Un framework de calcul pour la méthode des bases réduites : applications à des problèmes non-linéaires multi-physiques

Stéphane Veys

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques Appliquées**

Arrêté ministériel : du 7 août 2006

Présentée par

Stéphane Veys

Thèse dirigée par **Christophe Prud'homme**

préparée au sein **du Laboratoire Jean Kuntzmann**
et de **l'École doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Un framework de calcul pour la méthode des bases réduites : applications à des problèmes non-linéaires multi-physiques

Thèse soutenue publiquement le 26 novembre 2014 ,
devant le jury composé de :

M. Florian De Vuyst
Professeur, École Normale Supérieure de Cachan, Rapporteur
M. Damien Tromeur-Dervout
Professeur, Université Claude Bernard Lyon 1, Rapporteur
M, Frédéric Hecht
Professeur, Université Pierre et Marie Curie, Examinateur
Mme Clémentine Prieur
Professeur, Université Joseph Fourier, Examinateur
M. Christophe Prud'homme
Professeur, Université de Strasbourg, Directeur de thèse

The people who are crazy enough to think they can change the world,
are the ones who do.

Craig Tanimoto

Remerciements

Je tiens tout d'abord à remercier Christophe Prud'homme pour sa confiance, pour m'avoir proposé le sujet de ce travail de thèse ainsi que pour m'avoir encadré.

Merci à Florian De Vuyst et Damien Tromeur-Dervout d'avoir bien voulu rapporter sur ce manuscrit. Merci également à Frédéric Hecht et Clémentine Prieur d'avoir accepté de juger mon travail.

Vincent, partager le bureau avec toi fût un grand plaisir, merci pour tout. Je n'entends plus le son de ta guitare lors d'une petite pause, c'est dommage ! Merci à tous les autres occupants du bureau 64, Aymen, Afaf et nos discussions diverses et (très) variées ! Sans oublier Morgane et Abdoulaye. J'ai vraiment passé de bon moments dans ce bureau en votre compagnie.

Un grand merci à toi Cécile, ce fût très facile et très agréable de travailler avec toi. Cette thèse n'aurait sans doute pas été la même sans toi. Christophe, nos chemins se croisent depuis maintenant pas mal d'années. Sans toi il n'y aurait pas eu cette collaboration avec le LNCMI.

Merci également à Ranine pour m'avoir fait goûter à la cuisine libanaise (la vraie, pas la version occidentalisée) ainsi qu'à Alexandre pour les différentes discussions qu'on a pu avoir. Elisa, è stato un vero piacere collaborare con te, grazie. Ti auguro il meglio per l'avvenire.

Merci à l'ensemble de l'équipe de Feel++.

Je tiens aussi à adresser mes remerciements à tous les membres de l'équipe EDP dont j'ai fait parti pendant cette thèse. Je remercie aussi Juana et Hélène pour leur gentillesse et les nombreux services administratifs qu'elles m'ont rendus.

Valène, merci pour ton soutien sans faille depuis le début, et pour tout le reste !

Je tiens également à remercier ma famille qui m'a soutenue tout au long de cette thèse. Enfin, je remercie celle qui partage ma vie, m'a encouragé et soutenu au quotidien : un immense merci Emilie !

Table des matières

Notations

Méthodes

FE	:	élément fini – *Finite Element*.
FEM	:	méthode élément fini – *Finite Element Method*.
RB	:	base réduite – *Reduced Basis*.
RBM	:	méthode base réduite – *Reduced Basis Method*.
EIM	:	méthode dínterpolation empirique – *Empirical Interpolation Method*.
SCM	:	méthode des contraintes successives – *Successive Constraints Method*.
MsFE	:	élément fini multi-échelles – *Multiscale Finite Element*.
MsFEM	:	méthode élément fini multi-échelles – *Multiscale Finite Element Method*.

Géométrie et maillage

d	:	dimension géométrique $d = 1$, 2 ou 3.
Ω	:	domaine régulier de $\mathbb{R}^d$.
$\boldsymbol{n}$	:	vecteur normal unitaire de $\partial\Omega$ orienté vers l'extérieur.
$\mathcal{T}_h$	:	maillage du domaine Ω de taille caractéristique h.
n_t	:	nombre d'élément contenu dans $\mathcal{T}_h$.
K	:	un élément de $\mathcal{T}_h$.
$\hat{K}$	:	un élément de référence (de même type que K).
$\mathcal{F}_{\hat{K},K}$	:	transformation géométrique de $\hat{K}$ vers K.

Espaces fonctionnels

$\mathbb{P}_N(K)$	:	espace vectoriel des polynômes de degrès total inférieur ou égale à N sur K.
$\hat{\phi}_i$	:	i^{me} fonctions de base FE défini sur $\hat{K}$.
ϕ_i	:	i^{me} fonctions de base FE défini sur K.
ξ_i^{pr}	:	i^{me} fonctions de base primale RB.
ξ_i^{du}	:	i^{me} fonctions de base duale RB.
$L^2(\Omega)$	:	espace des fonctions qui sont de carré intégrable sur Ω.
$D^\alpha v$	:	la dérivée partielle généralisé de v au sens au sens des distributions.
$H^k(\Omega)$	:	$\{v$ tel que $D^\alpha v \in L^2(\Omega)$, $\forall \alpha$ tel que $\lvert\alpha\rvert \leq k\}$.
$W_{N_{pr}}$	:	espace d'approximation RB primal.
$W_{N_{pr}}$	:	espace d'approximation RB dual.

Aimants à haut champ

V	:	volts.
K	:	kelvin.
W	:	watt.
m	:	mètre.
S	:	Siemens.
u	:	potentiel électrique (V).
T	:	température exprimée (K).
σ_0	:	coefficient de conductivité électrique à température fixée $T = T_0$ ($S\ m^{-1}$).
$\sigma(T)$	:	coefficient de conductivité électrique ($S\ m^{-1}$).
$k(T)$	:	coefficient de conductivité thermique ($W\ m^{-1}\ K^{-1}$).
L	:	nombre de Lorentz.
α	:	ratio résistivité-température.

Introduction

De nombreux domaines de l'ingénierie requièrent de pouvoir résoudre numériquement des équations aux dérivées partielles (EDP) modélisant des phénomènes physiques multi-échelles. La particularité de ces phénomènes est de faire intervenir plusieurs échelles.

C'est le cas notamment lorsque nous voulons modéliser un phénomène de diffusion thermique à l'intérieur d'un matériau composite. Quand les fibres d'un matériau sont positionnées de manière aléatoire alors le coefficient de conductivité thermique peut subir des variations. La nature hétérogène du matériau composite impose de prendre en compte le comportement de la température dans les petites échelles, pour prédire de manière précise le comportement "global" de la température au niveau des grandes échelles.

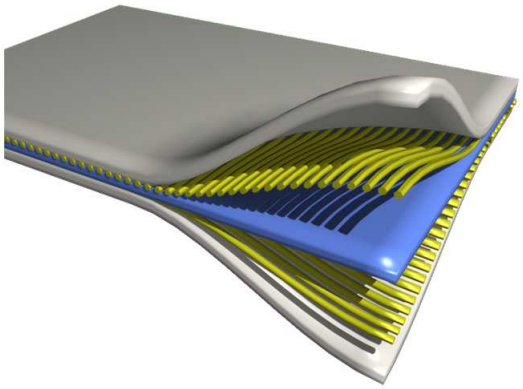


FIGURE 1 – Illustration d'un matériau composite. Source : http ://fr.wikipedia.org.

Il en est de même pour la modélisation d'écoulements en milieu poreux. Dans ce cas, la perméabilité du milieu varie suivant l'échelle considérée.

Même avec les ressources de calcul actuelles, il peut s'avérer très difficile, voir impossible, d'utiliser les méthodes numériques standards telles que les méthodes des éléments finis, volumes finis ou différences finies, pour discrétiser un domaine représentant un matériau composite à la plus petite échelle. Cependant, pour beaucoup de problèmes multi-échelles, il est suffisant de prédire le comportement macroscopique des propriétés du système étudié, tels que le coefficient de conductivité thermique ou la perméabilité du milieu.

Fonctions de base élément fini multi-échelles

Comme nous le verrons plus en détails dans le chapitre 6, il existe plusieurs méthodes permettant de résoudre ces problèmes multi-échelles avec un coût de calcul raisonnable.

Nous pouvons citer différentes méthodes : multigrille [21, 26, 130, 60], décomposition de domaines [115, 116, 104, 34], homogénéisation [89, 109, 11, 38], variationnelle multi-échelles [68, 69, 22], hétérogène multi-échelles [129, 1, 127, 128] ou encore élément fini multi-échelles [67, 66, 46].

Nous avons choisi de nous intéresser à la méthode des éléments finis multi-échelles. Cette méthode repose sur la prédiction du comportement de la solution – température à l'intérieur d'un matériau composite ou écoulements de fluides à travers un milieu poreux – à l'aide de la résolution du problème aux échelles macroscopiques, en ne faisant intervenir les contributions des échelles microscopiques que de manière indirecte.

Pour simplifier, considérons uniquement deux niveaux d'échelles : macroscopique et microscopique. Contrairement à la méthode élément fini, les fonctions de base de l'échelle macroscopique sont les solutions de problèmes locaux, définis au niveau de l'échelle microscopique. Chaque problème local est défini sur un seul élément géométrique du maillage "grossier" associé à l'échelle macroscopique.

Nous allons proposer une nouvelle construction de ces fonctions de base de l'échelle macroscopique. La clé de cette nouvelle construction repose sur l'utilisation d'une méthode de réduction d'ordre pour approcher, de manière rapide et fiable, ces fonctions de base. Nous verrons également qu'il est possible d'utiliser les nouvelles architectures permettant d'utiliser à la fois les CPU – *Central Processing Unit* – et les cartes graphiques – *Graphics Processing Unit (GPU)* – de manière efficace. En effet, autrefois les cartes graphiques ne pouvaient effectuer que des opérations basiques, mais de nos jours ces cartes sont de véritables atouts pour le calcul scientifique.

Pour mettre en place cette nouvelle construction des fonctions de base élément fini multi-échelles, il est important d'avoir un framework robuste pour appliquer une méthode de réduction d'ordre.

Framework base réduite

Plusieurs techniques de réduction existent. Une des méthodes les plus connues est la décomposition orthogonale aux valeurs propres [77, 65, 59, 85] – *Proper Orthogonal Decomposition (POD)* –. Avec cette technique, le temps est considéré comme un paramètre et des instantanés – *snapshots* – du champ inconnu sont obtenus numériquement ou via des manipulations expérimentales. Ces instantanés sont des vecteurs sur lesquels est appliquée une décomposition aux valeurs propres. L'espace d'approximation est alors engendré par les vecteurs propres associés aux plus grandes valeurs propres. L'étape de réduction consiste alors en une projection sur l'espace d'approximation qui vient d'être construit.

Nous verrons dans le chapitre 2 qu'il est possible de coupler la POD avec une autre méthode de réduction d'ordre : les bases réduites [101, 122, 123, 100, 103, 107] – *Reduced Basis Method (RBM)* –. Dans le cas de problèmes elliptiques, la méthode des bases réduites n'est pas couplée avec la POD, et l'espace d'approximation est engendré par différentes solutions de l'EDP paramétrée considérée.

Une autre technique de réduction d'ordre est la méthode de décomposition propre généralisée [8, 9, 36] – *Proper Generalized Decomposition (PGD)* – qui repose sur une séparation de variables et une construction incrémentale de l'espace d'approximation.

Nous avons choisi la méthode des bases réduites comme technique de réduction d'ordre.

Au cours de cette thèse un framework base réduite, supportant les architectures parallèles, a été développé. Ce framework permet d'appliquer la méthode de réduction d'ordre sur des problèmes stationnaires de type Navier-Stokes, mais également sur des problèmes multi-physiques non-linéaires sur des géométries industrielles pour lesquelles le parallélisme du framework est indispensable. Notons qu'il existe d'autres framework base réduite [90].

Cependant, le framework base réduite doit pouvoir s'appuyer sur un outil capable de mettre en oeuvre des méthodes numériques pour résoudre les EDPs. La librairie Feel++ [95, 102, 97, 99] – *Finite Element Embedded Language in* C++ – va remplir cette fonction.

Feel++ : un outil pour le calcul scientifique

La librairie Feel++ fournit une interface C++ pour résoudre des systèmes d'EDPs en 1D, 2D et 3D via la méthode des éléments finis, combinée aux méthodes d'approximation de Galerkin. Cette librairie a pour objectif de rassembler la communauté scientifique pour la mise en oeuvre de méthodes numériques avancées et le calcul haute performance.

Dans le but de rester très proche des mathématiques lorsque l'utilisateur écrit la formulation variationnelle de son problème, Feel++ propose un langage embarqué dans le C++ , il s'agit du paradigme des DSEL – *Domain Specific Embedded Language* – . Ainsi, la complexité des algorithmes informatiques est transparent pour l'utilisateur, tout en gardant une large flexibilité et une vision claire des mathématiques

Pour la résolution des systèmes algébriques, Feel++ dispose d'une interface avec la librairie `PETSc` [13, 12, 14, 114]. Combinée à cette puissante librairie de calcul, Feel++ offre un important choix de solveurs et de préconditionneurs qui peuvent être paramétrés directement en ligne de commande. Pour effectuer un calcul aux valeurs propres généralisé, Feel++ dispose également d'une interface avec la librairie `SLEPc` [64, 63].

D'un point de vue programmation informatique, Feel++ utilise le langage C++ avancé (métaprogrammation par *templates*) Ainsi, grâce à l'inférence des types, les nouveaux mots clés comme `auto` seront utilisés dans les exemples présentés dans cette thèse. Feel++ utilise également plusieurs parties de la librairie C++ Boost [20].

Illustrons nos propos en regardant comment résoudre le laplacien 2D dans un carré unité avec une approximation $\mathbb{P}_3$ (voir le code 1).

Listing 1 – Résolution du laplacien avec Feel++

```
#include <feel/feel.hpp>

int main(int argc, char**argv )
{
    using namespace Feel;
    Environment env(_argc=argc,_argv=argv);
    // definition du maillage et espace de fonction
    auto mesh = unitSquare();
    auto Vh = Pch<3>(mesh);
    auto u = Vh->element();
    auto v = Vh->element();
    // forme lineaire
    auto l = form1(_test=Vh);
    l = integrate(_range=elements(mesh),
                  _expr=id(v));
    // forme bilineaire
    auto a = form2(_trial=Vh,_test=Vh);
```

```
    a = integrate(_range=elements(mesh),
                  _expr=gradt(u)*trans(grad(v)) );
    a+=on(_range=boundaryfaces(mesh),_rhs=l,_element=u,
          _expr=cst(0.) );
    // resolution du systeme algebrique
    a.solve(_rhs=l,_solution=u);
    // extraction des resultats pour la visualisation
    auto e = exporter(_mesh=mesh,_name="laplacian" );
    e->add( "u", u ); e->save();
    return 0;
}
```

Note : si on remplace `unitSquare()` par `unitCube()`, ce code correspondrait à la résolution d'un laplacien 3D dans un cube.

Plan de la thèse

La première partie de cette thèse sera consacrée à la théorie. Nous commencerons par rappeler, dans le chapitre 1, des notions préliminaires utiles pour la lecture de ce manuscrit. Le chapitre 2 introduira en détails la méthode des bases réduites lorsque les problèmes à traiter sont linéaires – elliptiques ou paraboliques – et qu'il est possible d'avoir directement une décomposition affine en paramètres. Le chapitre 3 quant à lui exposera la méthodologie à utiliser pour appliquer la méthode RBM à des problèmes qui ne dépendent plus des paramètres de manière affine, et qui peuvent être également non-linéaires. Le framework base réduite permettant également de traiter des problèmes multi-physiques, nous verrons dans le chapitre 4 comment appliquer la méthode RBM à ce type de problèmes. Le chapitre 5 mettra en avant les difficultés associées à la résolution des problèmes avec non-linéarités quadratiques, via la méthode RBM et proposera une technique de projection permettant de les résoudre uniquement dans un espace de dimension réduite. A ce stade, la méthodologie base réduite permettant de traiter aussi bien des problèmes linéaires, non-linéaires, elliptiques, paraboliques, avec ou sans dépendance affine en paramètres, ou encore possédant des non-linéarités quadratiques, aura été exposée. Tous les ingrédients seront disponibles pour se tourner vers les problèmes de type multi-échelles. La chapitre 6 exposera une nouvelle construction des fonctions de base élément fini multi-échelles, basée sur l'utilisation de la méthode RBM.

Ensuite, une deuxième partie sera dédiée à l'implémentation du framework base réduite. Tout d'abord nous présenterons l'architecture logicielle dans le chapitre 7, nous verrons également que l'interface utilisateur a été pensée pour simplifier au maximum l'utilisation du framework base réduite. Puis nous verrons comment est mis en oeuvre le parallélisme du framework base réduite dans le chapitre 8.

Enfin, la dernière partie exposera les résultats numériques obtenus à l'aide du framework base réduite. Le chapitre 9 montrera les résultats obtenus en utilisant la technique de projection proposée dans le chapitre 5, sur un problème de type convection naturelle dans une cavité. Le chapitre 10 montrera que le framework base réduite peut être utilisé dans le cas de problèmes non-linéaires multi-physiques. Pour illustrer cette utilisation, une étude paramétrique ainsi qu'une étude de sensibilité sur un aimant – utilisé au Laboratoire National des Champs Magnétiques Intenses de Grenoble – sera présentée.

Publications

Articles publiés

[111] E. Schenone, S. Veys and C. Prud'Homme. High Performance Computing for the Reduced Basis Method. Application to Natural Convection. ESAIM : Proceedings, pages 255 – 273, December 2013.

[42] C. Daversin, S. Veys, C. Trophime and C. Prud'Homme. A reduced basis framework : Application to large scale nonlinear multi-physics problems. Esaim Proc., 43 :225–254, December 2013.

Conférences internationales

[126] S. Veys, C. Daversin, C. Prud'homme and C. Trophime. Reduced order modeling of high magnetic field magnets. Conference Record of the 11th International Workshop on Finite Elements for Microwave Engineering, 2012.

[125] S. Veys, R. Chakir, C. Daversin, C. Prud'homme, and C. Trophime. A computational framework for certified reduced basis methods : application to a multiphysic problem. Conference Record of the 6th European Congress on Computational Methods in Applied Sciences and Engineering, 2012.

[41] C. Daversin, C. Prud'homme, C. Trophime and S. Veys. Reduced order mode- ling of high magnetic field magnets. Conference Record of the 9th International Symposium of Electric and Magnetic Fields, 2013.

[40] C. Daversin, C. Prud'homme, C. Trophime and S. Veys. Applications of reduced order modeling to high magnetic field resistive magnets developments. Conference Record of the 23th International Conference on Magnet Technology, 2013.

[117] C. Trophime, C. Daversin, C. Prud'homme and S. Veys. Reduced Order Modeling of High Magnetic Field Magnets. Conference Record of the 17th U.S. National Congress on Theoretical and Applied Mechanics, 2014.

Première partie

Théorie mathématique

Chapitre 1

Notions préléminaires

Commençons par un premier chapitre dédié aux notions préliminaires, utiles pour la lecture de cette thèse. Dans un premier temps nous définirons les espaces de fonctions qui vont nous fournir un cadre mathématique qui permettra d'appliquer la méthode des éléments finis ou des bases réduites. Dans un deuxième temps, nous donnerons une définition de l'élément fini de Lagrange, qui est la brique de base autour de laquelle – pour cette thèse – vont s'articuler l'approximation élément fini, mais également base réduite.

1 Espace de fonctions

Considérons Ω un domaine borné régulier dans $\mathbb{R}^d$ (pour $d = 1, ..., 3$).

1.1 Espaces de Lebesgue

En notant $L^2(\Omega)$ l'espace des fonctions de carré intégrable sur Ω, nous avons

$$L^2(\Omega) = \Big\{ v \text{ tel que } \big|\big|v\big|\big|_{L^2(\Omega)} < \infty \Big\}, \tag{1.1}$$

où

$$\big|\big|v\big|\big|_{L^2(\Omega)} = \left(\int_\Omega v^2 \right)^{1/2}. \tag{1.2}$$

Plus généralement, un espace de Lebesgue $L^p(\Omega)$, pour $p \geq 1$, est défini par

$$L^p(\Omega) = \Big\{ v \text{ tel que } \big|\big|v\big|\big|_{L^p(\Omega)} < \infty \Big\}, \tag{1.3}$$

où

$$\big|\big|v\big|\big|_{L^p(\Omega)} = \left(\int_\Omega \big|v\big|^p \right)^{1/p}, \ 1 \leq p \leq \infty \text{ et } \big|\big|v\big|\big|_{L^\infty(\Omega)} = \operatorname*{ess\,sup}_{\boldsymbol{x} \in \Omega} \big|v(\boldsymbol{x})\big|. \tag{1.4}$$

La borne supérieure essentielle – ou *essential supremum* – de la fonction v définie sur Ω, notée $\operatorname{ess\,sup}_{\boldsymbol{x}\in\Omega} v(\boldsymbol{x})$, est le plus petit des "presque majorants" de v. Rappelons que si C_v est un presque majorant de v, alors l'ensemble $\big\{ \boldsymbol{x} \in \Omega, \text{ tel que } v(\boldsymbol{x}) < C_v \big\}$, est inclus dans un ensemble de mesure nulle.

Tout espace $L^p(\Omega)$, avec $p \geq 1$, est un espace de Banach et désigne un ensemble de fonctions qui ne diffèrent que sur un ensemble négligeable.

1.2 Espaces de Hilbert

Les espaces de Hilbert sont notés $H^k(\Omega)$, avec k un entier positif. Nous avons

$$H^k(\Omega) \ = \Big\{ v \text{ tel que } \ D^\alpha v \in L^2(\Omega) \ , \forall \alpha \text{ tel que } \ |\alpha| \leq k \Big\}, \tag{1.5}$$

où $D^\alpha v$ est la dérivée partielle généralisée de v au sens des distributions. Pour le multi-indices $\alpha = (\alpha^1, ..., \alpha^d)$ tel que $\alpha^i \geq 0$ pour $i = 1, ..., d$, et pour tout $\boldsymbol{x} = (x^1, ..., x^d) \in \Omega$, nous avons

$$D^\alpha \equiv \frac{\partial^\alpha}{\partial_{x^1}^{\alpha^1} ... \partial_{x^d}^{\alpha^d}}, \text{ et } |\alpha| = \sum_{i=1}^{d} \alpha_i. \tag{1.6}$$

Notons que le produit scalaire associé à $H^k(\Omega)$ est donné par

$$\big(u, v\big)_{H^k(\Omega)} \ \equiv \sum_{|\alpha| \leq k} \int_\Omega D^\alpha u \ D^\alpha v, \tag{1.7}$$

et la norme induite,

$$\big|\big|u\big|\big|_{H^k(\Omega)} \ \equiv \left(\sum_{|\alpha| \leq k} \int_\Omega \big| D^\alpha u \big|^2 \right)^{1/2}. \tag{1.8}$$

Les espaces de Hilbert étendent la notion d'espaces euclidiens dans le cas d'un espace de dimension quelconque.

Remarquons que l'espace $L^2(\Omega)$, qui est également l'espace $H^0(\Omega)$, est le seul espace de Lebesgue qui est un espace de Hilbert.

Par la suite nous aurons besoin d'utiliser l'inégalité de Cauchy-Schwarz. Tout espace de Hilbert possède un produit scalaire et une norme induite, l'inégalité de Cauchy-Schwarz peut donc s'écrire

$$\big|\big(u, v\big)_{H^k(\Omega)} \ \big| \leq \big|\big|u\big|\big|_{H^k(\Omega)} \ \ \big|\big|v\big|\big|_{H^k(\Omega)} \ . \tag{1.9}$$

1.3 Espaces de Sobolev

Introduisons à présent les espaces qui serviront de cadre à l'écriture des EDPs sous forme variationnelle. Cette formulation est utilisée afin d'appliquer la méthode des éléments finis.

Notons $W^{k,p}(\Omega)$, avec k un entier positif et $p \geq 1$, l'espace de Sobolev défini par

$$W^{k,p}(\Omega) = \Big\{ v \text{ tel que } \ D^\alpha v \in L^p(\Omega) \ , \forall \alpha \text{ tel que } \ |\alpha| \leq k \Big\}. \tag{1.10}$$

Les espaces de Sobolev sont des espaces de Banach avec les normes suivantes :

$$\big|\big|v\big|\big|_{W^{k,p}(\Omega)} \equiv \left(\sum_{|\alpha| \leq k} \int_\Omega \big| D^\alpha v \big|^p \right)^{1/p}, \text{ pour } 1 \leq p < \infty, \tag{1.11}$$

et

$$\big|\big|v\big|\big|_{W^{k,\infty}(\Omega)} \equiv \max_{|\alpha| \leq k} \operatorname*{ess\,sup}_{\boldsymbol{x} \in \Omega} \big| D^\alpha v(\boldsymbol{x}) \big|. \tag{1.12}$$

Le cas $p = 2$ est un cas particulier intéressant. En effet les espaces $W^{k,2}(\Omega)$, avec $k \geq 0$, ont une structure d'espace de Hilbert. Notons également que lorsque nous prenons $k = 0$, les espaces $W^{0,p}(\Omega)$ sont les espaces $L^p(\Omega)$. Il apparaît alors clairement que les espaces de Lebesgue sont inclus dans les espaces de Sobolev.

1.4 Espaces dual et norme associée

Dans ce manuscrit, la méthodologie des bases réduites certifiées sera détaillée. Avant de définir les estimateurs d'erreur a posteriori dans le cadre des bases réduites, nous avons besoin de définir la notion de "norme duale". Notons cependant que cette norme est également utilisée pour définir la propriété de stabilité linéaire d'un problème par rapport aux données (1.19), afin de montrer que ce dernier est bien posé.

Considérons un espace de Hilbert, noté X, muni d'un produit scalaire $(.,.)_X$, et d'une norme $||.||$, ainsi qu'une forme linéaire $\ell : X \to \mathbb{R}$. L'espace dual, X', est alors défini par

$$X' \equiv \Big\{ \ell\big(v\big) \text{ tel que } \big|\big|\ell\big(v\big)\big|\big|_{X'} < \infty \Big\}, \tag{1.13}$$

où la norme duale est donnée par

$$\big|\big|\ell\big(v\big)\big|\big|_{X'} \equiv \sup_{v \in X} \frac{\ell\big(v\big)}{\big|\big|v\big|\big|_X}. \tag{1.14}$$

L'espace dual X' est un espace de Hilbert. Lorsque $X = H^k(\Omega)$, alors nous avons $X' = H^{-k}(\Omega)$.

Dans la pratique, pour évaluer la norme duale $||\ell(v)||_{X'}$, nous nous servons du théorème de la représentation de Riesz. En effet, d'après ce théorème, pour toute forme linéaire $\ell \in X'$, il existe un unique élément $\hat{e} \in X$, tel que

$$\big(\hat{e}, v\big)_X = \ell\big(v\big), \ \forall v \in X. \tag{1.15}$$

D'après l'inégalité de Cauchy-Schwarz (1.9), nous pouvons écrire

$$\big|\big|\ell\big(v\big)\big|\big|_{X'} = \sup_{v \in X} \frac{\big(\hat{e}, v\big)_X}{\big|\big|v\big|\big|_X} = \big|\big|\hat{e}\big|\big|_X. \tag{1.16}$$

2 Approximation élément fini

Nous présentons ici, brièvement, la manière dont est construite l'approximation élément fini [37] – ou *Finite Element (FE)* – en utilisant la méthode de Galerkin. Cette approximation va servir par la suite de référence pour l'approximation base réduite.

2.1 Exemple introductif

Commençons par illustrer le fonctionnement de la méthode des éléments finis avec un exemple introductif abstrait. Soient $U \equiv H^1(\Omega)$ et $V \equiv H^1(\Omega)$, deux espaces de Hilbert. Nous supposons que la formulation variationnelle (ou faible) de notre problème s'écrit comme, chercher $u \in U$ tel que

$$a\big(u, v\big) = f\big(v\big), \ \forall v \in V, \tag{1.17}$$

avec $a : U \times V \to \mathbb{R}$ et $f : V \to \mathbb{R}$, les formes bilinéaire et linéaire associées à notre problème. Les espaces U et V sont respectivement les espaces "trial" et "test". Dans la suite de ce manuscrit nous supposons que ces espaces sont identiques, et nous notons cet

unique espace $X \equiv H^1(\Omega)$, ainsi que $\big|\big|.\big|\big|_X$ sa norme. La formulation variationnelle (1.17) peut s'écrire alors comme chercher $u \in X$ tel que

$$a\big(u,v\big) = f\big(v\big),\ \forall v \in X, \tag{1.18}$$

avec $a : X \times X \to \mathbb{R}$ et $f : X \to \mathbb{R}$.

Introduisons maintenant la propriété de stabilité linéaire de (1.18) par rapport aux données :

$$\exists\ c > 0 \text{ tel que } \ \forall f \in X',\ \big|\big|u\big|\big|_X \leq c\big|\big|f\big|\big|_{X'}. \tag{1.19}$$

Si le problème (1.18) a une et une seule solution, et si de plus la propriété de stabilité linéaire (1.19) est vérifiée, alors on dit que ce problème est bien posé au sens de J.Hadamard [61]. Dans la suite nous supposons que le problème traité est bien posé.

L'espace X étant de dimension infinie, nous allons résoudre le problème (1.18) sur un espace de dimension finie, $X_{\mathcal{N}}$ – de dimension $\mathcal{N}$ –, grâce à la discrétisation FE. Tout d'abord il faut mailler le domaine d'étude Ω. Définissons $\mathcal{T}_h = \{K_i, i = 1, ..., n_t\}$, avec n_t un entier strictement positif, un ensemble de n_t éléments formant une partition de Ω. $\mathcal{T}_h$ est alors un maillage, composé de n_t éléments, de Ω. L'espace de fonctions $X_{\mathcal{N}} = span\big\{\phi_1, ..., \phi_{\mathcal{N}}\big\}$ est défini sur $\mathcal{T}_h$. Nous supposons dans cet exemple que les fonctions ϕ_i, pour $i = 1, ..., \mathcal{N}$ sont des fonctions polynomiales, continues, affines par morceaux. Soit $u_{\mathcal{N}}$ un élément de l'espace $X_{\mathcal{N}}$, nous pouvons décomposer $u_{\mathcal{N}}$ dans la base de $X_{\mathcal{N}}$ en écrivant

$$u_{\mathcal{N}} = \sum_{i=1}^{\mathcal{N}} u^i_{\mathcal{N}}\ \phi_i. \tag{1.20}$$

Le problème (1.18) est alors approché par le problème suivant : chercher $u_{\mathcal{N}} \in X_{\mathcal{N}}$ tel que

$$A\ u_{\mathcal{N}} = F, \tag{1.21}$$

où $u_{\mathcal{N}} \in \mathbb{R}^{\mathcal{N}}$, la matrice (creuse) $A \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ ainsi que le vecteur $F \in \mathbb{R}^{\mathcal{N}}$ sont définis par

$$u_{\mathcal{N}}\big(i\big) = u^i_{\mathcal{N}},\quad A\big(i,j\big) = a\big(\phi_j, \phi_i\big) \text{ et } F\big(i\big) = f\big(\phi_i\big). \tag{1.22}$$

Nous avons projeté le problème (1.18) sur l'espace $X_{\mathcal{N}}$. La relation d'orthogonalité de Galerkin nous permet d'écrire

$$a\big(u - u_{\mathcal{N}}, v\big) = 0,\ \forall v \in X_{\mathcal{N}}. \tag{1.23}$$

De plus, d'après le lemme de Céa, si la taille caractéristique du maillage tend vers zéro alors $u_{\mathcal{N}}$ tend vers u dans X. Ici, $u - u_{\mathcal{N}}$ représente l'erreur commise par l'approximation de (1.18) par (1.21).

Introduisons maintenant la notion de coercivité. La forme bilinéaire a est dite coercive s'il existe une constante $\alpha > 0$, telle que

$$a\big(u,u\big) \geq \alpha\big|\big|u\big|\big|^2_X,\ \forall u \in X. \tag{1.24}$$

Si a est continue, symétrique et coercive, nous pouvons définir le produit scalaire

$$\big(\big(\big(u,v\big)\big)\big)_a \equiv a\big(u,v\big),\ \forall u, v \in X, \tag{1.25}$$

ainsi que la norme associée

$$\big|\big|\big|u\big|\big|\big|_a \equiv \sqrt{a\big(u,u\big)},\ \forall u \in X. \tag{1.26}$$

Dans ce cas, et d'après (1.23), $u_{\mathcal{N}}$ est la projection orthogonale – en utilisant le produit scalaire défini en (1.25) – de u sur l'espace $X_{\mathcal{N}}$.

2.2 Définition

Nous définissons maintenant un élément fini. Ce manuscrit traitant notamment de la méthodologie base réduite, il est important de définir la notion d'élément fini, mais ce n'est cependant pas le coeur des travaux présentés. Une définition des éléments finis plus détaillée se trouve dans les thèses [91, 32]. Les simulations numériques, réalisées dans le cadre de cette thèse, reposent toutes sur l'utilisation de maillages composés de simplexes. Un simplexe est une entité élémentaire utilisée pour construire un maillage, il s'agit d'un segment en 1D, triangle en 2D ou tétraèdre en 3D. Dans la suite nous ne considérons pas les éléments finis associés à d'autres types de maillages. Une définition complète des éléments finis peut être trouvée dans [28, 72, 25, 37].

Un élément fini est défini par le triplet $\big(K, P, \Sigma\big)$. K est un élément géométrique compact, connexe et d'intérieur non vide, dont la frontière est lipschitzienne. P est un espace – de dimension finie – de fonctions $p : K \to \mathbb{R}^{\nu}$ où ν est un entier positif. Σ est un ensemble de N_{ldofs} formes linéaires, notées σ_i pour $i = 1, ..., N_{ldofs}$, définies sur P, et telles que l'application linéaire :

$$P \ni p \longmapsto \Big(\sigma_1\left(p\right), ..., \sigma_{N_{ldofs}}\left(p\right)\Big)^T \in \mathbb{R}^{\mathcal{N}} \tag{1.27}$$

soit bijective. Les formes linéaires $\big(\sigma_1, ..., \sigma_{N_{ldofs}}\big)$ sont appelées degré de liberté de l'élément fini.

Il existe plusieurs éléments finis. Nous pouvons citer par exemple les éléments de Lagrange, Hermite, Raviart-Thomas, Crouzeix-Raviart ou encore Nedelec, mais cette liste n'est pas exhaustive. Tous les résultats présentés dans ce manuscrit se basent sur l'élément fini de Lagrange, nous ne considérons donc que cet élément par la suite.

2.3 Élément fini de Lagrange

Notons $\mathbb{P}_k(K)$ l'espace de polynômes de degré total inférieur ou égal à k définis sur l'élément géométrique K, et $\Sigma = \big\{\sigma_1, ..., \sigma_{N_{ldofs}}\big\}$ l'ensemble qui contient les formes linéaires définies, pour $i = 1, ..., N_{ldofs}$, par

$$\begin{array}{rccl} \sigma_i : \mathbb{P}_k(K) & \longrightarrow & \mathbb{R} \\ p & \longmapsto & p\left(\boldsymbol{t}_i\right), \end{array} \tag{1.28}$$

où les $\boldsymbol{t}_i$, pour $i = 1, ..., N_{ldofs}$, sont les points d'interpolation définis sur l'élément géométrique K.

L'élément de Lagrange est alors défini par le triplet $\big(K, \mathbb{P}_k(K), \Sigma\big)$. Nous constatons que le triplet définissant l'élément est associé à $K \in \mathcal{T}_h$. Cependant dans la pratique nous nous limitons à un seul élément géométrique de référence $\hat{K}$. Notons que pour tout $K \in \mathcal{T}_h$, il est possible de passer de $\hat{K}$ à K, à l'aide d'une transformation géométrique adaptée (voir la section 2.4).

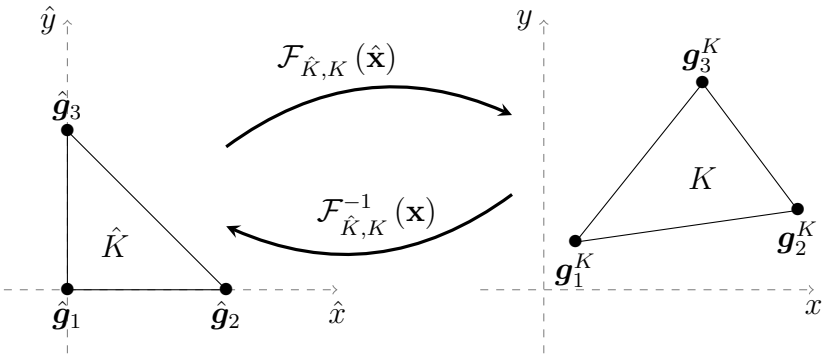
Nous allons donc nous concentrer sur l'élément de Lagrange d'ordre k défini sur $\hat{K}$ par le triplet $\big(\hat{K}, \hat{\mathbb{P}}_k(\hat{K}), \hat{\Sigma}\big)$, où $\hat{\Sigma} = \big\{\hat{\sigma}_1, ..., \hat{\sigma}_{N_{ldofs}}\big\}$ est l'ensemble contenant les N_{ldofs} formes linéaires définies sur $\hat{\mathbb{P}}_k(\hat{K})$ et à valeur dans $\mathbb{R}$. Introduisons $\hat{\phi}_i$, pour $i = 1, ..., N_{ldofs}$, les fonctions de base de l'espace $\hat{\mathbb{P}}_k(\hat{K})$ telles que

$$\hat{\sigma}_j\big(\hat{\phi}_i\big) = \delta_{ij}, \quad 1 \leq i, j \leq N_{ldofs}, \tag{1.29}$$

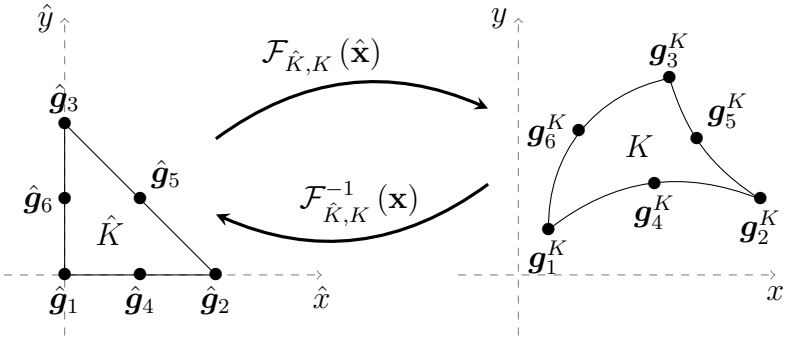
où δ_{ij} est le symbole de Kronecker.

2.4 Transformation géométrique

Nous venons de voir comment définir une base de l'espace $\hat{\mathbb{P}}_k(\hat{K})$, défini sur l'élément de référence $\hat{K}$. Nous allons maintenant montrer comment définir la transformation géométrique $\mathcal{F}_{\hat{K},K} : \hat{K} \to K$ telle que $\mathcal{T}_h = \left\{K = \mathcal{F}_{\hat{K},K}(\hat{K})\right\}$.



(a) Transformation linéaire sur un triangle



(b) Transformation quadratique sur un triangle

FIGURE 1.1 – Transformations géométriques d'ordre 1 et 2 sur un triangle.

En effet, dans FEEL++, les fonctions de base de l'espace $\mathbb{P}_k(K)$ ne sont jamais construites directement sur l'élément K, mais elles sont obtenues à partir de celles de $\hat{\mathbb{P}}_k(\hat{K})$ à l'aide de la transformation $\mathcal{F}_{\hat{K},K}$. De cette manière, les calculs élémentaires – par exemple évaluations et dérivations des fonctions de base aux points de quadratures – sont effectués uniquement sur l'élément de référence $\hat{K}$. La transformation géométrique $\mathcal{F}_{\hat{K},K}$ est une application inversible dès que l'élément K n'est pas dégénéré. Nous faisons l'hypothèse que $\mathcal{F}_{\hat{K},K}$ est C^1-difféomorphisme. En notant $\mathcal{F}^{-1}_{\hat{K},K}$ l'application inverse de $\mathcal{F}_{\hat{K},K}$, nous pouvons écrire

$$\begin{array}{rccc} \mathcal{F}_{\hat{K},K} : & \hat{K} \in \mathbb{R}^d & \to & K \in \mathbb{R}^d \\ & \hat{\boldsymbol{x}} & \to & \boldsymbol{x} \end{array} \quad \text{et} \quad \begin{array}{rccc} \mathcal{F}^{-1}_{\hat{K},K} : & K \in \mathbb{R}^d & \to & \hat{K} \in \mathbb{R}^d \\ & \boldsymbol{x} & \to & \hat{\boldsymbol{x}}. \end{array} \tag{1.30}$$

Nous nous servons de l'élément fini de Lagrange d'ordre k pour définir la transformation géométrique d'ordre k. Soient $\left\{\boldsymbol{g}_1^K, ..., \boldsymbol{g}_{N_{ldofs}}^K\right\}$ l'ensemble des noeuds géométriques de l'élément K . L'application $\mathcal{F}_{\hat{K},K}$ s'exprime alors comme une combinaison linéaire des

fonctions de base définies sur l'élément de référence $\hat{K}$,

$$\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}) = \sum_{i=1}^{N_{ldofs}} \boldsymbol{g}_i^K \ \hat{\phi}_i. \tag{1.31}$$

Introduisons $J_{\mathcal{F}_{\hat{K},K}}$, $J^{-1}_{\mathcal{F}_{\hat{K},K}}$, $J^{-T}_{\mathcal{F}_{\hat{K},K}}$ et $|J_{\mathcal{F}_{\hat{K},K}}|$ comme étant respectivement la matrice jacobienne de $\mathcal{F}_{\hat{K},K}$, la matrice inverse de $J_{\mathcal{F}_{\hat{K},K}}$, la transposée de $J^{-1}_{\mathcal{F}_{\hat{K},K}}$ et le déterminant de $J_{\mathcal{F}_{\hat{K},K}}$. Ces expressions sont indispensables lorsque l'on souhaite faire le changement de variable $\mathcal{F}_{\hat{K},K}$ dans une intégrale sur K. En effet, aucune intégrale n'est calculée directement sur l'élément K. Toutes les intégrales sont calculées sur l'élément de référence $\hat{K}$. On se propose de voir comment calculer – sur l'élément de référence $\hat{K}$ – différentes intégrales qui portent sur les fonctions $f : K \to \mathbb{R}$ et $g : K \to \mathbb{R}$:

$$\int_K f(\boldsymbol{x}) = \int_{\hat{K}} f\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\big) \ |J_{\mathcal{F}_{\hat{K},K}}|, \tag{1.32}$$

$$\int_K \nabla f(\boldsymbol{x}) = \int_{\hat{K}} \nabla f\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\big) J^{-T}_{\mathcal{F}_{\hat{K},K}} |J_{\mathcal{F}_{\hat{K},K}}|, \tag{1.33}$$

et

$$\int_K \nabla f(\boldsymbol{x}) \cdot \nabla g(\boldsymbol{x}) = \int_{\hat{K}} \nabla f\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\big) \ \Big(J^{-T}_{\mathcal{F}_{\hat{K},K}} J^{-1}_{\mathcal{F}_{\hat{K},K}}\Big) \nabla g\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\big) \ |J_{\mathcal{F}_{\hat{K},K}}|. \tag{1.34}$$

Résumé

Nous avons vu le cadre mathématique permettant de résoudre les EDPs. Les principes fondamentaux de la méthode des éléments finis, ainsi que l'élément fini de Lagrange ont été introduits. Ces notions servent de briques de base pour la suite des travaux présentés dans ce manuscrit. Ce chapitre était donc axé sur les notions préliminaires.

Chapitre 2

La méthode des bases réduites appliquée aux problèmes linéaires

La méthode des bases réduites a été introduite à la fin des années 1970, début des années 1980, pour l'analyse mécanique non-linéaire [4, 88, 86, 87]. Par la suite elle a été plus largement mise en oeuvre dans d'autres domaines [51, 92, 93, 105, 17, 15].

Nous nous intéressons ici à un modèle mathématique qui décrit le comportement physique d'un système en s'appuyant sur une ou plusieurs EDPs paramétrée dont la solution peut être un champ scalaire $u(\boldsymbol{\mu})$ ou vectoriel $\boldsymbol{u}(\boldsymbol{\mu})$. L'ensemble des paramètres $\boldsymbol{\mu}$ sert à identifier une configuration particulière du système décrit par le modèle (et donc les EDPs sous-jacentes) et sera identifié comme entrée du modèle. Notons $\mathcal{D}$ l'espace des paramètres de dimension P, avec $P \geq 1$. Nous avons la relation $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^P$. Ces paramètres d'entrée peuvent êtres des variables de caractérisation (par exemple des propriétés physiques de matériaux), des variables de design (paramètres géométriques) ou encore des variables de contrôle (la puissance électrique d'un système). Définissons à présent les sorties du modèle. Ces sorties sont des quantités d'intérêts qui dépendent de la solution des EDPs sous-jacentes au modèle. On peut citer comme exemple de sortie la température moyenne du système, ou un flux de chaleur. Il existe donc un lien fort entre les entrées du modèle et les sorties.

Considérons une sortie de notre modèle $s(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle d'un champ scalaire $u(\boldsymbol{\mu})$ dépendant de la valeur des paramètres d'entrée du modèle. Nous avons alors une relation *entrées-sorties* $\boldsymbol{\mu} \rightarrow s(\boldsymbol{\mu})$ qui exige la connaissance de la solution $u(\boldsymbol{\mu})$ d'une EDP paramétrée.

La méthode des bases réduites a été pensée pour être appliquée dans un contexte temps-réel (nécessaire pour faire du contrôle commande) et également dans un contexte demandant beaucoup d'évaluations des sorties du modèle (typiquement pour des méthodes d'optimisation ou d'analyse de sensibilités).

Soit Ω un domaine borné, régulier, dans $\mathbb{R}^d$ (pour $d = 1, ..., 3$). Introduisons l'espce de fonction X tel que $H_0^1(\Omega) \subset X \subset H^1(\Omega)$, ainsi que l'espace d'approximation FE, $X_{\mathcal{N}}$, de (grande) dimension $\mathcal{N}$. La méthode RBM s'appuie sur le fait que l'évolution de la solution de l'EDP $u(\boldsymbol{\mu})$ n'occupe pas tout l'espace X dans lequel elle est recherchée, mais seulement une variété $\mathcal{M}$ de bien plus faible dimension, lisse et induite par la dépendance paramétrique. Dans le cas elliptique, la variété $\mathcal{M}$ est définie par :

$$\mathcal{M} = \Big\{ u(\boldsymbol{\mu}) \in X, \ \forall \boldsymbol{\mu} \in \mathcal{D} \Big\}. \tag{2.1}$$

De la même manière que nous approchons l'espace X par l'espace discret $X_{\mathcal{N}}$, nous construisons une approximation $\mathcal{M}_{\mathcal{N}}$ de $\mathcal{M}$ de la manière suivante :

$$\mathcal{M}_{\mathcal{N}} = \Big\{ u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}},\ \forall \boldsymbol{\mu} \in \mathcal{D} \Big\}. \tag{2.2}$$

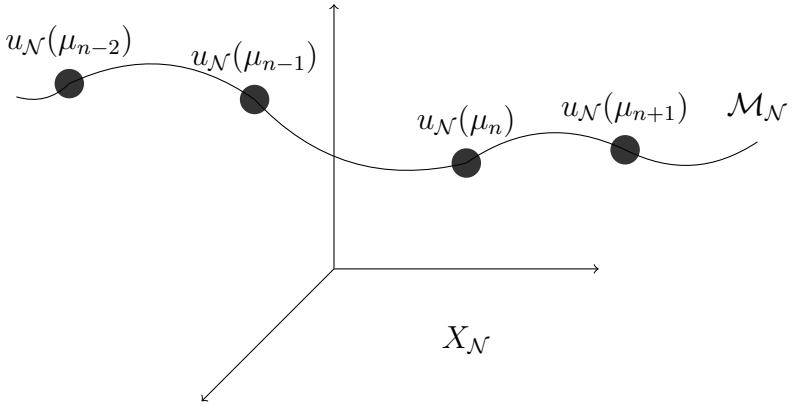


FIGURE 2.1 – Illustration de l'espace d'approximation $\mathcal{M}_{\mathcal{N}}$ de faible dimension par rapport à la dimension de $X_{\mathcal{N}}$.

La méthode des bases réduites repose sur une projection de type Galerkin sur un espace W_N de dimension N, avec $N << \mathcal{N}$. Nous verrons comment construire l'espace W_N afin qu'il approche au mieux de $\mathcal{M}_{\mathcal{N}}$.

Nous allons décrire la méthode CRBM [101, 122, 123, 100, 103, 107, 90] – *Certified Reduced Basis Method* (CRBM) – qui a vu le jour au début des années 2000 et qui peut être utilisée pour avoir une résolution rapide et fiable d'EDPs, elliptiques ou paraboliques, paramétrées par un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$. Dans ce chapitre nous nous limiterons aux problèmes linéaires. La méthode CRBM se démarque des travaux plus anciens par l'utilisation d'estimateurs d'erreur efficaces et rapides à évaluer, basés sur une stratégie *hors-ligne/en-ligne* que nous expliciterons dans ce chapitre. L'espace des paramètres peut être de grande dimension, alors que dans les précédents travaux il était souvent unidimensionnel ou de très petite dimension. De plus, l'espace d'approximation W_N est global. Les estimateurs d'erreur permettent non seulement de quantifier et de contrôler l'erreur commise par la méthode, mais aussi d'explorer l'espace des paramètres de manière optimale. Ils sont indispensables à la construction de l'espace d'approximation W_N via l'algorithme glouton – ou *greedy* –. L'algorithme glouton sélectionne les paramètres pour lesquels nous approchons le moins bien la sortie du modèle. Il est à noter que dans le cas parabolique, bien que dans un premier temps l'algorithme glouton fût utilisé seul pour construire W_N [55, 53], il est généralement couplé avec une méthode de décomposition orthogonale aux valeurs propres [77, 65, 59, 85] – *Proper Orthogonal Decomposition* (POD) –. La méthode que nous allons d'écrire dans ce chapitre suppose d'avoir accès aux opérateurs des EDPs, mais depuis quelques années une méthode des bases réduites non intrusives [33] – *Non Intrusive Reduced Basis (NIRB)* – a émergée. La méthode des NIRB permet de considérer la librairie utilisée pour résoudre les EDPs comme une boîte noire. Il est demandé à cette boîte noire de fournir des maillages pour différentes tailles de mailles données, ainsi que les solutions calculées sur ces maillages. Ensuite, à condition d'avoir des opérateurs

d'interpolation permettant de passer d'un maillage à l'autre, et de pouvoir utiliser les produits scalaires $L^2(\Omega)$ et $H^1(\Omega)$, la méthode des NIRB peut être mise en place.

Nous détaillerons dans ce chapitre la méthode CRBM appliquée aux problèmes elliptiques et paraboliques, admettant une dépendance affine en paramètres. Notons que le chapitre 3 proposera des solutions lorsque les opérateurs de l'EDP ne dépendent plus des paramètres de manière affine. Le calcul des estimateurs d'erreur sera explicité dans la section 3. Ensuite une méthode permettant de calculer, de manière efficace et fiable, une borne inférieure pour la constante de coercivité d'une forme bilinéaire sera étudiée. Enfin nous présenterons les résultats numériques obtenus sur deux modèles, elliptique et parabolique.

1 Problèmes elliptiques

1.1 Énoncé du problème général

Pour un jeu de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous nous intéressons à l'évaluation d'une sortie $s(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $u(\boldsymbol{\mu})$:

$$s(\boldsymbol{\mu}) = \ell\Big(u(\boldsymbol{\mu}); \boldsymbol{\mu}\Big), \tag{2.3}$$

pour un opérateur linéaire $\ell(.;\boldsymbol{\mu}) : X \times \mathcal{D} \to \mathbb{R}$ approprié. La formulation variationnelle de l'EDP consiste à chercher $u(\boldsymbol{\mu}) \in X$ tel que

$$a\Big(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}\Big),\ \forall v \in X, \tag{2.4}$$

où $a(.,.;\boldsymbol{\mu}) : X \times X \times \mathcal{D} \to \mathbb{R}$ et $f(.;\boldsymbol{\mu}) : X \times \mathcal{D} \to \mathbb{R}$ sont respectivement les formes bilinéaire et linéaire associées à l'EDP. Nous supposons ici que a est une forme bilinéaire symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((w,z)\big)_{a,\boldsymbol{\mu}} \equiv a\big(w,z;\boldsymbol{\mu}\big),\ \forall w,z \in X. \tag{2.5}$$

Notons que la norme associée à ce produit scalaire est

$$|||w|||_{a,\boldsymbol{\mu}} \equiv \sqrt{\big((w,w)\big)_{a,\boldsymbol{\mu}}},\ \forall w \in X. \tag{2.6}$$

De plus, nous supposons que la forme bilinéaire a est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, il existe une constante de continuité $\gamma^a(\boldsymbol{\mu})$ telle que

$$\gamma^a(\boldsymbol{\mu}) \equiv \sup_{v \in X} \frac{a(v,v;\boldsymbol{\mu})}{|||v|||^2_{a,\boldsymbol{\mu}_{ref}}} < \infty, \tag{2.7}$$

ainsi qu'une constante de coercivité $\alpha^a(\boldsymbol{\mu})$ telle que

$$\alpha^a(\boldsymbol{\mu}) \equiv \inf_{v \in X} \frac{a(v,v;\boldsymbol{\mu})}{|||v|||^2_{a,\boldsymbol{\mu}_{ref}}} > 0. \tag{2.8}$$

Un ingrédient important qui permet de rendre la méthode CRBM efficace est le développement d'une stratégie *hors-ligne/ en-ligne*. Cet ingrédient assure la viabilité de la méthode dans la pratique. Afin de mettre en oeuvre cette stratégie *hors-ligne/ en-ligne*. de manière efficace, a, f et l doivent pouvoir s'exprimer en faisant apparaître une dépendance affine en paramètres. Autrement dit il doit exister des entiers positifs Q_a, Q_f et Q_ℓ tels que $a(.,.;\boldsymbol{\mu})$, $f(.;\boldsymbol{\mu})$ et $\ell(.,\boldsymbol{\mu})$ puissent s'écrire de la manière suivante :

$$\begin{cases} a\left(u,v;\boldsymbol{\mu}\right) = \displaystyle\sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ a^q(u,v),\ \forall u,v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ f\left(v;\boldsymbol{\mu}\right) = \displaystyle\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu})\ f^q(v),\ \forall v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ \ell\left(v;\boldsymbol{\mu}\right) = \displaystyle\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu})\ \ell^q(v),\ \forall v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \end{cases} \tag{2.9}$$

où $\theta_a^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_a$, $\theta_f^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_f$ et $\theta_\ell^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_\ell$ sont des fonctions qui dépendent de $\boldsymbol{\mu}$.

1.2 Méthode des bases réduites

Étudions la construction de l'espace d'approximation de dimension réduite. Pour un ensemble de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, on commence avec la discrétisation FE des problèmes (2.3)-(2.4) qui consiste en l'évaluation de

$$s_{\mathcal{N}}(\boldsymbol{\mu}) = \ell\Big(u_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu}\Big), \tag{2.10}$$

avec $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ vérifiant

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),v;\boldsymbol{\mu}\Big) = f\Big(v;\boldsymbol{\mu}\Big),\ \forall v \in X_{\mathcal{N}}, \tag{2.11}$$

où nous avons $a : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, $f : X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$ et $\ell : X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$. Introduisons maintenant une séquence d'espaces d'approximation emboîtés $W_{N_{pr}}$ tels que $W_{1_{pr}} \subset W_{2_{pr}} \subset ... \subset W_{N_{max_{pr}}} \subset X_{\mathcal{N}}$, avec N_{max} un entier positif. Rappelons que l'espace $W_{N_{pr}}$ est de dimension N avec $N << \mathcal{N}$. Soit $S_N = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N\}$ un ensemble de N jeux de paramètres sélectionnés à l'aide de l'algorithme glouton dont une implémentation est donnée par l'algorithme (1). Soit S_N^u l'ensemble qui va contenir, pour chaque jeu de paramètres $\boldsymbol{\mu}$ dans S_N, la solution de (2.11). Cet ensemble s'écrit

$$S_N^u = \Big\{u_{\mathcal{N}}(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in S_N\Big\}. \tag{2.12}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble S_N^u pour avoir des fonctions de bases orthonormalisées ξ_n^{pr}, $1 \leq n \leq N$. Cette étape d'orthonormalisation est importante L'espace d'approximation $W_{N_{pr}}$ est défini par

$$W_{N_{pr}} = span\Big\{\xi_n^{pr},\ 1 \leq n \leq N\Big\}. \tag{2.13}$$

Quant à la solution réduite $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$, elle s'écrit

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu})\ \xi_i^{pr}. \tag{2.14}$$

Nous allons à présent utiliser la dépendance affine en paramètres pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{pr}$, pour $n = 1, ..., N$, alors pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ est solution de

$$\sum_{i=1}^{N} \left(\sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a^q\left(\xi_n^{pr}, \xi_i^{pr}\right) \right) u_{N_i}(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(\xi_n^{pr}) \right), \tag{2.15}$$

qui peut être également écrit sous la forme matricielle :

$$\left(\sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) A_N^q \right) u_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) F_N^q \right), \tag{2.16}$$

avec

$$\left(u_N(\boldsymbol{\mu})\right)_i = u_{N_i}(\boldsymbol{\mu}),\ \left(A_N^q\right)_{in} = a^q\left(\xi_n^{pr}, \xi_i^{pr}\right),\ \text{et}\ \left(F_N^q\right)_n = f^q\left(\xi_n^{pr}\right). \tag{2.17}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}) \ell^q(u_N(\boldsymbol{\mu})) \right),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.18}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}) {L_N^q}^T \right) u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.19}$$

avec $(L_N^q)_i = \ell^q(\xi_i^{pr})$.

La décomposition *hors-ligne/en-ligne* est maintenant évidente. Les fonctions de base ξ_i^{pr}, $1 \leq i \leq N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible la construction des matrices $A_N^q \in \mathbb{R}^{N \times N}$, $1 \leq q \leq Q_a$, et des vecteurs $F_N^q \in \mathbb{R}^N$, $1 \leq q \leq Q_f$ et $L_N^q \in R^N$, $1 \leq q \leq Q_\ell$. À l'étape *en-ligne*, pour chaque vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, on assemble la matrice $A_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) A_N^q$, ainsi que les vecteurs $F_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) F_N^q$ et $L_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}) L_N^q$. Enfin, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, nous résolvons le système réduit

$$A_N(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}) = F_N(\boldsymbol{\mu}), \tag{2.20}$$

et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = L_N^T(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.21}$$

Remarque 1. *Le fait d'avoir orthonormalisé les éléments de l'ensemble S_N^u permet de garantir un conditionnement de la matrice $A_N(\boldsymbol{\mu})$ borné, pour tout $\boldsymbol{\mu} \in \mathcal{D}$. En effet nous pouvons écrire*

$$cond\Big(A_N(\boldsymbol{\mu})\Big) \geq \frac{\gamma^a(\boldsymbol{\mu})}{\alpha^a(\boldsymbol{\mu})},\ \forall \mu \in \mathcal{D}. \tag{2.22}$$

Dans le cas contraire le conditionnement de la matrice de dimension réduite peut devenir très important et mener à des instabilités numériques.

Jusqu'ici nous avons exposé une approche uniquement *primale* pour évaluer la sortie. Dans le cas de sorties dites "souples" (ou *compliant*) – auquel cas a est symétrique et $l = f$ – cette approche suffit pour avoir une convergence quadratique. En effet, grâce à la symétrie de a et à la propriété d'orthogonalité de Galerkin, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\begin{aligned} s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu}) &= \ell\big(u_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}\big) - \ell\big(u_N(\boldsymbol{\mu}); \boldsymbol{\mu}\big) \\ &= f\big(u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}); \boldsymbol{\mu}\big) \\ &= a\big(u_{\mathcal{N}}, u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}); \boldsymbol{\mu}\big) \\ &= a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}), u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}); \boldsymbol{\mu}\Big). \end{aligned} \tag{2.23}$$

Donc nous avons bien

$$s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu}) = \Big|\Big|\Big|u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\Big|\Big|\Big|^2_{a,\boldsymbol{\mu}},\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.24}$$

Cependant dans le cas plus général de sorties non "souples", nous perdons cette convergence quadratique. Afin de la retrouver nous allons à présent introduire une approche *primale-duale*. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, le problème dual associé à la sortie du modèle consiste à chercher $\Psi(\boldsymbol{\mu}) \in X$ tel que

$$a\Big(v, \Psi(\boldsymbol{\mu}); \boldsymbol{\mu}\Big) = -\ell\Big(v; \boldsymbol{\mu}\Big),\ \forall v \in X. \tag{2.25}$$

Ψ est la variable duale – dans le cas de sorties souples nous avons $\Psi = -u$ –. Pour un jeu de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné la discrétisation FE du problème (2.25) consiste à chercher $\Psi_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que

$$a\Big(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}\Big) = -\ell\Big(v; \boldsymbol{\mu}\Big),\ \forall v \in X_{\mathcal{N}}. \tag{2.26}$$

De la même manière que pour l'approche *primale*, nous introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{du}}$, $1 \leq N \leq N_{max}$. S_N^{Ψ} est l'ensemble des solutions de (2.26), nous pouvons écrire

$$S_N^{\Psi} = \Big\{\Psi_{\mathcal{N}}(\boldsymbol{\mu}_i),\ \forall \boldsymbol{\mu}_i \in S_N\Big\}. \tag{2.27}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble S_N^{Ψ} pour avoir des fonctions de bases orthonormalisées ξ^{du}, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{du}}$ est défini par

$$W_{N_{du}} = span\Big\{\xi_n^{du},\ 1 \leq n \leq N\Big\}. \tag{2.28}$$

Quant à la solution réduite $\Psi_N(\boldsymbol{\mu}) \in W_{N_{du}}$, elle s'écrit

$$\Psi_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} \Psi_{Ni}(\boldsymbol{\mu})\ \xi_i^{du}. \tag{2.29}$$

Notons que nous avons présenté une approche ségrégée – nous aurions pu choisir une approche intégrée avec un seul espace d'approximation comprenant à la fois les fonctions de base primale et duale – et dans ce cas l'évaluation de la sortie est donnée par

$$s_N(\boldsymbol{\mu}) = \ell\big(u_N(\boldsymbol{\mu});\boldsymbol{\mu}\big) - r_{pr}\big(\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}\big),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.30}$$

avec

$$r_{pr}\big(v;\boldsymbol{\mu}\big) = f\big(v;\boldsymbol{\mu}\big) - a\big(u_N(\boldsymbol{\mu}),v;\boldsymbol{\mu}\big)\ \text{ et }\ r_{du}\big(v;\boldsymbol{\mu}\big) = -\ell\big(v;\boldsymbol{\mu}\big) - a\big(v,\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}\big). \tag{2.31}$$

Grâce à l'introduction du problème dual, nous avons récupéré une convergence quadratique. En effet pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\begin{array}{rcl} \big|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})\big| & = & \big|\ell\big(u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu});\boldsymbol{\mu}\big)\big| \\ & = & \big|a\big(u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}),\Psi_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu}\big)\big| \\ & = & \big|a\big(u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}),\Psi_{\mathcal{N}}(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}\big)\big|, \end{array} \tag{2.32}$$

puis en nous servant de l'inégalité de Cauchy-Schwarz, nous pouvons écrire

$$\big|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})\big| \le \big|\big|\big|u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\big|\big|\big|_{a,\boldsymbol{\mu}}\ \big|\big|\big|\Psi_{\mathcal{N}}(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu})\big|\big|\big|_{a,\boldsymbol{\mu}},\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.33}$$

Avec cette approche *primale-duale*, pour construire l'approximation base réduite – *Reduced Basis (RB)* –, il est nécessaire de résoudre le problème primal et le problème dual :

$$a(u_N(\boldsymbol{\mu}),v;\boldsymbol{\mu}) = f(v;\boldsymbol{\mu}),\ \forall v \in W_{N_{pr}}\ \text{et}\ a(v,\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}) = -\ell(v;\boldsymbol{\mu}),\ \forall v \in W_{N_{du}}. \tag{2.34}$$

Nous allons maintenant utiliser la dépendance affine en paramètres pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{du},\ n = 1,...,N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $\Psi_N(\boldsymbol{\mu})$ est solution de

$$-\sum_{i=1}^{N}\left(\sum_{q=1}^{Q_a}\theta_a^q(\boldsymbol{\mu})a^q\left(\xi_n^{du},\xi_i^{du}\right)\right)\ \Psi_{N_i}(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu})\ell^q(\xi_n^{du})\right), \tag{2.35}$$

qui peut être également écrit sous la forme matricielle :

$$-\left(\sum_{q=1}^{Q_a}\theta_a^q(\boldsymbol{\mu})A_N^q\right)\Psi_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu})L_N^q\right), \tag{2.36}$$

avec

$$\Big(\Psi_N(\boldsymbol{\mu})\Big)_i = \Psi_{N_i}(\boldsymbol{\mu}),\ \Big(A_N^q\Big)_{in} = a^q\Big(\xi_n^{du},\xi_i^{du}\Big)\ \text{et}\ \Big(L_N^q\Big)_n = \ell^q\Big(\xi_n^{du}\Big). \tag{2.37}$$

L'apparition dans (2.30) du terme de correction $r_{pr}(\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu})$ nécessite de manipuler des matrices $A_N^{prdu,q} \in \mathbb{R}^{N\times N}$, $1 \le q \le Q_a$, construites à partir des éléments de la base primale et duale. D'un point de vue matriciel, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, la sortie s'écrit

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu}){L_N^q}^T\right)\ u_N(\boldsymbol{\mu})\ -\ R_{pr}^{\Psi_N(\boldsymbol{\mu})}, \tag{2.38}$$

avec

$$R_{pr}^{\Psi_N(\boldsymbol{\mu})} = \left(\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) F_N^{du,q} \right) \Psi_N(\boldsymbol{\mu}) \; - \; \Psi_N(\boldsymbol{\mu})^T \left(\sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) A_N^{prdu,q} \right) u_N(\boldsymbol{\mu}), \quad (2.39)$$

ainsi que

$$\left(A_N^{prdu,q} \right)_{in} = a^q\left(\xi_n^{pr}, \xi_i^{du} \right) \text{ et } \left(F_N^{du,q} \right)_n = f^q\left(\xi_n^{du} \right). \quad (2.40)$$

Revenons sur la construction de l'ensemble S_N par l'algorithme glouton. Afin d'introduire cet algorithme, qui sert également à construire les espaces d'approximation $W_{N_{pr}}$ et $W_{N_{du}}$ durant la partie *hors-ligne*, nous définissons un échantillonnage Ξ_{train}, de taille n_{train}, de l'espace des paramètres dans lequel les paramètres seront sélectionnés. Étudions comment est construit un tel échantillonnage. L'espace des paramètres $\mathcal{D} \subset \mathbb{R}^P$ peut être vu comme une boîte défini par $\mathcal{D} \equiv [\mu_{min,1}, \mu_{max,1}] \times ... \times [\mu_{min,P}, \mu_{max,P}]$, où $\mu_{min,1}$,...,$\mu_{min,P}$ sont les bornes inférieures de $\mathcal{D}$ et $\mu_{max,1}$,...,$\mu_{max,P}$ en sont les bornes supérieures. Commençons par considérer le cas où l'espace de paramètre $\mathcal{D}$ est de dimension 1, c'est à dire $\mathcal{D} \subset \mathbb{R}^1$ est défini par $\mathcal{D} \equiv [\mu_{min,1}, \mu_{max,1}]$. Lorsque nous avons

$$0 \leq \mu_{min,1} < \mu_{max,1}, \quad (2.41)$$

alors les éléments de Ξ_{train} sont définis de la manière suivante :

$$\mu_i = exp\Big(ln(\mu_{min,1}) + \big(\; ln(\mu_{max,1}) - ln(\mu_{min,1}) \; \big) \; rand \Big), \; 1 \leq i \leq n_{train}, \quad (2.42)$$

où *rand* est une variable aléatoire uniformément distribuée sur $[0, 1]$. Dans le cas où (2.41) n'est pas vérifiée, c'est à dire que nous avons $\mu_{min,1} < 0$ alors les éléments de Ξ_{train} sont définis par :

$$\mu_i = \mu_{min,1} + \big(\mu_{max,1} - \mu_{min,1} \big) \; rand, \; 1 \leq i \leq n_{train}. \quad (2.43)$$

Dans le cas où l'espace de paramètre $\mathcal{D}$ est de dimension supérieure à 1 alors chaque composante des éléments de Ξ_{train} est construit en appliquant (2.42) ou (2.43).

Algorithm 1 Algorithme glouton
$[S_N, W_{N_{pr}}, W_{N_{du}}] = glouton(N_{max}, \epsilon_{tol}, \Xi_{train})$

$\boldsymbol{\mu}_1$ est choisi aléatoirement dans Ξ_{train} ;
$\epsilon \leftarrow \infty$;
$S_1 \leftarrow \boldsymbol{\mu}_1$;
$N \leftarrow 1$;
$W_{1_{pr}} \leftarrow u_{\mathcal{N}}(\boldsymbol{\mu}_1)$; $W_{1_{du}} \leftarrow \Psi_{\mathcal{N}}(\boldsymbol{\mu}_1)$;
while $\epsilon > \epsilon_{tol}$ et $N \leq N_{max}$ **do**
  $\boldsymbol{\mu}_N \leftarrow \arg \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta_{N-1}^s(\boldsymbol{\mu})$;
  $\epsilon \leftarrow \Delta_{N-1}^s(\boldsymbol{\mu}_N)$;
  $S_N \leftarrow S_{N-1} \cup \boldsymbol{\mu}_N$;
  $N \leftarrow N + 1$;
  $W_{N_{pr}} \leftarrow W_{N-1_{pr}} \cup span\{u_{\mathcal{N}}(\boldsymbol{\mu}_N)\}$; $W_{N_{du}} \leftarrow W_{N-1_{du}} \cup span\{\Psi_{\mathcal{N}}(\boldsymbol{\mu}_N)\}$;
end while

Le premier ensemble de paramètres $\boldsymbol{\mu}_1$ est choisi de manière aléatoire dans Ξ_{train}. En revanche les suivants maximisent l'erreur commise entre l'approximation RB et FE sur la sortie, calculée par $\Delta^s_{N-1}(\boldsymbol{\mu})$ pour chaque élément de Ξ_{train}. Il serait évidemment trop coûteux de calculer la véritable erreur $e^s(\boldsymbol{\mu}) = s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})$, c'est pourquoi un estimateur d'erreur est utilisé. $\Delta^s_{N-1}(\boldsymbol{\mu})$ fournit une borne supérieure de $e^s(\boldsymbol{\mu})$ en se servant des $N-1$ premiers éléments de la base réduite. La section 3 est dédiée à la mise en place d'un tel estimateur d'erreur. À chaque itération de l'algorithme glouton, Pour chaque nouveau $\boldsymbol{\mu}$ dans S_N, les espaces $W_{N_{pr}}$ et $W_{N_{du}}$ sont enrichis par les solutions des problèmes (2.11) et (2.26), puis ces nouveaux éléments RB sont orthonormalisés.

Durant l'étape *hors-ligne* il est nécessaire de faire $2N$ résolutions FE (coûteuses) des problèmes primal et dual et $\mathcal{O}(Q_a N^2 \mathcal{N})$ produits scalaires pour les étapes de projection sur la base réduite. En revanche la complexité de l'étape *en-ligne* ne dépend plus de la (grande) dimension $\mathcal{N}$ puisqu'elle implique $\mathcal{O}(Q_a N^2)$ multiplications pour assembler le système réduit, primal ou dual, et $\mathcal{O}(N^3)$ pour le résoudre. Ensuite, il faut compter $\mathcal{O}(N)$ produits scalaires pour avoir la sortie du modèle en utilisant (2.19) et $\mathcal{O}(N^3)$ lorsque le terme de correction est ajouté (2.38).

2 Problèmes paraboliques

2.1 Énoncé du problème général

Pour les problèmes paraboliques, nous introduisons l'intervalle de temps $I = (0; T_f)$ où T_f est le temps final. Pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, et un temps $t \in I$, nous nous intéressons à l'évaluation d'une sortie $s(\boldsymbol{\mu}, t) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $u(\boldsymbol{\mu}, t)$:

$$s(\boldsymbol{\mu}, t) = \ell\Big(u(\boldsymbol{\mu}, t); \boldsymbol{\mu}; t\Big), \tag{2.44}$$

pour un opérateur linéaire $\ell(.; \boldsymbol{\mu}) : X \times \mathcal{D} \times I \to \mathbb{R}$ approprié. La formulation variationnelle de l'EDP consiste à chercher $u(\boldsymbol{\mu}, t) \in X$ tel que

$$m\Big(\frac{\partial u(\boldsymbol{\mu}, t)}{\partial t}, v; \boldsymbol{\mu}\Big) + a\Big(u(\boldsymbol{\mu}, t), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}; t\Big),\ \forall v \in X,\ \forall t \in I, \tag{2.45}$$

où $m(., .; \boldsymbol{\mu}) : X \times X \times \mathcal{D} \to \mathbb{R}$, $a(., .; \boldsymbol{\mu}) : X \times X \times \mathcal{D} \to \mathbb{R}$ et $f(.; \boldsymbol{\mu}; t) : X \times \mathcal{D} \times I \to \mathbb{R}$ sont respectivement les formes bilinéaires et linéaire associées à l'EDP. Comme précédemment, nous supposons que a est symétrique, définie postive, continue et coercive, voir (2.7) et (2.8). Nous supposons également que m est symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((., .)\big)_{m,\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((w, z)\big)_{m,\boldsymbol{\mu}} \equiv m\big(w, z; \boldsymbol{\mu}\big),\ \forall w, z \in X. \tag{2.46}$$

Notons que la norme associée à ce produit scalaire est

$$|||w|||_{m,\boldsymbol{\mu}} \equiv \sqrt{\big((w, w)\big)_{m,\boldsymbol{\mu}}},\ \forall w \in X. \tag{2.47}$$

De plus, nous suposons que m est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ il existe une constante de continuité $\gamma^m(\boldsymbol{\mu})$ telle que

$$\gamma^m(\boldsymbol{\mu}) \equiv \sup_{v \in X} \frac{m(v, v; \boldsymbol{\mu})}{|||v|||^2_{m,\boldsymbol{\mu}_{ref}}} < \infty, \tag{2.48}$$

ainsi qu'une constante de coercivité $\alpha^m(\boldsymbol{\mu})$ telle que

$$\alpha^m(\boldsymbol{\mu}) \equiv \inf_{v \in X} \frac{m(v, v; \boldsymbol{\mu})}{|||v|||^2_{m, \boldsymbol{\mu}_{ref}}} > 0. \tag{2.49}$$

Afin de mettre en oeuvre une stratégie *hors-ligne/en-ligne* de manière efficace, nous supposons qu'il existe des entiers positifs Q_m, Q_a, Q_f et Q_ℓ tels que $m(.,.;\boldsymbol{\mu})$, $a(.,.;\boldsymbol{\mu})$, $f(.;\boldsymbol{\mu};t)$ et $\ell(.,\boldsymbol{\mu};t)$ puissent s'écrire de la manière suivante :

$$\begin{cases} m(u, v; \boldsymbol{\mu}) = \displaystyle\sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu})\ m^q(u, v),\ \forall u, v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ a(u, v; \boldsymbol{\mu}) = \displaystyle\sum_{q=1}^{Q_q} \theta_a^q(\boldsymbol{\mu})\ a^q(u, v),\ \forall u, v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ f(v; \boldsymbol{\mu}; t) = \displaystyle\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}; t)\ f^q(v),\ \forall v \in X,\ \forall t \in I,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ \ell(v; \boldsymbol{\mu}; t) = \displaystyle\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}; t)\ \ell^q(v),\ \forall v \in X,\ \forall t \in I,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \end{cases} \tag{2.50}$$

où $\theta_m^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_m$, $\theta_a^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_a$, $\theta_f^q : \mathcal{D} \times I \to \mathbb{R}$, $1 \leq q \leq Q_f$ et $\theta_\ell^q : \mathcal{D} \times I \to \mathbb{R}$, $1 \leq q \leq Q_\ell$ sont des fonctions qui dépendent de $\boldsymbol{\mu}$.

2.2 Méthode des bases réduites

Étudions la construction de l'espace d'approximation de dimension réduite. Pour un jeu de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné on commence avec la discrétisation – FE en espace, différences finies en temps – des problèmes (2.44)-(2.45). Pour cela nous divisons l'intervalle de temps I en K sous-intervalles avec K un entier positif. Notons Δt le pas de temps donné par $\Delta t = \frac{T_f}{K}$ et par conséquent nous avons $t^k = k\ \Delta t$, $k \in \mathbb{K}$ avec $\mathbb{K} = \{0, ..., K\}$. Pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ et $k \in \mathbb{K}$ donnés, la discrétisation des problèmes (2.44)-(2.45) consiste en l'évaluation de

$$s_{\mathcal{N}}(\boldsymbol{\mu}, t^k) = \ell\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}; t^k\Big), \tag{2.51}$$

avec $u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}$ vérifiant, pour tout $v \in X_{\mathcal{N}}$:

$$\begin{cases} \dfrac{1}{\Delta t} m\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_{\mathcal{N}}(\boldsymbol{\mu}, t^{k-1}), v; \boldsymbol{\mu}\Big) + a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}; t^k\Big),\ 1 \leq k \leq K, \\ \Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^0), v\Big)_{L^2(\Omega)} = \Big(u_{ini}(\boldsymbol{\mu}), v\Big)_{L^2(\Omega)}, \end{cases} \tag{2.52}$$

où $u_{ini}(\boldsymbol{\mu}) \in L^2(\Omega)$, $a : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, $m : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, $f : X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$ et $\ell : X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$.

Nous avons fait le choix d'utiliser un schéma en temps d'Euler, cependant l'utilisation d'autres schémas tels que BDF – Backward Differentiation Formula – d'ordre supérieur à 1, ou encore Crank-Nicholson est possible.

Comme dans le cas elliptique, nous introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{pr}}$ tels que $W_{1_{pr}} \subset W_{2_{pr}} \subset ... \subset W_{N_{max_{pr}}} \subset X_{\mathcal{N}}$, avec N_{max} un entier positif. $S_{\bar{N}} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_{\bar{N}}\}$ est un ensemble de $\bar{N}$ jeux de paramètres, avec $1 \leq \bar{N} \leq N$, sélectionnés à l'aide de l'algorithme POD/glouton – ou *POD/greedy* – qui combine une approche POD en temps et glouton en espace, dont une implémentation est donnée par l'algorithme (3). Présentons la décomposition orthogonale aux valeurs propres via la méthode des instantanés – ou *POD via snapshots method* – (voir l'algorithme 2). Considérons un entier N_m tel que $1 \leq N_m \leq K$, ainsi que pour un vecteur de paramètres $\boldsymbol{\mu}$ donné l'ensemble des solutions de (2.52) $Snap^u(\boldsymbol{\mu}) = \{u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}, 1 \leq k \leq K\}$. Soit $Y_{N_m}(\boldsymbol{\mu})$ un espace linéaire de dimension N_m, et de sorte que $Y_{N_m}(\boldsymbol{\mu}) \subset Snap^u(\boldsymbol{\mu})$. L'algoritme 2 renvoie N_m fonctions orthonormales au sens du produit scalaire $(.,.)_X$ $\{\rho_i(\boldsymbol{\mu}) \in X_{\mathcal{N}}, 1 \leq i \leq N_m\}$ de sorte que l'espace $\tilde{\rho}_{N_m}(\boldsymbol{\mu})$ engendré par ces fonctions soit au plus près des instantanés [76], dans le sens où l'on a

$$\tilde{\rho}_{N_m}(\boldsymbol{\mu}) = \arg \inf_{Y_{N_m}} \left(\frac{1}{K} \sum_{k=1}^{K} \inf_{v \in Y_{N_m}} \left| \left| \left| u_{\mathcal{N}}^k(\boldsymbol{\mu}) - v \right| \right| \right|_{a,\boldsymbol{\mu}_{ref}}^2 \right). \tag{2.53}$$

Notons que nous avons la relation :

$$\bar{N} = \frac{N}{N_m}. \tag{2.54}$$

Algorithm 2 $\{\rho_i(\boldsymbol{\mu}) \in X_{\mathcal{N}}, 1 \leq i \leq N_m\} = POD(Snap^u(\boldsymbol{\mu}), N_m)$

Construire la matrice M^{POD} telle que, pour $i = 1, ..., K$ et $j = 1, ..., K$:

$$\left(M^{POD}\right)_{ij} = \left(u_{\mathcal{N}}(\boldsymbol{\mu}, t^i), u_{\mathcal{N}}(\boldsymbol{\mu}, t^j)\right)_{a,\boldsymbol{\mu}_{ref}}. \tag{2.55}$$

Résoudre $M^{POD}\ \varphi_i = \lambda_i\ \varphi_i$ pour $(\varphi_i \in \mathbb{R}^K, \lambda_i \in \mathbb{R})_{1 \leq i \leq N_m}$ associés aux N_m plus grandes valeurs propres de M^{POD}.
Construire

$$\rho_i(\boldsymbol{\mu}) = \sum_{k=1}^{K} \varphi_k\ u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \text{ avec } 1 \leq i \leq N_m. \tag{2.56}$$

Notons $S_{\bar{N}}^u$ l'ensemble qui va contenir, pour chaque $\boldsymbol{\mu} \in S_{\bar{N}}$, les N_m modes propres principaux issus de $POD(Snap^u(\boldsymbol{\mu}), N_m)$. Cet ensemble s'écrit

$$S_{\bar{N}}^u = \left\{ POD(Snap^u(\boldsymbol{\mu}), N_m),\ \forall \boldsymbol{\mu} \in S_{\bar{N}} \right\}. \tag{2.57}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\left((.,.)\right)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble $S_{\bar{N}}^u$ pour avoir des fonctions de bases orthonormalisées ξ_n^{pr}, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{pr}}$ est défini par

$$W_{N_{pr}} = span\left\{ \xi_n^{pr},\ 1 \leq n \leq N \right\}. \tag{2.58}$$

Quant à la solution réduite $u_N(\boldsymbol{\mu}, t^k) \in W_{N_{pr}}$ elle s'écrit

$$u_N(\boldsymbol{\mu}, t^k) = \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu}, t^k)\ \xi_i^{pr}. \tag{2.59}$$

Nous allons maintenant utiliser la dépendance affine en paramètres pour construire une stratégie *hors-ligne/ en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{pr}$, $i = n, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $u_N(\boldsymbol{\mu}, t^k)$ est solution de

$$\begin{aligned} &\sum_{i=1}^{N} \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) m^q\left(\xi_n^{pr}, \xi_i^{pr}\right) + \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a^q\left(\xi_n^{pr}, \xi_i^{pr}\right) \right)\ u_{N_i}(\boldsymbol{\mu}; t^k) \\ &= \left(\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}; t^k) f^q(\xi_n^{pr}) + \frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) m^q\Big(u_N(\boldsymbol{\mu}; t^{k-1}), \xi_n^{pr}\Big) \right),\ 1 \le k \le K, \end{aligned} \tag{2.60}$$

qui peut être également écrit sous la forme matricielle :

$$\begin{aligned} &\left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) M_N^q + \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) A_N^q \right) u_N(\boldsymbol{\mu}, t^k) \\ &= \left(\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}; t^k) F_N^q + \frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) M_N^q\ u_N(\boldsymbol{\mu}; t^{k-1}) \right),\ 1 \le k \le K, \end{aligned} \tag{2.61}$$

avec

$$\begin{aligned} &\Big(u_N(\boldsymbol{\mu}, t^k)\Big)_i = u_{N_i}(\boldsymbol{\mu}; t^k),\ \Big(M_N^q\Big)_{in} = m^q\Big(\xi_n^{pr}, \xi_i^{pr}\Big), \\ &\Big(A_N^q\Big)_{in} = a^q\Big(\xi_n^{pr}, \xi_i^{pr}\Big) \text{ et } \Big(F_N^q\Big)_n = f^q\Big(\xi_n^{pr}\Big), \end{aligned} \tag{2.62}$$

et $\big(u_N(\boldsymbol{\mu}, t^0), v\big)_{L^2(\Omega)} = \big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^0), v\big)_{L^2(\Omega)}$, $\forall v \in W_{N_{pr}}$ à l'instant initial.
La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}; t^k) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}; t^k) \ell^q\Big(u_N(\boldsymbol{\mu}, t^k)\Big) \right),\ \forall \boldsymbol{\mu} \in \mathcal{D},\ \forall k \in \mathbb{K}, \tag{2.63}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}, t^k) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}; t^k) {L_N^q}^T \right) u_N(\boldsymbol{\mu}, t^k),\ \forall \boldsymbol{\mu} \in \mathcal{D},\ \forall k \in \mathbb{K}, \tag{2.64}$$

avec $(L_N^q)_n = \ell^q(\xi_n^{pr})$.

La construction des matrices $A_N^q \in \mathbb{R}^{N \times N}$, $1 \le q \le Q_a$, M_N^q,$1 \le q \le Q_m$, et des vecteurs $F_N^q \in \mathbb{R}^N$, $1 \le q \le Q_f$ et $L_N^q \in \mathbb{R}^N$, $1 \le q \le Q_\ell$ se fait durant l'étape *hors-ligne*. À l'étape *en-ligne*, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et $k \in \mathbb{K}$ donnés, on assemble les matrices $A_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ A_N^q$ et $M_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu})\ M_N^q$ ainsi que les vecteurs $F_N(\boldsymbol{\mu}; t^k) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}; t^k)\ F_N^q$ et $L_N(\boldsymbol{\mu}; t^k) = \sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}; t^k)\ L_N^q$. Enfin, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et $k = 1, ..., K$, nous résolvons le système réduit

$$\left(\frac{1}{\Delta t} M_N(\boldsymbol{\mu}) + A_N(\boldsymbol{\mu}) \right)\ u_N(\boldsymbol{\mu}, t^k) = \left(F_N(\boldsymbol{\mu}; t^k) + \frac{1}{\Delta t} M_N(\boldsymbol{\mu}) u_N(\boldsymbol{\mu}; t^{k-1}) \right), \tag{2.65}$$

et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}, t^k) = L_N^T(\boldsymbol{\mu}; t^k)\ u_N(\boldsymbol{\mu}, t^k),\ \forall \boldsymbol{\mu} \in D,\ \forall k \in \mathbb{K}. \tag{2.66}$$

Avec cette approche *primale*, si on ne considère que les sorties "souples", nous avons une convergence quadratique. Si on introduit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, la norme suivante :

$$\Big|\Big|\Big|w(t^k)\Big|\Big|\Big|_{pr,\boldsymbol{\mu}} \equiv \sqrt{m\Big(w(t^k), w(t^k); \boldsymbol{\mu}\Big) + \sum_{k'=1}^{k} a\Big(w(t^{k'}), w(t^{k'}); \boldsymbol{\mu}\Big)\Delta t},\ \forall w \in X_{\mathcal{N}}, 1 \leq k \leq K, \tag{2.67}$$

alors, de manière analogue au cas elliptique, nous avons

$$s_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - s_N(\boldsymbol{\mu}, t^k) = \frac{1}{\Delta t}\Big|\Big|\Big|u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_N(\boldsymbol{\mu}, t^k)\Big|\Big|\Big|^2_{pr,\boldsymbol{\mu}},\ \forall \boldsymbol{\mu} \in \mathcal{D},\ 1 \leq k \leq K. \tag{2.68}$$

Comme dans le cas elliptique, nous allons introduire à présent une approche *primale-duale*. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, le problème dual associé à la sortie du modèle consiste à chercher $\Psi(\boldsymbol{\mu}, t) \in X$ tel que

$$m\Big(v, \frac{\partial \Psi(\boldsymbol{\mu}, t)}{\partial t}; \boldsymbol{\mu}\Big) + a\Big(v, \Psi(\boldsymbol{\mu}, t); \boldsymbol{\mu}\Big) = 0,\ \forall v \in X, \forall t \in I, \tag{2.69}$$

la discrétisation – FE en espace, différences finies en temps – du problème (2.69) consiste à chercher, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $\Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}$ tel que, pour tout $v \in X_{\mathcal{N}}$:

$$\begin{cases} \dfrac{1}{\Delta t} m\Big(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}\Big) + a\Big(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}\Big) = \dfrac{1}{\Delta t} m\Big(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^{k+1}); \boldsymbol{\mu}\Big),\ K \geq k \geq 1 \\ m\Big(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^{K+1}); \boldsymbol{\mu}\Big) = \ell\Big(v; \boldsymbol{\mu}; t^{K+1}\Big). \end{cases} \tag{2.70}$$

De la même manière que pour l'approche *primale* nous introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{du}}$, $1 \leq N \leq N_{max}$. Définissons $Snap^{\Psi}(\boldsymbol{\mu})$ l'ensemble des solution du problème dual (2.70) pour $\boldsymbol{\mu} \in \mathcal{D}$ fixé. Nous avons $Snap^{\Psi}(\boldsymbol{\mu}) = \{\Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}, K \geq k \geq 1\}$. $S_{\bar{N}}^{\Psi}$ est l'ensemble défini par

$$S_{\bar{N}}^{\Psi} = \Big\{POD(Snap^{\Psi}(\boldsymbol{\mu}), N_m),\ \forall \boldsymbol{\mu} \in S_{\bar{N}}\Big\}. \tag{2.71}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble $S_{\bar{N}}^{\Psi}$ pour avoir des fonctions de bases orthonormalisées ξ^{du}, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{du}}$ est défini par

$$W_{N_{du}} = span\Big\{\xi_n^{du},\ 1 \leq n \leq N\Big\}. \tag{2.72}$$

Quant à la solution réduite $\Psi_N(\boldsymbol{\mu}, t^k) \in W_{N_{du}}$ elle s'écrit

$$\Psi_N(\boldsymbol{\mu}, t^k) = \sum_{i=1}^{N} \Psi_{Ni}(\boldsymbol{\mu}, t^k)\ \xi_i^{du}. \tag{2.73}$$

L'évaluation de la sortie, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et $k \in \mathbb{K}$, est donnée par

$$s_N(\boldsymbol{\mu}, t^k) = \ell\Big(u_N(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}; t^k\Big) + \sum_{k'=1}^{k} r_{pr}\Big(\Psi_N(\boldsymbol{\mu}, t^{K-k+k'}); \boldsymbol{\mu}; t^{k'}\Big)\ \Delta t, \tag{2.74}$$

avec pour tout $v \in X_{\mathcal{N}}$ et $1 \leq k \leq K$:

$$r_{pr}(v; \boldsymbol{\mu}; t^k) = f\left(v; \boldsymbol{\mu}; t^k\right) - a\left(u_N(\boldsymbol{\mu}, t^k), v; \boldsymbol{\mu}\right) - \frac{1}{\Delta t} m\left(u_N(\boldsymbol{\mu}, t^k) - u_N(\boldsymbol{\mu}, t^{k-1}); \boldsymbol{\mu}\right), \tag{2.75}$$

ainsi que, pour tout $v \in X_{\mathcal{N}}$ et $K \geq k \geq 1$:

$$r_{du}(v; \boldsymbol{\mu}; t^k) = -a\left(v, \Psi_N(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}\right) - \frac{1}{\Delta t} m\left(v, \Psi_N(\boldsymbol{\mu}, t^k) - \Psi_N(\boldsymbol{\mu}, t^{k+1}); \boldsymbol{\mu}\right). \tag{2.76}$$

Nous allons maintenant utiliser la dépendance affine en paramètres pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{du},\ i = n, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $\Psi_N(\boldsymbol{\mu}, t^k)$ est solution de

$$\begin{aligned} &\sum_{i=1}^{N} \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) m^q\left(\xi_n^{du}, \xi_i^{du}\right) + \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a^q\left(\xi_n^{du}, \xi_i^{du}\right) \right)\ \Psi_{N_i}(\boldsymbol{\mu}; t^k) \\ &= \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) m^q\Big(\Psi_N(\boldsymbol{\mu}, t^{k+1}), \xi_n^{du}\Big) \right),\ K \leq k \leq 1, \end{aligned} \tag{2.77}$$

qui peut être également écrit, pour $k = K, ..., 1$, sous la forme matricielle :

$$\left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) M_N^q + \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) A_N^q \right) \Psi_N(\boldsymbol{\mu}, t^k) = \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) M_N^q\ \Psi_N(\boldsymbol{\mu}, t^{k+1}) \right), \tag{2.78}$$

avec

$$\Big(\Psi_N(\boldsymbol{\mu}, t^k)\Big)_i = \Psi_{N_i}(\boldsymbol{\mu}; t^k),\ \Big(M_N^q\Big)_{in} = m^q\Big(\xi_n^{du}, \xi_i^{du}\Big),\ \Big(A_N^q\Big)_{in} = a^q\Big(\xi_n^{du}, \xi_i^{du}\Big). \tag{2.79}$$

Au temps final t^{K+1} nous avons

$$\Big(\sum_{q=1}^{Q_m} \theta_m^q(\boldsymbol{\mu}) M_N^q \Big) \Psi_N(\boldsymbol{\mu}, t^k) = \Big(\sum_{q=1}^{Q_\ell} \theta_m^q(\boldsymbol{\mu}; t^{K+1}) L_N^q \Big) \tag{2.80}$$

avec

$$\Big(L_N^q\Big)_n = \ell^q\Big(\xi_n^{du}\Big). \tag{2.81}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}, t^k) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}; t^k) {L_N^q}^T \right) u_N(\boldsymbol{\mu}, t^k) + \sum_{k'=1}^{k} R_{pr}^{\Psi_N(\boldsymbol{\mu}, t^{K-k+k'})},\ \forall \boldsymbol{\mu} \in \mathcal{D},\ \forall k \in \mathbb{K}, \tag{2.82}$$

avec

$$\begin{aligned} R_{pr}^{\Psi_N(\boldsymbol{\mu},t^{K-k+k'})} &= \left(\sum_{q=1}^{Q_f}\theta_f^q(\boldsymbol{\mu};t^{k'})F_N^{du,q}\right)\Psi_N(\boldsymbol{\mu},t^{K-k+k'}) \\ &\quad -\Psi_N(\boldsymbol{\mu},t^{K-k+k'})^T\left(\sum_{q=1}^{Q_a}\theta_a^q(\boldsymbol{\mu})A_N^{prdu,q}\right)u_N(\boldsymbol{\mu},t^k) \\ &\quad -\frac{1}{\Delta t}\Psi_N(\boldsymbol{\mu},t^{K-k+k'})^T\left(\sum_{q=1}^{Q_m}\theta_a^q(\boldsymbol{\mu})M_N^{prdu,q}\right)\Big(u_N(\boldsymbol{\mu},t^k)-u_N(\boldsymbol{\mu},t^{k-1})\Big) \end{aligned} \tag{2.83}$$

ainsi que

$$\left(A_N^{prdu,q}\right)_{in}=a^q\left(\xi_n^{pr},\xi_i^{du}\right),\left(M_N^{prdu,q}\right)_{in}=m^q\left(\xi_n^{pr},\xi_i^{du}\right)\text{ et }\left(F_N^{du,q}\right)_n=f^q\left(\xi_n^{du}\right), \tag{2.84}$$

où $A_N^{prdu,q}\in\mathbb{R}^{N\times N}$, $1\le q\le Q_a$, $M_N^{prdu,q}\in\mathbb{R}^{N\times N}$, $1\le q\le Q_m$ et $F_N^{du,q}\in\mathbb{R}^N$, $1\le q\le Q_f$.

L'introduction du problème dual nous a permis d'avoir une convergence quadratique pour les sorties non "souples". En effet en définissant la norme suivante pour tout $\boldsymbol{\mu}\in\mathcal{D}$

$$\Big|\Big|\Big|w(t^k)\Big|\Big|\Big|_{du,\boldsymbol{\mu}}\equiv\sqrt{m\Big(w(t^k),w(t^k);\boldsymbol{\mu}\Big)+\sum_{k'=k}^{K}a\Big(w(t^{k'}),w(t^{k'});\boldsymbol{\mu}\Big)\Delta t},\ \forall w\in X_{\mathcal{N}},K\ge k\ge 1, \tag{2.85}$$

de manière analogue au cas elliptique, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ et pour tout $k\in\mathbb{K}$, nous pouvons écrire

$$e_N^s(\boldsymbol{\mu},t^k)\le\frac{1}{\Delta t}\Big|\Big|\Big|u_{\mathcal{N}}(\boldsymbol{\mu},t^k)-u_N(\boldsymbol{\mu},t^k)\Big|\Big|\Big|_{pr,\boldsymbol{\mu}}\Big|\Big|\Big|\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^{K-k+1})-\Psi_N(\boldsymbol{\mu},t^{K-k+1})\Big|\Big|\Big|_{du,\boldsymbol{\mu}}, \tag{2.86}$$

où l'erreur $e_N^s(\boldsymbol{\mu},t^k)$ est donnée par $e_N^s(\boldsymbol{\mu},t^k)=s_{\mathcal{N}}(\boldsymbol{\mu},t^k)-s_N(\boldsymbol{\mu},t^k)$. Étudions maintenant la construction de l'ensemble $S_{\bar{N}}$ par l'algorithme *POD/glouton*. Pour cela définissons, pour un temps t^k et un jeu de paramètres $\boldsymbol{\mu}^*$ donnés, l'erreur de projection sur l'espace d'approximation primal et dual définie par

$$e_{\Pi_{W_{N_{pr}}}}^{N,k}(\boldsymbol{\mu}^*;t^k)=u_{\mathcal{N}}(\boldsymbol{\mu}^*,t^k)-\Pi_{W_{N_{pr}}}\Big(\ u_{\mathcal{N}}(\boldsymbol{\mu}^*,t^k)\ \Big), \tag{2.87}$$

et

$$e_{\Pi_{W_{N_{du}}}}^{N,k}(\boldsymbol{\mu}^*;t^k)=\Psi_{\mathcal{N}}(\boldsymbol{\mu}^*,t^k)-\Pi_{W_{N_{du}}}\Big(\ \Psi_{\mathcal{N}}(\boldsymbol{\mu}^*,t^k)\ \Big), \tag{2.88}$$

où $\Pi_{W_{N_{pr}}}(.)$ et $\Pi_{W_{N_{du}}}(.)$ désignent la projection, en utilisant produit scalaire $(.,.)_X$, respectivement sur $W_{N_{pr}}$ et $W_{N_{du}}$. Comme dans la section 1, Ξ_{train} est un échantillonnage fin de l'espace des paramètres dans lequel les paramètres seront sélectionnés.

