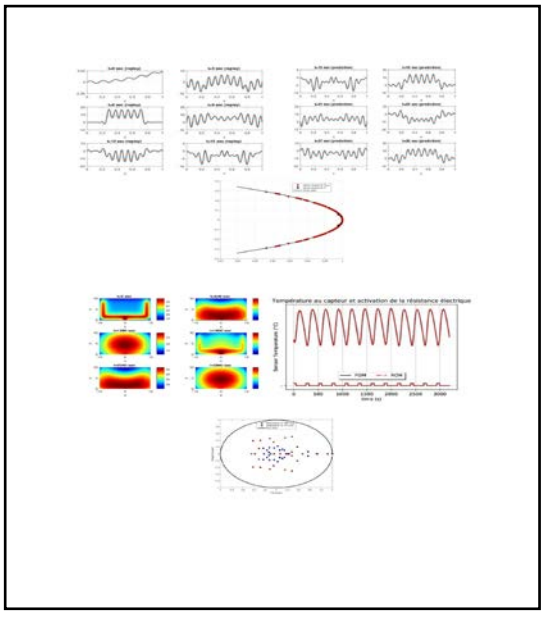
Réduction de modèles et apprentissage de solutions spatio-temporelles paramétrées à partir de données : application à des couplages EDP-EDO

Tarik Fahlaoui

Par **Tarik FAHLAOUI**

Réduction de modèles et apprentissage de solutions spatio-temporelles paramétrées à partir de données : application à des couplages EDP-EDO

Thèse présentée pour l'obtention du grade de Docteur de l'UTC



Soutenue le 16 janvier 2020
Spécialité : Mathématiques Appliquées et Problèmes Inverses : Laboratoire de Mathématiques Appliquées de Compiègne (Unité de recherche EA-2222)

D2535

THÈSE

pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

ÉCOLE DOCTORALE 71 : Sciences pour l'ingénieur

Spécialité

Mathématiques Appliquées - Problèmes Inverses

présentée et soutenue par

Tarik FAHLAOUI

le 16 Janvier 2020

Réduction de modèles et apprentissage de solutions spatio-temporelles paramétrées à partir de données

Application à des couplages EDP-EDO

Devant le Jury composé de

Mme Ghislaine GAYRAUD	PROFESSEURE, UTC - LMAC	Présidente du Jury
M Stéphane CLÉNET	PROFESSEUR, ENSAM ParisTech - Lille	Rapporteur
M Iraj MORTAZAVI	PROFESSEUR, CNAM - Paris	Rapporteur
M Amine AMMAR	PROFESSEUR, ENSAM ParisTech - Angers	Examinateur
M Georges OPPENHEIM	PROFESSEUR, UPEM - LAMA	Examinateur
M Pierre VILLON	PROFESSEUR, UTC - Roberval	Invité
M Florian DE VUYST	PROFESSEUR, UTC - LMAC	Directeur de thèse

Remerciements

Lorsque j'étais au Japon, j'ai tiré un *Omikuji* où il était écrit

"N'oubliez pas de remercier ceux grâce à qui vous réussissez."

Aujourd'hui, cette phrase prend tout son sens ...

Je tiens en premier lieu à remercier Florian De Vuyst, pour sa bienveillance et son attention tout au long de cette thèse. Ce fut un vrai plaisir de travailler avec lui. Je tiens également à remercier Ghislaine Gayraud pour sa gentillesse et sa disponibilité, de mon arrivée au laboratoire à ma soutenance de thèse, ainsi que Salim Bouzebda pour sa bonne humeur communicative.

Je remercie les membres du jury, Ghislaine Gayraud, Stéphane Clénet, Iraj Mortazavi, Amine Ammar, Georges Oppenheim, Pierre Villon et Florian De Vuyst, d'avoir accepté d'évaluer ce travail et pour les échanges enrichissants que nous avons eu. Mais également les rapporteurs Stéphane Clénet et Iraj Mortazavi d'avoir consacré du temps à mon manuscrit ainsi que pour leurs rapports détaillés et constructifs.

Les séminaires doctorants m'ont beaucoup appris et ce fut un plaisir d'y prendre part, ainsi je remercie Joseph, Wang, Joséphine, Ivan, Tea, Brenda et Boutheina.

Je remercie aussi mes amis, Alex, David, Giu, Jordan, Kévin, Kogu, Mathieu, Marvin, Oscar, et Thomas pour leurs soutiens et leurs encouragements.

Je remercie également ma famille.

Enfin, je remercie Inès pour son soutien infaillible, et les (très) nombreuses heures passées à la relecture de ce manuscrit.

Titre : Réduction de modèles et apprentissage de solutions spatio-temporelles paramétrées à partir de données

Résumé On s'intéresse dans cette thèse à l'apprentissage d'un modèle réduit précis et stable, à partir de données correspondant à la solution d'une *équation aux dérivées partielles* (EDP), et générées par un *solveur haute fidélité* (HF). Pour ce faire, on utilise la méthode *Dynamic Mode Decomposition* (**DMD**) ainsi que la méthode de réduction *Proper Orthogonal Decomposition* (**POD**). Le modèle réduit appris est facilement interprétable, et par une analyse spectrale a posteriori de ce modèle on peut détecter les anomalies lors de la phase d'apprentissage. Les extensions au cas de couplage EDP-EDO, ainsi qu'au cas d'EDP d'ordre deux en temps sont présentées. L'apprentissage d'un modèle réduit dans le cas d'un système dynamique contrôlé par commutation, où la règle de contrôle est apprise à l'aide d'un *réseau de neurones artificiel* (**ANN**), est également traité. Un inconvénient de la réduction POD, est la difficile interprétation de la représentation basse dimension. On proposera alors l'utilisation de la méthode *Empirical Interpolation Method* (**EIM**). La représentation basse dimension est alors intelligible, et consiste en une restriction de la solution en des points sélectionnés. Cette approche sera ensuite étendue au cas d'EDP dépendant d'un paramètre, et où l'algorithme *Kernel Ridge Regression* (**KRR**) nous permettra d'apprendre la variété solution. Ainsi, on présentera l'apprentissage d'un modèle réduit paramétré. L'extension au cas de données bruitées ou bien au cas d'EDP d'évolution non linéaire est présentée en ouverture.

Mots-clés : Réduction de modèles ; Data-driven ; EDP paramétrées ; DMD ; POD ; EIM ; ANN ; Apprentissage automatique

Title : Data-based learning of parametrized spatio-temporal solutions and model order reduction

Abstract In this thesis, an algorithm for learning an accurate reduced order model from data generated by a *high fidelity solver* (HF solver) is proposed. To achieve this goal, we use both *Dynamic Mode Decomposition* (**DMD**) and *Proper Orthogonal Decomposition* (**POD**). Anomaly detection, during the learning process, can be easily done by performing an a posteriori spectral analysis on the reduced order model learnt. Several extensions are presented to make the method as general as possible. Thus, we handle the case of coupled ODE/PDE systems or the case of second order hyperbolic equations. The method is also extended to the case of switched control systems, where the switching rule is learnt by using an *Artificial Neural Network* (**ANN**). The reduced order model learnt allows to predict time evolution of the POD coefficients. However, the POD coefficients have no interpretable meaning. To tackle this issue, we propose an interpretable reduction method using the *Empirical Interpolation Method* (**EIM**). This reduction method is then adapted to the case of third-order tensors, and combining with the *Kernel Ridge Regression* (**KRR**) we can learn the solution manifold in the case of parametrized PDEs. In this way, we can learn a parametrized reduced order model. The case of non-linear PDEs or disturbed data is finally presented in the opening.

Keywords : Model order reduction ; Data-driven ; Parametrized PDEs ; DMD ; POD ; EIM ; ANN ; Machine Learning

Thèse préparée au **Laboratoire de Mathématiques Appliquées de Compiègne EA 2222, Université de Technologie de Compiègne, Alliance Sorbonne Université, Compiègne France**
Sous la direction de **Florian De Vuyst, Professeur des Universités, Université de Technologie de Compiègne, LMAC**

Table des matières

Table des figures

Liste des tableaux

Nomenclature

Notation mathématique

$\boldsymbol{x}$	Variable d'espace
t	Variable de temps
$\boldsymbol{\mu}$	Paramètre
$\mathcal{L}$	Opérateur linéaire
Δ	Laplacien
$\nabla\cdot$	Divergence
$\mathcal{S}$	Matrice de snapshots
$\phi_k(\boldsymbol{x})$	Mode POD continu
$\boldsymbol{\phi}_k$	Mode POD discret
$\boldsymbol{\Phi}_r$	Matrice contenant les modes POD discrets
$\mathbf{a}_k$	Coefficient POD
$\mathbf{A}$	Matrice contenant les coefficients POD
W^K	Espace POD
$\mathbf{U}$	Matrice de données contenant la solution de l'EDP générée par le solveur HF
$\boldsymbol{\Theta}$	Matrice de données contenant la solution de l'EDO générée par le solveur HF
$\mathbf{M}$	Matrice de masse
$\mathbf{K}$	Matrice de rigidité
$\otimes$	Produit de Kronecker
X	Ensemble d'entrée d'entrainement

Y	Ensemble de sortie d'entrainement
$\mathcal{A}$	Donnée grande dimension
A	Donnée basse dimension
$\hat{A}$	Donnée apprises
$\tilde{A}$	Donnée projetée
$A^{\mathbb{S}}$	Donnée empilée
$A^{\dagger}$	Pseudo-inverse de Moore-Penrose de A
$\|A\|_F$	Norme de Frobenius de A
$\rho(A)$	Rayon spectral de A
σ	Application permettant de sélectionner des points
N_x	Nombre de degrés de libertés
N_t	Nombre de pas de temps
δ_x	Pas d'espace
δ_t	Pas de temps
K	Dimension du modèle réduit

Symbol physique

κ	Conductivité thermique
c	Célérité de l'onde
ω	Pulsation

Chapitre 1

Introduction

Avec l'émergence des premiers ordinateurs, peu après la Seconde Guerre mondiale, les mathématiques appliquées se sont révélées être un outil fondamentales dans le monde de l'industrie. En effet, les mathématiques appliquées peuvent être résumées comme la modélisation mathématique de phénomènes réels divers et variés (provenant de la physique, de la biologie ou encore de l'économie), et le calcul de la solution à ce modèle ([4]). Cela permet alors de simuler ce phénomène avec pour seul outil un ordinateur, et donc s'éviter le coût de mesures expérimentales. L'augmentation de la puissance des ordinateurs, ainsi que les nombreux travaux en mathématiques appliquées ont permis de simuler des problèmes de plus en plus complexes. Parallèlement, plusieurs disciplines reposant sur la modélisation mathématique et la simulation numérique, telle que l'optimisation de formes ou le contrôle optimal, se sont développées afin de répondre à des questions industrielles, tel que la conception optimale d'un matériau. Ces disciplines requièrent de résoudre de nombreuses fois le modèle mathématique. Par ailleurs, les modèles mathématiques sont devenus de plus en plus complexes, et les solutions calculées de plus en plus précises. Malgré le développement des ordinateurs et des algorithmes de parallélisation, les ingénieurs se sont retrouvés confronté au choix entre le calcul d'une solution peu précise mais obtenue rapidement et d'une solution très précise mais qui requiert un long temps d'attente. C'est dans ce contexte que la réduction de modèles a fait son apparition, en proposant un compromis à ce choix.

Récemment, plusieurs auteurs se sont abstenus du modèle mathématique, pour la simulation numérique. Les motivations reposent soit sur le travail laborieux et biaisé de modélisation, ou bien la difficulté d'exploration du code de simulation pour construire un modèle réduit. Cette approche partage de nombreuses problématiques avec l'apprentissage automatique, et de nombreux auteurs ont utilisé des algorithmes issus de l'apprentissage automatique. Cependant, les simulations numériques et les

mesures expérimentales étant coûteuses, la simulation numérique à partir de données est réalisée dans un contexte *Small Data*. Une attention toute particulière doit alors être portée au choix des algorithmes utilisés. L'historique des travaux de modélisation et de simulation numérique permet d'avoir des connaissances a priori sur les données, et en quelque sorte de pouvoir s'approcher du modèle mathématique que l'on recherche. C'est d'ailleurs dans cet esprit que Maday & al ont introduit l'algorithme *PBDW* qui consiste a apprendre seulement le biais du modèle ([118], [76]).

Le premier paragraphe sera consacré à présenter la modélisation mathématique et la simulation numérique. Dans un second paragraphe, on présentera l'apprentissage automatique et les quelques algorithmes utilisés. Enfin, dans un dernier paragraphe, nous présenterons les récents travaux portant sur la simulation numérique à partir de données.

1.1 Modélisation mathématique et simulation numérique

1.1.1 Modélisation mathématique

La modélisation mathématique peut se résumer comme l'écriture abstraite d'un phénomène concret observé. Cette modélisation demande ainsi de comprendre le phénomène, puis de le traduire à l'aide de différents objets mathématiques.
Le plus souvent ces modèles s'écrivent sous forme d'équations différentielles, plus particulièrement, sous forme d'équations différentielles ordinaires (EDO) et/ou d'équations aux dérivées partielles (EDP).
Une EDO relie une fonction inconnue $t \mapsto x(t)$, et ses dérivées successives, que l'on formalise comme suit :

$$F\Big(t, x(t), \dot{x}(t), \ddot{x}(t), \cdots\Big) = 0$$

Tandis qu'une EDP relie une fonction inconnue $\boldsymbol{x} \mapsto u(\boldsymbol{x})$, à d variables, et ses dérivées partielles successives, que l'on formalise :

$$F\left(\boldsymbol{x}, u(\boldsymbol{x}), \frac{\partial u(\boldsymbol{x})}{\partial \boldsymbol{x}_1}, \cdots, \frac{\partial u(\boldsymbol{x})}{\partial \boldsymbol{x}_d}, \cdots, \frac{\partial^2 u(\boldsymbol{x})}{\partial \boldsymbol{x}_1 \boldsymbol{x}_2}, \cdots\right) = 0$$

Lorsque la fonction inconnue u dépend aussi du temps, on parle alors d'EDP d'évolution, que l'on écrit sous la forme :

$$F\left(t, \boldsymbol{x}, u(\boldsymbol{x},t), \frac{\partial u(\boldsymbol{x},t)}{\partial t}, \frac{\partial^2 u(\boldsymbol{x},t)}{\partial t^2}, \cdots, \frac{\partial u(\boldsymbol{x},t)}{\partial \boldsymbol{x}_1}, \cdots, \frac{\partial u(\boldsymbol{x},t)}{\partial \boldsymbol{x}_d}, \cdots, \frac{\partial^2 u(\boldsymbol{x},t)}{\partial \boldsymbol{x}_1 \boldsymbol{x}_2}, \cdots\right) = 0.$$

Les premiers travaux de modélisation utilisant des EDP remontent au XVIII$^{\text{e}}$ siècle ([27]). En 1746, d'Alembert modélisa la propagation d'une onde en une dimension en étudiant les vibrations d'une corde de violon. Pour cela, il considéra une chaîne de masses ponctuelles interconnectées, puis en faisant tendre le nombre de masses vers l'infini([163]), il trouva le modèle suivant :

$$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{\partial^2 u(x,t)}{\partial x^2},$$

où la fonction $u(x,t)$ représente le déplacement de la corde au point d'espace x et de temps t. Ce modèle fut par la suite étendu en dimension deux par Euler en 1759, puis en dimension trois par Bernoulli en 1762.
Autour des années 1780, Pierre-Simon de Laplace introduisit l'équation de Laplace pour des besoins en mécanique newtonienne. Laplace montra alors que le potentiel gravitationnel $u(\boldsymbol{x})$ satisfait l'équation :

$$\Delta\, u(\boldsymbol{x}) := \frac{\partial^2 u(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_1} + \frac{\partial^2 u(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_2} + \frac{\partial^2 u(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_3} = 0.$$

Cette équation fut par la suite utilisée pour modéliser de nombreux phénomènes en astronomie, en mécanique des fluides ou encore en thermie.
En 1822, Joseph Fourier modélisa la diffusion de la chaleur en observant que le flux de chaleur était proportionnel à l'opposé du gradient de température ([12]), autrement dit, le flux de chaleur se déplace du plus chaud vers le plus froid. Cette observation fut modélisée par la loi de Fourier (établie mathématiquement par Biot en 1804, [70]), i.e.

$$\mathbf{q} = -\kappa \nabla u.$$

En ajoutant la conservation de l'énergie, Fourier modélisa alors la diffusion de la chaleur par l'équation :

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \kappa\, \Delta u(\boldsymbol{x},t),$$

avec κ la conductivité thermique, et u la température.

D'un point de vue mathématique, ces trois modèles représentent les trois principales classes d'EDP à savoir : hyperbolique, elliptique et parabolique. Mais ils sont aussi la base de nombreux modèles mathématiques traduisant des phénomènes plus complexes et indispensables dans le monde de l'industrie. C'est par exemple le cas, en aérodynamique, du système d'équations de Navier-Stokes, établi par Navier et Stokes en 1845 ([70]), qui permet de décrire le mouvement de fluides newtoniens, et s'obtient par des lois de conservations. Une des écriture de ce système est la suivante :

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot \Big(\rho \mathbf{V}\Big) = 0, \\ \frac{\partial \rho \mathbf{V}}{\partial t} + \nabla \cdot \Big(\rho \mathbf{V} \otimes \mathbf{V} + p\Big) + \nabla \cdot \boldsymbol{\tau} = 0, \\ \frac{\partial \rho E}{\partial t} + \nabla \cdot \Big(\rho H \mathbf{V}\Big) + \nabla \cdot \Big(\mathbf{V}^T \boldsymbol{\tau} - \mathbf{q}\Big) = 0, \end{cases}$$

où ρ est la densité du fluide, $\mathbf{V}$ le champ de vitesses, p la pression du fluide, E l'énergie totale du système, H est l'enthalpie, $\boldsymbol{\tau}$ le tenseur des contraintes, et $\mathbf{q}$ le flux de chaleur défini par la loi de Fourier. Ce modèle complexe permet notamment de simuler l'écoulement de l'air autour d'une aile d'avion ou d'une voiture.
Un autre modèle à la base de nombreuses innovations comme la radio ou la télévision ([164]) est dû à Maxwell en 1862. Ce modèle écrit sous la forme d'un système d'EDP exprime le comportement des champs électriques et magnétiques. Une écriture de ce modèle est :

$$\begin{cases} \nabla \cdot \mathbf{D} = \rho, \\ \nabla \cdot \mathbf{B} = 0, \\ \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \\ \nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \end{cases}$$

avec $\mathbf{E}$ l'intensité du champ électrique, $\mathbf{H}$ l'intensité du champ magnétique, $\mathbf{D}$ la densité du champ électrique, $\mathbf{B}$ la densité du champ magnétique, $\mathbf{J}$ la densité du courant électrique, et ρ la densité de charge électrique. Enfin en finance, le modèle Black-Scholes, proposé par Fischer Black et Myron Scholes ([19]) en 1973, décrit l'évolution du prix de l'option d'achat d'une action au cours du temps. Ce modèle est défini par

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial^2 S} + rS\frac{\partial V}{\partial S} - rV = 0,$$

où V est le prix de l'option et est une fonction du prix de l'option initial S et du temps t, r est le taux d'intérêt et σ la volatilité de l'action.

Si les équations présentées ci-dessus expriment mathématiquement des phénomènes complexes, les quantités d'intérêts telles que le prix de l'option dans le cas du modèle Black-Scholes, ne sont connues qu'implicitement comme étant les solutions de certaines

équations. Ainsi pour pouvoir simuler le phénomène, il nous faut résoudre ces équations. Le paragraphe suivant présente quelques méthodes numériques permettant de résoudre ces équations. Ces méthodes numériques reposent souvent sur une analyse mathématique de ces équations, dont de nombreux auteurs s'y sont intéressés ([60], [116], [11], [26], [68], [157], [44], [73], [105]).

Remarque 1. *Nous ne nous intéresserons dans cette thèse qu'à des équations différentielles déterministes. Notons tout de même que les EDO et les EDP ont des analogues dans le cas stochastique à savoir les équations différentielles stochastiques (EDS), et les équations aux dérivées partielles stochastiques (EDPS, [182]), et qui correspondent à des équations différentielles perturbées par un bruit blanc.*

1.1.2 Méthodes numériques pour la résolution d'EDP

Pour la très grande majorité des modèles, trouver la solution analytique n'est pas possible. On doit donc résoudre numériquement l'équation, c'est-à-dire calculer une approximation de la solution à partir d'un ordinateur. Cela nécessite de retranscrire notre modèle continu en un modèle discret. Il existe de nombreuses méthodes permettant d'obtenir une version discrète de notre modèle et une bonne connaissance du comportement de la solution du modèle continu permet de choisir une méthode numérique adéquate. On présentera ici deux méthodes numériques largement utilisées pour résoudre des EDP. Afin de simplifier la présentation on considérera l'équation de la chaleur en une dimension. Par ailleurs, le but étant in fine de résoudre les équations, on ne s'attardera pas sur l'analyse de ces méthodes mais plutôt sur leurs mises en œuvre.

1.1.2.1 Méthodes des différences finies (FDM)

La méthode des différences finies (FDM) est une des méthodes numériques les plus anciennes, elle remonte aux travaux d'Euler en 1768 ([59]) et est l'une des plus simples à mettre en œuvre ([160], [4]). Elle consiste à discrétiser le domaine et les opérateurs. Afin de mieux comprendre cette méthode, reprenons l'équation de la chaleur présentée au paragraphe précédent, mais en une dimension et pour une conductivité thermique égale à 1. Afin que le problème soit bien posé, c'est-à-dire qu'il admette une unique solution, il faut imposer une température initiale ainsi que des conditions aux limites. La température au temps $t = 0$ sera alors donnée par une fonction u_0, tandis qu'on imposera $u = 0$ sur le bord du domaine pour tout temps. Le modèle que l'on souhaite

discrétiser s'écrit alors

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} = \dfrac{\partial^2 u(x,t)}{\partial x^2}, & \forall (x,t) \in [0;L] \times (0,T), \\ u(0,t) = u(L,t) = 0, & \forall t \in (0,T), \\ u(x,0) = u_0(x), & \forall x \in [0;L]. \end{cases} \tag{1.1}$$

Il nous faut, en premier lieu, discrétiser l'espace et le temps, c'est-à-dire sélectionner N_x points dans l'intervalle $[0;L]$, et N_t points dans l'intervalle $(0,T)$. Le plus simple est de les choisir de façon uniforme. On prendra alors un pas d'espace $\delta x := \frac{L}{N_x - 1}$, et un pas de temps $\delta t := \frac{T}{N_t - 1}$. Puis on sélectionnera les points en espace de la manière suivante :

$$x_i = (i-1)\,\delta x, \qquad \forall 1 \leq i \leq N_x,$$

et les points en temps, comme ceci :

$$t^n = (n-1)\,\delta t, \qquad \forall 1 \leq n \leq N_t.$$

La deuxième étape consiste à remplacer les dérivées par des différences finies et repose sur la formule de Taylor. Appliquée à $u(x_{i+1}, t)$ au voisinage de (x_i, t), la formule de Taylor nous donne :

$$u(x_{i+1},t) = u(x_i + \delta x, t) = u(x_i,t) + \delta x \,\frac{\partial u(x_i,t)}{\partial x} + \frac{\delta x^2}{2}\,\frac{\partial^2 u(x_i,t)}{\partial x^2} + \frac{\delta x^3}{3!}\,\frac{\partial^3 u(x_i,t)}{\partial x^3} + o(\delta x^4),$$

en faisant de même avec $u(x_{i-1}, t)$, on trouve

$$u(x_{i-1},t) = u(x_i - \delta x, t) = u(x_i,t) - \delta x \,\frac{\partial u(x_i,t)}{\partial x} + \frac{\delta x^2}{2}\,\frac{\partial^2 u(x_i,t)}{\partial x^2} - \frac{\delta x^3}{3!}\,\frac{\partial^3 u(x_i,t)}{\partial x^3} + o(\delta x^4),$$

finalement en sommant ces deux égalités et en divisant le tout par δx^2, on obtient,

$$\frac{u(x_{i+1},t) - 2u(x_i,t) + u(x_{i-1},t)}{\delta x^2} = \frac{\partial^2 u(x_i,t)}{\partial x^2} + o(\delta x^2).$$

On a ainsi approché la dérivée en espace par une différence finie. Et cette approximation dépend du pas d'espace, ainsi plus le pas d'espace sera petit, plus l'approximation sera bonne. En ce qui concerne la dérivée en temps, on procède de la même façon en appliquant la formule de Taylor à $u(x, t^{n+1})$ voisinage de (x, t^n) et on obtient

$$u(x,t^{n+1}) = u(x,t^n + \delta t) = u(x,t^n) + \delta t \frac{\partial u(x,t^n)}{\partial t} + o(\delta t^2),$$

et ainsi on a pour la dérivée en temps

$$\frac{u(x,t^{n+1})-u(x,t^n)}{\delta t}=\frac{\partial u(x,t^n)}{\partial t}+o(\delta t).$$

C'est donc à partir de ces approximations que nous allons résoudre numériquement l'équation (1.1). En effet, on remplace les deux dérivées dans l'équation (1.1) par les approximations présentées ci-dessus. Ainsi en notant u_i^n l'approximation obtenue de la solution u au point x_i et au temps t^n, on obtient le modèle discret, que l'on appelle généralement schéma numérique, suivant :

$$\begin{cases} \dfrac{u_i^{n+1}-u_i^n}{\delta t} = \dfrac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{\delta x^2}, \quad \forall 2\leq i\leq N_x-1, \text{ et } 1\leq n\leq N_t-1, \\ u_1^n=u_{N_x}^n=0, \qquad \forall 1\leq n\leq N_t, \\ u_i^1=u_0(x_i), \qquad \forall 2\leq N_x-1. \end{cases} \tag{1.2}$$

La dernière étape consiste à réécrire ce modèle discret sous la forme du système linéaire :

$$\begin{cases} \mathbf{u}^{n+1}=\mathbf{A}\,\mathbf{u}^n, \qquad \forall 1\leq n\leq N_t-1 \\ \mathbf{u}_1^n=\mathbf{u}_{N_x}^n=0, \qquad \forall 1\leq n\leq N_t \\ \mathbf{u}^1=\big(u_0(x_i)\big)_{2\leq i\leq N_x-1} \end{cases}$$

Ce schéma, introduit en 1924 par Schmidt ([152],[5]), est dit *explicite* car la température est connue par une simple multiplication matrice-vecteur et ne nécessite pas d'inversion de matrice. Plus généralement, on définit le θ-schéma par :

$$\frac{u_i^{n+1}-u_i^n}{\delta t} = \theta\,\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\delta x^2}+(1-\theta)\,\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{\delta x^2}$$

Lorsque $\theta=0$, on retrouve alors le schéma (1.2). Cependant, pour des équations de transport cela peut poser des problèmes de stabilité, on préférera alors utiliser un schéma implicite (cas où $\theta=1$). Par ailleurs, lorsque le problème est fortement dynamique, on choisira un schéma d'ordre 2 en temps (cas où $\theta=\frac{1}{2}$, on parle alors de schéma de Crank-Nicholson).

La méthode des différences finies est souvent utilisée car elle est simple à mettre en place et permet d'approcher les solutions de nombreuses équations de façon précise. Il existe par ailleurs de nombreuses analyses de cette méthode et cela pour différentes équations ([43],[160], [166], [174]). Un des inconvénients de cette méthode numérique est sa difficile généralisation à des géométries complexes. Ce n'est pas le cas de la méthode suivante.

1.1.2.2 Méthode des éléments finis (FEM)

La méthode des éléments finis a été introduite au milieu du XX^{e} siècle pour l'analyse des structures ([188]), et est devenue la plus utilisée pour la simulation en mécanique des solides. L'analyse numérique de cette méthode a quant à elle été introduite en 1973 par Strang et Fix ([165]). Si cette méthode consiste aussi à discrétiser le domaine, on ne discrétise pas les opérateurs mais la solution de notre modèle en cherchant à approcher sa projection dans un espace de dimension finie. Comme précédemment, on prendra l'équation (1.1), afin de comprendre la méthode. La première étape est de réécrire l'équation dans un cadre Hilbertien afin de pouvoir définir la notion de projection. Cela se fait en passant d'une formulation dite *forte* (i.e. le modèle (1.1)), à une formulation dite *faible* (aussi appelé formulation variationnelle). Pour ce faire, considérons une fonction $\psi \in H_0^1([0;L])$, autrement dit continue (car H^1 s'injecte continument dans C^0 en dimension un), dont la dérivée est de carrée intégrable et telle que $\psi(0) = \psi(L) = 0$. Multiplions ensuite la première équation de (1.1) par ψ et intégrons sur tout le domaine $[0;L]$, on a alors

$$\int_0^L \frac{\partial u(x,t)}{\partial t}\psi(x)dx - \int_0^L \frac{\partial^2 u(x,t)}{\partial x^2}\psi(x)dx = 0, \qquad \forall t \in (0,T),$$

en intégrant par parties le second terme du membre de gauche, on obtient,

$$\int_0^L \frac{\partial u(x,t)}{\partial t}\psi(x)dx + \int_0^L \frac{\partial u(x,t)}{\partial x}\frac{\partial \psi(x)}{\partial x}dx = 0, \qquad \forall t \in (0,T).$$

Jusqu'à présent, nous avons formulé différemment l'équation (1.1), mais nous avons toujours une équation continue en temps et en espace. La méthode des éléments finis ne permet pas de discrétiser en temps, ainsi en utilisant la méthode des différences finies, on discrétise la dérivée en temps par un schéma cette fois-ci implicite. L'équation discrétisée en temps est alors

$$\begin{cases} \displaystyle\int_0^L \frac{u^{n+1}(x)}{\delta t}\psi(x)dx + \int_0^L \frac{\partial u^{n+1}(x)}{\partial x}\frac{\partial \psi(x)}{\partial x}dx = \int_0^L \frac{u^n(x)}{\delta t}\psi(x)dx, \quad \forall 1 \leq n \leq N_t - 1, \\ u^1(x) = u_0(x), \qquad \forall x \in [0;L]. \end{cases} \tag{1.3}$$

La seconde étape est de discrétiser le domaine spatial, ici l'intervalle $[0;L]$. Contrairement aux différences finies, on peut avec cette méthode prendre une subdivision non uniforme sans grande difficulté. Cependant nous conserverons ici les points x_i introduits précédemment. On parlera alors de maillage, les points x_i seront les nœuds

du maillage, les segments $[x_i; x_{i+1}]$ les éléments du maillage, et les points x_1 et x_{N_x} les bords du maillage.

Une fois que l'on a notre formulation variationnelle et notre maillage, la dernière étape consiste à choisir un espace de dimension finie dans lequel on va chercher notre solution u. Le choix de l'espace a une importance cruciale, et doit être en cohérence avec les propriétés mathématiques de la solution u. Dans notre cas, l'espace des polynômes de degré 1 par morceaux convient. On notera alors V_h cet espace et on le définira comme

$$V_h := \{w : [0; L] \mapsto \mathbb{R}, \quad w \in C^0([0; L]), \quad w_{|_{[x_i;x_{i+1}]}} \in \mathbb{P}^1, \quad w(0) = w(L) = 0\}.$$

Cet espace est engendré par $N_x - 2$ fonctions notées ϕ_i, que l'on nomme généralement "fonctions chapeaux" en raison de leur forme. On définit ϕ_i, pour tout $2 \leq i \leq N_x - 1$ par

$$\begin{cases} \phi_i(x) = \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & \text{si } x \in [x_{i-1}; x_i], \\ \phi_i(x) = \dfrac{x_{i+1} - x}{x_{i+1} - x_i}, & \text{si } x \in [x_i; x_{i+1}], \\ \phi_i(x) = 0, & \text{sinon .} \end{cases}$$

Ainsi elles vérifient la propriété

$$\phi_i(x_j) = \delta_{i,j}, \tag{1.4}$$

où $\delta_{i,j}$ est le symbole de Kronecker qui vaut 1 si $i = j$ et 0 sinon.

On a alors l'équivalence

$$v_h \in V_h \iff \exists \big(\alpha_i\big)_{2 \leq i \leq N_x - 1} \text{ tel que } \quad v_h(x) = \sum_{i=2}^{N_x - 1} \alpha_i \, \phi_i(x), \quad \forall x \in [0; L],$$

et en utilisant la propriété (1.4), on obtient que

$$\alpha_i = v_h(x_i), \qquad \forall 2 \leq i \leq N_x - 1.$$

Projetons ensuite la formulation faible semi-discrète (1.3) dans l'espace V_h, et ainsi le problème totalement discrétisé devient,

$$\begin{cases} \text{Trouver } u_h \in V_h, \text{ telle que :} \\ \displaystyle\int_0^L \frac{u_h^{n+1}(x)}{\delta t} v_h(x) dx + \int_0^L \frac{\partial u_h^{n+1}(x)}{\partial x} \frac{\partial v_h(x)}{\partial x} dx = \int_0^L \frac{u_h^n(x)}{\delta t} v_h(x) dx \\ \forall v_h \in V_h, \text{ et } \forall 1 \leq n \leq N_t - 1, \\ u_h^1 = \Pi^{V_h} u_0. \end{cases} \tag{1.5}$$

où Π^{V_h} est la projection sur l'espace V_h. On a ainsi obtenu un modèle discret, et pour tout temps t^n la solution à ce modèle discret u_h^n est unique d'après le théorème de Lax-Milgram ([58]) et vérifie,

$$\|u(\cdot,t^n) - u_h^n\|_{L^2([0;L])} \leq C\delta x^2 |u(\cdot,t^n)|_{H^2([0;L])},$$

d'après le lemme d'Aubin-Nitsche ([58]). Cela implique donc que sous réserve que la solution u du modèle continu (1.1) soit suffisamment régulière, l'erreur commise par le modèle discret (on parle d'erreur de discrétisation) est proportionnelle au carré du pas d'espace. On est donc sûr de converger vers la solution exacte en prenant un maillage de plus en plus fin.

La dernière étape est, comme précédemment, d'écrire le modèle discret (1.5), sous forme matricielle afin de pouvoir le résoudre à l'aide d'un ordinateur. Pour cela, il suffit de substituer u_h^n par son développement sur la base des ϕ_i, i.e.

$$u_h^n(x) = \sum_{i=2}^{N_x-1} u_h^n(x_i)\,\phi_i(x), \qquad \forall n$$

et de poser $v_h = \phi_j$ en faisant varier l'indice j, on retrouve alors,

$$\begin{aligned}
&\sum_{i=2}^{N_x-1} u_h^{n+1}(x_i)\left(\frac{1}{\delta t}\int_0^L \phi_i(x)\,\phi_j(x)dx + \int_0^L \frac{\partial\phi_i(x)}{\partial x}\,\frac{\partial\phi_j(x)}{\partial x}dx\right)\\
&= \sum_{i=2}^{N_x-1} u_h^n(x_i)\,\frac{1}{\delta t}\int_0^L \phi_i(x)\,\phi_j(x)dx, \qquad \forall 2\leq j\leq N_x-1,
\end{aligned}$$

en définissant les matrices de masse $\mathbf{M}$ et de rigidité $\mathbf{K}$ par

$$\mathbf{M}_{i,j} = \int_0^L \phi_j(x)\,\phi_i(x)dx, \qquad \mathbf{K}_{i,j} = \int_0^L \frac{\partial\phi_j(x)}{\partial x}\,\frac{\partial\phi_i(x)}{\partial x}dx,$$

on obtient finalement le système linéaire

$$\begin{cases}
\left(\frac{1}{\delta t}\mathbf{M} + \mathbf{K}\right)\mathbf{u}^{n+1} = \frac{1}{\delta t}\mathbf{M}\,\mathbf{u}^n,\\
\mathbf{u}_1^n = \mathbf{u}_{N_x}^n = 0, \qquad \forall 2\leq n\leq N_t,\\
\mathbf{u}^1 = \left(u_0(x_i)\right)_{1\leq i\leq N_x}.
\end{cases}$$

où $\mathbf{u}_i^n = u_h^n(x_i)$ pour l'espace V_h utilisé ici. Dans le cas général, $\mathbf{u}^n$ contiendra les coefficients de u_h^n dans la base ϕ_i et on parlera de degrés de liberté.

Cette méthode a l'avantage de s'adapter à tous types de géométrie et est ainsi largement utilisée, notamment en mécanique des structures. De nombreux espaces discrets ont été proposés afin de coller au plus près de la solution du modèle continu, on peut citer les éléments finis de Raviart-Thomas permettant d'assurer la continuité du flux entre chaque élément du maillage ([143],[66]), ou bien encore les éléments finis de Nédélec pour l'équation de Maxwell ([125], [15]). Il va sans dire que cette méthode a aussi été analysée par de nombreux auteurs ([58], [165], [173], [28]).

Dans les cas de lois de conservation hyperboliques, comme c'est le cas pour les équations de Navier-Stokes compressibles, la méthode des volumes finis est mieux adaptée ([114], [111], [61], [175], [75], [74]), et est devenue la plus utilisée pour la simulation en mécanique des fluides (CFD). Cette méthode a par ailleurs été étendue pour des équations elliptiques et paraboliques, comme l'équation de la chaleur ([61], [186], [169]). Il existe de nombreuses autres méthodes numériques comme par exemple les méthodes spectrales ([31], [16]) qui reposent sur le développement de la solution u dans une base, telle que les séries de Fourier par exemple, puis à trouver les coefficients dans cette base satisfaisants au mieux l'équation. On peut aussi citer la méthode de Galerkin discontinue ([41], [53]) qui fait le lien entre la méthode des volumes finis et la méthode des éléments finis. En effet, cette méthode consiste à approcher la solution u par un polynôme discontinu, on obtient ainsi une généralisation de la méthode des éléments finis dans le cas où l'on considère un espace V_h discontinu, mais c'est aussi une généralisation de la méthode des volumes finis.

