(m-1)}$. On cherche à trouver l'ensemble optimal $\{\gamma\}^{(m)} = \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$ de N fonctions paramétriques solution du problème de minimisation suivant :

$$\min_{\{\gamma\}^{(m)}} \left\| \boldsymbol{R}^{(m)} - \prod_{i=1}^{N} \gamma_i \right\| \tag{1.15}$$

où on a noté $\boldsymbol{R}^{(m)} = \boldsymbol{X} - \boldsymbol{X}_{PGD}^{(m-1)}$ l'erreur de l'approximation à $m-1$ modes que l'on cherche à corriger. La norme $\|\bullet\|$ est une norme de $\mathscr{S}_\mu$. On choisit par exemple la norme L^2 définie via le produit scalaire :

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \int_{\boldsymbol{\Sigma}_\mu} \boldsymbol{A}(\boldsymbol{\mu}) \boldsymbol{B}(\boldsymbol{\mu}) d\boldsymbol{\mu} = \int_{\boldsymbol{I}_1} \int_{\boldsymbol{I}_2} \dots \int_{\boldsymbol{I}_N} \boldsymbol{A}(\boldsymbol{\mu}) \boldsymbol{B}(\boldsymbol{\mu}) d\mu_1 d\mu_2 \dots d\mu_N \tag{1.16}$$

et donc $\|\bullet\|^2 = \langle \bullet, \bullet \rangle$. Soit la fonction test $\delta\boldsymbol{X} = \sum_{i=1}^{N} \delta\gamma_i \prod_{j \neq i} \gamma_j$ de $\mathscr{S}_\mu$. La minimisation de (1.15) est équivalente dans ce contexte à sa formulation sous forme faible :

$$\forall \delta\boldsymbol{X} = \sum_{i=1}^{N} \delta\gamma_i \prod_{j \neq i} \gamma_j, \quad \left\langle \boldsymbol{R} - \prod_{i=1}^{N} \gamma_i, \delta\boldsymbol{X} \right\rangle = 0 \tag{1.17}$$

En particulier, c'est en annulant tous les $\delta\gamma_i$ sauf un qu'on obtient chaque équation du système :

$$\forall \delta\gamma_i \in \mathscr{I}_i, \begin{cases} \displaystyle\int_{\boldsymbol{I}_i} \delta\gamma_i \left[\boldsymbol{A}\gamma_i - \boldsymbol{B}(\mu_i) \right] d\mu_i = 0 \\ \text{avec} \quad \boldsymbol{A} = \displaystyle\int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \prod_{j \neq i} \gamma_j^2 d\overline{\boldsymbol{\mu}}_i \qquad \Rightarrow \gamma_i = \dfrac{\boldsymbol{B}(\mu_i)}{\boldsymbol{A}} \\ \text{et} \quad \boldsymbol{B}(\mu_i) = \displaystyle\int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \boldsymbol{R}^{(m)} \prod_{j \neq i} \gamma_j d\overline{\boldsymbol{\mu}}_i \end{cases} \tag{1.18}$$

où $\overline{\boldsymbol{\Sigma}_{\mu_i}} = \bigtimes_{j \neq i} \boldsymbol{I}_j$ et $d\overline{\boldsymbol{\mu}}_i = \prod_{j \neq i} d\mu_j$. Après initialisation de $\{\gamma\}^{(m)}$, ces N équations sont résolues successivement un certain nombre de fois jusqu'à atteindre un critère de convergence. La structure de cet algorithme de point fixe est donnée ALG.1. Pour éviter les divergences numériques, on normalise toutes les fonctions sauf une, la fonction γ_{i_N} par exemple qui va porter le poids du mode. On peut aussi choisir de normaliser toutes les fonctions et de les pondérer par une constante comme dans (1.13).

Remarque 6 *La résolution* (1.18) *est triviale, les opérations potentiellement coûteuses dans cette procédure sont les intégrations multiples pour les grands nombres de paramètres.*

Vision discrète

Numériquement, on travaille sur l'espace discrétisé $\boldsymbol{\Sigma}_\mu = \bigtimes_{k=1}^{N} \mathbf{I}_k$. Sur chacune des dimensions, on peut approcher le produit scalaire L^2 par une opération matricielle via la relation :

$$\forall (a,b) \in \left(\mathbb{R}^{\mathbf{I}_i}\right)^2, \quad \langle a, b \rangle \approx \underline{a}^T \underline{\underline{M}}_i \underline{b} \tag{1.19}$$

où $(\underline{a}, \underline{b}) \in \left(\mathbb{R}^{\mathbf{I}_i}\right)^2$ sont les discrétisations des fonctions continues a et b. Les matrices d'intégration $\underline{\underline{M}}_i$ sont par exemple obtenues via la méthode des trapèzes pour des discrétisations unidimensionnelles. Ainsi, l'intégrale $\boldsymbol{A}$ définie dans (1.18) se calcule avec des grandeurs discontinues via le produit $\boldsymbol{A} = \prod_{j \neq i} \underline{\boldsymbol{\gamma}}_j^T \underline{\underline{M}}_j \underline{\boldsymbol{\gamma}}_j$.

Algorithm 1 PGD d'un champ multiparamétrique donné $\boldsymbol{X}$

Initialiser : $\boldsymbol{X}_{PGD} = 0$
while $\|\boldsymbol{X}_{PGD} - \boldsymbol{X}\| > \epsilon$ **do** ▷ Exemple de critère d'arrêt
  Initialiser : $\forall i \in [\![1, N]\!]$, $\gamma_i = \gamma_i^{init}$ ▷ Création d'un nouveau mode
  for $p = 1 : P$ **do** ▷ Point fixe : nombre fixe d'itérations
    for $i = 1 : N$ **do** ▷ Pour chaque paramètre
      $\gamma_i = \frac{\boldsymbol{B}(\boldsymbol{X}_{PGD}, \boldsymbol{X}, \{\gamma_j\}_{j\neq i})}{\boldsymbol{A}(\{\gamma_j\}_{j\neq i})}$ ▷ Voir équation (1.18)
      Normalisation de γ_i sauf si $i = i_N$
    end for
  end for
  $\boldsymbol{X}_{PGD} = \boldsymbol{X}_{PGD} + \prod_{i=1}^{N} \gamma_i$ ▷ Mise à jour de l'approximation
end while

L'intégrale $\boldsymbol{B}$ est définie dans (1.18) à partir du tenseur discret $\mathbf{R}$. Après discrétisation, la succession de produits suivante permet de la déterminer :

$$\boldsymbol{B} = \left(\left(\left((\mathbf{R} \times_1 \underline{\underline{M}}_1 \underline{\gamma}_1) \cdots \times_{i-1} \underline{\underline{M}}_{i-1} \underline{\gamma}_{i-1} \right) \times_{i+1} \underline{\underline{M}}_{i+1} \underline{\gamma}_{i+1} \cdots \right) \times_N \underline{\underline{M}}_N \underline{\gamma}_N \right) \tag{1.20}$$

Cette notation, certes relativement lourde, a l'avantage de rester proche de la physique décrite : on intègre bien tout le tenseur suivant chacune de ses dimensions sauf une. L'ordre des produits est important car à chaque opération la dimension du tenseur décroit et la numérotation des dimensions est modifiée.

Dans le cas des très grandes dimensions, la construction complète d'un tenseur $\mathbf{X}$ est impossible et la procédure décrite ci-dessous n'est utilisée que pour compresser des données déjà présentes sous un format compact. En particulier, si $\mathbf{X}$ est déjà connu sous une forme à variables séparées (1.12), on peut noter simplement $\mathbf{R}$ sous cette forme : $\mathbf{R} = \sum_{r=1}^{R} \bigotimes_{i=1}^{N} \underline{\gamma}_i^{(r)}$. Le calcul de $\boldsymbol{B}$ est simplifié et ne nécessite plus de produit $\times_n$ imposant la manipulation de tenseurs complets :

$$\boldsymbol{B} = \sum_{r=1}^{R} \underline{\gamma}_i^{(r)} \prod_{j\neq i} \underline{\gamma}_j^{(r)^T} \underline{\underline{M}}_j \underline{\gamma}_j$$

Vocabulaire

La diversité des noms de formats et des méthodes permettant de les obtenir ne reflète pas toujours la richesse réelle des formulations utilisées. Comme nous l'avons vu dans la partie précédente avec la PCA et la transformation de Karhunen-Loève qui représente la même méthode en dimension finie, de nombreuses formulations similaires portent des noms différents en fonction de leur origine ou de leur application. Pour le cas particulier qui nous intéresse, nous avons vu que les formulations à variables séparées, ou « forme PGD » en mécanique du solide, se nomment différemment dans le domaine de l'analyse tensorielle : décomposition canonique (canonical decomposition, abrégé CANDECOMP dans [Carroll et Chang, 1970]), facteurs parallèles (parallel

factors, ou PARAFAC dans [Harshman, 1970]), deux noms regroupés dans les années 2000 par [Kiers, 2000] pour donner le format CP (CANDECOMP/PARAFAC) qui peut parfois être interprété comme « Canonical Polyadic » en reprenant l'appellation historique introduite dans [Hitchcock, 1927].

La même difficulté apparaît quand on s'intéresse aux méthodes permettant d'obtenir ces approximations. La « méthode PGD » présentée dans ce chapitre est basée sur la norme associée à (1.16). Il s'agit d'une légère généralisation des procédures « greedy » présentées dans les articles à dominante mathématique tels que [Kolda et Bader, 2009] et qui se basent sur la norme de Frobénius :

$$\|\mathbf{X}\|_{\mathscr{F}} = \sqrt{\sum_{i_1=1}^{n_1} \dots \sum_{i_N=1}^{n_N} (\mathbf{X}(i_1, \dots i_N))^2} \tag{1.21}$$

On la retrouve exactement en choisissant des matrices d'intégration $\underline{\underline{M}}_i$ égales à l'identité. De telles méthodes sont utilisées en mathématiques appliquées sous des noms tels que « greedy ALS » (par exemple dans [Kolda *et al.*, 2005]) et présentées comme des cas particuliers de la procédure décrite ci-après. Remarquons également que cette méthode a été ré-introduite sous le nom de HO-PGD (High Order PGD) dans [Modesto *et al.*, 2015].

Quelle est la méthode la plus pertinente? Des auteurs se sont interrogés et ont effectué des comparaisons, notamment en ce qui concerne les méthodes de minimisation utilisées pour résoudre les problèmes de type (1.15) ou (1.22). Il apparaît dans [Faber *et al.*, 2003] que la méthode ALS basée sur un ensemble de résolutions de problèmes monodimensionnels via des points fixes est une de celles qui donnent les approximations les plus précises à nombre de modes donné. Elle est présentée ci-dessous.

2.2.3 Technique de compression : CP-ALS

L'algorithme CP-ALS est une généralisation de la PGD précédente, il permet de construire directement une approximation optimale sur R modes (R donné) à partir d'un tenseur $\mathbf{X}$ connu. Cette méthode est classique en analyse tensorielle et est souvent présentée en utilisant des notations et outils différents de ceux utilisés en réduction de modèle mécanique.

Remarque 7 *ALS est l'abréviation de* Alternative Least Squares *(moindres carrés alternés) et décrit la structure en point fixe de l'algorithme. En effet, une succession de problèmes de minimisation monodimensionnels au sens des moindres carrés sont résolus en alternant chaque dimension. C'est exactement ce que nous faisons à une nuance près, celle de la norme utilisée. Il s'agit bien d'une minimisation au sens des moindres carrés quand la norme retenue dans* (1.15) *et* (1.22) *est la norme de Frobenius. Nous avons cependant conservé cette appellation car les méthodes sont identiques aux facteurs* $\underline{\underline{M}}_i$ *près.*

On cherche à résoudre le problème de minimisation suivant :

$$\min_{\{\Gamma\}} \left\| \boldsymbol{X} - \sum_{r=1}^{R} \prod_{i=1}^{N} \gamma_i^{(r)} \right\| \tag{1.22}$$

où la famille $\{\Gamma\}$ est composée des $N \times R$ fonctions $\gamma_i^{(r)}$. On introduit la notation vectorielle $\underline{\Gamma}_i = \left[\gamma_i^{(1)}, \ldots, \gamma_i^{(R)}\right]^T$. Après discrétisation, on stockera ces fonctions sous la forme de N matrices $\underline{\underline{\boldsymbol{\Gamma}}}_i = Col(\underline{\boldsymbol{\gamma}}_i^{(r)})_{r=1}^{R}$.

On considère la fonction-test $\delta\gamma = \sum_{r=1}^{R} \delta\gamma^{(r)}$ avec $\delta\gamma^{(r)} = \sum_{i=1}^{N} \delta\gamma_i^{(r)} \prod_{j\neq i} \gamma_j^{(r)}$. La minimisation (1.22) donne, en utilisant la même démarche que dans (1.17) et en annulant toutes les fonctions $\gamma_i^{(r)}$ sauf celles associées à une dimension i particulière :

$$\forall (\delta\gamma_i^{(1)}, \ldots, \delta\gamma_i^{(R)}) \in (\mathscr{I}_i)^R, \begin{cases} \displaystyle\sum_{r_1=1}^{R} \int_{\boldsymbol{I}_i} \delta\gamma_i^{(r_1)} \left[\sum_{r_2=1}^{R} \boldsymbol{A}_{r_1 r_2} \gamma_i^{(r_2)} - \boldsymbol{B}_{r_1} \right] d\mu_i = 0 \\ \text{avec} \quad \boldsymbol{A}_{r_1 r_2} = \displaystyle\int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \prod_{j\neq i} \gamma_j^{(r_1)} \gamma_j^{(r_2)} d\overline{\boldsymbol{\mu}}_i \qquad \Rightarrow \underline{\underline{\boldsymbol{A}}}\, \underline{\Gamma}_i = \underline{\underline{\boldsymbol{B}}} \\ \text{et} \quad \boldsymbol{B}_{r_1}(\mu_i) = \displaystyle\int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \boldsymbol{X} \prod_{j\neq i} \gamma_j^{(r_1)} d\overline{\boldsymbol{\mu}}_i \end{cases} \tag{1.23}$$

Le système (1.23) représente une itération du point fixe et nécessite la résolution d'un système carré de taille $R \times R$. Si R est de taille raisonnable, c'est *a priori* une opération peu coûteuse, ce qui n'est pas le cas des nombreuses intégrations en grandes dimensions.

Algorithm 2 CP-ALS

1: Choix du nombre de modes : R
2: Initialiser : $\forall i \in [\![1, N]\!]$, $\underline{\Gamma}_i = \underline{\Gamma}_i^{init}$ ▷ Création des R modes
3: **while** $\left\| \boldsymbol{X}_{CP} - \boldsymbol{X}_{CP}^{prec} \right\| > \epsilon$ **do** ▷ Point fixe : critère de stagnation et/ou nombre d'itérations maximum
4: **for** $i = 1 : N$ **do** ▷ Pour chaque paramètre
5: Construire $\underline{\underline{\boldsymbol{A}}}(\{\underline{\Gamma}_j\}_{j\neq i})$ et $\underline{\underline{\boldsymbol{B}}}(\{\underline{\Gamma}_j\}_{j\neq i}, \boldsymbol{X})$
6: Résoudre $\underline{\underline{\boldsymbol{A}}}\, \underline{\Gamma}_i = \underline{\underline{\boldsymbol{B}}}$ ▷ Voir équation (1.23)
7: Normalisation des fonctions de $\underline{\Gamma}_i$ si $i \neq i_N$
8: **end for**
9: **end while**

Cette procédure s'applique en pratique à des espaces discrétisées et sa structure est

résumée ALG.2. Le système (1.23) devient alors :

$$\underline{\underline{\boldsymbol{A}}}\,\underline{\underline{\boldsymbol{\Gamma}}}_i^T = \underline{\underline{\boldsymbol{B}}} \text{ avec } \begin{cases} \underline{\underline{\boldsymbol{A}}} = \prod_{j\neq i}^{*} \underline{\underline{\boldsymbol{\Gamma}}}_j^T \underline{\underline{M}}_j \underline{\underline{\boldsymbol{\Gamma}}}_j \quad \text{de taille } R\times R \\ \text{et} \quad \underline{\underline{\boldsymbol{B}}} = [\underline{B}_1,\ldots,\underline{B}_R]^T \quad \text{de taille } R\times n_i \\ \\ \text{avec} \quad \forall r\in[\![1,R]\!], \\ \underline{B}_r = \left(\left((\mathbf{X}\times_1 \underline{\underline{M}}_1\underline{\boldsymbol{\gamma}}_1^{(r)})\cdots\times_{i-1}\underline{\underline{M}}_{i-1}\underline{\boldsymbol{\gamma}}_{i-1}^{(r)}\right)\times_{i+1}\underline{\underline{M}}_{i+1}\underline{\boldsymbol{\gamma}}_{i+1}^{(r)}\right)\cdots\right) \end{cases} \tag{1.24}$$

On remarquera la notation $\prod^*$ qui correspond au produit termes à termes (ou produits de Hadamard) des différentes matrices concernées. Comme pour l'algorithme précédent, on peut constater que le calcul de $\underline{\underline{\boldsymbol{B}}}$ se simplifie lorsque le champ à approximer est déjà connu sous la forme à variables séparées (1.12), on a alors :

$$\forall r\in[\![1,R]\!],\quad \underline{B}_r = \sum_{k=1}^{M}\underline{\boldsymbol{\gamma}}_i^{(k)}\prod_{j\neq i}\underline{\boldsymbol{\gamma}}_j^{(k)\,T}\underline{\underline{M}}_j\underline{\boldsymbol{\gamma}}_j^{(r)} \Longleftrightarrow \underline{\underline{\boldsymbol{B}}}^T = \sum_{k=1}^{M}\underline{\boldsymbol{\gamma}}_i^{(k)}\prod_{j\neq i}^{*}\underline{\boldsymbol{\gamma}}_j^{(k)\,T}\underline{\underline{M}}_j\underline{\underline{\boldsymbol{\Gamma}}}_j \tag{1.25}$$

Cet algorithme est décrit dans [Kolda et Bader, 2009] par exemple. Le formalisme utilisé est particulièrement différent et c'est la norme de Frobenius qui est toujours utilisée. En particulier le calcul de $\underline{\underline{\boldsymbol{B}}}$ dans (1.24) peut être noté de manière plus compacte en utilisant des opérateurs tensoriels spécifiques tels que la matricisation et les produits de Khatri-Rao. Nous préférons ici conserver nos notations certes plus lourdes mais plus simples à interpréter mécaniquement, notamment du point de vue de la définition des normes. De plus, la simplification (1.25) pour un tenseur déjà sous forme canonique est très simple à visualiser ici. Remarquons enfin que les méthodes utilisées sont différentes. D'une part, les minimisations présentées ci-dessus sont réalisées via multiplication de la fonction à minimiser par un champ virtuel avant intégration. D'autre part, les démonstrations mathématiques présentées dans [Kolda et Bader, 2009] sont basées sur une écriture du problème (1.22) sous une forme matricielle (problème non carré). Il est ensuite résolu grâce à la théorie de l'inversion généralisée de Moore-Penrose qui permet de régulariser ces problèmes potentiellement mal posés. Ces deux méthodes conduisent bien aux mêmes fonctions optimales.

Remarque 8 (Matlab Tensor Toolbox) *L'ensemble des exemples numériques présentés dans cette thèse sont résolus grâce à Matlab. En particulier, les données sont formatées en utilisant les formats proposés dans la Tensor Toolbox développée par les laboratoires Sandia [Bader* et al.*, 2015]. Ainsi, tous les champs à variables séparées sont stockés sous la forme de k-tensors* (1.13)*. De plus, leur version de l'algorithme CP-ALS en partie compilée et très rapide est largement utilisée pour compacter des données sous cette forme. D'autres algorithmes peuvent donner des approximations de meilleur qualité, mais à un coût plus élevé [Tomasi et Bro, 2006].*

2.2.4 Format de Tucker

Le format de Tucker [Tucker, 1966] n'est pas adapté aux espaces de très grandes dimensions car il ne permet pas de s'affranchir de la « malédiction de la dimensionnalité ». Par exemple, pour un problème à N paramètres, la représentation de Tucker du tenseur $\mathbf{X} \in \mathbf{\Sigma}_\mu$ peut être écrite sous la forme :

$$\mathbf{X}_T = \mathscr{G} \times_1 \underline{\underline{\boldsymbol{A}}}^{(1)} \times_2 \underline{\underline{\boldsymbol{A}}}^{(2)} \cdots \times_N \underline{\underline{\boldsymbol{A}}}^{(N)} \tag{1.26}$$

où les $\underline{\underline{\boldsymbol{A}}}^{(i)}$ sont les matrices de facteurs et $\mathscr{G}$ le tenseur central ou cœur de l'approximation. On peut interpréter cette approximation comme une réduction de la dimension de chacun des espaces $\mathbb{R}^{\mathbf{I}_i}$. En effet, chaque matrice $\underline{\underline{\boldsymbol{A}}}^{(i)}$ est de taille $n_i \times n_r^{(i)}$ avec $n_r^{(i)} < n_i$. La dimension réduite de l'espace $\mathbb{R}^{\mathbf{I}_i}$ est donc $n_r^{(i)}$ et la matrice $\underline{\underline{\boldsymbol{A}}}^{(i)}$ contient l'équivalent des vecteurs propres associés à cette dimension. La HO-SVD (High Order SVD) est une méthode classique permettant d'approximer un tenseur par sa forme de Tucker [De Lathauwer *et al.*, 2000].

On peut remarquer qu'un tenseur sous sa forme canonique (1.12) peut être interprété comme un tenseur de Tucker dont les matrices de facteurs sont les $\underline{\underline{\mathbf{\Gamma}}}_i$ et le cœur un tenseur diagonal de dimension M^N.

Dans le cas général $\mathscr{G}$ n'est pas diagonal. Le nombre d'éléments à stocker est donc $(n_r)^N$ si on suppose que n_r est la taille de chaque nouvel espace paramétrique compressé. Pour des problèmes de plus de 10 paramètres, il est impossible d'utiliser cette représentation.

2.2.5 Diagrammes tensoriels et autres formats génériques

Deux autres formats génériques classiques sont fréquemment mentionnés dans la littérature : le format de Tucker hiérarchique [Hackbusch et Kühn, 2009] et un cas particulier qui lui est associé, les trains de tenseurs [Oseledets, 2011]. Plutôt que de donner la forme mathématique générale de ces tenseurs, parfois lourde et nécessitant de nouvelles notations, on se contentera d'introduire un outil graphique permettant de visualiser leur structure.

Les diagrammes tensoriels sont des schémas qui permettent de présenter visuellement le format d'une décomposition tensorielle et de mettre en évidence les interactions entre différents tenseurs sans se soucier de la taille de chaque dimension. Ils sont composés de points et de branches, chaque point représentant un tenseur différent et chaque branche une dimension. Un vecteur est donc représenté par un point et une branche, une matrice par un point et deux branches et un tenseur de dimension N par un point d'où sont issues N branches (voir FIG.1.3).

Un produit matrice/tenseur au n-ème mode tel que défini précédemment est représenté par la jonction entre deux points. Il est donc nécessaire que les dimensions mises en jeu dans les deux tenseurs soient compatibles. Par exemple, le produit entre deux matrices est symbolisé FIG.1.4. Quel que soit le format représenté, le nombre de branches « libres », qui ne sont reliées qu'à un point, reste le même pour un tenseur

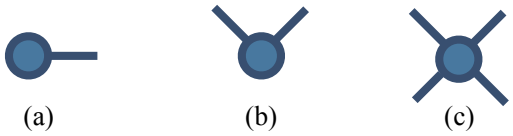


FIGURE 1.3 : Diagrammes tensoriels d'un vecteur (a), d'une matrice (b) et d'un tenseur de dimension 4 (c)

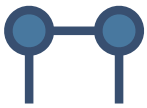


FIGURE 1.4 : Diagramme tensoriel d'un produit de deux matrices

donné et est égal à sa dimension. Le format de Tucker est représenté FIG.1.5a et on reconnait bien son corps au centre, tenseur de dimension N, et les matrices facteurs tout autour. Notons que cette représentation fonctionne aussi pour un tenseur sous forme canonique avec un cœur diagonal.

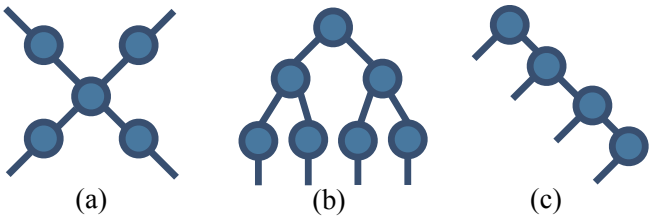


FIGURE 1.5 : Diagrammes tensoriels d'un tenseur de dimension 4 : format de Tucker ou forme canonique (a), format hiérarchique (b) et train de tenseurs (c).

Le tenseur hiérarchique [Hackbusch et Kühn, 2009, Grasedyck, 2010] est le format le plus générique possible et inclut les autres représentations discutées ici. Il consiste à structurer l'ensemble de nœuds qui mènent aux N branches finales du diagramme tensoriel. La FIG.1.5b en donne un exemple balancé. Il a été adapté à des algorithmes de réduction de modèle pour la résolution des EDP dans [Nouy, 2017] par exemple. Il est cependant délicat à utiliser car il faut définir en amont une structure dont le choix n'est pas trivial.

Le Tensor-Train format ([Holtz *et al.*, 2012, Oseledets, 2011], train de tenseurs ou TT-format) est comme le format canonique une représentation compacte adaptable aux grandes dimensions. Comme on peut le visualiser FIG.1.5c, un Tensor-Train est composé de $N-2$ tenseurs de dimension 3 et de deux matrices. La relative simplicité

de se format et la qualité des approximations d'ordre faible qu'il permet d'obtenir en font une représentation particulièrement utilisée. TAB.1.1 montre l'espace de stockage gagné grâce à ce format pour les champs bruts issus des nouveaux algorithmes étudiés dans cette thèse. Selon les problèmes, il peut être plus ou moins compact que la forme canonique. Le code utilisé pour compresser les données sous Matlab est fourni par [Oseledets *et al.*, 2012].

	problème à 4 dimensions	problème à 7 dimensions
Tenseur complet	2.8 Mb	1.4 Gb
Approximation après 2 itérations de l'algorithme PM-PGD sans compression (0,8-1,4% d'erreur)	400 kb	470 kb
Compression du tenseur approché via un Tensor Train (2% d'erreur)	19 kb	37 kb
Compression du tenseur approché sous forme canonique (2% d'erreur)	27 kb	18 kb

TABLEAU 1.1 : Coût de stockage de tenseurs de dimensions 4 et 7 issus de la résolution d'un problème de mécanique

2.3 Format adapté aux problèmes mécaniques multiparamétriques

Cette section présente un format adapté à la résolution de problèmes mécaniques à paramètres distribués spatialement. On revient au problème (1.3) qu'on réécrit dans un espace discrétisé spatialement sur d degrés de liberté (ddl). On ne détaille pas la méthode retenue, des éléments finis (EF) par exemple, et on se contente de noter $\underline{\underline{\mathbf{K}}}$ l'opérateur linéaire qui en résulte :

$$\begin{aligned} &\text{Trouver } \underline{\mathbf{X}} \in \mathcal{X} \text{ où } \mathcal{X} = \left\{ \underline{\mathbf{X}} : \boldsymbol{\Sigma}_\mu \to \mathbb{R}^d \,\middle|\, \int_{\boldsymbol{\Sigma}_\mu} \left\| \underline{\mathbf{X}}(\boldsymbol{\mu}) \right\|^2 d\boldsymbol{\mu} < \infty \right\} \\ &\text{tel que : } \quad \forall \boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu \qquad \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \underline{\mathbf{X}}(\boldsymbol{\mu}) = \underline{\mathrm{F}} \end{aligned} \tag{1.27}$$

On reconnait en $\underline{\mathbf{X}}$ un tenseur de dimension $N_p + 1$. L'expression (1.27) donne sa valeur pour la dimension spatiale (ou fibre, voir partie 2.2) pour chaque jeu $\boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu$ de paramètres. Nous allons effectuer quelques manipulations sur l'opérateur $\underline{\underline{\mathbf{K}}}$ afin de mettre en valeur l'importance du caractère localisé des paramètres.

Énergie d'une solution

Afin par exemple de construire des estimateurs d'erreurs ou de comparer des solutions entre elles, on définit l'énergie totale d'une solution comme étant l'intégrale sur le domaine paramétrique de l'énergie de déformation :

$$\int_{\boldsymbol{\Sigma}_\mu} \int_\Omega Tr\left(\boldsymbol{\sigma}\boldsymbol{\varepsilon}\right) d\Omega d\boldsymbol{\mu} \tag{1.28}$$

Si l'espace est discrétisé par une méthode EF d'opérateur de rigidité $\underline{\underline{\mathbf{K}}}$ et que $\underline{\mathbf{A}}$ et $\underline{\mathbf{B}}$ sont des champs de grandeur homogène à $\underline{\mathbf{X}}$ (ce que l'on note $\underline{\mathbf{A}} \propto \underline{\mathbf{B}} \propto \underline{\mathbf{X}}$), on introduit la forme bilinéaire suivante :

$$\left\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \right\rangle_{\mathscr{E}} = \int_{\boldsymbol{\Sigma}_\mu} \underline{\mathbf{A}}^T(\boldsymbol{\mu}) \underline{\underline{\mathbf{K}}}^0 \underline{\mathbf{B}}(\boldsymbol{\mu}) d\boldsymbol{\mu} \quad \text{avec} \quad \underline{\mathbf{A}} \propto \underline{\mathbf{B}} \propto \underline{\mathbf{X}} \tag{1.29}$$

où $\underline{\underline{\mathbf{K}}}^0 = \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}_0)$ avec $\boldsymbol{\mu}_0$ valeur moyenne des paramètres de $\boldsymbol{\Sigma}_\mu$. $\left\langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \right\rangle_{\mathscr{E}}$ donne une approximation de l'énergie (1.28). Si $\underline{\mathbf{A}}$ et $\underline{\mathbf{B}}$ ont des dimensions différentes, on introduit les définitions alternatives :

$$\begin{aligned} \left\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \right\rangle_{\mathscr{E}} &= \int_{\boldsymbol{\Sigma}_\mu} \underline{\mathbf{A}}^T(\boldsymbol{\mu}) \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\mathbf{B}}(\boldsymbol{\mu}) d\boldsymbol{\mu} \quad \text{si} \quad \underline{\mathbf{A}} \propto \underline{\mathbf{B}} \propto \underline{\mathrm{F}} \\ \left\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \right\rangle_{\mathscr{E}} &= \int_{\boldsymbol{\Sigma}_\mu} \underline{\mathbf{A}}^T(\boldsymbol{\mu}) \underline{\mathbf{B}}(\boldsymbol{\mu}) d\boldsymbol{\mu} \quad \text{si} \quad \underline{\mathbf{A}} \propto \underline{\mathrm{F}} \text{ et } \underline{\mathbf{B}} \propto \underline{\mathbf{X}} \end{aligned} \tag{1.30}$$

Remarque 9 (Semi-normes énergétiques) *Les formes bilinéaires présentées ne sont pas définies (au sens mathématique) et pas toujours symétriques, ce ne sont pas des produits scalaires. Elles ont la dimension d'une énergie et on qualifiera abusivement de « norme énergétique » le produit $\langle \bullet, \bullet \rangle_{\mathscr{E}}$, bien qu'il ne soit que la norme L^2 en paramètre d'une semi-norme matricielle associée à l'opérateur moyen en espace. On notera $\langle \bullet, \bullet \rangle_{\mathscr{E}} = |||\bullet|||^2$.*

Remarque 10 (Normes discrètes) *On n'introduira pas de nouvelle notation pour ces semi-normes après discrétisation de l'espace paramétrique $\boldsymbol{\Sigma}_\mu$ en $\boldsymbol{\Sigma}_\mu$.*

On introduit deux autres notations associées à des semi-normes particulières dérivées de $|||\bullet|||$:

- $\|\bullet\|$ est une norme exclusivement spatiale : elle quantifie l'énergie d'un champ spatial sans l'intégrer sur $\boldsymbol{\Sigma}_\mu$.
- $|||\bullet|||_E$ est une norme locale spatialement : elle se calcule à partir de données connues sur Ω_E. On intègre donc spatialement sur ce sous-domaine, ce qui revient à utiliser des produits scalaires construits sur l'opérateur de rigidité local $\underline{\underline{\mathbf{K}}}^0_E$ dans (1.30).

2.3.1 Principe de Saint-Venant

Au milieux du XIXème siècle, Adhémar Barré, comte de Saint-Venant, énonce le principe qui porte son nom [de Saint-Venant, 1856]. Il s'agit alors d'un principe de résistance des matériaux appliqué aux poutres. Il statue que les contraintes et déformations dans une région d'un solide en équilibre suffisamment éloignée des points d'application d'efforts extérieurs ne dépendent que de la résultante de ces efforts. Ce principe peut aujourd'hui être considéré comme un théorème pour les problèmes plaques et poutres [Ladevèze, 1985]. La FIG.1.6 illustre une conséquence de ce principe en 3D : une perturbation locale n'a qu'une influence localisée.

Nous proposons ci-dessous de développer mathématiquement les conséquences de ce principe quand on l'applique à un problème à grand nombre de paramètres locaux tel que celui présenté FIG.1.1.

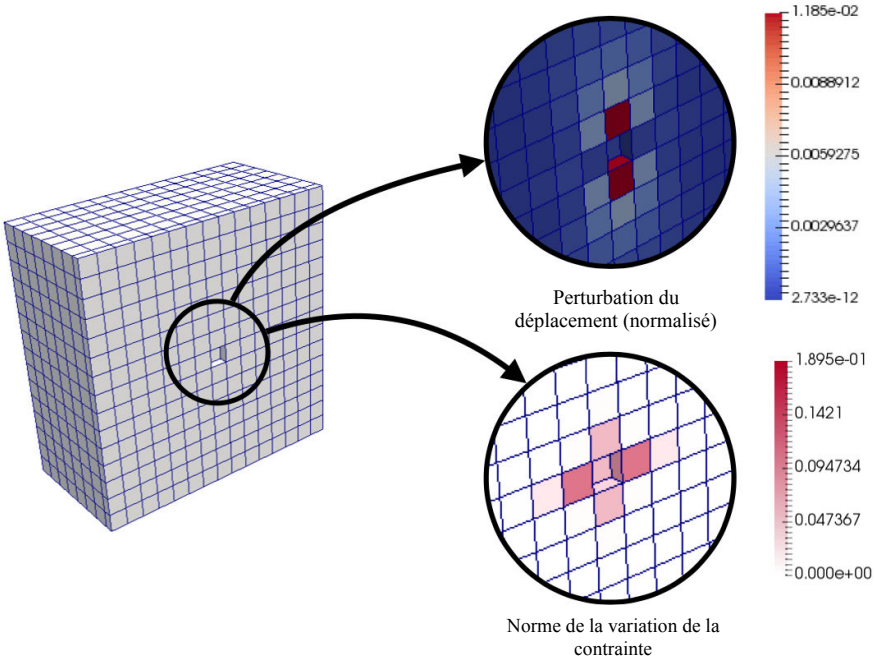


FIGURE 1.6 : Cube en traction, influence d'une perturbation locale (élément manquant) : différence entre les solutions perturbée et non perturbée

2.3.2 Développement de la solution

Réécrivons $\underline{\underline{\mathbf{K}}}$ en fonction de sa variation autour de sa valeur moyenne :

$$\underline{\underline{\mathbf{K}}} = \underline{\underline{\mathbf{K}}}^0 [\, \underline{\underline{\mathbf{1}}} - \underbrace{(\underline{\underline{\mathbf{1}}} - \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\underline{\mathbf{K}}})}_{\underline{\underline{\Delta}}} \,] \tag{1.31}$$

où $\underline{\underline{\mathbf{1}}}$ est la matrice identité. En définissant le champ moyen $\underline{\mathrm{X}}^0 = \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\mathrm{F}}$, on écrit la solution grâce au développement de Neumann de $(\underline{\underline{\mathbf{1}}} - \underline{\underline{\Delta}})^{-1}$ autour de celui-ci :

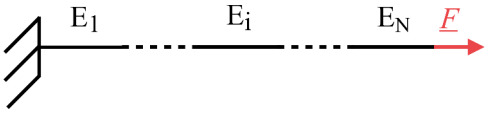
$$\underline{\mathbf{X}}(\boldsymbol{\mu}) : \underline{\mathrm{X}}^0 + \underbrace{\underline{\underline{\Delta}}\, \underline{\mathrm{X}}^0}_{\underline{\mathbf{X}}^1(\boldsymbol{\mu})} + \underbrace{\underline{\underline{\Delta}}^2 \underline{\mathrm{X}}^0}_{\underline{\mathbf{X}}^2(\boldsymbol{\mu})} + \ldots \tag{1.32}$$

$\underline{\mathbf{X}}^1$ étant une fonction linéaire en μ, $\underline{\mathbf{X}}^2$ un polynôme d'ordre 2, cette méthode s'apparente aux techniques d'approximation polynomiales. En pratique, plus faibles seront les variations des paramètres et plus petite sera la norme de $\underline{\underline{\Delta}}$, donc plus rapide sera la vitesse de convergence de l'approximation (1.32). Cette série ne converge que si les valeurs propres de $\underline{\underline{\Delta}}$ sont de valeurs absolues inférieures à 1, ce qu'on a pu vérifier dans la plage de données traitées ($\epsilon \in]0,2[$ dans la formulation (1.5)).

Cette formulation converge vers le champs solution exact $\underline{\mathbf{X}}$ et si on la tronque après un nombre fini et limité de termes, on obtient une approximation de $\underline{\mathbf{X}}$ donnée sous la

forme analytique (1.32). Cette forme est très compacte et ne se heurte pas à la « malédiction de la dimensionnalité ». Elle illustre ainsi comment on peut théoriquement calculer la solution en fonction des paramètres sans avoir recours au calcul complet de l'ensemble des solutions.

On a quantifié l'apport des premiers termes pour un problème monodimensionnel. FIG.1.7a illustre une poutre en traction composée de N éléments de modules d'Young indépendants, correspondant chacun à un paramètre. On peut calculer analytiquement la solution de ce problème très simple ainsi que les valeurs des différents termes du développement. Le résultat est donné TAB.1.7b pour des variations de modules d'Young de 50% autour de leurs valeurs moyennes. L'estimateur retenu est l'erreur en norme énergétique moyenne issue de (1.29). On constate qu'après l'ajout du terme quadratique du développement, l'erreur est déjà bien inférieure à 1%.



(a) Problème modèle

Développement	Erreur
$\underline{X}^0$	29%
$\underline{X}^0 + \underline{X}^1$	10%
$\underline{X}^0 + \underline{X}^1 + \underline{X}^2$	4%

(b) Erreur d'approximation du développement (1.32)

FIGURE 1.7 : Problème monodimensionnel à N paramètres

Soit un vecteur spatial élémentaire $\underline{Z}_E$, défini sur un sous-domaine Ω_E. On peut extrapoler ce vecteur local sur tout l'espace Ω pour obtenir un vecteur nul partout sauf sur Ω_E. Soit d_E le nombre de degrés de libertés associés à Ω_E et d le nombre de degrés de liberté total du système. On définit la matrice $\underline{\underline{\Pi}}_E \in \mathbb{R}^d \times \mathbb{R}^{d_E}$ telle que le vecteur élémentaire $\underline{Z}_E$ extrapolé sur tout l'espace s'obtienne via le produit $\underline{\underline{\Pi}}_E \underline{Z}_E$. Ainsi, le vecteur $\underline{Z}$ complet peut s'écrire en fonction de ses contributions locales élémentaires :

$$\underline{Z} = \sum_{E \in \mathbf{E}} \underline{\underline{\Pi}}_E \underline{Z}_E$$

Soit $\underline{\underline{\mathbf{K}}}_E$ un opérateur élémentaire. On peut aussi l'extrapoler sur tout l'espace, on a alors l'opérateur global $\underline{\underline{\Pi}}_E \underline{\underline{\mathbf{K}}}_E \underline{\underline{\Pi}}_E^T$. On suppose que chaque rigidité locale $\underline{\underline{\mathbf{K}}}_E$ dépend linéairement du paramètre μ_E. C'est le cas par exemple dans notre problème modèle via la relation (1.5). On a donc sur chaque sous-domaine : $\underline{\underline{\mathbf{K}}}_E = \underline{\underline{\mathbf{K}}}_E^0 + \mu_E \underline{\underline{\widehat{\mathbf{K}}}}_E$ où $\underline{\underline{\mathbf{K}}}_E^0$ est l'opérateur local moyen et $\underline{\underline{\widehat{\mathbf{K}}}}_E$ sa partie linéaire en μ_E. L'assemblage des matrices locales $\underline{\underline{\mathbf{K}}}_E$ s'écrit :

$$\underline{\underline{\mathbf{K}}} = \bigwedge_{E \in \mathbf{E}} \underline{\underline{\mathbf{K}}}_E = \sum_{E \in \mathbf{E}} \underline{\underline{\Pi}}_E \underline{\underline{\mathbf{K}}}_E \underline{\underline{\Pi}}_E^T \tag{1.33}$$

Cette notation permet de développer le premier terme de la série (1.32) :

$$\underline{\mathbf{X}}^1(\boldsymbol{\mu}) = \underline{\underline{\boldsymbol{\Delta}}}\,\underline{\mathrm{X}}^0 = -\underline{\underline{\mathbf{K}}}^{0^{-1}}\left[\sum_{E\in\mathbf{E}}\mu_E\underline{\underline{\Pi}}_E\underline{\underline{\widehat{\mathbf{K}}}}_E\underline{\underline{\Pi}}_E^T\underline{\mathrm{X}}^0\right] = \sum_{E\in\mathbf{E}}-\mu_E\underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\underline{\Pi}}_E\underbrace{\left[\underline{\underline{\widehat{\mathbf{K}}}}_E\underline{\mathrm{X}}_E^0\right]}_{\underline{\mathrm{Y}}_E} \tag{1.34}$$

en notant $\underline{\underline{\Pi}}_E^T\underline{\mathrm{X}}^0 = \underline{\mathrm{X}}_E^0$ la restriction de $\underline{\mathrm{X}}^0$ sur le sous-domaine Ω_E. Le terme $\underline{\underline{\Pi}}_E\underline{\mathrm{Y}}_E$ est une contrainte localement équilibrée sur Ω_E. Supposons que l'on applique cette contrainte locale au solide Ω complet de rigidité moyenne $\underline{\underline{\mathbf{K}}}^0$. D'après le principe de Saint-Venant, la solution associée est localisée dans un voisinage de Ω_E que l'on notera $\mathbf{C}_E$ et à qui on associe un opérateur de restriction $\underline{\underline{\Pi}}_{\mathbf{C}_E}$. On introduit donc la solution locale $\underline{\mathrm{Z}}^1_{\mathbf{C}_E}$ et son expression sur chaque sous-domaine $\Omega_{E'} \in \mathbf{C}_E$, $\underline{\mathrm{Z}}^1_{\mathbf{C}_E,E'}$, ce qui donne :

$$\underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\underline{\Pi}}_E\underline{\mathrm{Y}}_E \simeq \underline{\underline{\Pi}}_{\mathbf{C}_E}\underline{\mathrm{Z}}^1_{\mathbf{C}_E} = \sum_{E'\in\mathbf{C}_E}\underline{\underline{\Pi}}_{E'}\underline{\mathrm{Z}}^1_{\mathbf{C}_E,E'} \tag{1.35}$$

Cela revient à négliger l'influence de $\underline{\mathrm{Y}}_E$ hors du voisinage $\mathbf{C}_E$. D'où :

$$\underline{\mathbf{X}}^1(\boldsymbol{\mu}) \simeq \sum_{E\in\mathbf{E}}-\mu_E\underline{\underline{\Pi}}_{\mathbf{C}_E}\underline{\mathrm{Z}}^1_{\mathbf{C}_E} \tag{1.36}$$

On en déduit qu'à l'ordre 1, la solution a des dépendances paramétriques en μ_E qui sont linéaires et localisées dans un voisinage de Ω_E. De même, à l'ordre 2 on a :

$$\begin{aligned}
\underline{\mathbf{X}}^2(\boldsymbol{\mu}) &\simeq \sum_{E\in\mathbf{E}}\sum_{E'\in\mathbf{C}_E}\mu_{E'}\mu_E\underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\underline{\Pi}}_E\underline{\underline{\widehat{\mathbf{K}}}}_E\underline{\underline{\Pi}}_E^T\left(\underline{\underline{\Pi}}_{E'}\underline{\mathrm{Z}}^1_{\mathbf{C}_E,E'}\right)\\
&\simeq \sum_{E\in\mathbf{E}}\mu_E^2\underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\underline{\Pi}}_E\underline{\underline{\widehat{\mathbf{K}}}}_E\underline{\mathrm{Z}}^1_{\mathbf{C}_E,E} + \sum_{E\in\mathbf{E}}\sum_{\substack{E'\in\mathbf{C}_E\\E\neq E'}}\mu_E\mu_{E'}\underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\underline{\Pi}}_E\underline{\underline{\widehat{\mathbf{K}}}}_E\underline{\mathrm{Z}}^1_{\mathbf{C}_E,E'}\\
&\simeq \sum_{E\in\mathbf{E}}\mu_E^2\underline{\mathrm{Z}}^2_E + \sum_{E\in\mathbf{E}}\sum_{\substack{E'\in\mathbf{C}_E\\E'\neq E}}\mu_E\mu_{E'}\underline{\mathrm{Z}}^2_{\mathbf{C}_E,E'}
\end{aligned} \tag{1.37}$$

où $\underline{\mathrm{Z}}^2_E = \underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\underline{\Pi}}_E\underline{\underline{\widehat{\mathbf{K}}}}_E\underline{\mathrm{Z}}^1_{\mathbf{C}_E,E}$ est comme précédemment localisé sur $\mathbf{C}_E$ tandis que chacun des $\underline{\mathrm{Z}}^2_{\mathbf{C}_E,E'}$ reste localisé sur $\mathbf{C}_{E'}$. Ainsi, on en déduit que l'ordre 2 du développement (1.32) est quadratique par rapport aux paramètres μ_E dans un voisinage de Ω_E et linéaire par rapport aux paramètres dans un voisinage plus étendu, incluant tous les voisinages $\mathbf{C}_{E'}$ des sous-domaines composants $\mathbf{C}_E$. De ces remarques aux ordres 1 et 2, on retient les deux propriétés suivantes pour un paramètre μ_E :

— la majorité de la dépendance paramétrique, potentiellement non-linéaire, est localisée dans un voisinage $\mathbf{C}_E$ de Ω_E,
— hors de $\mathbf{C}_E$, la dépendance paramétrique reste linéaire.

Ces deux propriétés sont résumées visuellement FIG.1.8.

2.3.3 Format retenu

Le format retenu va donc chercher à tirer parti de ces deux échelles. On conserve le formalisme associé aux méthodes PGD et les champs sont décrits sous une forme à

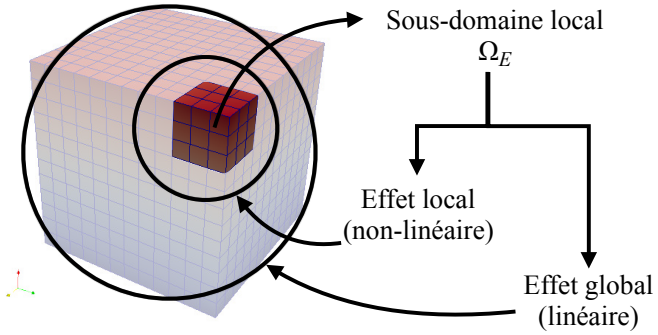


FIGURE 1.8 : Influence à deux échelles d'un paramètre local

variables séparées. On y inclut en plus une information sur la richesse des variations paramétriques en différenciant des fonctions « macro » et « micro ». Ces deux échelles permettent de représenter de deux manières différentes les mêmes champs, précisément (fonction micro) ou non (fonction macro) suivant le contexte d'utilisation. Cette technique s'inspire des représentations multiéchelles en espace [Dhia et Rateau, 2005, Ladevèze et Dureisseix, 2000] ou en temps [Ladeveze et Nouy, 2003]. Comme schématisées FIG.1.9, les fonctions « micro » seront discrétisées finement sur leur domaine de variation tandis que les fonctions « macro » sont choisies linéaires.