Le premier vecteur de paramètres $\boldsymbol{\mu}^*$ est choisi de manière aléatoire dans Ξ_{train}. En revanche les suivants maximisent l'erreur commise entre l'approximation RB et FE sur la sortie, calculée par $\Delta_{N-1}^s(\boldsymbol{\mu},t^k)$ pour chaque élément de Ξ_{train}. À l'instant t^k, pour $\boldsymbol{\mu}^*$ donné, cette erreur s'écrit $e^s(\boldsymbol{\mu}^*,t^k)=s_{\mathcal{N}}(\boldsymbol{\mu}^*,t^k)-s_{\mathcal{N}}(\boldsymbol{\mu}^*,t^k)$. De manière similaire au cas elliptique, $\Delta_{N-1}^s(\boldsymbol{\mu}^*,t^k)$ fournit une borne supérieure de $e^s(\boldsymbol{\mu}^*,t^K)$ en se servant des $N-1$ premiers éléments de la base réduite. Nous rappelons que la section 3 est dédiée à la mise

Algorithm 3 Algorithme POD/glouton
$[S_{\bar{N}}, W_{N_{pr}}, W_{N_{du}}] = POD/glouton(N_{max}, \epsilon_{tol}, N_m, \Xi_{train})$

$\boldsymbol{\mu}^*$ est choisi aléatoirement dans Ξ_{train} ;
$\epsilon \leftarrow \infty$, $N \leftarrow 0$;
$W_{0_{pr}} \leftarrow \emptyset$, $W_{0_{du}} \leftarrow \emptyset$;
$\bar{N} \leftarrow 1$;
$S_{\bar{N}} \leftarrow \boldsymbol{\mu}^*$;
while $\epsilon > \epsilon_{tol}$ et $N + N_m \leq N_{max}$ **do**
  $\bar{N} \leftarrow \bar{N} + 1$;
  for $i = 1, ..., N_m$ **do**
    $W_{N+i_{pr}} \leftarrow W_{N_{pr}} \oplus POD\left(\left\{e^{N,k}_{\Pi_{W_{N_{pr}}}}(\boldsymbol{\mu}^*; t^k), 1 \leq k \leq K\right\}, i\right)$;
    $W_{N+i_{du}} \leftarrow W_{N_{du}} \oplus POD\left(\left\{e^{N,k}_{\Pi_{W_{N_{du}}}}(\boldsymbol{\mu}^*; t^k), K \geq k \geq 1\right\}, i\right)$;
  end for
  $\boldsymbol{\mu}^* \leftarrow \arg \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta^s_{N-1}\left(\boldsymbol{\mu}, t^K\right)$;
  $\epsilon \leftarrow \Delta^s_{N-1}\left(\boldsymbol{\mu}^*, t^K\right)$;
  $S_{\bar{N}} \leftarrow S_{\bar{N}-1} \cup \boldsymbol{\mu}^*$;
  $N \leftarrow N + N_m$;
end while

en place d'un tel estimateur d'erreur. À chaque itération de l'algorithme POD/glouton, pour chaque nouveau jeu de paramètres dans $S_{\bar{N}}$, les espaces $W_{N_{pr}}$ et $W_{N_{du}}$ sont enrichis par les N_m premiers modes de l'erreur de projection, puis ces nouveaux éléments RB sont orthonormalisés.

Remarque 2. *Une autre manière de générer les éléments de la base réduite, mentionnée dans [103], consiste à appliquer une POD sur l'ensemble des modes issus eux-même des POD appliquées pour chaque $\boldsymbol{\mu} \in S_{\bar{N}}$, pour obtenir les espaces $W_{N_{pr}}$ et $W_{N_{du}}$. Dans ce cas l'algorithme 3 devient l'agorithme 4 (version alternative).*

Regardons la complexité algorithmique de la partie *en-ligne* dans le cas où nous nous servons de k pas de temps. Il faut $\mathcal{O}(Q_a N^2 + Q_m N^2)$ multiplications pour assembler le terme de gauche du système réduit primal (2.65) ou dual (2.78). La construction du second membre implique $\mathcal{O}(kN^2)$ opérations, aussi bien pour le système primal que dual. La résolution du problème, primal ou dual, nécessite $\mathcal{O}(kN^3)$ opérations. Ensuite, dans le cas de sorties "souples", il faut compter $\mathcal{O}(kN)$ produits scalaires pour avoir la sortie du modèle via (2.64), mais si l'on souhaite intégrer le terme de correction pour les sorties non "souples" (2.82) cela en nécessite alors $\mathcal{O}(kN^3)$. L'étape *en-ligne* reste indépendante de la (grande) dimension $\mathcal{N}$.

3 Estimation d'erreur a posteriori

À l'aide des formules (2.30) et (2.74), pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, et – pour les problèmes paraboliques – un temps t^k donné, $1 \leq k \leq K$, nous pouvons construire rapidement une approximation de la sortie du modèle. Dans cette section, nous introduisons des estimateurs d'erreur a posteriori [101, 85, 39, 123, 53] qui vont nous

Algorithm 4 Algorithme POD/glouton – version alternative – $[S_{\bar{N}}, W_{N_{pr}}, W_{N_{du}}] = POD/glouton(N_{max}, \epsilon_{tol}, N_m, \Xi_{train})$

$\boldsymbol{\mu}^*$ est choisi aléatoirement dans Ξ_{train} ;
$\epsilon \leftarrow \infty$, $N \leftarrow 0$, $Z^u \leftarrow \emptyset$, $Z^\Psi \leftarrow \emptyset$;
$W_{0_{pr}} \leftarrow \emptyset$, $W_{0_{du}} \leftarrow \emptyset$;
$\bar{N} \leftarrow 1$;
$S_{\bar{N}} \leftarrow \boldsymbol{\mu}^*$;
while $\epsilon > \epsilon_{tol}$ et $N + N_m \leq N_{max}$ **do**
  $\bar{N} \leftarrow \bar{N} + 1$;
  $\left\{\bar{Z}^u_m, 1 \leq m \leq N_m\right\} \leftarrow POD\Big(Snap^u(\boldsymbol{\mu}^*), N_m\Big)$;
  $\left\{\bar{Z}^\Psi_m, 1 \leq m \leq N_m\right\} \leftarrow POD\Big(Snap^\Psi(\boldsymbol{\mu}^*), N_m\Big)$;
  $Z^u \leftarrow \Big\{Z^u, \{\bar{Z}^u_m, 1 \leq m \leq N_m\}\Big\}$; $Z^\Psi \leftarrow \Big\{Z^\Psi, \{\bar{Z}^\Psi_m, 1 \leq m \leq N_m\}\Big\}$;
  $W_{N_{pr}} \leftarrow POD\Big(Z^u, N\Big)$; $W_{N_{du}} \leftarrow POD\Big(Z^\Psi, N\Big)$;
  $\boldsymbol{\mu}^* \leftarrow \arg \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta^s_{N-1}\Big(\boldsymbol{\mu}, t^K\Big)$;
  $\epsilon \leftarrow \Delta^s_{N-1}\Big(\boldsymbol{\mu}^*, t^K\Big)$;
  $S_{\bar{N}} \leftarrow S_{\bar{N}-1} \cup \boldsymbol{\mu}^*$;
  $N \leftarrow N + N_m$;
end while

permettre de savoir, rapidement et de manière fiable, si l'approximation des solutions primale et duale, ainsi de la sortie du modèle sont suffisament précises.

3.1 Ingrédients

Introduisons les constantes de coercivité $\alpha^{a,\mathcal{N}}(\boldsymbol{\mu})$ et $\alpha^{m,\mathcal{N}}(\boldsymbol{\mu})$ appliquées à la discrétisation FE, telles que pour tout $\boldsymbol{\mu} \in \mathcal{D}$

$$\alpha^{a,\mathcal{N}}(\boldsymbol{\mu}) = \inf_{w \in X_\mathcal{N}} \frac{a(w, w; \boldsymbol{\mu})}{|||w|||^2_{a,\boldsymbol{\mu}_{ref}}} > 0 \text{ et } \alpha^{m,\mathcal{N}}(\boldsymbol{\mu}) = \inf_{w \in X_\mathcal{N}} \frac{m(w, w; \boldsymbol{\mu})}{|||w|||^2_{m,\boldsymbol{\mu}_{ref}}} > 0. \tag{2.89}$$

Soient $\alpha^a_{LB}(\boldsymbol{\mu})$ et $\alpha^m_{LB}(\boldsymbol{\mu})$ les bornes inférieures postitives telles que nous pouvons écrire

$$0 \leq \alpha^a_{LB}(\boldsymbol{\mu}) \leq \alpha^{a,\mathcal{N}}(\boldsymbol{\mu}) \ \forall \boldsymbol{\mu} \in \mathcal{D} \text{ et } 0 \leq \alpha^m_{LB}(\boldsymbol{\mu}) \leq \alpha^{m,\mathcal{N}}(\boldsymbol{\mu}) \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.90}$$

où le temps d'évaluation de $\boldsymbol{\mu} \to \alpha^a_{LB}(\boldsymbol{\mu})$ et de $\boldsymbol{\mu} \to \alpha^m_{LB}(\boldsymbol{\mu})$ doivent être indépendants de la (grande) dimension $\mathcal{N}$ afin d'évaluer rapidement les estimateurs d'erreur que nous allons introduire dans cette section. Une manière de calculer $\alpha^a_{LB}(\boldsymbol{\mu})$ et $\alpha^m_{LB}(\boldsymbol{\mu})$ est exposée dans la section 4.

3.2 Problèmes elliptiques

Commençons par introduire la norme duale des résidus primal $\epsilon_{N_{pr}}(\boldsymbol{\mu})$ et dual $\epsilon_{N_{du}}(\boldsymbol{\mu})$:

$$\epsilon_{N_{pr}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_\mathcal{N}} \frac{r_{pr}(v; \boldsymbol{\mu})}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \big|\big|\big|\hat{e}_{pr}(\boldsymbol{\mu})\big|\big|\big|_{a,\boldsymbol{\mu}_{ref}}, \tag{2.91}$$

$$\epsilon_{N_{du}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{r_{du}(v;\boldsymbol{\mu})}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = |||\hat{e}_{du}(\boldsymbol{\mu})|||_{a,\boldsymbol{\mu}_{ref}}. \tag{2.92}$$

Notons que pour définir la norme duale, la représentation de Riesz des résidus primal et dual, respectivement $\hat{e}_{pr}(\boldsymbol{\mu})$ et $\hat{e}_{du}(\boldsymbol{\mu})$, a été introduite dans (2.91) et (2.92). À présent définissons l'estimation d'erreur a posteriori sur l'approximation de la sortie du modèle. Pour tout N tel que $1 \leq N \leq N_{max}$ et pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\left|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})\right| \leq \Delta_N^s(\boldsymbol{\mu}) \equiv \ \alpha_{LB}^a(\boldsymbol{\mu})\ \Delta_N^{pr}(\boldsymbol{\mu})\ \Delta_N^{du}(\boldsymbol{\mu}), \tag{2.93}$$

où $\Delta_N^{pr}(\boldsymbol{\mu})$ et $\Delta_N^{du}(\boldsymbol{\mu})$ sont les estimations d'erreur a posteriori respectivement pour les solutions, primale et duale, définies par

$$\Delta_N^{pr}(\boldsymbol{\mu}) = \frac{\sqrt{\epsilon_{N_{pr}}(\boldsymbol{\mu})^2}}{\alpha_{LB}^a(\boldsymbol{\mu})} \text{ et } \Delta_N^{du}(\boldsymbol{\mu}) = \frac{\sqrt{\epsilon_{N_{du}}(\boldsymbol{\mu})^2}}{\alpha_{LB}^a(\boldsymbol{\mu})}. \tag{2.94}$$

Les erreurs commisent sur les solutions peuvent être majorées de la façon suivante

$$|||u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})|||_{a,\boldsymbol{\mu}_{ref}} \leq \Delta_N^{pr}(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.95}$$

$$|||\Psi_{\mathcal{N}}(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu})|||_{a,\boldsymbol{\mu}_{ref}} \leq \Delta_N^{du}(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.96}$$

Stratégie *hors-ligne/en-ligne*

Nous exposons ici la stratégie *hors-ligne/en-ligne* à mettre en oeuvre afin d'évaluer les estimations d'erreurs définis dans (2.93) et (2.94) de manière efficace. En nous appuyant sur la dépendance affine en paramètres de a et f décrite dans (2.9) nous pouvons exprimer le résidu primal de la manière suivante :

$$r_{pr}(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(v) - \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) a^q(\xi_i^{pr}, v), \quad \forall v \in X_{\mathcal{N}}. \tag{2.97}$$

Le représentant de Riesz $\hat{e}_{pr}(\boldsymbol{\mu})$ vérifie

$$\Big(\Big(\hat{e}_{pr}(\boldsymbol{\mu}), v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(v) - \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) a^q(\xi_i^{pr}, v), \quad \forall v \in X_{\mathcal{N}}. \tag{2.98}$$

Ensuite, en utilisant le principe de superposition linéaire – ou *linear superposition* – , nous pouvons écrire

$$\hat{e}_{pr}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) \Gamma_{N_{pr}}^q + \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) \Upsilon_{N_{pr}}^{qi}, \tag{2.99}$$

avec

$$\Big(\Big(\Gamma_{N_{pr}}^q, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = f^q(v)\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_f, \tag{2.100}$$

$$\Big(\Big(\Upsilon_{N_{pr}}^{qi}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -a^q(\xi_i^{pr}, v),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_a,\ 1 \leq i \leq N. \tag{2.101}$$

Par conséquent nous avons

$$\epsilon_{N_{pr}}(\boldsymbol{\mu})^2 = C^{ff}_{N_{pr}}(\boldsymbol{\mu}) + 2\sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu}) C^{fa}_{N_{pr}i}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{i'=1}^{N} u_{Ni}(\boldsymbol{\mu}) u_{Ni'}(\boldsymbol{\mu}) C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu}), \quad (2.102)$$

avec pour $1 \leq i, i' \leq N$:

$$C^{ff}_{N_{pr}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\sum_{q'=1}^{Q_f} \theta^q_f(\boldsymbol{\mu})\ \theta^{q'}_f(\boldsymbol{\mu})\ \Phi^{qq'}_{N_{pr},ff}, \quad (2.103)$$

$$C^{fa}_{N_{pr}i}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_f} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_f(\boldsymbol{\mu})\ \Phi^{qiq'}_{N_{pr},fa}, \quad (2.104)$$

$$C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_a(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{pr},aa}. \quad (2.105)$$

où on peut écrire

$$\Phi^{qq'}_{N_{pr},ff} = \left(\left(\Gamma^q_{N_{pr}}, \Gamma^{q'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qiq'}_{N_{pr},fa} = \left(\left(\Upsilon^{qi}_{N_{pr}}, \Gamma^{q'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ , \quad (2.106)$$

$$\Phi^{qiq'i'}_{N_{pr},aa} = \left(\left(\Upsilon^{qi}_{N_{pr}}, \Upsilon^{q'i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}. \quad (2.107)$$

À présent, condidérons le résidu dual qui s'écrit

$$r_{du}(v;\boldsymbol{\mu}) = -\sum_{q=1}^{Q_\ell} \theta^q_\ell(\boldsymbol{\mu}) \ell^q(v) - \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta^q_a(\boldsymbol{\mu}) \Psi_{N_i}(\boldsymbol{\mu}) a^q(v, \xi^{du}_i), \quad \forall v \in X_{\mathcal{N}}. \quad (2.108)$$

Le représentant de Riesz $\hat{e}_{du}(\boldsymbol{\mu})$ vérifie

$$\hat{e}_{du}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \theta^q_\ell(\boldsymbol{\mu}) \Gamma^q_{N_{du}} + \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta^q_a(\boldsymbol{\mu}) \Psi_{N_i}(\boldsymbol{\mu}) \Upsilon^{qi}_{N_{du}},\ \forall v \in X_{\mathcal{N}}, \quad (2.109)$$

avec

$$\left(\left(\Gamma^q_{N_{du}}, v\right)\right)_{a,\boldsymbol{\mu}_{ref}} = -\ell^q(v),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_\ell, \quad (2.110)$$

$$\left(\left(\Upsilon^{qi}_{N_{du}}, v\right)\right)_{a,\boldsymbol{\mu}_{ref}} = -a^q(v, \xi^{du}_i),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_a,\ 1 \leq i \leq N. \quad (2.111)$$

Nous pouvons donc écrire

$$\epsilon_{N_{du}}(\boldsymbol{\mu})^2 = C^{\ell\ell}_{N_{du}}(\boldsymbol{\mu}) + 2\sum_{i=1}^{N} \Psi_{N_i} C^{\ell a}_{N_{du}i}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{i'=1}^{N} \Psi_{N_i}\Psi_{N_{i'}} C^{aa}_{N_{du}ii'}(\boldsymbol{\mu}), \quad (2.112)$$

avec pour $1 \leq i, i' \leq N$:

$$C^{\ell\ell}_{N_{du}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell}\sum_{q'=1}^{Q_\ell} \theta^q_\ell(\boldsymbol{\mu})\ \theta^{q'}_\ell(\boldsymbol{\mu})\ \Phi^{qq'}_{N_{du}}, \quad (2.113)$$

$$C^{\ell a}_{N_{du}i}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_\ell} \theta_a^q(\boldsymbol{\mu})\ \theta_\ell^{q'}(\boldsymbol{\mu})\ \Phi^{qiq'}_{N_{du}}, \tag{2.114}$$

$$C^{aa}_{N_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ \theta_a^{q'}(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{du}}. \tag{2.115}$$

où on a

$$\Phi^{qq'}_{N_{du},\ell\ell} = \left(\left(\Gamma^q_{N_{du}}, \Gamma^{q'}_{N_{du}}\right)\right)_{a,\boldsymbol{\mu}_{ref}},\ \Phi^{qiq'}_{N_{du},\ell a} = \left(\left(\Upsilon^{qi}_{N_{du}}, \Gamma^{q'}_{N_{du}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}, \tag{2.116}$$

$$\Phi^{qiq'i'}_{N_{du},aa} = \left(\left(\Upsilon^{qi}_{N_{du}}, \Upsilon^{q'i'}_{N_{du}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}. \tag{2.117}$$

Durant la phase *hors-ligne*, les quantités qui ne dépendent pas de $\boldsymbol{\mu}$ sont calculées puis stockées dans une base de données. Elles pourront ainsi être réutilisées lors de la phase *en-ligne* au cours de laquelle, pour $\boldsymbol{\mu} \in \mathcal{D}$ donné, les termes qui dépendent de $\boldsymbol{\mu}$ sont évalués et la norme duale des résidus est assemblée en utilisant l'approximation RB des solutions. De cette manière le calcul de l'estimation d'erreur ne dépend pas de la (grande) dimension FE.

Il est à noter que dans la pratique, l'évaluation de certains termes définis précédemment peut être sensible aux erreurs d'arrondis. Dans [30] il est proposé une solution pour remédier à de tels problèmes d'arrondis.

Pendant la phase *hors-ligne* il faut résoudre $(Q_f+2Q_aN+Q_\ell)$ problèmes de type Poisson qui dépendent de la dimension FE, $\mathcal{N}$, voir (2.100), (2.101), (2.110) et (2.111). Bien sûr cela sous entend que les quantités $\Gamma^q_{N_{pr}}$ pour $1 \leq N \leq Q_f$, $\Gamma^q_{N_{du}}$ pour $1 \leq N \leq Q_\ell$, $\Upsilon^{qi}_{N_{pr}}$ pour $1 \leq q \leq Q_a$ et $1 \leq i \leq N$, $\Upsilon^{qi}_{N_{du}}$ pour $1 \leq q \leq Q_a$ et $1 \leq i \leq N$ sont stockées. Dans le cas où aucune de ces quantités ne sont stockées, $\mathcal{O}(2N^2Q_a^2)$ résolutions de problèmes de type Poisson sont nécessaires. Cependant la proposition 1 permet de réduire ce nombre de résolutions.

Proposition 1. *Soient q, q', i et i' quatre entiers tels que $1 \leq q, q' \leq Q_a$ ainsi que $1 \leq i, i' \leq N$, nous avons alors*

$$\Phi^{qiq'i'}_{N_{pr},aa} = \Phi^{q'i'qi}_{N_{pr},aa} \quad et \quad \Phi^{qiq'i'}_{N_{du},aa} = \Phi^{q'i'qi}_{N_{du},aa}. \tag{2.118}$$

Les égalités énoncées dans (2.118) sont justifiées par la propriété de symétrie du produit scalaire. En nous appuyant sur ces égalités nous constatons que $\mathcal{O}(N^2(Q_a+1)Q_a)$ résolutions de problèmes de type Poisson suffisent lors de la phase *hors-ligne*, soit une économie de $\mathcal{O}(N^2(Q_a-1)Q_a)$ résolutions. Notons qu'il est également envisageable de chercher un équilibre entre le stockage d'une partie des solutions des problèmes de type Poisson et les résolutions.

Intéressons-nous maintenant à la phase *en-ligne*. Supposons que nous connaissons déjà les coefficients $u_{N_i}(\boldsymbol{\mu})$ et $\Psi_{N_i}(\boldsymbol{\mu})$, $1 \leq i \leq N$. Supposons également que les fonctions $\theta_f^q(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_f$, $\theta_\ell^q(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_\ell$ et $\theta_a^q(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_a$ sont déjà évaluées. Pour construire les termes introduits par (2.103)-(2.105) et (2.113)-(2.115), $\mathcal{O}(2Q_a^2+Q_f^2+Q_\ell^2)$ opérations sont nécessaires puisque les produits scalaires définis par (2.106)-(2.107) et (2.116)-(2.117) ont été évalués pendant la phase *hors-ligne*. Une fois

ces termes construits, nous pouvons évaluer les sommes définies en (2.102) et (2.112) en $\mathcal{O}(2Q_a^2N^2)$ opérations. Ainsi, la complexité *en-ligne* de l'estimation d'erreur est en $\mathcal{O}(2Q_a^2N^2)$ opérations.

3.3 Problèmes paraboliques

Définissons la norme duale du résidu primal $\epsilon_{N_{pr}}(\boldsymbol{\mu}, t^k)$ et dual $\epsilon_{N_{du}}(\boldsymbol{\mu}, t^k)$:

$$\epsilon_{N_{pr}}(\boldsymbol{\mu}, t^k) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{r_{pr}(v; \boldsymbol{\mu}; t^k)}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \left|||\hat{e}_{pr}(\boldsymbol{\mu}, t^k)\right|||_{a,\boldsymbol{\mu}_{ref}} \tag{2.119}$$

$$\epsilon_{N_{du}}(\boldsymbol{\mu}, t^k) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{r_{du}(v; \boldsymbol{\mu}; t^k)}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \left|||\hat{e}_{du}(\boldsymbol{\mu}, t^k)\right|||_{a,\boldsymbol{\mu}_{ref}}. \tag{2.120}$$

Notons que pour définir la norme duale, la représentation de Riesz des résidus primal et dual, respectivement $\hat{e}_{pr}(\boldsymbol{\mu}, t^k)$ et $\hat{e}_{du}(\boldsymbol{\mu}, t^k)$, a été introduit dans (2.119) et (2.120). À présent définissons l'estimation d'erreur a posteriori sur l'approximation de la sortie du modèle. Pour chaque N et pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\left|s_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - s_N(\boldsymbol{\mu}, t^k)\right| \leq \Delta_N^s(\boldsymbol{\mu}, t^k) \equiv \ \Delta_N^{pr}(\boldsymbol{\mu}, t^k)\ \Delta_N^{du}(\boldsymbol{\mu}, t^{K-k+1}),\ 1 \leq k \leq K, \tag{2.121}$$

où $\Delta_N^{pr}(\boldsymbol{\mu}; t^k)$ et $\Delta_N^{du}(\boldsymbol{\mu}, t^k)$ sont les estimations d'erreur a posteriori respectivement pour les solutions, primale et duale, définies par

$$\Delta_N^{pr}(\boldsymbol{\mu}, t^k) = \sqrt{\frac{\Delta t}{\alpha_{LB}^a(\boldsymbol{\mu})} \sum_{k'=1}^{k} \epsilon_{N_{pr}}(\boldsymbol{\mu}, t^{k'})^2},\ 1 \leq k \leq K, \tag{2.122}$$

$$\Delta_N^{du}(\boldsymbol{\mu}, t^k) = \sqrt{\frac{\Delta t}{\alpha_{LB}^a(\boldsymbol{\mu})} \sum_{k'=k}^{K} \epsilon_{N_{du}}(\boldsymbol{\mu}, t^{k'})^2 + \alpha_{LB}^m(\boldsymbol{\mu}) \Delta_N^{\Psi final}(\boldsymbol{\mu})^2},\ K \geq k \geq 1. \tag{2.123}$$

Le terme $\Delta_N^{\Psi final}(\boldsymbol{\mu})$ provient de l'erreur commise au temps t^{K+1} sur la variable duale. En posant $e^{du}(\boldsymbol{\mu}, t^{K+1}) = \Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^{K+1}) - \Psi_N(\boldsymbol{\mu}, t^{K+1})$ on peut écrire

$$\left|\left|\left|e^{du}(\boldsymbol{\mu}, t^{K+1})\right|\right|\right|_{m,\boldsymbol{\mu}_{ref}} \leq \Delta_N^{\Psi final}(\boldsymbol{\mu}) \equiv \frac{\epsilon_{N_{du}}^{\Psi final}(\boldsymbol{\mu}; t^{K+1})}{\alpha_{LB}^m(\boldsymbol{\mu})}, \tag{2.124}$$

où $\epsilon_{N_{du}}^{\Psi final}(\boldsymbol{\mu}; t^{K+1})$ est la norme duale du résidu associé à la condition finale. Notons que dans (2.122) la contribution du résidu associé à la condition initiale n'apparaît pas car nous supposons ici que $u_{\mathcal{N}}(\boldsymbol{\mu}, t^0) \in W_{N_{pr}}$. Néanmoins, si $u_{\mathcal{N}}(\boldsymbol{\mu}, t^0) \notin W_{N_{pr}}$ alors il est nécessaire d'ajouter un terme supplémentaire dans (2.122). Les erreurs commisent sur les solutions peuvent être majorées de la façon suivante

$$\left|\left|\left|u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_N(\boldsymbol{\mu}, t^k)\right|\right|\right|_{pr,\boldsymbol{\mu}_{ref}} \leq \Delta_N^{pr}(\boldsymbol{\mu}, t^k),\ \forall \boldsymbol{\mu} \in \mathcal{D},\ 1 \leq k \leq K, \tag{2.125}$$

$$\left|\left|\left|\Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - \Psi_N(\boldsymbol{\mu}, t^k)\right|\right|\right|_{du,\boldsymbol{\mu}_{ref}} \leq \Delta_N^{du}(\boldsymbol{\mu}, t^k),\ \forall \boldsymbol{\mu} \in \mathcal{D},\ K \geq k \geq 1. \tag{2.126}$$

Stratégie *hors-ligne/en-ligne*

Tout comme dans le cas elliptique,nous exposons ici la stratégie *hors-ligne/en-ligne* à mettre en oeuvre afin d'évaluer les estimations d'erreurs définis dans (2.121), (2.122) et (2.123) de manière efficace. En nous appuyant sur la dépendance affine en paramètres de m, a et f décrite dans (2.50) nous pouvons exprimer le résidu primal, pour tout $v \in X_{\mathcal{N}}$ et $1 \leq k \leq K$, de la manière suivante :

$$\begin{aligned} r_{pr}\big(v;\boldsymbol{\mu};t^k\big) = \; & \sum_{q=1}^{Q_f} \theta_f^q\big(\boldsymbol{\mu};t^k\big) f^q\big(v\big) - \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta_a^q\big(\boldsymbol{\mu}\big) u_{Ni}\big(\boldsymbol{\mu};t^k\big) a^q\big(\xi_i^{pr},v\big) \\ & -\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{i=1}^{N} \theta_m^q\big(\boldsymbol{\mu}\big)\Big(u_{Ni}(\boldsymbol{\mu};t^k) - u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big) m^q(\xi_i^{pr},v). \end{aligned} \tag{2.127}$$

Le représentant de Riesz $\hat{e}_{pr}(\boldsymbol{\mu},t^k)$ vérifie

$$\Big(\Big(\hat{e}_{pr}(\boldsymbol{\mu},t^k),v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = r_{pr}\big(v;\boldsymbol{\mu};t^k\big),\ \forall v \in X_{\mathcal{N}}. \tag{2.128}$$

Ensuite, en utilisant le principe de superposition linéaire – ou *linear superposition* – , nous pouvons écrire

$$\begin{aligned} \hat{e}_{pr}(\boldsymbol{\mu},t^k) = \; & \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu};t^k)\Gamma_{N_{pr}}^q + \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu})\Upsilon_{N_{pr}}^{qi} \\ & +\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{i=1}^{N} \theta_m^q(\boldsymbol{\mu})\Big(u_{Ni}(\boldsymbol{\mu};t^k) - u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big)\Lambda_{N_{pr}}^{qi}, 1 \leq k \leq K, \end{aligned} \tag{2.129}$$

avec

$$\Big(\Big(\Gamma_{N_{pr}}^q,v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = f^q(v),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_f, \tag{2.130}$$

$$\Big(\Big(\Upsilon_{N_{pr}}^{qi},v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -a^q(\xi_i^{pr},v),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_a,\ 1 \leq i \leq N, \tag{2.131}$$

$$\Big(\Big(\Lambda_{N_{pr}}^{qi},v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -m^q(\xi_i^{pr},v),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_m,\ 1 \leq i \leq N. \tag{2.132}$$

Par conséquent nous avons

$$\begin{aligned} \epsilon_{N_{pr}}(\boldsymbol{\mu},t^k)^2 = \; & 2\sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu};t^k) C_{N_{pr}i}^{fa}(\boldsymbol{\mu};t^k) + \sum_{i=1}^{N}\sum_{i'=1}^{N} u_{Ni}(\boldsymbol{\mu};t^k) u_{Ni'}(\boldsymbol{\mu};t^k) C_{N_{pr}ii'}^{aa}(\boldsymbol{\mu}) \\ & +\frac{1}{\Delta t^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\Big(u_{Ni}(\boldsymbol{\mu};t^k) - u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big)\Big(u_{Ni'}(\boldsymbol{\mu};t^k) - u_{Ni'}(\boldsymbol{\mu};t^{k-1})\Big) C_{N_{pr}ii'}^{mm}(\boldsymbol{\mu}) \\ & +\frac{2}{\Delta t}\sum_{i=1}^{N}\sum_{i'=1}^{N}\Big(u_{Ni}(\boldsymbol{\mu};t^k) - u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big) u_{Ni'}(\boldsymbol{\mu};t^k) C_{N_{pr}ii'}^{am}(\boldsymbol{\mu}) \\ & +\frac{2}{\Delta t}\sum_{i=1}^{N}\Big(u_{Ni}(\boldsymbol{\mu};t^k) - u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big) C_{N_{pr}i}^{fm}(\boldsymbol{\mu};t^k) + C_{N_{pr}}^{ff}(\boldsymbol{\mu};t^k),\ 1 \leq k \leq K, \end{aligned} \tag{2.133}$$

avec pour $1 \leq i, i' \leq N$:

$$C^{ff}_{N_{pr}}(\boldsymbol{\mu}; t^k) = \sum_{q=1}^{Q_f} \sum_{q'=1}^{Q_f} \theta^q_f(\boldsymbol{\mu}; t^k)\ \theta^{q'}_f(\boldsymbol{\mu}; t^k)\ \Phi^{qq'}_{N_{pr},ff}\ , \tag{2.134}$$

$$C^{fa}_{N_{pr}i}(\boldsymbol{\mu}; t^k) = \sum_{q=1}^{Q_a} \sum_{q'=1}^{Q_f} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_f(\boldsymbol{\mu}; t^k)\ \Phi^{qiq'}_{N_{pr},fa}\ , \tag{2.135}$$

$$C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{q'=1}^{Q_a} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_a(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{pr},aa}\ , \tag{2.136}$$

$$C^{am}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{q'=1}^{Q_m} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_m(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{pr},am}\ , \tag{2.137}$$

$$C^{fm}_{N_{pr}i}(\boldsymbol{\mu}; t^k) = \sum_{q'=1}^{Q_m} \sum_{q'=1}^{Q_f} \theta^q_m(\boldsymbol{\mu}) \theta^{q'}_f(\boldsymbol{\mu}; t^k)\ \Phi^{qiq'}_{N_{pr},fm}\ , \tag{2.138}$$

$$C^{mm}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m} \sum_{q'=1}^{Q_m} \theta^q_m(\boldsymbol{\mu})\ \theta^{q'}_m(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{pr},mm}\ , \tag{2.139}$$

où on a

$$\Phi^{qq'}_{N_{pr},ff} = \left(\left(\Gamma^q_{N_{pr}}, \Gamma^{q'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}},\ \Phi^{qiq'i'}_{N_{pr},am} = \left(\left(\Upsilon^{qi}_{N_{pr}}, \Lambda^{q'i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{2.140}$$

$$\Phi^{qiq'i'}_{N_{pr},aa} = \left(\left(\Upsilon^{qi}_{N_{pr}}, \Upsilon^{q'i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qiq'}_{N_{pr},fa} = \left(\left(\Upsilon^{qi}_{N_{pr}}, \Gamma^{q'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{2.141}$$

$$\Phi^{qiq'i'}_{N_{pr},mm} = \left(\left(\Lambda^{qi}_{N_{pr}}, \Lambda^{q'i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qiq'}_{N_{pr},fm} = \left(\left(\Lambda^{qi}_{N_{pr}}, \Gamma^{q'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}\ . \tag{2.142}$$

À présent, considérons le résidu dual qui s'écrit, pour tout $v \in X_{\mathcal{N}}$

$$r_{du}(v; \boldsymbol{\mu}; t^k) = -a\left(v, \Psi_N(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}\right) - \frac{1}{\Delta t} m\left(v, \Psi_N(\boldsymbol{\mu}, t^k) - \Psi_N(\boldsymbol{\mu}, t^{k+1})\right), \tag{2.143}$$

avec $K \geq k \geq 1$. Le représentant de Riesz $\hat{e}_{du}(\boldsymbol{\mu}, t^k)$ vérifie

$$\left(\left(\hat{e}_{du}(\boldsymbol{\mu}, t^k), v\right)\right)_{a,\boldsymbol{\mu}_{ref}} = r_{du}\big(v; \boldsymbol{\mu}; t^k\big),\ \forall v \in X_{\mathcal{N}}. \tag{2.144}$$

Ensuite, pour $K \geq k \geq 1$ et en utilisant le principe de superposition linéaire, nous pouvons écrire

$$\begin{aligned} \hat{e}_{du}(\boldsymbol{\mu}, t^k) = \ & -\sum_{q=1}^{Q_a} \sum_{i=1}^{N} \theta^q_a(\boldsymbol{\mu}) \Psi_{N_i}(\boldsymbol{\mu}; t^k) \Upsilon^{qi}_{N_{du}} \\ & -\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \sum_{i=1}^{N} \theta^q_m(\boldsymbol{\mu}) \Big(\Psi_{N_i}(\boldsymbol{\mu}; t^k) - \Psi_{N_i}(\boldsymbol{\mu}; t^{k+1})\Big) \Lambda^{qi}_{N_{du}}, \end{aligned} \tag{2.145}$$

avec

$$\left(\left(\Upsilon^{qi}_{N_{du}}, v\right)\right)_{a,\boldsymbol{\mu}_{ref}} = -a^q(\xi^{du}_i, v),\ \forall v \in X_{\mathcal{N}},\ 1 \leq q \leq Q_a,\ 1 \leq i \leq N, \tag{2.146}$$

$$\Big(\Big(\Lambda^{qi}_{N_{du}}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -m^q(\xi^{du}_i, v),\ \forall v \in X_{\mathcal{N}},\ 1 \le q \le Q_m,\ 1 \le i \le N. \tag{2.147}$$

Par conséquent nous avons, pour $k = K, ..., 1$:

$$\begin{aligned}
\epsilon_{N_{du}}(\boldsymbol{\mu}, t^k)^2 &= \sum_{i=1}^{N}\sum_{i'=1}^{N} \Psi_{N_i}(\boldsymbol{\mu}; t^k)\Psi_{N_i}(\boldsymbol{\mu}; t^k) C^{aa}_{N_{du}i}(\boldsymbol{\mu}) \\
&+\frac{2}{\Delta t}\sum_{i=1}^{N}\sum_{i'=1}^{N}\Big(\Psi_{N_i}(\boldsymbol{\mu}; t^k) - \Psi_{N_i}(\boldsymbol{\mu}; t^{k+1})\Big)\Psi_{N_{i'}}(\boldsymbol{\mu}; t^k) C^{am}_{N_{du}ii'}(\boldsymbol{\mu}) \\
&+\frac{1}{\Delta t^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\Big(\Psi_{N_i}(\boldsymbol{\mu}; t^k) - \Psi_{N_i}(\boldsymbol{\mu}; t^{k+1})\Big)\Big(\Psi_{N_{i'}}(\boldsymbol{\mu}; t^k) - \Psi_{N_{i'}}(\boldsymbol{\mu}; t^{k+1})\Big) C^{mm}_{N_{du}ii'}(\boldsymbol{\mu}),
\end{aligned} \tag{2.148}$$

avec pour $1 \le i, i' \le N$:

$$C^{aa}_{N_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_a(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{du},aa}, \tag{2.149}$$

$$C^{am}_{N_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_m} \theta^q_a(\boldsymbol{\mu})\ \theta^{q'}_m(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{du},am}, \tag{2.150}$$

$$C^{mm}_{N_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m}\sum_{q'=1}^{Q_m} \theta^q_m(\boldsymbol{\mu})\ \theta^{q'}_m(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{du},mm}, \tag{2.151}$$

où on a

$$\Phi^{qiq'i'}_{N_{du},aa} = \big((\Upsilon^{qi}_{N_{du}}, \Upsilon^{q'i'}_{N_{du}})\big)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qiq'i'}_{N_{du},mm} = \big((\Lambda^{qi}_{N_{du}}, \Lambda^{q'i'}_{N_{du}})\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{2.152}$$

$$\Phi^{qiq'i'}_{N_{du},am} = \big((\Upsilon^{qi}_{N_{du}}, \Lambda^{q'i'}_{N_{du}})\big)_{a,\boldsymbol{\mu}_{ref}}\ . \tag{2.153}$$

Intéressons-nous pour finir au résidu associé à la condition finale

$$r^{\Psi final}_{du}(v; \boldsymbol{\mu}; t^{K+1}) = \ell(v; t^{K+1}) - m(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}, t^{K+1}); \boldsymbol{\mu}),\ \forall v \in X_{\mathcal{N}}. \tag{2.154}$$

Le représentant de Riesz $\hat{e}_{du}(\boldsymbol{\mu}, t^{K+1})$ vérifiant

$$\Big(\Big(\hat{e}_{du}(\boldsymbol{\mu}, t^{K+1}), v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = r^{\Psi final}_{du}\big(v; \boldsymbol{\mu}; t^{K+1}\big),\ \forall v \in X_{\mathcal{N}}, \tag{2.155}$$

nous pouvons écrire

$$\hat{e}_{du}(\boldsymbol{\mu}, t^{K+1}) = -\sum_{q=1}^{Q_\ell} \theta^q_\ell(\boldsymbol{\mu}; t^{K+1})\Gamma^q_{N_{du}} - \sum_{q=1}^{Q_m}\sum_{i=1}^{N} \theta^q_m(\boldsymbol{\mu})\Psi_{N_i}(\boldsymbol{\mu}; t^{K+1})\Lambda^{qi}_{N_{du}}, \tag{2.156}$$

avec

$$\Big(\Big(\Gamma^q_{N_{du}}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = \ell^q(\xi^{du}_i, v)\ \forall v \in X_{\mathcal{N}},\ 1 \le q \le Q_\ell\ , \tag{2.157}$$

$$\Big(\Big(\Lambda^{qi}_{N_{du}}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -m^q(\xi^{du}_i, v)\ \forall v \in X_{\mathcal{N}},\ 1 \le q \le Q_m,\ 1 \le i \le N. \tag{2.158}$$

Par conséquent nous avons, pour $k = K, ..., 1$:

$$\begin{aligned}\epsilon^{\Psi final}_{N_{du}}(\boldsymbol{\mu}; t^{K+1})^2 = \; & C^{\ell\ell}_{N_{du}}\big(\boldsymbol{\mu}; t^{K+1}\big) + 2\sum_{i=1}^{N}\big(\Psi_{N_i}(\boldsymbol{\mu}; t^{K+1})\big)C^{\ell m}_{N_{du}i}\big(\boldsymbol{\mu}; t^{K+1}\big) \\ & + \sum_{i=1}^{N}\sum_{i'=1}^{N}\Psi_{N_i}\big(\boldsymbol{\mu}; t^{K+1}\big)\Psi_{N_{i'}}\big(\boldsymbol{\mu}; t^{K+1}\big)C^{mm}_{N_{du}ii'}\big(\boldsymbol{\mu}\big),\end{aligned} \tag{2.159}$$

avec pour $1 \leq i, i' \leq N$:

$$C^{\ell\ell}_{N_{du}}(\boldsymbol{\mu}; t^{K+1}) = \sum_{q=1}^{Q_\ell}\sum_{q'=1}^{Q_\ell}\theta^q_\ell(\boldsymbol{\mu}; t^{K+1})\ \theta^{q'}_\ell(\boldsymbol{\mu}; t^{K+1})\ \Phi^{qq'}_{N_{du},\ell\ell}, \tag{2.160}$$

$$C^{\ell m}_{N_{du}i}(\boldsymbol{\mu}; t^{K+1}) = \sum_{q=1}^{Q_m}\sum_{q'=1}^{Q_\ell}\ \theta^q_m(\boldsymbol{\mu})\ \theta^{q'}_\ell(\boldsymbol{\mu}; t^{K+1})\ \Phi^{qiq'}_{N_{du},\ell m}, \tag{2.161}$$

$$C^{mm}_{N_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m}\sum_{q'=1}^{Q_m}\theta^q_a m(\boldsymbol{\mu})\ \theta^{q'}_m(\boldsymbol{\mu})\ \Phi^{qiq'i'}_{N_{du},mm}, \tag{2.162}$$

où on a

$$\Phi^{qq'}_{N_{du},\ell\ell} = \big(\big(\Gamma^q_{N_{du}}, \Gamma^{q'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qiq'i'}_{N_{du},mm} = \big(\big(\Lambda^{qi}_{N_{du}}, \Lambda^{q'i'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{2.163}$$

$$\Phi^{qiq'}_{N_{du},\ell m} = \big(\big(\Lambda^{qi}_{N_{du}}, \Gamma^{q'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ . \tag{2.164}$$

Tout comme dans le cas elliptique durant la phase *hors-ligne*, les quantités qui ne dépendent pas du jeu de paramètres $\boldsymbol{\mu}$ sont calculées puis stockées dans une base de données. Elles pourront ainsi être réutilisées lors de la phase *en-ligne* au cours de laquelle, pour $\boldsymbol{\mu} \in \mathcal{D}$ et un temps donnés, les termes qui dépendent de $\boldsymbol{\mu}$ sont évalués et la norme duale des résidus est assemblée en utilisant l'approximation RB des solutions. Pendant la phase *hors-ligne* il faut résoudre $(Q_f+Q_\ell+2Q_aN+3Q_mN)$ problèmes de type Poisson qui dépendent de la dimension FE, $\mathcal{N}$, voir (2.130)-(2.132), (2.146)-(2.147) et (2.157)-(2.158). De manière analogue au cas ellitptique, cela sous entend que les quantités $\Gamma^q_{N_{pr}}$ pour $1 \leq N \leq Q_f$, $\Gamma^q_{N_{du}}$ pour $1 \leq N \leq Q_\ell$, $\Upsilon^{qi}_{N_{pr}}$ pour $1 \leq q \leq Q_a$ et $1 \leq i \leq N$, $\Upsilon^{qi}_{N_{du}}$ pour $1 \leq q \leq Q_a$ et $1 \leq i \leq N$, $\Lambda^{qi}_{N_{pr}}$ pour $1 \leq q \leq Q_a$ et $1 \leq i \leq N$, $\Lambda^{qi}_{N_{du}}$ pour $1 \leq q \leq Q_a$ et $1 \leq i \leq N$ sont stockées. Dans le cas où aucune de ces quantités ne sont stockées, $\mathcal{O}(2N^2Q_a^2 + 3N^2Q_m^2)$ résolutions de problèmes de type Poisson sont nécessaires.

Proposition 2. *Soient q, q', i et i' quatre entiers tels que $1 \leq q, q' \leq Q_m$ ainsi que $1 \leq i, i' \leq N$, nous avons alors*

$$\Phi^{qiq'i'}_{N_{pr},mm} = \Phi^{q'i'qi}_{N_{pr},mm} \quad et \quad \Phi^{qiq'i'}_{N_{du},mm} = \Phi^{q'i'qi}_{N_{du},mm}. \tag{2.165}$$

Les égalités énoncées dans (2.165) sont justifiées par la propriété de symétrie du produit scalaire. En nous appuyant sur les propositions 1 et 2 nous constatons que $\mathcal{O}(N^2(Q_a + 1)Q_a + 3/2N^2(Q_m + 1)Q_m)$ résolutions de problèmes de type Poisson suffisent lors de la

phase *hors-ligne*, soit une économie de $\mathcal{O}(\ N^2(Q_a-1)Q_a+3N^2/2(Q_m-1)Qm\ )$ résolutions. Notons qu'il est également envisageable de chercher un équilibre entre le stockage d'une partie des solutions des problèmes de type Poisson et les résolutions.

Intéressons-nous maintenant à la phase *en-ligne*. Supposons que nous connaissons déjà les coefficients $u_{N_i}(\boldsymbol{\mu};t^k)$ et $\Psi_{N_i}(\boldsymbol{\mu};t^k)$, $1 \leq i \leq N$, $1 \leq k \leq K$. Supposons également que les fonctions $\theta_f^q(\boldsymbol{\mu};t^k)$ pour $1 \leq q \leq Q_f$, $\theta_\ell^q(\boldsymbol{\mu};t^k)$ pour $1 \leq q \leq Q_\ell$, $\theta_a^q(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_a$ et $\theta_m^q(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_m$ sont déjà évaluées avec $1 \leq k \leq K$. Pour construire les termes introduits par (2.134)-(2.139), (2.150)-(2.149) et (2.160)-(2.162), $\mathcal{O}(2Q_a^2+3Q_m^2)$ opérations sont nécessaires puisque les produits scalaires définis par (2.140)-(2.142), (2.152)-(2.153), et (2.163)-(2.164) ont été évalués pendant la phase *hors-ligne*. Notons que seuls les termes (2.134), (2.135) et (2.138) doivent être reconstruits à chaque pas de temps. Les termes (2.136), (2.137), (2.139), (2.149), (2.150) et (2.151) n'évoluent pas au cours du temps. Une fois ces termes construits, nous pouvons évaluer les sommes définies en (2.133), (2.148) et (2.159), en $\mathcal{O}(2Q_a^2N^2 + 3Q_m^2N^2)$ opérations. Ainsi, la complexité *en-ligne* de l'estimation d'erreur est en $\mathcal{O}(2Q_a^2N^2 + 3Q_m^2N^2)$ opérations.

4 Méthode des contraintes successives

Dans cette section nous présentons comment calculer une borne inférieure $\alpha_{LB}^a(\boldsymbol{\mu})$ pour la constante de coercivité $\alpha^{a,\mathcal{N}}(\boldsymbol{\mu})$ via la méthode des contraintes successives – ou *Successive Constraints Method (SCM)* – [71, 35, 119, 107]. Le calcul de $\alpha_{LB}^m(\boldsymbol{\mu})$ se fait de manière similaire. Cette méthode se base sur une stratégie efficace *hors-ligne/en-ligne* qui aboutit à un problème d'optimisation linéaire . À l'instar de la méthode RBM qui, durant la partie *en-ligne* ne dépend pas de la (grande) dimension $\mathcal{N}$, le problème d'optimisation linéaire à résoudre durant la partie *en-ligne* ne dépend pas non plus de la dimension $\mathcal{N}$.

Rappelons que la forme bilinéaire paramétrée $a(.,.;\boldsymbol{\mu})$ dépend affinement du jeu de paramètres $\boldsymbol{\mu}$ et peut s'exprimer

$$a\left(u,v;\boldsymbol{\mu}\right) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ a^q(u,v) \quad \forall u,v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{2.166}$$

Nous nous plaçons dans le cas où a est symétrique et coercive.

Remarque 3. *Dans le cas où a n'est ni symétrique ni coervice, on s'intéresse alors à la borne inférieure de la condition inf-sup [71, 112].*

Afin de mettre en oeuvre une stratégie *hors-ligne/en-ligne* efficace, nous allons reformuler l'expression de la constante de coercivité $\alpha^{a,\mathcal{N}}(\boldsymbol{\mu})$ donnée dans (2.89) en introduisant la fonction objective $\mathcal{J}^{obj} : \mathcal{D} \times \mathbb{R}^{Q_a} \to \mathbb{R}$ définie par

$$\mathcal{J}^{obj}(\boldsymbol{\mu};y) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ y^q, \tag{2.167}$$

avec $y \in \mathbb{R}^{Q_a}$. La constante de coercivité (qui dépend de $\mathcal{N}$) est alors définie par

$$\alpha^{a,\mathcal{N}}(\boldsymbol{\mu}) = \inf_{y \in \mathcal{Y}} \mathcal{J}^{obj}(\boldsymbol{\mu};y), \tag{2.168}$$

où l'ensemble $\mathcal{Y} \in \mathbb{R}^{Q_a}$ est défini par

$$\mathcal{Y} = \left\{ y \in \mathbb{R}^{Q_a} \mid \exists w_y \in X_{\mathcal{N}} \text{ tel que } y^q = \frac{a^q(w_y, w_y)}{|||w_y|||^2_{a,\boldsymbol{\mu}_{ref}}},\ 1 \leq q \leq Q_a \right\}. \tag{2.169}$$

La construction de la constante de coercivité via (2.168) implique la résolution d'un problème de minimisation d'une fonction linéaire sur $\mathcal{Y} \subset \mathbb{R}^{Q_a}$. La méthode des contraintes successives repose sur la construction des deux ensembles $\mathcal{Y}_{LB}$ et $\mathcal{Y}_{UB}$ tels que $\mathcal{Y}_{UB} \subset \mathcal{Y} \subset \mathcal{Y}_{LB}$. La fonction objective $\mathcal{J}^{obj}$ est alors minimisée sur ces deux ensembles pour obtenir une borne inférieure et supérieure de la constante de coercivité. La construction de $\mathcal{Y}_{LB}$ et $\mathcal{Y}_{UB}$ sont détaillées dans les paragraphes 4.2 et 4.3. La borne inférieure α^a_{LB} est définie par

$$\alpha^a_{LB}(\boldsymbol{\mu}; C_J) = \min_{y \in \mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)} \mathcal{J}^{obj}(\boldsymbol{\mu}; y), \tag{2.170}$$

la borne supérieure quant à elle est définie par

$$\alpha^a_{UB}(\boldsymbol{\mu}; C_J) = \min_{y \in \mathcal{Y}_{UB}(C_J)} \mathcal{J}^{obj}(\boldsymbol{\mu}; y), \tag{2.171}$$

où l'ensemble C_J est défini par

$$C_J = \left\{ \boldsymbol{\mu}^1 \in \mathcal{D}, ..., \boldsymbol{\mu}^J \in \mathcal{D} \right\}. \tag{2.172}$$

4.1 Construction de l'échantillon des contraintes C_J

Nous allons voir maintenant comment construire C_J, en utilisant un algorithme glouton dans l'étape *hors-ligne* (voir l'algorithme 5). Rappelons que Ξ_{train} est un échantillonnage fin de l'espace des paramètres.Nous avons besoin également d'une tolérance $\epsilon \in (0, 1)$ pour contrôler l'erreur sur la prédiction de la borne inférieure. On commence, lorsque $J = 1$, en

Algorithm 5 Construction de l'ensemble C_J via un algorithme glouton.

$J \leftarrow 1$;
$\boldsymbol{\mu}^1$ est choisi aléatoirement dans Ξ_{train} ;
$C_1 \leftarrow \boldsymbol{\mu}^1$;
while $\max_{\boldsymbol{\mu} \in \Xi_{train}} \left[\frac{\alpha^a_{UB}(\boldsymbol{\mu}, C_J) - \alpha^a_{LB}(\boldsymbol{\mu}, C_J)}{\alpha^a_{UB}(\boldsymbol{\mu}, C_J)} \right] > \epsilon$ **do**
  $\boldsymbol{\mu}^{J+1} = arg \max_{\boldsymbol{\mu} \in \Xi_{train}} \left[\frac{\alpha^a_{UB}(\boldsymbol{\mu}, C_J) - \alpha^a_{LB}(\boldsymbol{\mu}, C_J)}{\alpha^a_{UB}(\boldsymbol{\mu}, C_J)} \right]$;
  $C_{J+1} \leftarrow C_J \cup \boldsymbol{\mu}^{J+1}$;
  $J \leftarrow J + 1$;
end while

choisissant le premier vecteur de paramètres aléatoirement dans Ξ_{train}. Ensuite à partir de $J = 2$, $\boldsymbol{\mu}^J$ maximise l'écart entre les prédictions de la borne inférieure et de la borne supérieure de la constante de coercivité. Autrement dit, à chaque itération l'ensemble C_J est enrichie du vecteur de paramètres $\boldsymbol{\mu} \in \Xi_{train}$ qui possède la pire approximation de la borne inférieure $\alpha^a_{LB}(\boldsymbol{\mu}, C_J)$. Comme pour chaque $\boldsymbol{\mu} \in C_J$ nous avons $\alpha^a_{LB}(\boldsymbol{\mu}, C_J) = \alpha^a_{UB}(\boldsymbol{\mu}, C_J)$, et grâce à la continuité, il s'ensuit que pour un nombre d'itérations suffisamment grand la tolérance ϵ est atteinte.

4.2 Construction de l'ensemble $\mathcal{Y}_{LB}$

Définissons la boîte des contraintes de continuité $\mathcal{B}$ de la manière de telle manière à avoir $\mathcal{Y} \subset \mathcal{B}$.

$$\mathcal{B} = \prod_{q=1}^{Q_a} \left[\inf_{w \in X_{\mathcal{N}}} \frac{a^q(w,w)}{|||w|||^2_{a,\boldsymbol{\mu}_{ref}}}, \sup_{w \in X_{\mathcal{N}}} \frac{a^q(w,w)}{|||w|||^2_{a,\boldsymbol{\mu}_{ref}}} \right]. \tag{2.173}$$

Pour construire l'ensemble $\mathcal{Y}_{LB}$ on se donne également deux entiers strictement positifs : M_α et M_+. Nous introduisons également une nouvelle notation : $P_M^{\boldsymbol{\mu},E}$ l'ensemble de M points dans E le plus proche au sens de la norme euclidienne d'un jeu de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, avec M un entier strictement positif.

Version originale

Dans [71] l'ensemble $\mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J)$ est défini de la manière suivante :

$$\begin{aligned} \mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J) \equiv \Bigg\{ & y \in \mathbb{R}^{Q_a} \mid y \in \mathcal{B}, \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}') y^q \geq \alpha^{a,\mathcal{N}}(\boldsymbol{\mu}') \ , \ \forall \boldsymbol{\mu}' \in P_{M_\alpha}^{\boldsymbol{\mu},C_J} \\ & \text{et} \ \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}') y^q \geq 0 \ , \ \forall \boldsymbol{\mu}' \in P_{M_+}^{\boldsymbol{\mu},\ \Xi_{train}} \Bigg\}. \end{aligned} \tag{2.174}$$

Le problème (2.170) est un problème d'optimisation linéaire – ou *Linear Programm (LP)* –. Il contient Q_a variables de design, ainsi que $2Q_a + M_\alpha + M_+$ inégalités liées aux contraintes. La construction de l'ensemble C_J doit se faire durant la partie *hors-ligne*. Il est important de noter que pour une boîte de contraintes de continuité $\mathcal{B}$ donnée, la complexité de l'évaluation $\boldsymbol{\mu} \to \alpha_{LB}^a(\boldsymbol{\mu})$ est indépendante de $\mathcal{N}$.

Première amélioration

Dans [35], au lieu de contraindre la borne inférieure recherchée à n'être seulement positive pour tout vecteur de paramètres $\boldsymbol{\mu}' \in P_{M_+}^{\boldsymbol{\mu},\ \Xi_{train}}$, l'évaluation de $\alpha_{LB}^a(\boldsymbol{\mu}', C_{J-1})$ est utilisée. l'ensemble $\mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J)$ est défini de la manière suivante :

$$\begin{aligned} \mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J) \equiv \Bigg\{ & y \in \mathbb{R}^{Q_a} \mid y \in \mathcal{B}, \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}') y^q \geq \alpha^{a,\mathcal{N}}(\boldsymbol{\mu}'), \ \forall \boldsymbol{\mu}' \in P_{M_\alpha}^{\boldsymbol{\mu},C_J} \\ & \text{et} \ \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}') y^q \geq \alpha_{LB}^a(\boldsymbol{\mu}', C_{J-1}), \ \forall \boldsymbol{\mu}' \in P_{M_+}^{\boldsymbol{\mu},\ \Xi_{train} \setminus C_J} \Bigg\}. \end{aligned} \tag{2.175}$$

Pour la première itération, on définit $\alpha_{LB}^a(\boldsymbol{\mu}', C_0) \equiv 0, \ \forall \boldsymbol{\mu} \in \Xi_{train}$. Grâce à cette redéfinition de l'ensemble $\mathcal{Y}_{LB}$, lorsque J augmente – c'est-à-dire que nous augmentons la taille de l'ensemble des contraintes – la borne inférieure $\alpha_{LB}^a(\boldsymbol{\mu}, C_J)$ ne décroît pas, il en est de même pour la borne supérieure $\alpha_{UB}^a(\boldsymbol{\mu}, C_J)$, et enfin la quantité $\frac{\alpha_{UB}^a(\boldsymbol{\mu},C_J) - \alpha_{LB}^a(\boldsymbol{\mu},C_J)}{\alpha_{UB}^a(\boldsymbol{\mu},C_J)}$ ne croît pas, quelque soit $\boldsymbol{\mu} \in \Xi_{train}$.

De plus, les contraintes imposées sur $\mathcal{Y}_{LB}$ étant plus strictes, le problème de minimisation (2.170) devrait être résolu plus rapidement qu'avec la version originale.

Deuxième amélioration

Construction *hors-ligne* de l'échantillon C_J :

La construction de $\mathcal{Y}_{LB}$ que nous présentons ici est à appliquer uniquement lors de la construction de l'échantillons C_J durant la partie *hors-ligne*. Nous venons de voir une construction de l'ensemble $\mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)$ qui dépend de deux entiers M_α et M_+. Dans [119] il est proposé de ne plus dépendre ces deux nombres dans la construction de $\mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)$. Remarquons que dans (2.175) il y a deux sortes de contraintes :

— pour M_α contraintes, la constante de coercivité $\alpha^{a,\mathcal{N}}(\boldsymbol{\mu}')$ a été calculée, avec $\boldsymbol{\mu}' \in P_{M_\alpha}^{\boldsymbol{\mu},C_J}$, que nous appelerons contraintes de premier ordre ;
— les M_+ contraintes restantes se servent uniquement de la borne inférieure $\alpha_{LB}^a(\boldsymbol{\mu}', C_{J-1})$, avec $\boldsymbol{\mu}' \in P_{M_+}^{\boldsymbol{\mu},\ \Xi_{train}\setminus C_J}$, que nous appelerons contraintes de second ordre.

L'idée développée par les auteurs repose sur le fait que lors de la construction de l'espace des contraintes C_J, l'approximation des bornes inférieures sont assez pauvres, surtout lorsque J est petit. Les contraintes de second ordre ne sont pas aussi performantes que les contraintes de premier ordre, nous allons donc les abandonner. D'autre part, lorsque le minimum est atteint via un programme d'optimisation linéaire à Q_a variables, il y a au plus Q_a contraintes actives. Aussi, si une contrainte de premier ordre n'est pas active au rang J, elle ne le sera pas pour les rangs supérieurs. Par conséquent il est intéressant de garder une trace des contraintes actives du premier ordre durant la construction de l'espace des contraintes C_J. De cette façon nous n'aurons jamais plus de Q_a contraintes actives du premier ordre et donc l'entier M_α n'est plus utile. Comme rien de prédit que pour $\boldsymbol{\mu} \in \mathcal{D}$ fixé la contrainte active se trouve dans le voisinnage $\boldsymbol{\mu}' \in P_{M_\alpha}^{\boldsymbol{\mu},C_J}$, la construction de $\mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)$ est donc moins restreinte que dans la version originale. De plus, construire un voisinage d'un jeu de paramètres $\boldsymbol{\mu}$ dans Ξ_{train} peut être coûteux si l'échantillon Ξ_{train} comporte beaucoup d'éléments, alors qu'avec cette nouvelle approche le voisinage n'est plus construit. L'ensemble $\mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)$ est alors défini par

$$\mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J) \equiv \left\{ y \in \mathbb{R}^{Q_a} \mid y \in \mathcal{B}, \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}') y^q \geq \alpha^{a,\mathcal{N}}(\boldsymbol{\mu}'),\ \forall \boldsymbol{\mu}' \in \mathcal{A}(\boldsymbol{\mu}, C_{J-1}) \right. \\ \left. \text{et } \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}^J) y^q \geq \alpha^{a,\mathcal{N}}(\boldsymbol{\mu}^J) \right\}, \tag{2.176}$$

où $\mathcal{A}(\boldsymbol{\mu}, C_{J-1})$ regroupe les contraintes actives pour le calcul de la borne inférieure $\alpha_{LB}^a(\boldsymbol{\mu}, C_{J-1})$.

Calcul *en-ligne* de α_{LB}^a :

Pour le calcul de la borne inférieure de la constante de coercivité $\alpha_{LB}^a(\boldsymbol{\mu}; C_J)$ lors de la partie *en-ligne*, les auteurs de [119] proposent d'utiliser toutes les contraintes se trouvant dans l'ensemble C_J. En effet, bien que les meilleures contraintes sont celles de premier ordre – et que seulement quelques unes sont actives –, dans l'étape *en-ligne* on doit pouvoir calculer $\alpha_{LB}^a(\boldsymbol{\mu}; C_J)$ pour n'importe quel jeu de paramètres $\boldsymbol{\mu}$, même si $\boldsymbol{\mu}$ ne se trouve pas dans l'ensemble Ξ_{train}. Or dans ce dernier cas – $\boldsymbol{\mu} \notin \Xi_{train}$ – , l'ensemble des contraintes actives n'a pas été construit durant la partie *hors-ligne*. Cela aboutit à une minimisation linéaire avec peu de variables et beaucoup de contraintes. Les auteurs remarquent que ce problème peut être résolu très rapidement en introduisant le problème

dual.

Nous venons de voir différentes manières de construire l'ensemble $\mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)$. Rappelons que la borne inférieure $\alpha^a_{LB}(\boldsymbol{\mu}; C_J)$ est définie par

$$\alpha^a_{LB}(\boldsymbol{\mu}; C_J) = \min_{y \in \mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)} \mathcal{J}^{obj}(\boldsymbol{\mu}; y). \tag{2.177}$$

Dans la littérature (par exemple [107]) on peut trouver les preuves que $\mathcal{Y} \subset \mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J)$ et que, quelque soit le vecteur de paramètres $\boldsymbol{\mu} \in D$, on a bien $\alpha^a_{LB}(\boldsymbol{\mu}; C_J) \leq \alpha^{a,\mathcal{N}}(\boldsymbol{\mu})$.

4.3 Construction de l'ensemble $\mathcal{Y}_{UB}$

Afin de mettre en oeuvre l'algorithme 5 il est nécessaire, pour un vecteur de paramètres donné $\boldsymbol{\mu} \in \mathcal{D}$ ainsi qu'un échantillon de contraintes C_J, d'avoir une borne supérieure $\alpha^a_{UB}(\boldsymbol{\mu})$ pour la constante de coercivité. Cette borne supérieure est donnée par (2.171). Nous définissons l'ensemble $\mathcal{Y}_{UB}(C_J)$ de la manière suivante :

$$\mathcal{Y}_{UB}(C_J) = \{y^*(\boldsymbol{\mu'}) \mid \boldsymbol{\mu'} \in C_J\}, \tag{2.178}$$

avec

$$y^*(\boldsymbol{\mu'}) = arg \min_{y \in \mathcal{Y}} \mathcal{J}^{obj}(\boldsymbol{\mu'}; y). \tag{2.179}$$

La construction de l'ensemble $\mathcal{Y}_{UB}(C_J)$ peut se faire de manière incrémentale durant la phase *hors-ligne*. En effet, à chaque fois que l'échantillon des contraintes C_J est enrichi avec un nouveau jeu de paramètres $\boldsymbol{\mu'}$, on peut résoudre un problème aux valeurs propres généralisé pour obtenir $y^*(\boldsymbol{\mu'})$ via (2.179). Rappelons que la borne supérieure $\alpha^a_{UB}(\boldsymbol{\mu})$ est définie par

$$\alpha^a_{UB}(\boldsymbol{\mu}; C_J) = \min_{y \in \mathcal{Y}_{UB}(C_J)} \mathcal{J}^{obj}(\boldsymbol{\mu}; y). \tag{2.180}$$

Une fois l'ensemble $\mathcal{Y}_{UB}(C_J)$ construit, pour évaluer la borne supérieure $\alpha^a_{UB}(\boldsymbol{\mu}; C_J)$ durant la phase *en-ligne*, il suffit de trouver l'élément de $\mathcal{Y}_{UB}(C_J)$ qui permet de minimiser $\mathcal{J}^{obj}(\boldsymbol{\mu}; y)$. L'évaluation $\boldsymbol{\mu} \to \alpha^a_{UB}(\boldsymbol{\mu}; C_J)$ est donc indépendante de la dimension $\mathcal{N}$. Du fait que nous ayons $\mathcal{Y}_{UB}(C_J) \subset \mathcal{Y}$, $\alpha^a_{UB}(\boldsymbol{\mu}; C_J)$ est bien une borne supérieure pour la constante de coercivité quelque soit $\boldsymbol{\mu} \in \mathcal{D}$.

Remarque 4. *Nous venons de présenter la méthode des contraintes successives, mais ce n'est pas la seule méthode qui permet de déterminer une borne inférieure pour les constantes de coercivité, il est notamment possible de procéder via l'approche "min-θ" [78, 124, 53, 39]. Avec cette méthode, dans le cas d'opérateurs symétriques coercifs, la borne inférieure de la constante de coercivité est calculée de la manière suivante :*

$$\alpha^a_{LB}(\boldsymbol{\mu}) = \left(\min_{q \in \{1,\dots,Q_a\}} \frac{\theta^q_a(\boldsymbol{\mu})}{\theta^q_a(\boldsymbol{\mu}_{ref})} \right) \alpha^{a,\mathcal{N}}(\boldsymbol{\mu}_{ref}), \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.181}$$

sous l'hypothèse d'avoir $\theta^q_a(\boldsymbol{\mu}) > 0$ $\forall \boldsymbol{\mu} \in \mathcal{D}$, $1 \leq q \leq Q_a$. La constante de coercivité associée à $\boldsymbol{\mu}_{ref}$ est calculée une seule fois durant la phase hors-ligne, puis ensuite le coefficient minimum est évalué pendant la partie en-ligne.

5 Résultats numériques

On se propose ici de vérifier que le framework base réduite, décrit dans la deuxième partie de ce manuscrit, permet de retrouver numériquement les résultats de convergence a priori.

Pour les problèmes linéaires, les premiers travaux sur la convergence a priori de la méthode RBM ne considéraient qu'un espace de paramètre à une dimension, autrement dit $\mathcal{D} \subset \mathbb{R}^1$. Mais au fil des années les résultats on été étendus à un espace des paramètres de dimension $P > 1$, c'est à dire $\mathcal{D} \subset \mathbb{R}^P$ [80, 81, 101, 107, 53, 19, 58, 27]. Nous savons à présent que la méthode RBM converge de manière exponentielle en $c_1 e^{-c_2 N}$, c_1 et c_2 étant des constantes. Ce résultat théorique est obtenu en utilisant un algorithme glouton très proche de l'algorithme 1. Cependant, la sélection des jeux de paramètres diffère légèrement. Dans la théorie, le $N^{\text{ème}}$ vecteur de paramètres est sélectionné de la manière suivante :

$$\boldsymbol{\mu}_N \leftarrow \arg\max_{\boldsymbol{\mu}\in\mathcal{D}} \left| s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu}) \right|, \ N \geq 2. \tag{2.182}$$

Cette méthode de sélection n'est pas applicable dans la pratique, c'est pourquoi dans l'algorithme 1 l'échantillonnage Ξ_{train} de l'espace des paramètres et l'estimation d'erreur a posteriori $\Delta^s_{N-1}(\boldsymbol{\mu})$ défini en (2.93) sont utilisés. Le $N^{\text{ème}}$ vecteur de paramètres est sélectionné comme suit :

$$\boldsymbol{\mu}_N \leftarrow \arg\max_{\boldsymbol{\mu}\in\Xi_{train}} \Delta^s_{N-1}(\boldsymbol{\mu}), \ N \geq 2. \tag{2.183}$$

La même remarque s'applique aux problèmes paraboliques.

Dans cette section, pour chaque problème, en premier lieu nous nous intéresserons à la convergence des approximations RB. Dans un deuxième temps nous nous concentrerons sur les estimateurs d'erreur a posteriori.

5.1 Problème des blocs thermiques

Présentons ici le problème des blocs thermiques qui est une variation du problème elliptique introduit dans [107, 90, 113]. Prenons le carré unité comme domaine d'étude. Nous avons $\Omega \equiv [0,1] \times [0,1]$. On se donne neuf sous-domaines Ω_i, $1 \leq i \leq 9$, de telle manière à avoir $\bigcup_{i=1}^{9} \Omega_i = \Omega$ comme représenté par la figure 2.2. Nous imposons une température nulle sur le bord $\partial\Omega_{haut}$ situé en $y = 1$. Le bord $\partial\Omega_{bas}$ situé en $y = 0$ est exposé à un flux de chaleur égal à 1. Enfin les bords $\partial\Omega_{iso}$ situés en $x = 0$ et $x = 1$ sont isolés. Nous résolvons l'équation de la chaleur dans chaque bloc Ω_i, $1 \leq i \leq 9$. Nous considérons que les coefficients de conductivité thermique peuvent varier d'un bloc à l'autre. Ces coefficients sont nos paramètres. Dans la pratique nous fixons à 1 le coefficient de conductivité thermique μ^1 associé au sous-domain Ω_1. Les coefficients de conductivité thermique des autres sous-domaines, normalisés, sont notés μ^i, $2 \leq i \leq 9$. Nous avons donc notre vecteur de paramètres défini par $\boldsymbol{\mu} = (\mu^2, ..., \mu^9) \in \mathcal{D} \subset \mathbb{R}^P$ avec $P = 8$. Introduisons le réel μ^r comprit entre 1 et l'infini. Nous allons nous servir de μ^r pour déterminer les valeurs minimales et maximales de nos paramètres. Nous avons $\mathcal{D} = [\mu_{min}, \mu_{max}]^P$ où $\mu_{min} = \frac{1}{\sqrt{\mu^r}}$ et $\mu_{max} = \sqrt{\mu^r}$. Ainsi μ^r contrôle le ratio entre μ_{min} et μ_{max} et nous avons $\frac{\mu_{max}}{\mu_{min}} = \mu^r$. Dans notre cas posons $\mu^r = 100$, l'espace des paramètres est donc défini par $\mathcal{D} = [0.1, 10]^P$.

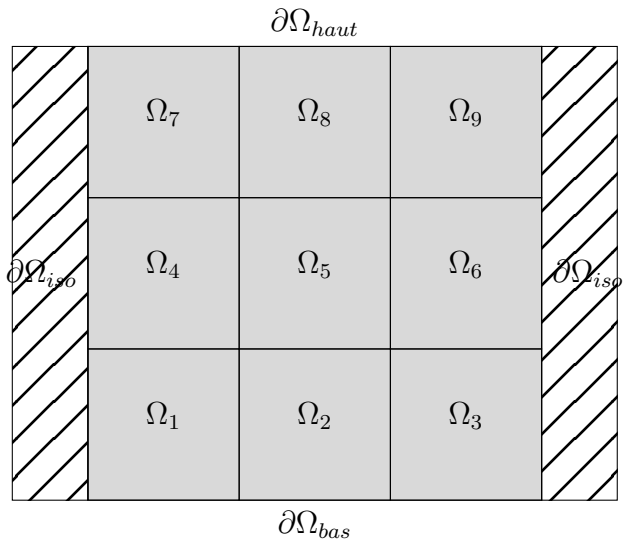


FIGURE 2.2 – Domaine d'étude Ω, composé de 9 sous-domaines, pour le problème des blocs thermiques.

Notons que u est continue à travers les interfaces des sous-domaines. On se donne à présent un espace de fonctions $X_{\mathcal{N}}$ qui approche $X \equiv \{v \in H^1(\Omega), v_{\partial\Omega_{haut}} = 0\}$ tel que $X_{\mathcal{N}} \subset X$. Nous avons fait le choix de traiter la condition de Dirichlet sur le bord $\partial\Omega_{haut}$ de manière faible y. Notons γ_D le paramètre de pénalisation et h_s la taille caractéristique du maillage. La formulation variationnelle de notre problème consiste alors à chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que pour tout $\boldsymbol{\mu} \in \mathcal{D}$

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}\Big), \ \forall v \in X_{\mathcal{N}}. \tag{2.184}$$

La forme bilinéaire a et la forme linéaire f s'écrivent

$$\begin{aligned} a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) \ = \ & \int_{\Omega_1} \nabla u_{\mathcal{N}}(\boldsymbol{\mu}) \cdot \nabla v + \sum_{i=2}^{9} \mu^i \int_{\Omega_i} \nabla u_{\mathcal{N}}(\boldsymbol{\mu}) \cdot \nabla v + \sum_{i=7}^{9} \int_{\Omega_i^{haut}} \frac{\gamma_D}{h_s} u_{\mathcal{N}}(\boldsymbol{\mu}) \ v \\ & - \sum_{i=7}^{9} \mu^i \int_{\Omega_i^{haut}} \Big(\nabla u_{\mathcal{N}}(\boldsymbol{\mu}) \cdot \mathbf{n} \ v + \nabla v \cdot \mathbf{n} \ u_{\mathcal{N}}(\boldsymbol{\mu})\Big), \end{aligned} \tag{2.185}$$

où Ω_i^{haut} est le bords haut situé en $y = 1$ du sous-domaine Ω_i, et

$$f\Big(v; \boldsymbol{\mu}\Big) = \sum_{i=1}^{3} \int_{\Omega_i^{bas}} v, \tag{2.186}$$

où Ω_i^{bas} est le bord bas situé en $x = 0$ du sous-domaine Ω_i. La sortie de ce problème est dite "souple". En effet la forme bilinéaire a est symétrique et la quantité d'intérêt est la valeur de la solution sur le bord $\partial\Omega_{bas}$, autrement dit nous avons $\ell(v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu})$. Rappelons enfin que pour tout $\boldsymbol{\mu} \in \mathcal{D}$ la sortie est donnée par $s_{\mathcal{N}}(\boldsymbol{\mu}) = \ell(u_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu})$.

Prenons une discrétisation FE $\mathbb{P}_1$ avec 47 600 degrés de liberté distribués sur 16 processeurs et illustrons l'impact des paramètres μ^i, $2 \leq i \leq 9$. La figure 2.3 illustre le champ de température, solution de (2.184) lorsque les coefficients de conductivité thermique associés aux sous-domaine Ω_i, $2 \leq i \leq 9$, sont fixés au minimum : 0.1, puis au maximum :

10. La température obtenue sur la figure 2.4 est également solution de (2.184) mais cette fois les paramètres μ^i, pour $i = 2, ..., 9$, ont des valeurs différentes deux à deux et prennent comme valeur soit 0.1 soit 10. Nous obtenons alors un champ de température très hétérogène d'un sous-domaine à l'autre. Afin de capter au mieux cette hétérogénéité, la base réduite utilisée doit être suffisamment riche et robuste.

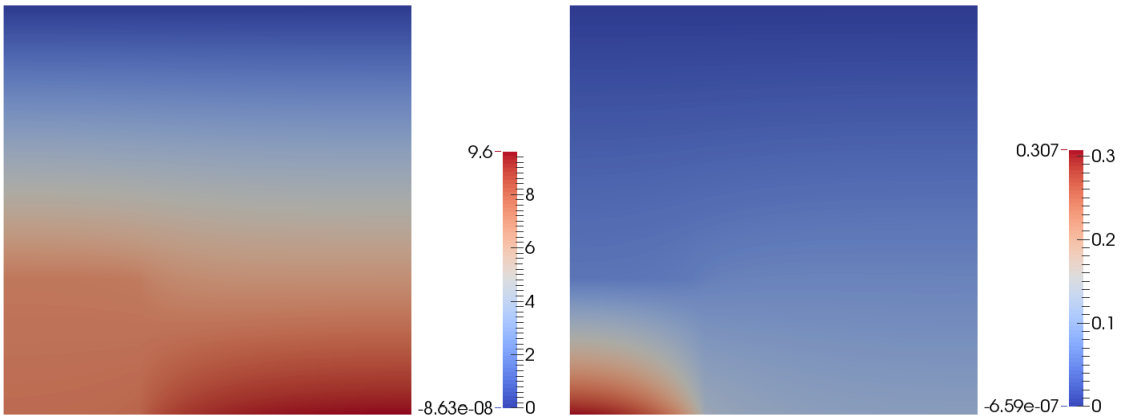


FIGURE 2.3 – Température obtenue en fixant les coefficients de conductivité thermique à 0.1 (à gauche) et 10 (à droite).

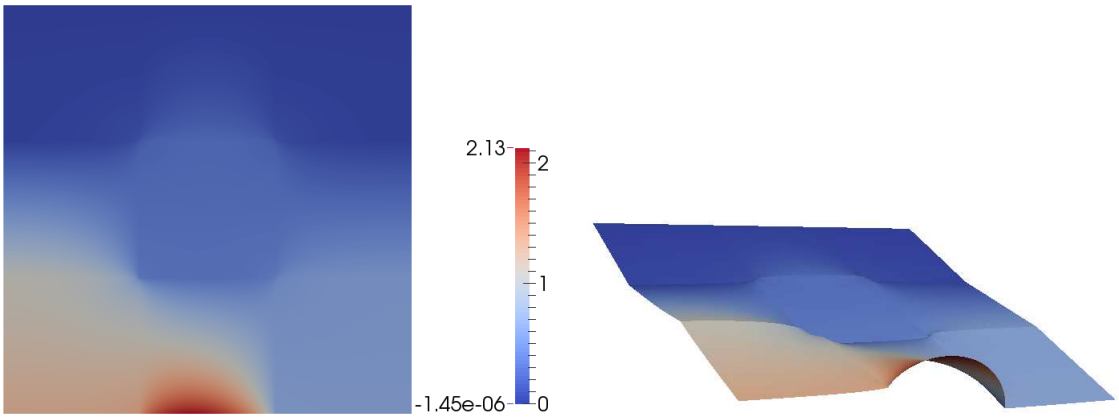


FIGURE 2.4 – Température obtenue lorsque les coefficients de conductivité thermique varient entre 0.1 et 10 d'un bloc à l'autre. À gauche : vue 2D classique, à droite le domaine est déformé suivant de la valeur du champ de température.

Convergence

À présent intéressons-nous à la convergence de l'approximation – via la méthode CRBM – de la solution du problème primal ainsi que de la sortie. Pour cela introduisons les notations suivantes :

$$e_N^{pr,rel} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|\left| u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}) \right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}}{\left|\left|\left| u_{\mathcal{N}}(\boldsymbol{\mu}) \right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}} \quad \text{et} \quad e_N^{s,rel} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left| s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu}) \right|}{\left| s_{\mathcal{N}}(\boldsymbol{\mu}) \right|}, \quad (2.187)$$

où Ξ_{train} est un échantillon de 1 400 vecteurs de paramètres sélectionnés de manière aléatoire dans l'espace $\mathcal{D}$. Les quantités $e_N^{pr,rel}$ et $e_N^{s,rel}$ désignent le maximum de l'erreur relative commise – respectivement sur la solution du problème primal et la sortie – lorsque nous parcourons l'échantillon Ξ_{train}. Le jeu de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$ est composé de $\mu_{ref}^i = 1$ pour $i = 2, ..., 9$. Rappelons que P désigne la dimension de l'espace des paramètres et que nous avons $P = 8$. Intéressons-nous à l'évolution des quantités $e_N^{pr,rel}$ et $e_N^{s,rel}$ en fonction du nombre d'éléments N de la base réduite.

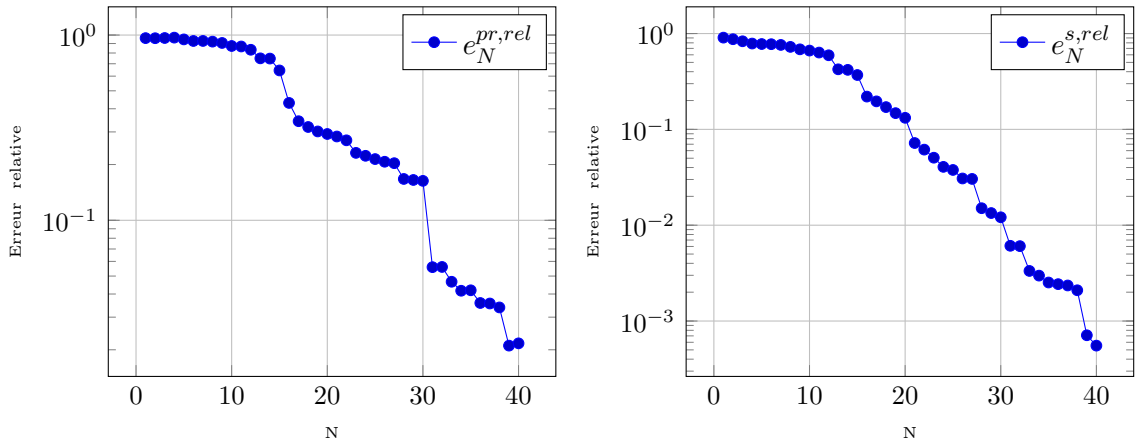


FIGURE 2.5 – Convergence de l'approximation de la solution du problème primal (à gauche) et de la sortie (à droite) – configuration P=8, $Card(\Xi_{train}) = 1400$ et $\mathcal{N} = 47\ 600$.

Nous distinguons deux phases sur les graphiques de la figure 2.5. Il apparaît clairement qu'à partir de 13 d'éléments dans la base réduite, la convergence des erreurs $e_N^{pr,rel}$ et $e_N^{s,rel}$ devient plus rapide. Nous allons donc scinder l'étude de ces courbes en deux. Pour chaque erreur – $e_N^{pr,rel}$ et $e_N^{s,rel}$ – nous étudions son évolution sur les douze premiers éléments de la base réduite, puis sur les trente-deux restants.

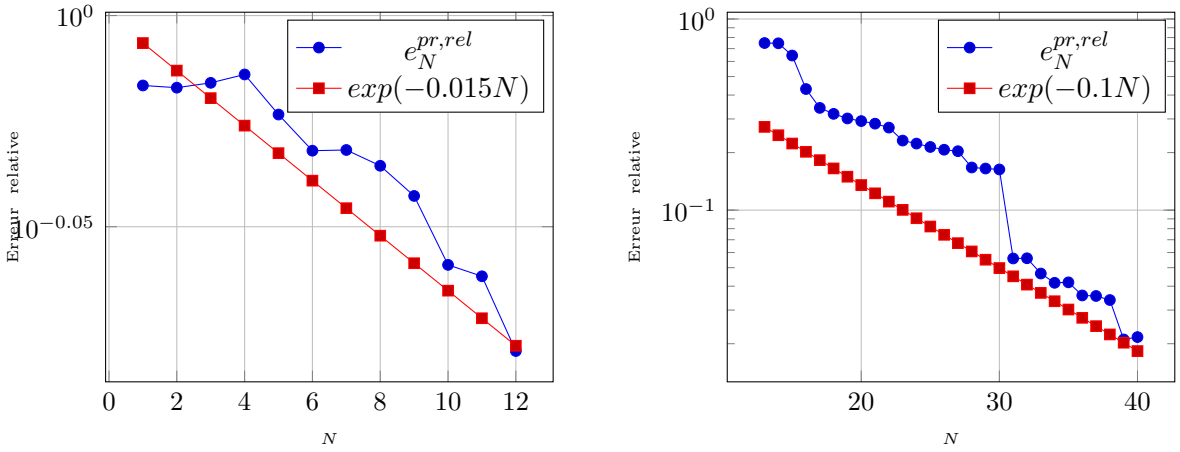


FIGURE 2.6 – Convergence de l'approximation de la solution primale pour $N = 1, ..., 12$ (à droite) et $N = 13, ..., 40$ (à gauche) – configuration P=8, $Card(\Xi_{train}) = 1400$ et $\mathcal{N} = 47\ 600$.

Concentrons-nous sur l'erreur $e_N^{pr,rel}$ (figure 2.6). Pour $1 \leq N \leq 12$, la convergence est proche de $e^{-0.015N}$ pour ensuite se rapprocher de $e^{-0.1N}$ lorsque $13 \leq N \leq 40$. Si l'on regarde l'erreur $e_N^{s,rel}$ (figure 2.7) nous constatons que la convergence est proche de $e^{-0.03N}$ pour $1 \leq N \leq 13$, et qu'ensuite elle se rapproche de $e^{-0.2N}$. Il est intéressant de remarquer c'est que la relation (2.24) est vérifiée, c'est à dire que nous avons une convergence quadratique. L'erreur commise sur la sortie du modèle converge de manière quadratique par rapport à l'erreur portant sur la solution primale.

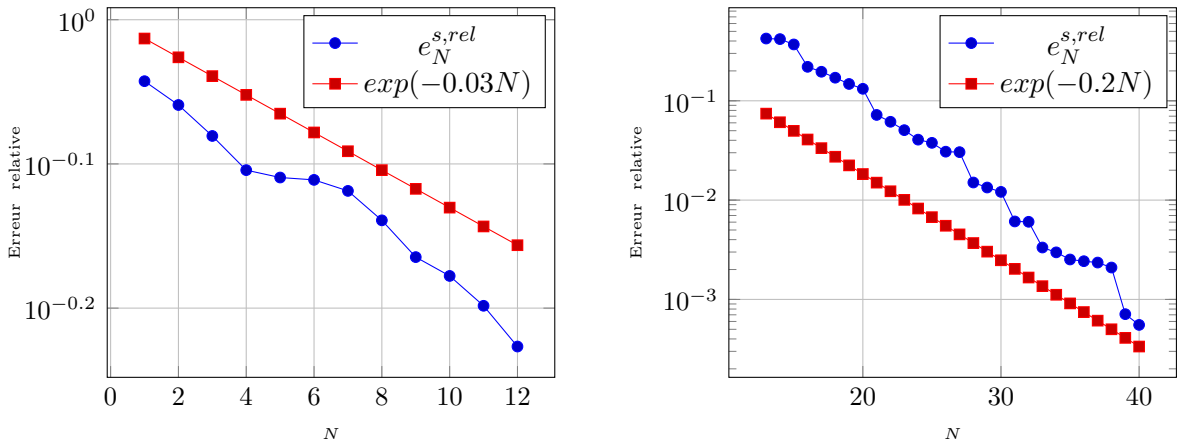


FIGURE 2.7 – Convergence de l'approximation de la sortie pour $N = 1, ..., 12$ (à droite) et $N = 13, ..., 40$ (à gauche) – configuration P=8, $Card(\Xi_{train}) = 1400$ et $\mathcal{N} = 47\ 600$.

Gardons à l'esprit que plus le nombre de paramètres qui varient est grand, plus le problème qu'il nous faut traiter est riche. Par conséquent la convergence des erreurs $e_N^{pr,rel}$ et $e_N^{s,rel}$ sera d'autant plus lente. Afin d'illustrer ce phénomène, la figure 2.8 montre les différentes convergences de $e_N^{s,rel}$ obtenues pour P allant de 1 à 8. Les coefficients de conductivité thermique associés aux sous-domaines Ω_i, pour $1 \leq i \leq 9 - P$, sont alors fixés à 1.

Estimation d'erreur a posteriori

Commençons par introduire, pour $N = 1, ..., N_{max}$, le maximum de l'estimation de l'erreur commise la sortie lorsque nous parcourons un échantillon Ξ_{train} de 1 400 vecteurs de paramètres sélectionnés de manière aléatoire dans l'espace $\mathcal{D}$:

$$\Delta_{N,max}^s = \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta_N^s(\boldsymbol{\mu}), \tag{2.188}$$

Afin de quantifier la qualité de l'estimation de l'erreur portant sur la sortie, nous introduisons les indicateurs d'efficacité pour tout $\boldsymbol{\mu} \in \mathcal{D}$, définis comme suit

$$\eta_N^s(\boldsymbol{\mu}) = \frac{\Delta_N^s(\boldsymbol{\mu})}{\left| s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu}) \right|}, \quad 1 \leq N \leq N_{max}. \tag{2.189}$$

Nous définissons également l'efficacité maximale et moyenne, respectivement $\eta_{N,max}^s$ et $\eta_{N,moy}^s$, pour $N = 1, ..., N_{max}$ par

$$\eta_{N,max}^s = \max_{\boldsymbol{\mu} \in \Xi_{train}} \eta_N^s(\boldsymbol{\mu}) \quad \text{et} \quad \eta_{N,moy}^s = \frac{1}{Card(\Xi_{train})} \sum_{\boldsymbol{\mu} \in \Xi_{train}} \eta_N^s(\boldsymbol{\mu}). \tag{2.190}$$

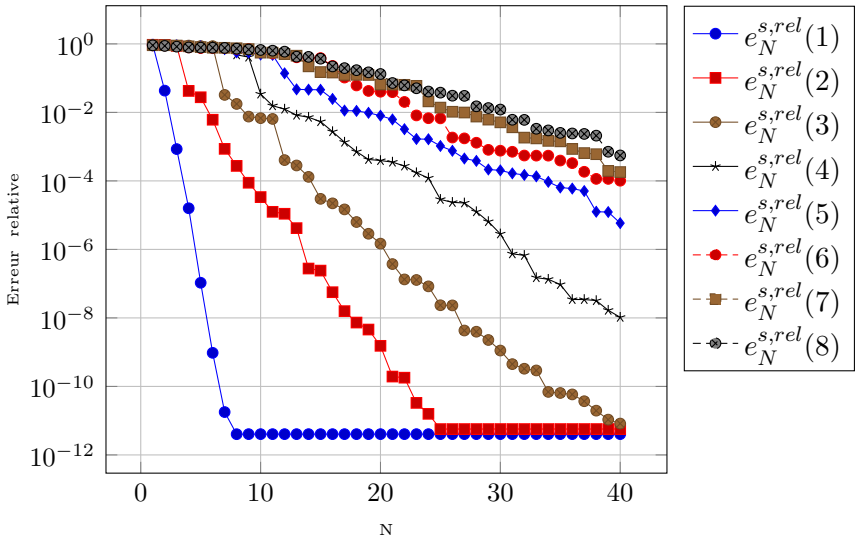


FIGURE 2.8 – Convergence de l'approximation de la sortie pour $N = 1, ..., 40$ et pour $P = 1, ..., 8$, $Card(\Xi_{train}) = 1400$ et $\mathcal{N} = 47\ 600$.

Notons que l'indicateur $\eta^s_{N,max}$ mesure la pire estimation d'erreur alors que $\eta^s_{N,moy}$ reflète le comportement moyen attendu de l'estimation d'erreur. Comme indiqué dans [90], dans le cas où l'espace des paramètres est de dimension 1 ($P = 1$), la théorie nous dit que nous pouvons majorer les indicateurs $\eta^s_{N,max}$ et $\eta^s_{N,moy}$ pour $N = 1, ..., N_{max}$ par

$$\eta^s_{N,max} \leq \eta^s_{max,UB} \equiv \sqrt{\mu^r} \qquad \text{et} \qquad \eta^s_{N,moy} \leq \eta^s_{moy,UB} \equiv \frac{2\left(\sqrt{\mu^r} - 1\right)}{ln\ \mu^r}. \tag{2.191}$$

Dans le cas où $P \geq 1$ alors nous avons

$$\eta^s_{max,UB} \equiv \mu^r. \tag{2.192}$$

Les tableaux 2.1 et 2.2 montrent l'évolution des quantités $\Delta^s_{N,max}$, $\eta^s_{N,max}$ et $\eta^s_{N,moy}$ en fonction de N, respectivement dans les cas $P = 1$ et $P = 8$. Au regard de ces tableaux nous constatons que les majorations théoriques données par (2.191) et (2.192) sont bien respectées. Dans le cas $P = 1$ nous avons bien $\eta^s_{N,max} \leq 10$ ainsi que $\eta^s_{N,moy} \leq 3.90$, et d'autre part pour $P = 8$ nous pouvons vérifier que $\eta^s_{N,max} \leq 100$.

N	$\Delta^s_{N,max}$	$\eta^s_{N,max}$	$\eta^s_{N,moy}$
1	1.2903×10^{0}	2.9145	1.5863
2	8.7591×10^{-2}	5.5289	2.2106
3	2.5717×10^{-3}	5.5291	2.3148
4	3.4022×10^{-5}	5.4349	2.2962
5	5.7924×10^{-7}	5.5832	2.3180
6	1.0819×10^{-8}	8.3138	2.2925

TABLE 2.1 – Estimations de l'erreur sur la sortie et efficacités pour le modèle des blocs thermiques avec $P = 1$, $Card(\Xi_{train}) = 1\ 400$ et $\mathcal{N} = 47\ 600$.

N	$\Delta^s_{N,max}$	$\eta^s_{N,max}$	$\eta^s_{N,moy}$
5	1.0106×10^{1}	25.7507	7.55098
10	9.9401×10^{0}	27.0787	8.23131
15	5.2582×10^{0}	30.4195	9.95733
20	3.6713×10^{0}	35.6983	10.6656
25	1.0853×10^{0}	37.4563	11.197
30	1.689×10^{-1}	39.8081	11.9026
35	8.1478×10^{-2}	39.4007	12.3345
40	2.0950×10^{-2}	47.6986	11.0619

TABLE 2.2 – Estimations de l'erreur sur la sortie et efficacités pour le modèle des blocs thermique avec $P = 8$, $Card(\Xi_{train}) = 1\ 400$ et $\mathcal{N} = 47\ 600$.

Gain de temps de calcul *hors-ligne* grâce à la proposition 1

Nous illustrons ici le gain de temps réalisé en appliquant le résultat énoncé dans la proposition 1. Nous nous plaçons dans le cas $P = 8$ et où aucune des solutions des problème de type Poisson (2.101) ne sont stockées. En nous servant des résultats de la proposition 1, seulement $\frac{N^2}{2}(Q_a+1)Q_a$ résolutions du problème (2.101) sont nécessaires pour ensuite, durant la phase *en-ligne* pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, construire le terme $C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu})$ défini par (2.105) avec $1 \leq N \leq N_{max}$ et $1 \leq i, i' \leq N$.

Introduisons t_1 le temps mis pour calculer l'ensemble des termes $\Phi^{qiq'i'}_{N_{pr},aa}$, avec $1 \leq q, q' \leq Q_a$ et $1 \leq i, i' \leq N$ pour $N = 1, ..., 15$, soit $N^2 Q_a^2$ résolutions du problème (2.101). Notons t_2 le temps mis pour avoir accès aux mêmes quantités mais en ne faisant que $\frac{N^2}{2}(Q_a+1)Q_a$ résolutions du problème (2.101) grâce à la proposition 1. Notons que dans notre cas nous avons $Q_a = 10$.

