Année 2009-2010
Master 2 « Techniques Avancées en Calcul de Structures »

Cadre d'approximation espace-temps pour le traitement d'hyper-problèmes

J. Berthe

Responsables : D. Néron
H. Leclerc

LMT-Cachan
ENS Cachan / CNRS / UPMC / PRES UniverSud Paris
61 avenue du Président Wilson, F-94235 Cachan cedex, France

Table des matières

Introduction

De nos jours, la conception de structures complexes, telles que les avions ou les lanceurs, nécessite toujours un grand panel d'essais, ce qui demande beaucoup de temps et d'argent. Le souhait des concepteurs est de pouvoir remplacer une partie de ces essais par des simulations numériques. Or pour se rapprocher du comportement réel de telles structures, il est nécessaire d'utiliser des modèles extrêmement complexes : matériaux très hétérogènes, géométrie complexe, phénomènes non linéaires... C'est par exemple le cas chez Airbus qui a récemment lancé le programme MAAXIMUS dont l'un des principaux objectifs est de simuler un tronçon complet d'avion, ce qui conduit à un calcul non linéaire à un milliard de degrés de liberté, que les codes commerciaux ne permettent pas de réaliser.

Depuis de nombreuses années, le LMT-Cachan développe des méthodes de calculs alternatives afin de résoudre plus efficacement les problèmes de grande taille. L'une d'entre elles repose sur une stratégie de décomposition de modèle mixte, basée sur la méthode LATIN multiéchelle avec homogénéisation en espace et en temps. Elle consiste en une procédure itérative conduisant à la résolution de problèmes à une échelle fine « micro » et à une échelle homogénéisée « macro ». Elle tire ainsi parti des ordinateurs à architecture parallèle. Un des points principaux de la méthode LATIN est l'utilisation de la Proper Generalized Decomposition, qui consiste en la représentation d'une fonction de l'espace et du temps à l'aide d'une somme de produits d'une fonction de l'espace par une fonction du temps. Cela permet de réduire considérablement les coûts de calcul et de stockage.

Cette technique peut être vue comme une méthode de réduction de modèles *a priori* qui permet de construire une approximation de la solution sans l'imposer. Cependant la construction des opérateurs du modèle réduit reste coûteuse, notamment à cause de la manipulation de grandeurs à variables séparées et de grandeurs qui dépendent de plusieurs variables dans le calcul de certaines intégrales. L'idée est d'introduire des « points de référence » et des « instants de référence » pour généraliser le concept de fonctions à variables séparées afin de proposer un cadre algébrique dans lequel tous les champs peuvent être représentés et toutes les opérations élémentaires effectuées pour un coût modique. L'utilisation de cette algèbre permet d'envisager une réduction supplémentaire des coûts de calcul et stockage des données.

Dans ce mémoire, le premier chapitre introduit le concept de la Proper Orthogonal Decomposition (POD) et ses applications pour la réduction de modèle *a posteriori*. Le second chapitre se penchera sur une méthode de réduction de modèle *a priori* avec la Proper Generalized Decomposition (PGD) et son application dans le cadre de la méthode

LATIN. Le troisième chapitre introduit la problématique précise de ce stage et propose une première solution. Le quatrième chapitre propose plusieurs méthodes de reconstructions de fonctions pour résoudre les problèmes évoqués dans le troisième chapitre. La solution la plus efficace sera implémentée dans la méthode LATIN dans le chapitre 5. Enfin, les améliorations et perspectives de ces travaux seront détaillées.

Chapitre 1

Méthodes de réduction de modèle basées sur la Proper Orthogonal Decomposition (POD)

Ce premier chapitre présente une première approche, dite a posteriori, pour la réduction de modèle, basée sur la POD.

Sommaire

1 Introduction

Aujourd'hui, dans tous les domaines, les résultats d'expériences ou de simulations sont constitués de millions de données. Le traitement de ces données, ainsi que leur stockage, a conduit de nombreux scientifiques à développer des méthodes d'approximation des solutions afin d'en extraire les informations essentielles. La POD (*Proper Orthogonal Decomposition*) a été développée, dans un premier temps, afin d'approximer à moindre coût la description de phénomènes complexes (vibrations de structure, écoulements turbulents...). Cette méthode est connue sous diverses appelations : Karhunen-Loeve Decomposition, Principal Component Analysis ou Singular Value Decomposition (SVD) ([Liang *et al.*, 2002]). En effet, la POD a fréquemment été « ré-inventée » dans ses différents domaines d'applications : traitement du signal, compression de données, traitement d'images, dynamique des fluides, thermique transitoire... Les développements récents autour de cette méthode ont pris une toute autre orientation. La POD n'est plus seulement considérée comme une technique de traitement de données, mais également comme une méthode de réduction de modèle *a posteriori*, permettant de réduire de façon considérable la taille des systèmes à résoudre.

Dans ce chapitre, la POD sera d'abord présentée dans le cadre général des méthodes d'approximations. Ensuite, deux exemples de l'utilisation de la POD comme une méthode de compression de données seront développés. Enfin, la POD sera utilisée comme une méthode de réduction de modèle *a posteriori*, afin de diminuer les temps de calculs pour la résolution de problèmes complexes.

2 Méthodes d'approximation à variables séparées

Soit $f(x,t)$ la fonction représentant la solution d'un problème. Les variables x et t peuvent être les variables classiques représentant respectivement l'espace et le temps, mais pas nécessairement. Elles peuvent par exemple être associées chacune à une variable d'espace (x_1 et x_2). Le but est d'approcher la fonction f par une somme finie de produits de fonctions de x et de fonctions de t :

$$f(x,t) = \tilde{f}(x,t) \simeq \sum_{k=1}^{m} a_k(t)\phi_k(x) \tag{1.1}$$

L'objectif est d'avoir une variable m « pas trop grande », afin de réduire la quantité de données nécessaire à la représentation de la fonction, et que lorsque m tend vers l'infini l'approximation soit exacte.

En considérant que la variable x représente une variable d'espace, plusieurs familles de fonctions peuvent être envisagées pour choisir les $\{\phi_k(x)\}_{k=1}^m$: les polynômes de Tchebychev, les polynômes de Legendre, les séries de Fourier... Et pour chacune de ces familles, une famille différente de $\{a_k(t)\}_{k=1}^m$ sera obtenue à l'aide du problème de minimisation suivant :

$$\arg\min_{a_k(t)} \| f(x,t) - \sum_{k=1}^{m} a_k(t)\phi_k(x) \|^2 \tag{1.2}$$

Ce qui permet de dire que la décomposition de la formule (1.1) n'est pas unique. Pour obtenir l'unicité de la décomposition, il faut ajouter des critères.

3 Principe de la Proper Orthogonalized Decomposition

Il est possible de présenter la POD comme une méthode d'approximation de la solution d'un problème [Chatterjee, 2000]. Dans ce cas, on impose que la famille des $\{\phi_k(x)\}_{k=1}^m$ de l'équation (1.1) soit une famille de vecteurs orthonormés :

$$\int_\Omega \phi_k(x)\phi_l(x)dx = \delta_{kl} \tag{1.3}$$

La POD est construite de sorte que l'approximation d'ordre m minimise la distance à la fonction pour une norme. Classiquement la norme utilisée est la norme associée au produit scalaire sur L^2 :

$$(f,g) = \int_\Omega fgdx \quad \text{et} \quad \|f\|^2 = (f,f) \tag{1.4}$$

Lorsque les fonctions $\{\phi_k(x)\}_{k=1}^m$ sont connues, il est possible d'obtenir les $\{a_k(t)\}_{k=1}^m$ en réalisant le produit scalaire entre la fonction $f(x,t)$ et les $\{\phi_k(x)\}_{k=1}^m$:

$$(f(x,t),\phi_k(x)) = \int_\Omega f(x,t)\phi_k(x)dx = \sum_{l=1}^m a_l(t)\int_\Omega \phi_l(x)\phi_k(x)dx = a_k(t) \tag{1.5}$$

Il suffit donc de construire la base des $\{\phi_k(x)\}_{k=1}^m$ pour construire la POD de la fonction $f(x,t)$. Pour cela on suppose la fonction $f(x,t)$ connue en N_x points x_i de l'espace et en N_t instants t_i. Alors, d'après tout ce qui précède, les fonctions $\{\phi_k(x)\}_{k=1}^m$ sont obtenues par résolution du problème de minimisation suivant :

$$\arg\min_{\phi_k(x)} \sum_{l=1}^{N_t} \|f(x,t_i) - \sum_{k=1}^m (f(x,t_i),\phi_k(x))\phi_k(x)\|^2 \tag{1.6}$$

Cette formulation est une formulation continue. Or, si les données que l'on souhaite approximer sont issues d'un essai expérimental, celles-ci sont discrètes et proviennent d'un certain nombre de capteurs. Dans ce cas, il est commode d'introduire la matrice A des « Snapshot » [Cordier et Bergmann, 2006], qui stocke l'ensemble des réalisations de f :

$$A = \begin{pmatrix} f(x_1,t_1) & f(x_1,t_2) & . & . & f(x_1,t_{N_t}) \\ f(x_2,t_1) & f(x_2,t_2) & . & . & f(x_2,t_{N_t}) \\ . & . & & & . \\ . & . & & & . \\ . & . & & & . \\ f(x_{N_x},t_1) & f(x_{N_x},t_2) & . & . & f(x_{N_x},t_{N_t}) \end{pmatrix} \tag{1.7}$$

Dans le cas discret, la solution du problème de minimisation (1.6) est équivalente à la Décomposition aux Valeurs Singulières de la matrice A tronquée à l'ordre m.

4 La Décomposition aux Valeurs Singulières (SVD)

Soit la matrice A de taille $N_x \times N_t$ à coefficients réels définie précédement. Alors la Décomposition aux Valeurs Singulières de A s'écrit de la façon suivante :

$$A = U\Sigma V^T \tag{1.8}$$

avec U et V des matrices orthogonales respectivement de taille $N_x \times N_x$ et $N_t \times N_t$ et Σ une matrice diagonale de dimension $N_x \times N_t$ contenant les valeurs singulières de A notées $\sigma_1, ..., \sigma_p$ avec $p = \min(N_x, N_t)$. Les valeurs singulières sont telles que : $\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_r > \sigma_{r+1} = ... = \sigma_p = 0$ avec r le rang de la matrice A. Les r premières colonnes de V et de U sont dénommées vecteurs singuliers droit et gauche de A.

D'après ce qui précède, il est possible d'écrire :

$$A^T A = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T \tag{1.9}$$

Or $A^T A$ est une matrice symétrique réelle. Elle est donc diagonalisable dans une base orthonormée de vecteurs propres :

$$A^T A = W\Lambda W^{-1} = W\Lambda W^T \tag{1.10}$$

De plus $A^T A$ est définie positive, donc ses valeurs propres sont positives ou nulles. Par comparaison des équations (1.9) et (1.10), il donc possible de dire que $\Sigma^2 = \Lambda$ et que $W = V$. Donc (V, Λ) représentent la décomposition aux valeurs propres de $A^T A$. On retrouve ici une propriété des valeurs singulières de la matrice A, celles-ci sont égales à la racine carré des valeurs propres de $A^T A$: $\sigma_i = \sqrt{\lambda_i}$.

En appliquant la même démarche à la matrice AA^T qui est également symétrique définie positive, on obtient : $AA^T = U\Sigma^2 U^T = W\Lambda W^T$. Ce qui conduit à la conclusion suivante : (U, Λ) représentent la décomposition aux valeurs propres de AA^T.

L'expression de la POD est obtenue en prenant pour les fonctions du temps les colonnes de la matrice $U\Sigma$ et pour les fonctions de l'espace les colonnes de la matrice V^T.

5 Applications de la POD

Pour illustrer la puissance de la POD mais aussi ses limites, deux exemples vont être développés : le premier a pour but d'approximer à moindre coût une fonction connue et le deuxième de compresser une image afin de diminuer l'espace occupé par celle-ci dans la mémoire [Long, 1983, Andrews et Patterson, 1975].

5.1 Approximation de fonctions

Soit la fonction $f(x,t) = e^{-|(x-0.5)(t-1)|} + \sin(xt)$ avec $x \in [0;1]$ et $t \in [0;2]$. Afin d'obtenir la POD de f, l'espace est discrétisé en 25 intervalles et le temps est discrétisé en 50 intervalles. Sur la figure 1.1, le premier graphique représente le tracé de la fonction

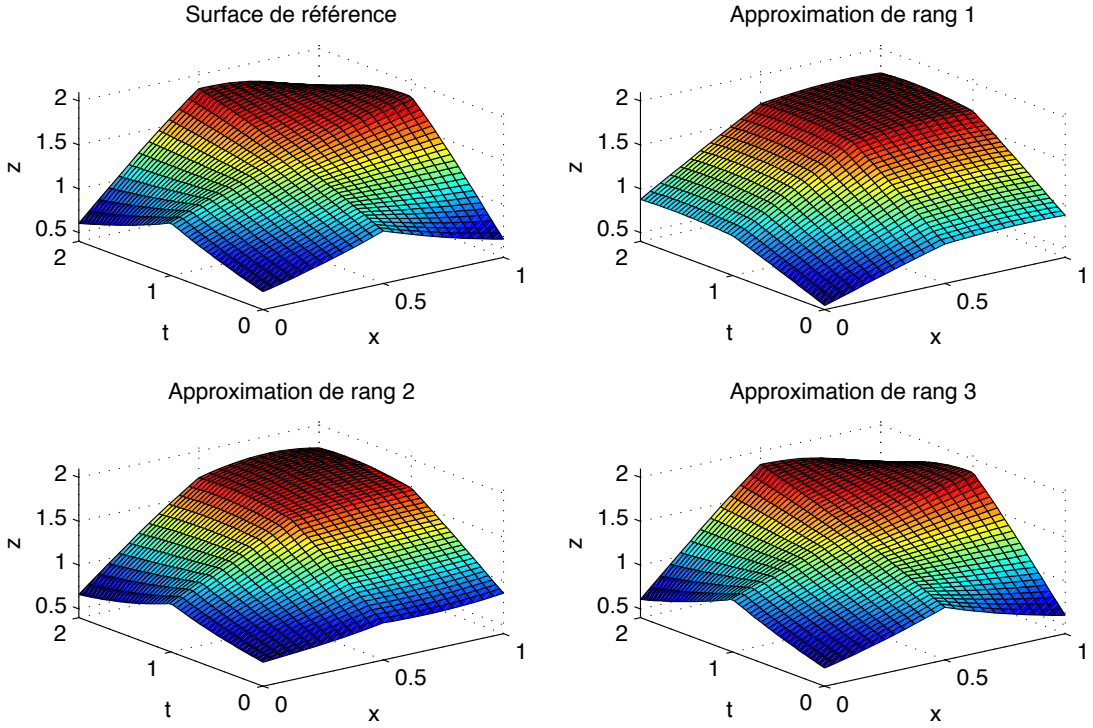


FIGURE 1.1: Tracé de la fonction f sans approximation, ensuite tracé de la fonction approximée à l'aide de la POD d'ordre 1, 2 et 3

f en fonction de x et de t sans approximation, les trois graphiques suivant représentent la fonction f approximée à l'ordre 1, 2 et 3. Il est quasiment impossible de voir à l'aide de ces graphiques la différence entre le tracé de l'approximation à l'ordre 3 et le tracé de la fonction sans approximation. Il est possible de définir une erreur relative entre la fonction approximée $\tilde{f}$ et la fonction exacte f :

$$\eta_p = \frac{\|f - \tilde{f}\|_{I\times\Omega}}{\|\frac{1}{2}(f+\tilde{f})\|_{I\times\Omega}} \tag{1.11}$$

avec $\|\cdot\|_{I\times\Omega}$ la norme associée au produit scalaire :

$$(f,g)_{I\times\Omega} = \int_{I\times\Omega} fg\, dtd\Omega \tag{1.12}$$

Dans cette exemple, l'erreur commise est de 5,6% pour l'approximation de rang 1, 4% pour l'approximation de rang 2 et 0,8% pour l'approximation de rang 3. La POD semble donc être très efficace pour approximer cette fonction.

Considérons maintenant la fonction $g(x,t) = e^{-a(x-t)^2}$ avec $x \in [-2;2]$ et $t \in [2;2]$. L'espace est discrétisé en 80 intervalles et le temps est discrétisé en 80 intervalles. Sur la figure 1.2, le premier graphique représente le tracé de la fonction g en fonction de x et de

t sans approximation, les trois graphiques suivants représentent la fonction f approximée à l'ordre 1, 2 et 3. Cette fois la POD semble rencontrer plus de difficultés pour approximer la fonction avec un ordre d'approximation faible, l'erreur pour l'approximation de rang 3 est de 62%. Afin de comprendre pourquoi l'approximation avec un ordre faible

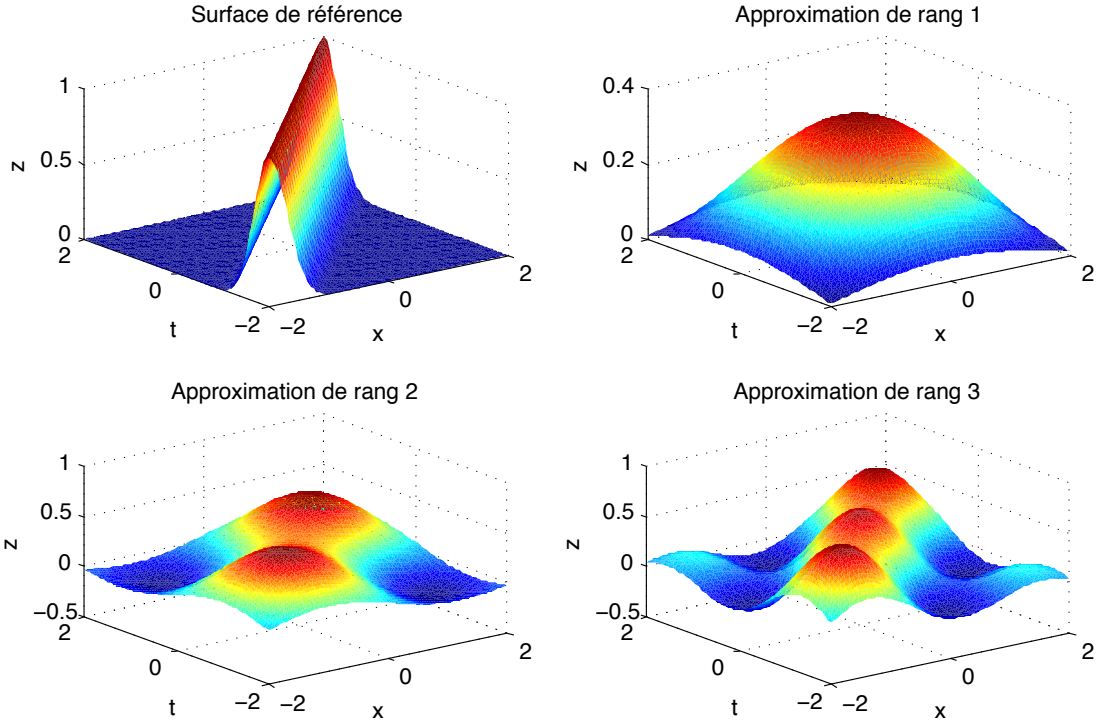


FIGURE 1.2: Tracé de la fonction g sans approximation, ensuite tracé de la fonction approximée à l'aide de la POD d'ordre 1, 2 et 3

fonctionne très bien avec la fonction f et pas avec la fonction g, l'amplitude des valeurs singulières de f et de g est représentée sur la figure 1.3. Comme la théorie le prévoit, l'amplitude des valeurs singulières σ_i décroit lorsque i augmente. Par contre, l'amplitude des valeurs singulières décroit plus vite pour la fontion f que pour la fonction g, ce qui explique la fonction f soit plus facile à approximer avec un ordre faible que la fonction g. Cette différence entre les deux fonctions provient principalement du fait que la fonction g présente des variations brusques de ses valeurs. Ces phénomènes localisés compliquent la représentation de la fonction par une approximation d'ordre faible.