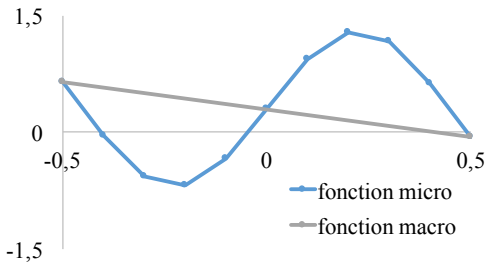


FIGURE 1.9 : Fonction micro et fonction macro associée

Ainsi, le nouveau format est défini localement en choisissant pour chaque sous-domaine Ω_E une expression différente composée de M modes :

$$\underline{\mathbf{X}}_E(\boldsymbol{\mu}) = \sum_{k=1}^{M} \underline{\widetilde{\mathbf{X}}}_E^{(k)} \prod_{E'' \notin \widehat{\mathbf{C}}_E} \underline{\boldsymbol{\gamma}}_{E'',(E)}^{M(k)}(\mu_{E''}) \prod_{E' \in \widehat{\mathbf{C}}_E} \underline{\boldsymbol{\gamma}}_{E',(E)}^{m(k)}(\mu_{E'}) \tag{1.38}$$

où $\underline{\widetilde{\mathbf{X}}}_E$ est un vecteur spatial localisé sur Ω_E, $\underline{\boldsymbol{\gamma}}^M$ et $\underline{\boldsymbol{\gamma}}^m$ sont respectivement des fonctions « macro » et « micro » des paramètres. $\widehat{\mathbf{C}}_E$ est un voisinage de Ω_E qui définit l'impact « micro » en espace du paramètre μ_E. On peut par exemple choisir de prendre tous les sous-domaines ayant une surface commune avec Ω_E comme illustré FIG.1.10.

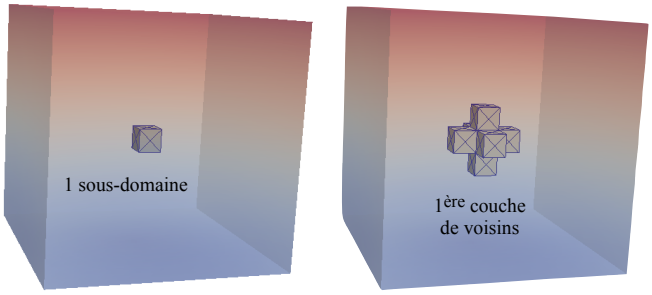


FIGURE 1.10 : Un sous-domaine et un des voisinages possibles associé

La solution complète du problème est l'ensemble de ces tenseurs locaux : $\underline{\underline{\mathbf{X}}}(\boldsymbol{\mu}) = \left\{\underline{\underline{\mathbf{X}}}_E(\boldsymbol{\mu})\right\}_{E\in\mathbf{E}}$.

Remarque 11 *Le format* (1.38) *peut être relativement lourd à manipuler pour donner des solutions globales car il faut reconstruire la solution sur chacun des sous-domaines avant d'obtenir un champ complet, mais est idéal pour déterminer des grandeurs locales en espace.*

3 Résolution d'EDP : construction *a priori* de bases adaptées

En mécanique des structures, les méthodes les plus classiques de réduction de modèle ne font pas appel aux formats de donnés les plus complexes présentés. En outre, on ne cherche pas toujours à donner une expression explicite d'une solution dans le domaine paramétrique comme dans l'exemple de la POD.

3.1 POD : Proper Orthogonal Decomposition

Cette méthode, confondue par certains auteurs avec la décomposition de KL ou la PCA, est introduite en mécanique dans les années 60 [Lumley, 1967]. [Kerschen *et al.*, 2005] en résume le principe ainsi que diverses applications dans notre domaine. Cette méthode a l'avantage d'être simple, non intrusive et ne nécessite pas de connaissance d'un modèle mathématique du comportement étudié. L'algorithme présenté ci-dessous est la POD-Galerkin.

On s'intéresse au problème discrétisé (1.27). Au lieu de chercher sa solution complète dans l'espace $\mathscr{U}^h$ pour chaque nouveau jeu de paramètres, on va se placer dans une base de petite taille. Soit $\underline{\underline{\Phi}} = Col(\underline{\phi}_k)_{k=1}^M$ une telle base de dimension M et $\mathscr{U}^M$ le sous-espace de $\mathscr{U}^h$ associé. La solution de (1.27) recherchée a la forme :

$$\underline{u}(\boldsymbol{\mu}) \approx \underline{u}_{POD}(\boldsymbol{\mu}) = \sum_{k=1}^{M} \alpha_k(\boldsymbol{\mu}).\underline{\phi}_k \tag{1.39}$$

Choix de la base

La phase offline de cette méthode consiste à construire la base $\underline{\underline{\Phi}}$. On choisit généralement de la calculer via des algorithmes de type PCA tels que ceux présentés en 2.1. Il est donc nécessaire de résoudre le problème (1.27) pour un ensemble de jeux paramétriques $\boldsymbol{\mu}^{(i)}$ qui vont donner les snapshots $\underline{u}^{(i)}$. Le choix de ces solutions particulières est crucial car la minimisation (1.7) qui permet de construire une base $\underline{\underline{\Phi}}$ optimale n'est relative qu'à ces tirages. Les méthodes de type Reduced Basis étudiées en 3.2 permettent de choisir des solutions particulières optimales afin de garantir la qualité de la base retenue pour approcher un jeu de paramètres donnés.

Selon l'utilisation recherchée du modèle réduit, deux approches liées à la POD sont possibles : l'utilisation simple de la base réduite pour résoudre le problème en temps réel ou la construction complète d'un abaque virtuel associé à celle-ci.

Projection du problème

On introduit $\underline{\boldsymbol{\alpha}}_{\boldsymbol{\mu}} = [\alpha_1(\boldsymbol{\mu}), \ldots, \alpha_M(\boldsymbol{\mu})]$ le vecteur qui représente l'approximation de $\underline{u}(\boldsymbol{\mu})$ dans la base $\underline{\underline{\Phi}}$ pour un jeu $\boldsymbol{\mu}$ fixé. Ce vecteur est solution du problème approché :

$$\underline{\underline{\mathbf{K}}}_{\boldsymbol{\mu}}^{POD} \underline{\boldsymbol{\alpha}}_{\boldsymbol{\mu}} = \underline{\mathrm{F}}^{POD} \quad \text{avec} \quad \underline{\underline{\mathbf{K}}}_{\boldsymbol{\mu}}^{POD} = \underline{\underline{\Phi}}^T \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \underline{\underline{\Phi}} \quad \text{et} \quad \underline{\mathrm{F}}^{POD} = \underline{\underline{\Phi}}^T \underline{\mathrm{F}} \tag{1.40}$$

Ce problème est de petite taille, il nécessite l'inversion de la matrice $\underline{\underline{\mathbf{K}}}_{\boldsymbol{\mu}}^{POD}$ de taille $M \times M$ ce qui est bien plus rapide que la résolution du système complet de taille $d \times d$. Cependant, on remarque qu'il faut quand même assembler le système $\underline{\underline{\mathbf{K}}}_{\boldsymbol{\mu}}^{POD}$ qui dépend de $\boldsymbol{\mu}$. Cette méthode est la plus simple et la plus ancienne utilisation de la POD. Un exemple a été traité dans [Paillet, 2016b].

Interpolation dans l'espace paramétrique

Le calcul online présenté consiste à déterminer le vecteur $\underline{\boldsymbol{\alpha}}_{\boldsymbol{\mu}}$. Le stockage de tous ces coefficients sous une forme réduite permet une étape de réduction supplémentaire. Il faut donc déterminer M interpolations différentes pour chaque coefficient α_i. Une méthode très simple consiste à échantillonner régulièrement ou pseudo-aléatoirement l'espace paramétrique $\boldsymbol{\Sigma}_\mu$ sur une grille $\mathscr{G}_\mu$ puis à déterminer la valeur de $\underline{\boldsymbol{\alpha}}$ en chacun de ces points. Lors de la phase online, il suffit alors de projeter le jeu de paramètre d'intérêt $\boldsymbol{\mu}$ sur la grille $\mathscr{G}_\mu$ pour obtenir immédiatement les coefficients $\underline{\boldsymbol{\alpha}}_{\boldsymbol{\mu}}$. Un exemple d'une telle interpolation a été réalisé dans [Paillet, 2016b].

Des méthodes d'interpolation plus sophistiquées telle que la méthode EIM (Empirical Interpolation Method [Barrault *et al.*, 2004]) permettent de choisir des points d'interpolation de manière plus pertinente. En particulier, l'EIM utilise une procédure greedy pour choisir à la fois les points d'échantillonnage $\boldsymbol{\mu}^{(i)}$ du modèle réduit et construire les fonctions d'interpolation entre ces points.

3.2 RB : Reduced Basis

Les Reduced Basis sont, comme la POD, des méthodes de projection de l'opérateur sur une base réduite [Maday et Ronquist, 2004, Patera *et al.*, 2007, Rozza, 2014]. Elles diffèrent de part le choix de cette base, qui est générée à partir de snapshots choisis de manière gloutonne (greedy). Une fois un ensemble de solutions $\underline{u}^{(i)} = u(\boldsymbol{\mu}^{(i)})$ retenu, on ne cherche pas à extraire les modes propres de cette famille. Les $\underline{u}^{(i)}(\boldsymbol{\mu}^{(i)})$ ont été choisis pour l'information nouvelle qu'ils apportent au modèle et on va les garder comme vecteurs de base de notre modèle réduit : $\mathscr{U}^M = Vect(\underline{u}^{(1)}(\boldsymbol{\mu}^{(1)}), \ldots, \underline{u}^{(M)}(\boldsymbol{\mu}^{(M)}))$. On peut éventuellement construire une base orthonormale en les orthogonalisant via le processus de Gram-Schmidt.

Une méthode de construction est donnée dans [Rozza *et al.*, 2007]. On initialise cette base en choisissant aléatoirement un jeu de paramètres $\boldsymbol{\mu}^{(1)}$. On suppose qu'on a déjà sélectionné m jeux de paramètres $\boldsymbol{\mu}^{(i)}$, on va retenir la solution la moins bien décrite par la base déjà construite, en norme énergétique par exemple :

$$\boldsymbol{\mu}^{m+1} = \arg \max_{\boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu} \left\| \underline{u}(\boldsymbol{\mu}) - \Pi^m \underline{u}(\boldsymbol{\mu}) \right\| \simeq \arg \max_{\boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu} \left[\Delta^m(\boldsymbol{\mu}) \right] \tag{1.41}$$

où Π^m représente la projection de $\underline{u}(\boldsymbol{\mu})$ dans la base $\mathscr{U}^m$. L'estimateur d'erreur $\Delta^m(\boldsymbol{\mu})$ est un opérateur scalaire qui doit être choisi [Rozza *et al.*, 2007, Patera *et al.*, 2007]. On peut par exemple retenir une fonction proportionnelle au résidu de l'équation d'équilibre pour chaque jeu de paramètres $\boldsymbol{\mu}$:

$$\Delta^m(\boldsymbol{\mu}) = \left\| \mathbf{R}^m(\mu) \right\| = \left\| \underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}\, \underline{u}^m(\boldsymbol{\mu}) \right\| \tag{1.42}$$

où $\underline{u}^m(\boldsymbol{\mu})$ est la solution exacte associée au jeu de paramètres $\boldsymbol{\mu}$ et projetée sur $\mathscr{U}^M$. En grande dimension, trouver ce minimum n'est pas trivial et nécessite de faire appel à des algorithmes d'optimisation spécifiques.

Cette procédure permet donc de choisir de manière optimale une base réduite pour décrire un ensemble de solutions dépendant de paramètres potentiellement nombreux. Elle permet aussi de garantir une erreur maximale réalisée lors d'une résolution dans l'espace d'approximation.

3.3 Méthodes d'hyper-réduction

En combinant une méthode de réduction de modèle telle que la POD ou une RB avec une technique d'interpolation comme l'EIM par exemple, on obtient un niveau supplémentaire de réduction. On nomme hyper-réduction (HR) les techniques associées à ce type de combinaisons utilisées par exemple dans [Carlberg *et al.*, 2013] ou [Ryckelynck, 2009].

En effet, même si la base réduite est de dimension faible, de nombreuses opérations restent coûteuses, notamment les intégrations comme mentionné REM.6. On peut accélérer ces étapes grâce à des méthodes d'interpolation (voir 3.1) ou de projection lacunaire comme dans [Ryckelynck, 2005].

La Reference Point Method (RPM) [Ladevèze, 1997] est une méthode d'interpolation qui, associée à la PGD décrite ci-dessous, forme une méthode d'HR et permet des gains en temps de calcul conséquent pour les problèmes non-linéaires [Capaldo *et al.*, 2017]. Celle-ci a été appliquée à des problèmes à grands nombres de paramètres pendant cette thèse avec un succès mitigé. Son adaptation en grandes dimensions et les résultats obtenus sont présentés en Annexe B.

4 La PGD, Proper Generalized Decomposition

Dans les années 1980, la méthode PGD, Proper Generalised Decomposition, a été introduite au LMT Cachan par Pierre Ladevèze sous le nom « d'approximation radiale » [Ladevèze, 1985, Ladevèze, 1989]. C'était à l'origine un point technique de la méthode LaTIn décrite dans [Ladevèze, 1996] et conçue pour résoudre des problèmes de comportement matériau non-linéaire. Elle a depuis été utilisée pour de nombreuses applications en mécanique des structures et [Ladevèze, 2014] donne une idée de la variété des utilisations étudiées. Le nom de la méthode a été modifié en 2010 par P. Ladevèze et F. Chinesta afin de mettre en valeur sa généralité : la PGD est une extension de l'approche POD décrite en 3.1 et applicable à de nombreux domaines.

Les problèmes non-linéaires [Ladevèze, 1996, Boisse *et al.*, 1990] forment l'application d'origine de cette méthode et certains perfectionnements des techniques sont toujours en cours [Vitse *et al.*, 2014]. Elle est aujourd'hui employée aussi pour résoudre des problèmes multiéchelles [Cremonesi *et al.*, 2013, Chinesta *et al.*, 2010] ou de décomposition de domaine [Nazeer *et al.*, 2014]. La rapidité d'utilisation des modèles réduits issus de la PGD fait d'elle un outil privilégié pour l'étude de problèmes de validation [Bouclier *et al.*, 2013] ou d'identification [Passieux *et al.*, 2015] en temps réel.

Son utilisation dépasse les frontières de la mécanique des structures et elle peut par exemple être employée pour traiter des problèmes de thermique [Lamari *et al.*, 2010] ou d'électromagnétisme [Pineda-Sanchez *et al.*, 2010] et a connu de nombreux développements en mécanique des fluides [Ammar *et al.*, 2007] et rhéologie [Chinesta *et al.*, 2011]. Elle peut être adaptée à un contexte stochastique [Nouy, 2007] et son application à des problèmes multiphysiques [Dureisseix *et al.*, 2003, Néron *et al.*, 2015] et multiparamétriques [Pruliere *et al.*, 2010] permet de créer des modèles réduits très riches.

L'idée principale de cette méthode consiste à calculer la base optimale de représentation d'un problème et sa solution simultanément en utilisant une approche itérative gloutonne.

4.1 Algorithmes gloutons classiques

Par souci de cohérence avec les algorithmes à grands nombres de paramètres traités aux chapitres suivants, on présentera les algorithmes PGD classiques pour des problèmes espace-paramètres. La méthode est bien sûr plus générale et on

trouve par exemple la présentation détaillée de la PGD dans plusieurs contextes dans [Chinesta *et al.*, 2013a].

La technique utilisée est très proche de celle étudiée en 2.2.2 pour approcher un champ donné. On a choisi de présenter ici une formulation discrétisée en espace et indépendante de la méthode de discrétisation spatiale retenue. On conservera cependant une représentation continue de l'espace paramétrique moins lourde à introduire, comme on l'a constaté en 2.2.2. Le problème traité a déjà été énoncé en (1.27), on en cherche une solution sous la forme à variables séparées :

$$\underline{\mathbf{X}}(\boldsymbol{\mu}) = \sum_{k=1}^{M} \underline{\widetilde{\mathrm{X}}}^{(k)} \prod_{i=1}^{N_p} \gamma_i^{(k)}(\mu_i) \tag{1.43}$$

On suppose que l'approximation $\underline{\mathbf{X}}_{PGD}^{(m-1)}$ sur les $m-1$ premiers modes est connue. Soit la fonction test $\delta\underline{\mathbf{X}} = \delta\underline{\widetilde{\mathrm{X}}} \prod_{i=1}^{N_p} \gamma_i + \sum_{i=1}^{N_p} \delta\gamma_i \underline{\widetilde{\mathrm{X}}} \prod_{j\neq i} \gamma_j$.

4.1.1 Progressive Galerkin PGD

Cette formulation est la plus ancienne [Ladevèze, 1999]. Dans le cas général, il n'existe pas de preuve de la convergence de cette méthode, c'est pourtant la plus utilisée et son efficacité est reconnue pour de nombreux cas d'applications.

On cherche à déterminer le mode suivant en posant le problème sous forme faible, on utilise ici le produit scalaire énergétique :

$$\forall \delta\underline{\mathbf{X}} = \delta\underline{\widetilde{\mathrm{X}}} \prod_{i=1}^{N_p} \gamma_i + \sum_{i=1}^{N_p} \delta\gamma_i \underline{\widetilde{\mathrm{X}}} \prod_{j\neq i} \gamma_j, \quad \left\langle \delta\underline{\mathbf{X}}, \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \left[\underline{\mathbf{X}}_{PGD}^{(m-1)} + \underline{\widetilde{\mathrm{X}}} \prod_{i=1}^{N_p} \gamma_i \right] - \underline{\mathrm{F}} \right\rangle_{\mathscr{E}} = 0 \tag{1.44}$$

ce qui donne les $N_p + 1$ équations suivantes en particularisant (1.44) pour l'espace et pour chacun des paramètres :

$$\left\{ \begin{array}{l} \forall \delta\underline{\widetilde{\mathrm{X}}} \in \mathbb{R}^d, \quad \delta\underline{\widetilde{\mathrm{X}}}^T \displaystyle\int_{\boldsymbol{\Sigma}_\mu} \underline{\underline{\mathbf{K}}}\, \underline{\widetilde{\mathrm{X}}} \prod_{i=1}^{N_p} \gamma_j^2 d\boldsymbol{\mu} = \delta\underline{\widetilde{\mathrm{X}}}^T \int_{\boldsymbol{\Sigma}_\mu} \left(\underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}\, \underline{\mathbf{X}}_{PGD}^{(m-1)} \right) \prod_{i=1}^{N_p} \gamma_i d\boldsymbol{\mu}_i \text{(a)} \\ \forall i \in [\![1, N_p]\!] : \\ \forall \delta\gamma_i \in \mathscr{I}_i, \quad \displaystyle\int_{\boldsymbol{I}_i} \delta\gamma_i \left(\gamma_i \int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \underline{\widetilde{\mathrm{X}}}^T \underline{\underline{\mathbf{K}}}\, \underline{\widetilde{\mathrm{X}}} \prod_{j\neq i} \gamma_j^2 d\overline{\boldsymbol{\mu}}_i - \right. \\ \qquad\qquad \left. \displaystyle\int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \underline{\widetilde{\mathrm{X}}}^T \left(\underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}\, \underline{\mathbf{X}}_{PGD}^{(m-1)} \right) \prod_{j\neq i} \gamma_j d\overline{\boldsymbol{\mu}}_i \right) d\mu_i = 0 \quad \text{(b)} \end{array} \right. \tag{1.45}$$

Ces différentes équations sont résolues successivement via un point fixe. La structure complète de l'algorithme est donnée ALG.3. Contrairement à la PGD d'un champ donné, les intégrations ne sont pas les opérations limitantes pour des exemples classiques (peu de paramètres). En effet, la résolution de (1.45)(a) impose l'inversion d'un

Algorithm 3 Résolution PGD d'un problème espace-paramètre (1.27)

Initialiser : $\underline{\mathbf{X}}_{PGD} = 0$
while $\left\|\underline{\widetilde{\mathrm{X}}}\right\| > \epsilon$ **do** ▷ Exemple de critère d'arrêt : stagnation
  Initialiser : $\underline{\widetilde{\mathrm{X}}} = \underline{\widetilde{\mathrm{X}}}^{init}$, $\forall i \in [\![1, N_p]\!]$, $\gamma_i = \gamma_i^{init}$ ▷ Création d'un nouveau mode
  for $p = 1 : P$ **do** ▷ Point fixe : nombre fixe d'itérations
    Calcul de $\underline{\widetilde{\mathrm{X}}}$ via (1.45)(a)
    for $i = 1 : N_p$ **do** ▷ Pour chaque paramètre
      Calcul de γ_i via (1.45)(b)
      Normalisation de γ_i
    end for
  end for
  $\underline{\mathbf{X}}_{PGD} = \underline{\mathbf{X}}_{PGD} + \underline{\widetilde{\mathrm{X}}} \prod_{i=1}^{N_p} \gamma_i$ ▷ Mise à jour de l'approximation
end while

système de taille $d \times d$, coûteuse pour les problèmes à grands nombres de degrés de liberté.

Classiquement, pour les problèmes à faibles nombres de paramètres, les points fixes convergent très rapidement (2-3 itérations) et on choisit un nombre fixe d'itérations à réaliser pour chaque mode plutôt qu'un critère de convergence. La procédure gloutonne associée à la PGD permet dans tous les cas de corriger d'éventuelles erreurs liées à la convergence incomplète du point fixe à l'itération suivante. On a cependant observé que ce faible nombre d'itérations n'est pas suffisant pour les plus grands nombres de paramètres (au delà de la vingtaine). C'est donc un critère de convergence qui a été utilisé dans les exemples de PGD classiques traités ci-après, FIG.1.12 par exemple.

4.1.2 Minimal Residual PGD

Cette formulation est a priori plus robuste que la précédente car on peut prouver mathématiquement sa convergence [Nouy et Ladevèze, 2004]. Elle consiste à minimiser à chaque itération le résidu associé à la solution PGD. Ainsi, à l'itération m on cherche à résoudre le problème :

$$\min_{(\underline{\widetilde{\mathrm{X}}}, \{\gamma_i\})} \left\| \underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \left[\underline{\mathbf{X}}_{PGD}^{(m-1)} + \underline{\widetilde{\mathrm{X}}} \prod_{i=1}^{N_p} \gamma_i \right] \right\| \tag{1.46}$$

En dérivant (1.46) par rapport à $\underline{\widetilde{\mathrm{X}}}$ et aux γ_i, on obtient $N_P + 1$ équations qu'on peut résoudre grâce à une stratégie de point fixe exactement comme dans ALG.3. Les détails de cette différenciation sont développés dans un cas particulier en Annexe A. On remarque notamment que les intégrales impliquées sont plus complexes, les coûts de calculs sont donc bien supérieurs à la méthode Progressive Galerkin quand le nombre de paramètres est élevé.

Aucune de ces deux méthodes ne donne généralement la base (spatiale) optimale en terme de nombre de modes qu'une PCA par exemple pourrait fournir à partir d'un ensemble de snapshots balayant tout l'espace paramétrique mais des techniques permettent de s'approcher de l'optimalité [Giacoma *et al.*, 2015]. Par ailleurs, à deux dimensions et notamment en espace-temps, la base temporelle est classiquement « mise à jour » à chaque itération. Cette étape consiste à déterminer grâce à une POD (voir 3.1) un nouveau jeu de fonctions temporelles optimales associées à la base spatiale générée à une itération donnée. Cette technique est cependant coûteuse à mettre en œuvre dès que le nombre de dimensions est supérieur à trois.

Pour retrouver une base optimale, des démarches similaires à celle présentée en 2.2.3 avec l'algorithme CP-ALS permettent aussi de déterminer l'ensemble des modes en une étape. Dans le contexte classique de la PGD à deux dimensions (espace et temps), une méthode telle que l'Optimal Galerkin PGD présentée dans [Nouy, 2010] permet de retrouver exactement cette base.

L'algorithme ALG.3 fait appel à un point fixe mais on peut aussi choisir de résoudre le problème couplé (1.45) via des problèmes aux valeurs propres. En effet, en supposant qu'on n'ait qu'un seul paramètre, on peut toujours substituer la valeur de cette fonction paramétrique donnée par (1.45)(b) dans (1.45)(a) et obtenir un problème proche d'un problème aux valeurs propres généralisées en $\underline{\underline{\widetilde{X}}}$. Cette substitution se généralise aisément pour N_p paramètres.

En particulier, on peut se contenter de ne générer qu'un mode dans une seule dimension si seul celui-ci est nécessaire. Cette stratégie a été retenue dans l'algorithme Extended-PGD présenté en Annexe A et développé dans [Paillet, 2016a]. Cette procédure isole le calcul des fonctions spatiales de l'ensemble des fonctions paramétriques afin d'accélérer la convergence des points fixes à grands nombres de paramètres.

4.2 Limites pour les grands nombres de paramètres

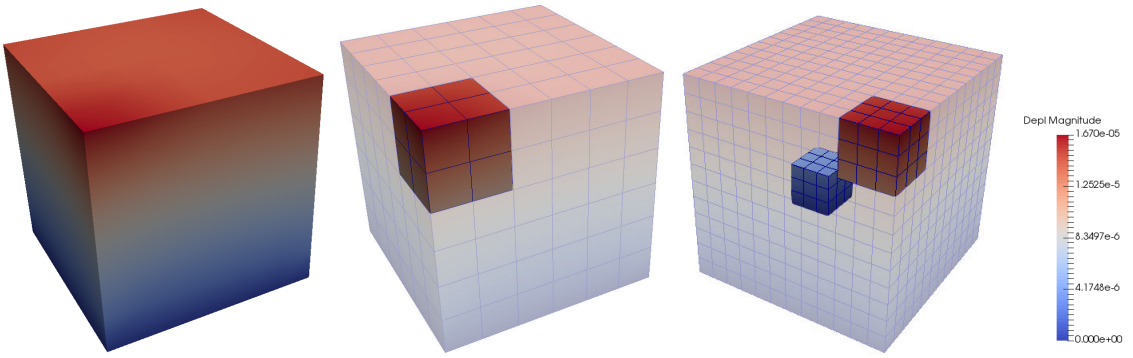
Pour illustrer les limites des algorithmes classiques, on s'est intéressé à un cas particulier du problème 1.1 illustré FIG.1.11. Un cube en traction est décomposé en sous-domaines réguliers, eux-mêmes composés d'éléments réguliers cubiques.

La convergence de l'algorithme ALG.3 est illustrée FIG.1.12 dans trois cas tests, comportant respectivement 8 (comme présenté FIG.2), 27 ou 64 (FIG.1.11) paramètres, en utilisant comme indicateur d'erreur le résidu normalisé :

$$\mathscr{E}^{(m)} = \frac{\left|\left|\left| \underline{\underline{\mathbf{K}}}\,\underline{\mathbf{X}}^{(m)}_{PGD} - \underline{\mathrm{F}} \right|\right|\right|}{\left|\left|\left| \underline{\mathrm{F}} \right|\right|\right|} \tag{1.47}$$

Un tel indicateur représente uniquement la précision de la réduction PGD. Les autres sources d'erreurs telles que les approximations liées à la discrétisation spatiale ne sont pas prises en compte.

La convergence ne peut pas être atteinte en un nombre raisonnable d'itérations quand le nombre de paramètres dépasse 30 et une nouvelle approche doit être intro-



(a) Solution particulière du problème à 27 paramètres

(b) Problème à 27 paramètres : mise en valeur d'un sous-domaine

(c) Problème à 64 paramètres : mise en valeur de deux sous-domaines

FIGURE 1.11 : Problèmes modèles à sous-domaines réguliers

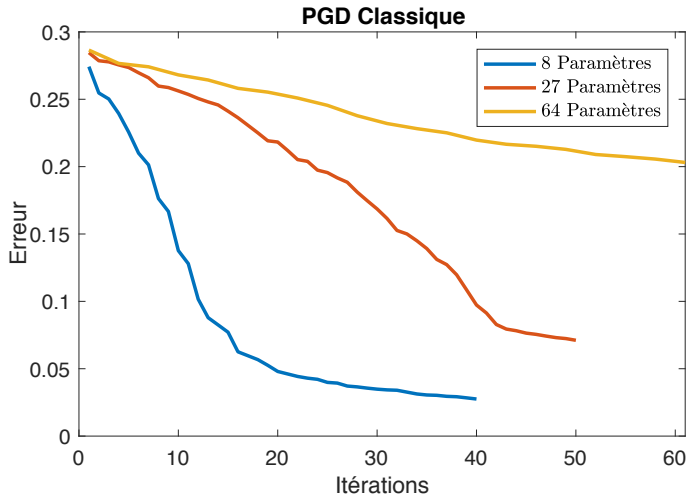
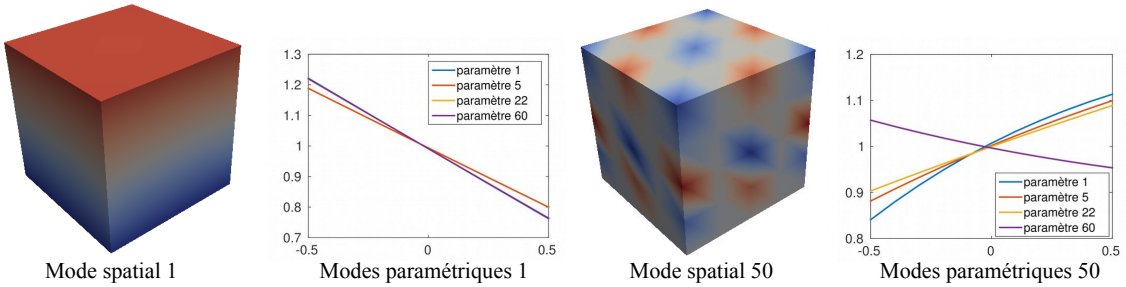


FIGURE 1.12 : Convergence de l'algorithme Progressive Galerkin PGD pour différents nombres de paramètres

duite. On a représenté FIG.1.13 l'allure de la fonction spatiale et de quelques fonctions paramétriques associées au premier et au cinquantième mode de la PGD. On remarque que le mode spatial 50 est un mode très global en espace, il couple fortement des sous-domaines pourtant éloignés. C'est pour éviter ce type de comportement que la nouvelle approche « Multiéchelle en Paramètres » présentée aux chapitres suivants est bâtie. La localisation de la zone d'application des paramètres nous a déjà permis de construire un format de données spécifique, elle est aussi à l'origine des algorithmes de réduction qui utilisent ce format.



Mode spatial 1 | Modes paramétriques 1 | Mode spatial 50 | Modes paramétriques 50

FIGURE 1.13 : Fonctions spatiales et paramétriques pour deux modes d'une PGD classique, problème à 64 paramètres

Chapitre 2

PGD multiéchelle en paramètres : méthodologie discontinue

On introduit dans ce chapitre une nouvelle méthode de discrétisation spatiale : la méthode WTDG. Celle-ci est ensuite utilisée par un algorithme de réduction de modèle consacré aux problèmes à grand nombre de paramètres distribués, la PM-PGD. Des cas-tests prenant en compte jusqu'à 1000 paramètres sont présentés.

Avant d'introduire un nouvel algorithme de réduction de modèle adapté aux problèmes à grandes dimensions, on s'intéresse à l'utilisation pratique du nouveau format introduit en partie 2.3. Celui-ci est défini localement pour chaque sous-domaine, un champ ainsi décrit est donc potentiellement discontinu d'un sous-domaine à l'autre. Si c'est une difficulté pour des EF classiques par exemple, il existe de nombreuses méthodes de discrétisation spatialement discontinues en espace pour lesquelles cette propriété du format ne sera pas une limitation.

1 Discrétisation spatiale discontinue

Les méthodes de Galerkin Discontinue (Discontinuous Galerkin, DG, [Arnold *et al.*, 2002]) ou les volumes finis [Eymard *et al.*, 2000] sont des procédures classiques qui n'imposent pas la continuité des champs approchés entre les éléments de discrétisation. D'autres méthodes telles que les EF étendus (X-FEM,

[Moës *et al.*, 1999]) permettent d'ajouter des discontinuités à des champs initialement continus. Toutes ces approches sont potentiellement compatibles avec la représentation (1.38). Nous développons dans cette partie une approche originale issue des méthodes de Trefftz.

1.1 Méthodes discontinues

Les méthodes de Trefftz [Kita et Kamiya, 1995] forment une catégorie de techniques proches des éléments finis de frontières (BEM, Boundary Elements Methods) [Xiao, 2015]. Celles-ci permettent de résoudre de manière approchée des EDP sans résoudre d'équation localement à l'intérieur du domaine. Elles ne nécessitent qu'une discrétisation de la frontière qui en pratique doit rester simple géométriquement.

Ces techniques sont aujourd'hui largement supplantées par des méthodes plus souples d'utilisation et adaptables aux géométries complexes. C'est le cas des méthodes DG par exemple, très couramment utilisées en dynamique ou en mécanique des fluides notamment du fait de leurs excellentes propriétés de description des problèmes de transport ou de convection. La méthode que nous introduisons est une combinaison de ces deux approches. En ajoutant différentes contraintes aux méthodes de Trefftz, on obtient une formulation très proche des méthodes DG les plus classiques, tout en conservant un fort lien avec des propriétés fondamentales de mécanique des structures.

La notion de sous-domaine associé à un paramètre est dans tout ce chapitre confondue avec celle d'élément, Ω_E étant composé d'un seul élément de la discrétisation spatiale discontinue introduite. Dans la deuxième partie du chapitre, on associe un paramètre à chaque Ω_E pour décrire la solution sous la forme (1.38) localement. Pour l'instant, on résout simplement notre problème modèle (1.3) pour un jeu de paramètres particuliers, ce qui conduit au problème purement spatial suivant :

$$\begin{cases} \underline{u} \in \mathscr{U} = [H^1(\Omega)]^3 \\ \underline{\mathrm{div}}(\boldsymbol{\sigma}) + \underline{f}_d = 0 \quad \text{sur} \quad \Omega \\ \underline{u} = \underline{u}_d \quad \text{sur} \quad \partial_1\Omega \\ \boldsymbol{\sigma}\underline{n} = \underline{F}_d \quad \text{sur} \quad \partial_2\Omega \\ \boldsymbol{\sigma} = \boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} = \frac{1}{2}(\boldsymbol{\nabla}\underline{u} + \boldsymbol{\nabla}\underline{u}^T) \end{cases} \tag{2.1}$$

La loi de Hooke $\boldsymbol{H}_{\boldsymbol{\mu}}$ est une valeur particulière de $\boldsymbol{H}(\boldsymbol{\mu})$ considérée comme constante dans cette partie.

1.1.1 Méthodes de Trefftz

L'esprit de ces méthodes consiste à concentrer tout le problème sur ses conditions limites (CL). On cherche à résoudre un problème très classique (ici l'élasticité linéaire). On en connait analytiquement des solutions homogènes valables en tous points intérieurs du domaine. Dans notre cas, il s'agit de tous les champs à divergence donnée.

Ces solutions sont utilisées comme base de résolution que l'on projette sur le bord du domaine pour en déterminer une combinaison particulière qui respecte au mieux les CL.

Sur l'intérieur Ω du domaine, on choisit donc une base d'approximation composée de fonctions qui respectent localement l'équilibre. En élasticité, on a :

$$\mathscr{U}^T = \left\{ \begin{array}{c} \underline{u}^T \in \left[H^1(\Omega)\right]^3 \\ \underline{\mathrm{div}}\left(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u}^T)\right) - \underline{f}_d = 0 \end{array} \right\} \tag{2.2}$$

Numériquement, on va approcher l'espace $\mathscr{U}^T$ par un espace de dimension finie $\mathscr{U}^{T,P}$, par exemple un ensemble de polynômes d'ordre P fixé. On cherche notre solution approchée $\underline{u}$ sous la forme d'une combinaison linéaire des fonctions $\underline{u}^T \in \mathscr{U}^{T,P}$ qui satisfasse les CL. On cherche donc à avoir :

$$\left\{ \begin{array}{l} \underline{u} - \underline{u}_d = 0 \quad \text{sur} \quad \partial_1\Omega \\ \boldsymbol{\sigma}\,\underline{n} - \underline{F}_d = 0 \quad \text{sur} \quad \partial_2\Omega \end{array} \right. \tag{2.3}$$

On ne peut pas forcément respecter ces égalités exactement en dimension finie, on se contente donc de les satisfaire au mieux via un problème de minimisation. On introduit $\boldsymbol{\sigma}^* = \boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u}^*)$. La formulation faible ci-dessous s'obtient via la méthode de Galerkin :

$$\forall \underline{u}^* \in \mathscr{U}^T \quad \int_{\partial_1\Omega} \boldsymbol{\sigma}^*\,\underline{n}.\left(\underline{u} - \underline{u}_d\right) d\Gamma + \alpha \int_{\partial_2\Omega} \underline{u}^*.\left(\boldsymbol{\sigma}\,\underline{n} - \underline{\mathrm{F}}_d\right) d\Gamma = 0 \tag{2.4}$$

Elle donne une solution unique si $\alpha = 1$ [Hochard *et al.*, 1997]. Il s'agit d'une méthode de Trefftz particulière, différente des méthodes les plus anciennes qui sont généralement basées sur des minimisations de type moindres carrés et qui donnent des problèmes mal conditionnés [Kita et Kamiya, 1995]. Cette formulation, bien que non-symétrique, est bien mieux conditionnée.

L'équation (2.4) est proche de l'esprit original des méthodes de Trefftz [Zieliński, 1988]. Mais seules des fonctions globales sur Ω sont définies, ce qui n'est pas adapté à des domaines non homogènes ou de formes complexes. On introduit donc une décomposition de Ω en différents éléments spatiaux. Les fonctions d'approximation retenues satisfont donc l'équilibre (2.2) sur chaque élément de manière indépendantes. Ainsi, on obtient une représentation potentiellement discontinue de notre solution qui se rapproche d'une méthode EF.

Entre chaque élément (voir FIG.2.1), la solution exacte est continue et les efforts équilibrés. On cherche donc à imposer sur chaque frontière (ou interface) $\Gamma_{EE'}$ entre deux éléments E et E' :

$$\left\{ \begin{array}{l} \underline{u}_E - \underline{u}_{E'} = 0 \quad \text{sur} \quad \Gamma_{EE'} \\ \boldsymbol{\sigma}_E\,\underline{n}_E + \boldsymbol{\sigma}_{E'}\,\underline{n}_{E'} = 0 \quad \text{sur} \quad \Gamma_{EE'} \end{array} \right. \tag{2.5}$$

On note que les normales sont définies sortantes de chaque élément, voir FIG.2.1. Comme précédemment, on résout le problème sous forme faible et la solution du problème sera unique avec la formulation suivante :

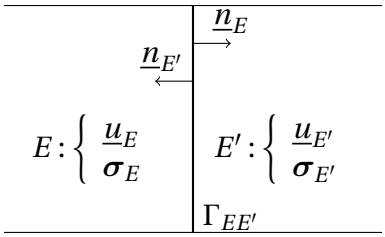


FIGURE 2.1 : Interface entre deux éléments E et E'

$$\begin{aligned} \forall \underline{u}^* \in \mathscr{U}^T(\mathbf{E}) \quad & \int_{\partial_1\Omega} \boldsymbol{\sigma}^* \underline{n}.\left(\underline{u}-\underline{u}_d\right) d\Gamma + \int_{\partial_2\Omega} \underline{u}^*.\left(\boldsymbol{\sigma}\underline{n}-\underline{\mathrm{F}}_d\right) d\Gamma \\ & + \sum_{E\in\mathscr{E}} \sum_{E'\in\widehat{\mathbf{C}}_E^1} \frac{1}{2} \int_{\Gamma_{EE'}} \left(\underline{u}_E^*.\left(\boldsymbol{\sigma}_E\underline{n}_E + \boldsymbol{\sigma}_{E'}\underline{n}_{E'}\right) + \boldsymbol{\sigma}_E^*\underline{n}.\left(\underline{u}_E - \underline{u}_{E'}\right)\right) d\Gamma = 0 \end{aligned} \tag{2.6}$$

où $\widehat{\mathbf{C}}_E^1$ représente l'ensemble des éléments partageant une interface avec E. On remarque qu'on cherche maintenant des fonctions dans l'ensemble $\mathscr{U}^T(\mathbf{E}) = \left\{\underline{u} \middle| \forall E \in \mathbf{E}, \underline{u}_{|E} \in \mathscr{U}^T(\Omega_E)\right\}$, c'est à dire des fonctions respectant l'équilibre intérieur indépendamment sur chaque élément. On peut regrouper les intégrales associées à la même interface en introduisant $\boldsymbol{\Gamma}_{int}$ l'ensemble des interfaces internes de Ω :

$$\begin{aligned} \forall \underline{u}^* \in \mathscr{U}_T(\mathbf{E}) \quad & \int_{\partial_1\Omega} \boldsymbol{\sigma}^* \underline{n}.\left(\underline{u}-\underline{u}_d\right) d\Gamma + \int_{\partial_2\Omega} \underline{u}^*.\left(\boldsymbol{\sigma}\underline{n}-\underline{\mathrm{F}}_d\right) d\Gamma \\ & + \sum_{\Gamma_{EE'}\in\boldsymbol{\Gamma}_{int}} \frac{1}{2} \int_{\Gamma_{EE'}} \left(\left(\underline{u}_E^* + \underline{u}_{E'}^*\right).\left(\boldsymbol{\sigma}_E\underline{n}_E + \boldsymbol{\sigma}_{E'}\underline{n}_{E'}\right) + \left(\boldsymbol{\sigma}_E^*\underline{n}_E - \boldsymbol{\sigma}_{E'}^*\underline{n}_{E'}\right).\left(\underline{u}_E - \underline{u}_{E'}\right)\right) d\Gamma = 0 \end{aligned} \tag{2.7}$$

Remarque 12 *La méthode de Trefftz adaptée à un ensemble de sous-structures est présentée ainsi dans [Hochard* et al.*, 1997]. Elle est dans cet article encore très différente dans l'esprit d'une méthode de type EF. En effet, toutes les sous-structures sont très régulières, de grandes tailles et elles même décomposées via des EF pour décrire les très riches fonctions de formes de* $\mathscr{U}^{T,P}$.

Notation

Introduisons des notations qui permettent à la fois de compacter les expressions présentées et de se rapprocher des formulations classiques associées aux méthodes DG. Sur une interface $\Gamma_{EE'}$ entre deux éléments E et E' orientée de E vers E' (voir FIG.2.1), on définit la moyenne de la contrainte normale et du déplacement par :

$$\left\langle \underline{u} \right\rangle_{EE'} = \frac{\underline{u}_E + \underline{u}_{E'}}{2} \quad \text{et} \quad \left\langle \boldsymbol{\sigma}\underline{n} \right\rangle_{EE'} = \frac{\boldsymbol{\sigma}_E\underline{n}_E - \boldsymbol{\sigma}_{E'}\underline{n}_{E'}}{2} \tag{2.8}$$

Les normales $\underline{n}_E$ et $\underline{n}_{E'}$ sont sortantes, on a bien $\underline{n}_E = -\underline{n}_{E'}$ sur l'interface. De même, le saut d'une grandeur sur $\Gamma_{EE'}$ se note :

$$\left[\underline{u}\right]_{EE'} = \underline{u}_E - \underline{u}_{E'} \quad \text{et} \quad \left[\boldsymbol{\sigma}\underline{n}\right]_{EE'} = \boldsymbol{\sigma}_E\underline{n}_E + \boldsymbol{\sigma}_{E'}\underline{n}_{E'} \tag{2.9}$$

Ces notations nous permettent de réécrire l'équation (2.7) sous la forme :

$$\begin{aligned}\forall \underline{u}^* \in \mathscr{U}^T(\mathbf{E}) \quad & \int_{\partial_1\Omega} \boldsymbol{\sigma}^* \underline{n}.\left(\underline{u}-\underline{u}_d\right) d\Gamma + \int_{\partial_2\Omega} \underline{u}^*.\left(\boldsymbol{\sigma}\underline{n}-\underline{F}_d\right) d\Gamma \\ & + \sum_{\Gamma_{EE'}\in\mathbf{\Gamma}_{int}} \int_{\Gamma_{EE'}} \left(\left\langle \underline{u}^* \right\rangle_{EE'} . \left[\boldsymbol{\sigma}\underline{n}\right]_{EE'} + \left\langle \boldsymbol{\sigma}^*\underline{n} \right\rangle_{EE'} . \left[\underline{u}\right]_{EE'} \right) d\Gamma = 0\end{aligned} \tag{2.10}$$

1.1.2 WTDG : Construction de la formulation

La condition (2.2) est contraignante à mettre en place et impose de choisir analytiquement des bases complexes. On peut se contenter de fonctions qui respectent l'équilibre en moyenne, c'est à dire sous forme faible. C'est ce qui est à la racine des méthodes de Trefftz Faibles (Weak Trefftz, WT) [Ladevèze, 2011]. On cherche une solution donnée par des fonctions respectant :

$$\left\{ \begin{array}{l} \mathscr{U}^{E,WT} \subset \left[H^1(\Omega_E)\right]^3 \\ \text{et} \quad \left\{ \begin{array}{l} \underline{u} \in \mathscr{U}^{E,WT} \\ \underline{u}^* \mathscr{U}_0^{E,WT} \end{array} \right. , \quad \int_{\Omega_E} \underline{u}^*.\left(\underline{\mathrm{div}}\left(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u})\right) - \underline{f}_d\right) d\Omega = 0 \end{array} \right. \tag{2.11}$$

où on a choisi une fonction test admissible à zéro satisfaisant de la même façon :

$$\left\{ \begin{array}{l} \mathscr{U}_0^{E,WT} \subset \left[H^1(\Omega_E)\right]^3 \\ \text{et} \quad (\underline{u},\underline{u}^*) \in \left(\mathscr{U}_0^{E,WT}\right)^2 \quad \int_{\Omega_E} \underline{u}.\underline{\mathrm{div}}\left(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u}^*)\right) d\Omega = 0 \end{array} \right. \tag{2.12}$$

En pratique, on choisit une base de fonctions quelconques, discontinues d'un élément à l'autre $\mathscr{U}^{WT,h} = \left\{ \underline{u} \middle| \forall E \in \mathbf{E}, \underline{u}_{|E} \in \left[H^1(\Omega_E)\right]^3 \right\}$. On impose les minimisations en moyenne (2.11) et (2.12) directement dans la formulation :

$$\begin{aligned}\forall \underline{u}^* \in \mathscr{U}^{WT,h} \quad & \int_{\partial_1\Omega} \boldsymbol{\sigma}^* \underline{n}.\left(\underline{u}-\underline{u}_d\right) d\Gamma + \int_{\partial_2\Omega} \underline{u}^*.\left(\boldsymbol{\sigma}\underline{n}-\underline{F}_d\right) d\Gamma \\ & + \sum_{\Gamma_{EE'}\in\mathbf{\Gamma}_{int}} \int_{\Gamma_{EE'}} \left(\left\langle \underline{u}^* \right\rangle_{EE'} . \left[\boldsymbol{\sigma}\underline{n}\right]_{EE'} + \left\langle \boldsymbol{\sigma}^*\underline{n} \right\rangle_{EE'} . \left[\underline{u}\right]_{EE'} \right) d\Gamma \\ & + \frac{1}{2} \sum_{E\in\mathscr{E}} \int_{\Omega_E} \left(\underline{u}^*.\left(\underline{\mathrm{div}}(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u})) - \underline{f}_d\right) + \underline{u}.\underline{\mathrm{div}}\left(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u}^*)\right) \right) d\Omega = 0\end{aligned} \tag{2.13}$$

La formulation précédente n'est malheureusement pas coercive et on ne sait pas démontrer l'unicité de la solution dans le cas général. Cependant, dans l'ensemble des cas que nous avons traités dans cette thèse, les problèmes discrétisés étaient bien inversibles. Par ailleurs, cette formulation n'est pas symétrique, ce qui n'est pas une limitation en soi mais ne permet pas de définir simplement des normes énergétiques comme on a pu le faire en (1.29) par exemple.