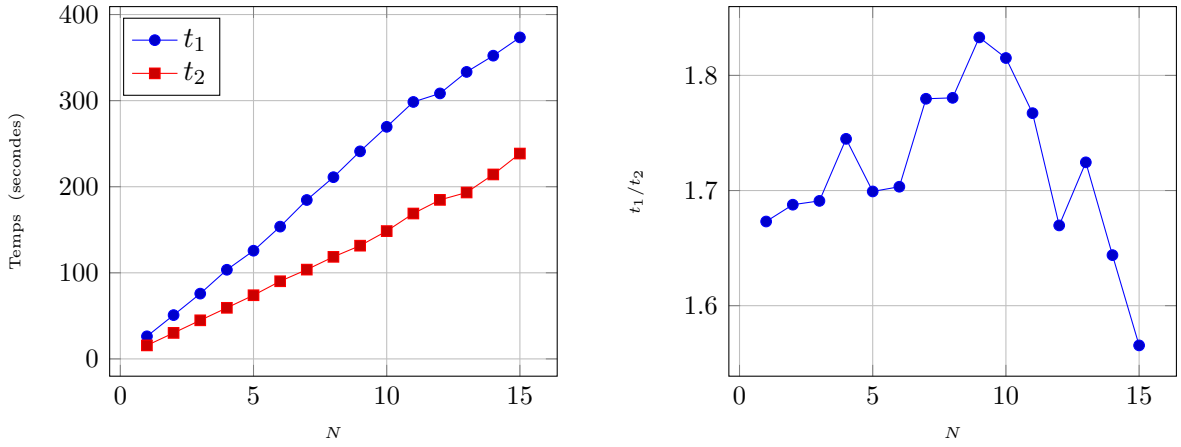


FIGURE 2.9 – Évolution des temps t_1 et t_2 en fonction de N (à gauche) ainsi que le ratio t_1/t_2 (à droite) – configuration P=8 et $\mathcal{N} = 3\ 200$.

Pour avoir accès à l'ensemble des termes $\Phi^{qiq'i'}_{N_{pr},aa}$, avec $1 \leq q, q' \leq Q_a$ et $1 \leq i, i' \leq N$ pour $N = 1, ..., 15$, lorsque nous nous servons de la proposition 1 nous allons au minimum 1.55 plus vite que lorsque nous construisons tous les termes.

Temps de calcul *en-ligne*

Considérons un entier N et un $\boldsymbol{\mu}$ donnés, tels que $1 \leq N \leq N_{max}$ et $\boldsymbol{\mu} \in \mathcal{D}$. Nous nous proposons de quantifier le gain de temps de calcul pour l'évaluation de la sortie $s_N(\boldsymbol{\mu})$ certifiée (avec une estimation de l'erreur) en comparaison de $s_{\mathcal{N}}(\boldsymbol{\mu})$. Pour cela on se place dans le cas $P = 8$, avec une discrétisation $\mathbb{P}_1$ et 3 200 degrés de liberté sur un seul processeur. Nous avons donc $\mathcal{N} = 3\ 200$. Le tableau 2.3 contient le temps de calcul moyen – sur 20 000 simulations – nécessaire pour évaluer la sortie $s_N(\boldsymbol{\mu})$ ainsi que l'estimation d'erreur associée $\Delta_N^s(\boldsymbol{\mu})$, en fonction de la dimension N de la base réduite. Ces temps de calculs sont normalisés par rapport au temps moyen mis pour évaluer la sortie $s_{\mathcal{N}}(\boldsymbol{\mu})$ en utilisant la méthode FE. Afin d'être le plus général possible, nous traitons notre

N	$s_N(\boldsymbol{\mu})$	$\Delta_N^s(\boldsymbol{\mu})$	$s_{\mathcal{N}}(\boldsymbol{\mu})$
10	1.4263×10^{-3}	1.2249×10^{-3}	1
20	1.8977×10^{-3}	1.5402×10^{-3}	1
30	1.6260×10^{-3}	1.7690×10^{-3}	1
40	2.0917×10^{-3}	1.9145×10^{-3}	1

TABLE 2.3 – Temps de calcul *en-ligne* moyen – sur 20 000 simulations – pour évaluer la sortie et l'estimation d'erreur associée. Ces temps sont normalisés par rapport au temps de calcul moyen de $s_{\mathcal{N}}(\boldsymbol{\mu})$ – configuration P=8 avec $\mathcal{N} = 3\ 200$.

sortie "souple" comme s'il s'agissait d'une sortie "non souple" nécessitant la résolution d'un problème dual. Notons que l'évaluation de la sortie $s_N(\boldsymbol{\mu})$ implique la résolution des problèmes réduits primal et dual, ainsi que l'ajout du terme de correction – voir (2.38) –. Nous remarquons que pour $N = 40$, il est plus rapide – avec un facteur 250 – d'évaluer $s_N(\boldsymbol{\mu})$ de manière certifiée que d'évaluer $s_{\mathcal{N}}(\boldsymbol{\mu})$. Cependant, lorsque $\mathcal{N}$ augmente, le temps de calcul *en-ligne* reste le même car il est indépendant de $\mathcal{N}$. Relançons les simulations en prenant cette fois $\mathcal{N} = 5\ 800$, nous obtenons alors le tableau 2.4. En restant sur un

N	$s_N(\boldsymbol{\mu})$	$\Delta_N^s(\boldsymbol{\mu})$	$s_{\mathcal{N}}(\boldsymbol{\mu})$
10	8.7006×10^{-4}	7.4717×10^{-4}	1
20	9.9183×10^{-4}	9.3950×10^{-4}	1
30	1.1575×10^{-3}	1.0790×10^{-3}	1
40	1.2759×10^{-3}	1.1678×10^{-3}	1

TABLE 2.4 – Temps de calcul *en-ligne* moyen – sur 20 000 simulations – pour évaluer la sortie et l'estimation d'erreur associée. Ces temps sont normalisés par rapport au temps de calcul moyen de $s_{\mathcal{N}}(\boldsymbol{\mu})$ – configuration P=8 avec $\mathcal{N} = 5\ 800$.

seul processeur, lorsque $\mathcal{N}$ passe de 3 200 à 5 800, pour $N = 40$ le facteur de gain passe de 250 à 410.

5.2 Problème du bouclier thermique

Considérons le problème de bouclier thermique introduit dans la thèse de M.Grepl [53] qui est un problème parabolique. Le domaine d'étude $\Omega \equiv \{[0, 10]\times[0, 4]\}\setminus\{(]1, 3[\times]1, 3[)\cup$

$(]4,6[\times]1,3[) \cup (]7,9[\times]1,3[)\}$ – un segment du bouclier thermique 2D – est illustré par la figure (2.10). Notons $\partial\Omega_{ext}$ le bord gauche, situé en $x = 0$ et $0 \leq y \leq 4$. Le bord inté-

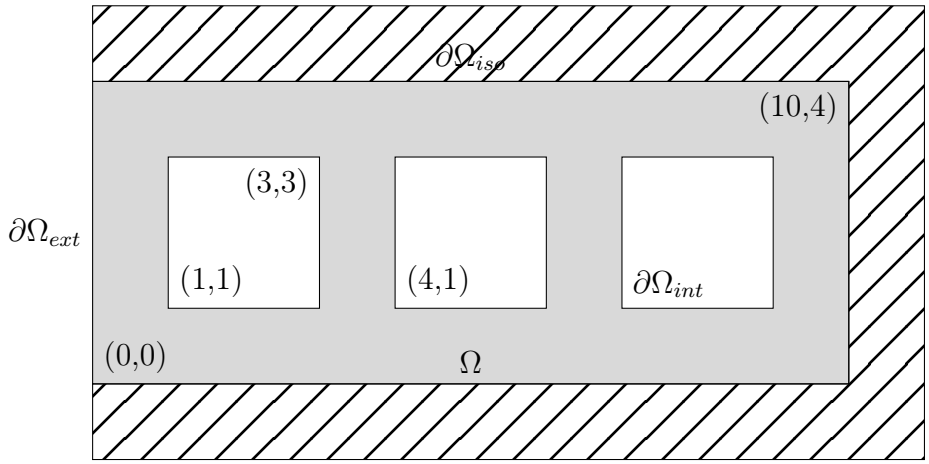


FIGURE 2.10 – Domaine d'étude Ω pour le problème du bouclier thermique.

rieur des trous qui correspondent aux surfaces des canaux de refroidissement $]1,3[\times]1,3[$, $]4,6[\times]1,3[$ et $]7,9[\times]1,3[$. Enfin notons $\partial\Omega_{iso}$ les bords extérieurs restants, à savoir les bords haut, bas et droit. Durant toute la durée de la simulation le bord $\partial\Omega_{in}$ est exposé à une haute température $T_{ext} = 1$, normalisée à l'unité. Quant aux canaux de refroidissement $\partial\Omega_{int}$, ils sont en contact avec de l'air à température $T_{air} = 0$. Les bords regroupés sous la dénomination $\partial\Omega_{iso}$ sont thermiquement isolés. Les coefficients de transfert thermique adimensionnalisés pour $\partial\Omega_{ext}$ et $\partial\Omega_{int}$ sont donnés par le nombre de Biot, respectivement B_{ext} et B_{int}. Ces deux coefficients de transfert thermique sont nos paramètres pour ce problème. Nous avons donc $\boldsymbol{\mu} = (\mu_1, \mu_2) = (B_{ext}, B_{int}) \in \mathcal{D} \equiv [0.01, 0.5] \times [0.001, 0.1] \subset \mathbb{R}^{P=2}$. Nous considérons que la simulation s'arrête à $T_f = 20$ et prenons un pas de temps $\Delta t = 0.2$ ce qui entraîne que nous avons $K = 100$. Dans ce cas test la quantité d'intérêt est la température moyenne dans le domaine Ω. Rappelons que $X_{\mathcal{N}}$ est un espace de fonctions approximant X tel que $X_{\mathcal{N}} \subset X \equiv H^1(\Omega)$. La formulation variationnelle consiste à chercher $u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}$ tel que, pour tout $v \in X_{\mathcal{N}}$ et $1 \leq k \leq K$:

$$\frac{1}{\Delta t} m\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_{\mathcal{N}}(\boldsymbol{\mu}, t^{k-1}), v; \boldsymbol{\mu}\Big) + a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}; t^k\Big), \tag{2.193}$$

avec à l'instant initial $u_{\mathcal{N}}(\boldsymbol{\mu}, t^0) = 0$. Les formes bilinéaires m et a ainsi que les formes linéaires f et ℓ s'écrivent :

$$m\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_{\mathcal{N}}(\boldsymbol{\mu}, t^{k-1}), v; \boldsymbol{\mu}\Big) = \int_\Omega \Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_{\mathcal{N}}(\boldsymbol{\mu}, t^{k-1})\Big) v, \tag{2.194}$$

$$\begin{aligned} a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}, t^k), v; \boldsymbol{\mu}\Big) = & \int_\Omega \nabla u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \cdot \nabla v \\ & + \mu_1 \int_{\partial\Omega_{ext}} u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) v + \mu_2 \int_{\partial\Omega_{int}} u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) v, \end{aligned} \tag{2.195}$$

$$f\Big(v; \boldsymbol{\mu}; t^k\Big) = \int_{\partial\Omega_{ext}} v, \quad 1 \leq k \leq K, \tag{2.196}$$

et

$$\ell\Big(v;\boldsymbol{\mu};t^k\Big) = \frac{1}{|\Omega|}\int_\Omega v, \quad 1 \leq k \leq K. \tag{2.197}$$

Prenons une discrétisation FE $\mathbb{P}_2$ avec 33 000 degrés de liberté distribués sur 16 processeurs et illustrons l'impact des paramètres μ^0 et μ^1. La figure 2.11 illustre le champ de température, solution de (2.193) au temps final t^{100} lorsque les paramètres sont fixés au minimum puis au maximum.

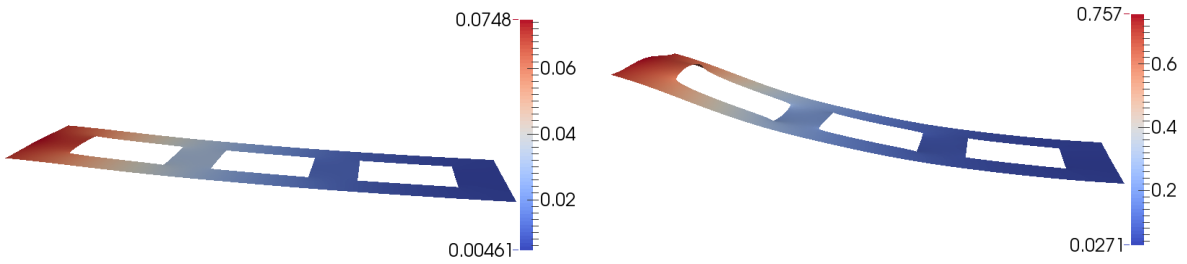


FIGURE 2.11 – Température obtenue en fixant $\boldsymbol{\mu} = (10^{-2}, 10^{-3})$ (à gauche) et $\boldsymbol{\mu} = (5 \times 10^{-1}, 10^{-1})$ (à droite). Le domaine est déformé suivant de la valeur du champ de température.

Convergence

Nous présentons ici la convergence de l'approximation – via la méthode CRBM – de la solution du problème primal, mais également du problème dual, ainsi que de la sortie. Nous considérons le cas où le modèle est stationnaire – en faisant abstraction des contributions de la forme bilinéaire m – et le cas où il est transitoire (2.193).

Soit Ξ_{train} un échantillon de 1 000 vecteurs de paramètres sélectionnés de manière aléatoire dans l'espace $\mathcal{D}$. Introduisons les notations suivantes :

$$e_N^{pr,rel} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\Big|\Big|\Big|u_\mathcal{N}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\Big|\Big|\Big|_{a,\boldsymbol{\mu}_{ref}}}{\Big|\Big|\Big|u_\mathcal{N}(\boldsymbol{\mu})\Big|\Big|\Big|_{a,\boldsymbol{\mu}_{ref}}}, \tag{2.198}$$

$$e_N^{du,rel} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\Big|\Big|\Big|\Psi_\mathcal{N}(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu})\Big|\Big|\Big|_{a,\boldsymbol{\mu}_{ref}}}{\Big|\Big|\Big|\Psi_\mathcal{N}(\boldsymbol{\mu})\Big|\Big|\Big|_{a,\boldsymbol{\mu}_{ref}}}, \tag{2.199}$$

$$e_N^{s,rel} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\Big|s_\mathcal{N}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})\Big|}{\Big|s_\mathcal{N}(\boldsymbol{\mu})\Big|}, \tag{2.200}$$

$$e_N^{pr,rel}(t^k) = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\Big|\Big|\Big|u_\mathcal{N}(\boldsymbol{\mu},t^k) - u_N(\boldsymbol{\mu},t^k)\Big|\Big|\Big|_{pr,\boldsymbol{\mu}_{ref}}}{\Big|\Big|\Big|u_\mathcal{N}(\boldsymbol{\mu},t^k)\Big|\Big|\Big|_{pr,\boldsymbol{\mu}_{ref}}}, \quad 1 \leq k \leq K, \tag{2.201}$$

$$e_N^{du,rel}(t^k) = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\Big|\Big|\Big|\Psi_\mathcal{N}(\boldsymbol{\mu},t^k) - \Psi_N(\boldsymbol{\mu},t^k)\Big|\Big|\Big|_{du,\boldsymbol{\mu}_{ref}}}{\Big|\Big|\Big|\Psi_\mathcal{N}(\boldsymbol{\mu},t^k)\Big|\Big|\Big|_{du,\boldsymbol{\mu}_{ref}}}, \quad 1 \leq k \leq K, \tag{2.202}$$

$$e_N^{s,rel}(t^k) = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|s_{\mathcal{N}}(\boldsymbol{\mu},t^k) - s_N(\boldsymbol{\mu},t^k)\right|}{\left|s_{\mathcal{N}}(\boldsymbol{\mu},t^k)\right|}, \quad 1 \leq k \leq K. \tag{2.203}$$

Le jeu de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$ qui a été choisi pour ce modèle est $\boldsymbol{\mu}_{ref} = (10^{-2}, 10^{-3})$. Les quantités $e_N^{pr,rel}$, $e_N^{du,rel}$ et $e_N^{s,rel}$ désignent le maximum de l'erreur relative commise – respectivement sur la solution du problème primal, dual et la sortie – lorsque nous parcourons l'échantillon Ξ_{train}. Pour un entier k donné tel que $1 \leq k \leq K$, les quantités $e_N^{pr,rel}(t^k)$, $e_N^{du,rel}(t^k)$ et $e_N^{s,rel}(t^k)$ désignent le maximum de l'erreur relative commise – respectivement sur la solution du problème primal, dual et la sortie – au temps t^k lorsque nous parcourons l'échantillon Ξ_{train}.

Commençons par considérer le cas stationnaire et intéressons-nous à l'évolution des quantités $e_N^{pr,rel}$, $e_N^{du,rel}$ (figure 2.12) et $e_N^{s,rel}$ (figure 2.13) en fonction du nombre d'éléments N de la base réduite.

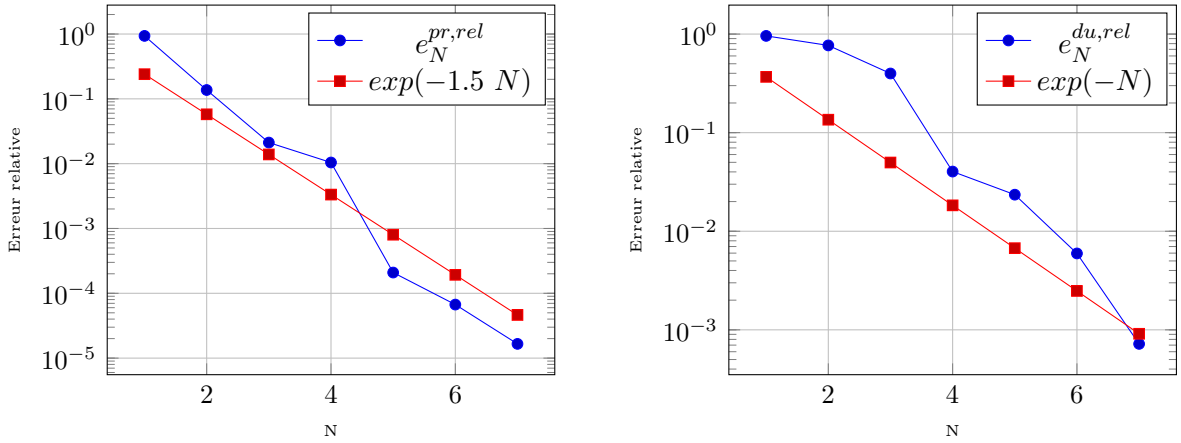


FIGURE 2.12 – Convergence de l'approximation de la solution du problème primal (à gauche) et dual (à droite) – configuration stationnaire avec $\Xi_{train} = 1\ 000$ et $\mathcal{N} = 33\ 000$.

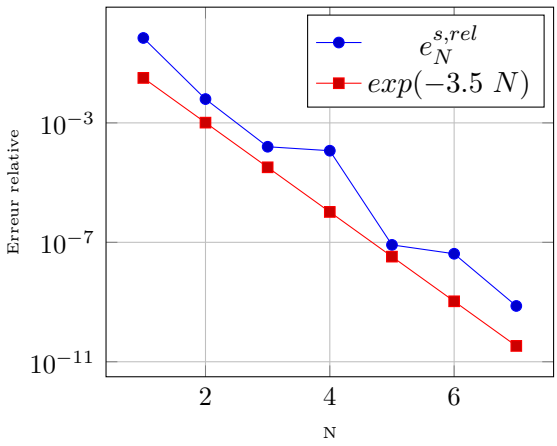


FIGURE 2.13 – Convergence de l'approximation de la sortie – configuration stationnaire avec $\Xi_{train} = 1\ 000$ et $\mathcal{N} = 33\ 000$.

Comme attendu nous avons bien une convergence exponentielle des différentes approximations. Étudions l'approximation de la sortie. Nous remarquons que la relation

(2.33) est vérifiée, c'est à dire que même lorsque la sortie n'est pas "souple" nous avons une convergence quadratique grâce à l'introduction du problème dual.

Considérons maintenant le cas transitoire et intéressons-nous à l'évolution des quantités $e_N^{pr,rel}(t^K)$, $e_N^{du,rel}(t^K)$ (figure 2.12) et $e_N^{s,rel}(t^K)$ (figure 2.13) au temps final t^K en fonction N. Rappelons que nous utilisons un algorithme couplé $POD/glouton$ pour traiter les problèmes paraboliques et que nous avons noté N_m est le nombre de modes propres utilisé pour enrichir la base réduite à chaque itération. Nous avons pris ici $N_m = 3$.

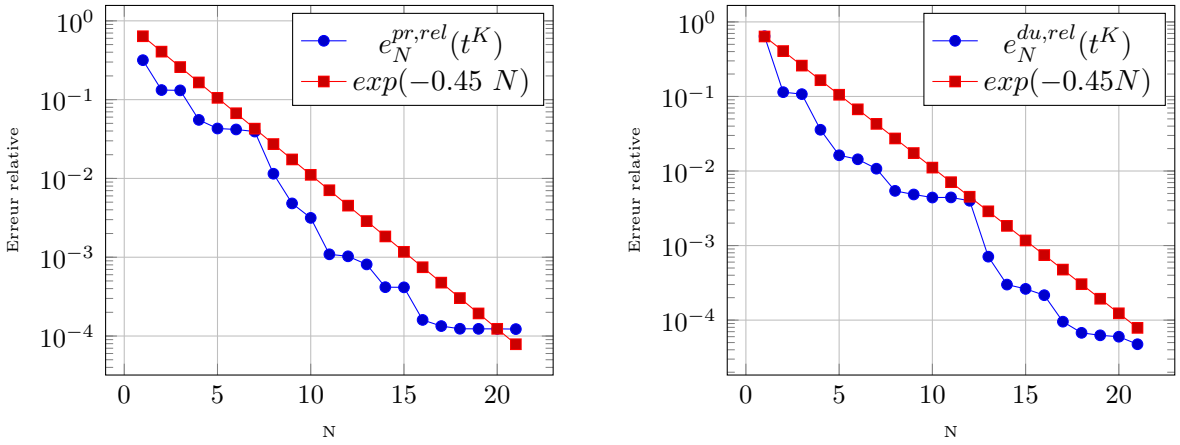


FIGURE 2.14 – Convergence de l'approximation de la solution du problème primal (à gauche) et dual (à droite) au temps final t^K – configuration transitoire avec $\Xi_{train} = 1\ 000$ et $\mathcal{N} = 33\ 000$.

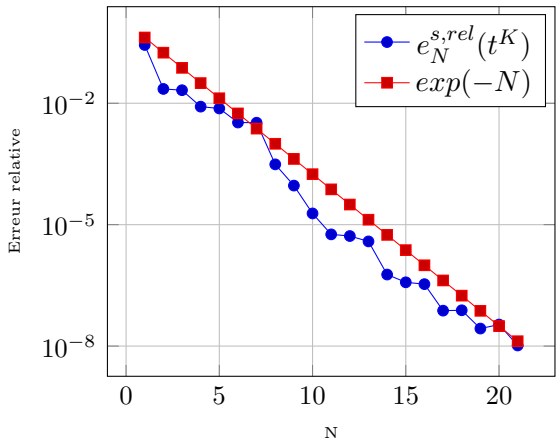


FIGURE 2.15 – Convergence de l'approximation de la sortie au temps final t^K – configuration transitoire avec $\Xi_{train} = 1\ 000$ et $\mathcal{N} = 33\ 000$.

Là encore nous avons une convergence exponentielle des différentes approximations. Nous remarquons que la relation (2.86) est vérifiée, c'est à dire que nous avons une convergence quadratique de l'approximation de la sortie bien qu'elle ne soit pas "souple".

Estimation d'erreur a posteriori

Commençons par introduire, pour $N = 1, ..., N_{max}$, les estimations des erreurs relatives commises sur la sortie

$$\Delta_N^{s,rel}(\boldsymbol{\mu}) = \frac{\Delta_N^s(\boldsymbol{\mu})}{s_{\mathcal{N}}(\boldsymbol{\mu})}, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{2.204}$$

$$\Delta_N^{s,rel}(\boldsymbol{\mu}, t^k) = \frac{\Delta_N^s(\boldsymbol{\mu}, t^k)}{s_{\mathcal{N}}(\boldsymbol{\mu}, t^k)}, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \quad 1 \leq k \leq K. \tag{2.205}$$

Définissons maintenant les maximums des estimations d'erreurs relatives commises sur la sortie lorsque nous parcourons un échantillon Ξ_{train} de 1 000 vecteurs de paramètres sélectionnés de manière aléatoire dans l'espace $\mathcal{D}$:

$$\Delta_{N,max}^{s,rel} = \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta_N^{s,rel}(\boldsymbol{\mu}), \quad 1 \leq N \leq N_{max}, \tag{2.206}$$

$$\Delta_{N,max}^{s,rel}(t^k) = \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta_N^{s,rel}(\boldsymbol{\mu}, t^k), \quad 1 \leq N \leq N_{max}, \quad 1 \leq k \leq K. \tag{2.207}$$

Afin de quantifier la qualité des estimations d'erreur, nous introduisons les indicateurs d'efficacité pour tout $\boldsymbol{\mu} \in \mathcal{D}$, définis comme suit

$$\eta_N^{s,rel}(\boldsymbol{\mu}) = \frac{\Delta_N^{s,rel}(\boldsymbol{\mu})}{e_N^{s,rel}}, \quad 1 \leq N \leq N_{max}, \tag{2.208}$$

$$\eta_N^{s,rel}(\boldsymbol{\mu}, t^k) = \frac{\Delta_N^{s,rel}(\boldsymbol{\mu}, t^k)}{e_N^{s,rel}(t^k)}, \quad 1 \leq N \leq N_{max}. \tag{2.209}$$

Nous définissons également les efficacités moyennes $\eta_{N,moy}^{s,rel}$ et $\eta_{N,moy}^{s,rel}(t^k)$, pour $N = 1, ..., N_{max}$ et $1 \leq k \leq K$ par

$$\eta_{N,moy}^{s,rel} = \frac{1}{Card(\Xi_{train})} \sum_{\boldsymbol{\mu} \in \Xi_{train}} \eta_N^{s,rel}(\boldsymbol{\mu}), \tag{2.210}$$

$$\eta_{N,moy}^{s,rel}(t^k) = \frac{1}{Card(\Xi_{train})} \sum_{\boldsymbol{\mu} \in \Xi_{train}} \eta_N^{s,rel}(\boldsymbol{\mu}, t^k). \tag{2.211}$$

Les indicateurs $\eta_{N,moy}^{s,rel}$ et $\eta_{N,moy}^{s,rel}(t^k)$,$1 \leq k \leq K$, reflètent le comportement moyen attendu de l'estimation d'erreur. Le tableau 2.5 montre l'évolution des quantités $e_N^{s,rel}$, $\Delta_{N,max}^{s,rel}$, et $\eta_{N,moy}^{s,rel}$ en fonction de N pour la configuration stationnaire.

N	$e_N^{s,rel}$	$\Delta_{N,max}^{s,rel}$	$\eta_{N,moy}^{s,rel}$
1	6.9780×10^{-1}	2.1813×10^{0}	4.1008
2	6.2731×10^{-3}	2.2681×10^{-2}	5.4421
3	1.5877×10^{-4}	2.8642×10^{-3}	24.2519
4	1.1698×10^{-4}	1.9189×10^{-4}	8.6667
5	8.1996×10^{-8}	6.1485×10^{-7}	32.1043
6	4.1363×10^{-8}	1.1621×10^{-7}	7.6043
7	7.3884×10^{-10}	2.4091×10^{-9}	47.5920

TABLE 2.5 – Estimations de l'erreur sur la sortie et efficacités pour le modèle du bouclier thermique – configuration stationnaire avec $\Xi_{train} = 1\ 000$ et $\mathcal{N} = 33\ 000$.

Le tableau 2.6 quant à lui montre l'évolution des quantités $e_N^{s,rel}(t^K)$, $\Delta_{N,max}^{s,rel}(t^K)$, et $\eta_{N,moy}^{s,rel}(t^K)$ au temps final t^K en fonction de N pour la configuration transitoire.

N	$e_N^{s,rel}(t^K)$	$\Delta_{N,max}^{s,rel}(t^K)$	$\eta_{N,moy}^{s,rel}(t^K)$
3	2.1009×10^{-2}	1.64939×10^{-1}	14.4049
6	3.3396×10^{-3}	1.3039×10^{-2}	4.4482
9	9.2305×10^{-5}	1.23459×10^{-4}	5.3056
12	5.2391×10^{-6}	1.6893×10^{-5}	31.48
15	3.7650×10^{-7}	1.9469×10^{-6}	29.31
18	7.6675×10^{-8}	1.7472×10^{-7}	10.714
21	1.0338×10^{-8}	1.0906×10^{-7}	46.751

TABLE 2.6 – Estimations de l'erreur sur la sortie et efficacités pour le modèle du bouclier thermique au temps final t^K – configuration transitoire avec $\Xi_{train} = 1\ 000$ et $\mathcal{N} = 33\ 000$.

Nous remarquons que les indicateurs d'efficacités présentés dans les tableaux 2.5 et 2.6 fluctuent en fonction de N. Les résultats présentés dans [53] sur ce modèle du bouclier thermique montrent cette même tendance. Pour l'heure nous n'avons pas encore d'explications à cette fluctuation.

Temps de calcul *en-ligne*

Considérons un entier N et un $\boldsymbol{\mu}$ donnés, tels que $1 \leq N \leq N_{max}$ et $\boldsymbol{\mu} \in \mathcal{D}$. Nous nous proposons de quantifier le gain de temps de calcul pour l'évaluation de la sortie $s_N(\boldsymbol{\mu}, t^K)$ certifiée (avec une estimation de l'erreur) au temps final t^K en comparaison de $s_{\mathcal{N}}(\boldsymbol{\mu}, t^K)$. Pour cela on se place dans la configuration transitoire, avec une discrétisation $\mathbb{P}_1$ et 3 000 degrés de liberté . Nous avons donc $\mathcal{N} = 3\ 000$. Le tableau 2.7 contient le temps de calcul moyen – sur 20 000 simulations – nécessaire pour évaluer la sortie $s_N(\boldsymbol{\mu}, t^K)$ ainsi que l'estimation d'erreur associée $\Delta_N^s(\boldsymbol{\mu}, t^K)$, en fonction de la dimension N de la base réduite. Ces temps de calculs sont normalisés par rapport au temps moyen mis pour évaluer la sortie $s_{\mathcal{N}}(\boldsymbol{\mu}, t^K)$ en utilisant la méthode FE. Notons que l'évaluation de la sortie

N	$s_N(\boldsymbol{\mu}, t^K)$	$\Delta_N^s(\boldsymbol{\mu}, t^K)$	$s_{\mathcal{N}}(\boldsymbol{\mu}, t^K)$
5	3.4982×10^{-3}	5.7433×10^{-4}	1
10	3.8960×10^{-3}	5.8151×10^{-4}	1
15	4.1372×10^{-3}	5.9779×10^{-4}	1
21	4.5376×10^{-3}	6.1688×10^{-4}	1

TABLE 2.7 – Temps de calcul *en-ligne* pour évaluer la sortie au temps final t^K et l'estimation d'erreur associée. Ces temps sont normalisés par rapport au temps de calcul moyen – sur 20 000 simulations – de $s_{\mathcal{N}}(\boldsymbol{\mu}, t^K)$. $\mathcal{N} = 3\ 000$.

$s_N(\boldsymbol{\mu}, t^K)$ implique, pour chaque pas de temps, la résolution des problèmes réduits primal et dual, ainsi que l'ajout du terme de correction – voir (2.82) –. Nous remarquons que pour $N = 21$, il est plus rapide – avec un facteur proche de 200 – d'évaluer $s_N(\boldsymbol{\mu}, t^K)$ de manière certifiée que d'évaluer $s_{\mathcal{N}}(\boldsymbol{\mu}, t^K)$.

Résumé

Nous avons vu dans ce chapitre comment appliquer la méthode CRBM aux problèmes linéaires, aussi bien elliptiques que paraboliques. Le calcul de la norme duale des résidus de manière à profiter de l'efficacité de la stratégie hors-ligne/en-ligne a été détaillé dans la section 3. Dans la section 4 nous avons exposé une méthode – avec les dernières améliorations – permettant d'obtenir rapidement l'évaluation de la borne inférieure de la constante de coercivité, ingrédient indispensable à l'estimateur d'erreur a posteriori. Les propositions 1 et 2 permettent d'optimiser le temps de calcul hors-ligne en vue d'utiliser l'estimation d'erreur a posteriori. Enfin, la dernière section a permis de présenter différents résultats numériques obtenus sur des modèles 2D elliptique (problème des blocs thermiques) et parabolique (problème du bouclier thermique), à l'aide du framework base réduite décrit dans la deuxième partie de ce manuscrit.

Chapitre 3

La méthode des bases réduites appliquée aux problèmes non-affines et non-linéaires

Le chapitre 2 expose la mise en oeuvre de la méthode CRBM dans le cas de problèmes elliptiques et paraboliques linéaires qui dépendent de manière affine du vecteur de paramètres $\boldsymbol{\mu}$. Cette dépendance est décrite en (2.9) et (2.50). Nous allons maintenant nous intéresser dans ce chapitre aux problèmes, linéaires et non-linéaires, qui ne dépendent pas de manière affine de $\boldsymbol{\mu}$. Lorsque l'on perd la dépendance affine en paramètres, la décomposition affine – qui joue un role clé pour la mise en oeuvre de la méthode CRBM de manière efficace – ne peut plus être écrite directement par inspection des forme linéaires et bilinéaires. Afin d'illustrer ces propos, considérons un domaine Ω borné, régulier, dans $\mathbb{R}^2$, ainsi que la forme bilinéaire $a : X_{\mathcal{N}} \times X_{\mathcal{N}} \times E^{\zeta} \to \mathbb{R}$, avec $\boldsymbol{x} \equiv (x^0, x^1) \in \Omega \in \mathbb{R}^2$ et $\boldsymbol{\mu} \equiv (\mu^0, \mu^1) \in \mathcal{D} \subset \mathbb{R}^2$, et $E^{\zeta} = \mathcal{D} \times \mathbb{R}^2$,

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big) = \underbrace{\mu^0 \int_{\Omega} \nabla u_{\mathcal{N}}(\boldsymbol{\mu}) \cdot \nabla v}_{a_0\left(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}\right)} + \underbrace{\int_{\Omega} \zeta(\boldsymbol{\mu}; \boldsymbol{x})\ u_{\mathcal{N}}(\boldsymbol{\mu})\ v}_{a_1\left(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\right)}, \tag{3.1}$$

où la fonction $\zeta(\boldsymbol{\mu}; \boldsymbol{x}) : E^{\zeta} \to \mathbb{R}$ est définie par

$$\zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}\Big) = \frac{1}{\sqrt{(x^0 - \mu^0)^2 + (x^1 - \mu^1)^2}}. \tag{3.2}$$

Comme indiqué par (3.1) il est possible de décomposer la forme bilinéaire a via

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big) = a_0\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) + a_1\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big). \tag{3.3}$$

Nous remarquons immédiatement, au vu de (3.2), que $a_1(u, v; \zeta(\boldsymbol{\mu}; \boldsymbol{x}))$ n'est pas affine en $\boldsymbol{\mu}$. Par conséquent, pour tout nouveau $\boldsymbol{\mu} \in \mathcal{D}$, l'évaluation de $\zeta(\boldsymbol{\mu}; \boldsymbol{x})$ dépend de la (grande) dimension $\mathcal{N}$. Notons que la fonction ζ peut également dépendre de la solution de l'EDP $u_{\mathcal{N}}(\boldsymbol{\mu})$, nous avons alors $\zeta : \mathcal{D} \times \mathbb{R}^2 \times X_{\mathcal{N}} \to \mathbb{R}$ définie par :

$$\zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) = \frac{x^0}{1 + \mu^0\Big(u_{\mathcal{N}}(\boldsymbol{\mu}) - \mu^1\Big)}. \tag{3.4}$$

Heureusement la méthode d'interpolation empirique [16, 56, 53, 79, 29, 47, 74] – *ou Empirical Interpolation Method (EIM)* – permet d'obtenir une approximation de la décomposition affine utilisée par la méthode CRBM. La méthode EIM occupe une place très importante dans Feel++ car elle peut être utilisée pour traiter des problèmes linéaires affines, non-affines ou encore non-linéaires [42]. Dans le cas des problèmes linéaires admettant directement une dépendance affine en paramètres la méthode EIM renvoie exactement la même décomposition affine que le problème initial. De plus, cela permet d'unifier l'interface d'accès à la méthode CRBM depuis les applications.

Dans ce chapitre, nous commencerons par présenter la méthode EIM. Ensuite nous exposerons comment cette méthode est intégrée dans la méthode CRBM pour traiter des problèmes linéaires, elliptiques et paraboliques, qui ne sont s'expriment pas naturellement de manière affine en $\boldsymbol{\mu}$. Nous nous intéresserons ensuite au cas des problèmes non-linéaires elliptiques, et enfin nous présenterons les résultats numériques obtenus sur un modèle de diffusion, linéaire, non-affine en paramètres.

1 Méthode d'interpolation empirique

Nous présentons dans cette section la méthode d'interpolation empirique qui, à l'instar de la méthode CRBM, est basée sur une stratégie *hors-ligne/en-ligne* efficace.

1.1 Calcul des coefficients de l'expansion EIM

Considérons la fonction non-linéaire suffisamment régulière $\zeta\big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\big) : \mathcal{D}\times\mathbb{R}^d\times X_{\mathcal{N}}$, où $1\leq d\leq 3$, non-affine par rapport à $\boldsymbol{\mu}$ et qui dépend d'une solution $u_{\mathcal{N}}(\boldsymbol{\mu})$ issue d'une EDP paramétrée. La méthode EIM permet d'avoir une approximation de $\zeta\big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\big)$ par une expansion RB collaterale $\zeta_M\big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\big)$. Cette expansion est affine en paramètres et est construite de telle manière à avoir

$$\zeta_M\Big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\Big)=\sum_{m=1}^{M}\beta^m\Big(\boldsymbol{\mu};u_{\mathcal{N}}(\boldsymbol{\mu})\Big)\ \hat{\zeta}^m(\boldsymbol{x}). \tag{3.5}$$

Pour ce faire, introduisons un ensemble d'échantillons emboîtés $S_M=\{\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_M\}$ ainsi que les espace emboîtés associés $W_M^{\zeta}=span\{\xi_m\equiv\zeta\big(\boldsymbol{\mu}_m;\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu}_m)\big),\ 1\leq m\leq M\}$ dans lesquels l'approximation $\zeta_M\big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\big)$ résidera. Soit Ξ_{train} qui, comme introduit dans le chapitre 2, est un échantillonnage de l'espace des paramètres $\mathcal{D}$. Le premier jeu de paramètres $\boldsymbol{\mu}_1$ est tiré aléatoirement dans Ξ_{train}. En supposant $\xi_1\neq 0$ on peut définir

$$S_1=\Big\{\boldsymbol{\mu}_1\Big\},\ \xi_1\equiv\zeta\Big(\boldsymbol{\mu}_1;\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu}_1)\Big)\text{ et }W_1^{\zeta}=span\Big\{\xi_1\Big\}. \tag{3.6}$$

Pour $M\geqslant 2$, $\boldsymbol{\mu}_M$ est déterminé à l'aide d'un algorithme de type glouton et on en déduit la fonction de base ξ_M associée :

$$\boldsymbol{\mu}_M=\arg\max_{\boldsymbol{\mu}\in\Xi_{train}}\inf_{z\in W_{M-1}^{\zeta}}\Big|\Big|\zeta\big(\boldsymbol{\mu};.;.\big)-z\Big|\Big|_{L^{\infty}(\Omega)}\text{ et }\xi_M=\zeta\Big(\boldsymbol{\mu}_M;\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu}_M)\Big). \tag{3.7}$$

À partir de là, nous pouvons finaliser le processus en enrichissant l'échantillon S_M ainsi que l'espace W_M^{ζ} de la manière suivante :

$$S_M=S_{M-1}\cup\big\{\boldsymbol{\mu}_M\big\}\text{ et }W_M^{\zeta}=W_{M-1}^{\zeta}\oplus span\big\{\xi_M\big\}. \tag{3.8}$$

Les coefficients β^m, $m = 1, ..., M$, qui apparaissent dans la combinaison linéaire (3.5) sont déterminés grâce aux points d'interpolation $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_M \in \Omega$ de sorte à avoir

$$\sum_{m=1}^{M} \beta^m\Big(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big)\ \hat{\zeta}^m(\boldsymbol{t}_i) = \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_i; u_{\mathcal{N}}(\boldsymbol{\mu})\Big), \forall\ \boldsymbol{t}_i,\ 1 \leq i \leq M. \tag{3.9}$$

Le premier point d'interpolation $\boldsymbol{t}_1$ est sélectionné de manière à maximiser la première fonction de base ξ_1, et $\hat{\zeta}^1$ est la normalisation de ξ_1, nous avons donc

$$\boldsymbol{t}_1 = \arg\sup_{\boldsymbol{x}\in\Omega} \big|\xi_1(\boldsymbol{x})\big|,\ \hat{\zeta}^1 = \frac{\xi_1(\boldsymbol{x})}{\xi_1(\boldsymbol{t}_1)} \text{ et } B^1_{11} = \hat{\zeta}^1(\boldsymbol{t}_1) = 1. \tag{3.10}$$

À présent pour $M \geqslant 2$, nous cherchons le vecteur $\boldsymbol{\sigma}^{M-1} = \big(\sigma_i^{M-1}\big)$ avec $1 \leq i \leq M-1$, obtenu grâce aux points d'interpolation

$$\sum_{j=1}^{M-1} \sigma_j^{M-1} \hat{\zeta}^j(\boldsymbol{t}_i) = \xi_M(\boldsymbol{t}_i),\ 1 \leq i \leq M-1, \tag{3.11}$$

ensuite nous pouvons évaluer le résidu

$$r_M(\boldsymbol{x}) = \xi_M(\boldsymbol{x}) - \sum_{j=1}^{M-1} \sigma_j^{M-1} \hat{\zeta}^j(\boldsymbol{x}). \tag{3.12}$$

Ce résidu nous permet de calculer le prochain point d'interpolation t_M ainsi que la fonction de base $\hat{\zeta}^M(x)$ pour $M \geqslant 2$. À cela s'ajoute la matrice d'interpolation B^M que l'on peut également remplir via

$$\boldsymbol{t}_M = \arg\sup_{\boldsymbol{x}\in\Omega} \big|r_M(\boldsymbol{x})\big|,\ \hat{\zeta}^M(\boldsymbol{x}) = \frac{r_M(\boldsymbol{x})}{r_M(\boldsymbol{t}_M)} \text{ et } B^M_{ij} = \hat{\zeta}^j(\boldsymbol{t}_i),\ 1 \leq i, j \leq M. \tag{3.13}$$

Une fois que tous les points d'interpolation t_M et toutes les fonctions de base $\hat{\zeta}^M$ sont calculées durant la phase *hors-ligne*, le calcul de l'approximation $\zeta_M\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big)$ pour un $\boldsymbol{\mu}$ donné durant la phase *en-ligne* consiste à déterminer les coefficients $\beta^m\big(\boldsymbol{\mu}, u_{\mathcal{N}}(\boldsymbol{\mu})\big)$ en résolvant le système

$$\sum_{j=1}^{M} B^M_{ij} \beta^j\Big(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) = \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_i; u_{\mathcal{N}}(\boldsymbol{\mu})\Big),\ 1 \leq i \leq M. \tag{3.14}$$

Autrement dit, les coefficients de l'expansion EIM sont solutions de

$$\begin{pmatrix} \hat{\zeta}^1(\boldsymbol{t}_1) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \hat{\zeta}^1(\boldsymbol{t}_{M-1}) & \cdots & \hat{\zeta}^{M-1}(\boldsymbol{t}_{M-1}) & 0 \\ \hat{\zeta}^1(\boldsymbol{t}_M) & \cdots & \cdots & \hat{\zeta}^M(\boldsymbol{t}_M) \end{pmatrix} \begin{pmatrix} \beta^1 \\ \vdots \\ \beta^{M-1} \\ \beta^M \end{pmatrix} = \begin{pmatrix} \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_1; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) \\ \vdots \\ \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_{M-1}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) \\ \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_M; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) \end{pmatrix}. \tag{3.15}$$

Remarque 5. *Dans le cas général, la fonction ζ dépend non seulement de $\boldsymbol{x}$ mais également de la solution $u_{\mathcal{N}}(\boldsymbol{\mu})$ du problème. Or, durant la phase en-ligne, nous avons besoin d'évaluer rapidement la fonction ζ aux points d'interolations (voir 3.15). Par conséquent, cela implique d'avoir une évaluation rapide de la solution $u_{\mathcal{N}}(\boldsymbol{\mu})$ qui est en réalité remplacée par son approximation RB $u_N(\boldsymbol{\mu})$. Notons qu'il est également nécessaire de précalculer les fonctions de bases associées à $u_N(\boldsymbol{\mu})$ aux points d'interpolation $(\boldsymbol{t}_i)$, $1 \leq i \leq M$.*

Notons que la méthode EIM peut être vue comme une méthode de séparation de variables, d'un côté les variables qui dépendent de l'espace et de l'autre celles qui dépendent des paramètres.

1.2 Estimation d'erreur

Intéressons-nous maintenant au calcul de l'estimation de l'erreur $e(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu}))$ – a posteriori – commise via l'utilisation de l'expansion EIM. Cette erreur est définie par

$$e\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big) = \big|\zeta\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big) - \zeta_M\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big)\big|. \tag{3.16}$$

Notons que contrairement à la méthode CRBM, l'estimation d'erreur n'intervient pas dans la construction des fonctions de base. Soit M_{max} le nombre maximal de termes de l'expansion EIM. Posons

$$\epsilon_M(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})) = \hat{\epsilon}_M(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu}))\ \hat{\zeta}^{M+1}(\boldsymbol{x}),\ \text{avec } M \leq M_{max} - 1, \tag{3.17}$$

et

$$\hat{\epsilon}_M(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu})) = \big|\zeta\big(\boldsymbol{\mu}; \boldsymbol{t}_{M+1}; u_{\mathcal{N}}(\boldsymbol{\mu})\big) - \zeta_M\big(\boldsymbol{\mu}; \boldsymbol{t}_{M+1}; u_{\mathcal{N}}(\boldsymbol{\mu})\big)\big|. \tag{3.18}$$

Remarque 6. *Dans [56] il est noté que d'une manière générale on a $\epsilon_M(\boldsymbol{\mu}; \boldsymbol{u}(\boldsymbol{\mu})) \geq \hat{\epsilon}_M(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu}))$ puisque $\epsilon_M(\boldsymbol{\mu}; \boldsymbol{u}(\boldsymbol{\mu})) = \big|\big|\zeta\big(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\big) - \zeta_M\big(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\big)\big|\big|_{L^\infty(\Omega)}$ et que pour tout $\boldsymbol{x} \in \Omega$ on peut écrire $\epsilon_M(\boldsymbol{\mu}; \boldsymbol{u}(\boldsymbol{\mu})) \geq \big|\zeta\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big) - \zeta_M\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big)\big|$, donc cette inégalité est également vérifiée pour $\boldsymbol{x} = \boldsymbol{t}_{M+1}$.*

L'estimateur d'erreur $\hat{\epsilon}_M(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu}))$ défini par (3.18) est une borne inférieure pour l'erreur $e(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu}))$. De plus, dans le cas où $\zeta\big(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\big) \in W^{\zeta}_{M+1}$, on peut alors écrire

— $e\big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\big) = \pm\epsilon_M(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu}))$;
— $\big|\big|\zeta\big(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\big) - \zeta_M\big(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\big)\big|\big|_{L^\infty(\Omega)} = \hat{\epsilon}_M(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu}))$.

L'estimateur d'erreur défini en (3.18) ne permet d'obtenir une borne supérieure pour l'erreur, néanmoins il s'avère suffisamment précis dans de nombreux cas. Il est à noter que [47] présente la construction d'un estimateur a posteriori rigoureux.

2 Problèmes elliptiques linéaires non-affines

Nous nous proposons ici d'étudier comment la méthode EIM peut être intégrée dans la méthode CRBM [16].

2.1 Énoncé du problème général

Commençons par rappeler la définition de l'espace $E^{\zeta} = \mathcal{D} \times \mathbb{R}^d$. Comme pour les problèmes linéaires affines, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous nous intéressons à l'évaluation d'une sortie $s_{\mathcal{N}}(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$:

$$s_{\mathcal{N}}(\boldsymbol{\mu}) = \ell\Big(u_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big), \tag{3.19}$$

pour un opérateur linéaire $\ell(.; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})) : X_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta} \to \mathbb{R}$ approprié qui dépend d'une fonction ζ non-affine en $\boldsymbol{\mu}$. La formulation variationnelle de l'EDP consiste à chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big) = f\Big(v; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big), \ \forall v \in X_{\mathcal{N}}, \tag{3.20}$$

où $a\big(., .; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta} \to \mathbb{R}$ et $f\big(.; \boldsymbol{\mu}; \zeta(\boldsymbol{\mu}; \boldsymbol{x})\big) : X_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta} \to \mathbb{R}$ sont respectivement les formes bilinéaire et linéaire associées à l'EDP. Remarquons qu'une seule fonction non-affine est impliquée dans le problème (3.20). Afin de généraliser le problème (3.20), considérons que N_{na} fonctions non-affines sont utilisées. Dans la notation N_{na}, l'indice na signifie "non-affine". Notons $\zeta^q(\boldsymbol{\mu}; \boldsymbol{x}) : \mathcal{D} \times \mathbb{R}^d \to \mathbb{R}$, $1 \leq q \leq N_{na}$, la $q^{\text{ème}}$ fonction non-affine dont une approximation via la méthode EIM s'écrit, pour un entier strictement positif M^q donné, $\zeta^q\big(\boldsymbol{\mu}; \boldsymbol{x}\big) \approx \zeta^q_{M^q}\big(\boldsymbol{\mu}; \boldsymbol{x}\big) = \sum_{m=1}^{M^q} \beta^{qm}\big(\boldsymbol{\mu}\big)\ \hat{\zeta}^{qm}\big(\boldsymbol{x}\big)$. Nous introduisons alors les espaces $E^{\zeta^q} = \mathcal{D} \times \mathbb{R}^d$, pour $q = 1, ..., N_{na}$. La formulation variationnelle de l'EDP consiste alors à chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que, pour tout $v \in X_{\mathcal{N}}$,

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \zeta^1(\boldsymbol{\mu}; \boldsymbol{x}); ...; \zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})\Big) = f\Big(v; \boldsymbol{\mu}; \zeta^1(\boldsymbol{\mu}; \boldsymbol{x}); ...; \zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})\Big), \tag{3.21}$$

avec $a\big(., .; \boldsymbol{\mu}; \zeta^1(\boldsymbol{\mu}; \boldsymbol{x}); ...; \zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta^1} \times ... \times E^{\zeta^{N_{na}}} \to \mathbb{R}$ et $f\big(.; \boldsymbol{\mu}; \zeta^1(\boldsymbol{\mu}; \boldsymbol{x}); ...; \zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})\big) : X_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta^1} \times ... \times E^{\zeta^{N_{na}}} \to \mathbb{R}$. Afin d'alléger l'écriture, lorsqu'une forme (bi-)linéaire dépend de N_{na} fonctions non-affines $\zeta^1(\boldsymbol{\mu}; \boldsymbol{x})$,..., $\zeta^q(\boldsymbol{\mu}; \boldsymbol{x})$ ou de leur approximation $\zeta^1_{M^1}(\boldsymbol{\mu}; \boldsymbol{x})$,..., $\zeta^q_{M^q}(\boldsymbol{\mu}; \boldsymbol{x})$, alors nous faisons le choix de représenter cette dépendance en ne mentionnant uniquement que la fonction non-affine qui possède l'indice le plus élevé et nous la soulignons. Ainsi, avec cette notation, la formulation variationnelle (3.21) peut se réécrire comme : chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})}\Big) = f\Big(v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})}\Big), \ \forall v \in X_{\mathcal{N}}. \tag{3.22}$$

Posons $E^{\underline{\zeta^{Nna}}} = E^{\zeta^1} \times ... \times E^{\zeta^{N_{na}}}$. Nous supposons ici que $a : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{Nna}}} \to \mathbb{R}$ est une forme bilinéaire symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((w, z)\big)_{a,\boldsymbol{\mu}} \equiv a\big(w, z; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})}\big), \ \forall w, z \in X_{\mathcal{N}}. \tag{3.23}$$

Notons que la norme associée à ce produit scalaire est

$$|||w|||_{a,\boldsymbol{\mu}} \equiv \sqrt{\big((w, w)\big)_{a,\boldsymbol{\mu}}}, \ \forall w \in X_{\mathcal{N}}. \tag{3.24}$$

De plus, nous supposons que a est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ il existe une constante de continuité $\gamma^{a,\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\gamma^{a,\mathcal{N}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{a(v, v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})})}{|||v|||^2_{a,\boldsymbol{\mu}_{ref}}} < \infty, \tag{3.25}$$

ainsi qu'une constante de coercivité $\alpha^{a,\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\alpha^{a,\mathcal{N}}(\boldsymbol{\mu}) \equiv \inf_{v \in X_{\mathcal{N}}} \frac{a(v, v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})})}{|||v|||^2_{a,\boldsymbol{\mu}_{ref}}} > 0. \tag{3.26}$$

Rappelons qu'un ingrédient important de la méthode CRBM est le développement d'une stratégie *hors-ligne/ en-ligne* efficace. Afin de mettre en oeuvre cette stratégie de manière efficace, a, f et ℓ doivent pouvoir s'exprimer en faisant apparaître une dépendance affine en paramètres. Or justement, nous venons de voir que dans le cas de problèmes non-affines, a, f and ℓ ne dépendent pas des paramètres de manière affine. Nous allons donc approcher, via la méthode EIM, les termes pour lequels il n'est pas possible d'écrire une décomposition affine directement par inspection. Autrement dit nous supposons qu'il existe des entiers positifs Q_a, Q_f et Q_ℓ, avec $1 \leq Q_a, Q_f, Q_\ell \leq N_{na}$, de sorte que la méthode EIM détermine $\big(M_a^q\big)_{q=1,\dots,Q_a}$, $\big(M_f^q\big)_{q=1,\dots,Q_f}$ et $\big(M_\ell^q\big)_{q=1,\dots,Q_\ell}$ tels que, pour $n_a = \sum_{q=1}^{Q_a} M_a^q$, $n_f = \sum_{q=1}^{Q_f} M_f^q$ et $n_\ell = \sum_{q=1}^{Q_\ell} M_\ell^q$, ainsi que pour tout $\boldsymbol{\mu} \in \mathcal{D}$, les formes bilinéaire et linéaires $a(.,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$, $f(.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$ et $\ell(.,\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$ puissent être approchées de la manière suivante :

$$\begin{cases} a\big(u, v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{a}\big(u, v; \boldsymbol{\mu}; \underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) a^{qm}(u, v; \hat{\zeta}^{qm}(\boldsymbol{x})), \\ f\big(v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{f}\big(v; \boldsymbol{\mu}; \underline{\zeta^{n_f}_{M^{n_f}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_f} \sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}) f^{qm}\big(v; \hat{\zeta}^{qm}(\boldsymbol{x})\big), \\ \ell\big(v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{\ell}\big(v; \boldsymbol{\mu}; \underline{\zeta^{n_\ell}_{M^{n_\ell}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu}) \ell^{qm}\big(v; \hat{\zeta}^{qm}(\boldsymbol{x})\big), \end{cases} \tag{3.27}$$

pour tout $u \in X_{\mathcal{N}}$ et pour tout $v \in X_{\mathcal{N}}$, où $\beta_a^{qm} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_a$, $1 \leq m \leq M_a^q$, $\beta_f^{qm} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_f$, $1 \leq m \leq M_f^q$ et $\beta_\ell^{qm} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_\ell$, $1 \leq m \leq M_\ell^q$ sont déterminées par la méthode EIM. Traitant des problèmes linéaires, les coefficients de l'expansion EIM ne dépendent que de $\boldsymbol{\mu}$ et non de la solution $u_{\mathcal{N}}(\boldsymbol{\mu})$. Posons $E^{\zeta^q}_{M^q} = \mathcal{D} \times \mathbb{R}^d$, pour $q = 1, ..., N_{na}$, ainsi que $E^{\underline{\zeta^{n_a}_{M^{n_a}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_a}_{M^{n_a}}}$, $E^{\underline{\zeta^{n_f}_{M^{n_f}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_f}_{M^{n_f}}}$, $E^{\underline{\zeta^{n_\ell}_{M^{n_\ell}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_\ell}_{M^{n_\ell}}}$. Notons que nous avons $\tilde{a}\big(.,.;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_a}_{M^{n_a}}}} \to \mathbb{R}$, $\tilde{f}\big(.;\boldsymbol{\mu};\underline{\zeta^{n_f}_{M^{n_f}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_f}_{M^{n_f}}}} \to \mathbb{R}$ et $\tilde{\ell}\big(.;\boldsymbol{\mu};\underline{\zeta^{n_\ell}_{M^{n_\ell}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_\ell}_{M^{n_\ell}}}} \to \mathbb{R}$.

En nous servant des approximations décrites dans (3.27), pour un $\boldsymbol{\mu} \in \mathcal{D}$ fixé, (3.22) revient à chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que

$$\tilde{a}\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\Big) = \tilde{f}\Big(v; \boldsymbol{\mu}; \underline{\zeta^{n_f}_{M^{n_f}}(\boldsymbol{\mu};\boldsymbol{x})}\Big), \ \forall v \in X_{\mathcal{N}}. \tag{3.28}$$

Remarque 7. *La méthode EIM nous permet d'approcher l'opérateur a par $\tilde{a}$ et devrait garantir la coercivité de $\tilde{a}$. Toutefois rien ne la garantie. Il faudra donc vérifier a posteriori que l'opérateur $\tilde{a}$ est bien coercif.*

Nous supposons ici que $\tilde{a}$ est une forme bilinéaire symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((.,.)\big)_{\tilde{a},\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((w,z)\big)_{\tilde{a},\boldsymbol{\mu}} \equiv \tilde{a}\big(w,z;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big),\ \forall w,z \in X_{\mathcal{N}}. \tag{3.29}$$

Notons que la norme associée à ce produit scalaire est

$$|||w|||_{\tilde{a},\boldsymbol{\mu}} \equiv \sqrt{\big((w,w)\big)_{\tilde{a},\boldsymbol{\mu}}},\ \forall w \in X_{\mathcal{N}}. \tag{3.30}$$

De plus, nous supposons que $\tilde{a}$ est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ il existe une constante de continuité $\gamma^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\gamma^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{\tilde{a}(v,v;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{na}}(\boldsymbol{\mu};\boldsymbol{x})})}{|||v|||^2_{\tilde{a},\boldsymbol{\mu}_{ref}}} < \infty, \tag{3.31}$$

ainsi qu'une constante de coercivité $\alpha^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\alpha^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu}) \equiv \inf_{v \in X_{\mathcal{N}}} \frac{\tilde{a}(v,v;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{na}}(\boldsymbol{\mu};\boldsymbol{x})})}{|||v|||^2_{\tilde{a},\boldsymbol{\mu}_{ref}}} > 0. \tag{3.32}$$

2.2 Méthode des bases réduites

De la même manière que pour les problèmes elliptiques linéaires affines, nous introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{pr}}$ tels que $W_{1_{pr}} \subset W_{2_{pr}} \subset ... \subset W_{N_{max_{pr}}} \subset X_{\mathcal{N}}$, avec N_{max} un entier positif. En reprenant les notations du chapitre 2, $S_N = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N\}$ est un ensemble de N jeux de paramètres sélectionnés à l'aide de l'algorithme glouton et S^u_N est l'ensemble qui va contenir, pour tout $\boldsymbol{\mu}$ dans S_N, la solution de (3.28). Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble S^u_N pour avoir des fonctions de bases orthonormalisées ξ^{pr}_n, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{pr}}$ est défini par

$$W_{N_{pr}} = span\Big\{\xi^{pr}_n,\ 1 \leq n \leq N\Big\}. \tag{3.33}$$

Quant à la solution réduite $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ elle s'écrit

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu})\ \xi^{pr}_i. \tag{3.34}$$

Remarque 8. *Ici $u_N(\boldsymbol{\mu})$ est solution de (3.22) qui implique N_{na} fonctions non-affines. Comme indiqué par (3.27), chacune de ces fonctions $non-affines$ est approchée en utilisant la méthode EIM. La précision de $u_N(\boldsymbol{\mu})$ dépend donc du nombre de termes M^1, ..., $M^{N_{na}}$ de chaque approximation, mais pour des raisons de simplicité, uniquement le nombre d'éléments N de la base réduite est utilisé dans la notation de la solution $u_N(\boldsymbol{\mu})$.*

Nous allons maintenant utiliser les approximations décrites dans (3.27) pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{pr}$, $n = 1, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $u_N(\boldsymbol{\mu})$ est solution de

$$\sum_{i=1}^{N} \left(\sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) a^{qm}\Big(\xi_n^{pr}, \xi_i^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \right) u_{N_i}(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_f} \sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}) f^{qm}\Big(\xi_n^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \right), \tag{3.35}$$

qui peut être également écrit sous la forme matricielle :

$$\left(\sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) A_N^{qm} \right) u_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_f} \sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}) F_N^{qm} \right), \tag{3.36}$$

avec

$$\Big(u_N(\boldsymbol{\mu})\Big)_i = u_{N_i}(\boldsymbol{\mu}) \ , \ \Big(A_N^{qm}\Big)_{in} = a^{qm}\Big(\xi_n^{pr}, \xi_i^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \text{ et } \Big(F_N^{qm}\Big)_n = f^{qm}\Big(\xi_n^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big). \tag{3.37}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}\Big(\boldsymbol{\mu}\Big) \ell^{qm}\Big(u_N(\boldsymbol{\mu}); \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \right), \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.38}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu}) L_N^{qm\ T} \right) u_N(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.39}$$

avec $(L_N^{qm})_i = \ell^{qm}(\xi_i^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x}))$. La décomposition *hors-ligne/en-ligne* est maintenant évidente. Les fonctions de base ξ_i^{pr}, $1 \leq i \leq N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible la construction des matrices $A_N^{qm} \in \mathbb{R}^{N \times N}$, $1 \leq q \leq Q_a$, $1 \leq m \leq M_a^q$ et des vecteurs $F_N^{qm} \in \mathbb{R}^N$, $1 \leq q \leq Q_f$, $1 \leq m \leq M_f^q$ et $L_N^{qm} \in \mathbb{R}^N$, $1 \leq q \leq Q_\ell$, $1 \leq m \leq M_\ell^q$. À l'étape *en-ligne*, pour chaque vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, on assemble la matrice $A_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^q(\boldsymbol{\mu}) A_N^{qm}$, ainsi que les vecteurs $F_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \sum_{m=1}^{M_f^a} \beta_f^q(\boldsymbol{\mu}) F_N^{qm}$ et $L_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^a} \beta_\ell^q(\boldsymbol{\mu}) L_N^{qm}$. Enfin, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, nous résolvons le système réduit

$$A_N(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}) = F_N(\boldsymbol{\mu}), \tag{3.40}$$

et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = L_N^T(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.41}$$

Jusqu'ici nous avons exposé une approche uniquement *primale* pour évaluer la sortie. De la même manière que dans le chapitre 2 nous introduisons à présent une approche *primale-duale*. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, le problème dual associé à la sortie du modèle consiste à chercher $\Psi_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que, pour $1 \leq n_a \leq Q_a$ et $1 \leq n_\ell \leq Q_\ell$,

$$\tilde{a}\Big(v, \Psi_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu}; \boldsymbol{x})}\Big) = -\tilde{\ell}\Big(v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu}; \boldsymbol{x})}\Big), \ \forall v \in X_{\mathcal{N}}. \tag{3.42}$$

Nous introduisons également une séquence d'espaces d'approximation emboîtés $W_{N_{du}}$, $1 \leq N \leq N_{max}$. S_N^Ψ est l'ensemble qui va contenir, pour tout $\boldsymbol{\mu}$ dans S_N, la solution de (3.42). Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble S_N^Ψ pour avoir des fonctions de bases orthonormalisées ξ_n^{du}, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{du}}$ est défini par

$$W_{N_{du}} = span\Big\{\xi_n^{du},\ 1 \leq n \leq N\Big\}. \tag{3.43}$$

Quant à la solution réduite $\Psi_N(\boldsymbol{\mu}) \in W_{N_{du}}$ elle s'écrit

$$\Psi_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} \Psi_{Ni}(\boldsymbol{\mu})\ \xi_i^{du}. \tag{3.44}$$

L'évaluation de la sortie, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, est donnée par

$$s_N(\boldsymbol{\mu}) = \tilde{\ell}\Big(u_N(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x})}\Big) - \tilde{r}_{pr}\Big(\Psi_N(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu};\boldsymbol{x})}\Big), \tag{3.45}$$

avec $1 \leq n_r \leq N_{na}$, et

$$\tilde{r}_{pr}\Big(v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu};\boldsymbol{x})}\Big) = \tilde{f}\Big(v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_f}}^{n_f}(\boldsymbol{\mu};\boldsymbol{x})}\Big) - \tilde{a}\Big(u_N(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu};\boldsymbol{x})}\Big). \tag{3.46}$$

Nous allons maintenant utiliser les approximations décrites dans (3.27) pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{du},\ n = 1, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $\Psi_N(\boldsymbol{\mu})$ est solution de

$$\begin{aligned} -\sum_{i=1}^{N} \left(\sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) a^{qm}\left(\xi_n^{du}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right) \right)\ \Psi_{N_i}(\boldsymbol{\mu}) \\ = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu}) \ell^{qm}\left(\xi_n^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right) \right), \end{aligned} \tag{3.47}$$

qui peut être également écrit sous la forme matricielle :

$$-\left(\sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) A_N^{qm} \right) \Psi_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu}) L_N^{qm} \right), \tag{3.48}$$

avec

$$\Big(\Psi_N(\boldsymbol{\mu})\Big)_i = \Psi_{N_i}(\boldsymbol{\mu}), \quad \text{et} \quad \Big(L_N^{qm}\Big)_n = \ell^{qm}\Big(\xi_n^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big). \tag{3.49}$$

et

$$\Big(A_N^{qm}\Big)_{in} = a^{qm}\Big(\xi_n^{du}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big). \tag{3.50}$$

L'apparition du terme de correction $\tilde{r}_{pr}\big(\Psi_N(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu};\boldsymbol{x})}\big)$ dans (3.45) nécessite de manipuler les matrices $A_N^{prdu,qm} \in \mathbb{R}^{N\times N}$, $1 \leq q \leq Q_a$ et $1 \leq m \leq M_a^q$, construites à partir

des éléments des bases primale et duale. D'un point de vue matriciel, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, la sortie s'écrit

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_a^q} \beta_\ell^{qm}(\boldsymbol{\mu}) L_N^{qm\ T} \right)\ u_N(\boldsymbol{\mu})\ -\ R_{pr}^{\Psi_N(\boldsymbol{\mu})}, \tag{3.51}$$

avec

$$R_{pr}^{\Psi_N(\boldsymbol{\mu})} = \left(\sum_{q=1}^{Q_f} \sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}) F_N^{du,qm} \right) \Psi_N(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu})^T \left(\sum_{q=1}^{Q_a} \sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) A_N^{prdu,qm} \right) u_N(\boldsymbol{\mu}), \tag{3.52}$$

ainsi que

$$\left(A_N^{prdu,qm} \right)_{in} = a^{qm}\left(\xi_n^{pr}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x}) \right) \ \text{ et } \ \left(F_N^{du,qm} \right)_n = f^{qm}\left(\xi_n^{du}; \hat{\zeta}^{qm}(\boldsymbol{x}) \right). \tag{3.53}$$

Durant l'étape *hors-ligne* il est nécessaire de faire $2N$ résolutions FE (coûteuses) des problèmes primal et dual et $\mathcal{O}(Q_a M_a^q N^2 \mathcal{N})$ produits scalaires pour les étapes de projection sur la base réduite. En revanche la complexité de l'étape *en-ligne* ne dépend plus de la (grande) dimension $\mathcal{N}$ puisqu'elle implique $\mathcal{O}(Q_a M_a^q N^2)$ multiplications pour assembler le système réduit, primal ou dual, et $\mathcal{O}(N^3)$ pour le résoudre. Ensuite il faut compter $\mathcal{O}(N)$ produits scalaires pour avoir la sortie du modèle en utilisant (3.39) et $\mathcal{O}(N^3)$ lorsque le terme de correction est ajouté (3.51).

2.3 Estimation d'erreur a posteriori

À l'aide de la formule (3.45), pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous pouvons construire rapidement une approximation de la sortie du modèle. Ici nous introduisons des estimateurs d'erreur a posteriori pour les problèmes elliptiques linéaires non-affines. La différence avec les estimateurs d'erreur présentés dans la section 3 du chapitre 2 réside dans le fait que nous devons estimer l'erreur commise par l'approximation des termes non-affines via la méthode EIM. Ces estimateurs vont nous permettre de savoir, rapidement et de manière fiable, si l'approximation de la solution primale, duale, et de la sortie du modèle sont suffisament précises.

Commençons par introduire la norme duale des résidus primal $\epsilon_{N_{pr}}(\boldsymbol{\mu})$ et dual $\epsilon_{N_{du}}(\boldsymbol{\mu})$:

$$\epsilon_{N_{pr}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_\mathcal{N}} \frac{\tilde{r}_{pr}(v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})})}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \left|||\hat{e}_{pr}(\boldsymbol{\mu})\right|||_{a,\boldsymbol{\mu}_{ref}}, \tag{3.54}$$

$$\epsilon_{N_{du}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_\mathcal{N}} \frac{\tilde{r}_{du}(v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})})}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \left|||\hat{e}_{du}(\boldsymbol{\mu})\right|||_{a,\boldsymbol{\mu}_{ref}}. \tag{3.55}$$

Notons que pour définir la norme duale, la représentation de Riesz des résidus primal et dual, respectivement $\hat{e}_{pr}(\boldsymbol{\mu})$ et $\hat{e}_{du}(\boldsymbol{\mu})$, a été introduit dans (3.54) et (3.55). Définissons également la norme duale du résidu primal $\epsilon_{N_{pr}^{eim}}(\boldsymbol{\mu})$ – respectivement dual $\epsilon_{N_{du}^{eim}}(\boldsymbol{\mu})$ – en nous limitant aux termes comportant des fonctions non-affines.

$$\epsilon_{N_{pr}^{eim}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_\mathcal{N}} \frac{\tilde{r}_{pr}(v; \boldsymbol{\mu}; \underline{\zeta^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})} - \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})})}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \left|||\hat{e}_{pr}^{eim}(\boldsymbol{\mu})\right|||_{a,\boldsymbol{\mu}_{ref}}, \tag{3.56}$$

$$\epsilon_{N_{du}^{eim}}(\boldsymbol{\mu}) \equiv \sup_{v\in X_{\mathcal{N}}} \frac{\tilde{r}_{du}(v;\boldsymbol{\mu};\underline{\zeta^{n_r}(\boldsymbol{\mu};\boldsymbol{x})} - \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu};\boldsymbol{x})})}{|||v|||_{a,\boldsymbol{\mu}_{ref}}} = \left|||\hat{e}_{du}^{eim}(\boldsymbol{\mu})\right|||_{a,\boldsymbol{\mu}_{ref}}. \tag{3.57}$$

À présent introduisons l'estimation d'erreur a posteriori sur l'approximation de la sortie du modèle. Pour tout N tel que $1 \leq N \leq N_{max}$ et pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\left|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})\right| \approx \Delta_N^s(\boldsymbol{\mu}) \equiv \ \alpha_{LB}^a(\boldsymbol{\mu})\ \Delta_N^{pr}(\boldsymbol{\mu})\ \Delta_N^{du}(\boldsymbol{\mu}), \tag{3.58}$$

où $\Delta_N^{pr}(\boldsymbol{\mu})$ et $\Delta_N^{du}(\boldsymbol{\mu})$ sont les estimations d'erreur a posteriori respectivement pour les solutions, primale et duale, définies par

$$\Delta_N^{pr}(\boldsymbol{\mu}) = \frac{\sqrt{\epsilon_{N_{pr}}(\boldsymbol{\mu})^2} + \sqrt{\epsilon_{N_{pr}^{eim}}(\boldsymbol{\mu})^2}}{\alpha_{LB}^a(\boldsymbol{\mu})} \ \text{ et } \ \Delta_N^{du}(\boldsymbol{\mu}) = \frac{\sqrt{\epsilon_{N_{du}}(\boldsymbol{\mu})^2} + \sqrt{\epsilon_{N_{du}^{eim}}(\boldsymbol{\mu})^2}}{\alpha_{LB}^a(\boldsymbol{\mu})}. \tag{3.59}$$

Nous avons

$$\left|||u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})|||\right|_{a,\boldsymbol{\mu}_{ref}} \approx \Delta_N^{pr}(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.60}$$

$$\left|||\Psi_{\mathcal{N}}(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu})|||\right|_{a,\boldsymbol{\mu}_{ref}} \approx \Delta_N^{du}(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.61}$$

Remarque 9. *Les estimateurs d'erreur proposés via (3.58) et (3.59) sont des estimateurs rigoureux uniquement dans le cas où $\zeta_{M^i}^i \in W_{M^i}^{\zeta^i}$, pour $i = 1, ..., N_{na}$.*

Stratégie *hors-ligne/en-ligne*

Nous exposons ici la stratégie *hors-ligne/en-ligne* à mettre en oeuvre afin d'évaluer les estimations d'erreur définies dans (3.58) et (3.59) de manière efficace. En nous appuyant sur les approximations décrites dans (3.27), nous pouvons exprimer le résidu primal, pour tout $v \in X_{\mathcal{N}}$, de la manière suivante :

$$\begin{aligned}\tilde{r}_{pr}(v;\boldsymbol{\mu};\underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu};\boldsymbol{x})}) \ &= \ \sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}) f^{qm}(v;\hat{\zeta}^{qm}(\boldsymbol{x})) \\ &\quad - \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{i=1}^{N} \beta_a^{qm}(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) a^{qm}(\xi_i^{pr}, v;\hat{\zeta}^{qm}(\boldsymbol{x})),\end{aligned} \tag{3.62}$$

Le représentant de Riesz $\hat{e}_{pr}(\boldsymbol{\mu})$ vérifie, pour tout $v \in X_{\mathcal{N}}$,

$$\Big(\Big(\hat{e}_{pr}(\boldsymbol{\mu}), v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = \tilde{r}_{pr}(v;\boldsymbol{\mu};\underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu};\boldsymbol{x})}). \tag{3.63}$$

Ensuite, en utilisant le principe de superposition linéaire nous pouvons écrire

$$\hat{e}_{pr}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}) \Gamma_{N_{pr}}^{qm} + \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{i=1}^{N} \beta_a^{qm}(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) \Upsilon_{N_{pr}}^{qmi}, \tag{3.64}$$

avec, pour tout $v \in X_{\mathcal{N}}$,

$$\Big(\Big(\Gamma_{N_{pr}}^{qm}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = f^{qm}\big(v;\hat{\zeta}^{qm}(\boldsymbol{x})\big),\ 1 \leq q \leq Q_f,\ 1 \leq m \leq M_f^q, \tag{3.65}$$

$$\Big(\Big(\Upsilon^{qmi}_{N_{pr}}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -a^{qm}\big(\xi^{pr}_i, v; \hat{\zeta}^{qm}(\boldsymbol{x})\big),\ 1 \le q \le Q_a,\ 1 \le m \le M^q_a,\ 1 \le i \le N. \quad (3.66)$$

Par conséquent nous avons

$$\epsilon_{N_{pr}}(\boldsymbol{\mu})^2 = C^{ff}_{N_{pr}}(\boldsymbol{\mu}) + 2\sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu}) C^{fa}_{N_{pr}i}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{i'=1}^{N} u_{Ni}(\boldsymbol{\mu}) u_{Ni'}(\boldsymbol{\mu}) C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu}), \quad (3.67)$$

avec pour $1 \le i, i' \le N$:

$$C^{ff}_{N_{pr}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\sum_{m=1}^{M^q_f}\sum_{q'=1}^{Q_f}\sum_{m'=1}^{M^q_f} \beta^{qm}_f(\boldsymbol{\mu})\ \beta^{q'm'}_f(\boldsymbol{\mu})\ \Phi^{qmq'm'}_{N_{pr},ff}, \quad (3.68)$$

$$C^{fa}_{N_{pr}i}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{m=1}^{M^q_a}\sum_{q'=1}^{Q_f}\sum_{m'=1}^{M^q_f} \beta^{qm}_a(\boldsymbol{\mu})\ \beta^{q'm'}_f(\boldsymbol{\mu})\ \Phi^{qmiq'm'}_{N_{pr},fa}, \quad (3.69)$$

$$C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{m=1}^{M^q_a}\sum_{q'=1}^{Q_a}\sum_{m'=1}^{M^q_a} \beta^{qm}_a(\boldsymbol{\mu})\ \beta^{q'm'}_a(\boldsymbol{\mu})\ \Phi^{qmiq'm'i'}_{N_{pr},aa}. \quad (3.70)$$

où on peut écrire

$$\Phi^{qmq'm'}_{N_{pr},ff} = \big(\big(\Gamma^{qm}_{N_{pr}}, \Gamma^{q'm'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}},\ \Phi^{qmiq'm'}_{N_{pr},fa} = \big(\big(\Upsilon^{qmi}_{N_{pr}}, \Gamma^{q'm'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}, \quad (3.71)$$

$$\Phi^{qmiq'm'i'}_{N_{pr},aa} = \big(\big(\Upsilon^{qmi}_{N_{pr}}, \Upsilon^{q'm'i'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}. \quad (3.72)$$

À présent, condidérons le résidu dual qui s'écrit, pour tout $v \in X_{\mathcal{N}}$,

$$\begin{aligned} \tilde{r}_{du}(v; \boldsymbol{\mu}; \underline{\zeta^{n_r}_{M^{nr}}(\boldsymbol{\mu}; \boldsymbol{x})}) \ &= \ -\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M^q_\ell} \beta^{qm}_\ell(\boldsymbol{\mu}) \ell^{qm}(v; \hat{\zeta}^{qm}(\boldsymbol{x})) \\ &\quad -\sum_{q=1}^{Q_a}\sum_{m=1}^{M^q_a}\sum_{i=1}^{N} \beta^{qm}_a(\boldsymbol{\mu}) \Psi_{N_i}(\boldsymbol{\mu}) a^{qm}(v, \xi^{du}_i; \hat{\zeta}^{qm}(\boldsymbol{x})). \end{aligned} \quad (3.73)$$

Le représentant de Riesz $\hat{e}_{du}(\boldsymbol{\mu})$ vérifie, pour tout $v \in X_{\mathcal{N}}$,

$$\hat{e}_{du}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell}\sum_{m=1}^{M^q_\ell} \beta^{qm}_\ell(\boldsymbol{\mu}) \Gamma^{qm}_{N_{du}} + \sum_{q=1}^{Q_a}\sum_{m=1}^{M^q_a}\sum_{i=1}^{N} \beta^{qm}_a(\boldsymbol{\mu}) \Psi_{N_i}(\boldsymbol{\mu}) \Upsilon^{qmi}_{N_{du}}, \quad (3.74)$$

avec, pour tout $v \in X_{\mathcal{N}}$,

$$\big(\big(\Gamma^{qm}_{N_{du}}, v\big)\big)_{a,\boldsymbol{\mu}_{ref}} = -\ell^{qm}\big(v; \hat{\zeta}^{qm}(\boldsymbol{x})\big),\ 1 \le q \le Q_\ell,\ 1 \le m \le M^q_\ell, \quad (3.75)$$

$$\big(\big(\Upsilon^{qmi}_{N_{du}}, v\big)\big)_{a,\boldsymbol{\mu}_{ref}} = -a^{qm}(v, \xi^{du}_i; \hat{\zeta}^{qm}(\boldsymbol{x})),\ 1 \le q \le Q_a,\ 1 \le m \le M^q_a,\ 1 \le i \le N. \quad (3.76)$$

Nous pouvons donc écrire

$$\epsilon_{N_{du}}(\boldsymbol{\mu})^2 = C^{\ell\ell}_{N_{du}}(\boldsymbol{\mu}) + 2\sum_{i=1}^{N} \Psi_{N_i} C^{\ell a}_{N_{du}i}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{i'=1}^{N} \Psi_{N_i}\Psi_{N_{i'}} C^{aa}_{N_{du}ii'}(\boldsymbol{\mu}), \quad (3.77)$$

avec pour $1 \leq i, i' \leq N$:

$$C^{\ell\ell}_{N_{du}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \sum_{m=1}^{M^q_\ell} \sum_{q'=1}^{Q_\ell} \sum_{m'=1}^{M^q_\ell} \beta^{qm}_\ell(\boldsymbol{\mu})\ \beta^{q'm'}_\ell(\boldsymbol{\mu})\ \Phi^{qmq'm'}_{N_{du}}, \tag{3.78}$$

$$C^{\ell a}_{N_{du}i}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{m=1}^{M^q_a} \sum_{q'=1}^{Q_\ell} \sum_{m'=1}^{M^q_\ell} \beta^{qm}_a(\boldsymbol{\mu})\ \beta^{q'm'}_\ell(\boldsymbol{\mu})\ \Phi^{qmiq'm'}_{N_{du}}, \tag{3.79}$$

$$C^{aa}_{N_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{m=1}^{M^q_a} \sum_{q'=1}^{Q_a} \sum_{m'=1}^{M^q_a} \beta^{qm}_a(\boldsymbol{\mu})\ \beta^{q'm'}_a(\boldsymbol{\mu})\ \Phi^{qmiq'm'i'}_{N_{du}}. \tag{3.80}$$

où on a

$$\Phi^{qmq'm'}_{N_{du},\ell\ell} = \big(\big(\Gamma^{qm}_{N_{du}}, \Gamma^{q'm'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qmiq'm'}_{N_{du},\ell a} = \big(\big(\Upsilon^{qmi}_{N_{du}}, \Gamma^{q'm'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{3.81}$$

$$\Phi^{qmiq'm'i'}_{N_{du},aa} = \big(\big(\Upsilon^{qmi}_{N_{du}}, \Upsilon^{q'm'i'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}. \tag{3.82}$$

Du fait que nous devons approcher les fonctions non-affines en utilisant la méthode EIM, cela engendre des erreurs supplémentaires que nous devons quantifier. Pour cela considérons la $q^{\text{ème}}$ fonction non-affine $\zeta^q(\boldsymbol{\mu};\boldsymbol{x})$ approchée par $\zeta^q_{M^q}(\boldsymbol{\mu};\boldsymbol{x})$, et introduisons les notations suivantes :

$$\epsilon_{M^q}(\boldsymbol{\mu};\boldsymbol{x}) = \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\zeta}^{qM^q+1}(\boldsymbol{x}), \tag{3.83}$$

avec

$$\hat{\epsilon}_{M^q}(\boldsymbol{\mu}) = \big|\zeta^q\big(\boldsymbol{\mu};\boldsymbol{t}_{M^q+1}\big) - \zeta^q_{M^q}\big(\boldsymbol{\mu};\boldsymbol{t}_{M^q+1}\big)\big|. \tag{3.84}$$

Ainsi, l'erreur due à l'utilisation de $\zeta^q_{M^q}(\boldsymbol{\mu};\boldsymbol{x})$ à la place de $\zeta^q(\boldsymbol{\mu};\boldsymbol{x})$ peut être approchée par

$$\big|\zeta^q\big(\boldsymbol{\mu};\boldsymbol{x}\big) - \zeta^q_{M^q}\big(\boldsymbol{\mu};\boldsymbol{x}\big)\big| \approx \epsilon_{M^q}(\boldsymbol{\mu};\boldsymbol{x}). \tag{3.85}$$

En nous servant de (3.85) nous pouvons écrire, pour tout $v \in X_{\mathcal{N}}$,

$$\begin{aligned}
&\tilde{r}_{pr}\big(v;\boldsymbol{\mu};\underline{\zeta^{n_r}(\boldsymbol{\mu};\boldsymbol{x})} - \underline{\zeta^{n_r}_{M^{nr}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \\
&\qquad = \tilde{r}_{pr}\big(v;\boldsymbol{\mu};\zeta^1(\boldsymbol{\mu};\boldsymbol{x}) - \zeta^1_{M^1}(\boldsymbol{\mu};\boldsymbol{x}), ..., \zeta^{nr}(\boldsymbol{\mu};\boldsymbol{x}) - \zeta^{nr}_{M^{nr}}(\boldsymbol{\mu};\boldsymbol{x})\big) \\
&\qquad \approx \tilde{r}_{pr}\big(v;\boldsymbol{\mu};\epsilon_{M^1}(\boldsymbol{\mu};\boldsymbol{x}), ..., \epsilon_{M^{nr}}(\boldsymbol{\mu};\boldsymbol{x})\big).
\end{aligned} \tag{3.86}$$

Le représentant de Riesz $\hat{e}^{eim}_{pr}(\boldsymbol{\mu})$ vérifie, pour tout $v \in X_{\mathcal{N}}$,

$$\begin{aligned}
\Big(\Big(\hat{e}^{eim}_{pr}(\boldsymbol{\mu}), v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} &= \sum_{q=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ f^{q(M^q_f+1)}\big(v;\hat{\zeta}^{q(M^q_f+1)}(\boldsymbol{x})\big) \\
&\quad - \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ u_{Ni}(\boldsymbol{\mu})\ a^{q(M^q_a+1)}\big(\xi^{pr}_i, v;\hat{\zeta}^{q(M^q_a+1)}(\boldsymbol{x})\big).
\end{aligned} \tag{3.87}$$

Ensuite, en utilisant le principe de superposition linéaire, nous pouvons écrire

$$\hat{e}^{eim}_{pr}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\Gamma^{q(M^q_f+1)}_{N_{pr}} + \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})u_{Ni}(\boldsymbol{\mu})\Upsilon^{q(M^q_a+1)i}_{N_{pr}}, \tag{3.88}$$

avec, pour tout $v \in X_{\mathcal{N}}$,

$$\Big(\Big(\Gamma_{N_{pr}}^{q(M_f^q+1)}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = f^{q(M_f^q+1)}\big(v; \hat{\zeta}^{q(M_f^q+1)}(\boldsymbol{x})\big),\ 1 \le q \le Q_f, \tag{3.89}$$

$$\Big(\Big(\Upsilon_{N_{pr}}^{q(M_a^q+1)i}, v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -a^{q(M_a^q+1)}\big(\xi_i^{pr}, v; \hat{\zeta}^{q(M_a^q+1)}(\boldsymbol{x})\big),\ 1 \le q \le Q_a,\ 1 \le i \le N. \tag{3.90}$$

Nous avons donc

$$\epsilon_{N_{pr}^{eim}}(\boldsymbol{\mu})^2 = C_{N_{pr}^{eim}}^{ff}(\boldsymbol{\mu}) + 2\sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu}) C_{N_{pr}^{eim}i}^{fa}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{i'=1}^{N} u_{Ni}(\boldsymbol{\mu}) u_{Ni'}(\boldsymbol{\mu}) C_{N_{pr}^{eim}ii'}^{aa}(\boldsymbol{\mu}), \tag{3.91}$$

avec pour $1 \le i, i' \le N$:

$$C_{N_{pr}^{eim}}^{ff}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\sum_{q'=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi_{N_{pr}^{eim},ff}^{q(M_f^q+1)q'(M_f^q+1)}, \tag{3.92}$$

$$C_{N_{pr}^{eim}i}^{fa}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi_{N_{pr}^{eim},fa}^{q(M_a^q+1)iq'(M_f^q+1)}, \tag{3.93}$$

$$C_{N_{pr}^{eim}ii'}^{aa}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi_{N_{pr}^{eim},aa}^{q(M_a^q+1)iq'(M_a^q+1)i'}. \tag{3.94}$$

où on peut écrire

$$\Phi_{N_{pr}^{eim},ff}^{q(M_f^q+1)q'(M_f^q+1)} = \big(\big(\Gamma_{N_{pr}}^{q(M_f^q+1)}, \Gamma_{N_{pr}}^{q'(M_f^q+1)}\big)\big)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.95}$$

$$\Phi_{N_{pr}^{eim},fa}^{q(M_a^q+1)iq'(M_f^q+1)} = \big(\big(\Upsilon_{N_{pr}}^{q(M_a^q+1)i}, \Gamma_{N_{pr}}^{q'(M_f^q+1)}\big)\big)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.96}$$

$$\Phi_{N_{pr}^{eim},aa}^{q(M_a^q+1)iq'(M_a^q+1)i'} = \big(\big(\Upsilon_{N_{pr}}^{q(M_a^q+1)i}, \Upsilon_{N_{pr}}^{q'(M_a^q+1)i'}\big)\big)_{a,\boldsymbol{\mu}_{ref}}. \tag{3.97}$$

Toujours en nous servant de (3.85), le représentant de Riesz $\hat{e}_{du}^{eim}(\boldsymbol{\mu})$ vérifie, pour tout $v \in X_{\mathcal{N}}$,

$$\hat{e}_{du}^{eim}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\Gamma_{N_{du}}^{q(M_\ell^q+1)} + \sum_{q=1}^{Q_a}\sum_{i=1}^{N} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\Psi_{N_i}(\boldsymbol{\mu})\Upsilon_{N_{du}}^{q(M_a^q+1)i}, \tag{3.98}$$

avec, pour tout $v \in X_{\mathcal{N}}$,

$$\big(\big(\Gamma_{N_{du}}^{q(M_\ell^q+1)}, v\big)\big)_{a,\boldsymbol{\mu}_{ref}} = -\ell^{q(M_\ell^q+1)}\big(v; \hat{\zeta}^{q(M_\ell^q+1)}(\boldsymbol{x})\big),\ 1 \le q \le Q_\ell, \tag{3.99}$$

$$\big(\big(\Upsilon_{N_{du}}^{q(M_a^q+1)i}, v\big)\big)_{a,\boldsymbol{\mu}_{ref}} = -a^{q(M_a^q+1)}(v, \xi_i^{du}; \hat{\zeta}^{q(M_a^q+1)}(\boldsymbol{x})),\ 1 \le q \le Q_a,\ 1 \le i \le N. \tag{3.100}$$

Nous pouvons donc écrire

$$\epsilon_{N_{du}^{eim}}(\boldsymbol{\mu})^2 = C_{N_{du}^{eim}}^{\ell\ell}(\boldsymbol{\mu}) + 2\sum_{i=1}^{N} \Psi_{N_i} C_{N_{du}^{eim}i}^{\ell a}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{i'=1}^{N} \Psi_{N_i}\Psi_{N_{i'}} C_{N_{du}^{eim}ii'}^{aa}(\boldsymbol{\mu}), \tag{3.101}$$

avec pour $1 \leq i, i' \leq N$:

$$C^{\ell\ell}_{N^{eim}_{du}}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell}\sum_{q'=1}^{Q_\ell} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi^{q(M^q_\ell+1)q'(M^q_\ell+1)}_{N^{eim}_{du}}, \tag{3.102}$$

$$C^{\ell a}_{N^{eim}_{du}i}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_\ell} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi^{q(M^q_a+1)iq'(M^q_\ell+1)}_{N^{eim}_{du}}, \tag{3.103}$$

$$C^{aa}_{N^{eim}_{du}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi^{q(M^q_a+1)iq'(M^q_a+1)i'}_{N^{eim}_{du}}. \tag{3.104}$$

où on a

$$\Phi^{q(M^q_\ell+1)q'(M^q_\ell+1)}_{N^{eim}_{du},\ell\ell} = \big(\big(\Gamma^{q(M^q_\ell+1)}_{N_{du}}, \Gamma^{q'(M^q_\ell+1)}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{3.105}$$

$$\Phi^{q(M^q_a+1)iq'(M^q_\ell+1)}_{N^{eim}_{du},\ell a} = \big(\big(\Upsilon^{q(M^q_a+1)i}_{N_{du}}, \Gamma^{q'(M^q_\ell+1)}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{3.106}$$

$$\Phi^{q(M^q_a+1)iq'(M^q_a+1)i'}_{N^{eim}_{du},aa} = \big(\big(\Upsilon^{q(M^q_a+1)i}_{N_{du}}, \Upsilon^{q'(M^q_a+1)i'}_{N_{du}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}. \tag{3.107}$$

Pendant la phase *hors-ligne* il faut résoudre $(Q_f M^q_f + 2Q_a M^q_a N + Q_\ell M^q_\ell)$ problèmes de type Poisson qui dépendent de la dimension FE, $\mathcal{N}$, voir (3.65), (3.66), (3.75) et (3.76). Intéressons-nous maintenant à la phase *en-ligne*. Supposons que nous connaissons déjà les coefficients $u_{N_i}(\boldsymbol{\mu})$ et $\Psi_{N_i}(\boldsymbol{\mu})$, $1 \leq i \leq N$. Supposons également que les fonctions $\beta^{qm}_f(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_f$, $1 \leq m \leq M^q_f$, $\beta^{qm}_\ell(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_\ell$, $1 \leq m \leq M^q_\ell$ et $\beta^{qm}_a(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_a$, $1 \leq m \leq M^q_a$ sont déjà évaluées. Pour construire les termes introduits par (3.68)-(3.70) et (3.78)-(3.80), $\mathcal{O}(2Q^2_a M^{q2}_a + Q^2_f M^{q2}_f + Q^2_\ell M^{q2}_\ell)$ opérations sont nécessaires, ainsi que $\mathcal{O}(2Q^2_a + Q^2_f + Q^2_\ell)$ pour construire les termes (3.92)-(3.94) et (3.102)-(3.104). Une fois ces termes construits, nous pouvons évaluer les sommes définies en (3.67) et (3.77) en $\mathcal{O}(2Q^2_a M^{q2}_a N^2)$ opérations. Il faut compter $\mathcal{O}(2Q^2_a N^2)$ opérations pour évaluer les sommes définies en (3.91) et (3.101). Ainsi, la complexité *en-ligne* de l'estimation d'erreur est en $\mathcal{O}(2Q^2_a M^{q2}_a N^2 + 2Q^2_a N^2)$ opérations.

3 Problèmes paraboliques linéaires non-affines

3.1 Énoncé du problème général

Comme pour les problèmes paraboliques linéaires, nous considérons l'intervalle de temps $I = (0; T_f)$ où T_f est le temps final. Nous utilisons une discrétisation FE en espace et différences finies en temps. Pour cela divisons I en K sous-intervalles avec K un entier positif. Notons Δt le pas de temps donné par $\Delta t = \frac{T_f}{K}$ et par conséquent nous avons $t^k = k\ \Delta t$, $k \in \mathbb{K}$ avec $\mathbb{K} = \{0, ..., K\}$. Introduisons également l'ensemble $\mathbb{I} = \{t^0, ..., t^K\}$. Nous considérons, comme dans la section 2, que nous avons N_{na} fonctions non-affines , $\zeta^1(\boldsymbol{\mu}; \boldsymbol{x})$,..., $\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})$. En reprenant les notations introduites à la section 2 (3.22), pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ et $k \in \mathbb{K}$ donnés, nous nous intéressons à l'évaluation d'une sortie $s_\mathcal{N}(\boldsymbol{\mu}, t^k) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $u(\boldsymbol{\mu}, t)$:

$$s_\mathcal{N}(\boldsymbol{\mu}, t^k) = \ell\Big(u_\mathcal{N}(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}; t^k; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})}\Big), \tag{3.108}$$

pour un opérateur linéaire $\ell\big(.;\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X\times\mathcal{D}\times I\times E^{\underline{\zeta^{N_{na}}}}\to\mathbb{R}$ approprié, avec $u_{\mathcal{N}}(\boldsymbol{\mu},t^k)\in X_{\mathcal{N}}$ vérifiant

$$\left\{\begin{array}{l}\dfrac{1}{\Delta t}m\Big(u_{\mathcal{N}}(\boldsymbol{\mu},t^k)-u_{\mathcal{N}}(\boldsymbol{\mu},t^{k-1}),v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big)+a\Big(u_{\mathcal{N}}(\boldsymbol{\mu},t^k),v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big)\\ \qquad\qquad =f\Big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big),\ \forall v\in X_{\mathcal{N}},1\le k\le K,\\ \Big(u_{\mathcal{N}}(\boldsymbol{\mu},t^0),v\Big)_{L^2(\Omega)}=\Big(u_{ini}(\boldsymbol{\mu}),v\Big)_{L^2(\Omega)},\ \forall v\in X_{\mathcal{N}},\end{array}\right. \tag{3.109}$$

où $u_{ini}(\boldsymbol{\mu})\in L^2(\Omega)$, $m\big(.,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X\times X\times\mathcal{D}\times E^{\underline{\zeta^{N_{na}}}}\to\mathbb{R}$, $a\big(.,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X\times X\times\mathcal{D}\times E^{\underline{\zeta^{N_{na}}}}\to\mathbb{R}$, et $f\big(.;\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X\times\mathcal{D}\times I\times E^{\underline{\zeta^{N_{na}}}}\to\mathbb{R}$.