Ainsi les méthodes numériques permettent de passer d'une EDP linéaire à un système linéaire. Cependant afin de rester fidèle à l'EDP, ce système linéaire est souvent de grande dimension. La section suivante présente différentes méthodes permettant de réduire la dimension de ce système linéaire.

1.1.3 Réduction de modèles

Le développement de méthodes numériques a permis de simuler des problèmes de plus en plus complexes. Cependant bien que la puissance de calcul et la capacité mémoire des ordinateurs aient considérablement augmenté au fil des années, certaines simulations numériques requièrent un temps de calcul déraisonnable. En effet, en $3D$ un maillage peut rapidement comporter un million de nœuds, ce qui demande alors de stocker et d'inverser une matrice de taille un million par un million. De plus, à partir de 2005, l'augmentation de la performance des ordinateurs ne fut souvent possible qu'en augmentant le nombre de cœurs par processeur ([55]). Cette problématique a ainsi

développé le calcul scientifique parallèle ([119]), qui consiste à répartir les calculs ainsi que le stockage en mémoire sur plusieurs processeurs. Il faut ainsi gérer les échanges entre chaque processeur mais aussi partager les calculs de manière égale. De nombreuses méthodes de décomposition de domaine ont alors été proposées afin de mettre au point des codes de calculs parallélisables et optimiser la répartition sur chaque processeur ([55], [140]). La parallélisation a alors permis de rendre abordable la simulation de problèmes très complexes, dont notamment des problèmes combinant plusieurs modèles (on parle de simulation multi-physiques) et ce, avec une grande précision.

Beaucoup de modèles en ingénierie dépendent de deux voire trois variables à savoir l'espace, le temps et un paramètre (physique, géométrique, etc). En ce qui concerne la discrétisation de l'espace, le nombre de nœud du maillage peut vite devenir très grand. Soit parce que l'on s'intéresse à un problème sur un grand domaine, soit parce que l'on désire avoir une simulation très précise et qu'on utilise donc un maillage très fin. Pour ce qui est de la discrétisation du temps, là aussi le nombre d'itérations peut rapidement devenir important. C'est le cas, lorsque l'on simule un phénomène sur une longueur période ou lorsque l'on simule un problème non-linéaire. Ainsi on augmente considérablement le nombre de calculs à réaliser. Enfin, il arrive aussi que l'on souhaite simuler plusieurs fois le même phénomène mais en faisant varier des paramètres géométriques. Comme par exemple, en optimisation de formes où l'on va chercher à concevoir un objet de façon à satisfaire une condition. Mis bout à bout les simulations deviennent très coûteuses, et l'on se retrouve alors à choisir entre une simulation de mauvaise qualité mais obtenue rapidement, et une simulation précise mais très coûteuse.

C'est dans ce contexte que sont apparus les modèles réduits, dont le but est de chercher un compromis entre la précision de la simulation et le temps de calcul.

1.1.3.1 POD-Galerkin

La méthode POD-Galerkin permet de construire un modèle réduit. Cette méthode se décompose en deux étapes, dans la première on cherche à construire un espace basse dimension à l'aide de la *méthode de décomposition orthogonales aux valeurs propres* (POD, [158]). Pour cela on doit résoudre l'équation pour une condition initiale fixée, $u_0(\boldsymbol{x})$. Dans la seconde, on projette la solution de l'EDP sur cet espace basse dimension à l'aide de la méthode de Galerkin. Ainsi, on obtient un modèle réduit nous permettant de résoudre l'EDP pour une condition initiale différente et à moindre coût.

On présentera dans cette section une application de cette méthode à l'équation de la chaleur, définie par :

$$\begin{cases} \dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = \kappa\,\Delta u(\boldsymbol{x},t), & \forall(\boldsymbol{x},t) \in \Omega \times (0,T), \\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), & \forall \boldsymbol{x} \in \Omega. \end{cases} \tag{1.6}$$

À l'aide d'une méthode numérique, on obtient l'approximation $u_h^n(\boldsymbol{x})$ de la solution du modèle (1.6), pour tout temps t^n.

La première étape consiste à trouver un sous-espace vectoriel de petite dimension W^K de $L^2(\Omega)$, de dimension K, permettant de représenter au mieux chaque fonction $u_h^n(\boldsymbol{x})$. Pour ce faire, on construit la matrice dite de *corrélation* $\mathcal{S}$ définie par :

$$\mathcal{S}_{n,m} := \int_\Omega u_h^n(\boldsymbol{x})\, u_h^m(\boldsymbol{x}) d\boldsymbol{x},$$

puis on la décompose en valeurs propres, i.e.

$$\mathcal{S} = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^T.$$

Finalement on obtient une base $\left\{\phi_k(\boldsymbol{x})\right\}_k$, permettant d'engendrer le sous-espace W^K, par la formule :

$$\phi_k(\boldsymbol{x}) := \lambda_k^{-\frac{1}{2}} \sum_{n=1}^{N_t} u_h^n(\boldsymbol{x})\, \mathbf{v}_k^n.$$

On appelle *modes POD* les fonctions $\phi_k(\boldsymbol{x})$. Par ailleurs, pour tout élément $\chi \in L^2(\Omega)$, sa projection dans l'espace W^K est :

$$\Pi^K \chi := \sum_{k=1}^{K} \mathbf{a}_k(\chi)\, \phi_k,$$

et on appelle *coefficients POD* les coefficients $\mathbf{a}_k(\chi)$.

Afin de construire un modèle réduit permettant de rapidement prédire la température, la deuxième étape de cette méthode consiste à projeter la solution $u(\boldsymbol{x},t)$ dans l'espace W^K, par une méthode de Galerkin. Pour cela considérons la formulation faible semi-discrétisée de l'équation (1.6), i.e.

$$\int_\Omega \frac{u^{n+1}(\boldsymbol{x}) - u^n(\boldsymbol{x})}{\delta t}\, \psi(\boldsymbol{x}) d\boldsymbol{x} + \int_\Omega \kappa \nabla u^{n+1}(\boldsymbol{x}) \cdot \nabla \psi(\boldsymbol{x}) d\boldsymbol{x} = 0, \quad \forall \psi \in V \tag{1.7}$$

En projetant chaque u^n dans le sous-espace W^K, on obtient :

$$\sum_{k=1}^{K} \frac{\mathbf{a}_k^{n+1} - \mathbf{a}_k^n}{\delta t} \int_\Omega \phi_k(\boldsymbol{x})\, \psi(\boldsymbol{x}) d\boldsymbol{x} + \sum_{k=1}^{K} \mathbf{a}_k^{n+1} \int_\Omega \kappa \nabla \phi_k(\boldsymbol{x}) \cdot \nabla \psi(\boldsymbol{x}) d\boldsymbol{x} = 0, \quad \forall \psi \in V \tag{1.8}$$

où l'on note

$$\mathbf{a}^n := \mathbf{a}(u^n).$$

Finalement en posant $\psi = \phi_l$, on obtient un système linéaire de taille K, appelé *modèle réduit*, permettant de rapidement calculer les coefficients $\mathbf{a}^n$:

$$\left(\frac{1}{\delta t}\, \mathbf{M}_r + \mathbf{K}_r\right) \mathbf{a}^{n+1} = \frac{1}{\delta t}\, \mathbf{M}_r\, \mathbf{a}^n,$$

avec les matrices, de masse et de rigidité, basses dimensions définies par :

$$\Big(\mathbf{M}_r\Big)_{k,l} := \int_\Omega \phi_l(\boldsymbol{x}) \phi_k(\boldsymbol{x}) d\boldsymbol{x}, \qquad \Big(\mathbf{K}_r\Big)_{k,l} := \int_\Omega \kappa \nabla \phi_l(\boldsymbol{x}) \cdot \nabla \phi_k(\boldsymbol{x}) d\boldsymbol{x}.$$

Remarque 2. *Une fois les coefficients $\mathbf{a}^n$ calculés par le modèle réduit, on obtient la température u_r^n en utilisant les modes POD ϕ_k, i.e.*

$$u_r^n(\boldsymbol{x}) := \sum_{k=1}^{K} \mathbf{a}_k^n\, \phi_k(\boldsymbol{x}).$$

1.1.3.2 Méthode des bases réduites (RB)

La méthode des bases réduites a été introduite à la fin des années 70 en analyse non-linéaire des structures, puis fut développer durant les années 80-90. Nous présentons dans cette section la version introduite en 2002 par Prud'homme & al ([138], [139], [144], [85]) qui présente l'avantage de fournir une estimation d'erreur a posteriori. Le cadre d'application classique est celui d'une EDP elliptique dépendant d'un paramètre. On cherche alors à construire un modèle réduit permettant de prédire en temps réel la solution de cette EDP, que l'on notera $u(\cdot; \boldsymbol{\mu})$ pour un paramètre $\boldsymbol{\mu}$ donné. Cette méthode se décompose en deux phases, la première dite *OFFLINE* consistant à construire une base réduite ainsi qu'à construire le modèle réduit, et une seconde dite *ONLINE* consistant à calculer la solution $u(\cdot; \boldsymbol{\mu})$ pour un paramètre $\boldsymbol{\mu}$ donné.

Dans cette méthode, la base réduite est construite à partir de solutions de l'EDP pour des paramètres particuliers. Le sous-espace de petite dimension est ici défini par, $V_{rb} := Span\{u(\cdot; \boldsymbol{\mu}_k), \quad k = 1, \cdots, K\}$. Le choix de ce sous-espace est motivé par des

résultats théoriques sur l'espace des solutions de cette EDP pour tout paramètre $\boldsymbol{\mu}$ ([42]). Cependant, le choix de ces paramètres est primordial, et pour ce faire deux principales méthodes ont été proposées. La première s'appuie sur la méthode POD et consiste à considérer des solutions pour un grand nombre de paramètres puis à compresser l'information contenue par toutes ces solutions. La seconde méthode, qualifiée de gloutonne, consiste quant à elle à sélectionner au fur et à mesure les paramètres, en prenant à chaque itération le paramètre pour lequel la solution est la moins bien représentée.

Une fois l'espace V_{rb} construit, il reste à construire un modèle réduit. Cela se fait par la méthode de Galerkin en projetant l'EDP dans cet espace réduit. Cette méthode a connu un fort succès et fut analysée et adaptée à de nombreux types d'EDP ([104], [52], [87], [47], [29], [18]).

1.1.3.3 Méthode d'interpolation empirique (EIM)

Lorsque l'EDP dépend du paramètre de manière non affine, la méthode des bases réduites ne peut plus être employée. Afin d'y remédier, Barrault & al ont introduit en 2004, la méthode d'interpolation empirique (EIM, [13]). Afin de comprendre le problème de dépendance non affine, considérons le cas d'une équation de diffusion stationnaire telle que la conductivité thermique est une fonction de l'espace et d'un paramètre $\boldsymbol{\mu}$, ainsi l'EDP s'écrit

$$-\nabla \cdot \Big(\kappa(\boldsymbol{x};\boldsymbol{\mu})\,\nabla u(\boldsymbol{x};\boldsymbol{\mu})\Big) = 0, \quad \forall \boldsymbol{x} \in \Omega,$$

la formulation variationnelle de ce problème est alors

$$\int_\Omega \kappa(\boldsymbol{x};\boldsymbol{\mu})\,\nabla u(\boldsymbol{x};\boldsymbol{\mu}) \cdot \nabla \psi(\boldsymbol{x}) d\boldsymbol{x} = 0, \quad \forall \psi \in V,$$

si nous étions dans le cas d'une dépendance affine, i.e.

$$\kappa(\boldsymbol{x};\boldsymbol{\mu}) = \sum_{q=1}^{Q} \boldsymbol{\theta}^q(\boldsymbol{\mu})\,\kappa_q(\boldsymbol{x}),$$

on pourrait alors réécrire la formulation variationnelle comme

$$\sum_{q=1}^{Q} \boldsymbol{\theta}^q(\boldsymbol{\mu}) \int_\Omega \kappa_q(\boldsymbol{x})\,\nabla u(\boldsymbol{x};\boldsymbol{\mu}) \cdot \nabla \psi(\boldsymbol{x}) d\boldsymbol{x} = 0, \quad \forall \psi \in V,$$

et ainsi les matrices de rigidité correspondant à chaque conductivité κ_q pourraient être calculées lors de la phase *OFFLINE*. En revanche, lorsque ce n'est pas le cas, nous ne pouvons pas construire un modèle réduit dans la phase *OFFLINE*, car la matrice de masse dépend de manière non affine du paramètre $\boldsymbol{\mu}$. Afin de contourner ce problème, la méthode EIM sélectionne de manière itérative M points $\left(\boldsymbol{x}_m\right)_{1\leq m\leq M}$ ainsi que M fonctions $\left(q_m(\boldsymbol{x})\right)_{1\leq m\leq M}$ permettant d'obtenir une bonne approximation de la conductivité avec un simple produit scalaire de taille M. On a alors

$$\kappa(\boldsymbol{x};\boldsymbol{\mu}) \approx \sum_{m=1}^{M} \beta_m(\boldsymbol{\mu})\, q_m(\boldsymbol{x}),$$

où les β_m sont obtenus en résolvant le système linéaire :

$$\sum_{m'=1}^{M} \beta_{m'}(\boldsymbol{\mu})\, q_{m'}(\boldsymbol{x}_m) = \kappa(\boldsymbol{x}_m;\boldsymbol{\mu}), \qquad \forall 1 \leq m \leq M.$$

Ce qui nous permet alors de rebasculer (à l'erreur d'approximation près) dans un cadre de dépendance affine. Cette méthode a connu un fort succès et s'est avérée très utile dans de nombreuses applications ([117]). Nous utiliserons d'ailleurs une variante de cette méthode dans cette thèse.

1.1.3.4 Décomposition propre généralisée (PGD)

Les deux méthodes présentées ci-dessus nécessitent la connaissance de la solution de l'EDP pour certains paramètres ou conditions initiales. On parle alors de méthode *a posteriori*. Ce n'est pas le cas de la méthode de *décomposition propre généralisée* (PGD), qui fait partie des méthodes de réduction dites *a priori*. En effet, la méthode PGD introduite au milieu des années 80 par Pierre Ladevèze ([102]), consiste à approcher la solution d'une EDP par une somme tensorisée, en séparant chacune des variables ([39], [103], [38]) . Autrement dit, on approche la solution d'une EDP d'évolution $u(\boldsymbol{x},t)$ par :

$$u(\boldsymbol{x},t) \approx \sum_{k=1}^{K} X_k(\boldsymbol{x})\, T_k(t).$$

Les fonctions X_k et T_k sont alors déterminées de manière itérative en remplaçant u dans l'EDP par son approximation à l'itération m, puis en résolvant un problème non-linéaire par la méthode du point fixe.

Cette approche s'avère être très intéressante notamment dans le cas de problème qui requiert un pas d'espace très petit et demande une simulation sur un temps long.

La méthode PGD a également été introduite afin de découpler les variables spatiales, i.e.

$$u(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_d) \approx \sum_{k=1}^{K} X_k^1(\boldsymbol{x}_1)\, X_k^2(\boldsymbol{x}_2) \cdots X_k^d(\boldsymbol{x}_d),$$

mais aussi dans le cas de solutions paramétriques,

$$u(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{k=1}^{K} X_k(\boldsymbol{x})\, T_k(t)\, M_k(\boldsymbol{\mu}),$$

et a ainsi permis d'aborder de nombreux problèmes tels que les simulations multi-échelles, l'optimisation ou bien les problèmes inverses ([37], [39], [77], [6]).

1.1.4 Limites de l'approche basée sur un modèle EDP

Lorsque l'on souhaite simuler un phénomène afin de s'abstenir de nombreuses mesures expérimentales, la première chose à faire est de modéliser ce phénomène par des équations, en général des équations aux dérivées partielles. Pour ce faire, on utilise généralement des modèles existants, et l'on spécifie les paramètres géométriques et physiques correspondant à notre problème. Une fois cette tâche accomplie, il nous faut ensuite construire un modèle discret en utilisant une méthode numérique, et si possible parallélisable. Enfin, les méthodes de réduction sont souvent nécessaires afin de réduire considérablement le temps de calcul. Cela représente ainsi un travail important, par ailleurs à chaque étape on commet des erreurs : erreurs de modélisation, erreurs d'approximation, erreurs numériques et erreurs de réduction. L'approche basée sur un modèle EDP présente donc des limites. Il est donc naturel de se demander si l'on ne pourrait pas automatiser ces tâches en demandant à l'ordinateur d'apprendre directement un modèle à partir de données expérimentales. On aurait alors simplement à générer des données en réalisant quelques expériences, puis l'ordinateur construirait par lui-même un modèle, si possible réduit, permettant de prédire le comportement futur du phénomène. Lorsque les données sont générées par un code de simulation, on parle alors de données synthétiques, l'approche basée sur un modèle EDP présente là aussi des limites. En effet, beaucoup de codes de simulation industrielle fonctionnent comme des boîtes noires, soit car nous n'avons pas accès aux codes sources, soit car le code est très complexe et donc difficile à modifier. Or comme nous l'avons vu, la plupart des méthodes de réduction sont intrusives, c'est-à-dire qu'elles demandent une modification du code de simulation (PGD, bases réduites, etc). Ainsi il est préférable

de n'utiliser que les données générées par le code de simulation afin de construire notre modèle réduit.

L'apprentissage automatique de modèle à partir de données a été étudiée depuis de nombreuses années. C'est pourquoi dans la section suivante nous présenterons le domaine de l'apprentissage automatique ainsi que les principaux algorithmes utilisés.

1.2 Apprentissage automatique

Bien que l'idée de concevoir des machines intelligentes remonte à la nuit des temps, la première utilisation de l'expression *apprentissage automatique* est apparue à la fin des années 50 pour décrire un programme informatique développé par Arthur Samuel. Ce programme consistait à jouer aux Dames, tout en s'améliorant à chaque partie ([123], [148]). Arthur Samuel définit d'ailleurs en 1959 ([72]) l'apprentissage automatique comme étant :

> "la discipline donnant aux ordinateurs la capacité d'apprendre sans qu'il soient explicitement programmés."

Ce n'est que bien plus tard, dans les années 90, que l'apprentissage automatique fut démocratisé, avec son utilisation pour créer un filtre anti-spam apprenant de lui-même ([147]). À partir du XXI^{e} siècle, ce domaine a connu un énorme succès grâce aux résultats impressionnants obtenus ce qui a permis de mettre en lumière toutes les opportunités qu'offrait ce domaine. En 2012, le laboratoire de Google à fait parler de lui en créant un processus d'apprentissage capable de reconnaître des chats sur des vidéos YouTube [40]).

Si l'apprentissage automatique connaît un tel succès aujourd'hui c'est qu'il a permis d'automatiser des tâches, mais aussi de résoudre des problèmes jusque-là inabordables. L'automatisation des tâches ne permet pas seulement de s'éviter un travail laborieux, mais permet aussi de s'adapter au changement sans aucune intervention humaine. L'autre avantage de l'apprentissage automatique est qu'il permet de résoudre des problèmes complexes tels que la reconnaissance vocale ou la météorologie.

Il existe principalement deux manières de mettre en place des algorithmes d'apprentissage automatique. La première est de faire des hypothèses sur les données, et ainsi de construire un modèle, on parle d'*apprentissage par modèle*. La seconde, est de reproduire des choses déjà vues, comme par exemple vérifier si un mail ressemble à un mail déjà identifié comme étant un spam. On parle alors d'*apprentissage par observations*. Nous choisirons de présenter dans cette section quelques principaux algorithmes en les séparant selon leurs classes : supervisés et non-supervisés. Il existe d'autres

classes telles que les algorithmes semi-supervisés ([35], [187]) ou bien l'apprentissage par renforcement ([171], [97]), cependant nous nous restreindrons aux types d'algorithmes utilisés dans cette thèse.

1.2.1 Algorithmes supervisés

Le principe de ce type d'algorithmes est de prédire une sortie y associée à une entrée $\mathbf{x}$, à partir d'un ensemble de couples entrées-sorties, appelé *ensemble d'entrainement*. On notera $\mathbf{x}_{train}^{(i)}$ la i-ème donnée d'entrée d'entrainement, et $y_{train}^{(i)}$ la i-ème donnée de sortie d'entrainement. Ces algorithmes ont deux phases, la première dite d'entrainement ; où l'on ajuste les paramètres de notre modèle, dans le cas d'apprentissage par modèle, ou bien où l'on recopie les données d'entrainement, dans le cas d'apprentissage par observations ; et la seconde dite de prédiction où l'on prédit la valeurs y pour un $\mathbf{x}$ donné. Lorsque la sortie y est quantitative, comme le prix d'une maison par exemple, on parle d'algorithme de régression. Tandis que si la sortie est qualitative on parle alors d'algorithme de classification.

1.2.1.1 Régression

Il existe de nombreux algorithmes de régression, et le choix de l'algorithme est propre au problème d'apprentissage. Voici quelques algorithmes de régressions.

Régression linéaire

Il s'agit de l'algorithme de régression le plus simple ([69]). Son principe consiste à trouver les deux paramètres β_0 et $\boldsymbol{\beta}_1$, minimisant la quantité,

$$\sum_{i=1}^{N_{train}} |y_{train}^{(i)} - \langle \boldsymbol{\beta}_1, \mathbf{x}_{train}^{(i)} \rangle - \beta_0|^2,$$

Cet algorithme est donc un algorithme d'apprentissage par modèle. Bien qu'il soit simple, il peut fournir de bons résultats, notamment lorsque les données suivent une relation linéaire.

Régression par machines à vecteurs de support (SVR)

En utilisant la régression linéaire présentée ci-dessus, on obtient l'hyperplan minimisant la distance moyenne pour toutes les données d'entraînements. Cependant

lorsque certaines données sont aberrantes, on aimerait ne pas en tenir compte. Pour ce faire, pénalisons la norme de $\boldsymbol{\beta}_1$ de façon à maximiser la distance entre l'hyperplan et chaque donnée d'entrainement. Par ailleurs, on minimisera la distance entre chaque donnée d'entrainement $\left(\mathbf{x}_{train}^{(i)}, y_{train}^{(i)}\right)$ et l'hyperplan par le dessus avec $\boldsymbol{\xi}_{(i)}$ et par le dessous avec $\boldsymbol{\xi}_{(i)}^*$. On est alors amener à résoudre le problème d'optimisation :

$$\min_{\beta_0, \boldsymbol{\beta}_1, \boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}_*^{(i)}} \frac{1}{2}\|\boldsymbol{\beta}_1\|_2^2 + C \sum_{i=1}^{N_{train}} \left(\boldsymbol{\xi}_{(i)} + \boldsymbol{\xi}_{(i)}^*\right) \tag{1.9}$$

$$\text{tels que } \begin{cases} y_{train}^{(i)} - \langle \boldsymbol{\beta}_1, \mathbf{x}_{train}^{(i)} \rangle - \beta_0 \leq \boldsymbol{\xi}_{(i)} \\ y_{train}^{(i)} - \langle \boldsymbol{\beta}_1, \mathbf{x}_{train}^{(i)} \rangle - \beta_0 \geq \boldsymbol{\xi}_{(i)}^* \\ \boldsymbol{\xi}_{(i)}, \boldsymbol{\xi}_{(i)}^* \geq 0. \end{cases} \tag{1.10}$$

L'hyperparamètre C permet d'ajuster l'éloignement entre l'hyperplan et les données d'entraînement. Lorsque le nombre de variables prédictives est plus grand que le nombre de données d'entrainement, on préférera résoudre le problème d'optimisation duale :

$$\max_{\boldsymbol{\alpha}_{(i)}, \boldsymbol{\alpha}_{(i)}^*} \frac{1}{2} \sum_{i,j=1}^{N_{train}} \left(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}_{(i)}^*\right)\left(\boldsymbol{\alpha}_{(j)} - \boldsymbol{\alpha}_{(j)}^*\right)\langle \mathbf{x}_{train}^{(i)}, \mathbf{x}_{train}^{(j)} \rangle + \sum_{i=1}^{N_{train}} y_{train}^{(i)}\left(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}_{(i)}^*\right) \tag{1.11}$$

$$\text{tels que } \begin{cases} \sum_{i=1}^{N_{train}} \left(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}_{(i)}^*\right) = 0, \quad \text{et } \boldsymbol{\alpha}_{(i)}, \boldsymbol{\alpha}_{(i)}^* \in [0; C]. \end{cases} \tag{1.12}$$

On prédit alors la sortie y associée à une entrée $\mathbf{x}$ par

$$\sum_{i=1}^{N_{train}} \left(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}_{(i)}^*\right)\langle \mathbf{x}_{train}^{(i)}, \mathbf{x} \rangle + \rho,$$

notons que le terme ρ n'intervient pas dans la formulation duale, plusieurs approches ont alors été proposées pour le déterminer. Cette méthode correspond aux machines à vecteurs de support linéaires ([161], [30]).

Lorsque les données ne satisfont pas une relation linéaire, cette approche ne semble pas être la bonne. On introduit alors des observables non-linéaires des $\mathbf{x}_{train}^{(i)}$ à l'aide de la fonction vectorielle $\boldsymbol{\phi}$. Cependant calculer les observables non-linéaires pour chaque donnée d'entrée d'entrainement est coûteux. En reprenant la formulation duale (1.11), on s'aperçoit que l'on ne doit évaluer que le produit scalaire entre chaque donnée d'entrée d'entrainement. Ainsi on considèrera le produit scalaire entre chaque donnée d'entrée d'entrainement dans l'espace augmenté par les observables non-linéaires. De plus, on ne spécifiera pas ces observables mais simplement le noyau correspondant à ce

produit scalaire, que l'on définira par :

$$k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle.$$

Le noyau k doit vérifier certaines conditions dictées par le théorème de Mercer ([124]), il est classique d'utiliser un noyau *Radial Basis Function* (RBF), défini par

$$k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x}-\mathbf{x}'\|^2}.$$

Cette approche repose sur la théorie des espaces de Hilbert à noyaux reproduisant (RKHS), et est appelée *kernel trick* ([32], [155], [153]). Il nous faut alors comme précédemment résoudre le problème d'optimisation :

$$\max_{\boldsymbol{\alpha}_{(i)}, \boldsymbol{\alpha}^*_{(i)}} \frac{1}{2} \sum_{i,j=1}^{N_{train}} \Big(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}^*_{(i)}\Big)\Big(\boldsymbol{\alpha}_{(j)} - \boldsymbol{\alpha}^*_{(j)}\Big)\, k\Big(\mathbf{x}^{(i)}_{train}, \mathbf{x}^{(j)}_{train}\Big) + \sum_{i=1}^{N_{train}} y^{(i)}_{train}\Big(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}^*_{(i)}\Big)$$

$$\text{tels que } \begin{cases} \sum\limits_{i=1}^{N_{train}} \Big(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}^*_{(i)}\Big) = 0, \quad \text{et } \boldsymbol{\alpha}_{(i)}, \boldsymbol{\alpha}^*_{(i)} \in [0; C]. \end{cases}$$

Et on prédit de même la sortie y associée à une entrée $\mathbf{x}$ par

$$\sum_{i=1}^{N_{train}} \Big(\boldsymbol{\alpha}_{(i)} - \boldsymbol{\alpha}^*_{(i)}\Big)\, k\Big(\mathbf{x}^{(i)}_{train}, \mathbf{x}\Big) + \rho,$$

D'autres choix de noyaux sont possibles, comme par exemple le noyau polynomial défini par :

$$k(\mathbf{x}, \mathbf{x}') = \Big(\gamma \, \langle \mathbf{x}, \mathbf{x}' \rangle + r\Big)^d.$$

Réseau de neurones artificiels (ANN)

Les réseaux de neurones artificiels appartiennent eux aussi à la classe des algorithmes d'apprentissage par modèle. Une des utilisations courantes des réseaux de neurones est la régression. Leur principe est inspiré du fonctionnement des neurones biologiques. En effet, un réseau de neurones artificiel est composé de plusieurs couches de neurones ([127]). On distingue trois types de couches :

— **La couche d'entrée** :
Dans cette couche le nombre de neurones est égal à la dimension de l'entrée $\mathbf{x}$. Aucun calcul n'est réalisé dans cette couche, elle ne sert qu'à introduire l'entrée $\mathbf{x}$ dans le réseau.

— **Les couches cachées** :
Ces couches représentent le cœur du réseau de neurones. À chaque neurone de ces couches on associe une fonction dite d'*activation* noté σ. Plaçons-nous dans la j-ème couche cachée, et considérons le i-ème neurone de cette couche que l'on notera $\mathbf{n}_i^j$. Notons N_j le nombre de neurones de cette couche. La sortie de ce neurone, que l'on notera $\mathbf{s}_i^j$ sera alors

$$\sigma\Big(\sum_{k=1}^{N_{j-1}} \mathbf{w}_{i,k}^j \, \mathbf{s}_k^{j-1} + \mathbf{b}_i^j\Big),$$

où $\mathbf{w}$ et $\mathbf{b}$ sont respectivement les poids et les biais du réseau et sont des paramètres à optimiser.

— **La couche de sortie** :
Cette couche contient un neurone et renvoie la somme pondérée des sorties de la dernière couche cachée.

Les couches cachées et le nombre de neurones qui les constituent définissent l'architecture du réseau de neurones. C'est d'ailleurs des hyperparamètres à fixer avant l'entrainement du modèle. De même, pour les fonctions d'activations σ, où l'on choisit généralement une fonction `sigmoide` définie par :

$$\sigma(x) = \frac{1}{1 + e^{-\lambda x}},$$

ou bien `ReLu`, lorsque l'on souhaite réduire le temps d'entrainement, définie par

$$\sigma(x) = \begin{cases} 0, & \text{si } x \leq 0, \\ x, & \text{sinon} \end{cases}$$

En revanche les poids et biais sont optimisés durant la phase d'entrainement afin de minimiser une certaine fonctionnelle. L'optimisation des poids et des biais est généralement réalisée par une descente de gradient, en utilisant la méthode de *rétropropagation* du gradient ([146], [109]).

Une des raisons du succès des réseaux de neurones est qu'ils permettent de représenter un grand nombre de fonctions avec une très grande précision. Cybenko a d'ailleurs démontré en 1989, que l'espace des réseaux de neurones à une seule couche cachée et avec une fonction d'activation sigmoidale est dense dans l'espace des fonctions continues ([46]).

Notons enfin que la régression linéaire, est un cas particulier de réseau de neurones. Reprenons la prédiction faite à l'aide d'une régression linéaire. Étant donné une entrée

$\mathbf{x}$, on prédit alors la sortie y par :

$$\langle \boldsymbol{\beta_1}, \mathbf{x} \rangle + \beta_0.$$

Cette sortie peut être modélisée par le réseau de neurones ci-dessous :

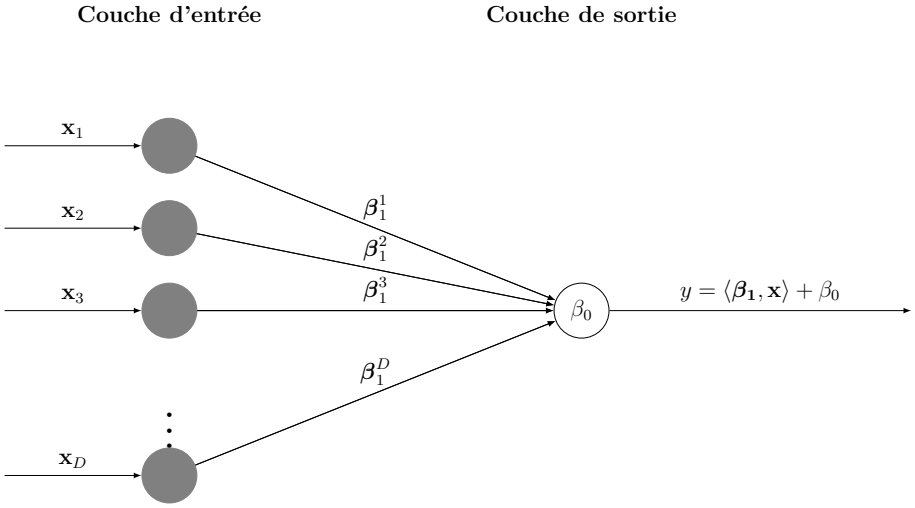


FIGURE 1.1 Schéma d'une régression linéaire par un réseau de neurones

Méthode des k plus proches voisins (kNN)

La méthode des k plus proches voisins fait en revanche partie des méthodes d'apprentissage par observations. La prédiction d'une sortie y associée à une entrée $\mathbf{x}$ se fait en deux phases. Lors de la première phase, on sélectionne les k entrées dans l'ensemble d'entrainement étant les plus proches de $\mathbf{x}$. Puis dans une seconde phase, on effectue la moyenne des k sorties associées aux k entrées sélectionnées.

La sélection des k entrées est en général coûteuse. L'algorithme *Brute Force* qui consiste à calculer la distance entre chaque entrée de l'ensemble d'entrainement et l'entrée $\mathbf{x}$ a une complexité en $\mathcal{O}(DN_{train}^2)$ avec D la dimension de $\mathbf{x}$. Afin de réduire ce temps de calcul, plusieurs alternatives ont été proposées telles que l'algorithme *K-D Tree* ([14], [17]) ou l'algorithme *Ball Tree* ([131]). En ce qui concerne la seconde phase, il peut être pertinent de pondérer la moyenne par l'inverse des distances entre les entrées sélectionnées et l'entrée $\mathbf{x}$.

Bien que cette méthode soit simple, elle donne en général de bons résultats, notamment lorsque les données d'entrainement recouvrent bien l'espace.

1.2.1.2 Classification

Dans le cas d'une classification, la sortie y est appelée *label* et est un entier correspondant au numéro d'une classe. On souhaite alors attribuer à un individu une classe à partir de ses *caractéristiques* contenues dans un vecteur $\mathbf{x}$. Comme dans le cas d'une régression, nous disposons pour cela d'un ensemble de couples entrées-sorties que l'on appelle ensemble d'entrainement ([99]).

La plus part des algorithmes de régression définis précédemment peuvent être utilisés pour résoudre un problème de classification avec quelques légères modifications, nous présenterons ces quelques modifications.

Classification par machines à vecteurs de support (SVC)

C'est le cas des machines à vecteurs de support. Considérons le cas d'une classification binaire, et où y vaut soit 1 soit -1. Le problème primal pour les machines à vecteurs de support linéaires restera le même que le problème (1.9), excepté la contrainte qui deviendra :

$$\begin{cases} y_{train}^{(i)}\Big(\langle \boldsymbol{\beta}_1, \mathbf{x}_{train}^{(i)} \rangle + \beta_0\Big) \geq 1 - \boldsymbol{\xi}^{(i)}, \\ \boldsymbol{\xi}^{(i)} \geq 0, \quad \forall 1 \leq i \leq N_{train}. \end{cases}$$

Ainsi comme dans le cas d'une régression, on cherchera à séparer les données d'entrée d'entrainement $\mathbf{x}_{train}^{(i)}$ par un hyperplan de sorte que les $\mathbf{x}_{train}^{(i)}$ soient le plus éloignés possible de l'hyperplan, tout en commettant le moins d'erreurs.

Réseau de neurones (ANN)

En ce qui concerne les réseaux de neurones, quelques changements sont nécessaires dans la couche de sortie. Tout d'abord, il faudra prendre autant de neurones que de classes, excepté dans le cas binaire où un seul neurone suffit. Puis appliquer une fonction `softmax`, noté σ, à la sortie de chaque neurone. De sorte que, si l'on note $\mathbf{s}$ le vecteur contenant les sorties de N neurones de la couche de sortie (N étant ici le nombre de classes), la sortie du j-ème neurone vaudra :

$$\begin{cases} 1, & \text{si} \quad \sigma(\mathbf{s})_j > 0.5, \\ 0, & \text{sinon}, \end{cases}$$

avec la fonction `softmax` σ définie par

$$\sigma(\mathbf{s})_i = \frac{e^{\mathbf{s}_i}}{\sum_{j=1}^{N} e^{\mathbf{s}_j}}, \qquad \forall 1 \leq i \leq N.$$

Méthode des k plus proches voisins

Enfin, en ce qui concerne les k plus proches voisins, on sélectionnera les k individus les plus proches de l'entrée $\mathbf{x}$ que l'on souhaite classer ([45]). L'individu ayant comme caractéristique $\mathbf{x}$, sera ensuite assigné à la classe majoritaire des k individus sélectionnés.

1.2.2 Algorithmes non-supervisés

Pour ce groupe d'algorithmes, on ne dispose que des entrées $\mathbf{x}_i$. Les applications sont alors de natures différentes, on cherchera principalement à analyser nos données. Lorsqu'on cherchera à regrouper nos données à partir de similarités, on parlera de partitionnement (*clustering* en anglais). Et lorsqu'on cherchera à réduire nos données pour conserver l'information contenue dans celles-ci tout en réduisant leur volume de stockage, on parlera de réduction de dimension.