5.2 Compression d'image

L'utilisation de la SVD pour la compression d'image est connue depuis de nombreuses annéees [Long, 1983][Andrews et Patterson, 1975]. La figure 1.4 vient illustrer ce propos. L'image originale est de taille 375 x 500 pixels. Le stockage de cette image sous la forme

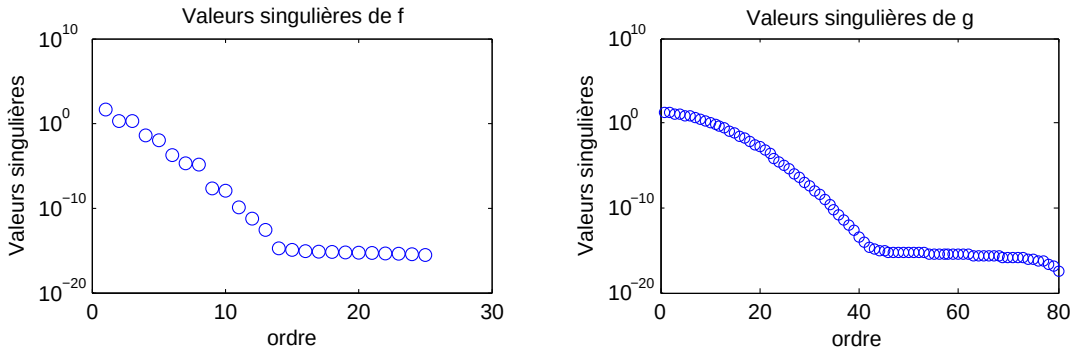


FIGURE 1.3: Amplitude des valeurs singulières de f (à gauche) et de g (à droite) (échelle logarithmique)

d'une matrice de niveaux de gris implique la présence en mémoire de 187500 réels. La décomposition en valeurs singulières de cette matrice est réalisé à l'aide de Matlab et une image approximée est reconstruite pour différentes valeurs du rang d'approximation (1, 2, 5, 10, 25 et 50). Dès l'approximation de rang 10, l'image approximée est proche de l'image originale et pour l'approximation de rang 50 la différence entre les deux images est minime. Le stockage de la matrice associée à l'image approximée au rang 50 implique la présence en mémoire de 2500 réels. La compression d'image à l'aide de la SVD permet donc des gains importants d'espace de stockage.

6 Réduction de modèle à l'aide de la POD

Jusqu'à présent la POD a été présentée comme une méthode d'approximation et de compression, c'est à dire comme une méthode permettant d'extraire l'information principale d'un ensemble de données. Mais cette méthode peut également être utilisée comme une méthode de réduction *a posteriori* d'un problème. Ces méthodes de réductions de modèles ont été utilisées dans de nombreux domaines : optimisation [Schmidt *et al.*, 2010], dynamique des fluides [Cordier et Bergmann, 2006][Gunzburger *et al.*, 2007], conduction thermique [Białecki *et al.*, 2005]... Une première série de calculs, appelée *snapshots*, est menée à l'aide des méthodes classiques, spécifiques au domaine d'application, afin de pouvoir construire la base POD qui servira pour l'approximation. Il y a deux manières de réaliser ces *snapshots* : la première consiste à utiliser le modèle complet et à réaliser un calcul fin sur un intervalle de temps limité (cf. figure 1.5 (a)). La deuxième consiste à réaliser une série de calculs sur l'intervalle de temps complet mais avec un maillage grossier (cf. figure 1.5 (b)). Une fois les *snapshots* réalisés, une POD de ceux-ci est réalisée afin d'en extraire une base de dimension m. Ensuite les équations du problème sont projetées sur cette base afin de construire le nouveau système d'équation. Le nouveau système d'équation à résoudre est de taille m (ordre d'approximation de la POD) alors que la taille du système initial est N (N peut-être de l'ordre de quelques millions), ce qui conduit a

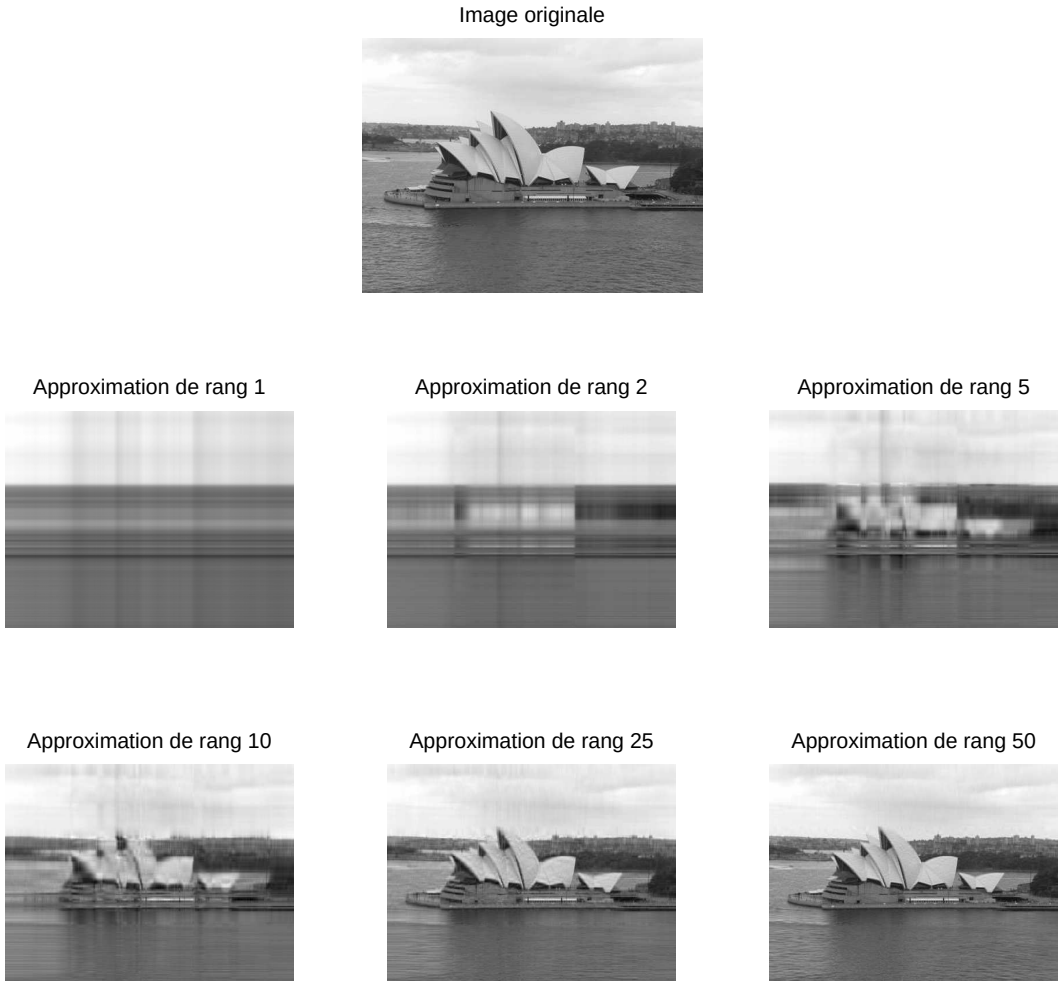
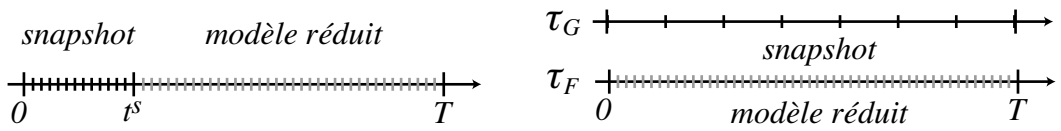


FIGURE 1.4: Utilisation de la SVD pour la compression d'images : visualisation du résultat en fonction du rang de l'approximation



(a) Calcul partiel (b) Calcul approché

FIGURE 1.5: Description des *snapshots* [Coelho et Breitkopf, 2009]

des réductions des temps de calculs car $m \ll N$. Cette méthode se révèle particulièrement efficace dans le cadre des méthodes d'optimisation par exemple, car un grand nombre de calculs sont réalisés pour des configurations qui changent très peu, donc une fois la base obtenue les différentes configurations sont étudiées à moindre coût. Par contre, le principal inconvénient de cette méthode réside dans le choix des *snapshots*. En effet la POD est incapable de représenter une information qui n'était pas présente initialement dans la base de données qui a été utilisée pour sa création. Il n'est donc pas assuré que la base construite soit une base optimale pour la description des phénomènes intervant dans le problème générale. C'est le principal inconvénient de cette méthode.

Chapitre 2

Proper Generalized Decomposition (PGD)

Ce second chapitre présente la PGD et plus particulièrement son application dans le cadre de la méthode LATIN afin de réduire les coûts de calcul.

Sommaire

1 Introduction

Dans le chapitre précédent, l'efficacité de la POD pour la réduction de problèmes a été démontrée, ainsi que ces limites. Dans ce qui va suivre, une technique de réduction de modèle *a priori* va être développée. Cette méthode, connue sous le nom de Proper Generalized Decomposition (PGD), a été introduite dans les années 80 par Pierre Ladevèze dans la méthode LATIN sous le nom de décomposition radiale [Ladevèze, 1985]. L'objectif de cette méthode est de générer la meilleure approximation possible d'une solution avec une représentation séparée des variables. Cette méthode a récemment été appliquée à la résolution de problèmes stochastiques [Nouy, 2007] et à la résolution de problèmes avec un grand nombre de variables [Chinesta *et al.*, 2008]. Ce chapitre détaille l'utilisation de la PGD pour réduire les coûts de calcul de la méthode LATIN.

2 La méthode LATIN

2.1 Le problème de référence

Le problème de réference est défini sur une structure Ω et sur l'intervalle d'étude $I = [0,T]$ pour une évolution quasi-statique et isotherme sous l'hypothèse des petites perturbations. Cette structure est soumise à des efforts extérieurs volumiques $\underline{f}_d$ et surfacique $\underline{F}_d$ sur sa frontière $\partial_2\Omega$. Sur la partie complémentaire $\partial_1\Omega$ de sa frontière, le déplacement $\underline{u}_d$ est imposé (Figure 2.1). Les conditions initiales sont toutes nulles, dans un souci de simplicité, sauf le déplacement $\underline{U}$ dont la condition initiale est notée $\underline{U}_0$.

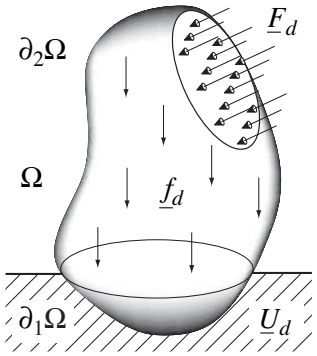


FIGURE 2.1: Problème de référence

L'état de la structure est entièrement définie par la connaisance des champs ($\dot{\varepsilon}_p$, $\dot{\mathbf{X}}$, σ, $\mathbf{Y}$) avec :

- ε_p qui désigne la partie anelastique de la déformation ε associée au champ de déplacement $\underline{U}$. En effet, la déformation ε peut-être décomposée en une partie élastique ε_e et une partie anélastique ε_p de la façon suivante : $\varepsilon_p = \varepsilon - \varepsilon_e$

– $\mathbf{X}$ qui désigne les autres variables internes
– σ qui désigne le tenseur des contraintes de Cauchy
– $\mathbf{Y}$ qui désigne l'ensemble des variables conjuguées de $\mathbf{X}$

Toutes les quantités sont supposées suffisament régulières sur le domaine $I \times \Omega$ et vérifient les conditions initiales en $t = 0$. La notation suivante est introduite pour les champs primaux et duaux :

$$\mathbf{u} = \begin{bmatrix} U \\ 0 \end{bmatrix}, \ \mathbf{e}_p = \begin{bmatrix} \varepsilon_p \\ -\mathbf{X} \end{bmatrix}, \ \mathbf{e} = \begin{bmatrix} \varepsilon \\ 0 \end{bmatrix}, \mathbf{e}_e = \begin{bmatrix} \varepsilon_e \\ \mathbf{X} \end{bmatrix}, \text{ avec } \mathbf{e}_p = \mathbf{e} - \mathbf{e}_e \text{ et } \mathbf{f} = \begin{bmatrix} \sigma \\ \mathbf{Y} \end{bmatrix} \quad (2.1)$$

Le taux de dissipations s'écrit donc :

$$\int_\Omega (\dot{\varepsilon}_p : \sigma - \dot{\mathbf{X}} \cdot \mathbf{Y}) d\Omega = \int_\Omega (\dot{\mathbf{e}}_p \circ \mathbf{f}) d\Omega \quad (2.2)$$

avec $\cdot$ le produit scalaire adaptée à la nature tensorielle de $\mathbf{X}$ et de $\mathbf{Y}$ et $\circ$ l'opérateur associé. La forme bilinéaire dissipation est également introduite :

$$\langle \mathbf{s}, \mathbf{s}' \rangle = \int_{I \times \Omega} (1 - \frac{1}{T})(\dot{\mathbf{e}}_p \circ \mathbf{f}' + \dot{\mathbf{e}}'_p \circ \mathbf{f}) d\Omega dt \quad (2.3)$$

avec $\mathbf{E}$ et $\mathbf{F}$, les espaces associés aux champs $\dot{\mathbf{e}}_p$ et $\mathbf{f}$ compatibles avec l'équation (2.3). Ces espaces définissent l'espace $\mathbf{S} = \mathbf{E} \times \mathbf{F}$ dans lequel l'état de la structure $\mathbf{s} = (\dot{\mathbf{e}}_p, \mathbf{f})$ est recherché.

Une formulation dite « normale » est utilisée pour décrire le comportement des matériaux. Les équations d'état sont donc les suivantes :

$$\begin{aligned} \sigma &= \rho \frac{\partial \psi}{\partial \varepsilon_e} = \mathbf{K} \varepsilon_e \\ \mathbf{Y} &= \rho \frac{\partial \psi}{\partial \mathbf{X}} = \Lambda \mathbf{X} \end{aligned} \quad (2.4)$$

avec $\mathbf{K}$ le tenseur de Hooke, Λ un tenseur constant symétrique défini positif, ρ la masse volumique et $\rho\psi(\varepsilon_e, \mathbf{X})$ l'énergie libre. Il est possible de réécrire ces équations de la façon suivante :

$$\mathbf{f} = \mathbf{A}\mathbf{e}_e \quad \text{avec} \quad \mathbf{A} = \begin{bmatrix} \mathbf{K} & 0 \\ 0 & \Lambda \end{bmatrix} \quad (2.5)$$

La loi d'évolution, qui peut être non linéaire, est obtenue par l'opérateur différentiel positif $\mathbf{B}$ tel que :

$$\dot{\mathbf{e}}_p = \mathbf{B}(\mathbf{f}) = \begin{bmatrix} \partial_\sigma \phi^* \\ \partial_\mathbf{Y} \phi^* \end{bmatrix} \quad (2.6)$$

avec $\phi^*(\sigma, \mathbf{Y})$ le pseudo-potentiel des dissipations.

Ce formalisme est valable pour un grand nombre de modèles de matériaux et une description détaillée des opérateurs $\mathbf{A}$ et $\mathbf{B}$ est disponible dans [Ladevèze, 1999] pour les matériaux viscoélastiques et viscoplastiques.

Afin de finir la définition du problème de référence, il faut introduire des sous-espaces fonctionnels de $\mathbf{S}$ correspondant aux conditions d'admissibilité des champs solutions :

- **L'espace $\mathcal{U}$ des champs cinématiquement admissibles u** qui correspond à l'ensemble des champs de déplacement $\underline{U}$ qui sont égaux à $\underline{U}_d$ sur $\partial_1\Omega$ et qui vérifient les conditions initiales :

$$\underline{U}_{|\partial_1\Omega} = \underline{U}_d \quad \text{et} \quad \underline{U}_{|t=0} = \underline{U}_0 \tag{2.7}$$

- **L'espace $\mathcal{F}$ des champs statiquement admissibles f** qui correspond à l'ensemble des champs de contrainte σ qui sont en équilibre avec les forces surfaciques $\underline{F}_d$ sur $\partial_2\Omega$ et les forces volumiques $\underline{f}_d$ sur Ω :

$$\forall \mathbf{u}^\star \in \mathcal{U}^\star,\ -\int_{I\times\Omega} \mathbf{f}\circ\mathbf{e}(\dot{\underline{U}}^\star)d\Omega dt + \int_{I\times\Omega} \underline{f}_d\cdot\dot{\underline{U}}^\star d\Omega dt + \int_{I\times\partial_2\Omega} \underline{F}_d\cdot\dot{\underline{U}}^\star dSdt = 0 \tag{2.8}$$

- **L'espace $\mathcal{E}$ des champs cinématiquement admissibles $\dot{\mathbf{e}}$** dont le champ de déformation ε dérive d'un champs de déplacement $\underline{U}$ appartenant à $\mathcal{U}$:

$$\forall \mathbf{f}^\star \in \mathcal{F}^\star,\ -\int_{I\times\Omega} \mathbf{f}^\star\circ\dot{\mathbf{e}}\ d\Omega dt + \int_{I\times\partial_1\Omega} \sigma^\star\underline{n}\cdot\dot{\underline{U}}^\star_d\ dSdt = 0 \tag{2.9}$$

- **L'espace $\mathbf{A_d}$ des champs admissibles s** pour lesquels **f** est statiquement admissible, $\dot{\mathbf{e}}_p$ et **f** vérifie les équations d'état (2.5) et dont le champ $\dot{\mathbf{e}}$ est cinématiquement admissible :

$$\mathbf{f}\in\mathcal{F},\ (\mathbf{A^{-1}}\dot{\mathbf{f}}+\dot{\mathbf{e}}_\mathbf{p})\in\mathcal{E} \tag{2.10}$$

- **L'espace Γ des champs admissibles s** pour lesquels $\dot{\mathbf{e}}_p$ et **f** vérifie la loi d'évolution (2.6) :

$$\dot{\mathbf{e}}_p = \mathbf{B}(\mathbf{f}) \tag{2.11}$$

La solution $\mathbf{s_{ref}}$ du problème sur l'espace-temps $\Omega\times I$ peut donc être vue comme l'intersection des deux espaces $\mathbf{A_d}$ et Γ. La recherche de la solution du problème de référence consiste donc à :

$$\text{Trouver } \mathbf{s_{ref}}\in\mathbf{A_d}\cap\Gamma \tag{2.12}$$

Remarque : Les champs appartenant à $\mathbf{A_d}$ sont solutions d'un système d'équations linéaires et globales, alors que les champs appartenant à Γ sont solutions d'un système d'équations pouvant être non linéaires et locales en temps et en espace.