Comme précédemment, nous supposons que a est symétrique, définie positive, continue et coercive, voir (3.25) et (3.26). Nous supposons également que m est symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((.,.)\big)_{m,\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref}\in\mathcal{D}$), par

$$\big((w,z)\big)_{m,\boldsymbol{\mu}}\equiv m\big(w,z;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big),\ \forall w,z\in X_{\mathcal{N}}. \tag{3.110}$$

Notons que la norme associée à ce produit scalaire est

$$|||w|||_{m,\boldsymbol{\mu}}\equiv\sqrt{\big((w,w)\big)_{m,\boldsymbol{\mu}}},\ \forall w\in X. \tag{3.111}$$

De plus nous supposons que m est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ il existe une constante de continuité $\gamma^{m,\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\gamma^{m,\mathcal{N}}(\boldsymbol{\mu})\equiv\sup_{v\in X_{\mathcal{N}}}\frac{m(v,v;\boldsymbol{\mu})}{|||v|||^2_{m,\boldsymbol{\mu}_{ref}}}<\infty, \tag{3.112}$$

ainsi qu'une constante de coercivité $\alpha^{m,\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\alpha^{m,\mathcal{N}}(\boldsymbol{\mu})\equiv\inf_{v\in X_{\mathcal{N}}}\frac{m(v,v;\boldsymbol{\mu})}{|||v|||^2_{m,\boldsymbol{\mu}_{ref}}}>0. \tag{3.113}$$

Afin de mettre en oeuvre une stratégie *hors-ligne/en-ligne* efficace nous allons maintenant approcher, via la méthode EIM, les termes pour lequels il n'est pas possible d'écrire une décomposition affine directement par inspection. Autrement dit nous supposons qu'il existe des entiers positifs Q_a, Q_m, Q_f et Q_ℓ, avec $1\le Q_a,Q_m,Q_f,Q_\ell\le N_{na}$, de sorte que la méthode EIM détermine $\big(M_a^q\big)_{q=1,\dots,Q_a}$, $\big(M_m^q\big)_{q=1,\dots,Q_m}$, $\big(M_f^q\big)_{q=1,\dots,Q_f}$ et $\big(M_\ell^q\big)_{q=1,\dots,Q_\ell}$ tels que, pour $n_a=\sum_{q=1}^{Q_a}M_a^q$, $n_m=\sum_{q=1}^{Q_m}M_m^q$, $n_f=\sum_{q=1}^{Q_f}M_f^q$ et $n_\ell=\sum_{q=1}^{Q_\ell}M_\ell^q$, ainsi que pour tout $\boldsymbol{\mu}\in\mathcal{D}$ et $k\in\mathbb{K}$, les formes bilinéaires et linéaires $m(.,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$, $a(.,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$, $f(.;\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$ et $\ell(.,\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$ puissent être appro-

chées de la manière suivante :

$$\begin{cases} m\big(u,v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{m}\big(u,v;\boldsymbol{\mu};\underline{\zeta^{n_m}_{M^{n_m}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q} \beta_m^{qm}(\boldsymbol{\mu}) m^{qm}(u,v;\hat{\zeta}^{qm}(\boldsymbol{x})), \\ a\big(u,v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{a}\big(u,v;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu}) a^{qm}(u,v;\hat{\zeta}^{qm}(\boldsymbol{x})), \\ f\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{f}\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{n_f}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu};t^k) f^{qm}\big(v;\hat{\zeta}^{qm}(\boldsymbol{x})\big), \\ \ell\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{\ell}\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{n_\ell}_{M^{n_\ell}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu};t^k) \ell^{qm}\big(v;\hat{\zeta}^{qm}(\boldsymbol{x})\big), \end{cases} \tag{3.114}$$

pour tout $u \in X_{\mathcal{N}}$ et pour tout $v \in X_{\mathcal{N}}$, où $\beta_m^{qm} : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_m$, $1 \le m \le M_m^q$, $\beta_a^{qm} : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_a$, $1 \le m \le M_a^q$, $\beta_f^{qm} : \mathcal{D} \times \mathbb{I} \to \mathbb{R}$, $1 \le q \le Q_f$, $1 \le m \le M_f^q$ et $\beta_\ell^{qm} : \mathcal{D} \times \mathbb{I} \to \mathbb{R}$, $1 \le q \le Q_\ell$, $1 \le m \le M_\ell^q$ sont déterminées par la méthode EIM. Posons $E^{\underline{\zeta^{n_m}_{M^{n_m}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_m}_{M^{n_m}}}$. Notons que nous avons $\tilde{a}\big(.,.;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_a}_{M^{n_a}}}} \to \mathbb{R}$, $\tilde{m}\big(.,.;\boldsymbol{\mu};\underline{\zeta^{n_m}_{M^{n_m}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_m}_{M^{n_m}}}} \to \mathbb{R}$, $\tilde{f}\big(.;\boldsymbol{\mu};t^k;\underline{\zeta^{n_f}_{M^{n_f}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times \mathcal{D} \times I \times E^{\underline{\zeta^{n_f}_{M^{n_f}}}} \to \mathbb{R}$ et $\tilde{\ell}\big(.;\boldsymbol{\mu};t^k;\underline{\zeta^{n_\ell}_{M^{n_\ell}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times \mathcal{D} \times I \times E^{\underline{\zeta^{n_\ell}_{M^{n_\ell}}}} \to \mathbb{R}$.

En nous servant des approximations décrites dans (3.114), pour $\boldsymbol{\mu} \in \mathcal{D}$ fixé, (3.109) revient à chercher $u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}$ tel que

$$\begin{cases} \dfrac{1}{\Delta t}\tilde{m}\Big(u_{\mathcal{N}}(\boldsymbol{\mu},t^k) - u_{\mathcal{N}}(\boldsymbol{\mu},t^{k-1}), v;\boldsymbol{\mu};\underline{\zeta^{n_m}_{M^{n_m}}(\boldsymbol{\mu};\boldsymbol{x})}\Big) + \tilde{a}\Big(u_{\mathcal{N}}(\boldsymbol{\mu},t^k), v;\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\Big) \\ \qquad\qquad = \tilde{f}\Big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{n_f}_{M^{n_f}}(\boldsymbol{\mu};\boldsymbol{x})}\Big),\ \forall v \in X_{\mathcal{N}}, 1 \le k \le K, \\ \Big(u_{\mathcal{N}}(\boldsymbol{\mu},t^0), v\Big)_{L^2(\Omega)} = \Big(u_{ini}(\boldsymbol{\mu}), v\Big)_{L^2(\Omega)},\ \forall v \in X_{\mathcal{N}}. \end{cases} \tag{3.115}$$

Comme précédemment, nous supposons que $\tilde{a}$ est symétrique, définie positive, continue et coercive, voir (3.31) et (3.32). Nous supposons également que $\tilde{m}$ est symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((.,.)\big)_{\tilde{m},\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((w,z)\big)_{\tilde{m},\boldsymbol{\mu}} \equiv m\big(w,z;\boldsymbol{\mu};\underline{\zeta^{n_m}_{M^{n_m}}(\boldsymbol{\mu};\boldsymbol{x})}\big),\ \forall w,z \in X_{\mathcal{N}}. \tag{3.116}$$

Notons que la norme associée à ce produit scalaire est

$$\big|\big|\big|w\big|\big|\big|_{\tilde{m},\boldsymbol{\mu}} \equiv \sqrt{\big((w,w)\big)_{\tilde{m},\boldsymbol{\mu}}},\ \forall w \in X. \tag{3.117}$$

De plus nous supposons que $\tilde{m}$ est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ il existe une constante de continuité $\gamma^{\tilde{m},\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\gamma^{\tilde{m},\mathcal{N}}(\boldsymbol{\mu}) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{\tilde{m}(v,v;\boldsymbol{\mu})}{\big|\big|\big|v\big|\big|\big|^2_{\tilde{m},\boldsymbol{\mu}_{ref}}} < \infty, \tag{3.118}$$

ainsi qu'une constante de coercivité $\alpha^{\tilde{m},\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\alpha^{\tilde{m},\mathcal{N}}(\boldsymbol{\mu}) \equiv \inf_{v \in X_{\mathcal{N}}} \frac{\tilde{m}(v,v;\boldsymbol{\mu})}{\big|\big|\big|v\big|\big|\big|^2_{\tilde{m},\boldsymbol{\mu}_{ref}}} > 0. \tag{3.119}$$

3.2 Méthode des bases réduites

De manière similaire au cas linéaire, nous introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{pr}}$ tels que $W_{1_{pr}} \subset W_{2_{pr}} \subset ... \subset W_{N_{max_{pr}}} \subset X_{\mathcal{N}}$, avec N_{max} un entier positif. Pour un $\boldsymbol{\mu}$ donné définissons l'ensemble des solutions de (3.115) $Snap^u(\boldsymbol{\mu}) = \{u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) \in X_{\mathcal{N}}, 1 \leq k \leq K\}$. $S_{\bar{N}} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_{\bar{N}}\}$ est un ensemble de $\bar{N}$ jeux de paramètres, avec $1 \leq \bar{N} \leq N$, sélectionnés à l'aide de l'algorithme POD/glouton (se reporter à l'algorithme 3). Notons $S^u_{\bar{N}}$ l'ensemble qui va contenir, pour un $\boldsymbol{\mu} \in S_{\bar{N}}$, les N_m modes propres principaux issus de $POD(Snap^u(\boldsymbol{\mu}), N_m)$. Cet ensemble s'écrit

$$S^u_{\bar{N}} = \Big\{ POD(Snap^u(\boldsymbol{\mu}), N_m),\ \forall \boldsymbol{\mu} \in S_{\bar{N}} \Big\}. \tag{3.120}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble $S^u_{\bar{N}}$ pour avoir des fonctions de bases orthonormalisées ξ^{pr}_n, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{pr}}$ est défini par

$$W_{N_{pr}} = span\Big\{ \xi^{pr}_n,\ 1 \leq n \leq N \Big\}. \tag{3.121}$$

Quant à la solution réduite $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ elle s'écrit

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu})\ \xi^{pr}_i. \tag{3.122}$$

Nous allons maintenant utiliser les approximations décrites dans (3.114) pour construire une stratégie *hors-ligne/ en-ligne* efficace. En choisissant les fonctions test comme $v = \xi^{pr}_n$, $i = n, ..., N$, alors pour $k = 1, ..., K$, $u_N(\boldsymbol{\mu}, t^k)$ est solution de

$$\begin{aligned}
&\sum_{i=1}^{N} \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \sum_{m=1}^{M^q_m} \beta^{qm}_m(\boldsymbol{\mu}) m^{qm}\Big(\xi^{pr}_n, \xi^{pr}_i; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \right. \\
&\qquad \left. + \sum_{q=1}^{Q_a} \sum_{m=1}^{M^q_a} \beta^q_a(\boldsymbol{\mu}) a^{qm}\Big(\xi^{pr}_n, \xi^{pr}_i; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \right) u_{N_i}(\boldsymbol{\mu}; t^k) \\
&= \left(\sum_{q=1}^{Q_f} \sum_{m=1}^{M^q_f} \beta^{qm}_f(\boldsymbol{\mu}; t^k) f^{qm}\Big(\xi^{pr}_n; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) + \frac{1}{\Delta t} \sum_{q=1}^{Q_m} \sum_{m=1}^{M^q_m} \beta^{qm}_m(\boldsymbol{\mu}) m^{qm}\Big(u_N(\boldsymbol{\mu}; t^{k-1}), \xi^{pr}_n\Big) \right),
\end{aligned} \tag{3.123}$$

qui peut être également écrit, pour $k = 1, ..., K$, sous la forme matricielle :

$$\begin{aligned}
&\left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \sum_{m=1}^{M^q_m} \beta^{qm}_m(\boldsymbol{\mu}) M^{qm}_N + \sum_{q=1}^{Q_a} \sum_{m=1}^{M^q_a} \beta^{qm}_a(\boldsymbol{\mu}) A^{qm}_N \right) u_N(\boldsymbol{\mu}, t^k) \\
&= \left(\sum_{q=1}^{Q_f} \sum_{m=1}^{M^q_f} \beta^{qm}_f(\boldsymbol{\mu}; t^k) F^{qm}_N + \frac{1}{\Delta t} \sum_{q=1}^{Q_m} \sum_{m=1}^{M^q_m} \beta^{qm}_m(\boldsymbol{\mu}) M^{qm}_N\ u_N(\boldsymbol{\mu}; t^{k-1}) \right),
\end{aligned} \tag{3.124}$$

avec

$$\begin{aligned}
&\Big(u_N(\boldsymbol{\mu}, t^k)\Big)_i = u_{N_i}(\boldsymbol{\mu}; t^k),\ \Big(M^{qm}_N\Big)_{in} = m^{qm}\Big(\xi^{pr}_n, \xi^{pr}_i; \hat{\zeta}^{qm}(\boldsymbol{x})\Big), \\
&\Big(A^{qm}_N\Big)_{in} = a^{qm}\Big(\xi^{pr}_n, \xi^{pr}_i; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \text{ et } \Big(F^{qm}_N\Big)_n = f^{qm}\Big(\xi^{pr}_n; \hat{\zeta}^{qm}(\boldsymbol{x})\Big).
\end{aligned} \tag{3.125}$$

et $\big(u_N(\boldsymbol{\mu},t^0),v\big)_{L^2(\Omega)}=\big(u_{\mathcal{N}}(\boldsymbol{\mu},t^0),v\big)_{L^2(\Omega)}$, $\forall v\in W_{N_{pr}}$ à l'instant initial.
La sortie s'exprime alors, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ et pour tout $k\in\mathbb{K}$, de la manière suivante :

$$s_N(\boldsymbol{\mu};t^k)=\left(\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q}\beta_\ell^{qm}(\boldsymbol{\mu};t^k)\ell^{qm}\Big(u_N(\boldsymbol{\mu},t^k);\hat{\zeta}^{qm}(\boldsymbol{x})\Big)\right), \tag{3.126}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu},t^k)=\left(\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q}\beta_\ell^{qm}(\boldsymbol{\mu};t^k)L_N^{qm\ T}\right)u_N(\boldsymbol{\mu},t^k),\ \forall\boldsymbol{\mu}\in\mathcal{D},\ \forall k\in\mathbb{K}, \tag{3.127}$$

avec $(L_N^{qm})_n=\ell^{qm}\big(\xi_n^{pr};\hat{\zeta}^{qm}(\boldsymbol{x})\big)$. La construction des matrices $A_N^{qm}\in\mathbb{R}^{N\times N}$, $1\le q\le Q_a$, $1\le m\le M_a^q$, $M_N^{qm}\in\mathbb{R}^{N\times N}$, $1\le q\le Q_m$, $1\le m\le M_m^q$, et des vecteurs $F_N^{qm}\in\mathbb{R}^N$, $1\le q\le Q_f$, $1\le m\le M_f^q$ et $L_N^{qm}\in\mathbb{R}^N$,$1\le q\le Q_\ell$, $1\le m\le M_\ell^q$ se fait durant l'étape *hors-ligne*. À l'étape *en-ligne*, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ et $k\in\mathbb{K}$ donnés, on assemble les matrices $A_N(\boldsymbol{\mu})=\sum_{q=1}^{Q_a}\sum_{m=1}^{M_q^a}\beta_a^{qm}(\boldsymbol{\mu})\ A_N^{qm}$ et $M_N(\boldsymbol{\mu})=\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^a}\beta_m^{qm}(\boldsymbol{\mu})\ M_N^{qm}$ ainsi que les vecteurs $F_N(\boldsymbol{\mu};t^k)=\sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^a}\beta_f^{qm}(\boldsymbol{\mu};t^k)\ F_N^{qm}$ et $L_N(\boldsymbol{\mu};t^k)=\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^a}\beta_\ell^{qm}(\boldsymbol{\mu};t^k)$ L_N^{qm}. Enfin, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ et $k=1,...,K$, nous résolvons le système réduit

$$\left(\frac{1}{\Delta t}M_N(\boldsymbol{\mu})+A_N(\boldsymbol{\mu})\right)u_N(\boldsymbol{\mu},t^k)=\left(F_N(\boldsymbol{\mu};t^k)+\frac{1}{\Delta t}M_N(\boldsymbol{\mu})u_N(\boldsymbol{\mu};t^{k-1})\right), \tag{3.128}$$

et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu},t^k)=L_N^T(\boldsymbol{\mu};t^k)\ u_N(\boldsymbol{\mu},t^k),\ \forall\boldsymbol{\mu}\in D,\ \forall k\in\mathbb{K}. \tag{3.129}$$

Nous venons de voir l'approche *primale*. Comme précédemment, nous allons introduire l'approche *primale-duale*. Pour tout $\boldsymbol{\mu}\in\mathcal{D}$, le problème dual associé à la sortie du modèle consiste à chercher $\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^k)\in X_{\mathcal{N}}$ tel que

$$\begin{cases}\dfrac{1}{\Delta t}m\Big(\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^k),v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big)+a\Big(\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^k),v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big)\\ \qquad=\dfrac{1}{\Delta t}m\Big(v,\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^{k+1});\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big),\ \forall v\in X_{\mathcal{N}},K\ge k\ge 1,\\ m\Big(v,\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^{K+1});\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big)=\ell\Big(v;\boldsymbol{\mu};t^{K+1};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})}\Big),\ \forall v\in X_{\mathcal{N}}.\end{cases} \tag{3.130}$$

De la même manière que pour l'approche *primale* nous introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{du}}$, $1\le N\le N_{max}$. Définissons $Snap^{\Psi}(\boldsymbol{\mu})$ l'ensemble des solutions du problème dual (3.130) pour $\boldsymbol{\mu}\in\mathcal{D}$ fixé. Nous avons $Snap^{\Psi}(\boldsymbol{\mu})=\{\Psi_{\mathcal{N}}(\boldsymbol{\mu},t^k)\in X_{\mathcal{N}},K\ge k\ge 1\}$. $S_{\bar{N}}^{\Psi}$ est l'ensemble défini par

$$S_{\bar{N}}^{\Psi}=\Big\{POD(Snap^{\Psi}(\boldsymbol{\mu}),N_m),\ \forall\boldsymbol{\mu}\in S_{\bar{N}}\Big\}. \tag{3.131}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble $S_{\bar{N}}^{\Psi}$ pour avoir des fonctions de bases orthonormalisées ξ^{du}, $1\le n\le N$. L'espace d'approximation $W_{N_{du}}$ est défini par

$$W_{N_{du}}=span\Big\{\xi_n^{du},\ 1\le n\le N\Big\}. \tag{3.132}$$

Quant à la solution réduite $\Psi_N(\boldsymbol{\mu}, t^k) \in W_{N_{du}}$ elle s'écrit

$$\Psi_N(\boldsymbol{\mu}, t^k) = \sum_{i=1}^{N} \Psi_{Ni}(\boldsymbol{\mu}, t^k)\ \xi_i^{du}. \tag{3.133}$$

L'évaluation de la sortie, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et $k \in \mathbb{K}$, est donnée par

$$\begin{aligned} s_N(\boldsymbol{\mu}, t^k) \ &= \ \tilde{\ell}\left(u_N(\boldsymbol{\mu}, t^k); \boldsymbol{\mu}; t^k; \underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu}; \boldsymbol{x})}\right) \\ &\quad + \sum_{k'=1}^{k} \tilde{r}_{pr}\left(\Psi_N(\boldsymbol{\mu}, t^{K-k+k'}); \boldsymbol{\mu}; t^{k'}; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})}\right)\ \Delta t, \end{aligned} \tag{3.134}$$

avec $1 \le n_r \le N_{na}$, ainsi que pour tout $v \in X_{\mathcal{N}}$ et $1 \le k \le K$:

$$\begin{aligned} \tilde{r}_{pr}\left(v; \boldsymbol{\mu}; t^k; \underline{\zeta_{M^{n_r}}^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})}\right) \ &= \ \tilde{f}\left(v; \boldsymbol{\mu}; t^k; \underline{\zeta_{M^{n_f}}^{n_f}(\boldsymbol{\mu}; \boldsymbol{x})}\right) \\ &\quad -\tilde{a}\left(u_N(\boldsymbol{\mu}, t^k), v; \boldsymbol{\mu}; \underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu}; \boldsymbol{x})}\right) \\ &\quad -\frac{1}{\Delta t}\tilde{m}\left(u_N(\boldsymbol{\mu}, t^k) - u_N(\boldsymbol{\mu}, t^{k-1}); \boldsymbol{\mu}; \underline{\zeta_{M^{n_m}}^{n_m}(\boldsymbol{\mu}; \boldsymbol{x})}\right). \end{aligned} \tag{3.135}$$

Nous allons maintenant utiliser les approximations décrites dans (3.114) pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{du},\ i = n, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $\Psi_N(\boldsymbol{\mu}, t^k)$ est solution de

$$\begin{aligned} &\sum_{i=1}^{N}\left(\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\beta_m^{qm}(\boldsymbol{\mu})m^{qm}\left(\xi_n^{du}, \xi_i^{du}; \underline{\zeta_{M^{n_m}}^{n_m}(\boldsymbol{\mu}; \boldsymbol{x})}\right)\right. \\ &\qquad \left. + \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\beta_a^{qm}(\boldsymbol{\mu})a^{qm}\left(\xi_n^{du}, \xi_i^{du}; \underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu}; \boldsymbol{x})}\right)\right)\ \Psi_{N_i}(\boldsymbol{\mu}; t^k) \\ &= \left(\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\beta_m^{qm}(\boldsymbol{\mu})m^{qm}\left(\Psi_N(\boldsymbol{\mu}, t^{k+1}), \xi_n^{du}; \underline{\zeta_{M^{n_m}}^{n_m}(\boldsymbol{\mu}; \boldsymbol{x})}\right)\right),\ K \le k \le 1, \end{aligned} \tag{3.136}$$

qui peut être également écrit, pour $k = K, ..., 1$, sous la forme matricielle :

$$\begin{aligned} &\left(\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\beta_m^{qm}(\boldsymbol{\mu})M_N^{qm} + \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\beta_a^{qm}(\boldsymbol{\mu})A_N^{qm}\right)\Psi_N(\boldsymbol{\mu}, t^k) \\ &\qquad = \left(\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\beta_m^{qm}(\boldsymbol{\mu})M_N^{qm}\ \Psi_N(\boldsymbol{\mu}, t^{k+1})\right), \end{aligned} \tag{3.137}$$

avec

$$\begin{aligned} &\left(\Psi_N(\boldsymbol{\mu}, t^k)\right)_i = \Psi_{N_i}(\boldsymbol{\mu}; t^k),\ \left(M_N^{qm}\right)_{in} = m^{qm}\left(\xi_n^{du}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right), \\ &\text{et } \left(A_N^{qm}\right)_{in} = a^{qm}\left(\xi_n^{du}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right). \end{aligned} \tag{3.138}$$

Au temps final t^{K+1} nous avons

$$\Big(\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\beta_m^{qm}(\boldsymbol{\mu})M_N^{qm}\Big)\Psi_N(\boldsymbol{\mu}, t^k) = \Big(\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q}\beta_m^{qm}(\boldsymbol{\mu}; t^{K+1})L_N^{qm}\Big) \tag{3.139}$$

avec

$$\left(L_N^{qm}\right)_n = \ell^{qm}\left(\xi_n^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right). \tag{3.140}$$

La sortie s'exprime alors, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour tout $k \in \mathbb{K}$, de la manière suivante :

$$s_N(\boldsymbol{\mu}, t^k) = \left(\sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu}; t^k) L_N^{qm\ T}\right) u_N(\boldsymbol{\mu}, t^k) + \sum_{k'=1}^{k} R_{pr}^{\Psi_N(\boldsymbol{\mu}, t^{K-k+k'})}, \tag{3.141}$$

avec

$$\begin{aligned} R_{pr}^{\Psi_N(\boldsymbol{\mu}, t^{K-k+k'})} \quad = \quad & \left(\sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu}; t^{k'}) F_N^{du,q}\right) \Psi_N(\boldsymbol{\mu}, t^{K-k+k'}) \\ & -\Psi_N(\boldsymbol{\mu}, t^{K-k+k'})^T \left(\sum_{q=1}^{Q_a}\sum_{m=1}^{M_m^q} \beta_a^{qm}(\boldsymbol{\mu}) A_N^{prdu,q}\right) u_N(\boldsymbol{\mu}, t^k) \\ & -\frac{1}{\Delta t}\Psi_N(\boldsymbol{\mu}, t^{K-k+k'})^T \left(\sum_{q=1}^{Q_m}\sum_{q=1}^{M_m^q} \beta_a^{q}(\boldsymbol{\mu}) M_N^{prdu,q}\right) \Big(u_N(\boldsymbol{\mu}, t^k) - u_N(\boldsymbol{\mu}, t^{k-1})\Big), \end{aligned} \tag{3.142}$$

ainsi que

$$\begin{aligned} &\left(A_N^{prdu,qm}\right)_{in} = a^{qm}\left(\xi_n^{pr}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right), \left(M_N^{prdu,qm}\right)_{in} = m^{qm}\left(\xi_n^{pr}, \xi_i^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right) \\ &\text{et} \quad \left(F_N^{du,qm}\right)_n = f^{qm}\left(\xi_n^{du}; \hat{\zeta}^{qm}(\boldsymbol{x})\right). \end{aligned} \tag{3.143}$$

Regardons la complexité algorithmique de la partie *en-ligne* dans le cas où nous nous servons de k pas de temps. Il faut $\mathcal{O}(Q_a M_a^q N^2 + Q_m M_m^q N^2)$ multiplications pour assembler le terme de gauche du système réduit primal (3.124) ou dual (3.137). La construction du second membre implique $\mathcal{O}(kN^2)$ opérations, aussi bien pour le système primal que dual. La résolution du problème, primal ou dual, nécessite $\mathcal{O}(kN^3)$ opérations. Ensuite, dans le cas de sorties "souples", il faut compter $\mathcal{O}(kN)$ produits scalaires pour avoir la sortie du modèle via (3.127), mais si l'on souhaite intégrer le terme de correction pour les sorties non "souples" (3.141) cela en nécessite alors $\mathcal{O}(kN^3)$. L'étape *en-ligne* reste indépendante de la (grande) dimension $\mathcal{N}$.

3.3 Estimation d'erreur a posteriori

À l'aide de la formule (3.134), pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné et un temps t^k donné, $1 \leq k \leq K$, nous pouvons construire rapidement une approximation de la sortie du modèle. Ici nous introduisons des estimateurs d'erreur a posteriori pour les problèmes paraboliques linéaires non-affines. La différence avec les estimateurs d'erreur présentés dans la section 3 du chapitre 2 réside dans le fait que nous devons estimer l'erreur commise par l'approximation des termes non-affines via la méthode EIM. Ces estimateurs vont nous permettre de savoir, rapidement et de manière fiable, si l'approximation de la solution primale, duale, et de la sortie du modèle sont suffisament précises.

Contrairement à ce qui a été fait dans cette thèse jusqu'à maintenant, nous allons proposer une estimation d'erreur sans nous servir du problème dual, c'est à dire qu'il s'agit d'un estimateur d'erreur pour la sortie définie en (3.127). Commençons par définir, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $1 \leq k \leq K$ et pour tout $w \in X_{\mathcal{N}}$, la norme suivante :

$$\left|\left|\left|w(t^k)\right|\right|\right|_{pr,\boldsymbol{\mu}} \equiv \sqrt{m\Big(w(t^k), w(t^k); \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})}\Big) + \sum_{k'=1}^{k} a\Big(w(t^{k'}), w(t^{k'}); \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x})}\Big)\Delta t} \tag{3.144}$$

Introduisons la norme duale du résidu primal $\epsilon_{N_{pr}}(\boldsymbol{\mu}, t^k)$:

$$\epsilon_{N_{pr}}(\boldsymbol{\mu}, t^k) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{\tilde{r}_{pr}(v; \boldsymbol{\mu}; t^k; \underline{\zeta^{n_r}_{M^{nr}}(\boldsymbol{\mu}; \boldsymbol{x})})}{\left|\left|\left|v\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}} = \left|\left|\left|\hat{e}^{eim}_{pr}(\boldsymbol{\mu}, t^k)\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}} \tag{3.145}$$

Notons que pour définir la norme duale, la représentation de Riesz des résidus primal $\hat{e}_{pr}(\boldsymbol{\mu}, t^k)$ a été introduit dans (3.145). Introduisons également la norme duale du résidu primal $\hat{e}^{eim}_{pr}(\boldsymbol{\mu}, t^k)$ en nous limitant aux termes comportant des fonctions non-affines.

$$\epsilon_{N^{eim}_{pr}}(\boldsymbol{\mu}, t^k) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{\tilde{r}_{pr}(v; \boldsymbol{\mu}; t^k; \underline{\zeta^{n_r}(\boldsymbol{\mu}; \boldsymbol{x})} - \underline{\zeta^{n_r}_{M^{nr}}(\boldsymbol{\mu}; \boldsymbol{x})})}{\left|\left|\left|v\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}} = \left|\left|\left|\hat{e}^{eim}_{pr}(\boldsymbol{\mu}, t^k)\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}} \tag{3.146}$$

À présent définissons l'estimation d'erreur a posteriori sur l'approximation de la sortie du modèle. Pour chaque N et pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\left|s_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - s_N(\boldsymbol{\mu}, t^k)\right| \approx \Delta^s_N(\boldsymbol{\mu}, t^k) \equiv \sup_{v \in X_{\mathcal{N}}} \frac{\ell(v)}{\left|\left|\left|v\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}} \Delta^{pr}_N(\boldsymbol{\mu}, t^k),\ 1 \leq k \leq K, \tag{3.147}$$

où $\Delta^{pr}_N(\boldsymbol{\mu}; t^k)$ est l'estimation d'erreur a posteriori pour la solution primale, défini par

$$\Delta^{pr}_N(\boldsymbol{\mu}, t^k) = \sqrt{\frac{2\Delta t}{\alpha^a_{LB}(\boldsymbol{\mu})} \sum_{k'=1}^{k} \epsilon_{N_{pr}}(\boldsymbol{\mu}, t^{k'})^2 + \frac{2\Delta t}{\alpha^a_{LB}(\boldsymbol{\mu})} \sum_{k'=1}^{k} \epsilon_{N^{eim}_{pr}}(\boldsymbol{\mu}, t^{k'})^2}, \tag{3.148}$$

pour $k = 1, ..., K$, et nous avons

$$\left|\left|\left|u_{\mathcal{N}}(\boldsymbol{\mu}, t^k) - u_N(\boldsymbol{\mu}, t^k)\right|\right|\right|_{pr,\boldsymbol{\mu}_{ref}} \approx \Delta^{pr}_N(\boldsymbol{\mu}, t^k),\ \forall \boldsymbol{\mu} \in \mathcal{D},\ 1 \leq k \leq K. \tag{3.149}$$

Remarque 10. *Les estimateurs d'erreur proposés via (3.147) et (3.148) sont des estimateurs rigoureux uniquement dans le cas où $\zeta^i_{M^i} \in W^{\zeta^i}_{M^i}$, pour $i = 1, ..., N_{na}$.*

Stratégie *hors-ligne/en-ligne*

Tout comme dans le cas elliptique, nous exposons ici la stratégie *hors-ligne/en-ligne* à mettre en oeuvre afin d'évaluer les estimations d'erreur définies dans (3.147) et (3.148) de manière efficace. En nous appuyant sur les approximations décrites dans (3.114), nous pouvons exprimer le résidu primal, pour tout $v \in X_{\mathcal{N}}$ et pour $1 \leq k \leq K$, de la manière

suivante :

$$\begin{aligned}\tilde{r}_{pr}\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{n_r}_{M^{n_r}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \;& \sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q}\beta_f^{qm}\big(\boldsymbol{\mu};t^k\big)f^{qm}\big(v;\hat{\zeta}^{qm}(\boldsymbol{x})\big)\\ &-\sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{i=1}^{N}\beta_a^{qm}\big(\boldsymbol{\mu}\big)u_{Ni}\big(\boldsymbol{\mu};t^k\big)a^{qm}\big(\xi_i^{pr},v;\hat{\zeta}^{qm}(\boldsymbol{x})\big)\\ &-\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\sum_{i=1}^{N}\beta_m^{qm}\big(\boldsymbol{\mu}\big)u_{Ni}(\boldsymbol{\mu};t^k)m^{qm}\big(\xi_i^{pr},v;\hat{\zeta}^{qm}(\boldsymbol{x})\big)\\ &+\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\sum_{i=1}^{N}\beta_m^{qm}\big(\boldsymbol{\mu}\big)u_{Ni}(\boldsymbol{\mu};t^{k-1})m^{qm}\big(\xi_i^{pr},v;\hat{\zeta}^{qm}(\boldsymbol{x})\big)\end{aligned} \tag{3.150}$$

Le représentant de Riesz $\hat{e}_{pr}(\boldsymbol{\mu},t^k)$ vérifie, pour tout $v\in X_{\mathcal{N}}$,

$$\Big(\Big(\hat{e}_{pr}(\boldsymbol{\mu},t^k),v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = \tilde{r}_{pr}\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{n_r}_{M^{n_r}}(\boldsymbol{\mu};\boldsymbol{x})}\big). \tag{3.151}$$

Ensuite, en utilisant le principe de superposition linéaire, nous pouvons écrire, pour $k=1,...,K$

$$\begin{aligned}\hat{e}_{pr}(\boldsymbol{\mu},t^k) = \;& \sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q}\beta_f^{qm}(\boldsymbol{\mu};t^k)\Gamma^{qm}_{N_{pr}}+\sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{i=1}^{N}\beta_a^{qm}(\boldsymbol{\mu})u_{Ni}(\boldsymbol{\mu})\Upsilon^{qmi}_{N_{pr}}\\ &+\frac{1}{\Delta t}\sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\sum_{i=1}^{N}\beta_m^{qm}(\boldsymbol{\mu})\Big(u_{Ni}(\boldsymbol{\mu};t^k)-u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big)\Lambda^{qmi}_{N_{pr}},\end{aligned} \tag{3.152}$$

avec, pour tout $v\in X_{\mathcal{N}}$,

$$\Big(\Big(\Gamma^{qm}_{N_{pr}},v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = f^{qm}\big(v;\hat{\zeta}^{qm}(\boldsymbol{x})\big),\ 1\le q\le Q_f,\ 1\le m\le M_f^q, \tag{3.153}$$

ainsi que pour $1\le i\le N$,

$$\Big(\Big(\Upsilon^{qmi}_{N_{pr}},v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -a^{qm}\big(\xi_i^{pr},v;\hat{\zeta}^{qm}(\boldsymbol{x})\big),\ 1\le q\le Q_a,\ 1\le m\le M_a^q, \tag{3.154}$$

$$\Big(\Big(\Lambda^{qmi}_{N_{pr}},v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} = -m^{qm}\big(\xi_i^{pr},v;\hat{\zeta}^{qm}(\boldsymbol{x})\big),\ 1\le q\le Q_m,\ 1\le m\le M_m^q. \tag{3.155}$$

Par conséquent nous avons

$$\begin{aligned}\epsilon_{N_{pr}}(\boldsymbol{\mu},t^k)^2 = \;& 2\sum_{i=1}^{N}u_{Ni}(\boldsymbol{\mu};t^k)C^{fa}_{N_{pr}i}(\boldsymbol{\mu};t^k)+\sum_{i=1}^{N}\sum_{i'=1}^{N}u_{Ni}(\boldsymbol{\mu};t^k)u_{Ni'}(\boldsymbol{\mu};t^k)C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu})\\ &+\frac{1}{\Delta t^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\Big(u_{Ni}(\boldsymbol{\mu};t^k)-u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big)\Big(u_{Ni'}(\boldsymbol{\mu};t^k)-u_{Ni'}(\boldsymbol{\mu};t^{k-1})\Big)C^{mm}_{N_{pr}ii'}(\boldsymbol{\mu})\\ &+\frac{2}{\Delta t}\sum_{i=1}^{N}\sum_{i'=1}^{N}\Big(u_{Ni}(\boldsymbol{\mu};t^k)-u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big)u_{Ni'}(\boldsymbol{\mu};t^k)C^{am}_{N_{pr}ii'}(\boldsymbol{\mu})\\ &+\frac{2}{\Delta t}\sum_{i=1}^{N}\Big(u_{Ni}(\boldsymbol{\mu};t^k)-u_{Ni}(\boldsymbol{\mu};t^{k-1})\Big)C^{fm}_{N_{pr}i}(\boldsymbol{\mu};t^k)+C^{ff}_{N_{pr}}(\boldsymbol{\mu};t^k),\quad 1\le k\le K,\end{aligned} \tag{3.156}$$

avec pour $1 \leq i, i' \leq N$:

$$C^{ff}_{N_{pr}}(\boldsymbol{\mu};t^k) = \sum_{q=1}^{Q_f}\sum_{m=1}^{M_f^q}\sum_{q'=1}^{Q_f}\sum_{m'=1}^{M_f^q} \beta_f^{qm}(\boldsymbol{\mu};t^k)\ \beta_f^{q'm'}(\boldsymbol{\mu};t^k)\ \Phi^{qmq'm'}_{N_{pr},ff}, \tag{3.157}$$

$$C^{fa}_{N_{pr}i}(\boldsymbol{\mu};t^k) = \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{q'=1}^{Q_f}\sum_{m'=1}^{M_f^q} \beta_a^{qm}(\boldsymbol{\mu})\ \beta_f^{q'm'}(\boldsymbol{\mu};t^k)\ \Phi^{qmiq'm'}_{N_{pr},fa}, \tag{3.158}$$

$$C^{aa}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{q'=1}^{Q_a}\sum_{m'=1}^{M_a^q} \beta_a^{qm}(\boldsymbol{\mu})\ \beta_a^{q'm'}(\boldsymbol{\mu})\ \Phi^{qmiq'm'i'}_{N_{pr},aa}, \tag{3.159}$$

$$C^{am}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{m=1}^{M_a^q}\sum_{q'=1}^{Q_m}\sum_{m'=1}^{M_m^q} \beta_a^{qm}(\boldsymbol{\mu})\ \beta_m^{q'm'}(\boldsymbol{\mu})\ \Phi^{qmiq'm'i'}_{N_{pr},am}, \tag{3.160}$$

$$C^{fm}_{N_{pr}i}(\boldsymbol{\mu};t^k) = \sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\sum_{q'=1}^{Q_f}\sum_{m'=1}^{M_f^q} \beta_m^{q'm'}(\boldsymbol{\mu})\ \beta_f^{q'm'}(\boldsymbol{\mu};t^k)\ \Phi^{qmiq'm'}_{N_{pr},fm}, \tag{3.161}$$

$$C^{mm}_{N_{pr}ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m}\sum_{m=1}^{M_m^q}\sum_{q'=1}^{Q_m}\sum_{m'=1}^{M_m^q} \beta_m^{qm}(\boldsymbol{\mu})\ \beta_m^{q'm'}(\boldsymbol{\mu})\ \Phi^{qmiq'm'i'}_{N_{pr},mm}, \tag{3.162}$$

où on a

$$\Phi^{qmq'm'}_{N_{pr},ff} = \big(\big(\Gamma^{qm}_{N_{pr}},\Gamma^{q'm'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qmiq'm'i'}_{N_{pr},am} = \big(\big(\Upsilon^{qmi}_{N_{pr}},\Lambda^{q'm'i'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{3.163}$$

$$\Phi^{qmiq'm'i'}_{N_{pr},aa} = \big(\big(\Upsilon^{qmi}_{N_{pr}},\Upsilon^{q'm'i'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qmiq'm'}_{N_{pr},fa} = \big(\big(\Upsilon^{qmi}_{N_{pr}},\Gamma^{q'm'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ , \tag{3.164}$$

$$\Phi^{qmiq'm'i'}_{N_{pr},mm} = \big(\big(\Lambda^{qmi}_{N_{pr}},\Lambda^{q'm'i'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ ,\ \Phi^{qmiq'm'}_{N_{pr},fm} = \big(\big(\Lambda^{qmi}_{N_{pr}},\Gamma^{q'm'}_{N_{pr}}\big)\big)_{a,\boldsymbol{\mu}_{ref}}\ . \tag{3.165}$$

Tournons-nous à présent vers la prise en compte de l'erreur engendrée par l'approximation des fonctions non-affines en utilisant la méthode EIM. En nous servant de (3.85) nous pouvons écrire, pour tout $v \in X_{\mathcal{N}}$ et $1 \leq k \leq K$,

$$\tilde{r}_{pr}\big(v;\boldsymbol{\mu};t^k;\underline{\zeta^{n_r}(\boldsymbol{\mu};\boldsymbol{x})} - \underline{\zeta^{n_r}_{M^{n_r}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{r}_{pr}\big(v;\boldsymbol{\mu};t^k;\epsilon_{M^1}(\boldsymbol{\mu};\boldsymbol{x}),...,\epsilon_{M^{nr}}(\boldsymbol{\mu};\boldsymbol{x})\big). \tag{3.166}$$

Le représentant de Riesz $\hat{e}^{eim}_{pr}(\boldsymbol{\mu},t^k)$ vérifie, pour tout $v \in X_{\mathcal{N}}$ et $1 \leq k \leq K$,

$$\begin{aligned}
\Big(\Big(\hat{e}^{eim}_{pr}(\boldsymbol{\mu},t^k),v\Big)\Big)_{a,\boldsymbol{\mu}_{ref}} \;=\; & \sum_{q=1}^{Q_f}\hat{\epsilon}_{M^q}(\boldsymbol{\mu})f^{q(M_f^q+1)}\big(v;\hat{\zeta}^{q(M_a^q+1)}(\boldsymbol{x})\big) \\
& -\sum_{q=1}^{Q_a}\sum_{i=1}^{N}\hat{\epsilon}_{M^q}(\boldsymbol{\mu})u_{Ni}\big(\boldsymbol{\mu};t^k\big)a^{q(M_a^q+1)}\big(\xi^{pr}_i,v;\hat{\zeta}^{q(M_a^q+1)}(\boldsymbol{x})\big) \\
& -\frac{1}{\Delta t}\sum_{i=1}^{N}\hat{\epsilon}_{M^q}(\boldsymbol{\mu})u_{Ni}(\boldsymbol{\mu};t^k)m^{q(M_m^q+1)}\big(\xi^{pr}_i,v;\hat{\zeta}^{q(M_m^q+1)}(\boldsymbol{x})\big) \\
& +\frac{1}{\Delta t}\sum_{i=1}^{N}\hat{\epsilon}_{M^q}(\boldsymbol{\mu})u_{Ni}(\boldsymbol{\mu};t^{k-1})m^{q(M_m^q+1)}\big(\xi^{pr}_i,v;\hat{\zeta}^{q(M_m^q+1)}(\boldsymbol{x})\big)
\end{aligned} \tag{3.167}$$

Ensuite, en utilisant le principe de superposition linéaire, nous pouvons écrire, pour $k = 1, ..., K$

$$\begin{aligned}\hat{e}_{pr}(\boldsymbol{\mu}, t^k) = & \sum_{q=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu}) \Gamma_{N_{pr}}^{q(M_f^q+1)} + \sum_{q=1}^{Q_a} \sum_{i=1}^{N} \hat{\epsilon}_{M^q}(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) \Upsilon_{N_{pr}}^{q(M_a^q+1)i} \\ & + \frac{1}{\Delta t} \sum_{q=1}^{Q_m} \sum_{i=1}^{N} \hat{\epsilon}_{M^q}(\boldsymbol{\mu}) \Big(u_{Ni}(\boldsymbol{\mu}; t^k) - u_{Ni}(\boldsymbol{\mu}; t^{k-1}) \Big) \Lambda_{N_{pr}}^{q(M_m^q+1)i},\end{aligned} \tag{3.168}$$

avec, pour tout $v \in X_{\mathcal{N}}$,

$$\Big(\Big(\Gamma_{N_{pr}}^{q(M_f^q+1)}, v \Big) \Big)_{a, \boldsymbol{\mu}_{ref}} = f^{q(M_f^q+1)}(v; \hat{\zeta}^{q(M_f^q+1)}(\boldsymbol{x})), \ 1 \leq q \leq Q_f, \tag{3.169}$$

ainsi que pour $1 \leq i \leq N$,

$$\Big(\Big(\Upsilon_{N_{pr}}^{q(M_a^q+1)i}, v \Big) \Big)_{a, \boldsymbol{\mu}_{ref}} = -a^{q(M_a^q+1)}\big(\xi_i^{pr}, v; \hat{\zeta}^{q(M_a^q+1)}(\boldsymbol{x}) \big), \ 1 \leq q \leq Q_a, \tag{3.170}$$

$$\Big(\Big(\Lambda_{N_{pr}}^{q(M_m^q+1)i}, v \Big) \Big)_{a, \boldsymbol{\mu}_{ref}} = -m^{q(M_m^q+1)}(\xi_i^{pr}, v; \hat{\zeta}^{q(M_f^q+1)}(\boldsymbol{x})), \ 1 \leq q \leq Q_m. \tag{3.171}$$

Nous avons donc

$$\begin{aligned}\epsilon_{N_{pr}^{eim}}(\boldsymbol{\mu}, t^k)^2 = & C_{N_{pr}^{eim}}^{ff}(\boldsymbol{\mu}; t^k) + 2 \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu}; t^k) C_{N_{pr}^{eim} i}^{fa}(\boldsymbol{\mu}; t^k) \\ & + \sum_{i=1}^{N} \sum_{i'=1}^{N} u_{Ni}(\boldsymbol{\mu}; t^k) u_{Ni'}(\boldsymbol{\mu}; t^k) C_{N_{pr}^{eim} ii'}^{aa}(\boldsymbol{\mu}) \\ & + \frac{2}{\Delta t} \sum_{i=1}^{N} \sum_{i'=1}^{N} \Big(u_{Ni}(\boldsymbol{\mu}; t^k) - u_{Ni}(\boldsymbol{\mu}; t^{k-1}) \Big) u_{Ni'}(\boldsymbol{\mu}; t^k) C_{N_{pr}^{eim} ii'}^{am}(\boldsymbol{\mu}) \\ & + \frac{1}{\Delta t^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \Big(u_{Ni}(\boldsymbol{\mu}; t^k) - u_{Ni}(\boldsymbol{\mu}; t^{k-1}) \Big) \Big(u_{Ni'}(\boldsymbol{\mu}; t^k) - u_{Ni'}(\boldsymbol{\mu}; t^{k-1}) \Big) C_{N_{pr}^{eim} ii'}^{mm}(\boldsymbol{\mu}) \\ & + \frac{2}{\Delta t} \sum_{i=1}^{N} \Big(u_{Ni}(\boldsymbol{\mu}; t^k) - u_{Ni}(\boldsymbol{\mu}; t^{k-1}) \Big) C_{N_{pr}^{eim} i}^{fm}(\boldsymbol{\mu}; t^k), \quad 1 \leq k \leq K,\end{aligned} \tag{3.172}$$

avec pour $1 \leq i, i' \leq N$:

$$C_{N_{pr}^{eim}}^{ff}(\boldsymbol{\mu}; t^k) = \sum_{q=1}^{Q_f} \sum_{q'=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu}) \ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu}) \ \Phi_{N_{pr}^{eim}, ff}^{q(M_f^q+1)q'(M_f^q+1)}, \tag{3.173}$$

$$C_{N_{pr}^{eim} i}^{fa}(\boldsymbol{\mu}; t^k) = \sum_{q=1}^{Q_a} \sum_{q'=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu}) \ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu}) \ \Phi_{N_{pr}^{eim}, fa}^{q(M_a^q+1)iq'(M_f^q+1)}, \tag{3.174}$$

$$C_{N_{pr}^{eim} ii'}^{aa}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{q'=1}^{Q_a} \hat{\epsilon}_{M^q}(\boldsymbol{\mu}) \ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu}) \ \Phi_{N_{pr}^{eim}, aa}^{q(M_a^q+1)iq'(M_a^q+1)i'}, \tag{3.175}$$

$$C^{am}_{N^{eim}_{pr} ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \sum_{q'=1}^{Q_m} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi^{q(M^q_a+1)iq'(M^q_m+1)i'}_{N^{eim}_{pr},am}, \tag{3.176}$$

$$C^{fm}_{N^{eim}_{pr} i}(\boldsymbol{\mu}; t^k) = \sum_{q'=1}^{Q_m} \sum_{q'=1}^{Q_f} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi^{q(M^q_m+1)iq'(M^q_f+1)}_{N^{eim}_{pr},fm}, \tag{3.177}$$

$$C^{mm}_{N^{eim}_{pr} ii'}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_m} \sum_{q'=1}^{Q_m} \hat{\epsilon}_{M^q}(\boldsymbol{\mu})\ \hat{\epsilon}_{M^{q'}}(\boldsymbol{\mu})\ \Phi^{q(M^q_m+1)iq'(M^q_m+1)i'}_{N^{eim}_{pr},mm}, \tag{3.178}$$

où on a

$$\Phi^{q(M^q_f+1)q'(M^q_f+1)}_{N^{eim}_{pr},ff} = \left(\left(\Gamma^{q(M^q_f+1)}_{N_{pr}}, \Gamma^{q'(M^q_f+1)}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.179}$$

$$\Phi^{q(M^q_a+1)iq'(M^q_m+1)i'}_{N^{eim}_{pr},am} = \left(\left(\Upsilon^{q(M^q_a+1)i}_{N_{pr}}, \Lambda^{q'(M^q_m+1)i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.180}$$

$$\Phi^{q(M^q_a+1)iq'(M^q_a+1)i'}_{N^{eim}_{pr},aa} = \left(\left(\Upsilon^{q(M^q_a+1)i}_{N_{pr}}, \Upsilon^{q'(M^q_a+1)i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.181}$$

$$\Phi^{q(M^q_a+1)iq'(M^q_f+1)}_{N^{eim}_{pr},fa} = \left(\left(\Upsilon^{q(M^q_a+1)i}_{N_{pr}}, \Gamma^{q'(M^q_f+1)}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.182}$$

$$\Phi^{q(M^q_m+1)iq'(M^q_m+1)i'}_{N^{eim}_{pr},mm} = \left(\left(\Lambda^{q(M^q_m+1)i}_{N_{pr}}, \Lambda^{q'(M^q_m+1)i'}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}, \tag{3.183}$$

$$\Phi^{q(M^q_m+1)iq'(M^q_f+1)}_{N^{eim}_{pr},fm} = \left(\left(\Lambda^{q(M^q_m+1)i}_{N_{pr}}, \Gamma^{q'(M^q_f+1)}_{N_{pr}}\right)\right)_{a,\boldsymbol{\mu}_{ref}}. \tag{3.184}$$

Pendant la phase *hors-ligne* il faut résoudre $(Q_f M^q_f + Q_a M^q_a N + Q_m M^q_m N)$ problèmes de type Poisson qui dépendent de la dimension FE : $\mathcal{N}$, voir (3.153)-(3.155). Intéressons-nous maintenant à la phase *en-ligne*. Supposons que nous connaissons déjà les coefficients $u_{N_i}(\boldsymbol{\mu}; t^k)$, $1 \leq i \leq N$, $1 \leq k \leq K$. Supposons également que les fonctions $\beta^{qm}_f(\boldsymbol{\mu}; t^k)$ pour $1 \leq q \leq Q_f$,$1 \leq m \leq M^q_f$, $\beta^{qm}_a(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_a$, $1 \leq m \leq M^q_a$ et $\beta^{qm}_m(\boldsymbol{\mu})$ pour $1 \leq q \leq Q_m$, $1 \leq m \leq M^q_m$ sont déjà évaluées avec $1 \leq k \leq K$. La complexité *en-ligne* de l'estimation d'erreur est en $\mathcal{O}(Q^2_a M^{q\,2}_a N^2 + Q^2_m M^{q\,2}_m N^2)$ opérations.

4 Problèmes elliptiques non-linéaires

Nous nous tournons maintenant vers les problèmes elliptiques dépendant de N_{na} fonctions non-affines en paramètres et qui dépendent de la solution $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ de l'EDP pour tout $\boldsymbol{\mu} \in \mathcal{D}$. Nous présentons ici comment est imbriquée l'approximation issue de la méthode EIM dans la construction de l'approximation base réduite. Cependant nous ne présentons pas dans cette section la méthode CRBM, en ce sens où nous traitons uniquement l'étape de prédiction. Nous n'évoquerons pas le problème dual ni la construction d'estimateurs d'erreur a posteriori. Le lecteur intéressé peut se reporter, entre autres, à [29] pour avoir plus de précisions sur la méthode CRBM appliquée aux problèmes non-linéaires nécessitant également l'intégration d'approximations issues de la méthode EIM.

4.1 Énoncé du problème général

Introduisons les espaces $E^{\zeta^q} = \mathcal{D} \times \mathbb{R}^d \times X_\mathcal{N}$, pour $q = 1, ..., N_{na}$. Posons maintenant $E^{\underline{\zeta^{N_{na}}}} = E^{\zeta^1} \times ... \times E^{\zeta^{N_{na}}}$. Pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous nous intéressons à l'évaluation d'une sortie $s_\mathcal{N}(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $u_\mathcal{N}(\boldsymbol{\mu}) \in X_\mathcal{N}$:

$$s_\mathcal{N}(\boldsymbol{\mu}) = \ell\Big(u_\mathcal{N}(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big), \tag{3.185}$$

pour un opérateur linéaire $\ell\big(.; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\big) : X_\mathcal{N} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na}}}} \to \mathbb{R}$ approprié qui dépend de N_{na} fonctions non-linéaires. Remarquons que contrairement au cas des problèmes non-affines, ici les fonctions ζ^i, $1 \leq i \leq N_{na}$, dépendent également de la solution $u_\mathcal{N}(\boldsymbol{\mu})$ de l'EDP. La formulation variationnelle de l'EDP consiste à chercher $u_\mathcal{N}(\boldsymbol{\mu}) \in X_\mathcal{N}$ tel que

$$g\Big(u_\mathcal{N}(\boldsymbol{\mu}), v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big) = 0, \ \forall v \in X_\mathcal{N}, \tag{3.186}$$

où $g\big(., ., \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\big) : X_\mathcal{N} \times X_\mathcal{N} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na}}}} \to \mathbb{R}$. La solution $u_\mathcal{N}(\boldsymbol{\mu})$ est donc la racine du résidu g. L'algorithme de Newton est utilisé pour résoudre (3.186). Une version basique en est présentée par l'algorithme 6. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour une solution

Algorithm 6 $u_\mathcal{N}(\boldsymbol{\mu}) = Newton\Big(g\big(u^k_\mathcal{N}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\big), j\big(\delta u^k_\mathcal{N}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\big), \text{valeur\_initiale}, tol, k_{max}\Big)$

k $\leftarrow 0$
$u^0_\mathcal{N}(\boldsymbol{\mu}) \leftarrow$ valeur_initiale
e $\leftarrow \infty$
while e $>$ tol et $k < k_{max}$ **do**
  résoudre :
    $j\Big(\delta u^k_\mathcal{N}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big)\Big[u^k_\mathcal{N}(\boldsymbol{\mu})\Big] = -g\Big(u^k_\mathcal{N}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big)$
  $u^{k+1}_\mathcal{N}(\boldsymbol{\mu}) \leftarrow u^k_\mathcal{N}(\boldsymbol{\mu}) + \delta u^k_\mathcal{N}(\boldsymbol{\mu})$
  $e \leftarrow \Big|\dfrac{u^{k+1}_\mathcal{N}(\boldsymbol{\mu}) - u^k_\mathcal{N}(\boldsymbol{\mu})}{u^k_\mathcal{N}(\boldsymbol{\mu})}\Big|$
  $k \leftarrow k + 1$
end while
$u_\mathcal{N}(\boldsymbol{\mu}) \leftarrow u^{k+1}_\mathcal{N}(\boldsymbol{\mu})$

initiale $u^0(\boldsymbol{\mu})$ donnée, à chaque itération k de l'algorithme – avec $1 \leq k \leq k_{max}$, où k_{max} est en entier strictement positif donné –, nous cherchons $\delta u^k_\mathcal{N}(\boldsymbol{\mu}) \in X_\mathcal{N}$ tel que pour tout $v \in X_\mathcal{N}$,

$$j\Big(\delta u^k_\mathcal{N}(\boldsymbol{\mu}); v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big)\Big[u^k_\mathcal{N}(\boldsymbol{\mu})\Big] = -g\Big(u^k_\mathcal{N}(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big), \tag{3.187}$$

où $\delta u^k_\mathcal{N}(\boldsymbol{\mu}) = u^{k+1}_\mathcal{N}(\boldsymbol{\mu}) - u^k_\mathcal{N}(\boldsymbol{\mu})$ et où $j : X_\mathcal{N} \times X_\mathcal{N} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na}}}} \to \mathbb{R}$ est la forme bilinéaire associée à la jacobienne, nous avons

$$j\Big(\delta u^k_\mathcal{N}(\boldsymbol{\mu}); v; \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big)\Big[u^k_\mathcal{N}(\boldsymbol{\mu})\Big] = \frac{\partial g\Big(u^k_\mathcal{N}(\boldsymbol{\mu}); \boldsymbol{\mu}; \underline{\zeta^{N_{na}}(\boldsymbol{\mu}; \boldsymbol{x}; u_\mathcal{N}(\boldsymbol{\mu}))}\Big)}{\partial u}. \tag{3.188}$$

Notons $\zeta^q(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu}))$, $1 \leq q \leq N_{na}$, la $q^{\text{ème}}$ fonction non-linéaire dont une approximation via la méthode EIM s'écrit, pour un entier strictement positif M^q donné, $\zeta^q\big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\big) \approx \zeta^q_{M^q}\big(\boldsymbol{\mu};\boldsymbol{x};u_{\mathcal{N}}(\boldsymbol{\mu})\big) = \sum_{m=1}^{M^q} \beta^{qm}\big(\boldsymbol{\mu};u_{\mathcal{N}}(\boldsymbol{\mu})\big)\ \hat{\zeta}^{qm}\big(\boldsymbol{x}\big)$. Afin de mettre en oeuvre une stratégie *hors-ligne/en-ligne* efficace nous allons maintenant approcher, via la méthode EIM, les termes pour lequels il n'est pas possible d'écrire une décomposition affine directement par inspection. Autrement dit nous supposons qu'il existe des entiers positifs Q_j, Q_r et Q_ℓ, avec $1 \leq Q_j, Q_r, Q_\ell \leq N_{na}$, de sorte que la méthode EIM détermine $\big(M_j^q\big)_{q=1,...,Q_j}$, $\big(M_g^q\big)_{q=1,...,Q_g}$ et $\big(M_\ell^q\big)_{q=1,...,Q_\ell}$ tels que, pour $n_j = \sum_{q=1}^{Q_j} M_j^q$, $n_g = \sum_{q=1}^{Q_g} M_g^q$ et $n_\ell = \sum_{q=1}^{Q_\ell} M_\ell^q$, ainsi que pour tout $\boldsymbol{\mu} \in \mathcal{D}$, les formes bilinéaire et linéaires $j(.,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x})})$, $g(u,.;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x};.)})$ et $\ell(.,\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x};.)})$ puissent être approchées de la manière suivante :

$$j\big(u,v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) \approx \tilde{j}\big(u,v;\boldsymbol{\mu};\underline{\zeta^{n_j}_{M^{n_j}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) = \sum_{q=1}^{Q_j}\sum_{m=1}^{M_j^q} \beta_j^{qm}(\boldsymbol{\mu};u) j^{qm}(u,v;\hat{\zeta}^{qm}(\boldsymbol{x})), \tag{3.189}$$

$$g\big(u,v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) \approx \tilde{g}\big(u,v;\boldsymbol{\mu};\underline{\zeta^{n_g}_{M^{n_g}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) = \sum_{q=1}^{Q_g}\sum_{m=1}^{M_g^q} \beta_g^{qm}(\boldsymbol{\mu};u) g^{qm}\big(u,v;\hat{\zeta}^{qm}(\boldsymbol{x})\big), \tag{3.190}$$

$$\ell\big(v;\boldsymbol{\mu};\underline{\zeta^{N_{na}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) \approx \tilde{\ell}\big(v;\boldsymbol{\mu};\underline{\zeta^{n_\ell}_{M^{n_\ell}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) = \sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu};u) \ell^{qm}\big(v;\hat{\zeta}^{qm}(\boldsymbol{x})\big), \tag{3.191}$$

pour tout $u \in X_{\mathcal{N}}$ et pour tout $v \in X_{\mathcal{N}}$, où $\beta_j^{qm} : \mathcal{D}\times X_{\mathcal{N}} \to \mathbb{R}$, $1 \leq q \leq Q_j$, $1 \leq m \leq M_j^q$, $\beta_g^{qm} : \mathcal{D} \times X_{\mathcal{N}} \to \mathbb{R}$, $1 \leq q \leq Q_g$, $1 \leq m \leq M_g^q$ et $\beta_\ell^{qm} : \mathcal{D} \times X_{\mathcal{N}} \to \mathbb{R}$, $1 \leq q \leq Q_\ell$, $1 \leq m \leq M_\ell^q$ sont déterminées par la méthode EIM. Posons $E^{\zeta^q}_{M^q} = \mathcal{D} \times \mathbb{R}^d \times X_{\mathcal{N}}$, pour $q = 1, ..., N_{na}$, ainsi que $E^{\underline{\zeta^{n_j}_{M^{n_j}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_j}_{M^{n_j}}}$, $E^{\underline{\zeta^{n_g}_{M^{n_g}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_g}_{M^{n_g}}}$, $E^{\underline{\zeta^{n_\ell}_{M^{n_\ell}}}} = E^{\zeta^1}_{M^1} \times ... \times E^{\zeta^{n_\ell}_{M^{n_\ell}}}$. Notons que nous avons $\tilde{j}\big(.,.;\boldsymbol{\mu};\underline{\zeta^{n_j}_{M^{n_j}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_j}_{M^{n_j}}}} \to \mathbb{R}$, $\tilde{g}\big(u,.;\boldsymbol{\mu};\underline{\zeta^{n_g}_{M^{n_g}}(\boldsymbol{\mu};\boldsymbol{x};u)}\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_g}_{M^{n_g}}}} \to \mathbb{R}$ et $\tilde{\ell}\big(.;\boldsymbol{\mu};\underline{\zeta^{n_\ell}_{M^{n_\ell}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : X_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{n_\ell}_{M^{n_\ell}}}} \to \mathbb{R}$. En nous servant des approximations décrites dans (3.189)-(3.191), pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour une solution initiale $u^0(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ donnée, à chaque itération k de l'algorithme – avec $1 \leq k \leq k_{max}$ –, nous cherchons $\delta u^k_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que pour tout $v \in X_{\mathcal{N}}$,

$$\tilde{j}\Big(\delta u^k_{\mathcal{N}}(\boldsymbol{\mu});v;\boldsymbol{\mu};\underline{\zeta^{n_j}_{M^{n_j}}(\boldsymbol{\mu};\boldsymbol{x};u^k_{\mathcal{N}}(\boldsymbol{\mu}))}\Big)\Big[u^k_{\mathcal{N}}(\boldsymbol{\mu})\Big] = -\tilde{g}\Big(u^k_{\mathcal{N}}(\boldsymbol{\mu}),v;\boldsymbol{\mu};\underline{\zeta^{n_g}_{M^{n_g}}(\boldsymbol{\mu};\boldsymbol{x};u^k_{\mathcal{N}}(\boldsymbol{\mu}))}\Big), \tag{3.192}$$

4.2 Méthode des bases réduites

Introduisons une séquence d'espaces d'approximation emboîtés $W_{N_{pr}}$ tels que $W_{1_{pr}} \subset W_{2_{pr}} \subset ... \subset W_{N_{max_{pr}}} \subset X_{\mathcal{N}}$, avec N_{max} un entier positif. Comme précédemment, $S_N = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N\}$ est un ensemble de N jeux de paramètres et S_N^u est l'ensemble qui va contenir, pour $\boldsymbol{\mu}$ dans S_N, la solution de (3.192) à l'itération finale. Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $(.,.)_{X_{\mathcal{N}}}$

– associé à l'espace $X_{\mathcal{N}}$ –, aux éléments de l'ensemble S_N^u pour avoir des fonctions de bases orthonormalisées ξ_n^{pr}, $1 \leq n \leq N$. L'espace d'approximation $W_{N_{pr}}$ est défini par

$$W_{N_{pr}} = span\Big\{\xi_n^{pr},\ 1 \leq n \leq N\Big\}. \tag{3.193}$$

Quant à la solution réduite $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$, elle s'écrit

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu})\ \xi_i^{pr}. \tag{3.194}$$

Définissons l'incrément, à l'itération k de l'algorithme de Newton, comme suit

$$\delta u_N^k(\boldsymbol{\mu}) = u_N^{k+1}(\boldsymbol{\mu}) - u_N^k(\boldsymbol{\mu}), \tag{3.195}$$

nous avons donc

$$\delta u_N^k(\boldsymbol{\mu}) = \sum_{i=1}^{N} \delta u_{N_i}^k(\boldsymbol{\mu}) = \sum_{i=1}^{N} \Big(u_{N_i}^{k+1}(\boldsymbol{\mu}) - u_{N_i}^k(\boldsymbol{\mu})\Big)\xi_i^{pr}. \tag{3.196}$$

Nous allons maintenant utiliser les approximations décrites dans (3.189)-(3.191) pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \xi_n^{pr}$, $n = 1, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour une solution initiale $u_N^0(\boldsymbol{\mu})$ donnée, à chaque itération k de l'algorithme de Newton – avec $1 \leq k \leq k_{max}$ –, nous cherchons $\delta u_N^k(\boldsymbol{\mu}) \in W_N$ tel que

$$\begin{aligned}
&\sum_{i=1}^{N} \left(\sum_{q=1}^{Q_j} \sum_{m=1}^{M_j^q} \beta_j^{qm}\Big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\Big)\ j^{qm}\Big(\xi_n^{pr}, \xi_i^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big)\right)\ \delta u_{N_i}^k(\boldsymbol{\mu}) \\
&= -\sum_{i=1}^{N} \left(\sum_{q=1}^{Q_g} \sum_{m=1}^{M_g^q} \beta_g^{qm}\Big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\Big)\ g^{qm}\Big(u_N^k(\boldsymbol{\mu}), \xi_n^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big)\right)
\end{aligned} \tag{3.197}$$

qui peut être également écrit sous la forme matricielle :

$$\left(\sum_{q=1}^{Q_j} \sum_{m=1}^{M_j^q} \beta_j^{qm}(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})) J_N^{qm} \right) \delta u_N^k(\boldsymbol{\mu}) = -\left(\sum_{q=1}^{Q_g} \sum_{m=1}^{M_g^q} \beta_g^{qm}(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})) G_N^{qm} \right), \tag{3.198}$$

avec

$$\Big(J_N^{qm}\Big)_{in} = j^{qm}\Big(\xi_n^{pr}, \xi_i^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \quad \text{et} \quad \Big(G_N^{qm}\Big)_n = g^{qm}\Big(u_N^k(\boldsymbol{\mu}), \xi_n^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x})\Big). \tag{3.199}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}\Big(\boldsymbol{\mu}\Big) \ell^{qm}\Big(u_N(\boldsymbol{\mu}); \hat{\zeta}^{qm}(\boldsymbol{x})\Big) \right),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.200}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}\Big(\boldsymbol{\mu}\Big) L_N^{qm\ T} \right) u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.201}$$

avec $(L_N^{qm})_i = \ell^{qm}(\xi_i^{pr}; \hat{\zeta}^{qm}(\boldsymbol{x}))$. La décomposition *hors-ligne/en-ligne* est maintenant évidente. Les fonctions de base ξ_i^{pr}, $1 \leq i \leq N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible la construction des matrices $J_N^{qm} \in \mathbb{R}^{N\times N}$, $1 \leq q \leq Q_j$, $1 \leq m \leq M_j^q$ et des vecteurs $G_N^{qm} \in \mathbb{R}^N$, $1 \leq q \leq Q_g$, $1 \leq m \leq M_g^q$ et $L_N^{qm} \in \mathbb{R}^N$, $1 \leq q \leq Q_\ell$, $1 \leq m \leq M_\ell^q$. À l'étape *en-ligne*, pour chaque $\boldsymbol{\mu} \in \mathcal{D}$ donné, à la $k^{\text{ème}}$ itération de l'algorithme de Newton on assemble la matrice $J_N\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big) = \sum_{q=1}^{Q_j}\sum_{m=1}^{M_j^q} \beta_j^{qm}\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big) J_N^{qm}$, ainsi que le vecteur $G_N\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big) = \sum_{q=1}^{Q_g}\sum_{m=1}^{M_g^q} \beta_g^{qm}\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big) G_N^{qm}$ puis nous résolvons

$$J_N\Big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\Big)\ \delta u_N^k(\boldsymbol{\mu}) = -G_N\Big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\Big). \tag{3.202}$$

Une fois que l'algorithme de Newton a convergé, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, nous assemblons le vecteur $L_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_\ell^q} \beta_\ell^{qm}(\boldsymbol{\mu}) L_N^{qm}$, et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = L_N^T(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{3.203}$$

La complexité de la partie *en-ligne* est indépendante de la (grande) dimension $\mathcal{N}$. Elle est étudiée en détail sur un exemple dans le chapitre 11 (annexe).

5 Résultats numériques

Nous illustrons ici l'utilisation de d'approximations obtenues via la méthode EIM dans le processus de construction d'approximations de type base réduite. Nous nous intéressons à un problème de diffusion, linéaire, non-affine, introduit dans [56].

Il s'agit d'un problème elliptique linéaire non-affine en 2D. Soit Ω le carré unité défini par $\Omega \equiv]0,1[^2 \in \mathbb{R}^2$. L'espace des paramètres est défini par $\mathcal{D} \equiv [-1,-0.01]^2$. Considérons le problème suivant : chercher $u_\mathcal{N}(\boldsymbol{\mu}) \in X_\mathcal{N} \equiv H_0^1(\Omega)$ tel que pour tout $\boldsymbol{\mu} \equiv (\mu^0, \mu^1) \in \mathcal{D}$ et pour tout $\boldsymbol{x} \equiv \big(x^0, x^1\big) \in \Omega$,

$$a_0\Big(u_\mathcal{N}(\boldsymbol{\mu}), v\Big) + a_1\Big(u_\mathcal{N}(\boldsymbol{\mu}), v, \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big) = f\Big(v; \boldsymbol{\mu}\Big),\ \forall v \in X_\mathcal{N}, \tag{3.204}$$

avec

$$a_0\Big(u_\mathcal{N}(\boldsymbol{\mu}), v\Big) = \int_\Omega \nabla u_\mathcal{N}(\boldsymbol{\mu}) \cdot \nabla v,\ a_1\Big(u_\mathcal{N}(\boldsymbol{\mu}), v, \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big) = \int_\Omega \zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}\Big)\ u\ v, \tag{3.205}$$

$$f\Big(v, \zeta(\boldsymbol{\mu}; \boldsymbol{x})\Big) = \int_\Omega \zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}\Big)\ v, \tag{3.206}$$

et où

$$\zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}\Big) = \frac{1}{\sqrt{(x^0-\mu^0)^2 + (x^1-\mu^1)^2}}, \tag{3.207}$$

Nous sommes intéressés par l'évaluation de la sortie du modèle $s_\mathcal{N}(\boldsymbol{\mu}) = \ell(u_\mathcal{N}(\boldsymbol{\mu}))$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, avec

$$\ell\Big(v\Big) = \int_\Omega v. \tag{3.208}$$

Prenons une discrétisation FE $\mathbb{P}_1$ avec 1 500 degrés de liberté et illustrons l'impact des paramètres μ^0 et μ^1 sur la solution. Les figures 3.1 et 3.2 illustrent la solution de (3.204) lorsque les paramètres sont fixés respectivement au minimum et au maximum. Plus les paramètres se rapprochent de $\mu^0 = \mu^1 = -0.01$, plus le pic se déplace en direction du coin situé en $(0,0)$ de Ω où se trouve une singularité, développant ainsi un phénomène similaire à une couche limite au voisinage de ce coin.

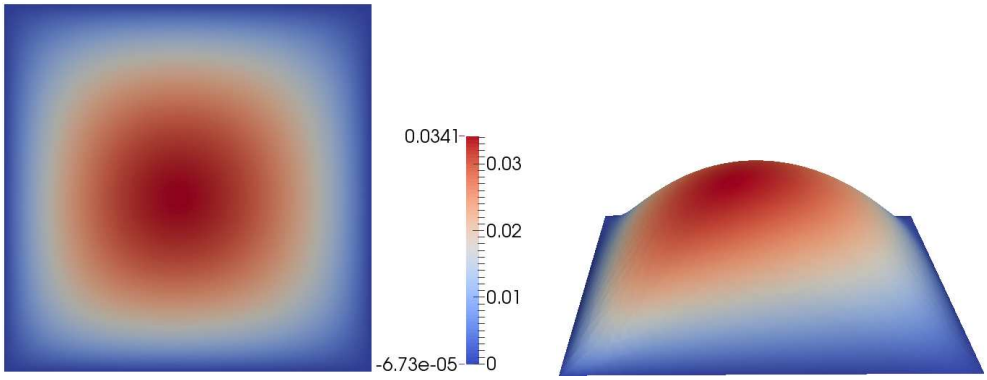


FIGURE 3.1 – À gauche, vue 2D classique. À droite, le domaine est déformé suivant de la valeur du champ solution. Dans les deux cas $\mu^0 = \mu^1 = -1$.

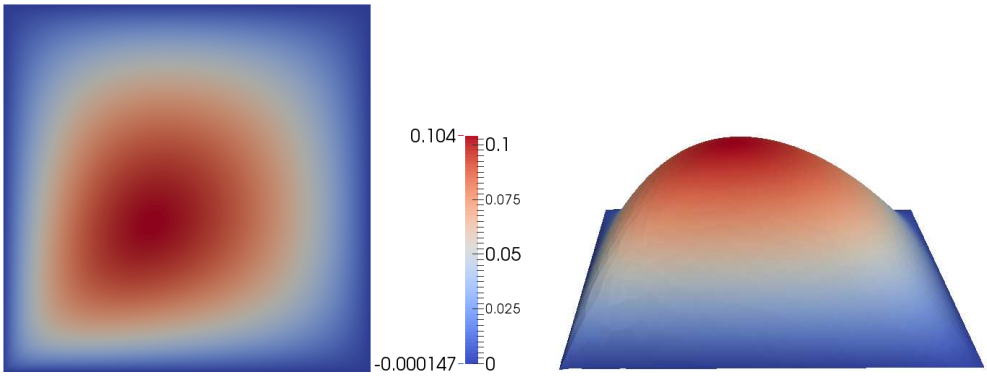


FIGURE 3.2 – À gauche, vue 2D classique. À droite, le domaine est déformé suivant de la valeur du champ solution. Dans les deux cas $\mu^0 = \mu^1 = -0.01$.

5.1 Convergence

Afin de pouvoir appliquer une stratégie *hors-ligne/en-ligne* efficace, le problème que nous traitons nécessite d'avoir une approximation ζ_M de la fonction non-affine ζ – voir (3.5) –. Rappelons que l'indice M indique le nombre de termes utilisés par la méthode EIM pour générer l'approximation ζ_M. Nous présentons ici la convergence de l'approximation – via la méthode CRBM – de la solution du problème primal, dual, ainsi que de la sortie en fonction de la qualité de l'approximation ζ_M. Introduisons les notations suivantes :

$$e_N^{pr,rel}(\zeta_M) = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|\left|u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}}{\left|\left|\left|u_{\mathcal{N}}(\boldsymbol{\mu})\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}}, \tag{3.209}$$

$$e_N^{du,rel}(\zeta_M) = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|\left|\Psi_{\mathcal{N}}(\boldsymbol{\mu}) - \Psi_N(\boldsymbol{\mu})\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}}{\left|\left|\left|\Psi_{\mathcal{N}}(\boldsymbol{\mu})\right|\right|\right|_{a,\boldsymbol{\mu}_{ref}}}, \tag{3.210}$$

$$e_N^{s,rel}(\zeta_M) = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})\right|}{\left|s_{\mathcal{N}}(\boldsymbol{\mu})\right|}, \tag{3.211}$$

où Ξ_{train} est un échantillon de 1 000 vecteurs de paramètres sélectionnés de manière aléatoire dans l'espace $\mathcal{D}$. Les quantités $e_N^{pr,rel}(\zeta_M)$, $e_N^{du,rel}(\zeta_M)$ et $e_N^{s,rel}(\zeta_M)$ désignent le maximum de l'erreur relative commise – respectivement sur la solution du problème primal, dual et la sortie – lorsque nous parcourons l'échantillon Ξ_{train} et que ζ_M est utilisée pour approcher la fonnction non-affine ζ. Le jeu de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$ qui a été choisi pour ce modèle est $\boldsymbol{\mu}_{ref} = (-1, -1)$.

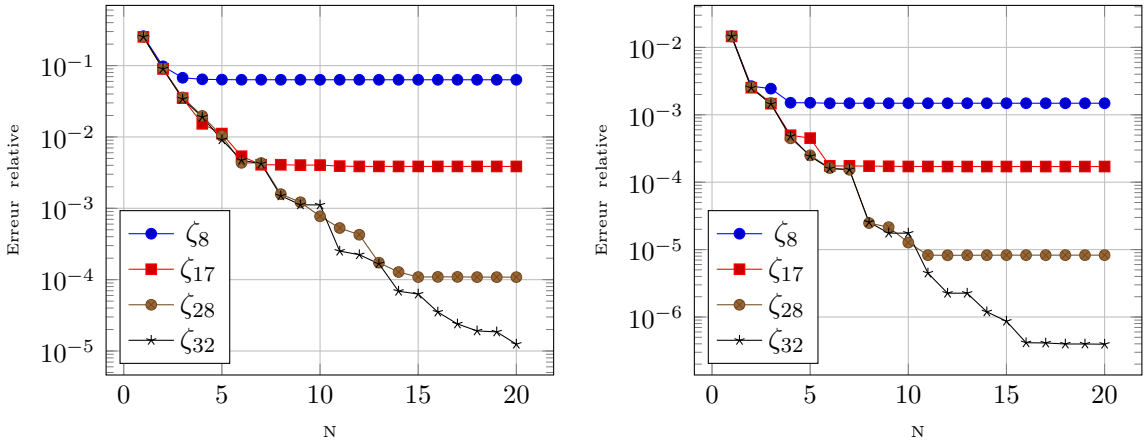


FIGURE 3.3 – Évolution des quantités $e_N^{pr,rel}(\zeta_M)$ (à gauche) et $e_N^{du,rel}(\zeta_M)$ (à droite) en fonction du nombre d'éléments dans la base réduite et de la qualité de l'approximation ζ_M, pour $M = 8, 17, 28$ et 32. $Card(\Xi_{train})$ =1 000 et $\mathcal{N} = 1\ 500$.

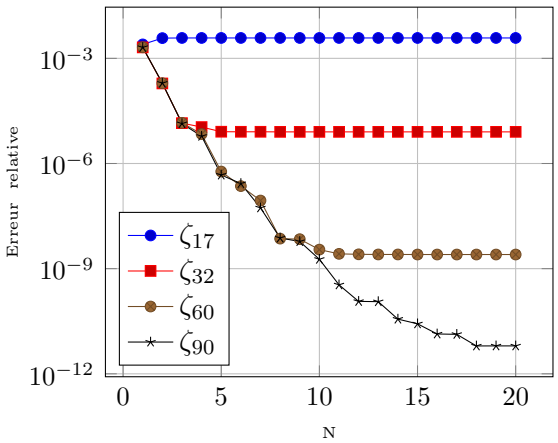


FIGURE 3.4 – Évolution de la quantité $e_N^{s,rel}(\zeta_M)$ en fonction du nombre d'éléments dans la base réduite et de la qualité de l'approximation ζ_M, pour $M = 17, 32, 60$ et 90. $Card(\Xi_{train})$ =1 000 et $\mathcal{N} = 1\ 500$.

Étudions également la qualité de l'approximation de ζ par ζ_M. Définissons

$$e_M^{rel,\zeta} = \max_{\boldsymbol{\mu} \in \Xi'_{train}} \frac{\left|\left|\zeta\left(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\right) - \zeta_M\left(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\right)\right|\right|_{L^\infty(\Omega)}}{\left|\left|\zeta\left(\boldsymbol{\mu}; .; u_{\mathcal{N}}(\boldsymbol{\mu})\right)\right|\right|} \tag{3.212}$$

où Ξ'_{train} est un échantillon de 1 000 vecteurs de paramètres sélectionnés de manière

aléatoire dans l'espace $\mathcal{D}$. La quantité $e_M^{rel,\zeta}$, désigne le maximum de l'erreur relative commise en utilisant l'approximation ζ_M lorsque nous parcourons l'échantillon Ξ'_{train}.

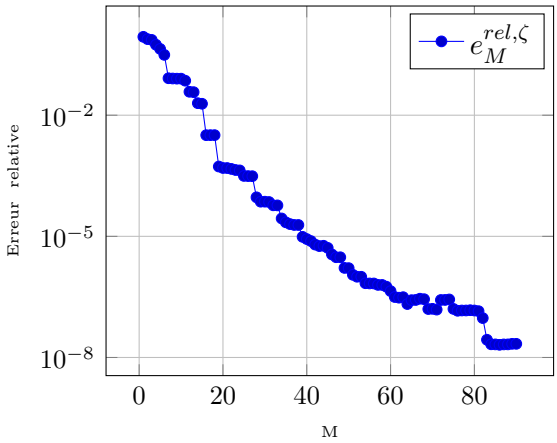


FIGURE 3.5 – Qualité de l'approximation ζ_M en fonction de M. $Card(\Xi'_{train}) = 1\ 000$ et $\mathcal{N} = 1\ 500$.

Estimation d'erreur a posteriori

Commençons par introduire, pour $N = 1, ..., N_{max}$, l'estimation de l'erreur relative commise sur la sortie

$$\Delta_N^{s,rel}(\boldsymbol{\mu}) = \frac{\Delta_N^s(\boldsymbol{\mu})}{s_{\mathcal{N}}(\boldsymbol{\mu})}, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{3.213}$$

ainsi que le maximum de $\Delta_N^{s,rel}(\boldsymbol{\mu})$ lorsque nous parcourons l'échantillon Ξ_{train},

$$\Delta_{N,max}^{s,rel} = \max_{\boldsymbol{\mu} \in \Xi_{train}} \Delta_N^{s,rel}(\boldsymbol{\mu}), \quad 1 \leq N \leq N_{max}. \tag{3.214}$$

Afin de quantifier la qualité de l'estimation d'erreur, nous introduisons l'indicateur d'efficacité pour tout $\boldsymbol{\mu} \in \mathcal{D}$, défini par

$$\eta_N^{s,rel}(\boldsymbol{\mu}) = \frac{\Delta_N^{s,rel}(\boldsymbol{\mu})}{e_N^{s,rel}}, \quad 1 \leq N \leq N_{max}, \tag{3.215}$$

Nous définissons également l'efficacité moyenne $\eta_{N,moy}^{s,rel}$, pour $N = 1, ..., N_{max}$ par

$$\eta_{N,moy}^{s,rel} = \frac{1}{Card(\Xi_{train})} \sum_{\boldsymbol{\mu} \in \Xi_{train}} \eta_N^{s,rel}(\boldsymbol{\mu}). \tag{3.216}$$

L'indicateur $\eta_{N,moy}^{s,rel}$ reflète le comportement moyen attendu de l'estimation d'erreur. Le tableau 3.1 montre l'évolution des quantités $e_N^{s,rel}$, $\Delta_{N,max}^{s,rel}$, et $\eta_{N,moy}^{s,rel}$ en fonction de N. La fonction non-affine ζ est approchée par ζ_M avec $M = 90$.

N	$e_N^{s,rel}$	$\Delta_{N,max}^{s,rel}$	$\eta_{N,moy}^{s,rel}$
5	4.69562×10^{-7}	1.8898×10^{-6}	9.4034
10	1.8548×10^{-9}	1.4552×10^{-8}	27.7708
15	2.6693×10^{-11}	4.9849×10^{-11}	6.0477
20	6.2613×10^{-12}	7.3573×10^{-12}	1.59386

TABLE 3.1 – Estimations de l'erreur sur la sortie et efficacités pour le problème de diffusion non-affine. L'approximation ζ_M est utilisée avec $M = 90$. $Card(\Xi_{train})$ =1 000 et $\mathcal{N} = 1$ 500.

Temps de calcul *en-ligne*

Considérons un entier N et un $\boldsymbol{\mu}$ donnés, tels que $1 \leq N \leq N_{max}$ et $\boldsymbol{\mu} \in \mathcal{D}$. Nous nous proposons de quantifier le gain de temps de calcul pour l'évaluation de la sortie $s_N(\boldsymbol{\mu})$ certifiée (avec une estimation de l'erreur) en comparaison de $s_{\mathcal{N}}(\boldsymbol{\mu})$. Le tableau 3.2 contient le temps de calcul moyen – sur 1 000 simulations – nécessaire pour évaluer la sortie $s_N(\boldsymbol{\mu})$ ainsi que l'estimation d'erreur associée $\Delta_N^s(\boldsymbol{\mu})$, en fonction de la dimension N de la base réduite. La fonction non-affine ζ est approchée par ζ_M avec $M = 90$. Les temps de calculs sont normalisés par rapport au temps moyen mis pour évaluer la sortie $s_{\mathcal{N}}(\boldsymbol{\mu})$ en utilisant la méthode FEM. Nous remarquons que l'évaluation $s_N(\boldsymbol{\mu})$ ne fait gagner qu'un facteur

N	$s_N(\boldsymbol{\mu})$	$\Delta_N^s(\boldsymbol{\mu})$	$s_{\mathcal{N}}(\boldsymbol{\mu})$
5	5.4832×10^{-1}	1.3785×10^{0}	1
10	5.5043×10^{-1}	1.4056×10^{0}	1
15	5.5405×10^{-1}	1.4500×10^{0}	1
20	5.5850×10^{-1}	1.4907×10^{0}	1

TABLE 3.2 – Temps de calcul *en-ligne* pour évaluer la sortie et l'estimation d'erreur associée. Ces temps sont normalisés par rapport au temps de calcul moyen de $s_{\mathcal{N}}(\boldsymbol{\mu})$ – sur 1 000 simulations – avec $\mathcal{N} = 1$ 500.

deux par rapport à l'évaluation de $s_{\mathcal{N}}(\boldsymbol{\mu})$. D'autre part le temps mis pour calculer une estimation d'erreur est supérieur à l'évaluation de $s_{\mathcal{N}}(\boldsymbol{\mu})$. Cela s'explique par le fait qu'en théorie la partie *en-ligne* ne dépend pas de la dimension FE : $\mathcal{N}$, mais que dans la pratique nous n'avons pas encore supprimé totalement cette dépendance. En effet, dans l'étape *en-ligne* de la méthode EIM , pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous utilisons l'approximation élément fini de la solution, $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$, pour l'évaluation de la fonction $\zeta\big(\boldsymbol{\mu}, ., u_{\mathcal{N}}(\boldsymbol{\mu})\big)$, alors que nous devrions utiliser l'approximation base réduite, $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$, et donc évaluer $\zeta\big(\boldsymbol{\mu}, ., u_N(\boldsymbol{\mu})\big)$ dans l'équation (3.14). Le chapitre 7 explique en détails, du point de vue informatique, pourquoi nous utilisons l'évaluation de $\zeta\big(\boldsymbol{\mu}, ., u_{\mathcal{N}}(\boldsymbol{\mu})\big)$, et donc pourquoi nous dépendons encore de la dimension FE. Cette dépendance en $\mathcal{N}$ est mise en évidence par les résultats du tableau qui contient les temps de calculs des mêmes quantités que le tableau 3.2 mais en prenant cette fois $\mathcal{N} = 4$ 200. Si le temps de calcul de $s_N(\boldsymbol{\mu})$ et $\Delta_N^s(\boldsymbol{\mu})$ ne dépend pas de $\mathcal{N}$ alors nous devrions avoir un facteur proche de 2.3 entre les valeurs des tableaux 3.2 et 3.3. Or ce n'est pas le cas.

N	$s_N(\boldsymbol{\mu})$	$\Delta_N^s(\boldsymbol{\mu})$	$s_{\mathcal{N}}(\boldsymbol{\mu})$
5	4.6329×10^{-1}	1.0155×10^{0}	1
10	4.6417×10^{-1}	1.030×10^{0}	1
15	4.6557×10^{-1}	1.051×10^{0}	1
20	4.6769×10^{-1}	1.072×10^{0}	1

TABLE 3.3 – Temps de calcul *en-ligne* pour évaluer la sortie et l'estimation d'erreur associée. Ces temps sont normalisés par rapport au temps de calcul moyen de $s_{\mathcal{N}}(\boldsymbol{\mu})$ – sur 1 000 simulations – avec $\mathcal{N} = 4\ 200$.

Résumé

Nous avons vu dans les sections 2 et 3 comment appliquer la méthode CRBM à des problèmes linéaires, respectivement elliptiques et paraboliques qui n'admettent pas une dépendance affine en paramètres. Les termes qui ne s'expriment pas de manière affine en $\boldsymbol{\mu}$ ont donc été approchés par une expansion donnée par la méthode EIM. Les sections 2.3 et 3.3 ont montré comment ces approximations doivent être prises en compte lors du calcul de l'estimation d'erreur a posteriori. L'application de la méthode RBM dans le cas de problèmes non-linéaire a été explicitée dans la section 4. Enfin, différents résultats numériques obtenus à l'aide du framework base réduite (décrit dans la deuxième partie de ce manuscrit) sur un modèle 2D linéaire de diffusion non-affine en paramètres ont été exposés dans la dernière section.

Chapitre 4

Traitement des problèmes elliptiques multi-physiques

Dans le but de modéliser des phénomènes de plus en plus complexes, beaucoup de problèmes aujourd'hui font intervenir plusieurs physiques. C'est notamment le cas lors de la modélisation d'aimants à haut champ de type résistifs où interviennent des phénomènes éléctro-thermiques, éléctro-thermo-mécaniques, magnétostatiques ou encore thermo-hydrauliques. Plus de détails sur la modélisation de ce type d'aimants sont donnés dans le chapitre 10, consacré à l'aspect éléctro-thermique (couplage potentiel éléctrique et température).

Nous présenterons dans ce chapitre la méthode des bases réduites appliquée aux problèmes multi-physiques. La particularité de ces problèmes est qu'ils sont composés de $Neqs$ équations, avec $Neqs > 0$. Notons que pour que le problème soit considéré comme multi-physique il faut avoir $Neqs > 1$. Cependant la méthodologie décrite dans ce chapitre reste valable dans le cas $Neqs = 1$, dans ce cas nous retombons exactement sur la méthodologie décrite dans le chapitre 2. Pour plus de simplicité, nous allons considérer dans ce chapitre que les différentes inconnues associées au problème multi-physique sont des champs scalaires – par exemple la température ou le potentiel éléctrique –. Dans le cas où nous devons traiter des champs vectoriels – par exemple le champ magnétique –, la méthodologie reste identique. De plus nous allons également supposer que les problèmes sont elliptiques, mais la méthodologie peut être facilement étendue au cas des problèmes paraboliques. Nous ne détaillerons pas non plus ici les aspects liés à la mise en place des estimateurs d'erreur ou à la résolution du problème dual.

Considérons Ω un domaine borné, régulier, dans $\mathbb{R}^d$, pour $d = 1, ..., 3$. Introduisons maintenant les espaces de fonctions $X_{\mathcal{N}_i}$ – de (grande) dimension $\mathcal{N}_i$ – pour $i = 1, ..., Neqs$, tels que $H_0^1(\Omega) \subset X_{\mathcal{N}_i} \subset H^1(\Omega)$. Soit $\boldsymbol{X}_{\mathcal{N}}$ un espace produit de dimension $\mathcal{N}$ défini par

$$\boldsymbol{X}_{\mathcal{N}} = X_{\mathcal{N}_1} \times X_{\mathcal{N}_2} \times \ldots \times X_{\mathcal{N}_{Nequs}}, \text{ avec } \mathcal{N} = \sum_{i=1}^{Neqs} \mathcal{N}_i. \tag{4.1}$$

Notons $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ l'inconnue de notre problème multi-physique. La question qui se pose alors est comment construire une base de dimension N – avec $N << \mathcal{N}$ –, sur laquelle un système de $Neqs$ équations va être projeté, afin d'obtenir une approximation de $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu})$ dans un espace de dimension N ? Dans ce chapitre nous allons répondre à cette question

en considérant d'abord les problèmes qui admettent une dépendance affine en paramètres, puis nous nous pencherons sur le cas des problèmes pour lesquels cette dépendance n'est plus affine. Enfin nous terminerons par l'étude des problèmes non-linéaires.

1 Problèmes linéaires affines

Nous nous intéressons ici aux problèmes elliptiques linéaires qui admettent une dépendance affine en paramètres. Considérons $Neqs$ formes bilinéaires $a_i : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$ et linéaires $f_i : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, pour $i = 1, ..., Neqs$. Nous supposons ici que les a_i, pour $i = 1, ..., Neqs$, sont des formes bilinéaires symétriques, définies positives. Nous pouvons donc introduire les produits scalaires $\big((.,.)\big)_{a_i,\boldsymbol{\mu}}$ définis, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((\boldsymbol{w}, \boldsymbol{z})\big)_{a_i,\boldsymbol{\mu}} \equiv a_i\big(\boldsymbol{w}, \boldsymbol{z}; \boldsymbol{\mu}\big), \ \forall \boldsymbol{w}, \boldsymbol{z} \in \boldsymbol{X}_{\mathcal{N}}, \ 1 \leq i \leq Neqs. \tag{4.2}$$

Notons que les normes associées à ces produits scalaires sont

$$\big|\big|\big|\boldsymbol{w}\big|\big|\big|_{a_i,\boldsymbol{\mu}} \equiv \sqrt{\big((\boldsymbol{w}, \boldsymbol{w})\big)_{a_i,\boldsymbol{\mu}}}, \ \forall \boldsymbol{w} \in \boldsymbol{X}_{\mathcal{N}}, \ 1 \leq i \leq Neqs. \tag{4.3}$$

De plus, nous supposons que les a_i, $1 \leq i \leq Neqs$, sont continues et coercives. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, il existe des constantes de continuité $\gamma^{a_i,\mathcal{N}}(\boldsymbol{\mu})$ telles que

$$\gamma^{a_i,\mathcal{N}}(\boldsymbol{\mu}) \equiv \sup_{\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}} \frac{a_i(\boldsymbol{v}, \boldsymbol{v}; \boldsymbol{\mu})}{\big|\big|\big|\boldsymbol{v}\big|\big|\big|^2_{a_i,\boldsymbol{\mu}_{ref}}} < \infty, \tag{4.4}$$

ainsi que des constantes de coercivité $\alpha^{a_i,\mathcal{N}}(\boldsymbol{\mu})$ telles que

$$\alpha^{a_i,\mathcal{N}}(\boldsymbol{\mu}) \equiv \inf_{\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}} \frac{a_i(\boldsymbol{v}, \boldsymbol{v}; \boldsymbol{\mu})}{\big|\big|\big|\boldsymbol{v}\big|\big|\big|^2_{a_i,\boldsymbol{\mu}_{ref}}} > 0. \tag{4.5}$$

Les formes bilinéaires a_i et linéaires f_i, $1 \leq i \leq Neqs$, sont les contributions de chaque équation au problème multi-physique. Afin d'évaluer la sortie du modèle, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous utilisons les opérateurs appropriés $\ell_i : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, pour $i = 1, ..., Neqs$. Nous faisons l'hypothèse que a_i, f_i et ℓ_i, pour $i = 1, ..., Neqs$, admettent une dépendance affine en paramètres. Nous supposons donc qu'il existe des entiers positifs Q_{a_i}, Q_{f_i} et Q_{ℓ_i}, pour $i = 1, ..., Neqs$, tels qu'il soit possible d'écrire

$$\begin{cases} a_i\big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}\big) = \displaystyle\sum_{q=1}^{Q_{a_i}} \theta_{a_i}^{iq}\big(\boldsymbol{\mu}\big) \ a_i^{iq}\big(\boldsymbol{u}, \boldsymbol{v}\big), \ \forall \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ f_i\big(\boldsymbol{v}; \boldsymbol{\mu}\big) = \displaystyle\sum_{q=1}^{Q_{f_i}} \theta_{f_i}^{iq}\big(\boldsymbol{\mu}\big) \ f_i^{iq}\big(\boldsymbol{v}\big), \ \forall \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \ \forall \boldsymbol{\mu} \in \mathcal{D}, \\ \ell_i\big(\boldsymbol{v}; \boldsymbol{\mu}\big) = \displaystyle\sum_{q=1}^{Q_{\ell_i}} \theta_{\ell_i}^{iq}\big(\boldsymbol{\mu}\big) \ \ell_i^{iq}\big(\boldsymbol{v}\big), \ \forall \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \ \forall \boldsymbol{\mu} \in \mathcal{D}, \end{cases} \tag{4.6}$$

où, pour $i = 1, ..., Neqs$, $\theta_{a_i}^{iq} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_{a_i}$, $\theta_{f_i}^{iq} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_f$ et $\theta_{\ell_i}^{iq} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_\ell$ sont des fonctions qui dépendent de $\boldsymbol{\mu}$.