La formulation (2.13) peut être mal conditionnée. On lui ajoute un terme de pénalisation sur le saut de déplacement à chaque interface $\Gamma_{EE'}$: $J_{EE'} = \int_{\Gamma_{EE'}} \left[\underline{u}^*\right].\left[\underline{u}\right] d\Gamma$ et sur chaque bord à déplacement imposé $\Gamma_{EE} \in \partial_1\Omega$, $J_{EE} = \int_{\Gamma_{EE}} \underline{u}^*_E.\left(\underline{u}_E - \underline{u}_d\right) d\Gamma$. Ce terme ne change pas la nature du problème résolu. On a finalement la formulation :

$$\begin{aligned}\forall \underline{u}^* \in \mathscr{U}^h_{WT} \quad & \int_{\partial_1\Omega} \boldsymbol{\sigma}^* \underline{n}.\left(\underline{u}-\underline{u}_d\right) d\Gamma + \int_{\partial_2\Omega} \underline{u}^*.\left(\boldsymbol{\sigma}\underline{n}-\underline{\mathrm{F}}_d\right) d\Gamma \\ & + \sum_{\Gamma_{EE'}\in\boldsymbol{\Gamma}_{int}} \int_{\Gamma_{EE'}} \left(\left\langle \underline{u}^* \right\rangle_{EE'} . \left[\boldsymbol{\sigma}\underline{n}\right]_{EE'} + \left\langle \boldsymbol{\sigma}^*\underline{n} \right\rangle_{EE'} . \left[\underline{u}\right]_{EE'} \right) d\Gamma \\ & + \frac{1}{2} \sum_{E\in\mathscr{E}} \int_{\Omega_E} \left(\underline{u}^*.\left(\underline{\mathrm{div}}(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u})) - \underline{f}_d\right) + \underline{u}.\underline{\mathrm{div}}\left(\boldsymbol{H}_{\boldsymbol{\mu}}\boldsymbol{\varepsilon}(\underline{u}^*)\right) \right) d\Omega \\ & + \sum_{\Gamma_{EE'}\in\boldsymbol{\Gamma}_{int}} \int_{\Gamma_{EE'}} k_0.\left[\underline{u}^*\right].\left[\underline{u}\right] d\Gamma + \sum_{\Gamma_{EE}\in\partial_1\Omega} \int_{\Gamma_{EE}} k_0.\underline{u}^*_E.\left(\underline{u}_E-\underline{u}_d\right) d\Gamma = 0 \end{aligned} \tag{2.14}$$

On choisit d'associer à la pénalisation un poids tel que J soit du même ordre de grandeur que les autres termes de cette minimisation, ce qui conduit dans notre cas-test à choisir :

$$k_0 = \frac{E_0}{L}$$

où L est une longueur caractéristique du domaine Ω et E_0 son module d'Young moyen. Ce raisonnement basé uniquement sur l'homogénéité est simpliste et ce coefficient dépend probablement de la longueur caractéristique des éléments comme pour les méthodes DG classiques.

Remarque 13 *L'appellation DG, Discontinuous Galerkin, est retenue car la méthode de Galerkin est effectivement utilisée pour les minimisations et la solution obtenue sera potentiellement discontinue d'un élément à l'autre. La formulation* (2.14) *n'est cependant pas une méthode DG classique telle que celles formalisées dans [Arnold* et al.*, 2002] et décrites en 1.1.3.*

Remarque 14 *Les méthodes de Trefftz ont été conçues pour utiliser des fonctions de forme très riches, mais avec peu de sous-domaines. Des fonctions sinusoïdales sont sélectionnées pour résoudre des problèmes de vibrations de moyenne fréquence avec la TVRC par exemple [Ladevèze et Riou, 2014]. Ici, on multiplie le nombre de sous-structures qui deviennent simplement nos éléments, tout en simplifiant les fonctions de forme.*

1.1.3 DG : application aux problèmes elliptiques

Les méthodes DG sont nées dans les années 70 [Reed et Hill, 1973], à l'origine pour résoudre des équations hyperboliques. Adaptées aux problèmes de transport, de convection, elles sont tout comme les méthodes de volumes finis particulièrement efficaces appliquées à des équations de conservation. Elles ont ensuite été adaptées aux problèmes elliptiques [Bassi et Rebay, 1997] tels que l'élasticité, même si les conditions de stabilité sont plus contraignantes. L'engouement pour ces méthodes vient du fait qu'elles semblent capable de combiner deux qualités :

— Elles sont comme les volumes finis extrêmement performantes pour les problèmes hyperboliques et peuvent s'adapter à des maillages non conformes. Certains de ces problèmes ont des termes elliptiques non négligeables (convection/diffusion par exemple) [Arnold *et al.*, 2000] et nécessitent des méthodes DG adaptées.

— Comme les éléments finis, elles permettent d'utiliser aisément des fonctions d'ordre élevé et de les adapter sur chaque élément (*hp*-adaptativity [Babuška et Guo, 1992]).

Et bien sûr, elles peuvent s'adapter à des problèmes de géométrie complexe, ce qui n'est pas le cas des Différences Finies par exemple.

On se place dans un espace d'approximation potentiellement discontinu : d'un élément à l'autre, la représentation peut être différente. On cherche la solution approchée $\underline{u}$ du problème (2.1) dans l'espace suivant (broken Sobolev Space) :

$$\left[H^1(\mathbf{E})\right]^3 = \left\{u \in \left[\mathrm{L}^2(\Omega)\right]^3 \,\middle|\, \forall E \in \mathbf{E}, u_{|E} \in \left[H^1(E)\right]^3\right\} \tag{2.15}$$

Numériquement, on se placera dans un espace de dimension finie qui approxime $\left[H^1(\mathbf{E})\right]^3$ en choisissant par exemple des fonctions polynomiales indépendantes sur chaque élément. On utilise classiquement des polynômes de Legendre, ce qui a été appliqué dans [Michoski *et al.*, 2017] par exemple. On n'explicitera pas dans ce manuscrit la méthode de construction de cette approximation bien plus classique que la WTDG et que l'on trouvera par exemple au deuxième chapitre de [Rivière, 2008]. Pour résoudre le problème (2.1), on utilise la formulation :

$$\begin{aligned}
\forall \underline{u}^* \in \left[H^1(\mathbf{E})\right]^3, \quad & \sum_{E\in\mathscr{E}} \int_{\Omega_E} Tr\left(\boldsymbol{H_\mu}\boldsymbol{\varepsilon}(\underline{u})\boldsymbol{\varepsilon}(\underline{u}^*)\right) d\Omega - \int_\Omega \underline{f}_d.\underline{u}^* d\Omega \\
& + \sum_{\Gamma_{EE'}\in\mathbf{\Gamma}_{int}} \int_{\Gamma_{EE'}} \left(-\left\langle \boldsymbol{\sigma}\,\underline{n}\right\rangle_{EE'}.\left[\underline{u}^*\right]_{EE'} + \eta\left\langle \boldsymbol{\sigma}^*\,\underline{n}\right\rangle_{EE'}.\left[\underline{u}\right]_{EE'}\right) d\Gamma \\
& + \int_{\partial_1\Omega} \boldsymbol{\sigma}^*\,\underline{n}.\left(\underline{u}_E - \underline{u}_d\right) d\Gamma - \int_{\partial_2\Omega} \underline{u}^*.\underline{F}_d d\Gamma \\
& + \sum_{\Gamma_{EE'}\in\mathbf{\Gamma}_{int}} \int_{\Gamma_{EE'}} k_0.\left[\underline{u}^*\right].\left[\underline{u}\right] d\Gamma + \sum_{\Gamma_{EE}\in\partial_1\Omega} \int_{\Gamma_{EE}} k_0.\underline{u}^*_E.\left(\underline{u}_E - \underline{u}_d\right) d\Gamma = 0
\end{aligned} \tag{2.16}$$

En fonction du coefficient η retenu, on retrouve plusieurs méthodes DG classiques aux propriétés de convergences différentes. En particulier, en choisissant $\eta = -1$, on obtient une formulation non symétrique (comme WTDG), la NIPG (Nonsymmetric Interior Penalty Galerkin, [Rivière *et al.*, 1999]). La forme bilinéaire associée est coercive, la solution du problème existe et est unique sous certaines conditions sur k_0.

Certains termes tels que ceux associés à la pénalisation sont identiques à ceux de la formulation WTDG (2.14) mais le choix d'ajouter la dénomination DG à la méthode de Weak Trefftz telle que présentée en partie précédente vient surtout de la forme générale des termes de la formulation. Elle est composée d'une somme de termes intérieurs aux éléments et de termes d'interface composés des moyennes ou des sauts des différentes grandeurs. L'esprit de ces deux méthodes est cependant très différent. Tandis que les méthodes DG partent de l'équation d'équilibre locale intégrée faiblement sur chaque élément, la méthode WTDG est construite à partir du respect des conditions aux interfaces et aux bords.

1.2 Mise en œuvre de la WTDG

La méthode WTDG a été implémentée dans le contexte de la mécanique des structures pour la première fois à l'occasion de cette thèse. Elle avait été utilisée sous une forme proche pour résoudre des problèmes d'acoustique de moyenne fréquence dans [Ladevèze et Riou, 2014]. Sa mise en œuvre elle-même n'est pas classique car les fonctions élémentaires retenues sont très différentes des fonctions EF ou DG. On a donc choisi de présenter quelques détails techniques de son implémentation pour trouver une solution particulière de (2.1) pour un jeu de paramètres donné. On se place dans un cas particulièrement simple : le déplacement sera approximé linéairement sur chaque élément. On a donc une contrainte constante par élément et par conséquent, $\underline{\mathrm{div}}(\boldsymbol{\sigma}) = 0$ sur chaque Ω_E. Ici, la version « faible » de la méthode de Trefftz retenue est équivalente à sa version forte.

1.2.1 Choix d'une base

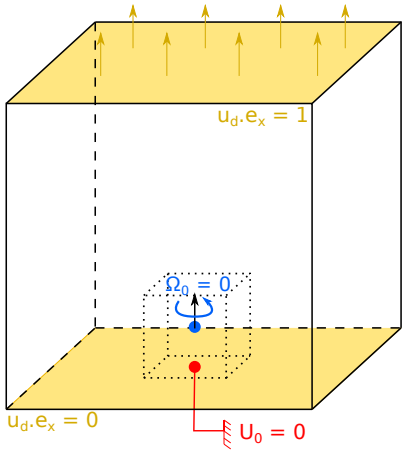


FIGURE 2.2 : Problème modèle : conditions limites

Pour représenter nos fonctions linéaires, on pourrait par exemple utiliser des fonctions de forme P_1 classiques. On préfèrera une formulation plus maniable et bien adaptée au calcul des termes de la forme $\boldsymbol{\sigma}.\underline{u}$ qui n'apparaissent jamais quand des éléments finis sont utilisés (ils sont transformés via les intégrations par parties). Sur un élément de référence dont le déplacement est linéaire, on a :

$$\underline{u}(\underline{x}) = \underline{\mathrm{U}}_{0,E} + \underline{\Omega}_{0,E} \wedge \underline{x} + \boldsymbol{\varepsilon}\,\underline{x} \tag{2.17}$$

avec $\underline{\mathrm{U}}_{0,E}$ et $\underline{\Omega}_{0,E}$, respectivement le déplacement et la rotation de corps rigide de l'élément et $\boldsymbol{\varepsilon}$ la matrice de déformation (constante sur l'élément). On notera le vecteur de déplacement élémentaire :

$$\underline{\mathrm{U}}_E = \begin{bmatrix} \underline{\mathrm{U}}_{0,E} \\ \underline{\Omega}_{0,E} \\ \underline{\hat{\boldsymbol{\varepsilon}}}_E \end{bmatrix} \tag{2.18}$$

où

— $\underline{\mathrm{U}}_{0,E}$ est le déplacement de corps rigide de l'élément
— $\underline{\Omega}_{0,E}$ est la rotation de corps rigide de l'élément par rapport à son centre
— $\underline{\hat{\boldsymbol{\varepsilon}}}$ est la déformation de l'élément en notation de Voigt

Pour construire la formulation, on définit les opérateurs $\underline{\underline{M}}_{pos}$ et $\underline{\underline{M}}_{def}$ tels que :

$$\begin{cases} \underline{u}(x,y,z) = \underline{\underline{M}}_{pos}(x,y,z)\underline{\mathrm{U}}_E \\ \underline{\hat{\boldsymbol{\varepsilon}}}(x,y,z) = \underline{\underline{M}}_{def}(x,y,z)\underline{\mathrm{U}}_E \end{cases} \tag{2.19}$$

En coordonnées cartésiennes et pour des éléments cubiques, on a

$$\underline{\underline{M}}_{pos}(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & 0 & -z & y & x & 0 & 0 & \frac{y}{\sqrt{2}} & \frac{z}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & z & 0 & -x & 0 & y & 0 & \frac{x}{\sqrt{2}} & 0 & \frac{z}{\sqrt{2}} \\ 0 & 0 & 1 & -y & x & 0 & 0 & 0 & z & 0 & \frac{x}{\sqrt{2}} & \frac{y}{\sqrt{2}} \end{bmatrix}$$

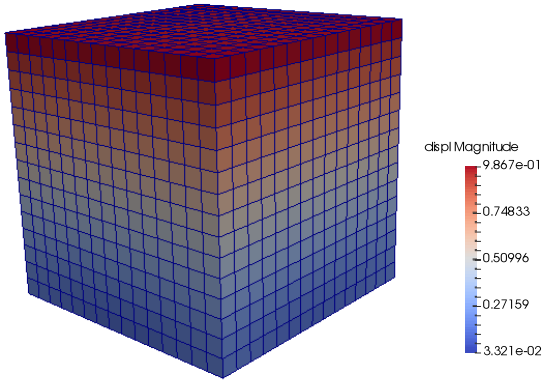
et $\underline{\underline{M}}_{def} = \begin{bmatrix} 0 & \cdots & 0 & 1 & & (0) \\ \vdots & & \vdots & & \ddots & \\ 0 & \cdots & 0 & (0) & & 1 \end{bmatrix}$.

Ainsi, tous les termes de la formulation WTDG discrétisée peuvent être calculés sous forme matricielle. Les conditions limites associées à ce problème sont visibles FIG.2.2. On autorise du glissement sur la face où le cube est encastré et on remarque qu'avec la formulation (2.17) il est très simple de bloquer les mouvements de corps rigide. Contrairement à un problème de rigidité homogène, on n'a pas de plan de symétrie dans ce problème car les modules d'Young de chaque élément sont indépendants et potentiellement différents.

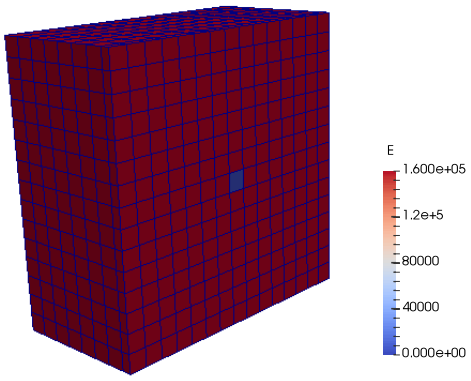
1.2.2 Résultats tridimensionnels

La formulation précédente a été implémentée pour le problème poutre présenté FIG.1.7 et utilisée dans [Ladevèze *et al.*, 2018]. Elle est exacte dans ce cas et permet bien de retrouver la solution analytique du problème. De même en 3D, un cas-test associé à une poutre en traction composée d'un seul élément cubique par section permet de retrouver ces résultats. Le problème modèle implémenté choisi est un problème régulier composé d'éléments cubiques associés au vecteur inconnu (2.18). On observe FIG.2.3 le principe de Saint-Venant, comme on avait pu l'illustrer précédemment avec des éléments finis (voir FIG.1.6). Cette fois-ci cependant, on n'a pas eu besoin d'assembler une matrice de rigidité construite à partir de tous les éléments sauf un. Il a suffi de choisir un module d'Young nul en jouant sur la valeur du paramètre associé à un élément seulement. La formulation (2.14) reste bien inversible.

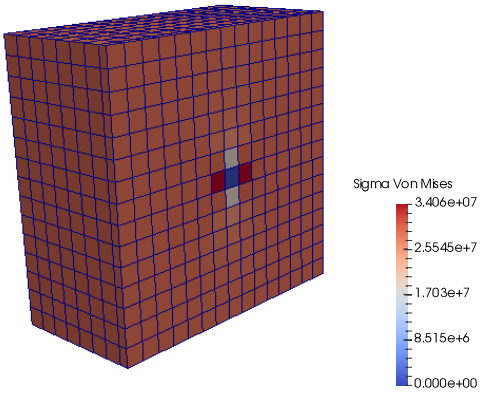
Par ailleurs, la formulation retenue est particulièrement robuste et peut prendre en compte le cas d'un ensemble d'éléments de rigidité nulle. Dans ce cas, certaines parties de la solution n'ont plus de sens physique, mais le problème reste inversible. C'est le cas traité FIG.2.4 par exemple : l'élément matériel central, entouré d'éléments de rigidité nulle « flotte » dans le vide et la solution sur cet élément n'est pas interprétable. Ce problème a cependant été résolu sans difficulté grâce à la formulation (2.14).



(a) Solution en déplacement

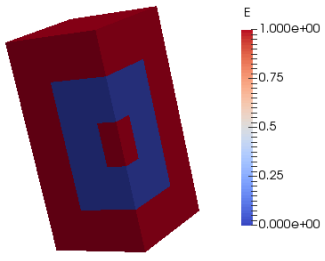


(b) Coupe : valeur élémentaire du module d'Young

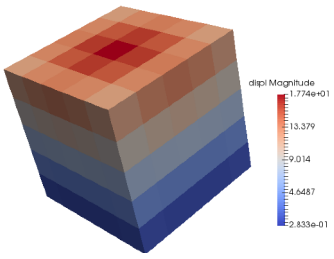


(c) Coupe : contrainte de Von Mises associée

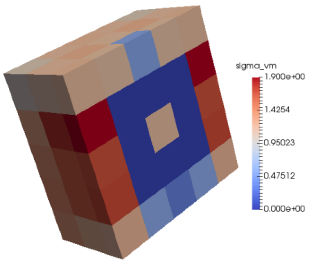
FIGURE 2.3 : Décomposition spatiale WTDG, solution avec élément de rigidité nulle



(a) Un quart du domaine : Module d'Young



(b) Solution complète : déplacement



(c) Coupe : contrainte de Von Mises

FIGURE 2.4 : Résolution via WTDG d'un problème creux

Remarque 15 *Cette formulation étant non-symétrique, certains algorithmes de réduction de modèle classiques tels que* ALG.*3 ne sont plus utilisables et doivent être adaptés [Cancès, E.* et al.*, 2013] si une telle discrétisation spatiale eist retenue.*

2 La Parameter-Multiscale PGD

Cet algorithme est le premier à avoir été implémenté durant de cette thèse et permet de résoudre des problèmes prenant en compte jusqu'à 1000 paramètres. Il comporte cependant des imperfections qui sont corrigés dans l'algorithme introduit au chapitre 3.

De premiers tests monodimensionnels présentés en 2.2.1 permettent de vérifier les ordres de grandeur des gains réalisés grâce à cette nouvelle procédure. Son implémentation tridimensionnelle est ensuite effectuée en parallèle de la mise en œuvre de la WTDG. Matlab est utilisé pour toutes les simulations présentées dans cette partie et les rendus tridimensionnels sont réalisés sous ParaView. Les données tensorielles sont formatées grâce à la Tensor Toolbox déjà mentionnée REM.8. En particulier, la formulation (1.38) est stockée comme en ensemble de k-tensors. La structure globale du code est grandement inspirée de celle développée par Matthieu Vitse dans [Vitse, 2016].

Remarque 16 (PM-PGD) *Le format* (1.38) *est un format à variables séparées (ou « format PGD ») qui introduit deux échelles dans l'espace paramétrique, d'où le nom de PGD multiéchelle en paramètres, ou Parameter Multiscale PGD (PM-PGD). Par extension, on nomme aussi l'algorithme associé PM-PGD. Bien qu'elle soit très éloignée d'un algorithme de PGD classique, la PM-PGD reste une procédure de réduction de modèle greedy basée sur la séparation de variables.*

2.1 Justification théorique et construction de l'algorithme

On cherche à résoudre le problème (1.27) que l'on rappelle ici sous forme discrète en précisant simplement que c'est la méthode WTDG qui a été utilisée pour effectuer la discrétisation spatiale, l'opérateur de rigidité $\underline{\underline{\mathbf{K}}}$ est donc issus de la formulation (2.14) :

$$\begin{aligned} &\text{Trouver } \underline{\mathbf{X}} \in \mathcal{X}^{WT} \text{ où } \mathcal{X}^{WT} = \left\{ \underline{\mathbf{X}} : \boldsymbol{\Sigma}_\mu \to \mathscr{U}^{WT,h} \middle| \int_{\boldsymbol{\Sigma}_\mu} \left\| \underline{\mathbf{X}}(\boldsymbol{\mu}) \right\|^2 d\boldsymbol{\mu} < \infty \right\} \\ &\text{tel que : } \quad \forall \boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu \qquad \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \underline{\mathbf{X}}(\boldsymbol{\mu}) = \underline{\mathrm{F}} \end{aligned} \tag{2.20}$$

2.1.1 Initialisation

On initialise la solution avec le champ moyen, déjà utilisé comme base du développement (1.32) en résolvant le problème :

$$\underline{\underline{\mathbf{K}}}^0 \underline{\mathrm{X}}^0 = \underline{\mathrm{F}} \tag{2.21}$$

On en déduit sur chaque sous-domaine Ω_E une déformation $\boldsymbol{\varepsilon}_E^0$ et donc une contrainte $\boldsymbol{\sigma}_E^0 = \boldsymbol{H}_E^0(\boldsymbol{\varepsilon}_E^0)$. L'opérateur $\boldsymbol{H}_E^0$ est ici considéré comme un opérateur qui traduit l'application de loi de comportement moyenne. On a utilisé la WTDG, on a donc accès très simplement à la déformation via (2.17). Pour simplifier les notations, on suppose dorénavant que $\boldsymbol{\varepsilon}$ et $\boldsymbol{\sigma}$ sont des vecteurs sous leur notation de Voigt et donc que $\boldsymbol{H}_E^0$ est un opérateur matriciel. Dès cette initialisation, les champs doivent donc être stockés numériquement localement sur chaque sous-domaine Ω_E, les différents opérateurs locaux $\boldsymbol{H}_E$ introduisant des variations paramétriques différentes. L'erreur $\mathbf{R}_E$ associée au non respect de la loi de comportement est donnée localement par :

$$\mathbf{R}_E^0 = \boldsymbol{\sigma}_E^0 - \boldsymbol{H}_E\boldsymbol{\varepsilon}_E^0 = (\boldsymbol{H}_E^0 - \boldsymbol{H}_E)\boldsymbol{\varepsilon}_E^0 = -\mu_E\boldsymbol{\sigma}_E^0 \tag{2.22}$$

L'idée fondamentale du nouvel algorithme présenté consiste à chercher à corriger de manière indépendante chacune de ces erreurs locales. L'énergie (ou semi-norme énergétique, voir REM.9) de celle-ci vaut :

$$\left|\left|\left|\mathbf{R}_E^0\right|\right|\right|^2 = \int_{\boldsymbol{\Sigma}_\mu}\int_{\Omega_E} {\mathbf{R}_E^0}^T {\boldsymbol{H}_E^0}^{-1}\mathbf{R}_E^0 d\Omega d\boldsymbol{\mu} = \int_{\boldsymbol{\Sigma}_\mu}\int_{\Omega_E} \mu_E^2.({\boldsymbol{\sigma}_E^0}^T {\boldsymbol{H}_E^0}^{-1}\boldsymbol{\sigma}_E^0) d\Omega d\boldsymbol{\mu} \tag{2.23}$$

Pour définir un estimateur d'erreur local, on la normalise par rapport à l'énergie de la déformation moyenne, qui vaut dans le cas particulier d'un solide déformé de manière homogène : $\left|\left|\left|\boldsymbol{\sigma}_E^0\right|\right|\right|^2 = \int_{\Omega_E} {\boldsymbol{\sigma}_E^0}^T {\boldsymbol{H}_E^0}^{-1}\boldsymbol{\sigma}_E^0 d\Omega$, ici identique sur chaque sous-domaine. Ce qui donne finalement pour notre cas-test :

$$\mathscr{E}_E^0 = \sqrt{\int_{\boldsymbol{\Sigma}_\mu} \mu_E^2 d\boldsymbol{\mu}} \tag{2.24}$$

Comme $\mu_E \in [-\frac{1}{2}, \frac{1}{2}]$, on obtient par exemple $\mathscr{E}_E^0 = 29\%$ pour $\epsilon = 1$ (variation de module d'Young de 50% autour de sa valeur moyenne). On introduit de même un estimateur d'erreur global :

$$\mathscr{E} = \frac{\left|\left|\left|\mathbf{R}\right|\right|\right|}{\left|\left|\left|\boldsymbol{\sigma}^0\right|\right|\right|} \tag{2.25}$$

qui a la même valeur que l'ensemble des erreurs locales après l'initialisation. Tous les estimateurs introduits sont résumés au chapitre suivant TAB.3.1.

2.1.2 Correction des erreurs locales

Sur un sous-domaine Ω_E, l'erreur vaut $\mathbf{R}_E^0 = -\mu_E\boldsymbol{\sigma}_E^0$. On corrige la solution approchée en lui ajoutant un terme destiné à compenser cette erreur locale. Posons :

$$\boldsymbol{\varepsilon}_E^1 = \boldsymbol{H}_E^{-1}\mathbf{R}_E^0 = -\frac{\mu_E}{1+\epsilon\mu_E}{\boldsymbol{H}_E^0}^{-1}\boldsymbol{\sigma}_E^0 \tag{2.26}$$

En ajoutant cette correction $\boldsymbol{\varepsilon}_E^1$, on obtient sur Ω_E la déformation exacte fonction de μ_E.

On ne peut cependant pas ajouter simplement ce terme local à la solution car il a une influence globale. On peut la déterminer exactement via le problème (2.27). On introduit pour cela le tenseur inconnu sur tout le domaine espace-paramètres $\underline{\underline{\mathbf{X}}}_{(\boldsymbol{\varepsilon}_E)}$ tel que :

$$\begin{cases} \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})\underline{\underline{\mathbf{X}}}_{(\boldsymbol{\varepsilon}_E)} = \underline{\mathrm{F}} \\ \boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_E \underline{\underline{\mathbf{X}}}_{(\boldsymbol{\varepsilon}_E)}\right) = \boldsymbol{\varepsilon}_E^1 = \frac{-\mu_E}{1+\epsilon\mu_E} \boldsymbol{H}_E^{0\,-1} \boldsymbol{\sigma}_E^0 \end{cases} \tag{2.27}$$

Seule la déformation $\boldsymbol{\varepsilon}_E^1$ sur Ω_E est connue et considérée comme une déformation imposée. Ce problème est aussi difficile que le problème (2.20) car l'opérateur $\underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})$ est toujours aussi complexe. Si on pouvait le résoudre exactement, on obtiendrait aussi la solution exacte de (2.20).

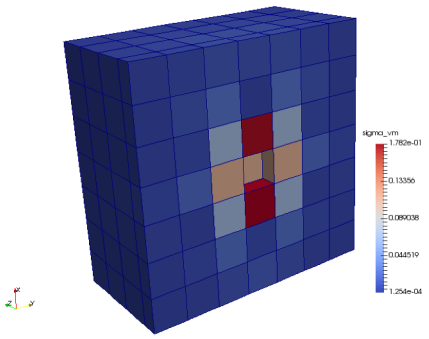


FIGURE 2.5 : Solution du problème (2.28) pour un paramètre localisé au centre de Ω (vue en coupe, 343 paramètres)

On approxime (2.27) pour ne conserver que l'influence spatiale $\underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)}$ de cette déformation imposée $\boldsymbol{\varepsilon}_E^1$ via l'opérateur moyen :

$$\begin{cases} \underline{\underline{\mathbf{K}}}^0 \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)} = \underline{\mathrm{F}} \\ \boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_E \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)}\right) = \boldsymbol{H}_E^{0\,-1} \boldsymbol{\sigma}_E^0 \end{cases} \tag{2.28}$$

Ce problème est purement spatial et le résultat pour un sous-domaine particulier est visible FIG.2.5. On post-traite ensuite la solution de (2.28) pour rétablir les dépendances paramétriques et enrichir le modèle réduit :

$$\underline{\mathbf{X}}^1 = \underline{\mathrm{X}}^0 + \frac{-\mu_E}{1+\epsilon\mu_E} \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)} \tag{2.29}$$

A l'issue de cette étape, l'erreur locale sur Ω_E est parfaitement corrigée. Mais seule la moyenne de la loi de comportement a été utilisée sur l'ensemble des autres sous-domaines, cette approximation induit une nouvelle erreur sur chaque $\Omega_{E'}, E' \neq E$:

$$\mathbf{R}_{E'}^1 = \mathbf{R}_{E'}^0 + \mu_{E'} \frac{\mu_E}{1+\epsilon\mu_E} \boldsymbol{H}_E^0 \boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_{E'} \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)}\right) \tag{2.30}$$

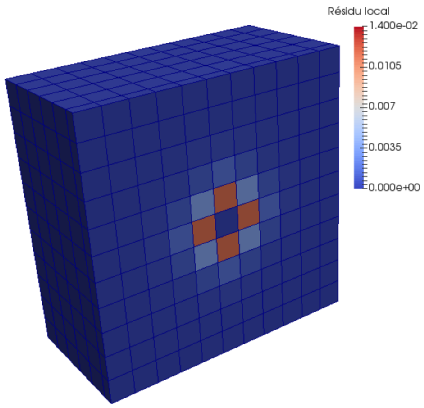


FIGURE 2.6 : Une étape locale : résidu associé à un paramètre localisé sur l'élément central. (1331 éléments)

La norme de $\mathbf{R}^1_{E'}$ est visible FIG.2.6. On exprime sur chaque sous-domaine cette nouvelle erreur sous la forme multiéchelle en paramètres (1.38). Après avoir appliqué cette procédure sur chaque sous-domaine, toutes les erreurs initiales $\mathbf{R}^0_E$ sont localement annulées sur l'ensemble des Ω_E et les erreurs résultantes se somment :

$$\mathbf{R}^{(1,tot)}_{E'} = \sum_{E \neq E'} \mu_{E'} \frac{\mu_E}{1+\epsilon\mu_E} \boldsymbol{H}^0_E \, \boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_{E'} \underline{\widetilde{X}}_{(\boldsymbol{\varepsilon}_E)}\right) \tag{2.31}$$

L'estimateur $\mathscr{E}$ associé est représenté FIG.2.7, il est très homogène pour le problème régulier traité. Pour $\epsilon = 1$, il vaut environ 3.5% quel que soit le nombre d'éléments considérés. On qualifie d'itération globale de notre algorithme cet ensemble de N_p problèmes spatiaux (2.28).

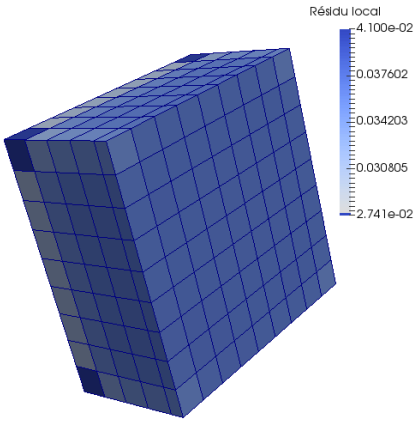


FIGURE 2.7 : Erreur locale après une itération globale, vue en coupe (1000 paramètres)

2.1.3 Structure de l'algorithme

La méthodologie appliquée pour cette première itération se généralise. En effet, à l'itération globale n, on a une erreur composée de N_R termes de la forme :

$$\mathbf{R}_E^n = \sum_{r=1}^{N_R} f^{(r)}(\boldsymbol{\mu}) \widetilde{\mathbf{R}}_E^{(r)} \tag{2.32}$$

On peut l'annuler localement en ajoutant $\boldsymbol{\varepsilon}_E^{n+1} = -\boldsymbol{H}_E^{-1}\mathbf{R}_E^n = \sum_{r=1}^{N_R} \boldsymbol{\varepsilon}_{R_E}^{(r)}$ à la solution et chercher ensuite l'influence globale de cette déformation via l'opérateur moyen :

$$\forall r \in [\![1, N_R]\!], \quad \begin{cases} \underline{\underline{\mathbf{K}}}_0 \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_{R_E})}^{(r)} = \underline{\mathrm{F}} \\ \boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_E \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_{R_E})}^{(r)}\right) = \boldsymbol{\varepsilon}_{R_E}^{(r)} \end{cases} \tag{2.33}$$

puis on somme ces contributions grâce au principe de superposition :

$$\underline{\mathbf{X}}^{n+1} = \underline{\mathbf{X}}^n + \sum_{r=1}^{N_R} \frac{f^{(r)}(\boldsymbol{\mu})}{1+\epsilon\mu_E} \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_{R_E})}^{(r)} \tag{2.34}$$

Algorithm 4 PM-PGD, vision discontinue

Initialisation $\underline{\mathrm{X}}^0 = \underline{\underline{\mathbf{K}}}^{0\,-1}\underline{\mathrm{F}}$, $\underline{\mathbf{X}} = \underline{\mathrm{X}}^0$ ▷ Solution moyenne
$\forall E \in \mathbf{E}$, $\mathbf{R}_E = \boldsymbol{\sigma}_E^0 - \underline{\underline{\boldsymbol{H}}}_E \boldsymbol{\varepsilon}_E^0$ ▷ Résidus locaux
for $i = 0 : I_G$ **do** ▷ I_G, Nombre d'itérations globales
  for $E \in \mathbf{E}$ **do** ▷ Sous-itérations
    Compensation approchée de l'erreur locale sur Ω_E :
    for $r = 1 : N_R$ **do** ▷ N_R, Nombre de modes de $\mathbf{R}_E = \sum_{r=1}^{N_R} f^{(r)}(\boldsymbol{\mu})\widetilde{\mathbf{R}}_E^{(r)}$
      Calcul de $\underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_{R_E})}^{(r)}$ via le problème spatial (2.33)
    end for
    Nouvelle erreur : $\forall E' \neq E \quad \mathbf{R}_{E'} = \mathbf{R}_{E'} + (\boldsymbol{H}_{E'}^0 - \boldsymbol{H}_{E'}) \sum_{r=1}^{N_R} f^{(r)}(\boldsymbol{\mu}) \boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_{E'} \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_{R_E})}^{(r)}\right)$
  end for
  for $E \in \mathbf{E}$ **do** ▷ Mise à jour de la solution
    $\underline{\mathbf{X}}_E = \underline{\mathbf{X}}_E + \sum_{E' \in \mathbf{E}} \sum_{r=1}^{N_R} \frac{f^{(r)}(\boldsymbol{\mu})}{1+\epsilon\mu_{E'}} \underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_{R_{E'}})|E}^{(r)}$ ▷ Format multiéchelle (1.38)
  end for
end for

Comme précédemment, toutes les erreurs locales associées à chacun des sous-domaines sont compensées pour réaliser une itération complète. Chacune de ces nouvelles corrections engendre une erreur sur tous les éléments $E' \neq E$. L'algorithme complet est résumé ALG.4.

2.2 Résultats et discussion

2.2.1 Le cas particulier monodimensionnel en espace

Le cas monodimentionel (problème poutre), en plus d'être simple à implémenter, permet de comparer les solutions obtenues aux solutions analytiques.

Le cas-test retenu est celui d'une poutre en traction telle qu'illustrée FIG.1.7. Les modules d'Young de chacun des éléments sont indépendants et deux solutions particulières à déplacement imposé sont présentées FIG.2.8. On peut constater la robustesse de la méthode discontinue déjà mentionnée en 1.2.2 et l'importance de s'intéresser à un cas test à déplacement imposé pour pouvoir résoudre le problème.

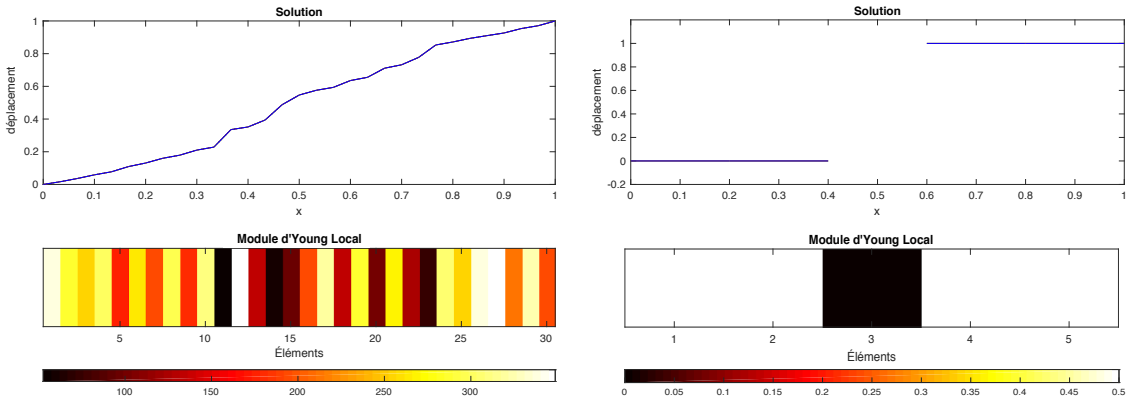


FIGURE 2.8 : Poutre en traction à déplacement imposé : 2 solutions particulières du problème 1D, un paramètre par élément

Cet exemple est cependant très mal adapté à l'étude de la PM-PGD car le principe de Saint-Venant n'est pas respecté par la modélisation poutre. En effet, comme on peut le constater FIG.2.9, une modification locale de la rigidité d'un élément engendre un mouvement de corps rigide (MCR) global. Dans ce cas, le principe de Saint-Venant n'est respecté qu'en contrainte et en déformation. La démarche présentée ALG.4 ne fonctionne qu'avec l'addition d'un fonction globale décrivant ce mouvement de corps rigide.

Le cas monodimensionnel est aussi particulier car la solution exacte est obtenue en une seule itération globale. On peut visualiser le résultat FIG.2.10. Les erreurs obtenues sont indépendantes du nombre de paramètres, ce qui permet de surmonter la « malédiction de la dimensionnalité ».

Dans ce cas particulier, la solution analytique $\underline{\mathbf{X}}_A$ est connue, on peut utiliser l'estimateur d'erreur suivant :

$$\mathscr{E}_{ex} = \frac{|||\underline{\mathbf{X}}_A - \underline{\mathbf{X}}_{PM-PGD}|||}{|||\underline{\mathbf{X}}_A|||} \tag{2.35}$$

Comme toujours, on ne s'intéresse qu'à l'erreur liée au modèle réduit. On compare donc la solution discrète $\underline{\mathbf{X}}_{PM-PGD}$ obtenue à la solution analytique $\underline{\mathbf{X}}_A$ discrétisée aux

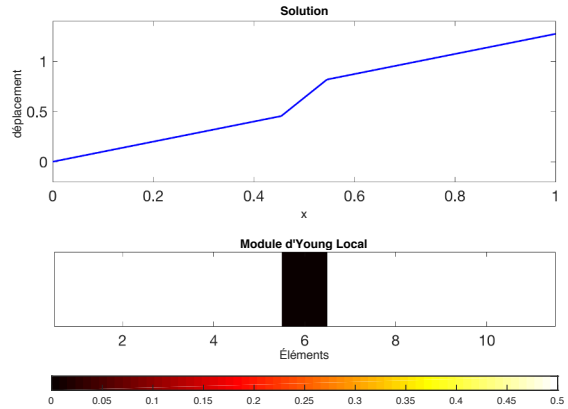


FIGURE 2.9 : Poutre en traction à déplacement imposé : un mouvement de corps rigide affecte tous les éléments situés à droite de l'élément modifié

mêmes points. Trois résultats sont représentés FIG.2.10 :

- En violet la solution associée à l'algorithme PM-PGD présenté ALG.4. Cette erreur relativement élevée est liée à la non prise en compte des mouvements de corps rigides associés à la résolution du problème (2.33) en une dimension.
- En bleu clair la solution obtenue est exacte car on a ajouté un terme correspondant au mouvement de corps rigide visible FIG.2.9. Il s'agit cependant d'une version dégénérée de la PM-PGD puisque que le format multiéchelle (1.38) n'a pas été respecté, toutes les fonctions paramétriques retenues étant des fonctions micro. Cette méthode redonne bien la solution analytique du problème en une itération globale.
- En vert, le format multiéchelle (1.38) a été utilisée, ce qui signifie que les mouvements de corps rigide globaux sont décrits via des fonctions macro linéaires. Le choix a été fait ici de ne garder comme voisinage $\widehat{\mathbf{C}}_E$ que l'élément Ω_E lui-même, qui est le seul déformé par une variation de μ_E (principe de Saint-Venant en déformation). Ainsi l'erreur effectuée est exclusivement liée au format retenu et à la différence entre les fonctions paramétriques globales et leurs approximations linéaires macro. Un voisinage $\widehat{\mathbf{C}}_E$ plus large aurait permis de décrire parfaitement le mouvement de corps rigide sur plusieurs éléments et donc d'obtenir une erreur plus faible.

Le format multiéchelle (1.38) a été conçu en appliquant le principe de Saint-Venant, il n'est pas surprenant que son utilisation engendre d'importantes erreurs dans un cas où il n'est pas respecté. On constatera en partie 2.2.2 que l'utilisation de ce format est tout à fait justifiée pour un problème tridimensionnel.

Remarque 17 *L'erreur exacte $\mathscr{E}_{ex}$ associée à l'utilisation de la PM-PGD n'est pas nulle car le format multiéchelle* (1.38) *a été utilisé. Ce format comporte des approximations liées à la linéarisation de certaines fonctions paramétriques, les fonctions macro. Au contraire, l'estimateur global $\mathscr{E}$ défini en* (2.25) *est égal à 0 à l'issue de cette procédure car il est*

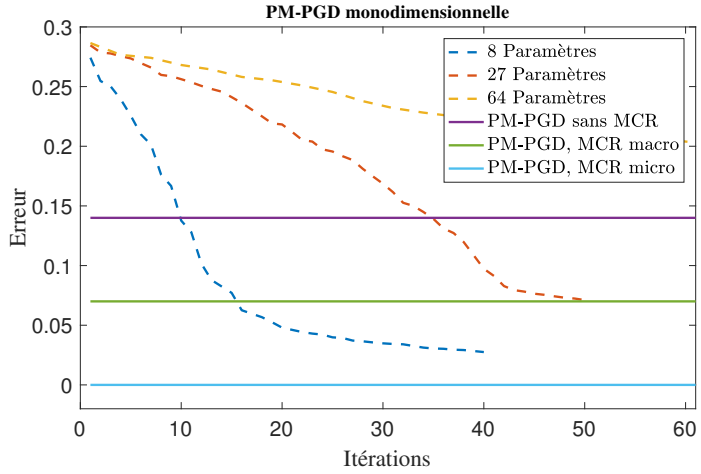


FIGURE 2.10 : Solution monodimensionnelle : comparaison entre la solution issue d'une PGD classique pour différents nombres de paramètres et les erreurs $\mathscr{E}_{ex}$ issues de plusieurs variantes de la PM-PGD

calculé à partir du champ **R**, *lui même décrit sous forme multiéchelle.*

Malgré les difficultés liées à l'aspect monodimensionnel du cas-test, l'algorithme ALG.4 permet bien de résoudre un problème à très grand nombre de paramètres, cette procédure étant indépendante de ce nombre. Ces résultats ont été publiés dans [Ladevèze *et al.*, 2018].

2.2.2 Résultats tridimensionnels en espace

Le problème présenté FIG.2.2 a été résolu dans un cas où chacun des éléments cubiques de la WTDG est associé à un paramètre μ_E proportionnel à son module d'Young. Il s'agit ici d'un choix et non d'une nécessité de la méthode comme en 1.1.

Comme en 1D, la PM-PGD donne des résultats sensiblement identiques quel que soit le nombre de paramètres pris en compte, comme on peut le constater TAB.2.1. Les erreurs locales en espace $\mathscr{E}_E$ associées sont réparties de manière homogène et visibles par exemple FIG.2.7. Ces résultats ont été publiés dans [Paillet *et al.*, 2018].

On s'arrête à l'issue de la deuxième itération qui permet de passer sous le pourcent d'erreur. Il s'agit d'une estimation de l'erreur $\mathscr{E}$, qui présente des biais tels que celui présenté REM.17 . L'utilisation d'autres estimateurs est discutée partie III.2.3.1.

Par ailleurs, la procédure présentée ALG.4 donne des résultats à très grands nombres de modes. A chaque itération globale, on doit ajouter autant de termes que de composantes N_R des erreurs locales $\mathbf{R}_E$, ce qui donne une augmentation exponentielle. Pour obtenir des solutions plus légères et rapides à manipuler, des stratégies de compression des résultats basées sur la CP-ALS (voir ALG.2) par exemple permettent de limiter cet effet et sont présentées en détails en partie III.2.4.4. Les performances de la PM-

	Erreur $\mathscr{E}$ associée à la solution moyenne	Erreur $\mathscr{E}$ après la première itération globale	Erreur $\mathscr{E}$ après la deuxième itération globale
125 paramètres	29%	3.5%	0.49%
216 paramètres	29%	3.7%	0.55%
343 paramètres	29%	3.6%	0.51%
729 paramètres	29%	3.5%	0.53%
1000 paramètres	29%	3.5%	0.53%

TABLEAU 2.1 : Convergence des deux premières itérations globales de l'algorithme ALG.4, problème tridimensionnel, $\epsilon = 1$.

PGD, en terme de coût de stockage entre autre, sont étudiées sur des cas à plus grands nombres de degrés de liberté en partie IV.1.3.

2.3 Influence des paramètres de l'algorithme

2.3.1 Troncature

Comme on peut le constater FIG.2.6, les $\boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_{E'}\underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)}\right)$ sont non négligeables seulement sur un voisinage de Ω_E. On a choisit de profiter de cette propriété en décrivant via des fonctions « macro » l'influence de μ_E sur les sous-domaines éloignés de Ω_E. On peut aller plus loin pour les problèmes à grands nombres de paramètres en définissant un voisinage $\widehat{\mathbf{C}}_E^t$ de Ω_E hors duquel la solution n'est plus calculée. Cette troncature du résultat s'exprime par exemple dans les calculs des nouvelles erreurs (2.31) par la sommation partielle :

$$\mathbf{R}_{E'}^{(1,tot)} = \sum_{E\in\widehat{\mathbf{C}}_{E'}^t} \mu_{E'}\frac{\mu_E}{1+\epsilon\mu_E}\boldsymbol{H}_E^0\,\boldsymbol{\varepsilon}\left(\underline{\underline{\Pi}}_{E'}\underline{\widetilde{\mathrm{X}}}_{(\boldsymbol{\varepsilon}_E)}\right) \tag{2.36}$$

L'ensemble $\widehat{\mathbf{C}}_E^t$ est inclus dans la zone d'influence micro $\widehat{\mathbf{C}}_E$. En pratique, on a retenu les sous-domaines partageant une surface avec Ω_E (voir FIG.1.10), qu'on nommera voisins de Ω_E. En ajoutant les voisins des voisins, puis les voisins des sous-domaines ainsi sélectionnés, on obtient trois niveaux de voisinage qui forment l'ensemble $\widehat{\mathbf{C}}_E^t$ dans nos exemples. Pour les problèmes à plus de 100 paramètres ce choix n'a pas d'influence visible sur les erreurs $\mathscr{E}$ présentées TAB.2.1. Par conséquent, la solution est décrite sur un nombre plus limité de modes. Cette troncature est indispensable pour calculer la seconde itération et éviter une augmentation trop importante du nombre de modes négligeables.

2.3.2 Influence des variations paramétriques

La FIG.2.11 présente les erreurs $\mathscr{E}$ réalisées à l'issue de la deuxième itération globale de l'algorithme pour différentes valeurs de ϵ. Il s'agit ici du cas à 125 paramètres. On

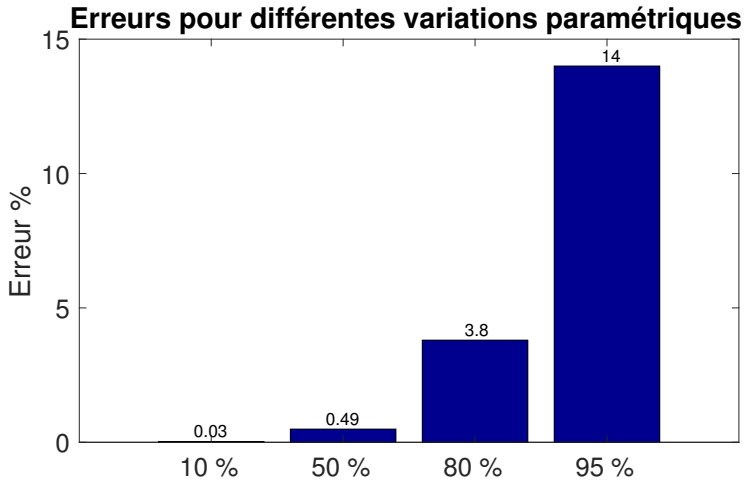


FIGURE 2.11 : Influence des variations paramétriques : variation du paramètre de 10% à 95% autour de sa moyenne ($\epsilon = 0.2$ à 1.9), 125 paramètres

constate que le problème que l'on a retenu (50% de variation autour de la moyenne) est particulièrement facile par rapport à un problème qui s'approche de la prise en compte d'élément à module d'Young égal à zéro, modélisant des éléments vides. Par ailleurs, il a fallu augmenter de beaucoup la précision des fonctions micro pour les cas à plus de 80%. En effet les variations sont loin d'être linéaires et on doit discrétiser les fonctions sur $n_E > 50$ points pour limiter les erreurs d'interpolation.