1.2.2.1 Partitionnement

Il existe de nombreux algorithmes permettant de partitionner des données ([93]). Pour obtenir un bon partitionnement le but des algorithmes est de maximiser la distance inter-classe tout en minimisant la distance intra-classe. Nous présenterons ici deux algorithmes couramment utilisés.

Algorithme des k-moyennes

Cet algorithme requiert un nombre de classes à spécifier, que l'on notera k. On cherche à répartir les individus en k classes de façon à minimiser la distance entre le barycentre des points d'une classe et les points de cette classe ([91], [92]). En notant S_i l'ensemble des individus appartenant à la i-ème classe, et $\boldsymbol{\mu}_i$ le barycentre des points de la i-ème classe (aussi appelé centroïde), cela revient à résoudre le problème

d'optimisation :

$$\underset{(S_1,S_2,\cdots,S_k)}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2.$$

Un des algorithmes pour résoudre ce problème d'optimisation est :

- **Initialisation**
  Choisir aléatoirement k points $\mathbf{x}_i$ afin de définir les k centroïdes $\boldsymbol{\mu}_i^{(1)}$.
- **Répéter tant que** $\min\limits_{(S_1,S_2,\cdots,S_k)} \sum\limits_{i=1}^{k} \sum\limits_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 > tol$
  — On définit la classe i à l'itération l, par

$$S_i^{(l)} := \{\mathbf{x}_j : \quad \|\mathbf{x}_j - \boldsymbol{\mu}_i^{(l)}\| \leq \|\mathbf{x}_j - \boldsymbol{\mu}_{i'}^{(l)}\|, \quad \forall i' \neq i.\}.$$

  — On définit le centroïde associé à la classe i à l'itération $l+1$, par

$$\boldsymbol{\mu}_i^{(l+1)} := \frac{1}{\#S_i^{(l)}} \sum_{\mathbf{x}_j \in S_i^{(l)}} \mathbf{x}_j.$$

L'initialisation a une importance cruciale dans le bon choix des classes, aussi l'algorithme *k-moyennes++* ([8]) propose une approche probabiliste du choix des centroïdes initiaux. La norme choisie a aussi une importance, elle permet de mesurer correctement la similarité entre les individus représentés par leurs caractéristiques $\mathbf{x}_i$.

Partitionnement spectral

L'exemple classique montrant les limites du partitionnement par l'algorithme des k-moyennes est le cas où les points sont alignés le long de deux cercles de rayons différents mais avec un centre commun. La figure ci-dessous présente cet exemple :

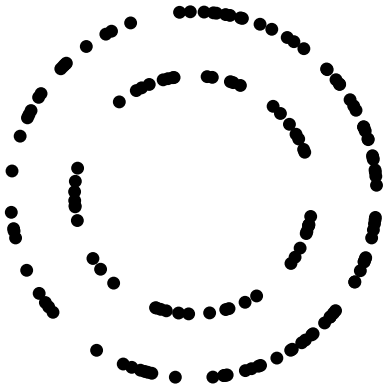


FIGURE 1.2 Points alignés le long de deux cercles différents

On aimerait ainsi partitionner les points en fonction du cercle sur lequel ils sont alignés. Seulement en utilisant l'algorithme des k-moyennes on ne peut pas prendre en compte ce genre de structure. L'algorithme se contentera de séparer les points par un hyperplan. Pour remédier à cela, il nous faudrait changer de repère de façon à ce que la structure apparaisse dans les coordonnées de chaque point. On aurait ensuite qu'à partitionner les points dans le nouveau repère à l'aide de l'algorithme des k-moyennes. C'est justement le principe de fonctionnement du partitionnement spectral ([179], [126]).

L'idée est de construire un graphe permettant de relier les points à regrouper. Par exemple, le graphe des k plus proches voisins connecte deux nœuds i et j si le point $\mathbf{x}_j$ fait partie des k plus proches voisins du point $\mathbf{x}_i$. Une fois le graphe construit, on le représente à l'aide de sa matrice Laplacienne, noté $\mathbf{L}$. Puis on décompose la matrice $\mathbf{L}$ en valeurs propres, i.e.

$$\mathbf{L} = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^T.$$

On conserve ensuite les K vecteurs propres $\mathbf{v}_k$ correspondant aux K plus grandes valeurs propres λ_k dans la matrice $\mathbf{V}_r \in \mathbb{R}^{N\times K}$, avec N le nombre de points à partitionner. Les points définis dans le nouveau repère sont alors les lignes de $\mathbf{V}_r$, i.e.

$$\mathbf{y}_i = \left(\mathbf{V}_r\right)_{i,\cdot}.$$

On partitionne alors ces points en k classes $C_1, \cdots, C_k$, à l'aide de l'algorithme des k-moyennes. Et finalement, les classes renvoyées par l'algorithme de partitionnement spectral, notées S_i, sont définies par

$$S_i := \{j \quad \mathbf{y}_j \in C_i\}.$$

Algorithme des k-médoïdes

Un autre inconvénient à l'algorithme des k-moyennes, est sa sensibilité aux données aberrantes. En effet, chaque centroïde est construit comme étant le barycentre des éléments d'une même classe. Un moyen d'éviter cela et ainsi de rendre l'algorithme plus robuste, est de considérer comme élément central de la classe, celui ayant le moins de dissimilitude avec les autres. Cet élément est appelé médoïde. Ainsi contrairement à l'algorithme des k-moyennes, l'élément central d'une classe est présent dans les données à partitionner.

1.2.2.2 Réduction de dimension

Une autre application de l'apprentissage non-supervisé est la réduction de dimension. Lorsque l'on souhaite entrainer un modèle de régression ou de classification, la dimension D des données d'entrée d'entrainement est parfois importante. Cela occasionne un stockage et un temps d'entrainement important. De plus, il nous faudrait un grand nombre de données d'entrainement afin de couvrir l'espace contenant l'ensemble des entrées possibles. On parle alors du *fléau de la dimension*. Cependant, les données sont souvent corrélées, et occupent donc en réalité un espace beaucoup moins important. Ainsi les méthodes de réduction projettent les données dans un espace de petite dimension tout en conservant le plus d'information possible. On peut distinguer deux types de méthodes de réduction, à savoir les méthodes de *sélection de caractéristiques* et les méthodes d'*extraction de caractéristiques*

Analyse en composantes principales (PCA)

Considérons la matrice $\mathbf{X}$ contenant l'ensemble des entrées $\mathbf{x}_i$ définie comme,

$$\mathbf{X} := \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \\ | & | & & | \end{bmatrix},$$

on notera N le nombre de ses entrées et D la dimension de ses entrées. D étant très grand, on souhaiterais trouver un vecteur $\mathbf{w} \in \mathbb{R}^D$, tel que pour chaque individu i on

ait

$$\mathbf{x}_i \approx t_i \, \mathbf{w}, \quad \text{avec} \quad t_i := \langle \mathbf{x}_i, \mathbf{w} \rangle \quad \text{par définition de la projection orthogonale.}$$

Ainsi au lieu de conserver pour chaque individu un vecteur de taille D, on ne conservera qu'un scalaire t_i. Seulement, il faut que ce vecteur $\mathbf{w}$ soit représentatif de tous les individus. Cela revient alors à chercher un vecteur $\mathbf{w}$ pour lequel les coefficients t_i sont les plus différents possible. Ainsi on aura en quelque sorte préservé la diversité de l'information contenue dans les entrées $\mathbf{x}_i$.

Statistiquement, cela revient à maximiser la variance du vecteur $\mathbf{t}$ contenant tous les coefficients t_i. Pour ce faire, commençons par centrer nos entrées, en modifiant la matrice $\mathbf{X}$,

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 - \bar{\mathbf{x}}_1 & \mathbf{x}_2 - \bar{\mathbf{x}}_2 & \cdots & \mathbf{x}_N - \bar{\mathbf{x}}_N \\ | & | & & | \end{bmatrix},$$

avec $\bar{\mathbf{x}}_i$ la moyenne de $\mathbf{x}_i$. Ainsi on cherchera simplement à maximiser la norme du vecteur $\mathbf{t}$. De plus, sans perte de généralité cherchons un vecteur $\mathbf{w}$ unitaire. Le problème d'optimisation devient alors :

$$\mathbf{w}^* = \underset{\|\mathbf{w}\|=1}{\arg\max} \, \|\mathbf{t}\|^2 = \underset{\|\mathbf{w}\|=1}{\arg\max} \sum_{i=1}^{N} |\langle \mathbf{x}_i, \mathbf{w} \rangle|^2.$$

En réécrivant ce problème, on obtient :

$$\mathbf{w}^* = \underset{\|\mathbf{w}\|=1}{\arg\max} \, \|\mathbf{X}^T \, \mathbf{w}\|^2 = \underset{\|\mathbf{w}\|=1}{\arg\max} \, \mathbf{w}^T \, \mathbf{X} \, \mathbf{X}^T \, \mathbf{w}$$

Par définition du quotient de Rayleigh, on retrouve alors le premier vecteur propre de la matrice $\mathbf{C} := \mathbf{X} \, \mathbf{X}^T$, qui n'est rien d'autre que la matrice de covariance.

Si projeter les données dans un espace à une dimension ne suffit pas, on peut exhiber un deuxième vecteur orthogonal à $\mathbf{w}$ et ainsi de suite. En fait, l'espace de dimension K maximisant la variance est celui engendré par les K premiers vecteurs propres de la matrice de covariance, et l'on notera $\mathbf{W}$ la matrice les contenants.

Les étape de l'analyse en composantes principales sont donc ([95]) :

- **Centrer les données**
- **Construire la matrice de covariance** :
  $\mathbf{C} := \mathbf{X} \, \mathbf{X}^T$

- **Calculer les K premiers vecteurs propres de la matrice $\mathbf{C}$** :
  En réalisant la décomposition en valeurs propres $\mathbf{C} = \mathbf{W}\,\mathbf{\Lambda}\,\mathbf{W}^T$, puis en ne conservant que les K colonnes de la matrice $\mathbf{W}$ associées aux K plus grandes valeurs propres λ_k.
- **Projeter les données** :
  La nouvelle matrice de données est alors $\mathbf{T} = \mathbf{W}^T\,\mathbf{X}$.

Cette méthode est largement utilisée en apprentissage automatique. Des variantes ont été proposées dont la PCA randomisée, afin d'accélérer le calcul des vecteurs propres, [183]. Et la PCA incrémentale, qui évite d'utiliser toutes les données en même temps pour la génération des composantes principales, [7]). Enfin cette méthode a été étendu au cas non-linéaire à l'aide du *kernel trick* ([154]).

Méthode de sélection de sous-ensembles de colonnes (CSSP)

Le stockage massif de données dans le contexte du Big Data, a induit le développement de nombreux algorithmes permettant un stockage moins coûteux sans pour autant perdre l'information. La décomposition en valeurs singulières (SVD) a longtemps été la méthode la plus employée notamment pour son optimalité. Cependant, cette méthode présente l'inconvénient de ne pas, en général, conserver la sparsité de la matrice. Par ailleurs, les données stockées sont peu interprétables. Afin de conserver la sparsité, et l'interprétabilité de la matrice réduite, de nombreux auteurs se sont intéressés à trouver un sous-ensemble de colonnes, que l'on notera S, tel que l'erreur

$$\|\mathbf{U} - \mathbf{C}\,\mathbf{U}_S\|_F, \tag{1.13}$$

soit minimale ([56],[22]), où $\mathbf{U}_S$ est la matrice contenant les colonnes sélectionnées, et $\mathbf{C}$ une matrice permettant de reconstruire les données.

L'idée étant de sélectionner les colonnes, soit selon une méthode probabiliste ([51], [82]), ce qui requiert un grand nombre de colonnes sélectionnées afin d'obtenir une erreur (1.13) raisonnable. Soit de choisir de façon déterministe les colonnes ce qui entraîne un temps de calcul assez long. Nous choisirons ici de présenter deux algorithmes efficaces et peu couteux.

- **Greedy Column Subset Selection**
  Afin d'obtenir une erreur de reconstruction petite même pour un petit nombre de colonnes sélectionnées, tout en conservant une complexité raisonnable, Farahat & al ont introduit l'algorithme *Greedy Column Subset Selection* (GCSS, [63]). L'idée est de choisir de façon gloutonne l'indice des colonnes sélectionnées de

façon à minimiser la fonctionnelle $J(S) := \|\mathbf{U} - \mathbf{C}\,\mathbf{U}_S\|_F$. L'évaluation de cette fonctionnelle étant coûteuse, les auteurs proposent de calculer cette fonctionnelle de façon itérative. Ainsi cette algorithme s'inscrit dans le même esprit que la méthode EIM.

- **Near-Optimal Column Selection**
  Boutsidis & al, introduisent un algorithme probabiliste basé sur un échantillonnage adaptatif ([51]), qui permet de sélectionner de façon rapide des colonnes tout en assurant une reconstruction précise et ce même pour un petit nombre de colonnes sélectionnées ([21]). Par ailleurs, on a l'estimation d'erreur suivante :
  Théorème 1 (Boutsidis & al ([21]))**.** *Soit une matrice* $\mathbf{U} \in \mathbb{R}^{m\times n}$, $\mathbf{U}_S \in \mathbb{R}^{m\times K}$ *la matrice contenant les* K *colonnes sélectionnées par l'algorithme NOCS, on a alors :*
  $$\mathbb{E}\|\mathbf{U} - \mathbf{U}_S\mathbf{U}_S^{\dagger}\mathbf{U}\|_F^2 \leq \Big(1+\varepsilon\Big)\|\mathbf{U} - \mathbf{U}_k\|_F^2$$
  avec $\mathbf{U}_k$ *la décomposition en valeurs singulières de* A *tronquée à l'ordre* k *et* $\varepsilon = \frac{2k}{K}\Big(1 + o(1)\Big)$.
  Ce résultat nous assure ainsi une quasi-optimalité de l'algorithme..

1.3 Simulation numérique à partir de données

Lorsque l'on s'affranchit du modèle, en considérant simplement des données, on passe alors d'un formalisme continu à un formalisme discret. Aussi, la solution à notre modèle EDP ne sera plus la fonction $u(\boldsymbol{x}, t)$ mais la matrice $\mathbf{U} \in \mathbb{R}^{N_x\times N_t}$ avec N_x le nombre de degrés de liberté et N_t le nombre de pas de temps.

1.3.1 Apprentissage de solution spatio-temporelle

Pour la méthode POD-Galerkin, nous avions réduit la dimension à l'aide de la méthode POD puis avions construit notre modèle réduit à l'aide d'une projection de Galerkin. Ce n'est ici bien-sûr pas possible, n'ayant pas accès au modèle EDP.

La grande majorité des travaux portant sur l'apprentissage de solution spatio-temporelle, ont en commun d'utiliser la méthode POD pour construire un espace basse dimension W^K. En effet, la solution discrète $\mathbf{u}^n$, est un vecteur de très grande taille et l'on souhaite éviter le fléau de la dimension. En revanche, ils différent sur la façon de prédire la solution. Certains travaux utilisent des algorithmes issus de l'apprentissage automatique, tels que les réseaux de neurones ou bien les processus gaussiens, tandis que d'autres essayent d'identifier l'EDP.

1.3.1.1 Prédiction des coefficients POD

Lorsque l'on réduit notre matrice de données $\mathbf{U}$, on obtient alors pour chaque temps discret t^n, un vecteur de coefficients POD, $\mathbf{a}^n$. L'idée est alors de chercher à partir de l'historique des coefficients POD $\mathbf{a}^n$ à en prédire le futur. Plusieurs auteurs se sont alors intéressés à l'apprentissage d'une fonction $\hat{\boldsymbol{f}}$ telle que :

$$\mathbf{a}^{n+1} = \hat{\boldsymbol{f}}(\mathbf{a}^n, \mathbf{a}^{n-1}, \cdots, \mathbf{a}^{n-p}),$$

Xiao & al ont utilisés une interpolation RBF afin d'apprendre cette fonction et ainsi obtenir un modèle réduit pour l'équation de Navier-Stokes ([185]). Maulik & al ont proposé l'utilisation d'un réseau de neurones *Long Short Term Memory* (LSTM) et d'un réseau de neurones *Neural Ordinary Differential Equations* (NODE) afin d'apprendre cette fonction $\hat{\boldsymbol{f}}$ pour l'équation de Burger visqueuse ([122]). Enfin, Guo & al utilisent un Processus Gaussien afin d'apprendre cette fonction, leur travail est appliqué à l'équation de Burger visqueuse ainsi qu'à l'équation de Navier-Stokes incompressible ([81]).

1.3.1.2 Identification de l'EDP : Méthode DMD

L'autre approche consiste à identifier l'EDP, en retrouvant les vecteurs propres et valeurs propres de l'opérateur. Présentons ainsi la méthode *Décomposition en modes dynamiques* (DMD) introduite en 2010 par Schmid ([151]), puis proposée dans sa forme actuelle par Tu & al en 2013 ([176], [101]). Cette méthode a été introduite dans le cas d'EDP d'évolution linéaire.

Considérons les matrices $\mathbf{X}$ et $\mathbf{X}'$ définies par

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^{N_t-1} \\ | & | & & | \end{bmatrix} \quad \text{et} \quad \mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{u}^2 & \mathbf{u}^3 & \cdots & \mathbf{u}^{N_t} \\ | & | & & | \end{bmatrix}$$

On cherche alors la matrice $\mathbf{A}$ satisfaisant $\mathbf{A}\,\mathbf{X} = \mathbf{X}'$. Pour ce faire, la première étape est de décomposer en valeurs singulières la matrice $\mathbf{X}$, puis de la tronquer à l'ordre K, de telle sorte que l'on ait :

$$\mathbf{X} \approx \boldsymbol{\Phi}_r\, \boldsymbol{\Sigma}_r\, \mathbf{V}_r^T.$$

La deuxième étape consiste à calculer la matrice $\mathbf{A}$ en utilisant cette décomposition, on obtient ainsi :

$$\mathbf{A} = \mathbf{X}'\, \mathbf{V}_r\, \boldsymbol{\Sigma}_r^{-1}\, \boldsymbol{\Phi}_r^T.$$

Cependant, cette matrice est de très grande taille ($N_x \times N_x$), ainsi il ne semble pas judicieux de la décomposer en valeurs propres. C'est pourquoi, on travaillera avec sa représentation basse dimension en la projetant dans la base réduite contenue dans la matrice $\mathbf{\Phi}_r$. On notera alors $\tilde{\mathbf{A}}$ cette matrice, et on la définira par

$$\tilde{\mathbf{A}} := \mathbf{\Phi}_r^T \, \mathbf{A} \, \mathbf{\Phi}_r = \mathbf{\Phi}_r^T \, \mathbf{X}' \, \mathbf{V}_r \, \mathbf{\Sigma}_r^{-1}.$$

À présent nous pouvons analyser spectralement la représentation basse dimension de la matrice $\mathbf{A}$, en effectuant la décomposition en valeurs propres :

$$\tilde{\mathbf{A}} = \mathbf{W} \, \mathbf{\Lambda} \, \mathbf{W}^T,$$

et les modes DMD (aussi appelé modes exactes car ce sont les modes de l'opérateur) sont donnés par la formule :

$$\mathbf{\Psi}_r := \mathbf{X}' \, \mathbf{V}_r \, \mathbf{\Sigma}_r^{-1} \, \mathbf{W}.$$

On peut alors prédire la solution au temps t^{n+1} par :

$$\mathbf{u}^{n+1} := \mathbf{\Psi}_r \, e^{\mathbf{\Omega} \, t^{n+1}} \mathbf{b},$$

avec

$$\mathbf{\Omega} := \frac{1}{\delta t} \, ln(\mathbf{\Lambda}), \qquad \mathbf{b} := \mathbf{\Psi}_r^{\dagger} \mathbf{u}^1.$$

Remarque 3. *Cette méthode a été étendue ensuite au cas d'EDP d'évolution non-linéaire en considérant des observables non-linéaires (EDMD,[184]).*

1.3.2 Apprentissage de solution paramétrée

Lorsque l'on souhaite construire un modèle réduit paramétré à partir de données, la méthode des bases réduites n'est pas envisageable car nous ne pouvons pas projeter l'équation sur l'espace basse dimension. De plus, contrairement au cas de solutions spatio-temporelles, il n'existe pas de relation entre une solution pour un paramètre $\boldsymbol{\mu}_n$ et un paramètre $\boldsymbol{\mu}_{n+1}$. Nous n'avons alors pas d'autres choix que de prédire la solution pour un paramètre donné à partir de solutions $u(\cdot; \boldsymbol{\mu}_k)$ calculées durant la phase OFFLINE. Comme précédemment, on ne cherche pas la solution *grande dimension* mais sa représentation *basse dimension* obtenue à l'aide d'une méthode POD ou bien d'une construction gloutonne. Il faudra alors prédire la représentation basse dimension pour un paramètre donné. Chen & al propose l'utilisation d'interpolation RBF afin de

prédire les coefficients POD ([36]), et appliquent cette méthodologie à un problème de diffusion ainsi qu'à l'équation de Navier-Stokes incompressible. Hesthaven & al utilisent quant à eux un réseau de neurones afin de prédire les coefficients POD pour l'équation de Poisson non-linéaire ainsi que pour l'équation de Navier-Stokes incompressible stationnaire. Les paramétrisations considérées sont autant de nature physiques que géométriques.([88]). Enfin Swischuk & al comparent différents algorithmes issues de l'apprentissage automatique, tel que le réseau de neurones, les k-plus proches voisins ou bien encore les arbres de décisions ([172]).

Remarque 4. *Une alternative a été proposée par Chakir & al ([34][33]) qui consiste à calculer la solution $u_H(\cdot;\boldsymbol{\mu})$, où l'indice H fait référence au fait que cette solution a été calculé avec un solveur grossier, puis à projetter cette solution grossière sur l'espace des bases réduites. Enfin on rectifie cette projection par un post-traitement faisant intervenir les solutions $u(\cdot;\boldsymbol{\mu}_k)$.*

1.4 Contributions

Les données synthétiques étant coûteuses à produire, l'apprentissage d'un modèle réduit est réalisé dans un contexte *Small Data*. Cela représente un frein pour l'entraînement, on doit alors choisir entre un modèle simple avec le risque qu'il soit une mauvaise approximation de l'EDP ou bien un modèle complexe avec le risque qu'il soit peu stable et que sa prédiction se dégrade en temps long. Mais le peu de données est aussi un frein pour la validation, il nous est alors difficile d'évaluer la qualité du modèle réduit appris. Dans cette thèse, nous souhaitons proposer une méthodologie, basée sur la méthode DMD, permettant à la fois d'apprendre un modèle assez complexe pour être fidèle à l'EDP mais aussi interprétable afin de s'assurer de son bon comportement en temps long. Pour ce faire, on utilise la connaissance a priori issue de la physique et de la simulation numérique afin d'obtenir un modèle réduit dont la forme est au plus proche de celle de l'EDP. Bien-sûr tout en restant dans un contexte non-intrusive et donc sans connaître le modèle EDP. Ainsi, nous pouvons apprendre un modèle réduit d'une grande précision. De plus, cela facilite grandement l'interprétation et la validation de ce modèle réduit. On peut alors par une analyse matricielle et spectrale a posteriori du modèle réduit s'assurer du bon comportement en temps long de la prédiction.

C'est dans cet esprit que nous présentons la méthode au **Chapitre 2**, où l'on utilise la méthode POD afin de réduire la dimension. De plus on adaptera la méthode DMD au cas d'EDP d'évolution d'ordre deux en temps, ainsi qu'au couplage EDP-EDO. Dans le **Chapitre 3**, nous étendrons la méthode au cas de système contrôlé par commutation

en utilisant un réseau de neurones artificiel afin d'apprendre le contrôle. Une application au cas d'un commutateur thermique sera présentée. Ce travail a été accepté dans la revue *Comptes Rendus Mécanique* ([62]). L'optimisation de la règle de contrôle à l'aide d'un réseau de neurones pour le commutateur thermique, est présentée en **Annexe B**. Au **Chapitre 4**, on proposera une réduction de dimension par échantillonnage utilisant l'algorithme EIM. Cela permettra alors d'apprendre un modèle réduit dont la solution est interprétable. Enfin au **Chapitre 5**, on proposera une méthode de réduction de tenseurs, ce qui permettra alors d'apprendre un modèle réduit paramétré. Par ailleurs, la réduction proposée pourra être vue comme une nouvelle approximation CUR, et l'application à un problème d'optimisation de formes non-intrusive sera présentée. Enfin, différentes perspectives seront présentées au **Chapitre 6**, tel que l'extension au cas d'EDP d'évolution non-linéaire ou bien le cas de données bruitées. Enfin l'apprentissage d'un modèle réduit utilisant EIM permet d'adapter en direct notre modèle réduit aux données assimilées. Une application d'apprentissage en ligne sera présentée au **Chapitre 6**.

Chapitre 2

Identification d'une EDP d'évolution linéaire à partir de données

Résumé Ce chapitre est consacré à l'identification d'équations aux dérivées partielles d'évolution linéaires ainsi qu'à l'apprentissage d'un modèle réduit, à partir de données synthétiques générées par un solveur dit *haute-fidélité* (HF). La méthode proposée dans ce chapitre repose sur la méthode de *décomposition en modes dynamiques* (DMD) ainsi que sur la méthode de *décomposition orthogonale aux valeurs propres* (POD). Une fois la dynamique identifiée, un modèle réduit sera appris afin de pouvoir rapidement prédire le comportement de la solution. L'adaptation de cette méthode à divers modèles EDP sera présentée. Par ailleurs, les performances théoriques de la méthode seront quantifiées par une analyse numérique. Enfin, les performances numériques seront quant à elles quantifiées sur un problème de diffusion et un problème d'ondes, afin de comparer le comportement de la méthode sur différents phénomènes physiques.

Mots-clés. Data-driven model ; System identification ; Reduced Order Model ; DMD ; POD.

Introduction

Les simulations numériques sont rendues nécessaires dans de nombreuses applications industrielles afin de pouvoir anticiper le comportement de quantités physiques d'intérêts en s'abstenant d'expériences coûteuses. Cependant, ces simulations sont

souvent réalisées par des codes complexes, en raison de la modélisation mathématique et des méthodes numériques utilisées. Cela entraîne une difficulté à explorer le code ainsi qu'un temps de calcul assez long pour obtenir une prédiction. Par ailleurs, dans de nombreuses applications les données de simulations sont obtenues par une *boîte noire*, l'accès au code de simulations n'est alors pas possible. On appellera solveur *haute-fidélité* (HF) cette boîte noire. On souhaite dans ce chapitre, identifier le modèle ayant servit pour la simulation en ayant seulement accès aux données générées par le solveur HF. De plus, on apprendra un modèle réduit qui permettra de prédire rapidement le comportement de la quantité physique d'intérêt.

Un domaine en pleine expansion en apprentissage automatique, est celui de la prédiction de série temporelle, appelée *Time series forecasting* ([24]). Cela consiste à prédire l'évolution d'une quantité au cours du temps. Ces principales applications sont la prédiction des marchés financiers ([10], [80]) et des consommations énergétiques ([1], [50]). La problématique est semblable à celle abordée dans ce chapitre. À la différence près que la quantité que l'on souhaite prédire est ici solution d'une équation différentielle. Deux méthodes sont couramment utilisées pour la prédiction de séries temporelles, à savoir les *réseaux de neurones récurrents* (RNN,[134]), et plus particulièrement leur variante *Long Short Term Memory* (LSTM, [89],[133]), et les *Processus Gaussien* (GP, [25], [141]). Ces deux méthodes présentent l'avantage de prendre en compte l'historique du comportement de la série temporelle afin d'en prédire le futur. La méthode *Dynamic Mode Decomposition* (DMD, [151], [176]) consiste quant à elle à trouver une application linéaire permettant de relier la solution au temps t^{n+1} à la solution au temps t^n. Ainsi on ne cherche pas à apprendre la série temporelle, mais à identifier le modèle mathématique guidant l'évolution de cette série. Les données provenant de simulations de phénomènes physiques, on aimerait que le modèle réduit conserve certaines propriétés physiques, telles que la positivité de la densité ou bien la conservation de la masse. Si cela peut être imposé par la méthode DMD et ses variantes ([181]), c'est en revanche plus compliqué avec les deux autres méthodes. Enfin, l'interprétabilité du modèle permettant la prédiction est importante car elle permet de s'assurer de l'identification du bon phénomène physique. Dans le cas de la méthode DMD, cela peut être effectué grâce à une analyse spectrale a posteriori. Une analyse similaire semble difficilement possible, dans le cas des deux autres méthodes.

Nous présenterons dans ce chapitre une variante de la méthode DMD, mieux adaptée au cas de données simulées. Par ailleurs, dans ce travail nous étendrons la méthode DMD à divers modèles EDP et termes sources. L'identification de la dynamique et du terme source peut s'apparenter à un problème inverse non paramétrique. Enfin,

les données étant générées par un solveur HF et non expérimentales, la méthode de réduction utilisée sera la méthode de *décomposition orthogonale aux valeurs propres* (POD, [178]). Nous présenterons, tout d'abord, le contexte ainsi que la mise en place du solveur HF. On construira dans la seconde partie, un espace basse dimension à l'aide de la méthode POD. Dans une troisième partie, la méthode DMD sera introduite, afin d'identifier la dynamique et d'apprendre un modèle réduit. Une analyse numérique permettra de quantifier théoriquement les performances de la méthode. Différentes extensions de la méthode DMD seront présentées dans la quatrième partie. Enfin, quelques applications numériques viendront appuyer la pertinence de la méthode DMD sur différents phénomènes physiques.

2.1 Contexte et génération des données

De nombreux phénomènes physiques sont modélisés par une *équation aux dérivées partielles* (EDP). La quantité physique d'intérêt que nous appellerons u n'est connue que implicitement à travers l'équation

$$\partial_t u(\boldsymbol{x}, t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x}, t) = 0, \qquad \forall (\boldsymbol{x}, t) \in \Omega \times [0, T]. \tag{2.1}$$

Dans tout ce chapitre, nous supposerons l'opérateur $\mathcal{L}$ linéaire.

La solution analytique u de l'équation (2.1) n'est souvent pas connue. En pratique, on calcule pour chaque temps discret t^n, à l'aide d'un solveur *haute-fidélité* (HF), une approximation $u_h(\cdot, t^n)$, que l'on suppose suffisamment proche de $u(\cdot, t^n)$, afin de pouvoir les confondre. On a accès à cette approximation à travers une matrice notée $\mathbf{U}$ de taille $N_x \times N_t$. La colonne j de cette matrice correspond à l'approximation au j-ème temps discret. Tandis que la ligne i correspond au i-ème degré de liberté. Par ailleurs, pour chaque méthode numérique, il existe une base $\big(\psi_i\big)_{1 \leq i \leq N_x}$ telle que

$$u(\boldsymbol{x}, t^n) \approx \sum_{i=1}^{N_x} \psi_i(\boldsymbol{x}) \mathbf{U}_{i,n} =: u_h(\boldsymbol{x}, t^n) \tag{2.2}$$

Les lignes de la matrice des données $\mathbf{U}$ ne seront alors rien d'autres que les coefficients de $u_h(\cdot, t^n)$ dans la base $\big(\psi_i\big)_{1 \leq i \leq N_x}$.

Dans la suite de ce travail, nous n'aurons accès qu'à la matrice de données U, et nous chercherons à retrouver l'opérateur $\mathcal{L}$ de l'équation (2.1)**, on parlera alors d'identification de la dynamique**.

2.2 Réduction de la dimensionnalité : projection dans l'espace POD

Afin d'obtenir une approximation u_h fidèle à la solution u, le solveur HF a dû utiliser un maillage très fin. En conséquence N_x, le nombre de lignes de la matrice $\mathbf{U}$, est très grand. Cependant, il est courant que l'information contenue dans la matrice $\mathbf{U}$ ne soit pas si riche. Nous allons donc réduire l'information contenue dans la matrice $\mathbf{U}$ à l'aide de la méthode POD (aussi nommée *Principal Component Analysis* (PCA, [95]) en statistiques, ou *Karhunen Loève Expansion* (KL,[3]) en probabilité).
L'idée est de trouver un sous-espace vectoriel W^K de $\mathbb{R}^{N_x}$ de dimension $K \ll N_x$, tel que

$$W^K := \underset{dim(W^K)=K}{\arg\min} \sum_{n=1}^{N_t} \|\Pi^K \mathbf{u}^n - \mathbf{u}^n\|^2, \tag{2.3}$$

avec $\mathbf{u}^n := \mathbf{U}_{\cdot,n}$, et Π^K la projection orthogonale sur W^K. Ce sous-espace vectoriel peut être engendré par une base $\left(\boldsymbol{\phi}_k\right)_{1\leq k\leq K}$, c'est donc cette base que nous cherchons à trouver. On appelle modes POD les éléments de cette base. Pour ce faire, considérons la matrice dite de *corrélation* $\mathcal{S}$ définie par :

$$\mathcal{S}_{n,m} = \int_\Omega u_h(\boldsymbol{x}, t^n) u_h(\boldsymbol{x}, t^m) d\boldsymbol{x}.$$

En utilisant la décomposition (2.2), on a alors

$$\mathcal{S}_{n,m} = \sum_{i,j}^{N_x} \mathbf{U}_{i,n}\mathbf{U}_{j,m} \int_\Omega \psi_k(\boldsymbol{x})\psi_l(\boldsymbol{x}) d\boldsymbol{x} = \mathbf{U}^T\mathbf{M}\mathbf{U}.$$

où $\mathbf{M}$ est la matrice de masse, et on définira le produit scalaire $\langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle_{\mathbf{M}} := \boldsymbol{\phi}^T \mathbf{M} \boldsymbol{\psi}$.
Décomposons à présent $\mathcal{S}$ en valeurs propres, on a alors :

$$\mathcal{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T,$$

où $\boldsymbol{\Lambda}$ est la matrice diagonale contenant les valeurs propres réelles de $\mathcal{S}$, et $\mathbf{V}$ est la matrice orthogonale contenant les vecteurs propres $\mathbf{v}_k \in \mathbb{R}^{N_t}$. Après avoir ordonné les valeurs propres de la plus grande valeur à la plus petite, on tronque à l'ordre K par :

$$\mathbf{V}_r = \mathbf{V}_{\cdot,1\cdots K} \quad \text{et} \quad \boldsymbol{\Lambda}_r = \boldsymbol{\Lambda}_{1\cdots K,1\cdots K}.$$

Et finalement la matrice $\mathbf{\Phi}_r$ orthogonale (pour le produit scalaire $\langle \cdot, \cdot \rangle_{\mathbf{M}}$) et contenant l'ensemble $\left(\boldsymbol{\phi}_k\right)_{1 \leq k \leq K}$ sera obtenue par la formule :

$$\mathbf{\Phi}_r = \mathbf{U}\mathbf{V}_r\mathbf{\Lambda}_r^{-\frac{1}{2}}. \tag{2.4}$$

On a ainsi construit notre sous-espace vectoriel de dimension K sur lequel nous allons projeter les données contenues dans la matrice $\mathbf{U}$. Par ailleurs, comme nous le verrons par la suite, il vérifie bien la condition (2.3).
Il nous reste, à présent, à trouver pour chaque $\mathbf{u}^n$ les coefficients dans la base $\left(\boldsymbol{\phi}_k\right)_{1 \leq k \leq K}$, que l'on notera $\mathbf{a}_k^n$ (que l'on appelle coefficient POD). Par définition de la projection orthogonale, nous avons

$$\mathbf{a}_k^n = \langle \mathbf{u}^n, \boldsymbol{\phi}_k \rangle_{\mathbf{M}}.$$

En utilisant (2.4) et la décomposition en valeurs propres de $\mathcal{S}$, nous avons

$$\mathbf{A} = \mathbf{\Phi}_r^T\mathbf{M}\mathbf{U} = \mathbf{\Lambda}_r^{-\frac{1}{2}}\mathbf{V}_r^T\mathbf{U}^T\mathbf{M}\mathbf{U} = \mathbf{\Lambda}_r^{-\frac{1}{2}}\mathbf{V}_r^T\mathcal{S} = \mathbf{\Sigma}_r\mathbf{V}_r^T, \tag{2.5}$$

où l'on note $\mathbf{A}_{k,n} = \mathbf{a}_k^n$, et $\mathbf{\Sigma}_r := \mathbf{\Lambda}_r^{\frac{1}{2}}$.
On appellera *représentation basse dimension* de $\mathbf{U}$ la matrice $\mathbf{A} \in \mathbb{R}^{K \times N_t}$. Afin de re-basculer dans l'espace $\mathbb{R}^{N_x \times N_t}$, il suffit alors de multiplier à gauche $\mathbf{A}$ par $\mathbf{\Phi}_r$, et on obtient alors l'approximation,

$$\mathbf{U}_r := \mathbf{\Phi}_r\, \mathbf{A} \approx \mathbf{U},$$

et cette approximation est quantifiée par le résultat ci-dessous :

Théorème (Eckart–Young–Mirsky, [57])**.** *En conservant les notations ci-dessus, nous avons :*

$$\Big(\sum_{n=1}^{N_t} \|\mathbf{u}^n - \Pi^K \mathbf{u}^n\|^2 \Big)^{\frac{1}{2}} = \Big(\sum_{k>K} \lambda_k \Big)^{\frac{1}{2}},$$

avec λ_k les valeurs propres de la matrice $\mathcal{S}$.
De plus, nous avons le résultat d'optimalité :

$$\frac{1}{2} \sum_{n=1}^{N_t} \|\mathbf{u}^n - \Pi^K \mathbf{u}^n\|^2 \leq \min_{rank(\mathbf{V})=K} \frac{1}{2} \sum_{n=1}^{N_t} \|\mathbf{u}^n - \mathbf{v}^n\|^2.$$

Dans la suite de ce travail, on appellera erreur de réduction la quantité :

$$\Big(\sum_{n=1}^{N_t} \|\mathbf{u}^n - \Pi^K \mathbf{u}^n\|^2 \Big)^{\frac{1}{2}}$$

.