Introduisons maintenant les formes bilinéaires $a : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$ et linéaires $f : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$, $\ell : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$ définies par

$$a\Big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}\Big) = \sum_{e=1}^{Neqs} a_e\Big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}\Big), \ \forall \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{4.7}$$

$$f\Big(\boldsymbol{v}; \boldsymbol{\mu}\Big) = \sum_{e=1}^{Neqs} f_e\Big(\boldsymbol{v}; \boldsymbol{\mu}\Big), \ \forall \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{4.8}$$

et

$$\ell\Big(\boldsymbol{v}; \boldsymbol{\mu}\Big) = \sum_{e=1}^{Neqs} \ell_e\Big(\boldsymbol{v}; \boldsymbol{\mu}\Big), \ \forall \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{4.9}$$

Pour un jeu de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous nous intéressons à l'évaluation d'une sortie $s(\boldsymbol{\mu}) \in \mathbb{R}$ définie par

$$s_{\mathcal{N}}(\boldsymbol{\mu}) = \ell\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}\Big), \tag{4.10}$$

avec $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ vérifiant

$$a\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\Big) = f\Big(\boldsymbol{v}; \boldsymbol{\mu}\Big), \ \forall \boldsymbol{v} \in X_{\mathcal{N}}, \tag{4.11}$$

Introduisons maintenant une séquence d'espaces d'approximation emboîtés $\boldsymbol{W}_{N_{pr}}$ tels que $\boldsymbol{W}_{1_{pr}} \subset \boldsymbol{W}_{2_{pr}} \subset ... \subset \boldsymbol{W}_{N_{max_{pr}}} \subset \boldsymbol{X}_{\mathcal{N}}$, avec N_{max} un entier positif. L'espace $\boldsymbol{W}_{N_{pr}}$ est de dimension N avec $N << \mathcal{N}$. Soit $S_N = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N\}$ un ensemble de N jeux de paramètres sélectionnés à l'aide de l'algorithme glouton. Considérons également $S_N^{\boldsymbol{u}}$ l'ensemble qui va contenir, pour chaque jeu de paramètres $\boldsymbol{\mu}$ dans S_N, la solution de (4.11). Cet ensemble s'écrit

$$S_N^{\boldsymbol{u}} = \Big\{\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in S_N\Big\}. \tag{4.12}$$

Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $(.,.)_{\boldsymbol{X}_{\mathcal{N}}}$ – associé à l'espace $\boldsymbol{X}_{\mathcal{N}}$ –, aux éléments de l'ensemble $S_N^{\boldsymbol{u}}$ pour avoir des fonctions de bases orthonormalisées $\boldsymbol{\xi}_n^{pr}$, $1 \leq n \leq N$. L'espace d'approximation $\boldsymbol{W}_{N_{pr}}$ est défini par

$$\boldsymbol{W}_{N_{pr}} = span\Big\{\boldsymbol{\xi}_n^{pr}, \ 1 \leq n \leq N\Big\}. \tag{4.13}$$

Quant à la solution réduite $\boldsymbol{u}_N(\boldsymbol{\mu}) \in \boldsymbol{W}_{N_{pr}}$, elle s'écrit

$$\boldsymbol{u}_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} \boldsymbol{u}_{Ni}(\boldsymbol{\mu}) \ \boldsymbol{\xi}_i^{pr}. \tag{4.14}$$

Nous allons à présent utiliser la dépendance affine en paramètres pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $\boldsymbol{v} = \boldsymbol{\xi}_n^{pr}$, pour $n = 1, ..., N$, alors pour tout $\boldsymbol{\mu} \in \mathcal{D}$, $\boldsymbol{u}_N(\boldsymbol{\mu}) \in \boldsymbol{W}_{N_{pr}}$ est solution de

$$\sum_{i=1}^{N} \left(\sum_{e=1}^{Neqs} \sum_{q=1}^{Q_{a_e}} \theta_{a_e}^{eq}(\boldsymbol{\mu}) a_e^{eq}(\boldsymbol{\xi}_n^{pr}, \boldsymbol{\xi}_i^{pr}) \right) \ \boldsymbol{u}_{N_i}(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs} \sum_{q=1}^{Q_{f_e}} \theta_{f_e}^{eq}(\boldsymbol{\mu}) f_e^{eq}(\boldsymbol{\xi}_n^{pr}) \right), \tag{4.15}$$

qui peut être également écrit sous la forme matricielle :

$$\left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{a_e}}\theta_{a_e}^{eq}(\boldsymbol{\mu})\boldsymbol{A}_{e,N}^{eq}\right)\boldsymbol{u}_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_{f_e}}\theta_{f_e}^{eq}(\boldsymbol{\mu})\boldsymbol{F}_{e,N}^{eq}\right), \tag{4.16}$$

avec

$$\Big(\boldsymbol{u}_N(\boldsymbol{\mu})\Big)_i = \boldsymbol{u}_{N_i}(\boldsymbol{\mu}),\ \Big(\boldsymbol{A}_{e,N}^{eq}\Big)_{in} = a_e^{eq}\Big(\boldsymbol{\xi}_n^{pr},\boldsymbol{\xi}_i^{pr}\Big),\ \text{et}\ \Big(\boldsymbol{F}_{e,N}^{eq}\Big)_n = f_e^{eq}\Big(\boldsymbol{\xi}_n^{pr}\Big). \tag{4.17}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\theta_{\ell_e}^{eq}(\boldsymbol{\mu})\ell_e^{eq}(\boldsymbol{u}_N(\boldsymbol{\mu}))\right),\ \forall\boldsymbol{\mu}\in\mathcal{D}, \tag{4.18}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\theta_{\ell_e}^{eq}(\boldsymbol{\mu})\boldsymbol{L}_{e,N}^{eq}\right)u_N(\boldsymbol{\mu}),\ \forall\boldsymbol{\mu}\in\mathcal{D}, \tag{4.19}$$

avec $(\boldsymbol{L}_{e,N}^{eq})_i = \ell_e^{eq}(\boldsymbol{\xi}_i^{pr})$. La décomposition *hors-ligne/en-ligne* est maintenant claire. Les fonctions de base $\boldsymbol{\xi}_i^{pr}$, $1\le i\le N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible – pour $e=1,...,Neqs$ – la construction des matrices $\boldsymbol{A}_{e,N}^{eq}\in\mathbb{R}^{N\times N}$, $1\le q\le Q_a$, et des vecteurs $\boldsymbol{F}_{e,N}^{eq}\in\mathbb{R}^N$, $1\le q\le Q_f$ et $\boldsymbol{L}_{e,N}^{eq}\in R^N$, $1\le q\le Q_\ell$. À l'étape *en-ligne*, pour chaque vecteur de paramètres $\boldsymbol{\mu}\in\mathcal{D}$ donné, on assemble la matrice $\boldsymbol{A}_N(\boldsymbol{\mu}) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{a_e}}\theta_{a_e}^{eq}(\boldsymbol{\mu})\boldsymbol{A}_{e,N}^{eq}$, ainsi que les vecteurs $\boldsymbol{F}_N(\boldsymbol{\mu}) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{f_e}}\theta_{f_e}^{eq}(\boldsymbol{\mu})\boldsymbol{F}_{e,N}^{eq}$ et $\boldsymbol{L}_{e,N}^{eq}(\boldsymbol{\mu})$ $= \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\theta_{\ell_e}^{eq}(\boldsymbol{\mu})\boldsymbol{L}_{e,N}^{eq}$. Enfin, pour tout $\boldsymbol{\mu}\in\mathcal{D}$, nous résolvons le système réduit

$$\boldsymbol{A}_N(\boldsymbol{\mu})\ \boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{F}_N(\boldsymbol{\mu}), \tag{4.20}$$

et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = \boldsymbol{L}_N^T(\boldsymbol{\mu})\ \boldsymbol{u}_N(\boldsymbol{\mu}),\ \forall\boldsymbol{\mu}\in\mathcal{D}. \tag{4.21}$$

Définissons l'entier positif $Q_a^e = \sum_{e=1}^{Neqs}Q_{a_e}$. Durant l'étape *hors-ligne* il est nécessaire de faire N résolutions FE (coûteuses) du problème primal $\mathcal{O}(Q_a^eN^2\mathcal{N})$ produits scalaires pour les étapes de projection sur la base réduite. Cependant la complexité de l'étape *en-ligne* ne dépend plus de la (grande) dimension $\mathcal{N}$ puisqu'elle implique $\mathcal{O}(Q_a^eN^2)$ multiplications pour assembler le système réduit primal, et $\mathcal{O}(N^3)$ pour le résoudre. Ensuite, il faut compter $\mathcal{O}(N)$ produits scalaires pour avoir la sortie du modèle en utilisant (4.21).

2 Problèmes linéaires non-affines

Tournons-nous maintenant vers les problèmes elliptiques qui n'admettent pas de dépendance affine en paramètres. Commençons par rappeler, en reprenant les notations introduites dans le chapitre 3, que la fonction non-affine ζ est définie sur l'espace $E^\zeta = \mathcal{D}\times\mathbb{R}^d$. Chaque équation se compose de N_{na} fonctions non-affines, nous notons donc N_{na_i}

le nombre de fonctions non-affines associées à la $i^{\text{ème}}$ équation. Lorsque N_{na_i} fonctions non-affines $\zeta_1, ..., \zeta_{N_{na_i}}$, interviennent, elles sont définies sur $E^{\zeta^1} \times ... \times E^{\zeta^{N_{na_i}}} \equiv E^{\underline{\zeta^{N_{na_i}}}}$. Soit $\zeta^{iq}(\boldsymbol{\mu}; \boldsymbol{x})$ la $q^{\text{ème}}$ fonction non-affine de la $i^{\text{ème}}$ équation du problème multi-physique, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pout tout $\boldsymbol{x} \in \Omega$. Pour un entier strictement positif M_i^q donné, cette fonction est approchée par $\zeta_{M_i^q}^{iq}(\boldsymbol{\mu}; \boldsymbol{x}) = \sum_{m=1}^{M_i^q} \beta^{iqm}(\boldsymbol{\mu})\ \hat{\zeta}^{iqm}$.

Considérons $Neqs$ formes bilinéaires $a_i : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na_i}}}} \to \mathbb{R}$ et linéaires $f_i : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na_i}}}} \to \mathbb{R}$, pour $i = 1, ..., Neqs$. Nous supposons ici que les a_i, pour $i = 1, ..., Neqs$, sont des formes bilinéaires symétriques, définies positives. Nous pouvons donc introduire les produits scalaires $\big((.,.)\big)_{a_i,\boldsymbol{\mu}}$ définis, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$), par

$$\big((\boldsymbol{w}, \boldsymbol{z})\big)_{a_i,\boldsymbol{\mu}} \equiv a_i\big(\boldsymbol{w}, \boldsymbol{z}; \boldsymbol{\mu}; \underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x})}\big),\ \forall \boldsymbol{w}, \boldsymbol{z} \in \boldsymbol{X}_{\mathcal{N}},\ 1 \leq i \leq Neqs. \tag{4.22}$$

Notons que les normes associées à ces produits scalaires sont

$$\big|\big|\big|\boldsymbol{w}\big|\big|\big|_{a_i,\boldsymbol{\mu}} \equiv \sqrt{\big((\boldsymbol{w}, \boldsymbol{w})\big)_{a_i,\boldsymbol{\mu}}},\ \forall \boldsymbol{w} \in \boldsymbol{X}_{\mathcal{N}},\ 1 \leq i \leq Neqs. \tag{4.23}$$

De plus, nous supposons que les a_i, $1 \leq i \leq Neqs$, sont continues et coercives. Autrement dit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, il existe des constantes de continuité $\gamma^{a_i,\mathcal{N}}(\boldsymbol{\mu})$ telles que

$$\gamma^{a_i,\mathcal{N}}(\boldsymbol{\mu}) \equiv \sup_{\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}} \frac{a_i(\boldsymbol{v}, \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x})})}{\big|\big|\big|\boldsymbol{v}\big|\big|\big|^2_{a_i,\boldsymbol{\mu}_{ref}}} < \infty, \tag{4.24}$$

ainsi que des constantes de coercivité $\alpha^{a_i,\mathcal{N}}(\boldsymbol{\mu})$ telles que

$$\alpha^{a_i,\mathcal{N}}(\boldsymbol{\mu}) \equiv \inf_{\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}} \frac{a_i(\boldsymbol{v}, \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x})})}{\big|\big|\big|\boldsymbol{v}\big|\big|\big|^2_{a_i,\boldsymbol{\mu}_{ref}}} > 0. \tag{4.25}$$

Comme précédemment les formes bilinéaires a_i et linéaires f_i, $1 \leq i \leq Neqs$, sont les contributions de chaque équation au problème multi-physique. Afin d'évaluer la sortie du modèle, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous utilisons les opérateurs appropriés ℓ_i : $\boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na_i}}}} \to \mathbb{R}$, pour $i = 1, ..., Neqs$. Pour de mettre en oeuvre une stratégie *hors-ligne/en-ligne* efficace, a_i, f_i et ℓ_i, pour $i = 1, ..., Neqs$, doivent pouvoir s'exprimer en faisant apparaître une dépendance affine en paramètres. Nous allons donc approcher, via la méthode EIM, les termes pour lequels il est impossible d'écrire directement une décomposition affine. Nous supposons donc que, pour $i = 1, ..., Neqs$, il existe des entiers positifs, Q_{a_i}, Q_{f_i} et Q_{ℓ_i}, avec $1 \leq Q_{a_i}, Q_{f_i}, Q_{\ell_i} \leq N_{na_i}$, de sorte que la méthode EIM détermine $\big(M_{a_i}^q\big)_{q=1,...,Q_{a_i}}$, $\big(M_{f_i}^q\big)_{q=1,...,Q_{f_i}}$ et $\big(M_{\ell_i}^q\big)_{q=1,...,Q_\ell}$ tels que, pour $n_{a_i} = \sum_{q=1}^{Q_{a_i}} M_{a_i}^q$, $n_{f_i} = \sum_{q=1}^{Q_{f_i}} M_{f_i}^q$ et $n_{\ell_i} = \sum_{q=1}^{Q_{\ell_i}} M_{\ell_i}^q$, ainsi que pour tout $\boldsymbol{\mu} \in \mathcal{D}$, les formes bilinéaire et linéaires $a_i(.,.;\boldsymbol{\mu}; \underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x})})$, $f_i(.;\boldsymbol{\mu}; \underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x})})$ et $\ell_i(.,\boldsymbol{\mu}; \underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x})})$ puissent être

approchées de la manière suivante :

$$\begin{cases} a_i\big(\boldsymbol{u},\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{a}_i\big(\boldsymbol{u},\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{a_i}}}^{i\ n_{a_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_{a_i}}\sum_{m=1}^{M_{a_i}^q} \beta_{a_i}^{iqm}(\boldsymbol{\mu}) a_i^{iqm}(\boldsymbol{u},\boldsymbol{v};\hat{\zeta}^{iqm}(\boldsymbol{x})), \\ f_i\big(\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{f}_i\big(\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{f_i}}}^{i\ n_{f_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_{f_i}}\sum_{m=1}^{M_{f_i}^q} \beta_{f_i}^{iqm}(\boldsymbol{\mu}) f_i^{iqm}\big(\boldsymbol{v};\hat{\zeta}^{iqm}(\boldsymbol{x})\big), \\ \ell_i\big(\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta^{i\ N_{na_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) \approx \tilde{\ell}_i\big(\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{\ell_i}}}^{i\ n_{\ell_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) = \displaystyle\sum_{q=1}^{Q_{\ell_i}}\sum_{m=1}^{M_{\ell_i}^q} \beta_{\ell_i}^{iqm}(\boldsymbol{\mu}) \ell_i^{iqm}\big(\boldsymbol{v};\hat{\zeta}^{iqm}(\boldsymbol{x})\big), \end{cases} \tag{4.26}$$

pour tout $\boldsymbol{u} \in \boldsymbol{X}_{\mathcal{N}}$ et pour tout $\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}$, et où pour $i = 1, ..., Neqs$, $\beta_{a_i}^{iqm} : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_{a_i}$, $1 \le m \le M_{a_i}^q$, $\beta_{f_i}^{iqm} : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_{f_i}$, $1 \le m \le M_{f_i}^q$ et $\beta_{\ell_i}^{iqm} : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_{\ell_i}$, $1 \le m \le M_{\ell_i}^q$ sont déterminées par la méthode EIM. Pour $i = 1, ..., Neqs$, posons $E_{M^q}^{\zeta^q} = \mathcal{D} \times \mathbb{R}^d$ avec $q = 1, ..., N_{na_i}$, ainsi que $E^{\underline{\zeta_{M^{n_{a_i}}}^{n_{a_i}}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_{a_i}}}^{n_{a_i}}}$, $E^{\underline{\zeta_{M^{n_{f_i}}}^{n_{f_i}}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_{f_i}}}^{n_{f_i}}}$, $E^{\underline{\zeta_{M^{n_{\ell_i}}}^{n_{\ell_i}}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_{\ell_i}}}^{n_{\ell_i}}}$. Nous remarquons que nous avons, pour $i = 1, ..., Neqs$, $\tilde{a}_i\big(.,.;\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{a_i}}}^{i\ n_{a_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_{a_i}}}^{n_{a_i}}}} \to \mathbb{R}$, $\tilde{f}_i\big(.;\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{f_i}}}^{i\ n_{f_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_{f_i}}}^{n_{f_i}}}} \to \mathbb{R}$ et $\tilde{\ell}_i\big(.;\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{\ell_i}}}^{i\ n_{\ell_i}}(\boldsymbol{\mu};\boldsymbol{x})}\big) : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_{\ell_i}}}^{n_{\ell_i}}}} \to \mathbb{R}$.

Définissons $\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x})}$ comme étant la somme des approximations des différentes fonctions non-affines présentes dans les équations du problème multi-physique. Posons $n_\ell = \sum_{i=1}^{Neqs} n_{\ell_i}$ et $E^{\underline{\zeta_{M^{n_\ell}}^{n_\ell}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_\ell}}^{n_\ell}}$. De la même manière nous posons $n_a = \sum_{i=1}^{Neqs} n_{a_i}$ et $E^{\underline{\zeta_{M^{n_a}}^{n_a}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_a}}^{n_a}}$, ainsi que $n_f = \sum_{i=1}^{Neqs} n_{f_i}$ et $E^{\underline{\zeta_{M^{n_f}}^{n_f}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_f}}^{n_f}}$. En nous servant des approximations décrites dans (4.26), pour un $\boldsymbol{\mu} \in \mathcal{D}$ fixé, nous nous intéressons à l'évaluation d'une sortie $s_{\mathcal{N}}(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$:

$$s_{\mathcal{N}}(\boldsymbol{\mu}) = \tilde{\ell}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu};\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x})}\Big), \tag{4.27}$$

où $\tilde{\ell}\Big(.;\boldsymbol{\mu};\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x})}\Big) : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_\ell}}^{n_\ell}}} \to \mathbb{R}$ est définie par

$$\tilde{\ell}\Big(.;\boldsymbol{\mu};\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x})}\Big) = \sum_{i=1}^{Neqs} \tilde{\ell}_i\Big(.;\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{\ell_i}}}^{i\ n_{\ell_i}}(\boldsymbol{\mu};\boldsymbol{x})}\Big). \tag{4.28}$$

La solution $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ vérifie

$$\tilde{a}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu};\boldsymbol{x})}\Big) = \tilde{f}\Big(\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_f}}^{n_f}(\boldsymbol{\mu};\boldsymbol{x})}\Big),\ \forall \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}, \tag{4.29}$$

où $\tilde{a}\Big(.;.;\boldsymbol{\mu};\underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu};\boldsymbol{x})}\Big) : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_a}}^{n_a}}} \to \mathbb{R}$ et $\tilde{f}\Big(.;\boldsymbol{\mu};\underline{\zeta_{M^{n_f}}^{n_f}(\boldsymbol{\mu};\boldsymbol{x})}\Big) : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_f}}^{n_f}}} \to \mathbb{R}$ sont définies par

$$\tilde{a}\Big(.;.;\boldsymbol{\mu};\underline{\zeta_{M^{n_a}}^{n_a}(\boldsymbol{\mu};\boldsymbol{x})}\Big) = \sum_{i=1}^{Neqs} \tilde{a}_i\Big(.;.;\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{a_i}}}^{i\ n_{a_i}}(\boldsymbol{\mu};\boldsymbol{x})}\Big) \tag{4.30}$$

et

$$\tilde{f}\Big(.;\boldsymbol{\mu};\underline{\zeta^{n_f}_{M^{n_f}}(\boldsymbol{\mu};\boldsymbol{x})}\Big)=\sum_{i=1}^{Neqs}\tilde{f}_i\Big(.;\boldsymbol{\mu};\underline{\zeta^{i\ n_{f_i}}_{M_i^{n_{f_i}}}(\boldsymbol{\mu};\boldsymbol{x})}\Big). \tag{4.31}$$

Nous supposons ici que $\tilde{a}$ est une forme bilinéaire symétrique, définie positive. Nous pouvons donc introduire le produit scalaire $\big((.,.)\big)_{\tilde{a},\boldsymbol{\mu}}$ défini, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ (et donc en particulier pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref}\in\mathcal{D}$), par

$$\big((\boldsymbol{w},\boldsymbol{z})\big)_{\tilde{a},\boldsymbol{\mu}}\equiv\tilde{a}\big(\boldsymbol{w},\boldsymbol{z};\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})}\big),\ \forall\boldsymbol{w},\boldsymbol{z}\in X_{\mathcal{N}}. \tag{4.32}$$

Notons que la norme associée à ce produit scalaire est

$$\big|\big|\big|\boldsymbol{w}\big|\big|\big|_{\tilde{a},\boldsymbol{\mu}}\equiv\sqrt{\big((\boldsymbol{w},\boldsymbol{w})\big)_{\tilde{a},\boldsymbol{\mu}}},\ \forall\boldsymbol{w}\in X_{\mathcal{N}}. \tag{4.33}$$

De plus, nous supposons que $\tilde{a}$ est continue et coercive. Autrement dit, pour tout $\boldsymbol{\mu}\in\mathcal{D}$ il existe une constante de continuité $\gamma^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\gamma^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu})\equiv\sup_{\boldsymbol{v}\in\boldsymbol{X}_{\mathcal{N}}}\frac{\tilde{a}(\boldsymbol{v},\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})})}{\big|\big|\big|bv\big|\big|\big|^2_{\tilde{a},\boldsymbol{\mu}_{ref}}}<\infty, \tag{4.34}$$

ainsi qu'une constante de coercivité $\alpha^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu})$ telle que

$$\alpha^{\tilde{a},\mathcal{N}}(\boldsymbol{\mu})\equiv\inf_{\boldsymbol{v}\in X_{\mathcal{N}}}\frac{\tilde{a}(\boldsymbol{v},\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta^{n_a}_{M^{n_a}}(\boldsymbol{\mu};\boldsymbol{x})})}{\big|\big|\big|\boldsymbol{v}\big|\big|\big|^2_{\tilde{a},\boldsymbol{\mu}_{ref}}}>0. \tag{4.35}$$

De la même manière que pour les problèmes elliptiques linéaires affines, nous introduisons une séquence d'espaces d'approximation emboîtés $\boldsymbol{W}_{N_{pr}}$ tels que $\boldsymbol{W}_{1_{pr}}\subset\boldsymbol{W}_{2_{pr}}\subset...\subset\boldsymbol{W}_{N_{maxpr}}\subset\boldsymbol{X}_{\mathcal{N}}$, avec N_{max} un entier positif. Nous considérons $S_N=\{\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_N\}$ un ensemble de N jeux de paramètres sélectionnés à l'aide de l'algorithme glouton et $S_N^{\boldsymbol{u}}$ l'ensemble qui va contenir, pour tout $\boldsymbol{\mu}$ dans S_N, la solution de (4.29). Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a,\boldsymbol{\mu}_{ref}}$, aux éléments de l'ensemble $S_N^{\boldsymbol{u}}$ pour avoir des fonctions de bases orthonormalisées $\boldsymbol{\xi}_n^{pr}$, $1\leq n\leq N$. L'espace d'approximation $\boldsymbol{W}_{N_{pr}}$ est défini par

$$\boldsymbol{W}_{N_{pr}}=span\Big\{\boldsymbol{\xi}_n^{pr},\ 1\leq n\leq N\Big\}. \tag{4.36}$$

Quant à la solution réduite $\boldsymbol{u}_N(\boldsymbol{\mu})\in\boldsymbol{W}_{N_{pr}}$ elle s'écrit

$$\boldsymbol{u}_N(\boldsymbol{\mu})=\sum_{i=1}^{N}\boldsymbol{u}_{Ni}(\boldsymbol{\mu})\ \boldsymbol{\xi}_i^{pr}. \tag{4.37}$$

Nous allons maintenant utiliser les approximations décrites dans (4.26) pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $\boldsymbol{v}=\boldsymbol{\xi}_n^{pr}$, $n=1,...,N$, pour tout $\boldsymbol{\mu}\in\mathcal{D}$, $\boldsymbol{u}_N(\boldsymbol{\mu})$ est solution de

$$\begin{aligned}&\sum_{i=1}^{N}\Bigg(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{a_e}}\sum_{m=1}^{M^q_{a_e}}\beta^{eqm}_{a_e}(\boldsymbol{\mu})a^{eqm}_e\Big(\boldsymbol{\xi}_n^{pr},\boldsymbol{\xi}_i^{pr};\hat{\zeta}^{eqm}(\boldsymbol{x})\Big)\Bigg)\ \boldsymbol{u}_{N_i}(\boldsymbol{\mu})\\&=\Bigg(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{f_e}}\sum_{m=1}^{M^q_{f_e}}\beta^{eqm}_{f_e}(\boldsymbol{\mu})f^{eqm}_e\Big(\boldsymbol{\xi}_n^{pr};\hat{\zeta}^{eqm}(\boldsymbol{x})\Big)\Bigg),\end{aligned} \tag{4.38}$$

qui peut être également écrit sous la forme matricielle :

$$\left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{a_e}}\sum_{m=1}^{M_{a_e}^q}\beta_{a_e}^{eqm}(\boldsymbol{\mu})\boldsymbol{A}_{e,N}^{eqm}\right)\boldsymbol{u}_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{f_e}}\sum_{m=1}^{M_{f_e}^q}\beta_{f_e}^{eqm}(\boldsymbol{\mu})\boldsymbol{F}_{e,N}^{eqm}\right), \tag{4.39}$$

avec

$$\begin{aligned}&\Big(\boldsymbol{u}_N(\boldsymbol{\mu})\Big)_i = \boldsymbol{u}_{N_i}(\boldsymbol{\mu})\ ,\ \Big(\boldsymbol{A}_{e,N}^{eqm}\Big)_{in} = a_e^{eqm}\Big(\boldsymbol{\xi}_n^{pr},\boldsymbol{\xi}_i^{pr};\hat{\zeta}^{eqm}(\boldsymbol{x})\Big)\\ &\text{et}\ \ \Big(\boldsymbol{F}_{e,N}^{eqm}\Big)_n = f_e^{eqm}\Big(\boldsymbol{\xi}_n^{pr};\hat{\zeta}^{eqm}(\boldsymbol{x})\Big).\end{aligned} \tag{4.40}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\sum_{m=1}^{M_{\ell_e}^q}\beta_{\ell_e}^{eqm}\Big(\boldsymbol{\mu}\Big)\ell_e^{eqm}\Big(\boldsymbol{u}_N(\boldsymbol{\mu});\hat{\zeta}^{eqm}(\boldsymbol{x})\Big)\right),\ \forall\boldsymbol{\mu}\in\mathcal{D}, \tag{4.41}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell}}\sum_{m=1}^{M_{\ell_e}^q}\beta_{\ell}^{eqm}(\boldsymbol{\mu})\boldsymbol{L}_{e,N}^{eqm\ T}\right)\boldsymbol{u}_N(\boldsymbol{\mu}),\ \forall\boldsymbol{\mu}\in\mathcal{D}, \tag{4.42}$$

avec $(\boldsymbol{L}_{e,N}^{eqm})_i = \ell_e^{eqm}(\boldsymbol{\xi}_i^{pr};\hat{\zeta}^{eqm}(\boldsymbol{x}))$. Regardons comment s'articule la décomposition *hors-ligne/en-ligne*. Les fonctions de base $\boldsymbol{\xi}_i^{pr}$, $1\le i\le N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible – pour $e=1,...,Neqs$ – la construction des matrices $\boldsymbol{A}_{e,N}^{eqm}\in\mathbb{R}^{N\times N}$, $1\le q\le Q_{a_e}$, $1\le m\le M_{a_e}^q$ et des vecteurs $\boldsymbol{F}_{e,N}^{eqm}\in\mathbb{R}^N$, $1\le q\le Q_{f_e}$, $1\le m\le M_{f_e}^q$ et $\boldsymbol{L}_{e,N}^{eqm}\in\mathbb{R}^N$, $1\le q\le Q_{\ell_e}$, $1\le m\le M_{\ell_e}^q$. À l'étape *en-ligne*, pour chaque vecteur de paramètres $\boldsymbol{\mu}\in\mathcal{D}$ donné, on assemble la matrice $\boldsymbol{A}_N(\boldsymbol{\mu}) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{a_e}}\sum_{m=1}^{M_{a_e}^q}\beta_{a_e}^{eqm}(\boldsymbol{\mu})\boldsymbol{A}_{e,N}^{eqm}$, ainsi que les vecteurs $\boldsymbol{F}_N(\boldsymbol{\mu}) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{f_e}}\sum_{m=1}^{M_{f_e}^a}\beta_{f_e}^{eqm}(\boldsymbol{\mu})\boldsymbol{F}_{e,N}^{eqm}$ et $\boldsymbol{L}_N(\boldsymbol{\mu}) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\sum_{m=1}^{M_{\ell_e}^a}\beta_{\ell_e}^{eqm}(\boldsymbol{\mu})\boldsymbol{L}_{e,N}^{eqm}$. Enfin, pour tout $\boldsymbol{\mu}\in\mathcal{D}$, nous résolvons le système réduit

$$\boldsymbol{A}_N(\boldsymbol{\mu})\ \boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{F}_N(\boldsymbol{\mu}), \tag{4.43}$$

et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = \boldsymbol{L}_N^T(\boldsymbol{\mu})\ \boldsymbol{u}_N(\boldsymbol{\mu}),\ \forall\boldsymbol{\mu}\in\mathcal{D}. \tag{4.44}$$

Nous définissons les entiers positifs $Q_a^e = \sum_{e=1}^{Neqs}Q_{a_e}$ et $M_a^{qe} = \sum_{e=1}^{Neqs}M_{a_e}^q$. Durant l'étape *hors-ligne* il est nécessaire de faire N résolutions FE (coûteuses) du problème primal et $\mathcal{O}(Q_a^eM_a^{qe}N^2\mathcal{N})$ produits scalaires pour les étapes de projection sur la base réduite. En revanche la complexité de l'étape *en-ligne* ne dépend plus de la (grande) dimension $\mathcal{N}$ puisqu'elle implique $\mathcal{O}(Q_a^eM_a^{qe}N^2)$ multiplications pour assembler le système réduit primal, et $\mathcal{O}(N^3)$ pour le résoudre. Ensuite il faut compter $\mathcal{O}(N)$ produits scalaires pour avoir la sortie du modèle en utilisant (4.42).

3 Problèmes non-linéaires elliptiques

Le dernier type de problème que nous abordons dans ce chapitre concerne les problèmes elliptiques coercifs non-linéaires. Rappelons que dans ce cas, les fonctions non-affines dépendent de la solution $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ de l'EDP pour tout $\boldsymbol{\mu} \in \mathcal{D}$. Commençons par introduire les espaces $E^{\zeta^{iq}} = \mathcal{D} \times \mathbb{R}^d \times X_{\mathcal{N}}$, pour $q = 1, ..., N_{na}$ et pour $i = 1, ..., Neqs$. En reprenant la notation introduite précédemment nous avons, pour un entier N_{na_i} positif donné, $E^{\zeta^1} \times ... \times E^{\zeta^{N_{na_i}}} \equiv E^{\underline{\zeta^{N_{na_i}}}}$. Considérons maintenant $\zeta^{iq}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu})) \in E^{\zeta^{iq}}$ la $q^{\text{ème}}$ fonction non-affine de la $i^{\text{ème}}$ équation du problème multi-physique, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, pour tout $\boldsymbol{x} \in \Omega$ et pour tout $u_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$. Pour un entier strictement positif M_i^q donné, cette fonction est approchée par $\zeta_{M_i^q}^{iq}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu})) = \sum_{m=1}^{M_i^q} \beta^{iqm}\big(\boldsymbol{\mu}; \boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu})\big)\ \hat{\zeta}^{iqm}$. Considérons les contributions de chaque équation, à savoir les résidus $g_i : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na_i}}}} \to \mathbb{R}$ ainsi que les formes bilinéaires associées aux jacobiennes $j_i : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na_i}}}} \to \mathbb{R}$, pour $i = 1, ..., Neqs$. Afin d'évaluer la sortie du modèle, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous utilisons les opérateurs appropriés $\ell_i : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta^{N_{na_i}}}} \to \mathbb{R}$, pour $i = 1, ..., Neqs$.

Pour mettre en oeuvre une stratégie *hors-ligne/ en-ligne* efficace nous allons maintenant approcher, via la méthode EIM, les termes qui pour lequels il est impossible d'écrire directement une décomposition affine. Autrement dit nous supposons qu'il existe des entiers positifs Q_{j_i}, Q_{r_i} et Q_{ℓ_i}, avec $1 \leq Q_{j_i}, Q_{r_i}, Q_{\ell_i} \leq N_{na}$ et pour $i = 1, ..., Neqs$, de sorte que la méthode EIM détermine $\big(M_{j_i}^q\big)_{q=1,...,Q_{j_i}}$, $\big(M_{g_i}^q\big)_{q=1,...,Q_{g_i}}$ et $\big(M_{\ell_i}^q\big)_{q=1,...,Q_{\ell_i}}$ tels que, pour $n_{j_i} = \sum_{q=1}^{Q_{j_i}} M_{j_i}^q$, $n_{g_i} = \sum_{q=1}^{Q_{g_i}} M_{g_i}^q$ et $n_{\ell_i} = \sum_{q=1}^{Q_{\ell_i}} M_{\ell_i}^q$, ainsi que pour tout $\boldsymbol{\mu} \in \mathcal{D}$, les formes bilinéaire et linéaires $j_i(.,.;\boldsymbol{\mu}; \underline{\zeta^{N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x}; .)})$, $g_i(\boldsymbol{u}, .; \boldsymbol{\mu}; \underline{\zeta^{N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})})$ et $\ell_i(., \boldsymbol{\mu}; \underline{\zeta^{N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x}; .)})$ puissent être approchées de la manière suivante :

$$\begin{aligned} j_i\big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})}\big) \approx\ & \tilde{j}_i\big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta_{M^{n_j}}^{n_j}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})}\big) = \\ & \sum_{q=1}^{Q_{j_i}} \sum_{m=1}^{M_{j_i}^q} \beta_{j_i}^{iqm}(\boldsymbol{\mu}; \boldsymbol{u}) j_i^{iqm}\big(\boldsymbol{u}, \boldsymbol{v}; \hat{\zeta}^{iqm}(\boldsymbol{x})\big), \end{aligned} \tag{4.45}$$

$$\begin{aligned} g_i\big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})}\big) \approx\ & \tilde{g}_i\big(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta_{M^{n_g}}^{n_g}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})}\big) = \\ & \sum_{q=1}^{Q_{g_i}} \sum_{m=1}^{M_{g_i}^q} \beta_{g_i}^{iqm}(\boldsymbol{\mu}; \boldsymbol{u}) g_i^{iqm}\big(\boldsymbol{u}, \boldsymbol{v}; \hat{\zeta}^{iqm}(\boldsymbol{x})\big), \end{aligned} \tag{4.46}$$

et

$$\begin{aligned} \ell_i\big(\boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta^{N_{na_i}}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})}\big) \approx\ & \tilde{\ell}_i\big(\boldsymbol{v}; \boldsymbol{\mu}; \underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u})}\big) = \\ & \sum_{q=1}^{Q_{\ell_i}} \sum_{m=1}^{M_{\ell_i}^q} \beta_{\ell_i}^{iqm}(\boldsymbol{\mu}) \ell_i^{iqm}\big(\boldsymbol{v}; \hat{\zeta}^{iqm}(\boldsymbol{x})\big), \end{aligned} \tag{4.47}$$

pour tout $\boldsymbol{u} \in \boldsymbol{X}_{\mathcal{N}}$ et pour tout $\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}$, où pour $i = 1, ..., Neqs$, $\beta_{j_i}^{iqm} : \mathcal{D} \times \boldsymbol{X}_{\mathcal{N}} \to \mathbb{R}$, $1 \leq q \leq Q_{j_i}$, $1 \leq m \leq M_{j_i}^q$, $\beta_{g_i}^{iqm} : \mathcal{D} \times \boldsymbol{X}_{\mathcal{N}} \to \mathbb{R}$, $1 \leq q \leq Q_{g_i}$, $1 \leq m \leq M_{g_i}^q$ et $\beta_{\ell_i}^{iqm} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_{\ell_i}$, $1 \leq m \leq M_{\ell_i}^q$ sont déterminées par la méthode EIM. Posons $E_{M^q}^{\zeta^q} = \mathcal{D} \times \mathbb{R}^d \times \boldsymbol{X}_{\mathcal{N}}$, pour $q = 1, ..., N_{na}$, ainsi que $E^{\underline{\zeta_{M^{n_{j_i}}}^{n_{j_i}}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_{j_i}}}^{n_{j_i}}}$, $E^{\underline{\zeta_{M^{n_{g_i}}}^{n_{g_i}}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_{g_i}}}^{n_{g_i}}}$, $E^{\underline{\zeta_{M^{n_{\ell_i}}}^{n_{\ell_i}}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_{\ell_i}}}^{n_{\ell_i}}}$. Notons que nous avons

$\tilde{j}_i\big(.,.;\boldsymbol{\mu};\underline{\zeta_{M^{n_{j_i}}}^{n_{j_i}}(\boldsymbol{\mu};\boldsymbol{x};.)}\big) : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_{j_i}}}^{n_{j_i}}}} \to \mathbb{R}$, $\tilde{g}_i\big(\boldsymbol{u},.;\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_i}}}^{n_{g_i}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u})}\big) : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_{g_i}}}^{n_{g_i}}}} \to \mathbb{R}$ et $\tilde{\ell}_i\big(.;\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{\ell_i}}}^{i\ n_{\ell_i}}(\boldsymbol{\mu};\boldsymbol{x};.)}\big) : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_{\ell_i}}}^{n_{\ell_i}}}} \to \mathbb{R}$.

Posons $n_\ell = \sum_{i=1}^{Neqs} n_{\ell_i}$ et $E^{\underline{\zeta_{M^{n_\ell}}^{n_\ell}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_\ell}}^{n_\ell}}$. Nous définissons alors $\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x};.)} \in E^{\underline{\zeta_{M^{n_\ell}}^{n_\ell}}}$ comme étant la somme des approximations des différentes fonctions non-affines présentes dans les équations du problème multi-physique. Nous posons également $n_j = \sum_{i=1}^{Neqs} n_{j_i}$ et $E^{\underline{\zeta_{M^{n_j}}^{n_j}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_j}}^{n_j}}$, ainsi que $n_g = \sum_{i=1}^{Neqs} n_{g_i}$ et $E^{\underline{\zeta_{M^{n_g}}^{n_g}}} = E_{M^1}^{\zeta^1} \times ... \times E^{\zeta_{M^{n_g}}^{n_g}}$. En nous servant des approximations décrites dans (4.45)-(4.47), pour un $\boldsymbol{\mu} \in \mathcal{D}$ fixé, nous nous intéressons à l'évaluation d'une sortie $s_{\mathcal{N}}(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$:

$$s_{\mathcal{N}}(\boldsymbol{\mu}) = \tilde{\ell}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu};\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big), \tag{4.48}$$

où $\tilde{\ell}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu};\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) : \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\underline{\zeta_{M^{n_\ell}}^{n_\ell}}} \to \mathbb{R}$ est définie par

$$\tilde{\ell}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu};\underline{\zeta_{M^{n_\ell}}^{n_\ell}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) = \sum_{i=1}^{Neqs} \tilde{\ell}_i\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu};\underline{\zeta_{M_i^{n_{\ell_i}}}^{i\ n_{\ell_i}}(\boldsymbol{\mu};\boldsymbol{x};.)}\Big). \tag{4.49}$$

La solution $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ vérifie

$$\tilde{g}_i\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_i}}}^{n_{g_i}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) = 0, \quad \forall \boldsymbol{v} \in \boldsymbol{X},\ 1 \le i \le Nequs. \tag{4.50}$$

En définissant $\tilde{g} : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta_{M^{n_g}}^{n_g}} \to \mathbb{R}$ par

$$\tilde{g}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_g}}^{n_g}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) = \begin{pmatrix} \tilde{g}_1\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_1}}}^{n_{g_1}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) \\ \tilde{g}_2\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_2}}}^{n_{g_2}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) \\ \vdots \\ \tilde{g}_{Neqs}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_{Neqs}}}}^{n_{g_{Neqs}}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) \end{pmatrix}, \tag{4.51}$$

l'équation (4.50) s'écrit alors

$$\tilde{g}\Big(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_g}}^{n_g}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big) = 0, \quad \forall \boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}. \tag{4.52}$$

L'algorithme de Newton est utilisé pour résoudre (4.52), voir l'algorithme 6. Soit $\tilde{j} : \boldsymbol{X}_{\mathcal{N}} \times \boldsymbol{X}_{\mathcal{N}} \times \mathcal{D} \times E^{\zeta_{M^{n_j}}^{n_j}} \to \mathbb{R}$, la jacobienne associée au résidu g et $\delta\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ l'incrément défini par

$$\delta\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}) = \boldsymbol{u}_{\mathcal{N}}^{k+1}(\boldsymbol{\mu}) - \boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}). \tag{4.53}$$

Pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour une solution initiale $\boldsymbol{u}^0(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ donnée, à chaque itération k de l'algorithme – avec $1 \le k \le k_{max}$, où k_{max} est en entier strictement positif donné –, nous cherchons $\delta u_{\mathcal{N}}^k(\boldsymbol{\mu}) \in \boldsymbol{X}_{\mathcal{N}}$ tel que pour tout $\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}$,

$$\tilde{j}\Big(\delta\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu});\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_j}}^{n_j}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}))}\Big)\Big[\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu})\Big] = -\tilde{g}\Big(\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}),v;\boldsymbol{\mu};\underline{\zeta_{M^{n_g}}^{n_g}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}))}\Big), \tag{4.54}$$

où on a

$$j\Big(\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_j}}^{n_j}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}))}\Big)\Big[\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu})\Big] = \left(\begin{array}{ccc} \dfrac{\partial g_1\Big(\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_1}}}^{n_{g_1}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big)}{\partial u_{\mathcal{N}}^1(\boldsymbol{\mu})} & \cdots & \dfrac{\partial g_1\Big(\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{g_1}}}^{n_{g_1}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big)}{\partial u_{\mathcal{N}}^{Neqs}(\boldsymbol{\mu})} \\ & \ddots & \\ \dfrac{\partial g_{N_{\text{equs}}}\Big(\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{gNeqs}}}^{n_{gNeqs}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big)}{\partial u_{\mathcal{N}}^1(\boldsymbol{\mu})} & \cdots & \dfrac{\partial g_{N_{\text{equs}}}\Big(\boldsymbol{u}_{\mathcal{N}}^k(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu};\underline{\zeta_{M^{n_{gNeqs}}}^{n_{gNeqs}}(\boldsymbol{\mu};\boldsymbol{x};\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}))}\Big)}{\partial u_{\mathcal{N}}^{Neqs}(\boldsymbol{\mu})} \end{array}\right) \tag{4.55}$$

avec $u_{\mathcal{N}}^i(\boldsymbol{\mu}) \in X_{\mathcal{N}_i}$, pour $i = 1, ..., Neqs$.

Comme précédemment nous introduisons une séquence d'espaces d'approximation emboîtés $\boldsymbol{W}_{N_{pr}}$ tels que $\boldsymbol{W}_{1_{pr}} \subset \boldsymbol{W}_{2_{pr}} \subset ... \subset \boldsymbol{W}_{N_{maxpr}} \subset \boldsymbol{X}_{\mathcal{N}}$, avec N_{max} un entier positif. Nous considérons $S_N = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_N\}$ un ensemble de N jeux de paramètres sélectionnés à l'aide de l'algorithme glouton et $S_N^{\boldsymbol{u}}$ l'ensemble qui va contenir, pour tout $\boldsymbol{\mu}$ dans S_N, la solution de (4.54) à l'itération finale. Ensuite nous appliquons le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $(.,.)_{X_{\mathcal{N}}}$ – associé à l'espace $\boldsymbol{X}_{\mathcal{N}}$, aux éléments de l'ensemble $S_N^{\boldsymbol{u}}$ pour avoir des fonctions de bases orthonormalisées $\boldsymbol{\xi}_n^{pr}$, $1 \leq n \leq N$. L'espace d'approximation $\boldsymbol{W}_{N_{pr}}$ est défini par

$$\boldsymbol{W}_{N_{pr}} = span\Big\{\boldsymbol{\xi}_n^{pr},\ 1 \leq n \leq N\Big\}. \tag{4.56}$$

Quant à la solution réduite $\boldsymbol{u}_N(\boldsymbol{\mu}) \in \boldsymbol{W}_{N_{pr}}$, elle s'écrit

$$\boldsymbol{u}_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} \boldsymbol{u}_{N_i}(\boldsymbol{\mu})\ \boldsymbol{\xi}_i^{pr}. \tag{4.57}$$

En nous basant sur la définition de l'incrément (4.53) nous avons

$$\delta\boldsymbol{u}_N^k(\boldsymbol{\mu}) = \sum_{i=1}^{N} \delta\boldsymbol{u}_{N_i}^k(\boldsymbol{\mu}) = \sum_{i=1}^{N} \Big(\boldsymbol{u}_{N_i}^{k+1}(\boldsymbol{\mu}) - \boldsymbol{u}_{N_i}^k(\boldsymbol{\mu})\Big)\boldsymbol{\xi}_i^{pr}. \tag{4.58}$$

Nous allons maintenant utiliser les approximations décrites dans (4.45)-(4.47) pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \boldsymbol{\xi}_n^{pr}$, $n = 1, ..., N$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour une solution initiale $\boldsymbol{u}_N^0(\boldsymbol{\mu}) \in \boldsymbol{W_N}$ donnée, à chaque itération k de l'algorithme de Newton – avec $1 \leq k \leq k_{max}$ –, nous cherchons $\delta\boldsymbol{u}_N^k(\boldsymbol{\mu}) \in \boldsymbol{W_N}$ tel que

$$\begin{aligned} &\sum_{i=1}^{N}\left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{je}}\sum_{m=1}^{M_{je}^q} \beta_{j_e}^{eqm}\Big(\boldsymbol{\mu};\boldsymbol{u}_N^k(\boldsymbol{\mu})\Big)\ j^{eqm}\Big(\boldsymbol{\xi}_n^{pr},\boldsymbol{\xi}_i^{pr};\hat{\zeta}^{eqm}(\boldsymbol{x})\Big)\right)\ \delta\boldsymbol{u}_{N_i}^k(\boldsymbol{\mu}) \\ &= -\sum_{i=1}^{N}\left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{ge}}\sum_{m=1}^{M_{ge}^q} \beta_{g_e}^{eqm}\Big(\boldsymbol{\mu};\boldsymbol{u}_N^k(\boldsymbol{\mu})\Big)\ g^{eqm}\Big(\boldsymbol{u}_N^k(\boldsymbol{\mu}),\boldsymbol{\xi}_n^{pr};\hat{\zeta}^{bqm}(\boldsymbol{x})\Big)\right) \end{aligned} \tag{4.59}$$

qui peut être également écrit sous la forme matricielle :

$$\left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{je}}\sum_{m=1}^{M_{je}^q} \beta_{j_e}^{eqm}(\boldsymbol{\mu};\boldsymbol{u}_N^k(\boldsymbol{\mu}))J_N^{eqm}\right)\delta\boldsymbol{u}_N^k(\boldsymbol{\mu}) = -\left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{ge}}\sum_{m=1}^{M_{ge}^q} \beta_{g_e}^{eqm}(\boldsymbol{\mu};\boldsymbol{u}_N^k(\boldsymbol{\mu}))G_N^{eqm}\right), \tag{4.60}$$

avec

$$\left(\boldsymbol{J}_{e,N}^{eqm}\right)_{in} = j_e^{eqm}\left(\boldsymbol{\xi}_n^{pr}, \boldsymbol{\xi}_i^{pr}; \hat{\zeta}^{eqm}(\boldsymbol{x})\right) \text{ et } \left(\boldsymbol{G}_{e,N}^{eqm}\right)_n = g_e^{eqm}\left(\boldsymbol{u}_N^k(\boldsymbol{\mu}), \boldsymbol{\xi}_n^{pr}; \hat{\zeta}^{eqm}(\boldsymbol{x})\right). \tag{4.61}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\sum_{m=1}^{M_{\ell_e}^q} \beta_{\ell_e}^{eqm}\Big(\boldsymbol{\mu}\Big)\ell_e^{eqm}\Big(\boldsymbol{u}_N(\boldsymbol{\mu}); \hat{\zeta}^{eqm}(\boldsymbol{x})\Big)\right), \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{4.62}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\sum_{m=1}^{M_{\ell_e}^q} \beta_{\ell_e}^{eqm}\Big(\boldsymbol{\mu}\Big)\boldsymbol{L}_{e,N}^{eqm\ T}\right) u_N(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{4.63}$$

avec $(\boldsymbol{L}_{e,N}^{eqm})_i = \ell_e^{eqm}(\boldsymbol{\xi}_i^{pr}; \hat{\zeta}^{eqm}(\boldsymbol{x}))$. La décomposition *hors-ligne/en-ligne* est maintenant claire. Les fonctions de base $\boldsymbol{\xi}_i^{pr}$, $1 \leq i \leq N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible – pour $e = 1, ..., Neqs$ – la construction des matrices $\boldsymbol{J}_{e,N}^{eqm} \in \mathbb{R}^{N\times N}$, $1 \leq q \leq Q_{j_e}$, $1 \leq m \leq M_{j_e}^q$ et des vecteurs $\boldsymbol{G}_{e,N}^{eqm} \in \mathbb{R}^N$, $1 \leq q \leq Q_{g_e}$, $1 \leq m \leq M_{g_e}^q$ et $\boldsymbol{L}_{e,N}^{eqm} \in \mathbb{R}^N$, $1 \leq q \leq Q_{\ell_e}$, $1 \leq m \leq M_{\ell_e}^q$. À l'étape *en-ligne*, pour chaque $\boldsymbol{\mu} \in \mathcal{D}$ donné, à la $k^{\text{ème}}$ itération de l'algorithme de Newton on assemble la matrice $\boldsymbol{J}_N\big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\big) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{j_e}}\sum_{m=1}^{M_{j_e}^q} \beta_{j_e}^{eqm}\big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\big)\boldsymbol{J}_{e,N}^{eqm}$, ainsi que le vecteur $\boldsymbol{G}_N\big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\big) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{g_e}}\sum_{m=1}^{M_{g_e}^q} \beta_{g_e}^{eqm}\big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\big)\boldsymbol{G}_{e,N}^{eqm}$ puis nous résolvons

$$\boldsymbol{J}_N\Big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\Big)\ \delta\boldsymbol{u}_N^k(\boldsymbol{\mu}) = -\boldsymbol{G}_N\Big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\Big). \tag{4.64}$$

Une fois que l'algorithme de Newton a convergé, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, nous assemblons le vecteur $\boldsymbol{L}_N(\boldsymbol{\mu}) = \sum_{e=1}^{Neqs}\sum_{q=1}^{Q_{\ell_e}}\sum_{m=1}^{M_{\ell_e}^q} \beta_{\ell_e}^{eqm}\big(\boldsymbol{\mu}; \boldsymbol{u}_N^k(\boldsymbol{\mu})\big)\boldsymbol{L}_{e,N}^{eqm}$, et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = \boldsymbol{L}_N^T(\boldsymbol{\mu})\ \boldsymbol{u}_N(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{4.65}$$

Si on s'intéresse à la complexité de l'étape *en-ligne*, le coût de l'application de l'algorithme de Newton est prédominant sur le reste des opérations. L'assemblage de $\boldsymbol{J}_N$ et de $\boldsymbol{G}_N$, ainsi que la résolution de (4.64), est indépendante de la (grande) dimension $\mathcal{N}$.

Remarque 11. *Nous concluons ce chapitre sur une remarque portant sur le déroulement de l'étape hors-ligne de l'algorithme EIM. Actuellement, lorsque la fonction non-linéaire que nous voulons approcher dépend de la solution de l'EDP, alors la sélection du $i^{ème}$ vecteur de paramètres, telle que présentée dans le chapitre 3, pour $i \geq 2$, dépend de la dimension élément fini via l'utilisation de $u_{\mathcal{N}}(\boldsymbol{\mu})$, – voir (3.7) –. Or, le calcul de $u_{\mathcal{N}}(\boldsymbol{\mu})$ pour chaque élément de l'échantillon de l'espace des paramètres utilisé dans (3.7) peut s'avérer extrêmement coûteux. Jusqu'à présent, nous commençons par effectuer la totalité de l'étape hors-ligne de l'algorithme EIM afin d'avoir une décomposition affine des formes linéaires et bilinéaires utilisées, puis nous nous servons de cette décomposition affine pour construire la base réduite. Cependant, nous pensons qu'il est envisageable de développer une stratégie permettant de coupler la construction des fonctions de base EIM avec la*

construction des éléments de la base réduite. De cette manière, durant la sélection du $i^{ème}$ vecteur de paramètres ($i \geq 2$) il serait possible de remplacer $u_{\mathcal{N}}(\boldsymbol{\mu})$ par l'approximation base réuduite $u_N(\boldsymbol{\mu})$. Cela permettrait de réduire le coût de calcul hors-ligne lié à la mise en oeuvre de l'algorithme EIM pour des problèmes non-linéaires multi-physiques complexes. Il s'agit d'une réflexion en cours.

Résumé

Nous avons vu dans ce chapitre comment appliquer la méthode des bases réduites aux problèmes multi-physiques elliptiques coercifs. La méthodologie présentée ici peut être facilement étendue au cas des problèmes paraboliques. Dans la section 1 nous nous sommes intéressés aux problèmes qui admettent une dépendance affine en paramètres, alors que dans la la section 2 les problèmes considérés n'admettent plus cette dépendance affine en paramètres. Enfin, dans la section 3, nous avons considéré les problèmes non-linéaires.

Chapitre 5

La méthode des bases réduites appliquée aux problèmes stationnaires avec non-linéarités quadratiques

Dans ce chapitre nous allons mettre en avant les difficultés associées à la résolution de problèmes avec non-linéarités quadratiques via la méthode des bases réduites. Les précédents travaux portant sur l'application de la méthode RBM aux équations – non-linéaires – de type Navier-Stokes se concentrent sur le choix optimal de la base et sur la mise en oeuvre d'estimateurs d'erreur [121, 84, 54, 39, 123]. Nous ne nous intéressons pas ici à ces aspects. Nous détaillerons une technique de projection, indépendante du choix de la base, qui peut être appliquée à tout problème avec non-linéarités quadratiques afin de les résoudre uniquement dans un espace réduit, sans utiliser de projections sur l'espace (de grande dimension) élément fini. Ce travail a été réalisé en collaboration avec E.Schenone durant le CEMRACS 2012 et est présenté dans [111].

1 Énoncé du problème général

Pour un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ donné, nous nous intéressons à l'évaluation d'une sortie $s(\boldsymbol{\mu}) \in \mathbb{R}$ exprimée comme fonctionnelle du champ $u(\boldsymbol{\mu})$:

$$s(\boldsymbol{\mu}) = \ell\Big(u(\boldsymbol{\mu}); \boldsymbol{\mu}\Big), \tag{5.1}$$

pour un opérateur $\ell(.;\boldsymbol{\mu})$ approprié. La formulation variationnelle du problème que nous traitons consiste à chercher $u(\boldsymbol{\mu}) \in X$ tel que

$$a\Big(u(\boldsymbol{\mu}), u(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) + b\Big(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}\Big), \ \forall v \in X. \tag{5.2}$$

où $a(.,.,.;\boldsymbol{\mu}) : X \times X \times X \times \mathcal{D} \to \mathbb{R}$, $b(.,.;\boldsymbol{\mu}) : X \times X \times \mathcal{D} \to \mathbb{R}$ et $f(.;\boldsymbol{\mu}) : X \times \mathcal{D} \to \mathbb{R}$ sont respectivement les formes trilinéaire, bilinéaire et linéaire associées à l'EDP. Nous faisons l'hypothèse que $a(.,.,.;\boldsymbol{\mu})$, $b(.,.;\boldsymbol{\mu})$, $f(.;\boldsymbol{\mu})$ et $\ell(.;\boldsymbol{\mu})$ dépendent de $\boldsymbol{\mu}$ de manière affine, autrement dit nous supposons qu'il existe des entiers positifs Q_a, Q_b, Q_f et Q_ℓ tels

que

$$\begin{cases} a\left(u,u,v;\boldsymbol{\mu}\right)=\displaystyle\sum_{q=1}^{Q_a}\theta_a^q(\boldsymbol{\mu})\ a^q(u,u,v), \quad \forall u,v\in X,\ \forall\boldsymbol{\mu}\in\mathcal{D}, \\ b\left(u,v;\boldsymbol{\mu}\right)=\displaystyle\sum_{q=1}^{Q_b}\theta_b^q(\boldsymbol{\mu})\ b^q(u,v), \quad \forall u,v\in X,\ \forall\boldsymbol{\mu}\in\mathcal{D}, \\ f\left(v;\boldsymbol{\mu}\right)=\displaystyle\sum_{q=1}^{Q_f}\theta_f^q(\boldsymbol{\mu})\ f^q(v), \quad \forall v\in X,\ \forall\boldsymbol{\mu}\in\mathcal{D}, \\ \ell\left(v;\boldsymbol{\mu}\right)=\displaystyle\sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu})\ \ell^q(v), \quad \forall v\in X,\ \forall\boldsymbol{\mu}\in\mathcal{D}, \end{cases} \tag{5.3}$$

où $\theta_a^q:\mathcal{D}\to\mathbb{R}$, $1\leq q\leq Q_a$, $\theta_b^q:\mathcal{D}\to\mathbb{R}$, $1\leq q\leq Q_b$, $\theta_f^q:\mathcal{D}\to\mathbb{R}$, $1\leq q\leq Q_f$ et $\theta_\ell^q:\mathcal{D}\to\mathbb{R}$, $1\leq q\leq Q_\ell$ sont des fonctions qui dépendent de $\boldsymbol{\mu}$.

2 Discrétisation élément fini

Introduisons à présent une discrétisation FE de (5.2). Pour cela commençons par définir l'espace discret $X_{\mathcal{N}}\subset X$ par

$$X_{\mathcal{N}}=vect\big\{\phi_1,...,\phi_{\mathcal{N}}\big\}, \tag{5.4}$$

ainsi que la projection de Galerkin $u_{\mathcal{N}}(\boldsymbol{\mu})$ de la solution $u(\boldsymbol{\mu})$ de (5.2) sur la base $\big\{\phi_1,...,\phi_{\mathcal{N}}\big\}$,

$$u_{\mathcal{N}}(\boldsymbol{\mu})=\sum_{i=1}^{\mathcal{N}}u_i(\boldsymbol{\mu})\ \phi_i. \tag{5.5}$$

D'après (5.5), la formulation discrète de (5.2) consiste à chercher $u_{\mathcal{N}}(\boldsymbol{\mu})\in X_{\mathcal{N}}$ tel que, pour $k=1,...,\mathcal{N}$,

$$\sum_{i=1}^{\mathcal{N}}\sum_{j=1}^{\mathcal{N}}u_i(\boldsymbol{\mu})u_j(\boldsymbol{\mu})a\Big(\phi_i,\phi_j,\phi_k;\boldsymbol{\mu}\Big)+\sum_{i=1}^{\mathcal{N}}u_i(\boldsymbol{\mu})b\Big(\phi_i,\phi_k;\boldsymbol{\mu}\Big)=f\Big(\phi_k;\boldsymbol{\mu}\Big). \tag{5.6}$$

L'algorithme de Newton est utilisé pour résoudre (5.6). Pour tout $\boldsymbol{\mu}\in\mathcal{D}$ et pour une solution initiale $u_{\mathcal{N}}^0(\boldsymbol{\mu})$ donnée, à chaque itération n de l'algorithme – avec $1\leq n\leq n_{max}$, où n_{max} est en entier strictement positif donné –, nous cherchons $u_{\mathcal{N}}^n(\boldsymbol{\mu})\in X_{\mathcal{N}}$ tel que

$$J\Big(\boldsymbol{\mu};u_{\mathcal{N}}^n(\boldsymbol{\mu})\Big)\Big(u_{\mathcal{N}}^{n+1}(\boldsymbol{\mu})-u_{\mathcal{N}}^n(\boldsymbol{\mu})\Big)=G\Big(\boldsymbol{\mu};u_{\mathcal{N}}^n(\boldsymbol{\mu})\Big), \tag{5.7}$$

où $J\big(\boldsymbol{\mu};u_{\mathcal{N}}^n(\boldsymbol{\mu})\big)\in\mathbb{R}^{\mathcal{N}}\times\mathbb{R}^{\mathcal{N}}$ est la matrice jacobienne, et $G\big(u_{\mathcal{N}}^n(\boldsymbol{\mu});\boldsymbol{\mu}\big)\in\mathbb{R}^{\mathcal{N}}$ est le vecteur résidu. Dans le cas des problèmes qui s'écrivent sous la forme de (5.2), nous pouvons écrire les termes de la jacobienne et du résidu de manière générique. Étudions comment construire la jacobienne. L'élément de la matrice situé à la $k^{\text{ème}}$ ligne et $i^{\text{ème}}$ colonne s'écrit

$$J_{ki}\Big(\boldsymbol{\mu};u_{\mathcal{N}}(\boldsymbol{\mu})\Big)=\sum_{j=1}^{\mathcal{N}}u_j(\boldsymbol{\mu})a\Big(\phi_i,\phi_j,\phi_k;\boldsymbol{\mu}\Big)+\sum_{j=1}^{\mathcal{N}}u_j(\boldsymbol{\mu})a\Big(\phi_j,\phi_i,\phi_k;\boldsymbol{\mu}\Big)+b\Big(\phi_i,\phi_k;\boldsymbol{\mu}\Big). \tag{5.8}$$

La $k^{\text{ème}}$ composante du vecteur résidu s'écrit

$$G_k\Big(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) = -a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), u_{\mathcal{N}}(\boldsymbol{\mu}), \phi_k; \boldsymbol{\mu}\Big) - b\Big(u, \phi_k; \boldsymbol{\mu}\Big) + f\Big(\phi_k; \boldsymbol{\mu}\Big). \tag{5.9}$$

3 Méthode des bases réduites

Regardons à présent comment mettre en oeuvre la méthode RBM sur ce type de problème. Nous proposons ici une approche générale qui peut être appliquée à tout problème, affine en paramètres, possédant des non-linéarités quadratiques. La technique proposée est basée sur l'idée de stocker le maximum d'information dans l'espace réduit. Par conséquent nous projetons aussi bien les tenseurs que les matrices et les vecteurs. Afin de décrire l'application de l'approximation RB nous introduisons, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, le tenseur $A(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}\times\mathcal{N}}$, la matrice $B(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ et enfin le vecteur $F(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$ définis par

$$\Big(A(\boldsymbol{\mu})\Big)_{ijk} = a\Big(\phi_i, \phi_j, \phi_k; \boldsymbol{\mu}\Big),\ 1 \le i, j, k \le \mathcal{N}, \tag{5.10}$$

$$\Big(B(\boldsymbol{\mu})\Big)_{ki} = b\Big(\phi_k, \phi_i; \boldsymbol{\mu}\Big),\ 1 \le i, k \le \mathcal{N}, \tag{5.11}$$

et

$$\Big(F(\boldsymbol{\mu})\Big)_{k} = f\Big(\phi_k; \boldsymbol{\mu}\Big),\ 1 \le k \le \mathcal{N}. \tag{5.12}$$

Notons que le tenseur $A(\boldsymbol{\mu})$ peut être considéré comme un vecteur de matrices. En d'autres termes, pour $k = 1, ..., \mathcal{N}$, nous pouvons définir $A^k(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ comme

$$A^k(\boldsymbol{\mu})_{ij} = \Big(A(\boldsymbol{\mu})\Big)_{ijk},\ 1 \le i, j \le \mathcal{N}. \tag{5.13}$$

En utilisant la notation introduite en (5.13), nous pouvons réécrire (5.6) sous forme matricielle :

$$u_{\mathcal{N}}(\boldsymbol{\mu})^T A^k(\boldsymbol{\mu}) u_{\mathcal{N}}(\boldsymbol{\mu}) + \Big(B(\boldsymbol{\mu})u\Big)_k = \Big(F(\boldsymbol{\mu})\Big),\ 1 \le k \le \mathcal{N}, \tag{5.14}$$

où $\big(B(\boldsymbol{\mu})u\big)_k$ et $\big(F(\boldsymbol{\mu})\big)_k$ désignent respectivement la $k^{\text{ème}}$ composante des vecteurs $B(\boldsymbol{\mu})$ et $F(\boldsymbol{\mu})$. Penchons-nous sur la mise en place de l'algorithme de Newton. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, à chaque itération $n = 1, ..., n_{max}$, nous cherchons $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que

$$J_{k,\cdot}\Big(\boldsymbol{\mu}; u^n_{\mathcal{N}}(\boldsymbol{\mu})\Big)\Big(u^{n+1}_{\mathcal{N}}(\boldsymbol{\mu}) - u^n_{\mathcal{N}}(\boldsymbol{\mu})\Big) = G_k\Big(\boldsymbol{\mu}; u^n_{\mathcal{N}}(\boldsymbol{\mu})\Big),\ 1 \le k \le \mathcal{N}, \tag{5.15}$$

où $J_{k,\cdot}\Big(\boldsymbol{\mu}; u^n_{\mathcal{N}}(\boldsymbol{\mu})\Big) \in \mathbb{R}^{\mathcal{N}}$ est la $k^{\text{ème}}$ ligne de la matrice jacobienne et $G_k\Big(\boldsymbol{\mu}; u^n_{\mathcal{N}}(\boldsymbol{\mu})\Big)$ est la $k^{\text{ème}}$ composante du vecteur résidu. Regardons comment s'exprime la matrice jacobienne

à l'aide de (5.10)-(5.12). Nous pouvons écrire, pour $1 \leq k, i \leq \mathcal{N}$:

$$\begin{aligned} J_{ki}\Big(\boldsymbol{\mu}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) &= \frac{\partial}{\partial u_i(\boldsymbol{\mu})} a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), u_{\mathcal{N}}(\boldsymbol{\mu}), \phi_k; \boldsymbol{\mu}\Big) + \frac{\partial}{\partial u_i(\boldsymbol{\mu})} b\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), \phi_k; \boldsymbol{\mu}\Big) \\ &\quad - \frac{\partial}{\partial u_i(\boldsymbol{\mu})} f\Big(\phi_k, \boldsymbol{\mu}\Big) \\ &= \sum_{l=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{N}} \frac{\partial}{\partial u_i(\boldsymbol{\mu})} u_l(\boldsymbol{\mu}) u_j(\boldsymbol{\mu}) a\Big(\phi_l, \phi_j, \phi_k; \boldsymbol{\mu}\Big) + \sum_{j=1}^{\mathcal{N}} \frac{\partial}{\partial u_i(\boldsymbol{\mu})} u_j(\boldsymbol{\mu}) b\Big(\phi_j, \phi_k; \boldsymbol{\mu}\Big) \\ &= \sum_{l=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{N}} \frac{\partial}{\partial u_i(\boldsymbol{\mu})} \Big(u_l(\boldsymbol{\mu}) \ \big(A(\boldsymbol{\mu})\big)_{ljk} \ u_j(\boldsymbol{\mu})\Big) + \sum_{j=1}^{\mathcal{N}} \frac{\partial}{\partial u_i(\boldsymbol{\mu})} \Big(\big(B(\boldsymbol{\mu})_{kj}\big) \ u_j(\boldsymbol{\mu})\Big) \\ &= \sum_{j=1}^{\mathcal{N}} u_j(\boldsymbol{\mu}) \ \big(A(\boldsymbol{\mu})\big)_{jik} + \sum_{j=1}^{\mathcal{N}} \big(A(\boldsymbol{\mu})\big)_{ijk} \ u_j(\boldsymbol{\mu}) + B_{ki}. \end{aligned} \tag{5.16}$$

En général, lorsque nous utilisons la discrétisation FE, $A(\boldsymbol{\mu})$ n'est jamais assemblée du fait de sa grande dimension : $\mathcal{N}^3$. Rappelons cependant que lorsque nous travaillons avec un espace de dimension réduite, nous pouvons alors nous permettre d'assembler entièrement le tenseur $A(\boldsymbol{\mu})$. Introduisons l'espace réduit W_N, de dimension N avec $N << \mathcal{N}$, engendré par les fonctions $\{\xi_1, ..., \xi_N\}$. Chaque fonction de base ξ_i, pour $i = 1, ..., N$, est la solution de (5.6) pour un vecteur de paramètres $\boldsymbol{\mu}_i$ donné. Nous ne développons pas ici la manière de sélectionner les vecteurs de paramètres. Définissons la projection $u_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ de $u_{\mathcal{N}}(\boldsymbol{\mu})$ sur l'espace W_N par

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu}) \xi_i. \tag{5.17}$$

Introduisons maintenant

$$\Xi = \big[\xi_1, ..., \xi_N\big] \ \in \mathbb{R}^{\mathcal{N} \times N}, \tag{5.18}$$

qui contient les fonctions de base de W_N exprimées dans la base $\big\{\phi_1, ..., \phi_{\mathcal{N}}\big\}$. Notons également $\hat{\xi}_{i,j}$ la $j^{\text{ème}}$ composante de ξ_i dans la base $\big\{\phi_1, ..., \phi_{\mathcal{N}}\big\}$. Nous avons

$$\hat{\xi}_{i,j} = \Big(\xi_i, \phi_j\Big)_{X_{\mathcal{N}}}, \tag{5.19}$$

où $\big(.,.\big)_{X_{\mathcal{N}}}$ est le produit scalaire associé à $X_{\mathcal{N}}$. En utilisant (5.18), nous pouvons réécrire (5.17)

$$u_N(\boldsymbol{\mu}) = \Xi^T u_{\mathcal{N}}(\boldsymbol{\mu}). \tag{5.20}$$

Les coefficients $u_{N_i}(\boldsymbol{\mu})$ sont donnés par

$$u_{N_i}(\boldsymbol{\mu}) = \Big(u_{\mathcal{N}}(\boldsymbol{\mu}), \xi_i\Big)_{X_{\mathcal{N}}} = \Big(u_{\mathcal{N}}(\boldsymbol{\mu}), \sum_{j=1}^{\mathcal{N}} \hat{\xi}_{i,j} \phi_i\Big)_{X_{\mathcal{N}}} = \sum_{j=1}^{\mathcal{N}} u_j \ \hat{\xi}_{i,j}, \tag{5.21}$$

Nous remarquons que la description de la $n^{\text{ème}}$ itération de l'algorithme de Newton donnée en (5.15) reste valable ici, à condition d'utiliser la version discrète des espaces et de remplacer $X_{\mathcal{N}}$ par W_N. Cela correspond à utiliser, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, la projection des

termes de (5.15) sur l'espace réduit W_N. En effet, en utilisant (5.20), nous commençons par définir $F_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ comme étant la projection de $F(\boldsymbol{\mu})$,

$$F_N(\boldsymbol{\mu}) = \Xi^T F(\boldsymbol{\mu}), \tag{5.22}$$

puis $B_N(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ la projection de $B(\boldsymbol{\mu})$,

$$B_N(\boldsymbol{\mu}) = \Xi^T B(\boldsymbol{\mu})\Xi, \tag{5.23}$$

et enfin $A_N(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N\times N}$ le tenseur de petite dimension. Nous proposons de construire $A_N(\boldsymbol{\mu})$ comme étant la projection de $A(\boldsymbol{\mu})$ sur W_N. Pour $1 \le i,j,k \le N$, nous avons

$$\begin{aligned}\Big(A_N(\boldsymbol{\mu})\Big)_{ijk} &= a\Big(\xi_i,\xi_j,\xi_k;\boldsymbol{\mu}\Big) = \sum_{l,m,h}^{\mathcal{N}} \hat{\xi}_{i,l}\hat{\xi}_{j,m}\hat{\xi}_{k,h}a\Big(\phi_l,\phi_m,\phi_h;\boldsymbol{\mu}\Big)\\ &= \sum_{l,m,h}^{\mathcal{N}} \hat{\xi}_{i,l}\hat{\xi}_{j,m}\hat{\xi}_{k,h}\Big(A(\boldsymbol{\mu})\Big)_{lmh} = \sum_{h=1}^{\mathcal{N}} \hat{\xi}_{k,h}\big(\Xi^T A^h(\boldsymbol{\mu})\Big)_{ij}\\ &= \sum_{h=1}^{\mathcal{N}} \hat{\xi}_{k,h}\Big(A_N^h(\boldsymbol{\mu})\Big)_{ij},\end{aligned} \tag{5.24}$$

où $A_N^h(\boldsymbol{\mu}) = \Xi^T A^h(\boldsymbol{\mu})\Xi$ est la projection de $A^h(\boldsymbol{\mu})$ sur W_N, pour $k = 1,...,\mathcal{N}$. Définissons le produit tensoriel $\star : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ comme suit :

$$a^T \star \Big(b^T A c\Big) \equiv \sum_{h=1}^{\mathcal{N}} a^h\Big(b^T A^h c\Big),\ \forall a,b,c \in \mathbb{R}^{\mathcal{N}},\ \forall A \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}\times\mathcal{N}}. \tag{5.25}$$

Nous pouvons alors écrire

$$\Big(A_N(\boldsymbol{\mu})\Big)_{ijk} = \sum_{h=1}^{\mathcal{N}} \hat{\xi}_{k,h}\Big(\xi_i^T A^h(\boldsymbol{\mu})\xi_j\Big) = \xi_k^T \star \Big(\xi_i^T A(\boldsymbol{\mu})\xi_j\Big),\ 1 \le i,j,k \le N. \tag{5.26}$$

Cependant, dans la pratique, nous proposons de réaliser la projection du tenseur $A(\boldsymbol{\mu})$ à l'aide d'un tenseur hybride $\Lambda(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}\times N}$ dont les éléments sont définis par

$$\Lambda_{ijk}(\boldsymbol{\mu}) = \Big(\Lambda^k(\boldsymbol{\mu})\Big)_{ij} = a\Big(\phi_i,\phi_j,\xi_k;\boldsymbol{\mu}\Big),\ 1 \le i,j \le \mathcal{N},\ 1 \le k \le N. \tag{5.27}$$

Nous pouvons alors utiliser ce tenseur hybride pour redéfinir les éléments de $A_N(\boldsymbol{\mu})$ comme suit :

$$\begin{aligned}\Big(A_N(\boldsymbol{\mu})\Big)_{ijk} &= a\Big(\xi_i,\xi_j,\xi_k;\boldsymbol{\mu}\Big) = \sum_{l,m=1}^{\mathcal{N}} \hat{\xi}_{i,l}\hat{\xi}_{j,m}a\Big(\phi_l,\phi_m,\xi_k;\boldsymbol{\mu}\Big)\\ &= \sum_{l,m=1}^{\mathcal{N}} \Xi_{li}\ \Xi_{mj}\ \Big(\Lambda^k(\boldsymbol{\mu})\Big)_{lm} = \Big(\Xi^T\ \Lambda^k\ \Xi\Big)_{ij}.\end{aligned} \tag{5.28}$$

Nous venons de montrer que nous pouvons écrire

$$A_N^k = \Xi^T\ \Lambda^k\ \Xi. \tag{5.29}$$

Cela implique que nous pouvons calculer seulement N matrices Λ^k, $k = 1, \ldots, N$, puis les projeter sur l'espace réduit W_N pour obtenir le tenseur A_N.

En utilisant (5.23) et (5.26) nous obtenons la projection de la matrice jacobienne $J_N \in \mathbb{R}^{N\times N}$. Pour chaque itération n de l'algorithme de Newton et pour tout $\boldsymbol{\mu} \in \mathcal{D}$ nous avons

$$\Big(J_N(\boldsymbol{\mu}; u_N^n(\boldsymbol{\mu}))\Big)_{ik} = \Big(A_N^k(\boldsymbol{\mu})\Big)_{.i}^T u_N^n(\boldsymbol{\mu}) + \Big(A_N^k(\boldsymbol{\mu})\Big)_{i.} u_N^n(\boldsymbol{\mu}) + \Big(B\Big)_{ik}, \tag{5.30}$$

où $\big(A_N^k(\boldsymbol{\mu})\big)_{.i}$ et $\big(A_N^k(\boldsymbol{\mu})\big)_{i.}$ sont respectivement la i^{eme} colonne et ligne de $A_N^k(\boldsymbol{\mu})$. En détaillant nous pouvons écrire

$$\Big(A_N^k(\boldsymbol{\mu})\Big)_{.i} = \Big[\xi_k^T \star \big(\xi_1^T A(\boldsymbol{\mu})\xi_i\big), ..., \xi_k^T \star \big(\xi_N^T A(\boldsymbol{\mu})\xi_i\big)\Big],\ 1 \le i, k \le N \tag{5.31}$$

$$\Big(A_N^k(\boldsymbol{\mu})\Big)_{i.} = \Big[\xi_k^T \star \big(\xi_i^T A(\boldsymbol{\mu})\xi_1\big), ..., \xi_k^T \star \big(\xi_i^T A(\boldsymbol{\mu})\xi_N\big)\Big],\ 1 \le i, k \le N. \tag{5.32}$$

De manière similaire la projection du vecteur résidu $G_N \in \mathbb{R}^N$ s'écrit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour $k = 1, ..., N$,

$$\Big(G_N(\boldsymbol{\mu}; u_N^n(\boldsymbol{\mu}))\Big)_k = \Big(F_N(\boldsymbol{\mu})\Big)_k - \Big(u_N^n(\boldsymbol{\mu})\Big)^T A_N^k(\boldsymbol{\mu}) u_N^n(\boldsymbol{\mu}) - \Big(B_N(\boldsymbol{\mu}) u_N^n(\boldsymbol{\mu})\Big)_k. \tag{5.33}$$

La sortie s'exprime alors de la manière suivante :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q\Big(\boldsymbol{\mu}\Big)\ell^q\Big(u_N(\boldsymbol{\mu})\Big)\right),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.34}$$

ou encore sous une forme vectorielle :

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \theta_\ell^q\Big(\boldsymbol{\mu}\Big) {L_N^q}^T\right) u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.35}$$

avec $(L_N^q)_i = \ell^q(\xi_i^{pr})$.

Regardons comment s'articule la décomposition *hors-ligne/en-ligne*. Les fonctions de base ξ_i^{pr}, $1 \le i \le N$, sont calculées à l'étape *hors-ligne*, ce qui rend possible la construction des matrices J_N^q, $1 \le q \le Q_j$, ainsi que des vecteurs G_N^q, $1 \le q \le Q_g$, et L_N^q , $1 \le q \le Q_\ell$. À l'étape *en-ligne*, pour chaque $\boldsymbol{\mu} \in \mathcal{D}$ donné, à la $n^{\text{ème}}$ itération de l'algorithme de Newton on assemble la matrice $J_N\big(\boldsymbol{\mu}; u_N^n(\boldsymbol{\mu})\big)$, et le vecteur $G_N\big(\boldsymbol{\mu}; u_N^n(\boldsymbol{\mu})\big)$, puis nous résolvons

$$J_N\Big(\boldsymbol{\mu}; u_N^n(\boldsymbol{\mu})\Big)\ \delta u_N^n(\boldsymbol{\mu}) = G_N\Big(\boldsymbol{\mu}; u_N^n(\boldsymbol{\mu})\Big), \tag{5.36}$$

où $\delta u_N^n(\boldsymbol{\mu}) = u_N^{n+1}(\boldsymbol{\mu}) - u_N^n(\boldsymbol{\mu})$. Une fois que l'algorithme de Newton a convergé, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, nous assemblons le vecteur $L_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \theta_\ell^q\big(\boldsymbol{\mu}\big) L_N^q$, et nous pouvons évaluer la sortie

$$s_N(\boldsymbol{\mu}) = L_N^T(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{5.37}$$

Étudions la complexité de l'étape *en-ligne*. Le coût de l'algorithme de Newton est prédominant sur le reste des opérations. À chaque itération nous devons assembler la matrice jacobienne et le vecteur résidu, ce qui necessite $\mathcal{O}(N^3)$ opérations. Ensuite le système (5.36) est résolu en $\mathcal{O}(N^3)$. La complexité de l'étape *en-ligne* ne dépend donc que de la (petite) dimension N, ainsi que du nombre d'itérations de l'algorithme de Newton. Elle ne dépend pas de la (grande) dimension $\mathcal{N}$.

Résumé

Nous avons vu dans ce chapitre comment appliquer la méthode RBM à des problèmes stationnaires avec non-linéarités quadratiques. Nous avons présenté une technique de projection, indépendante du choix de la base, qui peut être appliquée à tous les problèmes avec non-linéarités quadratiques afin de les résoudre uniquement dans un espace réduit, sans utiliser de projections sur l'espace (de grande dimension) élément fini. Quelques résultats numériques, obtenus en utilisant cette technique, seront présentés dans le chapitre 9.

Chapitre 6

Nouvelle construction de fonctions de base pour la méthode multi-échelles

Nous nous tournons à présent vers les problèmes de type multi-échelles. La particularité de tels problèmes est de manipuler un ensemble de phénomènes associés à des échelles différentes. C'est le cas par exemple des problèmes faisant intervenir des matériaux composites, lorsque les fibres sont positionnées de manière aléatoire alors le coefficient de conductivité thermique – ou électrique – peut subir des variations. C'est également le cas lorsque nous nous intéressons à des écoulements en milieu poreux.

La principale difficulté lorsque nous voulons traiter ce type de problèmes réside dans la taille (importante) du problème à résoudre une fois que ce dernier est discrétisé. En effet, afin de pouvoir capturer au mieux les variations de la solution au niveau des petites échelles, il faut avoir une discrétisation très fine du domaine étudié. Dans le même temps la solution est également influencée par d'autres phénomènes, à plus grande échelle. Pour la plupart de ces problèmes, il est très difficile de discrétiser l'ensemble du domaine étudié en se basant sur la plus petite échelle impliquée. Puis, en admettons que nous ayons une discrétisation très fine, se pose ensuite le problème de la résolution. Même si de nos jours de plus en plus d'algorithmes sont parallélisés, la taille du problème à résoudre ne diminue pas et sa résolution peut s'avérer extrêmement coûteuse.

Il existe plusieurs techniques permettant de prendre en compte cette cascade d'échelles. Certaines techniques se reposent sur l'utilisation de méthodes numériques très efficaces pour résoudre le problème multi-échelles discrétisé suivant chaque niveau d'échelle. C'est le cas de la méthode multigrille [21, 26, 130, 60], et de la méthode de décomposition de domaine [115, 116, 104, 34]. D'autres se basent sur le fait qu'il est parfois suffisant d'avoir le comportement "global" de la solution du problème, c'est-à-dire celui associé aux grandes échelles, sans devoir résoudre le problème sur les petites échelles. C'est le cas des méthodes d'homogénéisation [89, 109, 11, 38], variationnelle multi-échelles [68, 69, 22], hétérogène multi-échelles [129, 1, 127, 128] et élément fini multi-échelles [67, 66, 46].

Ici nous ne nous intéressons uniquement à la méthode élément fini multi-échelles. La première partie de ce chapitre est dédiée à l'énoncé du problème général, puis la méthodologie de la méthode élément fini multi-échelles – *Multiscale Finite Element Method (MsFEM)* – est présentée, enfin nous proposons une nouvelle construction des fonctions de base pour la méthode MsFEM.

1 Énoncé du problème général

Soit Ω un domaine borné, périodique, régulier, dans $\mathbb{R}^d$ (pour $d = 1, ..., 3$), de période ϵ. Définissons $\mathcal{T}_h = \{K_i, i = 1, ..., n_t\}$, avec n_t un entier strictement positif, un ensemble de n_t éléments formant une partition de Ω. Introduisons l'espace de fonctions X tel que $H_0^1(\Omega) \subset X \subset H^1(\Omega)$, ainsi que l'espace d'approximation FE, $X_{\mathcal{N}} = span\{\phi_1, ..., \phi_{\mathcal{N}}\}$ de (grande) dimension $\mathcal{N}$. L'utilisation de la méthode MsFEM permet de ne pas résoudre le problème sur les petites échelles, mais d'avoir le comportement de la solution sur les grandes échelles. L'ingrédient clé est la construction des fonctions de base. En effet, les fonctions de base élément fini multi-échelles – *Multiscale Finite Element (MsFE)* – s'adaptent aux propriétés locales du problème (par exemple le coefficient de conductivité thermique d'un matériau composite). La méthode MsFEM propose de construire ces fonctions de base en résolvant un problème local à chaque élément $K \in \mathcal{T}_h$.

Considérons le problème suivant : chercher $u \in X$ tel que, pour tout $\boldsymbol{x} \in \Omega$,

$$\begin{cases} \nabla \cdot \Big(\alpha_\epsilon(\boldsymbol{x}; \mu^0)\nabla u\Big) &= g, \quad \text{dans } \Omega, \\ u &= 0, \quad \text{sur } \partial\Omega, \end{cases} \tag{6.1}$$

où $\mu^0 \in \mathcal{D}^{\mu^0}$ est le paramètre d'oscillation, $\mathcal{D}^{\mu^0} \subset \mathbb{R}^1$ étant l'espace du paramètre μ^0. La fonction $\alpha_\epsilon(\boldsymbol{x}; \mu^0)$ oscille fortement – avec une période ϵ – à l'intérieur du domaine Ω et $g \in L^2(\Omega)$. Notons que ce problème sera notre problème de référence tout au long de ce chapitre. Il s'agit d'un problème de diffusion à coefficients variables qui peut être utilisé, par exemple, afin de connaître la présence d'huile et de gaz dans un milieu poreux. L'inconnue u est alors la pression du fluide et le coefficient $\alpha_\epsilon(\boldsymbol{x}; \mu^0)$ est la perméabilité du milieu. La formulation faible du problème (6.1) – en utilisant la discrétisation FE – consiste à chercher $u_{\mathcal{N}} \in X_{\mathcal{N}}$ tel que

$$a\Big(u_{\mathcal{N}}, v; \mu^0\Big) = f\Big(v\Big), \ \forall v \in X_{\mathcal{N}}. \tag{6.2}$$

Les formes bilinéaire $a : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D}^{\mu^0} \to \mathbb{R}$ et linéaire $f : X_{\mathcal{N}} \to \mathbb{R}$ sont définies par

$$a\Big(u_{\mathcal{N}}, v; \mu^0\Big) = \int_\Omega \alpha_\epsilon(\boldsymbol{x}; \mu^0) \ \nabla u_{\mathcal{N}} \cdot \nabla v \text{ et } f\Big(v\Big) = \int_\Omega g \ v. \tag{6.3}$$

La méthode de stabilisation des bulles à résidu libre – *ou residual free bubbles (RFB)* – [10, 22, 24, 52] peut être utilisée pour résoudre les problèmes du type (6.2). Définissons l'espace des fonctions bulles $X_{\mathcal{N}}^B$ par

$$X_{\mathcal{N}}^B = \prod_{K \in \mathcal{T}_h} X_{\mathcal{N}}^B(K), \tag{6.4}$$

où $X_{\mathcal{N}}^B(K)$ est défini sur l'élément $K \in \mathcal{T}_h$ par $X_{\mathcal{N}}^B(K) = H_0^1(K)$. L'espace $X_{\mathcal{N}}$ est alors enrichi en ajoutant des fonctions bulles sur chaque élément $K \in \mathcal{T}_h$. L'espace augmenté $X_{\mathcal{N}}^A$ est défini par

$$X_{\mathcal{N}}^A = X_{\mathcal{N}} \oplus X_{\mathcal{N}}^B. \tag{6.5}$$

Quant aux éléments de l'espace augmenté, ils s'écrivent sous la forme

$$u_{\mathcal{N}}^A = u_{\mathcal{N}} + u_{\mathcal{N}}^B, \tag{6.6}$$

avec $u_{\mathcal{N}} \in X_{\mathcal{N}}$ et $u^B_{\mathcal{N}} \in X^B_{\mathcal{N}}$. Lorsque nous appliquons la méthode des bulles à résidu libre, nous cherchons $u^A_{\mathcal{N}} \in X^A_{\mathcal{N}}$ tel que

$$a\Big(u^A_{\mathcal{N}}, v; \mu^0\Big) = f\Big(v\Big),\ \forall v \in X^A_{\mathcal{N}}. \tag{6.7}$$

2 Méthode élément fini multi-échelles

À présent introduisons l'espace d'approximation multi-échelles $X^{MS}_{\mathcal{N}} = span\{w^1_{\mathcal{N}^m}, ..., w^{\mathcal{N}}_{\mathcal{N}^m}\}$ de (grande) dimension $\mathcal{N}$, engendré par les fonctions de base multi-échelles. Un élément $u^{MS}_{\mathcal{N}}$ de cet espace s'écrit

$$u^{MS}_{\mathcal{N}} = \sum_{i=1}^{\mathcal{N}} u^{MS}_i\ w^i_{\mathcal{N}^m}. \tag{6.8}$$

Pour chaque élément $K \in \mathcal{T}_h$, les fonctions de base multi-échelles $w^i_{\mathcal{N}^m} \in X^{\phi_i}_{\mathcal{N}^m}(K)$ vérifient

$$a^i_{MS}\big(w^i_{\mathcal{N}^m}, v; \mu^0\big) = 0,\ \forall v \in X^{\phi_i}_{\mathcal{N}^m}(K),\ \text{pour } i = 1, ..., \mathcal{N}, \tag{6.9}$$

où pour $i = 1, ..., \mathcal{N}$, les formes bilinéaires $a^i_{MS} : X^{\phi_i}_{\mathcal{N}^m}(K) \times X^{\phi_i}_{\mathcal{N}^m}(K) \times \mathcal{D}^{\mu^0} \to \mathbb{R}$ sont définies par

$$a^i_{MS}\Big(w^i_{\mathcal{N}^m}, v; \mu^0\Big) = \int_K \alpha_\epsilon(\boldsymbol{x}; \mu^0)\ \nabla w^i_{\mathcal{N}^m} \cdot \nabla v \tag{6.10}$$

où h^K_s est la taille caractéristique du maillage de K et l'espace $X^{\phi_i}_{\mathcal{N}^m}(K)$ est un espace d'approximation FE de (grande) dimension $\mathcal{N}^m$ défini sur K qui impose la contrainte $w^i_{\mathcal{N}^m|\partial K} = \phi_{i|\partial K}$. Rappelons que les fonctions ϕ_i, $1 \leq i \leq \mathcal{N}$, sont les fonctions de base de l'espace d'approximation $X_{\mathcal{N}}$. Le problème (6.2) peut être reformulé de la manière suivante : chercher $u^{MS}_{\mathcal{N}} \in X^{MS}_{\mathcal{N}}$ tel que

$$a\Big(u^{MS}_{\mathcal{N}}, v; \mu^0\Big) = f\Big(v\Big),\ \forall v \in X^{MS}_{\mathcal{N}}. \tag{6.11}$$

Nous avons alors, d'après [23], que $u^{MS}_{\mathcal{N}}$ coïncide nodalement avec $u^A_{\mathcal{N}}$.

Regardons à présent comment mettre en oeuvre la méthode MsFEM. Pour cela nous nous plaçons sur l'élément de référence $\hat{K}$. Rappelons que pour tout $K \in \mathcal{T}_h$ nous notons $\mathcal{F}_{\hat{K},K}$ la transformation géométrique, inversible, de $\hat{K}$ vers K et $\mathcal{F}^{-1}_{\hat{K},K}$ son inverse (voir la section 2.4 du chapitre 1). Ces applications sont définies par

$$\begin{array}{rccc} \mathcal{F}_{\hat{K},K} : & \hat{K} \in \mathbb{R}^d & \to & K \in \mathbb{R}^d \\ & \hat{\boldsymbol{x}} & \to & \boldsymbol{x} \end{array} \quad \text{et} \quad \begin{array}{rccc} \mathcal{F}^{-1}_{\hat{K},K} : & K \in \mathbb{R}^d & \to & \hat{K} \in \mathbb{R}^d \\ & \boldsymbol{x} & \to & \hat{\boldsymbol{x}}. \end{array} \tag{6.12}$$

Nous pouvons alors écrire $\mathcal{T}_h = \{K = \mathcal{F}_{\hat{K},K}(\hat{K})\}$. En utilisant la méthode MsFEM, le problème (6.2) peut se reformuler comme suit : chercher $u^{MS}_{\mathcal{N}} \in X^{MS}_{\mathcal{N}}$ tel que pour tout $\hat{\boldsymbol{x}} \in \hat{K}$,

$$\sum_{K \in \mathcal{T}_h} \int_K \alpha_\epsilon\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}); \mu^0\big) \nabla u^{MS}_{\mathcal{N}} \cdot \nabla w^j_{\mathcal{N}^m} = \sum_{K \in \mathcal{T}_h} \int_K g\ w^j_{\mathcal{N}^m},\ 1 \leq j \leq \mathcal{N}. \tag{6.13}$$

Introduisons $J_{\mathcal{F}_{\hat{K},K}}$, $J^{-1}_{\mathcal{F}_{\hat{K},K}}$, $J^{-T}_{\mathcal{F}_{\hat{K},K}}$ et $|J_{\mathcal{F}_{\hat{K},K}}|$ comme étant respectivement la matrice jacobienne de $\mathcal{F}_{\hat{K},K}$, la matrice inverse de $J_{\mathcal{F}_{\hat{K},K}}$, la transposée de $J^{-1}_{\mathcal{F}_{\hat{K},K}}$ et le déterminant de $J_{\mathcal{F}_{\hat{K},K}}$. Notons N_{ldofs} le nombre de degrés de liberté sur l'élément de référence $\hat{K}$. Considérons les fonctions $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, définies sur l'élément $\hat{K}$ telles que

$$\int_K w^i_{\mathcal{N}^m} = \int_{\hat{K}} \hat{w}^{\hat{\imath}}_{\mathcal{N}^m} |J_{\mathcal{F}_{\hat{K},K}}|,\ 1 \leq i \leq \mathcal{N},\ \hat{\imath} = \mathcal{G}^{loc}_{K\to\hat{K}}(i), \tag{6.14}$$

et

$$\int_K \nabla w^i_{\mathcal{N}^m} = \int_{\hat{K}} \nabla \hat{w}^{\hat{\imath}}_{\mathcal{N}^m} J^{-T}_{\mathcal{F}_{\hat{K},K}} |J_{\mathcal{F}_{\hat{K},K}}|,\ 1 \leq i \leq \mathcal{N},\ \hat{\imath} = \mathcal{G}^{loc}_{K\to\hat{K}}(i), \tag{6.15}$$

où $\mathcal{G}^{loc}_{K\to\hat{K}}(i)$ est une application permettant d'obtenir l'indice local du degré de liberté associé à l'indice (global) i. En nous plaçant sur l'élément de référence $\hat{K}$ nous pouvons réécrire (6.13) comme suit :

$$\begin{aligned} &\sum_{i=1}^{\mathcal{N}} u_i^{MS} \sum_{K\in\mathcal{T}_h} \int_{\hat{K}} \alpha_\epsilon\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}); \mu^0\big)\ \nabla \hat{w}^{\hat{\imath}}_{\mathcal{N}^m} \Big(J^{-T}_{\mathcal{F}_{\hat{K},K}} J^{-1}_{\mathcal{F}_{\hat{K},K}}\Big) \nabla \hat{w}^{\hat{\jmath}\,T}_{\mathcal{N}^m}\ |J_{\mathcal{F}_{\hat{K},K}}| \\ &= \sum_{K\in\mathcal{T}_h} \int_{\hat{K}} g\circ\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\ \hat{w}^{\hat{\jmath}}_{\mathcal{N}^m}\ |J_{\mathcal{F}_{\hat{K},K}}|,\ 1 \leq \hat{\jmath} \leq N_{ldofs}, \end{aligned} \tag{6.16}$$

où $h^{\hat{K}}_s$ est la taille caractéristique du maillage de $\hat{K}$, ainsi que $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m} = w^i_{\mathcal{N}^m} \circ \mathcal{F}_{\hat{K},K}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$. Localement, pour tout élément $K \in \mathcal{T}_h$, nous devons donc assembler la matrice $M_K \in \mathbb{R}^{N_{ldofs}\times N_{ldofs}}$ et le vecteur $F_K \in \mathbb{R}^{N_{ldofs}}$ de la manière suivante :

$$M_K(\hat{\imath},\hat{\jmath}) = \int_{\hat{K}} \alpha_\epsilon\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}); \mu^0\big)\ \nabla \hat{w}^{\hat{\imath}}_{\mathcal{N}^m} \Big(J^{-T}_{\mathcal{F}_{\hat{K},K}} J^{-1}_{\mathcal{F}_{\hat{K},K}}\Big) \nabla \hat{w}^{\hat{\jmath}\,T}_{\mathcal{N}^m}\ |J_{\mathcal{F}_{\hat{K},K}}| \tag{6.17}$$

et

$$F_K(\hat{\jmath}) = \int_{\hat{K}} g\circ\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\ \hat{w}^{\hat{\jmath}}_{\mathcal{N}^m}\ |J_{\mathcal{F}_{\hat{K},K}}|,\ 1 \leq \hat{\imath} \leq N_{ldofs}. \tag{6.18}$$

Pour cela nous devons construire, sur chaque élément $K \in \mathcal{T}_h$, les fonctions de base multi-échelles $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m} \in X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K})$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, telles que

$$a^{\hat{\imath}}_{MS}\Big(\hat{w}^{\hat{\imath}}_{\mathcal{N}^m}, \hat{v}; \mu^0\Big) = 0,\ \forall \hat{v} \in X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K}),\ \hat{\imath} = 1, ..., N_{ldofs}, \tag{6.19}$$

où l'espace de fonctions $X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K})$ de (grande) dimension $\mathcal{N}^m$ est défini sur $\hat{K}$ et impose la contrainte $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m|\partial\hat{K}} = \hat{\phi}_{\hat{\imath}|\partial\hat{K}}$. Les formes bilinéaires $a^{\hat{\imath}}_{MS} : X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K}) \times X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K}) \times \mathcal{D}^{\mu^0} \to \mathbb{R}$, $1 \leq \hat{\imath} \leq N_{ldofs}$, sont définies par

$$a^{\hat{\imath}}_{MS}\Big(\hat{w}^{\hat{\imath}}_{\mathcal{N}^m}(\boldsymbol{\mu}), \hat{v}; \mu^0\Big) = \int_{\hat{K}} \alpha_\epsilon\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}); \mu^0\big)\ \nabla \hat{w}^{\hat{\imath}}_{\mathcal{N}^m} \Big(J^{-T}_{\mathcal{F}_{\hat{K},K}} J^{-1}_{\mathcal{F}_{\hat{K},K}}\Big) \nabla \hat{v}\ |J_{\mathcal{F}_{\hat{K},K}}|, \tag{6.20}$$

avec $\hat{\phi}_{\hat{\imath}} = \phi_i \circ \mathcal{F}_{\hat{K},K}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, ainsi que $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m} = w^i_{\mathcal{N}^m} \circ \mathcal{F}_{\hat{K},K}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$ et $\hat{v} = v \circ \mathcal{F}_{\hat{K},K}$.