Les calculs pour des variations paramétriques de 100% ne sont pas accessibles via cet algorithme, non à cause des résolutions spatiales qui fonctionnent pour un élément vide comme nous l'avons illustré FIG.2.3, mais à cause de l'inversion des lois de comportement locales $\boldsymbol{H}_E$. On peut constater que l'expression (2.26) par exemple n'est pas définie pour $\epsilon = 2$.

2.4 Limites de l'algorithme

Cet algorithme a permis avec succès de résoudre des problèmes à plusieurs centaines de paramètres inaccessibles via les méthodes de réductions classiques présentées au chapitre 1.

Cependant l'estimateur d'erreur $\mathscr{E}$ défini en (2.25) ne décrit pas forcément précisément la qualité de la solution. En effet, comme précisé REM.17, il ne prend pas en compte les erreurs liés aux linéarisations des fonctions micro en fonctions macro lors de l'étape (2.34) de l'algorithme. Un estimateur permettant de surmonter cette limite sera introduit en partie III.2.3.1, il est malheureusement trop coûteux pour être utilisé en pratique pour les très grands nombres de paramètres (plusieurs centaines).

De plus, cet algorithme basé sur la discrétisation spatiale WTDG, ne peut pas être adapté simplement à des géométries complexes. Il faudrait pour cela disposer d'un

solveur général WTDG couplé à un mailleur qui puisse prendre en compte différentes formes d'éléments et de géométries. Ce solveur n'existe pas et n'a été implémenté pendant cette thèse. Même si cela avait été fait, la méthode présentée aurait été extrêmement difficile à transférer au milieu industriel : en plus de proposer une méthode de résolution originale, celle-ci nécessiterait un solveur spatial unique. Le choix a été fait d'adapter l'algorithme à des solveurs bien plus courants, et une version compatible avec les éléments finis est présentée au chapitre suivant.

Chapitre 3

PGD multiéchelle en paramètres compatible avec les EF

Une nouvelle version de la PM-PGD est introduite, compatible avec des discrétisations continues en espace telles que les EF. Plusieurs estimateurs sont présentés et utilisés pour analyser quantitativement les sources d'erreurs de la méthode. Des problèmes à grands nombres de paramètres et de géométries complexes sont résolus.

L'algorithme présenté dans ce chapitre permet comme ALG.4 de résoudre des problèmes à très grand nombre de paramètres. Il est conçu pour surmonter les limites mentionnées en II.2.4. En particulier il est compatible avec une discrétisation spatiale de type éléments finis (EF).

On introduira en plus du cas-test illustré FIG.1.1 un cas test de géométrie plus complexe, ainsi que des sous-domaines de formes irrégulières. Il n'est plus nécessaire de choisir des éléments cubiques et on discrétise le domaine spatial grâce à des éléments tétraédriques. Le solveur EF utilisé, *Romlab*, est développé sous Matlab par [Nachar *et al.*, 2019]. La version proposée de l'algorithme est intrusive : on ne peut pas se contenter de l'utilisation d'un solveur considéré comme une « boîte noire » et il est indispensable d'avoir accès aux opérations d'assemblage du code. La possibilité de rendre cette procédure totalement non-intrusive est discutée en partie 3.

1 Compensation de l'erreur

On ne rappelle pas dans cette partie la méthode des EF décrite par exemple dans [Bonnet et Frangi, 2006]. On redonne cependant précisément la définition des espaces de recherche de nos solutions que l'on généralise pour traiter des problèmes paramétriques. On aboutit à la définition d'une erreur qu'on cherche à corriger itérativement grâce à une procédure très proche de celle présentée en II.2.1.2.

1.1 Discrétisation du problème

On se place dans le contexte des EF en déplacement et on cherche des solutions cinématiquement admissibles. Les solutions spatiales du problème de référence (1.3) sont définies dans les espaces suivants :

$$\begin{aligned} \mathscr{U}^{ad} &= \left\{ \underline{u} \middle| \underline{u} \in \left[H^1(\Omega)\right]^3 \text{ et } \underline{u} = \underline{u}_d \text{ sur } \partial_1\Omega \right\} \\ \mathscr{U}_0^{ad} &= \left\{ \underline{u} \middle| \underline{u} \in \left[H^1(\Omega)\right]^3 \text{ et } \underline{u} = 0 \text{ sur } \partial_1\Omega \right\} \end{aligned} \tag{3.1}$$

Remarque 18 (Champs réguliers) *On définit souvent les champs admissibles sous la forme :*

$\mathscr{U}^{ad} = \left\{ \underline{u} \middle| \underline{u} \text{ régulier sur } \Omega \text{ et } \underline{u} = \underline{u}_d \text{ sur } \partial_1\Omega \right\}$

En effet, suivant les CL $\underline{u}_d$ retenues, la condition de « régularité » peut différer [Ladevèze, 1999], en particulier pour les problèmes présentant de fortes discontinuités (fissures ...). On n'aborde pas dans ce manuscrit la question parfois délicate de l'unicité des solutions des problèmes modèles retenus. On se contente de supposer que les CL choisies sont suffisamment « régulières » pour donner des problèmes à solution unique et définie dans $\left[H^1(\Omega)\right]^3$ en 3 dimensions.

Sous sa forme faible en espace, le problème (1.3) devient :

$$\begin{aligned} &\text{Trouver } \boldsymbol{X} \in \mathscr{X} \text{ où } \mathscr{X} : \left\{ \boldsymbol{X} : \boldsymbol{\Sigma}_\mu \to \mathscr{U}^{ad} \middle| \int_{\boldsymbol{\Sigma}_\mu} \|\boldsymbol{X}\|^2 \, d\boldsymbol{\mu} < \infty \right\} \text{ tel que :} \\ &\forall \boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu, \forall \underline{u}^* \in \mathscr{U}_0^{ad}, \\ &\quad -\int_\Omega Tr\left(\boldsymbol{H}(\boldsymbol{\mu})\boldsymbol{\varepsilon}\left(\boldsymbol{X}(\boldsymbol{\mu})\right)\boldsymbol{\varepsilon}\left(\underline{u}^*\right)\right) d\Omega + \int_\Omega \underline{f}_d \underline{u}^* \, d\Omega + \int_{\partial_2\Omega} \underline{F}_d \underline{u}^* \, dS = 0 \end{aligned} \tag{3.2}$$

On discrétise (3.2) sur une base EF de dimension d du sous-espace $\mathscr{U}^h \subset \mathscr{U}^{ad}$. Soit $\underline{u} \in \mathscr{U}^{ad}$, on note $\underline{\widetilde{X}} \in \mathbb{R}^d$ le vecteur des coordonnées dans la base EF de $\underline{u}^h$, projection de $\underline{u}$ sur $\mathscr{U}^h$.

Remarque 19 *Le vecteur $\underline{\widetilde{X}}$ est défini comme étant cinématiquement admissible au sens des EF (voir partie 1.2).*

A l'issue de cette étape et en discrétisant l'espace paramétrique on obtient le problème suivant :

$$\begin{aligned} &\text{Trouver } \underline{\mathbf{X}} \in \mathcal{X} \text{ où } \mathcal{X} : \left\{ \underline{\mathbf{X}} : \boldsymbol{\Sigma}_\mu \to \mathbb{R}^d \middle| \int_{\boldsymbol{\Sigma}_\mu} \left\|\underline{\mathbf{X}}\right\|^2 d\boldsymbol{\mu} < \infty \right\} \text{ tel que :} \\ &\forall \boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu \qquad \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})\underline{\mathbf{X}}(\boldsymbol{\mu}) = \underline{\mathrm{F}} \end{aligned} \tag{3.3}$$

L'opérateur de rigidité EF $\underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})$ est symétrique, contrairement aux opérateurs construits en II.1.1 via la méthode de WTDG par exemple.

1.2 Erreur et admissibilité au sens des EF

1.2.1 Respect de la loi de comportement

La solution du problème (3.2) donne une solution Cinématiquement Admissible (CA) dont on peut extraire la déformation $\boldsymbol{\varepsilon}$. Or, la solution complète du problème mécanique est composée de ce champ de déformation et du champ de contrainte associé $\boldsymbol{\sigma}$ Statiquement Admissible (SA), c'est à dire :

$$\boldsymbol{\sigma} \in \mathscr{S}^{ad} = \left\{ \boldsymbol{\sigma} \middle| \boldsymbol{\sigma} \in \left[\mathrm{L}^2(\Omega)\right]^6 \ , \ \underline{\mathrm{div}}(\boldsymbol{\sigma}) + \underline{f}_d = 0 \text{ sur } \Omega \ , \ \boldsymbol{\sigma}\,\underline{n} = \underline{F}_d \text{ sur } \partial_2\Omega \right\} \tag{3.4}$$

REM.18 s'applique également à l'espace $\mathscr{S}^{ad}$, on a supposé une certaine « régularité » de $\underline{f}_d$ et $\underline{F}_d$. De plus, la solution exacte du problème $(\boldsymbol{\varepsilon}, \boldsymbol{\sigma})$ doit respecter la loi de comportement, ici la loi de Hooke :

$$\boldsymbol{\sigma} = \boldsymbol{H}(\boldsymbol{\mu})\boldsymbol{\varepsilon} \tag{3.5}$$

Soit une solution approchée du problème composée des deux champs $\boldsymbol{\sigma}^a \in \mathscr{S}^{ad}$ et $\boldsymbol{\varepsilon}^a\left(\underline{u}^a\right)$, $\underline{u}^a \in \mathscr{U}^{ad}$. Un bon indicateur de la précision de cette approximation consiste à vérifier le respect de la loi de comportement : $\mathscr{E}_{CR} = \boldsymbol{\sigma}^a - \boldsymbol{H}\boldsymbol{\varepsilon}^a$. Dans le contexte des EF en déplacement, la construction d'un tel indicateur impose de construire un champ SA à partir d'une solution $\boldsymbol{\varepsilon}^a$ issue de la résolution du problème EF. Celle-ci est délicate et a été particulièrement étudiée en validation des simulations pour construire l'Erreur en Relation de Comportement (ERC, [Ladevèze et Chamoin, 2016]).

Dans notre cas, nous allons simplement vérifier si nos champs EF vérifient la loi de comportement. Pour cela, on considère un déplacement $\underline{\widetilde{\mathrm{U}}}$ CA au sens des EF, c'est à dire :

$$\underline{\widetilde{\mathrm{U}}} \in \mathscr{U}^{ad,h} = \left\{ \underline{\widetilde{\mathrm{U}}} = \{u_i\}_{i=1}^d \middle| \underline{u}^h = \sum_{i=1}^{d} u_i \underline{\phi}_i \ , \ u_i = \underline{u}_d(x_i) \text{ si } x_i \in \Gamma_1 \text{ et } \underline{u}^h \in \mathscr{U}^h \right\} \tag{3.6}$$

où les $\underline{\phi}_i$ sont les fonctions de la base EF et Γ_1 la discrétisation EF de $\partial_1\Omega$. De même, on définit un champ SA au sens des EF par :

$$\boldsymbol{\sigma}^h \in \mathscr{S}^{ad,h} = \left\{ \boldsymbol{\sigma} \middle| \forall \underline{u}^* \in \mathscr{U}_0^h, \int_V \left(\boldsymbol{\sigma} : \boldsymbol{\varepsilon}\left(\underline{u}^*\right) - \underline{f}_d \underline{u}^*\right) d\Omega + \int_{\Gamma_2} \left(\boldsymbol{\sigma}\,\underline{n} - \underline{F}_d\right) \underline{u}^* \, dS = 0 \right\} \tag{3.7}$$

où V et Γ_2 correspondent aux maillages de Ω et $\partial_2\Omega$. Plus généralement, on définit les champs :

$$\boldsymbol{\sigma}^T \in \left\{ \boldsymbol{\sigma} \middle| \forall \underline{u}^* \in \mathscr{U}_0^h, \int_V \left(\boldsymbol{\sigma} : \boldsymbol{\varepsilon}\left(\underline{u}^*\right) - \boldsymbol{T} : \boldsymbol{\varepsilon}\left(\underline{u}^*\right)\right) d\Omega = 0 \right\} \tag{3.8}$$

que l'on qualifiera de SA à $\boldsymbol{T}$ au sens des EF, ou SA-$\boldsymbol{T}$ au sens des EF.

$\boldsymbol{\varepsilon}^h\left(\underline{\widetilde{U}}\right), \underline{\widetilde{U}} \in \mathscr{U}^{ad,h}$ et $\boldsymbol{\sigma}^h \in \mathscr{S}^{ad,h}$ étant données, une erreur au sens des EF peut être définie par :

$$\mathbf{R} = \boldsymbol{\sigma}^h - \boldsymbol{H}\boldsymbol{\varepsilon}^h \tag{3.9}$$

Pour un problème de statique non paramétrique résolu par la méthode des EF, cette erreur est triviale et égale à zéro. Cependant dans notre cas, comme $\boldsymbol{\sigma}^h$, $\boldsymbol{\varepsilon}^h$ et $\boldsymbol{H}$ dépendent de tous les paramètres, cette erreur donne une bonne indication de la précision de notre approximation sur tout le domaine paramétrique.

Remarque 20 (Interprétation) *On peut interpréter les définitions précédentes « au sens des EF » simplement : les champs concernés sont des champs discrets solutions d'un problème EF donné et bien posé.*

1.2.2 Correction globale de l'erreur

On reprend à présent des notations continues indépendantes de la discrétisation EF. On retient simplement qu'après discrétisation on définit l'erreur $\mathbf{R}$ grâce aux champs solutions « au sens des EF ». La définition des contraintes admissibles (3.7) dans ce sens est très simple et découle naturellement du produit de la déformation par la loi de comportement. Il est inutile d'employer les outils de reconstruction utilisés dans le contexte de l'ERC par exemple.

L'erreur $\mathbf{R}$ est utilisée à chaque étape de notre algorithme. Celui-ci est conçu pour la corriger aussi précisément que possible à chaque itération. Posons $\mathbf{R}^i = \boldsymbol{\sigma}^i - \boldsymbol{H}\boldsymbol{\varepsilon}^i$ l'erreur à une itération donnée de l'algorithme. Le problème suivant va compenser cette erreur :

$$\begin{cases} \forall \underline{v}^* \in \mathscr{U}_0^{ad}, \quad \displaystyle\int_\Omega Tr\left((\Delta\boldsymbol{\sigma} - \mathbf{R}^i)\Delta\boldsymbol{\varepsilon}(\underline{v}^*)\right) d\Omega = 0 \\ \Delta\boldsymbol{\sigma} = \boldsymbol{H}'\Delta\boldsymbol{\varepsilon}(\underline{u}) \end{cases} \tag{3.10}$$

$\Delta\boldsymbol{\varepsilon}$ est CA-0 et $\Delta\boldsymbol{\sigma}$ est SA-$\mathbf{R}^i$, donc $(\Delta\boldsymbol{\sigma} - \mathbf{R}^i)$ est SA-0. $\boldsymbol{H}'$ est une approximation de la relation de comportement $\boldsymbol{H}$ discutée ci-après. Ainsi, comme l'approximation initiale du problème est donnée par $\boldsymbol{\varepsilon}^i$, qui est CA, et $\boldsymbol{\sigma}^i$, qui est SA, on peut définir une nouvelle solution admissible :

$$\begin{cases} \boldsymbol{\varepsilon}^{i+1} = \boldsymbol{\varepsilon}^i + \Delta\boldsymbol{\varepsilon} \\ \boldsymbol{\sigma}^{i+1} = \boldsymbol{\sigma}^i + (\Delta\boldsymbol{\sigma} - \mathbf{R}^i) \end{cases} \tag{3.11}$$

La nouvelle erreur associée est donnée par $\mathbf{R}^{i+1} = \boldsymbol{\sigma}^{i+1} - \boldsymbol{H}\boldsymbol{\varepsilon}^{i+1} = \boldsymbol{\sigma}^i + (\Delta\boldsymbol{\sigma} - \mathbf{R}^i) - \boldsymbol{H}(\boldsymbol{\varepsilon}^i + \Delta\boldsymbol{\varepsilon}) = \Delta\boldsymbol{\sigma} - \boldsymbol{H}\Delta\boldsymbol{\varepsilon}$.

Si on a résolu le problème en utilisant la loi exacte $\boldsymbol{H}' = \boldsymbol{H}$, alors cette solution est exacte au sens des EF. Cependant, les dépendances paramétriques de $\boldsymbol{H}$ sont trop complexes et seule une approximation de cette loi est utilisée, donnant une nouvelle erreur de la forme :

$$\mathbf{R}^{i+1} = (\boldsymbol{H}' - \boldsymbol{H})\Delta\boldsymbol{\varepsilon} \tag{3.12}$$

2 Construction de l'algorithme et implémentation

2.1 Vision continue en paramètres

On s'intéresse toujours à un problème où chaque paramètre μ_E est associé à un sous-domaine spatial Ω_E et proportionnel au module Young local suivant la relation (1.5).

La procédure présentée ci-dessus en 1.2.2 est basée sur la compensation de l'erreur $\mathbf{R}$ définie en (3.9). Naturellement, comme la loi de comportement $\boldsymbol{H}(\boldsymbol{\mu})$ est définie localement sur chaque sous-domaine, cette erreur sera aussi décomposée en erreurs locales qu'on notera $\mathbf{R}_E$ sur chaque sous domaine Ω_E. Comme en ALG.4, on initialise la solution par le champ $\underline{\widetilde{X}}^0$ associé à la valeur moyenne des paramètres.

Après l'initialisation, on dispose d'une première erreur définie par sous-domaine $\mathbf{R}^0 = \{\mathbf{R}^0_E, E \in \mathbf{E}\}$. On peut compenser cette erreur localement, en corrigeant chaque terme indépendamment, puis sommer les contributions ainsi obtenues. Pour cela, on résout pour chaque $E \in \mathbf{E}$ le problème (3.10). On ne peut cependant pas utiliser la loi de comportement exacte qui est de trop grande dimension. On en utilise une version simplifiée de même nature que celle retenue dans la version discontinue au chapitre précédent. Sur le sous-domaine Ω_E dont on cherche à corriger l'erreur on utilise l'expression exacte de $\boldsymbol{H}$ qui ne dépend que d'un paramètre : μ_E. Ailleurs, on se contentera de la loi de comportement approchée par sa valeur moyenne $\boldsymbol{H}^0$. Ainsi, le problème (3.10) peut se reformuler de la manière suivante :

$$\forall \underline{v}^* \in \mathscr{U}_0^{ad}, \quad \int_{\Omega_E} Tr\left((\boldsymbol{H}_E(\Delta\boldsymbol{\varepsilon}) - \mathbf{R}^0_E)\Delta\boldsymbol{\varepsilon}(\underline{v}^*)\right) d\Omega + \int_{\overline{\Omega}_E} Tr\left((\boldsymbol{H}^0(\Delta\boldsymbol{\varepsilon}))\Delta\boldsymbol{\varepsilon}(\underline{v}^*)\right) d\Omega = 0 \tag{3.13}$$

Rappelons que $\overline{\Omega}_E$ décrit l'ensemble des sous-domaines sauf Ω_E. Cette formulation ne dépend que du seul paramètre μ_E, les autres paramètres étant facilement factorisés (voir (3.18) ci-après). Après résolution de ce problème (3.13), l'erreur devient :

$$\begin{cases} \mathbf{R}^1_E = 0 \text{ sur } \Omega_E \\ \mathbf{R}^1_{E'} = \mathbf{R}^0_{E'} + (\boldsymbol{H}^0 - \boldsymbol{H}_{E'})\Delta\boldsymbol{\varepsilon} \ , \ E' \neq E \end{cases} \tag{3.14}$$

En effet, on a complètement compensé l'erreur locale $\mathbf{R}^0_E$ et les champs correctifs respectent exactement le comportement sur Ω_E. Par contre, ce problème ne corrige en rien les erreurs sur les autres éléments. Ainsi un terme en théorie faible voir complètement négligeable loin de Ω_E s'ajoute hors de ce sous-domaine car la loi de comportement n'y est vérifiée qu'en moyenne. Ces propriétés sont visibles FIG.3.1, sur laquelle on peut aussi vérifier la validité du principe de Saint-Venant sur un sous-domaine de forme quelconque car l'erreur est principalement localisée à la frontière de celui-ci.

En résolvant les problèmes (3.13) pour chaque $E \in \mathbf{E}$, on réalise une itération complète de notre algorithme. Le nouveau résidu obtenu est composé d'un plus grand nombre de termes qui peuvent à leur tour être corrigés de la même manière. La structure de l'algorithme est résumée ALG.5.

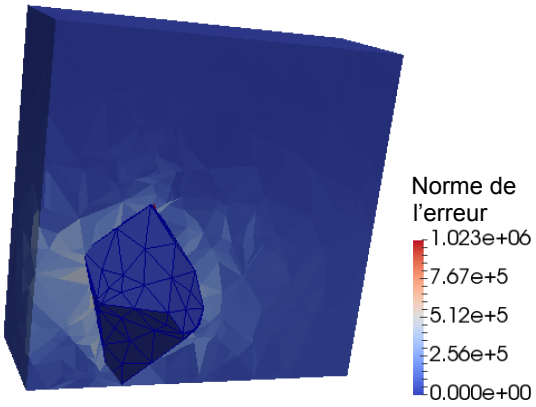


FIGURE 3.1 : Une étape locale : erreur à l'issue d'une sous-itération associée à un paramètre localisé sur le sous-domaine mis en relief (problème à 60 paramètres)

Algorithm 5 PM-PGD compatible avec les EF, présentation continue

Initialisation : $\underline{\widetilde{X}}^0$, $\boldsymbol{\sigma}^0$, $\boldsymbol{\varepsilon}^0$
Erreur initiale : $\mathbf{R}^0 = \boldsymbol{\sigma}^0 - \boldsymbol{H}\boldsymbol{\varepsilon}^0 = \{\mathbf{R}^0_E, E \in \mathbf{E}\}$
for $i = 0 : I_G$ **do** ▷ I_G, Nombre d'itérations globales
  for $E \in \mathbf{E}$ **do** ▷ Sous-itérations
    Calcul de $\Delta\underline{\mathbf{X}}_{(E)}$, $\Delta\boldsymbol{\sigma}$, $\Delta\boldsymbol{\varepsilon}$, solution de (3.13) ▷ Correction de $\mathbf{R}_E$
    $\mathbf{R}^i_E = \Delta\boldsymbol{\sigma} - \boldsymbol{H}\Delta\boldsymbol{\varepsilon}$
  end for
  $\mathbf{R}^i = \sum_{E\in\mathbf{E}} \mathbf{R}^i_E$ ▷ Nouvelle erreur
  $\underline{\mathbf{X}}^i = \underline{\mathbf{X}}^{i-1} + \sum_{E\in\mathbf{E}} \Delta\underline{\mathbf{X}}_{(E)}$ ▷ Mise à jour de la solution
end for

2.2 Algorithme détaillé et discrétisation de l'espace paramétrique

2.2.1 Problème bidimensionnel espace-paramètre

On s'intéresse à présent aux champs espace-paramètres représentés via la structure multiéchelle (1.38) retenue au chapitre 1. Pour alléger les notations, la procédure présentée dans cette partie est décrite sous un format continu. La discrétisation de l'espace paramétrique se fait ensuite très naturellement. On note de manière générale le résidu associé à chaque Ω_E sous la forme :

$$\mathbf{R}_E = \sum_{r=1}^{N_R} \mathbf{R}_E^{(r)} \text{ avec } \mathbf{R}_E^{(r)} = \widetilde{\mathbf{R}}_E^{(r)} \prod_{i=1}^{N_p} \gamma_{i,(E)}^{(r)}(\mu_i) \tag{3.15}$$

Nous avons nommé de manière générale $\gamma_{i,(E)}$ chaque fonction paramétrique locale de $\mathbf{R}_E$ sans préciser son caractère micro ou macro car celui-ci n'a pas d'influence sur la

formulation (3.13). Ainsi, pour chaque composante r de $\mathbf{R}_E$, le problème de minimisation devient :

$$\forall \underline{v}^*(\boldsymbol{\mu}) \in \mathscr{X}_0, \quad \int_{\boldsymbol{\Sigma}_\mu} \int_\Omega Tr\left[\left(\boldsymbol{H}' \Delta\boldsymbol{\varepsilon} - \mathbf{R}_E^{(r)}\right)\boldsymbol{\varepsilon}\left(\underline{v}^*(\boldsymbol{\mu})\right)\right] d\Omega d\boldsymbol{\mu} = 0 \tag{3.16}$$

où $\mathscr{X}_0 = \left\{\boldsymbol{X} : \boldsymbol{\Sigma}_\mu \to \mathscr{U}_0^{ad} \middle| \int_{\boldsymbol{\Sigma}_\mu} \|\boldsymbol{X}\|^2 d\boldsymbol{\mu} < \infty\right\}$. On cherche une solution de (3.16) sous la forme à variables séparées $\Delta\boldsymbol{\varepsilon} = \Delta\widetilde{\boldsymbol{\varepsilon}} \prod_{i=1}^{N_p} g_i(\mu_i)$. On utilise les fonctions-tests suivantes de $\mathscr{U}_0^{ad} \times \mathscr{S}_\mu$:

$$\boldsymbol{\varepsilon}\left(\underline{v}^*(\boldsymbol{\mu})\right) = \widetilde{\boldsymbol{\varepsilon}}(\underline{v}^*) \prod_{j=1}^{N_p} g_j + \Delta\widetilde{\boldsymbol{\varepsilon}} \sum_{i=1}^{N_p} \lambda_i^*(\mu_i) \prod_{j\neq i} g_j \tag{3.17}$$

Pour chaque paramètre $i \neq E$, en choisissant toutes les fonctions-tests nulles sauf $\lambda_i^*(\mu_i)$, on particularise l'expression (3.16) et on obtient un problème de la forme :

$$\int_{\boldsymbol{I}_i} \left(C_1 g_i(\mu_i) + C_2 \gamma_{i,(E)}^{(r)}(\mu_i)\right) \lambda_i^*(\mu_i) d\mu_i = 0 \tag{3.18}$$

où C_1 et C_2 sont des constantes d'intégration. Comme les fonctions paramétriques sont normalisées, on obtient $g_i = \gamma_{i,(E)}^{(r)}$ pour chaque $i \neq E$. Il ne reste donc plus qu'à résoudre un problème à deux paramètres : l'espace et μ_E.

$$\begin{aligned} \forall \underline{v}^*(\mu_E) \in \mathscr{X}_{0,E}, \quad & \int_{\boldsymbol{I}_E} \int_{\Omega_E} Tr\left[\left(\boldsymbol{H}_E \Delta\boldsymbol{\varepsilon} - \mathbf{R}_E^{(r)}\right)\boldsymbol{\varepsilon}\left(\underline{v}^*(\mu_E)\right)\right] d\Omega d\mu_E \\ & + \int_{\boldsymbol{I}_E} \int_{\overline{\Omega}_E} Tr\left[\left(\boldsymbol{H}^0 \Delta\boldsymbol{\varepsilon}\right)\boldsymbol{\varepsilon}\left(\underline{v}^*(\mu_E)\right)\right] d\Omega d\mu_E = 0 \end{aligned} \tag{3.19}$$

où $\mathscr{X}_{0,E} = \left\{\boldsymbol{X} : \boldsymbol{I}_E \to \mathscr{U}_0^{ad} \middle| \int_{\boldsymbol{I}_E} \|\boldsymbol{X}\|^2 d\boldsymbol{\mu} < \infty\right\}$. La formulation (3.19) se résout via une PGD classique telle que celle présentée en partie I.4.

Remarque 21 (Résolution par POD) *On pourrait aussi choisir de ne pas appliquer la méthode de Galerkin sur $\boldsymbol{I}_E$ et considérer que* (3.19) *est un problème purement spatial mais dépendant d'un paramètre. En échantillonnant celui-ci on obtient un ensemble de solutions sur lesquelles on peut appliquer une POD (voir I.3.1). En particulier on peut construire la solution complète espace-paramètre en utilisant la discrétisation « micro » associée à μ_E. Ce champ étant bidimensionnel, une approximation à variables séparées optimale est obtenue simplement grâce à une SVD (voir I.2.1).*

Une PGD et une approximation du champ bidimensionnel via une SVD (comme décrit REM.21) ont été codées et donnent des résultats équivalents. Lors de la première itération globale, un seul mode suffit à donner la solution exacte du problème (3.19). Pour les itérations suivantes, les solutions sont très fortement séparables et seuls $M = 2$ ou $M = 3$ modes sont conservés, générant des erreurs de troncature de moins de 2%.

Algorithm 6 PM-PGD compatible avec les EF, présentation discrétisée en paramètres

Initialisation : $\underline{\widetilde{\mathrm{X}}}^0$, $\boldsymbol{\sigma}^0$, $\boldsymbol{\varepsilon}^0$

$\underline{\mathbf{X}}^0 = \underline{\widetilde{\mathrm{X}}}^0 \bigotimes_{i=1}^{N_p} \underline{1}_i = \left\{ \underline{\widetilde{\mathrm{X}}}^0_E \bigotimes_{i=1}^{N_p} \underline{1}_{i,(E)}, E \in \mathbf{E} \right\}$

Erreur initiale : $\mathbf{R}^0 = \boldsymbol{\sigma}^0 - \boldsymbol{H}\boldsymbol{\varepsilon}^0 = \left\{ \widetilde{\mathbf{R}}_E \bigotimes_{i=1}^{N_p} \underline{\gamma}_{i,(E)}, E \in \mathbf{E} \right\} = \left\{ \mathbf{R}_E, E \in \mathbf{E} \right\}$

for $i = 1 : I_G$ **do** ▷ I_G, Nombre d'itérations globales

  for $E \in \mathbf{E}$ **do** ▷ Sous-itérations

    for $r = 1 : N_R$, composantes de $\mathbf{R}_E$ **do**

      Calcul de M modes, solution du problème (3.19) : ▷ $M = 2$ ou 3

      $\Delta\underline{\widetilde{\mathrm{X}}}^{(m)}$, $\Delta\widetilde{\boldsymbol{\varepsilon}}^{(m)}$, $\Delta\widetilde{\boldsymbol{\sigma}}^{(m)}$ et $\underline{g}_E^{(m)}$, $m \in [\![1, M]\!]$

      Correction de la solution : $\Delta\underline{\mathbf{X}}^{(r)}_{(E)}(\mu_E) \approx \sum_{m=1}^{M} \Delta\underline{\widetilde{\mathrm{X}}}^{(m)} \otimes \underline{g}_E^{(m)}$

      Nouvelle erreur associée :

      $\mathbf{R}_E^{(r)} = \left\{ \left(\sum_{m=1}^{M} \left[\Delta\widetilde{\boldsymbol{\sigma}}^{(m)}_{|E'} - \boldsymbol{H}_{E'} \Delta\widetilde{\boldsymbol{\varepsilon}}^{(m)}_{|E'} \right] \otimes \underline{g}_E^{(m)} \right) \bigotimes_{j \neq E} \underline{\gamma}^{(r)}_{j,(E)}, \forall E' \neq E \right\}$

    end for

    Sommation des contributions de l'erreur : $\mathbf{R}_E^i = \sum_{r=1}^{N_R} \mathbf{R}_E^{(r)}$, $\circledast^1$

    Correction de la solution : $\Delta\underline{\mathbf{X}}^i_{(E)} = \left\{ \sum_{r=1}^{N_R} \Delta\underline{\mathbf{X}}^{(r)}_{(E)|E'} \bigotimes_{j \neq E} \underline{\gamma}^{(r)}_{j,(E)}, \forall E' \in \mathbf{E} \right\}$

  end for

  $\mathbf{R}^i = \sum_{E \in \mathbf{E}} \mathbf{R}^i_E$ ▷ Nouvelle erreur

  $\underline{\mathbf{X}}^i = \underline{\mathbf{X}}^{i-1} + \sum_{E \in \mathbf{E}} \Delta\underline{\mathbf{X}}^i_{(E)}$, $\circledast^2$ ▷ Incrémentation de la solution

  $\underline{\mathbf{X}}^1 = \underline{\mathbf{X}}^0 + \Delta\underline{\mathbf{X}}_{(E)}$

end for

Sauf mention contraire c'est la PGD, moins coûteuse dans ce contexte, qui a été retenue pour tous les exemples numériques présentés dans ce chapitre et le suivant.

ALG.6 résume la procédure décrite dans cette partie. On a choisi d'y représenter les champs sous leur forme discrétisée en paramètres.

Remarque 22 (Nombre de modes) *Comme on l'a déjà constaté pour l'algorithme précédent en II.2.2.2, le nombre de modes augmente exponentiellement à chaque itération globale. En pratique, on a théoriquement 1 mode après l'initialisation et* $(N_p \times M)^i$ *après la i-ème itération globale. Des étapes de compressions indiquées par les symboles* $\circledast$ *dans* ALG.6 *permettent de conserver des solutions de tailles raisonnables et sont discutées en 2.4.4.*

2.2.2 Mise à jour de l'erreur et de la solution

Les composantes spatiales de chaque terme correctif $\Delta\underline{\mathbf{X}}^{(r)}_{(E)}$ solution du problème (3.19) sont globales en espace, elles ont donc une influence sur tout le domaine. On constate ALG.6 lignes 10 et 13 qu'il est indispensable de décrire les nouveaux champs sous une forme locale en espace pour respecter le format retenu. On doit donc extraire les restrictions des champs spatialement sur chaque sous-domaine $\Omega_{E'}$, opération notée $\widetilde{A}_{|E'}$ pour un champ quelconque $\widetilde{A}$.

Par ailleurs, il faut aussi adapter le format (micro ou macro) des fonctions paramétriques pour qu'il soit compatible avec le format de chaque sous-domaine. Par exemple, la solution est mise à jour (ALG.6 ligne 13) en calculant pour chaque $\Omega_{E'}$ la représentation micro-macro associée en 4 étapes :

- on extrait du champ global $\Delta\underline{\widetilde{\mathbf{X}}}^{(r)}_{(E)}$ les degrés de liberté spatiaux associés à $\Omega_{E'}$, $\Delta\underline{\widetilde{\mathbf{X}}}^{(r)}_{(E)|E'}$.
- les fonctions macro $\underline{\boldsymbol{\gamma}}^{(r)}_{j,(E)}$ avec $j \notin \widehat{\mathbf{C}}_E$ et $j \notin \widehat{\mathbf{C}}_{E'}$ sont simplement reproduites, de même que les fonctions micro communes à $\widehat{\mathbf{C}}_E$ et $\widehat{\mathbf{C}}_{E'}$.
- les fonctions macro $\underline{\boldsymbol{\gamma}}^{(r)}_{j,(E)}$ avec $j \notin \widehat{\mathbf{C}}_E$ mais $j \in \widehat{\mathbf{C}}_{E'}$ sont transformées en fonction micro : on les interpole linéairement sur tous les points de la discrétisation micro.
- les fonctions micro $\underline{\boldsymbol{\gamma}}^{(r)}_{j,(E)}$ avec $j \in \widehat{\mathbf{C}}_E$ mais $j \notin \widehat{\mathbf{C}}_{E'}$ ou la fonction $\underline{g}^{(m)}_E$ si $E \notin \widehat{\mathbf{C}}_{E'}$ sont transformées en fonctions macro, on conserve les deux points extrémaux (interpolation linéaire).

Remarque 23 (Perte de continuité) *Cette dernière étape est d'une grande importance : il s'agit de la seule opération de tout l'algorithme où des discontinuités peuvent apparaitre entre deux sous-domaines. Ces approximations restent très raisonnables, car elles sont basées sur le principe de Saint-Venant qui a conduit à la formulation multiéchelle* (1.38)*. Cependant, elles ont un inconvénient majeur : la solution n'est plus continue spatialement et n'est donc plus CA. Ces approximations créent des erreurs qui sont simplement négligées dans la procédure. Nous avons étudié et quantifié ces approximations pour montrer qu'elles sont effectivement minimes en partie 2.4.1.*

2.3 Résultats

L'utilisation des EF permet une grande liberté dans le choix des cas traités. Deux géométries de domaines Ω ont été testées, le cube en traction déjà présenté et une forme « en L » (ou L-shape, voir FIG.3.2). Les sous-domaines qui les décomposent peuvent être des cubes réguliers ou bien des polyèdres de formes aléatoires qui imitent les structures naturelles des solides polycristalins. Ces sous-structures complexes sont réalisées grâce au logiciel Neper [Quey *et al.*, 2011] qui utilise un algorithme de Voronoï pour générer de tels cristaux.

Les cas à grands nombres de degrés de libertés et grands nombres de paramètres sont présentés au chapitre 4 lors de l'étude des performances de la PM-PGD.

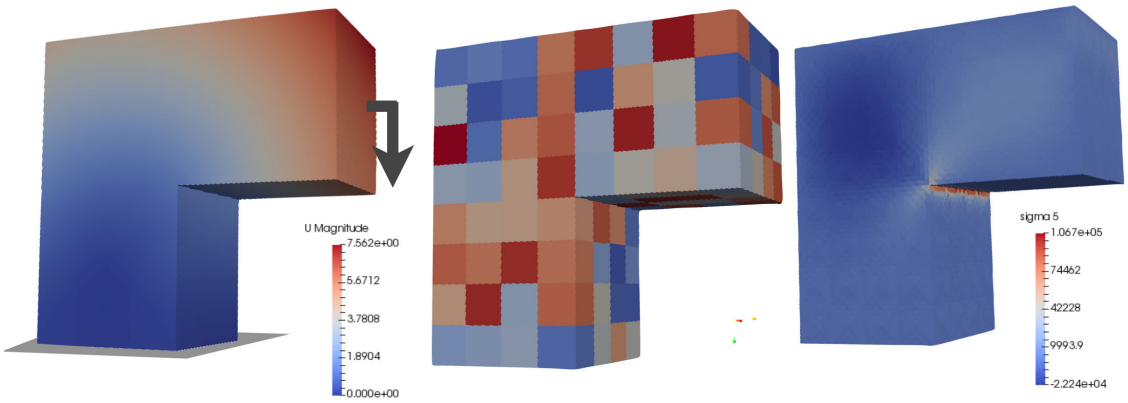


FIGURE 3.2 : Problème « en L » (L-shape) : conditions limites et solution moyenne en déplacement ; exemple de décomposition en 192 sous-domaines ; solution moyenne en contrainte

2.3.1 Différents estimateurs d'erreur

On a déjà introduit depuis le chapitre 2 l'estimateurs d'erreur $\mathscr{E}$ qui correspond à la normalisation de l'erreur $\mathbf{R}$ minimisée itérativement à chaque étape de ALG.5 et qu'on cherche à compenser exactement dans sa version discontinue ALG.4. Cette erreur quantifie le non-respect de la loi de comportement paramétrique au sens des EF.

Or cet estimateur est défini à partir de champs décrits sous le format multiéchelle (1.38). Rappelons la forme de cette erreur à l'initialisation par exemple : $\mathbf{R}^0 = \boldsymbol{\sigma}^0 - \boldsymbol{H}\boldsymbol{\varepsilon}^0 = \left\{ \tilde{\mathbf{R}}^0_E \bigotimes_{i=1}^{N_p} \underline{\gamma}^0_{i,(E)}, E \in \mathbf{E} \right\} = \{\mathbf{R}_E, E \in \mathbf{E}\}$. Cette expression étant stockée sous un format différent sur chaque sous-domaine, il est nécessaire de calculer indépendant chaque erreur locale grâce à la norme $|||\bullet|||_E$ avant de les sommer pour obtenir l'erreur globale.

Cet indicateur ne permet pas de prendre en compte l'erreur liée au format multiéchelle, comme on l'a déjà constaté pour le cas discontinu REM.17. Dans cette nouvelle version de l'algorithme, l'apparition de ces erreurs est liée exclusivement au changement de format lors des étapes de mise à jour de la solution ou des erreurs $\mathbf{R}$ (voir REM.23). On peut simplement évaluer cette erreur en comparant la solution PM-PGD à une solution monoéchelle discrétisée exclusivement avec des fonctions micro.

On introduit les nouvelles notations $\mathbf{R}_{mM}$ et $\mathbf{R}_m$ qui décrivent respectivement la valeur de l'erreur en utilisant l'approximation micro-macro (1.38) et sa valeur pour une description exclusivement micro. On note leur différence $\Delta\mathbf{R} = \mathbf{R}_{mM} - \mathbf{R}_m$ et on définit l'estimateur d'erreur micro-macro par :

$$\mathscr{E}_{mM} = \frac{|||\boldsymbol{\sigma} - \boldsymbol{H}\boldsymbol{\varepsilon} + \Delta\mathbf{R}|||}{|||\boldsymbol{\sigma}_0|||} \tag{3.20}$$

Comme le calcul de $\Delta\mathbf{R}$ impose d'une part l'interpolation de toutes les fonctions

Notation	Expression	Description	Coût
$\mathscr{E}$	$\frac{\|\|\|\boldsymbol{\sigma} - \boldsymbol{H}\boldsymbol{\varepsilon}\|\|\|}{\|\|\|\boldsymbol{\sigma}_0\|\|\|}$	Erreur minimisée dans les algorithmes PM-PGD. Elle surestime la qualité du modèle réduit calculé car toutes les approximations ne sont pas prises en compte	modéré
$\mathscr{E}_{mM}$	$\frac{\|\|\|\boldsymbol{\sigma} - \boldsymbol{H}\boldsymbol{\varepsilon} + \Delta\mathbf{R}\|\|\|}{\|\|\|\boldsymbol{\sigma}_0\|\|\|}$	Corrige l'estimateur précédent pour prendre en compte toutes les approximations	très élevé
$\mathscr{E}_{loc}$	$\max_{i\in I_{part}} \frac{\left\|\underline{\mathrm{X}}^{(i)}_{exact} - \underline{\mathrm{X}}^{(i)}_{PM-PGD}\right\|}{\left\|\underline{\mathrm{X}}^{(i)}_{exact}\right\|}$	Cet estimateur local (dans le domaine paramétrique) donne l'erreur maximale réalisée par le modèle réduit sur I_{part} jeux de paramètres choisis aléatoirement	faible si $\|I_{part}\|$ reste raisonnable
$\mathscr{E}_{ex}$	$\frac{\|\|\|\underline{\mathbf{X}}_A - \underline{\mathbf{X}}_{PM-PGD}\|\|\|}{\|\|\|\underline{\mathbf{X}}_A\|\|\|}$	Calcul de l'erreur exacte, n'est possible que si la solution analytique du problème est connue, en 1D par exemple	modéré

TABLEAU 3.1 : Différents estimateurs d'erreur utilisés pour contrôler la qualité des modèles réduits

macro de $\mathbf{R}_{mM}$ sur la grille micro et d'autre part un deuxième calcul complet en n'utilisant pas le format (1.38), cet estimateur d'erreur est extrêmement coûteux et n'est jamais calculé pour les plus gros cas-tests. Les différents indicateurs d'erreur introduits sont résumés TAB.3.1.

Remarque 24 (Quantification des troncatures) *Les étapes de troncatures décrites précisément en II.2.3.1 pourraient aussi être prises en compte par cet estimateur, mais son coût est prohibitif pour les très grands nombres de paramètres.*

On peut déterminer l'erreur réellement commise par le modèle réduit pour un jeu particulier de paramètres $\boldsymbol{\mu}$ en calculant :

$$\mathscr{E}_{part} = \frac{\left\|\underline{\mathrm{X}}_{exact} - \underline{\mathrm{X}}_{PM-PGD}\right\|}{\left\|\underline{\mathrm{X}}_{exact}\right\|} \tag{3.21}$$

où les champs spatiaux $\underline{\mathrm{X}}_{exact}$ et $\underline{\mathrm{X}}_{PM-PGD}$ sont respectivement la solution exacte du problème (3.3) pour le jeu $\boldsymbol{\mu}$ et l'interpolation du modèle réduit $\underline{\mathbf{X}}$ issu de ALG.5 pour les paramètres $\boldsymbol{\mu}$.

En pratique, on calcule cette erreur particulière (3.21) pour un certain nombre I_{part} de jeux de paramètres et on en déduit un estimateur d'erreur en gardant la valeur maximale obtenue. Les tirages sont réalisés aléatoirement suivant une loi uniforme sur le domaine de définition des paramètres comme précisé TAB.3.1. En pratique, on a retenu 50 solutions particulières dans les cas présentés dans ce manuscrit. C'est un indicateur peu coûteux mais il est finalement proche de l'erreur $\mathscr{E}_{mM}$ comme on le vérifie par exemple FIG.3.3. On l'utilisera en particulier pour les cas à très grands nombres de paramètres.

Cet estimateur a de plus l'avantage de prendre en compte l'éventuelle erreur d'interpolation dans l'espace paramétrique car on n'impose pas de choisir des points sur la grille micro pour déterminer les I_{part} solutions. Il est par contre dépendant des tirages aléatoires retenus et ne présente aucune garantie .

Remarque 25 (Solution de référence) *Quel que soit l'estimateur retenu, on ne considère jamais l'erreur liée à la discrétisation spatiale. La solution EF est toujours considérée comme notre solution de référence. On pourrait affiner nos estimateurs en y incluant l'Erreur en Relation de Comportement [Ladevèze et Chamoin, 2016] associée à la discrétisation EF afin de prendre en compte les erreurs liées aux maillages parfois grossiers qu'on utilise.*

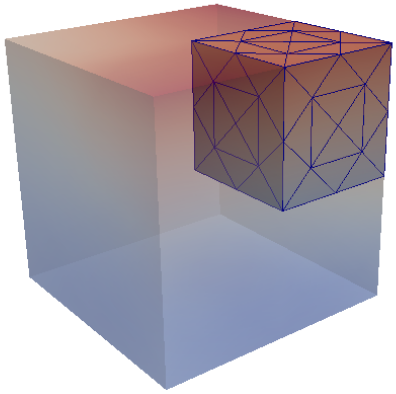
Remarque 26 (Solutions particulières structurées) *Les estimateurs $\mathscr{E}$ et $\mathscr{E}_{mM}$ sont intégrés sur l'espace paramétrique, ce sont des estimateurs en moyenne qui ne donnent pas d'indication sur la qualité d'une solution particulière. Par ailleurs, les solutions utilisées dans $\mathscr{E}_{loc}$ sont choisies aléatoirement et la probabilité de tirer un cas spécifique est quasi-nulle pour les grands nombres de paramètres. Or, on peut être amené à faire appel aux modèles réduits pour résoudre des problèmes « structurés », c'est à dire pour lesquels les paramètres forment des motifs réguliers dans l'espace (par exemple si tous les paramètres sont rigides, souples, si on a présence seulement d'une inclusion ou d'un trou ...). Aucun de nos estimateurs ne nous donne de réelle indication pour connaitre la solution des ces problèmes particuliers qui peuvent être très mal approchés par nos modèles réduits.*

2.3.2 Résolution exacte

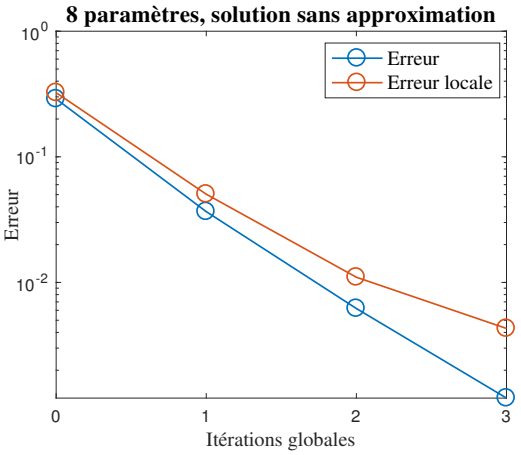
Le premier exemple présenté permet de tester la convergence de la PM-PGD introduite ALG.5. On a donc cherché à éliminer toutes les approximations réalisées.

En ne choisissant que des fonctions micro (donc $\Delta\mathbf{R} = 0$), la solution obtenue doit converger vers la solution EF exacte car chaque erreur $\mathbf{R}_E$ est compensée parfaitement sur Ω_E à chaque itération. Par ailleurs, on n'a pas utilisé d'approximation PGD pour résoudre le problème bidimensionnel (3.19) mais sa solution exacte. La seule source d'erreur qui n'est pas complètement annulée est l'erreur d'interpolation dans l'espace paramétrique. On a cherché à la minimiser en discrétisant les fonctions micro sur 100 points.