D'un point de vue pratique, la matrice de corrélation $\mathcal{S}$ sera générée par le solveur HF en même temps que la génération de la matrice des données $\mathbf{U}$.

Commentaires et Notations

Afin d'être le plus rigoureux possible, et pour plus de clarté, nous adopterons les notations suivantes :

(1) $\mathbf{\Phi}_r$ est la matrice orthogonale contenant les modes POD discrets.

(2) $\Big(\boldsymbol{\phi}_k\Big)_{1\leq k\leq K}$ sont les modes POD discrets, de sorte que $\boldsymbol{\phi}_k$ est la k-ème colonne de $\mathbf{\Phi}_r$.

(3) On notera $\Big(\phi_k(\boldsymbol{x})\Big)_{1\leq k\leq K}$ les modes POD continus, définies pour tout k par

$$\phi_k(\boldsymbol{x}) = \sum_{i=1}^{N_x} \psi_i(\boldsymbol{x})\Big(\boldsymbol{\phi}_k\Big)_i.$$

(4) $\mathbf{A}$ la matrice contenant les coefficients POD de $\mathbf{U}$.

(5) $\mathbf{a}^n$ est la n-ème colonne de $\mathbf{A}$ et contient les coefficients POD de $\mathbf{u}^n$.

Notons que d'après (3), l'ensemble $\Big\{\phi_k(\boldsymbol{x}), \quad \forall 1 \leq k \leq K\Big\}$ est orthogonal dans $L^2(\Omega)$, en effet :

$$\int_\Omega \phi_k(\boldsymbol{x})\phi_l(\boldsymbol{x})d\boldsymbol{x} = \sum_{i,j=1}^{N_x} \Big(\boldsymbol{\phi}_k\Big)_i\Big(\boldsymbol{\phi}_l\Big)_j \int_\Omega \psi_i(\boldsymbol{x})\psi_j(\boldsymbol{x})d\boldsymbol{x} = \boldsymbol{\phi}_k^T\mathbf{M}\boldsymbol{\phi}_l = \delta_{k,l}$$

d'après (2.4) et l'orthogonalité des vecteurs propres $\mathbf{v}_k$.

2.3 Identification de la dynamique et apprentissage d'un modèle réduit

Nous souhaitons dans cette section identifier la dynamique, autrement dit, trouver un opérateur linéaire faisant correspondre $u_h(\boldsymbol{x}, t^n)$ à $u_h(\boldsymbol{x}, t^{n+1})$. En tenant compte de la décomposition (2.2), cela revient à trouver une matrice $\mathcal{A} \in \mathbb{R}^{N_x\times N_x}$, telle que

$$\mathbf{u}^{n+1} = \mathcal{A}\,\mathbf{u}^n. \tag{2.6}$$

2.3.1 Première approche

Cherchons une matrice $\mathcal{A} \in \mathbb{R}^{N_x \times N_x}$ satisfaisant (2.6). Pour ce faire, introduisons les ensembles d'entrées et de sorties d'entrainement $\mathcal{X}$ et $\mathcal{Y}$ définis par :

$$\mathcal{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^{N_t-1} \\ | & | & & | \end{bmatrix} \quad \text{et} \quad \mathcal{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^2 & \mathbf{u}^3 & \cdots & \mathbf{u}^{N_t} \\ | & | & & | \end{bmatrix}. \tag{2.7}$$

Ainsi la matrice $\mathcal{A}$ devra satisfaire l'équation matricielle

$$\mathcal{A}\,\mathcal{X} = \mathcal{Y}. \tag{2.8}$$

Un des moyens les plus courant est de chercher $\mathcal{A}$ par une méthode des moindres carrés. Définissons alors la matrice $\mathcal{A}^*$ comme

$$\mathcal{A}^* = \arg\min_{\mathcal{A}} J(\mathcal{A}) := \frac{1}{2}\|\mathcal{A}\,\mathcal{X} - \mathcal{Y}\|_F^2$$

La fonctionnelle J étant quadratique par rapport à $\mathcal{A}$, le minimum de J est atteint au point pour lequel son gradient s'annule. Autrement dit, pour

$$\langle dJ(\mathcal{A}), \mathcal{H}\rangle_F = 0, \qquad \text{pour toute perturbation } \mathcal{H}.$$

$\mathcal{A}^*$ est alors définie implicitement à travers l'équation matricielle

$$\mathcal{A}^*\,\mathcal{X}\mathcal{X}^T = \mathcal{Y}\mathcal{X}^T \tag{2.9}$$

Si la matrice $\mathcal{X}\mathcal{X}^T$ était inversible, nous aurions alors une expression analytique de la matrice $\mathcal{A}^*$, et notre problème d'identification serait alors résolu. Cependant, cette condition n'est en générale pas satisfaite, et cela est dû au rang de la matrice $\mathcal{X}\mathcal{X}^T$. En effet, la matrice $\mathcal{X}$ est de taille $N_x \times N_t$, ainsi le rang de la matrice $\mathcal{X}\mathcal{X}^T$ vaut au plus $min(N_t, N_x)$. En général $N_t < N_x$, ainsi le rang de $\mathcal{X}\mathcal{X}^T$ vaut au plus N_t et donc la matrice n'est pas inversible.

Afin de contourner ce problème, une façon naturelle consiste à régulariser la fonctionnelle J en pénalisant la norme de Frobenius de la matrice $\mathcal{A}$. Cela correspond à une régularisation de Tikhonov, que l'on retrouve régulièrement en Apprentissage

automatique et en Problème inverse. Le problème d'identification devient alors :

$$\mathcal{A}^*_\alpha = \arg\min_{\mathcal{A}} J_\alpha(\mathcal{A}) := \frac{1}{2}\|\mathcal{A}\,\mathcal{X} - \mathcal{Y}\|_F^2 + \frac{\alpha}{2}\|\mathcal{A}\|_F^2$$

De même que précédemment, la fonctionnelle J_α est quadratique, et donc le minimum est atteint pour

$$\langle dJ_\alpha(\mathcal{A}), \mathcal{H}\rangle_F = 0, \qquad \text{pour toute perturbation } \mathcal{H}.$$

La matrice $\mathcal{A}^*_\alpha$ est alors définie par :

$$\mathcal{A}^*_\alpha \left(\mathcal{X}\mathcal{X}^T + \alpha \mathbf{I}_{N_x}\right) = \mathcal{Y}\mathcal{X}^T \tag{2.10}$$

Cette fois-ci la matrice $\mathcal{X}\mathcal{X}^T + \alpha\mathbf{I}_{N_x}$ est inversible et on a alors l'expression analytique de $\mathcal{A}^*_\alpha$

$$\mathcal{A}^*_\alpha = \mathcal{Y}\mathcal{X}^T\left(\mathcal{X}\mathcal{X}^T + \alpha \mathbf{I}_{N_x}\right)^{-1}$$

En faisant tendre α vers 0, on obtient finalement l'expression analytique de $\mathcal{A}^*$. Notons par ailleurs que l'on retrouve aussi la définition du pseudo-inverse de Moore-Penrose $\mathcal{X}^\dagger := \lim_{\alpha\to 0} \mathcal{X}^T\left(\mathcal{X}\mathcal{X}^T + \alpha \mathbf{I}_{N_x}\right)^{-1}$. Ainsi on obtient

$$\mathcal{A}^* = \mathcal{Y}\mathcal{X}^\dagger.$$

On a donc finalement pu trouver une matrice satisfaisant la condition (2.8). En revanche, rien ne nous indique que cette application est fidèle au modèle (2.1). En effet il n'y a pas unicité de la matrice $\mathcal{A}^*$. En ajoutant le terme de pénalisation, nous avons tout simplement choisit l'application ayant le plus faible spectre en norme ℓ^2 dans l'espace des applications satisfaisant l'équation (2.8). Procéder ainsi n'est donc pas adapté à l'identification de la dynamique.

2.3.2 Identification de la dynamique

Afin de correctement identifier la dynamique, il nous faut donc définir un problème de minimisation ayant existence et unicité. Pour cela nous allons utiliser l'espace vectoriel, W^K introduit à la section précédente. Ainsi, nous utiliserons la *représentation basse dimension* $\mathbf{A}$ de la matrice de données $\mathbf{U}$. Les ensembles d'arrivées et de sorties

d'entrainement deviennent alors

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ | & | & & | \end{bmatrix} \quad \text{et} \quad Y = \begin{bmatrix} | & | & & | \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ | & | & & | \end{bmatrix}. \tag{2.11}$$

Cette fois-ci, nous chercherons la matrice A^* telle que

$$A^* = \arg\min_A \, J(A) := \frac{1}{2}\|A\,X - Y\|_F^2$$

En effet, la matrice XX^T est inversible. Et cela est une conséquence directe de la représentation basse dimension des données. On a alors identifier la dynamique (2.1) au travers de la matrice A^* définie par

$$A^* = Y X^\dagger.$$

La formule ci-dessus sera appelée *formule DMD*.

2.3.3 Apprentissage d'un modèle réduit

Utiliser la représentation basse dimension des données n'a en fait pas seulement permis d'obtenir un problème d'identification bien posé. Cela nous a aussi permis de trouver une matrice A de petite dimension. Ainsi prédire le comportement de la solution u sera peu coûteux. Nous avons finalement appris un modèle réduit de la forme

$$\mathbf{a}^{n+1} = A\,\mathbf{a}^n, \quad \text{et } \mathbf{u}_r^{n+1} = \mathbf{\Phi}_r\,\mathbf{a}^{n+1},$$

fidèle au modèle

$$\partial_t\, u(\boldsymbol{x}, t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x}, t) = 0,$$

à partir des données d'entrainement et permettant une rapide prédiction de la solution u.

L'algorithme 1 résume la procédure d'identification de la dynamique et l'apprentissage d'un modèle réduit.

Algorithme 1 : Identification de la dynamique et apprentissage d'un modèle réduit

Entrées : La matrice de données $\mathbf{U}$ et la matrice de corrélation $\mathcal{S}$

Sorties : Le modèle réduit $\mathbf{a}^{n+1} = A\,\mathbf{a}^n, \quad$ et $\mathbf{u}_r^{n+1} = \mathbf{\Phi}_r\,\mathbf{a}^{n+1}$.

Réduction de la dimensionnalité :

- Décomposer $\mathcal{S}$ en valeurs propres, i.e. $\mathcal{S} \leftarrow \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$
- Tronquer les valeurs propres et vecteurs propres, i.e. $\mathbf{V}_r \leftarrow \mathbf{V}_{\cdot,1:K}$, $\mathbf{\Lambda}_r \leftarrow \mathbf{\Lambda}_{1:K,1:K}$
- Construire les modes POD et coefficients POD, i.e. $\mathbf{\Phi}_r \leftarrow \mathbf{U}\mathbf{V}_r\mathbf{\Lambda}_r^{-\frac{1}{2}}$, $\mathbf{A} \leftarrow \mathbf{\Lambda}_r^{\frac{1}{2}}\mathbf{V}_r^T$

Construction des ensembles d'entraînement :

$$X \leftarrow \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ | & | & & | \end{bmatrix} \quad \text{et} \quad Y \leftarrow \begin{bmatrix} | & | & & | \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ | & | & & | \end{bmatrix}.$$

Identification de la dynamique :

$$A \leftarrow Y X^{\dagger}.$$

2.3.4 Analyse numérique de l'identification

Les données contenues dans la matrice $\mathbf{U}$ sont issues de la résolution du modèle (2.1). Cette EDP étant linéaire, il existe alors une matrice $\mathcal{A} \in \mathbb{R}^{N_x \times N_x}$ telle que

$$\mathcal{A}\,\mathcal{X} = \mathcal{Y}, \quad \text{avec } \mathcal{X}, \mathcal{Y} \text{ définis en (5.2).}$$

Cependant, réduire les données revient à introduire un bruit artificiel et ainsi il n'existe pas de matrice $A \in \mathbb{R}^{K \times K}$, telle que les ensembles d'entrées et de sorties d'entrainement réduits X et Y vérifient

$$A\,X = Y, \quad \text{avec } X, Y \text{ définis en (2.11),}$$

ou du moins pas une matrice A fidèle à la matrice $\mathcal{A}$.

En fait, à travers la méthode DMD nous souhaiterions retrouver la représentation basse dimension $\tilde{A}$ de la matrice $\mathcal{A}$, définie par

$$\tilde{A} = \mathbf{\Phi}_r^T \mathbf{M} \mathcal{A} \mathbf{\Phi}_r$$

Le résultat ci-dessous quantifie l'erreur pour la représentation basse dimension $\tilde{A}$.

Proposition 1. *Soit $\tilde{A}$ la matrice basse dimension, on a alors*

$$\|\tilde{A}X - Y\|_F \leq \Big(\sum_{k>K} \lambda_k \Big)^{\frac{1}{2}} \rho(\mathcal{A})$$

avec K l'ordre de troncature, et λ_k la k-ème valeur propre de la matrice $\mathcal{S}$.

Démonstration. La première étape de la preuve, est de donner une expression analytique de l'erreur $\tilde{A}\,X - Y$.
Par hypothèse, nous avons

$$\mathcal{A}\mathcal{X} = \mathcal{Y},$$

on a ainsi

$$\mathcal{A}\big(\mathcal{X} - \tilde{\mathcal{X}}\big) + \mathcal{A}\tilde{\mathcal{X}} = \big(\mathcal{Y} - \tilde{\mathcal{Y}}\big) + \tilde{\mathcal{Y}},$$

avec $\tilde{\mathcal{X}} := \mathbf{\Phi}_r X$, et $\tilde{\mathcal{Y}} := \mathbf{\Phi}_r Y$. La représentation basse dimension de cette égalité est

$$\mathbf{\Phi}_r^T \mathbf{M} \mathcal{A}\, \mathcal{E}_{\mathcal{X}} + \underbrace{\mathbf{\Phi}_r^T \mathbf{M} \mathcal{A} \mathbf{\Phi}_r}_{=\tilde{A}} X = \underbrace{\mathbf{\Phi}_r^T \mathbf{M} \mathcal{E}_{\mathcal{Y}}}_{=\mathbf{0}} + \underbrace{\mathbf{\Phi}_r^T \mathbf{M} \mathbf{\Phi}_r}_{=\mathbf{I}_K} Y.$$

avec $\mathcal{E}_{\mathcal{X}} := \mathcal{X} - \tilde{\mathcal{X}}$, et de même $\mathcal{E}_{\mathcal{Y}} := \mathcal{Y} - \tilde{\mathcal{Y}}$. Ainsi, le résidu vaut

$$\boxed{\tilde{A}\,X - Y = -\mathbf{\Phi}_r^T \mathbf{M} \mathcal{A}\, \mathcal{E}_{\mathcal{X}}} \tag{2.12}$$

À présent, cherchons à majorer la norme de Frobenius de la matrice $\mathbf{\Phi}_r^T \mathbf{M} \mathcal{A}\, \mathcal{E}_{\mathcal{X}}$.
On a

$$\|\mathbf{\Phi}_r^T \mathbf{M} \mathcal{A}\, \mathcal{E}_{\mathcal{X}}\|_F^2 = \sum_{k=1}^{K} \sum_{n=1}^{N_t - 1} \Big| \Big\langle \boldsymbol{\phi}_k \,,\, \mathcal{A}\, \mathcal{E}_{\mathcal{X}}^n \Big\rangle_{\mathbf{M}} \Big|^2$$

Or les $\big(\boldsymbol{\phi}_k\big)_{1 \leq k \leq K}$ forment un ensemble orthonormal sur l'espace $\Big(\mathbb{R}^{N_x}, \|\cdot\|_{\mathbf{M}}\Big)$, en le complétant on obtient

$$\|\mathbf{\Phi}_r^T \mathbf{M} \mathcal{A}\, \mathcal{E}_{\mathcal{X}}\|_F^2 \leq \sum_{n=1}^{N_t - 1} \|\mathcal{A} \mathcal{E}_{\mathcal{X}}^n\|_{\mathbf{M}}^2.$$

En définissant la norme spectrale pondérée par la matrice de Masse

$$\|\mathcal{A}\|_{\mathbf{M}} := \sup_{\boldsymbol{\phi}\neq 0} \frac{\|\mathcal{A}\boldsymbol{\phi}\|_{\mathbf{M}}}{\|\boldsymbol{\phi}\|_{\mathbf{M}}} = \rho(\mathcal{A}).$$

Nous avons

$$\|\mathcal{A}\,\mathcal{E}_{\mathcal{X}}^n\|_{\mathbf{M}}^2 \leq \|\mathcal{A}\|_{\mathbf{M}}^2 \|\mathcal{E}_{\mathcal{X}}^n\|_{\mathbf{M}}^2, \qquad \forall n \in \{1, \cdots, N_t - 1\}.$$

Finalement, en remarquant que

$$\sum_{n=1}^{N_t-1} \|\mathcal{E}_{\mathcal{X}}^n\|_{\mathbf{M}}^2 \leq \sum_{n=1}^{N_t} \|\mathbf{u}^n - \Pi^K \mathbf{u}^n\|_{\mathbf{M}}^2 = \sum_{k>K} \lambda_k,$$

nous retrouvons

$$\boxed{\|\boldsymbol{\Phi}_r^T \mathbf{M} \mathcal{A}\,\mathcal{E}_{\mathcal{X}}\|_F^2 \leq \Big(\sum_{k>K} \lambda_k\Big)\rho(\mathcal{A})^2} \tag{2.13}$$

ce qui conclut la preuve. □

Ainsi si l'erreur de réduction est trop importante, l'erreur commise par la matrice $\tilde{A}$ sera importante. Minimiser cette quantité afin de retrouver $\tilde{A}$ n'est donc pas forcément une bonne chose. En effet on risque d'identifier des corrélations artificielles entre les données. En revanche, lorsque l'erreur de réduction est suffisamment petite, l'erreur commise par la matrice $\tilde{A}$ est cette fois-ci faible et la méthode DMD se montrera efficace. Notons enfin, que le rayon spectral de la matrice $\mathcal{A}$ intervient dans la majoration de l'erreur. Cela correspond à la vitesse à laquelle l'erreur de projection se propage. Intuitivement, l'erreur de réduction se conservera pour un problème advectif, tandis qu'elle se dissipera pour un problème diffusif.

Si ce résultat nous donne une idée sur la précision du modèle que nous pourrions espérer apprendre, il ne nous donne cependant pas de résultat théorique quant à la précision du modèle que nous apprenons par la méthode DMD. En revanche, la proposition ci-dessous nous donne le résultat suivant :

Proposition 2. *Soit la matrice $\hat{A} := YX^\dagger$ apprise par la méthode DMD, on a alors*

$$\|\hat{A}X - Y\|_F \leq \Big(\sum_{k>K} \lambda_k\Big)^{\frac{1}{2}} \rho(\mathcal{A})\Big(1 + \kappa_F(X)\Big),$$

avec $\kappa_F(X) = \|X\|_F \|X^\dagger\|_F$ le conditionnement de la matrice X pour la norme de Frobenius .

Démonstration. Reprenons l'égalité

$$\boxed{\tilde{A}\,X - Y = -\mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\,\mathcal{E}_{\mathcal{X}}}.$$

En multipliant à droite par la matrice $X^\dagger X$ puis en ajoutant et soustrayant Y au membre de gauche, on obtient l'égalité

$$\tilde{A}X - Y - \Big(\hat{A}X - Y\Big) = -\mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\,\mathcal{E}_{\mathcal{X}} X^\dagger X,$$

on peut alors estimer l'erreur $\hat{A}X - Y$ en norme de Frobenius, i.e.

$$\|\hat{A}X - Y\|_F \leq \Big(\sum_{k>K} \lambda_k\Big)^{\frac{1}{2}} \rho(\mathcal{A})\Big(1 + \|X^\dagger X\|_F\Big).$$

enfin le résultat s'obtient par définition du conditionnement de Frobenius. □

Contrairement à la majoration de l'erreur pour la matrice $\tilde{A}$, ce résultat fait intervenir le conditionnement de la matrice X.

Remarque 5. *En statistiques, le problème de multicolinéarité est bien connu. ([159], [64]). Ce problème intervient lorsque plusieurs variables prédictives d'un modèle statistique sont colinéaires. Cela entraîne une mauvaise estimation des paramètres du modèle. Afin de s'assurer que les variables prédictives ne sont pas colinéaires, on peut calculer le conditionnement de la matrice des prédicteurs ([20]). Dans notre cas, les variables prédictives sont tout simplement les coefficients POD. Et la matrice des prédicteurs la matrice X. Ainsi il est naturel de retrouver le conditionnement dans l'estimation de l'erreur. Notons que dans notre cas la multicolinéarité est dû à une erreur de réduction trop petite. Il faut donc choisir un K qui satisfasse à la fois une erreur de réduction assez petite afin que la matrice $\hat{A}$ n'apprenne pas de corrélations artificielles, mais aussi assez grande pour que le problème d'identification ne soit pas instable.*

Enfin, nous pouvons finalement donner le théorème suivant :

Théorème 2. *Soit $u_h(\boldsymbol{x}, t^n)$ la solution du modèle* (2.1) *générée par le solveur haute-fidélité, et $\hat{u}_h(\boldsymbol{x}, t^n)$ la solution obtenue par le modèle réduit. On a alors*

$$\Big(\sum_{n=1}^{N_t} \|u_h(\cdot,t^n) - \hat{u}_h(\cdot,t^n)\|^2_{L^2(\Omega)}\Big)^{\frac{1}{2}} \leq \Big(\sum_{k>K} \lambda_k\Big)^{\frac{1}{2}} \left\{1 + \rho(\mathcal{A})\Big(1 + \kappa_F(X)\Big)\Big(\sum_{m=1}^{N_t-1} \|\hat{A}^{m-1}\|^2\Big)^{\frac{1}{2}}\right\}.$$

Démonstration. Nous utiliseront la notation chapeau pour désigner toutes les données obtenues à l'aide du modèle réduit. On cherche à majorer la quantité $\sum_{n=1}^{N_t} \|\mathbf{u}^n - \hat{\mathbf{u}}^n\|^2_{\mathbf{M}}$.

En utilisant la décomposition POD, nous avons

$$\mathbf{U} = \mathbf{U}_r + \mathcal{E}, \quad \text{avec } \mathcal{E} \text{ le résidu.}$$

Par ailleurs $\hat{\mathbf{U}}$ étant reconstruite à l'aide de la base POD, nous avons $\hat{\mathbf{U}} = \mathbf{\Phi}_r \begin{bmatrix} X^1 & \hat{Y} \end{bmatrix}$, avec la notation $X^j = X_{\cdot,j}$.
Enfin, d'après les notations introduites précédemment, on a $\mathbf{U}_r = \mathbf{\Phi}_r \begin{bmatrix} X^1 & Y \end{bmatrix}$. Ce qui nous permet de donner le premier résultat intermédiaire

$$\Big(\sum_{n=1}^{N_t} \|\mathbf{u}^n - \hat{\mathbf{u}}^n\|_{\mathbf{M}}^2 \Big)^{\frac{1}{2}} \leq \|Y - \hat{Y}\|_F + \Big(\sum_{k>K} \lambda_k \Big)^{\frac{1}{2}}. \tag{2.14}$$

Dans un second temps nous chercherons à estimer l'erreur $\|Y - \hat{Y}\|_F$ en fonction de l'erreur $\|Y - \hat{A}X\|_F$.
En effet, le modèle réduit utilise la condition initiale. On a donc $\hat{Y}^n = \hat{A}^n X^1$, si bien qu'en remarquant que $X^n = Y^{n-1}$, on peut en déduire la formule suivante :

$$Y^n - \hat{Y}^n = \sum_{m=1}^{n} \hat{A}^{m-1} \left(Y^{n-m+1} - \hat{A}\, X^{n-m+1} \right). \tag{2.15}$$

On a donc

$$\begin{aligned}
\sum_{n=1}^{N_t-1} \|Y^n - \hat{Y}^n\|^2 &\leq \sum_{n=1}^{N_t-1} \sum_{m=1}^{n} \|\hat{A}^{m-1} \left(Y^{n-m+1} - \hat{A}\, X^{n-m+1} \right)\|^2 \\
&\leq \sum_{n=1}^{N_t-1} \sum_{m=1}^{n} \|\hat{A}^{m-1}\|^2 \|Y^{n-m+1} - \hat{A}\, X^{n-m+1}\|^2 \\
&\leq \sum_{m=1}^{N_t-1} \|\hat{A}^{m-1}\|^2 \Big(\sum_{n=m}^{N_t-1} \|Y^{n-m+1} - \hat{A}\, X^{n-m+1}\|^2 \Big) \\
&\leq \Big(\sum_{m=1}^{N_t-1} \|\hat{A}^{m-1}\|^2 \Big) \|Y - \hat{A}X\|_F^2.
\end{aligned}$$

Enfin le résultat s'obtient en utilisant la **proposition 2** . □

En plus des termes présent dans la majoration de la **proposition 2**, on a aussi la norme spectrale des itérés de la matrice apprise qui intervient. Ainsi le spectre de $\hat{A}$ a une grande importance. On peut donc espérer avoir une bonne approximation $\hat{u}_h(\boldsymbol{x}, t^n)$ si le spectre de $\hat{A}$ est inférieur à 1.

Remarque 6. *Le théorème ci-dessus fait intervenir le rayon spectral de la matrice apprise $\hat{A}$. Trois cas sont alors à spécifier :*

- $\rho(\hat{A}) < 1$ *on a alors :* $\left(\sum_{m=1}^{N_t-1} \|\hat{A}^{m-1}\|^2\right)^{\frac{1}{2}} \leq \left(\rho(\hat{A})^2\big(N_t-1\big)\right)^{\frac{1}{2}}$.
- $\rho(\hat{A}) = 1$ *on a alors :* $\left(\sum_{m=0}^{N_t-1} \|\hat{A}^{m}\|^2\right)^{\frac{1}{2}} = \big(N_t-1\big)^{\frac{1}{2}}$
- $\rho(\hat{A}) > 1$, *auquel cas* $\left(\sum_{m=0}^{N_t-1} \|\hat{A}^{m}\|^2\right)^{\frac{1}{2}}$ *explosera avec* N_t.

Terminons par donner quelques résultats sur le rayon spectral des matrices $\tilde{A}$ et $\hat{A}$. Tout d'abord nous avons :

Proposition 3. *Soit $\mathcal{A}$ la matrice "grande dimension", et $\tilde{A}$ sa représentation "basse dimension" dans l'espace POD, on a alors*

$$\rho(\tilde{A}) \leq \rho(\mathcal{A}).$$

Démonstration. Rappelons que $\tilde{A} = \mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\mathbf{\Phi}_r$, et que $\mathbf{\Phi}_r^T \mathbf{M}\mathbf{\Phi}_r = \mathbf{I}_K$.

Soit $\mathbf{v}$ un vecteur de taille K, on a alors

$$\mathbf{v}^T\tilde{A}\mathbf{v} = \mathbf{v}^T\mathbf{\Phi}_r^T\mathbf{M}\mathcal{A}\mathbf{\Phi}_r\mathbf{v},$$

posons ensuite $\mathbf{w} = \mathbf{\Phi}_r\mathbf{v}$, on a alors que

$$\mathbf{v}^T\tilde{A}\mathbf{v} = \mathbf{w}^T\mathbf{M}\mathcal{A}\mathbf{w}.$$

Par ailleurs

$$\mathbf{v}^T\mathbf{v} = \mathbf{w}^T\mathbf{M}\mathbf{w},$$

si bien que pour tout $\mathbf{v} \in \mathbb{R}^K$ et tout $\mathbf{w} = \mathbf{\Phi}_r\mathbf{v}$,

$$\mathcal{R}(\tilde{A}, \mathbf{v}) = \mathcal{R}_{\mathbf{M}}(\mathcal{A}, \mathbf{w}),$$

avec $\mathcal{R}$ le quotient de Rayleigh et $\mathcal{R}_{\mathbf{M}}$ le quotient de Rayleigh pour la matrice de Masse. On peut alors conclure que

$$\rho(\tilde{A}) = \sup_{\mathbf{v}} \mathcal{R}(\tilde{A}, \mathbf{v}) = \sup_{\mathbf{w}\in Range(\mathbf{\Phi}_r)} \mathcal{R}_{\mathbf{M}}(\mathcal{A}, \mathbf{w}) \leq \sup_{\mathbf{w}} \mathcal{R}_{\mathbf{M}}(\mathcal{A}, \mathbf{w}) = \rho(\mathcal{A}).$$

□

Donnons enfin un résultat sur le rayon spectral de la matrice $\hat{A}$.

Théorème 3. *Soit $\hat{A}$ la matrice obtenue par la méthode DMD, on a alors*

$$\rho(\hat{A}) \leq \rho(\mathcal{A})\Big(1 + \big(\sum_{k>K} \lambda_k\big)^{\frac{1}{2}} \big(\sigma_{min}(X)\big)^{-1}\Big).$$

avec $\sigma_{min}(X)$ la plus petite valeur singulière de la matrice X.

Démonstration. D'après (2.12), nous avons

$$\hat{A} = \mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\mathcal{E}_{\mathcal{X}} X^\dagger + \tilde{A}.$$

Par ailleurs,

$$\|\mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\mathcal{E}_{\mathcal{X}} X^\dagger\|_2 \leq \|\mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\mathcal{E}_{\mathcal{X}}\|_F \|X^\dagger\|_2.$$

D'autre part, considérons la décomposition en valeurs singulières de X, i.e. $X = U\Sigma V^T$, alors son pseudo-inverse est égal à $X^\dagger = V\Sigma^\dagger U^T$, et donc $\|X^\dagger\|_2 = \frac{1}{\sigma_{min}(X)}$. Finalement d'après (2.13), on obtient que

$$\|\mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\mathcal{E}_{\mathcal{X}} X^\dagger\|_2 \leq \big(\sum_{k>K} \lambda_k\big)^{\frac{1}{2}} \rho(\mathcal{A}) \frac{1}{\sigma_{min}(X)},$$

ce qui conclut la preuve. □

2.4 Extension de la méthode

L'identification de la dynamique et l'apprentissage d'un modèle réduit ont été présentés dans le cas où les données satisfaisaient la relation linéaire :

$$\mathcal{A}\,\mathcal{X} = \mathcal{Y}. \tag{2.16}$$

Cependant, ce n'est pas toujours le cas, mais de légères modifications, permettent de retrouver ce cadre. Nous présenterons dans cette section, l'extension de la méthode DMD à un modèle non homogène ainsi qu'à un modèle d'ordre 2 en temps.

2.4.1 Identification d'un terme source statique

L'absence de terme source dans le modèle (2.1) est assez restrictif, en effet dans de nombreuses applications le système est excité par une force extérieure. Considérons le

modèle suivant :

$$\partial_t u(\boldsymbol{x}, t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x}, t) = f(\boldsymbol{x}). \tag{2.17}$$

Une discrétisation du modèle nous montre que dans ce cas, les données satisfont la relation affine

$$\mathcal{A}\,\mathcal{X} + \mathcal{B} = \mathcal{Y},$$

avec

$$\mathcal{B} = \begin{bmatrix} | & | & & | \\ \mathcal{B}^f & \mathcal{B}^f & \cdots & \mathcal{B}^f \\ | & | & & | \end{bmatrix},$$

et où $\mathcal{B}^f$ est la discrétisation du terme source f.

Nous voudrions alors apprendre un modèle réduit de cette forme, afin d'être fidèle au modèle (2.17).
Reprenons pour cela les ensembles d'entraînement X et Y introduits en (2.11), et considérons le problème de minimisation suivant :

$$\left(\hat{A}, \hat{B}\right) = \underset{(A,B)}{\arg\min}\ \mathfrak{L}(A, B) := \frac{1}{2}\|A\,X + B\,Z - Y\|_F^2,$$

avec $Z := \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{N_t - 1}$. on a ainsi

$$\langle \frac{\partial \mathfrak{L}(A, B)}{\partial A}, H_A \rangle_F = \langle AXX^T + B\,ZX^T - YX^T, H_A \rangle_F, \quad \text{pour toute perturbation } H_A,$$

de même

$$\langle \frac{\partial \mathfrak{L}(A, B)}{\partial B}, H_B \rangle_F = \langle AXZ^T + BZ^T - YZ^T, H_B \rangle_F, \quad \text{pour toute perturbation } H_B.$$

On en déduit alors le système matriciel

$$\begin{cases} \hat{A} + \hat{B}ZX^\dagger = YX^\dagger, \\ \hat{A}XZ^\dagger + \hat{B} = YZ^\dagger \end{cases} \tag{2.18}$$

Finalement, on retrouve les expressions analytiques :

$$\begin{cases} \hat{B} = \frac{1}{ZX^\dagger XZ^\dagger - 1}\Big(YX^\dagger XZ^\dagger - YZ^\dagger\Big), \\ \hat{A} = YX^\dagger - \hat{B}ZX^\dagger \end{cases} \tag{2.19}$$

On obtient alors le modèle réduit suivant

$$\mathbf{a}^{n+1} = \hat{A}\,\mathbf{a}^n + \hat{B}, \qquad \mathbf{u}^{n+1} = \mathbf{\Phi}_r\,\mathbf{a}^{n+1}. \tag{2.20}$$

Le cas d'un terme source dynamique, est en revanche plus complexe car il nécessite d'apprendre également son évolution en temps. Nous verrons dans la section suivante un moyen d'y arriver.

2.4.2 Identification d'un terme source dynamique

À présent considérons le cas d'un terme source dynamique, en émettant l'hypothèse qu'il puisse s'écrire sous la forme tensorisée suivante :

$$f(\boldsymbol{x}, t) = \sum_{m=1}^{M} g_m(\boldsymbol{x})\theta_m(t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle.$$

avec $\boldsymbol{\theta}$ solution de l'EDO suivante :

$$\dot{\boldsymbol{\theta}}(t) = C\,\boldsymbol{\theta}(t).$$

On considère alors dans cette section le modèle

$$\partial_t\, u(\boldsymbol{x}, t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) \tag{2.21}$$

Une discrétisation du modèle (2.21), nous donne

$$\mathcal{A}\,\mathcal{X} + \mathcal{B} = \mathcal{Y},$$

avec cette fois-ci

$$\mathcal{B} = \begin{bmatrix} | & | & & | \\ \sum_{m=1}^{M} \mathcal{B}^{g_m}\theta_m^1 & \sum_{m=1}^{M} \mathcal{B}^{g_m}\theta_m^2 & \cdots & \sum_{m=1}^{M} \mathcal{B}^{g_m}\theta_m^{N_t-1} \\ | & | & & | \end{bmatrix},$$

et $\mathcal{B}^{g_m}$ le vecteur correspondant au terme source g_m.

Comme précédemment, nous aimerions que notre modèle réduit soit de cette forme. Pour ce faire, nous introduisons le problème d'optimisation

$$\left(\hat{A}, \hat{B}, \hat{C}\right) = \underset{(A,B,C)}{\arg\min}\ \mathfrak{L}(A,B,C) := \frac{1}{2}\|A\,X + B\,Z - Y\|_F^2 + \frac{1}{2}\|C\,Z - Z'\|_F^2,$$

avec

$$Z = \begin{bmatrix} | & | & & | \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ | & | & & | \end{bmatrix}, \quad \text{et} \quad Z' = \begin{bmatrix} | & | & & | \\ \boldsymbol{\theta}^2 & \boldsymbol{\theta}^3 & \cdots & \boldsymbol{\theta}^{N_t} \\ | & | & & | \end{bmatrix},$$

les ensembles d'entraînement permettant d'apprendre l'évolution de $\boldsymbol{\theta}(t)$. En procédant de manière analogue, on retrouve le système matriciel

$$\begin{cases} \hat{A} + \hat{B}ZX^\dagger = YX^\dagger, \\ \hat{A}XZ^\dagger + \hat{B} = YZ^\dagger, \\ \hat{C} = Z'Z^\dagger \end{cases} \tag{2.22}$$

Les expressions analytiques sont alors données par :

$$\begin{cases} \hat{B} = \left(YX^\dagger XZ^\dagger - YZ^\dagger\right)\left(ZX^\dagger XZ^\dagger - \mathbf{I}_M\right)^{-1}, \\ \hat{A} = YX^\dagger - \hat{B}ZX^\dagger, \\ \hat{C} = Z'Z^\dagger. \end{cases} \tag{2.23}$$

et l'on retrouve finalement le modèle réduit

$$\begin{cases} \mathbf{a}^{n+1} = \hat{A}\,\mathbf{a}^n + \hat{B}\,\mathbf{z}^n, \\ \mathbf{z}^{n+1} = \hat{C}\,\mathbf{z}^n, \\ \mathbf{u}^{n+1} = \boldsymbol{\Phi}_r\,\mathbf{a}^{n+1}. \end{cases}$$

Remarque 7. *Lorsque $\boldsymbol{\theta}$ n'est pas solution d'une EDO linéaire, il faut pouvoir prédire son comportement futur à partir des données observées. La procédure d'identification du modèle reste la même, excepté l'apprentissage de $\boldsymbol{\theta}$ qui sera réalisé par un algorithme plus complexe tel que les réseaux de neurones récurrent.*

Une analyse numérique de cette extension est présentée en annexe A .