2.2 Principe de la méthode LATIN

Pour résoudre le problème précédemment défini, il est possible d'utiliser une méthode incrémentale. L'intervalle de temps I est discrétisé en une série de petits intervalles $[t_i;t_{i+1}]$ avec $t_{i+1} = t_i + \Delta t$ où Δt représente l'incrément de temps. Ces méthodes s'avèrent très coûteuses surtout si le problème est non linéaire.

Afin d'éviter ce problème, une méthode non incrémentale va être utilisée, la méthode LATIN (LArge Time INcrement). Originalement introduite pour la résolution de problèmes non linéaire d'évolution [Ladevèze, 1985], la LATIN est une méthode de résolution des problèmes de mécanique sur tout l'espace temps. En effet, à chaque itération, le

problème est résolu sur l'ensemble de la structure Ω et sur l'ensemble de l'intervalle de temps I. Cette méthode repose sur trois grandes idées.

La première est de séparer les difficultés. Le problème de référence consiste à trouver $\mathbf{s_{ref}} \in \mathbf{A_d} \cap \Gamma$ avec $\mathbf{A_d}$ qui contient des équations linéaires et globales et Γ qui contient des équations pouvant être non linéaires et locales en temps et en espace. Les équations contenues dans $\mathbf{A_d}$ sont faciles à résoudre car elles sont linéaires, mais cette opération est coûteuse car elles sont globales. Par contre, les équations contenues dans Γ sont difficiles à résoudre car non linéaire mais cette opération est peu coûteuse car locale. La séparation des difficultés dans la méthode LATIN est donc basée sur les propriétés de ces deux espaces.

La deuxième grande idée de la LATIN est d'utiliser un schéma itératif en deux étapes, qui résout successivement chacun des deux groupes d'équations. La solution est construite alternativement, en construisant dans un premier temps un champ solution de Γ au cours de l'étape locale et en construisant dans un deuxième temps un champ solution de $\mathbf{A_d}$ au cours de l'étape globale (voir Figure 2.2). A la fin de l'itération $n+1$, $\hat{\mathbf{s}}_{n+1/2} \in \Gamma$ et $\mathbf{s}_{n+1} \in \mathbf{A_d}$ sont définis sur $I \times \Omega$, c'est pourquoi cette méthode est dite non incrémentale.

$$\cdots \rightarrow \mathbf{s_n} \in \mathbf{A_d} \underbrace{\xrightarrow{\text{Etape locale}} \hat{\mathbf{s}}_{n+1/2} \in \Gamma \xrightarrow{\text{Etape linéaire}} \mathbf{s}_{n+1} \in \mathbf{A_d}}_{\text{Itération } n+1} \rightarrow \hat{\mathbf{s}}_{n+3/2} \in \Gamma \rightarrow \cdots$$

FIGURE 2.2: Etape locale et linéaire à l'itération $n+1$

La troisième grande idée de la méthode LATIN est l'utilisation d'une représentation des inconnues adaptées au problème. Cette représentation est basée sur la PGD et qui était originalement appelée dans la méthode LATIN approximation radiale. Cette troisième idée sera détaillée dans la partie 3.

La figure 2.3 tente de donner une représentation graphique de la méthode dans le plan $(\mathbf{f},\dot{\mathbf{e}}_p)$. Elle montre la nécessité d'introduire les directions de recherche $\mathbf{E}^+$ et $\mathbf{E}^-$, qui permettent de passer d'un espace à l'autre afin de fermer le problème. Ceci va être fait dans la suite.

2.3 L'étape locale à l'itération n+1

Cette étape consiste à trouver $\hat{\mathbf{s}}_{n+1/2} \in \Gamma$, connaissant $\mathbf{s}_n \in \mathbf{A_d}$, à l'aide de la direction de recherche ascendante $\mathbf{E}^+$ définie par :

$$\mathbf{E}^+ = \left\{ \delta\mathbf{s} = (\delta\dot{\mathbf{e}}_p, \delta\mathbf{f}) \mid \delta\dot{\mathbf{e}}_p + \mathbf{H}^+\delta\mathbf{f} = 0 \right\} \tag{2.13}$$

avec $\delta\mathbf{s} = \hat{\mathbf{s}}_{n+1/2} - \mathbf{s}_n$ et $\mathbf{H}^+$ un opérateur symétrique défini positif qui est un paramètre de la méthode. La recherche de $\hat{\mathbf{s}}_{n+1/2} = (\hat{\dot{\mathbf{e}}}_{p_{n+1/2}}, \hat{\mathbf{f}}_{n+1/2})$ qui appartient à Γ et à $\mathbf{E}^+$ correspond à la résolution du problème non linéaire suivant :

$$\mathrm{B}(\hat{\mathbf{f}}_{n+1/2}) + \mathbf{H}^+\hat{\mathbf{f}}_{n+1/2} = \dot{\mathbf{e}}_{p_n} + \mathbf{H}^+\mathbf{f_n} \tag{2.14}$$

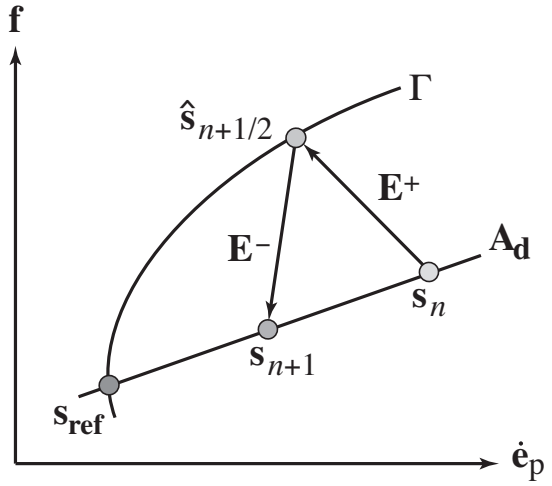


FIGURE 2.3: Représentation graphique de la méthode LATIN dans le plan ($\mathbf{f}$,$\dot{\mathbf{e}}_p$)

Remarque : Le membre de droite est connue à ce stade et ces équations sont locales en temps et en espace (ce qui justifie l'appellation d'étape locale). Une fois le problème (2.14) résolu, le terme $\hat{\mathbf{e}}_p$ est obtenu à l'aide de la direction de recherche $\mathbf{E}^+$: $\hat{\mathbf{e}}_{p_{n+1/2}} = \dot{\mathbf{e}}_{p_n} + \mathbf{H}^+\mathbf{f_n} - \mathbf{H}^+\hat{\mathbf{f}}_{n+1/2}$

2.4 L'étape linéaire à l'itération n+1

Cette étape consiste à trouver $\mathbf{s}_{n+1} \in \mathbf{A_d}$, connaissant $\hat{\mathbf{s}}_{n+1/2} \in \Gamma$, à l'aide de la direction de recherche descendante $\mathbf{E}^-$ définie par :

$$\mathbf{E}^- = \left\{ \delta\mathbf{s} = (\delta\dot{\mathbf{e}}_p, \delta\mathbf{f}) \mid \delta\dot{\mathbf{e}}_p - \mathbf{H}^-\delta\mathbf{f} = 0 \right\} \tag{2.15}$$

avec $\delta\mathbf{s} = \mathbf{s}_{n+1} - \hat{\mathbf{s}}_{n+1/2}$ et $\mathbf{H}^-$ un opérateur symétrique défini positif qui est un paramètre de la méthode. La recherche de $\mathbf{s}_{n+1} = (\dot{\mathbf{e}}_{p_{n+1}}, \mathbf{f}_{n+1})$ qui appartient à $\mathbf{A_d}$ et à $\mathbf{E}^-$ correspond à la résolution du problème global suivant tel que $\forall \mathbf{f}^\star \in \mathcal{F}^\star$:

$$\int_{I\times\Omega} \mathbf{f}^\star \circ \left(\mathbf{A^{-1}}\dot{\mathbf{f}}_{\mathbf{n+1}} + \mathbf{H}^-\mathbf{f_{n+1}}\right) d\Omega dt = \int_{I\times\Omega} \mathbf{f}^\star \circ \hat{\mathbf{a}} d\Omega dt + \int_{I\times\partial_1\Omega} \boldsymbol{\sigma}^\star \underline{n} \cdot \underline{\dot{U}_d} dS dt = 0 \tag{2.16}$$

avec $\hat{\mathbf{a}} = \hat{\mathbf{e}}_{p_{n+1/2}} - \mathbf{H}^-\hat{\mathbf{f}}_{n+1/2}$ connue à ce stade et $\underline{U_d}$ la condition de déplacement imposé. Ce problème est linéaire (ce qui justifie l'appellation d'étape linéaire) mais qui dépend du temps. C'est pour la résolution de ce problème que la PGD est utilisée. Une fois le problème (2.16) résolu, le terme $\dot{\mathbf{e}}_{p_{n+1}}$ est obtenu à l'aide de la direction de recherche $\mathbf{E}^-$: $\dot{\mathbf{e}}_{p_{n+1}} = \hat{\mathbf{a}} - \mathbf{H}^-\mathbf{f}_{n+1}$

2.5 Convergence de la méthode

La convergence de la méthode a été démontrée dans [Ladevèze, 1999] dans le cas où l'opérateur $\mathbf{B}$ est maximal monotone, l'opérateur $\mathbf{A}$ est symétrique défini positif et les directions de montée et descente sont telles que : $\mathbf{H}^+ = \mathbf{H}^- = \mathbf{L}$ avec $\mathbf{L}$ un opérateur symétrique défini positif.

Afin d'assurer la convergence de la méthode pour une large gamme de comportement matériau, une technique de relaxation est mise en place. En renommant $\bar{s}_{n+1}$ la solution de l'étape linéaire à l'itération n+1, la solution s_{n+1} de l'étape de relaxation à l'intération n+1 est obtenu par la formule :

$$s_{n+1} = \mu \bar{s}_{n+1} + (1-\mu) s_{n+1} \tag{2.17}$$

avec μ le paramètre de relaxation généralement pris égal à 0,8.

2.6 Indicateur d'erreur

Afin de pouvoir stopper l'algorithme, il faut définir un indicateur d'erreur. La solution $\mathbf{s_{ref}}$ étant à l'intersection entre $\mathbf{A_d}$ et Γ, la distance entre $\hat{\mathbf{s}}_{n+1/2}$ et $\mathbf{s}_n$ est un bon indicateur d'erreur qui permettra de vérifier la convergence de l'algorithme [Néron et Ladevèze, 2010] :

$$\eta = \frac{\|\hat{\mathbf{s}}_{n+1/2} - \mathbf{s}_n\|}{\frac{1}{2}\|\hat{\mathbf{s}}_{n+1/2} + \mathbf{s}_n\|} \tag{2.18}$$

avec :

$$\|\mathbf{s}\|^2 = \frac{1}{2}\int_{I\times\Omega} (\dot{\boldsymbol{\varepsilon}}_p \circ \mathbf{H}^{-1}\dot{\boldsymbol{\varepsilon}}_p + \boldsymbol{\sigma} \circ \mathbf{H}\boldsymbol{\sigma}) d\Omega dt \tag{2.19}$$

3 Proper Generalized Decomposition et méthode LATIN

3.1 Description de l'approximation

Afin de résoudre le problème (2.16) à l'aide de la PGD, il faut modifier l'écriture de celui-ci. A l'itération $n+1$, le problème n'est plus de trouver $\mathbf{s}_{n+1} \in \mathbf{A_d}$, mais de trouver $\Delta\mathbf{s}$, terme correctif de la solution $\mathbf{s}_n$, tel que $\mathbf{s}_{n+1} = (\dot{\boldsymbol{\varepsilon}}_{p_{n+1}}, \boldsymbol{\sigma}_{n+1}) = \mathbf{s}_n + \Delta\mathbf{s}$.

Si la solution initiale $\mathbf{s}_0$ (souvent choisie comme la solution du problème en élasticité linéaire) appartient à $\mathbf{A_d}$, alors tous les termes correctifs $\Delta\mathbf{s} = (\Delta\dot{\boldsymbol{\varepsilon}}_p, \Delta\boldsymbol{\sigma})$ sont recherchés dans l'espace $\mathbf{A_d^\star}$, qui correspond à l'espace $\mathbf{A_d}$ avec des conditions homogènes. Avec l'hypothèse $\mathbf{H}^- = \mathbf{H}$ où $\mathbf{H}$ est un opérateur symétrique définie positif, la direction de recherche $\mathbf{E}^-$ définie par l'équation (2.15) devient :

$$\Delta\dot{\boldsymbol{\varepsilon}}_p - \mathbf{H}\Delta\boldsymbol{\sigma} - \Delta\hat{\mathbf{a}} = 0 \tag{2.20}$$

avec $\Delta\hat{\mathbf{a}} = (\hat{\dot{\boldsymbol{\varepsilon}}}_p - \dot{\boldsymbol{\varepsilon}}_{p_n}) - \mathbf{H}(\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}_n)$ qui est connue à cette étape. Cette équation peut-être réécrite comme un problème de minimisation :

$$\Delta\mathbf{s} = \mathrm{Arg}\min_{\Delta\mathbf{s}\in\mathbf{A_d^\star}} e_{CR}^2(\Delta\mathbf{s}) = \|\Delta\dot{\boldsymbol{\varepsilon}}_p - \mathbf{H}\Delta\boldsymbol{\sigma} - \Delta\hat{\mathbf{a}}\|_{\mathbf{M}}^2 \tag{2.21}$$

Cette nouvelle expression du problème nécessite l'introduction d'une norme définie par :

$$\|\square\|^2_{\mathbf{M}} = \int_{I\times\Omega} \square \circ \mathbf{M}\square d\Omega dt \tag{2.22}$$

avec $\mathbf{M}$ un opérateur symétrique défini positif généralement pris tel que $\mathbf{M} = (1 - \frac{t}{T})\mathbf{H}$. La formulation (2.21) est équivalente à la formulation (2.16), le fait de chercher la solution de l'itération $n+1$ à l'aide d'un terme correctif que l'on ajoute à la solution de l'itération n n'entraine pas d'approximation de cette solution.

Il est maintenant possible d'approximer $\Delta\mathbf{s}$ en utilisant sa décomposition PGD $\Delta\check{\mathbf{s}}$, ce qui conduit aux champs approximés suivant :

$$\begin{aligned} \Delta\dot{\boldsymbol{\varepsilon}}_p(t,\underline{M}) &\simeq \Delta\check{\dot{\boldsymbol{\varepsilon}}}_p(t,\underline{M}) = \dot{a}(t)\mathbf{E}_p(\underline{M}) \\ \Delta\mathbf{f}(t,\underline{M}) &\simeq \Delta\check{\mathbf{f}}(t,\underline{M}) = b(t)\mathbf{F}(\underline{M}) \\ \Delta\dot{\boldsymbol{\varepsilon}}(t,\underline{M}) &\simeq \Delta\check{\dot{\boldsymbol{\varepsilon}}}(t,\underline{M}) = \dot{c}(t)\mathbf{E}(\underline{M}) \end{aligned} \tag{2.23}$$

Dans un but de simplification des équations qui vont suivre, dans les équations (2.23) une seule paire de fonction espace-temps a été utilisée. Il n'y a aucune difficulté a étendre ce qui va suivre au cas avec plusieurs paires.

Remarque : Les trois termes correctifs ne sont pas indépendants car $\Delta\check{\mathbf{s}}$ est recherché dans $\mathbf{A}^\star_{\mathbf{d}}$ ce qui traduit le fait que $\Delta\check{\mathbf{f}} \in \mathcal{F}$ et $\Delta\check{\dot{\boldsymbol{\varepsilon}}} = \mathbf{A}^{-1}\Delta\check{\mathbf{f}} + \Delta\check{\dot{\boldsymbol{\varepsilon}}}_p \in \mathcal{E}$. Ces deux conditions conduisent à l'équation suivante :

$$\forall\dot{\boldsymbol{\varepsilon}}^\star \in \mathcal{E},\ \int_{I\times\Omega} \dot{\boldsymbol{\varepsilon}}^\star \circ \mathbf{A}\Delta\check{\dot{\boldsymbol{\varepsilon}}}\, d\Omega dt = \int_{I\times\Omega} \dot{\boldsymbol{\varepsilon}}^\star \circ \mathbf{A}\Delta\check{\dot{\boldsymbol{\varepsilon}}}_p\, d\Omega dt \tag{2.24}$$

L'équation (2.24) montre que si a et $\mathbf{E}_p$ sont connus, alors $c = a$ et $\mathbf{E}$ peut-être obtenu à l'aide de l'équation suivante :

$$\forall\mathbf{E}^\star \in \mathcal{E}^\star,\ \int_\Omega \mathbf{E}^\star \circ \mathbf{A}(\mathbf{E} - \mathbf{E}_p)d\Omega = 0 \tag{2.25}$$

Ce problème est un problème classique, indépendant du temps, qui consiste à chercher $\mathbf{E} = [\boldsymbol{\varepsilon}(\Delta\underline{U})\ \ 0]^T$ avec $\Delta\underline{U} \in \mathcal{U}^\star$ connu :

$$\forall\underline{U}^\star \in \mathcal{U}^\star,\ \int_\Omega \boldsymbol{\varepsilon}(\underline{U}^\star) : \mathbf{K}(\boldsymbol{\varepsilon}(\Delta\underline{U}) - \boldsymbol{\varepsilon}_p)d\Omega = 0 \tag{2.26}$$

Une fois que les termes correctifs $\Delta\check{\dot{\boldsymbol{\varepsilon}}}$ et $\Delta\check{\dot{\boldsymbol{\varepsilon}}}_p$ sont connus, le dernier incrément correctif $\Delta\check{\mathbf{f}}$ est obtenu par : $b = a$ et $\mathbf{F} = \mathbf{A}(\mathbf{E} - \mathbf{E}_p)$. Donc la connaissance de $\Delta\check{\mathbf{s}}$ est entièrement liée à la connaissance des fonctions $a(t)$ et $\mathbf{E}_p$. Il est donc possible de réécrire le problème (2.21) de la façon suivante :

$$\Delta\check{\mathbf{s}} = \mathrm{Arg}\min_{\Delta\check{\mathbf{s}}\in\mathbf{A}^\star_{\mathbf{d}}} e^2_{CR}(\Delta\check{\mathbf{s}}) = \|\dot{a}\mathbf{E}_p - a\mathbf{H}\mathbf{F} - \Delta\hat{\mathbf{a}}\|^2_{\mathbf{M}} \tag{2.27}$$

qui est resolu par la méthode décrite dans le paragraphe suivant.