Le problème retenu est un problème à sous-domaines cubiques à 8 paramètres (voir FIG.3.3), les variations des modules d'Young de chaque sous-domaine cubique



(a) Solution particulière et mise en valeur d'un sous-domaine



(b) Erreur $\mathscr{E} = \mathscr{E}_{mM}$ et Erreur locale $\mathscr{E}_{loc}$

FIGURE 3.3 : Cas-test à 8 paramètres sans approximation

sont de 50% autour de leur valeur moyenne. L'évolution des différents estimateurs d'erreur est visible FIG.3.3. On observe une décroissance régulière de l'erreur $\mathscr{E}$. L'erreur locale à un comportement proche et le léger ralentissement après les deux premières itérations est lié à l'erreur d'interpolation dans l'espace paramétrique.

On est contraint dans ce contexte de s'arrêter après trois itérations car le nombre de modes à calculer dépasserait largement la mémoire disponible sur un ordinateur de bureau (voir REM.22). On observe le même comportement sur un problème à seulement 3 paramètres jusqu'à la quatrième itération.

Cet exemple permet de vérifier que la procédure proposée converge effectivement vers la solution multiparamétrique exacte. Si les temps de calcul restent très raisonnables, la taille finale du modèle réduit est extrêmement élevée, sans commune mesure avec les quelques dizaines de modes qui permettent d'atteindre des erreurs de quelques pourcents via une PGD classique telle que celle présentée FIG.1.12. Une compression du résultat obtenu est indispensable et peut être réalisée grâce à une CP-ALS (voir I.2.2.3) par exemple.

2.3.3 Convergence et analyse du cas à 60 paramètres

Le problème modèle a été résolu pour un cas à 60 paramètres associé à des sous-domaines polyédriques et une variation du module d'Young de 50%. Les champs d'erreur $\mathbf{R}$ et de solution $\underline{\mathbf{X}}$ n'ont pas été compressés dans cet exemple.

Les sous-domaines sont visibles FIG.3.5. Le voisinage $\widehat{\mathbf{C}}_E$ associé à un sous-domaine particulier Ω_E est présenté dans deux cas différents, si une ou deux « couches » de voisins sont retenues. On a choisi d'inclure dans le voisinage tous les sous-domaines présents dans un rayon r donné autour de Ω_E. En effet, choisir tous les

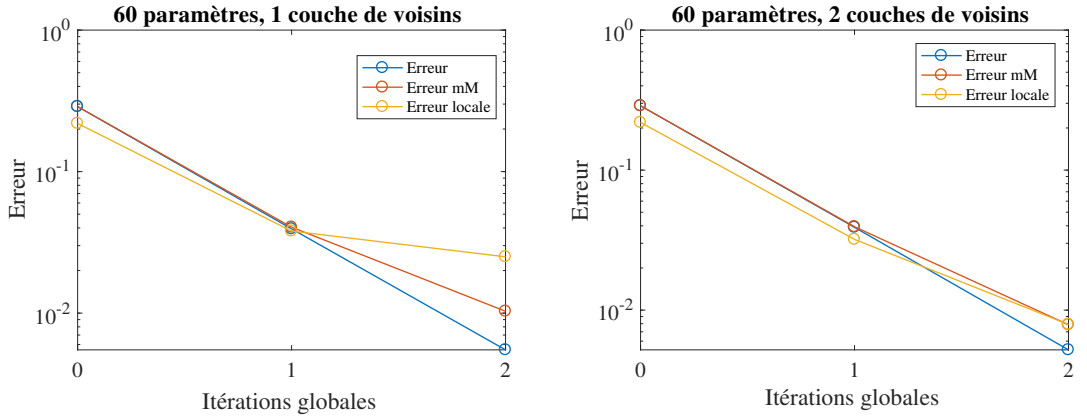


FIGURE 3.4 : Erreurs pour un problème à 60 paramètres

sous-domaines partageant une surface avec Ω_E comme on l'a fait pour des décompositions régulières (voir FIG.1.10) produit des voisinages extrêmement déséquilibrés. En notant l_E une longueur caractéristique de Ω_E, on choisit la première couche de voisins en posant $r = 2l_E$ et la deuxième avec $r = 3l_E$.

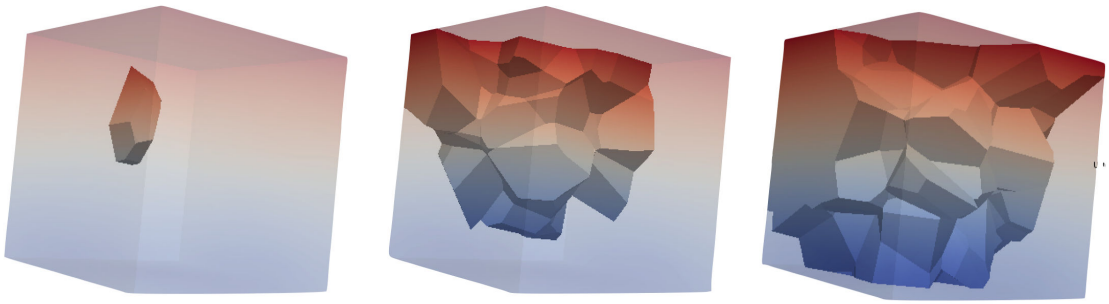


FIGURE 3.5 : Problème à 60 paramètres : 1 sous-domaine ; première couche de voisins ; deuxième couche de voisins

L'Erreur $\mathscr{E}$ visible FIG.3.4 décroit régulièrement quel que soit le voisinage retenu. Elle peut être interprétée comme une erreur au sens de la discrétisation micro-macro et décrit la qualité de la solution par rapport au format de donnée multiéchelle retenu. Elle ne prend pas en compte les erreurs liées à ce format, qui ne peuvent pas être négligées à partir de la deuxième itération. On le constate par exemple en observant que l'erreur locale $\mathscr{E}_{loc}$ ne décroit presque plus lorsque peu de fonctions micro sont utilisées. Comme on l'a noté REM.17, $\mathscr{E}$ n'est pas un estimateur fiable dans ce contexte. Il continue à décroitre alors que la qualité du modèle réduit vérifiée en certains points par l'erreur $\mathscr{E}_{loc}$ n'est pas améliorée.

Au contraire, pour une représentation multiéchelle suffisamment riche, on constate

que $\mathscr{E}$ et $\mathscr{E}_{mM}$ sont très proches. Dans ce cas, l'erreur locale décroit avec ces deux estimateurs. L'erreur $\mathscr{E}_{mM}$ étant extrêmement coûteuse, on se contente d'utiliser l'estimateur local pour vérifier la convergence de notre algorithme pour les grands nombres de paramètres.

2.4 Paramètres de l'algorithme

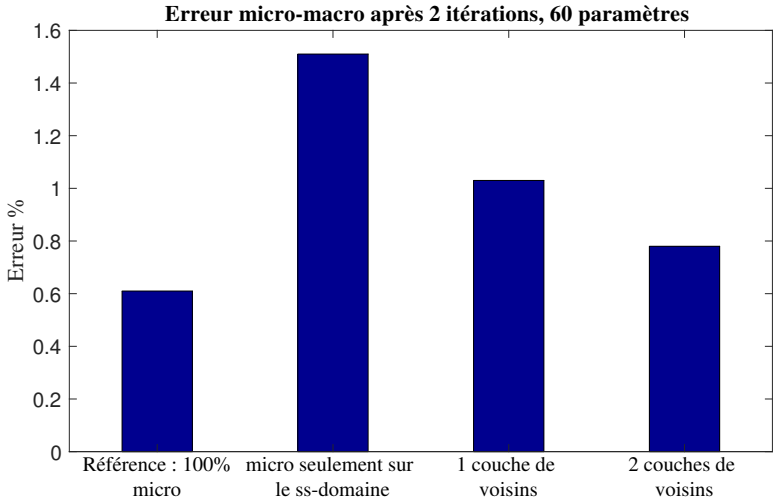


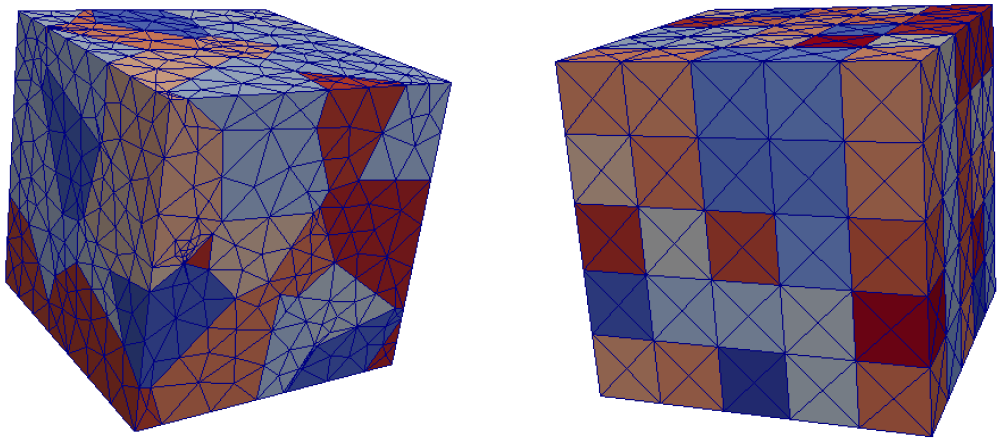
FIGURE 3.6 : Influence du voisinage (problème à 60 paramètres, variation paramétrique de 50%)

2.4.1 Choix du voisinage

Différents voisinages possibles pour le problème à 60 paramètres sont représentés FIG.3.5. Quelle que soit la solution retenue, le temps de calcul du modèle réduit (calcul offline) reste du même ordre de grandeur. Cependant, les solutions sont bien plus légères et rapides à utiliser si peu de fonctions micro sont conservées, au prix d'une certaine perte de précision. On a tracé FIG.3.6 différentes erreurs $\mathscr{E}_{mM}$ après deux itérations pour trois différentes tailles de voisinage que l'on compare avec une solution de référence exclusivement composée de fonctions micro.

On constate que pour chaque représentation, l'erreur reste du même ordre de grandeur. Pour réaliser un compromis entre la taille de la solution et la qualité du modèle réduit calculé, ce sont les représentations associées à une ou deux couches de voisins qui ont été retenues dans les exemples présentés dans la suite de cette thèse, en particulier pour les grands nombres de paramètres en IV.1.1.

Par ailleurs, on remarque aussi que la taille des sous-domaines doit être prise en compte. Certes, des sous-domaines de grandes tailles auront une influence spatiale plus importante, mais comme leurs voisinages sont aussi plus étendus, ils concentrent



(a) Sous-domaines polyédriques, 60 paramètres

(b) Sous-domaines cubiques, 125 paramètres

FIGURE 3.7 : Maillage et décomposition en sous-domaines de deux cubes

quand même la majorité des variations paramétriques. Au contraire, pour des sous-domaines de petite taille, l'influence paramétrique s'étendra facilement au delà de la première couche plus fine de voisins. C'est pour cette raison que deux couches de voisins ont systématiquement été retenues pour les cas décomposés régulièrement car leurs sous-domaines sont petits (25 éléments chacun, voir FIG.3.7b par exemple).

2.4.2 Influence de la géométrie

Pour un problème donné, le cube en traction présenté FIG.1.1 par exemple, on peut modifier complètement la géométrie du problème traité en choisissant des sous-domaines de formes très différentes, tels que ceux présentés FIG.3.7. Un second problème « en L » a aussi été traité pour des sous-domaines réguliers (FIG.3.2) ou polyédriques (FIG.3.8).

Les résultats pour les deux premières itérations de l'algorithme sont présentés TAB.3.2. On constate qu'ils sont du même ordre de grandeur d'un problème à l'autre. Cette observation est à rapprocher des résultats discontinus TAB.2.1 : la géométrie du problème et son nombre de paramètres ont très peu d'influence sur la convergence de l'algorithme.

Le problème « en L » qui présente une concentration de contraintes (voir FIG.3.2) pourrait être plus difficile du fait de la présence de sous-domaines particulièrement sollicités. En effet, l'erreur pour ces cas-tests est légèrement plus élevée que pour un cube en traction, mais le principe de Saint-Venant y est parfaitement respecté et la convergence de l'algorithme est aussi rapide.

	Erreur locale, solution moyenne	Erreur locale, itération 1	Erreur locale, itération 2
Cube, sous-domaines réguliers, 125 paramètres	21,2%	3,3%	0,98%
Cube, sous-domaines polyédriques, 60 paramètres	21,9%	3,2%	0,79%
L-shape, sous-domaines réguliers, 192 paramètres	25,7%	4,3%	1,1%
L-shape, sous-domaines polyédriques, 120 paramètres	25,9%	4,8%	1,2%

TABLEAU 3.2 : Erreurs locales pour des problèmes de géométries différentes (2 couches de voisins, variation paramétrique de 50%)

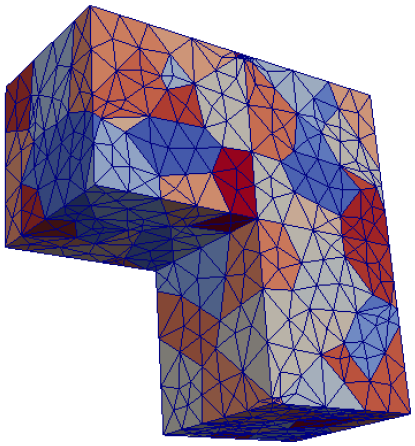


FIGURE 3.8 : Maillage et décomposition en sous-domaines du problème en « L » pour 120 paramètres

2.4.3 Influence des variations paramétriques

Tous les exemples présentés jusqu'à présent dans ce chapitre étudient des variations paramétriques de 50% autour de la moyenne. Notre paramètre étant le module d'Young, le problème devient bien plus difficile lorsque les variations augmentent, comme on a pu le constater avec l'algorithme précédent en II.2.4.1.

Les résultats présentés FIG.3.9 montrent l'influence de cette variation paramétrique dans le cas du problème à 120 paramètres illustré FIG.3.8. La solution moyenne est naturellement une meilleure approximation pour les problèmes à faibles variations

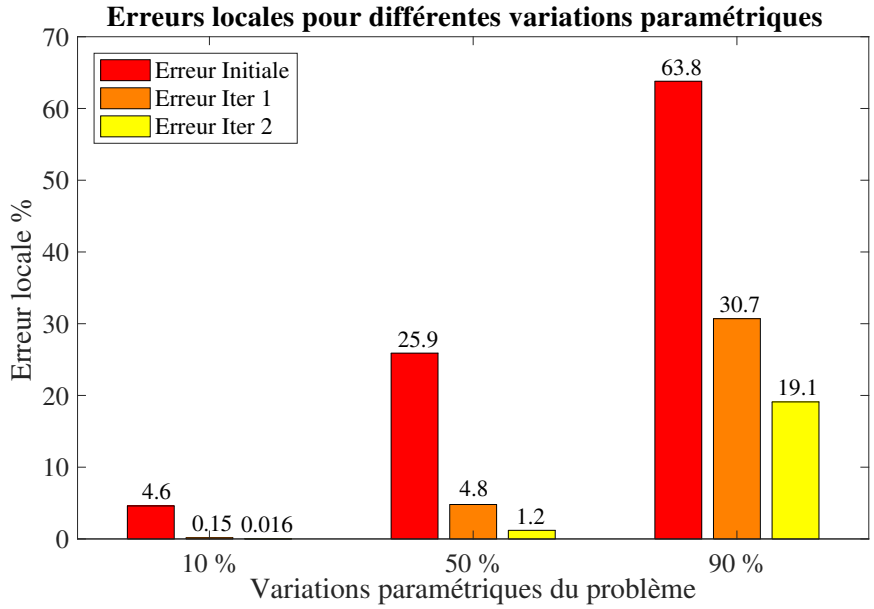


FIGURE 3.9 : Influence des variations paramétriques, problème à 120 paramètres

paramétriques, mais on constate également que le modèle réduit converge vers la solution exacte bien plus rapidement pour les problèmes les plus simples.

Contrairement à l'algorithme discontinu, une variation de 100% qui conduit à un problème à module d'Young nul sur un sous-domaine n'est pas supportée par le solveur EF. Par contre, il possible de résoudre le problème (3.19) dans le cas particulier où le sous-domaine Ω_E de module d'Young nul est vide en assemblant un matrice de rigidité spécifique pour la valeur $\mu_E = -1/2$ associée. On prend ensuite en compte cette solution particulière complétée par un champ de déplacement nul sur Ω_E au même titre que les solutions spatiales associées à toutes les autres valeurs de μ_E. Une POD permet de donner la solution de (3.19) sur quelques modes (voir REM.21). C'est cependant une procédure lourde à mettre en place et qui n'a pas été codée.

2.4.4 Compression de la solution

On peut à plusieurs moments de l'algorithme compresser les données générées, c'est même indispensable dès qu'il y a un trop grand nombre de paramètres comme on le constate REM.22.

La première compression a lieu lors de la résolution du problème (3.19) pour obtenir une solution espace-paramètres sous la forme d'une somme de produits. Cette compression consiste généralement à interrompre la PGD après la génération des 2 ou 3 modes qui permettent de décrire précisément la solution.

On a déjà discuté des troncatures possibles de la solution en II.2.3.1. L'influence d'une correction locale $\Delta\underline{\mathbf{X}}_{(E)}$ est négligée pour les éléments trop éloignés de Ω_E. Cette

mise à jour partielle peut être interprétée comme un espace d'approximation légèrement différent. Ces troncatures ont exactement les mêmes conséquences que la perte de continuité d'un sous-domaine à l'autre : l'erreur **R** est aussi approximée et ne va pas décrire toutes les erreurs qui apparaissent durant le processus.

Deux autres compressions sont possibles à deux moments de l'algorithme, précisés par les symboles $\circledast$ dans ALG.6. Celles-ci permettent de réduire le nombre de modes de champs déjà stockés sous une forme à variables séparées. L'algorithme utilisé, la CP-ALS (voir I.2.2.3), est fourni par une fonction de la Tensor Toolbox [Bader *et al.*, 2015].

Compression directe de la solution

On peut compresser directement les solutions. Ces étapes, notées $\circledast^2$ dans ALG.6 sont réalisées après la sommation des nouvelles contributions et ont plusieurs caractéristiques :

— Elles sont particulièrement efficaces : en effet, comme elles arrivent après la sommation des contributions provenant de nombreuses solutions globales à des problèmes différents, elles compressent des données particulièrement redondantes, notamment pour les itérations supérieures à 1 (voir TAB.3.3).

— Mais elles touchent directement la solution. Ainsi, l'erreur de compression, aussi minime soit-elle, ne se retrouvera pas dans l'erreur **R**. Il s'agit d'une approximation qu'on ne pourra plus corriger par la suite. Si cette compression est réalisée dès le début de l'algorithme, il faut être conscient que l'erreur finale ne pourra plus être inférieure à cette erreur de compression.

— On privilégiera donc ce type de compression à l'issue de la dernière itération, elle sert principalement à rendre le modèle réduit plus léger et efficace à utiliser.

	Compression globale de la solution, itération 1	Compression locale de la solution, itération 2		
Nombre de modes avant compression	122	~2000	~2000	~2000
Nombre de modes après compression	61	100 (par sous-domaine)	20 (par sous-domaine)	10 (par sous-domaine)
Erreur de compression	2.5%	~1%	~2%	~3%

TABLEAU 3.3 : Compression de la solution ($\circledast^2$ dans ALG.6), problème à 60 paramètres, $d \sim 40.10^3$

Remarque 27 (Compression locale) *Si la solution est stockée localement sur chaque sous-domaine, avec une description multiéchelle en paramètres, il faut réaliser une compression par sous-domaine. Celles-ci sont extrêmement efficaces (voir* TAB.*3.3). On comprend pourquoi grâce au principe de St-Venant : les contributions des voisins lointains*

sont en effet localement négligeables, et la compression ne corrige pas vraiment une redondance d'information mais plutôt le fait que de nombreuses informations sont en réalité minimes.

Compression de l'erreur

Les compressions réalisées sur les erreurs **R** avant des les compenser (notées $\circledast^1$ dans ALG.6) sont assez différentes (voir TAB.3.4) :

— Elles se font sur des erreurs locales, les compressions sont donc très efficaces (voir REM.27).
— On modifie le nombre de modes de l'erreur, donc directement le nombre de problèmes PGD bidimensionnels à résoudre. Cette étape est essentielle pour que l'algorithme tourne en un temps raisonnable.
— En imposant le nombre de modes à compenser sur chaque résidu, on impose indirectement le nombre de modes de la solution finale (avant compression éventuelle).
— Comme précédemment, les itérations suivantes de l'algorithme ne peuvent plus corriger les erreurs ainsi créées qui ne se retrouvent pas dans la nouvelle valeur de **R**.

	Sans compression	5 modes conservés	10 modes conservés	15 modes conservés
Erreur locale finale	0.6% (extrapolé d'un cas plus léger)	2.1%	1.2%	0.8%
Nombre de modes finaux avant compression de $\underline{\mathbf{X}}$	$\sim 14.10^3$ (extrapolé d'un cas plus léger)	~700	~1300	~1900
Poids de la solution	~10 Go (extrapolé d'un cas plus léger)	420 Mo	840 Mo	1.3 Go

TABLEAU 3.4 : Compression de **R** à l'itération 2, problème à 60 paramètres, $d \sim 40.10^3$

TAB.3.3 et TAB.3.4 illustrent les erreurs effectuées lors des étapes de compression : elles sont facilement de l'ordre de quelques pourcents. Les gains en terme de nombre de modes et donc de poids de la solution sont conséquents et ces étapes sont indispensables pour l'utilisation pratique des modèles réduits.

Les paramètres de compaction (Quand compresser? Combien de modes conserver?) sont les paramètres les plus délicats des procédures PM-PGD et tous les choix effectués dans cette thèse sont empiriques, issus de tests tels que ceux présentés dans cette partie. Des études rigoureuses seront à mener dans le futur.

Remarque 28 (Erreur de compression) *On constate* TAB.*3.2 que pour différentes géométries, les erreurs à l'issue de la première itération sont de l'ordre de 3 à 4 pourcents. Si*

l'ensemble des compressions nécessaires pour obtenir une solution utilisable génère des erreurs d'approximation qui sont elles aussi de quelques pourcents, la deuxième itération de l'algorithme n'améliore quasiment plus la solution finale.

3 Comparaison avec la méthodologie discontinue

La limite majeure de la méthodologie discontinue présentée en II.2.4 est surmontée par ce nouvel algorithme : des géométries variées et complexes sont accessibles pour des niveaux de performance similaires.

Au delà du format multiéchelle utilisé dans les deux versions de la PM-PGD, l'esprit de ces deux procédures est très proche, d'où le choix de conserver le même nom. En effet, elles sont basées sur une même idée : corriger itérativement les erreurs locales $\mathbf{R}_E$ via des problèmes simplifiés. La simplification retenue (utilisation de la loi de comportement moyenne hors du sous-domaine Ω_E associé à $\mathbf{R}_E$) est la même. La principale différence provient de la méthode de résolution des problèmes de correction :

— La procédure discontinue consiste à compenser cette erreur en imposant fortement un champ correctif connu sous sa forme analytique (problème (2.27)).
— Au contraire, la méthode présentée dans ce chapitre cherche à annuler cette erreur locale sous forme faible (problème (3.19)) pour ne pas perdre les propriétés de continuité de la solution aux interfaces entre les sous-domaines.

L'estimateur $\mathscr{E}_{mM}$ étant trop coûteux et l'erreur $\mathscr{E}$ masquant les approximations liées au format retenu, on se contentera pour les cas à très grands nombres de paramètres de l'estimateur le plus simple, $\mathscr{E}_{loc}$. On peut vérifier FIG.3.4 que l'évolution de cet estimateur est bien corrélée à celle de $\mathscr{E}_{mM}$, cette observation a été confirmée sur de nombreux cas-tests durant cette thèse.

Un des objectifs premiers lors de la conception de cet algorithme était de le rendre non-intrusif. On aurait voulu pouvoir faire appel à un solveur EF considéré comme une « boîte noire » capable de résoudre un problème mécanique donné. Ce n'est pas le cas en pratique, car la résolution via une PGD du problème à deux dimensions (3.19) nécessite la construction d'opérateurs spécifiques (voir partie I.4) qui est une opération très intrusive.

Cependant, on pourrait simplement se contenter de résoudre ce problème spatialement pour chaque valeur $\underline{\mu}_E(i)$ de la discrétisation paramétrique puis effectuer une POD comme précisé REM.21. Dans ce cas, chaque résolution peut théoriquement se faire en appelant un solveur EF extérieur. Encore faut-il pouvoir exprimer la « contrainte imposée » $\mathbf{R}_E\left(\underline{\mu}_E(i)\right)$ comme une CL en entrée de ce solveur. Cette condition n'est pas classique et a par exemple due être codée dans le solveur *Romlab*. En supposant que de telles CL puissent être prises en compte, cette méthode peut être implémentée en appelant de manière non-intrusive un solveur industriel.

Il faudrait donc assembler et résoudre autant de problèmes que de points de discrétisation de $\underline{\mu}_E$. Or tous ces problèmes sont très proches et ne diffèrent que par un facteur dans la matrice de rigidité. Cette propriété permet de gagner beaucoup de temps

lors des assemblages lorsqu'on a accès à ces fonctions dans le code EF.

Chapitre 4

Résultats et applications

La PM-PGD est appliquée à la construction de modèles réduits de très grandes dimensions et ses performances sont étudiées. On utilise ensuite ces abaques virtuels pour résoudre efficacement des problèmes inverses à grand nombre de paramètres.

La Parameter-Multiscale PGD, dans sa version continue est un algorithme robuste qui permet de traiter des problèmes à grand nombre de paramètres. Nous avons au chapitre précédent présenté des exemples en prenant en compte jusqu'à environ 200 (voir FIG.3.2 par exemple). Ces résultats sont déjà inaccessibles aux méthodes de réduction de modèles classiques. On souhaite augmenter encore le nombre de paramètres, ce qui est théoriquement possible puisque la convergence de l'algorithme ne dépend quasiment pas de celui-ci (voir III.2.4.2).

Cependant, certaines difficultés techniques apparaissent quand il s'agit de manipuler ou stocker des champs de très grandes dimensions et certains éléments de l'algorithme doivent être adaptés. Ces problématiques sont discutées en détails en partie 1.1.1.

Bien qu'il soit possible de générer des modèles réduits avec des précisions de l'ordre du pourcent, on a pu constater que de tels niveaux d'erreur imposent de garder un grand nombre de modes, les solutions associées sont donc très lourdes (voir TAB.3.3). Le temps nécessaire pour extraire une solution n'est pas négligeable dans ce cas et on étudie en partie 1.3 les gains réels en temps de calcul online liés à l'utilisation de la PM-PGD.

Par ailleurs, selon les applications objectifs de notre modèle réduit, les erreurs effectuées ne sont pas forcément négligeables (voir REM.26). Pour tester nos algorithmes dans des cas pratiques d'utilisation, on a retenu un problème particulier qui satisfait les critères suivants :

— il a un grand nombre de paramètres matériaux
— il fait appel un grand nombre de fois à des calculs similaires, mais associés à des paramètres matériaux différents

Des problèmes inverses tels que l'identification d'un champ de coefficients matériaux étudiée en partie 2 satisfont parfaitement ces critères. On se place donc dans un cas d'application classique de la réduction de modèle : après un calcul offline, on espère gagner en temps de calcul online en faisant appel de nombreuses fois au modèle réduit lors de procédures itératives.

Un autre avantage majeur du modèle réduit est sa forme : la solution et ses dépendances paramétriques sont connues explicitement. Ainsi, on peut analytiquement manipuler le modèle réduit et par exemple le dériver par rapport aux paramètres. Les applications présentées permettent de profiter de cette possibilité.

1 Grands nombres de degrés de libertés, grands nombres de paramètres

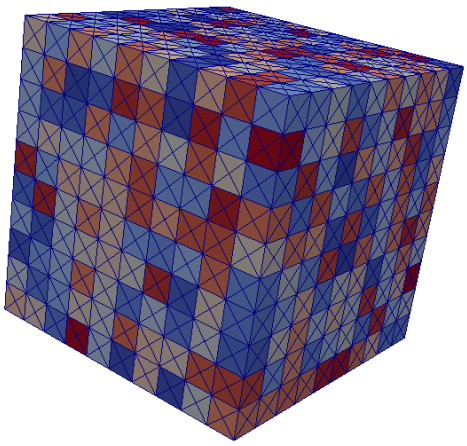
1.1 Cas-tests de grandes dimensions

On a discuté en III.2.4.4 de différentes stratégies de compression de la solution. Celles-ci sont en particulier basées sur l'utilisation de la CP-ALS qui permet d'approximer un champ déjà connu sous une forme à variables séparées par une approximation de même forme mais comprenant un nombre bien inférieur de modes.

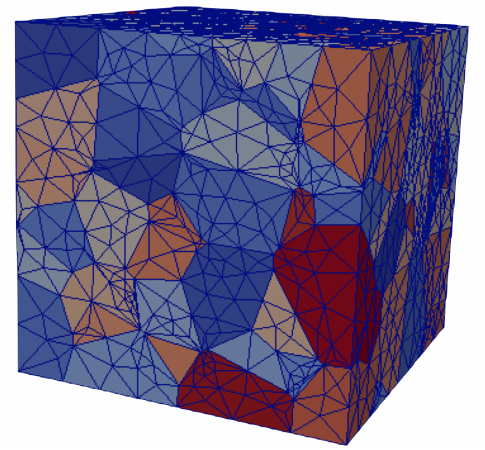
1.1.1 Contrôle de la CP-ALS

Dans notre contexte, un des principaux défauts de cet algorithme est son manque de souplesse. Le nombre de modes finaux doit être choisi en amont (voir I.2.2.3) et l'erreur finale ne peut pas être contrôlée comme lors de l'utilisation d'une procédure greedy telle que la PGD (voir I.2.2.2). C'est un problème majeur dans notre cas, car on ne veut pas que l'erreur de compression devienne du même ordre de grandeur que les corrections apportées par la procédure PM-PGD, ce qui rendrait l'ajout de nouveaux modes inutile (voir REM.28).

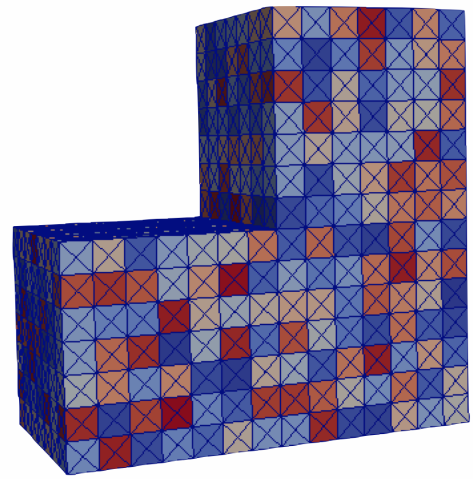
En pratique on s'appuie sur des résultats empiriques tels que ceux présentés TAB.3.3 pour choisir un taux de compression raisonnable, c'est à dire un rapport entre le nombre de modes avant compression et le nombre de termes de la solution compressée. La qualité de chaque CP-ALS est vérifiée *a posteriori* et le taux de compression est réduit si l'erreur d'approximation est trop importante. Dans tous les cas présentés ici, on a toléré une erreur inférieure à 2% sur la solution $\underline{\mathbf{X}}$ et à 3% sur les erreurs $\mathbf{R}$.



(a) Cube, 1000 paramètres



(b) Cube, 300 paramètres



(c) L-shape, 1029 paramètres

FIGURE 4.1 : Maillages et répartitions paramétriques de cas-tests de grandes dimensions

1.1.2 Compression par groupes de modes

La CP-ALS est donnée ALG.2. On rappelle qu'on doit en particulier construire à chaque itération un système matriciel $\underline{\underline{\boldsymbol{A}}}\,\underline{\Gamma} = \underline{\underline{\boldsymbol{B}}}$ dont les différents termes sont issus d'intégrations. Par exemple, pour construire la matrice $\underline{\underline{\boldsymbol{A}}}$, on doit réaliser des intégrations de la forme : $\int_{\overline{\boldsymbol{\Sigma}_{\mu_i}}} \prod_{j\neq i} \gamma_j^{(r_1)}\gamma_j^{(r_2)} d\overline{\boldsymbol{\mu}}_i = \prod_{j\neq i}\left(\int_{\mathbf{I}_j} \gamma_j^{(r_1)}\gamma_j^{(r_2)} d\mu_i\right) = \prod_{j\neq i} I_j$. Les différentes intégrales I_j ont la même forme et sont généralement du même ordre de grandeur. Pour les grands nombres de paramètres, ce produit de $(N_p - 1)$ termes peut donc prendre des valeurs extrêmes, très élevées en valeur absolue si les $|I_j|$ sont en moyenne

supérieurs à 1 ou atteignant le zéro numérique si ceux-ci sont inférieurs à 1. La matrice $\underline{\underline{\boldsymbol{A}}}$ est alors trop mal conditionnée pour permettre une résolution numérique.

On a pourtant pris soin de choisir un ensemble de définition tel que chaque fonction constante égale à 1 (c'est à dire un paramètre sans influence) ait une intégrale qui vaille 1 (voir REM.3). Cette précaution permet de limiter l'augmentation exponentielle des termes de $\underline{\underline{\boldsymbol{A}}}$ et $\underline{\underline{\boldsymbol{B}}}$, notamment à la première itération lorsque de nombreuses fonctions paramétriques sont constantes et la CP-ALS classique est dans ce cas utilisable jusqu'à une soixantaine de paramètres.

Pour surmonter cette difficulté et compacter des champs de très grandes dimensions on effectue une CP-ALS par groupes de modes. On sélectionne des modes qui ne dépendent que d'un petit nombre de paramètres, c'est à dire composés de fonctions $\gamma_E \neq 1$ seulement pour un nombre limités de μ_E. On extrait ensuite un « sous-tenseur » à partir de ces modes en excluant tous les paramètres non variables $\mu_{E'}$ associés à des fonctions $\gamma_{E'}$ constantes. Ce sous-tenseur peut être compacté via une CP-ALS avant d'effectuer une nouvelle sélection de modes dépendants de d'autres paramètres à compresser.

Cette technique est bien moins efficace qu'une compression de tous les modes d'un coup. Elle n'est bien sûr possible que si un grand nombre de fonctions sont constantes, ce qui permet d'extraire ces groupes de modes sous la forme de sous-tenseurs de faibles dimensions sans perdre d'information. Par construction, les champs issus de ALG.6 ont bien cette propriété. Il faut alors rajouter un paramètre à l'algorithme : le nombre de dimensions maximales des sous-tenseurs qu'on choisit.

1.2 Résultats à très grand nombre de paramètres

1.2.1 Options de la méthode et résultats

Des cas à très grand nombre de paramètres ont été implémentés. Deux dépassent 1000 paramètres, ce sont des cas à sous-domaines cubiques simples et réguliers. Leurs maillages sont visibles FIG.4.1a et FIG.4.1c. Des problèmes à sous-domaines polyédriques nécessitant des maillages plus raffinés ont été testés jusqu'à 300 paramètres (voir FIG.4.1b).

L'ensemble des paramètres de l'algorithme PM-PGD sont récapitulés TAB.4.1. Pour illustrer les choix effectués pendant cette thèse, on donne TAB.4.2 l'exemple de deux ensembles d'options retenues pour deux cas-tests différents. Le premier, présenté TAB.4.2a, est un cas composé d'un nombre raisonnable de paramètres (60) mais à grand nombre de degrés de liberté. Cet exemple est discuté en détail en partie 1.3. L'autre, TAB.4.2b, concerne un des cas-tests retenus dans cette partie et illustre les choix adaptés à la prise en compte d'un très grand nombre de paramètres. On y observe en particulier les différentes options de compaction discutées ci-dessus.

Les résultats sont présentés TAB.4.3. Nous n'avons calculé que la première itération des algorithmes. En effet, pour les plus grands nombres de paramètres il n'est pas possible de trouver de compromis satisfaisant entre la qualité des compressions et le

<table>
<tr><th colspan="2">Choix du problème</th></tr>
<tr><td colspan="2">• Ω, $\underline{f}_d$, $\underline{u}_d$, $\underline{F}_d$: Géométrie du problème et conditions limites
• $\{\Omega_E\}_{E\in\mathbf{E}}$: Ensemble des sous-domaines
$\Rightarrow N_p$: Nombre de paramètres
• ϵ : Variation paramétrique</td></tr>
<tr><th colspan="2">Options de la PM-PGD</th></tr>
<tr><td colspan="2">• I_G : Nombres d'itérations globales
• $\widehat{\mathbf{C}}_E$: Taille des voisinages
• n_E : Nombre de points de discrétisation des $\mathbf{I}_E$
• I_{part} : Nombre de solutions particulières pour $\mathscr{E}_{loc}$
• $\circledast^1$ = Oui/Non : Compression des erreurs $\mathbf{R}_E$
• $\circledast^2$ = Oui/Non : Compression de la solution $\underline{\mathbf{X}}$
• Troncature : Oui/Non
• Stockage de $\underline{\mathbf{X}}$: local ou global</td></tr>
<tr><th colspan="2">Options de Compression</th></tr>
<tr><th>CP-ALS classique</th><th>CP-ALS par groupes</th></tr>
<tr><td>• Itérations concernées
• Nombre de modes conservés (= taux de compression)
• Erreur de compression max
$\Rightarrow$ pour $\mathbf{R}$ et $\underline{\mathbf{X}}$</td><td>• Itérations concernées
• Taille maximale d'un bloc
• Taux de compression de chaque bloc
• Erreur de compression max
$\Rightarrow$ pour $\mathbf{R}$ et $\underline{\mathbf{X}}$</td></tr>
</table>

TABLEAU 4.1 : Ensemble des paramètres à fixer pour construire un modèle réduit PM-PGD

temps de calcul. Le nombre de modes étant extrêmement élevé, il est nécessaire d'effectuer de très importantes compactions des erreurs $\mathbf{R}_E$ notamment. Cela nuit tellement à la qualité du résultat que l'influence de la deuxième itération devient minime, comme décrit REM.28.

Il est cependant possible d'aller jusqu'à la deuxième itération pour le problème à 120 paramètres et les résultats ont été présentés TAB.3.2. Le modèle réduit est alors très lourd et son utilisation donne des performances médiocres (voir TAB.4.7).

Les erreurs observées sont exactement celles attendues. Nous avons déjà constaté plusieurs fois que le nombre de paramètres n'était pas un élément déterminant pour la convergence de la PM-PGD. On obtient donc des erreurs du même ordre de grandeur que celles déjà présentées TAB.3.2 par exemple. On retiendra que l'algorithme est capable de donner en un temps raisonnable un modèle réduit avec une précision de l'ordre du pourcent pour les problèmes de moins de 200 paramètres et de l'ordre de 3 à 5 pourcents pour les problèmes de 300 à 1000 paramètres, comme résumé TAB.4.4.

Choix du problème
• Cube, problème présenté FIG.1.1 • 60 sous-domaines, maillage présenté FIG.4.2 $\Rightarrow N_p = 60$ • $\epsilon = 1$ (Variation paramétrique = 50%)
Options de la PM-PGD
• $I_G = 2$ • $\widehat{\mathbf{C}}_E = 2$ couches de voisins • $n_E = 25$ • $I_{part} = 50$ • $\circledast^2$ = Oui : Compression de la solution $\underline{\mathbf{X}}$ à l'issue de l'itération 2 • $\circledast^1$ = Oui : Compression des erreurs $\mathbf{R}_E$ à l'issue de l'itération 1 • Troncature : Non • Stockage de $\underline{\mathbf{X}}$: global
Options de Compression
CP-ALS classique :
• Compression de $\mathbf{R}^1$: Taux de compression = $\frac{1}{8}$ Erreur de compression max = 3% • Compression de $\underline{\mathbf{X}}^2$: Taux de compression = $\frac{1}{2}$ Erreur de compression max = 2%

(a) Problème à 60 paramètres

Choix du problème
• L-shape, problème présenté FIG.3.2 • 1029 sous-domaines, maillage présenté FIG.4.1c $\Rightarrow N_p = 1029$ • $\epsilon = 1$ (Variation paramétrique = 50%)
Options de la PM-PGD
• $I_G = 1$ • $\widehat{\mathbf{C}}_E = 2$ couches de voisins • $n_E = 25$ • $I_{part} = 50$ • $\circledast^2$ = Oui : Compression de la solution $\underline{\mathbf{X}}$ à l'issue de l'itération 2 • $\circledast^1$ = Non • Troncature : Oui, 3 couches • Stockage de $\underline{\mathbf{X}}$: global
Options de Compression
CP-ALS par groupe :
• Compression de $\underline{\mathbf{X}}^1$: Dimension max des groupes = 40 Taux de compression = $\frac{1}{2}$ Erreur de compression max = 2%

(b) Problème à 1029 paramètres

TABLEAU 4.2 : Options de la méthode pour deux cas-tests différents

Ces résultats sont en cours de publication dans [Paillet *et al.*, 2019].

De telles performances peuvent sembler bien moins bonnes que celles obtenues grâce à la méthode discontinue présentée TAB.2.1. Il s'agit d'un biais lié au choix de l'estimateur d'erreur : seul l'erreur $\mathscr{E}$ a été calculée au chapitre 2 tandis qu'on a choisi aux chapitres 3 et 4 l'estimateur $\mathscr{E}_{loc}$ qui nous semble plus pertinent. Les deux méthodes ont bien des précisions équivalentes. A titre d'exemple, pour le problème à 125 paramètres, on observait une erreur $\mathscr{E} = 0.49\%$ avec la méthode discrète après deux itérations. Ce même estimateur donne précisément $\mathscr{E} = 0.59\%$ dans le cas discontinu.

Il aurait certes été plus cohérent de présenter dans ce manuscrit des résultats équivalents en utilisant des estimateurs comparables. Cependant, on a choisi de conserver la présentation des résultats tels qu'ils sont publiés dans [Ladevèze *et al.*, 2018] et

	Solution moyenne, erreur locale	Une itération globale, erreur locale
Cube, sous-domaines réguliers, 1000 paramètres	27,8%	3,9%
Cube, sous-domaines polyédriques, 300 paramètres	19.5%	3.4%
L-shape, sous-domaines réguliers, 1029 paramètres	18,0%	4,5%
L-shape, sous-domaines polyédriques, 120 paramètres	25,9%	4.8%

TABLEAU 4.3 : Erreur locale après la première itération (2 couches de voisins, variation paramétrique = 50%, troncature pour les problèmes à 1000 paramètres et plus)

[Paillet *et al.*, 2019]. Ils sont commentés de manière bien plus détaillés dans cette thèse et la différence illustre l'évolution des travaux durant les 3 ans passés sur ce sujet. La nouvelle version de l'algorithme PM-PGD n'a pas été que l'occasion de dépasser les limites présentées en II.2.4 mais aussi de corriger les imprécisions méthodologiques des premières tentatives de construction de modèles réduits à très grand nombre de paramètres.

	Moins de 100 paramètres	100 à 200 paramètres	Plus de 200 paramètres
Erreur locale	$< 1\%$	$\sim 1\%$	3-5%

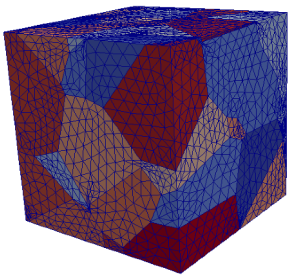
TABLEAU 4.4 : Erreurs locales accessibles en pratique en fonction du nombre de paramètres du problème pour une variation paramétrique de 50%

1.2.2 Manipulation des solutions

$\underline{\mathbf{X}}$ multiéchelle, avant compression	$\underline{\mathbf{X}}$ multiéchelle, après compression	$\underline{\mathbf{X}}$ monoéchelle, avant compression	$\underline{\mathbf{X}}$ monoéchelle, après compression
18 Go	1.6 Go	650 Mo	325 Mo

TABLEAU 4.5 : Coût de stockage des deux formats de solution, problème à 60 paramètres, $d \sim 40.10^3$, erreur locale $\mathscr{E}_{loc}$ finale inférieure à 1%

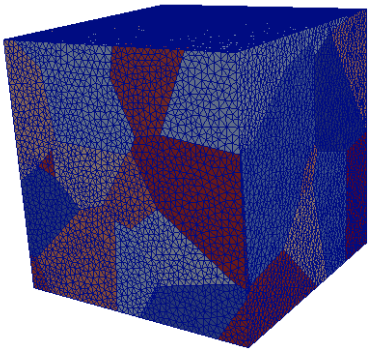
Une solution $\underline{\mathbf{X}}$ calculée grâce au format multiéchelle (1.38) est en pratique stockée comme un ensemble de champs locaux. Suivant l'utilisation escomptée du modèle réduit, ce format peut être très lourd à manipuler. En effet, pour construire un champ



Maillage et décomposition paramétrique

	Solution moyenne	Itération 1	Itération 2
Nombre de modes	1	122	1083
Erreur locale $\mathscr{E}_{loc}$	23%	3,3%	1,7%
Poids de la solution	300 ko	36 Mo	325 Mo

FIGURE 4.2 : Cube en traction, 60 paramètres, $d \sim 40.10^3$, variation paramétrique = 50%



Maillage et décomposition paramétrique

	Solution moyenne	Itération 1
Erreur locale $\mathscr{E}_{loc}$	23%	4,1%
Erreur $\mathscr{E}$	29%	4,3%

FIGURE 4.3 : Cube en traction, 60 paramètres, $d \sim 120.10^3$, variation paramétrique = 50%

solution complet il est nécessaire d'interpoler une solution paramétrique différente pour chaque sous-domaine, ce qui peut être coûteux. Or, on observe ALG.6 que la solution finale est composée de la somme de contributions globales $\Delta\underline{\mathbf{X}}_{(E)}$. On ne peut malheureusement pas les sommer directement car les formats des fonctions paramétriques sont les formats multiéchelles locaux associés aux sous-domaines Ω_E. La solution retenue (voir ALG.6 ligne 14) consiste à stoker séparément les contributions de $\Delta\underline{\mathbf{X}}_{(E)}$ restreintes à chaque sous-domaine $\Omega_{E'}$ en adaptant le format des fonctions paramétriques comme présenté en III.2.2.2.

Une autre solution consiste à sommer directement les différentes contributions globales $\Delta\underline{\mathbf{X}}_{(E)}$ en les modifiant. Pour qu'elles soient toutes compatibles, il faut choisir un format commun à l'ensemble des sous-domaines : les fonctions paramétriques sont donc toutes projetées sur la grille micro. Le format de stockage de la solution n'est

finalement plus multiéchelle, mais la démarche et les $\mathbf{R}_E$ par exemple restent associés au format (1.38).

TAB.4.5 illustre sur un cas particulier les gains en terme de coût de stockage que ce retour final au monoéchelle apporte. De plus, le temps nécessaire pour extraire une solution complète particulière est réduit de manière drastique. Par contre, pour des applications qui ne nécessitent qu'une connaissance des solutions localement en espace, l'utilisation de cette structure plus globale est bien moins efficace car l'unique champ global stocké a un nombre de modes bien plus important que le champ local d'intérêt sous la forme (1.38).

Enfin, l'étape de compaction de la solution est fortement modifiée. La solution n'est plus représentée que comme un unique tenseur dont une des dimensions (l'espace) est beaucoup plus grande que les autres. Les compressions via une CP-ALS sont bien moins efficaces car chaque mode issu d'un $\Delta\underline{\underline{\mathbf{X}}}_{(E)}$ particulier représente la correction d'un problème local indépendant et a une grande importance localement. On n'est plus dans le cas très favorable des champs locaux présenté REM.27.

1.3 Performances de la méthode

Pour étudier quantitativement les performances de la PM-PGD, on a choisi un cas-test à grand nombre de degrés de liberté (ddl). Le problème à 60 paramètres retenu est décomposé spatialement sur $d \simeq 40\,000$ ddl, ce qui correspond à une moyenne de l'ordre de 1800 éléments par sous-domaine (voir FIG.4.2). En raffinant plus finement encore le maillage utilisé, on a généré un modèle réduit associé à une décompostion spatiale sur $d \simeq 120\,000$ ddl. Les erreurs associées sont présentées FIG.4.3. Il faut plus de 6h pour terminer la première itération du modèle réduit, la deuxième est inaccessible dans ce cas pour des raisons de temps de calcul.