2.4.3 Identification d'une EDP d'évolution linéaire d'ordre 2 en temps

2.4.3.1 Cas homogène

Considérons le modèle d'ordre 2 en temps suivant :

$$\partial_{tt}\, u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = 0. \tag{2.24}$$

D'après la formule de Taylor, nous avons

$$\partial_{tt}\, u(\boldsymbol{x},t^n) = \frac{u(\boldsymbol{x},t^{n-1}) - 2u(\boldsymbol{x},t^n) + u(\boldsymbol{x},t^{n+1})}{\delta t^2} + o(\delta t^2).$$

Ainsi pour connaître la solution au temps t^{n+1} il nous faut les solutions aux temps t^{n-1} et t^n. Nous introduisons les deux ensembles d'entrées d'entraînement X_1 et X_2 définis par

$$X_1 = \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-2} \\ | & | & & | \end{bmatrix}, \qquad X_2 = \begin{bmatrix} | & | & & | \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t-1} \\ | & | & & | \end{bmatrix}, \tag{2.25}$$

et l'ensemble de sorties d'entraînement Y sera quant à lui défini par :

$$Y = \begin{bmatrix} | & | & & | \\ \mathbf{a}^3 & \mathbf{a}^4 & \cdots & \mathbf{a}^{N_t} \\ | & | & & | \end{bmatrix}. \tag{2.26}$$

Nous cherchons alors à résoudre le problème d'optimisation :

$$\left(\hat{A}_1, \hat{A}_2\right) := \underset{(A_1,A_2)}{\arg\min}\ \mathfrak{L}(A_1,A_2) := \frac{1}{2}\|A_1\, X_1 + A_2\, X_2 - Y\|_F^2.$$

On retrouve alors,

$$\begin{cases} \hat{A}_2 = \left(Y X_1^\dagger X_1 X_2^\dagger - Y X_2^\dagger\right)\left(X_2 X_1^\dagger X_1 X_2^\dagger - \mathbf{I}_K\right)^{-1}, \\ \hat{A}_1 = Y X_1^\dagger - \hat{A}_2 X_2 X_1^\dagger, \end{cases} \tag{2.27}$$

de telle sorte que le modèle réduit s'écrira

$$\mathbf{a}^{n+1} = A_1\, \mathbf{a}^{n-1} + A_2\, \mathbf{a}^n, \qquad \mathbf{u}^{n+1} = \Phi_r\, \mathbf{a}^{n+1}.$$

2.4.3.2 Cas non-homogène

Lorsque le modèle que l'on souhaite apprendre est de la forme

$$\partial_{tt}\, u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = f(\boldsymbol{x},t) \tag{2.28}$$

On peut là encore donner une expression analytique des matrices $\hat{A}_1, \hat{A}_2, \hat{B}$ et $\hat{C}$, de manière à ce que le modèle réduit soit de la forme

$$\begin{cases} \mathbf{a}^{n+1} = \hat{A}_1\, \mathbf{a}^{n-1} + \hat{A}_2\, \mathbf{a}^n + \hat{B}\, \mathbf{z}^n, \\ \mathbf{z}^{n+1} = \hat{C}\, \mathbf{z}^n, \\ \mathbf{u}^{n+1} = \boldsymbol{\Phi}_r\, \mathbf{a}^{n+1}. \end{cases}$$

Ces extensions ne sont pas exhaustives, et d'autres EDP d'évolution linéaires peuvent être traitées de manière analogue.

2.4.4 Accélération de l'identification par empilement

Nous avons fait le choix de présenter des extensions à la méthode en reconsidérant le problème d'optimisation introduit dans le cas linéaire, cela offre un cadre confortable pour l'analyse numérique. Une autre approche consisterait à empiler les ensembles d'entraînement afin de re-basculer dans un cas linéaire et ensuite d'appliquer la *formule DMD*, i.e.

$$\hat{A} = Y X^{\dagger}.$$

Cette approche est dans le même esprit que l'extension de la méthode DMD introduite par Williams & al sous le nom de *Extended Dynamic Mode Decomposition* (EDMD, [184]). En pratique, l'identification par empilement est moins coûteuse.

Le tableau ci-dessous présente les ensembles d'entraînement empilés dans chacun des cas étudiés plus haut.

Forme de l'EDP	**Ensemble de départ X**	**Ensemble d'arrivée Y**
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = f(\boldsymbol{x})$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \\ 1 & 1 & \cdots & 1 \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle$ (avec apprentissage de $\boldsymbol{\theta}(t)$)	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^2 & \boldsymbol{\theta}^3 & \cdots & \boldsymbol{\theta}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle$ (sans apprentissage de $\boldsymbol{\theta}(t)$)	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_{tt} u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = 0$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-2} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^3 & \mathbf{a}^4 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$

TABLE 2.1 Ensembles d'entraînement en fonction de la forme du modèle par lequel les données sont générées.

Forme de l'EDP	**Ensemble de départ X**	**Ensemble d'arrivée Y**
$\partial_{tt}\, u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = f(\boldsymbol{x})$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-2} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \\ 1 & 1 & \cdots & 1 \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^3 & \mathbf{a}^4 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_{tt}\, u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle$ (avec apprentissage de $\boldsymbol{\theta}(t)$)	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-2} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^3 & \mathbf{a}^4 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^2 & \boldsymbol{\theta}^3 & \cdots & \boldsymbol{\theta}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_{tt}\, u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle$ (sans apprentissage de $\boldsymbol{\theta}(t)$)	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-2} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t-1} \\ \vert & \vert & & \vert \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{a}^3 & \mathbf{a}^4 & \cdots & \mathbf{a}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$

TABLE 2.2 Ensembles d'entraînement en fonction de la forme du modèle par lequel les données sont générées (suite).

Le tableau ci-dessous présente les complexités de chaque approche.

Forme de l'EDP	Identification sans empilement	Identification par empilement
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = f(\boldsymbol{x})$	$\mathcal{O}\big(4N_tK^2\big)$	$\mathcal{O}\big(2N_tK^2\big)$
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t)\rangle$ (avec apprentissage de $\boldsymbol{\theta}(t)$)	$\mathcal{O}\Big(N_t\big(4K^2+5KM+4M^2\big)\Big)$	$\mathcal{O}\Big(N_t\big(2K^2+4KM+2M^2\big)\Big)$
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t)\rangle$ (sans apprentissage de $\boldsymbol{\theta}(t)$)	$\mathcal{O}\Big(N_t\big(4K^2+5KM+2M^2\big)\Big)$	$\mathcal{O}\Big(N_t\big(2K^2+3KM+M^2\big)\Big)$
$\partial_{tt} u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = 0$	$\mathcal{O}\big(12N_tK^2\big)$	$\mathcal{O}\big(6N_tK^2\big)$
$\partial_{tt} u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = f(\boldsymbol{x})$	$\mathcal{O}\big(16N_tK^2\big)$	$\mathcal{O}\big(6N_tK^2\big)$
$\partial_{tt} u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t)\rangle$ (avec apprentissage de $\boldsymbol{\theta}(t)$)	$\mathcal{O}\Big(N_t\big(16K^2+10KM+4M^2\big)\Big)$	$\mathcal{O}\Big(N_t\big(6K^2+7KM+2M^2\big)\Big)$
$\partial_{tt} u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t)\rangle$ (sans apprentissage de $\boldsymbol{\theta}(t)$)	$\mathcal{O}\Big(N_t\big(16K^2+10KM+2M^2\big)\Big)$	$\mathcal{O}\Big(N_t\big(6K^2+5KM+M^2\big)\Big)$

TABLE 2.3 Complexité des méthodes avec et sans empilement.

On remarque ainsi que les nombreuses multiplications matricielles de la méthode sans empilement l'a rende plus coûteuse que celle avec empilement. Cependant dans tous les cas la complexité ne dépend jamais de N_x (qui est souvent très grand) et dépend linéairement de N_t. Ainsi ces deux méthodes permettent une identification du modèle dynamique en très peu de temps.

2.5 Applications numériques

Dans cette partie, nous présenterons l'application de la méthode DMD à un problème de diffusion et à un problème d'ondes. Il nous faudra tout d'abord générer des données à l'aide d'un solveur HF. Dans chaque section, la première partie sera dédiée à la présentation du modèle EDP et à la génération des données. Tandis que dans la seconde partie, nous ferons abstraction de la connaissance du modèle EDP et utiliserons exclusivement les données générées par le solveur HF en considérant celui-ci comme une boîte noire. Nous utiliserons par ailleurs la méthode par empilement, présentée à la **section 2.4.4**.

2.5.1 Problème de diffusion dynamique

Considérons tout d'abord la diffusion de la chaleur rendue fortement dynamique par un couplage EDP-EDO. Dans ce cas là, l'opérateur $\mathcal{L}$ sera régularisant, ce qui

facilitera la réduction de la dimensionnalité. Cependant le terme source dynamique, issu du couplage avec l'EDO, compliquera l'identification du modèle.

2.5.1.1 Mise en place de la boîte noire

Notre problème sera modélisé par une EDP parabolique linéaire. En outre, nous considérerons un terme source dynamique, décomposable en somme tensorisée, dont la partie temporelle sera solution d'une EDO. La partie spatiale sera quant à elle composée de deux fonctions permettant de refroidir et de chauffer le système. Le modèle que l'on cherchera à identifier est défini par :

$$\begin{cases} \partial_t u(x,t) - \kappa \partial_{xx} u(x,t) = \langle \boldsymbol{g}(x), \boldsymbol{\theta}(t) \rangle, & \text{dans } [0,1] \times (0,T), \\ u(0,t) = 0, \quad u(1,t) = 0, & \forall t \in (0,T), \\ \dot{\boldsymbol{\theta}}(t) = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \boldsymbol{\theta}(t), \qquad \boldsymbol{\theta}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, & \\ \boldsymbol{g}_i(x) = (-1)^{i+1} q_i \Big(\frac{1}{\tau\sqrt{100\pi}} e^{-\frac{(x-x_0)^2}{2\tau^2}} + |x - x_0|^3 (1 - \cos(8\pi x)) \Big), & \\ u(x,0) = u_0(x), & \text{dans } [0,1] \end{cases} \tag{2.29}$$

On prendra la fonction nulle comme condition initiale u_0, de plus la conductivité thermique κ sera quant à elle égale à 0.01.
Afin de générer les données, nous utiliserons un schéma de Crank-Nicholson pour discrétiser u, tandis que nous utiliserons l'expression analytique de $\boldsymbol{\theta}$. L'approximation u_h et la solution $\boldsymbol{\theta}$ seront alors stockées dans les matrices $\mathbf{U}$ et $\boldsymbol{\Theta}$, définies par :

$$\mathbf{U}_{i,n} =: \mathbf{u}_i^n = u_h(x_i, t^n) \quad \text{et} \quad \boldsymbol{\Theta}_{k,n} = \boldsymbol{\theta}_k(t^n),$$

avec les notations habituelles $x_i := (i-1)\delta x$ et $t^n := (n-1)\delta t$.
Le tableau ci-dessous résume les paramètres physiques et numériques utilisés.

Paramètres	**Valeurs**
Espérance du premier terme source (x_0)	0.5
Puissance du terme source (q_1, q_2)	3
Variance du terme source (τ)	0.05
Fréquence de l'activation du terme source (ω)	4π
Conductivité thermique (κ)	$0.01\ W \cdot m^{-1} \cdot K^{-1}$
Condition initiale (u_0)	0
Temps final (T)	6 sec
Pas d'espace (δx)	2×10^{-3}
Pas de temps (δt)	5×10^{-3}

TABLE 2.4 Paramètres utilisés par le solveur HF

La figure ci-dessous représente la solution u à différents temps.

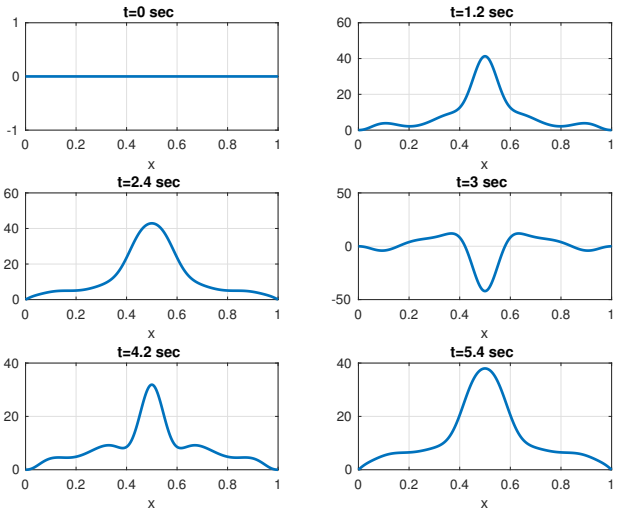


FIGURE 2.1 Solution HF à différents pas de temps

On peut ainsi remarquer que le problème est fortement dynamique autour du point $x_0 = 0.5$.

2.5.1.2 Identification de la dynamique et apprentissage d'un modèle réduit

À présent nous n'avons plus accès qu'aux matrices $\mathbf{U}$ et $\mathbf{\Theta}$. Nous allons donc, à partir de ces matrices de données tenter d'identifier la dynamique et d'apprendre un

modèle réduit.

Réduction de la dimensionnalité

Afin de réduire la dimensionnalité du problème d'identification, nous construisons une base POD. La figure ci-dessous présente les six premiers modes POD continus $\phi_k(x)$.

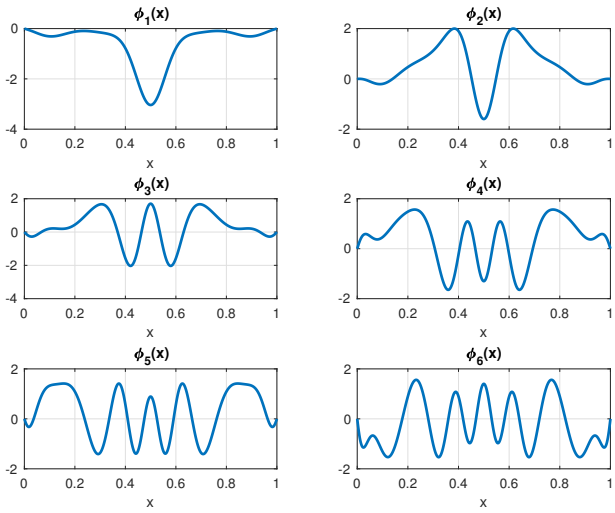


FIGURE 2.2 6 premiers modes POD continus

On peut ainsi voir que cette base est bien adaptée au caractère oscillant du problème.

Enfin nous avons vu que l'identification était dépendante de l'erreur de réduction, aussi nous présentons ci-dessous les erreurs en norme L^2 et H^1 définies par

$$err_{L^2} := \sqrt{\frac{\sum_{n=1}^{N_t} \| \sum_{k=1}^{K} \phi_k(\cdot)\mathbf{a}_k^n - u_h(\cdot,t^n)\|^2_{L^2(0,1)}}{\sum_{n=1}^{N_t} \|u_h(\cdot,t^n)\|^2_{L^2(0,1)}}}$$

et

$$err_{H^1} := \sqrt{\frac{\sum_{n=1}^{N_t} \| \sum_{k=1}^{K} \phi_k(\cdot)\mathbf{a}_k^n - u_h(\cdot,t^n)\|^2_{H^1(0,1)}}{\sum_{n=1}^{N_t} \|u_h(\cdot,t^n)\|^2_{H^1(0,1)}}}$$

Le tableau ci-dessous présente les résultats :

Nombre de modes POD (K)	**Erreur** L^2	**Erreur** H^1
1	0.2562	0.4001
2	0.0652	0.1516
3	0.0179	0.0404
4	0.0042	0.0110
5	9×10^{-4}	0.0027
10	8.6×10^{-7}	9.78×10^{-6}

TABLE 2.5 Erreurs relatives de réduction

On observe ainsi une rapide décroissance de l'erreur de réduction. Ainsi pour un rang de troncature égal à 10, l'erreur de réduction est tout à fait raisonnable.
Par la suite, nous choisirons donc un rang de troncature K égal à 10.

Identification de la dynamique sous l'hypothèse d'un terme source statique

Dans un premier temps, nous supposerons que le terme source est statique. Nous utiliserons donc l'extension de la méthode présentée au **paragraphe 2.4.1**. Par conséquent, le modèle réduit appris sera de la forme :

$$\mathbf{a}^{n+1} = \hat{A}\,\mathbf{a}^n + \hat{B}, \qquad \mathbf{u}^{n+1} = \mathbf{\Phi}_r\,\mathbf{a}^{n+1}.$$

Nous séparerons les données, la première moitié des pas de temps servira à l'entrainement et la seconde servira à la validation.
Comparons à présent la prédiction de la solution u obtenue en utilisant le modèle réduit avec celle générée par le solveur HF, sur l'ensemble d'entrainement et de validation. La prédiction est représentée par une ligne continue rouge, tandis que la solution HF est représentée par une ligne continue bleue.

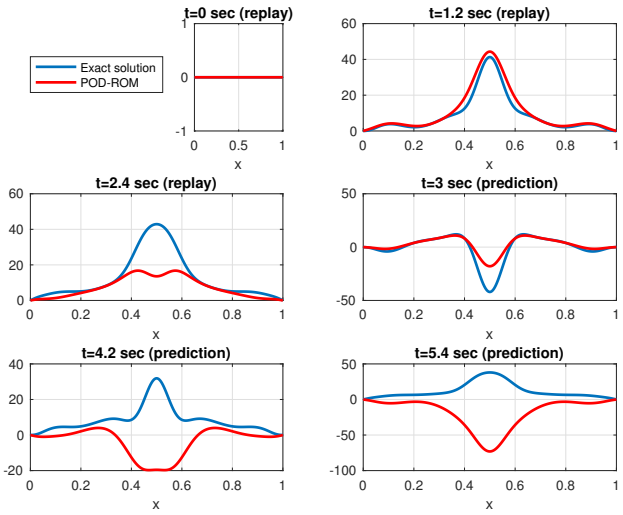


FIGURE 2.3 Prédiction et solution HF

La précision du modèle réduit est mauvaise, et comme attendu le modèle réduit semble avoir de grandes difficultés avec le caractère dynamique du terme source.
Une analyse spectrale du modèle réduit appris par la méthode DMD, permet une détection d'anomalie. En effet, nous souhaiterions que le spectre de la matrice apprise $\hat{A}$ soit fidèle au spectre de l'opérateur $\mathcal{L}$. Autrement dit, nous souhaiterions que toutes les valeurs propres de $\hat{A}$ soient réelles positives, décroissantes vers zéro et de module strictement inférieur à 1.
Représentons à présent le spectre de la matrice $\hat{A}$ afin d'analyser le modèle réduit appris.

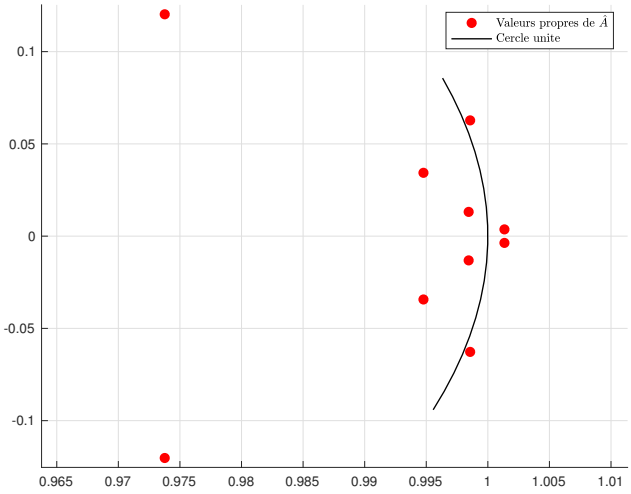


FIGURE 2.4 Spectre de $\hat{A}$ dans le plan complexe

Les valeurs propres de $\hat{A}$ ne sont donc pas toutes à l'intérieur du cercle unité, ce qui ne correspond pas à un problème de diffusion. D'autre part, deux valeurs propres semblent presque sur le cercle unité, ce qui laisse penser que le modèle réduit essaie d'apprendre le caractère conservatif de l'EDO guidant $\boldsymbol{\theta}(t)$.

Remarque 8. *Le terme source appris par le modèle réduit (i.e.* $\sum_{k=1}^{K} \phi_k(x)\hat{B}_k$*) est présenté ci-dessous :*

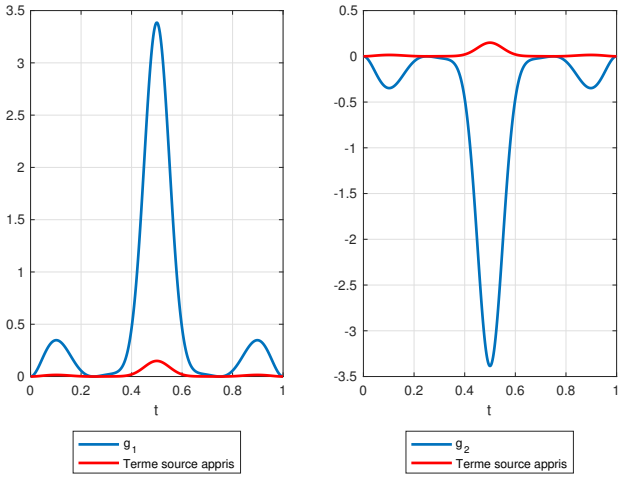


FIGURE 2.5 Terme source identifié

On peut observé que le terme source appris chauffe continûment mais peu par rapport au terme source du modèle (4.9).

Ainsi écrit sous cette forme nous ne sommes pas en mesure d'identifier la dynamique du problème (4.9).

Identification sous l'hypothèse d'un terme source dynamique

Pour pallier ce problème, nous allons utiliser l'extension présentée au **paragraphe 2.4.1**. Pour ce faire, introduisons les deux ensembles d'entraînement :

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ | & | & & | \\ | & | & & | \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ | & | & & | \end{bmatrix}, \qquad Y = \begin{bmatrix} | & | & & | \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ | & | & & | \\ | & | & & | \\ \boldsymbol{\theta}^2 & \boldsymbol{\theta}^3 & \cdots & \boldsymbol{\theta}^{N_t} \\ | & | & & | \end{bmatrix},$$

le modèle réduit sera alors de la forme :

$$\begin{cases} \mathbf{a}^{n+1} = \hat{A}\,\mathbf{a}^n + \hat{B}\,\boldsymbol{\theta}^n, \\ \boldsymbol{\theta}^{n+1} = \hat{D}\,\mathbf{a}^n + \hat{C}\,\boldsymbol{\theta}^n, \\ \mathbf{u}^{n+1} = \boldsymbol{\Phi}_r\,\mathbf{a}^{n+1}. \end{cases}$$

avec

$$\left(\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{D} & \hat{C} \end{array}\right) = Y X^{\dagger}$$

Notons que l'utilisation de la méthode avec empilement dans ce cas là, nous donne quatre sous-matrices.
Comme précédemment, comparons à présent la prédiction de la solution u obtenue en utilisant le modèle réduit avec la solution générée par le solveur HF. La prédiction est représentée par une ligne continue rouge, tandis que la solution HF est représentée par une ligne continue bleue.

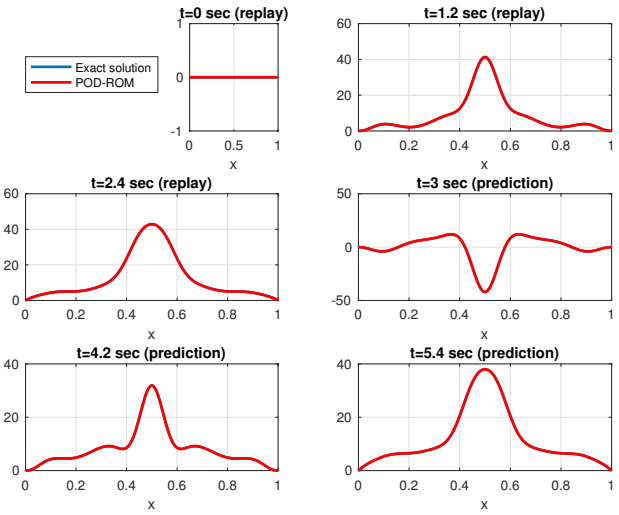


FIGURE 2.6 Prédiction du modèle réduit et solution HF : superposition parfaite des courbes à la norme de l'œil.

Cette fois-ci, le modèle réduit reproduit parfaitement la dynamique du modèle (4.9), et ce même pour un nombre de modes POD K faible comme le montre le tableau ci-dessous :

Nombre de modes POD (K)	**Erreur prédiction** L^2	**Erreur prédiction** H^1	**Erreur POD** L^2	**Erreur POD** H^1
1	0.3119	0.3698	0.2562	0.4001
2	0.0945	0.1614	0.0652	0.1516
3	0.0426	0.0522	0.0179	0.0404
4	0.0186	0.0246	0.0042	0.0110
5	0.0057	0.0101	9×10^{-4}	0.0027
10	1.083×10^{-5}	5.34×10^{-5}	8.6×10^{-7}	9.78×10^{-6}

TABLE 2.6 Erreurs relatives de prédiction et de réduction

Un autre moyen de s'assurer que le modèle (4.9) a bien été appris est de comparer les fonctions $\boldsymbol{g}_m$ avec les termes sources appris lors de l'identification i.e. $\sum_{k=1}^{K} \phi_k(x)\hat{B}_{k,m}$.

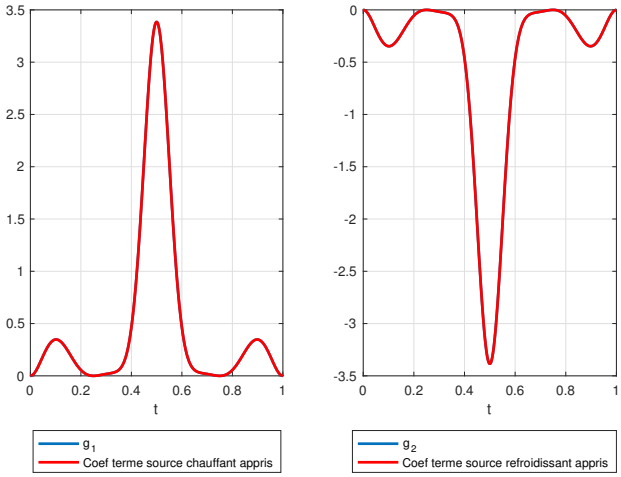


FIGURE 2.7 Les fonctions $\boldsymbol{g}_m$ (en bleu) et celle appris par le modèle réduit (en rouge)

On a donc parfaitement appris les deux régimes présents dans le terme source du modèle (4.9).

Validation de l'identification du modèle par analyse spectrale

Le modèle réduit appris, s'exprime sous la forme de la matrice suivante :

$$\hat{\mathbf{A}}_{DMD} := \left(\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{D} & \hat{C} \end{array}\right),$$

où la matrice $\hat{A}$ correspond à l'opérateur $\mathcal{L}$, $\hat{B}$ le terme source excitant le système, $\hat{D}$ est censé être nulle ($\boldsymbol{\theta}$ ne dépend pas de $\mathbf{a}^n$) et $\hat{C}$ correspond à la matrice de l'EDO. Afin de visualiser la matrice $\hat{\mathbf{A}}_{DMD}$, nous prendrons un rang de troncature $K = 6$. La matrice $\hat{\mathbf{A}}_{DMD}$ calculée par la méthode DMD vaut alors :

$$\left(\begin{array}{cccccc|cc}
\mathbf{0.9968} & -0.0044 & 0.0026 & -0.0005 & 0.0001 & 0.0000 & -1.0064 & 1.0064 \\
-0.0044 & \mathbf{0.9919} & 0.0078 & -0.0017 & 0.0007 & 0.0002 & -0.1532 & 0.1532 \\
0.0026 & 0.0078 & \mathbf{0.9805} & 0.0112 & -0.0051 & -0.0023 & 0.0303 & -0.0303 \\
-0.0005 & -0.0017 & 0.0112 & \mathbf{0.9823} & 0.0146 & 0.0056 & -0.0130 & 0.0130 \\
0.0001 & 0.0007 & -0.0051 & 0.0146 & \mathbf{0.9758} & -0.0156 & 0.0076 & -0.0076 \\
0.0000 & 0.0002 & -0.0022 & 0.0055 & -0.0154 & \mathbf{0.9700} & 0.0036 & -0.0036 \\
\hline
0.0000 & -0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.9980 & 0.0628 \\
-0.0000 & 0.0000 & 0.0000 & -0.0000 & 0.0000 & 0.0000 & -0.0628 & 0.9980
\end{array}\right)$$

Les coefficients en noir correspondent à la matrice $\hat{A}$. On remarque que cette matrice est proche de la matrice identité, autrement dit la solution u se dissipe lentement au cours du temps. Enfin cette matrice est symétrique, cette propriété bien que nullement imposée lors de la phase d'identification est importante (l'opérateur $\mathcal{L}$ est auto-adjoint).
Les coefficients violet et pourpre correspondent à la matrice $\hat{B}$. La première colonne, en violet, correspond à la fonction $\boldsymbol{g}_1$, tandis que la seconde colonne, en pourpre, correspond à la fonction $\boldsymbol{g}_2$. On peut voir que la relation "opposée" (i.e. $\boldsymbol{g}_1(x) = -\boldsymbol{g}_2(x), \quad \forall x$) entre les deux fonctions est parfaitement conservée.
Les coefficients en rouge correspondent quant à eux à la matrice $\hat{D}$ et sont, comme attendu, tous nuls.
Enfin, les coefficients en vert correspondent à la matrice $\hat{C}$ et sont fidèles au modèle (4.9), en effet une discrétisation de cette EDO nous donne la matrice

$$\begin{pmatrix} 1 & \omega\delta t \\ -\omega\delta t & 1 \end{pmatrix} \qquad \text{et} \qquad \omega\delta t = 0.0628.$$

Terminons cette section par une analyse spectrale du modèle réduit appris.

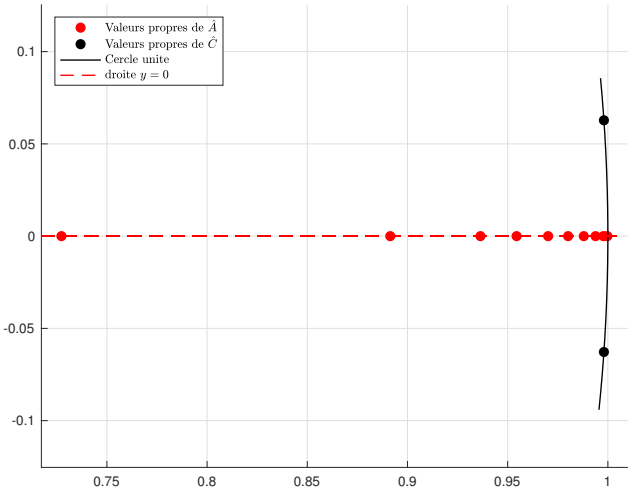


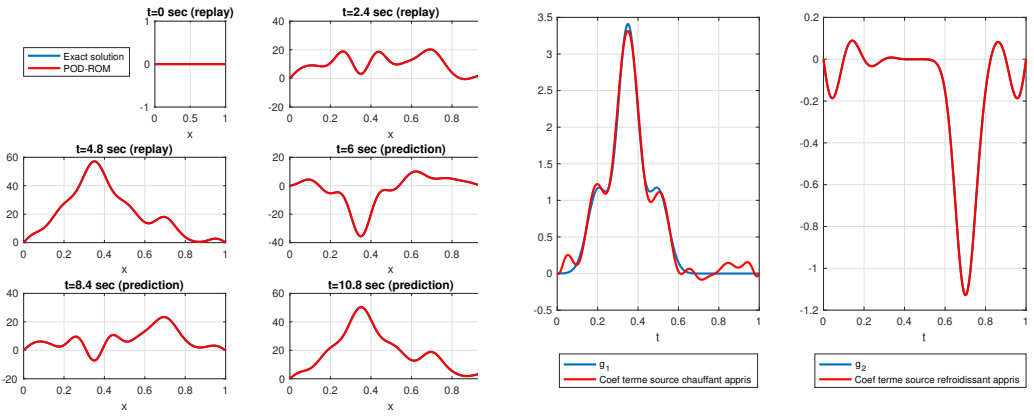
FIGURE 2.8 Spectre de $\hat{A}$ et $\hat{C}$ dans le plan complexe

Les valeurs propres de $\hat{A}$ correspondent bien à un phénomène diffusif, en effet, elles sont toutes de module strictement inférieur à 1 (la plus grande a pour module 0.9939) et décroissent vers 0. Par ailleurs, elles sont toutes réelles et positives.
En ce qui concerne les valeurs propres de $\hat{C}$, elles sont exactement inscrites sur le cercle unité. Cela est fidèle au caractère conservatif de l'EDO du système (4.9).

Remarque 9 (Termes sources non-symétriques)**.** *Si le système précédent a parfaitement été identifié, on peut penser que la symétrie des termes sources a pu faciliter cette identification. Ainsi nous considérons deux termes sources* $\boldsymbol{g}_1$ *et* $\boldsymbol{g}_2$ *ne présentant aucune relation linéaire. Nous les définissons ici comme*

$$\boldsymbol{g}_1(x) = \frac{1}{\tau\sqrt{100\pi}} e^{-\frac{(x-x_1)^2}{2\tau^2}} + 3\frac{1}{\tau\sqrt{100\pi}} e^{-\frac{(x-x_2)^2}{2\tau^2}} + \frac{1}{\tau\sqrt{100\pi}} e^{-\frac{(x-x_3)^2}{2\tau^2}},$$
$$\boldsymbol{g}_2(x) = -\frac{1}{\tau\sqrt{100\pi}} e^{-\frac{(x-x_4)^2}{2\tau^2}} + 2|x - x_0|^3 \sin(10\pi x),$$

avec $x_1 = 0.2$, $x_2 = 0.35$, $x_3 = 0.5$ *et* $x_4 = 0.7$, *de sorte que le refroidissement et le chauffage n'opèrent pas au même point. La figure ci-dessous présente l'identification de la dynamique ainsi que l'identification des fonctions* $\boldsymbol{g}_1$ *et* $\boldsymbol{g}_2$.



(a) Prédiction du modèle réduit (rouge) et solution HF (bleu) : superposition parfaite des courbes à la norme de l'œil

(b) Terme source identifié (en rouge) et terme source utilisé par le solveur HF (en bleu)

Si la prédiction de la solution u est là encore parfaite, le terme source $\boldsymbol{g}_1$ n'a pas parfaitement été identifié, une des raisons peut être la nature diffusive du problème, ainsi le rôle joué par le second membre est moins fort et donc plus difficile à identifier.

Il est bien connu que la réduction de la dimensionnalité pour un problème de diffusion est très efficace mais que ce n'est en revanche pas le cas pour des problèmes hyperboliques.

2.5.2 Problème d'ondes dynamique

Considérons à présent la propagation d'une onde perturbée par un terme source. Cette fois-ci, la difficulté ne sera pas seulement dû au terme source, mais également à la dynamique de l'EDP. Par ailleurs, la solution $\boldsymbol{\theta}$ était solution d'une EDO conservative, son comportement était alors périodique. Nous considérerons, ici le cas d'une EDO non-conservative, où la solution $\boldsymbol{\theta}$ se dissipera au cours du temps.

2.5.2.1 Mise en place de la boîte noire

On considérera également un terme source dynamique tensorisé. Mais ici la partie spatiale sera composée de quatre fonctions de formes distinctes, afin de mesurer l'identification du terme source dans un cas complexe.

Mathématiquement le modèle à identifier sera défini par :

$$
\begin{cases}
\partial_{tt}u(x,t) - c\partial_{xx}u(x,t) = \langle \boldsymbol{g}(x), \boldsymbol{\theta}(t)\rangle, \qquad \text{dans } [0,1]\times(0,T),\\
u(0,t)=0, \quad u(1,t)=0, \qquad \forall t\in(0,T),\\
\\
\dot{\boldsymbol{\theta}}(t) = \left(\begin{array}{cc|cc} -\alpha_1 & \omega_1 & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ -\omega_1 & -\alpha_1 & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{0}} & -\alpha_2 & \omega_2 \\ & & -\omega_2 & -\alpha_2 \end{array}\right)\boldsymbol{\theta}(t), \qquad \boldsymbol{\theta}(0) = \begin{pmatrix}0\\1\\0\\1\end{pmatrix},\\
\\
\boldsymbol{g}_1(x) = q_1\Big(\frac{1}{\tau\sqrt{100\pi}}e^{-\frac{(x-x_1)^2}{2\tau^2}} + 3\frac{1}{\tau\sqrt{100\pi}}e^{-\frac{(x-x_2)^2}{2\tau^2}} + \frac{1}{\tau\sqrt{100\pi}}e^{-\frac{(x-x_3)^2}{2\tau^2}}\Big),\\
\boldsymbol{g}_2(x) = -q_2\Big((x_4-|x-x_4|)^2 + 0.05\sin(20\pi x)\Big),\\
\boldsymbol{g}_3(x) = q_3\Big(\frac{1}{\tau\sqrt{100\pi}}e^{-\frac{(x-x_5)^2}{2\tau^2}} + 2(x-x_3)^3\sin(10\pi x)\Big),\\
\boldsymbol{g}_4(x) = -q_4\mathbb{1}_{[x_6,x_7]},\\
u(x,0) = u_0(x), \qquad\qquad \text{dans } [0,1]
\end{cases}
\tag{2.30}
$$

Le solution u sera obtenue à l'aide d'une méthode des différences finies, et son approximation u_h stockée dans la matrice $\mathbf{U}$. L'échantillonnage de $\boldsymbol{\theta}$ ainsi que son stockage seront réalisés de la même façon que précédemment. Le tableau ci-dessous présente les paramètres utilisés pour la génération des données.