3.2 Résolution du problème

Pour résoudre l'équation (2.27), il est possible de chercher le minimum en fonction du temps, ce qui conduit à un système d'équations différentielles, et de chercher le minimum en fonction de l'espace, ce qui conduit à un problème spatial, comme cela est fait dans l'algorithme 1 [Néron et Ladevèze, 2010].

Données : les résultats de l'approximation précédente $\mathbf{e}_{p_n} = a_0\mathbf{E}_{p_0} + \sum_{i=1}^{n} a_i\mathbf{E}_{p_i}$ et la quantité $\Delta\hat{\mathbf{a}}$
Etape 1 : **Utilisation de la base réduite**
Les fonctions spatiales $(\mathbf{E}_{p_i})_{i=1}^{n}$ sont fixées, recherche des nouvelles fonctions $(a_i)_{i=1}^{n}$ qui minimisent e_{CR}^2;
Etape 2 : **Ajout de nouvelles fonctions**
Initialisation : $a^0(t)$ (par exemple, $f(t) = \alpha t$);
pour $k = 1$ *à* k_{max} **faire**
  Problème spatial : les fonctions $a^k(t)$ sont fixées, recherche des fonctions $\mathbf{E}_p^k$ qui minimisent e_{CR}^2;
  Problème temporel : les fonctions $\mathbf{E}_p^k$ sont fixées, recherche des fonctions $a^k(t)$ qui minimisent e_{CR}^2;
  Orthonormalisation : des fonctions d'espace par rapport à l'ancienne base de fonctions d'espace;
fin

Algorithme 1 : Génération itérative du terme correctif PGD à l'itération $n+1$

On se place à l'itération n+1, alors les fonctions $(a_i, \mathbf{E}_{p_i})_{i=1}^{n}$ sont connues et il est possible de construire $\dot{\mathbf{e}}_{p_n} = \dot{a}_0\mathbf{E}_{p_0} + \sum_{i=1}^{n} \dot{a}_i\mathbf{E}_{p_i}$ et $\mathbf{f}_n = b_0\mathbf{F}_0 + \sum_{i=1}^{n} b_i\mathbf{F}_i$. La paire initiale $(\dot{a}_0\mathbf{E}_{p_0}, b_0\mathbf{F}_0)$ appartient à $\mathbf{A_d}$ alors que les n autres termes $(\dot{a}_i\mathbf{E}_{p_i}, b_i\mathbf{F}_i)$ appartiennent à $\mathbf{A_d^\star}$. L'objectif est de construire le nouveau terme correctif $\dot{a}\mathbf{E}_p$ pour les champs primaux et le nouveau terme $b\mathbf{F}$ pour les termes duaux. Les deux grandes étapes de l'algorithme sont :
Etape 1 : Utilisation de la base réduite. Cette phase sert à construire à moindre coût une approximation de la solution en réutilisant les fonctions de l'itération précedente. Les fonctions de l'espace sont fixées et les nouvelles fonctions du temps sont obtenues à l'aide de l'équation (2.27). Ces fonctions du temps sont solutions d'un système différentiel linéaire en temps relativement petit qui peut-être résolu à l'aide de méthodes classiques
Etape 2 : Ajout de nouvelles fonctions. A l'aide de la solution de l'étape précédente une nouvelle solution approchée est recherchée. La résolution du problème de minimisation par rapport aux variables d'espace conduit à un problème spatial sur Ω indépendant du temps. La résolution du problème de minimisation par rapport a la variable temporelle conduit à un problème différentiel scalaire sur tout l'intervalle de temps I. Une fois les deux problèmes résolus, les fonctions du temps sont orthogonalisées et ajoutées à la base réduite.

Au cours de ce processus de création de la base réduite, la construction des fonctions

d'espace est l'étape la plus couteuse, il est donc intéressant de stocker ces fonctions et de les réutiliser le plus possible. C'est pourquoi, les fonctions créées à l'itération n sont réutilisées à l'itération $n+1$ et que si à la fin de l'étape 1 de l'algorithme 1 l'approximation est « suffisament bonne » l'étape 2 n'est pas réalisée (cela nécessite l'introduction d'un critère que l'on définira plus loin).

3.3 Discussion sur les coûts de calcul

La résolution du problème (2.16) par une méthode incrémentale aurait conduit à la résolution de n_I (nombre de pas de temps) problèmes spatiaux à chaque itération de la méthode LATIN. Donc l'obtention de la solution finale $\mathbf{s_{ref}}$ aurait nécessité la résolution de $n_I \times n_{iter}$ problèmes spatiaux (avec n_{iter} le nombre d'itération de la LATIN). En utilisant la PGD, l'algorithme 1 converge très vite [Néron et Ladevèze, 2010][Ladevèze *et al.*, 2009]. En pratique k_{max} est égal à 2 ou 3. Ce qui conduit donc à la résolution de $k_{max} \times n_{iter}$ problèmes spatiaux pour obtenir la solution finale $\mathbf{s_{ref}}$. L'efficacité de la PGD pour réduire les coûts de calcul dans le cadre de la méthode LATIN.

Chapitre 3

La problématique

Ce troisième chapitre permet d'introduire le problème de référence qui illustrera ce mémoire et les problèmes liés à la construction du modèle réduit PGD dans le cadre de la méthode LATIN.

Sommaire

1 Le problème de référence

Le problème de référence qui sera utilisé pour illustrer ce mémoire est un problème d'évolution thermique 1D (voir Figure 3.1). La résolution du problème se traduit de la

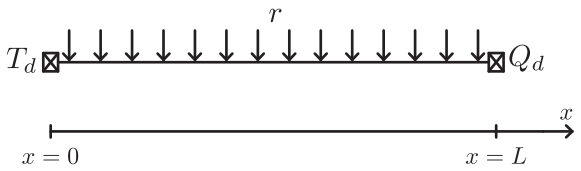


FIGURE 3.1: Description du problème d'évolution thermique 1D

façon suivante, avec T la température, $\underline{q}$ le vecteur densité de flux de chaleur, T_d la température imposée en $x = 0$, $\underline{q_d}$ le vecteur densité de flux de chaleur imposé en $x = L$, ρ la masse volumique du matériau, λ la conductivité thermique du matériau et c la chaleur spécifique massique du matériau :

$$\begin{aligned} &\rho c \frac{\partial T}{\partial t} = r - \text{div } \underline{q} \\ &T = T_d \text{ en } x = 0 \\ &\underline{q} \cdot \underline{n} = -q_d \text{ en } x = L \\ &\underline{q} = -\lambda \underline{\text{grad}}\ T \end{aligned} \tag{3.1}$$

Afin d'assurer la convergence de la méthode LATIN [Ladevèze, 1999], on pose $\underline{y} = -\underline{q}$ afin d'avoir une relation de comportement avec un opérateur positif. Cela conduit au système d'equations suivant :

$$\begin{aligned} &\rho c \frac{\partial T}{\partial t} = r + \text{div } \underline{y} \\ &T = T_d \text{ en } x = 0 \\ &\underline{y}.\underline{n} = \underline{q_d} \text{ en } x = L \\ &\underline{y} = \lambda \underline{\text{grad}}\ T \end{aligned} \tag{3.2}$$

Ce problème va être résolu à l'aide la méthode LATIN, la première étape consiste donc à séparer les équations en deux ensembles $\mathbf{A_d}$ et Γ :

$$\mathbf{A_d} : \begin{cases} \rho c \dfrac{\partial T}{\partial t} = r + \text{div } \underline{y} \\ T = T_d \text{ en } x = 0 \\ \underline{y} \cdot \underline{n} = q_d \text{ en } x = L \end{cases} \quad \text{et} \quad \Gamma : \ \underline{y} = \lambda \underline{\text{grad}}\ T \tag{3.3}$$

Il est maintenant possible d'expliciter les directions des recherches $\mathbf{E}^+$ pour l'étape locale et $\mathbf{E}^-$ pour l'étape linéaire :

$$\begin{aligned} &\mathbf{E}^+ : \mathbf{H}^+ \left(\underline{\hat{y}} - \underline{y}_n \right) + \left(\underline{\text{grad}}\ \hat{T} - \underline{\text{grad}}\ T_n \right) = 0 \\ &\mathbf{E}^- : \mathbf{H}^- \left(\underline{y}_{n+1} - \underline{\hat{y}} \right) - \left(\underline{\text{grad}}\ T_{n+1} - \underline{\text{grad}}\ \hat{T} \right) = 0 \end{aligned} \tag{3.4}$$

Dans la suite, on pose :

$$\begin{array}{l} \alpha_n = \mathbf{H}^+ \underline{y}_n + \underline{\text{grad}}\ T_n \\ \hat{\alpha} = \mathbf{H}^- \underline{\hat{y}} - \underline{\text{grad}}\ \hat{T} \end{array} \tag{3.5}$$

1.1 Etape locale

On suppose que $\left(T_n, \underline{y}_n\right) \in \mathbf{A_d}$ sont connues, on cherche alors $\left(\hat{T}, \underline{\hat{y}}\right) \in \Gamma$, c'est à dire :

$$\underline{\hat{y}} = \lambda \underline{\text{grad}}\ \hat{T} \tag{3.6}$$

et vérifiant la direction de recherche :

$$\mathbf{H}^+ \underline{\hat{y}} + \underline{\text{grad}}\ \hat{T} = \alpha_n \tag{3.7}$$

En injectant la première équation dans la seconde, on obtient :

$$\left\{ \begin{array}{l} \left[(\mathbf{H}^+)^{-1} + \lambda\right] \underline{\text{grad}}\ \hat{T} = (\mathbf{H}^+)^{-1} \alpha_n \\ \underline{\hat{y}} = \lambda \underline{\text{grad}}\ \hat{T} \end{array} \right. \tag{3.8}$$

La première équation permet d'obtenir $\underline{\text{grad}}\ \hat{T}$, donc elle permet d'obtenir $\hat{T}$. La seconde équation du système permet d'obtenir $\underline{\hat{y}}$ connaissant $\underline{\text{grad}}\ \hat{T}$. La résolution de l'étape locale est réalisée avec des grandeurs définies sur l'ensemble de l'espace et du temps, sans approximation radiale (PGD). Dans la suite, les grandeurs qui sont définies sur l'ensemble de l'espace et du temps seront appelées FULL.

1.2 Etape linéaire

On suppose que $\left(\hat{T}, \underline{\hat{y}}\right) \in \Gamma$ sont connues, on cherche alors $\left(T_{n+1}, \underline{y}_{n+1}\right) \in \mathbf{A_d}$, sous la forme $T_{n+1} = T_n + \Delta T$ et $\underline{y}_{n+1} = \underline{y}_n + \Delta \underline{y}$ avec $(\Delta T, \Delta \underline{y}) \in \mathbf{A_d^\star}$:

$$\left\{ \begin{array}{l} \rho c \Delta \dot{T} = \text{div}\ \Delta \underline{y} \\ \Delta T = 0 \text{ en } x = 0 \\ \Delta \underline{y} \cdot \underline{n} = 0 \text{ en } x = L \end{array} \right. \tag{3.9}$$

et vérifiant la direction de recherche :

$$\mathbf{H}^- \Delta \underline{y} - \underline{\text{grad}}\ \Delta T = \mathbf{H}^- \underline{\hat{y}} - \underline{\text{grad}}\ \hat{T} - \mathbf{H}^- \underline{y}_n + \underline{\text{grad}}\ T_n \tag{3.10}$$

En écrivant une formulation faible de la première équation, on obtient :

$$\forall T^\star \in \mathcal{T}^\star, \int_{I \times \Omega} \rho c \Delta \dot{T} T^\star d\Omega dt + \int_{I \times \Omega} \Delta \underline{y} \cdot \underline{\text{grad}}\ T^\star d\Omega dt = 0 \tag{3.11}$$

En combinant les équations (3.9) avec l'équation précédente et en posant $\Delta\underline{\hat{\alpha}} = \mathbf{H}^{-}\underline{\hat{y}} - \underline{\text{grad}}\,\hat{T} - \mathbf{H}^{-}\underline{y}_n + \underline{\text{grad}}\,T_n$, on obtient :

$$\forall T^\star \in \mathcal{T}^\star, \int_{I\times\Omega}\left[T^\star \rho c \Delta\dot{T} + \underline{\text{grad}}\,T^\star(\mathbf{H}^{-})^{-1}\underline{\text{grad}}\,\Delta T\right]d\Omega dt = -\int_{I\times\Omega}\underline{\text{grad}}\,T^\star(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega dt \tag{3.12}$$

On introduit maintenant la Proper Generalized Decomposition (PGD) qui consiste à écrire $\Delta T = \tau T$. Le champ test $T^\star$ va donc s'écrire $T^\star = \tau^\star T + \tau T^\star$. Le problème (3.12) consiste à chercher τ et $T \in \mathcal{F}$, tel que $\forall\tau^\star$ et $\forall T^\star \in \mathcal{F}^\star$:

$$\int_{I\times\Omega}(\tau^\star T + \tau T^\star)\rho c\dot{\tau}T + (\tau^\star\underline{\text{grad}}\,T + \tau\underline{\text{grad}}\,T^\star)(\mathbf{H}^{-})^{-1}\tau\underline{\text{grad}}\,T\,d\Omega dt = -\int_{I\times\Omega}(\tau^\star\underline{\text{grad}}\,T + \tau\underline{\text{grad}}\,T^\star)(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega dt \tag{3.13}$$

Si l'on factorise l'équation précédente à l'aide des champs tests, il vient que :

$$\begin{aligned}\int_I\tau^\star\int_\Omega &\dot{\tau}T\rho cT + \tau\underline{\text{grad}}\,T(\mathbf{H}^{-})^{-1}\underline{\text{grad}}\,T + \underline{\text{grad}}\,T(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega\,dt \\ &+\int_{I\times\Omega}T^\star\tau\dot{\tau}\rho cT + \underline{\text{grad}}\,T^\star\tau(\mathbf{H}^{-})^{-1}\tau\underline{\text{grad}}\,T + \tau(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega dt = 0\end{aligned} \tag{3.14}$$

Ce qui conduit au système d'équations suivant :

$$\begin{cases}\displaystyle\int_{I\times\Omega}T^\star\tau\dot{\tau}\rho cT + \underline{\text{grad}}\,T^\star(\tau(\mathbf{H}^{-})^{-1}\tau\underline{\text{grad}}\,T + \tau(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega dt = 0 & (a)\\ \displaystyle\dot{\tau}\int_\Omega T\rho cT\,d\Omega + \tau\int_\Omega\underline{\text{grad}}\,T(\mathbf{H}^{-})^{-1}\underline{\text{grad}}\,T\,d\Omega = -\int_\Omega\underline{\text{grad}}\,T(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega & (b)\end{cases} \tag{3.15}$$

Une méthode de point fixe, avec une initialisation par un τ aléatoire, est utilisée pour résoudre ce système (voir Figure 3.2) Le problème avec ces équations est qu'elles mé-

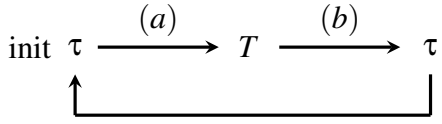


FIGURE 3.2: Principe du point fixe

langent des termes FULL avec des termes PGD ce qui n'est pas optimal. Par exemple, le second membre de l'équation (3.15)(b) est constitué de termes PGD (T) et de termes FULL ($\Delta\underline{\hat{\alpha}}$) issus de l'étape locale :

$$-\int_\Omega\underline{\text{grad}}\,T(\mathbf{H}^{-})^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega = -\int_\Omega\underline{\text{grad}}\,T(\mathbf{H}^{-})^{-1}\left[\mathbf{H}^{-}\underline{\hat{y}} - \underline{\text{grad}}\,\hat{T} - \mathbf{H}^{-}\underline{y}_n + \underline{\text{grad}}\,T_n\right]d\Omega \tag{3.16}$$

Le calcul de ce second membre oblige à reconstruire les termes PGD de façon FULL. Dans la mise en oeuvre du calcul de façon discrète, il faut ensuite sommer les termes sur chacun des éléments et chacun des points de Gauss. Cela implique un temps de calcul important des intégrales et donc pénalise le temps de calcul global de la méthode. De plus, il est impossible de calculer une fois pour toute des opérateurs réalisant les opérations. En effet lors de calculs non linéaire, il est nécessaire d'actualiser certains opérateurs comme, par exemple $\mathbf{H}^-$, pour assurer la convergence de la méthode. L'idée serait donc d'approximer les grandeurs FULL par des grandeurs à variables séparées afin de réduire les temps de calculs. Avant de pouvoir faire cela, il faut s'assurer que la méthode LATIN converge toujours malgrès l'utilisation d'un second membre approximé dans les équations. Dans la suite de ce chapitre, la version discrète de la méthode LATIN va être utilisée. Le calcul de référence sera effectué avec un découpage de l'espace en 100 points sur l'intervalle $[0,1]$ et un découpage du temps en 100 points sur l'intervalle $[0,30]$. Pour les conditions aux limites, des fonctions sinusoïdales ont été choisies :

$$\begin{aligned} T_d &= \sin\left(2\pi\left(1-\frac{t}{T}\right)\right) \\ q_d &= \sin\left(5\pi\left(1-\frac{t}{T}\right)\right) \\ r &= 10\sin\left(6\pi x\left(\frac{t}{T}\right)^3\right) \end{aligned} \tag{3.17}$$

L'initialisation de la méthode sera faite en choisissant $\left(\hat{T},\underline{\hat{y}}\right) = (0,\underline{0})$ et en cherchant un champs de température T_0 qui appartient à $\mathbf{A_d}$ et qui vérifie la direction de recherche $\mathbf{E}^-$.