Remarque 29 (Calculs séquentiels) *La génération du modèle réduit est réalisée de manière purement séquentielle. Pour pouvoir les comparer, tous les calculs sont lancés sur un unique processeur du cluster Fusion, commun à l'école CentraleSupélec et à l'ENS Paris-Saclay.*

C'est pour les cas les plus « lourds » spatialement que notre modèle réduit offre les meilleurs gains de temps. Si la construction d'une matrice d'assemblage EF et la résolution du système associé est coûteuse, il est rentable de faire appel à un modèle réduit qui consiste simplement à interpoler une solution à variables séparées au point de l'espace paramétrique souhaité.

Les temps de calcul associés à la génération du modèle réduit et son exploitation sont présentés TAB.4.6.

Remarque 30 (Parallélisation) *A l'intérieur de chaque itération globale de la PM-PGD, les étapes de corrections locales sont toutes indépendantes. Elles peuvent donc potentiellement être parallélisées et engendrer des gains de temps de calcul offline considérables.*

	Temps de calcul
Extraction d'une solution de $\underline{\mathbf{X}}^1$ après 1 itération	0,3 s
Extraction d'une solution de $\underline{\mathbf{X}}^2$ après 2 itérations	0,77 s
Calcul complet d'une solution (assemblage de la matrice + inversion)	4,8 s
Calcul de $\underline{\mathbf{X}}^0$: initialisation + solution moyenne	15 s
Calcul de $\underline{\mathbf{X}}^1$: 1 itération globale	19 min
Calcul de $\underline{\mathbf{X}}^2$: 2 itérations globales	3 h

TABLEAU 4.6 : Temps de calcul associés à différentes opérations, problème à 60 paramètres, options de la PM-PGD choisies TAB.4.2a

L'utilisation du modèle réduit engendre un gain de temps conséquent. Chaque calcul individuel passe d'environ 5 secondes à 0.3 ou 0.8 seconde suivant la précision du modèle retenu. Un tel gain n'est pas négligeable si ces calculs particuliers doivent être lancés des milliers de fois comme pour les problèmes d'inversion étudiés en partie 2 par exemple. Par ailleurs, on constate le déséquilibre entre les temps de calcul des deux premières itérations. Le nombre élevé de modes à corriger et l'ensemble des opérations de compression à effectuer allongent énormément la durée de calcul de la seconde. On comprend que lorsque la première itération a déjà duré plusieurs heures comme pour le cas présenté FIG.4.3, la deuxième itération est inaccessible en un temps raisonnable.

Ce cas-test est compressé assez fortement comme précisé TAB.4.2a. Ce choix a été fait pour obtenir des solutions sur un nombre de modes limité, environ 1000, et donc un coût de stockage raisonnable (voir FIG.4.2). Les gains de temps auraient été moindres avec des solutions plus précises mais plus lourdes à manipuler. On a vu en II.2.4.2 qu'on pouvait aisément atteindre une erreur locale inférieure au pourcent pour ce cas-test, certes discrétisé spatialement plus grossièrement.

On ne revient pas en détail sur le poids des solutions que l'on manipule et qui peut atteindre le giga-octet comme on l'a constaté par exemple TAB.3.4 ou TAB.4.5. Pour des cas encore plus gros, le stockage même du modèle réduit pourrait devenir une limite.

On retient que la deuxième itération est très coûteuse comparativement à la première, pour une augmentation assez modeste de la qualité de la solution. Ainsi, on se contentera de modèles réduits issus d'une seule itération globale quand c'est possible pour les applications présentées ci-après.

2 Problèmes inverses

Cette partie propose des exemples de problèmes qui font appel un grand nombre de fois à des solutions EF de problèmes proches. Ils fournissent donc un cadre d'utilisation idéal pour l'application de nos modèles réduits. Les exemples présentés permettent de tester le comportement de la PM-PGD, ils n'ont pas fait en eux-mêmes l'objet d'études détaillées et le bibliographie présentée sera par conséquent limitée.

Les problèmes inverses forment un thème de recherche très riche et dynamique. On s'est contenté d'une méthode simple et classique sans prendre en compte les avancées les plus récentes de ce domaine. Ces problèmes ne sont que des prétextes à appliquer nos modèles réduits dans des cas originaux pour étudier leur pertinence.

Tous les exemples traités jusqu'à présent dans cette thèse sont des problèmes directs : on choisit l'ensemble des paramètres (géométrie, chargement, matériaux) puis on détermine la réponse de notre système, par exemple sous la forme d'un champ de déplacement ou de contrainte. L'originalité de nos travaux réside dans la nature de la réponse de très grande dimension qu'on cherche à calculer.

Les problèmes inverses au contraire permettent de déterminer certains paramètres d'un problème à partir de la connaissance partielle de ses données et de l'observation des résultats. Ils sont en général mal posés au sens de Hadamard [Hadamard, 1932] : l'existence et l'unicité de la solution ne sont pas garanties, et celle-ci ne dépend pas forcément continument des données d'entrée. C'est la nature des observations qui est à l'origine du caractère mal posé de ce type de problèmes. Elles sont généralement discrètes, potentiellement en nombre limité par rapport à la taille du problème et peuvent être bruitées.

Afin de profiter des modèles réduits construits grâce à la PM-PGD qui permettent de prendre en compte des paramètres distribués spatialement, des problèmes d'identification de champs ont été retenus. A partir de mesures (simulées dans notre cas), nous cherchons à retrouver la valeur particulière d'un paramètre donné (la rigidité par exemple) qui peut varier dans l'espace.

Plaçons-nous dans le cas particulier suivant : le problème direct est le problème continu purement spatial (2.1), et le problème inverse associé consiste à retrouver la valeur des paramètres $\boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu$ connaissant une solution particulière $\underline{\mathrm{U}}_{\mathscr{M}}$ de (2.1) seulement sur un ensemble $\mathscr{M}$ de points. On formule le problème d'inversion comme une minimisation :

$$\min_{\substack{\boldsymbol{\mu}\in\boldsymbol{\Sigma}_\mu \\ \underline{u}\text{ solution de (2.1)}}} \underbrace{\frac{1}{2}\left\|\underline{u}_{|\mathscr{M}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right\|^2_{\mathscr{N}}}_{\Phi(\boldsymbol{\mu})} \tag{4.1}$$

où $\underline{u}_{|\mathscr{M}}(\boldsymbol{\mu})$ est la solution du problème (2.1) restreinte aux points de mesure. Plusieurs choix de norme $\|\bullet\|_{\mathscr{N}}$ peuvent être effectués et seront discutés ci-après.

Après discrétisation EF de l'espace spatial, le problème devient :

$$\min_{\substack{\boldsymbol{\mu}\in\boldsymbol{\Sigma}_\mu \\ \underline{\underline{\mathrm{K}}}\,\underline{\mathrm{X}}=\underline{\mathrm{F}}}} \underbrace{\frac{1}{2}\left\|\underline{\underline{\Pi}}_{\mathscr{M}}\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right\|^2_{\mathscr{N}}}_{\Phi(\boldsymbol{\mu})} \tag{4.2}$$

où $\underline{\mathbf{X}}(\boldsymbol{\mu})$ est une solution particulière du problème discret (3.3) et $\underline{\underline{\Pi}}_{\mathscr{M}}$ une matrice de projection sur les points de mesure.

Les méthodes d'identification purement mathématiques vont se concentrer sur la minimisation (4.1) et sur les moyens d'enrichir le problème, grâce à la régularisation de Tikhonov utilisée par exemple en (4.6). Au contraire, une technique telle que la méthode de l'Erreur en Relation de Comportement (ERC) modifiée [Ladevèze *et al.*, 1994] permet une résolution du problème inverse basée sur des considérations mécaniques. Cette méthode fonctionne particulièrement bien pour résoudre les problèmes qualifiés de « reconstitution de champs de module » tels que ce que l'on étudie dans la partie suivante [Bonnet et Aquino, 2015, Banerjee *et al.*, 2013]. Il est très courant de faire appel aux statistiques [Kaipio et Somersalo, 2006] pour résoudre ce type de problèmes : les mesures étant connues avec une certaine incertitude, les paramètres sont identifiés sous la forme de densités de probabilité.

Expérimentalement, la corrélation d'image (2D) ou la tomographie (3D) permettent de mesurer des déplacements sur tout le domaine Ω. En variant les tests expérimentaux, on peut profiter de la richesse de ces mesures pour identifier des lois de comportement potentiellement complexes [Cooreman *et al.*, 2008]. Ces champs de mesures donnent la solution en chaque nœud du maillage EF ($\underline{\underline{\Pi}}_{\mathscr{M}} = \underline{\underline{\mathbf{1}}}$) et la fonctionnelle Φ de (4.2) est alors celle utilisée dans le contexte de la méthode FEMU (Finite Element Model Updating, [Pagnacco *et al.*, 2013]) en choisissant une norme L^2. Un autre choix possible est la norme énergétique :

$$\Phi(\boldsymbol{\mu}) = \frac{1}{2}\left\|\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right\|^2_{\underline{\underline{\mathbf{K}}}(\mu)} = \frac{1}{2}\left(\left[\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right]^T \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})\left[\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right]\right) \tag{4.3}$$

Contrairement aux définitions des semi-normes énergétiques données REM.9, on conserve bien la variation paramétrique de l'opérateur $\underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})$. D'autres méthodes ont été spécifiquement développées pour mettre à profit ces mesures de champs telles que la méthode de l'écart à l'équilibre [Claire *et al.*, 2004] ou la méthode des champs virtuels [Grédiac, 1989] (voir [Avril *et al.*, 2008] pour un aperçu des différentes méthodes). Toutes ces techniques peuvent être interprétées comme des minimisations de la fonctionnelle Φ introduite en (4.2) au sens d'une certaine norme $\mathscr{N}$. En particulier, lorsque les champs de mesures sont connus sur l'ensemble des nœuds du maillage, la méthode de l'ERC se simplifie et revient simplement à minimiser la norme énergétique (4.3) [Barbarella *et al.*, 2016]. Dans un contexte stochastique, on peut définir une métrique optimale qui permette d'identifier les paramètres $\boldsymbol{\mu}$ avec une incertitude minimale [Roux et Hild, 2019].

Des modèles réduits espace-paramètres ont été utilisés pour identifier des paramètres matériaux grâce à des méthodes de type POD (par exemple dans [Bocciarelli *et al.*, 2014]). Dans ce contexte cependant, la PGD qui donne directement une solution espace-paramètres sous une forme à variables séparées offre une grande simplicité d'utilisation. Par exemple, [Signorini *et al.*, 2017, Chamoin *et al.*, 2016] ont utilisé les avantages de la PGD dans un contexte déterministe tandis que [Rubio *et al.*, 2019, Marchand *et al.*, 2016] l'ont appliquée à des environnements sto-

chastiques.

Il y a de nombreux moyens de minimiser (4.2). La solution analytique n'est généralement pas accessible et des procédures itératives telles que l'algorithme de Newton-Raphson permettent de converger vers un minimum. C'est la méthode retenue pour les exemples présentés dans cette thèse, elle est classiquement utilisée dans la procédure FEMU par exemple. Cette technique impose de calculer à chaque itération la Hessienne de la fonctionelle Φ (voir (4.7)), ce qui est une opération coûteuse. On peut l'éviter grâce à des résolutions effectuées via un « problème adjoint », ce qui revient à imposer fortement la contrainte $\underline{\underline{\mathbf{K}}}\,\underline{\mathbf{X}} = \underline{\mathrm{F}}$ dans la formulation grâce à un multiplicateur de Lagrange. Cette nouvelle formulation est résolue via un point fixe entre l'espace et les paramètres. Il s'agit de la méthode privilégiée pour minimiser l'ERC modifiée.

2.1 Méthode retenue

2.1.1 Problème modèle

Comme dans tout ce manuscrit, on s'intéresse à un problème d'élasticité linéaire. Le domaine spatial Ω est composé d'un nombre N_p de sous-domaines Ω_E associés chacun à un paramètre proportionnel au module d'Young local du matériau sur Ω_E. Le champ de paramètres $\boldsymbol{\mu}$ que l'on cherche à déterminer s'interprète comme l'ensemble des rigidités des sous-domaines Ω_E.

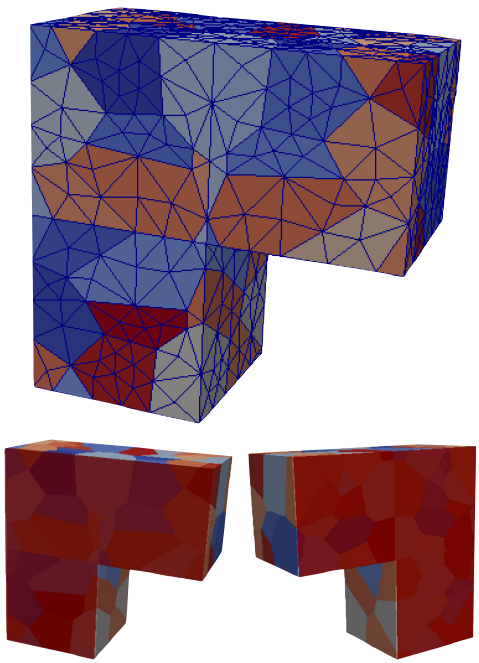
2 cas-tests sont présentés FIG.4.4. Le problème tridimensionnel à 120 paramètres « en L » est illustré FIG.3.2. Son modèle réduit présente une erreur locale $\mathscr{E}_{loc}$ de 4.8% après la première itération et 1.2% après la deuxième (voir TAB.3.2). Les modules d'Young de chacun des sous-domaines polyédriques sont générés aléatoirement et on cherche à les retrouver à partir de mesures sur 2 des faces latérales du cubes (en rouge FIG.4.4a).

Le second cas-test est un problème de traction bidimensionnelle à 1000 paramètres. Il a une rigidité moyenne sauf sur certains sous-domaines qui modélisent des inclusions souples ou rigides. On cherche à retrouver ces inclusions et leur rigidité à partir de mesures sur les bords non-encastrés. Dans ces deux cas-tests, les variations maximales du module d'Young autour de sa valeur moyenne sont de 50%

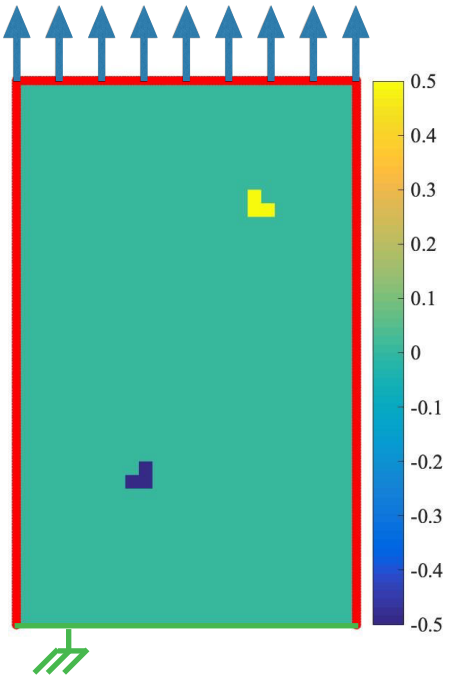
On choisit de réaliser des mesures uniquement en certains points du domaine. On définit en pratique $\underline{\underline{\Pi}}_{\mathscr{M}}$ l'opérateur permettant d'extraire d'un champ connu (ici simulé) un certain nombre de « points de mesure ». On simule par ailleurs un bruit en ajoutant un vecteur généré aléatoirement $\underline{\eta}$. On a donc construit nos « mesures » à partir d'une simulation donnée :

$$\underline{\mathrm{U}}_{\mathscr{M}} = \underline{\underline{\Pi}}_{\mathscr{M}} \underline{\mathbf{X}}(\boldsymbol{\mu}_{ex}) + \underline{\eta} \tag{4.4}$$

où $\boldsymbol{\mu}_{ex}$ est la valeur des paramètres qu'on cherchera ensuite à retrouver. En pratique, on quantifie le bruit par rapport à la moyenne du champ mesuré (en valeur absolue) $\langle |\underline{\mathbf{X}}| \rangle$. Par exemple, un bruit de 1% correspond à un vecteur généré aléatoirement suivant une loi uniforme à partir de valeurs comprises entre $-0.01 \langle |\underline{\mathbf{X}}| \rangle$ et $0.01 \langle |\underline{\mathbf{X}}| \rangle$.



(a) Problème à 120 paramètres, champ paramétrique aléatoire, les deux faces de mesures sont mises en évidence en rouge sur les figures du bas



(b) Problème à 1000 paramètres, inclusions de rigidités variables, mesures sur les bords non encastrés (en rouge).

FIGURE 4.4 : 2 problèmes inverses de grande dimension

A l'issue du problème d'identification, on a déterminé un jeu de paramètres $\boldsymbol{\mu}_{id}$ qu'on peut comparer au tirage exact $\boldsymbol{\mu}_{ex}$ grâce à l'estimateur :

$$\mathscr{E}_{\mu} = \frac{\|\boldsymbol{\mu}_{ex} - \boldsymbol{\mu}_{id}\|}{\|\boldsymbol{\mu}_{ex}\|} \tag{4.5}$$

La norme euclidienne est simplement utilisée ici.

Remarque 31 (Décroissance du résidu) *A partir de mesures réelles, on n'a pas accès à la valeur exacte $\boldsymbol{\mu}_{ex}$. L'estimateur utilisé est alors généralement le résidu minimisé $\left\| \underline{\underline{\Pi}}_{\mathscr{M}} \underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathbf{U}}_{\mathscr{M}} \right\|_{\mathscr{N}}$. Celui-ci converge toujours bien mieux que l'erreur $\mathscr{E}_{\mu}$ mais ne reproduit pas fidèlement la qualité de la solution obtenue.*

Si le problème est très mal posé ou sous-déterminé, il faut ajouter un terme à la fonctionnelle Φ pour améliorer le conditionnement. Dans notre cas, on régularise autour de $\boldsymbol{\mu}^0$ (valeur moyenne des paramètres sur leur domaine de définition) grâce à la méthode de Tikhonov :

$$\Phi(\boldsymbol{\mu}) = \frac{1}{2} \left\| \underline{\underline{\Pi}}_{\mathscr{M}} \underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathbf{U}}_{\mathscr{M}} \right\|_{\mathscr{N}}^2 + \frac{c}{2} \left\| \boldsymbol{\mu} - \boldsymbol{\mu}^0 \right\|^2 \tag{4.6}$$

Une norme euclidienne est simplement utilisée pour ce second terme. Le paramètre c doit aussi être déterminé, par exemple grâce à une $\mathscr{L}$-curve. Cette méthode suppose que la solution n'est pas trop éloignée de la solution moyenne $\underline{\mathrm{X}}^0$ utilisée pour initialiser le modèle réduit par exemple.

2.1.2 Résolution

On a retenu la méthode de Newton pour minimiser la fonction (4.6). On va donc déterminer le jeu de paramètres solution itérativement. Soit $\boldsymbol{\mu}^i$ le champ paramétrique identifié à l'itération i. On a :

$$\boldsymbol{\mu}^{i+1} = \boldsymbol{\mu}^i - [\nabla^2\Phi(\boldsymbol{\mu}^i)]^{-1}\nabla\Phi(\boldsymbol{\mu}^i) \tag{4.7}$$

Cette expression impose de calculer le Jacobien $\nabla\Phi$ et la Hessienne $\nabla^2\Phi$ de Φ. La norme $\|\bullet\|_{\mathscr{N}}$ retenue dans ces exemples est une norme énergétique moyenne liée à l'opérateur $\underline{\underline{\mathbf{K}}}^0$. Il s'agit d'un compromis entre le sens physique de la fonctionnelle qu'on minimise et la simplicité d'implémentation. En effet, la norme énergétique exacte présentée en (4.3) dépend de $\boldsymbol{\mu}$ ce qui accroit la complexité des différentiations de (4.7) et augmente de beaucoup le temps de calcul à chaque itération. Pour alléger les notations, on note $\underline{\underline{\Pi}} = \underline{\underline{\Pi}}_{\mathscr{M}}$. On pose $\underline{\underline{\mathbf{K}}}' = \underline{\underline{\Pi}}\,\underline{\underline{\mathbf{K}}}^0\underline{\underline{\Pi}}^T$ et on réécrit la fonction à minimiser sous la forme :

$$\Phi(\boldsymbol{\mu}) = \frac{1}{2}\left\|\underline{\underline{\Pi}}\,\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right\|^2_{\underline{\underline{\mathbf{K}}}'} + \frac{c}{2}\left\|\boldsymbol{\mu} - \boldsymbol{\mu}^0\right\|^2 \tag{4.8}$$

Dans notre cas particulier, la valeur moyenne des paramètres vaut 0. En différentiant (4.8), on obtient les expressions suivantes :

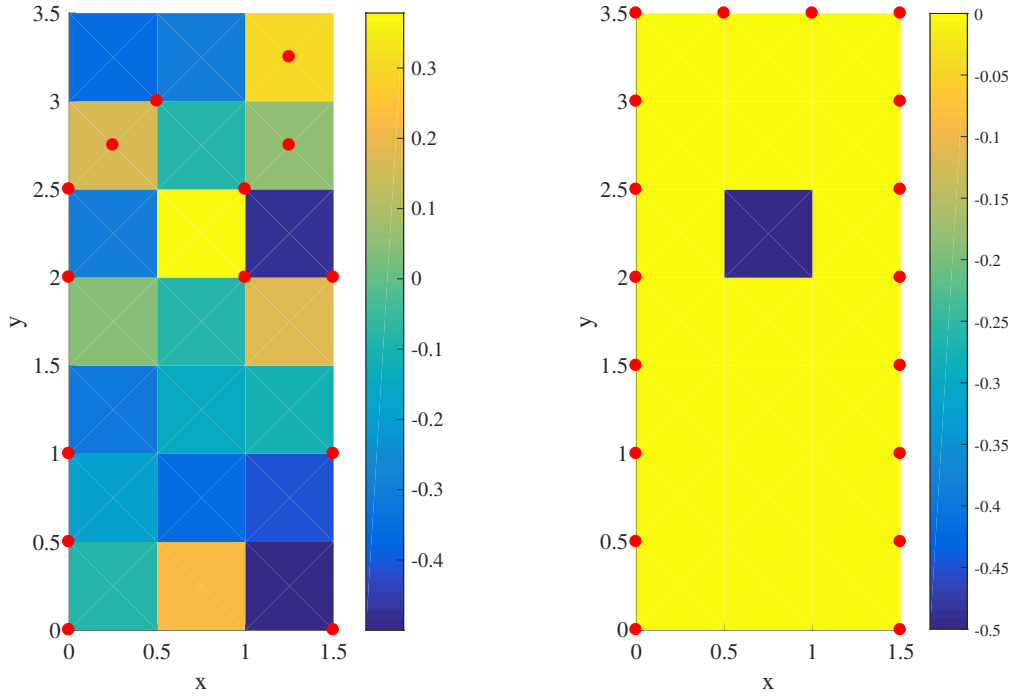
$$\begin{aligned}
\nabla\Phi &= \left[\underline{\underline{\Pi}}\nabla_{\boldsymbol{\mu}}\underline{\mathbf{X}}(\boldsymbol{\mu})\right]^T\underline{\underline{\mathbf{K}}}'\left(\underline{\underline{\Pi}}\,\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right) + c\boldsymbol{\mu} \\
\nabla^2\Phi &= \left[\underline{\underline{\Pi}}\nabla^2_{\boldsymbol{\mu}}\underline{\mathbf{X}}(\boldsymbol{\mu})\right]^T\underline{\underline{\mathbf{K}}}'\left(\underline{\underline{\Pi}}\,\underline{\mathbf{X}}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}}\right) + \left[\underline{\underline{\Pi}}\nabla_{\boldsymbol{\mu}}\underline{\mathbf{X}}(\boldsymbol{\mu})\right]^T\underline{\underline{\mathbf{K}}}'\underline{\underline{\Pi}}\nabla_{\boldsymbol{\mu}}\underline{\mathbf{X}}(\boldsymbol{\mu}) + c\underline{\underline{\mathbf{1}}}
\end{aligned} \tag{4.9}$$

Le premier terme de la Hessienne est d'ordre deux et d'un coût très élevé. Nous avons utilisé l'approximation classique qui consiste à le négliger. Ainsi, seul le Jacobien de Φ doit être calculé à chaque itération (4.7) de l'algorithme. On utilise pour cela la méthode des différences finies :

$$\nabla_{\boldsymbol{\mu}}\underline{\mathbf{X}}\left(\boldsymbol{\mu}^i\right) = \begin{bmatrix} | & & | \\ \frac{\partial\underline{\mathbf{X}}}{\partial\mu_1} & \cdots & \frac{\partial\underline{\mathbf{X}}}{\partial\mu_{N_E}} \\ | & & | \end{bmatrix} \text{ avec : } \frac{\partial\underline{\mathbf{X}}}{\partial\mu_k} \approx \frac{1}{\delta\mu_k}\left(\underline{\mathbf{X}}\left(\boldsymbol{\mu}^i + \begin{bmatrix} 0 \\ | \\ 0 \\ \delta\mu_k \\ 0 \\ | \\ 0 \end{bmatrix}\right) - \underline{\mathbf{X}}\left(\boldsymbol{\mu}^i\right)\right) \tag{4.10}$$

Remarque 32 (Coût du Jacobien) *Chaque calcul du Jacobien impose de calculer $N_p + 1$ solutions particulières du problème direct.*

2.1.3 Bruit et nombre de points de mesure



(a) Champ de paramètres aléatoire, points de mesures aléatoires

(b) Inclusion de faible rigidité, mesures au bord du domaine

FIGURE 4.5 : Problèmes à 21 paramètres

On présente FIG.4.5 deux problèmes-tests plus simples que les problèmes de grandes tailles présentés FIG.4.4. Il permettent de mettre en évidence l'influence de deux paramètres principaux de la méthode exposée ci-dessus.

FIG.4.6 présente la convergence de l'Erreur $\mathscr{E}_\mu$ selon le nombre de points de mesures choisi. Il n'y a pas de bruit dans ces problèmes et on constate que dès que suffisamment de points de mesures sont disponibles, le problème est bien posé et l'erreur converge vers des valeurs très faibles. C'est le cas par exemple si 30% des points sont sélectionnés comme illustré FIG.4.5a. Si seulement 20% des points sont des points de mesure, le problème est sous déterminé. C'est la régularisation de Tikhonov $c \neq 0$ qui permet la convergence de l'algorithme vers une solution, certes assez éloignée de la solution exacte. On visualise par ailleurs l'influence de la régularisation pour le problème où 30% des points sont conservés (courbes bleue et jaune) : l'erreur n'atteint plus le zéro numérique lorsque $c \neq 0$.

En ajoutant un bruit de mesure, on perturbe le problème. On constate FIG.4.7 que même des niveaux de bruit très faibles génèrent des erreurs élevées. FIG.4.8 permet

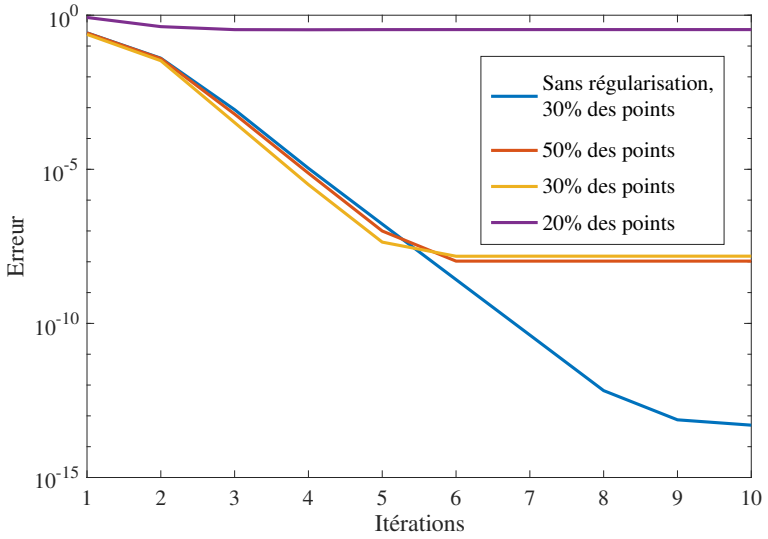


FIGURE 4.6 : Influence du nombre de points de mesures, problème à 21 paramètres à champ aléatoire, voir FIG.4.5a

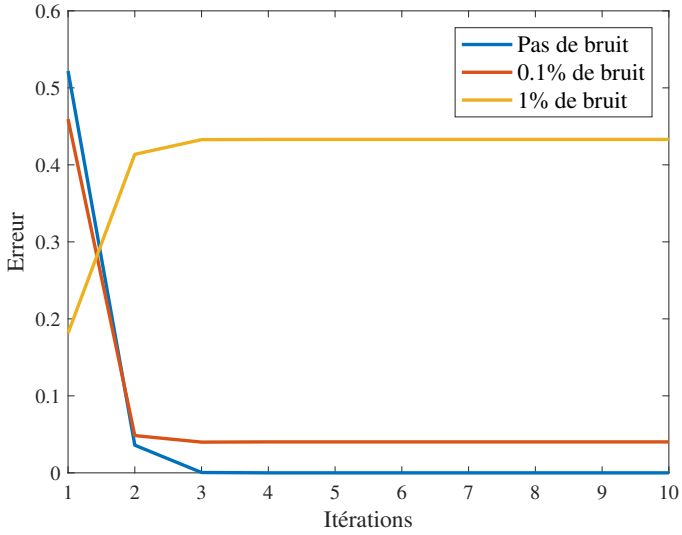
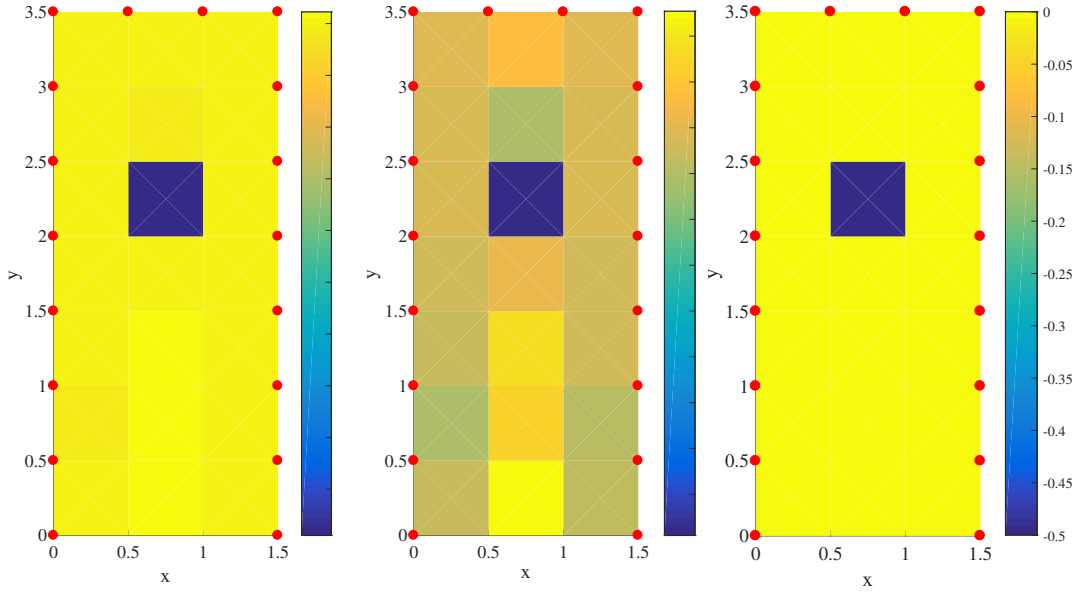


FIGURE 4.7 : Influence du bruit, problème à 21 paramètres avec inclusion, voir FIG.4.5b

de visualiser les champs identifiés à partir de mesures bruitées. En particulier, avec 1% de bruit l'erreur augmente avant de stagner. En effet, l'initialisation de l'algorithme (la solution moyenne) donne une erreur $\mathscr{E}_\mu$ inférieure à la valeur de cet estimateur pour la solution visible FIG.4.8b. Il est cependant bien plus satisfaisant d'obtenir ce champ qui met bien en évidence la présence d'une inclusion. Dans ce contexte, un autre es-

timateur tel que la valeur du résidu peut apporter des informations supplémentaires (voir REM.31). Dans ce cas particulier, le résidu décroit jusqu'à la 4ème itération avant de stagner.



(a) Bruit = 0.1%

(b) Bruit = 1%

(c) Solution avec ROM, 1 itération globable

FIGURE 4.8 : Solutions de problèmes à 21 paramètres avec inclusion (voir FIG.4.5b) pour différentes sources de perturbation

2.2 Résultats et performances

2.2.1 Utilisation du modèle réduit

On rappelle qu'il est nécessaire de calculer $N_p + 1$ solutions particulières du problème direct à chaque itération (voir REM.32). La première stratégie d'utilisation du modèle réduit consiste à y faire appel à chaque fois qu'on a besoin d'une solution simulée. On va donc minimiser la fonctionnelle suivante :

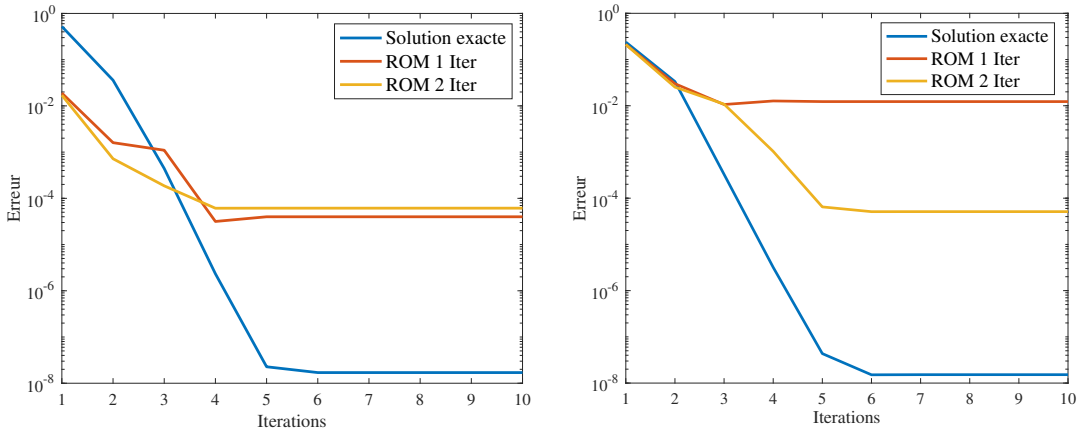
$$\Phi(\boldsymbol{\mu}) = \frac{1}{2} \left\| \underline{\underline{\Pi}} \underline{\mathbf{X}}_{ROM}(\boldsymbol{\mu}) - \underline{\mathrm{U}}_{\mathscr{M}} \right\|^2_{\underline{\underline{\mathbf{K}}}'} + \frac{c}{2} \left\| \boldsymbol{\mu} - \boldsymbol{\mu}^0 \right\|^2 \tag{4.11}$$

où $\underline{\mathbf{X}}_{ROM}$ est un modèle réduit issu de la PM-PGD. On fait en particulier appel à $\underline{\mathbf{X}}_{ROM}$ pour tous les calculs directs nécessaires à l'approximation de la Jacobienne en (4.10).

$\underline{\mathbf{X}}_{ROM}$ est un tenseur sous une forme à variables séparées (stockée sous un format monoéchelle, voir 1.2.2). On connait donc explicitement sa dépendance paramétrique.

La deuxième stratégie consiste à effectuer les dérivations par différences finies directement sur les fonctions $\underline{\gamma}$ des paramètres de $\underline{\mathbf{X}}_{ROM}$. Ces dérivées sont stockées et forment un nouveau modèle réduit du gradient de $\underline{\mathbf{X}}$ noté $\nabla\underline{\mathbf{X}}_{ROM}$. Il est exactement équivalent de faire appel à ce modèle réduit et de calculer les variations de $\underline{\mathbf{X}}$ via la formule (4.10) en utilisant $\underline{\mathbf{X}}_{ROM}$ car dans tous les cas des différences finies sont utilisées. Les temps de calcul associés sont cependant très différents (voir partie 2.2.2).

On constate FIG.4.8c que l'influence du modèle réduit n'est pas visible à l'œil nu dans ce cas particulier. FIG.4.9 illustre la convergence de l'algorithme pour les deux cas-tests présentés FIG.4.5. On constate que pour résoudre le problème à une inclusion, le modèle réduit permet d'obtenir une solution très satisfaisante même avec une seule itération globale. Ce n'est pas surprenant au regard du principe de construction de la PM-PGD : les contributions $\Delta\underline{\mathbf{X}}_{(E)}$ ajoutées à chaque itération locale de ALG.5 décrivent parfaitement l'influence d'une modification locale sur un sous-domaine Ω_E lorsque tous les autres ont une rigidité moyenne.



(a) Problème avec inclusion, FIG.4.5b (b) Problème aléatoire, FIG.4.5a

FIGURE 4.9 : Influence de l'utilisation d'un modèle réduit PM-PGD après une ou deux itérations globales

Au contraire, le problème à champ aléatoire présenté FIG.4.9b est approximé moins précisément par $\underline{\mathbf{X}}_{ROM}$ et il est nécessaire d'effectuer deux itérations complètes de la PM-PGD pour obtenir une solution avec une erreur $\mathscr{E}_\mu$ inférieure au pourcent.

Remarque 33 (Interprétation du modèle réduit) *En introduisant une perturbation dans les solutions simulées, l'utilisation du modèle réduit a une influence sur le système proche de celle de l'ajout d'un bruit au niveau des mesures.*

2.2.2 Performances

Les performances de l'inversion du problème tridimensionnel à 120 paramètres illustré FIG.4.4a sont présentés TAB.4.7. Dans cet exemple aucun bruit de mesure n'a

	Erreur $\mathscr{E}_\mu$ à convergence	Temps de calcul (secondes)
Résolution exacte	0 %	120 s
Utilisation du modèle réduit, 1 itération globale (appel au modèle)	42 %	64 s
Utilisation du modèle réduit, 1 itération globale (utilisation de $\nabla\underline{\mathbf{X}}_{ROM}$)	42 %	18 s
Utilisation du modèle réduit, 2 itérations globales (utilisation de $\nabla\underline{\mathbf{X}}_{ROM}$)	18 %	~3000 s

TABLEAU 4.7 : Performances de techniques de résolution, problème 3D à 120 paramètres FIG.4.4a

été rajouté. Dans tous les cas mentionnés, les différences entre les champs de paramètres exacts et les champs identifiés sont difficilement visibles à l'œil nu. Ce problème est suffisamment déterminé et sans bruit ni approximation de la solution via un modèle réduit l'erreur $\mathscr{E}_\mu$ tend vers 0.

La comparaison des temps de calcul (voir REM.29) permet de mettre en évidence l'apport d'un modèle réduit adapté tel que celui issu de la première itération de la PM-PGD. En particulier, l'utilisation du modèle réduit du Jacobien est très efficace et permet de trouver une solution 6 fois plus rapidement. Au contraire, si un modèle réduit trop lourd est utilisé, celui issu de la deuxième itération de la PM-PGD par exemple, la tendance s'inverse et le temps nécessaire aux interpolations dans $\nabla\underline{\mathbf{X}}_{ROM}$ dépasse de beaucoup celui consacré aux résolutions à faire pour déterminer la solution exacte du problème. Comme attendu, ce modèle réduit plus lourd permet bien d'obtenir une solution finale plus précise, même s'il n'est pas pertinent de l'utiliser en pratique.

	Sans utiliser le modèle réduit $\underline{\mathbf{X}}_{ROM}$	En appelant le modèle réduit $\underline{\mathbf{X}}_{ROM}$	En utilisant le modèle réduit $\nabla\underline{\mathbf{X}}_{ROM}$
300 paramètres, 10 itérations	~800 s	~650 s	~170 s
1000 paramètres, 5 itérations	~10800 s	~5400 s	~600 s

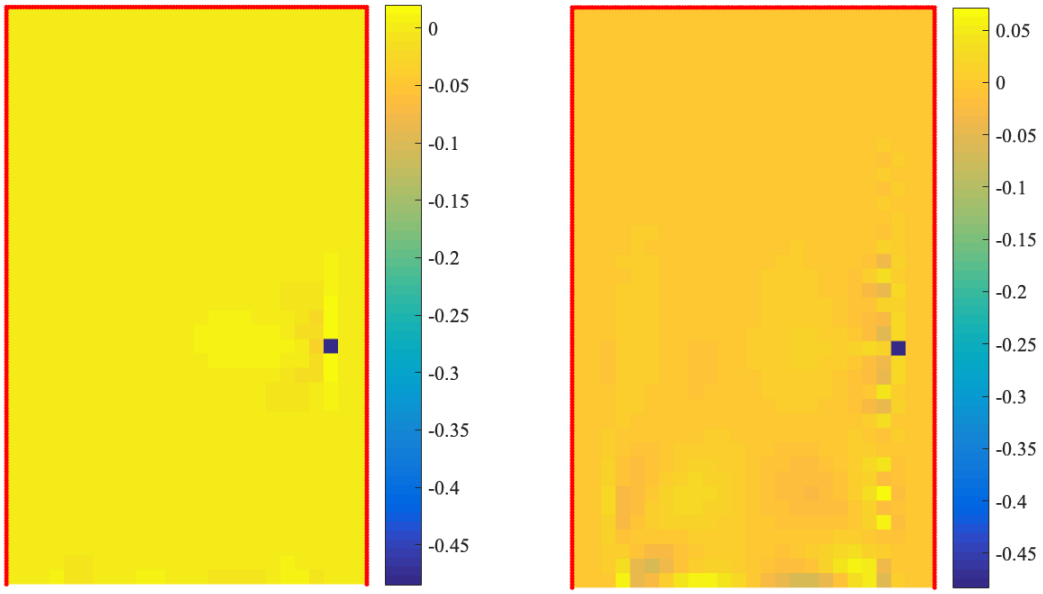
TABLEAU 4.8 : Temps de calcul associés à la résolution de problèmes bidimensionnels tels que présenté FIG.4.4b

En deux dimensions mais avec plus de paramètres, on constate également des gains de temps lorsqu'un modèle réduit issu de la première itération globale est utilisé. Il est de toute manière inutile de générer la deuxième itération dans ces cas de trop grandes dimensions (voir REM.28). On obtient une accélération allant jusqu'à un facteur 18

pour les problèmes les plus lourds à 1000 paramètres.

Remarque 34 *Les temps de calcul présentés* TAB.*4.8 sont indépendants des champs à identifier. Que ce soit pour un champ aléatoire ou une inclusion par exemple, le nombre de résolutions à effectuer est le même et seul le nombre d'itérations de Newton peut changer.*

Le calcul de $\nabla \underline{\mathbf{X}}_{ROM}$ à partir de $\underline{\mathbf{X}}_{ROM}$ est une opération simple et rapide (quelques secondes pour le cas à 1000 paramètres) possible grâce à la connaissance explicite des champs espace-paramètres. L'utilisation de ce modèle réduit du gradient dès que les variations paramétriques d'une fonctions sont nécessaires est donc un avantage considérable pour réduire les temps de calcul.

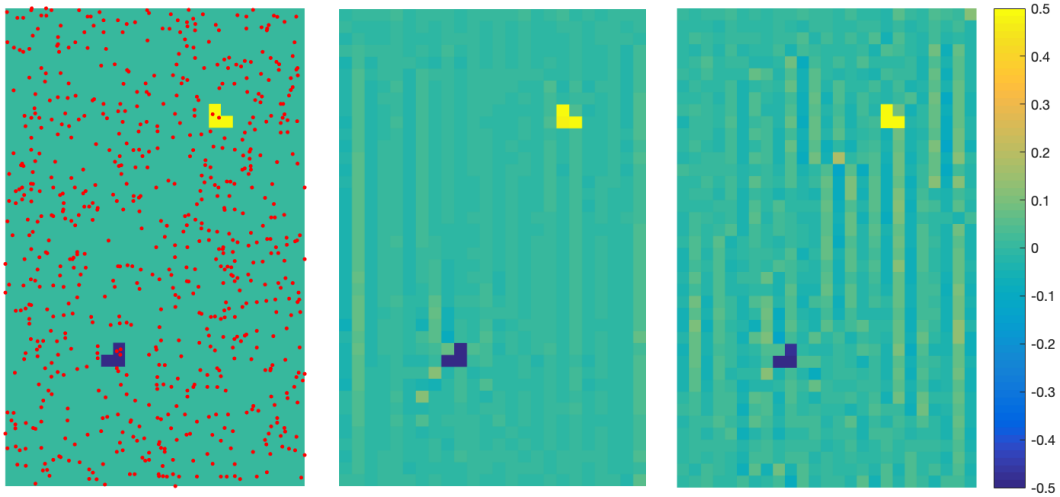


(a) Sans utilisation du modèle réduit : Erreur $\mathscr{E}_\mu = 0.17$

(b) Avec utilisation du modèle réduit, 1 Itération globale : Erreur $\mathscr{E}_\mu = 0.72$

FIGURE 4.10 : Problèmes inverse à 1000 paramètres : identification d'une inclusion via des mesures aux bords

Les problème inverses retenus peuvent être difficiles à résoudre car très mal posés. Par exemple, même sans bruit de mesure, on ne converge pas vers la solution pour le problème initialement présenté FIG.4.4b et ce indépendamment de l'utilisation d'un modèle réduit. Le problème à une seule inclusion illustré FIG.4.10 est plus facile à résoudre, notamment car l'inclusion est suffisamment proche d'un bord et donc d'une zone riche en information. On constate FIG.4.10a que, sans bruit de mesure, la solution n'utilisant pas d'approximation est quand même perturbée. Dans ce cas, l'utilisation d'un modèle réduit permet bien aussi de retrouver l'inclusion (voir FIG.4.10b), tout en offrant un gain de temps online d'un facteur 18.



(a) Champ à identifier et points de mesures

(b) Résolution avec utilisation du modèle réduit, pas de bruit $\mathscr{E}_\mu$ = 0.50

(c) Résolution avec utilisation du modèle réduit, bruit = 1%, $\mathscr{E}_\mu$ = 1.07

FIGURE 4.11 : Problèmes inverse à 1000 paramètres : identification d'inclusions via des mesures aléatoires

Remarque 35 (Perturbation de la solution) *Les solutions visibles* FIG.*4.10b ou* FIG.*4.11c par exemple sont perturbées par des oscillations de petites longueurs d'ondes. Celles-ci sont classiquement évitées grâce à une régularisation du gradient. On n'a pas utilisé cette correction dans nos cas-tests car on cherche à identifier des inclusions. On ne souhaite donc pas lisser les fortes discontinuités locales associées.*

On a réussi à retrouver des « motifs » formés par des inclusions tels que ceux illustrés FIG.4.4b à partir de mesures de bords uniquement pour un problème de plus petite taille à 300 paramètres. Pour le cas à 1000 paramètres, on a choisit un ensemble de points mesurés fournissant plus d'information : 1400 points choisis aléatoirement sur tout le domaine ont été retenus FIG.4.11a. Certes, ce choix a peu de sens physiquement mais il permet de tester notre méthode. Il est proche de celui utilisé par la technique FEMU (connaissance du déplacement expérimental en tous points grâce à la corrélation d'image) et particulièrement robuste au bruit de mesure. En particulier, l'utilisation du modèle réduit permet de déterminer efficacement la position et la rigidité des inclusions (FIG.4.11b) et ce même en présence d'un bruit de 1% (FIG.4.11c).

Il est délicat d'établir le lien entre la précision du modèle réduit utilisé et les erreurs qu'il va effectivement engendrer comme expliqué REM.26. Cette question ne possède de toute manière pas forcément de réponse pratique car il est souvent impossible (REM.28) ou très coûteux (voir TAB.4.7) d'augmenter la précision du modèle réduit.

Un autre inconvénient des modèles réduits présentés ici réside dans le fait qu'ils

ne sont adaptés qu'à une seule configuration géométrique et à des CL particulières. En les enrichissant, on pourrait y inclure ce type de paramètres. Ainsi, on disposerait d'un modèle réduit utilisable pour plusieurs problèmes différents, le problème d'identification ne portant pas sur l'ensemble des paramètres pris en compte dans cet abaque virtuel.

Malgré ces limites, l'utilisation des modèles réduits PM-PGD offre généralement des gains de temps considérables. Ils sont particulièrement intéressants si de nombreux problèmes d'inversion doivent être réalisés ou si dans un contexte industriel on ne souhaite pas attendre plusieurs heures un résultat mais seulement quelques minutes (voir TAB.4.8).