3 Utilisation de la méthode des bases réduites

Étudions à présent comment utiliser la méthode CRBM pour traiter des problèmes multi-échelles, et notamment proposer une nouvelle construction des fonctions de base MsFE du type (6.2). Cette nouvelle stratégie de construction est basée sur une idée de C.Prud'homme présentée lors d'une conférence en 2009 [96].

Pour tout élément $K \in \mathcal{T}_h$, la transformation géométrique $\mathcal{F}_{\hat{K},K}$ peut être paramétrée par les coordonnées des points d'interpolation sur l'élément K. Ces points sont notés s_i, pour $i = 1, ..., N_{ldofs}$ et on peut écrire

$$\mathcal{F}_{\hat{K},K} = \sum_{i=1}^{N_{ldofs}} s_i \ \hat{\phi}_i. \tag{6.21}$$

Notons $\mu^i = s_i$ pour $i = 1, ..., N_{ldofs}$. Nous avons déjà évoqué le paramètre d'oscillation noté μ^0. Soit $\boldsymbol{\mu} = (\mu^0, \mu^1, ..., \mu^{N_{ldofs}}) \in \mathcal{D}$, où $\mathcal{D} \subset \mathbb{R}^{Nldofs+1}$ est l'espace des paramètres. Nous remarquons que contrairement à ce qui a été présenté jusqu'à maintenant dans cette thèse, il n'est pas nécessaire de s'occuper de la discrétisation de l'espace des paramètres associé à μ^i, pour $i = 1, ..., N_{ldofs}$, puisque cette dernière est donnée par le maillage. Seul $\mathcal{D}^{\mu^0}$ a besoin d'une discrétisation, voir (2.42) et (2.43). Sur chaque élément $K \in \mathcal{T}_h$, nous cherchons les fonctions de base multi-échelles $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m}(\boldsymbol{\mu}) \in X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K})$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, telles que

$$a^{\hat{\imath}}_{MS_{RB}}\Big(\hat{w}^{\hat{\imath}}_{\mathcal{N}^m}(\boldsymbol{\mu}), \hat{v}; \boldsymbol{\mu}\Big) = 0, \ \forall \hat{v} \in X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K}), \ \hat{\imath} = 1, ..., N_{ldofs}, \tag{6.22}$$

où les formes bilinéaires $a^{\hat{\imath}}_{MS_{RB}} : X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K}) \times X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K}) \times \mathcal{D} \to \mathbb{R}$ sont définies par

$$a^{\hat{\imath}}_{MS_{RB}}\Big(\hat{w}^{\hat{\imath}}_{\mathcal{N}^m}(\boldsymbol{\mu}), \hat{v}; \boldsymbol{\mu}\Big) = \int_{\hat{K}} \alpha_\epsilon\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}); \mu^0\big) \ \nabla \hat{w}^{\hat{\imath}}_{\mathcal{N}^m}\Big(J^{-T}_{\mathcal{F}_{\hat{K},K}} J^{-1}_{\mathcal{F}_{\hat{K},K}}\Big) \nabla \hat{v} \ |J_{\mathcal{F}_{\hat{K},K}}|, \tag{6.23}$$

avec $\hat{\phi}_{\hat{\imath}} = \phi_i \circ \mathcal{F}_{\hat{K},K}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, ainsi que $\hat{w}^{\hat{\imath}}_{\mathcal{N}^m} = w^i_{\mathcal{N}^m} \circ \mathcal{F}_{\hat{K},K}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$ et $\hat{v} = v \circ \mathcal{F}_{\hat{K},K}$. Afin de pouvoir mettre en place une stratégie *hors-ligne/en-ligne* efficace, nous faisons l'hypothèse que les formes bilinéaires $a^{\hat{\imath}}_{MS_{RB}}$, $1 \leq \hat{\imath} \leq N_{ldofs}$ dépendent de manière affine de $\boldsymbol{\mu}$. Autrement dit nous supposons qu'il existe des entiers positifs $Q_{a^{\hat{\imath}}_{MS_{RB}}}$ avec $\hat{\imath} = 1, ..., N_{ldofs}$, tels que nous puissions écrire, pour tout $\hat{u} \in X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K})$ ainsi que pour tout $\hat{v} \in X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K})$ et pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$a^{\hat{\imath}}_{MS_{RB}}\Big(\hat{u}, \hat{v}; \boldsymbol{\mu}\Big) = \sum_{q=1}^{Q_{a^{\hat{\imath}}_{MS_{RB}}}} \theta^q_{a^{\hat{\imath}}_{MS_{RB}}}\Big(\boldsymbol{\mu}\Big) \ a^{\hat{\imath},q}_{MS}\Big(\hat{u}, \hat{v}\Big), \ 1 \leq \hat{\imath} \leq N_{ldofs}, \tag{6.24}$$

où $\theta^q_{a^{\hat{\imath}}_{MS_{RB}}} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_{a^{\hat{\imath}}_{MS_{RB}}}$ sont des fonctions qui dépendent de $\boldsymbol{\mu}$.

Introduisons maintenant N_{ldofs} séquences d'espaces d'approximation emboîtés $W^{\hat{\imath}}_{N_{pr}}$ de dimension N, avec $1 \leq \hat{\imath} \leq N_{ldofs}$ et $N << \mathcal{N}^m$, tels que $W^{\hat{\imath}}_{1_{pr}} \subset W^{\hat{\imath}}_{2_{pr}} \subset ... \subset W^{\hat{\imath}}_{N_{max_{pr}}} \subset X^{\hat{\phi}_{\hat{\imath}}}_{\mathcal{N}^m}(\hat{K})$, avec N_{max} un entier positif. Considérons $S^{\hat{\imath}}_N = \{\boldsymbol{\mu}^{\hat{\imath}}_1, ..., \boldsymbol{\mu}^{\hat{\imath}}_N\}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, les ensembles qui contiennent chacun N jeux de paramètres sélectionnés à l'aide de l'algorithme glouton. Soit $S^{w,\hat{\imath}}_N$, avec $1 \leq \hat{\imath} \leq N_{ldofs}$, les ensembles qui vont

contenir, pour chaque $\boldsymbol{\mu} \in S_N^{\hat{\imath}}$, les solutions de (6.22). Ces ensembles s'écrivent, pour $\hat{\imath} = 1, ..., N_{ldofs}$,

$$S_N^{w,\hat{\imath}} = \left\{ \hat{w}_{\mathcal{N}^m}^{\hat{\imath}}(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in S_N^{\hat{\imath}} \right\}. \tag{6.25}$$

Comme $a_{MS_{RB}}^{\hat{\imath}}$ est une forme bilinéaire symétrique, définie positive, nous introduisons le produit scalaire, pour $\hat{\imath} = 1, ..., N_{ldofs}$, défini pour tout $\boldsymbol{\mu} \in \mathcal{D}$ (et en particulier pour $\boldsymbol{\mu}_{ref} \in \mathcal{D}$) par

$$\big((w, z)\big)_{a_{MS_{RB}}^{\hat{\imath}}, \boldsymbol{\mu}} \equiv a_{MS_{RB}}^{\hat{\imath}}\big(w, z; \boldsymbol{\mu}\big),\ \forall w, z \in X_{\mathcal{N}^m}^{\hat{\phi}_{\hat{\imath}}}(\hat{K}). \tag{6.26}$$

Nous appliquons ensuite le processus d'orthonormalisation de Gram-Schmidt, en utilisant le produit scalaire $\big((.,.)\big)_{a_{MS_{RB}}^{\hat{\imath}}, \boldsymbol{\mu}_{ref}}$ aux éléments des ensembles $S_N^{w,\hat{\imath}}$, pour avoir des fonctions de bases orthonormalisées $\hat{\xi}_n^{pr,\hat{\imath}}$, $1 \leq n \leq N$ et $1 \leq \hat{\imath} \leq N_{ldofs}$. Les solutions réduites $\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}) \in W_{N_{pr}}^{\hat{\imath}}$ s'écrivent, pour $\hat{\imath} = 1, ..., N_{ldofs}$,

$$\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}) = \sum_{j=1}^{N} \hat{w}_{N_j}^{\hat{\imath}}(\boldsymbol{\mu})\ \hat{\xi}_j^{pr,\hat{\imath}}. \tag{6.27}$$

Nous allons à présent utiliser la dépendance affine en paramètres – (6.24) et (6.36) – pour construire une stratégie *hors-ligne/en-ligne* efficace. En choisissant les fonctions test comme $v = \hat{\xi}_n^{pr,\hat{\imath}}$, pour $n = 1, ..., N$, alors pour tout $\boldsymbol{\mu} \in \mathcal{D}$, les $\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu})$ sont solutions de

$$\sum_{j=1}^{N} \left(\sum_{q=1}^{Q_{a_{MS_{RB}}^{\hat{\imath}}}} \theta_{Q_{a_{MS_{RB}}^{\hat{\imath}}}}^{q}\big(\boldsymbol{\mu}\big)\ a_{MS}^{\hat{\imath},q}\big(\hat{\xi}_n^{pr,\hat{\imath}}, \hat{\xi}_j^{pr,\hat{\imath}}\big) \right)\ \hat{w}_{N_j}^{\hat{\imath}}(\boldsymbol{\mu}) = 0, \tag{6.28}$$

pour $\hat{\imath} = 1..., N_{ldofs}$. Nous pouvons réécrire (6.28) sous la forme matricielle :

$$\left(\sum_{q=1}^{a_{MS_{RB}}^{\hat{\imath}}} \theta_{a_{MS_{RB}}^{\hat{\imath}}}^{q}\big(\boldsymbol{\mu}\big) A_{MS_{RB},N}^{\hat{\imath},q} \right) \hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}) = 0,\ 1 \leq \hat{\imath} \leq N_{ldofs}, \tag{6.29}$$

avec, pour $\hat{\imath} = 1, ..., N_{ldofs}$,

$$\left(\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu})\right)_j = \hat{w}_{N_j}^{\hat{\imath}}(\boldsymbol{\mu}) \text{ et } \left(A_{MS_{RB},N}^{\hat{\imath},q}\right)_{jn} = a_{MS_{RB}}^{\hat{\imath},q}\left(\hat{\xi}_n^{pr,\hat{\imath}}, \hat{\xi}_j^{pr,\hat{\imath}}\right). \tag{6.30}$$

Nous venons de voir comment calculer les solutions réduites $\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}) \in W_{N_{pr}}^{\hat{\imath}}$ pour $\hat{\imath} = 1, ..., N_{ldofs}$. Nous ne détaillons pas ici les aspects liés à la mise en place des estimateurs d'erreur, le chapitre 2 traite déjà un cas similaire. Il ne faut pas perdre de vue que nous souhaitons utiliser les solutions $\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}) \in W_{N_{pr}}^{\hat{\imath}}$ pour $\hat{\imath} = 1, ..., N_{ldofs}$, dans l'assemblage de M_K et F_K pour tout $K \in \mathcal{T}_h$ – voir (6.17) et (6.18) –. Pour tout $K \in \mathcal{T}_h$, en nous servant des solutions réduites, pour un jeu de paramètre $\boldsymbol{\mu} \in \mathcal{D}$ approprié, nous avons

$$M_K^{\boldsymbol{\mu}}\left(\hat{\imath}, \hat{\jmath}\right) = a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}\left(\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}), \hat{w}_N^{\hat{\jmath}}(\boldsymbol{\mu}); \boldsymbol{\mu}\right),\ 1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}, \tag{6.31}$$

et

$$F_K^{\boldsymbol{\mu}}\left(\hat{\jmath}\right) = f_{\hat{K}}^{\hat{\jmath}}\left(\hat{w}_N^{\hat{\jmath}}(\boldsymbol{\mu}); \boldsymbol{\mu}\right),\ 1 \leq \hat{\jmath} \leq N_{ldofs}, \tag{6.32}$$

où les formes bilinéaires $a_{\hat{K}}^{\hat{\imath},\hat{\jmath}} : W_{N_{pr}}^{\hat{\imath}} \times W_{N_{pr}}^{\hat{\jmath}} \times \mathcal{D} \to \mathbb{R}$ et linéaires $f_{\hat{K}}^{\hat{\jmath}} : W_{N_{pr}}^{\hat{\jmath}} \times \mathcal{D} \to \mathbb{R}$, $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, sont définies par

$$a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}\Big(\hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu}), \hat{w}_N^{\hat{\jmath}}(\boldsymbol{\mu}); \boldsymbol{\mu}\Big) = \int_{\hat{K}} \alpha_\epsilon\big(\mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}}); \mu^0\big)\ \nabla \hat{w}_N^{\hat{\imath}}(\boldsymbol{\mu})\Big(J_{\mathcal{F}_{\hat{K},K}}^{-T} J_{\mathcal{F}_{\hat{K},K}}^{-1}\Big)\nabla \hat{w}_N^{\hat{\jmath}\,T}(\boldsymbol{\mu})\ |J_{\mathcal{F}_{\hat{K},K}}| \tag{6.33}$$

et

$$f_{\hat{K}}^{\hat{\jmath}}\Big(\hat{w}_N^{\hat{\jmath}}(\boldsymbol{\mu}); \boldsymbol{\mu}\Big) = \int_{\hat{K}} g \circ \mathcal{F}_{\hat{K},K}(\hat{\boldsymbol{x}})\ \hat{w}_N^{\hat{\jmath}}(\boldsymbol{\mu})\ |J_{\mathcal{F}_{\hat{K},K}}|. \tag{6.34}$$

Là encore nous faisons l'hypothèse que les formes bilinéaires $a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}$, $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, ainsi que les formes linéaires $f_{\hat{K}}^{\hat{\jmath}}$, $1 \leq \hat{\jmath} \leq N_{ldofs}$, dépendent de manière affine de $\boldsymbol{\mu}$. Autrement dit nous supposons qu'il existe des entiers positifs $Q_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}$,$Q_{f_{\hat{K}}^{\hat{\jmath}}}$ avec $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, tels que nous puissions écrire, pour tout $\hat{u} \in W_{N_{pr}}^{\hat{\imath}}$ ainsi que pour tout $\hat{v} \in W_{N_{pr}}^{\hat{\jmath}}$ et pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}\Big(\hat{u}, \hat{v}; \boldsymbol{\mu}\Big) = \sum_{q=1}^{Q_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}} \theta_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}^{q}\Big(\boldsymbol{\mu}\Big)\ a_{\hat{K}}^{\hat{\imath},\hat{\jmath},q}\Big(\hat{u}, \hat{v}\Big),\ 1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}, \tag{6.35}$$

et

$$f_{\hat{K}}^{\hat{\jmath}}\Big(\hat{v}; \boldsymbol{\mu}\Big) = \sum_{q=1}^{Q_{f_{\hat{K}}^{\hat{\jmath}}}} \theta_{f_{\hat{K}}^{\hat{\jmath}}}^{q}\Big(\boldsymbol{\mu}\Big)\ f_{\hat{K}}^{\hat{\jmath},q}\Big(\hat{v}\Big),\ 1 \leq \hat{\jmath} \leq N_{ldofs}, \tag{6.36}$$

où $\theta_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}^{q} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}$, et $\theta_{f_{\hat{K}}^{\hat{\jmath}}}^{q} : \mathcal{D} \to \mathbb{R}$, $1 \leq q \leq Q_{f_{\hat{K}}^{\hat{\jmath}}}$ sont des fonctions qui dépendent de $\boldsymbol{\mu}$. Introduisons les matrices $A_{\hat{K},N}^{\hat{\imath},\hat{\jmath},q} \in \mathbb{R}^{N \times N}$, pour $q = 1, ..., Q_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}$ et avec $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, telles que

$$A_{\hat{K},N}^{\hat{\imath},\hat{\jmath},q}\Big(i, j\Big) = a_{\hat{K}}^{\hat{\imath},\hat{\jmath},q}\Big(\hat{\xi}_i^{pr,\hat{\imath}}, \hat{\xi}_j^{pr,\hat{\jmath}}\Big),\ \text{pour } 1 \leq i, j \leq N. \tag{6.37}$$

Nous pouvons alors écrire, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ ainsi que pour $1 \leq i, j \leq N_{ldofs}$,

$$M_K^{\boldsymbol{\mu}}\Big(\hat{\imath}, \hat{\jmath}\Big) = \hat{w}_N^{\hat{\imath}\,T}(\boldsymbol{\mu})\ A_{\hat{K},N}^{\hat{\imath},\hat{\jmath}}\ \hat{w}_N^{\hat{\jmath}}(\boldsymbol{\mu}), \tag{6.38}$$

avec

$$A_{\hat{K},N}^{\hat{\imath},\hat{\jmath}} = \sum_{q=1}^{Q_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}} \theta_{a_{\hat{K}}^{\hat{\imath},\hat{\jmath}}}^{q}(\boldsymbol{\mu})\ A_{\hat{K},N}^{\hat{\imath},\hat{\jmath},q}. \tag{6.39}$$

Introduisons maintenant les vecteurs $F_{\hat{K},N}^{\hat{\jmath},q} \in \mathbb{R}^N$, pour $q = 1, ..., Q_{f_{\hat{K}}^{\hat{\jmath}}}$ et avec $1 \leq \hat{\jmath} \leq N_{ldofs}$, tels que

$$F_{\hat{K},N}^{\hat{\jmath},q}\Big(j\Big) = f_{\hat{K}}^{\hat{\jmath},q}\Big(\hat{\xi}_j^{pr,\hat{\jmath}}\Big),\ \text{pour } 1 \leq j \leq N. \tag{6.40}$$

Nous pouvons alors écrire, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ ainsi que pour $1 \leq j \leq N_{ldofs}$,

$$F_K^{\boldsymbol{\mu}}\Big(\hat{\jmath}\Big) = \hat{w}_N^{\hat{\jmath}\,T}(\boldsymbol{\mu})\ F_{\hat{K},N}^{\hat{\jmath}}, \tag{6.41}$$

avec

$$F_{\hat{K},N}^{\hat{\jmath}} = \sum_{q=1}^{Q_{f_{\hat{K}}^{\hat{\jmath}}}} \theta_{f_{\hat{K}}^{\hat{\jmath}}}^{q}\big(\boldsymbol{\mu}\big)\ F_{\hat{K},N}^{\hat{\jmath},q}. \tag{6.42}$$

Penchons-nous à présent sur l'articulation des phases *hors-ligne* et *en-ligne*. Pendant l'étape *hors-ligne*, il nous faut commencer par construire les N_{ldofs} bases réduites sur lesquelles vont s'appuyer les approximations des fonctions de base multi-échelles définies sur $\hat{K}$. Pour cela, NN_{ldofs} résolutions (coûteuses) de (6.22) sont nécessaires. Il faut ensuite $\mathcal{O}(N_{ldofs}Q_{a^{\hat{\imath}}_{MS_{RB}}}N^2\mathcal{N}^m)$ produits scalaires pour les étapes de projection sur les bases réduites. Puis, pour $\hat{\imath}$ et $\hat{\jmath}$ fixés tels que $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, il faut effectuer encore $\mathcal{O}(Q_{a^{\hat{\imath},\hat{\jmath}}_{\hat{K}}}N^2\mathcal{N}^m)$ produits scalaires pour construire les matrices $A^{\hat{\imath},\hat{\jmath},q}_{\hat{K},N}$, pour $q = 1, ..., Q_{a^{\hat{\imath},\hat{\jmath}}_{\hat{K}}}$ – voir (6.37) –. Une fois arrivé à l'étape *en-ligne*, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, il faut compter $\mathcal{O}(Q_{a^{\hat{\imath}}_{MS_{RB}}}N^2N_{ldofs})$ multiplications pour assembler les systèmes réduits et $\mathcal{O}(N^3)$ opérations pour les résoudre afin d'avoir les fonctions de base multi-échelles définies sur $\hat{K}$. Puis pour $\hat{\imath}$ et $\hat{\jmath}$ fixés tels que $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, $\mathcal{O}(Q_{a^{\hat{\imath},\hat{\jmath}}_{\hat{K}}}N^2)$ opérations sont nécessaires pour assembler les matrices $A^{\hat{\imath},\hat{\jmath},q}_{\hat{K},N}$, pour $q = 1, ...Q_{a^{\hat{\imath},\hat{\jmath}}_{\hat{K}}}$, ainsi que les vecteurs $F^{\hat{\jmath},q}_{\hat{K},N}$, pour $q = 1, ...Q_{f^{\hat{\jmath}}_{\hat{K}}}$. Les calculs de $M^{\boldsymbol{\mu}}_K\left(\hat{\imath},\hat{\jmath}\right)$ et de $F^{\boldsymbol{\mu}}_K\left(\hat{\jmath}\right)$, pour $1 \leq \hat{\imath}, \hat{\jmath} \leq N_{ldofs}$, entraînent $\mathcal{O}(N^2)$ opérations. La complexité de l'étape *en-ligne* ne dépend donc pas de la (grande) dimension $\mathcal{N}^m$.

Remarque 12. *Durant la partie en-ligne, du point de vue informatique, il peut être envisagé de solliciter les processeurs graphiques – Graphics Processing Units (GPU) – pour l'assemblage des contributions locales M_K et F_K pour tout $K \in \mathcal{T}_h$ – voir (6.17) et (6.18) –, puis résoudre sur le processeur central – Central Processing Unit (CPU) – le problème multi-échelle (6.11). Prenons deux éléments distincts $K_1 \in \mathcal{T}_h$ et $K_2 \in \mathcal{T}_h$. Le calcul de M_{K_1} et indépendant du calcul de M_{K_2}. De même le calcul de F_{K_1} et indépendant du calcul de F_{K_2}. Il est donc possible de calculer les différentes contributions locales M_K et F_K pour tout $K \in \mathcal{T}_h$ en parallèle. Rappelons que les processeurs graphiques permettent de faire des calculs, de préférence en parallèle, à l'aide d'un circuit intégré.*

Résumé

Après avoir présenté le problème de diffusion elliptique que nous avons traité tout au long du chapitre, le principe de la méthode MsFEM a été exposé. Ensuite, nous avons vu comment pouvait intervenir la méthode CRBM dans la construction des fonctions de base MsFE. L'intérêt de cette nouvelle construction est de ne plus dépendre de la dimension $\mathcal{N}^m$ et donc obtenir ces fonctions de base de manière très rapide et fiable. De plus, comme indiqué dans la remarque 12, cette nouvelle construction se prête à l'utilisation des architectures hybrides. Faute de temps, aucun résultat numérique n'a été présenté. Nous pensons cependant, au vu de la nature du coefficient α_ϵ, que l'approximation base réduite ne sera pas suffisante. Il faudra probablement envisager d'utiliser une approximation de type "hp" [48, 49].

Deuxième partie

Mise en oeuvre

Chapitre 7

Architecture logicielle et interface utilisateur

Nous ouvrons cette deuxième partie en présentant l'architecture logicielle sur laquelle repose l'implémentation de la méthode RBM dans Feel++. Nommons framework-RBM le framework dédié à la méthode RBM. Ce framework s'appuie sur Feel++, notamment pour fournir les approximations FE nécessaires à la méthode RBM ainsi que pour les aspects liés à la parallélisation du code (nous y reviendrons plus en détails dans le chapitre 8).

Au commencement de cette thèse, ce framework était capable de calculer la sortie du modèle, basée sur les solutions réduites primale et duale, de problèmes elliptiques linéaires coercifs dans le cas mono-physique (2.30). Une implémentation des différents ingrédients nécessaires au calcul de l'estimation d'erreur a posteriori – une approximation de la borne inférieure de la constante de coercivité via la méthode des contraintes successives (2.170), et le calcul de la norme duale des différents résidus (2.91) et (2.92) – était également disponible. Durant cette thèse, le framework-RBM a évolué afin

- de finaliser la certification de la sortie du modèle (2.93), ainsi que des solutions réduites primale et duale (2.94), dans le cas de problèmes elliptiques linéaires coercifs ;
- d'appliquer la méthode CRBM aux problèmes linéaires paraboliques coercifs (2.74), (2.122) ;
- de traiter les problèmes elliptiques avec non-linéarités quadratiques (5.37) ;
- d'appliquer la méthode CRBM aux problèmes linéaires coercifs n'admettant pas une décomposition affine en paramètres (3.45), (3.58) ;
- d'appliquer la méthode RBM aux problèmes non-linéaires ellitpiques coercifs multi-physiques (3.203), (4.19), (4.42), (4.63) ;
- de supporter les architectures parallèles (voir le chapitre 8).

De plus, à l'origine, le framework-RBM n'offrait qu'une vision algébrique de la méthode RBM. Une autre évolution a consisté à proposer une vision fonctionnelle et ainsi à pouvoir manipuler de la même manière un élément de l'espace d'approximation FE ou RB.

Tout au long de ce travail, il a été porté une attention particulière à ce que cette complexité croissante des modèles ne surcharge pas inutilement l'interface utilisateur.

La première partie de ce chapitre sera dédiée à l'architecture logicielle du framework-RBM alors que la deuxième partie sera tournée vers l'interface utilisateur.

1 Architecture logicielle

Tout d'abord nous présentons le framework-RBM de Feel++ de manière générale, avec entre autres, une vue d'ensemble du processus de réduction d'ordre en utilisant ce framework (figure 7.1). Après cela, nous nous penchons plus en détails sur l'espace d'approximation RB, qui permet d'avoir une vision fonctionnelle (à l'opposé d'algébrique) du framework-RBM. Puis nous verrons comment mettre en oeuvre, de manière efficace, la partie *en-ligne* de la méthode EIM – notamment en nous appuyant sur l'aspect fonctionnel du code et sur la méta-programmation –. Enfin, nous verrons quelques outils mis en place pour optimiser la gestion de la mémoire.

1.1 Présentation générale du framework-RBM

Le framework-RBM fournit une interface pour appliquer la méthodologie base réduite décrite dans la première partie de cette thèse (voir les chapitres 2 à 5). A la fin du processus de réduction, il est possible de lancer l'évaluation de la sortie du modèle de différentes façons : directement via la ligne de commande, ou en utilisant les interfaces de Python ou Octave, afin d'évaluer de manière rapide et fiable la sortie définie par

$$\left[s_N^1(\boldsymbol{\mu}), s_N^2(\boldsymbol{\mu}), \ldots, s_N^{n_s}(\boldsymbol{\mu})\right] = \ell(\boldsymbol{\mu}), \text{ où } n_s \text{ un entier strictement positif,} \tag{7.1}$$

ou alors, pour un temps t donné,

$$\left[s_N^1(\boldsymbol{\mu}, t), s_N^2(\boldsymbol{\mu}, t), \ldots, s_N^{n_s}(\boldsymbol{\mu}, t)\right] = \ell(\boldsymbol{\mu}; t). \tag{7.2}$$

À présent ℓ est le code de simulation qui prend en entrée le jeu de paramètres $\boldsymbol{\mu}$ et retourne les sorties $s_N^i(\boldsymbol{\mu})$ – ou $s_N^i(\boldsymbol{\mu}, t)$ – , pour $i = 1, ..., n_s$, qui sont un ensemble de métriques accompagnées, lorsque c'est possible, de l'estimation d'erreur associée.

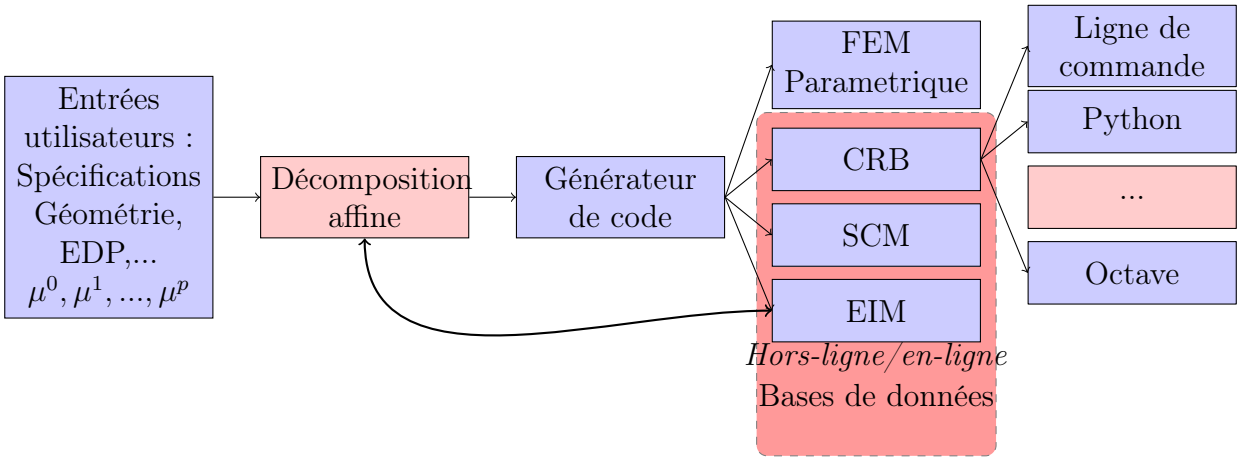


Figure 7.1 – Processus de réduction d'ordre en utilisant le Framework-RBM de Feel++

L'utilisateur doit fournir les spécifications du modèle sur lequel il souhaite appliquer la méthodologie base réduite : l'espace des paramètres $\mathcal{D}$ (extremums), la géométrie du

domaine, la formulation variationnelle du problème à traiter en faisant apparaître la décomposition affine en paramètres (la méthode EIM peut être utilisée au besoin, nous reviendrons en détails sur ce point dans dans la section 2.2). Enfin, le programme est créé via l'outil `CMAKE` [82, 83], qui génère le `makefile` utilisé pour la compilation. Le design des classes `C++` du framework-RBM est illustré sous forme de diagramme UML – *Unified Modeling Language* [5, 108] – par la figure 7.2.

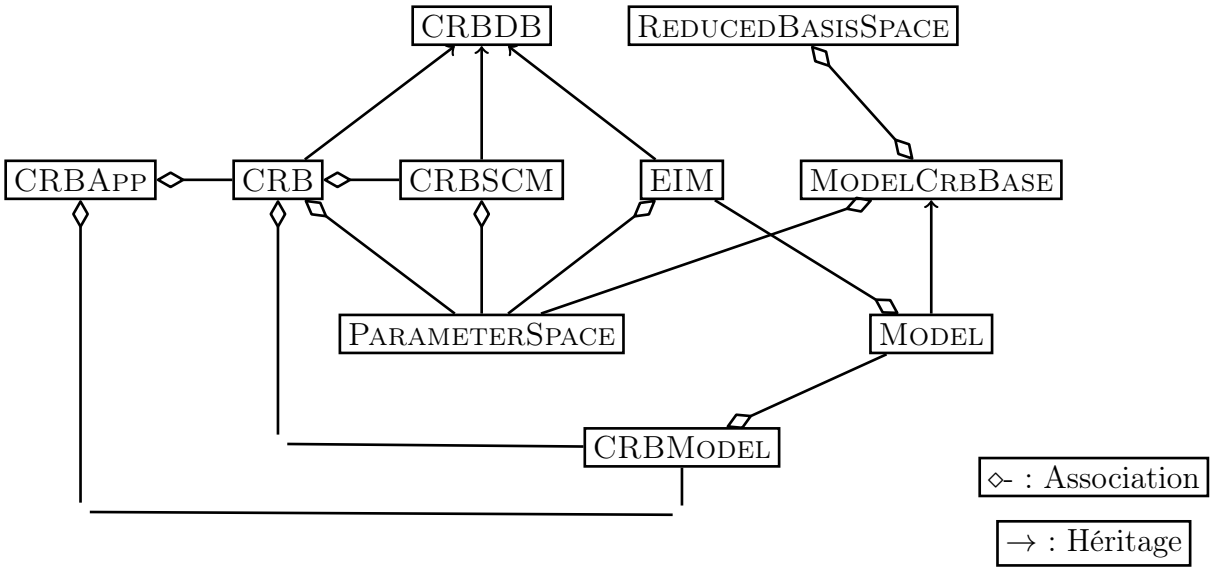


FIGURE 7.2 – Diagramme UML des classes utilisées par le framework-RBM.

La classe ParameterSpace génère et stock les échantillons de l'espace de paramètres – voir (2.42) et (2.43), mais d'autres stratégies sont proposées par le framework-RBM , il est possible de générer des paramètres log-équidistribués ou log-aléatoires –. Ces échantillons sont ensuite utilisés aussi bien par CRB, SCM que EIM.

CRBModel est l'interface que doivent respecter les modèles coercifs pour utiliser le framework. C'est à travers cette interface que l'utilisateur fournit les specifications du modèle mentionnées précédemment. Dans le cas des problèmes avec non-linéarités quadratiques, l'interface est donnée par CRBModelTrilinear.

Tous les modèles utilisateurs dérivent de ModelCrbBase. Cette classe permet d'absorber la complexité des modèles. En effet le framework-RBM demande à ce qu'un certain nombre de fonctions soient déclarées (et implémentées) par les modèles. Cependant les modèles qui ne se servent pas de la méthode EIM n'ont pas à se préoccuper des fonctions liées à l'utilisation de cette méthode car elles sont fournies par ModelCrbBase. Le même principe est appliqué lorsqu'il existe plusieurs manières de donner la même information, comme c'est le cas pour fournir les termes provenant de la décomposition affine (plus de détails sont donnés dans la section 2), l'utilisateur choisit d'implémenter une fonction sans se préoccuper des autres fonctions qui, au final, donnent les mêmes informations au framework-RBM.

Comme la phase de calculs *hors-ligne* peut s'avérer très coûteuse, les produits scalaires résultants de la projection des matrices et des vecteurs sur les espaces réduits (primal et dual), voir (2.17),(2.37),(2.62) et (2.79), sont sauvegardés dans une base de données propre à la classe CRB. Il en est de même pour les produits scalaires définis en (2.106),

(2.107), (2.116), (2.117) (2.140), (2.141), (2.142) (2.152) et (2.153) utilisés pour fournir une estimation d'erreur a posteriori. Concernant la méthode SCM, les quantités introduites par (2.169), (2.173) et (2.176) sont également stockées dans une base de donnée propre à la classe SCM. Pour la méthode EIM, la matrice d'interpolation ainsi que les fonctions de base (3.10), (3.13) sont également stockées dans une base de donnée propre à la classe EIM.

Pour stocker ces quantités, nous utilisons le concept de "sérialisation" introduit par le groupe de librairies dédié à la programmation `C++` : `Boost`. La classe CRBDB contient la stratégie de stockage des différentes bases de données. Elle s'appuie sur la librairie `Boost.serialization`. Pendant la phase *en-ligne* nous manipulons les matrices denses et les vecteurs via la librairie `Eigen` [57].

La classe CRB représente l'implémentation de la méthode RBM pour les problèmes coercifs, elliptiques et paraboliques, linéaires et non linéaires. Par défaut, lorsque ce sont des problèmes elliptiques qui sont traités, l'algorithme glouton est utilisé, alors que pour les problèmes paraboliques c'est l'algorithme POD/glouton – POD en temps et glouton en espace – qui est utilisé. Il est à noter que dans les deux cas, l'utilisateur a la possibilité de spécifier dans un fichier l'échantillon de paramètres à utiliser pour remplir S_N, qui sert à la construction des espaces d'approximation primal et dual. Dans le cas des problèmes avec non-linéarités quadratiques , l'implémentation de la méthode RBM se trouve dans la classe CRBTrilinear Pour les équations linéaires, les solveurs linéaires de `Eigen` sont utilisés, alors que pour les équations non-linéaires nous nous servons des solveurs de `PETSc`.

Afin d'interagir avec les solveurs de `PETSc`, nous avons mis en place un "pont" `Eigen :: Map<>` pour communiquer les données non-linéaires entre `Eigen` et `PETSc`. Sur le diagramme de la figure 7.2, pour plus de lisibilité, les classes CRBTrilinear et CRBModelTrilinear n'apparaissent pas. Elles dérivent respectivement des classes CRB et CRBModel.

La classe CRBSCM implémente la méthode des contraintes successives à l'aide des solveurs aux valeurs propres généralisés de la librairie `SLEPc`.

EIM quant à elle contient l'implémentation de la méthode EIM.

ReducedBasisSpace offre la possibilité de manipuler les éléments des epsaces d'approximation RB (évaluation, calcul du gradient, ...). Elle est également en charge de la base de données – via `Boost.serialization` – où sont stockés les fonctions de base ξ_i^{pr} et ξ_i^{du}, pour $i = 1, ..., N$, où N est la dimension RB.

Remarque 13. *Il est important que les fonctions de base RB soient stockées dans une base de données autre que celle associée à la classe* CRB. *En effet, si les espaces $W_{N_{pr}}$ et $W_{N_{du}}$ ont déjà été générés et que l'on ne souhaite pas les enrichir, alors seule la base de données associée à la classe* CRB *est chargée (rappelons qu'elle contient le résultat de la projection des matrices et des vecteurs sur les espaces réduits). De cette manière on ne dépend pas de la dimension FE : $\mathcal{N}$.*

La classe CRBApp sert de pilote pour le framework-RBM.

Enfin, une analyse de sensibilité peut être effectuée en utilisant la librairie scientifique `OpenTURNS` [131], spécialisée dans le traitement des incertitudes. À travers un script python, l'utilisateur génère un ensemble de paramètres d'entrées suivant une loi de distribution, à partir desquels `OpenTURNS` construit un échantillon de ces paramètres qu'il donne à la classe CRBApp. Ensuite, durant la phase *en-ligne* de la méthode RBM , CR-

Bapp fournit les sorties du modèle associées à `OpenTURNS` qui calcule des quantités telles que les indices de Sobol, l'écart-type ou les quantiles.

1.2 Espace d'approximation RB

Nous présentons ici la classe ReducedBasisSpace plus en détails. Tout d'abord nous faisons le lien avec la classe dédiée à l'espace d'approximation FE dans Feel++, puis nous introduisons la notion de Context qui est importante pour la suite. Enfin nous nous penchons sur la notion d'élément attaché à un espace de fonctions.

Lien avec l'espace d'approximation FE de Feel++

Commençons par nous intéresser à l'espace d'approximation FE, $X_{\mathcal{N}}$. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, nous considérons $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ défini par

$$u_{\mathcal{N}}(\boldsymbol{\mu}) = \sum_{j=1}^{\mathcal{N}} u_{\mathcal{N}}^{j}(\boldsymbol{\mu})\ \phi_j, \tag{7.3}$$

où ϕ_j, pour $j = 1, ..., \mathcal{N}$, sont les fonctions de base FE. Dans Feel++, le code associé aux espaces d'approximation FE se trouve dans la classe FunctionSpace. Maintenant considérons, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, l'élément RB $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ défini par

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_N^{i}(\boldsymbol{\mu})\ \xi_i^{pr}, \tag{7.4}$$

où ξ_i^{pr}, pour $i = 1, ..., N$, sont les fonctions de base RB. Comme ces fonctions de base RB sont des éléments de $X_{\mathcal{N}}$, nous pouvons également écrire

$$\xi_i^{pr} = \sum_{j=1}^{\mathcal{N}} \alpha_j\ \phi_j,\ 1 \leq i \leq N, \tag{7.5}$$

où les coefficients α_j, pour $j = 1, ..., \mathcal{N}$, sont des réels. Donc nous avons, pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} \sum_{j=1}^{\mathcal{N}} u_N^{i}(\boldsymbol{\mu})\ \alpha_j\ \phi_j. \tag{7.6}$$

L'espace d'approximation RB est une spécialisation de l'espace d'approximation FE, par conséquent la classe ReducedBasisSpace dérive de FunctionSpace.

Notion de Context

Nous introduisons ici la notion de Context, mais commençons par rappeler la notion de maillage et d'élément de référence. Soit Ω un domaine borné régulier, dans $\mathbb{R}^d$ (pour $d = 1, ..., 3$). Définissons $\mathcal{T}_h = \{K_i, i = 1, ..., n_t\}$, avec n_t un entier strictement positif, un ensemble de n_t éléments formant une partition de Ω. Notons $\hat{K}$ l'élément (ou maille) de

référence. Rappelons (voir la section 2.4 du chapitre 1) que pour tout $K \in \mathcal{T}_h$, $\mathcal{F}_{\hat{K},K}$ et $\mathcal{F}^{-1}_{\hat{K},K}$ les applications sont définies par

$$\begin{array}{rccc} \mathcal{F}_{\hat{K},K}: & \hat{K} \in \mathbb{R}^d & \rightarrow & K \in \mathbb{R}^d \\ & \hat{\boldsymbol{t}} & \rightarrow & \boldsymbol{t} \end{array} \quad \text{et} \quad \begin{array}{rccc} \mathcal{F}^{-1}_{\hat{K},K}: & K \in \mathbb{R}^d & \rightarrow & \hat{K} \in \mathbb{R}^d \\ & \boldsymbol{t} & \rightarrow & \hat{\boldsymbol{t}}. \end{array} \tag{7.7}$$

Nous pouvons alors écrire $\mathcal{T}_h = \{K = \mathcal{F}_{\hat{K},K}(\hat{K})\}$.

Dans Feel++, si on considère la discrétisation FE, la classe PolynomialSet possède une sous-classe Precompute qui permet de stocker les évaluations des fonctions de base $\hat{\phi}_{\hat{\imath}}$ sur l'élément de référence $\hat{K}$ en un ensemble de points – typiquement les points de quadrature – noté $\hat{npts}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, où N_{ldofs} est le nombre de degrés de liberté associé à $\hat{K}$. Maintenant considérons l'élément $K \in \mathcal{T}_h$, PolynomialSet possède également une sous-classe Context dédiée à l'image des évaluations des fonctions de base $\hat{\phi}_{\hat{\imath}}$ aux points de l'ensemble $\hat{npts}$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, par la transformation géométrique $\mathcal{F}_{\hat{K},K}(\hat{K})$ en s'appyant sur les pré-calculs effectués sur l'élément de référence $\hat{K}$.

Remarque 14. *Notons que nous parlons ici, pour plus de simplicité, que de l'évaluation des fonctions de base, mais bien d'autres quantités sont stockées dans un* Context *telles que les évaluations du gradient des fonctions de base, ou encore des dérivées partielles.*

Au cours de cette thèse, la notion de Context a été étendue afin de stocker l'évaluation des fonctions de base, non plus en un ensemble de points $\hat{npts}$ défini sur l'élément de référence $\hat{K}$, mais en un ensemble de points, noté $npts$, défini sur l'ensemble du maillage $\mathcal{T}_h$ de Ω. Une sous-classe Context a donc été ajoutée dans FunctionSpace pour calculer ces évaluations. Pour tout point $\boldsymbol{x} \in npts$, nous recherchons l'image $\hat{\boldsymbol{x}}$ de ce point sur l'élément de référence $\hat{K}$ pour pouvoir évaluer les fonctions de base en ce point : $\hat{\phi}_{\hat{\imath}}(\hat{\boldsymbol{x}})$, pour $\hat{\imath} = 1, ..., N_{ldofs}$, puis la transformation géométrique $\mathcal{F}_{\hat{K},K}(\hat{K})$ nous permet d'obtenir l'évaluation des fonctions de base en $\boldsymbol{x}$. Ainsi, l'évaluation de (7.3), en un point donné, est accélérée. Il est important de noter cependant que, pour un point $\boldsymbol{t} \in K \in \mathcal{T}_h$ donné, seules les fonctions de base FE définies sur l'élément K ne sont pas nulles. Nous allons donc nous limiter aux fonctions de base ϕ_i avec $i = \mathcal{G}^{glob}_{\hat{K}\rightarrow K}(\hat{\imath})$, pour $\hat{\imath} = 1, ..., N_{ldofs}$. $\mathcal{G}^{glob}_{\hat{K}\rightarrow K}(\hat{\imath})$ est une application qui repose sur la transformation géométrique $\mathcal{F}^{-1}_{\hat{K},K}$ et qui permet d'obtenir l'indice global du degré de liberté associé à l'indice (local) $\hat{\imath}$. Nous avons alors, pour tout $\boldsymbol{t} \in \Omega$ ayant été préalablement associé à FunctionSpace :: Context,

$$u_{\mathcal{N}}(\boldsymbol{\mu};\boldsymbol{t}) = \sum_i u^i_{\mathcal{N}}(\boldsymbol{\mu};\boldsymbol{t})\ \phi_i(\boldsymbol{x}),\ \text{avec } i = \mathcal{G}^{glob}_{\hat{K}\rightarrow K}(\hat{\imath}),\ 1 \leq \hat{\imath} \leq N_{ldofs}. \tag{7.8}$$

Le code 7.1 illustre comment associer les points d'interpolation au Context de FunctionSpace.

Listing 7.1 – Association des points d'interpolation à un Context .

```
//Le domaine d'etude est le carre unite
auto mesh=unitHypercube<2>();//T_h
auto Xh = Pch<1>(mesh);//espace d'approximation FE ( P_1 )
auto ctx = Xh->context();//context de FunctionSpace
node_type t1(2);//premier point d'interpolation
// ...
node_type tn(2);//n^ème point d'interpolation
```

```
ctx.add(t1);//association du premier point au context
// ...
ctx.add(tn);//association du n^ème point au context
```

Le même principe a été transposé à l'espace d'approximation RB. La classe ReducedBasisSpace contient deux sous-classes , ContextRB et ContextRBSet. La première dérive de PolynomialSet :: Context et contient l'évaluation des fonctions de base RB ξ_i^{pr}, pour $i = 1, ..., N$, en un seul point $\boldsymbol{t} \in \mathcal{T}_h$. Quant à ContextRBSet, elle contient une référence vers FunctionSpace :: Context et permet de faire correspondre, à tout point $\boldsymbol{t} \in \mathcal{T}_h$ préalablement ajouté, l'objet ContextRB associé. ContextRBSet a pour rôle de donner l'accès aux évaluations des fonctions de base RB, et ce pour un ensemble de points associés à ce context. Le code 7.2, écrit à l'intérieur d'un modèle, illustre comment associer des points d'interpolation au ContextRBSet de ReducedBasisSpace.

Listing 7.2 – Association des points d'interpolation à un ContextRBSet .

```
auto mesh=unitHypercube<2>();//T_h
//espace d'approximation RB
auto RbSpace = RbSpacePch(this->shared_from_this());
//contextRBSet de ReducedBasisSpace
auto rb_ctx = RbSpace->context();
node_type xt(2);//premier point d'interpolation
// ...
node_type tn(2);//n^ème point d'interpolation
ctx.add(t1);//association du premier point au context
// ...
ctx.add(tn);//association du n^ème point au context
```

Sur la figure 7.3 sont représentés les différents liens entre les sous-classes que nous venons d'évoquer.

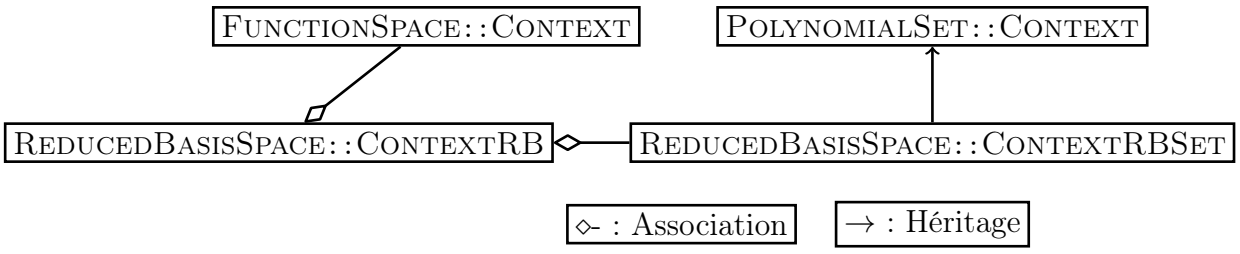


Figure 7.3 – Diagramme UML des sous-classes ContextRB et ContextRBSet du framework-RBM et leurs liens avec Feel++.

Notion d'élément

Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, l'élément $u_{\mathcal{N}}(\boldsymbol{\mu})$ de l'espace d'approximation FE, $X_{\mathcal{N}}$, est représenté par un vecteur contenant les coefficients $u^i_{\mathcal{N}}(\boldsymbol{\mu})$, pour $i = 1, ..., \mathcal{N}$, voir (7.3). La même définition peut être appliquée à un élément de l'espace d'approximation RB. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, l'élément $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ est représenté par un vecteur contenant les coefficients $u^i_N(\boldsymbol{\mu})$, pour $i = 1, ..., N$, voir (7.4). La classe ReducedBasisSpace offre la possibilité d'effectuer des opérations arithmétiques sur les éléments d'un même espace d'approximation RB, comme illustré par le code 7.3.

Listing 7.3 – Opérations arithmétiques sur les éléments d'un même espace d'approximation RB.

```
//Le domaine d'etude est le carre unite
auto mesh=unitHypercube<2>();//\mathcal{T}_h
auto Xh = Pch<1>( mesh );//Espace d'approximation FE ( \mathbb{P}_1 )
//Espace d'approximation RB
auto RbSpace = RbSpacePch( this->shared_from_this()  );

//Construction manuelle des elements de la base reduite
//en projetant les fonctions x et y sur Xh
auto basis_x = vf::project( Xh , elements(mesh), Px() );
auto basis_y = vf::project( Xh , elements(mesh), Py() );
//Ajout des fonctions de base
RbSpace->addPrimalBasisElement( basis_x );
RbSpace->addPrimalBasisElement( basis_y );

int N=RbSpace->size();//dimension de la base reduite

//Declaration de 3 elements de l'espace d'approximation RB
auto ucrb = RbSpace->element();
auto ucrb2 = RbSpace->element();
auto ucrb3 = RbSpace->element();

//Initialisation :
// ucrb = (1, 10), ucrb2=(2, 20) et ucrb3=(2, 25)
ucrb.setCoefficient(0, 1); ucrb.setCoefficient(1, 10);
ucrb2.setCoefficient(0, 2); ucrb2.setCoefficient(1, 20);
ucrb3.setCoefficient(0, 2); ucrb3.setCoefficient(1, 25);

//Operations arithmetiques
ucrb+=ucrb2;// maintenant, ucrb=(3, 30)
ucrb-=ucrb3;// maintenant, ucrb=(1, 5)
```

Penchons-nous maintenant sur l'évaluation d'une expression. Dans FEEL++, la notion d'expression fait référence à la représentation informatique d'une expression mathématique. Pour évaluer une expression qui dépend d'une inconnue FE ou RB, on doit appliquer à cette inconnue un ou plusieurs opérateurs tels que l'identité – *identity (id)* – ou le gradient – *gradient (grad)*–. Les éléments de l'espace d'approximation RB possèdent une implémentation de ces opérateurs. Le code 7.4 illustre l'utilisation de l'opérateur identité sur un élément de $W_{N_{pr}}$. Nous considérons le carré unité comme domaine d'étude Ω. Avec le code 7.3, nous avons construit l'espace $W_{N_{pr}} = \text{span}\{\xi_1^{pr}, \xi_2^{pr}\}$ avec $\xi_1^{pr} = x$ et $\xi_2^{pr} = y$. Après les opérations arithmétiques, l'élément $ucrb \in W_{N_{pr}}$ est la projection de la fonction $x + 5y$ sur $X_{\mathcal{N}}$. Dans 7.4 nous souhaitons évaluer la quantité i_ucrb définie par

$$i\_ucrb = \int_{\Omega} ucrb. \tag{7.9}$$

De manière analytique nous savons que $i\_ucrb = 3$, nous pouvons donc vérifier que le code retourne un résultat exact grâce à l'uitlisation de `FEELPP_ASSERT`.

Listing 7.4 – Utilisation de l'opérateur identité sur un élément de l'espace $W_{N_{pr}}$ pour calculer la quantité définie en (7.9) et la vérifier.

```
//Projection de la valeur de ucrb sur l'espace Xh
//en utilisant l'operateur identite (idv).
auto ucrb_proj = project( _space=Xh ,
```

```
                                _range=elements(mesh),
                                _expr=idv(ucrb) );
//i_ucrb = ∫_Ω ucrb
double i_ucrb = integrate( _range=elements( mesh ),
                            _expr=idv( ucrb_proj )
                          ).evaluate()( 0,0 );
//Calcul de l'erreur
double error_ucrb=math::abs(i_ucrb - 3);
//Verification
FEELPP_ASSERT( error_ucrb < 1e-14 )( error_ucrb )
             .error( "i_ucrb n'a pas la bonne valeur");
```

Notons qu'à partir d'un élément de $W_{N_{pr}}$ (RB), il est également possible d'avoir son équivalent sur l'espace $X_{\mathcal{N}}$ (FE) via la fonction `expansion`. Cette fonction permet d'appliquer la formule (7.4). Comme les fonctions de base RB sont des éléments de l'espace d'approximation FE, nous pouvons obtenir un élément de l'espace $X_{\mathcal{N}}$ à partir d'un élément de l'espace $W_{N_{pr}}$. Le code 7.5 montre comment faire les mêmes calculs que précédemment, mais cette fois-ci avec un élément de $X_{\mathcal{N}}$.

Listing 7.5 – Utilisation de l'opérateur identité sur un élément de l'espace $X_{\mathcal{N}}$ pour calculer la quantité définie en (7.9) et la vérifier.

```
auto ufem = u1.expansion();//ufem ∈ X_N alors que u1 ∈ W_Npr
double i_ufem = integrate( _range=elements( mesh ),
                            _expr=idv( ufem )
                          ).evaluate()( 0,0 );
//Calcul de l'erreur
double error_ufem=math::abs(i_ufem-3);
//Verification
FEELPP_ASSERT( error_ufem < 1e-14 )( error_ufem )
             .error( "i_ufem n'a pas la bonne valeur");
```

1.3 Mise en oeuvre d'une partie *en-ligne* efficace pour la méthode EIM

Cette partie est consacrée à la méthode EIM, plus particulièrement aux ingrédients nécessaires pour avoir une étape *en-ligne* efficace. Dans un premier temps, nous verrons comment utiliser la notion de CONTEXT pour évaluer rapidement un élément d'un espace d'approximation FE ou RB. Puis nous aborderons un aspect lié à la méta-programmation.

Evaluation efficace d'une expression en un ensemble de points

Nous nous concentrons ici sur les expressions qui dépendent de l'inconnue. En effet, considérons l'expression $Expr\Big(\boldsymbol{\mu};\boldsymbol{t};u_{\mathcal{N}}(\boldsymbol{\mu};\boldsymbol{t})\Big)$ qui dépend d'un vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$, d'un point du maillage $\boldsymbol{t} \in \mathcal{T}_h$ et d'un élément de l'espace d'approximation FE, $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$. Lorsque nous voulons appliquer la méthodologie EIM décrite dans le chapitre 3 pour avoir une approximation de l'expression Expr, nous devons alors résoudre – durant l'étape *en-ligne* – le système présenté en (3.14) afin de déterminer les coefficients β. Pour cela il faut évaluer l'expression Expr aux points d'interpolation $\boldsymbol{t}_i$, pour $i = 1, ..., M_t$, où M_t est en entier strictement positif. Cela implique de pouvoir évaluer, rapidement, l'inconnue $u_{\mathcal{N}}(\boldsymbol{\mu})$ en ces points. Pour cela, il a été implémenté au cours de cette thèse une

fonction, `evaluateFromContext`, qui permet d'évaluer une expression – qui peut dépendre de l'inconnue du problème traité – en un certain nombre de points en s'appuyant sur la notion de CONTEXT. Tout d'abord, les points d'interpolation sont ajoutés au CONTEXT de FUNCTIONSPACE (voir le code 7.1). Puis, pour chacun d'eux, nous mettons à jour l'expression en appliquant les opérateurs appropriés sur l'inconnue (identité, gradient, ...), tout en nous servant des pré-évaluations des fonctions de base FE – voir (7.8) –.

Listing 7.6 – Évaluation d'un élément de $W_{N_{pr}}$ via CONTEXT associé FUNCTIONSPACE.

```
//Rappel : Xh est un espace de fonctions FE
//ufem est un element de Xh
auto ctxfem = Xh->context();//Acces a FunctionSpace::context
node_type x1(Dim);//premier point du maillage
// ...
node_type xn(Dim);//n^ème point d'interpolation
ctxfem.add(x1);//association du premier point d'interpolation
// ...
ctxfem.add(xn);//association du n^ème point au context
//Evaluation de ufem
auto evalid = evaluateFromContext(_context=ctxfem,
                                  _expr=idv(ufem) );
//Evaluation du gradient de ufem
auto evalgrad = evaluateFromContext(_context=ctxfem,
                                    _expr=gradv(ufem) );
```

Notons que dans le cas où l'expression que l'on souhaite évaluer dépend de $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$, et non plus de $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$, le procédé reste identique à condition d'utiliser CONTEXTRBSET de REDUCEDBASISSPACE (voir le code 7.2) à la place du CONTEXT de FUNCTIONSPACE.

Listing 7.7 – Évaluation d'un élément de $W_{N_{pr}}$ via CONTEXTRBSET associé à REDUCEDBASISSPACE.

```
//Rappel : RbSpace est un espace de fonctions RB
//ucrb est un element de RbSpace
//Acces a ReducedBasisSpace::contextRBSet
auto ctxrb = RbSpace->context();
node_type x1(Dim);////premier point d'interpolation
// ...
node_type xn(Dim);//n^ème point d'interpolation
ctxrb.add(x1);//association du premier point au context
// ...
ctxrb.add(xn);//association du n^ème point au context
//Evaluation de ucrb
auto evalid = evaluateFromContext(_context=ctxrb,
                                  _expr=idv(ucrb) );
//Evaluation du gradient de ucrb
auto evalgrad = evaluateFromContext(_context=ctxrb,
                                    _expr=gradv(ucrb) );
```

Comme nous avons vu dans la section 1.2 de ce chapitre, pour évaluer un élément d'un espace de fonctions nous utilisons l'opérateur identité. L'implémentation de cet opérateur diffère suivant le type de CONTEXT utilisé. Lorsque nous voulons évaluer un élément de l'espace $X_{\mathcal{N}}$ via la fonction `evaluateFromContext`, et que nous utilisons un CONTEXT de type FUNCTIONSPACE::CONTEXT, alors pour chaque point d'interpolation $\boldsymbol{t}_i$, pour $i = 1, ..., M_t$, la formule (7.8) est utilisée pour implémenter l'opérateur identité.

Maintenant, considérons que nous voulons évaluer, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, l'élément $u_N(\boldsymbol{\mu}, \boldsymbol{t}_i) \in W_{N_{pr}}$ pour $i = 1, ..., M_t$, en utilisant le CONTEXT de type REDUCEDBASISSPACE::CONTEXTRBSET. Chaque fonction de base RB ξ_j^{pr}, pour $j = 1, ..., N$, est pré-évaluée durant la partie *hors-ligne* aux points d'interpolation en utilisant la formule (7.8). Ensuite, pendant la partie *en-ligne*, l'évaluation de cet élément peut s'écrire

$$u_N(\boldsymbol{\mu}, \boldsymbol{t}_i) = \sum_{j=1}^{N} u_N^j(\boldsymbol{\mu})\ \xi_j^{pr}(\boldsymbol{t}_i),\ 1 \leq i \leq M_t. \tag{7.10}$$

Pour $i = 1, ..., M_t$, l'évaluation de $u_N(\boldsymbol{\mu}, \boldsymbol{t}_i)$ telle que proposée dans (7.10) peut s'écrire sous la forme d'un produit matrice-vecteur. En effet, en définissant la matrice $\Xi \in \mathbb{R}^{M_t \times N}$ et le vecteur $u_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ par

$$\Xi = \begin{pmatrix} \xi_1^{pr}(\boldsymbol{t}_1) & \dots & \xi_N^{pr}(\boldsymbol{t}_1) \\ \vdots & & \vdots \\ \xi_1^{pr}(\boldsymbol{t}_{M_t}) & \dots & \xi_N^{pr}(\boldsymbol{t}_{M_t}), \end{pmatrix} \quad \text{et} \quad u_N(\boldsymbol{\mu}) = \begin{pmatrix} u_N^1(\boldsymbol{\mu}) \\ \vdots \\ u_N^N(\boldsymbol{\mu}) \end{pmatrix}, \tag{7.11}$$

alors le vecteur $u_N(\boldsymbol{\mu}, \boldsymbol{t}_1, ..., \boldsymbol{t}_{M_t}) \in \mathbb{R}^{M_t}$, qui contient l'évaluation de $u_N(\boldsymbol{\mu}, \boldsymbol{t}_i)$ pour $i = 1, ..., M_t$, est défini par

$$u_N(\boldsymbol{\mu}, \boldsymbol{t}_1, ..., \boldsymbol{t}_{M_t}) = \Xi\ u_N(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{7.12}$$

La complexité de l'évaluation présentée en (7.12) est en $\mathcal{O}(M_t N)$.

Utiliser un CONTEXTRBSET pour évaluer un élément de $W_{N_{pr}}$ est bien adapté pour un petit nombre de points d'interpolation (de l'ordre de la centaine), mais peut devenir moins adapté lorsqu'un grand nombre de points est considéré. En effet, pour chaque point associé au CONTEXT, l'élément du maillage sur lequel se situe ce point est localisé afin d'obtenir la transformation géométrique, les fonctions de base FE sont évaluées en ce point, ainsi que d'autres quantités telles que le gradient ou la divergence de ces fonctions de base. Toutes ces informations sont stockées dans un CONTEXT associé à POLYNOMIALSET. Ensuite, lorsque nous utilisons un CONTEXTRB, une couche supplémentaire d'information est stockée. Les fonctions de base RB ainsi que leurs gradients sont évalués en chaque point d'interpolation. Par conséquent, il est également possible d'évaluer l'élément $u_N(\boldsymbol{\mu}, \boldsymbol{t}_i) \in W_{N_{pr}}$ pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour $i = 1, ..., M_t^2$, où $M_t^2 >> M_t$ en utilisant la fonction `evaluateFromContext` avec un CONTEXT associé à FUNCTIONSPACE comme illustré par le code 7.8. Cela permet de ne pas stocker la couche supplémentaire d'information associée aux fonctions de base RB.

Listing 7.8 – Évaluation d'un élément de $W_{N_{pr}}$ via un CONTEXT associé à FUNCTIONSPACE.

```
//Rappel : Xh est un espace de fonctions FE
//ucrb est un element de RBSpace
//Acces a FunctionSpace::context
auto ctxfem = Xh->context();
node_type x1(Dim);////premier point
// ...
node_type xn(Dim);//n^ème point
ctxrb.add(x1);//association du premier point
// ...
```

```
ctxrb.add(xn);//association du n^ème point
//Evaluation de ucrb
auto evalid = evaluateFromContext(_context=ctxfem,
                                  _expr=idv(ucrb) );
//Evaluation du gradient de ucrb
auto evalgrad = evaluateFromContext(_context=ctxfem,
                                    _expr=gradv(ucrb) );
```

Dans ce cas, en nous servant de (7.5), l'évaluation de $u_N(\boldsymbol{\mu}, \boldsymbol{t}_i) \in W_{N_{pr}}$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$ et pour $i = 1, ..., M_t^2$, se déroule de la manière suivante :

$$u_N(\boldsymbol{\mu}, \boldsymbol{t}_i) = \sum_{k=1}^{N} \sum_{j} u_N^k(\boldsymbol{\mu})\ \alpha_j\ \phi_j(\boldsymbol{t}_i),\ \text{avec } j = \mathcal{G}_{\hat{K}\to K}^{glob}(\hat{\jmath}),\ 1 \leq \hat{\jmath} \leq N_{ldofs}. \tag{7.13}$$

Jusqu'ici nous avons illustré uniquement l'évaluation d'un élément d'un espace de fonctions (FE ou RB), mais il est également possible d'évaluer le gradient ainsi que les dérivées partielles suivant les variables d'espace comme montré par le code présenté en dans le chapitre 13 (annexe).

Remarque 15. *Comme nous venons de le voir, initialement la fonction* `evaluateFromContext` *a été mise en place pour répondre à un besoin bien précis dans le cadre du framework-RBM. Cependant son application dépasse maintenant largement le cadre de la méthodologie base réduite. G.Dollé l'utilise dans sa thèse [44] pour évaluer une expression en un certain nombre de points du maillage.*

Méta-programmation

Maintenant que nous avons vu comment évaluer rapidement l'inconnue de l'EDP – qu'elle soit un élément de l'espace d'approximation FE ou RB – aux points d'interpolation à l'aide de la fonction `evaluateFromContext`, regardons comment mettre à profit cette fonctionnalité afin de rendre efficace la partie *en-ligne* de la méthode EIM. La classe EIM possède l'expression $Expr\Big(\boldsymbol{\mu}; \boldsymbol{t}; u_{\mathcal{N}}(\boldsymbol{\mu}; \boldsymbol{t})\Big)$ que nous souhaitons approcher. Durant la phase *hors-ligne* nous n'avons accès qu'à la solution $u_{\mathcal{N}} \in X_{\mathcal{N}}$, donc le fait que $Expr$ dépende d'un élément de l'espace d'approximation FE ne pose aucun problème. Cependant, lors de la phase *en-ligne* il faut se servir de la solution base réduite $u_N \in W_{N_{pr}}$ qui est, rappellons-le, une approximation de $u_{\mathcal{N}} \in X_{\mathcal{N}}$. Dans le but de pouvoir évaluer l'expression $Expr$ aussi bien avec $u_{\mathcal{N}} \in X_{\mathcal{N}}$ qu'avec $u_N \in W_{N_{pr}}$, nous avons commencé à travailler sur la mise en place de lambda-expressions. Nous souhaitons que le principe d'une lambda-expression de Feel++ soit identique à celui proposée par le `C++11` pour une lambda-fonction, à savoir la distinction entre les variables liées et les variables libres. Les variables libres sont capturées au moment de l'exécution du code, dans notre cas il s'agit de l'inconnue de l'EDP qui être $u_{\mathcal{N}} \in X_{\mathcal{N}}$ ou $u_N \in W_{N_{pr}}$ suivant que l'on se trouve dans la phase *hors-ligne* ou *en-ligne*. À l'inverse, les variables liées sont les variables "standard" dont le type est connu avant l'exécution du code, telles que le paramètre $\boldsymbol{\mu} \in \mathcal{D}$ et les points d'interpolations $\boldsymbol{t}_i \in \mathcal{T}_h$, pour $i = 1, ..., M_t$. Cependant, les lambda-expressions de Feel++ n'ont pas encore atteint ce niveau de maturité.

Listing 7.9 – Utilisation de lambda-expressions dans Feel++.

```
auto mesh = unitSquare(); // carre unite
```

```
auto Xh = Pch<2>( mesh ); //espace de fonctions FE (P_2)
//u contient la projection de la variable x sur l'espace Xh
auto u = project( _space=Xh, _range=elements(mesh), _expr=Px() );
//Definition de l'integrale I = int_Omega _e1
//Ici le mot cle _e1 est une expression qui a vocation a etre
//substituee par la suite
auto I = integrate( elements(mesh), _expr=_e1 );
// Ix : evaluation de I lorsque _e1 = x^2
double Ix = I( Px()*Px() ).evaluate();
// Iu : evaluation de I lorsque _e1 = u^2
double Iu = I( idv(u)*idv(u) ).evaluate();
//Verification
FEELPP_ASSERT( math::abs(Ix-1./3.) < 1e-14 )( Ix )
             .error( "Ix n'a pas la bonne valeur");
FEELPP_ASSERT( math::abs(Iu-1./3.) < 1e-14 )( Iu )
             .error( "Ix n'a pas la bonne valeur");
//Il est egalement possible de complexifier la lambda-expression
//grace aux mots cles _e2 et _e3.
//Projection sur Xh de la constante 5
u = project(_space=Xh,_range=elements(mesh),_expr=cst(.5) );
//Projection sur Xh de l'expression x+1
auto w = project(_space=Xh,_range=elements(mesh),_expr=Px()+1 );
//Projection sur Xh de l'expression x+y
auto v = project(_space=Xh,_range=elements(mesh),_expr=Px()+Py() );
//Definition de la lambda-expression
auto I3 = integrate( elements(mesh), _expr=_e1*_e2*_e3);
//Evaluation de int_Omega (2u)(grad v . grad v)(w/(2(x+1)))
double eval = I3( cst(2.)*idv(u), gradv(v)*trans(gradv(v)),
                  idv(w)/(cst(2.0)*(Px()+1)) ).evaluate();
```

Dans le code 7.9, qui illustre l'utilisation de lambda-expressions dans le calcul d'intégrales, nous constatons que $\_e1$ est une expression générique qui peut être substituée aussi bien par une expression qui dépend d'un élément de l'espace de fonctions $X_{\mathcal{N}}$ que par une expression moins complexe qui ne fait intervenir que la variable "x". Notons que pour évaluer Iu nous avons appelé l'intégrale I en lui indiquant de substituer $\_e1$ par l'opérateur identité appliqué à un élément de $X_{\mathcal{N}}$. Il est donc nécessaire de connaître les opérateurs qui sont à appliquer au moment où on manipule la lambda-expression. Autrement dit il n'est pas possible, à ce niveau du développement, d'écrire

```
auto I = integrate( elements(mesh), _expr=idv(_e1)*idv(_e1) );
// ... mise a jour de u (element de Xh) ...
auto Iu = I ( u ).evaluate;
```

Alors que lorsque nous manipulons une expression, nous pouvons le faire en la considérant comme une boîte noire. En effet il est possible d'écrire

```
auto u=Xh->element();
u = project( _space=Xh, _range=elements(mesh), _expr=Px() );
//Supposons qu'il existe une expression Expr definie par idv(u)
auto I = integrate( elements(mesh), _expr=Expr );
double Iu = I.evaluate;
//mise a jour de u
u = project( _space=Xh, _range=elements(mesh), _expr=Py() );
Iu = I.evaluate; //Iu est mis a jour
```

Puisque nous sommes obligés, à l'heure actuelle, de connaître les opérateurs impliqués pour manipuler une lambda-expression, il n'est pas encore envisageable de remplacer, dans

la classe EIM, l'expression (dont on veut une approximation) par une lambda-expression.

Remarque 16. *Actuellement la classe EIM n'utilise pas de lambda-expressions. Cela veut dire que si l'expression que nous voulons approcher dépend de la solution de l'EDP, alors l'approximation élément fini $u_{\mathcal{N}} \in X_{\mathcal{N}}$ sera utilisée. Il n'est pas possible d'utiliser l'approximation base réduite $u_N \in W_{N_{pr}}$. Par conséquent durant la phase en-ligne, lorsque nous avons accès à $u_N \in W_{N_{pr}}$, nous faisons appel à la fonction* `expansion` *présente dans le code 7.5 afin d'obtenir un élément de l'espace d'approximation FE : $X_{\mathcal{N}}$. Dans la pratique, la phase en-ligne de la méthode EIM dépend donc de la dimension élément fini $\mathcal{N}$.*

1.4 Gestion de la mémoire

Penchons-nous maintenant sur un problème de mémoire que nous pouvons rencontrer. Afin d'illustrer la problématique de la gestion de la mémoire, nous allons nous intéresser à la résolution du problème elliptique suivant : chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ tel que

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}\Big) = f\Big(v; \boldsymbol{\mu}\Big),\ \forall v \in X_{\mathcal{N}}\ ,\ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{7.14}$$

où $a\big(.,.;\boldsymbol{\mu}\big) : X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$ et $f\big(.;\boldsymbol{\mu}\big) : X_{\mathcal{N}} \times \mathcal{D} \to \mathbb{R}$. Comme nous l'avons vu dans le chapitre 2, afin d'être efficace, la méthode des bases réduites repose sur l'hypothèse suivante : les formes linéaire et bilinéaire a et f admettent une décomposition affine en $\boldsymbol{\mu}$ – voir (2.9) –. Cette décomposition peut être à l'origine d'une partie non négligeable de la place mémoire occupée par le code. Afin de minimiser l'impact sur la mémoire de ces nouveaux termes, il a été développé des classes adaptées que nous présentons ici.

Traitement des formes bilinéaires

Rappelons qu'écrire la décomposition affine de a revient à dire qu'il existe un entier positif Q_a tel que a peut se décomposer de la manière suivante :

$$a\big(u, v; \boldsymbol{\mu}\big) = \sum_{q=1}^{Q_a} \theta_a^q\big(\boldsymbol{\mu}\big)\ a^q\big(u, v\big), \tag{7.15}$$

avec $\theta_a^q : \mathcal{D} \to \mathbb{R}$ pour $1 \leq q \leq Q_a$.

La décomposition affine fait apparaître Q_a formes bilinéaires définies sur $X_{\mathcal{N}} \times X_{\mathcal{N}} \times \mathcal{D}$. De plus, il est important de noter que lorsque a ne peut pas se décomposer suivant (7.15), alors nous approchons cette décomposition affine à l'aide de la méthode EIM, ce qui fait augmenter le nombre de formes bilinéaires utilisées.

Au niveau algébrique, la classe OperatorLinear est dédiée à la manipulation des formes bilinéaires. Chaque forme bilinéaire est représentée sous forme d'une matrice. Lorsque nous considérons un espace d'approximation FE de grande dimension alors l'espace mémoire occupé par le stockage des Q_a matrices peut devenir problématique. Dans Feel++ il est possible d'assembler la matrice associée à une forme bilinéaire à partir de l'expression de cette dernière. Le code 7.10 illustre comment assembler la matrice associée à une forme bilinéaire.

Listing 7.10 – Assemblage de la matrice associée à la forme bilinéaire $a(u,v) = \int_\Omega \nabla u \cdot \nabla v$.

```
//Xh : espace de fonctions FE, Xh
```

```
auto u=Xh->element(), v=Xh->element();
//A : matrice contenant la representation algebrique de a(u,v)
auto A = backend()->newMatrix( Xh, Xh );
//Assemblage de A
form2( _test=Xh, _trial=Xh, _matrix=A ) =
      integrate( _range=elements( mesh ),
                 _expr=gradt( u )*trans( grad( v ) ) )
```

Pour pallier au problème de mémoire dû au stockage des matrices provenant de la décomposition affine, deux classes ont été ajoutées dans FEEL++ : OPERATORLINEARFREE et OPERATORLINEARCOMPOSITE. La première permet de stocker l'expression associée à la forme bilinéaire, par conséquent nous n'avons plus besoin d'assembler une matrice. La seconde offre la possibilité d'effectuer les opérations arithmétiques nécessaires dans le cadre de la méthodologie base réduite, comme par exemple la construction du second membre de (7.15) qui demande de sommer plusieurs matrices ayant au préalable été multipliées par un scalaire. Pour faire cela, OPERATORLINEARCOMPOSITE fait correspondre un objet de type OPERATORLINEAR – les a^q, pour $q = 1, ..., Q_a$ – à l'indice q associé.

Le diagramme représenté sur la figure 7.4 illustre comment les classes OPERATORLINEARFREE et OPERATORLINEARCOMPOSITE s'intègrent dans FEEL++.

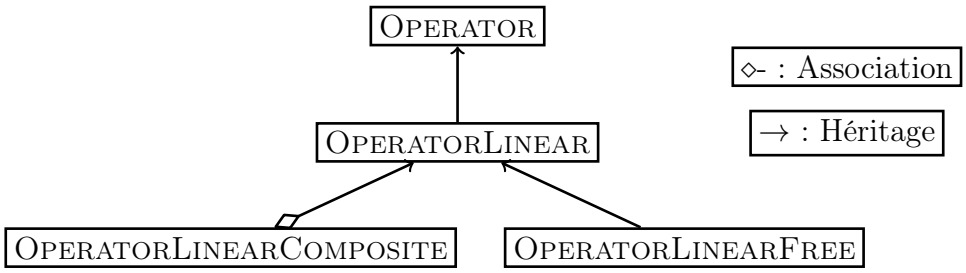


FIGURE 7.4 – Hiérarchie des classes traitant les opérateurs bilinéaires dans FEEL++.

Il est à noter que ces classes peuvent également servir dans le cas où la méthode EIM est utilisée pour retrouver une décomposition affine – voir par exemple (3.27) –. Dans ce cas, OPERATORLINEARCOMPOSITE fait correspondre un objet de type OPERATORLINEAR à une paire d'indices.

Traitement des formes linéaires

Intéressons-nous maintenant au traitement des formes linéaires. La décomposition affine de f fait qu'il existe un entier positif Q_f tel que l'on puisse avoir :

$$f(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu})\ f^q(v), \tag{7.16}$$

avec $\theta_f^q : \mathcal{D} \to \mathbb{R}$ pour $1 \leq q \leq Q_f$.

La décomposition affine fait apparaître Q_a formes bilinéaires définies sur $X_\mathcal{N} \times X_\mathcal{N} \times \mathcal{D}$. De plus il est important de noter que lorsque a ne peut pas se décomposer suivant (7.15), alors nous approchons cette décomposition affine à l'aide de la méthode EIM, ce qui fait augmenter le nombre de formes bilinéaires utilisées.

Au niveau algébrique, la classe FsFunctionalLinear est dédiée à la manipulation des formes linéaires. Chaque forme linéaire est représentée sous forme d'un vecteur. Dans Feel++ il est possible de construire le vecteur associé à une forme linéaire à partir de l'expression de cette dernière. Le code 7.11 illustre comment assembler le vecteur associé à une forme linéaire.

Listing 7.11 – Construction du vecteur associé à la forme linéaire $f(v) = \int_\Omega v$.

```
//Xh : espace de fonctions FE, Xh
auto v=Xh->element();
//F : vecteur contenant la representation algebrique de f(v)
auto F = backend()->newVector( Xh );
//Assemblage de F
form1( _test=Xh, _vector=F ) =
      integrate( _range=elements( mesh ),
                 _expr=id( v ) );
```

De manière similaire au traitement des formes bilinéaires, deux classes ont été ajoutées dans Feel++ : FsFunctionalLinearFree et FsFunctionalLinearComposite. La première permet de stocker l'expression associée à la forme linéaire. La seconde offre la possibilité d'effectuer les opérations arithmétiques nécessaires dans le cadre de la méthodologie base réduite, comme par exemple la construction du second membre de (7.16). Pour faire cela, FsFunctionalLinearComposite fait correspondre un objet de type FsFunctionalLinear – les f^q, pour $q = 1, ..., Q_f$ – à l'indice q associé.

Le diagramme représenté sur la figure 7.5 illustre comment les classes FsFunctionalLinearFree et FsFunctionalLinearComposite s'intègrent dans Feel++.

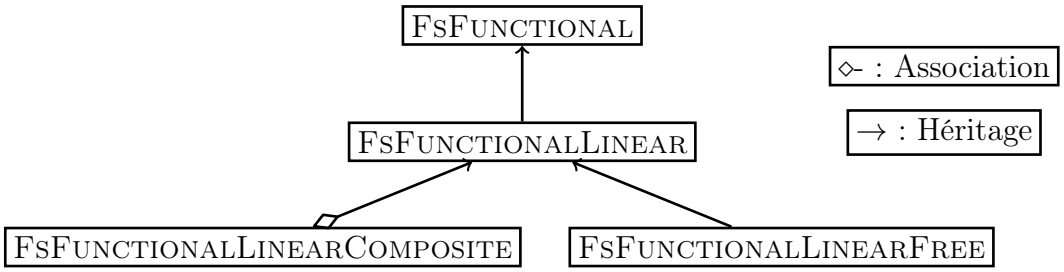


Figure 7.5 – Hiérarchie des classes traitant les opérateurs linéaires dans Feel++.

Là encore, il est à noter que ces classes peuvent également servir dans le cas où la méthode EIM est utilisée pour retrouver une décomposition affine – voir par exemple (3.27) –. Dans ce cas, FsFunctionalLinearComposite fait correspondre un objet de type FsFunctionalLinear à une paire d'indices.

Application au problème des blocs thermiques

Nous commençons par introduire l'utilisation, d'un point de vue utilisateur, des opérateurs "libres" que nous venons de présenter. Considérons le problème des blocs thermiques introduit dans le chapitre 2, section 5. La décomposition affine des formes bilinéaire et linéaire est donnée par (2.185) et (2.186).

Listing 7.12 – Utilisation des opérateurs "libres" pour la décomposition affine de a et f (problème des blocs thermiques).

```
// u et v sont des elements de Xh, l'espace de fonctions FE
```

```
// Xh est de type space_type
// definition des types utilises
// Creation d'un vecteur d'operateurs
std::vector<operator_ptrtype> Aq_free; //bilineaires
std::vector<functional_ptrtype> Fq_free; //lineaires
// Nous supposons que ces vecteurs sont dimensionne correctement
// Expression associee a la premiere forme bilineaire
auto expr_a0 = integrate(_range=markedelements(mesh,"domain-1"),
                         _expr=gradt( u )*trans( grad( v ) ) );
//Instanciation d'un objet de type OperatorLinearFree
auto operatorfree0=opLinearFree( _domainSpace=Xh,
                                 _imageSpace=Xh,
                                 _expr=expr_a0 );
Aq_free[0]=operatorfree0; // Remplissage de Aq_free
// ... procede identitque pour les autres formes bilineaires ...
//Instanciation d'un objet type operateur composite
auto CompositeA=opLinearComposite(_domainSpace=Xh,_imageSpace=Xh);
CompositeA->addList( Aq_free ); //Ajout des operateurs "libres"
// Expression associee a la forme lineaire
auto expr_f =
integrate(_range=markedfaces(mesh,"south_domain-1"),_expr=id(v) )
+integrate(_range=markedfaces(mesh,"south_domain-2"),_expr=id(v) )
+integrate(_range=markedfaces(mesh,"south_domain-3"),_expr=id(v) );
//Instanciation d'un objet de type FsFunctionalLinearFree
auto functionalfree = functionalLinearFree(_space=Xh,_expr=expr_f);
Fq_free[0]=functionalfree; // Remplissage de Fq_free
//Instanciation et remplissage de la fonctionnelle composite
auto CompositeF=functionalLinearComposite( _space=Xh );
CompositeF->addList( Fq_free );
```

Ensuite, les objets `CompositeA` et `CompositeF` sont récupérés dans le framework-RBM, puis il est possible d'assembler les seconds membres de (7.15) et (7.16) de la manière suivante

Listing 7.13 – Manipulation de `CompositeA` et `CompositeF` dans le framework-RBM.

```
std::vector<double> coeff_A;
std::vector<double> coeff_F;
//Nous supposons que pour un μ donne les coefficients θ_a^q(μ)
//pour q = 1,...,Q_a sont contenus dans le vecteur "coeff_A",
//les coefficients θ_f^q(μ) sont dans coeff_F.
compositeA->setScalars( coeff_A );
auto A = backend()->newMatrix( Xh, Xh );
//A va contenir le second membre de (7.15)
CompositeA->sumAllMatrices( A );
CompositeF->setScalars( coeff_F );
auto F = backend()->newVector( Xh );
//F va contenir le second membre de (7.16)
CompositeF->sumAllVectors( F );
```

Revenons à présent sur le problème des blocs thermiques afin d'étudier l'impact de l'utilisation de ces opérateurs "libres" sur la mémoire consommée par le code. Pour cela nous faisons appel à une fonctionnalité de `Petsc` qui permet de mesurer la mémoire résidente du code : `PetscMemoryGetCurrentUsage`. Nous considérons ici une discrétisation FE $\mathbb{P}_1$ avec 291 700 dofs degrés de liberté sur un seul processeur. La décomposition affine de a fait intervenir 10 termes. Lorsque nous stockons uniquement les expressions associées aux formes bilinéaires, nous gagnons alors 10.25% de mémoire par rapport au même code ayant stocké les 10 matrices. En revanche, comme nous devons assembler les matrices

associées aux formes bilinéaires à chaque fois que nous avons besoin de les manipuler, nous perdons en performance lors de l'éxécution. Un facteur 5 a été mesuré entre les deux versions du code. Si nous avons suffisamment de mémoire, le stockage des matrices est donc à préférer pour bénéficier au maximum des performances du code.

2 Interface utilisateur

Nous présentons ici comment l'utilisateur peut créer un modèle, sur lequel il souhaite appliquer la méthodologie des bases réduites, avec le framework-RBM. Ensuite nous illustrons comment l'utilisateur peut se servir de la méthode EIM pour fournir les différents termes de la décomposition affine.

2.1 Création de modèles

On se propose ici de détailler l'interface que doit respecter un modèle pour bénéficier du framework-RBM. Nous en profitons également pour montrer que lorsque c'est possible, le framework-RBM déduit certaines données afin d'alléger cette interface utilisateur.

Afin d'illustrer nos propos, nous présentons comment mettre en place le modèle des blocs thermiques introduit dans le chapitre 2, section 5.

Tout modèle possède une fonction d'initialisation `initModel`. Commençons par définir les espaces d'approximation FE et RB. Tout d'abord il faut définir une géométrie qu'un maillage associé. Pour cela on se base sur l'outil `GMSH`. Nous pouvons fournir le maillage de notre géométrie en passant par la fonction `loadGMSHMesh`. Dans un deuxième temps définir l'espace de fonctions élément fini $X_{\mathcal{N}}$. Enfin la fonction `setFunctionSpaces` (implémentée dans ModelCrbBase) permet de lier l'espace $X_{\mathcal{N}}$ à la classe mère ModelCrbBase et en même temps d'instancier un objet de type ReducedBasisSpace.

```
//le maillage que l'on souhaite charcher se trouve dans
//le fichier mon_maillage.msh
auto mesh = loadGMSHMesh( _mesh=new mesh_type,
                          _filename="mon_maillage.msh");
//Creation de l'espace de fonctions FE
auto Xh = functionspace_type::New( mesh ); // X_N
this->setFunctionSpaces( Xh );
```

Cependant il existe une autre façon de faire. À partir de la description de la géométrie, il est également possible de définir l'espace de fonctions FE en écrivant une seule ligne.

```
//Creation du maillage (de dimension 2) a partir de
//la description de la geometrie
//Creation de l'espace de fonctions X_N et instanciation
//d'un objet de type ReducedBasisSpace
this->setFunctionSpaces( Pch<1> (
                         loadMesh( _mesh=new Mesh<Simplex<2> > )
                          ) );
```

La fonction `loadMesh` va créer le maillage associé à la géométrie (s'il n'existe pas déjà) puis le charger. Cela suppose que le nom du fichier contenant la description de la géométrie soit indiqué dans le fichier `CMakeLists.txt`.

Maintenant que l'espace d'approximation FE est défini, nous pouvons déclarer les éléments $u \in X_{\mathcal{N}}$ et $v \in X_{\mathcal{N}}$ qui seront utiles pour définir les formes bilinéaires et linéaires.

```
auto u = Xh->element(); auto v = Xh->element();
```

Il nous faut également définir les extremums de l'espace des paramètres. Pour le modèle des blocs thermiques nous avons $\mathcal{D} \subset \mathbb{R}^P$ avec $P = 8$.

```
auto mu_min = Dmu->element(); auto mu_max = Dmu->element();
mu_min <<  0.1 , 0.1 , 0.1 , 0.1 , 0.1 , 0.1 , 0.1 , 0.1 ;
mu_max <<  10 , 10 , 10 , 10 , 10 , 10 , 10 , 10 ;
Dmu->setMin( mu_min ); Dmu->setMax( mu_max );
```

L'objet `Dmu` que nous manipulons représente l'espace des paramètres $\mathcal{D}$ et est défini dans la classe ModelCrbBase comme étant un pointeur sur un objet de type `parameterspace_type` provenant de la classe ParameterSpace.

Pour utiliser le framework-RBM, l'utilisateur doit fournir le produit scalaire à utiliser pour les différentes projections qui interviennent dans la méthodologie base réduite. Ici nous utilisons le produit scalaire associé à la norme d'énergie pour un vecteur de paramètres de référence $\boldsymbol{\mu}_{ref} \in \mathcal{D}$. La fonction `addEnergyMatrix` est utilisée pour que le framework-RBM puisse avoir accès à un pointeur sur la matrice M.

```
double muref=0.1; // μ_ref
//M=Matrice associee au produit scalaire defini par l'utilisateur
auto M = backend()->newMatrix( Xh , Xh );
form2( Xh, Xh, M ) =
  integrate( _range=markedelements( mesh, "domain-1" ),
             _expr=gradt( u )*trans( grad( v ) )  );
form2( Xh, Xh, M ) +=
  integrate( _range=markedelements( mesh, "domain-2" ),
             _expr=gradt( u )*trans( grad( v ) ) *muref );
// ...
//Une fois que la matrice M a fini d'etre assemblee :
this->addEnergyMatrix( M );
```

À présent il nous faut définir les termes qui composent la décomposition affine en paramètres de notre problème. Pour cela nous pouvons utiliser deux approches : fournir séparément les représentations algébriques des formes (bi-)linéaires et les évaluations des fonctions dépendant de $\boldsymbol{\mu} \in \mathcal{D}$; ou bien associer ces deux informations. Considérons le premier cas. Pour fournir les représentations algébriques des formes (bi-)linéaires il nous faut implémenter `computeAffineDecomposition` qui renvoie un vecteur de matrices et un vecteur de vecteur de vecteur. En effet nous avons fait le choix d'associer les formes linéaires des sorties du modèle avec celles provenant de la décomposition affine du second membre de l'EDP.

```
affine_decomposition_light_type computeAffineDecompositionLight()
{
  return boost::make_tuple( Aq, Fq );
}
```

Ici `Aq` et `Fq` sont des vecteurs, respectivement, de matrices et de vecteurs de vecteurs. `Aq[q]` contient la matrice associée à la $q^{\text{ème}}$ forme bilinéaire de la décomposition affine de a. `Fq[out][q]` contient le vecteur associé à la forme $q^{\text{ème}}$ forme linéaire de la décomposition affine de la sortie indicée par out. La sortie indicée par $out = 0$ est toujours la sortie "souple" du modèle, par conséquent `Fq[0]` contient l'ensemble des termes de la décomposition affine de f. `Aq` et `Fq` sont alors définis à l'aide des mots clés `form2` et `form1` – voir les codes 7.10 et 7.11 –.

Notons que nous n'avons pas implémenté la fonction `computeAffineDecomposition()`

mais `computeAffineDecompositionLight()`. Cela vient du fait que le framework-RBM traite tous les modèles de la même manière. Il considère que tous les modèles ont besoin de faire appel à la méthode EIM, et par conséquent il s'attend à récupérer non pas un vecteur de matrices, mais un vecteur de vecteur de matrices. L'utilisation de la version "allégée" de `computeAffineDecomposition()` permet à l'utilisateur de ne pas déclarer des termes inutiles. C'est le framework-RBM qui va transformer automatiquement les vecteurs en vecteurs de vecteurs afin de pouvoir appliquer la méthodologie base réduite.

Ensuite, les évaluations des fonctions dépendant de $\boldsymbol{\mu} \in \mathcal{D}$ sont données par la fonction `computeBetaQ` qui retourne des objets de type `beta_vector_light_type`, des vecteurs de coefficients. Il s'agit de la version "allégée" de `beta_vector_type`.

```
boost::tuple<
    beta_vector_light_type, std::vector<beta_vector_light_type>
>
computeBetaQ( parameter_type const& mu )
{
  //Nous supposons que betaAq et betaFq sont deja dimensionnes
  //Coefficient associe a la premiere forme bilineaire
  betaAq[0] = 1;
  for ( int i=1; i<9; i++ )
  {
    //Coefficient associe a la (i+1)ème forme bilineaire
    betaAq[i] = mu( i-1 );
  }
  //Coefficient associe a la 10ème forme bilineaire
  betaAq[9]=1;

  //Coefficient associe a la premiere forme lineaire
  betaFq[0][0] = 1; // sortie "souple"
  return boost::make_tuple( betaAq, betaFq );
}
```

Notons que dans le cas où la méthode EIM a besoin d'être utilisée alors ce sera la fonction `computeBetaQm` qui sera implémentée.

Nous venons de voir comment fournir, de manière séparée, les représentations algébriques des formes (bi-)linéaires et les évaluations des fonctions dépendant de $\boldsymbol{\mu} \in \mathcal{D}$. Il a été fait un effort pour simplifier cette interface et fournir des fonctions qui permettent de donner ces informations de manière plus corrélée. La fonction `addLhs` fait correspondre une forme bilinéaire (définie par son expression) et une fonction, fournie sous la forme d'une chaîne de caractères, qui peut dépendre de $\boldsymbol{\mu}$. Afin d'évaluer, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, la fonction qui dépend des paramètres nous utilisons la librairie `GiNaC` [18]. Le symbole `mui`, pour $i = 0, ..., P-1$ (rappelons que P est la dimension de l'espace des paramètres), représente la $i^{\text{ème}}$ composante du vecteur de paramètres $\boldsymbol{\mu}$. De la même manière la fonction `addRhs` fait correspondre une forme linéaire et une fonction qui peut dépendre de $\boldsymbol{\mu}$. Ces fonctions sont implémentées dans la classe ModelCrbBase.

```
auto a0 = form2( _trial=Xh, _test=Xh);
a0 = integrate( _range=markedelements( mesh, "domain-1" ),
                _expr=gradt( u )*trans( grad( v ) ) );
this->addLhs( { a0 , "1" } );
auto a1 = form2( _trial=Xh, _test=Xh);
a1 = integrate( _range=markedelements( mesh, "domain-2" ),
                _expr=gradt( u )*trans( grad( v ) ) );
this->addLhs( { a1 , "mu0" } );
// ...
```

```
  auto f0 = form1( _test=Xh );
  f0 =  integrate( _range=markedfaces( mesh,"south_domain-1" ),
                   _expr= id( v ) )
  + integrate( _range=markedfaces( mesh,"south_domain-2" ),
                  _expr= id( v ) )
  + integrate( _range=markedfaces( mesh,"south_domain-3" ),
                  _expr= id( v ) );
  this->addRhs( { f0, "1" } );
```

Pour les modèles qui ont une autre sortie en plus de la sortie "souple" (par défaut), la fonction `addOutput` peut être utilisée et fonctionne sur le même principe que `addRhs`. L'initialisation du modèle est maintenant terminée et le framework-RBM a toutes les informations nécessaires pour appliquer la méthodologie base réduite. Cependant l'utilisateur doit fournir encore une ultime information : le calcul de la sortie du modèle. En effet nous demandons à l'utilisateur de fournir le calcul de la sortie du modèle, sans se baser sur la décomposition affine, afin de nous assurer de la bonne qualité de l'approximation base réduite calculée par le framework-RBM

```
double output( int output_index, parameter_type const& mu ,
               element_type& u)
{
  if ( output_index==0 )
  {
    output =
      integrate( _range=markedfaces( mesh,"south_domain-1" ),
                 _expr= idv( u ) ).evaluate()(0,0)
    + integrate( _range=markedfaces( mesh,"south_domain-2" ),
                 _expr= idv( u ) ).evaluate()(0,0)
    + integrate( _range=markedfaces( mesh,"south_domain-3" ),
                 _expr= idv( u ) ).evaluate()(0,0);
  }
  else
    throw std::logic_error( Error with output_index : only 0 " );
  return output;
}
```

Dans le chapitre 12 (annexe), il est présenté de manière détaillée comment peut être mis en place le modèle correspondant au problème du bouclier thermique introduit dans le chapitre 2, section 5.