2 Convergence en présence de perturbation

Dans un premier temps, il faut vérifier que la méthode LATIN converge toujours malgrès l'utilisation d'un second membre approximé dans les équations. Si l'on note Q_n le second membre de la LATIN, alors on a :

$$\mathrm{Q}_n \text{ tel que } \mathrm{T}^{\star^{\mathrm{T}}}\mathrm{Q}_n = -\int_\Omega \underline{\mathrm{grad}}\ \mathrm{T}^\star(\mathbf{H}^-)^{-1}\Delta\underline{\hat{\alpha}}\,d\Omega \tag{3.18}$$

Ce second membre de la méthode LATIN va être modifié afin d'introduire de façon artificielle une perturbation ce qui permettra de vérifier la convergence de l'algorithme avec un second membre approximé. Le niveau d'erreur est calculé à l'aide de la formule suivante :

$$\text{perturbation} = \frac{\|\mathrm{Q}_n - \mathrm{Q}_n^{\text{modifié}}\|}{\frac{1}{2}\|\mathrm{Q}_n + \mathrm{Q}_n^{\text{modifié}}\|} \tag{3.19}$$

Dans la suite de ce mémoire, lorsque que l'on tracera une courbe d'erreur, l'erreur qui sera calculée et la différence entre une solution de référence et la solution issue de la méthode LATIN (formule (3.20)). La solution de référence est une solution calculée par

une méthode directe avec un maillage fin. Dans les 2 cas, les maillages sont identiques et supposés suffisament fin pour ne pas influencer la convergence de la méthode LATIN.

$$erreur = \frac{\|S_{exacte} - S_{LATIN}\|}{\frac{1}{2}\|S_{exacte} + S_{LATIN}\|} \tag{3.20}$$

Sur la Figure 3.3, l'évolution du taux de convergence de la méthode est tracée pour différents niveaux de perturbation sur le second membre. Malgré l'introduction d'un second membre modifié dans la méthode, celle-ci converge toujours. Par contre, le taux de convergence se dégrade quand la perturbation augmente. Il est donc possible d'envisager un compromis avec l'introduction dans la méthode d'un second membre approximé qui vient dégrader le taux de convergence dans la méthode mais qui permet de réduire les temps de calculs.

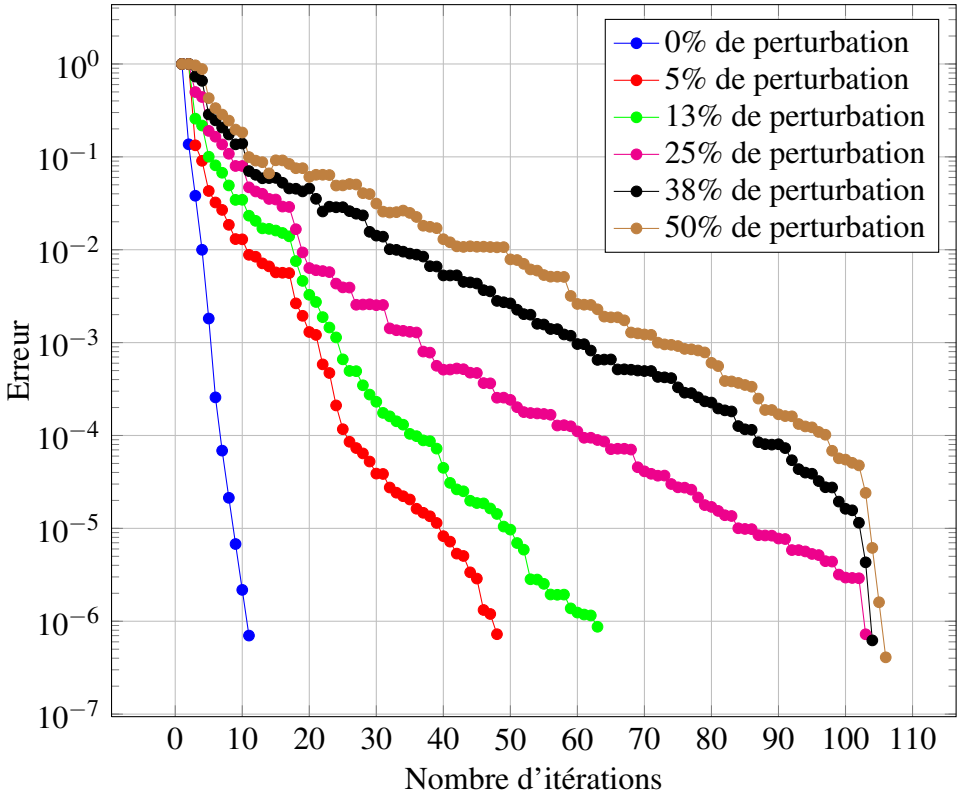


FIGURE 3.3: Graphique de l'évolution du nombre d'itérations de la méthode LATIN en fonction de la perturbation ajoutée sur le second membre

3 Approximation du second membre par la POD

Une première idée d'approximation qui peut être envisagée est l'utilisation d'une POD tronquée du second membre. En effet, l'utilisation de la POD permet d'avoir une écriture

en produits de fonctions à variables séparées qui s'adapte au formalise de l'étape linéaire de la LATIN. Dans notre cas, il faudrait calculer la POD de chacune des quantités issues de l'étape locale $(\hat{\cdot})$ afin de pouvoir écrire toutes les grandeurs comme des grandeurs à variables séparées. Dans la suite, afin de vérifier le bon fonctionnement de la POD dans le cadre de la méthode LATIN, l'approximation par la POD sera réalisée sur le second membre Q_n, c'est à dire sur le second membre déjà calculé.

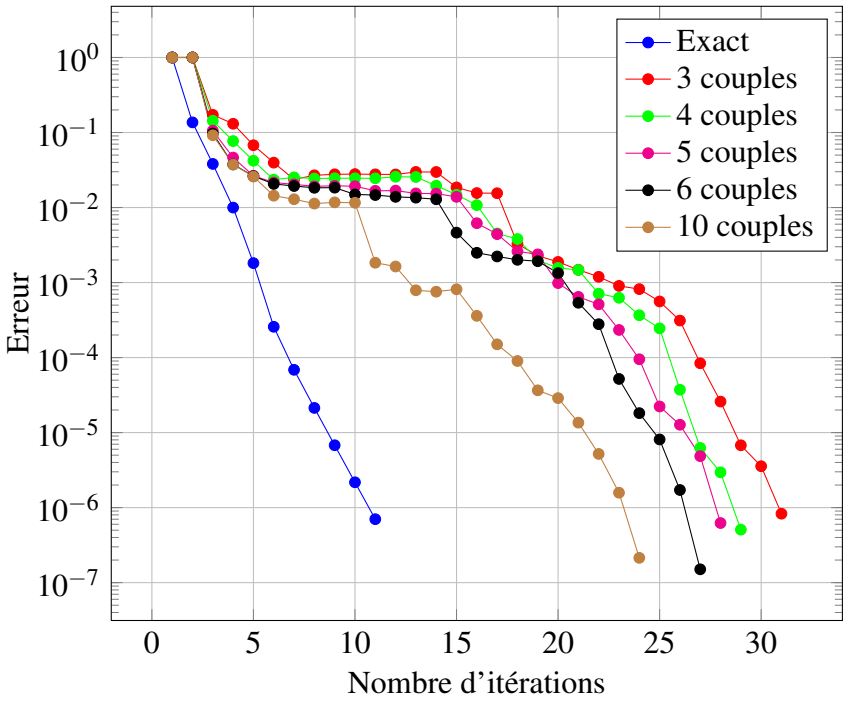


FIGURE 3.4: Graphique de l'évolution du nombre d'itérations de la méthode LATIN en fonction du nombre de couple dans l'approximation du second membre

L'évolution de l'erreur, pour différents niveaux d'approximation de la POD, est tracée en fonction du nombre d'itération de la LATIN sur la Figure 3.4. Tout d'abord, avec 1 couple et 2 couples la méthode ne converge pas. Plus exactement, l'algorithme s'arrête à cause d'un mauvais conditionnement des opérateurs. Par contre, à partir de 3 couples la méthode converge et plus on augmente le nombre de couples plus le taux de convergence s'approche de celui de la méthode sans approximation du second membre. Ces résultats sont intéressants mais le calcul de la POD du second membre sur l'ensemble de l'espace et du temps coûte cher et l'on ne sait pas a priori le nombre de couples à choisir.

Une seconde idée est de découper le domaine $I \times \Omega$ en plusieurs patchs espace-temps afin de réduire la taille du domaine sur lequel on réalise la POD et de ne conserver qu'un seul couple sur chacun des patchs. Cela permet de diminuer le coût de calcul de la POD mais pose un problème de discontinuité entre les patchs. Il faut donc vérifier que l'introduction d'une discontinuité entre les patchs ne fait pas diverger la méthode.

L'évolution de l'erreur, pour différents nombres de patchs, est tracée en fonction du

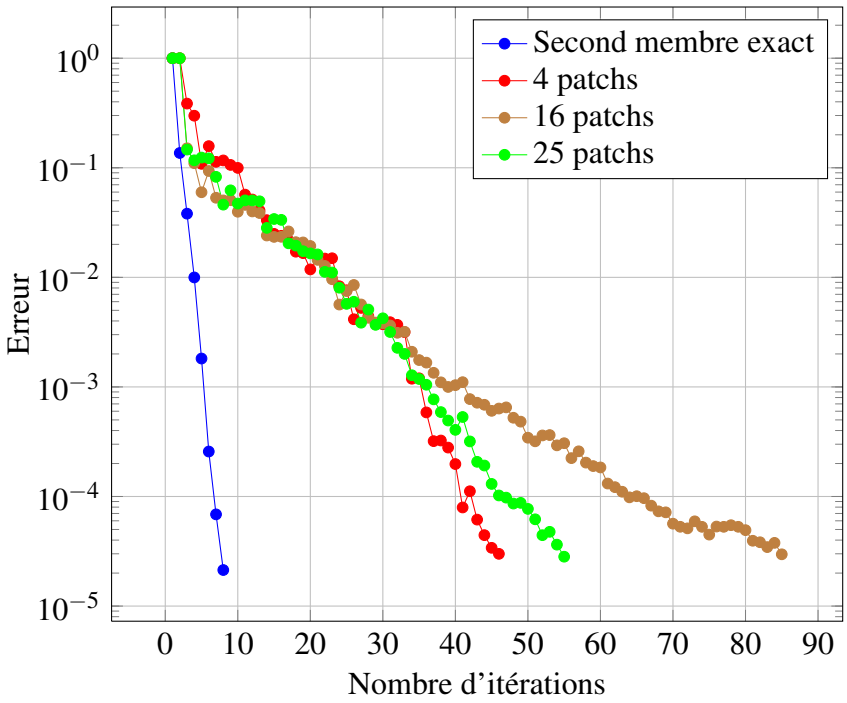


FIGURE 3.5: Graphique de l'évolution du nombre d'itérations de la méthode LATIN en fonction du nombre de patchs dans l'approximation du second membre

nombre d'itération de la LATIN sur la Figure 3.5. La méthode LATIN converge dans tous les cas, mais l'on s'aperçoit que les résultats sont peu prévisibles. En effet, le fait d'augmenter le nombre de patchs ne conduit pas forcément à un meilleur taux de convergence. Cela est peut-être lié au fait que l'on introduit des discontinuités entre les patchs et que suivant la position de ces discontinuités le taux de convergence varie plus ou moins. Néanmoins, la division de l'ensemble du domaine espace-temps en patchs semble efficace.

4 Conclusion

Toutes ces méthodes basées sur la POD permettent d'obtenir un second membre avec une écriture sous forme PGD qui est bien adaptée au formalisme de l'étape linéaire de la méthode LATIN. Par contre, ces méthodes coûtent chères, surtout s'il est nécessaire de réaliser une POD à chaque étape sur chacune des grandeurs de l'étape locale. De plus le nombre de couples nécessaires pour obtenir un niveau d'approximation donné n'est pas connu a priori, ce qui complique encore la tâche.

Dans le chapitre suivant, de nouvelles méthodes d'approximation du second membre vont être dévéloppées afin de proposer une solution efficace pour le calcul du second membre. Ces nouvelles méthodes reposent sur l'approximation du second membre à l'aide de l'évaluation de celui-ci en un nombre limité de points sur un patchs.

Chapitre 4

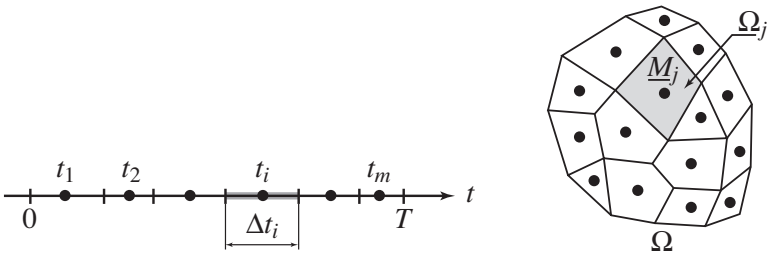
Nouvelles méthodes d'approximation

Ce quatrième chapitre introduit de nouvelles méthodes d'approximation de fonctions qui permettront à terme de transformer les termes FULL de la LATIN en des termes à variables séparées

Sommaire

1 Définitions

Dans un premier temps on définit les "points de référence" et les "instants de référence". Soit $\{I_i\}_{i=1...m}$ m sous intervalles de I de longueur $\{\Delta t_i\}_{i=1...m}$ (voir Figure 4.1(a)). Il est alors possible de définir les centres $\{t_i\}_{i=1...m}$ de ces sous intervalles appelés "instants de référence".



(a) Découpage de l'intervalle de temps

(b) Découpage de l'espace Ω

FIGURE 4.1: Représentation graphique des différentes notions

Pour l'espace Ω, on introduit m' points $\{M_j\}_{j=1...m'}$ et la partition de Ω en $\{\Omega_j\}_{j=1...m'}$ comme sur la Figure 4.1(b). Ces points sont appelés "points de référence" et la mesure de ces sous domaines est notée $\{\omega_j\}_{j=1...m'}$.

Le choix de ces points de référence n'est pas lié aux discrétisations classiques de I et de Ω. Le but est de représenté le champ f défini sur l'intervalle de temps I et sur le domaine Ω de la façon suivante :

$$\hat{a}_i^j(t) = \begin{cases} f(t,\underline{M}_j) & \text{si } t \in I_i \\ 0 & \text{sinon} \end{cases} \quad \text{and} \quad \hat{b}_i^j(\underline{M}) = \begin{cases} f(t_i,\underline{M}) & \text{si } \underline{M} \in \Omega_j \\ 0 & \text{sinon} \end{cases} \tag{4.1}$$

avec $i = 1...m$ et $j = 1...m'$. L'ensemble des couples $\{(\hat{a}_i^j, \hat{b}_i^j)\}_{i=1...m}^{j=1...m'}$ est appelé « coordonnées génaralisées de f ».

Dans ce qui va suivre l'idée est de mettre en place des outils permettant la reconstruction de la fonction f à partir des coordonnées généralisés de f.

2 Méthode privilégiant le temps

La méthode fut introduite dans [Néron et Ladevèze, 2010]. Le but est de construire une approximation $\tilde{f}$ de f sur chacun des patchs espace-temps $\Omega_j \times I_i$ de la façon suivante :

$$\tilde{f}(t,\underline{M}) = a_i^j(t) b_i^j(\underline{M}) \quad \forall (t,\underline{M}) \in I_i \times \Omega_j \tag{4.2}$$

avec les $\{(a_i^j, b_i^j)\}_{i=1...m}^{i=1...m'}$ définies à partir des $\{(\hat{a}_i^j, \hat{b}_i^j)\}_{i=1...m}^{i=1...m'}$.

Le choix qui a été fait est de donner un rôle particulier au temps car il y a beaucoup plus de degrés de liberté spatiaux que de degrés de liberté temporels. La fonction $\tilde{f}$ est alors définie par :

$$\tilde{f}(t,\underline{M}) = a_i(t)b_i(\underline{M}) \quad \forall (t,\underline{M}) \in I_i \times \Omega \tag{4.3}$$

Des produits scalaires sont introduits :

$$\langle f,g\rangle_{I_i} = \int_{I_i} fg dt \quad \text{et} \quad \langle f,g\rangle_{\Omega_j} = \int_{\Omega_j} fg d\Omega \tag{4.4}$$

Ce qui permet d'expliciter la fonction $J(a_i,b_i)$ qu'il faut minimiser afin d'obtenir $\{(a_i^j,b_i^j)\}_{i=1\ldots m}$:

$$J(a_i,b_i) = \sum_{j=1}^{m'} \left[\omega_j \|\hat{a}_i^j(t) - a_i(t)b_i(\underline{M}_j)\|_{I_i}^2 + \Delta t_i \|\hat{b}_i^j(\underline{M}) - a_i(t_i)b_i(\underline{M})\|_{\Omega_j}^2 \right] \tag{4.5}$$

Ce qui conduit aux résultats suivant :

$$a_i(t) = \frac{\sum_{j=1}^{m'} \omega_j \hat{a}_i^j(t) b_i(\underline{M}_j)}{\sum_{j=1}^{m'} \omega_j b_i^2(\underline{M}_j)} \quad \text{et} \quad b_i(\underline{M}) = \frac{\sum_{j=1}^{m'} \hat{b}_i^j(\underline{M})}{m' a_i(t_i)} \tag{4.6}$$

La reconstruction de f s'écrit donc :

$$f(t,\underline{M}) \approx \tilde{f}(t,\underline{M}) = a_i(t)b_i(\underline{M}) = \frac{\sum_{k=1}^{m'} \omega_j \hat{a}_i^k(t)\hat{a}_i^k(t_i)}{\sum_{k=1}^{m'} \omega_j \hat{a}_i^k(t_i)\hat{a}_i^k(t_i)} \hat{b}_i^j(\underline{M}) \quad \forall (t,\underline{M}) \in I_i \times \Omega_j \tag{4.7}$$

3 Méthode par patchs

La méthode par patchs est issue de ce stage. Elle est basée sur le même principe que la méthode précédente mais ne privilégie, ni le temps, ni l'espace. L'idée est de travailler sur chacun des patchs indépendement des autres.

Le but est de reconstruire f sur chacun des patchs espace-temps $\Omega_j \times I_i$ de la façon suivante :

$$\tilde{f}(t,\underline{M}) = a_i^j(t)b_i^j(\underline{M}) \quad \forall (t,\underline{M}) \in I_i \times \Omega_j \tag{4.8}$$

avec les $\{(a_i^j,b_i^j)\}_{i=1\ldots m}^{i=1\ldots m'}$ définies à partir des $\{(\hat{a}_i^j,\hat{b}_i^j)\}_{i=1\ldots m}^{i=1\ldots m'}$.