Conclusion et perspectives

Les résultats présentés dans ce manuscrit sont prometteurs : des modèles réduits à grand nombre de paramètres ont pu être construits (voir par exemple TAB.4.3). La PM-PGD est à notre connaissance le seul algorithme capable de traiter de tels problèmes. Le coût des solutions proposées peut cependant être élevé et les erreurs d'approximation liées à l'utilisation du modèle réduit construit sont loin d'être négligeables. On a cependant présenté en IV.2 des exemples d'utilisations pratiques pour lesquels des gains de temps *online* conséquents ont été réalisés.

De nombreux outils ont été mis en œuvre pour atteindre ces performances, notre conclusion est l'occasion de revenir sur les trois apports originaux majeurs de cette thèse.

Un nouveau **format de données** a été introduit (en I.2.3). Ce format tensoriel « multiéchelle en paramètres » est basé sur des considérations mécaniques, notamment le principe de Saint-Venant, ce qui en fait un outil idéal pour traiter nos problèmes. Son utilisation a amené de nombreux questionnements techniques, notamment sur sa manipulation et la manière de l'utiliser en s'appuyant sur les outils d'analyse tensorielle classiques. Ces réflexions ont façonné les algorithmes PM-PGD.

La **WTDG** est une méthode de discrétisation spatiale implémentée pour la première fois à l'occasion de cette thèse. Il s'agit d'une méthode n'imposant pas la continuité des champs d'un élément à l'autre, propriété très utile dans la prise en compte du nouveau format de données. C'est aussi une méthode particulièrement robuste basée sur la physique du problème. Son implémentation discutée en II.1.2 permet de compenser directement les erreurs locales lors de la procédure de PM-PGD.

Enfin, les **algorithmes PM-PGD** dans les deux versions présentées sont des techniques originales qui nous ont permis de construire ces modèles réduits de très grandes dimensions. Ces méthodes sont elles aussi basées sur l'aspect multiéchelle en paramètres des problèmes traités ce qui permet la construction de solutions approchées par superposition de corrections locales. Dans la première version de l'algorithme (chapitre 2), on compense rigoureusement les erreurs locales grâce aux propriétés de la WTDG. La technique développée au chapitre 3 utilise un solveur EF et cherche à minimiser les discontinuités d'un sous-domaine à l'autre lors des corrections locales. Elle s'est montrée tout aussi efficace.

A ce stade de la recherche, la supériorité de la deuxième version de la PM-PGD

semble avérée. Elle présente des performances similaires à la version incluant la WTDG tout en proposant en l'état actuel une plus grande souplesse d'utilisation. Une procédure totalement non-intrusive peut même être envisagée (voir III.3) ce qui permettrait une implémentation couplée avec des solveurs industriels simplifiée.

Par ailleurs, on a vu en IV.1.2.2 qu'il était pratique voire indispensable de stocker le modèle réduit final sous une forme monoéchelle. Pour une première utilisation de la PM-PGD, une version plus simple n'utilisant pas le format multiéchelle proposé resterait fonctionnelle pour des nombres de paramètres allant jusqu'à quelques centaines. L'utilisation du multiéchelle et en particulier des opérations de troncature est néanmoins indispensable pour résoudre des problèmes prenant en compte jusqu'au millier de paramètres.

D'importants développements restent à poursuivre à l'issue de cette thèse, que ce soient des questions théoriques à approfondir ou des points techniques à améliorer. Par ailleurs, les applications potentielles sont nombreuses. Certaines de ces perspectives sont listées ci-dessous en fonction de la durée prévisionnelle qu'il serait nécessaire d'y consacrer.

Perspectives à court terme :

— Une application à des problèmes d'optimisation topologique est en cours d'étude. Comme les problèmes inverses d'identification de champs, les problèmes de répartition optimale de matière pourraient être accélérés via l'utilisation de modèles réduits à grands nombres de paramètres.
— Les performances de l'adaptation en grande dimension de la Reference Point Method (RPM, voir Annexe B.2) peuvent sans doute être améliorées. Des études supplémentaires pourraient permettre de déterminer des techniques d'interpolation ou de construction des bases plus pertinentes.
— La conception de modèles réduits de grandes dimensions consacrés à l'étude de quantité d'intérêt (QI) permettrait d'obtenir des précisions bien supérieures pour certaines grandeurs données. En particulier l'utilisation du format multiéchelle est particulièrement adapté à des QI locales car on peut enrichir une solution localement sans modifier le modèle réduit sur tous les sous-domaines.

Perspectives à moyen terme :

— La WTDG a été implémentée avec succès dans un cas particulier simple mais peu flexible. Il serait nécessaire pour mettre en valeur tout son potentiel de disposer d'un solveur permettant d'utiliser des éléments de géométries variées et des fonctions de base d'ordres différents. On a mis en évidence la robustesse de cette méthode sur quelques exemples, il faudrait étudier d'autres paramètres tels que ses propriétés de convergence ou sa précision pour pouvoir la comparer aux autres solveurs continus ou discontinus classiquement utilisés en mécanique des structures.
— Il serait intéressant de tester les performances de la PM-PGD pour prendre en compte des nombres importants de paramètres globaux. Si ceux-ci ne sont plus

localisés spatialement, il est cependant possible de les discrétiser sur un grand nombre de sous-domaines locaux. Des paramètres géométriques pourraient servir de base à cette étude.

— Pour que la PM-PGD devienne accessible à des utilisateurs non expérimentés, il est nécessaire d'automatiser plusieurs étapes telles que les choix de compressions par exemple (voir IV.1.1). Une implémentation exclusivement non-intrusive dans un logiciel industriel serait alors envisageable. Les itérations locales sont alors plus coûteuses mais elles restent parallélisables et l'utilisation du calcul haute performance permettrait des gains de temps offline conséquents.

Enfin à plus long terme, la PM-PGD pourrait être étendue aux problèmes non-linéaires. On peut en effet envisager de la coupler avec la LaTIn en s'inspirant de la méthode LaTIn-PGD classique. Les objectifs seraient alors bien plus modestes en terme de nombre de paramètres mais la résolution de problèmes non-linéaires en incluant plusieurs dizaines reste un challenge aujourd'hui. Des stratégies basées sur la PM-PGD pourraient permettre d'y accéder.

Jusqu'à présent la PM-PGD a été implémentée et appliquée uniquement par l'auteur de ce manuscrit. Son développement ne devrait cependant pas s'arrêter là. Les perspectives énoncées en conclusion laissent entrevoir de nombreuses applications potentielles, mais aussi les améliorations nécessaires à une utilisation simplifiée par les ingénieurs de demain de cette méthode affectueusement surnommée *Big-PGD*.

Annexe A

Extended-PGD

Cette annexe présente un résumé de résultats détaillés dans le mémoire [Paillet, 2016a]. Ceux-ci n'ont pas été publiés mais ont fait l'objet de communications lors de conférences.

L'Extended-PGD a été développée à partir de [Ladevèze, 2014] et s'inspire des préconditionnements de la méthode du gradient conjugué utilisés par exemple en décomposition de domaine [Rey, 1998]. On décrit en partie I.4 une méthode classique de construction de la PGD qui impose de calculer une solution spatiale complète à chaque itération du point fixe. Cette opération est de loin la plus coûteuse de toute la procédure lorsqu'on l'applique à des problèmes industriels à plusieurs centaines de milliers de degrés de liberté. La méthode introduite dans cette annexe est conçue pour minimiser le nombre de ces résolutions.

Elle n'est donc pas spécifiquement associée aux problèmes à grands nombres de paramètres. Cependant, comme les points fixes en grandes dimensions convergent bien plus lentement et nécessitent plus que deux ou trois itérations pour fournir un résultat satisfaisant, c'est dans ces cas que l'Extended-PGD sera la plus efficace.

1 Algorithme

On s'intéresse toujours dans cette annexe au problème (1.3) traité dans la thèse. Les sous-domaines associés aux paramètres sont réguliers (voir FIG.A.1a par exemple) et discrétisés spatialement grâce à la méthode des éléments finis (EF). L'Extended-PGD est conçue pour générer séparément une base spatiale puis dans un second temps les fonctions paramétriques de manière plus classique. Cette méthode permet de minimiser le nombre de résolutions de problèmes spatiaux. On rappelle la forme du problème à résoudre après discrétisation EF :

$$\begin{aligned} &\text{Trouver } \underline{\mathbf{X}} \in \mathcal{X} \text{ où } \mathcal{X} = \left\{ \underline{\mathbf{X}} : \boldsymbol{\Sigma}_\mu \to \mathbb{R}^d \,\middle|\, \int_{\boldsymbol{\Sigma}_\mu} \left\| \underline{\mathbf{X}}(\boldsymbol{\mu}) \right\|^2 d\boldsymbol{\mu} < \infty \right\} \\ &\text{tel que : } \quad \forall \boldsymbol{\mu} \in \boldsymbol{\Sigma}_\mu \qquad \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \underline{\mathbf{X}}(\boldsymbol{\mu}) = \underline{\mathrm{F}} \end{aligned} \tag{A.1}$$

1.1 Construction d'une base spatiale

On initialise cet algorithme grâce à la solution nominale comme on le fait pour toutes les versions de la PM-PGD présentées dans ce manuscrit. Soit $\underline{\underline{\mathbf{K}}}^0$ la matrice de rigidité EF moyenne. On pose :

$$\underline{\mathrm{X}}^0 = \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\mathrm{F}} \tag{A.2}$$

On associe à $\underline{\mathrm{X}}^0$ sa composante spatiale $\underline{\widetilde{\mathrm{X}}}^{(0)}$ qui donne le premier mode de la base qu'on cherche à construire (dans ce cas particulier, toutes les composantes paramétriques de $\underline{\mathrm{X}}^0$ sont constantes et égales à 1 donc $\underline{\mathrm{X}}^0 = \underline{\widetilde{\mathrm{X}}}^{(0)}$). Le résidu associé à cette première base de dimension 1 est :

$$\underline{\mathbf{R}}^0(\boldsymbol{\mu}) = \underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})\underline{\mathrm{X}}^0 \tag{A.3}$$

Remarque 36 *La rigidité $\underline{\underline{\mathbf{K}}}$ dépend de l'ensemble des paramètres $\boldsymbol{\mu}$. Elle est connue analytiquement sous une forme à variables séparées qui découle de la définition des lois de comportement. En pratique, si les paramètres sont proportionnels aux modules d'Young comme présenté en* (1.5)*, on peut construire $\underline{\underline{\mathbf{K}}}$ exactement grâce à $N_p + 1$ modes. Ainsi, $\underline{\mathbf{R}}^0(\boldsymbol{\mu})$ est lui aussi un champ de grande dimension décrit exactement par $N_p + 2$ modes.*

La méthodologie suivie s'inspire de la méthode du gradient conjugué : on cherche à construire pas à pas une base de Krylov $\mathscr{K}_K$ pour approcher $\underline{\mathbf{R}}^0$:

$$\mathscr{K}_K = \left\{\underline{\widetilde{\mathrm{X}}}^{(0)}, \cdots, \underline{\widetilde{\mathrm{X}}}^{(K)}\right\} \tag{A.4}$$

Remarque 37 *On a nommé «*résidu*» le champ $\underline{\mathbf{R}}$. Au contraire, dans les deux versions de la PM-PGD, on cherche à compenser une «*erreur*» $\mathbf{R}$ qui traduit le non respect de la loi de comportement. Ces deux grandeurs de dimensions différentes jouent des rôles similaires dans les algorithmes. Elles traduisent l'erreur qui découle des variations paramétriques des lois de comportement choisies et qu'on cherche à corriger à chaque itération.*

La première étape de la construction de la base consiste à construire une décomposition PGD de $\underline{\mathbf{R}}^0$ telle que :

$$\underline{\mathbf{R}}^0 \simeq \underline{\widetilde{\mathbf{R}}}\, \boldsymbol{X}(\boldsymbol{\mu}) \tag{A.5}$$

On a choisi de garder la notation $\boldsymbol{X}$ introduite au chapitre 1 qui décrit une fonction multiparamétrique de grande dimension car il n'est pas nécessaire dans cette partie d'individualiser les fonctions de chaque paramètre. Une méthode possible et présentée en I.2.2.2 consiste à minimiser la différence $\left(\underline{\mathbf{R}}^0 - \underline{\widetilde{\mathbf{R}}}\, \boldsymbol{X}(\boldsymbol{\mu})\right)$ ce qui donne :

$$\forall \left(\underline{\widetilde{\mathbf{R}}}^*, \boldsymbol{X}^*\right) \in \mathbb{R}^d \times \boldsymbol{\Sigma}_\mu \quad \left\langle \underline{\widetilde{\mathbf{R}}}^* \boldsymbol{X} + \underline{\widetilde{\mathbf{R}}}\, \boldsymbol{X}^*, \underline{\mathbf{R}}^0(\boldsymbol{\mu}) - \underline{\widetilde{\mathbf{R}}}\, \boldsymbol{X}(\boldsymbol{\mu}) \right\rangle_{\mathscr{E}} = 0 \tag{A.6}$$

Ce produit scalaire énergétique intégré sur le domaine paramétrique a été introduit en (1.30). On pourrait résoudre ce problème via un point fixe comme en I.2.2.2. Par exemple, $\underline{\mathbf{R}}^* = 0$ donne le problème particulier :

$$\boldsymbol{X}(\boldsymbol{\mu}) = \frac{\underline{\widetilde{\mathbf{R}}}^T \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\mathbf{R}}^0}{\underline{\widetilde{\mathbf{R}}}^T \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\widetilde{\mathbf{R}}}} \tag{A.7}$$

Remarque 38 (Approximation du résidu connu $\underline{\mathbf{R}}$) *La construction de la base spatiale présentée dans cette partie ne fait pas appel aux méthodes PGD de résolution d'EDP décrites en I.4 mais uniquement aux techniques d'approximation d'un champs connu, ici le résidu* $\underline{\mathbf{R}}$.

Un autre moyen d'approximer $\mathbf{R}^0$ par des fonctions à variables séparées consiste à minimiser $\left|\left|\left|\mathbf{R}^0 - \underline{\widetilde{\mathbf{R}}}\,\boldsymbol{X}(\boldsymbol{\mu})\right|\right|\right|$. Cette PGD par minimisation du résidu est mentionnée dans le manuscrit en 4.1.2 dans le contexte de la résolution d'EDP.

En développant cette expression et en y intégrant l'équation (A.7), on obtient :

$$\left|\left|\left|\underline{\mathbf{R}}^0 - \underline{\widetilde{\mathbf{R}}}\,\boldsymbol{X}(\boldsymbol{\mu})\right|\right|\right| = \left|\left|\left|\underline{\mathbf{R}}^0\right|\right|\right| - \left\langle \underline{\mathbf{R}}^0, \underline{\widetilde{\mathbf{R}}}\,\boldsymbol{X}(\boldsymbol{\mu})\right\rangle_{\mathscr{E}} \tag{A.8}$$

On cherche donc à maximiser l'expression :

$$\mathscr{R}(\underline{\widetilde{\mathbf{R}}}) = \left\langle \underline{\mathbf{R}}^0, \underline{\widetilde{\mathbf{R}}}\,\boldsymbol{X}(\boldsymbol{\mu})\right\rangle_{\mathscr{E}} = \frac{\underline{\widetilde{\mathbf{R}}}^T \underline{\underline{\mathbf{K}}}^{0^{-1}} \left[\int_{\boldsymbol{\Sigma}_\mu} \underline{\mathbf{R}}^0 \underline{\mathbf{R}}^{0^T} d\boldsymbol{\mu}\right] \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\widetilde{\mathbf{R}}}}{\underline{\widetilde{\mathbf{R}}}^T \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\widetilde{\mathbf{R}}}} \tag{A.9}$$

Il s'agit de la forme classique d'un quotient de Rayleigh généralisé. En posant $\underline{\underline{\mathbf{R}}} = \int_{\boldsymbol{\Sigma}_\mu} \underline{\mathbf{R}}^0 \underline{\mathbf{R}}^{0^T} d\boldsymbol{\mu}$, on maximise cette expression en déterminant le vecteur propre généralisé associé à la valeur propre la plus élevée de $\underline{\underline{\mathbf{R}}}$ et $\underline{\underline{\mathbf{K}}}^{0^{-1}}$.

Il est nettement plus rapide de calculer les vecteurs spatiaux optimaux via cette méthode. En effet, le problème à résoudre consiste à rechercher la valeur propre maximale d'un système, opération qui ne nécessite aucune inversion matricielle. Le point fixe issu de A.6 permet aussi d'approximer le résidu mais nécessite plus de temps car il fournit une approximation complète, spatiale et paramétrique.

On définit le second vecteur de notre base de Krylov en normalisant :

$$\underline{\widetilde{\mathbf{X}}}^{(1)} = \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\widetilde{\mathbf{R}}} \tag{A.10}$$

On suppose qu'on vient de calculer une contribution spatiale $\underline{\widetilde{\mathbf{X}}}^{(i)}$. On complète la base par itérations successives en calculant la contribution spatiale issue du résidu $\underline{\mathbf{R}}^{(i)} = \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})\underline{\widetilde{\mathbf{X}}}^{(i)}$ puis en les multipliant par $\underline{\underline{\mathbf{K}}}^{0^{-1}}$. Un algorithme de Gram-Schmidt permet d'orthogonaliser ces modes à chaque itération.

Cette procédure permet donc de calculer une base de Krylov de K vecteurs sans effectuer de résolution spatiale. Il suffit d'inverser une seule fois la matrice $\underline{\underline{\mathbf{K}}}^0$ en début de procédure, celle-ci peut être interprétée comme un préconditionneur. On a constaté en pratique que le nombre de vecteurs de la base de Krylov a une influence assez faible. Certes, l'ajout de modes de Krylov est une opération peu coûteuse, mais la base finale est bien plus lourde à manipuler.

1.2 Calcul des fonctions paramétriques

On suppose qu'on a à notre disposition une base $\mathscr{K}_K$ de K vecteurs. On cherche à exprimer la solution dans cet espace en minimisant le résidu au sens de la norme

énergétique :

$$\left\| \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \left(\sum_{k=1}^{K} \widetilde{\underline{\mathrm{X}}}^{(k)} \prod_{E \in \mathbf{E}} \gamma_E^{(k)}(\mu_E) \right) - \underline{\mathrm{F}} \right\| \tag{A.11}$$

Il est délicat de construire les K familles $\left\{ \gamma_E^{(k)} \right\}_{E \in \mathbf{E}}$ d'un coup comme on le fait pour un champs connus grâce à la CP-ALS (voir I.2.2.3) car les dépendances paramétriques de $\underline{\underline{\mathbf{K}}}$ doivent être prises en compte. Si on construit les $\left\{ \gamma_E^{(k)} \right\}_{E \in \mathbf{E}}$ de manière itérative, on peut noter cette minimisation à l'itération k sous la forme :

$$\left\| \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu}) \left(\widetilde{\underline{\mathrm{X}}}^{(k)} \prod_{E \in \mathbf{E}} \gamma_E^{(k)}(\mu_E) \right) - \underline{\mathbf{R}}^{k}(\boldsymbol{\mu}) \right\| \tag{A.12}$$

Cette minimisation peut être effectuée via la méthode de Galerkin, ou bien directement par dérivation. Cette méthode qualifiée simplement de « minimisation du résidu » est plus complexe à implémenter mais garantit la décroissance de l'erreur en norme énergétique à chaque itération.

La différentiation de A.12 par rapport à $\gamma_E^{(k)}$ conduit à l'équation :

$$\int_{\overline{\boldsymbol{\Sigma}_{\mu_E}}} \underline{\mathbf{R}}^{(k)^T} \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\underline{\mathbf{K}}} \widetilde{\underline{\mathrm{X}}}^{(k)} \prod_{E' \neq E} \gamma_{E'}^{(k)} d\overline{\boldsymbol{\mu}}_E = \gamma_E^{(k)} \int_{\overline{\boldsymbol{\Sigma}_{\mu_E}}} \widetilde{\underline{\mathrm{X}}}^{(k)^T} \underline{\underline{\mathbf{K}}} \underline{\underline{\mathbf{K}}}^{0^{-1}} \underline{\underline{\mathbf{K}}} \widetilde{\underline{\mathrm{X}}}^{(k)} \prod_{E' \neq E} \gamma_{E'}^{(k)^2} d\overline{\boldsymbol{\mu}}_E \tag{A.13}$$

On peut déterminer de cette manière N_p équations qui sont résolues via un point fixe. Cette formulation par minimisation du résidu est plus lourde à mettre en place que la méthode de Galerkin, dont la formulation en paramètres est donnée équation (1.45). La dépendance paramétriques de l'opérateur rend les intégrales de (A.13) beaucoup plus longues à calculer.

Toujours dans une optique de minimisation du nombre de fonctions spatiales à générer, on peut chercher à profiter pleinement de chaque $\widetilde{\underline{\mathrm{X}}}^{(k)}$ en capturant au maximum la richesse des variations paramétriques associées à ce mode. Ainsi, après la détermination d'une correction $\widetilde{\underline{\mathrm{X}}}^{(k)} \prod_{E \in \mathbf{E}} \gamma_E^{(k)}(\mu_E)$ de la solution, on peut chercher le mode suivant en utilisant la même fonction spatiale $\widetilde{\underline{\mathrm{X}}}^{(k)}$. Cette étape d'enrichissement, permet d'accélérer grandement la convergence de l'algorithme si les modes ont été générés via le problème aux valeur propres (A.9). En effet, cette équation a été déterminée en supposant connu le champ paramétrique optimal (A.7) qui n'a aucune raison d'être représenté parfaitement par une seule fonction à variables séparées. On obtient finalement une correction définie grâce à S jeux de fonctions paramétriques :

$$\Delta \underline{\mathbf{X}}^{k} = \widetilde{\underline{\mathrm{X}}}^{(k)} \sum_{s=1}^{S} \prod_{E \in \mathbf{E}} \gamma_E^{(k,s)}(\mu_E) \tag{A.14}$$

Remarque 39 *Lorsque qu'une fonction espace-paramètres est générée via un point fixe sur l'espace et l'ensemble des paramètres via* (A.6) *par exemple, l'ajout de fonctions paramétriques supplémentaires associées au vecteur spatial est bien moins intéressant. En effet, ce mode spatial est optimisé en considérant que les dépendances paramétriques sont aussi sous une forme à variables séparées.*

1.3 Structure de l'algorithme

On suppose qu'on dispose d'une solution approchée sur un nombre i de modes, notée $\underline{\mathbf{X}}^i$. On peut alors calculer un nouveau résidu :

$$\underline{\mathbf{R}}^i(\boldsymbol{\mu}) = \underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}(\boldsymbol{\mu})\underline{\mathbf{X}}^i \tag{A.15}$$

On peut de nouveau chercher à approximer celui-ci en lui appliquant la procédure que l'on vient de présenter :

— génération d'une base de Krylov à partir du premier mode PGD spatial de $\underline{\mathbf{R}}^i$.
— orthogonalisation par rapport aux fonctions précédentes. En pratique, on n'orthogonalise que par rapport à quelques modes précédents pour éviter que les bases ne dégénèrent. On voit en (A.14) par exemple que des informations spatiales redondantes permettent bien d'enrichir la solution.
— calcul des fonctions paramétriques associées.
— mise à jour de la solution.

1.4 Résumé

ALG.7 résume l'algorithme présenté dans le cas particulier où les bases de Krylov retenues sont de taille 1 pour alléger les notations.

Algorithm 7 Extended-PGD $K = 1$

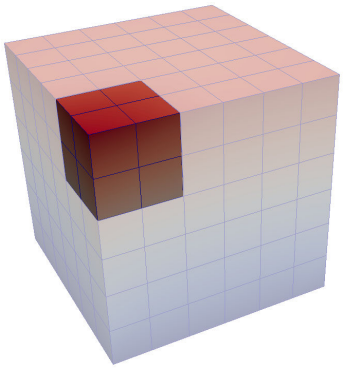
Initialisation $\underline{\widetilde{\mathrm{X}}}^{(0)} = \underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\mathrm{F}}$
On pose $\underline{\mathbf{X}}^0 = \underline{\widetilde{\mathrm{X}}}^{(0)}$
On pose : $\underline{\mathbf{R}}^0 = \underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}\,\underline{\mathbf{X}}^0$
for $i = 1$ to I_{max} **do** ▷ I_{max}, Nombre final de modes
    Calcul de $\underline{\widetilde{\mathbf{R}}}$, solution de (A.9).
    Calcul du mode $\underline{\widetilde{\mathrm{X}}}^{(i)} = \underline{\underline{\mathbf{K}}}^{0^{-1}}\underline{\widetilde{\mathbf{R}}}$ ▷ et éventuellement de toute la base $\mathscr{K}_K$
    Orthogonalisation de $\underline{\widetilde{\mathrm{X}}}^{(i)}$, $\underline{\widetilde{\mathrm{X}}}^{(i-1)}$ et $\underline{\widetilde{\mathrm{X}}}^{(i-2)}$
    Calcul de S solutions du problème (A.12) qui donne :
    $\Delta\underline{\mathbf{X}}^i = \underline{\widetilde{\mathrm{X}}}^{(i)} \sum_{s=1}^{S} \prod_{E \in \mathbf{E}} \gamma_E^{(i,s)}(\mu_E)$
    Mise à jour de la solution $\underline{\mathbf{X}}^i = \underline{\mathbf{X}}^{i-1} + \Delta\underline{\mathbf{X}}^i$
    On calcul $\underline{\mathbf{R}}^i = \underline{\mathrm{F}} - \underline{\underline{\mathbf{K}}}\,\underline{\mathbf{X}}^i$
end for

2 Résultats

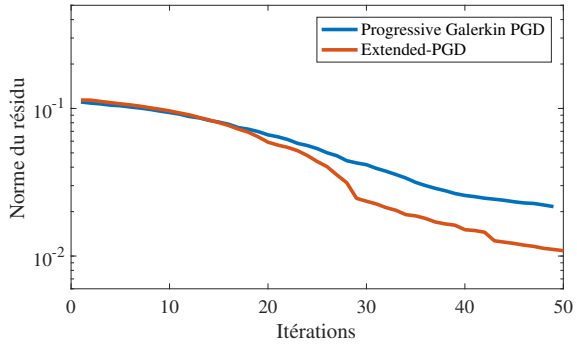
L'algorithme présenté ALG.7 possède de nombreuses variantes et paramètres. Par exemple, la tailles des bases de Krylov, le nombre S de fonctions paramétriques par

mode spatial ou le choix des méthodes de résolution (Galerkin ou minimisation du résidu) ont été étudiés précisément et se trouvent dans [Paillet, 2016a].

On se contente ici de présenter les résultats obtenus pour des nombres relativement élevés de paramètres, en choisissant des options qui optimisent la convergence de la méthode. En particulier seul le résultat du premier problème (A.10) est conservé pour construire la base $\mathscr{K}_K$ qui ne comporte donc qu'un vecteur par itération. On a choisi $S = 1$ et une génération des modes spacieux grâce à une méthode de Galerkin classique pour le cas à 27 paramètres illustré FIG.A.1 tandis qu'on a $S = 3$ et une résolution des points fixes paramétriques grâce à la minimisation du résidu présentée en (A.13) pour le cas à 64 paramètres (voir FIG.A.2).

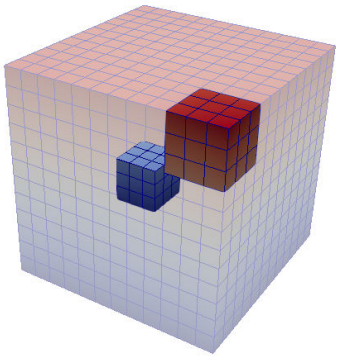


(a) Mise en valeur d'un sous-domaine

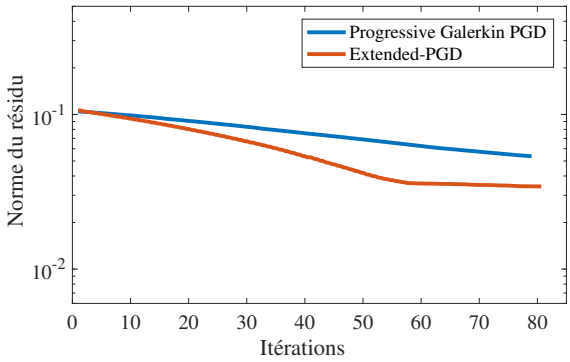


(b) Décroissance du résidu en fonction du nombre d'itérations globales

FIGURE A.1 : Problème à 27 paramètres



(a) Mise en valeur de deux sous-domaines



(b) Décroissance du résidu en fonction du nombre d'itérations globales

FIGURE A.2 : Problème à 64 paramètres

Avec de tels paramètres, l'Extended-PGD converge légèrement plus vite qu'une PGD classique. FIG.A.1 et FIG.A.2 illustrent ces performances et les comparent avec les résultats déjà obtenus FIG.1.12 avec la Progressive Galerkin PGD. La méthode Extended-PGD n'a nécessité qu'une seule inversion d'une matrice spatiale (la matrice $\underline{\underline{\mathbf{K}}}^0$). Au contraire, il a fallu inverser un système complet à chaque itération du point fixe pour la Progressive Galerkin PGD (voir équation (1.45)). Il s'agit d'un gain de temps considérable à l'échelle de problèmes industriels à très nombreux degrés de liberté spatiaux. Par contre, la solution obtenue à l'issue de l'Extended-PGD peut être plus lourde : quand $S = 3$, on a 2 modes supplémentaires à stocker à chaque itération.

Il nous faut cependant remarquer que les taux d'erreur restent élevés et les vitesses de convergence faibles. Les performances de cette méthode n'ont rien à voir avec celles de la PM-PGD pour les problèmes à grands nombres de paramètres. Il serait cependant intéressant d'étudier l'application de l'Extended-PGD à des problèmes linéaires paramétriques à grands nombres de degrés de liberté. Des gains importants de temps de calcul offline sont attendus par rapport aux stratégies plus classiques.

Annexe B

RPM en grande dimension

La RPM (*Reference Points Method*, Méthode des Points de Référence, [Ladevèze, 1997]) est une technique d'interpolation basée sur une idée simple : une solution de grande dimension est représentée localement par une approximation à variables séparées. A l'origine, elle a été introduite dans le contexte de la LaTIn-PGD pour accélérer les étapes d'intégration [Capaldo *et al.*, 2017].

L'objectif de cette thèse est d'être capable de fournir des « abaques virtuels » en très grandes dimensions. La PGD a l'avantage de proposer un format adapté à la prise en compte de paramètres mais sans utiliser de structure de données spécifique, des méthodes d'interpolation telles que l'EIM (voir I.3.1) peuvent être utilisées dans le domaine paramétrique. Cette annexe présente une autre méthode d'interpolation et son adaptation aux problèmes de grande dimension. Connaissant la valeur de la solution spatiale pour certains points particuliers de l'espace paramétrique ainsi qu'un moyen d'interpoler ces solutions, on construit bien un modèle réduit de la solution.

Cette annexe est structurée en deux parties. On rappelle d'abord les principes de construction de la RPM en 2D. Cet exemple bidimensionnel permet également de présenter visuellement les répartitions de points aléatoires qu'il est indispensable d'utiliser pour les grands nombres de paramètres. Une deuxième partie présente une généralisation de cette méthode en grande dimension que l'on applique aux cas-tests utilisés dans cette thèse pour étudier la PM-PGD dans sa version continue (voir chapitre 3).

1 Introduction : RPM à 2 dimensions

On cherche à approximer une fonction à deux variables définie par :

$$F : \left\{ \begin{array}{ccc} \Omega \times \boldsymbol{I}_\mu & \rightarrow & \mathbb{R} \\ x \times \mu & \rightarrow & F(x,\mu) \end{array} \right. \tag{B.1}$$

1.1 Formulation à variables séparées : 2 paramètres, 1 point

Soit un point $\underline{\mathrm{X}}_p \in \Omega \times \boldsymbol{I}_\mu$ de coordonnées (x_p, μ_p). On introduit les coupes de F en $\underline{\mathrm{X}}_p$:

$$\begin{aligned} \widehat{F}_p(x) &= F(x, \mu_p) \\ \widehat{F}_p(\mu) &= F(x_p, \mu) \end{aligned} \tag{B.2}$$

et la valeur de F en $\underline{\mathrm{X}}_p$: $F(x_p, \mu_p) = F_p$.

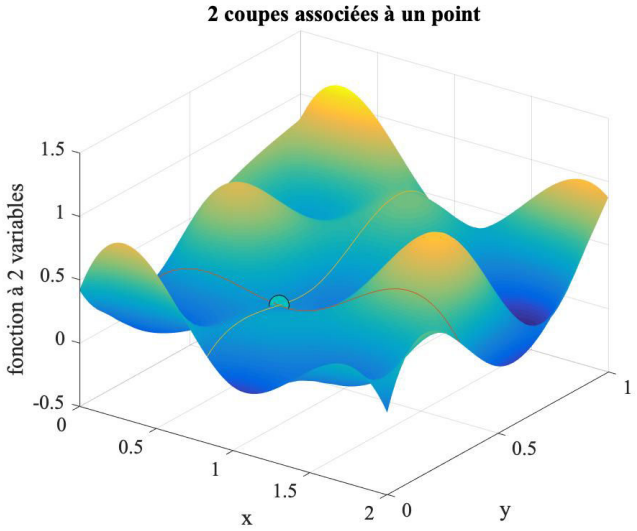


FIGURE B.1 : Fonction bidimensionnelle, un point de référence et ses deux coupes associées

On choisit d'approximer F sur la base donnée par les coupes en $\underline{\mathrm{X}}_p$ tout en respectant la valeur de F en $\underline{\mathrm{X}}_p$. On obtient donc une formulation à variables séparées de la forme :

$$F(x, \mu) \approx \breve{F}(x, \mu) = \frac{\widehat{F}_p(x)\widehat{F}_p(\mu)}{F_p} \tag{B.3}$$

Cette approximation très simple ne nécessite aucun calcul d'optimisation, seulement la connaissance des coupes, c'est à dire la valeur de la fonction à approximer F en un certain nombre de points.

1.2 RPM sur une grille régulière : 2 paramètres

On maille régulièrement les domaines Ω et $\boldsymbol{I}_\mu$ sur respectivement N_x et N_μ éléments. Au centre de chacun des $K = N_x \times N_\mu$ domaines rectangulaires Ω_p ainsi construits dans $\Omega \times \boldsymbol{I}_\mu$, on définit le point $\underline{\mathrm{X}}_p$. Comme précédemment, on peut définir sur chaque Ω_p une approximation sous forme de produit à variables séparées grâce aux deux coupes générées en ce point. On généralise cette définition via la formule :

$$\breve{F}^{(p)}(x,\mu) = \frac{\sum_{k=1}^{N_x} \widehat{F}_k(\mu) F(x_k,\mu_p).\lambda_{kp}^2}{\sum_{k=1}^{N_x} F(x_k,\mu_p)^2.\lambda_{kp}^2} \times \widehat{F}_p(x) \tag{B.4}$$

Cette approximation $\breve{F}^{(p)}$ est associée au point $\underline{\mathrm{X}}_p$ et définie sur tout le domaine $\Omega \times \boldsymbol{I}_\mu$. Elle vaut bien F_p en $\underline{\mathrm{X}}_p$ et est également exacte sur toute la coupe issue de $\underline{\mathrm{X}}_p$ dans la direction x.

Remarque 40 (Sens physique de l'approximation) *Les variables x et μ sont traitées de manière très différentes. Dans le cas ou les deux dimensions du champ approximé ne sont pas de même nature, il est plus pertinent de donner le rôle tenu par x dans* (B.4) *à la dimension qui présente les plus forts gradients car elle est plus difficile à interpoler.*

On a choisi dans notre cas de définir des sous-domaines autour des points de références $\underline{\mathrm{X}}_p$ répartis régulièrement. Ainsi, l'approximation $\breve{F}^{(p)}$ définie en $\underline{\mathrm{X}}_p$ n'est valable que sur Ω_p. On pourrait approximer la fonction dans tout l'espace par une combinaison linéaire des différentes fonctions $\breve{F}^{(p)}$ en définissant une interpolation. On note alors :

$$F_{RPM}(x,\mu) \approx \sum_{k=1}^{K} \breve{F}^{(k)}(x,\mu) \times I^{(k)}(x,\mu) \tag{B.5}$$

ou $I^{(k)}(x,\mu)$ est la fonction d'interpolation associée à $\underline{\mathrm{X}}_k$. Dans notre exemple, on a choisit $I^{(k)}(x,\mu) = 1$ sur Ω_k et 0 ailleurs. Il s'agit d'une interpolation de type Voronoï (ou Nearest-Neighbor (NN) en dimension supérieure à 2). On aurait pu choisir par exemple des fonctions EF de type P1 pour effectuer une interpolation linéaire. Cette interpolation aurait probablement un effet régularisant et minimiserait les oscillations visibles FIG.B.2.

La qualité de la solution complète peut être contrôlée en définissant une erreur exacte car on cherche à approcher un champ connu. On définit l'estimateur $\mathscr{E}_{ex}$:

$$\mathscr{E}_{ex} = \sqrt{\frac{\int_{\boldsymbol{I}_\mu} \int_\Omega (F_{RPM} - F)^2 dx d\mu}{\int_{\mathbf{I}_\mu} \int_\Omega F^2 dx d\mu}} \tag{B.6}$$

Dans (B.4), les valeurs des constantes λ_{kp} permettent de définir plusieurs méthodes différentes.

- Si $\lambda_{kp} = \delta_{kp}$, on retrouve la définition donnée en (B.3). FIG.B.2 illustre ce cas particulier sur une grille à 100 points de références.
- Si $\lambda_{kp} = 1$, on obtient la formulation qui sera utilisée par la suite. On la qualifie d'approximation par tranche car elle a la particularité d'être la même quelle que

soit la composante x_p pour un μ_p donné :

$$\breve{F}^{(p)}(x,\mu) = \frac{\sum_{k=1}^{N_x} \widehat{F}_k(\mu) F(x_k,\mu_p)}{\sum_{k=1}^{N_x} F(x_k,\mu_p)^2} \times \widehat{F}_p(x) \tag{B.7}$$

Le résultat est représenté FIG.B.3.

- D'autres valeurs de λ_{kp} peuvent être retenues. En particulier, il est intéressant de garder une valeur non nulle de λ_{kp} pour les coupes associées aux éléments voisins de Ω_p. Par exemple, 0.1% pour les voisins et 0 ailleurs donnait une valeur optimale dans l'exemple présenté par [Capaldo *et al.*, 2017]. On a de notre coté retenu les contributions des voisins sans les pondérer en choisissant $\lambda_{kp} = 1$ si $|p-k| \leq 1$ et $\lambda_{kp} = 0$ sinon.

	$\lambda_{kp} = \delta_{kp}$	$\lambda_{kp} = 1$	$\lambda_{kp} = 1$ si $\lvert p-k\rvert \leq 1$ et $\lambda_{kp} = 0$ sinon
Erreur	19%	13%	8%

TABLEAU B.1 : Approximation RPM sur 100 points de discrétisation, norme de l'erreur en fonction du choix de λ_{kp}

Physiquement, on peut interpréter la fonction de μ dans (B.4) (c'est à dire le quotient $\frac{\sum_{k=1}^{N_x} \widehat{F}_k(\mu) F(x_k,\mu_p).\lambda_{kp}^2}{\sum_{k=1}^{N_x} F(x_k,\mu_p)^2.\lambda_{kp}^2}$) comme une moyenne pondérée de l'influence de chacune des tranches de direction μ à x_p fixé. Ainsi, le choix $\lambda_{kp} = \delta_{kp}$ donne une solution particulièrement précise autour de $\underline{X}_p$. D'un autre coté, des λ_{kp} non nuls sur les éléments voisins de Ω_p permettent de lisser la fonction aux bords du sous-domaine et d'éviter en partie les oscillations observées FIG.B.2. Choisir la moyenne sur toutes les tranches ($\lambda_{kp} = 1$) peut sembler absurde et particulièrement grossier sur un maillage régulier, cette approximation donne d'ailleurs un résultat de mauvaise qualité dans [Capaldo *et al.*, 2017]. C'est cependant la seule méthode simple à utiliser sur un maillage irrégulier.

En observant les champs d'erreur, notamment FIG.B.2, on constate que celle-ci peut être particulièrement élevée très localement. Ces pics d'erreur apparaissent en particulier aux coins des sous-domaines Ω_p lorsque le point $\underline{X}_p$ est dans une zone à forts gradients dans les deux directions. L'aspect « régularisant » de l'influence des voisins a été testé simplement en choisissant $\lambda_{kp} = 1$ sur l'élément Ω_p et ses voisins directs. L'erreur finale est alors plus faible que dans les deux autres cas retenus comme présenté TAB.B.1.

L'erreur effectuée décroit avec le nombre de points retenus, mais à partir d'un certain seuil la convergence devient extrêmement lente et ce quel que soit le cas étudié

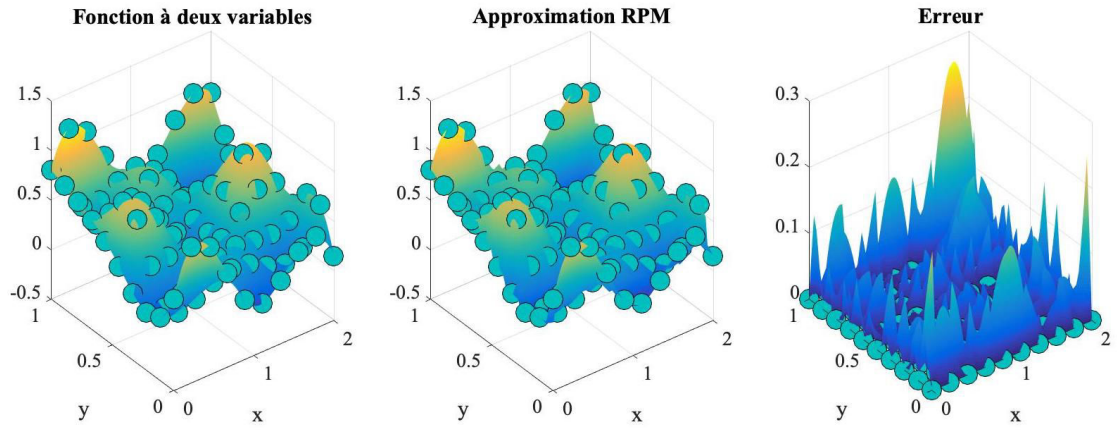


FIGURE B.2 : Approximation RPM d'une fonction sur une grille régulière à 100 points de référence avec $\lambda_{kp} = \delta_{kp}$; Norme finale de l'erreur = 19%

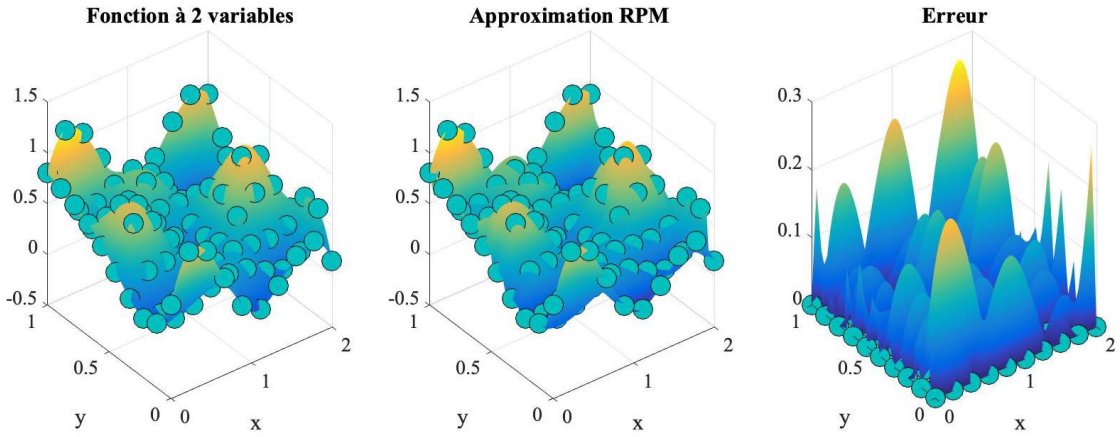


FIGURE B.3 : Approximation RPM d'une fonction sur une grille régulière à 100 points de référence avec $\lambda_{kp} = 1$; Norme finale de l'erreur = 13%

(voir FIG.B.4). On peut constater sur cette même figure que le choix des points est crucial lorsqu'on effectue une approximation grossière.

Remarque 41 (Grille non régulière) *Grâce à un estimateur d'erreur ($\mathscr{E}_\mu$ par exemple), on peut savoir où est-ce que l'approximation est la plus grossière. On pourrait construire itérativement des maillages plus adaptées au problème permettant d'améliorer significativement la précision de la RPM. Dans la partie suivante, on va choisir aléatoirement les points de référence. Bien que dans cet exemple bidimensionnel il soit possible de choisir des points pertinemment de manière à minimiser $\mathscr{E}_\mu$, nous ne nous sommes pas intéressé à de telles stratégies car elles ne sont pas aisément généralisables en grandes dimensions.*

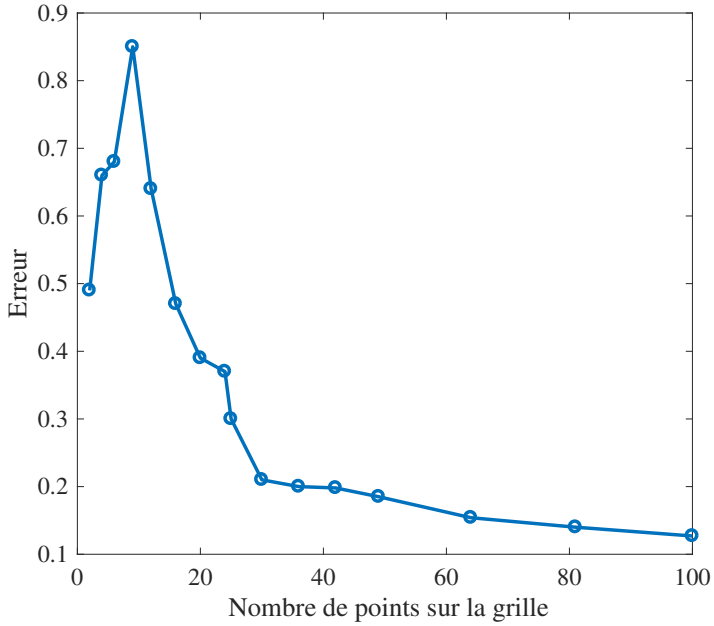


FIGURE B.4 : Approximation RPM d'une fonction avec des grilles régulières de différentes densités. $\lambda_{kp} = 1$ si $|p-k| \leq 1$ et $\lambda_{kp} = 0$ sinon

1.3 RPM sur une grille aléatoire : 2 paramètres

On choisit maintenant aléatoirement K points $\underline{X}_k$ sur le domaine $\Omega \times \boldsymbol{I}_\mu$. On ne peut maintenant plus définir de tranche pour un x_p ou un μ_p donné, on va donc conserver l'influence de tous les points dans la fonction $\breve{F}_p$. On définit :

$$\breve{F}_p(x,\mu) = \frac{\sum_{k=1}^{K} \widehat{F}_k(\mu) F(x_k,\mu_p).\lambda_{kp}^2}{\sum_{k=1}^{K} F(x_k,\mu_p)^2.\lambda_{kp}^2} \times \widehat{F}_p(x) \tag{B.8}$$

Il est maintenant beaucoup plus difficile de définir des voisinages, on se contentera donc de prendre soit $\lambda_{kp} = \delta_{kp}$, soit $\lambda_{kp} = 1$. Cette approximation est toujours exacte pour tout x à μ_p fixé. Par contre, il est indispensable d'utiliser une fonction d'interpolation comme dans (B.5) pour reconstruire le champ global. On a ici utilisé une interpolation constante par morceaux. Des sous-domaines rectangulaires sont construits suivant la direction x, chacun centré sur un point, puis redécoupés suivant la direction μ si plusieurs points de même abscisse sont présents. On peut interpréter cette méthode comme un algorithme de Voronoï « orthogonal », on détermine les zones *rectangulaires* les plus proches de chaque point. Cette interpolation n'est pas optimale mais à l'avantage d'être simple à implémenter. Le résultat est visible FIG.B.5

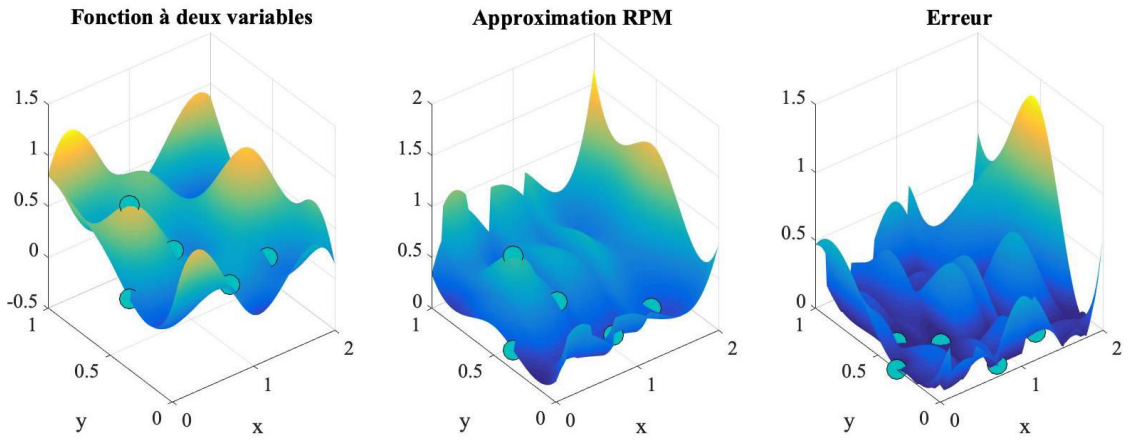


FIGURE B.5 : Approximation RPM d'une fonction avec une grille irrégulière à 6 points. $\lambda_{kp} = 1$, erreur = 61%, l'aspect de l'approximation « par tranche » est très visible.