2.2 Utilisation de la méthode EIM

Nous présentons ici un cas pratique d'utilisation de la classe EIM inspiré par le problème électro-thermique non-linéaire décrit dans le chapitre 10. La formulation variationnelle associée à ce problème peut s'exprimer de la manière suivante : chercher $\boldsymbol{u} = (u, T) \in \boldsymbol{X}_{\mathcal{N}} = X_h^u \times X_h^T$ tel que pour tout $\boldsymbol{v} \in \boldsymbol{X}_{\mathcal{N}}$ et pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$g(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}) = 0. \tag{7.17}$$

où u et T sont respectivement le potentiel électrique et la température, $\boldsymbol{X}_{\mathcal{N}}$ est un espace composite – produit de l'espace de fonctions X_h^u et X_h^T auxquels appartiennent u et T –. Dans ce modèle, le résidu g dépend du terme, non-affine en $\boldsymbol{\mu}$, $\sigma(T)$. Ce dernier est défini par

$$\sigma(T) = \frac{\boldsymbol{\mu}^0}{1 + \boldsymbol{\mu}^1 (T - T_0)}\ , \tag{7.18}$$

où T_0 est une constante donnée. Par conséquent, g étant non-affine en $\boldsymbol{\mu}$, la décomposition affine associée ne peut pas être obtenue sans avoir une approximation du terme $\sigma(T)$. Nous utilisons la méthode EIM pour obtenir cette approximation. Concrètement, pour cela, nous faisons appel à la classe EIM du framework-RBM en instanciant un objet `eim` comme illustré par le code 7.14. Cet objet `eim` donne directement accès aux fonctions de bases et coefficients de l'expansion EIM décrite en (3.5).

Listing 7.14 – Approximation EIM d'un terme non-affine en paralètres et dépendent de la solution de l'EDP

```
//Nous supposons que l'espace des parametres D
//denote Dmu dans le code, est deja rempli
parameter_type mu; // μ ∈ D
//Declaration d'un echantillonnage de D
auto Pset = Dmu->sampling();
//Taille de l'echantillonnage Pset
int eim_sampling_size = 1 000;
//Generation de Pset
Pset->randomize(eim_sampling_size);
//Expression non-lineaire de σ(T)
auto sigma = ref( mu(0) )/( 1+ref( mu(1) )*( idv(T)-T0 ) );

//Instanciation d'un objet eim
auto eim_sigma = eim( _model=solve( g(u,v;μ)=0 ),
                      _element=T,//inconnue dont nous avons besoin
                                 //pour l'evaluation de σ(T)
                      _parameter=mu,// μ
                      _expr=sigma,  // σ(T)
                      _space=X_h^T,//Espace de fonctions, voir(7.17)
                      _name="eim_sigma",
                      _sampling=Pset );

//Maintenant nous avons acces aux coefficient de l'expansion EIM
std::vector<double> beta_sigma = eim_sigma->beta( mu );
```

Dans le code 7.14, lors de la construction de l'objet `eim` nous utilisons l'abus de notation `_model = solve`$(g(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}) = 0)$ afin d'indiquer qu'il est nécessaire de résoudre le problème (7.17). Dans la pratique il faut fournir en argument un fonctor qui possède un opérateur – qui peut être appelé par la partie *hors-ligne* de la méthode EIM – afin de récupérer la solution élément fini du problème pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné. Puisque les fonctions de base de l'expansion EIM appartiennent à l'espace de fonctions donné par l'argument `_space`, nous avons donné l'espace X_h^T.

Notons qu'il est également possible de construire l'expansion EIM d'une fonction qui ne dépend pas de la solution du modèle. Considérons la fonction $\zeta(\boldsymbol{\mu}; \boldsymbol{x}) = \sin(\boldsymbol{\mu}^0\ \pi\ x)$, où μ^0 est un paramètre et $\boldsymbol{x}$ est un point situé sur le domaine sur lequel est défini le problème. Dans ce cas nous n'avons pas besoin de calculer de solution élément fini pour pouvoir approcher $\zeta(\boldsymbol{\mu}; x)$ par la méthode EIM. Nous utilisons donc le mot clé `eim_no_solve`, comme illustré par le code 7.15.

Listing 7.15 – Approximation EIM d'un terme non-affine en paramètres mais sans dépendance sur la solution de l'EDP

```
//Nous reprenons les memes notations que dans le code 7.14
//Expansion EIM de sin(μ^0 π x)
auto eim_sin = eim( _model=eim_no_solve(FemModel) ),//le model
```

```
                        //FemModel est donne en argument de eim_no_solve
                        _element=T,
                        _parameter=mu,// μ
                        _expr=sin( ref( mu(0) )*pi*Px() ),
                        _space=X_h^{FemModel},//Objet representant l'espace
                          //de fonctions utilise par le modele FemModel
                        _name="eim_sin",
                        _sampling=Pset );
```

Résumé

Dans la première partie de ce chapitre, le framework-RBM a été présenté, ce qui a permis de voir à quel point il est intégré dans Feel++. *Nous avons mis l'accent sur la vision fonctionnelle qu'offre désormais le framework-RBM grâce à la classe* ReducedBasisSpace, *ainsi que sur l'introduction des opérateurs "libres" et la fonction* `evaluateFromContext` *– utilisée dans* Feel++ *au-delà du cadre du framework-RBM –. Dans la deuxième partie nous avons présenté l'interface utilisateur qui a été développée pour que l'utilisateur saisisse le moins d'informations et le plus simplement possible. En lien avec cet objectif nous avons vu notamment une version plus intuitive de l'interface pour donner les termes de la décomposition affine. L'utilisation de la classe EIM a également été illustrée.*

Chapitre 8

Calcul haute performance

Avant les années 2000, la fréquence des processeurs augentait régulièrement. Ainsi, lorsque nous achetions un nouvel ordinateur, notre code gagnait automatiquement en performance, sans qu'aucune intervention de notre part n'intervienne. Cependant, ce procédé a atteint ses limites, notamment en raison des difficultés liées à la dissipation thermique. Suite à cette limite, les processeurs sont devenus multicoeurs, traitant ainsi plusieurs instructions simultanément. Sur une architecture à mémoire distribuée, la norme MPI – *Message Passing Interface* – est majoritairement utilisée pour assurer la communication entre les processeurs. Au contraire, sur une architecture à mémoire partagée, la technologie Multithreading – *MT* – sera préférée. Nous sommes alors entrés dans l'ère du calcul haute performance – *High Performance Computing (HPC)* –. Dans le but de profiter pleinement des ressources de calculs d'une machine, de plus en plus de personnes font de la programmation hybride en couplant ces deux approches. À cela peuvent s'ajouter l'utilisation des cartes graphiques – *Graphics Processing Unit (GPU)* – et plus récémment l'architecture MIC – *Many Integrated Core* –. Aujourd'hui, un code de calcul se doit d'avoir une stratégie HPC pour s'attaquer à la résolution de problèmes industriels.

Le framework-RBM prend en charge les architectures parallèles, en s'appuyant notamment sur les structures de données parallèles de Feel++. Dans ce chapitre, nous commencerons par donner les grandes lignes de la stratégie HPC adoptée par Feel++, puis nous présenterons les stratégies adoptées pour la parallélisation des méthodes CRBM et EIM. Nous verrons comment adapter la fonction `evaluateFromContext` pour qu'elle puisse être utilisée lorsque les points d'interpolation se trouvent répartis sur différents processeurs. Enfin, nous illustrerons la scalabilité du framework-RBM.

1 Stratégie HPC de Feel++

Nous présentons brièvement ici les grandes lignes de la stratégie HPC adoptée par Feel++. Une description détaillée se trouve dans la thèse de V.Chabannes [32].

Feel++ a pour objectif d'utiliser aussi bien les couches MPI que GPU comme illustré par la figure 8.1. Le module HARTS – *Hybrid Architecture Runtine System* – de Feel++ assure la communication entre les couches CPU et GPU, ainsi que la réalisation de différentes tâches sur GPU. Cependant, nous nous concentrons ici uniquement sur la couche MPI.

Soit n_p, un entier strictement positif, représentant le nombre de processeurs utilisés. Feel++ commence par créer n_p partitions du maillage associé au problème, à l'aide des

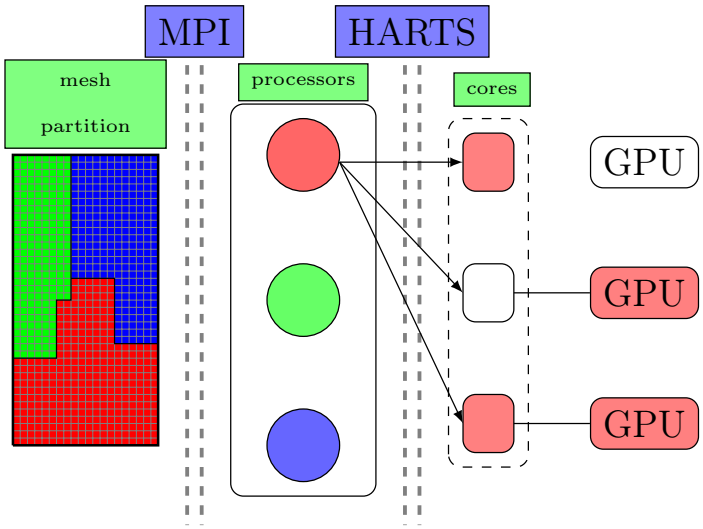


FIGURE 8.1 – Stratégie de calcul haute performance dans FEEL++.

partitionneurs fournis par `GMSH` que sont `Chaco` [62] et `Metis` [73]. Puis chaque partition est envoyée sur un processeur. Autrement dit, un processeur ne peut accéder qu'à une sous-partie du maillage. L'espace de fonctions FE, $X_{\mathcal{N}}$, étant défini sur l'ensemble du maillage, pour chaque vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$, l'élément $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}}$ est alors réparti sur l'ensemble des processeurs. Aucun processeur ne peut avoir accès à la représentation complète de $u_{\mathcal{N}}(\boldsymbol{\mu})$.

Il est donc nécessaire d'utiliser des structures algébriques adaptées, et de faire appel à des solveurs parallèles, pour résoudre une EDP. Pour cela, il a été développé une interface qui permet de profiter des outils adaptés pour le parallélisme fournis par `PETSc`. Il est donc possible de construire des matrices et des vecteurs ayant une structure permettant d'être répartis sur l'ensemble des processeurs, et d'utiliser ensuite des solveurs – linéaire et non-linéaires – et préconditionneurs parallèles.

Il est important de noter que le parallélisme de FEEL++, comme celui du framework-RBM , est transparent pour l'utilisateur. Les communications entre processeurs sont cachées dans la librairie.

2 La méthode CRBM

Nous présentons ici la stratégie de parallélisation des partie *hors-ligne* et *en-ligne*. Le sujet de l'enrichissement des espaces d'approximation RB, ainsi que la visualisation de la solution réduite sur le maillage, seront également évoqués.

2.1 Partie *hors-ligne*

Lorsque nous souhaitons faire tourner le code sur plusieurs processeurs, deux stratégies s'offrent à nous pour adapter la partie *hors-ligne* de l'algorithme des bases réduties.

La première consiste à ne pas partitionner le maillage – et donc ne pas utiliser de structures de données parallèles –. Le processeur p (avec $0 \leq p \leq n_p$) possède un vecteur de paramètres $\boldsymbol{\mu}_p \in \mathcal{D}$, partagé avec aucun des autres processeurs, et résout l'EDP de manière séquentielle. Ainsi, à chaque itération, n_p solutions FE sont calculées. Cependant,

l'inconvénient de cette approche réside dans le fait qu'il ne sera pas possible de résoudre de plus gros problèmes en utilisant plus de processeurs, puisque chaque résolution se fait de manière séquentielle. Or, résoudre un problème possédant un grand nombre de mailles entraîne une grande consommation de mémoire.

La seconde stratégie, que nous avons choisi de mettre en oeuvre pour le framework-RBM , est calquée sur celle de FEEL++. Le maillage est donc partitionné puis distribué sur l'ensemble des processeurs par FEEL++. Pour chaque vecteur de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ sélectionné, l'EDP est résolue à l'aide des structures de données et solveurs parallèles. Ainsi, à chaque itération, une seule solution FE est calculée, comme c'est le cas dans l'algorithme séquentiel présenté dans le chapitre 2. L'avantage de cette stratégie est de répartir la consommation mémoire, demandée par la résolution d'un gros problème, sur l'ensemble des processeurs.

Il faut cependant veiller à ce que tous les processeurs possèdent le même vecteur de paramètres $\boldsymbol{\mu}$, sinon la solution calculée n'a aucun sens. Pour cela, le framework-RBM a adapté l'algorithme gloutons – ou POD/glouton – utilisé pour construire l'ensemble S_N (voir chapitre 2). Rappelons que pour un échantillon de paramètres Ξ_{train} donné, ce processus fait appel à la partie *en-ligne* pour ensuite sélectionner le jeu de paramètres $\boldsymbol{\mu} \in \Xi_{train}$ maximisant l'estimation d'erreur associée. Afin de paralléliser ce processus, l'échantillon Ξ_{train} est généré par un seul processeur (cela permet d'éviter la présence de doublons), qui ensuite le découpe de manière équitable en n_p sous-échantillons, puis ces sous-échantillons sont envoyés aux autres processeurs en utilisant une communication point à point. De cette manière, chaque processeur peut sélectionner – localement – le meilleur candidat, puis envoie ce candidat ainsi que l'estimation d'erreur lui étant associé au processeur maître. Là encore des communications point à point. Enfin, maintenant que le processeur maître a récupéré les meilleurs candidats des autres processeurs, il peut sélectionner le meilleur candidat – global – en prenant le jeu de paramètres associé à la plus grande estimation d'erreur. Une fois cette sélection achevée, le processeur maître distribue le vecteur de paramètres sélectionné aux autres processeurs via une communication collective. Ainsi, tous les processeurs possèdent le même jeu de paramètres $\boldsymbol{\mu} \in \Xi_{train}$.

L'algorithme 7 présente la version parallèle de l'algorithme glouton dans le cas de problèmes elliptiques. Nous faisons l'hypothèse ici que le processeur maître est le processeur 1 et que la numérotation des processeurs commence à 1. Les commentaires sont précédés de deux barres obliques comme en `C++`. Les phases de communication entre processeurs apparaissent entre chevrons.

Notons que dans le cas où l'algorithme glouton n'est pas utilisé alors le vecteur de paramètres $\boldsymbol{\mu} \in \Xi_{train}$ est sélectionné par un seul processeur, qui ensuite le distribue aux autres à l'aide d'une communication collective.

2.2 Partie *en-ligne*

Lorsque la partie *en-ligne* est utilisée pour l'agorithme glouton – ou POD/glouton –, chaque processeur possède son propre sous-échantillon de Ξ_{train} qu'il ne partage avec aucun autre processeur.

Dans les autres cas d'utilisation, pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, tous les processeurs résolvent le même système réduit et possèdent tous la même solution réduite et la même estimation d'erreur associée. Cela vient du fait que tous les processeurs possèdent une version de chaque base de données évoquée dans la section 1.1 du chapitre 7. Celle associée à la classe

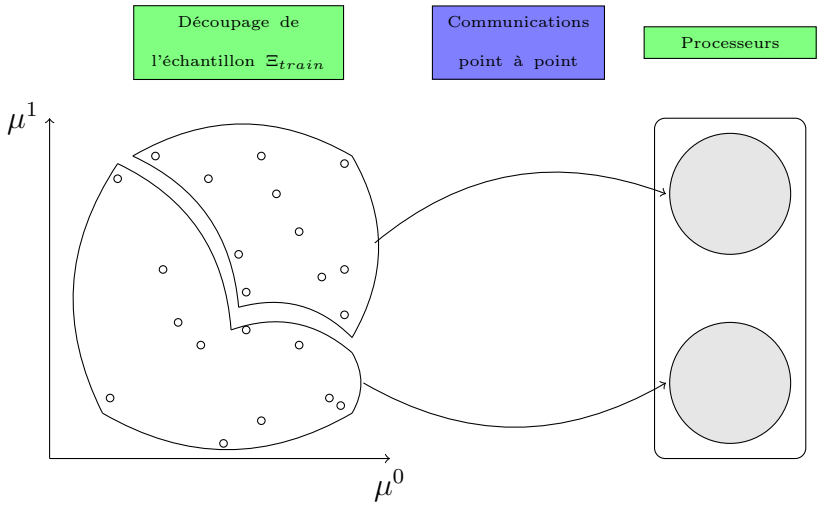


FIGURE 8.2 – Lors de la phase de sélection du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$, le processeur maître découpe l'échantillon Ξ_{train} puis envoie un sous-échantillon à chaque processeur. De cette manière le framework-RBM utilise l'ensemble des processeurs à sa disposition. Ici, l'espace des paramètres est de dimension deux et nous avons $n_p = 2$.

CRB contient les produits scalaires résultants de la projection des matrices et des vecteurs sur les espaces réduits (primal et dual), ainsi que les résultats des produits scalaires utilisés pour l'estimation d'erreur a posteriori. Ces quantités sont indépendantes des différentes partitions du maillage. Par conséquent, tous les processeurs possèdent la même version de la base de données associée à la classe CRB, et ont donc accès aux mêmes informations. Dans le cas où nous ne souhaitons pas visualiser la solution réduite sur le maillage, il n'est pas nécessaire de charger la base de données – associée à la classe REDUCEDBASISSPACE – contenant les fonctions de base RB. Autrement dit, si la construction de la base réduite a nécessité l'utilisation de n_p processeurs, il est possible de lancer la partie *en-ligne* sur un seul processeur, sans avoir besoin de charger le maillage.

2.3 Fonctions de base RB

Enrichissement

Les espaces d'approximation RB sont enrichis de manière incrémentale. A chaque itération, une solution FE est calculée puis orthonormalisée par rapport aux fonctions de base RB déjà présentes. Le framework-RBM offre donc la possibilité, une fois la partie *hors-ligne* achevée, d'ajouter de nouvelles fonctions de base RB afin d'améliorer la précision de la solution réduite. Pour cela, il est nécessaire de charger la base de données associée à REDUCEDBASISSPACE, afin que chaque processeur puisse avoir accès à sa sous-partie des fonctions de base RB – car, rappelons-le, les éléments de l'espace de fonctions FE sont répartis sur l'ensemble des processeurs –. Par conséquent, il est impératif d'avoir exactement le même partitionnement du maillage pour pouvoir charger les fonctions de base RB, et ainsi relancer l'étape *hors-ligne* afin d'enrichir les espaces d'approximation.

Algorithm 7 Algorithme glouton – version parallèle –
$[S_N, W_{N_{pr}}, W_{N_{du}}] = glouton(N_{max}, \epsilon_{tol})$

Processeur 1 construit l'échantillon Ξ_{train} ;
Processeur 1 découpe équitablement Ξ_{train} en n_p sous-échantillons $\Xi^1_{train}, ..., \Xi^{n_p}_{train}$;
$\ll$ chaque processeur reçoit un sous-échantillon $\gg$
// *Nous supposons que le processeur i reçoit le sous-échantillon* Ξ^i_{train}, $1 \leq i \leq n_p$
$\boldsymbol{\mu}_1$ est choisi aléatoirement dans Ξ^1_{train} par processeur 1 ;
$\ll$ Processeur 1 envoie $\boldsymbol{\mu}_1$ aux autres processeurs $\gg$
$\epsilon \leftarrow \infty$;
$S_1 \leftarrow \boldsymbol{\mu}_1$;
$N \leftarrow 1$;
$W_{1_{pr}} \leftarrow u_{\mathcal{N}}(\boldsymbol{\mu}_1)$; $W_{1_{du}} \leftarrow \Psi_{\mathcal{N}}(\boldsymbol{\mu}_1)$;
while $\epsilon > \epsilon_{tol}$ et $N \leq N_{max}$ **do**
  // *Chaque processeur sélectionne son meilleur candidat*
  $\boldsymbol{\mu}^i_N \leftarrow \arg \max_{\boldsymbol{\mu} \in \Xi^i_{train}} \Delta^{s,i}_{N-1}(\boldsymbol{\mu}),\ 1 \leq i \leq n_p$;
  $\ll$ Chaque processeur envoie son meilleur candidat $\boldsymbol{\mu}^i_N$,
  ainsi que l'estimation d'erreur $\Delta^{s,i}_{N-1}(\boldsymbol{\mu}^i_N)$ associée, au processeur 1 $\gg$
  //*Processeur 1 sélectionne le meilleur des meilleurs candidats*
  $\boldsymbol{\mu}_N = arg \max_{\boldsymbol{\mu}^i_N, i \in \{1,...,n_p\}} \Delta^{s,i}_{N-1}(\boldsymbol{\mu}^i_N)$
  $\ll$ Chaque processeur reçoit le vecteur de paramètres $\boldsymbol{\mu}_N$ $\gg$
  $\epsilon \leftarrow \Delta^s_{N-1}(\boldsymbol{\mu}_N)$;
  $S_N \leftarrow S_{N-1} \cup \boldsymbol{\mu}_N$;
  $N \leftarrow N + 1$;
  $W_{N_{pr}} \leftarrow W_{N-1_{pr}} \cup span\{u_{\mathcal{N}}(\boldsymbol{\mu}_N)\}$; $W_{N_{du}} \leftarrow W_{N-1_{du}} \cup span\{\Psi_{\mathcal{N}}(\boldsymbol{\mu}_N)\}$;
  //*Rappelons que les solutions* $u_{\mathcal{N}}(\boldsymbol{\mu}_N)$ *et* $\Psi_{\mathcal{N}}(\boldsymbol{\mu}_N)$ *sont distribuées*
  //*sur l'ensemble des processeurs*
end while

Visualisation

Le même problème apparaît lorsque nous voulons visualiser le champ solution RB. Rappelons que la solution RB $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ s'exprime, pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u^i_N(\boldsymbol{\mu})\ \xi^{pr}_i, \tag{8.1}$$

où N est la dimension de $W_{N_{pr}}$, ξ^{pr}_i pour $i = 1, ..., N$ sont les fonctions de base RB et $u^i_N(\boldsymbol{\mu})$, pour $i = 1, ..., N$, sont les coefficients de l'expansion RB. Les fonctions de base RB sont indispensables pour visualiser $u_N(\boldsymbol{\mu})$ sur le maillage. Par conséquent, une fois encore, il est impératif d'avoir exactement le même partitionnement du maillage pour pouvoir charger les fonctions de base RB.

3 La méthode EIM

Intéressons-nous maintenant à la méthode EIM. Nous présentons ce qui a été fait pour que cette méthode puisse être utilisée sur une architecture parallèle.

3.1 Partie *hors-ligne*

Durant la construction des fonctions de base, à chaque itération, la sélection du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$ nécessite le calcul du maximum d'une norme infinie d'une quantité $Q(\boldsymbol{\mu}, \boldsymbol{x})$ – voir (3.7) –. La quantité Q dépendant du maillage, pour un $\boldsymbol{\mu} \in \Xi_{train}$ fixé, chaque processeur évalue alors le maximum local – défini sur la partition du maillage qu'il a – de $Q(\boldsymbol{\mu}, \boldsymbol{x})$. Ce maximum local est ensuite envoyé au processeur maître, via des communications point à point, pour que ce dernier puisse calculer le maximum global – défini sur l'ensemble du maillage et associé à $\boldsymbol{\mu}$ –. Ce procédé est répété pour l'ensemble des vecteurs de paramètres contenus dans l'échantillon Ξ_{train}. Enfin, le processeur maître sélectionne le $\boldsymbol{\mu} \in \Xi_{train}$ pour lequel $Q(\boldsymbol{\mu}, \boldsymbol{x})$ atteint son maximum, puis envoie ce vecteur de paramètres aux autres processeurs en utilisant une communication collective.

L'algorithme 8 illustre comment est évaluée la norme infinie de la quantité Q dans un calcul de type $arg\max$, lorsque le maillage est réparti sur plusieurs processeurs. Notons Ω_i, $1 \leq i \leq n_p$, la $i^{\text{ème}}$ partition du maillage Ω sur lequel évolue Q. Nous supposons que le processeur i possède la partition Ω_i. $\boldsymbol{\mu}_i \in \Xi_{train}$ est le $i^{\text{ème}}$ vecteur de paramètres appartenant à l'échantillon Ξ_{train}, avec $1 \leq i \leq Card(\Xi_{train})$.

Algorithm 8 $Max = arg \max_{\boldsymbol{\mu} \in \Xi_{train}} ||Q(\boldsymbol{\mu}, \boldsymbol{x})||_{L^\infty(\Omega)}$

for $i = 1$ to $Card(\Xi_{train})$ **do**
  //Le processeur j calcule le maximum de $Q(\boldsymbol{\mu}_i, .)$ sur Ω_j
  $max\_local\_j \leftarrow \max_{\boldsymbol{x} \in \Omega_j} Q(\boldsymbol{\mu}^i, \boldsymbol{x})$;
  ≪ Chaque processeur envoie le maximum local au processeur maître ≫
  //Le processeur maître calcule $max^{\boldsymbol{\mu}_i} = ||Q(\boldsymbol{\mu}_i, \boldsymbol{x})||_{L^\infty(\Omega)}$
  $max^{\boldsymbol{\mu}_i} \leftarrow \max_{j \in \{1,\dots,n_p\}} max\_local\_j$
end for
//Le processeur maître calcule $Max = arg \max_{\boldsymbol{\mu} \in \Xi_{train}} Q(\boldsymbol{\mu}, \boldsymbol{x})$
$Max \leftarrow \max_{i \in \{1,\dots,Card(\Xi_{train})\}} max^{\boldsymbol{\mu}_i}$
≪ Le processeur maître envoie Max aux autres processeurs ≫

Remarque 17. *Attention, la quantité Q dépendant à la fois du maillage et de l'espace des paramètres, il est important que tous les processeurs possèdent le même $\boldsymbol{\mu}$. Sinon le calcul du maximum de Q n'a plus de sens. Il ne faut donc surtout pas reproduire la stratégie utilisée dans la méthode CRBM.*

De même, lorsque la méthode CRBM est mise en oeuvre et qu'elle fait appel à la méthode EIM pour approcher des termes de la décomposition affine (voir les sections 2.3 et 3.3 du chapitre 3), il devient impératif que tous les processeurs possèdent le même vecteur de paramètres $\boldsymbol{\mu}$. En effet, l'erreur commise en remplaçant certains termes par l'expansion EIM associée dépend à la fois de l'espace des paramètres et du maillage – voir (3.85) –.

3.2 Partie *en-ligne*

La base de données associée à la classe EIM contient la matrice d'interpolation – commune à tous les processeurs –, ainsi que les fonctions de base – réparties sur l'ensemble des processeurs –. Il est donc, là encore, indispensable d'avoir le maillage partitionné de la même manière que lors de la partie *hors-ligne*. Dans le cas contraire, la base de données ne peut être chargée, et les calculs effectués durant la partie *hors-ligne* sont perdus. Contrairement à la méthode CRBM, si la construction de la base EIM a nécessité l'utilisation de n_p processeurs, il n'est pas possible de lancer la partie *en-ligne* sur un seul processeur. Cependant, comme indiqué dans la remarque 16, l'étape *en-ligne* dépend de la dimension élément fini. Il faut donc toujours charger le maillage utilisé pour les calculs *hors-ligne*.

4 La fonction evaluateFromContext

Rappelons que la fonction `evaluateFromContext` permet d'évaluer, rapidement, une expression *Expr* aux points d'interpolation associés à un CONTEXT . Dans le cas où le maillage est partitionné et réparti sur plusieurs processeurs, les points d'interpolation se retrouvent eux aussi répartis sur l'ensemble des processeurs. Chaque processeur possède donc zéro, un, ou plusieurs points d'interpolation.

Chaque processeur parcourt les points d'interpolation qu'il a, évalue l'expression *Expr* en ces points, puis envoie le résultat au processeur maître en utilisant une communication point à point. Une fois que le processeur maître a reçu toutes les évaluations de tous les processeurs, il envoie à tous les autres l'évaluation de *Expr* à tous les points d'interpolation en utilisant une communication collective.

Notons que la communication point à point que nous venons d'évoquer peut être désactivée en utilisant l'option `_mpi_communications=false`. Cette option est utilisée notamment lorsque le CONTEXTRB associé à REDUCEDBASISSPACE souhaite évaluer les fonctions de base RB en un seul point.

En effet, lorsqu'un point est ajouté à un CONTEXTRBSET, alors le CONTEXTRB est construit uniquement sur le processeur ayant la partition du maillage sur lequel se trouve le point. Par conséquent tous les processeurs n'ont pas forcément de CONTEXTRB, et donc tous ne font pas appel à la fonction `evaluateFromContext` pour évaluer les fonctions de base RB. C'est pour cette raison que, dans ce cas, il est nécessaire de désactiver les communications collectives.

5 Scalabilité

Nous illustrons dans cette partie le comportement du framework-RBM en parallèle. Pour cela, nous considérons le problème du bouclier thermique introduit dans le chapitre 2, section 5. Nous nous proposons d'étudier la scalabilité forte du framework-RBM, c'est à dire que la taille du problème est fixée alors que le nombre de processeurs varie. Nous nous intéressons plus particulièrement au speed-up défini par

$$\text{speed-up} = \frac{t^{n_p=4}}{t^{n_p=p}}, \text{ pour } p = 4, 8, 12, 16, 24 \text{ ou } 32, \tag{8.2}$$

et où $t^{n_p=p}$ désigne le temps CPU mesuré en utilisant p processeurs. Pour résoudre le problème nous utilisons un solveur itératif BICGSTAB – Biconjugate Gradient Stabilized – qui utilise une méthode de projection sur un sous-espace de Krylov.

Les préconditionneurs LU, GASM-1 et GASM-2 sont également utilisés. `PETSc` permet d'accéder à la librairie `MUMPS` [6, 7] qui offre un préconditionneur LU parallèle. Avec un tel préconditionneur, le solveur peut alors être vu comme un solveur direct. La construction de ce préconditionneur peut s'avérer chère en terme de coût de calcul. Notons que pour les problèmes transitoires, il est possible de le construire une seule fois, puis de le réutiliser à chaque pas de temps – contrairement à ce qui a été fait pour obtenir les résultats présentés ici –. Il peut également consommer beaucoup de mémoire lorsqu'il est utilisé avec de gros problèmes.

Le préconditionneur GASM – Generalized Additive Schwarz Method – est un préconditionneur de type Schwarz additif avec recouvrement algébrique, bien adapté lorsque nous avons beaucoup de sous-domaines. Introduisons la notion de sous-domaine : lorsque n_p processeurs sont utilisés, le domaine de départ est alors divisé en n_p sous-domaines. Lorsqu'un niveau de recouvrement (respectivement deux) entre deux sous-domaines est utilisé, nous notons alors ce préconditionneur GASM-1 (respectivement GASM-2). Les résultats présentés ici ont été obtenus en utilisant un sous-préconditionneur LU dans chaque sous-domaine. Notons que dans chaque sous-domaine, le sous-préconditionneur fonctionne de manière séquentielle.

Une description détaillée des solveurs et préconditionneurs utilisés dans Feel++ est faite dans [32].

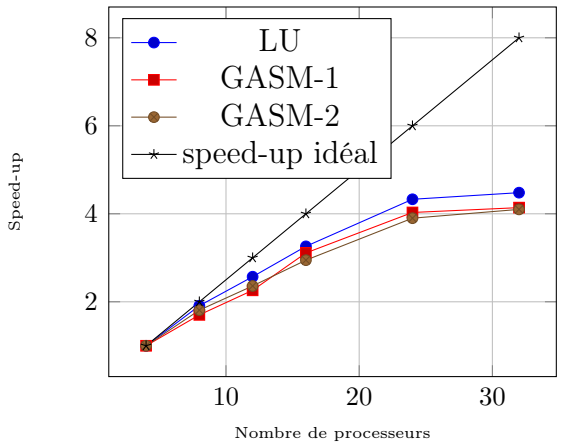
Les tests de scalabilité ont été réalisés sur des processeurs AMD Opteron(tm) 6386 SE, 2.7 Ghz.

Prenons une discrétisation élément fini $\mathbb{P}_3$ avec 815 000 degrés de liberté. Le nombre de pas de temps est fixé à 100. Le préconditionneur utilisé est reconstruit à chaque pas de temps. Nous introduisons également l'échantillon Ξ_{train} composé de 100 000 vecteurs de paramètres $\boldsymbol{\mu}$ sélectionnés de manière aléatoire dans l'espace des paramètres $\mathcal{D}$.

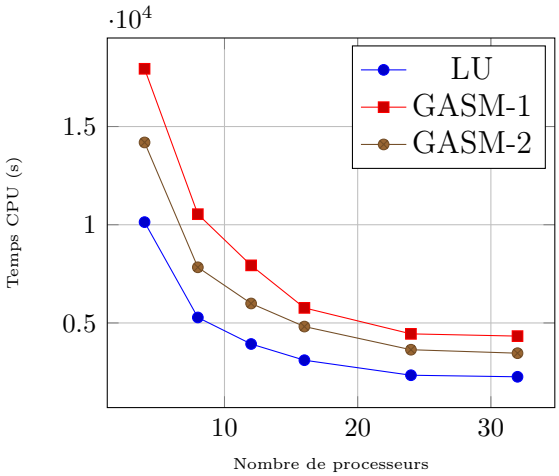
Commençons par étudier la scalabilité du framework-RBM lors de la construction du premier élément de la base réduite – voir les figures 8.3 (a) et (b) –. Cela comprend la résolution élément fini des problèmes primal et dual (2.52),(2.69), le calcul du mode propre principal (2.56), le calcul des termes définis en (2.140),(2.141),(2.142),(2.152), (2.153) qui sont utilisés pour le calcul de l'estimation d'erreur a posteriori, et enfin la sélection du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$, qui est celui qui maximise l'erreur estimée – voir l'algorithme 3 –. Le problème étant 2D et de dimension encore raisonnable, il est possible d'utiliser le préconditionneur LU. Si l'on regarde le temps CPU, les performances optimales sont obtenues logiquement en utilisant le préconditionneur LU, puis GASM-2 et enfin GASM-1. La scalabilité est quasiment identique avec les trois préconditionneurs utilisés.

Maintenant étudions plus en détails les différentes étapes qui interviennent lors de la construction du premier élément de la base réduite. Tout d'abord, intéressons-nous à la résolution élément fini du problème primal – voir les figures 8.3 (c) et (d) –. D'un point de vue performance, la conclusion est la même que précédemment. La scalabilité obtenue en utilisant GASM-1 et GASM-2 est cependant meilleure qu'en utilisant LU comme préconditionneur.

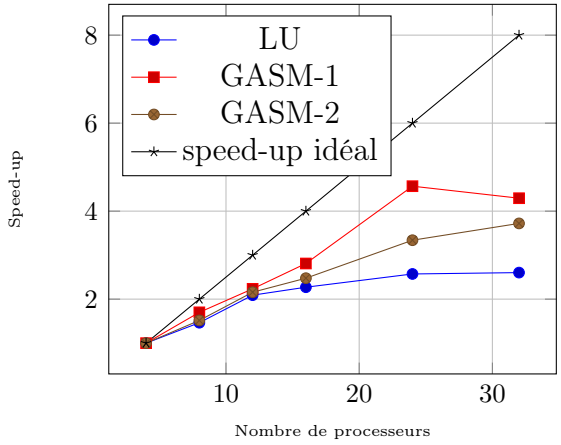
Au regard de la figure 8.3 (e), la scalabilité obtenue lors de l'assemblage de la matrice POD (2.55) est assez bonne.



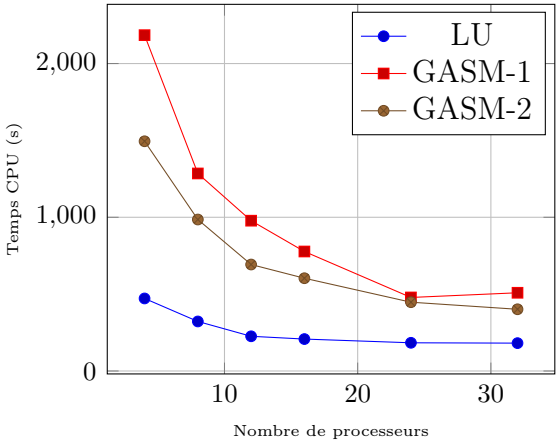
(a) Speed-up pour la construction complète du premier élément de la base réduite.



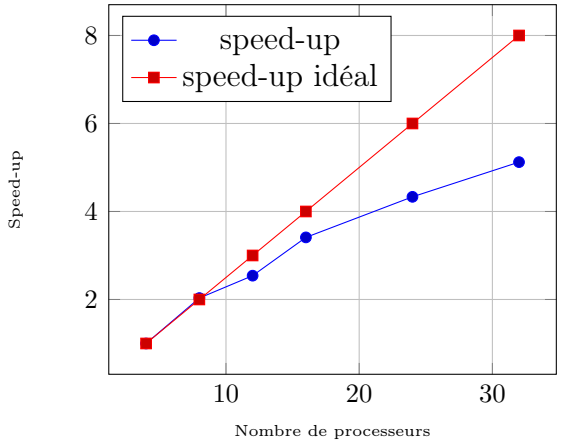
(b) Temps CPU pour la construction complète du premier élément de la base réduite.



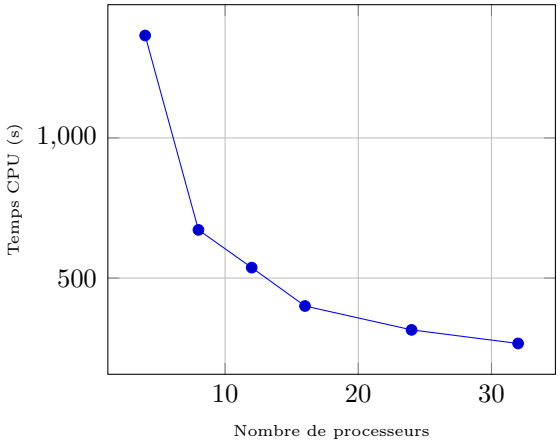
(c) Speed-up pour la résolution du problème primal FE.



(d) Temps CPU pour la résolution du problème primal FE.

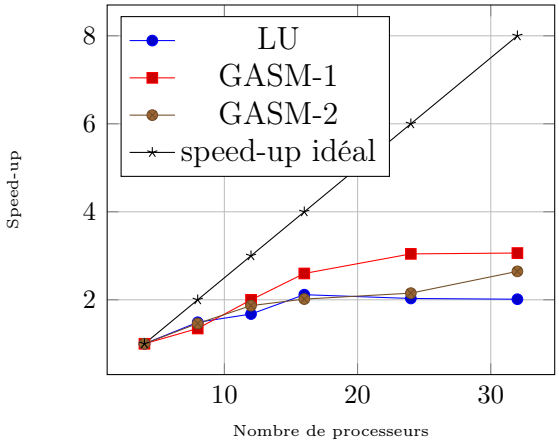


(e) Speed-up pour l'assemblage de la matrice POD.

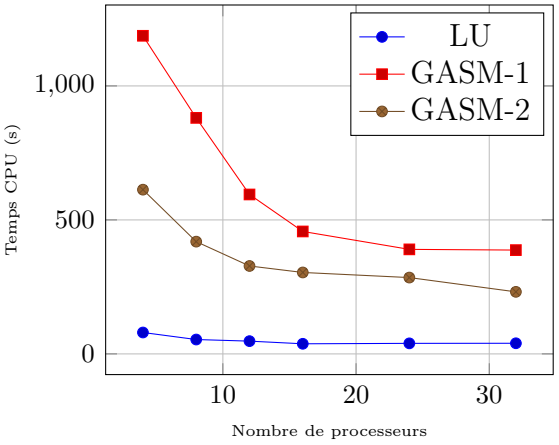


(f) Temps CPU pour l'assemblage de la matrice POD.

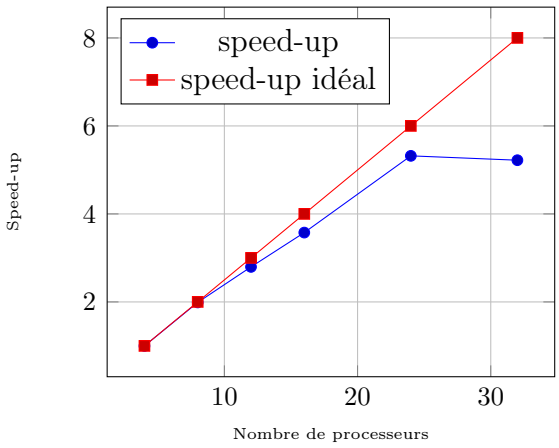
FIGURE 8.3 – Test de scalabilité lors de la création du premier élément de la base réduite. $\mathcal{N} = 815\ 000$, nombre de pas de temps : 100, $Card(\Xi_{train}) = 100\ 000$.



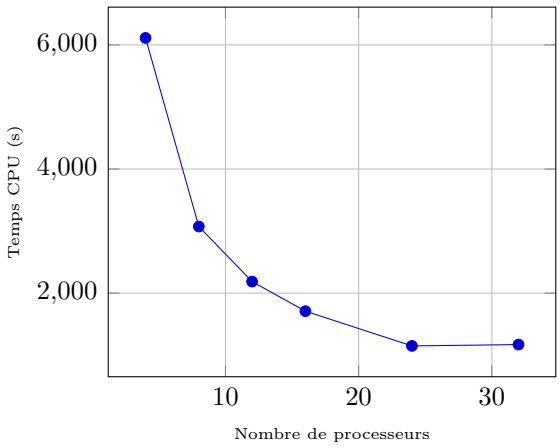
(a) Speed-up pour le calcul du terme $\Phi^{qiq'i'}_{N_{pr},aa}$, pour $1 \leq i, i' \leq N$ et $1 \leq q, q' \leq Q_a$.



(b) Temps CPU pour le calcul du terme $\Phi^{qiq'i'}_{N_{pr},aa}$, pour $1 \leq i, i' \leq N$ et $1 \leq q, q' \leq Q_a$.



(c) Speed-up pour la sélection du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$.



(d) Temps CPU pour la sélection du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$.

FIGURE 8.4 – Tests de scalabilité pour la construction du premier élément de la base réduite. Détails du calcul de $\Phi^{qiq'i'}_{N_{pr},aa}$, pour $1 \leq i, i' \leq N$ et $1 \leq q, q' \leq Q_a$ et du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$. $\mathcal{N} = 815\,000$, nombre de pas de temps : 100, $Card(\Xi_{train}) = 100\,000$ et $Q_a = 3$.

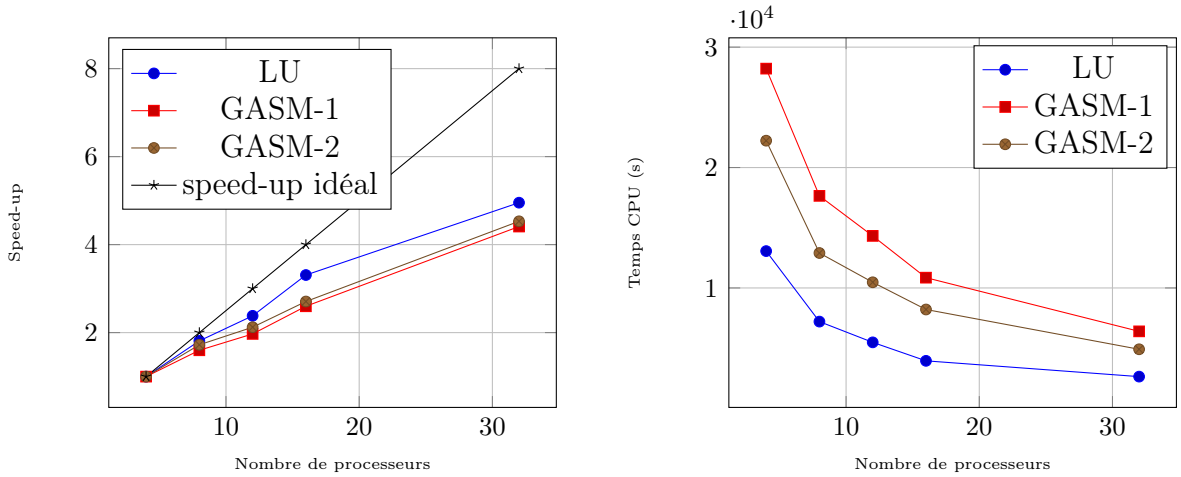
Notons que cette étape ne fait intervenir principalement que des calculs de produits scalaires entre solutions FE, par conséquent ce sont les communications collectives qui sont testées.

Les figures 8.4 (a) et (b) montrent respectivement la scalabilité et le temps CPU pour le calcul du terme $\Phi^{qiq'i'}_{N_{pr},aa}$ – utilisé pour le calcul de l'estimation d'erreur a posteriori et défini en (2.141) –, pour $1 \leq i, i' \leq N$ et $1 \leq q, q' \leq Q_a$, où N représente la dimension de la base réduite et Q_a le nombre de terme associés à la décomposition affine de la forme bilinéaire a. Nous constatons la formation d'un plateau lorsqu'on étudie la scalabilité, et ce avec les trois préconditionneurs utilisés. Cela est probablement dû au fait que le problème résolu soit de trop petite taille. Sans surprise, le préconditionneur LU offre la meilleure performance. Rappelons que pour calculer le terme $\Phi^{qiq'i'}_{N_{pr},aa}$, pour $1 \leq i, i' \leq N$ et $1 \leq q, q' \leq Q_a$, nous devons résoudre des problèmes de type Poisson, voir (2.131). Pour ces problèmes, seul le second membre change, donc le préconditionneur n'est pas

reconstruit. Dans le cas de l'utilisation du préconditionneur LU, cette réutilisation offre un gain de temps conséquent.

Enfin, nous terminons cette étude de scalabilité du framework-RBM lors de la construction du premier élément de la base réduite en nous concentrant sur le calcul du meilleur candidat $\boldsymbol{\mu} \in \Xi_{train}$ – voir les figures 8.4 (c) et (d) –. Rappelons que Ξ_{train} est de taille 100 000. Lorsque 32 processeurs sont utilisés, chacun reçoit un sous-échantillon de taille 3 125. Au regard de la courbe de scalabilité, nous sommes probablement passé sous un seuil au-delà duquel le découpage de Ξ_{train} est moins efficace.

En effet, la scalabilité est bonne jusqu'à 24 processeurs, puis un plateau se forme. Pour gagner en scalabilité la dimension de l'échantillon Ξ_{train} doit être augmentée et donc être supérieure à 100 000.



(a) Speed-up pour la construction complète de dix éléments de la base réduite.

(b) Temps CPU pour la construction complète de dix éléments de la base réduite.

FIGURE 8.5 – Test de scalabilité lors de la création des dix premiers éléments de la base réduite.

En se limitant à la construction de la première fonction de base, la projection du système initial se fait sur une base constituée d'un seul élément. Or, le temps de construction d'un élément de la base réduite augmente progressivement en fonction du nombre d'éléments déjà présents dans la base. On se propose donc d'étudier la scalabilité du framework-RBM sur la construction des dix premiers éléments de la base réduite.

Changeons la discrétisation élément fini qui devient $\mathbb{P}_2$ avec 131 000 degrés de liberté , mais le reste de la configuration reste inchangé.

D'après la figure 8.5, nous constatons que le framework-RBM passe à l'échelle même lorsqu'il y a plusieurs éléments dans la base réduite.

Résumé

Nous avons montré dans ce chapitre que le framework-RBM supportait les architectures parallèles en s'appuyant sur le parallélisme de la librairie FEEL++. *Nous avons également présenté les stratégies utilisées pour la parallélisation des méthodes impliquées dans le processus de réduction, ainsi que les modifications nécessaires à intégrer dans l'implémentation de la fonction* `evaluateFromContext` *lorsque le maillage est partitionné et distribué*

sur l'ensemble des processeurs. Enfin, un test de scalabilité forte du framework-RBM a été réalisé sur le modèle du bouclier thermique.

Troisième partie

Simulations numériques

Chapitre 9

Convection naturelle

Nous présentons dans ce chapitre une application de la technique de projection proposée dans le chapitre 5. Rappelons que ce travail a été effectué en collaboration avec E.Schenone et est présenté dans [111]. Intéressons-nous aux problèmes de type convection naturelle stationnaire. Ces derniers ont des applications dans plusieurs domaines de l'ingénierie. Être capable de mettre en oeuvre la méthode des bases réduites permet de gagner du temps de calcul lorsque la sortie du modèle doit être évaluée à plusieurs reprises.

1 Description du modèle

Considérons le problème de convection naturelle – stationnaire – paramétré par les nombres de Grashof et Prandtl [121, 84]. Il s'agit d'un fluide incompressible chauffé dans une cavité carrée ou cubique. Le fluide circule vers la température faible sous l'action de la densité de la différence de gravité. Introduisons les équations de Navier-Stokes adimensionnalisées couplées avec l'équation de la chaleur, en considérant une cavité en deux ou trois dimensions – voir la figure 9.1 –. Le problème s'énonce qui suit : chercher $(\boldsymbol{u}, p, T)$ tel que

$$\left\{\begin{array}{rcll} \boldsymbol{u}\cdot\nabla\boldsymbol{u}+\nabla p-\dfrac{1}{\sqrt{Gr}}\triangle\boldsymbol{u} &=& T\mathbf{e_2}, & \text{dans } \Omega \\ \nabla\cdot\boldsymbol{u} &=& 0, & \text{dans } \Omega \\ \boldsymbol{u}\cdot\nabla T-\dfrac{1}{\sqrt{Gr}Pr}\triangle T &=& 0, & \text{dans } \Omega \\ \boldsymbol{u} &=& \mathbf{0}, & \text{sur } \partial\Omega \\ T &=& 0, & \text{sur } \Gamma_1 \\ \dfrac{\partial T}{\partial \mathbf{n}} &=& 0, & \text{sur } \partial\Omega\setminus(\Gamma_1\cup\Gamma_3) \\ \dfrac{\partial T}{\partial \mathbf{n}} &=& 1, & \text{sur } \Gamma_3. \end{array}\right. \tag{9.1}$$

où $\Omega\subset\mathbb{R}^d$, $d=2,3$, $\boldsymbol{u}$, p et T sont repectivement le champ de vitesse, de pression, de température, adimensionnalisés. Gr et Pr sont les nombres de Grashof et the Prandtl, et $\mathbf{e_2}$ est le vecteur normal qui pointe vers l'intérieur de Ω associé au bord $\Gamma_2\subset\partial\Omega$. Le domaine 2D est une cavité rectangulaire de hauteur 1 et de longueur W, dans le cas 3D nous ajoutons une profondeur de 1. Un flux de chaleur est imposé sur le bord "droit" Γ_3 alors que la température est fixée sur le bord "gauche" Γ_1, enfin les bords sont isolés.

Des conditions limites similaires sont appliquées en 3D. Nous imposons des conditions de non-glissement au champ de vitesse sur l'ensemble des bords de la cavité.

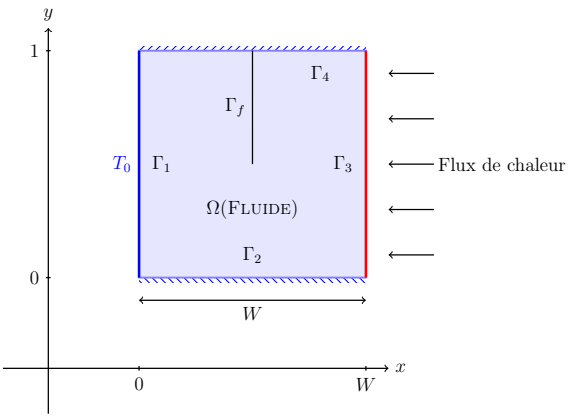


FIGURE 9.1 – Géométrie du modèle 2D. Dans le cas 3D il faut considérer une extrusion le long de l'axe z d'une longueur 1.

Nous considérons ici deux paramètres : les nombres de Grashof et de Prandtl, la sortie du modèle est la valeur moyenne de la température le long du bord Γ_3. Plus le nombre de Grashof – et/ou le nombre de Prandtl – augmente, plus la température moyenne augmente.

1.1 Discrétisation élément fini

Nous nous penchons ici sur la formulation du problème (9.1) en utilisant la discrétisation élément fini, ainsi que sur une méthode itérative pour le résoudre. La formulation variationnelle du problème (9.1) s'énonce comme suit : chercher $(\boldsymbol{u}, p, T) \in \boldsymbol{X} \times Q \times S$ tel que

$$\begin{aligned} a(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) - b(p,\boldsymbol{v}) + \frac{1}{\sqrt{Gr}} c(\boldsymbol{u},\boldsymbol{v}) - d(T,\boldsymbol{v}) = 0, \quad & \forall \boldsymbol{v} \in \boldsymbol{X}, \\ b(q,\boldsymbol{u}) = 0, \quad & \forall q \in Q, \\ e(T,\boldsymbol{u},s) + \frac{1}{\sqrt{Gr}Pr} f(T,s) - \frac{1}{\sqrt{Gr}Pr} h(s) = 0, \quad & \forall s \in S, \end{aligned} \tag{9.2}$$

où $\boldsymbol{X} \equiv [H_0^1(\Omega)]^d$, $Q \equiv L^2(\Omega)$, $S \equiv \{s \in H^1(\Omega) \text{ tel que } s|_{\Gamma_1} = 0\}$. Les formes trilinéaires $a : \boldsymbol{X} \times \boldsymbol{X} \times \boldsymbol{X} \to \mathbb{R}$ et $e : S \times \boldsymbol{X} \times S \to \mathbb{R}$ sont définies par

$$a(\boldsymbol{u},\boldsymbol{w},\boldsymbol{v}) = \int_\Omega (\boldsymbol{w} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v}, \ \forall \boldsymbol{u} \in \boldsymbol{X}, \forall \boldsymbol{w} \in \boldsymbol{X}, \boldsymbol{v} \in \boldsymbol{X}, \tag{9.3}$$

$$e(T,\boldsymbol{v},s) = \int_\Omega (\boldsymbol{v} \cdot \nabla T)\, s, \ \forall \boldsymbol{v} \in \boldsymbol{X}, \ \forall T \in S, \forall s \in S. \tag{9.4}$$

Les formes bilinéaires $b : Q \times \boldsymbol{X} \to \mathbb{R}$, $c : \boldsymbol{X} \times \boldsymbol{X} \to \mathbb{R}$, $d : S \times \boldsymbol{X} \to \mathbb{R}$ et $f : S \times S \to \mathbb{R}$ sont définies par

$$b(q,\boldsymbol{v}) = \int_\Omega q \nabla \cdot \boldsymbol{v}, \ \forall \boldsymbol{v} \in \boldsymbol{X}, \forall q \in Q, \tag{9.5}$$

$$c(\boldsymbol{w},\boldsymbol{v}) = \int_\Omega \nabla \boldsymbol{w} : \nabla \boldsymbol{v}, \ \forall \boldsymbol{w} \in \boldsymbol{X}, \forall \boldsymbol{v} \in \boldsymbol{X}, \tag{9.6}$$

$$d(s,\boldsymbol{v}) = \int_\Omega s\,\mathbf{e_2}\cdot\boldsymbol{v},\ \forall \boldsymbol{v}\in\boldsymbol{X},\ \forall s\in S, \tag{9.7}$$

$$f(T,s) = \int_\Omega \nabla T\cdot\nabla s,\ \forall T\in S,\ \forall s\in S. \tag{9.8}$$

Enfin l'opérateur linéaire $h : S \to \mathbb{R}$ est défini par

$$h(s) = \int_{\Gamma_3} s,\ \forall s\in S. \tag{9.9}$$

Introduisons à maintenant la formulation discrète de (9.2). Pour cela définissons les espaces discrets $\boldsymbol{X}_h \subset \boldsymbol{X}$, $Q_h \subset Q$, $S_h \subset S$ ainsi que la projection de la solution $(\boldsymbol{u}, p, T)$ de (9.2) par

$$\boldsymbol{X}_h \equiv \text{span}\{\boldsymbol{\phi}_1,\dots,\boldsymbol{\phi}_{\mathcal{N}_u}\},\ \boldsymbol{u}\simeq\sum_{j=1}^{\mathcal{N}_u}\boldsymbol{u}_j\boldsymbol{\phi}_j,$$

$$Q_h \equiv \text{span}\{\psi_1,\dots,\psi_{\mathcal{N}_p}\},\ p\simeq\sum_{j=1}^{\mathcal{N}_p}p_j\psi_j,$$

et

$$S_h \equiv \text{span}\{\varphi_1,\dots,\varphi_{\mathcal{N}_T}\},\ T\simeq\sum_{j=1}^{\mathcal{N}_T}T_j\varphi_j.$$

La formulation discrète de (9.2) s'écrit pour $k_1 = 1,...,\mathcal{N}_u$, $k_2 = 1,...,\mathcal{N}_p$, $k_3 = 1,...,\mathcal{N}_T$,

$$\begin{aligned}
\sum_{i,j=1}^{N_u}\boldsymbol{u}_i\boldsymbol{u}_j a(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j,\boldsymbol{\phi}_{k_1}) + \sum_{i=1}^{N_u}\frac{1}{\sqrt{Gr}}\mathbf{u}_i c(\boldsymbol{\phi}_i,\boldsymbol{\phi}_{k_1}) - \sum_{i=1}^{N_p}p_i b(\psi_i,\boldsymbol{\phi}_{k_1}) - \sum_{i=1}^{N_T}T_i d(\varphi_i,\boldsymbol{\phi}_{k_1}) &= 0,\\
\sum_{i=1}^{N_u}\boldsymbol{u}_i b(\psi_{k_2},\boldsymbol{\phi}_i) &= 0,\\
\sum_{i=1}^{N_T}\sum_{j=1}^{N_u}T_i\boldsymbol{u}_j e(\varphi_i,\boldsymbol{\phi}_j,\varphi_{k_3}) + \frac{1}{\sqrt{Gr}Pr}\sum_{i=1}^{N_T}T_i f(\varphi_i,\varphi_{k_3}) - \frac{1}{\sqrt{Gr}Pr}h(\varphi_{k_3}) &= 0.
\end{aligned} \tag{9.10}$$

Parce que lorsque les nombres de Grashof ou Prandtl sont élevés, la non-linéarité devient forte, une méthode itérative robuste doit être mise en place pour résoudre ce problème. Nous utilisons ici une méthode de Newton. Pour un vecteur de paramètres $\boldsymbol{\mu} = (\mu^0,\mu^1) = (Gr^{1/2}, Pr^{-1})$ donné ainsi qu'une valeur initiale $(\boldsymbol{u}^0, p^0, T^0)$, à chaque itération n de l'algorithme – avec $1 \le n \le n_{max}$, où n_{max} est en entier strictement positif donné –, nous cherchons $(\boldsymbol{u}^{n+1}, p^{n+1}, T^{n+1}) \in \mathbb{R}^{\mathcal{N}_u}\times\mathbb{R}^{\mathcal{N}_p}\times\mathbb{R}^{\mathcal{N}_T}$ tel que

$$J\Big(\boldsymbol{u}^n,p^n,T^n;\boldsymbol{\mu}\Big)\Big[\big(\boldsymbol{u}^{n+1},p^{n+1},T^{n+1}\big)-\big(\boldsymbol{u}^n,p^n,T^n\big)\Big] = G\Big(\boldsymbol{u}^n,p^n,T^n;\boldsymbol{\mu}\Big) \tag{9.11}$$

où $J = J(\boldsymbol{u}, p, T; \boldsymbol{\mu})$ est la matrice jacobienne, et $G = G(\boldsymbol{u}, p, T; \boldsymbol{\mu})$ est le vecteur résidu. Regardons comment construire J et G. Pour chaque ligne $k_1 = 1, ..., \mathcal{N}_u$ nous avons

$$\begin{aligned}
&J_{k_1 i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = \sum_{j=1}^{\mathcal{N}_u} \boldsymbol{u}_j a\Big(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j, \boldsymbol{\phi}_{k_1}\Big) + \sum_{j=1}^{\mathcal{N}_u} \boldsymbol{u}_j a\big(\boldsymbol{\phi}_j, \boldsymbol{\phi}_i, \boldsymbol{\phi}_{k_1}\Big) + \mu^0 c\big(\boldsymbol{\phi}_i, \boldsymbol{\phi}_{k_1}\big), \quad i = 1, ..., \mathcal{N}_u,\\
&J_{k_1 \mathcal{N}_u + i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = -b\big(\psi_i, \boldsymbol{\phi}_{k_1}\big), \qquad i = 1, ..., \mathcal{N}_p,\\
&J_{k_1 \mathcal{N}_u + \mathcal{N}_p + i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = -d\big(\varphi_i, \boldsymbol{\phi}_{k_1}\big), \qquad i = 1, ..., \mathcal{N}_T.
\end{aligned} \tag{9.12}$$

Pour chaque ligne $k_2 = \mathcal{N}_u + k$, $k = 1, ..., \mathcal{N}_p$, nous pouvons écrire

$$\begin{aligned}
&J_{k_2 i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = b\big(\psi_k, \boldsymbol{\phi}_i\big), \quad i = 1, ..., \mathcal{N}_u,\\
&J_{k_2 \mathcal{N}_u + i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = 0, \qquad i = 1, ..., \mathcal{N}_p,\\
&J_{k_2 \mathcal{N}_u + \mathcal{N}_p + i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = 0, \quad i = 1, ..., \mathcal{N}_T.
\end{aligned} \tag{9.13}$$

Enfin, pour chaque ligne $k_3 = \mathcal{N}_u + \mathcal{N}_p + k$, $k = 1, ..., \mathcal{N}_T$

$$\begin{aligned}
&J_{k_3 i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = \sum_{j=1}^{\mathcal{N}_T} T_j e\big(\varphi_j, \boldsymbol{\phi}_i, \varphi_k\big), \qquad i = 1, ..., \mathcal{N}_u,\\
&J_{k_3 \mathcal{N}_u + i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = 0, \qquad i = 1, ..., \mathcal{N}_p,\\
&J_{k_3 \mathcal{N}_u + \mathcal{N}_p + i}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = \sum_{j=1}^{\mathcal{N}_u} \boldsymbol{u}_j e\big(\varphi_i, \boldsymbol{\phi}_j, \varphi_k\big) + \mu^0 \mu^1 f\big(\varphi_i, \varphi_k\big), \quad i = 1, ..., \mathcal{N}_T.
\end{aligned} \tag{9.14}$$

De la même manière, chaque terme du vecteur résidu $G(\boldsymbol{u}, p, T; \boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}_u + \mathcal{N}_p + \mathcal{N}_T}$ peut être exprimé par

$$\begin{aligned}
&G_k\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = -a\big(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{\phi}_k\big) + b\big(p, \boldsymbol{\phi}_k\big) - \mu^0 c\big(\boldsymbol{u}, \boldsymbol{\phi}_k\big) + d\big(T, \boldsymbol{\phi}_k\big), \quad k = 1, ..., \mathcal{N}_u,\\
&G_{N_u + k}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = -b\big(\psi_k, \boldsymbol{u}\big), \qquad k = 1, ..., \mathcal{N}_p,\\
&G_{N_u + N_p + k}\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) = \mu^0 \mu^1 h\big(\varphi_k\big) - e\big(T, \boldsymbol{u}, \varphi_k\big) - \mu^0 \mu^1 f\big(T, \varphi_k\big), \quad k = 1, ..., \mathcal{N}_T.
\end{aligned}$$

Remarque 18. *Pour des valeurs de Gr et Pr élevées, la méthode de Newton peut ne pas être suffisante. Dans ce cas nous proposons d'utiliser l'algorithme de continuation 9. Pour des valeurs données de Gr et Pr, les valeurs minimales Gr_{min}, Pr_{min}, le nombre maximum d'itération n_{max} et la tolérance ϵ_{tol} utilisés pour l'algorithme de Newton, ainsi qu'une valeur initiale pour la solution sol^0, l'algorithme de continuation renvoie la solution du problème. Pour cela, plusieurs valeurs intermédiaires des paramètres Pr et Gr sont utilisées. Notons cependant que cet algorithme n'a pas été appliqué pour les résultats présentés dans la suite de ce chapitre.*

1.2 Discrétisation base réduite

Appliquons la technique de réduction présentée au chapitre 5 à notre problème. Pour cela, nous introduisons les matrices et tenseurs assiciés à (9.10). Rappelons que dans le chapitre 5 nous avons vu qu'un tenseur pouvait être considéré comme un vecteur de matrices. Définissons les tenseurs $A \in \mathbb{R}^{\mathcal{N}_u \times \mathcal{N}_u \times \mathcal{N}_u}$ and $E \in \mathbb{R}^{\mathcal{N}_T \times \mathcal{N}_u \times \mathcal{N}_T}$ par

$$A = \big[A_{ijk},\ i, j, k = 1, ..., \mathcal{N}_u\big],\ (A^k)_{ij} = A_{ijk} = a\big(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j, \boldsymbol{\phi}_k\big),$$

Algorithm 9 Stratégie de continuation pour des valeurs de Gr et Pr élevées
$sol = continuation(Gr, Pr, Gr_{min}, Pr_{min}, n_{max}, \epsilon_{tol}, sol^0)$

Calcul du nombre de valeurs intermédiaires de Pr et Gr :
$N = \max\Big\{1; \max\big\{\lceil \log(Gr/Gr_{min})\rceil), \lceil (\log(Pr/Pr_{min})\rceil\big\}\Big\}$
for i=1 :N **do**
Mise à jour des paramètres utilisés :
$Gr \leftarrow exp\{\log(Gr_{min}) + i(\log(Gr/Gr_{min}))/N)$
$Pr \leftarrow exp\{\log(Pr_{min}) + i(\log(Pr/Pr_{min}))/N)$
$n \leftarrow 0$
while $||G|| \geq \epsilon_{tol}$ **or** $n \leq n_{max}$ **do**
$n \leftarrow n + 1$
chercher sol^n tel que $J(sol^{n-1})(sol^n - sol^{n-1}) = G(sol^{n-1})$
end while
end for
return $sol \leftarrow sol^n$

$$E = \big[E_{ijk},\ i,k = 1,...,\mathcal{N}_T, j = 1,...,\mathcal{N}_u\big],\ (E^k)_{ij} = E_{ijk} = e\big(\varphi_i, \boldsymbol{\phi}_j, \varphi_k\big),$$

ainsi que les matrices $B \in \mathbb{R}^{\mathcal{N}_p \times \mathcal{N}_u}$, $C \in \mathbb{R}^{\mathcal{N}_u \times \mathcal{N}_u}$, $D \in \mathbb{R}^{\mathcal{N}_u \times \mathcal{N}_T}$, $F \in \mathbb{R}^{\mathcal{N}_T \times \mathcal{N}_T}$ par

$$B = \big[B_{ij},\ i = 1,...,\mathcal{N}_u, j = 1,...,\mathcal{N}_p\big],\ B_{ij} = b\big(\psi_j, \boldsymbol{\phi}_i\big),$$

$$C = \big[C_{ij},\ i,j = 1,...,\mathcal{N}_u\big],\ C_{ij} = c\big(\boldsymbol{\phi}_j, \boldsymbol{\phi}_i\big),$$

$$D = \big[D_{ij},\ i = 1,...,\mathcal{N}_u, j = 1,..., N_T\big],\ D_{ij} = d\big(\varphi_j, \boldsymbol{\phi}_i\big),$$

$$F = \big[F_{ij},\ i = 1,...,\mathcal{N}_T\big],\ F_{ij} = f\big(\varphi_j, \varphi_i\big),$$

et enfin le vecteur $H \in \mathbb{R}^{\mathcal{N}_T}$ par

$$H = [H_i,\ i = 1,...,\mathcal{N}_T],\ H_i = h\big(\varphi_i\big).$$

La matrice jacobienne s'écrit alors

$$J(\boldsymbol{u}, p, T; \boldsymbol{\mu}) = \begin{bmatrix} J^{Nl_1}(\boldsymbol{u}) + \mu^0 C & -B & -D \\ B^T & \mathbf{0} & \mathbf{0} \\ J^{Nl_2}(T) & \mathbf{0} & J^{Nl_3}(\boldsymbol{u}) + \mu^0\mu^1 F \end{bmatrix}. \tag{9.15}$$

Chaque élément des sous-matrices non-linéaires $J^{Nl_1}(\boldsymbol{u})$, $J^{Nl_2}(T)$ et $J^{Nl_3}(\boldsymbol{u})$ est donné par

$$\begin{aligned} J^{Nl_1}_{ki}\big(\boldsymbol{u}\big) &= \sum_{j=1}^{\mathcal{N}_u} \big((A^k)_{ij} + (A^k)_{ji}\big)\boldsymbol{u}_j = \big((A^k_{.i})^T + (A^k_{i.})\big)\boldsymbol{u}\ ,\ k,i = 1,...,\mathcal{N}_u, \\ J^{Nl_2}_{ki}\big(T\big) &= \sum_{j=1}^{\mathcal{N}_T} \big(E^k\big)_{ij} T_j = \big(E^k_{i.}\big)T\ ,\ k = 1,...,\mathcal{N}_T, i = 1,...,\mathcal{N}_u, \\ J^{Nl_3}_{ki}\big(\boldsymbol{u}\big) &= \sum_{j=1}^{\mathcal{N}_u} \big(E^k\big)_{ji} \boldsymbol{u}_j = \big(E^k_{.i}\big)^T \boldsymbol{u}\ ,\ k,i = 1,...,\mathcal{N}_T. \end{aligned} \tag{9.16}$$

où $\left(A^k\right)_{.i}$ et $\left(A^k\right)_{i.}$ sont respectivement la $i^{\text{ème}}$ colonne et la $i^{\text{ème}}$ ligne de A^k, pour $k = 1, ..., \mathcal{N}_u$. Les mêmes notations sont utilisées pour désigner un colonne ou une ligne d'une autre matrice. Chaque élément du vecteur résidu s'écrit

$$\begin{aligned} G_k\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) &= -\,\boldsymbol{u}^T A^k \boldsymbol{u} + B_{k.} p - \mu_1 C_{k.}\mathbf{u} + D_{k.} T \ , \ k = 1, ..., \mathcal{N}_u, \\ G_k\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) &= -\,(B_{.k})^T \boldsymbol{u} \ , \ k = \mathcal{N}_u + 1, ..., \mathcal{N}_u + \mathcal{N}_p, \\ G_k\big(\boldsymbol{u}, p, T; \boldsymbol{\mu}\big) &= \mu^0\mu^1 H_k - \boldsymbol{u}^T E^k T - \mu^0\mu^1 F_{k.} T \ , \ k = \mathcal{N}_u + \mathcal{N}_p + 1, ..., \mathcal{N}_u + \mathcal{N}_p + \mathcal{N}_T. \end{aligned} \tag{9.17}$$

Rappelons que pour mettre en oeuvre de manière efficace la stratégie *hors-ligne/en-ligne*, l'ingrédient clé est la décomposition affine – en paramètres – des termes utilisés dans la méthode de Newton. Cette décomposition est immédiate pour notre modèle. La matrice jacobienne peut s'écrire

$$J(\boldsymbol{u}, p, T; \boldsymbol{\mu}) = \sum_{q=1}^{Q_J} \theta_J^q(\boldsymbol{\mu}) J^q(\boldsymbol{u}, p, T) \tag{9.18}$$

avec ici $Q_J = 4$ et $\theta_J^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q, \leq Q_J$, sont définies par

$$\theta_J^1(\boldsymbol{\mu}) = \mu^0 = \frac{1}{\sqrt{Gr}}, \quad \theta_J^2(\boldsymbol{\mu}) = \mu^0\mu^1 = \frac{1}{\sqrt{Gr}Pr}, \quad \theta_J^3(\boldsymbol{\mu}) = \theta_J^4(\boldsymbol{\mu}) = 1. \tag{9.19}$$

Les matrices J^q, $1 \leq q \leq Q_J$ peuvent être décrites en utilisant les notations introduites ci-dessus,

$$\begin{aligned} J^1(\boldsymbol{u}, p, T) &= \begin{bmatrix} C & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad J^2(\boldsymbol{u}, p, T) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & F \end{bmatrix}, \\ J^3(\boldsymbol{u}, p, T) &= \begin{bmatrix} \mathbf{0} & -B & -D \\ B^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad J^4(\boldsymbol{u}, p, T) = \begin{bmatrix} J^{Nl_1}(\boldsymbol{u}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ J^{Nl_2}(T) & \mathbf{0} & J^{Nl_3}(\boldsymbol{u}) \end{bmatrix}. \end{aligned} \tag{9.20}$$

De même, le résidu admet la décomposition affine suivante :

$$G(\boldsymbol{u}, p, T; \boldsymbol{\mu}) = \sum_{q=1}^{Q_R} \theta_G^q(\boldsymbol{\mu}) G^q(\boldsymbol{u}, p, T) \tag{9.21}$$

avec ici $Q_R = 3$ et $\theta_G^q : \mathcal{D} \to \mathbb{R}$, $1 \leq q, \leq Q_G$, sont définies par

$$\theta_G^1(\boldsymbol{\mu}) = \mu^0 = \frac{1}{\sqrt{Gr}}, \quad \theta_G^2(\boldsymbol{\mu}) = \mu^0\mu^1 = \frac{1}{\sqrt{Gr}Pr}, \quad \theta_G^3(\boldsymbol{\mu}) = 1. \tag{9.22}$$

Les vecteurs G^q, $1 \leq q \leq Q_G$, peuvent être décrits par

$$\begin{aligned} G^1(\boldsymbol{u}, p, T) &= \begin{bmatrix} -C\boldsymbol{u} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \ G^2(\boldsymbol{u}, p, T) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ H - FT \end{bmatrix}, \\ G^3(\boldsymbol{u}, p, T) &= \begin{bmatrix} -\big(\boldsymbol{u}^T A_k \boldsymbol{u}\big)_{k=1}^{\mathcal{N}_u} + Bp + DT \\ B^T \boldsymbol{u} \\ -\big((\boldsymbol{u}^T E_k T)^T\big)_{k=1}^{\mathcal{N}_T} \end{bmatrix}. \end{aligned} \tag{9.23}$$

Nous devons à présent projeter ces équations sur l'espace de dimension réduite $\boldsymbol{W_N} = span\{\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_N\}$, avec $N << \mathcal{N}_u + \mathcal{N}_p + \mathcal{N}_T$ et $\boldsymbol{\xi}_i \in \boldsymbol{X}_h \times Q_h \times S_h$ pour $i = 1, ..., N$. La méthode de Newton s'écrit, pour chaque itération $n = 1, ..., n_{max}$,

$$J_N\Big({\boldsymbol{u}_N}^n, p_N^n, T_N^n\Big)\Big[({\boldsymbol{u}_N}^{n+1}, p_N^{n+1}, T_N^{n+1}) - ({\boldsymbol{u}_N}^n, p_N^n, T_N^n)\Big] = G_N\Big(\boldsymbol{u}_N^n, T_N^n\Big) \tag{9.24}$$

où $\boldsymbol{u}_N$, p_N et T_N sont les projections de $\boldsymbol{u}$, p et T sur l'espace $\boldsymbol{W_N}$. En reprenant les notations introduites dans le chapitre 5, la projection sur $\boldsymbol{W_N}$ est effectuée en utilisant la matrice Ξ définie par les fonctions de base $\boldsymbol{\xi}_i$, $1 \leq i \leq N$ – voir 5.22 et 5.23 –. Cependant, les tenseurs A et E sont de très grande dimension et doivent être traités de manière différente en utilisant un tenseur hybride comme indiqué dans le chapitre 5. Avec cette technique, la projection d'un tenseur sur l'espace réduit revient à projeter N matrices.

Remarque 19. *La pression est incluse dans l'espace d'approximation de dimension réduite $\boldsymbol{W_N}$ bien qu'il s'agisse d'un multiplicateur de Lagrange pour garantir la contrainte de la divergence nulle sur le champ de vitesse dans le problème original – discrétisé via la méthode FEM –. La pression peut être supprimée de $\boldsymbol{W_N}$ sans que cela impacte la solution réduite. En effet les fonctions de base de l'espace réduit sont déjà à divergence nulle. Si la pression est incluse dans $\boldsymbol{W_N}$, ce n'est que pour des raisons de commodité lors de la phase d'implémentation.*

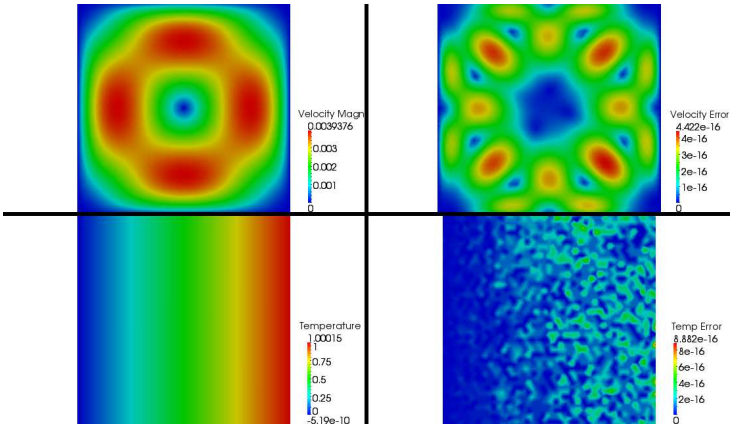
2 Résultats numériques

Nous présentons ici quelques résultats numériques. Tout d'abord nous comparons les profils d'écoulement obtenus en utilisant les discrétisations élément fini et base réduite, ainsi que les erreurs associées, dans le cas 2D et 3D. Puis, le temps de calcul ainsi que la température moyenne sont étudiés avec les deux discrétisations. Rappelons que les modèles 2D et 3D – élément fini et base réduite – utilisent la même librairie : FEEL++.

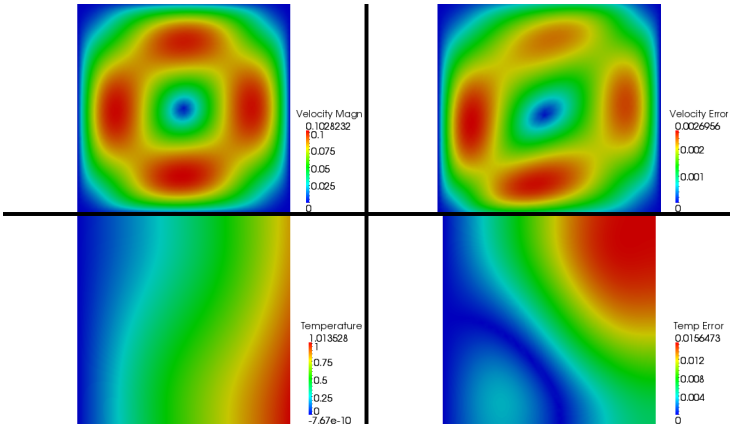
Quelque soit la dimension d de l'espace, une discrétisation élément fini de type $\mathbb{P}_3 - \mathbb{P}_2 - \mathbb{P}_3$ a été utilisée, c'est à dire que les fonctions de base sont des polynômes de degré 3 pour le champ de vitesse et de température, et de degré 2 pour le champ de pression. Pour le cas 2D, nous avons 6×10^4 degrés de liberté distribués sur 10 processeurs, et pour le cas 3D nous utilisons plus de 2×10^4 degrés de liberté distribués sur 24 processeurs. Rappelons que l'algorithme de continuation 9 n'est utilisé dans aucune simulation, ni élément fini ni base réduite. Nous avons utilisé seulement la méthode de Newton. Le solveur *GMRES* est utilisé dans tous les cas, mais dans le cas 2D nous utilisons une méthode de Schwarz additive comme préconditionneur – GASM –, alors que pour le cas 3D nous utilisons le préconditionneur *LU* de la librairie `MUMPS`. Pour tous les résultats présentés par la suite, la base de l'espace $\boldsymbol{W_N}$ contient 28 éléments non orthonormalisés. Le nombre de Prandtl ne varie pas et est fixé à 1, tandis que le nombre de Grashof varie de 1 à 10^6. Pour construire la base, ce paramètre varie suivant une loi de distribution uniforme. Une meilleure construction peut être obtenue en utilisant les estimateurs d'erreur a posteriori.

2.1 Profils d'écoulement

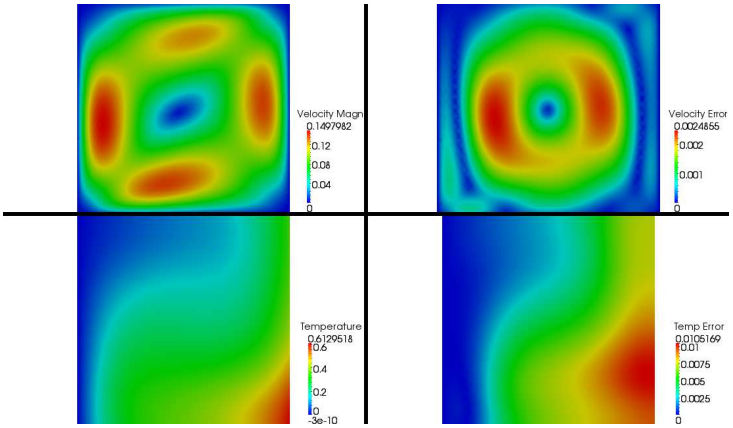
Commençons par étudier les solutions obtenues via l'utilisation des bases réduites, puis comparons les avec celles obtenues via la méthode FEM. Nous nous intéressons à



(a) Gr=1.e+00



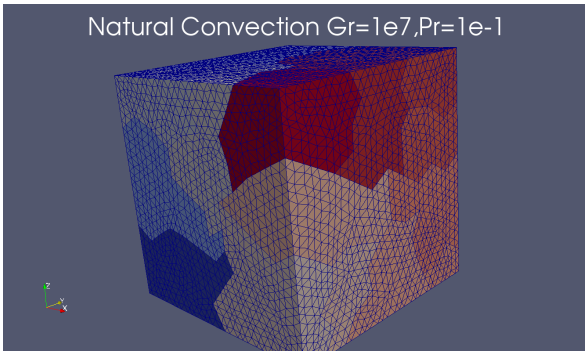
(b) Gr=1.e+03



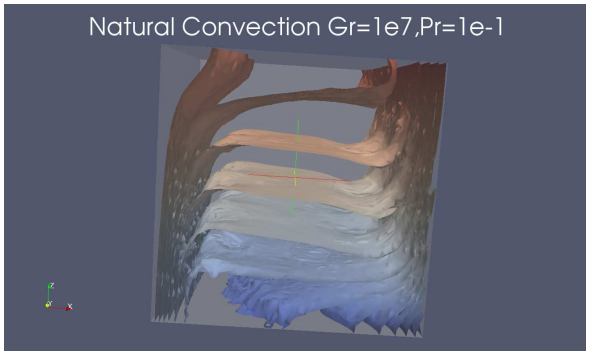
(c) Gr=1.e+05

FIGURE 9.2 – Comparaison des solutions obtenues en utilisant la méthode FEM et RBM pour $Gr = (1, 10^3, 10^5)$ et $Pr = 1$. À gauche, la magnitude du champ de vitesse (haut) et le profil de température (bas) pour les solutions base réduite. À droite, les erreurs associées aux champs de vitesse (haut) et de température (bas).

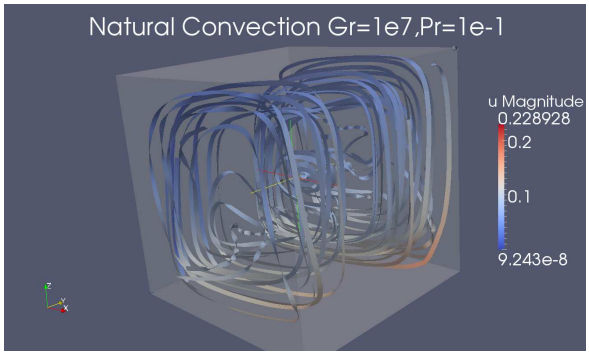
l'erreur relative entre les deux résolutions pour un écoulement de plus en plus turbulent – c'est à dire avec une valeur du nombre de Grashof de plus en plus grande – . La figure 9.2 montre les résultats sur la cavité 2D pour différentes valeurs du nombre de Grashof (1,10^3 et 10^5). À gauche se trouvent les solutions obtenues en utilisant la méthode RBM, pour chaque valeur du paramètre nous exhibons la magnitude du champ de vitesse (en haut) et le champ de température (en bas). Nous constatons que plus le nombre de Grashof augmente, plus l'écoulement est rapide, ce qui est le résultat attendu. Sur la droite se trouvent les erreurs relatives commises sur le champ de vitesse et de température. Nous remarquons que plus l'écoulement est turbulent, plus l'erreur entre la solution base réduite et élément fini est grande, ce qui est là encore le comportement attendu. De plus l'erreur est de l'ordre 2×10^{-3} sur la magnitude du champ de vitesse et 10^{-2} sur le profil de température. Ce sont des valeurs satisfaisantes. Notons que nous n'avons pas considéré le cas $Gr = 1$ car cela correspond à un paramètre utilisé pour construire la base réduite, dans ce cas l'erreur est de l'ordre de 10^{-16} comme attendu.



(a) Partitionnement du maillage (24 processeurs)



(b) Isosurfaces du champ de température



(c) Lignes de courant

FIGURE 9.3 – Calculs 3D pour $Gr = 10^7$ et $Pr = 0.1$

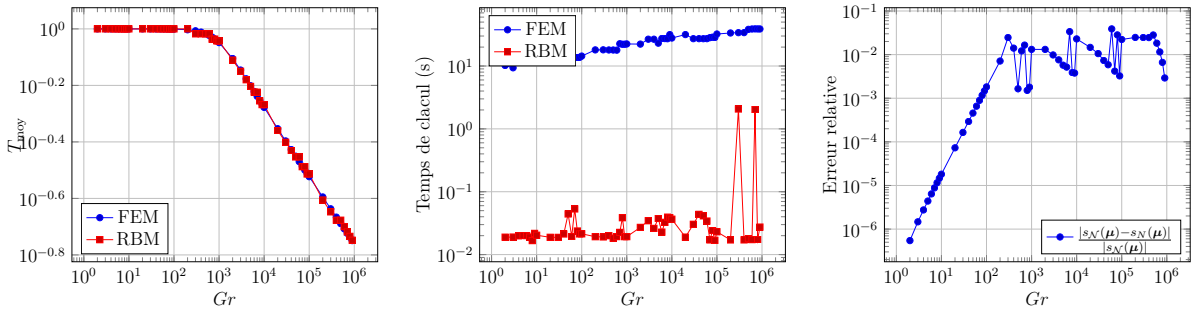
Nous analysons l'erreur sur les simulations 2D et 3D respectivement sur les figures 9.4(c) et 9.5(c), en augmentant la valeur du paramètre Gr. Notons $s_{\mathcal{N}}(\boldsymbol{\mu})$ et $s_N(\boldsymbol{\mu})$ respectivement la sortie évaluée en $\boldsymbol{\mu}$ à l'aide de la méthode FEM et RBM. L'erreur que nous étudions est l'erreur relative définie par $\frac{|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|}{|s_{\mathcal{N}}(\boldsymbol{\mu})|}$. Nous observons dans les deux cas que l'erreur augmente lorsque le nombre de Grashof est compris entre 1 et 10^3, puis l'erreur se stabilise pour les valeurs du paramètre supérieures à 10^3. Nous remarquons également que la solution dans le cas de la cavité 3D pour un haut Grashof représente un écoulement avec un dessin complexe – voir la figure 9.3 –.

2.2 Performances

Tournons-nous à présent vers les performances. Nous comparons ici le temps de calcul lorsque nous utilisons la méthode RBM et FEM. Ces temps de calcul sont montrés, en fonction de la valeur du nombre de Grashof, sur les figures 9.4(b) et 9.5(b) respectivement pour les simulations 2D et 3D. Dans les deux cas, le graphe du temps de calcul est tracé en utilisant une échelle log-log, en fonction de la valeur du paramètre Gr. Comme attendu, la méthode FEM est plus coûteuse, de plusieurs ordres de grandeur, que la méthode RBM sur ce problème. En particulier, pour un faible Grashof nous avons un facteur 6×10^2, puis ce facteur passe à 1.5×10^3 pour un haut Grashof dans le cas 2D, et ce facteur passe de 10^3 à 10^4 dans le cas 3D. De plus nous observons que le temps de calcul augmente lorsque la méthode FEM est utilisée, tandis qu'il reste quasiment constant avec la méthode RBM. Cependant, que ce soit dans le cas 2D ou 3D, nous avons trouvé quelques valeurs du paramètre pour lesquelles le temps de calcul via la méthode RBM est plus important. Cela est dû à l'augmentation du nombre d'itérations de l'algorithme de Newton pendant la phase *en-ligne*. Malgré cela, la méthode RBM conduit à de bonnes approximations au regard de la température moyenne – voir les figures 9.4(a) et 9.5(a) – .

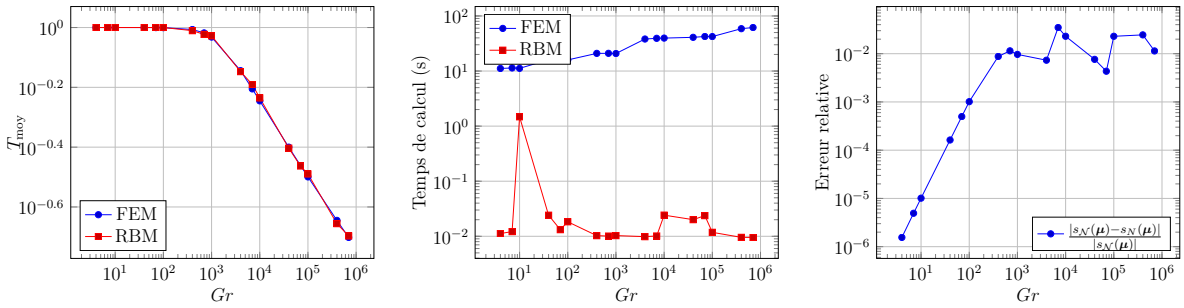
Attention, la phase *hors-ligne* n'est pas incluse dans le temps de calcul lors des comparaisons. Si nous le prenons en compte pour les comparaisons alors la méthode RBM devient compétitive à partir d'un nombre de simulations légèrement plus grand que la dimension de la base réduite N. En effet nous devons prendre en compte le coût de l'étape de projection qui est en $\mathcal{O}(N^3)$ à cause des tenseurs.

Comme nous avons vu, l'algorithme de continuation n'a été utilisé ni avec la méthode FEM ni avec la méthode RBM. Cependant dans le cas élément fini, cet algorithme peut devenir nécessaire lorsque la méthode de Newton ne converge pas, par exemple pour des valeurs de Grashof plus élevées ou une géométrie plus complexe. Comme valeur initiale de la solution pour appliquer l'algorithme de Newton, nous prenons la solution connue la plus proche dans le cas des bases réduites, tandis que dans le cas élément fini cette valeur initiale est nulle – bien que nous pouvons également nous servir de l'élément de la base réduite le plus proche pour initialiser la solution – .



(a) Température moyenne en fonction de Gr (b) Temps de calcul en fonction de Gr (c) Erreur en fonction de Gr

FIGURE 9.4 – Calculs 2D pour $Gr \in [1; 10^6]$, $Pr = 1$.



(a) Température moyenne en fonction de Gr (b) Temps de calcul en fonction de Gr (c) Erreur en fonction de Gr

FIGURE 9.5 – Calculs 3D pour $Gr \in [1; 10^6]$, $Pr = 1$.

2.3 Sortie du modèle

Pour finir, regardons la température moyenne sur le bord $\Gamma_3 = \bar{\Omega} \cap \{x = 1\}$ – voir la figure 9.1 pour la cavité 2D –

$$T_{\text{moy}} = \int_{\Gamma_3} T. \tag{9.25}$$

Cette quantité diminue lorsque le nombre de Grashof augmente car plus le nombre de Grashof est grand, plus la vitesse du fluide augmente et refroidi le bord Γ_3. Sur les figures 9.4(a) et 9.5(a), respectivement pour les cas 2D et 3D, la courbe en échelle logarithmique montre que les solutions obtenues via la méthode RBM ont le même comportement dans le cas élément fini. Ces résultats confirment donc la qualité de l'approximation base réduite pour les simulations 2D et 3D.

Résumé

Nous avons vu dans ce chapitre une application – à un problème de type convection naturelle – de la technique de projection proposée dans le chapitre 5. Cette technique peut être appliquée à tout type de problèmes ayant des non-linéarités quadratiques. Elle donne des résultats précis et efficaces. Nous avons constaté que plus le nombre de Grashof est élevé, plus cette efficacité est importante, que ce soit pour le cas 2D ou 3D. En terme de perspectives, il faudra s'intéresser à l'implémentation d'estimateurs d'erreur adaptés afin de pouvoir construire la base réduite à l'aide de l'algorithme glouton [121, 84, 123, 84].

Chapitre 10

Modélisation d'aimants résistifs à haut champ

Nous présentons dans ce chapitre l'utilisation du framework-RBM pour traiter des problèmes multi-physiques non-linéaires. Au cours de cette thèse, il a été effectué un travail en collaboration avec C.Daversin et C.Trophime, du Laboratoire National des Champs Magnétiques Intenses (LNCMI) sur la modélisation d'aimants résistifs à haut champ.

Commençons par définir cette notion de "champ intense" ou "haut champ". On parle de champ intense dès qu'un champ magnétique atteint 24 Tesla. À titre de comparaison, le champ magnétique terrestre est de 5.8×10^{-4} T. Les champs magnétiques intenses ont de nombreux domaines d'application, on peut citer par exemple la magnéto-science (étude des effets du champ magnétique sur les matériaux) ou l'étude des solides par résonance magnétique nucléaire – *Nuclear Magnetic Resonance (NMR)* –.

Ce chapitre reprend les résultats présentés dans [42] ainsi que de nouveaux résultats obtenus depuis.

La méthode des bases réduites est appliquée sans estimateur d'erreur, les paramètres utilisés pour construire la base réduite sont sélectionnés de manière aléatoire.

Après une présentation du LNCMI et des défis qu'il doit relever – nous verrons qu'étant données les ressources de calcul disponible, l'utilisation de méthode des bases réduites est nécessaire pour venir à bout de ces défis – nous introduirons l'approximation base réduite du problème traité. Enfin, nous présenterons les premiers résultats obtenus grâce au framework-RBM.

1 Le Laboratoire National des Champs Magnétiques Intenses

Le LNCMI est une unité propre de recherche (UPR3228) du Centre National de la Recherche Scientifique (CNRS). Il est affilié à l'Institut National des Sciences Appliquées (INSA) et l'Université Paul Sabatier (UPS) de Toulouse, ainsi que l'Université Joseph Fourier (UJF) de Grenoble. Il est qualifié d'Instrument de Recherche et permet à tout chercheur – venant du monde entier – de réaliser des expériences sous champ intense. Implanté sur deux sites, il offre des champs magnétiques statiques allant jusqu'à 36 Tesla sur son site de Grenoble, et des champs magnétiques pulsés allant jusqu'à 81 Tesla de manière

non-destructive (et 180 Tesla de manière semi-destructive) sur son site de Toulouse.

Dans la suite, nous ne nous intéresserons plus qu'aux champs magnétiques statiques mis en place sur le site de Grenoble.

1.1 Conception et optimisation des aimants

Les champs magnétiques sont obtenus en utilisant des aimants résistifs refroidis par eau – le débit atteint 300 litres par seconde – reliés à une alimentation de 24 MW [50]. Le LNCMI est en charge de la conception et de l'optimisation de ces aimants. Cela requiert la prédiction de quantités d'intérêt – ou métriques de performances – que l'on nommera sorties. Il s'agit par exemple de la valeur du champ magnétique au centre de l'aimant, la température moyenne et maximum ou encore la contrainte mécanique maximale.

Ces sorties sont exprimées comme fonctionnelles de champs inconnus associés à un ensemble d'équations couplées et paramétrées décrivant le comportement physique des aimants résistifs. Les paramètres d'entrées du modèle sont des variables de caractérisation tels que les propriétés physiques des matériaux, les coefficients de transfert thermique, la température de l'eau utilisée pour refroidir l'aimant ainsi que le débit associé, ou encore des variables géométriques pour les études d'optimisation. Pour évaluer ces relations *entrées-sorties*, les solutions de modèles multi-physiques impliquant de la thermo-électrique, magnétostatique, électro-thermo-mécanique ou thermo-hydraulique, sont requises. Il est à noter que les propriétés des matériaux utilisés dépendent de la température, par conséquent le modèle est non-linéaire. Dans la pratique, ces évaluations représentent un coût de calcul énorme mais sont indispensables pour améliorer la conception de l'aimant étant donné que nous ne pouvons plus compter sur le sens physique commun.

Deux technologies – Bitter et poly-hélices – sont développées au LNCMI pour les aimants résistifs. Typiquement, un aimant 24 MW consiste en un ensemble d'aimants poly-hélices et Bitter (ou "insert" dans la terminologie utilisée au LNCMI) alimenté séparément par 12 MW chacun. Les poly-hélices sont constituées de tubes en alliage de cuivre concentriques, dans lesquels les hélices ont été coupées par des techniques d'électroérosion, reliées électriquement en série. L'hélice coupée dans chaque tube peut être remplie avec de la colle ou laissée libre. Dans ce cas, des isolants sont introduits, périodiquement, pour éviter tout contact électrique entre les spires.

Le courant électrique appliqué dans chaque insert est de 30 kA. Cela engendre d'importantes pertes Joules à l'intérieur de l'insert. Un débit d'eau d'environ 300 litres par seconde est nécessaire pour refroidir l'insert et ainsi éviter que la température n'atteigne un seuil critique.