Avec les mêmes produits scalaires que précédemment, les couples $\{(a_i^j,b_i^j)\}_{i=1\ldots m}^{i=1\ldots m'}$ sont solutions de la minimisation de la fonctionnelle $J(a_i^j,b_i^j)$:

$$J(a_i^j,b_i^j) = \omega_j \|\hat{a}_i^j(t) - a_i^j(t)b_i^j(\underline{M}_j)\|_{I_i}^2 + \Delta t_i \|\hat{b}_i^j(\underline{M}) - a_i^j(t_i)b_i^j(\underline{M})\|_{\Omega_j}^2 \tag{4.9}$$

Ce qui conduit aux résultats suivant :

$$a_i^j(t) = \frac{\hat{a}_i^j(t)}{b_i^j(\underline{M}_j)} \quad \text{et} \quad b_i^j(\underline{M}) = \frac{\hat{b}_i^j(\underline{M})}{a_i^j(t_i)} \tag{4.10}$$

La reconstruction de f s'écrit donc :

$$f(t,\underline{M}) \approx \tilde{f}(t,\underline{M}) = a_i^j(t)b_i^j(\underline{M}) = \hat{a}_i^j(t)\frac{\hat{b}_i^j(\underline{M})}{\hat{b}_i^j(\underline{M}_j)} \qquad \forall (t,\underline{M}) \in I_i \times \Omega_j \tag{4.11}$$

4 Comparaison des reconstructions

Dans la suite, on notera f_{alg} la fonction f reconstruite à l'aide des méthodes introduites précédemment. Pour comparer les méthodes de reconstruction entre elle, il est nécessaire d'introduire un critère d'erreur :

$$erreur = \frac{\|f - f_{alg}\|}{\frac{1}{2}\|f + f_{alg}\|} \tag{4.12}$$

Dans ce qui va suivre, les fonctions seront choisies comme dépendantes du temps et de l'espace. La figure 4.2 permet de rappeler les définitions en 2D des "points de référence" et des "instants de référence", ainsi que les coordonnées généralisés d'une fonction.

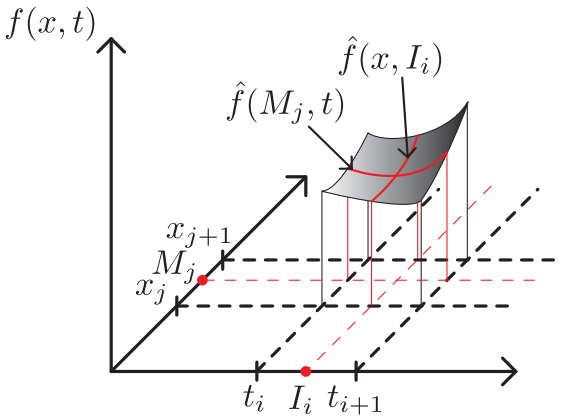


FIGURE 4.2: Illustration des "points de référence" et des "instants de référence"

4.1 Une fonction gentille

La première fonction qui sera étudiée dans ce comparatif est la fonction :

$$f(x,t) = e^{-|(x-0.5)(t-1)|} + \sin(xt) \qquad \forall x \in [0,1] \text{ et } \forall t \in [0,2] \tag{4.13}$$

Sur la Figure 4.4, la reconstruction de la fonction est réalisée avec 5 instants de référence et 5 points de référence. Cela n'a pas encore été dit, mais les deux méthodes introduisent des discontinuités lors de la reconstruction : la méthode privilégiant le temps est discontinue en espace et la méthode par patchs est discontinue en temps et en espace. Ces discontinuités pourront être un problème à l'avenir pour l'utilisation de ces méthodes dans la méthode LATIN. Sur cette exemple, il est possible de constater que la méthode par patchs semble être plus efficace que la méthode privilégiant le temps pour reconstruire la fonction. Cette affirmation est confirmée par la Figure 4.5, sur laquelle l'erreur (4.12) est représentée en fonction du nombre de points et d'instants de référence (qui sont égaux).

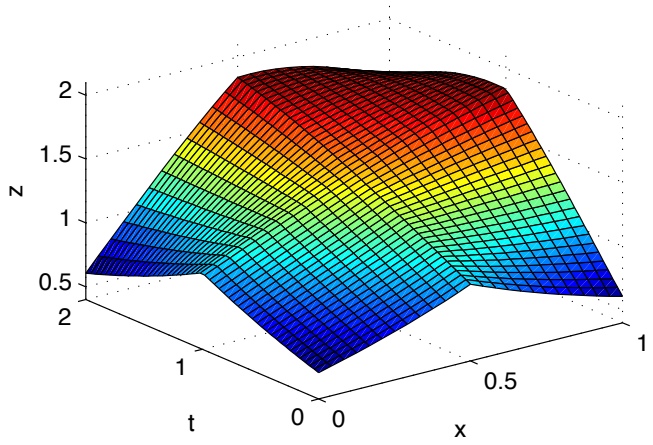
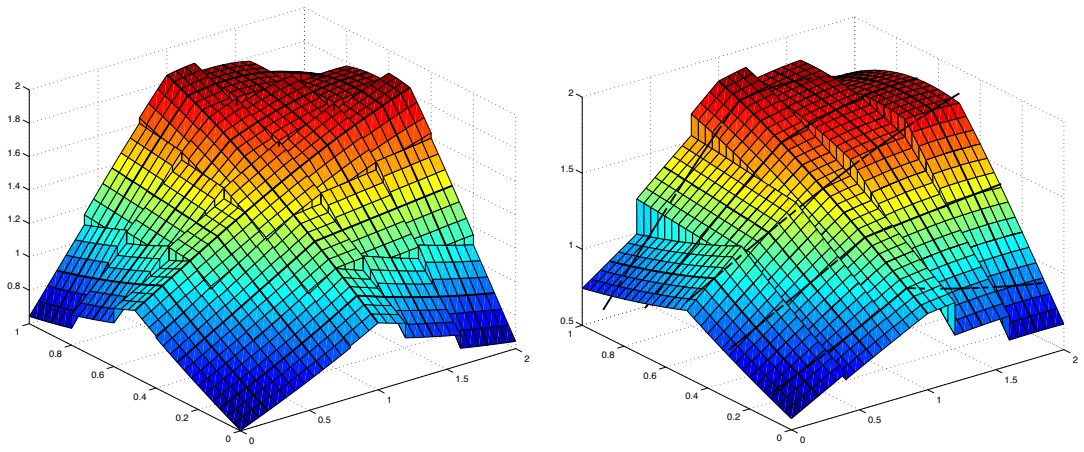


FIGURE 4.3: Représentation graphique de la première fonction



(a) Méthode par patchs

(b) Méthode privilégiant le temps

FIGURE 4.4: Représentation graphique des différentes reconstructions avec en noir les coordonnées généralisées

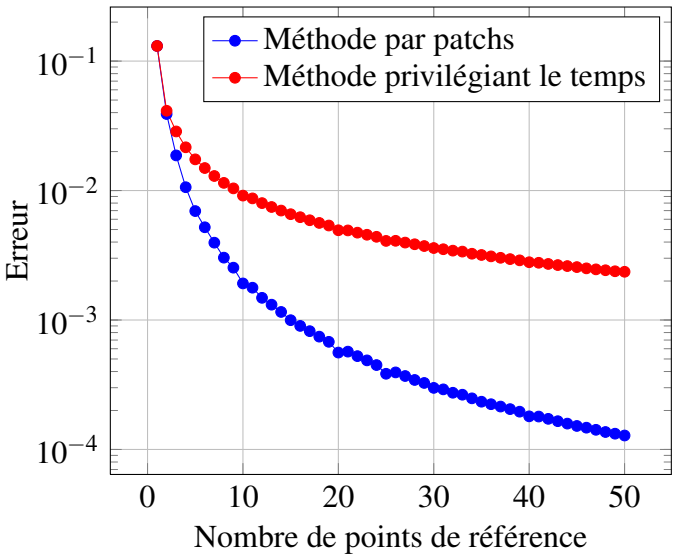


FIGURE 4.5: Erreur de reconstruction de la fonction en fonction du nombre de points de référence pour chacune des méthodes

L'étude de cette fonction assez simple a permis de mettre en évidence le potentiel de ces deux méthodes d'approximation. La méthode d'approximation par patchs semble être plus efficace que la méthode privilégiant le temps, mais celle-ce introduit plus de discontinuités ce qui pourra être un problème.

4.2 Une fonction "semblable" aux seconds membres de la LATIN

La seconde fonction qui sera étudié dans ce comparatif entre les méthodes est la fonction :

$$f(x,t) = \sin(7xt) \qquad \forall x \in [0,1] \text{ et } \forall t \in [0,2] \tag{4.14}$$

Sur la Figure 4.7, l'erreur (4.12) est représentée en fonction du nombre de points et d'instants de référence (qui sont égaux). Les résultats sont en contradiction avec ceux de la partie précédente. La méthode privilégiant le temps semble plus efficace que la méthode par patchs. Cela semble provenir du fait que la méthode par patchs a un comportement "imprévisible". En effet, si l'on s'intéresse par exemple aux cas avec 32 et 34 points de référence, on voit que l'erreur augmente alors qu'il y a plus de points de référence. Afin d'analyser ces résultats, sur la Figure 4.8 la représentation graphique de la fonction reconstruite avec 32 points de référence est tracée à gauche et la representation graphique de la fonction reconstruite avec 34 points de référence est tracée à droite. On constate que la méthode par patchs rencontre des problèmes dans certaines zones où l'on aperçoit des pics. Ces zones se caractérisent par deux choses : un gradient important et une valeur de la fonction au centre du patch proche de zéro.

Ce paragraphe a confirmé le potentiel de la méthode privilégiant le temps et a mis en

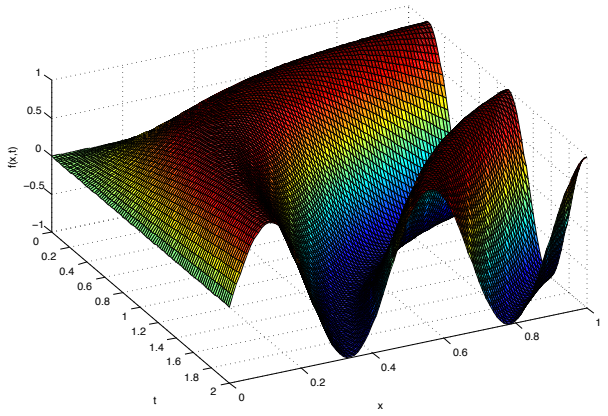


FIGURE 4.6: Représentation graphique de la seconde fonction

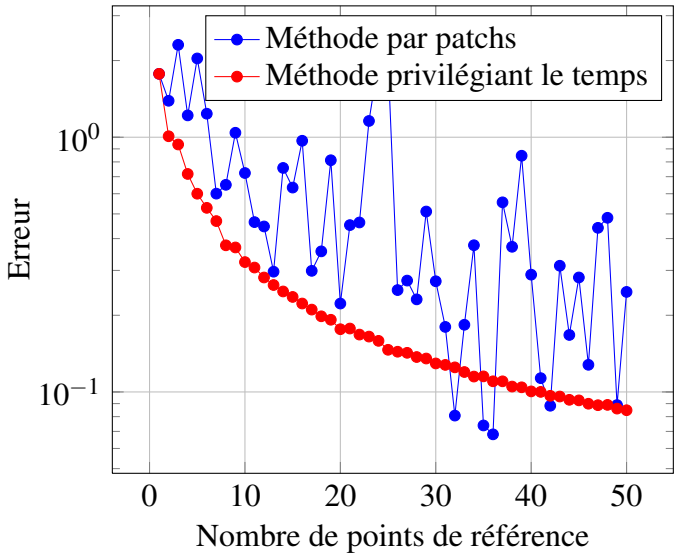


FIGURE 4.7: Erreur de reconstruction de la fonction en fonction du nombre de points de référence pour chacune des méthodes

avant un problème avec la méthode par patchs. Dans le paragraphe suivant une fonction plus simple va être utilisée afin de comprendre le problème que rencontre la méthode par patchs lors de la reconstruction de la fonction.

4.3 Compréhension du problème sur un cas simple

La fonction qui sera étudiée pour comprendre le phénomène des pics est :

$$f(x,t) = (x-0,5)^2 + (t-1)^2 + 0,001 \qquad \forall x \in [0,1] \text{ et } \forall t \in [0,2]$$

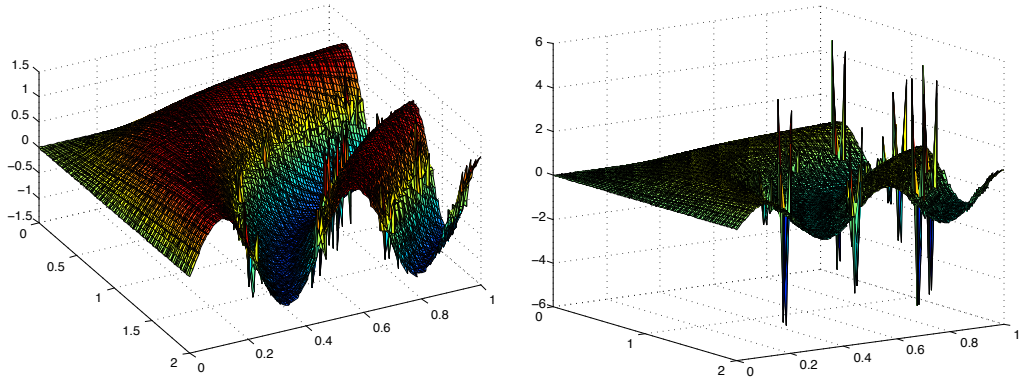


FIGURE 4.8: A gauche la fonction reconstruite avec 32 points de référence et à droite la fonction reconstruite avec 34 points de référence

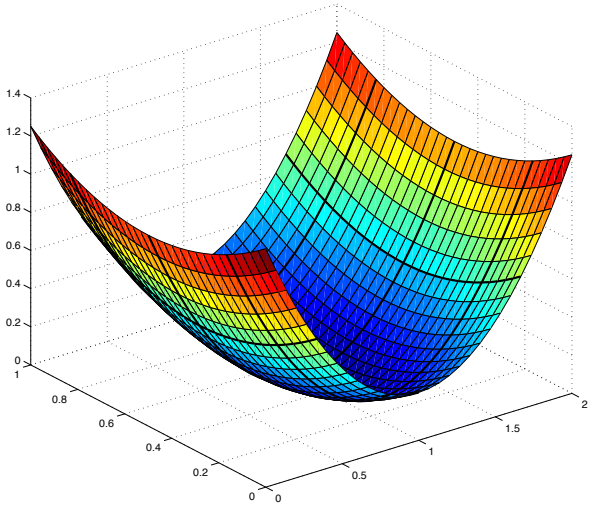


FIGURE 4.9: Représentation graphique de la fonction

La reconstruction de la fonction avec trois points de références est visible sur la Figure 4.10. Sur cette reconstruction, il n'y a que le patch central qui pose problème. Or sur ce patch, la valeur du point central est proche de zéro. Ce qui vient confirmer les constations du paragraphe précédent.

Si l'on s'intéresse à la formule de reconstruction de la fonction(4.15), on s'aperçoit que la valeur du point central du patch est au dénominateur.

$$f(t,\underline{M}) = a_i^j(t)b_i^j(\underline{M}) = \hat{a}_i^j(t)\frac{\hat{b}_i^j(\underline{M})}{\hat{b}_i^j(\underline{M}_j)} \qquad \forall (t,\underline{M}) \in I_i \times \Omega_j \tag{4.15}$$

Cette écriture n'est donc possible que si $\hat{b}_i^j(\underline{M}_j) \neq 0$ ce qui se traduit dans la mise ne œuvre numérique par $\hat{b}_i^j(\underline{M}_j)$ pas trop petit. Or, dans notre cas, $\hat{b}_2^2(\underline{M}_2) = 0,001$ ce qui explique l'erreur de reconstruction.

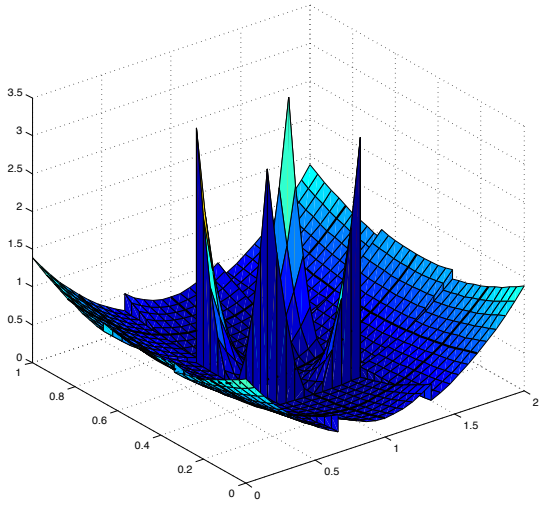


FIGURE 4.10: Représentation graphique de la fonction

Il est important de noter que la méthode de reconstruction privilégiant le temps peut également rencontrer ce genre de problème. En effet, la formule de reconstruction(4.16) présente un terme au dénominateur qui doit, lui aussi, ne pas être trop petit. Ce terme peut-être petit si tous les points centraux des patchs pour un instant de référence sont petits. La probabilité de rencontrer ce problème est plus faible mais n'est pas nulle.

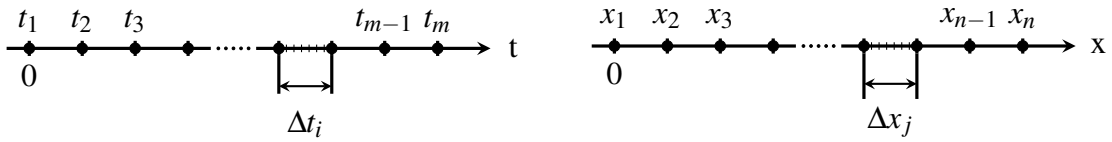
$$f(t,\underline{M}) = a_i(t)b_i(\underline{M}) = \frac{\sum_{k=1}^{m'} \omega_j \hat{a}_i^k(t)\hat{a}_i^k(t_i)}{\sum_{k=1}^{m'} \omega_j \hat{a}_i^k(t_i)\hat{a}_i^k(t_i)} \hat{b}_i^j(\underline{M}) \qquad \forall (t,\underline{M}) \in I_i \times \Omega_j \qquad (4.16)$$

4.4 Conclusion

Ces deux méthodes posent des problèmes de mise en œuvre numérique lorsque l'on souhaite représenter des fonctions qui passent par des valeurs nulles, ce qui est courant. La suite de ce chapitre a donc pour but de proposer de nouvelles méthodes permettant d'approximer les fonctions.

5 Modification des définitions

Afin de s'adapter aux méthodes qui vont être utilisées, la définition des instants et des points de référence doit être légèrement modifiée (voir Figure 4.11).



(a) Découpage de l'intervalle de temps (b) Découpage de l'espace

FIGURE 4.11: Représentation graphique des différentes notions

Si l'on considère un patch espace-temps, avec cette nouvelle définition des éléments de référence, les coordonnées généralisés de la fonction ne sont plus connus au centre du patch, mais sur les bords du patch, comme on peut le voir sur la Figure 4.12.