Paramètres	**Valeurs**
Espérances du terme source $\boldsymbol{g}_1$ (x_1, x_2, x_3)	$(0.2, 0.35, 0.5)$
Espérance du terme source $\boldsymbol{g}_2$ (x_4)	0.5
Espérance du terme source $\boldsymbol{g}_3$ (x_5)	0.8
Intervalle du terme source $\boldsymbol{g}_4$ $([x_6, x_7])$	$[0.75, 0.9]$
Variance du terme source (τ)	0.05
Intensité du terme source (q_1)	5×10^{-4}
Intensité du terme source (q_2)	10^{-3}
Intensité du terme source (q_3, q_4)	10^{-2}
Fréquence de l'activation du terme source (ω_1)	$4\,\pi$
Vitesse de dissipation du terme source (α_1)	0
Fréquence de l'activation du terme source (ω_2)	$6\,\pi$
Vitesse de dissipation du terme source (α_2)	0.25
Célérité de l'onde (c)	0.5
Condition initiale (u_0)	0
Temps final (T)	12 sec
Pas d'espace (δx)	5×10^{-3}
Pas de temps (δt)	$\frac{0.3}{c}\delta x$

TABLE 2.7 Paramètres physiques du problème

La figure ci-dessous représente la solution u à différents instants.

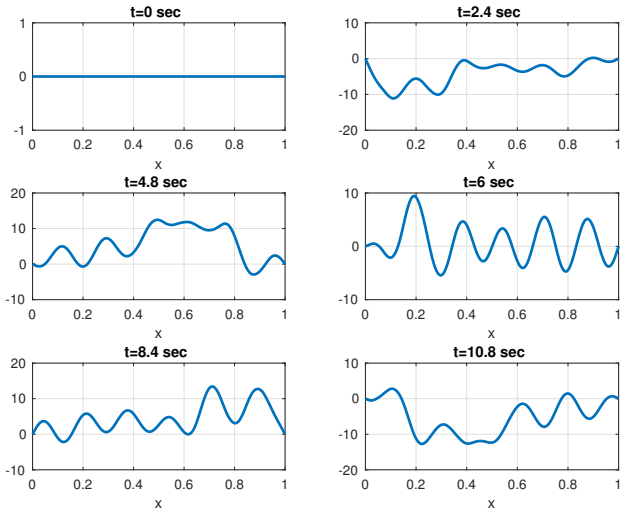


FIGURE 2.10 Données HF à différents pas de temps

On remarque que le problème est fortement dynamique et que l'information contenue dans **U** est donc très riche, ce qui rend la réduction difficile.

2.5.2.2 Identification de la dynamique et apprentissage d'un modèle réduit

À présent nous n'avons plus accès qu'aux matrices **U** et **Θ**. Nous allons donc, à partir de ces matrices de données tenter d'identifier la dynamique et d'apprendre un modèle réduit.

Réduction de la dimensionnalité

Nous allons à présent réduire l'information contenue dans la matrice **U** par la méthode POD. Les 6 premiers modes POD continus $\phi_k(x)$ sont présentés ci-dessous.

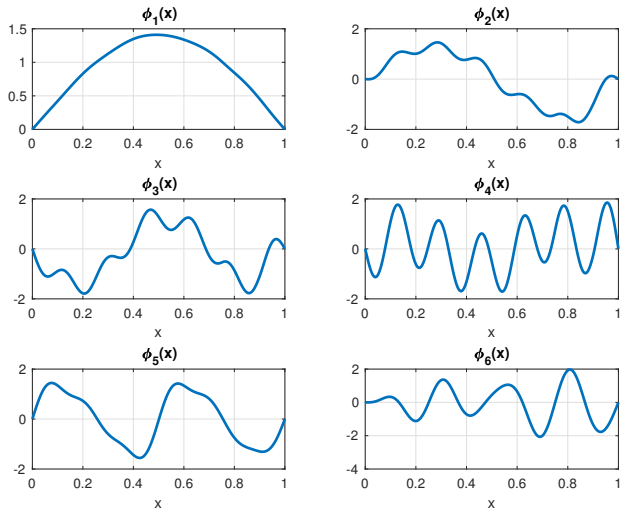


FIGURE 2.11 Les 6 premiers modes POD

La richesse de l'information contenue dans **U** explique la diversité des modes POD. Le tableau ci-dessous présente comme précédemment les erreurs de réduction en norme L^2 et en norme H^1.

Nombre de modes POD (K)	**Erreur** L^2	**Erreur** H^1
1	0.5847	0.9772
5	0.2663	0.5503
10	0.0670	0.1845
20	0.0060	0.0401
30	0.0017	0.0185
40	7.4810×10^{-4}	0.0117
50	4.2359×10^{-4}	0.0088
60	2.8966×10^{-4}	0.0070
80	1.4079×10^{-4}	0.0048
100	8.6458×10^{-5}	0.0034

TABLE 2.8 Erreurs relatives de réduction

Comme attendu, l'erreur de réduction dans ce cas décroît beaucoup moins vite, il nous faut 40 à 50 modes POD pour avoir une erreur de réduction tout juste raisonnable, et même avec 100 modes l'erreur en norme H^1 est de l'ordre de 10^{-3}. Cela s'explique par la richesse d'information de notre problème.
Afin de ne pas polluer les résultats par une erreur de réduction trop importante, nous prendrons $K = 50$.

Identification par un modèle d'ordre 1 en temps

Nous allons tout d'abord essayer d'apprendre un modèle réduit d'ordre 1 en temps de la forme :

$$\begin{cases} \mathbf{a}^{n+1} = \hat{A}\,\mathbf{a}^n + \hat{B}\,\boldsymbol{\theta}^n, \\ \boldsymbol{\theta}^{n+1} = \hat{D}\,\mathbf{a}^n + \hat{C}\,\boldsymbol{\theta}^n, \\ \mathbf{u}^{n+1} = \boldsymbol{\Phi}_r\,\mathbf{a}^{n+1}. \end{cases}$$

Comme précédemment, comparons la prédiction du modèle réduit et les données HF.

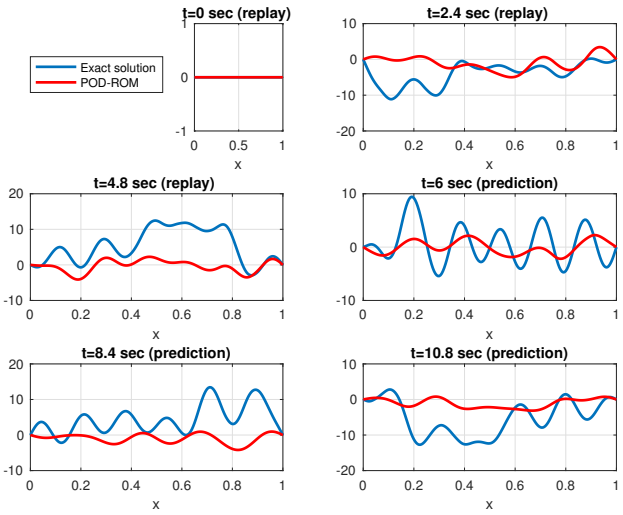


FIGURE 2.12 Comparaison entre les données HF (en bleu) et la prédiction du modèle réduit (en rouge)

On observe que le modèle réduit n'arrive pas à prédire correctement le comportement de l'onde au cours du temps.
Vérifions si cette mauvaise prédiction aurait pu être anticipée par l'analyse spectrale du modèle réduit. Pour cela, présentons les valeurs propres de la matrice apprise $\hat{A}$.

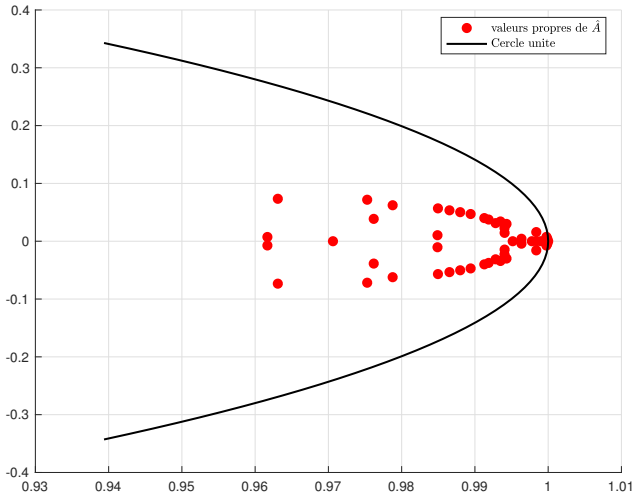


FIGURE 2.13 Valeurs propres du modèle réduit appris (points rouge) contenues dans le cercle unité (ligne continue noire)

On s'aperçoit que l'on a identifié un modèle EDP ne correspondant pas au système (2.30). En effet, l'équation des ondes est une EDP hyperbolique conservative, ainsi les valeurs propres devraient toutes être sur le cercle unité. Or aucune valeur propre n'est sur le cercle unité, nous avons donc un modèle réduit non-conservatif qui semble plus proche d'un problème diffusif.
Cela montre encore une fois, l'importance du choix de la forme du modèle réduit que l'on souhaite apprendre.

Identification par un modèle d'ordre 2 en temps

Afin d'identifier correctement la dynamique du modèle (2.30) et ainsi prédire avec précision le comportement de la solution u, nous utiliserons l'extension de la méthode DMD au cas d'un modèle dynamique d'ordre 2 en temps, décrite au **paragraphe 2.4.3**. Ainsi le modèle réduit sera de la forme :

$$\begin{cases} \mathbf{a}^{n+1} = \hat{A}_1\, \mathbf{a}^{n-1} + \hat{A}_2\, \mathbf{a}^{n} + \hat{B}\, \boldsymbol{\theta}^{n}, \\ \boldsymbol{\theta}^{n+1} = \hat{D}_1\, \mathbf{a}^{n-1} + \hat{D}_2\, \mathbf{a}^{n} + \hat{C}\, \boldsymbol{\theta}^{n}, \\ \mathbf{u}^{n+1} = \boldsymbol{\Phi}_r\, \mathbf{a}^{n+1}. \end{cases}$$

Cette fois-ci, nous avons prédit correctement la propagation de l'onde au cours du temps comme le montre la figure ci-dessous.

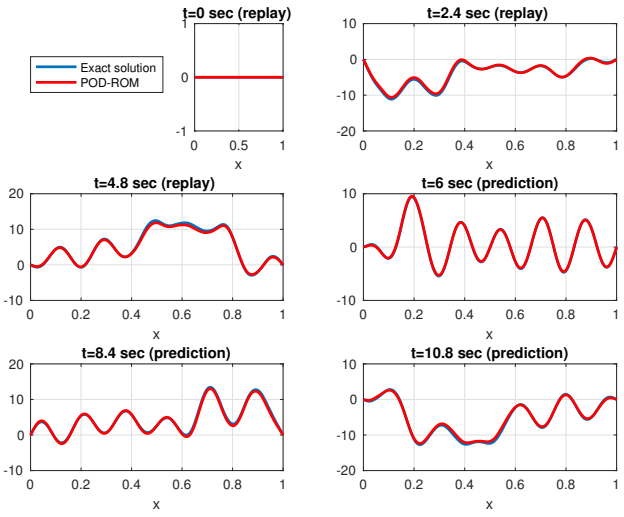
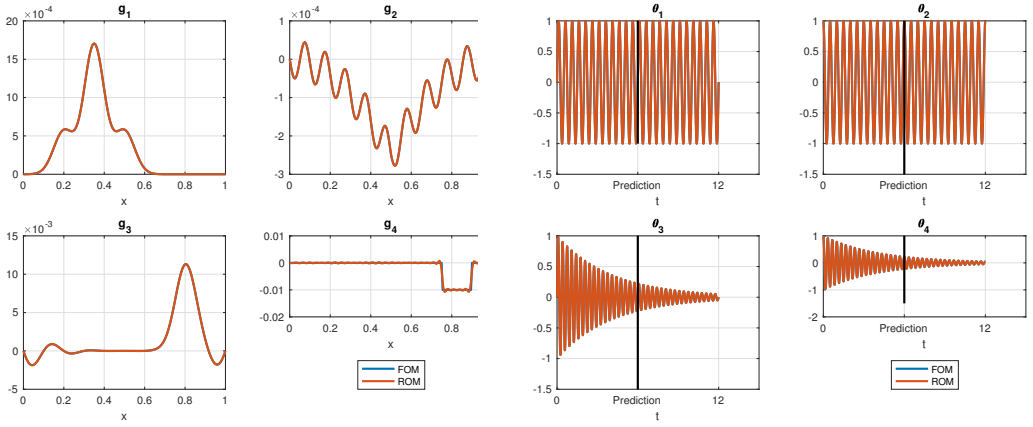


FIGURE 2.14 Comparaison entre la solution HF (en bleu) et la prédiction du du modèle réduit (en rouge)

Ceci est renforcé par les figures ci-dessous qui comparent les fonctions $\boldsymbol{g}_m$ et $\boldsymbol{\theta}_m$ utilisées par le solveur HF, avec celles apprises par le modèle réduit.



(a) Comparaison entre les fonctions $\boldsymbol{g}_m$ utilisées par le solveur HF (en bleu) et celles prédites par le modèle réduit (en rouge) : quasi superposition des courbes à la norme de l'œil

(b) Comparaison entre les fonctions $\boldsymbol{\theta}_m$ utilisées par le solveur HF (en bleu) et celles prédites par le modèle réduit (en rouge) : superposition parfaite des courbes à la norme de l'œil

On a parfaitement identifié les termes sources, même le terme source discontinu $\boldsymbol{g}_4$ a bien été identifié.

Remarque 10. *Il semble que dans le cas de l'équation des ondes, l'identification du terme source soit meilleure que dans le cas d'une équation de diffusion. Cela renforce l'intuition que le caractère dissipatif de l'équation de diffusion rend, en quelque sorte, moins important le rôle joué par le terme source. Ainsi on peut "coller" aux données sans avoir totalement appris les bons termes sources. Dans le cas de l'équation des ondes, une mauvaise identification des termes sources détériore sévèrement la qualité du modèle réduit.*

Validation de l'identification du modèle par analyse spectrale

Nous allons maintenant analyser le modèle réduit construit, afin de s'assurer de son bon comportement en temps long. Tout d'abord, rappelons que la matrice obtenue est de la forme :

$$\mathbf{A}_{DMD} := \left(\begin{array}{c|c|c} \hat{A}_1 & \hat{A}_2 & \hat{B} \\ \hline \hat{D}_1 & \hat{D}_2 & \hat{C} \end{array}\right),$$

les matrices $\hat{A}_1$ et $\hat{A}_2$ représentent l'équation des ondes, $\hat{B}$ représente les fonctions $\boldsymbol{g}_i$, et enfin $\hat{C}$ correspond à la matrice de l'EDO. Enfin les matrices $\hat{D}_1$ et $\hat{D}_2$ sont censées

être nulles. Le tableau ci-dessous présente la norme max (i.e. $\max_{i,j} |A_{i,j}|$) des matrices $\hat{D}_1$ et $\hat{D}_2$, afin de s'assurer qu'elles soient bien nulles.

Matrice	**Norme max**
$\hat{D}_1$	1.8474×10^{-13}
$\hat{D}_2$	2.0872×10^{-13}

TABLE 2.9 Norme max des matrices $\hat{D}_1$ et $\hat{D}_2$.

À présent étudions la matrice $\hat{C}$. Une discrétisation de la matrice de l'EDO issue du modèle (2.30), nous donne la matrice

$$\left(\begin{array}{cc|cc} 1-\alpha_1\delta t & \omega_1\delta t & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ -\omega_1\delta t & 1-\alpha_1\delta t & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{0}} & 1-\alpha_2\delta t & \omega_2\delta t \\ & & -\omega_2\delta t & 1-\alpha_2\delta t \end{array}\right), \quad \text{avec} \begin{cases} 1-\alpha_1\delta t = 1, \\ \omega_1\delta t = 0.0377, \\ 1-\alpha_2\delta t = 0.9992, \\ \omega_2\delta t = 0.0565. \end{cases}$$

La matrice $\hat{C}$ calculée vaut quant à elle

$$\begin{pmatrix} 0.9993 & 0.0377 & -0.0000 & -0.0000 \\ -0.0377 & 0.9993 & -0.0000 & -0.0000 \\ -0.0000 & -0.0000 & 0.9977 & 0.0565 \\ 0.0000 & -0.0000 & -0.0565 & 0.9977 \end{pmatrix}.$$

On a donc retrouvé la matrice permettant de faire évoluer la solution $\boldsymbol{\theta}$.

Enfin étudions le spectre de la matrice apprise $\begin{bmatrix} \hat{A}_1 & \hat{A}_2 \end{bmatrix}$, correspondant à l'équation des ondes. Cette matrice n'est cependant pas carrée, ainsi afin de pouvoir étudier son spectre, ajoutons l'égalité triviale $\mathbf{a}^n = \mathbf{0}_K\, \mathbf{a}^{n-1} + \mathbf{I}_K\, \mathbf{a}^n$, et étudions alors le spectre de la matrice

$$\hat{A}_{EDP} := \left(\begin{array}{c|c} \mathbf{0}_K & \mathbf{I}_K \\ \hline \hat{A}_1 & \hat{A}_2 \end{array}\right).$$

La figure ci-dessous présente le spectre de la matrice $\hat{A}_{EDP}$ ainsi que le spectre de la matrice $\hat{C}$ correspondant à la partie EDO.

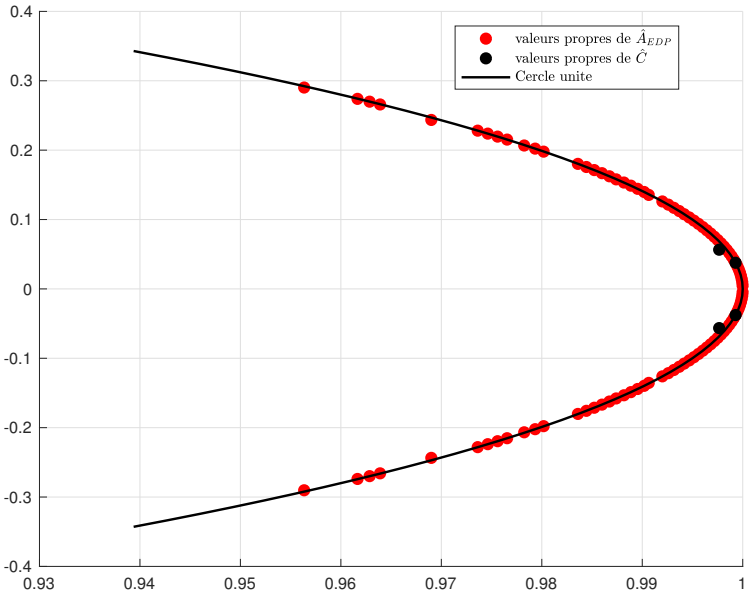


FIGURE 2.16 En rouge, les valeurs propres de la matrice $\hat{A}_{EDP}$ et en noire, les valeurs propres de la matrice $\hat{C}$

Toutes les valeurs propres de $\hat{A}_{EDP}$ sont sur le cercle unité, cela est cohérent avec l'équation des ondes. On a donc appris un modèle réduit conservatif. En ce qui concerne l'EDO, α_1 valant zéro, $\boldsymbol{\theta}_1$ et $\boldsymbol{\theta}_2$ valent alors

$$\boldsymbol{\theta}_1(t) = sin(\omega_1 t) \quad \text{et} \quad \boldsymbol{\theta}_2(t) = cos(\omega_1 t)$$

Il est logique que les deux premières valeurs propres de $\hat{C}$ soient alors de module égale à 1. En ce qui concerne les deux suivantes, α_2 valant 0.25, on alors

$$\boldsymbol{\theta}_3(t) = sin(\omega_2 t)\, e^{-0.25t}, \quad \text{et} \quad \boldsymbol{\theta}_4(t) = cos(\omega_2 t)\, e^{-0.25t}$$

Ainsi $\boldsymbol{\theta}_3$ et $\boldsymbol{\theta}_4$ se dissipent au cours du temps, ce qui explique les deux valeurs propres de module strictement inférieur à 1.

Conclusion

La méthode DMD et ses extensions proposées dans ce chapitre permettent l'apprentissage d'un modèle réduit précis et stable en temps long. De plus, la complexité du processus d'apprentissage du modèle réduit est linéaire en N_t (nombre de pas de temps).

En revanche, cela requiert un minimum de connaissance a priori du modèle EDP, telle que l'ordre de dérivée en temps. Cette connaissance nous permet ainsi de choisir la bonne forme pour notre modèle réduit. L'analyse numérique et les applications numériques viennent confirmer la pertinence de cette méthode pour des EDP paraboliques et hyperboliques, ainsi que pour des couplages EDP-EDO. Enfin, par une analyse spectrale du modèle réduit, nous pouvons détecter les anomalies lors de l'identification de la dynamique. Un cas particulier de terme source n'ayant pas été traité dans ce chapitre est celui où la fonction $\boldsymbol{\theta}$ dépend de la solution u, i.e. $\boldsymbol{\theta}(t) = \boldsymbol{\theta}(u(\boldsymbol{x}, t))$. Ce cas, très utilisé en ingénierie, correspond aux systèmes contrôlés par commutation et sera étudié au chapitre suivant.

Chapitre 3

Identification d'un système contrôlé par commutation

Résumé Ce chapitre est consacré à l'identification, à partir de données synthétiques générées par un solveur dit *haute-fidélité* (HF), d'un système dynamique contrôlé par commutation. Le modèle représentant ce système est non-linéaire, ce qui rend l'identification complexe. Dans ce travail, on apprendra un modèle réduit fidèle à chaque sous-système linéaire à l'aide de la méthode présentée au **Chapitre 2**. La fonction non-linéaire $\boldsymbol{\theta}$, guidant les commutations, sera quant à elle apprise grâce à un *Réseau de neurones artificiels* (ANN). La méthode présentée dans ce chapitre sera ensuite appliquée au cas d'un commutateur thermique afin de mesurer la précision du modèle réduit appris.

Mots-clés. Data-driven model ; Switched control ; Heating system ; Reduced Order Model ; DMD ; POD ; ANN ; SVC ; Spectral Clustering ; Machine Learning.

Introduction

Dans le chapitre précédent, nous avions présenté l'identification d'une EDP d'évolution linéaire comportant un second membre dynamique. Cependant, dans bien des cas chercher à faire dépendre le terme source de la seule variable temporelle semble restrictif. C'est par exemple le cas des systèmes contrôlés. Ce cas là n'a pas pu être abordé au chapitre précédent, car dès lors que le terme source dépend de la solution du système, celui-ci devient non-linéaire. Dans ce chapitre, on s'intéressera aux systèmes dynamiques contrôlés par commutation. Imaginons par exemple une balle rebondissant sur le sol sans perte d'énergie. Modéliser le système tel quel est complexe, en

revanche modéliser la phase où la balle tombe et la phase où la balle remonte est beaucoup plus simple ([110]). Ainsi on choisira de modéliser ce problème global par deux sous-systèmes représentant la dynamique de chaque phase et commutant à l'aide d'une règle de contrôle. Le changement de dynamique s'opère en fonction de la position de la balle. Cette classe de systèmes dynamiques, correspond aux systèmes contrôlés par commutation, et on appelle sous-système le système associé à chaque mode de commutation. Les systèmes contrôlés par commutation ont été largement étudiés ([115], [113], [170]) et on les retrouve dans de nombreux domaines tels que l'automobile ([2], [65]) ou encore la robotique.

Depuis plusieurs années, et principalement en robotique, des travaux ont porté sur l'identification des sous-systèmes afin de construire une synthèse de contrôle adéquate. Ainsi en apprenant l'action du robot sur son environnement on est en mesure de connaître les actions devant être réalisées par le robot. On peut citer les travaux de Lee & al ([110]), où les auteurs font l'hypothèse que le système contrôlé par commutation est composé de sous-systèmes lisses afin de détecter les commutations. Ainsi les auteurs arrivent à isoler chaque sous-système, puis à apprendre la dynamique de chacun à l'aide d'un *Processus Gaussien* (GP, [142]). Notons que ce travail porte sur des systèmes dynamiques de petites dimensions, tandis que notre cas la dimension N_x des systèmes dynamiques est supposée très grande, ce qui rend difficile l'identification. Dans le cas d'*équation aux dérivées partielles* (EDP), Peitz & al ([136]) transforment un problème de contrôle en un problème de contrôle par commutation. Puis en utilisant la théorie des opérateurs de Koopman appliquée aux systèmes dynamiques, ils identifient la dynamique des sous-systèmes ce qui permet alors d'accélérer la recherche d'un contrôle optimal. Enfin, Proctor & al ([137]) ont introduit la méthode *Dynamic Mode Decomposition with control* (DMDc) afin d'apprendre séparément la dynamique du système et l'action du contrôle sur le système.

Dans ce chapitre, nous présenterons tout d'abord les systèmes contrôlés par commutation. Dans une seconde partie nous expliquerons comment identifier la dynamique de chaque sous-système. La troisième partie sera quant à elle dévouée à l'apprentissage de la règle de contrôle lorsque les données issues d'un capteur sont disponibles. Lorsque ce n'est pas le cas, une autre approche sera présentée à la quatrième partie. Enfin la cinquième partie, traitera du cas où même l'historique des commutations n'est pas disponible. Dans la partie numérique, nous présenterons l'apprentissage d'un modèle réduit pour un commutateur thermique.

3.1 Modèle et solveur haute-fidélité

Notre système contrôlé par commutation sera composé d'une famille de M sous-systèmes, que nous considérerons linéaires, ainsi que d'une règle de contrôle guidant la commutation entre chaque sous-système. Cette règle de contrôle, que l'on notera H est dépendante de la solution du système et potentiellement du sous-système précédent. Dans bien des cas, le contrôle ne dépend pas de la solution u sur tout le domaine, mais seulement de son évaluation ponctuelle mesurée par des capteurs. Pour simplifier les notations, on supposera que le contrôle ne dépend que de la mesure d'un capteur, et on notera ℓ la forme linéaire correspondant à cette mesure. La figure 3.1 résume le fonctionnement de notre système.

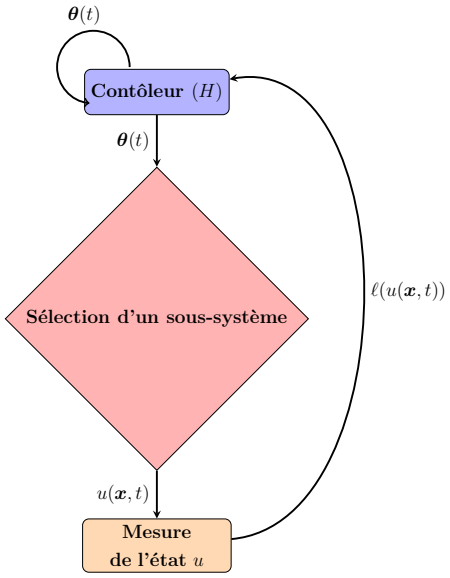


FIGURE 3.1 Schéma de contrôle par commutation

Pour chaque temps t, $\boldsymbol{\theta}(t)$ est un vecteur ayant une seule composante non nulle, celle de l'indice du sous-système sélectionné. Ainsi au temps t, si le contrôleur choisit le sous-système m, on aura $\boldsymbol{\theta}_n(t) = \delta_{m,n}$, pour tout $n \in \{1, \cdots, M\}$. Ensuite l'état du système évoluera selon la dynamique du sous-système m, puis sera réévalué au pas de temps suivant afin d'ajuster le contrôle.

Mathématiquement, notre système pourra être modélisé par :

$$\partial_t\, u(\boldsymbol{x},t) + \langle \boldsymbol{\mathcal{L}}(\boldsymbol{x})\, u(\boldsymbol{x},t), \boldsymbol{\theta}(t)\rangle = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t)\rangle, \tag{3.1}$$

$$\text{avec} \qquad \boldsymbol{\theta}(t) = H\Big(\ell\Big(u(\cdot,t)\Big), \boldsymbol{\theta}(t^-)\Big) \tag{3.2}$$

où

$$t^- := \underline{\lim}_{s \to t} s, \qquad \boldsymbol{\mathcal{L}} = \begin{bmatrix} \boldsymbol{\mathcal{L}}_1 \\ \boldsymbol{\mathcal{L}}_2 \\ \vdots \\ \boldsymbol{\mathcal{L}}_M \end{bmatrix}, \qquad \text{et} \quad \boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_M \end{bmatrix}$$

avec $\boldsymbol{\mathcal{L}}$ le vecteur contenant les opérateurs linéaires associés à chaque sous-système et $\boldsymbol{g}$ le vecteur contenant les termes sources associés à chaque sous-système.

Comme dans le chapitre précédent, nous souhaitons nous faire une idée de l'écriture matricielle du système que les données générées par notre *boîte noire* satisfont. On va donc approcher la solution u de l'équation (3.1) par une fonction u_h à partir d'une méthode numérique. Considérons, pour ce faire, un schéma d'Euler implicite pour la discrétisation en temps et la méthode des éléments finis pour la discrétisation spatiale. On a alors

$$\frac{1}{\delta t}\mathbf{M}\,\mathbf{u}^{n+1} + \sum_{m=1}^{M} \boldsymbol{\theta}_m^n\, \boldsymbol{\mathcal{L}}_{m,h}\,\mathbf{u}^{n+1} = \sum_{m=1}^{M} \boldsymbol{\theta}_m^n\, \mathcal{G}_m + \frac{1}{\delta t}\mathbf{M}\,\mathbf{u}^n \tag{3.3}$$

avec $\mathbf{M}$ la matrice de masse, $\boldsymbol{\theta}^n$ la discrétisation en temps de $\boldsymbol{\theta}$, $\boldsymbol{\mathcal{L}}_{m,h}$ l'opérateur $\boldsymbol{\mathcal{L}}_m$ discret, et $\mathcal{G}_m$ la discrétisation du terme source $\boldsymbol{g}_m$.
Étant donné que $\boldsymbol{\theta}$ a une unique composante égale à 1 et toutes les autres nulles, on peut alors réécrire (3.3), comme

$$\sum_{m=1}^{M} \boldsymbol{\theta}_m^n \left(\frac{1}{\delta t}\mathbf{M} + \boldsymbol{\mathcal{L}}_{m,h}\right)\mathbf{u}^{n+1} = \frac{1}{\delta t}\mathbf{M}\,\mathbf{u}^n + \sum_{m=1}^{M} \boldsymbol{\theta}_m^n\, \mathcal{G}_m \tag{3.4}$$

et finalement on obtient

$$\mathbf{u}^{n+1} = \sum_{m=1}^{M} \boldsymbol{\theta}_m^n \left(\underbrace{\frac{1}{\delta t}\Big(\frac{1}{\delta t}\mathbf{M} + \boldsymbol{\mathcal{L}}_{m,h}\Big)^{-1}\mathbf{M}}_{=:\,\mathcal{A}_m}\,\mathbf{u}^n + \underbrace{\Big(\frac{1}{\delta t}\mathbf{M} + \boldsymbol{\mathcal{L}}_{m,h}\Big)^{-1}\mathcal{G}_m}_{=:\,\mathcal{B}_m} \right) \tag{3.5}$$

Ainsi les données générées par notre solveur HF, satisfont un système de la forme :

$$\mathbf{u}^{n+1} = \sum_{m=1}^{M} \boldsymbol{\theta}_m^n \Big(\mathcal{A}_m\,\mathbf{u}^n + \mathcal{B}_m\Big). \tag{3.6}$$

Tout comme auparavant, les données générées seront stockées dans une matrice $\mathbf{U}$. Par ailleurs, le solveur HF générera aussi la matrice de corrélation $\mathcal{S}$ définie par

$$\mathcal{S} = \mathbf{U}^T\,\mathbf{M}\,\mathbf{U}.$$

Enfin l'historique des activations des sous-systèmes sera conservé dans la matrice $\mathbf{\Theta}$ définie par

$$\mathbf{\Theta}_{m,n} := \boldsymbol{\theta}_m^n.$$

Remarque 11. *En notant N_t le nombre de pas de temps et N_x le nombre de degrés de liberté, on a alors $\mathbf{U} \in \mathbb{R}^{N_x \times N_t}$ tandis que $\mathbf{\Theta} \in \mathbb{R}^{M \times N_t - 1}$.*

3.2 Identification de la dynamique et du terme source de chaque sous-système

Dans cette section, nous cherchons à identifier la dynamique de chaque sous-système, autrement dit à retrouver les matrices $\mathcal{A}_m$ et les vecteurs $\mathcal{B}_m$ du système (3.6). Ainsi à l'aide de la matrice de corrélation $\mathcal{S}$ générée au préalable, on construira une base POD, et on substituera $\mathbf{U}$ par sa représentation basse dimension $\mathbf{A}$. On ne cherchera alors plus des matrices $\mathcal{A}_m$ et des vecteurs $\mathcal{B}_m$, mais leurs représentations basses dimensions que l'on notera A_m et B_m. Il nous faut ensuite organiser les données des ensembles d'entrainement afin que le modèle réduit ait la forme (3.6). Comme précédemment, l'ensemble d'entrainement de sortie Y sera

$$Y = \begin{bmatrix} | & | & & | \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t} \\ | & | & & | \end{bmatrix},$$

En revanche l'ensemble d'entrainement d'entrée sera lui plus complexe.

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{\Theta}_{1,1}\mathbf{a}^1 & \mathbf{\Theta}_{1,2}\mathbf{a}^2 & \cdots & \mathbf{\Theta}_{1,N_t-1}\mathbf{a}^{N_t-1} \\ | & | & & | \\ \vdots & \vdots & \vdots & \vdots \\ | & | & & | \\ \mathbf{\Theta}_{M,1}\mathbf{a}^1 & \mathbf{\Theta}_{M,2}\mathbf{a}^2 & \cdots & \mathbf{\Theta}_{M,N_t-1}\mathbf{a}^{N_t-1} \\ | & | & & | \end{bmatrix},$$

Une écriture simplifiée de cet ensemble X peut être obtenue à l'aide du produit de Kronecker,

$$X = \Big(\mathbf{\Theta} \otimes X_1\Big)_{:,\pi}, \quad \text{avec} \quad X_1 = \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ | & | & & | \end{bmatrix}, \tag{3.7}$$

$$\text{et} \begin{cases} \pi : \{1, \cdots, N_t - 1\} \to \{1, \cdots, (N_t - 1)^2\}, \\ n \mapsto N_t(n-1) + 1 \end{cases} \tag{3.8}$$

Le terme source sera traité comme au chapitre précédent, en posant $Z = \mathbf{\Theta}$, puis en cherchant une matrice B telle que $B\,Z$ contiennent l'historique des excitations du système.
Nous avons, à présent, tout pour identifier la dynamique et le terme source de chaque sous-système. Considérons le problème d'optimisation suivant,

$$\Big(\hat{A}, \hat{B}\Big) = \underset{(A,B)}{\arg\min}\ \mathfrak{L}(A, B) := \frac{1}{2}\|A\,X + B\,Z - Y\|_F^2,$$

on a ainsi

$$\begin{cases} \hat{B} = \Big(YX^\dagger XZ^\dagger - YZ^\dagger\Big)\Big(ZX^\dagger XZ^\dagger - \mathbf{I}_M\Big)^{-1}, \\ \hat{A} = YX^\dagger - \hat{B}ZX^\dagger. \end{cases}$$

Finalement les matrices de chaque sous systèmes sont données par

$$\begin{bmatrix} \hat{A}_1 & \hat{A}_2 & \cdots \hat{A}_M \end{bmatrix} = \hat{A}.$$

Tandis que les vecteurs sont donnés par

$$\begin{bmatrix} \hat{B}_1 & \hat{B}_2 & \cdots \hat{B}_M \end{bmatrix} = \hat{B}.$$

Remarque 12. *La matrice X a KM lignes et $N_t - 1$ colonnes, ainsi la complexité de l'algorithme est en $\mathcal{O}\Big(N_t\big(2K^2M^2 + 2K^2M + 2KM^2\big)\Big)$. Et donc linéaire par rapport à N_t. En outre, K et M étant souvent petit, cela reste raisonnable.*

Accélération de l'identification par empilement

Afin d'accélérer le processus d'identification, nous allons utiliser la méthode par empi-

lement introduite au **Chapitre 2**. L'ensemble d'entrainement d'entrée X sera alors,

$$X = \Big(\mathbf{\Theta} \otimes X_1\Big)_{:,\pi}, \quad \text{avec} \quad X_1 = \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1} \\ | & | & & | \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \tag{3.9}$$

$$\text{et} \begin{cases} \pi : \{1, \cdots, N_t - 1\} \to \{1, \cdots, (N_t-1)^2\}, \\ n \mapsto N_t(n-1) + 1 \end{cases} \tag{3.10}$$

et on obtient alors

$$\Big[\Big[\hat{A}_1 \quad \hat{B}_1\Big] \quad \Big[\hat{A}_2 \quad \hat{B}_2\Big] \quad \cdots \Big[\hat{A}_M \quad \hat{B}_M\Big]\Big] = Y X^\dagger.$$

Cette fois-ci, la complexité est en $\mathcal{O}\Big(N_t\Big(K^2M^2 + K^2M\Big)\Big)$ pour le procédé d'identification.