Dans les conditions testées, cette approximation reste très mauvaise avec des erreurs de l'ordre de 50% sauf lorsque le nombre de points choisis tend vers le nombre de points de la grille fine retenue. Ces erreurs sont principalement concentrées aux bords des sous-domaines rectangulaires qui présentent de forts gradients.

C'est cette méthode qui est généralisée en partie suivante. Une meilleur technique d'interpolation (NN) y est utilisée et le choix des fonctions à interpoler est bien plus adapté à cette procédure. Ce sont les fonctions paramétriques de problèmes mécaniques tels que celui présenté FIG.1.1, elles sont bien plus régulières que les fonctions à deux variables étudiées dans cette partie.

2 RPM à N dimensions

2.1 Approximation

On se place maintenant dans un espace à 3 dimensions spatiales (domaine Ω) et N_p dimensions paramétriques (domaine $\boldsymbol{\Sigma}_\mu$). On cherche à approcher par exemple des champs de déplacement spatiaux $\boldsymbol{X} \in \mathscr{U}^{ad}(\Omega)$ discrétisés par $\underline{\mathbf{X}} \in \mathbb{R}^d$. On va construire l'interpolation RPM sur le domaine paramétrique entre des solutions spatiales particulières.

Soit $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}\}$, K jeux de paramètres choisis aléatoirement et $\underline{\mathbf{X}}(\boldsymbol{\mu}^{(k)})$ les solutions spatiales associées. On note $\boldsymbol{\mu}^{(k)} = [\mu_1^{(k)}, \ldots, \mu_{N_p}^{(k)}]$.

On définit la coupe de $\underline{\mathbf{X}}(\boldsymbol{\mu})$ au point $\boldsymbol{\mu}^{(k)}$ suivant la direction d par :

$$\underline{\hat{\mathbf{X}}}_{k,d}(\mu_d) = \underline{\mathbf{X}}\left(\mu_1^{(k)}, \ldots, \mu_{d-1}^{(k)}, \mu_d \ , \mu_{d+1}^{(k)}, \ldots, \mu_{N_p}^{(k)}\right) \tag{B.9}$$

Remarque 42 (Coupes et fibres) *On définit de la même manière les fibres d'un tenseur*

(voir partie I.2.2). Cependant, $\underline{\mathbf{X}}$ *n'est pas une fonction scalaire et les « fibres » définies en* (B.9) *sont à image dans* $\mathscr{U}^{ad}(\Omega)$.

Soit l'approximation partielle :

$$L_d^{(k)}\left(\mu_d,\overline{\boldsymbol{\mu}}_d\right)=\underbrace{\frac{\sum_{p=1}^{K}\left(\underline{\widehat{\mathbf{X}}}_{p,d}\left(\mu_d\right)\right)^T\underline{\underline{\mathbf{K}}}^0\underline{\widehat{\mathbf{X}}}_{p,d}(\mu_d^{(k)})}{\sum_{p=1}^{K}\left(\underline{\widehat{\mathbf{X}}}_{p,d}(\mu_d^{(k)})\right)^T\underline{\underline{\mathbf{K}}}^0\underline{\widehat{\mathbf{X}}}_{p,d}\left(\mu_d^{(k)}\right)}}_{\widetilde{L}_d^{(k)}(\mu_d)}\times\underline{\mathbf{X}}\left(\mu_d^{(k)},\overline{\boldsymbol{\mu}}_d\right) \tag{B.10}$$

$L_d^{(k)}$ est définie sur tout le domaine paramétrique, elle est exacte en $\boldsymbol{\mu}^{(k)}$ et précise au voisinage de ce point. On rappelle que la notation $\overline{\bullet}_d$ indique un vecteur de tout l'espace paramétrique sauf dans la dimension d. $\underline{\mathbf{X}}\left(\mu_d^{(k)},\overline{\boldsymbol{\mu}}_d\right)$ est toujours de grande dimension, c'est une fonction de tous les paramètres sauf 1 (à l'opposé d'une coupe). Cette définition est le pendant de B.8 avec $\lambda_{kp}=1$ pour une fonction à image dans $\mathscr{U}^{ad}(\Omega)$.

On peut ensuite approximer de la même manière $\underline{\mathbf{X}}\left(\mu_d^{(k)},\overline{\boldsymbol{\mu}}_d\right)$. On obtient par récurrence l'approximation :

$$\underline{\breve{\mathbf{X}}}^{(k)}(\boldsymbol{\mu})=\left[\prod_{d=1}^{N_p}\frac{\sum_{p=1}^{K}\left(\underline{\widehat{\mathbf{X}}}_{p,d}(\mu_d)\right)^T\underline{\underline{\mathbf{K}}}^0\underline{\widehat{\mathbf{X}}}_{p,d}\left(\mu_d^{(d)}\right)}{\sum_{p=1}^{K}\left(\underline{\widehat{\mathbf{X}}}_{p,d}\left(\mu_d^{(k)}\right)\right)^T\underline{\underline{\mathbf{K}}}^0\underline{\widehat{\mathbf{X}}}_{p,d}\left(\mu_d^{(k)}\right)}\right]\times\underline{\mathbf{X}}\left(\boldsymbol{\mu}^{(k)}\right)=\prod_{d=1}^{N_p}\widetilde{L}_d^{(k)}(\mu_d)\times\underline{\mathbf{X}}\left(\boldsymbol{\mu}^{(k)}\right) \tag{B.11}$$

Cette fonction approxime $\underline{\mathbf{X}}$ par une formulation à variable séparée. Elle est théoriquement valide uniquement dans un voisinage de $\boldsymbol{\mu}^{(k)}$. En particulier, $\underline{\mathbf{X}}\left(\boldsymbol{\mu}^{(k)}\right)=\underline{\breve{\mathbf{X}}}^{(k)}\left(\boldsymbol{\mu}^{(k)}\right)$. Afin de construire la solution dans tout l'espace paramétrique, on interpole ces différentes solutions pour obtenir une solution complète :

$$\underline{\mathbf{X}}(\boldsymbol{\mu})\approx\sum_{k=1}^{K}\underline{\breve{\mathbf{X}}}^{(k)}(\boldsymbol{\mu})I^{(k)}(\boldsymbol{\mu}) \tag{B.12}$$

2.2 Implémentation

Cette approximation ne dépend pas du problème traité ni de sa physique. L'aspect localisé des paramètres, à la base de la PM-PGD, n'est pas pris en compte ici. On a retenu le même problème modèle que dans le reste du manuscrit (voir FIG.1.1) et les résolutions spatiales sont effectuées grâce à un solveur EF utilisé en « boîte noire ». L'algorithme se décompose en trois étapes.

1 - Choix des points de référence On peut sélectionner les points de références suivant plusieurs méthodes qui donnent des résultats significativement différents. Dans le premier exemple présenté FIG.B.6, les points sont tirés aléatoirement sur la grille de discrétisation des fonctions paramétriques. On constate que c'est en choisissant les points de références aléatoirement sur une grille plus grossière que la grille micro qu'on obtient les tirages les plus pertinents, c'est ce qui est fait pour le cas à 120 paramètres présenté FIG.B.7.

2 - Calcul des coupes et des approximations Cette partie peut être extrêmement couteuse. Il faut en effet calculer l'ensemble des $\underline{\widehat{\mathbf{X}}}_{k,d}(\mu_d)$. Chaque point de discrétisation de l'espace paramétrique nécessite un calcul EF complet, assemblage de la matrice inclus car le solveur est utilisé en « boîte noire ». Une stratégie plus intrusive permettrait de gagner en temps de calcul, par exemple en générant tous les points d'une coupe sans reprendre l'opération d'assemblage à zéro. Il faut ensuite calculer les $\widetilde{L}_d^{(k)}(\mu_d)$ qui sont des combinaisons des projections précédentes.

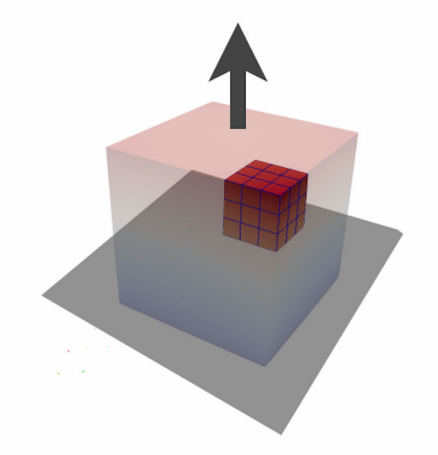
3 - Interpolation Pour reconstruire la solution, il faudrait théoriquement bâtir les fonctions d'interpolation $I^{(k)}$. En pratique, ce n'est pas utile car le champ complet n'est pas une donnée intéressante en lui-même dans un contexte de réduction de modèle. Il est de toute façon inaccessible pour les grands nombres de paramètres du fait de la malédiction de la dimensionnalité. Il faut par contre être capable de connaitre sa valeur en certains points en temps réel. Il suffit donc d'interpoler le champs $\underline{\mathbf{X}}$ pour les jeux de paramètres d'intérêt, ce qui est très facile avec une interpolation de type Nearest-Neighbor (NN). Cette méthode consiste à calculer la distance dans l'espace paramétrique entre le nouveau point d'intérêt $\boldsymbol{\mu}^{(i)}$ et tous les points $\boldsymbol{\mu}^{(k)}$ étudiés. On ne retient alors que l'approximation liée au point $\boldsymbol{\mu}^{(P)}$ le plus proche de $\boldsymbol{\mu}^{(i)}$ qui est donnée par : $\underline{\mathbf{X}}\left(\boldsymbol{\mu}^{(i)}\right) \approx \underline{\widetilde{\mathbf{X}}}^{(P)}\left(\boldsymbol{\mu}^{(i)}\right)$.

Remarque 43 (Méthode d'interpolation) *L'interpolation de type NN est la généralisation des algorithmes de Voronoï. Elle est très imprécise en grandes dimensions, comme le prouve les résultats qui vont suivre. Cependant, la plupart des autres méthodes d'interpolation envisagées nécessitent la construction de fonctions d'interpolation. C'est une étape délicate pour les grands nombres de paramètres car il est impossible de construire explicitement une grille ou un maillage de dimension N_p.*

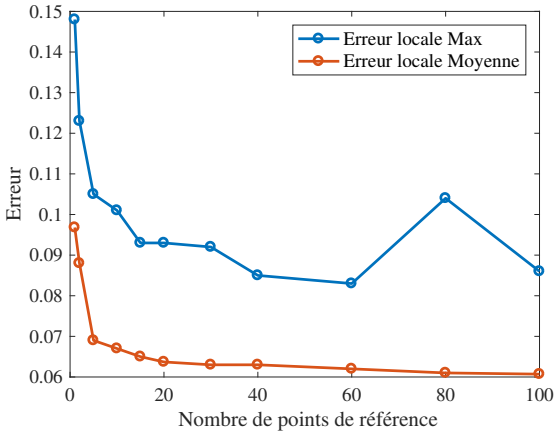
Erreur On estime l'erreur effectuée via cette méthode grâce à un estimateur local. On calcule pour $I_{part} = 100$ tirages paramétriques différents la solution exacte et la solution réduite, puis son erreur en norme énergétique. On définit alors deux estimateurs : $\mathscr{E}_{loc}^{Avr}$ en calculant la moyenne des erreurs calculées et $\mathscr{E}_{loc}^{Max}$ en en conservant le maximum.

$$\mathscr{E}_{loc}^{Max} = \max_{i \in I_{part}} \frac{\left\| \underline{\mathrm{X}}_{exact}^{(i)} - \underline{\mathrm{X}}_{RPM}^{(i)} \right\|}{\left\| \underline{\mathrm{X}}_{exact}^{(i)} \right\|} \qquad \mathscr{E}_{loc}^{Avr} = \left\langle \frac{\left\| \underline{\mathrm{X}}_{exact}^{(i)} - \underline{\mathrm{X}}_{RPM}^{(i)} \right\|}{\left\| \underline{\mathrm{X}}_{exact}^{(i)} \right\|} \right\rangle_{i \in I_{part}} \tag{B.13}$$

Les normes retenues sont les semi-normes énergétiques purement spatiales définies en I.2.3.

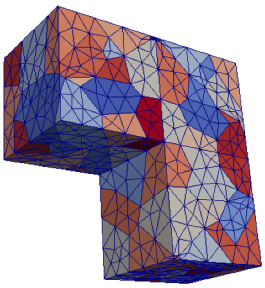


(a) Problème modèle : mise en évidence d'un sous-domaine



(b) Erreurs locales en fonction de nombre K de points de références

FIGURE B.6 : Problèmes réguliers à 64 paramètres, 50% de variation paramétrique, problème modèle décrit précisément en (1.3)



(a) Problème modèle : maillage et répartition paramétrique

Nombre de points de référence	Erreur locale $\mathscr{E}_{loc}^{Max}$	Erreur locale $\mathscr{E}_{loc}^{Avr}$
20	14.2%	7.9%
50	13.7%	7.1%
100	11.8%	7.2%

(b) Erreurs locales en fonction du nombre K de points de références

FIGURE B.7 : Problèmes à 120 paramètres, 50% de variation paramétrique, problème modèle présenté FIG.3.2

On observe FIG.B.6 et FIG.B.7 la décroissance de l'erreur lorsqu'on augmente le nombre de points de référence. On constate qu'on atteint très vite un seuil, après une

vingtaine de points environ et que tout enrichissement ultérieur a très peu d'influence. On a volontairement laissé FIG.B.6b le point singulier de l'erreur $\mathscr{E}_{loc}^{Max}$ pour 80 points de référence. Il ne s'agit que d'une conséquence du choix aléatoire de l'ensemble I_{part} de points du domaine paramétrique. Ce tirage contenait un point $\boldsymbol{\mu}^{(i)}$ particulièrement mal décrit par l'interpolation RPM. On constate cependant grâce à l'estimateur moyen que le comportement global de l'algorithme reste le même.

2.3 Limites de la méthode

La stagnation des erreurs quand on ajoute plus de points de référence est une limite majeure de cette version de la RPM.

Les distances retenues dans l'espace paramétrique qui permettent de réaliser l'interpolation NN sont simplement les distances euclidiennes dans l'espace $\boldsymbol{\Sigma}_\mu$ à N_p dimensions. Or, on constate qu'elles sont sensiblement les mêmes entre tous les points choisis aléatoirement, comme illustré FIG.B.8 pour un problème à 64 paramètres. Il n'y a ni point très proche, ni point très distant lorsqu'on tire des points aléatoirement, le choix de l'interpolation de type NN est donc très discutable. Est-il possible de surmonter ce problème en choisissant une autre définition de la distance ? En sélectionnant une autre méthode d'interpolation différente et compatible avec les grandes dimensions ? Ces questions n'ont pas été traitées dans le cadre de cette thèse.

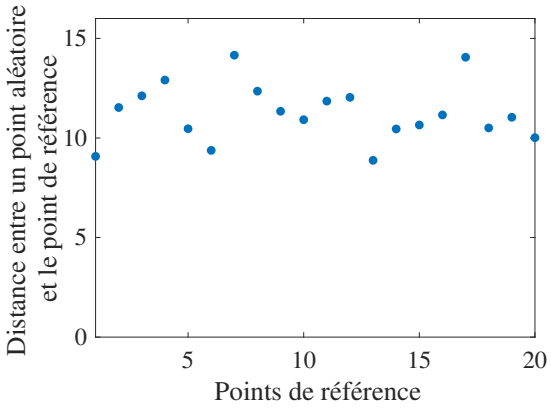


FIGURE B.8 : Problème à 64 paramètres : ensemble des distances euclidiennes entre un point tiré aléatoirement dans l'espace paramétrique et les 20 points de références utilisés pour construire une interpolation.

Pour des problèmes à plusieurs centaines de paramètres, le nombre de résolutions à effectuer pour le calcul des coupes devient vite très élevé. La Figure B.9 montre le nombre de ces calculs dans différents cas. On peut remarquer par exemple qu'un problème à 300 ou 1000 paramètres reste traitable avec une PM-PGD composée d'une seule itération, mais qu'il est très coûteux pour toutes les méthodes RPM ou pour la deuxième itération de la PM-PGD.

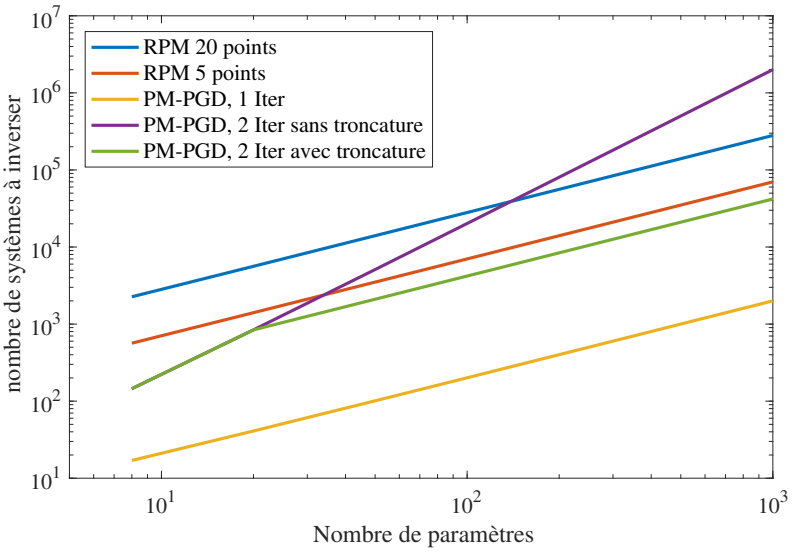


FIGURE B.9 : Nombre de systèmes à inverser pour des problèmes de tailles croissantes selon les méthodes utilisées.

Ce graphique ne prend pas en compte le fait que plus le problème est grand, plus chaque calcul est long et complexe et le temps de calcul n'augmente pas linéairement avec le nombre de systèmes matriciels à résoudre. Il apparait clairement que la PM-PGD est globalement une méthode moins coûteuse. De plus, les taux d'erreur que l'on atteint avec la PM-PGD présentés FIG.4.3 par exemple sont bien meilleurs.

Une méthode tensorielle réalise aussi une approximation de la même forme que la RPM sous le nom de cross-approximation [Espig *et al.*, 2009]. Cette approximation est composée de deux types de données :

— Des tenseurs de rang 1 définis par l'ensembles des fibres passant par une certaine composante (appelée *pivot index*). Il s'agit exactement des approximations (B.4) avec $\lambda_{kp} = \delta_{kp}$.

— Des fonctions d'interpolations permettant de définir en chaque point du tenseur approché quel poids associer aux tenseurs de rang 1 stockés.

La méthode de construction des cross-approximations est un algorithme greedy qui détermine itérativement les *pivot indexes* grâce à une procédure proche de celle utilisée par les RB par exemple (voir I.3.2). Ce choix optimal des points de référence (voir REM.41) n'est malheureusement pas généralisable simplement aux grandes dimensions [Aguado *et al.*, 2018].

Bibliographie

[Aguado *et al.*, 2018] AGUADO, J. V., BORZACCHIELLO, D., KOLLEPARA, K., CHINESTA, F. et HUERTA, A. (2018). Tensor representation of non-linear models using cross approximations. *Journal of Scientific Computing*.

[Ammar *et al.*, 2007] AMMAR, A., MOKDAD, B., CHINESTA, F. et KEUNINGS, R. (2007). A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids : Part ii : Transient simulation using space-time separated representations. *Journal of Non-Newtonian Fluid Mechanics*, 144(2-3):98–121.

[Arnold *et al.*, 2002] ARNOLD, D., BREZZI, F., COCKBURN, B. et MARINI, L. (2002). Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779.

[Arnold *et al.*, 2000] ARNOLD, D. N., BREZZI, F., COCKBURN, B. et MARINI, D. (2000). *Discontinuous Galerkin Methods for Elliptic Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Avril *et al.*, 2008] AVRIL, S., BONNET, M., BRETELLE, A.-S., GRÉDIAC, M., HILD, F., IENNY, P., LATOURTE, F., LEMOSSE, D., PAGANO, S., PAGNACCO, E. *et al.* (2008). Overview of identification methods of mechanical parameters based on full-field measurements. *Experimental Mechanics*, 48(4):381.

[Babuška et Guo, 1992] BABUŠKA, I. et GUO, B. (1992). The h, p and hp version of the finite element method ; basis theory and applications. *Advances in Engineering Software*, 15(3-4):159–174.

[Bachmayr *et al.*, 2017] BACHMAYR, M., COHEN, A., DUNG, D. et SCHWAB, C. (2017). Fully discrete approximation of parametric and stochastic elliptic pdes. *SIAM Journal on Numerical Analysis*, 55(5):2151–2186.

[Bader *et al.*, 2015] BADER, B. W., KOLDA, T. G. *et al.* (2015). Matlab tensor toolbox version 2.6. Available online.

[Banerjee *et al.*, 2013] BANERJEE, B., WALSH, T. F., AQUINO, W. et BONNET, M. (2013). Large scale parameter estimation problems in frequency-domain elastodynamics using an error in constitutive equation functional. *Computer methods in applied mechanics and engineering*, 253:60–72.

[Barbarella *et al.*, 2016] BARBARELLA, E., ALLIX, O., DAGHIA, F., LAMON, J. et JOLLIVET, T. (2016). A new inverse approach for the localization and characterization of defects based on compressive experiments. *Computational Mechanics*, 57(6):1061–1074.

[Barrault *et al.*, 2004] BARRAULT, M., MADAY, Y., NGUYEN, N. C. et PATERA, A. T. (2004). An 'empirical interpolation' method : application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672.

[Bassi et Rebay, 1997] BASSI, F. et REBAY, S. (1997). A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier–stokes equations. *Journal of computational physics*, 131(2):267–279.

[Bocciarelli *et al.*, 2014] BOCCIARELLI, M., BULJAK, V., MOY, C., RINGER, S. et RANZI, G. (2014). An inverse analysis approach based on a pod direct model for the mechanical characterization of metallic materials. *Computational Materials Science*, 95:302–308.

[Boisse *et al.*, 1990] BOISSE, P., BUSSY, P. et LADEVEZE, P. (1990). A new approach in non-linear mechanics : The large time increment method. *International journal for numerical methods in engineering*, 29(3):647–663.

[Bonnet et Aquino, 2015] BONNET, M. et AQUINO, W. (2015). Three-dimensional transient elastodynamic inversion using an error in constitutive relation functional. *Inverse Problems*, 31(3):035010.

[Bonnet et Frangi, 2006] BONNET, M. et FRANGI, A. (2006). *Analyse des solides déformables par la méthode des éléments finis*. Editions de l'Ecole Polytechnique.

[Bordeu, 2013] BORDEU, F. (2013). Pxdmf : A file format for separated variables problems, version 1.6. Available online.

[Bouclier *et al.*, 2013] BOUCLIER, R., LOUF, F. et CHAMOIN, L. (2013). Real-time validation of mechanical models coupling pgd and constitutive relation error. *Computational Mechanics*, 52(4):861–883.

[Bui-Thanh *et al.*, 2008] BUI-THANH, T., WILLCOX, K. et GHATTAS, O. (2008). Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288.

[Cagniart *et al.*, 2019] CAGNIART, N., MADAY, Y. et STAMM, B. (2019). Model order reduction for problems with large convection effects. *In Contributions to Partial Differential Equations and Applications*, pages 131–150. Springer.

[Cancès, E. *et al.*, 2013] CANCÈS, E., EHRLACHER, V. et LELIÈVRE, T. (2013). Greedy algorithms for high-dimensional non-symmetric linear problems. *ESAIM : Proc.*, 41:95–131.

[Capaldo *et al.*, 2017] CAPALDO, M., GUIDAULT, P.-A., NÉRON, D. et LADEVÈZE, P. (2017). The reference point method, a hyperreduction technique : Application to pgd-based nonlinear model reduction. *Computer Methods in Applied Mechanics and Engineering*, 322:483–514.

[Carlberg *et al.*, 2013] CARLBERG, K., FARHAT, C., CORTIAL, J. et AMSALLEM, D. (2013). The gnat method for nonlinear model reduction : effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647.

[Carroll et Chang, 1970] CARROLL, J. D. et CHANG, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319.

[Cattell, 1944] CATTELL, R. B. (1944). “parallel proportional profiles” and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283.

[Chamoin *et al.*, 2016] CHAMOIN, L., ALLIER, P.-E. et MARCHAND, B. (2016). Synergies between the constitutive relation error concept and pgd model reduction for simplified v&v procedures. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1):18.

[Chatterjee, 2000] CHATTERJEE, A. (2000). An introduction to the proper orthogonal decomposition. *Current Science*, 78(7):808–817.

[Chinesta *et al.*, 2010] CHINESTA, F., AMMAR, A. et CUETO, E. (2010). Proper generalized decomposition of multiscale models. *International Journal for Numerical Methods in Engineering*, 83(8-9):1114–1132.

[Chinesta *et al.*, 2011] CHINESTA, F., AMMAR, A., LEYGUE, A. et KEUNINGS, R. (2011). An overview of the proper generalized decomposition with applications in computational rheology. *Journal of Non-Newtonian Fluid Mechanics*, 166(11):578–592.

[Chinesta *et al.*, 2013a] CHINESTA, F., KEUNINGS, R. et LEYGUE, A. (2013a). *The Proper Generalized Decomposition for Advanced Numerical Simulations*.

[Chinesta *et al.*, 2013b] CHINESTA, F., LEYGUE, A., BORDEU, F., CUETO, E., GONZALEZ, D., AMMAR, A. et HUERTA, A. (2013b). PGD-Based Computational Vademecum for Efficient Design, Optimization and Control. *Archives of Computational Methods in Engineering*, 20(1):31 – 59.

[Claire *et al.*, 2004] CLAIRE, D., HILD, F. et ROUX, S. (2004). A finite element formulation to identify damage fields : the equilibrium gap method. *International journal for numerical methods in engineering*, 61(2):189–208.

[Cooreman *et al.*, 2008] COOREMAN, S., LECOMPTE, D., SOL, H., VANTOMME, J. et DEBRUYNE, D. (2008). Identification of mechanical material behavior through inverse modeling and dic. *Experimental Mechanics*, 48(4):421–433.

[Courard *et al.*, 2016] COURARD, A., LADEVÈZE, P., NÉRON, D. et BALLÈRE, L. (2016). Integration of PGD-virtual charts into an engineering design process. *Computational Mechanics*, 57(4):637 – 651.

[Cremonesi *et al.*, 2013] CREMONESI, M., NÉRON, D., GUIDAULT, P.-A. et LADEVÈZE, P. (2013). A pgd-based homogenization technique for the resolution of nonlinear multiscale problems. *Computer Methods in Applied Mechanics and Engineering*, 267:275–292.

[De Lathauwer *et al.*, 2000] DE LATHAUWER, L., DE MOOR, B. et VANDEWALLE, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

[de Saint-Venant, 1856] de SAINT-VENANT, M. (1856). *Mémoire sur la torsion des prismes : Avec des considérations sur leur flexion ainsi que sur l'équilibre intérieur des solides élastiques en général : Et des formules pratiques pour le calcul de leur résistance à divers efforts s' exerçant simultanément*. Imprimerie nationale.

[Dhia et Rateau, 2005] DHIA, H. B. et RATEAU, G. (2005). The arlequin method as a flexible engineering design tool. *International journal for numerical methods in engineering*, 62(11):1442–1462.

[Dureisseix *et al.*, 2003] DUREISSEIX, D., LADEVÈZE, P. et SCHREFLER, B. A. (2003). A latin computational strategy for multiphysics problems : application to poroelasticity. *International Journal for Numerical Methods in Engineering*, 56(10):1489–1510.

[Espig *et al.*, 2009] ESPIG, M., GRASEDYCK, L. et HACKBUSCH, W. (2009). Black box low tensor-rank approximation using fiber-crosses. *Constructive approximation*, 30(3): 557.

[Eymard *et al.*, 2000] EYMARD, R., GALLOUËT, T. et HERBIN, R. (2000). Finite volume methods. *Handbook of numerical analysis*, 7:713–1018.

[Faber *et al.*, 2003] FABER, N. K. M., BRO, R. et HOPKE, P. K. (2003). Recent developments in candecomp/parafac algorithms : a critical review. *Chemometrics and Intelligent Laboratory Systems*, 65(1):119–137.

[Giacoma *et al.*, 2015] GIACOMA, A., DUREISSEIX, D., GRAVOUIL, A. et ROCHETTE, M. (2015). Toward an optimal a priori reduced basis strategy for frictional contact problems with latin solver. *Computer Methods in Applied Mechanics and Engineering*, 283:1357–1381.

[Grasedyck, 2010] GRASEDYCK, L. (2010). Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054.

[Grédiac, 1989] GRÉDIAC, M. (1989). Principe des travaux virtuels et identification. *Comptes rendus de l'Académie des sciences. Série 2, Mécanique, Physique, Chimie, Sciences de l'univers, Sciences de la Terre*, 309(1):1–5.

[Hackbusch, 2012] HACKBUSCH, W. (2012). *Tensor spaces and numerical tensor calculus*, volume 42. Springer Science & Business Media.

[Hackbusch et Kühn, 2009] HACKBUSCH, W. et KÜHN, S. (2009). A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15(5):706–722.

[Hadamard, 1932] HADAMARD, J. (1932). *Le probleme de Cauchy et les équations aux dérivées partielles linéaires hyperboliques*. Hermann.

[Harshman, 1970] HARSHMAN, R. A. (1970). Foundations of the parafac procedure : Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

[Hitchcock, 1927] HITCHCOCK, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

[Hochard *et al.*, 1997] HOCHARD, C., LADEVÈZE, P. et PROSLIER, L. (1997). A trefftz simplified method for the computation of elastic structures. *Computer Assisted Mechanics and Engineering Sciences*, 4:383–396.

[Holtz *et al.*, 2012] HOLTZ, S., ROHWEDDER, T. et SCHNEIDER, R. (2012). On manifolds of tensors of fixed TT-rank. *Numerische Mathematik*, 120(4):701–731.

[Jolliffe, 2011] JOLLIFFE, I. (2011). Principal component analysis. *In International encyclopedia of statistical science*, pages 1094–1096. Springer.

[Kaipio et Somersalo, 2006] KAIPIO, J. et SOMERSALO, E. (2006). *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media.

[Karhunen, 1947] KARHUNEN, K. (1947). Under lineare methoden in der wahr scheinlichkeitsrechnung. *Annales Academiae Scientiarun Fennicae Series A1 : Mathematia Physica*, 47.

[Kerschen *et al.*, 2005] KERSCHEN, G., GOLINVAL, J.-c., VAKAKIS, A. F. et BERGMAN, L. A. (2005). The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems : an overview. *Nonlinear dynamics*, 41(1-3):147–169.

[Kiers, 2000] KIERS, H. A. (2000). Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 14(3):105–122.

[Kita et Kamiya, 1995] KITA, E. et KAMIYA, N. (1995). Trefftz method : an overview. *Advances in Engineering Software*, 24(1-3):3–12.

[Kolda et Bader, 2009] KOLDA, T. G. et BADER, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[Kolda *et al.*, 2005] KOLDA, T. G., BADER, B. W. et KENNY, J. P. (2005). Higher-order web link analysis using multilinear algebra. *In Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE.

[Ladevèze, 1985] LADEVÈZE, P. (1985). On Algorithm Family in Structural Mechanics. *Comptes rendus des séances de l'Academie des sciences. Série 2*, 300(2).

[Ladevèze, 1989] LADEVÈZE, P. (1989). The large time increment method for the analyse of structures with nonlinear constitutive relation described by internal variables (in french). *Comptes rendus des séances de l'Academie des sciences. Série 2*, 309(2):1095–1099.

[Ladevèze, 1996] LADEVÈZE, P. (1996). Mécanique non linéaire des structures. *Hermès*.

[Ladevèze, 1997] LADEVÈZE, P. (1997). A computational technique for the integrals over the time-space domain in connection with the latin method (in french). Rapport technique, Tech. Rep. 193, LMT-Cachan.

[Ladevèze, 1999] LADEVÈZE, P. (1999). *Nonlinear computational structural mechanics : new approaches and non-incremental methods of calculation*. Springer-Verlag New York.

[Ladevèze, 2011] LADEVÈZE, P. (2011). New variational formulations for discontinuous approximations (in french). Rapport technique, LMT Cachan.

[Ladevèze, 2014] LADEVÈZE, P. (2014). New methods with conditioner for PGD reduced order models (in french). Rapport technique, LMT Cachan.

[Ladevèze, 2014] LADEVÈZE, P. (2014). *Separated Representations and PGD-Based Model Reduction : Fundamentals and Applications*, chapitre PGD in linear and nonlinear Computational Solid Mechanics, pages 91–152. Springer Vienna, Vienna.

[Ladevèze, 2016] LADEVÈZE, P. (2016). On reduced models in nonlinear solid mechanics. *European Journal of Mechanics, A/Solids*, 60:227–237.

[Ladevèze et Chamoin, 2016] LADEVÈZE, P. et CHAMOIN, L. (2016). The constitutive relation error method : A general verification tool. *In Verifying Calculations-Forty Years On*, pages 59–94. Springer.

[Ladevèze et Dureisseix, 2000] LADEVÈZE, P. et DUREISSEIX, D. (2000). A micro/macro approach for parallel computing of heterogeneous structures. *International Journal for Computational Civil and Structural Engineering*, 1:18–28.

[Ladevèze *et al.*, 1994] LADEVÈZE, P., NEDJAR, D. et REYNIER, M. (1994). Updating of finite element models using vibration tests. *AIAA journal*, 32(7):1485–1491.

[Ladeveze et Nouy, 2003] LADEVEZE, P. et NOUY, A. (2003). On a multiscale computational strategy with time and space homogenization for structural mechanics. *Computer Methods in Applied Mechanics and Engineering*, 192(28-30):3061–3087.

[Ladevèze *et al.*, 2018] LADEVÈZE, P., PAILLET, C. et NÉRON, D. (2018). Extended-PGD model reduction for nonlinear solid mechanics problems involving many parameters. *Computational Methods in Applied Sciences*, 46:201–220.

[Ladevèze et Riou, 2014] LADEVÈZE, P. et RIOU, H. (2014). On Trefftz and weak Trefftz discontinuous Galerkin approaches for medium-frequency acoustics. *Computer Methods in Applied Mechanics and Engineering*, 278:729–743.

[Lamari *et al.*, 2010] LAMARI, H., AMMAR, A., CARTRAUD, P., LEGRAIN, G., CHINESTA, F. et JACQUEMIN, F. (2010). Routes for efficient computational homogenization of nonlinear materials using the proper generalized decompositions. *Archives of Computational methods in Engineering*, 17(4):373–391.

[Liang *et al.*, 2002] LIANG, Y., LEE, H., LIM, S., LIN, W., LEE, K. et WU, C. (2002). Proper orthogonal decomposition and its applicationspart i : Theory. *Journal of Sound and Vibration*, 252(3):527 – 544.

[Loève, 1955] LOÈVE, M. (1955). *Probability theory : foundations, random sequences*. van Nostrand Princeton, NJ.

[Lumley, 1967] LUMLEY, J. L. (1967). The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*.

[Maday et Ronquist, 2004] MADAY, Y. et RONQUIST, E. M. (2004). The reduced basis element method : application to a thermal fin problem. *SIAM Journal on Scientific Computing*, 26(1):240–258.

[Marchand *et al.*, 2016] MARCHAND, B., CHAMOIN, L. et REY, C. (2016). Real-time updating of structural mechanics models using kalman filtering, modified constitutive relation error, and proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, 107(9):786–810.

[Michoski *et al.*, 2017] MICHOSKI, C., ALEXANDERIAN, A., PAILLET, C., KUBATKO, E. J. et DAWSON, C. (2017). Stability of nonlinear convection–diffusion–reaction systems in discontinuous galerkin methods. *Journal of Scientific Computing*, 70(2):516–550.

[Modesto *et al.*, 2015] MODESTO, D., ZLOTNIK, S. et HUERTA, A. (2015). Proper generalized decomposition for parameterized helmholtz problems in heterogeneous and unbounded domains : Application to harbor agitation. *Computer Methods in Applied Mechanics and Engineering*, 295:127–149.

[Moës *et al.*, 1999] MOËS, N., DOLBOW, J. et BELYTSCHKO, T. (1999). A finite element method for crack growth without remeshing. *International journal for numerical methods in engineering*, 46(1):131–150.

[Nachar *et al.*, 2019] NACHAR, S., BOUCARD, P.-A., NERON, D. et BORDEU, F. (2019). Coupling multi-fidelity kriging & model-order reduction for the construction of virtual charts. *Computational Mechanics*.

[Nazeer *et al.*, 2014] NAZEER, S. M., BORDEU, F., LEYGUE, A. et CHINESTA, F. (2014). Arlequin based pgd domain decomposition. *Computational Mechanics*, 54(5):1175–1190.

[Néron *et al.*, 2015] NÉRON, D., BOUCARD, P. A. et RELUN, N. (2015). Time-space PGD for the rapid solution of 3D nonlinear parametrized problems in the many-query context. *International Journal for Numerical Methods in Engineering*, 103(4):275—-292.

[Niroomandi *et al.*,] NIROOMANDI, S., GONZALEZ, D., ALFARO, I., BORDEU, F., LEYGUE, A., CUETO, E. et CHINESTA, F. Real-time simulation of biological soft tissues : a pgd approach. *International Journal for Numerical Methods in Biomedical Engineering*, 29(5):586–600.

[Nouy, 2007] NOUY, A. (2007). A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 196(45-48):4521–4537.

[Nouy, 2010] NOUY, A. (2010). A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1603–1626.

[Nouy, 2017] NOUY, A. (2017). *Low-Rank Tensor Methods for Model Order Reduction*, pages 857–882. Springer International Publishing, Cham.

[Nouy et Ladevèze, 2004] NOUY, A. et LADEVÈZE, P. (2004). Multiscale computational strategy with time and space homogenization : a radial-type approximation technique for solving microproblems. *International Journal for Multiscale Computational Engineering*, 2(4).

[Oseledets *et al.*, 2012] OSELEDETS, I., DOLGOV, S., KAZEEV, V., LEBEDEVA, O. et MACH, T. (2012). Tt-toolbox 2.2. *available online, URL : http ://spring. inm. ras. ru/osel*.

[Oseledets, 2011] OSELEDETS, I. V. (2011). Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

[Pagnacco *et al.*, 2013] PAGNACCO, E., CARO-BRETELLE, A.-S. et IENNY, P. (2013). Parameter identification from mechanical field measurements using finite element model updating strategies. *Full-Field Measurements and Identification in Solid Mechanics*, pages 247–274.

[Paillet, 2016a] PAILLET, C. (2016a). Mémoire de Master; BigPGD : Nouveaux algorithmes de réduction de modèles pour le traitement des problèmes à très grand nombre de paramètres. Rapport technique, LMT, Cachan.

[Paillet, 2016b] PAILLET, C. (2016b). Rapport Bibliographique; BigPGD : Nouveaux algorithmes de réduction de modèles pour le traitement des problèmes à très grand nombre de paramètres. Rapport technique, LMT, Cachan.

[Paillet *et al.*, 2019] PAILLET, C., LADEVÈZE, P. et NÉRON, D. (2019). High-dimensional multiparametric reduced order models compatible with finite element solvers. *in preparation*.

[Paillet *et al.*, 2018] PAILLET, C., NÉRON, D. et LADEVÈZE, P. (2018). A door to model reduction in high-dimensional parameter space. *Comptes Rendus Mécanique*, 346(7):524 – 531. Model reduction, data-based and advanced discretization in computational mechanics.

[Passieux *et al.*, 2015] PASSIEUX, J.-C., BUGARIN, F., DAVID, C., PÉRIÉ, J.-N. et ROBERT, L. (2015). Multiscale displacement field measurement using digital image correlation : Application to the identification of elastic properties. *Experimental Mechanics*, 55(1):121–137.

[Patera *et al.*, 2007] PATERA, A. T., ROZZA, G. *et al.* (2007). Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations.

[Pineda-Sanchez *et al.*, 2010] PINEDA-SANCHEZ, M., CHINESTA, F., ROGER-FOLCH, J., RIERA-GUASP, M., PÉREZ-CRUZ, J. et DAIM, F. (2010). Simulation of skin effect via separated representations. *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering*, 29(4):919–929.

[Pruliere *et al.*, 2010] PRULIERE, E., CHINESTA, F. et AMMAR, A. (2010). On the deterministic solution of multidimensional parametric models using the proper generalized decomposition. *Mathematics and Computers in Simulation*, 81(4):791–810.

[Quey *et al.*, 2011] QUEY, R., DAWSON, P. et BARBE, F. (2011). Large-scale 3d random polycrystals for the finite element method : Generation, meshing and remeshing. *Computer Methods in Applied Mechanics and Engineering*, 200(17-20):1729–1745.

[Reed et Hill, 1973] REED, W. H. et HILL, T. (1973). Triangular mesh methods for the neutron transport equation. Rapport technique, Los Alamos Scientific Lab., N. Mex.(USA).

[Rey, 1998] REY, C. (1998). Reuse of krylov spaces in the solution of large-scale nonlinear elasticity problems.

[Rivière, 2008] RIVIÈRE, B. (2008). *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations*, chapitre Linear Elasticity, pages 109–115.

[Rivière *et al.*, 1999] RIVIÈRE, B., WHEELER, M. F. et GIRAULT, V. (1999). Improved energy estimates for interior penalty, constrained and discontinuous galerkin methods for elliptic problems. part i. *Computational Geosciences*, 3(3):337–360.

[Roux et Hild, 2019] ROUX, S. G. et HILD, F. (2019). Optimal procedure for the identication of constitutive parameters from experimentally measured displacement fields. *International Journal of Solids and Structures*.

[Rozza, 2014] ROZZA, G. (2014). *Separated Representations and PGD-Based Model Reduction : Fundamentals and Applications*, chapitre Fundamentals of reduced basis method for problems governed by parametrized PDEs and applications, pages 153–227. Springer Vienna, Vienna.

[Rozza *et al.*, 2007] ROZZA, G., HUYNH, D. B. P. et PATERA, A. T. (2007). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):1.

[Rubio *et al.*, 2019] RUBIO, P.-B., LOUF, F. et CHAMOIN, L. (2019). Transport maps sampling with pgd model reduction for fast dynamical bayesian data assimilation. *Submitted to International Journal for Numerical Methods in Engineering*.

[Ryckelynck, 2005] RYCKELYNCK, D. (2005). A priori hyperreduction method : an adaptive approach. *Journal of computational physics*, 202(1):346–366.

[Ryckelynck, 2009] RYCKELYNCK, D. (2009). Hyper-reduction of mechanical models involving internal variables. *International Journal for Numerical Methods in Engineering*, 77(1):75–89.

[Signorini *et al.*, 2017] SIGNORINI, M., ZLOTNIK, S. et DÍEZ, P. (2017). Proper generalized decomposition solution of the parameterized helmholtz problem : application to inverse geophysical problems. *International Journal for Numerical Methods in Engineering*, 109(8):1085–1102.

[Stewart, 1993] STEWART, G. W. (1993). On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–566.

[Tomasi et Bro, 2006] TOMASI, G. et BRO, R. (2006). A comparison of algorithms for fitting the parafac model. *Computational Statistics & Data Analysis*, 50(7):1700–1734.

[Tucker, 1966] TUCKER, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

[Vitse, 2016] VITSE, M. (2016). *Réduction de modèle pour l'analyse paramétrique de l'endommagement dans les structures en béton armé*. Thèse de doctorat, Paris Saclay.

[Vitse *et al.*, 2014] VITSE, M., NÉRON, D. et BOUCARD, P.-A. (2014). Virtual charts of solutions for parametrized nonlinear equations. *Computational Mechanics*, 54(6): 1529–1539.

[Xiao, 2015] XIAO, Y. (2015). Recent developments in t-trefftz and f-trefftz finite element methods. *International Journal of Scientific Research in Science, Engineering and Technology*, 1(2):441–459.

[Zieliński, 1988] ZIELIŃSKI, A. (1988). Trefftz method : elastic and elastoplastic problems. *Computer methods in applied mechanics and engineering*, 69(2):185–204.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre : Nouvelles démarches de réduction de modèles pour le traitement des problèmes à très grand nombre de paramètres

Mots clés : Modèles Réduits, PGD, Multiparamétrique

Résumé : Alors que la simulation numérique prend aujourd'hui une place essentielle dans de nombreuses branches de l'ingénierie, les évolutions incroyables des moyens de calculs peinent à compenser la complexité croissante des modèles que les ingénieurs sont amenés à traiter. Dans ce contexte, les modèles réduits sont de véritables outils d'aide à la décision car ils permettent, une fois construits, d'évaluer un très grand nombre de scénarios en temps quasi réel. En particulier, la méthode PGD (Proper Generalized Decomposition) initée au LMT a connu de très nombreux développements (problèmes non linéaires, multiéchelles, multiphysiques...) et conduit à des gains en temps CPU pouvant atteindre plusieurs ordres de grandeur.
Malheureusement, l'essor de ces modèles réduits est actuellement freiné par la difficulté à les calculer lorsque le nombre de paramètres à prendre en compte augmente. Toutes les techniques de réduction de modèles actuelles (PGD comprise) peinent à traiter des problèmes à très grand nombre de paramètres (la limite actuelle tourne autour de la vingtaine de paramètres), ce qui constitue un verrou scientifique majeur pour l'essor de ces techniques. Cette thèse présente une adaptation de la méthode PGD qui permet le traitement de tels problèmes.
Trois contributions principales ont permis d'atteindre de telles performances. D'une part, une nouvelle structure de données plus proche de la physique du probème a été développée. Elle introduit deux échelles de représentation des fonctions paramétriques et donne son nom à la méthode : la Parameter-Multiscale PGD. Par ailleurs, une discrétisation spactiale discontinue particulièrement adaptée à nos méthodes de résolution a été implémentée, la WTDG (Weak Trefftz Discontinuous Galerkin). Enfin de nouveaux algorithmes ont été développés pour construire des modèles réduits qui permettent des gains de temps conséquents pour des problèmes ayant jusqu'à mille paramètres.

Title : New model order reduction methods for problems with a high number of parameters

Keywords : Reduced Order Model (ROM), PGD, High number of parameters

Abstract : Numerical simulation is nowadays a major tool in a large number of engineering fields. Nevertheless, even the recent incredible improvements of the computational power can hardly compensate the increasing complexity of the models used by engineers. In this context, Reduced Order Models (ROM) can be major decision-maker tools because, once they have been computed, they can be used to evaluate a very large number of test cases in a duration close to real time. The PGD (Proper Generalized Decomposition) in particular, is a method introduced at the LMT which has been adapted to many cases (non-linear problems, multiscale, multiphysics) and leads to savings of CPU time reaching several orders of magnitude.
Unfortunately, it is currently difficult to build ROM with an increasing number of parameters. All the actual model reduction technics (including the PGD) can hardly solve problems with a high number of parameters (the current limit is about twenty parameters). It is a major barrier to a larger development of these methods. This PhD thesis presents a new methodology based on the PGD able to take into account high numbers of parameters.
This goal has been achived thanks to three major contributions. First, a new data structure faithfull to mecanical properties of the problem has been developed. To that end, two different scales are introduced in the parametric space, giving its name to our method : Parameter-Multiscale PGD. Furthermore, the WTDG (Weak Treffz Discontinuous Galerkin) method has been inpemented. It is a discontinuous spatial discretisation adapted to our resolution techniques. Finally, new algorithms have been developed to built reduced order models of problems taking into account up to one thousand parameters.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France