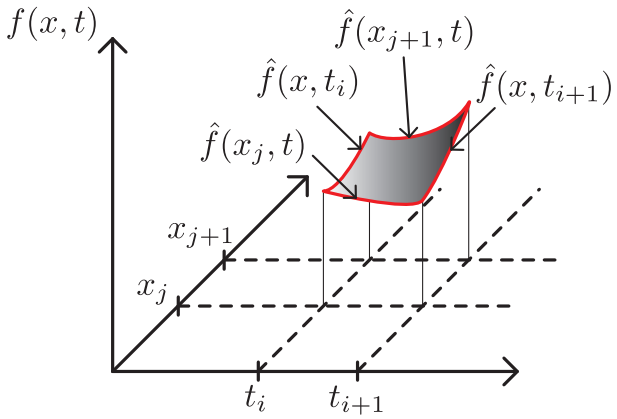


FIGURE 4.12: Illustration des "points de référence" et des "instants de référence"

Cette nouvelle définition permet d'envisager l'utilisation de nouvelles méthodes de reconstruction comme par exemple les carreaux de Coons.

6 Combinaison linéaire des bords

L'idée de cette méthode est d'utiliser une combinaison linéaire de produits des fonctions bords de chacun des patchs. En s'appuyant sur la Figure 4.12, la fonction reconstruite sur chacun des patchs est définie par $\forall x \in [x_j, x_{j+1}]$ et $\forall t \in [t_i, t_{i+1}]$:

$$\begin{aligned} f_{alg} &= \lambda_1 \hat{f}(x,ti)\hat{f}(x_j,t) + \lambda_2 \hat{f}(x,ti)\hat{f}(x_{j+1},t) \\ &\quad + \lambda_3 \hat{f}(x,t_{i+1})\hat{f}(x_j,t) + \lambda_4 \hat{f}(x,t_{i+1})\hat{f}(x_{j+1},t) \end{aligned} \tag{4.17}$$

avec $\{\lambda_i\}_{i=1..4}$ à déterminer sur chacun des patchs. Pour cela, un problème de minimisation est défini sur chacun des patchs. Il consiste à trouver la fonction f_{alg} qui minimise

l'écart à chaque coin du patch. Ce qui consiste à rechercher les $\{\lambda_i\}_{i=1..4}$ qui minimise sur chacun des patchs la fonction J suivante :

$$J(\lambda_1,\lambda_2,\lambda_3,\lambda_4) = \left[f(x_j,t_i)-f_{alg}(x_j,t_i)\right]^2+\left[f(x_{j+1},t_i)-f_{alg}(x_{j+1},t_i)\right]^2 \\ \left[f(x_j,t_{i+1})-f_{alg}(x_j,t_{i+1})\right]^2+\left[f(x_{j+1},t_{i+1})-f_{alg}(x_{j+1},t_{i+1})\right]^2 \quad (4.18)$$

Sur certains patchs, la solution de ce problème de minimisation n'est pas unique, ce qui conduit à un problème mal posé au sens d'Hadamard. Il existe plusieurs méthodes afin de régulariser les problèmes mal posés. Le choix qui a été fait ici est d'utiliser la méthode de régularisation de Tikhonov [Tikhonov et Arsénine, 1976] avec comme opérateur régularisant la matrice identité.

6.1 Choix du paramètre de régularisation de Tikhonov

Dans son ouvrage sur les méthodes de résolution de problèmes mal posée, A. Tikhonov propose plusieurs approches pour obtenir la valeur optimale du paramètre de régularisation [Tikhonov et Arsénine, 1976]. Ces différentes démarches sont basées, soit sur des méthodes d'optimisation qui demandent de résoudre plusieurs problèmes pour obtenir la valeur optimale, soit sur une analyse du bruit des données. Ces démarches ne sont donc pas adaptées.

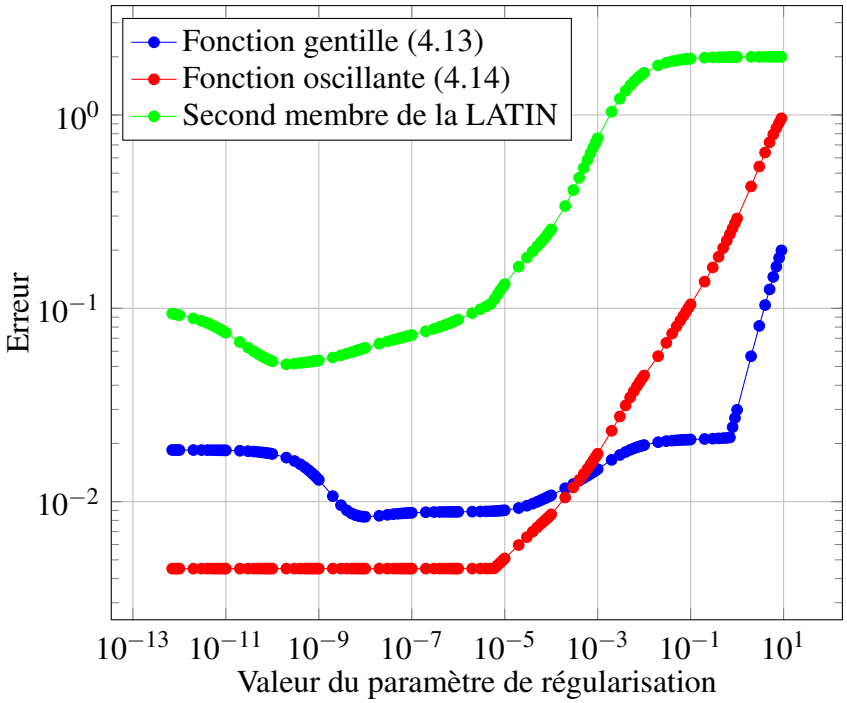


FIGURE 4.13: Erreur de reconstruction en fonction de la valeur du paramètre de Tikhonov pour différentes fonctions

La Figure 4.13, qui représente l'évolution de l'erreur (4.12) en fonction de la valeur du paramètre de régularisation pour différentes fonctions à reconstruire, permet d'affirmer que pour chacune des fonctions reconstruites, il y a une valeur optimale différente pour le paramètre de régularisation. Dans la suite, la valeur du paramètre de régularisation a été fixée à 10^{-8} qui représente un bon compromis pour une reconstruction optimale des trois fonctions.

7 Carreaux de Coons

Originalement introduit par S. Coons pour la reconstruction de surface et d'objets en CAO [Coons, 1967], ces carreaux permettent de trouver une surface qui interpole 4 courbes 2 à 2 opposées (Voir Figure 4.14). Un carreau est défini par deux paramètres u et v définies dans $[0,1]$. La frontière est définie par les quatres équations des courbes suivantes : $\hat{f}(u,0)$, $\hat{f}(u,1)$, $\hat{f}(0,v)$ et $\hat{f}(1,v)$.

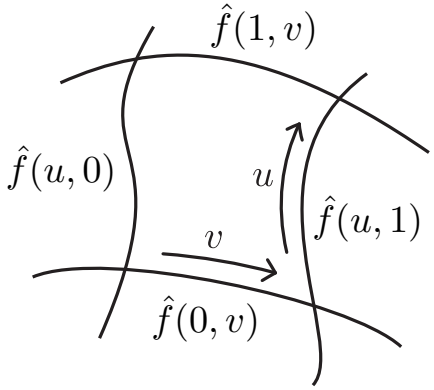


FIGURE 4.14: Définition d'un carreau de Coons

L'expression de la reconstruction sur un carreau $f_{alg}(u,v)$ est donnée par la formule suivante :

$$\begin{aligned} f_{alg}(u,v) &= \hat{f}(u,0).p_0(v)+\hat{f}(u,1).p_1(v)+\hat{f}(0,v).p_0(u)+\hat{f}(1,v).p_1(u) \\ &\quad -\hat{f}(0,0).p_0(u).p_0(v)-\hat{f}(0,1).p_0(u).p_1(v) \\ &\quad -\hat{f}(1,0).p_1(u).p_0(v)-\hat{f}(1,1).p_1(u).p_1(v) \end{aligned} \tag{4.19}$$

Les fonctions p_0 et p_1 sont des polynômes choisis en fonction des propriétés souhaitées pour la surface globale. La condition nécessaire au passage de la surface par les frontières sont : $p_0(0)=1$, $p_0(1)=0$, $p_1(0)=0$ et $p_1(1)=1$. Les polynômes $p_0(t)=1-t$ et $p_1(t)=t$ permettent de construire une surface limitées par les frontières définies. Pour qu'il y ait continuité en tangence entre les carreaux, Coons propose d'utiliser :

$$\begin{aligned} p_0(t) &= 2t^3-3t^2+1 \\ p_1(t) &= -2t^3+3t^2 \end{aligned}$$

Si une continuité en courbure entre les surfaces est souhaitée (il faut que les courbes limites soient également continues en courbure), Coons propose d'utiliser les polynômes :

$$p_0(t) = -6t^5 + 15t^4 - 10t^3 + 1$$
$$p_1(t) = 6t^5 - 15t^4 + 10t^3$$

Le choix qui a été fait ici est d'utiliser les polynômes qui permettent d'obtenir la continuité entre les surfaces : $p_0(t) = 2t^3 - 3t^2 + 1$ et $p_1(t) = -2t^3 + 3t^2$.

8 Comparaison des deux méthodes

Afin de comparer les deux méthodes, les fonctions (4.13)(4.14) ainsi qu'un second membre numérique issu de la LATIN vont être utilisés. Sur la Figure 4.15, l'erreur de reconstruction est tracée en fonction du nombre d'éléments de références (nombre d'instants de référence = nombre de points de référence = nombre d'éléments de référence).

Hormis pour la première fonction (4.13), les deux méthodes conduisent à un niveau d'erreur de reconstruction semblable pour un même nombre d'éléments de référence. Les deux méthodes présentes des avantages et des inconvénients. Les carreaux de Coons sont intéressants car ils proposent une reconstruction à moindre coût des fonctions, puisque la reconstruction est basée sur une formule explicite. La méthode basée sur les combinaisons linéaires des fonctions bords présentent comme avantage une écriture plus simple de l'expression de la fonction reconstruite, mais oblige à inverser une matrice 4x4 sur chaque patch.

Par contre, un point essentiel pour les deux méthodes est le choix du nombre d'éléments de référence. En effet, la Figure 4.15 permet de voir qu'en fonction de la complexité de la fonction à approximer, le nombre d'éléments de référence pour atteindre un niveau d'erreur donné varie. Dans la suite, une méthode de choix du nombre d'éléments de référence va être développée.

9 Choix du nombre d'éléments de référence

Pour obtenir le nombre d'éléments de référence nécessaires pour approximer la fonction, deux solutions peuvent être envisagées.

Une méthode d'analyse de l'erreur de reconstruction pourrait être envisagée. L'algorithme démarre avec un nombre d'éléments de référence faible, l'erreur de reconstruction est calculée. Si celle-ci est supérieure à une valeur cible, alors le nombre d'élements de référence est augmenté jusqu'à atteindre une erreur de reconstruction inférieure à la valeur cible. Cette méthode est longue et coûteuse, mais permet de s'assurer que l'erreur de reconstruction est bien inférieure à une valeur cible.

La seconde méthode, qui est celle qui sera utilisée par la suite, s'appuie sur la quantité d'information contenue dans la fonction que l'on souhaite approximer. Dans un premier temps, un seul patch espace-temps est considéré. Sur les bords du patch, la trace de la

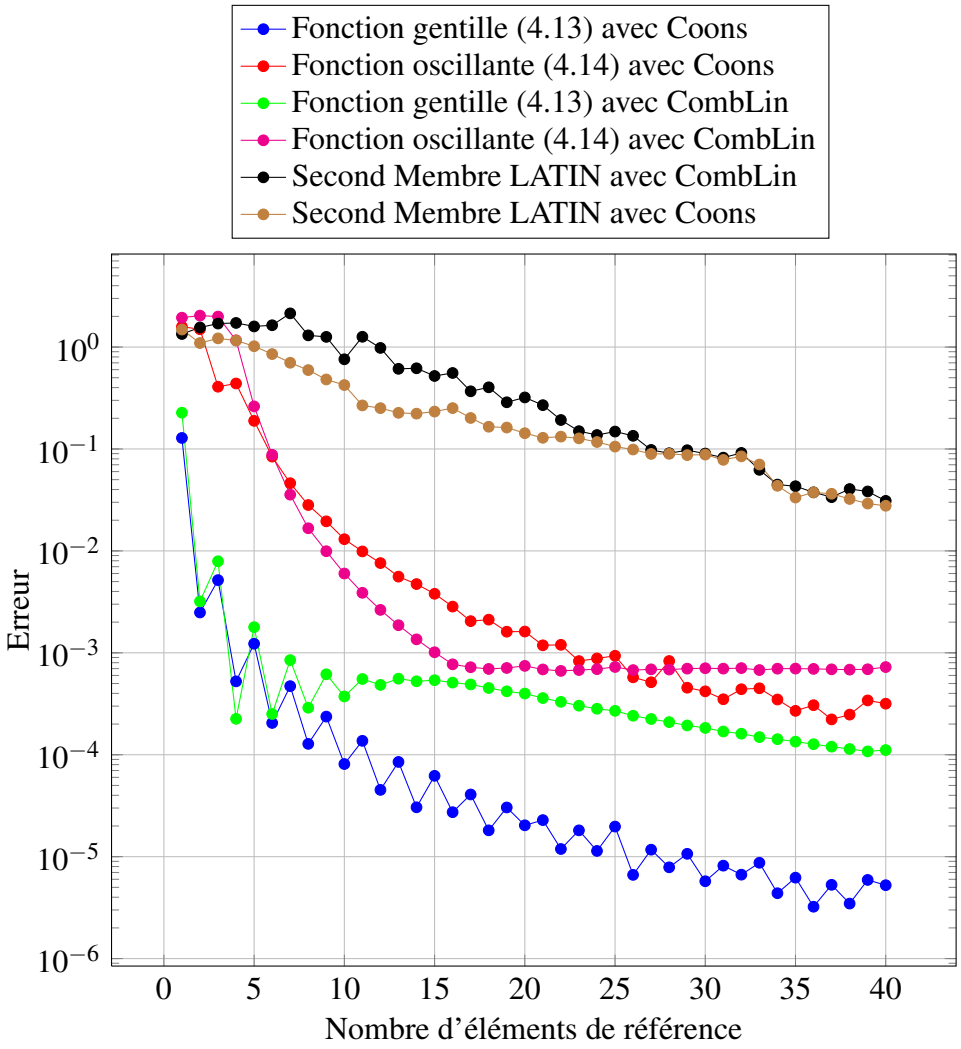
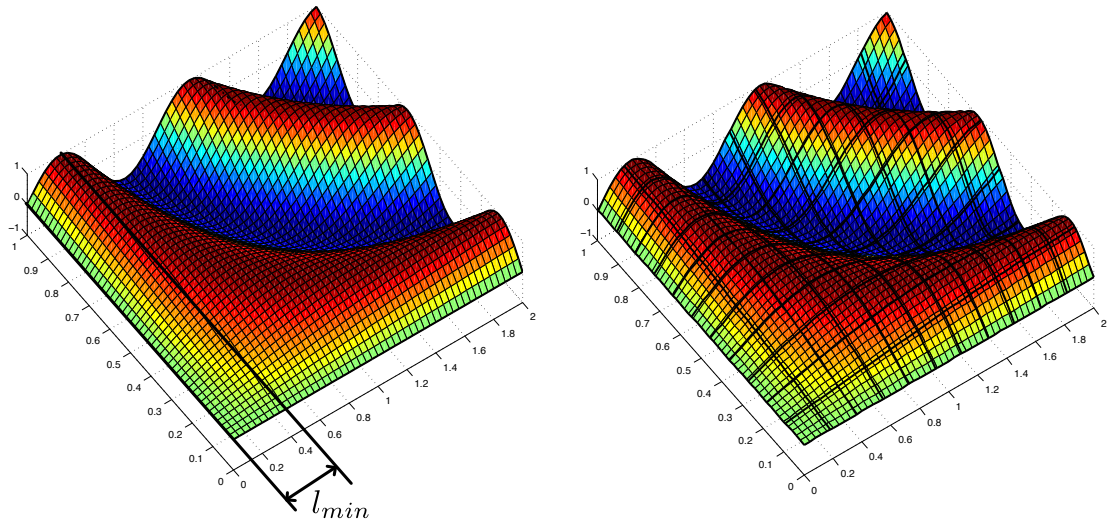


FIGURE 4.15: Erreur de reconstruction des différentes méthodes en fonction du nombre d'éléments de référence pour plusieurs fonctions

fonction est extraite. Le gradient de chacune des fonctions bords est calculé. L'idée de la méthode est de dire que la longueur maximale d'un patch doit être de l'ordre de la longueur minimal de variation du signe du gradient (Voir Figure 4.16). C'est cette longueur qui est ensuite utilisée afin de déterminer le nombre d'éléments de référence.



(a) Définition de la longueur minimale de variation du signe du gradient

(b) Résultat après découpage

FIGURE 4.16: Définition et résultats de la méthode de découpage de l'espace et du temps en éléments de référence

Les résultats de la méthode sont intéressant, car pour les fonctions (4.13)(4.14) l'erreur de reconstruction avec le choix automatique des éléments de référence est inférieure à 10^{-2}. Par contre, comme on peut le voir sur la Figure 4.16(b), dans cette première version de la méthode le découpage est uniforme ce qui n'est certainement pas optimal. Une solution envisageable serait de placer un élément de référence à chaque point de changement de signe du gradient des fonctions bords.

Chapitre 5

Intégration dans la méthode LATIN

Dans ce chapitre, les méthodes d'approximation présentées au chapitre 4 seront utilisées dans le cadre de la méthode LaTiN.

Sommaire

1 Introduction

Les deux méthodes, carreaux de Coons ou combinaison linéaire des fonctions bords, ont montré leur potentiel pour approximer des fonctions à plusieurs variables. L'avantage de la méthode de Coons est que l'obtention de la reconstruction ne nécessite pas d'inversion de systèmes, ce qui permet de gagner du temps de calcul. Par contre son extension aux fonctions à plus de deux variables n'est pas immédiate et nécessitera une réflexion de fond.

Dans la suite de ce chapitre, la méthode d'approximation des fonctions de deux variables qui sera utilisée est celle basée sur les carreaux de Coons. Des résultats similaires peuvent être obtenus avec la méthode basée sur la combinaison linéaire des fonctions bords.

2 Intégration dans la méthode LaTiN

Le problème de référence utilisé dans cette partie est celui défini au Chapitre 3 avec les mêmes discrétisations spatiale et temporelle. Le nombre d'instants et de points de référence est automatiquement déterminé à l'aide de la méthode du gradient sur le bord décrite précédement. Dans l'algorithme, le second membre après calcul des intégrales est approximé. Ces résultats sont donc une première étape afin de vérifier la convergence de la méthode.