On a donc identifier la dynamique et le terme source de chaque sous-système. À présent, il nous faut apprendre la règle de contrôle H, afin de pouvoir prédire la sélection des sous-systèmes. Nous considérerons dans la suite de ce travail trois niveaux de *non-intrusion* que l'on définira comme suit,

- **Quasi non-instrusif :** dans ce cadre, nous supposerons savoir que le contrôle est dirigé par la mesure d'un capteur. Les matrices $\mathbf{U}$, $\mathcal{S}$ et $\mathbf{\Theta}$ ainsi que l'historique des mesures du capteur sont à notre disposition.
- **Non-instrusif :** dans ce cadre, nous chercherons à contrôler le système qu'à partir de la solution sur tout le domaine. Les matrices $\mathbf{U}$ et $\mathcal{S}$ et $\mathbf{\Theta}$ sont à notre disposition.
- **Totalement non-instrusif :** dans ce cadre, nous ne connaissons pas même l'historique des décisions prises par le contrôle. Seules les matrices $\mathbf{U}$ et $\mathcal{S}$ sont à notre disposition.

3.3 Apprentissage de la règle de contrôle à partir des mesures du capteur

Dans cette section, nous serons dans le cadre *Quasi non-intrusif*. La commutation des sous-systèmes s'opère par la fonction H, nous voudrions donc apprendre cette fonction. Il paraît déraisonnable d'approcher la fonction H à l'aide d'une interpolation

polynomiale du fait du caractère discontinu de cette fonction. En réalité ce problème correspond plutôt à un problème de classification. À partir de *caractéristiques* : la mesure du capteur $\ell\big(u(\cdot, t^n)\big)$ et l'indice du sous-système précédemment choisi $\boldsymbol{\theta}^{n-1}$, nous voudrions associer à chaque *individu* : t^n, une *classe* qui correspondrait ici à l'indice d'un sous-système.

Afin de choisir un algorithme adapté à notre problème, essayons tout d'abord de comprendre comment fonctionne la règle de contrôle. Un sous-système est choisi à partir de la solution courante et de l'indice du sous-système précédemment sélectionné. Ainsi plusieurs tests simples doivent être effectués, comme par exemple s'assurer que la solution ne franchisse pas un certain seuil. Les réseaux de neurones artificiels semblent donc être les plus adaptés pour ce genre de problème. La figure ci-dessous présente un exemple d'architecture de réseau de neurones afin d'apprendre la règle de contrôle.

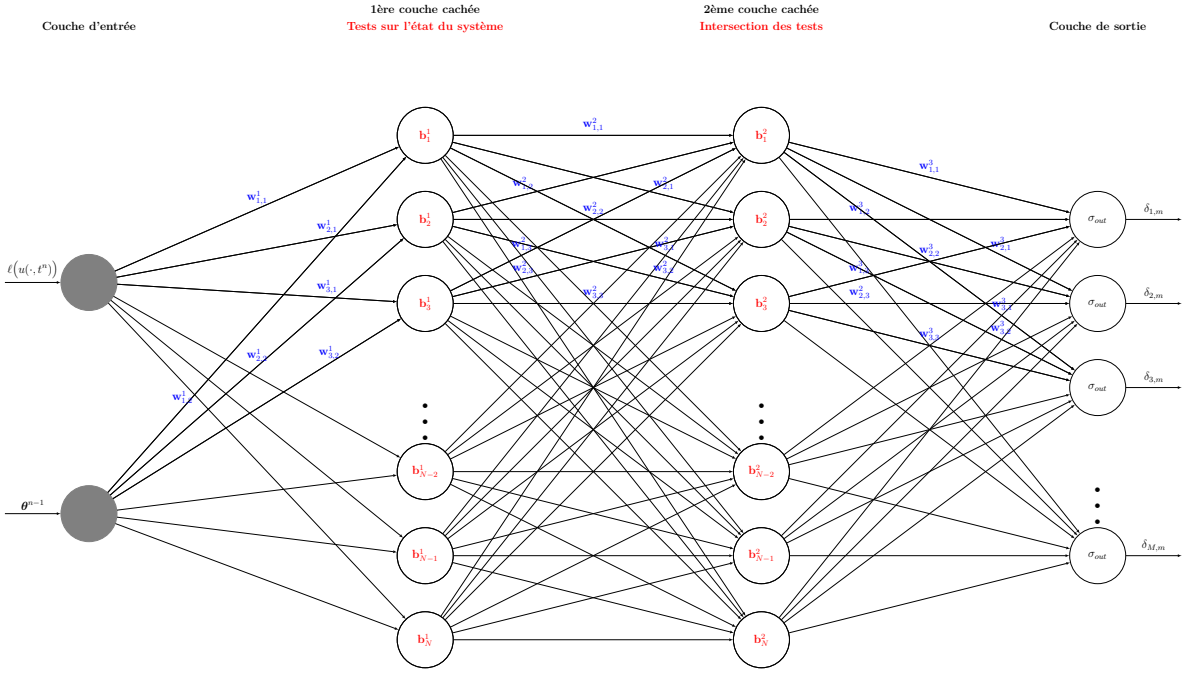


FIGURE 3.2 Architecture de réseau de neurones pour l'apprentissage de la règle de contrôle

La sortie du réseau de neurones est un vecteur de taille M tel que sa composante m vaut 1 et toutes les autres 0, afin de rester conforme aux données contenues dans la matrice $\boldsymbol{\Theta}$.

En entraînant notre réseau de neurones avec les données capteur, on peut ainsi prédire le sous-système devant être choisi. Cependant, pour en déduire un modèle réduit il nous faut connaître la forme linéaire ℓ ou du moins sa représentation dans

l'espace réduit. Ainsi à partir d'un vecteur de taille K contenant les coefficients POD de la solution au temps t^n on voudrait évaluer la mesure du capteur au temps t^n. En utilisant la décomposition POD, on a

$$\ell\Big(u_h(\cdot,t^n)\Big) \approx \ell\Big(\Pi_c^K\, u_h(\cdot,t^n)\Big) = \sum_{k=1}^{K} \mathbf{a}_k^n\, \ell(\phi_k(\cdot)),$$

avec on le rappelle Π_c^K la projection sur l'espace POD continu, et u_h l'approximation de u.
En posant $\tilde{\ell}_k = \ell(\phi_k(\cdot))$, on a alors

$$\ell\Big(u_h(\cdot,t^n)\Big) \approx \langle \tilde{\ell}, \mathbf{a}^n \rangle \quad \text{pour tout } t^n,$$

ou encore

$$\mathbf{A}^T\, \tilde{\ell} \approx \begin{bmatrix} \ell\Big(u_h(\cdot,t^1)\Big) \\ \ell\Big(u_h(\cdot,t^2)\Big) \\ \vdots \\ \ell\Big(u_h(\cdot,t^{N_t})\Big) \end{bmatrix},$$

on a donc un problème surdéterminé, et on trouve finalement par une approche de moindres carrés,

$$\hat{\ell} = \Big(\mathbf{A}\mathbf{A}^T\Big)^{-1}\mathbf{A} \begin{bmatrix} \ell\Big(u_h(\cdot,t^1)\Big) \\ \ell\Big(u_h(\cdot,t^2)\Big) \\ \vdots \\ \ell\Big(u_h(\cdot,t^{N_t})\Big) \end{bmatrix}. \tag{3.11}$$

Ainsi la mesure de la solution se fera en multipliant les coefficients POD prédits par le vecteur $\hat{\ell}$.

Remarque 13. *Le vecteur $\tilde{\ell}$ peut être vu comme le représentant de Riesz de la forme linéaire ℓ dans l'espace POD. En effet, pour toute fonction $\chi \in W^K$, on a*

$$\ell(\chi) = \langle \tilde{\ell}, \mathbf{a}(\chi) \rangle,$$

avec $\mathbf{a}_k(\chi) = \langle \chi, \boldsymbol{\phi}_k \rangle_{\mathbf{M}}$ les coefficients POD de χ.

L'algorithme ci-dessous résume la procédure.

Algorithme 2 : Construction d'un modèle réduit contrôlé par commutation à partir des données

input : La matrice de donnée $\mathbf{U}$, la matrice de corrélation $\mathcal{S}$, l'historique des commutations $\mathbf{\Theta}$ et les données capteur $\left\{\ell\Big(u_h(\cdot,t^n)\Big)\right\}_n$

output : Le modèle réduit

$$\mathbf{a}^{n+1} = \sum_{m=1}^{M} \hat{H}_m\Big(\langle\hat{\ell},\mathbf{a}^n\rangle,\hat{\boldsymbol{\theta}}^{n-1}\Big)\Big(\hat{A}_m\,\mathbf{a}^n+\hat{B}_m\Big), \quad \text{et } \mathbf{u}_r^{n+1}=\mathbf{\Phi}_r\,\mathbf{a}^{n+1}.$$

Réduction de la dimensionalité :

- Décomposer $\mathcal{S}$ en valeurs propres, i.e. $\mathcal{S}\leftarrow\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$
- Tronquer les valeurs propres et vecteurs propres, i.e. $\mathbf{V}_r\leftarrow\mathbf{V}_{\cdot,1:K}$, $\mathbf{\Lambda}_r\leftarrow\mathbf{\Lambda}_{1:K,1:K}$
- Construire les modes POD et coefficients POD, i.e. $\mathbf{\Phi}_r\leftarrow\mathbf{U}\mathbf{V}_r\mathbf{\Lambda}_r^{-\frac{1}{2}}$, $\mathbf{A}\leftarrow\mathbf{\Lambda}_r^{\frac{1}{2}}\mathbf{V}_r^T$

Construction des ensembles d'entraînement :

$$X\leftarrow\Big(\mathbf{\Theta}\otimes X_1\Big)_{:,\pi}, \quad \text{avec} \quad X_1\leftarrow\begin{bmatrix}| & | & & |\\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_t-1}\\ | & | & & |\\ 1 & 1 & \cdots & 1\end{bmatrix}, \quad \text{et } \pi(n)\leftarrow N_t(n-1)+1,$$

$$\text{et} \quad Y\leftarrow\begin{bmatrix}| & | & & |\\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_t}\\ | & | & & |\end{bmatrix}.$$

Identification de la dynamique :

$$\Big[\Big[\hat{A}_1 \quad \hat{B}_1\Big] \quad \Big[\hat{A}_2 \quad \hat{B}_2\Big] \quad \cdots\Big[\hat{A}_M \quad \hat{B}_M\Big]\Big]\leftarrow YX^{\dagger}.$$

Apprentissage de la forme linéaire :

$$\hat{\ell}\leftarrow\Big(\mathbf{A}\mathbf{A}^T\Big)^{-1}\mathbf{A}\begin{bmatrix}\ell\Big(u_h(\cdot,t^1)\Big)\\ \ell\Big(u_h(\cdot,t^2)\Big)\\ \vdots\\ \ell\Big(u_h(\cdot,t^{N_t})\Big)\end{bmatrix}.$$

Apprentissage de la règle de contrôle :

$$\hat{H}\leftarrow\texttt{train}\left(\begin{bmatrix}\ell\Big(u_h(\cdot,t^2)\Big) & \ell\Big(u_h(\cdot,t^3)\Big) & \cdots & \ell\Big(u_h(\cdot,t^{N_t-1})\Big)\\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-2}\end{bmatrix};\mathbf{\Theta}_{\cdot,2:N_t-1}\right)$$

3.4 Apprentissage d'un contrôle sans données capteur

Plaçons-nous à présent dans un cadre *Non-intrusif*. À présent, nous n'avons plus connaissance des données capteur, nous avons seulement connaissance de la solution $\mathbf{u}^n$. Comme habituellement nous utiliserons sa représentation basse dimension. Si dans le paragraphe ci-dessus les réseaux de neurones semblaient être le meilleur choix, dans le cas présent c'est moins évident. En effet, afin d'avoir une représentation fidèle à la solution, on prendra un ordre de troncature K assez grand. Cela aura pour incidence d'augmenter de manière significative le nombre de paramètres à optimiser, ce qui n'est pas souhaitable ici. D'autres algorithmes de classification tels que les $k-$plus proches voisins ou bien les machines à vecteurs de support, semblent être mieux adaptés dans ce cas. On cherchera ainsi à apprendre la fonction qui à $\mathbf{a}^n$ associe $\boldsymbol{\theta}^n$, en entraînant une machine à vecteurs de support avec les matrices $\mathbf{A}$ comme données d'entrée d'entrainement et la matrice $\boldsymbol{\Theta}$ comme données de sortie d'entrainement.

3.5 Partitionnement des temps discrets afin d'identifier les commutations

Considérons finalement le cas *Totalement non-intrusif*. Nous avions vu précédemment que la connaissance de l'historique des commutations, contenu dans la matrice $\boldsymbol{\Theta}$, était fondamentale pour entrainer nos algorithmes de classifications. Il nous faut, maintenant retrouver la matrice $\boldsymbol{\Theta}$ à partir de la matrice des données réduite $\mathbf{A}$.

Le meilleur moyen semble être de regrouper chaque vecteur $\mathbf{a}^n$ en M classes à l'aide d'un algorithme de partitionnement. Cette approche a été utilisée par Lee & al ([110]) avec un algorithme de partionnement spectral ([179]) dans le cas d'un système dynamique de petite dimension. Projetons l'équation (3.3) sur l'espace POD, on a obtient alors la relation :

$$\mathbf{a}^{n+1} \approx \sum_{m=1}^{M} \boldsymbol{\theta}_m^n \Big(\underbrace{A_m \, \mathbf{a}^n + B_m}_{=: f_m(\mathbf{a}^n)} \Big). \tag{3.12}$$

Autrement dit, à chaque pas de temps, $\mathbf{a}^n$ évolue selon une fonction f_m. On a alors en quelque sorte M "espaces de trajectoires", et chaque point $\mathbf{a}^n$ appartient à un de ces espaces. Retrouver les indices des sous-systèmes sélectionnés, revient à regrouper les points $\mathbf{a}^n$ suivant l'espace des trajectoires auquel ils appartiennent. Il faudra ainsi

utiliser un algorithme de partitionnement pouvant reconnaître cette structure, c'est la raison pour laquelle nous utiliserons l'algorithme de partitionnement spectral.

Une fois les temps discrets t^n regroupés en M classes, on peut alors apprendre la matrice $\boldsymbol{\Theta}$ et retrouver ainsi le cas *Non-intrusif*.

3.6 Application à un commutateur thermique

Afin d'évaluer les performances de la méthode introduite dans ce chapitre, nous présenterons dans cette section une application à un commutateur thermique, cette application provient de la thèse d'Adrien Le Coënt ([106], [107]). La première sous-partie sera dédiée à la présentation du problème ainsi qu'aux données générées par le solveur HF. Nous présenterons ensuite l'apprentissage d'un modèle réduit dans le cas où l'historique des commutations est disponible. Enfin le cas où cela n'est pas disponible sera présenté.

3.6.1 Mise en place d'une boîte noire

3.6.1.1 Présentation du problème

Considérons une plaque chauffée par une résistance électrique. Un capteur permettant de mesurer la température est positionné sur la plaque au point $\boldsymbol{x}_S$. Le but est alors de maintenir la température en $\boldsymbol{x}_S$ entre T_{min} et T_{max}. Ainsi les deux sous-systèmes modéliseront la diffusion de la chaleur avec et sans activation de la résistance électrique. La règle de contrôle consistera quant à elle à décider de l'activation ou non de la résistance électrique en fonction de la température au capteur afin de maintenir cette température entre T_{min} et T_{max}. Notons que la plaque sera placée dans un environnement plus froid que T_{min}, de telle sorte que l'activation de la résistance électrique sera rendue nécessaire.
Le problème ci-dessus est schématisé par la figure suivante :

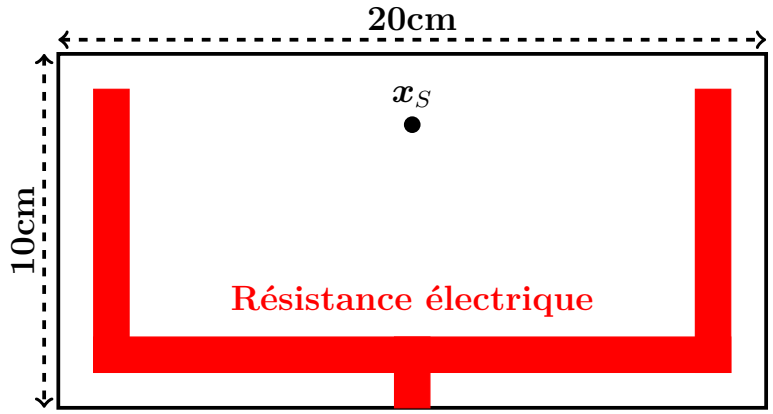


FIGURE 3.3 Schématisation du problème

Mathématiquement, ce problème est modélisé par :

$$\begin{cases} \partial_t u(\boldsymbol{x},t) - \nabla \cdot \Big(\kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x},t)\Big) = f(\boldsymbol{x},t), & \text{dans } \Omega \times (0,T), \\ \kappa \partial_{\mathbf{n}} u(\boldsymbol{x},t) = -\phi_{out}, & \text{sur } \partial\Omega \times (0,T), \\ u(\boldsymbol{x},0) = T_{min}, & \text{dans } \Omega, \end{cases} \tag{3.13}$$

La conductivité thermique κ est constante par morceaux, et vaut κ_R dans le domaine correspondant à la résistance électrique, noté Ω_R, et κ_P en dehors. Le terme source f peut quant à lui s'écrire comme un produit tensoriel espace-temps, et en conservant les notations introduites nous avons :

$$f(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle$$

avec

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{pmatrix} 0 \\ q_R\, \chi_{\Omega_R}(\boldsymbol{x}) \end{pmatrix}, \qquad \text{et} \quad \boldsymbol{\theta}(t) = H\Big(u(\boldsymbol{x}_S,t), \boldsymbol{\theta}(t^-)\Big),$$

et où

$$H\Big(u(\boldsymbol{x}_S,t), \boldsymbol{\theta}(t^-)\Big) = \begin{cases} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, & \text{si } u(\boldsymbol{x}_S,t) \geq T_{max}, \text{ et } \boldsymbol{\theta}(t^-) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, & \text{si } u(\boldsymbol{x}_S,t) \leq T_{min}, \text{ et } \boldsymbol{\theta}(t^-) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \end{cases} \tag{3.14}$$

Afin de générer des données synthétiques qui serviront à constituer un modèle réduit, nous utiliserons un schéma d'Euler implicite pour discrétiser l'opérateur temporel et la méthode des éléments finis pour discrétiser l'espace. Les données générées par le

solveur HF satisferont alors l'équation,

$$\mathbf{u}^{n+1} = \frac{1}{\delta t}\Big(\frac{1}{\delta t}\mathbf{M} + \mathbf{K}\Big)^{-1}\mathbf{M}\,\mathbf{u}^{n} + \left\langle \Big(\frac{1}{\delta t}\mathbf{M} + \mathbf{K}\Big)^{-1}\mathbf{G}, \boldsymbol{\theta}^{n} \right\rangle - \Big(\frac{1}{\delta t}\mathbf{M} + \mathbf{K}\Big)^{-1}\Phi_{out} \quad (3.15)$$

avec $\mathbf{G}$ la discrétisation de $\boldsymbol{g}$ et Φ_{out} la discrétisation du flux sortant ϕ_{out}.
Le tableau ci-dessous présente les paramètres physiques et numériques utilisés.

Paramètres	**Valeurs**
Conductivité thermique dans la résistance (κ_R)	$0.5\ W \cdot m^{-1} \cdot K^{-1}$
Conductivité thermique dans la plaque (κ_P)	$0.2\ W \cdot m^{-1} \cdot K^{-1}$
Température minimale (T_{min})	$20°C$
Température maximale (T_{max})	$21°C$
Flux de la résistance électrique (q_R)	$4W \cdot m^{-2}$
Flux sortant (ϕ_{out})	$0.5\ W \cdot m^{-1}$
Pas de temps (δt)	4
Nombre de sommets du maillage (N_x)	4997
Temps final (T)	$53\,min$

TABLE 3.1 Paramètres du problème

3.6.1.2 Données générées par le solveur HF

L'équation (3.15) sera résolue par le solveur HF à l'aide du logiciel FREEFEM++ ([84]). Les sorties du solveur HF seront :

- **la matrice U :** contenant la température en tout point du maillage et en tout temps discret t^n
- **la matrice de corrélation $\mathcal{S}$**
- **la matrice $\boldsymbol{\Theta}$:** contenant l'historique des commutations
- **le vecteur s :** contentant l'historique des températures au capteur, défini par $\mathbf{s}_n = u(\boldsymbol{x}_S, t^n)$.

La figure ci-dessous représente la température à différents instants ainsi que l'évolution de la température au capteur.

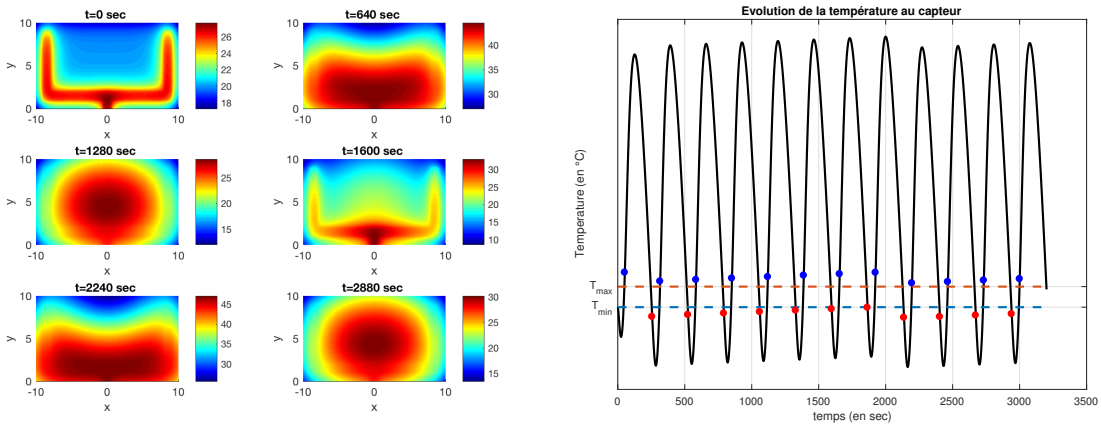


FIGURE 3.4 (Gauche) Température sur la plaque pour différents temps, (droite) Évolution de la température au capteur au cours du temps (ligne noire continue), activation de la résistance électrique (point rouge), désactivation de la résistance électrique (point bleu).

On remarque ainsi que la commutation des états de la résistance électrique rend ce problème fortement dynamique. Par ailleurs, du fait du caractère diffusif de la propagation de la chaleur, la température au capteur continue d'augmenter ou de diminuer au-delà du seuil acceptable.

3.6.2 Apprentissage dans le cadre quasi non-intrusif d'un modèle réduit

Dans toute cette partie nous aurons accès à la matrice $\mathbf{U}$, la matrice $\mathcal{S}$, la matrice $\mathbf{\Theta}$, et au vecteur $\mathbf{s}$. Cependant ni la géométrie du problème, ni la nature des mesures du capteur ne seront connues. Il nous faudra alors identifier la dynamique et les termes sources de chaque sous-système, apprendre la règle de contrôle H, et enfin apprendre le type de mesure effectuée par le capteur.

3.6.2.1 Identification de la dynamique et des termes sources pour chaque sous-système

Commençons par identifier la dynamique et le terme source de chaque sous-système. Afin de valider l'identification de chaque sous-système, nous utiliserons la première moitié des temps discrets pour l'entrainement. Les temps suivants serviront à la validation. On notera alors $N_{t,train} = \left\lfloor \frac{N_t}{2} \right\rfloor$.

Les ensembles d'entraînement seront ceux définis à la **section 3.2**, i.e.

$$X = \Big(\mathbf{\Theta} \otimes X_1\Big)_{:,\pi}, \quad \text{avec} \quad X_1 = \begin{bmatrix} | & | & & | \\ \mathbf{a}^1 & \mathbf{a}^2 & \cdots & \mathbf{a}^{N_{t,train}-1} \\ | & | & & | \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \tag{3.16}$$

$$\text{et} \begin{cases} \pi : \{1, \cdots, N_{t,train} - 1\} \to \{1, \cdots, (N_{t,train} - 1)^2\}, \\ n \mapsto N_{t,train}(n-1) + 1 \end{cases} \tag{3.17}$$

et

$$Y = \begin{bmatrix} | & | & & | \\ \mathbf{a}^2 & \mathbf{a}^3 & \cdots & \mathbf{a}^{N_{t,train}} \\ | & | & & | \end{bmatrix},$$

ainsi les matrices et vecteurs $\hat{A}_m$ et $\hat{B}_m$, seront donnés par :

$$\Big[\Big[\hat{A}_1 \quad \hat{B}_1\Big] \quad \Big[\hat{A}_2 \quad \hat{B}_2\Big]\Big] = YX^{\dagger}.$$

Nous allons à présent analyser les matrices $\hat{A}_1$ et $\hat{A}_2$ afin de s'assurer que nous avons appris la "bonne physique", puis s'assurer que les vecteurs $\hat{B}_1$ et $\hat{B}_2$ correspondent respectivement au cas où la résistance électrique est désactivée et au cas où elle est activée. Finalement, nous quantifierons la qualité de cette identification en utilisant la règle exacte H utilisée par le solveur HF.

Comme nous l'avions vu au **Chapitre 2**, les valeurs propres d'un modèle réduit correspondant à de la diffusion doivent être à l'intérieur du cercle unité et décroître vers 0. Afin de s'en assurer, présentons le spectre de ces deux matrices dans le plan complexe, pour $K = 30$.

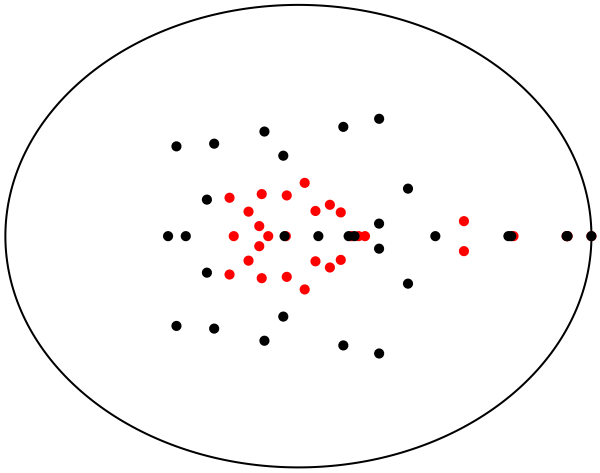


FIGURE 3.5 Spectre de $\hat{A}_1$ (en rouge) et $\hat{A}_2$ (en noir) dans le plan complexe, les valeurs propres sont donc bien toutes inscrites dans le cercle unité (ligne continue noire)

On voit donc que les valeurs propres respectent bien cette propriété. Par ailleurs, certaines valeurs propres ont une partie imaginaire non nulle, ce qui dénote un manque de symétrie des matrices $\hat{A}_1$ et $\hat{A}_2$.

À présent, étudions les vecteurs $\hat{B}_1$ et $\hat{B}_2$. Ainsi la figure ci-dessous représente les termes sources appris $\hat{\boldsymbol{g}}_1$ et $\hat{\boldsymbol{g}}_2$, reconstruits à l'aide des modes POD, i.e.

$$\hat{\boldsymbol{g}}_m := \sum_{k=1}^{K} \left(\hat{B}_m\right)_k \phi_k(\boldsymbol{x}).$$

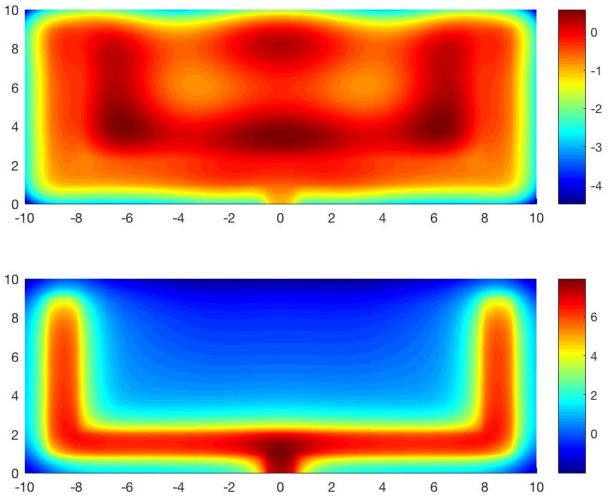


FIGURE 3.6 Reconstruction de $\hat{B}_1$ qui correspond au flux sortant ϕ_{out} (en haut) et $\hat{B}_2$ correspondant au flux sortant ϕ_{out} et à l'activation de la résistance électrique (en bas)

La fonction $\hat{\boldsymbol{g}}_1$ représente le cas où la résistance électrique est désactivée, seul le flux ϕ_{out} refroidi la plaque. C'est ce qu'on retrouve sur la figure du haut où la fonction $\hat{\boldsymbol{g}}_1$ est négative sur le bord du domaine, et nulle ailleurs. En ce qui concerne $\hat{\boldsymbol{g}}_2$, cette fois-ci la résistance électrique est activée et chauffe la plaque. Ainsi la fonction $\hat{\boldsymbol{g}}_2$ a bien appris l'action de la résistance électrique sur la plaque.

Ainsi il semble que l'on ait correctement identifié la dynamique et les termes sources de chaque sous-système. Quantifions à présent cela, à l'aide des données de validation. L'erreur sera mesurée en norme L^2.

Nombre de modes POD (K)	**Erreur sur l'ensemble d'entraînement**	**Erreur sur l'ensemble de validation**	**Erreur POD**
1	0.3394	0.3406	0.3395
5	0.0020	0.0014	0.0020
10	1.975×10^{-4}	1.530×10^{-4}	1.982×10^{-4}
20	1.893×10^{-4}	1.491×10^{-4}	1.899×10^{-4}
30	1.815×10^{-4}	1.436×10^{-4}	1.821×10^{-4}
60	1.653×10^{-4}	1.432×10^{-4}	1.659×10^{-4}

TABLE 3.2 Erreurs relatives de prédiction et de réduction

On a donc d'excellents résultats avec seulement 10 modes POD. Par ailleurs, l'erreur de prédiction semble bien dépendre de l'erreur de réduction. Enfin l'erreur commise sur l'ensemble de validation est même inférieure à celle commise sur l'ensemble d'entraînement. On peut ainsi espérer une très bonne prédiction en temps long. À présent présentons l'apprentissage de la règle de contrôle.

3.6.2.2 Apprentissage de la règle de contrôle

Dans cette section, nous cherchons à apprendre la règle de contrôle H. Pour ce faire, nous utiliserons un réseau de neurones avec deux couches cachées contenant chacune 15 neurones.

Remarque 14. *Notons que la règle de contrôle H peut s'écrire comme un réseau de neurones avec deux couches cachées contenant respectivement* 3 *et* 2 *neurones.*

Afin de confirmer le choix de l'algorithme utilisé pour l'apprentissage de la règle de contrôle H, nous le comparerons à différents algorithmes de classification très répandus en apprentissage automatique. Nous effectuerons une validation croisée sur les données d'entraînement générées par le solveur HF.

Le tableau ci-dessous représente les résultats de la validation croisée, où on a séparé 5 fois notre ensemble d'entrainement. L'implémentation utilisée de ces algorithmes de classification et de la validation croisée, est celle de la bibliothèque Scikit-Learn ([135]), implémentée en Python.

Algorithme	**Score**	**Intervalle de confiance**
Réseau de neurones	100 %	0 %
k plus proches voisins	99.75 %	1.01 %
SVC	100 %	0 %
Forêts aléatoires	100 %	0 %
Processus Gaussien	99.75 %	0.99 %
Classifier naïve de Bayes Gaussien	96.99 %	1.22%
AdaBoost	99.75 %	0.99 %
Arbre de décision	98.99 %	2.95 %
Analyse discriminante quadratique	96.99 %	1.22%

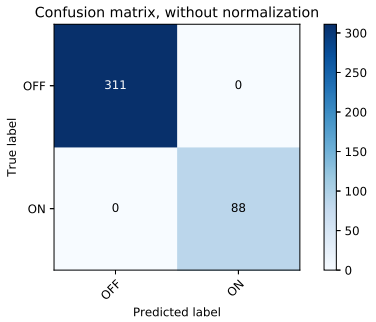
Table 3.3 Score pour l'apprentissage de la règle de contrôle

Le réseau de neurones présente à la fois le meilleur score mais aussi l'intervalle de confiance le plus faible, cela motive donc l'utilisation d'un réseau de neurones afin d'apprendre la règle de contrôle. Notons par ailleurs, que les machines à vecteurs de supports ainsi que les forêts aléatoires présentent des performances similaires.

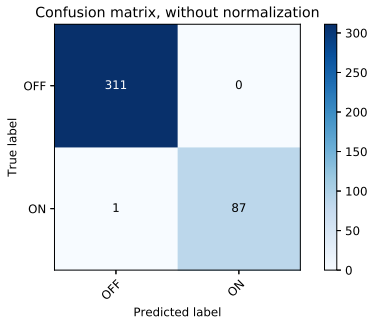
Affichons, à présent, la matrice de confusion pour chaque algorithme. Les lignes de cette matrice correspondent aux classes réelles, tandis que les colonnes aux classes

prédites. Ainsi l'algorithme aura correctement classé les entrées si cette matrice est diagonale.

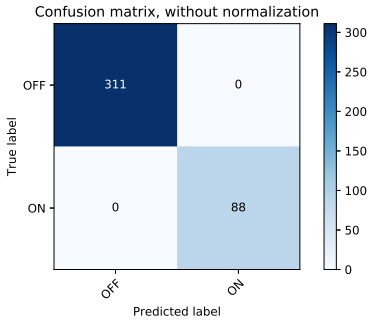
La figure ci-dessous représente la matrice de confusion pour chaque algorithme.



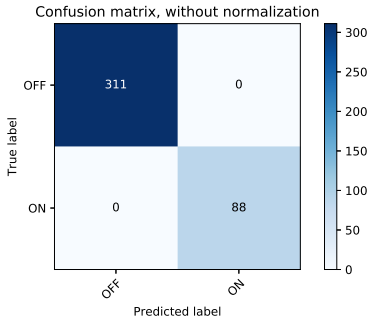
(a) Réseau de neurones



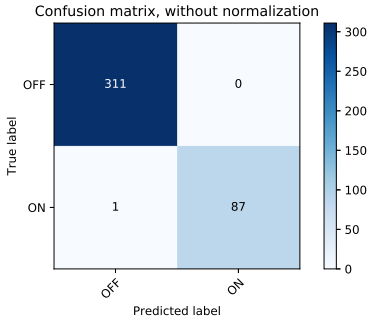
(b) k plus proches voisins



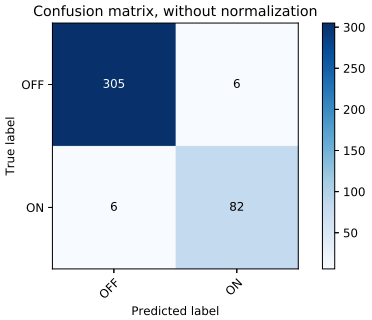
(c) SVC



(d) Forêts aléatoires



(e) Processus Gaussien



(f) Classifier naïve de Bayes Gaussien

FIGURE 3.7 Matrice de confusion des différents algorithmes.

3.6.2.3 Apprentissage des mesures du capteur

La dernière étape nécessaire à l'apprentissage d'un modèle réduit, est l'apprentissage du type de mesure réalisé par le capteur. Nous souhaitons donc prédire la quantité $\ell(\mathbf{u}^n)$ à partir des coefficients POD $\mathbf{a}^n$. Comme présenté à la **section 3.3** il suffit pour

cela de retrouver à partir des données, la valeur de chaque mode POD au capteur. On notera alors $\hat{\ell}$ le vecteur contenant ces valeurs, et on l'obtiendra par

$$\hat{\ell} = \left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_{N_{t,train}} \end{bmatrix}.$$

et ainsi

$$\ell(\mathbf{u}^n) \approx \langle \hat{\ell}, \mathbf{a}^n \rangle.$$

.

3.6.2.4 Validation du modèle réduit appris

Nous avons maintenant, tous les outils pour prédire le comportement de la température sur la plaque. En utilisant **l'algorithme 2**, nous pouvons prédire l'évolution de la température sur la plaque. La figure ci-dessous présente l'évolution de la température au capteur ainsi que l'évolution des activations de la résistance électrique générées par le solveur haute-fidélité (en noir) et le modèle réduit (en rouge), sur tous les temps.