Nous considérons dans cette thèse uniquement l'aspect thermo-électrique. Notons T la température à l'intérieur de l'aimant, et u le potentiel électrique. Le modèle couplé électro-thermique est donné par

$$\begin{cases} -\nabla \cdot \Big(\sigma\big(T\big)\ \nabla u\Big) = 0, \\ -\nabla \cdot \Big(k\big(T\big)\ \nabla T\Big) = \sigma\big(T\big)\ \nabla u \cdot \nabla u, \end{cases} \tag{10.1}$$

où $\sigma\big(T\big)$ et $k\big(T\big)$ – qui dépendent de la température – sont respectivement la conductivité électrique et thermique du matériau. Cette dépendance en température est à l'origine de

FIGURE 10.1 – Un insert poly-hélices. Les hélices sont des tubes en alliage de cuivre découpés par une technique d'érosion. La colle époxy peut être introduite dans la fente pour assurer l'isolation électrique entre les spires : il s'agit d'hélices à refroidissement longitudinal, car le débit d'eau est longitudinal. Une autre possibilité pour assurer cette isolation isolation électrique entre les spires consiste à introduire périodiquement des isolants dans la fente : on parle alors d'hélices à refroidissement radial, puisque l'eau s'écoule à partir de l'intérieur vers l'extérieur de la fente ouverte.

la non-linéarité du modèle. En effet, nous avons

$$\sigma\big(T\big) = \frac{\sigma_0}{1+\alpha\big(T-T_0\big)} \qquad \text{et} \qquad k\big(T\big) = \sigma\big(T\big) L\ T, \tag{10.2}$$

où σ_0 est la conductivité électrique a $T = T_0$. α et L sont des caractéristiques du matériau, appelées respectivement ratio résistivité-température et nombre de Lorentz.

Dans un aimant de type poly-hélices (figure 10.1) les solénoïdes de cuivre sont reliés électriquement en série. La circulation du courant électrique dans chacun d'eux peut être modélisé comme une différence de potentiel électrique entre le courant d'entrée et de sortie. La valeur de cette différence peut être évaluée à partir de l'intensité du courant – connue à partir de résultats expérimentaux – et est intégrée dans le modèle en utilisant des conditions aux limites de type Dirichlet. Sur les autres bords, nous appliquons les conditions de Neumann homogènes car aucun courant ne circule sur les aimants en dehors des zones supérieures et inférieures,

$$\begin{cases} u = 0 \text{ sur l'entrée de courant } (c_{in}) \text{ et } u = u_D \text{ sur la sortie du courant } (c_{out}), \\ -\sigma\big(T\big)\ \nabla u \cdot \mathbf{n} = 0 \text{ sur les autres bords.} \end{cases} \tag{10.3}$$

On peut supposer qu'il n'y a pas d'échange thermique sur les surfaces dites "de connexion" (entrée et sortie de courant) ni sur les interfaces des isolants. L'échange thermique entre le cuivre et l'eau est modélisé par une condition de convection forcée (avec h étant le coefficient de transfert de chaleur, et Tw la température de l'eau),

$$\begin{cases} -k\big(T\big)\ \nabla T \cdot \mathbf{n} = h\ \big(T - T_w\big) \text{ sur les bords de refroidissement thermique,} \\ -k\big(T\big)\ \nabla T \cdot \mathbf{n} = 0 \text{ sur les autres bords,} \end{cases} \tag{10.4}$$

où h est obtenu en faisant la moyenne des corrélations thermohydrauliques classiques étant donné le débit de l'eau.

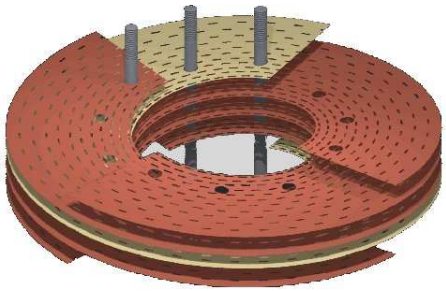


FIGURE 10.2 – Un insert Bitter consiste en un empilement de plaques en alliage de cuivre. Les isolants sont placés entre les plaques afin de créer un chemin de courant hélicoïdal. De petits trous sont percés afin de permettre à l'eau de s'écouler à travers l'aimant. Des tirants sont insérées pour assurer un bon contact électrique entre chaque plaque par application d'une pré-compression.

1.2 Défis à relever

Il existe une véritable course internationale entre les différents laboratoires de champs magnétiques intenses du monde pour fournir le champ magnétique le plus élevé. Dans cette course, le LNCMI pousse ses technologies d'aimants à leurs limites. La conception des inserts est principalement limitée par les contraintes thermiques – pour les parties intérieures – et par les contraintes mécaniques – pour les parties extérieures –. Deux solutions sont mises en place pour s'affranchir de ces problèmes. La première consiste à développer les hélices à refroidissement radial (voir le commentaire sous la figure 10.1) qui, de part leur construction, sont moins sensibles aux limites thermiques. La seconde consiste à chercher des matériaux avec des propriétés mécaniques améliorées. Notons que ce dernier aspect est une recherche à plus long terme.

Aujourd'hui, les projets du LNCMI impliquent l'utilisation d'hélices à refroidissement radial, ce qui permet de fournir des champs magnétiques intenses tout en brisant la limite associée à la contrainte thermique. Du point de vue de la conception cela requiert de pouvoir contrôler la température – moyenne ainsi que son écart-type – dans les aimants en fonction de plusieurs paramètres : ceux correspondant aux conditions de fonctionnement (le potentiel électrique appliqué u_D, ainsi que les paramètres liés aux conditions de refroidissement T_w,h), et ceux qui permettent de prendre en compte les incertitudes liées aux proprités des matériaux (la condutctivité électrique σ_0, ainsi que les coefficients α et L). En effet, les propriétés des matériaux ne sont pas précises. Par exemple, le cuivre acheté est fourni avec une borne supérieure et inférieure pour σ_0. Quant aux coefficients α and L, la littérature ne donne qu'un intervalle pour les définir.

Pour cela, le LNCMI a besoin d'une estimation rapide et fiable du champ de température. Il est possible de contrôler l'erreur a posteriori pour un champ T associé à un maillage anisotrope, obtenu à l'aide d'une stratégie d'adaptation de maillage [98]. Cependant, cette approche n'est ni suffisante ni efficace étant données les ressources de calcul requises. Une alternative intéressante est d'appliquer la méthodologie des bases réduites, couplée avec la méthode EIM pour retrouver une dépendance affine en paramètres provenant du second membre de (10.1) et (10.2) afin de réduire le coût de calcul. Cette nouvelle approche permet donc d'améliorer la conception des aimants car elle offre la possibilité de

réaliser des études paramétriques ainsi que des analyses de sensibilité à un coût de calcul raisonable.

2 Approximation base réduite du problème

Nous présentons maintenant l'approximation base réduite du problème. Dans la suite, on considère le vecteur de paramètres d'entrées $\boldsymbol{\mu}$ défini par

$$\boldsymbol{\mu} = \big(\sigma_0, \alpha, L, V_D, h, T_w\big) \in \mathcal{D} \subset \mathbb{R}^6. \tag{10.5}$$

Comme les propriétés du matériau rendent le modèle multi-physique paramétrisé non-affine en paramètres et non-linéaire, l'utilisation de la méthode des bases réduites dans ce contexte implique de procéder comme décrit dans la section 4 du chapitre 3. La sortie du modèle que nous considérons dans la suite est la valeur moyenne de la température $T(\boldsymbol{\mu})$. Nous avons donc

$$s\big(T(\boldsymbol{\mu})\big) = \frac{1}{|\Omega|}\int_\Omega T(\boldsymbol{\mu}), \tag{10.6}$$

où $|\Omega|$ désigne la surface du domaine d'étude.

Introduisons $X^u_{\mathcal{N}_u}$ et $X^T_{\mathcal{N}_T}$, deux espaces d'approximation FE, $\mathbb{P}_1$, de (grande) dimension $\mathcal{N}_u$ et $\mathcal{N}_T$. Soit $\boldsymbol{X}_\mathcal{N} = X^u_{\mathcal{N}_u} \times X^T_{\mathcal{N}_T}$ l'espace produit de dimension $\mathcal{N}$, avec $\mathcal{N} = \mathcal{N}_u + \mathcal{N}_T$. Notons que dans la suite nous avons $\mathcal{N}_u = \mathcal{N}_T$.

Nous avons fait le choix de traiter la condition de Dirichlet de manière faible en appliquant la méthode de Nitsche symétrique. Pour tout $\boldsymbol{\mu} \in \mathcal{D}$, les solutions $u_\mathcal{N}(\boldsymbol{\mu}) \in X^u_{\mathcal{N}_u}$ et $T(\boldsymbol{\mu}) \in X^T_{\mathcal{N}_T}$ du modèle couplé, défini dans la section 1.1 de ce chapitre, sont les racines respectives de $g_u(u(\boldsymbol{\mu}), \phi_u; \boldsymbol{\mu}) : X^u_{\mathcal{N}_u} \times X^u_{\mathcal{N}_u} \times \mathcal{D}$ et $g_T(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu}) : X^T_{\mathcal{N}_T} \times X^T_{\mathcal{N}_T} \times \mathcal{D}$, qui s'expriment de la manière suivante :

$$\begin{aligned} g_u\Big(u_\mathcal{N}(\boldsymbol{\mu}), \phi_u; \boldsymbol{\mu}\Big) = &\int_\Omega \sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)\nabla u_\mathcal{N}(\boldsymbol{\mu}) \cdot \nabla \phi_u - \int_{c_{in}\cup c_{out}} \sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)\Big(\nabla u_\mathcal{N}(\boldsymbol{\mu}) \cdot \mathbf{n}\Big)\phi_u \\ &+ \int_{c_{in}\cup c_{out}} \frac{\sigma\big(T\big)\gamma_D}{h_s} u_\mathcal{N}(\boldsymbol{\mu})\ \phi_u - \int_{c_{in}\cup c_{out}} \sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)(\nabla \phi_u \cdot \mathbf{n}) u_\mathcal{N}(\boldsymbol{\mu}) \\ &- \int_{c_{out}} \frac{\sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)\gamma_D}{h_s} u_D \phi_u + \int_{c_{out}} \sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)(\nabla \phi_u \cdot \mathbf{n}) u_D \end{aligned} \tag{10.7}$$

et

$$\begin{aligned} g_T\Big(T_\mathcal{N}(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu}\Big) = &\int_\Omega k\big(T_\mathcal{N}(\boldsymbol{\mu})\big)\nabla T_\mathcal{N}(\boldsymbol{\mu}) \cdot \nabla \phi_T + \int_{cooling} h\ T_\mathcal{N}(\boldsymbol{\mu})\ \phi_T \\ &- \int_\Omega \sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)\phi_T \nabla u_\mathcal{N}(\boldsymbol{\mu}) \cdot \nabla u_\mathcal{N}(\boldsymbol{\mu}) - \int_{cooling} h\ T_w\ \phi_T, \end{aligned} \tag{10.8}$$

où γ_D et h_s sont respectivement le coefficient de pénalisation – sa valeur est une constante indépendante de $\boldsymbol{\mu}$ – et la taille caractéristique du maillage. Notons que le terme $\frac{\gamma_D}{h_s}$ a été multiplié par $\sigma\big(T_\mathcal{N}(\boldsymbol{\mu})\big)$ afin de veiller à ce que le terme de pénalisation s'adapte correctement en fonction des autres termes sur le même bord. En faisant ça, il faut que le coefficient de pénalisation γ_D soit suffisamment grand ($\gamma_D \approx 30$) pour assurer la coercivité et faire en sorte que cette dernière ne dépende pas des paramètres.

Nous pouvons à présent utiliser l'algorithme de Newton pour résoudre

$$g\left(\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),T_{\mathcal{N}}(\boldsymbol{\mu})\Big),\Big(\phi_u,\phi_T\Big);\boldsymbol{\mu}\right)=g_u\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),\phi_u;\boldsymbol{\mu}\Big)+g_T\Big(T_{\mathcal{N}}(\boldsymbol{\mu}),\phi_T;\boldsymbol{\mu}\Big)=0. \quad (10.9)$$

Le résidu et la jacobienne s'écrivent respectivement

$$r\left(\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),T_{\mathcal{N}}(\boldsymbol{\mu})\Big),\Big(\phi_u,\phi_T\Big);\boldsymbol{\mu}\right)=g_u\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),\phi_u;\boldsymbol{\mu}\Big)+g_T\Big(T_{\mathcal{N}}(\boldsymbol{\mu}),\phi_T;\boldsymbol{\mu}\Big) \quad (10.10)$$

et

$$j\left(\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),T_{\mathcal{N}}(\boldsymbol{\mu})\Big),\Big(\phi_u,\phi_T\Big);\boldsymbol{\mu}\right)=\begin{pmatrix}\dfrac{\partial g_u\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),\phi_u;\boldsymbol{\mu}\Big)}{\partial u_{\mathcal{N}}(\boldsymbol{\mu})} & \dfrac{\partial g_u\Big(u_{\mathcal{N}}(\boldsymbol{\mu}),\phi_u;\boldsymbol{\mu}\Big)}{\partial T_{\mathcal{N}}(\boldsymbol{\mu})}\\[2ex] \dfrac{\partial g_T\Big(T_{\mathcal{N}}(\boldsymbol{\mu}),\phi_T;\boldsymbol{\mu}\Big)}{\partial u_{\mathcal{N}}(\boldsymbol{\mu})} & \dfrac{\partial g_T\Big(T_{\mathcal{N}}(\boldsymbol{\mu}),\phi_T;\boldsymbol{\mu}\Big)}{\partial T_{\mathcal{N}}(\boldsymbol{\mu})}\end{pmatrix}. \quad (10.11)$$

Rappelons que les termes σ et k, définis en (10.2), et qui apparaissent dans les expressions de g_u et g_T n'admettent pas une dépendance affine en paramètres. Par conséquent, nous devons appliquer la méthode EIM afin de construire, pour chacun de ces termes, une approximation affine en paramètres σ_{M_σ} et k_{M_k}. Il existe donc deux entiers positifs M_σ et M_k – déterminés par la méthode EIM – tels que nous puissions écrire

$$\sigma_{M_\sigma}\big(T_{\mathcal{N}}(\boldsymbol{\mu}),x;\boldsymbol{\mu}\big)=\sum_{m=1}^{M_\sigma}\beta_\sigma^m\big(T_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu}\big)\ \hat{\zeta}_\sigma^m(x) \quad (10.12)$$

et

$$k_{M_k}\big(T_{\mathcal{N}}(\boldsymbol{\mu}),x;\boldsymbol{\mu}\big)=\sum_{m=1}^{M_k}\beta_k^m\big(T_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu}\big)\ \hat{\zeta}_k^m(x), \quad (10.13)$$

où $\hat{\zeta}_\sigma^m$, pour $m=1,...,M_\sigma$ et $\hat{\zeta}_k^m$, pour $m=1,...,M_k$, sont respectivement les fonctions de base de l'expansion EIM des termes σ et k. Quant aux coefficients de l'expansion EIM, β_σ^m, pour $m=1,...,M_\sigma$ et β_k^m, pour $m=1,...,M_k$, ils sont solutions de (3.14).

Le produit scalaire $H^1(\Omega)$ – défini sur les espaces $X_{\mathcal{N}_u}^u$ et $X_{\mathcal{N}_T}^T$ – est utilisé pour les différentes projections sur l'espace d'approximation RB. Le code 10.1 illustre l'assemblage de la matrice associée au produit scalaire $H^1(\Omega)$ que l'utilisateur fournit au framework-RBM.

Listing 10.1 – Assemblage de la matrice associée au produit scalaire $H^1(\Omega)$.

```
//u et phi_u ∈ X^u_{N_u}, T et phi_T ∈ X^T_{N_T}.
//M : matrice associee au produit scalaire
M = M_backend->newMatrix( _test=Xh, _trial=Xh );
//produit scalaire H^1(Ω) defini sur X^u_{N_u}
form2( _test=Xh, _trial=Xh, _matrix=M ) =
  integrate( _range=elements( mesh ),
             _expr=id(phi_u)*idt(u) +
                   grad(phi_u)*trans( gradt(u) ) );
//produit scalaire H^1(Ω) defini sur X^u_{N_u}
```

```
form2( _test=Xh, _trial=Xh, _matrix=M ) +=
  integrate( _range=elements( mesh ),
             _expr=id(phi_T)*idt(T) +
                   grad(phi_T)*trans( gradt(T) ) );
```

3 Premiers résultats

Nous présentons maintenant les premiers résultats obtenus grâce au framework-RBM. Tout d'abord nous illustrerons la convergence de la méthode des bases réduites sur des géométries simples, puis nous présenterons une étude paramétrique réalisée sur un secteur de Bitter. Nous nous intéresserons ensuite à la quantification d'incertitudes. Enfin, nous évoquerons le gain, en terme de performance, lié à l'utilisation du framework-RBM.

La non-linéarité de ce modèle multi-physique est traitée à l'aide de l'algorithme de Picard. Un nombre maximal d'itération est fixé. Pour un $\boldsymbol{\mu} \in \mathcal{D}$ donné, l'algorithme de Picard peut s'interrompre parce que le nombre d'itérations maximal est atteint, et dans ce cas le critère d'erreur n'est pas satisfait. La solution peut donc perdre en précision. le critère d'erreur n'est pas satisfait. La solution numérique peut alors être très loin du résultat escompté.

Les résultats présentés sur un secteur d'aimant de type de Bitter ont été obtenus en faisant tourner le code sur 8 processeurs en utilisant le solveur `GMRES` et une méthode de type Schwarz additive – `GASM` – avec un seul niveau de recouvrement comme préconditionneur. Le maillage associé est anisotrope et comporte 9 700 tétraèdres, voir la figure 10.3(a). Le nombre maximal d'itérations de Picard a été fixé à 20.

Les résultats présentés sur un secteur d'aimant de type de hélice (figure 10.5) ont été obtenus en faisant tourner le code sur 15 processeurs en utilisant le solveur `GMRES` et le préconditionneur `GASM` avec un seul niveau de recouvrement. Le maillage associé comporte 28 000 tétraèdres. Le nombre maximal d'itérations de Picard a été fixé à 20.

Quant aux résultats présentés sur une hélice complète, le maillage, le nombre de processeurs et le nombre maximal d'itérations de Picard utilisés varient, ils seront donc précisés au cas par cas. En revanche le solveur utilisé est `GMRES` et le préconditionneur est `GASM` avec un seul niveau de recouvrement.

3.1 Convergence

On se propose de s'assurer de la convergence de la méthode base réduite. Nous considérons tout d'abord un secteur d'aimant de type Bitter puis un secteur d'aimant de type hélice.

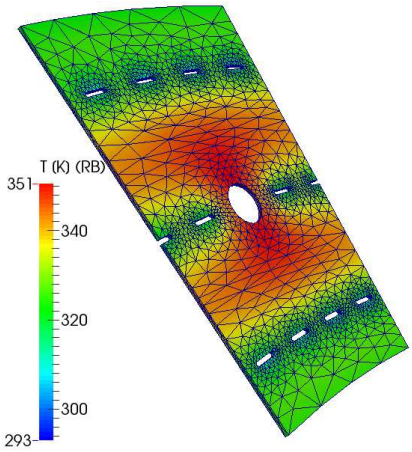
Secteur de Bitter

Nous commencons par étudier la convergence de la méthode EIM. Soit E une expression dont l'expansion EIM est notée E_{M_E}. E peut être par exemple σ,k ou $\sigma\nabla u \cdot \nabla u$. Soit Ξ_{train} un échantillon de 50 vecteurs de paramètres sélectionnés de manière aléatoire dans l'espace $\mathcal{D}$. Notons que nous avons $\Xi_{train} \bigcap S_N = \emptyset$, où S_N est l'ensemble des jeux de paramètres $\boldsymbol{\mu}$ ayant servi à construire la base réduite. Nous définissons alors le maximum de l'erreur relative commise par l'approximation EIM de E lorsque nous parcourons

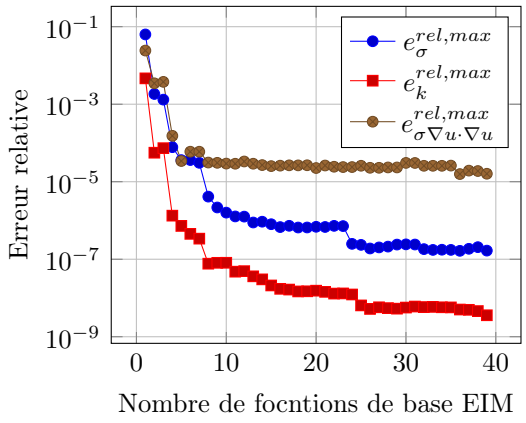
l'échantillon Ξ_{train} comme suit :

$$e_E^{rel,max} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|E - E_{M_E}\right|\right|_{L^2(\Omega)}}{\left|\left|E\right|\right|_{L^2(\Omega)}}. \tag{10.14}$$

Pour comprendre la formation du plateau observé sur la figure 10.3(b), il faut savoir qu'avec le framework-RBM, lorsque nous voulons approcher l'expression σ avec la méthode EIM, nous approchons en réalité π_σ, qui est la projection de σ sur l'espace de fonctions élément fini $X^u_{\mathcal{N}_u}$. Cela veut dire que les fonctions de base EIM sont calculées de manière à avoir l'expansion EIM de π_σ. À l'étape *en-ligne* de la méthodologie EIM, lors du calcul des coefficients de l'expansion (3.14), il est important d'évaluer, non pas l'expression σ, mais π_σ aux points d'interpolation. Dans le cas contraire nous observons alors le plateau montré par la figure 10.3(b).



(a) Visualisation d'une solution réduite (champ de température) sur un secteur d'aimant de type Bitter.



(b) Convergence de l'approximation EIM pour les expressions σ, k et $\sigma\nabla u \cdot \nabla u$.

FIGURE 10.3 – Champ de température sur un secteur de Bitter et convergence EIM pour différentes expressions.

Nous nous tournons maintenant vers la convergence de la méthode base réduite. Notons $T_N(\boldsymbol{\mu}) \in W_{N_{pr}}$, la solution réduite (champ de température) associée au vecteur de paramètres $\boldsymbol{\mu}$. Nous définissons alors les erreurs suivantes, pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$e_{N,T}^{rel,max} = \max_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|T_{\mathcal{N}}(\boldsymbol{\mu}) - T_N(\boldsymbol{\mu})\right|\right|_{L^2(\Omega)}}{\left|\left|T_N(\boldsymbol{\mu})\right|\right|_{L^2(\Omega)}}, \tag{10.15}$$

$$e_{N,T}^{rel,min} = \min_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|T_{\mathcal{N}}(\boldsymbol{\mu}) - T_N(\boldsymbol{\mu})\right|\right|_{L^2(\Omega)}}{\left|\left|T_N(\boldsymbol{\mu})\right|\right|_{L^2(\Omega)}}, \tag{10.16}$$

et

$$e_{N,T}^{rel,moy} = \frac{1}{Card(\Xi_{train})} \sum_{\boldsymbol{\mu}\in\Xi_{train}} \frac{\left|\left|T_{\mathcal{N}}(\boldsymbol{\mu}) - T_N(\boldsymbol{\mu})\right|\right|_{L^2(\Omega)}}{\left|\left|T_N(\boldsymbol{\mu})\right|\right|_{L^2(\Omega)}}. \tag{10.17}$$

Les quantités $e_{N,T}^{rel,max}$, $e_{N,T}^{rel,min}$ et $e_{N,T}^{rel,moy}$ désignent respectivement le maximum, le minimum et la moyenne de l'erreur relative commise sur la température lorsque nous parcourons l'échantillon Ξ_{train}.

D'après la figure 10.4, nous voyons que la convergence de la méthode des bases réduites sur un secteur d'aimant de type Bitter est assurée. Cependant, en regardant de plus près, bien qu'une suite d'espaces d'approximation RB emboîtés soit utilisé, nous constatons que la convergence n'est pas monotone. Cela peut provenir de la perte de précision de la solution obtenue par l'algorithme de Picard lorsque ce dernier atteint le nombre maximal d'itérations.

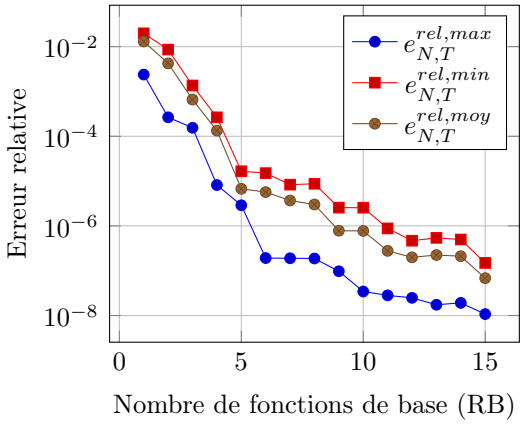


FIGURE 10.4 – Convergence de la méthode des bases réduites sur un secteur d'aimant de type Bitter.

Secteur d'hélice

Étudions la convergence de la méthode des bases réduites sur un secteur d'aimant de type hélice (voir la figure 10.5).

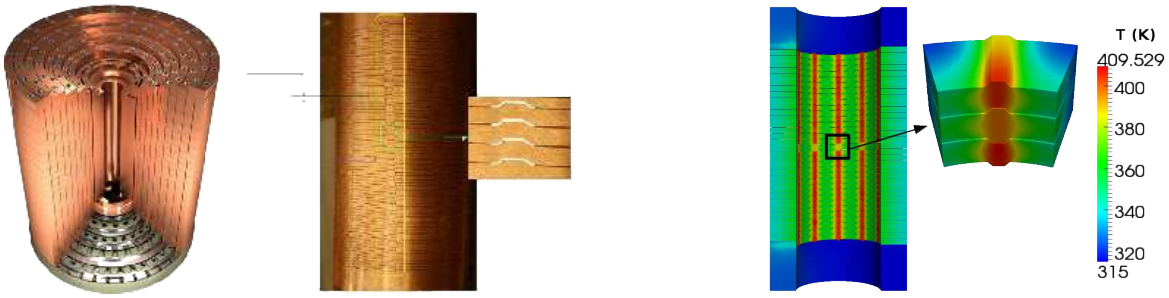


FIGURE 10.5 – À gauche : un exemple d'insert poly-hélices, suivi d'une vue détaillée de l'extérieur d'une hélice à refroidissement radial, avec un zoom sur les canaux de refroidissement et les isolants. À droite : Coupe transversale d'une hélice à refroidissement radial, champ de température – obtenu via l'utilisation du framework-RBM – de l'intérieur de l'hélice, avec un zoom sur les canaux de refroidissement et les isolants (il s'agit d'un secteur d'hélice).

D'après la figure 10.6, nous voyons que la convergence de la méthode des bases réduites sur un secteur d'aimant de type hélice est assurée. Là encore, nous constatons que la

convergence n'est pas monotone. Il s'agit du même phénomène que pour la convergence sur un secteur d'aimant de type Bitter.

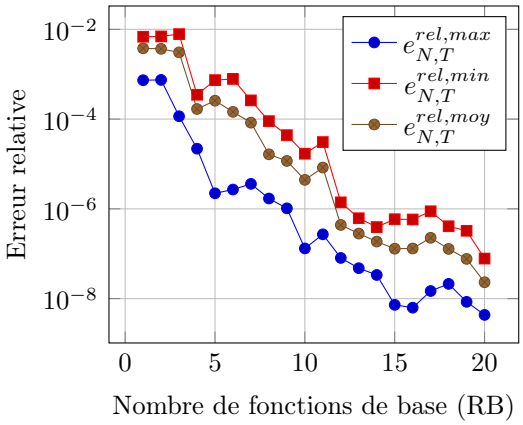


FIGURE 10.6 – Convergence de la méthode des bases réduites sur un secteur d'aimant de type hélice.

Notons que les quantités $e_{N,T}^{rel,max}$, $e_{N,T}^{rel,min}$ et $e_{N,T}^{rel,moy}$ sont les mêmes que celles définies en (10.15)-(10.17).

3.2 Étude paramétrique sur un secteur de Bitter

Aujourd'hui, le LNCMI est impliqué dans un projet de construction d'aimant hybride – *Hybrid Magnet Project* – qui consiste en l'assemblage d'une bobine résistive interne avec un supraconducteur externe. Cette technologie permet de générer le champ magnétique continu le plus élevé pour une installation de puissance électrique donnée (43 Tesla pour 24 MW en ce qui concerne le LNCMI). La bobine intérieure est une combinaison de deux technologies d'aimants (Bitter et poly-hélices) alimentés de manière indépendante [50] et [2].

Pour atteindre le champ magnétique demandé, l'aimant Bitter doit être repensé afin de générer plus de 9 Tesla ce qui correspond à la conception réelle. Ceci peut être obtenu en augmentant localement la densité de courant $j = \sigma(T)\nabla u$. En pratique, cela signifie modifier l'empilement de plaques de cuivre (voir le commentaire sous la figure 10.2). Ce changement entrainera également une augmentation locale la température.

L'étude paramétrique nous permet de quantifier de combien il est possible d'augmenter la densité de courant tout en conservant une température moyenne raisonnable afin de ne pas endommager l'aimant.

Du point de vue de l'ingénierie, pour ce genre d'aimants – composés de solénoïdes massifs, non bloqués – la densité de courant peut être approchée par

$$j(r) = \sigma_0 \left(\frac{u_D}{\theta\, r} \right), \tag{10.18}$$

où r est le rayon – en coordonnées cylindriques – et θ est l'angle du secteur de Bitter.

Le paramètre d'entrée u_D peut alors être choisi de telle sorte que la densité de courant $j(r_{int})$, sur le rayon intérieur r_{int} varie de 30×10^6 à 100×10^6 $A.m^{-2}$. Les autres paramètres sont fixés de la manière suivante : $\sigma_0 = 58 \times 10^6$ S, $\alpha = 3.5 \times 10^{-3}$ K^{-1}, $L = 2.5 \times 10^{-8}$, $h = 80\ 000$ $W \times m^{-2} \times K^{-1}$ et $T_w = 293$ K.

L'étude paramétrique a été effectuée avec la méthode classique élément fini, en exécutant les calculs pour un ensemble de paramètres donné, ainsi qu'avec la méthode base réduite, afin de valider l'utilisation du framework-RBM.

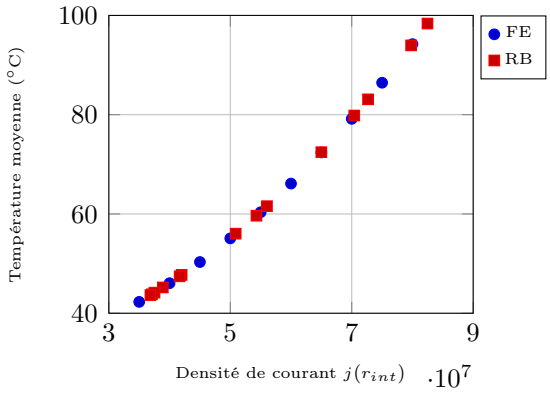


FIGURE 10.7 – Étude paramétrique sur un secteur de Bitter : température moyenne en fonction de la densité de courant.

D'un point de vue coût de calcul, l'utilisation du framework-RBM représente un gain de 2 ou 3 ordres de magnitude par rapport à l'utilisation du modèle classique élément fini.

Avec la conception actuelle, la température moyenne est d'environ $40°C$, or elle peut atteindre $60°C$ sans détériorer l'aimant. Atteindre $60°C$ permettrait d'augmenter la densité de courant d'un facteur 1.5 comme illustré par la figure 10.7. Cela permettrait de redéfinir l'empilement des plaques de cuivre pour atteindre en toute sécurité 10 Tesla, soit un gain de 1 Tesla. La nouvelle conception de l'aimant Bitter se fera sur la base de ce résultat.

3.3 Vers la quantification d'incertitudes

Comme mentionné précédemment, l'utilisation de la technique du refroidissement radial pour les hélices intérieures d'un insert poly-hélices permet d'appliquer une densité de courant plus élevée, et donc de fournir un champ magnétique plus élevé [50]. Cela est possible car les hélices à refroidissement radial sont, par construction, les hélices ayant le moins de limitations thermiques. Le débit de l'eau, qui s'écoule du rayon intérieur vers le rayon extérieur de l'hélice, est plus efficace pour refroidir.

Cependant, depuis que les isolants ont été introduits entre chaque spire (voir la figure 10.5), des points chauds sont apparus localement. Une estimation précise de la température est un élément clé pour la conception, surtout lorsque nous voulons que la température moyenne à l'intérieur de l'hélice reste sous un certain seuil critique. Prédire la température maximale atteinte au niveau des isolants nous permet de mieux les garder intacts. Pour cela, nous devons prendre en compte les incertitudes associées aux propriétés de l'alliage de cuivre (σ_0, α et L) ainsi que les incertitudes sur les paramètres de fonctionnement u_D et les conditions de refroidissement T_w, h. Une analyse de sensibilité permet de faire cela. Les plages de variation des paramètres d'entrée, indiquées dans le tableau 10.1, ont été choisies à partir de la littérature et de résultats expérimentaux.

L'analyse de sensibilité se fait en reliant le framework-RBM de FEEL++ avec la librairie `Openturns` (voir [45] et `http://www.openturns.org`) dédiée au traitement des incertitudes. Pour mener cette étude, `OpenTurns` construit un échantillon de paramètres d'entrée

Paramètre	Plage de variation
σ_0 [10^6 S]	[50 ; 50.2]
α [10^{-3}]	[3.3 ; 3.5]
L [10^{-8}]	[2.5 ; 2.9]
u_D [V]	[0.14 ; 0.15]
h [W K^{-1} m^{-2}]	[70000 ; 90000]
T_w [K]	[293 ; 313]

TABLE 10.1 – Plages de variation des paramètres d'entrée.

déterminé à partir d'une distribution de probabilités donnée. Le framework-RBM est ensuite utilisé pour évaluer les sorties du modèle associées, à partir desquelles la valeur moyenne et l'écart-type sont déduits par `OpenTurns`.

Comme nous le verrons dans la section 3.4, grâce au fait que le framework-RBM supporte les architectures parallèles, il est possible d'appliquer la méthode des bases réduites sur une géométrie aussi complexe que celle d'un aimant de type hélice. Cependant, dans le framework-RBM de FEEL++, rappelons que pour des problèmes non-linéaires, la partie *en-ligne* dépend de la (très grande) dimension élément fini. Le nombre de simulations nécessaires pour faire une analyse de sensibilité avec `OpenTurns` est si grand qu'il n'est pas possible de la faire avec les ressources de calcul disponibles. C'est pourquoi nous avons choisi d'effectuer l'analyse de sensibilité sur un secteur d'hélice (figure 10.5). Le secteur d'aimant de type d'hélice considéré est suffisamment représentatif car la température se comporte de manière régulière et symétrique.

Les résultats que nous présentons ont été obtenus à partir d'une distribution uniforme des paramètres d'entrées (les pages de variations sont données par le tableau 10.1), avec un échantillon de dimension 300. Notons que cette analyse de sensibilité a été réalisée sur différents échantillons de dimension de plus en plus grande. Nous avons alors remarqué qu'à partir d'un échantillon de dimension 300, la différence observée sur les quantités calculées devient négligeable. C'est pourquoi nous considérons qu'un échantillon de dimension 300 est suffisant pour obtenir des résultats satisfaisants en termes de précision.

Pour les paramètres d'entrée compris dans les pages de variations données par 10.1, le tableau 10.2 donne une valeur de référence pour la température moyenne, ainsi que l'écart-type qui peut être considéré comme un "intervalle de sécurité" associé à la valeur de référence précédente.

Valeur moyenne de la température [K]	368.66
Écart-type associé à la valeur moyenne de la température [K]	6.22

TABLE 10.2 – Valeur moyenne et écart-type.

Les valeurs données par 10.2 nous assure que la température moyenne dans l'aimant sera comprise dans l'interval $[362,44; 374,88]$ K.

La valeur maximale atteinte – la plus critique – peut être affinée via l'utilisation de quantiles. En effet, les quantiles donnent accès – pour une probablité γ donnée – au seuil $q(\gamma)$ que nous sommes assurés de ne pas dépasser. Considérons Y l'ensemble des températures moyennes obtenues à partir de l'échantillon des paramètres d'entrée, cela se

traduit par

$$\text{chercher } q(\gamma) \text{ tel que } \; P\Big(Y < q\big(\gamma\big)\Big) > \gamma. \tag{10.19}$$

Pour la même configuration que précédemment, le résultat quantiles pour $\gamma = 80\%$ et $\gamma = 99\%$ sont montrés par le tableau 10.3. Ces résultats sont importants pour le système

Probability (γ)	Quantile [K]
80%	371.297
99%	376.014

TABLE 10.3 – Quantiles

de contrôle de l'aimant. Il permet de détecter rapidement un comportement thermique suspect, et donc de mieux prévoir les incidents.

De plus, dans le but de simplifier et de mieux guider la campagne de mesures expérimentales, nous devons identifier les paramètres d'entrée qui sont les plus influents sur le comportement de la température. Cela est possible en se basant sur le calcul des indices de sensibilité. Les indices de Sobol d'ordre 1 qui figurent dans le tableau 10.8(gauche) ont été obtenus en utilisant un méta-modèle basé sur les polynômes de chaos sur l'échantillon précédent des paramètres d'entrée. Le graphique apparaissant sur la figure 10.8(droite) illustre la comparaison entre la température moyenne prédite par le framework-RBM et le méta-modèle. Cela nous permet de nous assurer que ce dernier corresponde bien au modèle de base.

Paramètre	Indice
σ_0	$6,85.10^{-5}$
α	$4,5.10^{-4}$
L	$9,2.10^{-3}$
V_D	$0,12$
h	$0,24$
T_w	$0,62$

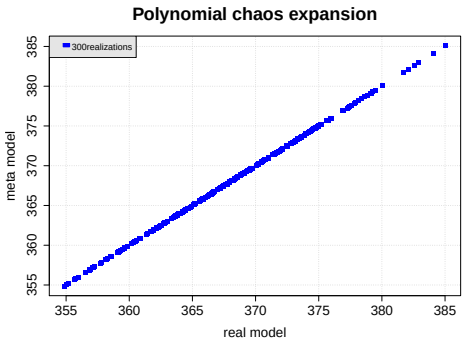


FIGURE 10.8 – À gauche : indices de Sobol. À droite : comparaison entre le modèle réel (RB) et le méta-modèle (polynômes du chaos).

Ces résultats montrent que les propriétés de l'alliage de cuivre ont bien moins d'influence sur la température moyenne que les paramètres de fonctionnement, spécialement ceux liés aux conditions de refroidissement. Par conséquent, le LNCMI doit se concentrer sur ces quantités en terme de qualité des mesures expérimentales, mais également au niveau de la compréhension du comportement de l'eau de refroidissement. Un tel investissement représenterait une réelle amélioration pour le modèle et apporterait plus de fiabilité.

3.4 Performances

Hélice complète

Considérons maintenant une hélice complète. L'espace d'approximation élément fini est composé de 5×10^5 degrés de liberté, répartis sur 15 processeurs de type Intel(R) Xeon(R) X5650 à 2.67 GHz. Le nombre maximal d'itérations de Picard est fixé à 20.

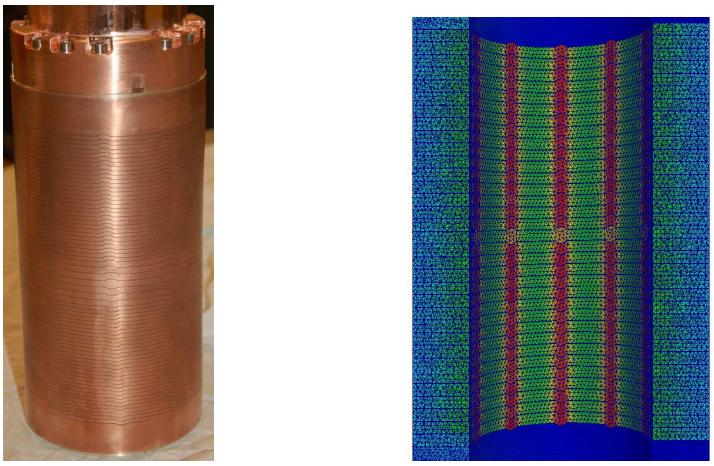


FIGURE 10.9 – À gauche : Vue extérieure d'une hélice à refroidissement radial. À droite : Coupe transversale d'une hélice à refroidissement radial, champ de température – obtenu via l'utilisation du framework-RBM – de l'intérieur de l'hélice avec le maillage associé.

20 fonctions de base ont été utilisées pour obtenir l'expansion EIM. La base réduite quant à elle est de dimension 10. Ainsi, l'erreur moyenne relative commise sur la température, définie en (10.17), est de l'ordre de 10^{-4}. Pour un jeu de paramètres donné, une simulation élément fini – pour obtenir la sortie du modèle – nécessite 45 minutes, contre 18 secondes via la partie *en-ligne* de la méthode des bases réduites (temps moyen calculé à partir de 20 évaluations). L'utilisation du framework-RBM permet de gagner un facteur 150 sur le temps de calcul d'une évaluation.

Si l'on considère à la fois le temps de calcul *hors-ligne* et *en-ligne*, la construction des bases EIM prend 2h30. La construction d'un élément de la base réduite nécessite 10 minutes.

En terme de temps de calcul, nous estimons que l'utilisation de la méthode des bases réduites devient intéressante – en comparaison avec la méthode des éléments finis – à partir de 6 évaluations. Cependant, puisque la partie *hors-ligne* nécessite autant de résolutions FE qu'il y a d'éléments dans la base réduite, l'utilisation de la méthode RB devient intéressante seulement après 10 évaluations. Cela est dû au temps – non négligeable – requis par le processus d'assemblage. Il est important de noter que, contrairement au modèle FE pour lequel les matrices sont assemblées à chaque simulation (aucun pré-calcul n'est possible), l'utilisation de la décomposition affine en paramètres par le framework-RBM permet d'assembler les matrices une seule fois au moment de l'étape *hors-ligne* quel que soit le nombre d'éléments dans la base réduite. Afin d'illustrer nos propos, on se propose de faire le même calcul, en comparant cette fois le modèle RB au modèle FE paramétrique. Le modèle FE paramétrique est le modèle FE qui utilise la décomposition affine en paramètres obtenue à l'aide des expansions EIM. Il est donc possible d'assembler

les matrices une seule fois. Cette comparaison confirme notre hypothèse puisque dans ce cas, l'utilisation du framework-RBM ne devient intéressante qu'au-delà de 13 évaluations.

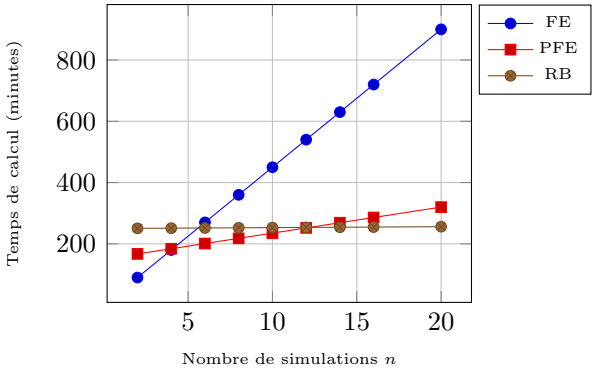


FIGURE 10.10 – Comparaison des temps de calcul (5×10^5 degrés de liberté).

La figure 10.10 illustre le temps observé sur plusieurs simulations n. Le temps FE correspond au temps nécessaire pour chaque évaluation (en prenant en compte le temps d'assemblage) du modèle FE, multiplié par n, soit 45 minutes $\times n$. Le modèle PFE quant à lui utilise la décomposition affine en paramètres donnée par EIM, ce qui permet d'assembler les matrices une seule fois. Le temps noté PFE est la somme du temps mis par l'algorithme EIM pour construire les différentes bases et du temps nécessaire pour une simlation, soit 8.5 minutes $\times n$. Enfin, le temps RB correspond à la somme du temps de construction de la base réduite (bases EIM comprises) et du temps mis pour une évaluation, soit 0.3 minutes $\times n$.

Prenons à présent un espace d'approximation élément fini composé de 2.4×10^6 degrés de liberté, répartis sur 16 processeurs de type Intel(R) Xeon(R) E5-2670 à 2.60 GHz. Le nombre maximal d'itérations de Picard est fixé à 6.

10 fonctions de base ont été utilisées pour obtenir l'expansion EIM, la base réduite quant à elle est de dimension 25. Pour un jeu de paramètres donné, une simulation élément fini – pour obtenir la sortie du modèle – nécessite 35 minutes, contre 2 minutes via la partie *en-ligne* de la méthode des bases réduites (temps moyen calculé à partir de 10 évaluations). L'utilisation du framework-RBM permet de gagner un facteur 17.5 sur le temps de calcul d'une évaluation.

Il est évident que le temps mis par la partie *en-ligne* de la méthode des bases réduites n'est pas satisfaisant. Cela est dû au fait que la partie *en-ligne* de EIM dépend encore de la (très grande) dimension élément fini, voir la remarque 16. Il reste donc encore du travail pour utiliser de manière efficace le framework-RBM sur une géométrie aussi complexe.

Résumé

Nous avons vu dans ce chapitre qu'il était possible d'utiliser le framework-RBM pour traiter des problèmes multi-physiques sur des géométries complexes. Le LNCMI a d'ores et déjà des premiers résultats pour une nouvelle conception d'aimants de type Bitter qui permettra de gagner 1 Tesla. Nous avons montré également, à l'aide d'une analyse de sensibilité, que les propriétés de l'alliage de cuivre ont bien moins d'influence sur la température moyenne que les paramètres de fonctionnement.

Conclusion et perspectives

Un des objectifs de cette thèse était de proposer une nouvelle construction des fonctions de base élément fini multi-échelles en s'appuyant sur les approximations base réduite, et dont le caractère intrinsèquement parallèle permettrait de profiter de la puissance des architectures hybrides. Cette nouvelle construction a été détaillée dans le chapitre 6, cependant sa mise en pratique nécessite l'utilisation d'un framework base réduite robuste.

Dans la première partie de ce manuscrit, après avoir rappelé quelques notions préliminaires, la méthode des bases réduites certifiées a été présentée dans le cas où des problèmes linéaires qui dépendent de manière affine des paramètres – elliptiques ou paraboliques – sont considérés. Nous avons pu vérifier que la convergence de la méthode RBM était bien celle attendue et que les estimateurs d'erreur fournissaient une borne supérieure raisonnable pour l'erreur. Nous avons également étudié deux propositions qui permettent d'optimiser le temps de calcul *hors-ligne* de certains termes nécessaires pour la construction des estimateurs d'erreur. Puis nous avons vu que lorsque cette dépendance affine était perdue, il était toujours possible de la retrouver, moyennant l'approximation de certains termes, à l'aide de la méthode EIM. L'approximation de ces termes conduit à une erreur supplémentaire qui doit être prise en compte par les estimateurs d'erreur. La qualité de l'approximation d'un terme non affine en paramètres via une expansion EIM a été vérifié, ainsi que l'approximation base réduite complète – incluant l'utilisation de la méthode EIM – dans le cas d'un problème linéaire non-affine. L'efficacité des estimateurs d'erreur a également été contrôlée. Ensuite, nous avons vu comment appliquer la méthode des bases réduites aux problèmes elliptiques multi-physiques, linéaires et non-linéaires – la méthodologie présentée peut facilement être étendue aux problèmes paraboliques –. Une technique de projection – ayant conduit à la publication [111] – permettant de résoudre les problèmes avec non-linéarités quadratiques uniquement sur un espace réduit a été présentée dans le chapitre 5. Enfin, la partie théorique a été clôturée par l'introduction, détaillée, d'une nouvelle construction des fonctions de base élément fini multi-échelles.

Dans la deuxième partie, orientée sur les développements informatiques, l'architecture logicielle du framework-RBM a été présentée. Au commencement de cette thèse, le framework-RBM était capable de calculer la sortie du modèle – basée sur les solutions réduites primale et duale – de problèmes elliptiques linéaires coercifs dans le cas mono-physique. Une implémentation des différents ingrédients nécessaires au calcul de l'estimation d'erreur a posteriori était également disponible. Il a donc fallu faire évoluer le framework afin de traiter les problèmes paraboliques linéaires et de prendre en compte les problèmes non-linéaires, multi-physiques. Cela implique évidemment d'implémenter l'algorithme EIM. Une autre contribution majeure a été d'offrir une vision fonctionnelle (à l'opposé d'une vision algébrique), il est désormais possible de manipuler de la même ma-

nière un élément de l'espace d'approximation FE ou RB. C'est notamment le cas grâce à la fonction `evaluateFromContext` dont l'utilisation dans Feel++ va au-delà du cadre des bases réduites. L'interface utilisateur, développée pour que l'utilisateur saisisse le moins d'informations et le plus simplement possible, a également été introduite. Ensuite, nous avons vu la stratégie mise en place pour que le framework-RBM puisse supporter les architectures parallèles. Cela implique la parallélisation de la méthode des bases réduites, mais également de la méthode EIM ou encore de la fonction `evaluateFromContext`.

La dernière partie quant à elle est consacrée aux simulations numériques réalisées à l'aide du framework-RBM. La technique de projection mentionnée précédemment pour appliquer la méthode des bases réduites aux problèmes ayant des non-linéarités quadratiques a été appliquée à un problème de type convection naturelle. Il a été montré que cette technique donnait des résultats précis et efficaces. Rappelons que ce travail est le fruit d'une collaboration avec E.Schenone. Il a ensuite été montré, lors d'une collaboration avec C.Daversin et C.Trophime du LNCMI, que le framework-RBM peut être utilisé pour appliquer la méthode RBM à des problèmes multi-physiques non-linéaires, et notamment pour mener des études paramétriques sur une portion d'aimant de type Bitter, ainsi que des analyses de sensibilités sur des secteurs d'aimant de type hélice. Ces études ont mis en évidence d'une part qu'il serait possible d'avoir une nouvelle conception des aimants de type Bitter qui permettrait de gagner 1 Tesla, et d'autre part que les propriétés de l'alliage de cuivre ont bien moins d'influence sur la température moyenne que les paramètres de fonctionnement.

De nombreuses perspectives se dégagent de ce travail. Tout d'abord, il faut poursuivre le travail commencé sur le traitement des problèmes ayant des non-linéarités quadratiques en l'étandant au cas transitoire, et en ajoutant l'implémentation d'estimateurs d'erreur adaptés afin de pouvoir construire la base réduite à l'aide de l'algorithme glouton [121, 84, 123, 84]. Il est possible que le framework-RBM soit amené à fournir des approximations de type "*hp*" [48, 49].

Il faut également continuer le travail commencé sur les lambda-expressions afin de ne plus dépendre de la dimension élément fini lors de l'étape *en-ligne* de l'algorithme EIM (voir la remarque 16). Ensuite, des estimateurs d'erreur – adaptés aux problèmes non-linéaires – pourront être utilisés afin d'optimiser le choix des éléments de la base réduite pour les problèmes du LNCMI. Du point de vue des applications, le modèle électro-thermique présenté dans le chapitre 10 devra être enrichi avec d'autres physiques telles que la magnétostatique ou l'élasticité. Le modèle devra également évoluer pour gérer des simulations transitoires. Les possibilités offertes par les simulations de type "légo" [120] associées à des techniques de décomposition de domaines sont très attractives pour ces grandes applications qui impliquent des simulations multi-physiques sur des géométries complexes.

La réflexion portant sur le développement d'une stratégie permettant de réduire le coût de calcul *hors-ligne* lié à la mise en oeuvre de l'algorithme EIM pour des problèmes non-linéaires multi-physiques complexes – voir la remarque 11 – devra être poursuivie.

Enfin, la nouvelle construction des fonctions de base élément fini multi-échelles proposée dans le chapitre 6 devra être implémentée dans Feel++. Pour cela, le framework-RBM a besoin de savoir gérer un espace de paramètres hybride, c'est-à-dire composé à la fois de paramètres dont la plage de variation doit être discrétisée (c'est le cas l'ensemble des paramètres considérés dans ce manuscrit) et de paramètres dont la plage de variation

est déjà discrétisée (comme c'est le cas lorsque nous considérons que les éléments d'un maillage sont des paramètres). Pour des performances optimales, le framework-RBM devra également pouvoir se servir des cartes graphiques pour la partie *en-ligne*.

Annexes

Chapitre 11

Étude de la complexité de la méthode RBM dans le cas de problèmes non-linéaires

Nous proposons dans cette annexe d'étudier la complexité de l'étape *en-ligne* de la méthode RBM, appliquée à un problème non-linéaire.

Considérons le problème de diffusion non-linéaire introduit dans [56]. Il s'agit d'un problème elliptique non-linéaire en 2D. Soit Ω le carré unité défini par $\Omega \equiv]0,1[^2 \in \mathbb{R}^2$. L'espace des paramètres est défini par $\mathcal{D} \equiv [0.01, 10]^2$. Considérons le problème suivant : chercher $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X_{\mathcal{N}} \equiv H_0^1(\Omega)$ tel que pour tout $\boldsymbol{\mu} \equiv (\mu^0, \mu^1) \in \mathcal{D}$ et pour tout $\boldsymbol{x} \equiv \left(x^0, x^1\right) \in \Omega$,

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v\Big) + \int_\Omega \zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) v = f\Big(v\Big), \ \forall v \in X_{\mathcal{N}}, \tag{11.1}$$

avec

$$a\Big(u_{\mathcal{N}}(\boldsymbol{\mu}), v\Big) = \int_\Omega \nabla u_{\mathcal{N}}(\boldsymbol{\mu}) \cdot \nabla v, \tag{11.2}$$

$$\zeta\Big(\boldsymbol{\mu}; \boldsymbol{x}; u_{\mathcal{N}}(\boldsymbol{\mu})\Big) = \mu^0 \ \frac{e^{\mu^1 u_{\mathcal{N}}(\boldsymbol{\mu})} - 1}{\mu^1}, \tag{11.3}$$

et

$$f\Big(v\Big) = 100 \int_\Omega sin\Big(2\pi x^0\Big) cos\Big(2\pi x^1\Big) \ v. \tag{11.4}$$

Nous sommes intéressés par l'évaluation de la sortie du modèle $s_{\mathcal{N}}(\boldsymbol{\mu}) = \ell(u_{\mathcal{N}}(\boldsymbol{\mu}))$, pour tout $\boldsymbol{\mu} \in \mathcal{D}$, avec

$$\ell\Big(v\Big) = \int_\Omega v. \tag{11.5}$$

Résolution en-ligne

Nous détaillons ici la construction de la matrice jacobienne et du vecteur résidu pour pouvoir résoudre (3.198) à chaque itération de l'algorithme de Newton. Nous étudierons également la complexité de l'étape *en-ligne* dans le cas de ce problème. Rappelons que

l'espace d'approximation $W_{N_{pr}}$ dans lequel est cherché la solution $u_N(\boldsymbol{\mu})$ est engendré par ξ_i^{pr}, $1 \leq i \leq N$. La solution réduite s'écrit, pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu})\ \xi_i^{pr}, \tag{11.6}$$

et la fonction non-linéaire $\zeta_M\big(\boldsymbol{\mu}; \boldsymbol{x}; u_N(\boldsymbol{\mu})\big)$ est approchée par

$$\zeta_M\Big(\boldsymbol{\mu}; \boldsymbol{x}; u_N(\boldsymbol{\mu})\Big) = \sum_{m=1}^{M} \beta^m\Big(\boldsymbol{\mu}; \boldsymbol{x}; u_N(\boldsymbol{\mu})\Big)\ \hat{\zeta}^m. \tag{11.7}$$

À l'étape *en-ligne* nous cherchons $u_N(\boldsymbol{\mu}) \in W_{N_{pr}}$ tel que pour tout $\boldsymbol{\mu} \in \mathcal{D}$,

$$a\Big(u_N(\boldsymbol{\mu}), v\Big) + \int_\Omega \zeta_M\Big(\boldsymbol{\mu}; \boldsymbol{x}; u_N(\boldsymbol{\mu})\Big) v = f\Big(v\Big),\ \forall v \in W_{N_{pr}}, \tag{11.8}$$

En nous servant de (11.6) et de (11.7), (11.8) peut se réécrire sous la forme

$$\sum_{i=1}^{N} \Big(A_N\Big)_{ji} u_{N_i}(\boldsymbol{\mu}) + \sum_{m=1}^{M} \Big(C_N\Big)_{jm} \beta^m\Big(\boldsymbol{\mu}; u_N(\boldsymbol{\mu})\Big) = \Big(F_N\Big)_j,\ j = 1, ..., N, \tag{11.9}$$

avec $A_N \in \mathbb{R}^{N\times N}$, $C_N \in \mathbb{R}^{N\times M}$ et $F_N \in \mathbb{R}^N$. Nous avons $\big(A_N\big)_{ji} = a\big(\xi_i^{pr}, \xi_j^{pr}\big)$ avec $1 \leq i, j \leq N$, $\big(C_N\big)_{jm} = \int_\Omega \hat{\zeta}^m \xi_j^{pr}$ avec $1 \leq j \leq N$ et $1 \leq m \leq M$, $\big(F_N\big)_j = f\Big(\xi_j^{pr}\Big)$, avec $1 \leq j \leq N$. De plus, les coefficients $\beta^m\big(\boldsymbol{\mu}; u_N(\boldsymbol{\mu})\big)$, $1 \leq m \leq M$, vérifient pour $j = 1, ..., M$ – voir (3.14) –,

$$\begin{aligned} \sum_{m=1}^{M} B_{jm}^M\ \beta^m\Big(\boldsymbol{\mu}; u_N(\boldsymbol{\mu})\Big) &= \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_m; u_N(\boldsymbol{\mu})\Big) \\ &= \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_m; \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu})\ \xi_i^{pr}\big(\boldsymbol{t}_m\big)\Big), \end{aligned} \tag{11.10}$$

où $\boldsymbol{t}_m$, $1 \leq m \leq M$ sont les points d'interpolation déterminés par la méthode EIM. Par conséquent, (11.9) peut également s'écrire, pour $j = 1, ..., N$,

$$\sum_{i=1}^{N} \Big(A_N\Big)_{ji} u_{N_i}(\boldsymbol{\mu}) + \sum_{m=1}^{M} \Big(D_N\Big)_{jm} \zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_m; \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu})\ \xi_i^{pr}\big(\boldsymbol{t}_m\big)\Big) = \Big(F_N\Big)_j, \tag{11.11}$$

avec $D_N \in \mathbb{R}^{N\times M}$ et nous avons $\big(D_N\big)_{jm} = C_N\big(B^M\big)^{-1}$. Afin de résoudre (11.11) nous appliquons l'algorithme de Newton. Considérons la $k^{\text{ème}}$ itération, nous cherchons l'incrément $\delta u_N^k(\boldsymbol{\mu}) \in W_{N_{pr}}$ tel que, pour $j = 1, ..., N$,

$$\sum_{i=1}^{N} \Big(J_N\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big)\Big)_{ji}\ \delta u_{N_j}^k(\boldsymbol{\mu}) = -G_N\Big(\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big)\Big)_j, \tag{11.12}$$

avec

$$G_N\Big(\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big)\Big)_j = \sum_{i=1}^{N}\Big(A_N\Big)_{ji} u_N^k(\boldsymbol{\mu}) + \sum_{m=1}^{M}\Big(D_N\Big)_{jm}\zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_m; \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu})\ \xi_i^{pr}\big(\boldsymbol{t}_m\big)\Big) - \Big(F_N\Big)_j,\ 1 \le j \le N, \tag{11.13}$$

ainsi que

$$\Big(J_N\big(\boldsymbol{\mu}; u_N^k(\boldsymbol{\mu})\big)\Big)_{ji} = \Big(A_N\Big)_{ji} + \Big(E_N\Big)_{ji},\ 1 \le i, j \le N, \tag{11.14}$$

où la matrice $E_N \in \mathbb{R}^{N\times N}$ doit être construite à chaque itération de la manière suivante :

$$\Big(E_N\Big)_{ji} = \sum_{m=1}^{M}\Big(D_N\Big)_{jm}\ \frac{\partial}{\partial u_{N_l}(\boldsymbol{\mu})}\zeta\Big(\boldsymbol{\mu}; \boldsymbol{t}_m; \sum_{n=1}^{N} u_{N_n}(\boldsymbol{\mu})\ \xi_n^{pr}\big(\boldsymbol{t}_m\big)\Big)\xi_i^{pr}\Big(\boldsymbol{t}_m\Big),\ 1 \le i, j, l \le N. \tag{11.15}$$

Enfin la sortie est évaluée comme suit

$$s_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{N_i}(\boldsymbol{\mu})\ \Big(L_N\Big)_j, \tag{11.16}$$

où $L_N \in \mathbb{R}^N$ et est défini par $\big(L_N\big)_i = \ell\big(\xi_i^{pr}\big)$, $1 \le i \le N$.

Étudions la complexité de l'étape *en-ligne* présentée ci-dessus. Le coût de l'application de l'algorithme de Newton est prédominant sur le reste des opérations. À chaque itération nous devons mettre à jour le résidu et construire E_N, ce qui necessite $\mathcal{O}(MN^2)$ opérations. La matrice jacobienne est ensuite mise à jour, puis le système (11.12) est résolu avec une complexité de $\mathcal{O}(N^3)$. La complexité de l'étape *en-ligne* ne dépend donc que des (petites) dimensions N et M, ainsi que du nombre d'itérations de l'algorithme de Newton. Elle ne dépend pas de la (grande) dimension $\mathcal{N}$.

Chapitre 12

Code complet du modèle correspondant au problème du bouclier thermique

Dans cette annexe est présenté le code complet nécessaire à la mise en place du modèle correspondant au problème du bouclier thermique introduit dans le chapitre 2, section 5. Avec ce code il est possible d'avoir un espace de fonctions élément fini d'ordre arbitraire grâce au template `Order`.

```
#include <feel/feel.hpp>
using namespace Feel;

template<int Order>
class BouclierThermique :
  public ModelCrbBase< // Derive de ModelCrbBase
    ParameterSpace<2>, //Espace de parametres de dimension 2
    // type de l'espace de fonctions FE
    decltype(Pch<Order>(Mesh<Simplex<2>>::New())) ,
    TimeDependent // le modele depend du temps
  >
{
public:

  typedef ModelCrbBase<
    ParameterSpace<2>,
    decltype(Pch<Order>(Mesh<Simplex<2>>::New())) ,
    TimeDependent > super_type;
  // Element de l'espace d'approximation FE
  typedef typename super_type::element_type element_type;
  // Parametre μ ∈ D
  typedef typename super_type::parameter_type parameter_type;
  // Objet utilise pour la discretisation en temps
  // BDF est l'acronyme de "Backward differentiation formula"
  typedef typename super_type::bdf_ptrtype bdf_ptrtype;

  void initModel();
  double output( int output_index, parameter_type const& mu,
                 element_type &u );
  //Retourne un pointeur sur l'objet de type bdf
  bdf_ptrtype bdfModel(){ return M_bdf; }

private:
    bdf_ptrtype M_bdf;
};
```

```
template<int Order>
void BouclierThermique<Order>::initModel()
{
  //Espace de fonctions FE et RB
  this->setFunctionSpaces(
          Pch<Order>(
             loadMesh( _mesh=new Mesh<Simplex<2> > )
          ) );

  //Acces au maillage
  auto mesh = this->Xh->mesh();

  //Informations sur l'espace de fonctions X_N
  //Seul le processeur maitre affiche les informations
  if( Environment::worldComm().isMasterRank() )
  {
    //Nombre de degres de liberte sur le processeur maitre
    std::cout << "Number of local dof " <<this->Xh->nLocalDof() ;
    //Nombre de degres de liberte global
    std::cout << " and number of dof " <<this->Xh->nDof()<<"\n";
  }

  //Calcul de la surface totale du maillage
  double surface = integrate( _range=elements( mesh ),
                              _expr=cst( 1. ) ).evaluate();

  //Constructeur de l'objet de type bdf
  M_bdf = bdf( _space=this->Xh )

  //Definition de l'espace des parametres
  auto mu_min = this->Dmu->element();
  auto mu_max = this->Dmu->element();
  mu_min << 1e-2 , 1e-3; mu_max << 0.5 , 0.1;
  this->Dmu->setMin( mu_min ); this->Dmu->setMax( mu_max );

  //Elements de l'espace de fonctions X_N
  auto u = this->Xh->element(); auto v = this->Xh->element();

  //Les bords marques "gamma_holes" correspondent a ∂Ω_int sur la
  //figure 2.10. Quant au bord "left" il correspond a ∂Ω_ext sur
  //la figure 2.10

  //Formes bilineaires provenant de la decomposition affine de a
  auto a0 = form2( _trial=this->Xh, _test=this->Xh);
  a0 = integrate( _range= elements( mesh ),
                  _expr= gradt( u )*trans( grad( v ) ) );
  this->addLhs( { a0 , "1" } );
  auto a1 = form2( _trial=this->Xh, _test=this->Xh);
  a1 = integrate( _range= markedfaces( mesh, "left" ),
                  _expr= idt( u )*id( v ) );
  this->addLhs( { a1 , "mu0" } );
  auto a2 = form2( _trial=this->Xh, _test=this->Xh);
  a2 = integrate( _range= markedfaces( mesh, "gamma_holes" ),
                  _expr= idt( u )*id( v ) );
  this->addLhs( { a2 , "mu1" } );

  //Contribution de la matrice de masse
```

```
  auto mass = form2( _trial=this->Xh, _test=this->Xh);
  mass = integrate ( _range=elements( mesh ),
                     _expr=idt( u )*id( v ) );
  this->addMass( { mass , "1" } );

  //Second membre
  auto f0 = form1( _test=this->Xh );
  f0 = integrate( _range=markedfaces( mesh,"left" ),
                  _expr= id( v ) ) ;
  this->addRhs( { f0, "mu0" } );

  //Sortie : ℓ(v) = ∫_Ω 1/|Ω| v
  auto out = form1( _test=this->Xh );
  out = integrate( _range=elements( mesh ),
                   _expr= (1./surface)*id( v ) ) ;
  this->addOutput( { out, "1" } );

  //Specification du produit scalaire pour un parametre de reference
  //ici μ_ref = (0.01, 0.001). Nous avons choisi le produit
  //scalaire associe a la norme d'energie
  auto energy = form2( _trial=this->Xh, _test=this->Xh);
  energy =
     integrate( _range=elements( mesh ),
                _expr=gradt( u )*trans( grad( v ) ) ) +
     integrate( _range= markedfaces( mesh, "left" ),
                _expr= 0.01 * idt( u )*id( v ) ) +
     integrate( _range= markedfaces( mesh, "gamma_holes" ),
                _expr= 0.001 * idt( u )*id( v ) );

  this->addEnergyMatrix( energy );
  this->addMassMatrix(mass);
}

//Calcul des sorties du modele pour un element u de X_N
template<int Order>
double BouclierThermique<Order>::output(int output_index,
                                        parameter_type const& mu,
                                        element_type &u )
{
    auto mesh = this->Xh->mesh();
    double output=0;
    if ( output_index==0 )
        output=integrate( _range=markedfaces( mesh,"left" ),
                          _expr= mu(0)*idv( u )).evaluate();
    if( output_index==1 )
        output=integrate( _range=elements( mesh ),
                          _expr= (1./surface)*idv( u )).evaluate();
    if( output_index > 1 )
        throw std::logic_error( "Error with output_index" );
    return output ;
}
```

Chapitre 13

Différentes utilisations de la fonction evaluateFromContext

Dans la section 1.3 du chapitre 7 nous avons illustré comment évaluer un élément de l'espace de fonctions FE ou RB – en un certain nombre de points – via la fonction `evaluateFromContext`. Nous montrons ici que nous pouvons également avoir accès au gradient ou aux dérivées partielles suivant les variables d'espace. De plus nous montrons qu'il est également possible de manipuler des éléments d'espaces vectoriels. Afin de vérifier les résultats obtenus nous utilisons les outils dédiés à la vérification de la librairie `Boost` comme `BOOST_CHECK`.

```
auto mesh=unitHypercube<2>(); //carre unite
//Espace d'approximation (vectoriel) FE, X_N
auto Xh = Pchv<Order>( mesh );
//Reduced basis space
auto RbSpace = RbSpacePch( this->shared_from_this() );
//Projection de (x , x^2 ) sur X_N
auto basis_1 = vf::project( Xh , elements(mesh),
                            vec( Px() , Px()*Px() ) );
//Projection de (y , x y ) sur X_N
auto basis_2 = vf::project( Xh , elements(mesh),
                            vec( Py() , Py()*Px() ) );

//Ces deux elements engendrent la base reduite
RbSpace->addPrimalBasisElement( basis_1 );
RbSpace->addPrimalBasisElement( basis_2 );

//Points que l'on va associer au context
node_type t1(2), t2(2), t3(2);
/*x*/ t1(0)=0.1;  /*y*/ t1(1)=0.2;
/*x*/ t2(0)=0.1;  /*y*/ t2(1)=0.8;
/*x*/ t3(0)=0.75; /*y*/ t3(1)=0.9;

//Creation des context FE et RB
auto ctxfem = Xh->context(); auto ctxrb = RbSpace->context();
// Ajout des points aux contexts
ctxfem.add( t1 ); ctxfem.add( t2 ); ctxfem.add( t3 );
ctxrb.add( t1 ); ctxrb.add( t2 ); ctxrb.add( t3 );

//Element de l'espace d'approximation RB
auto u_rb = RbSpace->element();
u_rb.setCoefficient( 1 , 1 );//u_rb = (0, 1)
```

```
auto u_fem = u_rb.expansion();
//les elements manipules sont (y, x y)

int x=0,y=1; //variables d'espace

// ------- Evaluations -------
//Pour les evaluations, evaluateFromContext retourne un vecteur qui,
//pour chaque point, contient l'evaluation de l'expression en
//chaque composante, on a :
// auto res=evaluateFromContext(...,_expr=idv()) avec
// res(0), res(1) : evaluation au premier point
// res(2), res(3) : evaluation au dexieme point ...

//Element : u_fem, context : ctxfem
auto fem_eval_ctxfem =
    evaluateFromContext( _context=ctxfem, _expr=idv(u_fem) );
//Element : u_rb, context : ctxfem
auto rb_eval_from_ctxrb =
    evaluateFromContext( _context=ctxfem, _expr=idv(u_rb) );
//Element : u_rb, context : ctxrb
auto rb_eval_ctxfem =
    evaluateFromContext( _context=ctxrb, _expr=idv(u_rb) );

// Verification
Eigen::VectorXd true_values( 6 ); // evaluations exactes
true_values(0)=t1(y); true_values(1)=t1(y)*t1(x);
true_values(2)=t2(y); true_values(3)=t2(y)*t2(x);
true_values(4)=t3(y); true_values(5)=t3(y)*t3(x);

double norm_fem_eval_ctxfem = fem_eval_ctxfem.norm();
double norm_rb_eval_ctxrb = rb_eval_ctxrb.norm();
double norm_rb_eval_ctxfem = rb_eval_ctxfem.norm();
double true_norm=true_values.norm();
double tol=1e-13; //tolerance
double diff=math::abs(norm_rb_eval_ctxrb - true_norm);
//La difference entre les normes calculees doit etre plus petite
//que la tolerance "tol"
BOOST_CHECK_SMALL(diff, 1e-13 );
diff=math::abs(norm_rb_eval_ctxrb - norm_rb_eval_ctxfem);
BOOST_CHECK_SMALL(diff, 1e-13 );
diff = math::abs(norm_rb_eval_ctxrb - norm_fem_eval_ctxfem)
BOOST_CHECK_SMALL( diff, 1e-13 );

// ------- Gradients -------
//Pour les gradient, evaluateFromContext retourne un vecteur qui,
//pour chaque point, pour chaque variable d'espace, contient la
//derivee partielle de chaque composante , en posant u=(u1,u2)
//on a :
// auto res=evaluateFromContext(...,_expr=gradv()) avec
// res(0) : evaluation de du1/dx au premier point
// res(1) : evaluation de du2/dx au premier point
// res(2) : evaluation de du1/dy au premier point
// res(3) : evaluation de du2/dy au premier point
// res(4) : evaluation de du1/dx au deuxieme point

//Element : u_fem, context : ctxfem
auto fem_grad_ctxfem =
    evaluateFromContext( _context=ctxfem, _expr=gradv(u_fem) );
```

```
auto rb_grad_ctxfem =
    evaluateFromContext( _context=ctxfem, _expr=gradv(u_rb) );
auto rb_grad_ctxrb  =
    evaluateFromContext( _context=ctxrb, _expr=gradv(u_rb) );

// Verification
Eigen::VectorXd true_valuesg( 12 )
/*du1/dx*/true_g( 0 ) = 0; /*du2/dx*/true_g( 1 ) = t1(y);
/*du1/dy*/true_g( 2 ) = 1; /*du2/dy*/true_g( 3 ) = t1(x);
/*du1/dx*/true_g( 4 ) = 0; /*du2/dx*/true_g( 5 ) = t2(y);
/*du1/dy*/true_g( 6 ) = 1; /*du2/dy*/true_g( 7 ) = t2(x);
/*du1/dx*/true_g( 8 ) = 0; /*du2/dx*/true_g( 9 ) = t3(y);
/*du1/dy*/true_g( 10 ) = 1; /*du2/dy*/true_g( 11 ) = t3(x);

double norm_fem_grad_ctxfem = fem_grad_ctxfem.norm();
double norm_rb_grad_ctxrb = rb_grad_ctxrb.norm();
double norm_rb_grad_ctxfem = rb_grad_ctxfem.norm();
double trueg_norm = true_g.norm();
diff = math::abs(norm_fem_grad_ctxfem - trueg_norm)
BOOST_CHECK_SMALL( diff, tol );
diff = math::abs(norm_rb_grad_ctxrb - trueg_grad)
BOOST_CHECK_SMALL( diff, tol );
diff = math::abs(norm_rb_grad_ctxrb - norm_rb_grad_ctxfem)
BOOST_CHECK_SMALL( diff, tol );

//Notons qu'il est egalement possible de tester chaque
//composante du vecteur resultat, par exemple :
diff = math::abs(rb_grad_ctxrb(0) - true_g(0) );
BOOST_CHECK_SMALL( diff, tol );

// ------- Derivees partielles -------
//Il est egalement possible d'avoir l'evaluation des
//derivees partielles. Cela retourne un vecteur contenant,
//pour chaque point, contient l'evaluation de l'expression en
//chaque composante. Comme pour les evaluations.
auto dy_fem_ctxfem=evaluateFromContext( _context=ctxfem,
                                        _expr=dyv(u_fem) );
auto dy_rb_ctxrb=evaluateFromContext( _context=ctxrb,
                                      _expr=dyv(u_rb) );
auto dy_rb_ctxrb=evaluateFromContext( _context=ctxrb,
                                      _expr=dyv(u_rb) );
auto dy_rb_ctxfem=evaluateFromContext( _context=ctxfem,
                                       _expr=dyv(u_rb) );
```

Bibliographie

[1] A. Abdulle, E. Weinan, B. Engquist, and E. Vanden-Eijnden. The heterogeneous multiscale method. *Acta Numerica*, 21 :1–87, 2012.

[2] A.Bourquard, D.Bresson, A.Daël, F.Debray, P.Fazilleau, B.Hervieu, W.Joss, F.P.Juster, C.Mayri, P.Pugnat, J.M.Rifflet, L.Ronayette, C.Trophime, C.Verwaerde, and L.Villars. The 42+ t hybrid magnet project at cnrs-lncmi-grenoble. *Journal of Low Temperature Physics*, 159 :332–335, April 2010.

[3] G. Allaire. A brief introduction to homogenization and miscellaneous applications. In *ESAIM : Proceedings*, volume 37, pages 1–49. EDP Sciences, 2012.

[4] B.O. Almroth, P. Stern, and F.A. Brogan. Automatic choice of global shape functions in structural analysis. *AIAA Journal*, 16(5) :525–528, 1978.

[5] Scott W Ambler. *The Elements of UML (TM) 2.0 Style*. Cambridge University Press, 2005.

[6] P.R. Amestoy, I.S. Duff, J.Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1) :15–41, 2001.

[7] P.R. Amestoy, A. Guermouche, J.Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2) :136–156, 2006.

[8] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics*, 139(3) :153–176, 2006.

[9] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids : Part ii : Transient simulation using space-time separated representations. *Journal of Non-Newtonian Fluid Mechanics*, 144(2) :98–121, 2007.

[10] C. Baiocchi, F. Brezzi, and L. P. Franca. Virtual bubbles and galerkin-least-squares type methods (ga. ls). *Computer Methods in Applied Mechanics and Engineering*, 105(1) :125–141, 1993.

[11] N. Bakhvalov and G Panasenko. *Homogenisation averaging processes in periodic media*. Springer, 1989.

[12] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, B. F. Smith, and H. Zhang. *PETSc Users Manual*, 2012.

[13] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, B. F. Smith, and H. Zhang. *PETSc Web page*, 2012.

[14] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith. *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*, 1997.

[15] E. Balmès. Parametric families of reduced finite element models. theory and applications. *Mechanical Systems and Signal Processing*, 10(4) :381–394, 1996.

[16] M. Barrault, Y. Maday, N. C. Nguyen, and A.T. Patera. An empirical interpolation method : application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9) :667–672, 2004.

[17] A. Barrett and G. Reddien. On the reduced basis method. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 75(7) :543–549, 1995.

[18] C. Bauer, A. Frink, and R. Kreckel. Introduction to the ginac framework for symbolic computation within the c++ programming language. *Journal of Symbolic Computation*, 33(1) :1–12, 2002.

[19] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3) :1457–1472, 2011.

[20] C++ Boost. Libraries, 2008. http ://www.boost.org.

[21] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138) :333–390, 1977.

[22] F. Brezzi, L.P. Franca, T.J.R. Hughes, and A. Russo. B= $\int$ g. *Computer Methods in Applied Mechanics and Engineering*, 145 :329–339, 1997.

[23] F. Brezzi and L.D. Marini. Augmented spaces, two-level methods, and stabilizing subgrids. *International journal for numerical methods in fluids*, 40(1-2) :31–46, 2002.

[24] F. Brezzi and A. Russo. Choosing bubbles for advection-diffusion problems. *Mathematical Models and Methods in Applied Sciences*, 4(04) :571–587, 1994.

[25] F.(Franco) Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, 1991.

[26] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*. Siam, 2000.

[27] A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme, and G. Turinici. A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM : Mathematical Modelling and Numerical Analysis*, 46(03) :595–603, 2012.

[28] C. Canuto, MY Hussaini, A. Quarteroni, and TA Zang. *Spectral methods*. Springer, 2006.

[29] C. Canuto, T. Tonn, and K. Urban. A posteriori error analysis of the reduced basis method for nonaffine parametrized nonlinear pdes. *SIAM Journal on Numerical Analysis*, 47(3) :2001–2022, 2009.

[30] F. Casenave. Accurate a posteriori error evaluation in the reduced basis method. *Comptes Rendus Mathematique*, 350(9) :539–542, 2012.

[31] F. Casenave, A. Ern, and T. Lelièvre. Accurate and efficient evaluation of the a posteriori error estimator in the reduced basis method. *arXiv preprint arXiv :1212.0970*, 2012.

[32] V. Chabannes. *Vers la simulation des écoulements sanguins*. PhD thesis, Université Joseph Fourier, Grenoble, july 2013.

[33] R. Chakir and Y. Maday. Une méthode combinée d'éléments finis à deux grilles/-bases réduites pour l'approximation des solutions d'une edp paramétrique. *Comptes Rendus Mathematique*, 347(7) :435–440, 2009.

[34] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta numerica*, 3 :61–143, 1994.

[35] Y. Chen, J. S Hesthaven, Y. Maday, and J. Rodríguez. Improved successive constraint method based a posteriori error estimate for reduced basis approximation of 2d maxwell s problem. 2008.

[36] F Chinesta, A Ammar, A Leygue, and Roland Keunings. An overview of the proper generalized decomposition with applications in computational rheology. *Journal of Non-Newtonian Fluid Mechanics*, 166(11) :578–592, 2011.

[37] P.G. Ciarlet. The finite element method for elliptic problems, vol. 40 of classics in applied mathematics, society for industrial and applied mathematics (siam), philadelphia, pa, 2002. reprint of the 1978 original. *Reprint of the 1978 original*, 1(1) :2–4.

[38] D. Cioranescu and P. Donato. An introduction to homogenization, volume 17 of oxford lecture series in mathematics and its applications. *The Clarendon Press Oxford University Press, New York*, 4 :118, 1999.

[39] N. N. Cuong, K. Veroy, and A. T. Patera. Certified real-time solution of parametrized partial differential equations. In *Handbook of Materials Modeling*, pages 1529–1564. Springer, 2005.

[40] C. Daversin, C. Prud'homme, C. Trophime, and S. Veys. Applications of reduced order modeling to high magnetic field resistive magnets developments. In *Conference Record of the 23th International Conference on Magnet Technology*, 2013.

[41] C. Daversin, C. Prud'homme, C. Trophime, and S. Veys. Reduced order modeling of high magnetic field magnets. In *Conference Record of the 9th International Symposium of Electric and Magnetic Fields*, 2013.

[42] C. Daversin, S. Veys, C. Trophime, and C. Prud'Homme. A reduced basis framework : Application to large scale nonlinear multi-physics problems. *Esaim Proc.*, 43 :225–254, December 2013.

[43] S. Deparis and G. Rozza. Reduced basis method for multi-parameter-dependent steady navier–stokes equations : applications to natural convection in a cavity. *Journal of Computational Physics*, 228(12) :4359–4378, 2009.

[44] G. Dollé. *Diffuse Optical Tomography for Tumour Detection*. PhD thesis, Université de Strasbourg, 2016.

[45] A. Dutfoy, I. Dutka-Malen, A. Pasanisi, R. Lebrun, F. Mangeant, J. Sen Gupta, M. Pendola, and T. Yalamas. OpenTURNS, an Open Source initiative to Treat Uncertainties, Risks'N Statistics in a structured industrial approach. In *41èmes Journées de Statistique, SFdS, Bordeaux (France)*, 2009.

[46] Y. R. Efendiev, T. Y. Hou, and X.H. Wu. Convergence of a nonconforming multiscale finite element method. *SIAM Journal on Numerical Analysis*, 37(3) :888–910, 2000.

[47] J. L. Eftang, M. A. Grepl, and A. T. Patera. *A posteriori* error bounds for the empirical interpolation method. *Comptes Rendus Mathematique*, 348(9) :575–579, 2010.

[48] J. L. Eftang, D. J. Knezevic, and A. T. Patera. An hp certified reduced basis method for parametrized parabolic partial differential equations. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4) :395–422, 2011.

[49] J.L. Eftang, D.B.P. Huynh, D.J. Knezevic, and A.T. Patera. A two-step certified reduced basis method. *Journal of Scientific Computing*, 51 :28–58, 2012.

[50] F.Debray, J.Dumas, C.Trophime, and N.Vidal. Dc high field magnets at the lncmi. *IEEE transactions on applied superconductivity,*, 22(3), June 2012.

[51] J.P. Fink and W.C. Rheinboldt. On the error behavior of the reduced basis technique for nonlinear finite element approximations. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 63(1) :21–28, 1983.

[52] L. P. Franca and A. Russo. Deriving upwinding, mass lumping and selective reduced integration by residual-free bubbles. *Applied mathematics letters*, 9(5) :83–88, 1996.

[53] M. A. Grepl. *Reduced-basis approximation a posteriori error estimation for parabolic partial differential equations*. PhD thesis, Massachusetts Institute of Technology, 2005.

[54] M. A. Grepl, N. C. Nguyen, K. Veroy, A. T. Patera, and G. R. Liu. Certified rapid solution of partial differential equations for real-time parameter estimation and optimization. In *Proceedings of the 2nd Sandia workshop of PDE-constrained optimization : Real-time PDE-constrained optimization. SIAM computational science and engineering book series. SIAM, Philadelphia*, pages 197–216, 2007.

[55] M. A. Grepl and A. T. Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM : Mathematical Modelling and Numerical Analysis*, 39(01) :157–181, 2005.

[56] M.A. Grepl, Y. Maday, N. C. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM : Mathematical Modelling and Numerical Analysis*, 41(03) :575–605, 2007.

[57] G. Guennebaud, B. Jacob, et al. Eigen v3. http ://eigen.tuxfamily.org, 2010.

[58] B. Haasdonk. Convergence rates of the pod-greedy method. *ESAIM : Mathematical Modelling and Numerical Analysis*, 1(1), 2012.

[59] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM : Mathematical Modelling and Numerical Analysis*, 42(02) :277–302, 2008.

[60] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer-Verlag Berlin, 1985.

[61] J. Hadamard. *Le probleme de Cauchy et les équations aux dérivées partielles linéaires hyperboliques*, volume 193. Paris, 1932.

[62] B. Hendrickson and R. Leland. The chaco user's guide version 2.0. Technical report, Technical Report SAND95-2344, Sandia National Laboratories, 1995.

[63] V. Hernandez, J. E. Roman, A. Tomas, and V. Vidal. A survey of software for sparse eigenvalue problems. Technical Report STR-6, Universitat Politècnica de València, 2009. Available at http ://www.grycap.upv.es/slepc.

[64] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc : A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3) :351–362, 2005.

[65] P. Holmes, J. L. Lumley, and G. Berkooz. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 1998.

[66] T. Hou, X.H. Wu, and Z. Cai. Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients. *Mathematics of Computation of the American Mathematical Society*, 68(227) :913–943, 1999.

[67] T. Y. Hou and X.H. Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of computational physics*, 134(1) :169–189, 1997.

[68] T. J. R. Hughes. Multiscale phenomena : Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer methods in applied mechanics and engineering*, 127(1) :387–401, 1995.

[69] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.B. Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer methods in applied mechanics and engineering*, 166(1) :3–24, 1998.

[70] T.J.R. Hughes, G. Scovazzi, and L.P. Franca. *Multiscale and stabilized methods*. Wiley Online Library, 2004.

[71] D.B.P. Huynh, G. Rozza, S. Sen, and A.T. Patera. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf–sup stability constants. *Comptes Rendus Mathematique*, 345(8) :473–478, 2007.

[72] G. Karniadakis and S.J. Sherwin. *Spectral/hp element methods for CFD*. Oxford University Press, USA, 1999.

[73] G. Karypis and V. Kumar. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*, 1998.

[74] D. Klindworth, M. A. Grepl, and G. Vossen. Certified reduced basis methods for parametrized parabolic partial differential equations with non-affine source terms. *Computer Methods in Applied Mechanics and Engineering*, 209 :144–155, 2012.

[75] D. J. Knezevic. Reduced basis approximation and a posteriori error estimates for a multiscale liquid crystal model. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4) :443–461, 2011.

[76] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2) :492–515, 2002.

[77] J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, pages 166–178, 1967.

[78] L. Machiels, Y. Maday, I. B. Oliveira, A. T. Patera, and D. V. Rovas. Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 331(2) :153–158, 2000.

[79] Y. Maday, N. C. Nguyen, A. T. Patera, and G. S.H. Pau. A general, multipurpose interpolation procedure : the magic points. 2007.

[80] Y. Maday, A. T. Patera, and G. Turinici. Global a priori convergence theory for reduced-basis approximations of single-parameter symmetric coercive elliptic partial differential equations. *Comptes Rendus Mathematique*, 335(3) :289–294, 2002.

[81] Y. Maday, A. T. Patera, and G. Turinici. A priori convergence theory for reduced-basis approximations of single-parameter elliptic partial differential equations. *Journal of Scientific Computing*, 17(1-4) :437–446, 2002.

[82] K. Martin and B. Hoffman. An open source approach to developing software in a small organization. *Software, IEEE*, 24(1) :46–53, 2007.

[83] K. Martin and B. Hoffman. *Mastering CMake 4th Edition*. Kitware, Inc., 2008.

[84] Y. Masayuki. A space-time Petrov-Galerkin certified reduced basis method : Application to the boussinesq equations. Submitted to SIAM Journal on Scientific Computing, December 2012.

[85] N.C. Nguyen, G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for parametrized parabolic pdes ; application to real-time bayesian parameter estimation. *Biegler, Biros, Ghattas, Heinkenschloss, Keyes, Mallick, Tenorio, van Bloemen Waanders, and Willcox, editors, Computational Methods for Large Scale Inverse Problems and Uncertainty Quantification, John Wiley & Sons, UK*, 2009.

[86] A. K. Noor. Recent advances in reduction methods for nonlinear problems. *Computers & Structures*, 13(1) :31–44, 1981.

[87] A. K. Noor. On making large nonlinear problems small. *Computer methods in applied mechanics and engineering*, 34(1) :955–985, 1982.

[88] A. K. Noor and Jeanne M. Peters. Reduced basis technique for nonlinear analysis of structures. *Aiaa journal*, 18(4) :455–462, 1980.

[89] G. Papanicolau, A. Bensoussan, and J.L. Lions. *Asymptotic analysis for periodic structures*. Elsevier, 1978.

[90] A.T. Patera and G. Rozza. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations. MIT Pappalardo Graduate Monographs in Mechanical Engineering, 2007. Copyright MIT 2006-2007.

[91] G. Pena. *Spectral element approximation of the incompressible Navier-Stokes equations evolving in a moving domain and applications*. PhD thesis, École Polytechnique Fédérale de Lausanne, November 2009. n°4529.

[92] J. S. Peterson. The reduced basis method for incompressible viscous flow calculations. *SIAM Journal on Scientific and Statistical Computing*, 10(4) :777–786, 1989.

[93] T.A. Porsching. Estimation of the error in the reduced basis method solution of nonlinear equations. *Mathematics of Computation*, 45(172) :487–496, 1985.

[94] C. Prud'Homme. *Contributions to Real-Time Reliable Simulations and Certain Aspects of Scientific Computing*. Habilitation à diriger des recherches, Université Pierre et Marie Curie – Paris VI, Décembre 2005.

[95] C. Prud'homme. A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming*, 14(2) :81–110, 2006.

[96] C. Prud'homme. A reduced basis multiscale method. Talk given at International Conference on Spectral and High Order Methods, ICOSAHOM'09. invited conference, June 2009.

[97] C. Prud'Homme, V. Chabannes, V. Doyeux, M. Ismail, A. Samake, and G. Pena. Feel++ : A Computational Framework for Galerkin Methods and Advanced Numerical Methods. *ESAIM : Proc.*, 38 :429 – 455, December 2012.

[98] C. Prud'Homme, V. Chabannes, V. Doyeux, M. Ismail, A. Samake, G. Pena, C. Daversin, and C. Trophime. Advances in feel++ : A domain specific embedded language in C++ for partial differential equations. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012), 2012.

[99] C. Prud'homme, V. Chabannes, and G. Pena. Feel++ : Finite Element Embedded Language in C++. Free Software available at http ://www.feelpp.org. Contributions from A. Samake, V. Doyeux, M. Ismail and S. Veys.

[100] C. Prud'homme and A.T. Patera. Reduced-basis output bounds for approximately parameterized elliptic coercive partial differential equations. *Computing and Visualization in Science*, 6(2-3) :147–162, 2004.

[101] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations : Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1) :70–80, 2002.

[102] C. Prud'homme. Life : Overview of a unified c++ implementation of the finite and spectral element methods in 1d, 2d and 3d. In Bo Kågström, Erik Elmroth, Jack Dongarra, and Jerzy Waśniewski, editors, *Applied Parallel Computing. State of the Art in Scientific Computing*, volume 4699 of *Lecture Notes in Computer Science*, pages 712–721. Springer Berlin Heidelberg, 2007.

[103] A. Quarteroni, G. Rozza, and A. Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1) :1–49, 2011.

[104] A. Quarteroni and A. Valli. *Domain decomposition methods for partial differential equations*. Number CMCS-BOOK-2009-019. Oxford University Press, 1999.

[105] Werner C. Rheinboldt. On the theory and error estimation of the reduced basis method for multi-parameter problems. Technical report, DTIC Document, 1992.

[106] D.V. Rovas, L. Machiels, and Y. Maday. Reduced-basis output bound methods for parabolic problems. *IMA journal of numerical analysis*, 26(3) :423–445, 2006.

[107] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3) :1–47, 2007.

[108] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education, 2004.

[109] E. Sánchez-Palencia. Non-homogeneous media and vibration theory. In *Non-homogeneous media and vibration theory*, volume 127, 1980.

[110] G. Sangalli. Capturing small scales in elliptic problems using a residual-free bubbles finite element method. *Multiscale Modeling & Simulation*, 1(3) :485–503, 2003.

[111] E. Schenone, S. Veys, and C. Prud'Homme. High Performance Computing for the Reduced Basis Method. Application to Natural Convection. *ESAIM : Proceedings*, pages 255 – 273, December 2013.

[112] S. Sen. *Reduced basis approximation and a posteriori error estimation for non-coercive elliptic problems : application to acoustics*. PhD thesis, Massachusetts Institute of Technology, 2007.

[113] S. Sen. Reduced-basis approximation and a posteriori error estimation for many-parameter heat conduction problems. *Numerical Heat Transfer, Part B : Fundamentals*, 54(5) :369–389, 2008.

[114] B. Smith. Petsc introductory tutorial. 2006.

[115] B.F. Smith, P. Bjorstad, and W. Gropp. Domain decomposition : parallel multilevel methods for elliptic partial differential equations, 1996.

[116] A. Toselli and O. Widlund. *Domain decomposition methods : algorithms and theory*, volume 3. Springer, 2005.

[117] C. Trophime, C. Daversin, C. Prud'homme, and S. Veys. Reduced order modeling of high magnetic field magnets. In *17th U.S. National Congress on Theoretical and Applied Mechanics*, june 2014.

[118] K. Urban and A. T. Patera. A new error bound for reduced basis approximation of parabolic partial differential equations. *Comptes Rendus Mathematique*, 350(3) :203–207, 2012.

[119] S. Vallaghé, M. Fouquembergh, A. Le Hyaric, and C. Prud'Homme. A successive constraint method with minimal offline constraints for lower bounds of parametric coercivity constant. 2011.

[120] S. Vallaghé and A.T. Patera. The static condensation reduced basis element method for a mixed-mean conjugate heat exchanger model. Submitted to SIAM Journal on Scientific Computing, August 2012.

[121] K. Veroy and A.T. Patera. Certified real-time solution of the parametrized steady incompressible navier-stokes equations : Rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Meth. Fluids*, 47 :773–788, 2005.

[122] K. Veroy, C. Prud'homme, and A.T. Patera. Reduced-basis approximation of the viscous Burgers equation : Rigorous *a posteriori* error bounds. *C. R. Acad. Sci. Paris, Série I*, 337(9) :619–624, November 2003.

[123] K. Veroy, C. Prud'homme, D. V. Rovas, and A.T. Patera. *A posteriori* error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic

partial differential equations (AIAA Paper 2003-3847). In *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, June 2003.

[124] K. Veroy, D. V. Rovas, and A. T. Patera. A posteriori error estimation for reduced-basis approximation of parametrized elliptic coercive partial differential equations :"convex inverse" bound conditioners. *ESAIM : Control, Optimisation and Calculus of Variations*, 8 :1007–1028, 2002.

[125] S. Veys, R. Chakir, C. Daversin, C. Prud'homme, and C. Trophime. A computational framework for certified reduced basis methods : application to a multiphysic problem. In *Conference Record of the 6th European Congress on Computational Methods in Applied Sciences and Engineering*, 2012.

[126] S. Veys, C. Daversin, C. Prud'homme, and C. Trophime. Reduced order modeling of high magnetic field magnets. In *Conference Record of the 11th International Workshop on Finite Elements for Microwave Engineering*, 2012.

[127] E. Weinan and B. Engquist. Multiscale modeling and computation. *Notices of the AMS*, 50(9) :1062–1070, 2003.

[128] E. Weinan and B. Engquist. The heterogeneous multi-scale method for homogenization problems. In *Multiscale methods in science and engineering*, pages 89–110. Springer, 2005.

[129] E. Weinan, B. Engquist, and Z. Huang. Heterogeneous multiscale method : a general methodology for multiscale modeling. *Physical Review B*, 67(9) :092101, 2003.

[130] P. Wesseling. Introduction to multigrid methods. Technical report, DTIC Document, 1995.

[131] EADS Innovation Works, EDF Research & Development, and PhiMECA. OpenTURNS Web page, 2012. http ://www.openturns.org.

Résumé

Aujourd'hui, dans de nombreux champs d'applications, de plus en plus de problèmes d'ingénierie demandent d'avoir une évaluation précise et efficace de quantités d'intérêt. Très souvent, ces quantités dépendent de la solution d'une équation aux dérivées partielles (EDP) paramétrée où les paramètres – physiques ou géométriques – sont les entrées du modèle et les quantités d'intérêt – valeurs moyennes – en sont les sorties. Les techniques de réduction d'ordre, notamment la méthode des bases réduites qui est la méthode utilisée tout au long de ces travaux, permettent de répondre à ces demandes. Dans cette thèse nous nous intéressons à la mise en place d'un framework en C++, supportant le calcul parallèle, permettant d'appliquer la méthode des bases réduites à des problèmes multi-physiques non-linéaires tels que les problèmes de convection naturelle (couplage fluide-thermique), ou encore la modélisation d'aimants de type résistifs à hauts champs (nous nous limitons au couplage thermo-electrique) aboutissant à une étude sur la quantification d'incertitude. La méthode des bases réduites s'appuie naturellement sur une approximation obtenue via la discrétisation élément fini du problème à traiter. Pour cela nous utilisons la librairie de calcul Feel++, spécialisée dans la résolution d'EDPs. Nous nous intéressons également aux problèmes de type multi-échelles. La particularité de ces problèmes est de manipuler un ensemble de phénomènes mettant en jeu des échelles différentes, comme c'est le cas par exemple lorsque nous considérons un écoulement en milieu poreux. La méthode des éléments finis multi-échelles permet d'avoir le comportement "global", associé aux grandes échelles, de la solution du problème sans devoir le résoudre sur les petites échelles. Nous proposons une nouvelle construction des fonctions de base élément fini multi-échelles basée sur la méthode des bases réduites.

Mots-clés : méthode élément fini multi-échelles, méthode base réduite, calcul parallèle, Feel++, C++ .