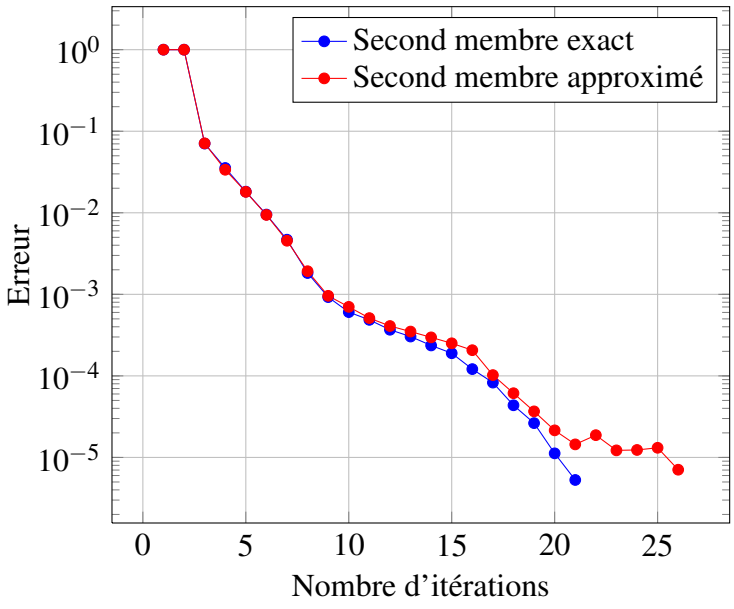


FIGURE 5.1: Graphique de convergence de la méthode LATIN avec un second membre exact ou approximé

Sur la Figure 5.1, les courbes de convergence de la méthode LATIN sont tracées pour

le cas du second membre exact et pour le cas du second membre approximé. Les deux courbes sont quasiment identiques jusqu'à un écart entre la solution exact et la solution LATIN de 10^{-4}. Ces résultats sont bons mais cela vient du fait que la méthode de détermination automatique du nombre d'éléments de référence en prend le maximum. Comme nous l'avons dit précédemment, le découpage est uniforme et est controlé par la plus petite longueur de variation du gradient. Or, comme on peut le voir sur la Figure 5.2, le second membre de la méthode LATIN présente des oscillations très rapides dans certaines zones. Ce qui conduit à un nombre d'éléments de référence important : il y a dans ce calcul 50 instants de référence et 50 points de référence pour un maillage fin de 100×100. Il faut donc envisager une évolution de la méthode de détermination automatique du nombre d'éléments de référence qui serait basée sur une répartition non uniforme de ceux-ci.

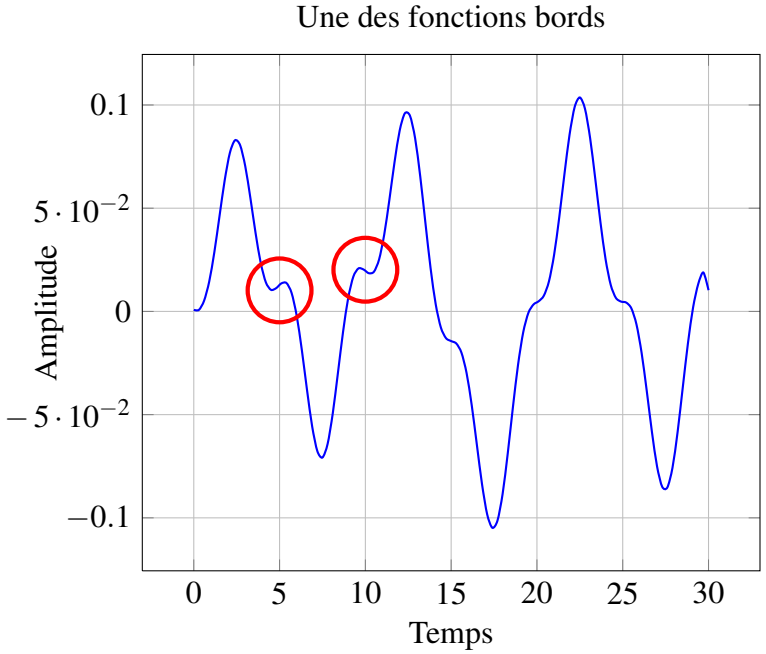


FIGURE 5.2: Graphique d'une des fonctions sur le bord permettant d'expliquer les problèmes de la méthode basée sur le gradient

3 Répartition non uniforme

Il a été montré précédement qu'une répartition non uniforme des éléments de référence est nécessaire. Pour cela, on conserve la méthode qui calcule le gradient de la fonction sur le bord du domaine espace-temps. Mais maintenant les éléments de références vont être placés à chaque point où le gradient change de signe. Si l'on applique cette méthode à un second membre de la LATIN, on obtient le résultat de la Figure 5.3. Les coordonnées généralisées sont représentées en noir et permettent de voir que dans les zones

où il y a beaucoup d'oscillations, il y a également beaucoup d'éléments de références.

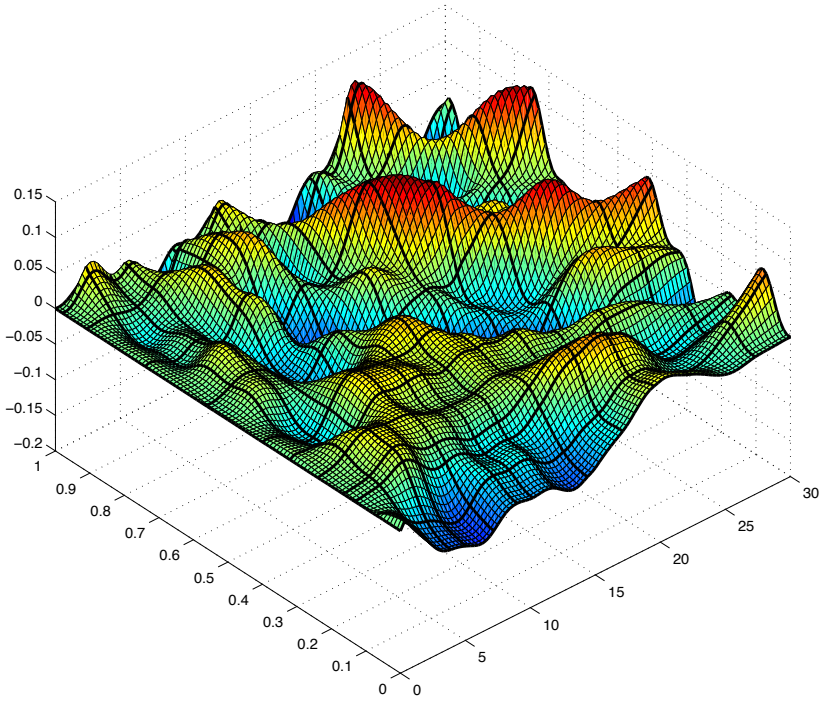


FIGURE 5.3: Répresentation d'un second membre de la méthode LATIN avec en noirs ses coordonnées généralisées

Il est maintenant possible d'intégrer cette nouvelle méthode de détermination automatique du nombre d'éléments de référence dans la LATIN. La Figure 5.4 permet de constater que le nombre d'éléments de référence choisi par la méthode automatique évolue au cours des itérations. Il ne dépasse pas 27 points de référence et 22 instants de référence, alors que précédemment, la méthode prenait 52 points de référence et 52 instants de référence. La Figure 5.5 confirme que la méthode converge. Les deux courbes sont proches jusqu'à une erreur de 10^{-3}, ce qui confirme l'efficacité de la méthode d'approximation. Le fait que la méthode approximée ait beaucoup plus de mal à atteindre un niveau d'erreur de 10^{-4} provient sans doute du fait que l'approximation n'est pas faite pour atteindre de telles niveaux d'erreur.

L'efficacité de la méthode LATIN avec approximation est confirmée par la Figure 5.6 qui permet de comparer la solution du problème de référence avec une méthode directe et la solution du problème de référence avec la méthode LATIN approximée.

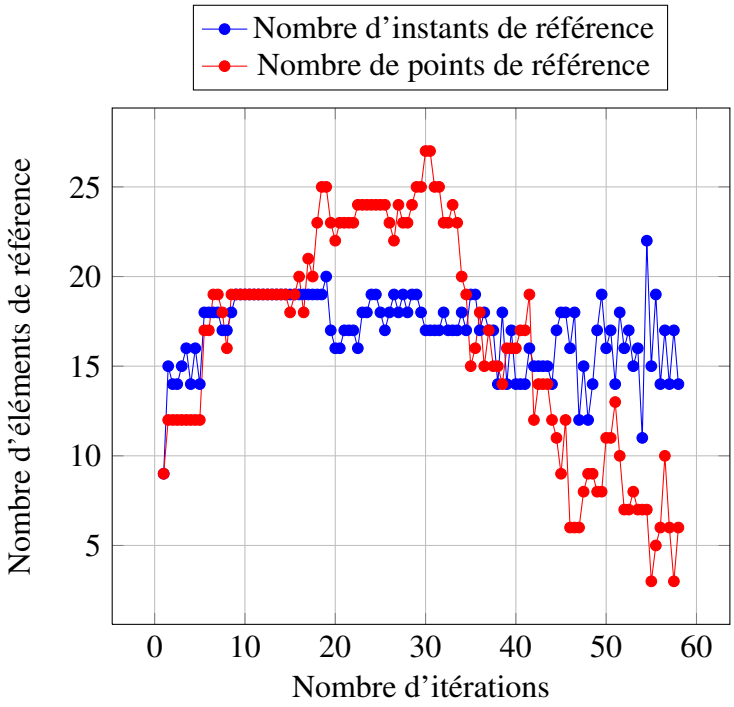


FIGURE 5.4: Graphique du nombre d'éléments de référence choisi par la méthode automatique au cours des itérations

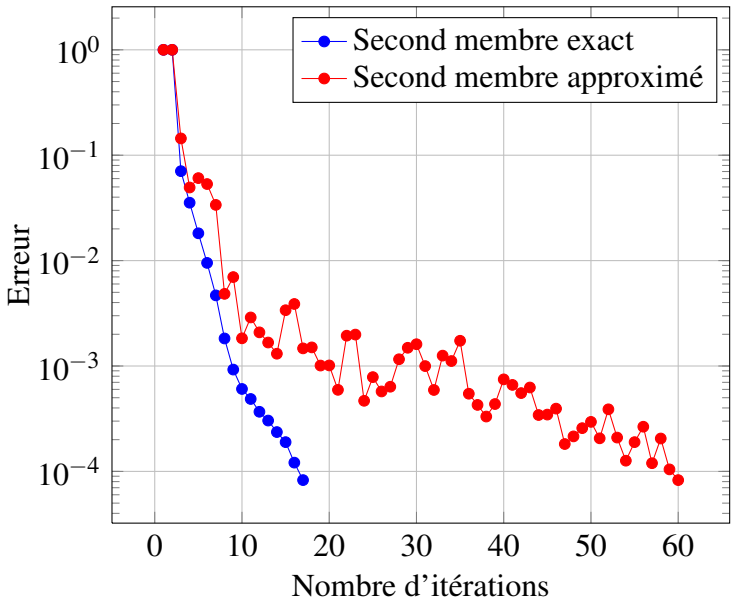


FIGURE 5.5: Graphique de convergence de la méthode LATIN avec un second membre exact ou approximé

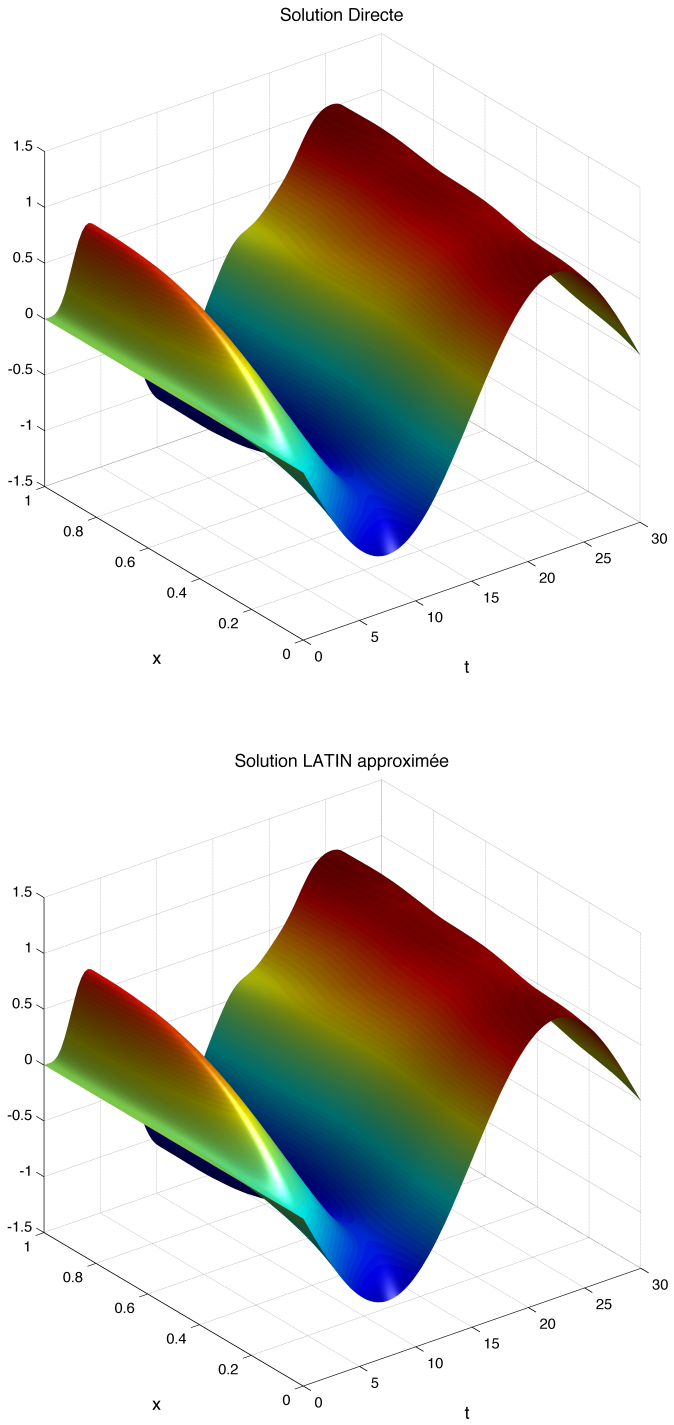


FIGURE 5.6: Solution du problème de référence par une méthode directe en haut et par la méthode LATIN approximée en bas

Conclusion

Dans ce travail, plusieurs méthodes d'approximation des fonctions ont été présentées et comparées afin d'obtenir une méthode robuste qui puisse être implémentée dans la méthode LATIN. Afin de facilité l'intégration de l'approximation dans la LATIN, une méthode de choix automatique des éléments de référence a été développée. Les premiers résultats sur le problème de thermique 1D sont prometteurs et permettent d'envisager d'importants gains en terme de temps de calcul et de stockage de données.

Afin de s'assurer de ces gains potentiels, il faudrait pousser l'approximation plus loin. Dans le cadre de ce stage, nous avons juste approximé le second membre dans son intégralité. L'étape suivante serait d'approximer chacun des termes issus de l'étape locale $\left(\hat{T},\underline{\hat{y}}\right)$, afin de pouvoir calculer le second membre uniquement avec des termes à variables séparées. Cela permettrait de s'assurer de la réduction du temps de calcul global de la méthode, grâce à la diminution du temps de calcul des opérateurs de l'étape linéaire, et ce malgré l'augmentation du nombre d'itérations.

Pour étendre la méthode à des cas 2D ou 3D, il faudra étendre la théorie des carreaux de Coons à des fonctions de plus de 2 variables afin de pouvoir reconstruire par exemple une fonction $f(x,y,z,t)$ dans le cas de problèmes 3D d'évolution.

Enfin à terme, ces méthodes d'approximation devraient conduire à l'écriture d'une « algèbre PGD », c'est à dire à une vision des opérations classiques dans le formalisme de la méthode LATIN.

Bibliographie

[Andrews et Patterson, 1975] ANDREWS, H. C. et PATTERSON, C. L. (1975). Outer product expansions and their uses in digital image processing. *The American Mathematical Monthly*, 82:1–13.

[Białecki *et al.*, 2005] BIAŁECKI, R., KASSAB, A. et FIC, A. (2005). Proper orthogonal decomposition and modal analysis for acceleration of transient FEM thermal analysis. *International Journal for Numerical Methods in Engineering*, 62:774–797.

[Chatterjee, 2000] CHATTERJEE, A. (2000). An introduction to the proper orthogonal decomposition. *Current Science*, 78:808–817.

[Chinesta *et al.*, 2008] CHINESTA, F., AMMAR, A., LEMARCHAND, F., BEAUCHENE, P. et BOUST, F. (2008). Alleviating mesh constraints : Model reduction, parallel time integration and high resolution homogenization. *Computer methods in applied mechanics and engineering*, 197:400–413.

[Coelho et Breitkopf, 2009] COELHO, R. F. et BREITKOPF, P. (2009). *Optimisation multidisciplinaire en mécanique - Tome 2, Réduction de modèles, robustesse, fiabilité, réalisations logicielles*. Hermes Science Publications.

[Coons, 1967] COONS, S. (1967). Surface for computer-aided design of space forms. Rapport technique.

[Cordier et Bergmann, 2006] CORDIER, L. et BERGMANN, M. (2006). Réduction de dynamique par décomposition orthogonale (pod). *Ecole de printemps OCET*.

[Gunzburger *et al.*, 2007] GUNZBURGER, M. D., PETERSON, J. S. et SHADID, J. S. (2007). Reduced-order modeling of time-dependant pdes with multiple parameters in the boundary data. *Computer methods in applied mechanics and engineering*, 196:1030–1047.

[Ladevèze, 1985] LADEVÈZE, P. (1985). Sur une famille d'algorithmes en mécanique des structures. *Compte rendu de l'académie des sciences*, 300(2):41–44.

[Ladevèze, 1999] LADEVÈZE, P. (1999). *Nonlinear computational structural mechanics*. Springer Verlag.

[Ladevèze *et al.*, 2009] LADEVÈZE, P., NÉRON, D. et PASSIEUX, J.-C. (2009). The LATIN multiscale computational method and the Proper Generalized Decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199:1287–1296.

[Liang *et al.*, 2002] LIANG, Y., LEE, H., LIM, S., LIN, W., LEE, K. et WU, C. (2002). Proper orthogonal decomposition and its applications - part 1 : theory. *Journal of Sound and Vibration*, 252:527–544.

[Long, 1983] LONG, C. (1983). Visualisation of matrix singular value decomposition. *Mathematics Magazine*, 56:161–167.

[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*, 196:4521–4537.

[Néron et Ladevèze, 2010] NÉRON, D. et LADEVÈZE, P. (2010). Proper generalized decomposition for multiscale and multiphysics problems. *Archives of Computational Methods in Engineering*, A paraître.

[Schmidt *et al.*, 2010] SCHMIDT, F., PIRC, N., MONGEAU, M. et CHINESTA, F. (2010). Efficient mold danscooling optimization by using model reduction. Rapport technique, Université de Toulouse.

[Tikhonov et Arsénine, 1976] TIKHONOV, A. et ARSÉNINE, V. (1976). *Méthodes de résolution de problèmes mal posés*. Edition MIR . MOSCOU.