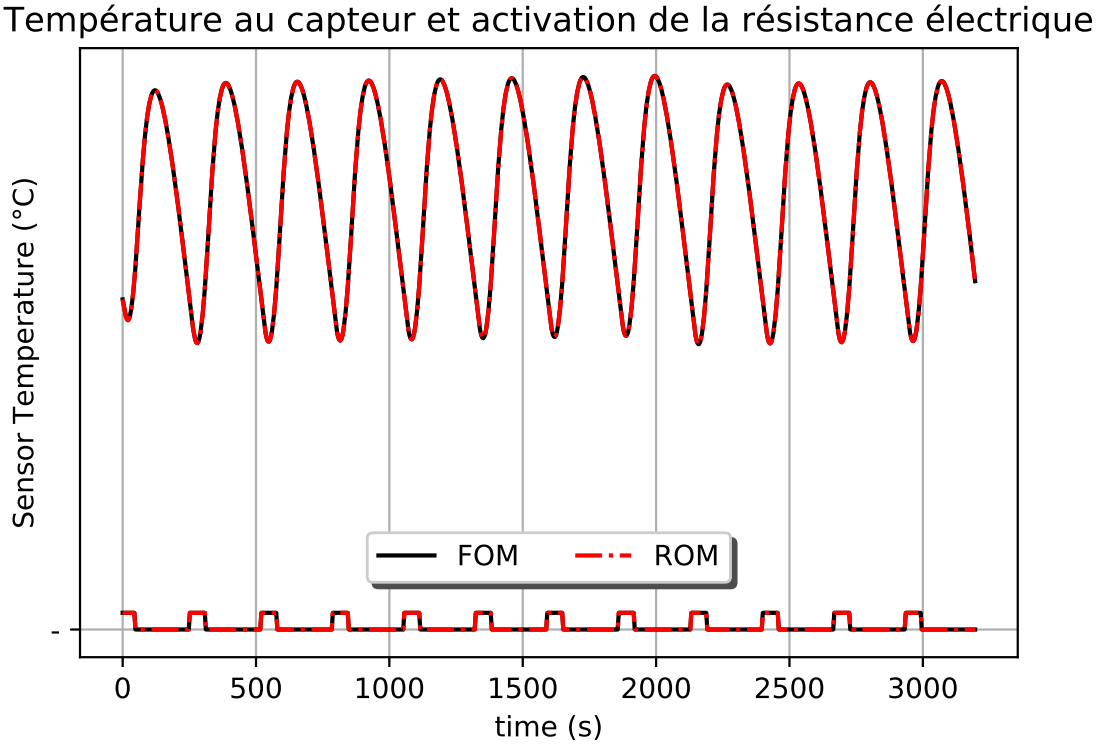


FIGURE 3.8 Comparaison entre les données HF et le modèle réduit appris

On a donc parfaitement appris le modèle (3.13). On est ainsi capable de rejouer le modèle et donc de prédire la température.

3.6.3 Apprentissage dans le cadre non-intrusif d'un modèle réduit

Dans cette section nous supposerons avoir accès aux matrices $\mathbf{U}$, $\mathcal{S}$, et $\mathbf{\Theta}$. Si l'identification des dynamiques et termes sources est analogue au cadre Quasi non-intrusif, la règle de contrôle apprise dépendra ici de la température sur toute la plaque. Rappelons que la température sur la plaque sera connue au travers des coefficients POD $\mathbf{a}^n$.

3.6.3.1 Apprentissage de la règle de contrôle

Pour apprendre la règle de contrôle, nous utiliserons une machine à vecteurs de support à noyau polynomial. Par ailleurs, nous la comparerons comme précédemment avec d'autres algorithmes de classification. Essayons tout d'abord de prédire l'état de

la résistance électrique en n'utilisant que les deux premiers coefficients POD. Cela permettra ainsi de visualiser les frontières de décisions. L'ensemble de visualisation sera la grille

$$\mathcal{G} := \left[\left(\min_{1\leq n\leq N_{t,train}} \mathbf{a}_1^n - \alpha_1\,,\ \max_{1\leq n\leq N_{t,train}} \mathbf{a}_1^n + \alpha_1\right) \times \left(\min_{1\leq n\leq N_{t,train}} \mathbf{a}_2^n - \alpha_2\,,\ \max_{1\leq n\leq N_{t,train}} \mathbf{a}_2^n + \alpha_2\right)\right].$$

Afin de comprendre la signification des deux premiers coefficients POD, représentons les deux premiers modes POD continus $\phi_1(\boldsymbol{x})$ et $\phi_2(\boldsymbol{x})$.

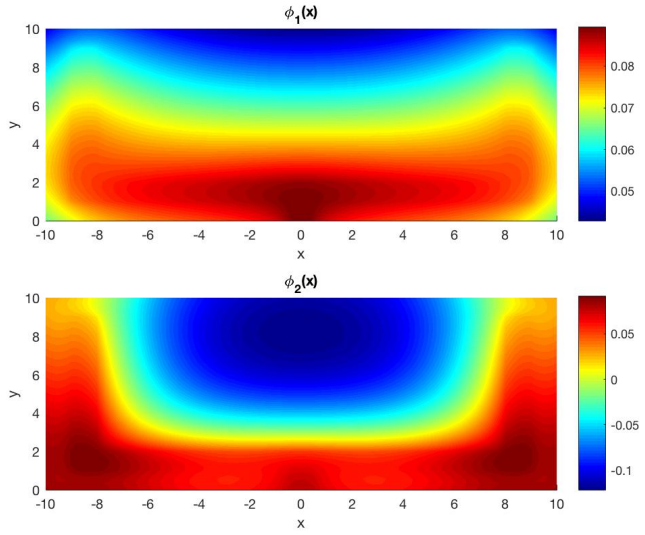


FIGURE 3.9 Les deux premiers modes POD continues ϕ_1 et ϕ_2

Il semble que le premier mode POD représente le cas où la résistance électrique n'est pas activée, tandis que le deuxième mode POD représente le cas où elle est activée. Ainsi, il semble que le test à effectuer est de vérifier que le premier coefficient POD est supérieur au deuxième. Bien-sûr, la difficulté réside à identifier exactement la séparation. Par ailleurs, cette séparation semble linéaire puisqu'on voudrait la définir par une droite du type $a_2 = \alpha a_1 + \beta$. Les résultats sont présentés ci-dessous, où la zone bleu correspond à la désactivation de la résistance électrique et la zone rouge à l'activation. Les points visibles sont les données d'entraînement.

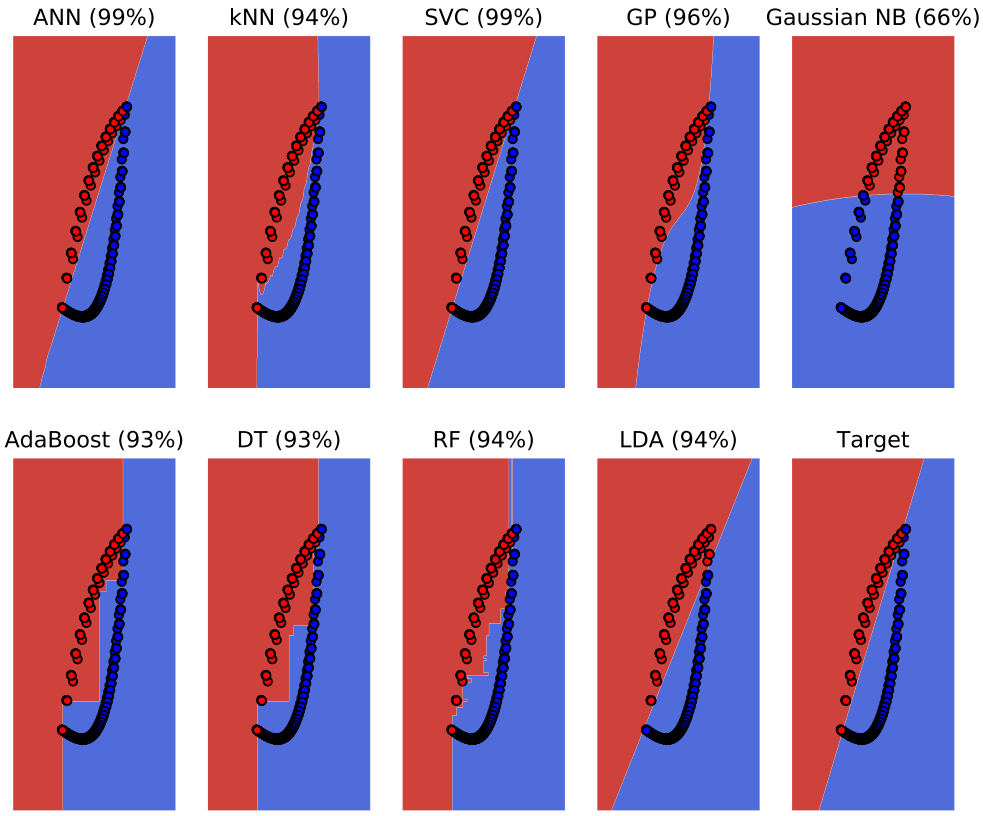


FIGURE 3.10 Frontière de décision obtenue par les différents algorithmes de classification utilisés pour la grille $\mathcal{G}$. L'abscisse correspond au premier coefficient POD, tandis que l'ordonnée correspond au second coefficient POD. La figure tout en bas à droite correspond à la frontière de décision obtenue avec la règle θ utilisée par le solveur HF.

Comme attendu la séparation semble linéaire. Notons qu'en considérant le comportement global du système, l'algorithme a aussi l'information sur l'état précédent de la résistance électrique. Cette fois-ci il semble que le réseau de neurones et les machines à vecteurs de support donnent les meilleurs résultats. Lorsque K augmente, la dimension du problème d'apprentissage augmente. La figure ci-dessous présente les scores pour $K = 20$.

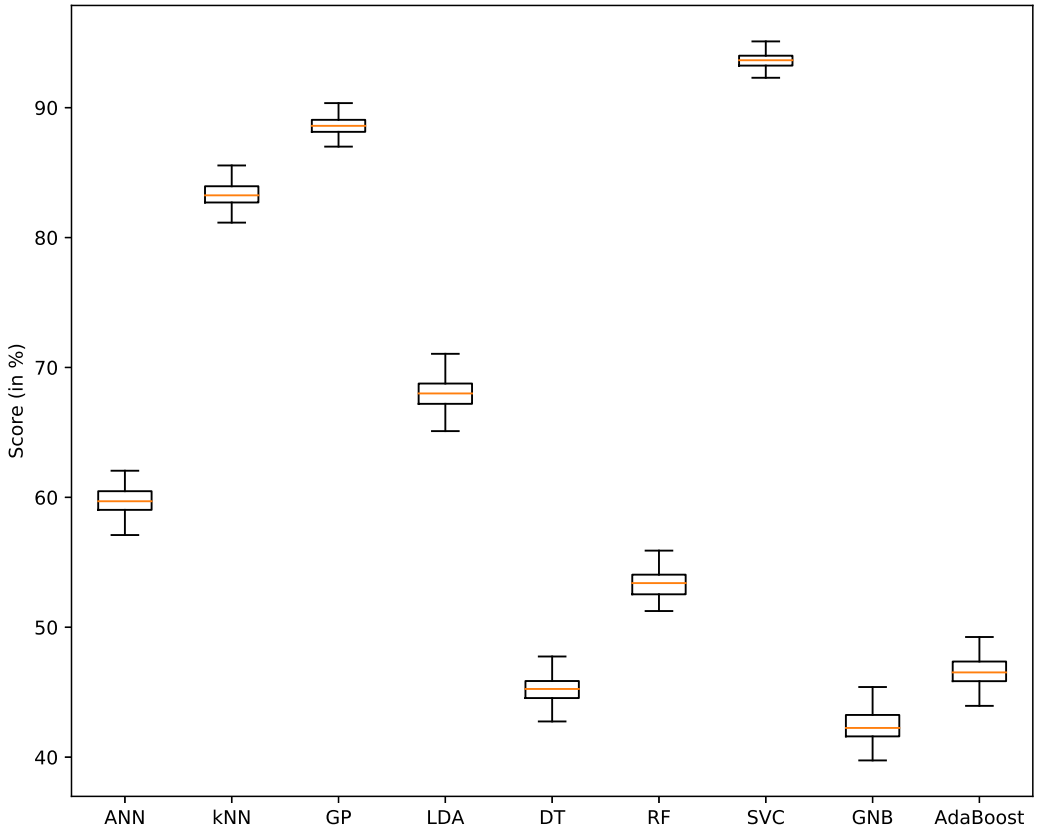


FIGURE 3.11 Score pour $K = 20$

L'augmentation de la dimension a fortement impactée les résultats pour la plus part des algorithmes. Seuls les algorithmes des $k-$plus proches voisins, des machines à vecteurs de support et les Processus Gaussiens semblent conserver un bon score.

3.6.3.2 Validation du modèle réduit

Nous allons à présent quantifier la précision du modèle réduit. Dans le tableau ci-dessous nous comparons l'erreur de prédiction, pour différents rangs de troncature K, en utilisant la règle de contrôle apprise $\hat{H}$ et celle utilisée par le solveur haute-fidélité H.

Nombre de modes POD (K)	Erreur sur l'ensemble d'entraînement avec $\hat{H}$	Erreur sur l'ensemble de validation avec $\hat{H}$	Erreur sur l'ensemble d'entraînement avec H	Erreur sur l'ensemble de validation avec H
2	0.2071	0.4555	0.1185	0.1406
5	0.0021	0.0015	0.0020	0.0014
10	2.018×10^{-4}	1.531×10^{-4}	1.975×10^{-4}	1.530×10^{-4}
20	1.912×10^{-4}	1.5×10^{-4}	1.893×10^{-4}	1.491×10^{-4}
30	1.815×10^{-4}	1.436×10^{-4}	1.815×10^{-4}	1.436×10^{-4}

TABLE 3.4 Erreurs relatives de prédiction avec règles de contrôle apprise et exacte

Lorsque nous considérons seulement deux coefficients POD, la prédiction du modèle réduit en utilisant la règle apprise est moins bonne que la prédiction en utilisant la règle exacte. En revanche, lorsque K augmente l'erreur commise en utilisant la règle apprise tend vers l'erreur commise avec règle exacte, ce qui signifie que les SVC à noyau polynomial sont bien capables d'apprendre la règle de contrôle, on a même la même erreur pour $K = 30$. On est ainsi en mesure d'apprendre le contrôle sans avoir accès aux données capteur.

Remarque 15. *On pourrait apprendre le contrôle avec un réseau de neurones ayant l'architecture suivante :*

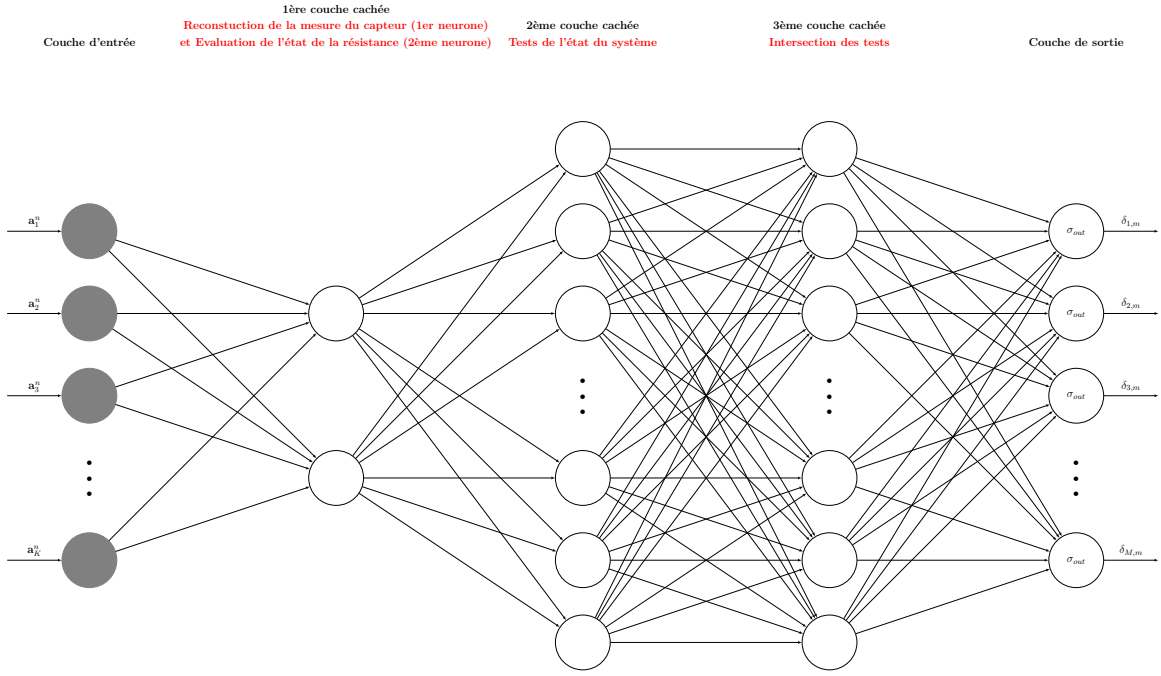


FIGURE 3.12 Architecture du réseau de neurones pour l'apprentissage de la règle de contrôle sans les mesures du capteur

La première couche interne servirait alors à reconstruire à la fois la mesure du capteur et l'état précédent de la résistance électrique. Ainsi pour le premier neurone

de cette couche, on aurait une fonction d'activation linéaire (aussi appelée identité) si bien que la sortie de ce neurone serait

$$\Big(\sum_{k=1}^{K} \mathbf{w}_{1,k}^1 \mathbf{a}_k^n + \mathbf{b}_1^1\Big),$$

et si l'optimisation des paramètres du réseau de neurones nous donne $\mathbf{b}_1^1 = 0$ *et* $\mathbf{w}_{1,k}^1 = \ell_k$*, on aura comme sortie*

$$\langle \ell, \mathbf{a}^n \rangle,$$

ce qui correspond à l'approximation de la mesure du capteur. On est donc capable de reconstruire la mesure du capteur avec cette architecture. En ce qui concerne le deuxième neurone de la première couche interne, un test du type $\sum_{1\leq k\leq K} \alpha_k \, \mathbf{a}_k^n \leq \beta$ *pourrait nous donner l'état de la résistance électrique compte tenu de la forme des modes POD, ce qui reviendrait à considérer une fonction d'activation de type* `sigmoid`*.*

Dans notre cas, le peu de données d'entrainement sont un frein à l'utilisation des réseaux de neurones, c'est pourquoi nous avons préféré utiliser des machines à vecteurs de support.

3.6.4 Apprentissage dans le cadre totalement non-intrusif d'un modèle réduit

Considérons enfin le cas où nous ne connaissons que la matrice des données $\mathbf{U}$ et la matrice de corrélation $\mathcal{S}$. Il nous faut alors identifier l'historique des activations de la résistance électrique afin de retrouver un cadre non-intrusif.

3.6.4.1 Identification des commutations

Nous souhaitons ainsi partitionner les $N_{t,train}$ temps discrets en deux groupes : activation ou désactivation de la résistance électrique, en fonction de leurs coefficients POD. Afin de visualiser les données que nous devront regrouper, représentons les deux (respectivement trois) premiers coefficients POD dans le plan (respectivement l'espace) avec les codes couleurs introduits précédemment (i.e. bleu pour une résistance électrique désactivée, rouge pour une résistance activée). Pour faciliter le partitionnement, on normalise les coefficients POD.

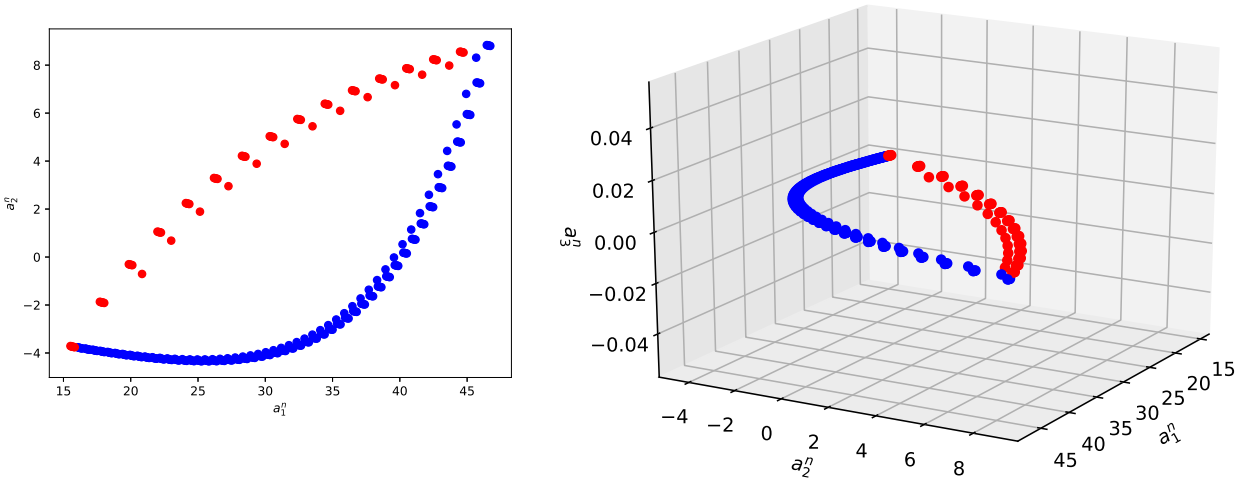


FIGURE 3.13 Données partitionnées en dimension 2 et 3

Il semble bien que les données correspondant à l'activation de la résistance électrique vivent sur une trajectoire différente de celle correspondant à la désactivation de la résistance électrique. On observe aussi que ces deux trajectoires s'intersectent, ce qui rend le partitionnement difficile. Il semblerait que les points correspondant à cette intersection soient en fait les temps au cours duquel une commutation s'opère. Cette intuition est appuyée par la figure ci-dessous.

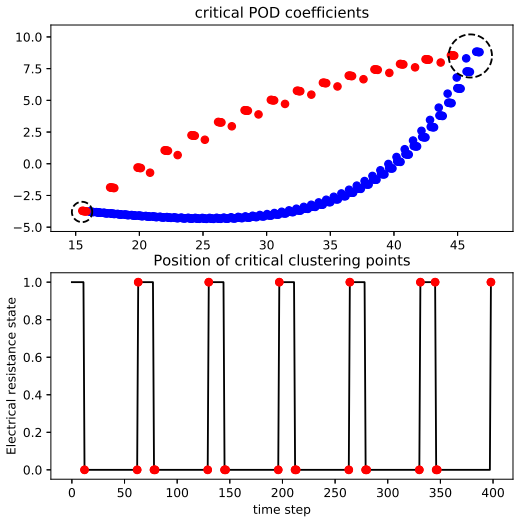


FIGURE 3.14 (En haut) Données groupées et position critique (en bas) correspondance des points critiques par rapport aux états de la résistance électrique

Les cercles en pointillés indiquent les zones où le partitionnement est difficile à réaliser. Le graphique du bas représente l'historique des états de la résistance électrique, ainsi que la position des points (en rouge) des données contenues dans les cercles.

En fait, il semble que le changement de dynamique lors d'une commutation n'opère pas un changement brutal du comportement de la température sur la plaque, ce qui entraîne des coefficients POD localement proche. Cela est dû au caractère régularisant de la diffusion.

Cherchons à présent à partitionner les données, en utilisant le partitionnement spectral. De plus, nous le comparerons avec différents algorithmes de partitionnement. La figure ci-dessous présente le partitionnement pour $K = 3$ coefficients POD. Là encore les algorithmes de partitionnement sont ceux issus de la bibliothèque Scikit-Learn.

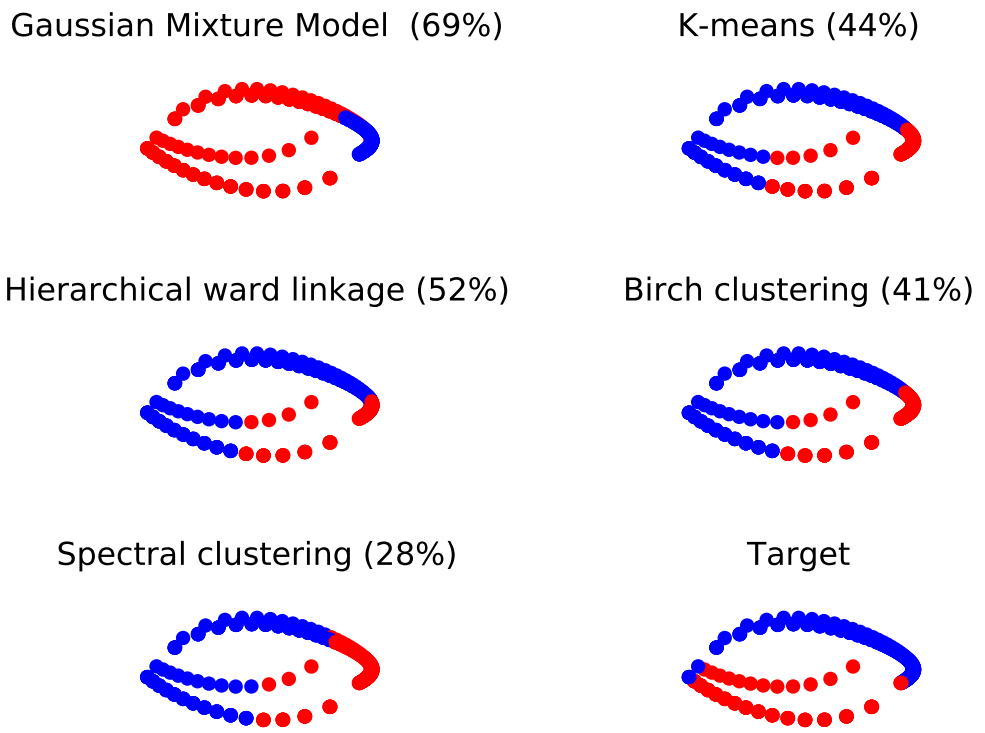


FIGURE 3.15 Partitionnement des données pour $K = 3$

Aucun algorithme de partitionnement ne regroupe correctement les données.

Afin de contourner ce problème une des solutions serait de nettoyer les données, en ne conservant que les points à une certaine distance des intersections. Les résultats sont présentés à la figure ci-dessous. On utilisera l'algorithme de partitionnement spectral.

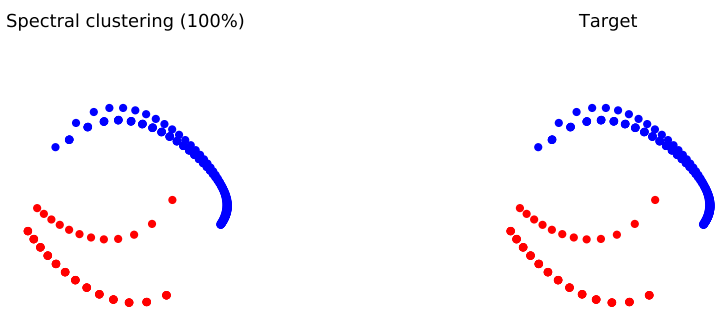


FIGURE 3.16 Regroupement des données nettoyées par l'algorithme SC

On a ainsi correctement partitionné les temps discrets. Ce qui nous permet alors de créer un ensemble d'entrainement, que l'on notera $\hat{\mathbf{\Theta}}$, afin d'identifier la dynamique et le terme source de chaque sous-système ainsi qu'à apprendre la règle de contrôle.

3.6.4.2 Identification de la dynamique en utilisant la règle de commutation apprise

La procédure d'identification reste la même qu'auparavant, notons que cette fois-ci nous disposons de moins de temps discrets. Cependant, comme on peut le voir sur la figure 3.16, le nombre de temps disponible est tout à fait raisonnable. Notons $\mathbf{i}$ le vecteur contenant les indices des temps discrets conservés après nettoyage des données, et $N_{t,clean}$ le nombre de temps conservés. En définissant X et Y comme

$$X = \left(\hat{\mathbf{\Theta}} \otimes X_1\right)_{:,\pi}, \quad \text{avec} \quad X_1 = \begin{bmatrix} | & | & & | \\ \mathbf{a}^{\mathbf{i}_1} & \mathbf{a}^{\mathbf{i}_2} & \cdots & \mathbf{a}^{\mathbf{i}_{N_{t,clean}-1}} \\ | & | & & | \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \tag{3.18}$$

$$\text{et} \begin{cases} \pi : \{1, \cdots, N_{t,clean} - 1\} \to \{1, \cdots, (N_{t,clean} - 1)^2\}, \\ n \mapsto N_{t,clean}(n-1) + 1 \end{cases} \tag{3.19}$$

et

$$Y = \begin{bmatrix} | & | & & | \\ \mathbf{a}^{\mathbf{i}_1+1} & \mathbf{a}^{\mathbf{i}_2+1} & \cdots & \mathbf{a}^{\mathbf{i}_{N_{t,clean}-1}+1} \\ | & | & & | \end{bmatrix},$$

on retrouve alors la dynamique avec et sans activation de la résistance électrique par :

$$\left[\begin{bmatrix}\hat{A}_1 & \hat{B}_1\end{bmatrix} \quad \begin{bmatrix}\hat{A}_2 & \hat{B}_2\end{bmatrix}\right] = YX^{\dagger}.$$

Afin de vérifier que nous avons bien appris la dynamique de nos sous-systèmes, nous allons comparer les erreurs de prédiction en utilisant l'historique des commutations exactes (i.e. la matrice $\boldsymbol{\Theta}$), et les commutations apprises i.e. la matrice $\hat{\boldsymbol{\Theta}}$). Le tableau ci-dessous présente les résultats

Nombre de modes POD (K)	**Erreur sur l'ensemble d'entraînement avec** $\hat{\Theta}$	**Erreur sur l'ensemble de validation avec** $\hat{\Theta}$	**Erreur sur l'ensemble d'entraînement avec** Θ	**Erreur sur l'ensemble de validation avec** Θ
1	0.3395	0.3405	0.3394	0.3406
5	0.0023	0.0021	0.0020	0.0014
10	8.716×10^{-4}	8.791×10^{-4}	1.975×10^{-4}	1.530×10^{-4}
20	9.126×10^{-4}	9.118×10^{-4}	1.893×10^{-4}	1.491×10^{-4}
30	0.0010	0.0010	1.815×10^{-4}	1.436×10^{-4}
60	0.0024	0.0026	1.653×10^{-4}	1.432×10^{-4}

TABLE 3.5 Erreurs relatives de prédiction

Bien que les résultats soient moins bons que dans le cas où nous avions connaissance de l'historique de l'activation de la résistance électrique, nous sommes en mesure d'identifier la dynamique avec une précision tout à fait satisfaisante.

3.6.4.3 Apprentissage de la règle de contrôle à partir des commutations apprises

À présent vérifions qu'avec le nouvel ensemble d'entrainement, nous sommes en mesure d'apprendre la règle de contrôle. La figure ci-dessous présente le résultat pour 2 coefficients POD, comme précédemment.

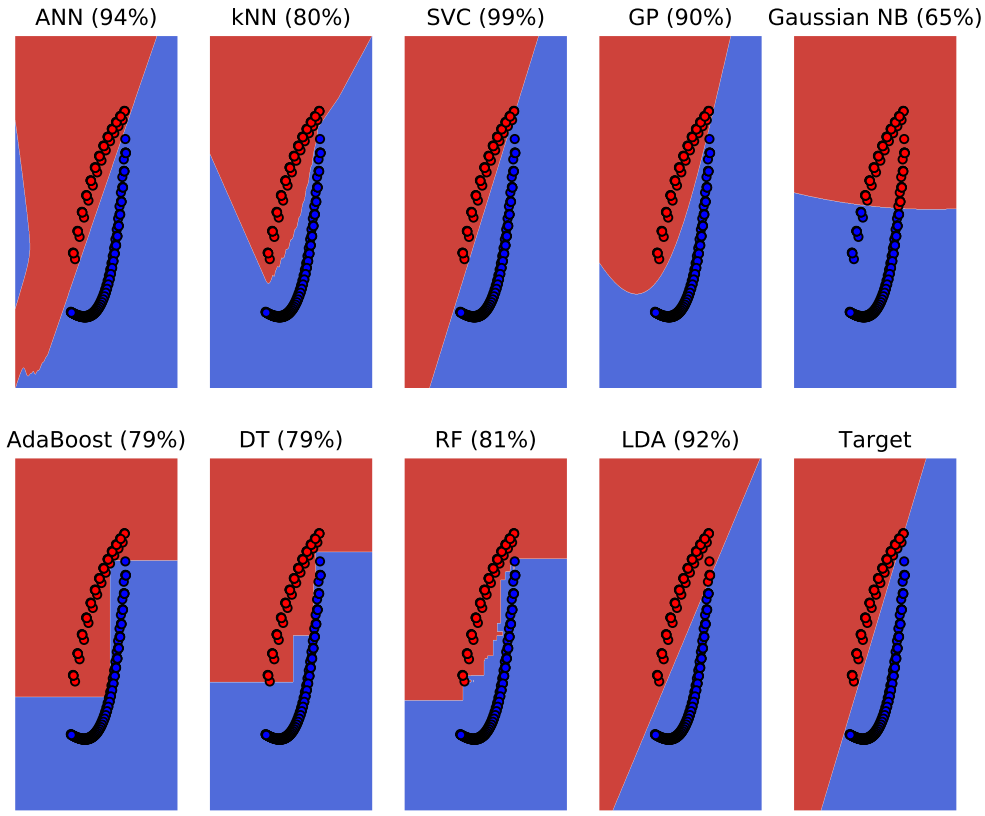


FIGURE 3.17 Validation des classifications pour $K = 2$

La plus part des algorithmes ne parviennent pas à apprendre la règle correctement. Les machines à vecteurs de support donnent cependant un résultat convenable.

3.6.4.4 Validation du modèle réduit

Nous avons à présent tout pour construire un modèle réduit. Malheureusement, si la dynamique semble bien avoir été identifiée, en nettoyant les données on a aussi retiré des points cruciaux pour le bon apprentissage de la règle de contrôle. Ainsi la règle de contrôle apprise sépare mal les données, comme le montre la figure ci-dessous, où la couleur noire correspond à une mauvaise prédiction et blanche à une bonne prédiction.

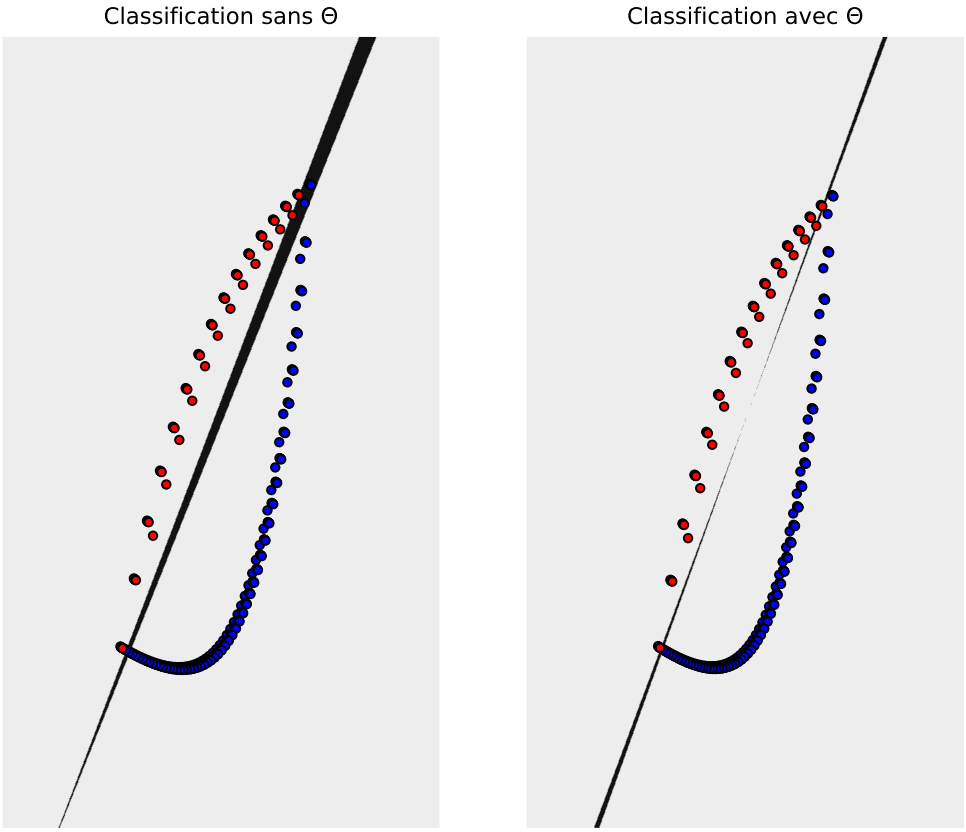


FIGURE 3.18 Erreur de classification, la zone blanche correspond à celle où le classifier a bien prédit l'état de la résistance électrique, tandis que la zone noire signifie que le classifier a mal prédit, les données d'entrainement avant nettoyage sont représentés en bleu (état OFF), et en rouge (état ON). La figure de gauche présente l'erreur commise en utilisant les données prédites par le regroupement, tandis que la figure de droite présente l'erreur en utilisant l'historique des activations générées par le solveur HF.

L'erreur commise pour la frontière de décision est donc plus importante avec $\hat{\boldsymbol{\Theta}}$. Cette erreur entraîne alors un retard d'activation de la résistance électrique. Si bien que la température prédite par le modèle réduit ne correspond plus à celle générée par le solveur HF. Pour appuyer cela affichons la température au capteur générée par le solveur HF et celle prédite par le modèle réduit. On affichera aussi l'historique des activations pour le solveur HF et le modèle réduit.

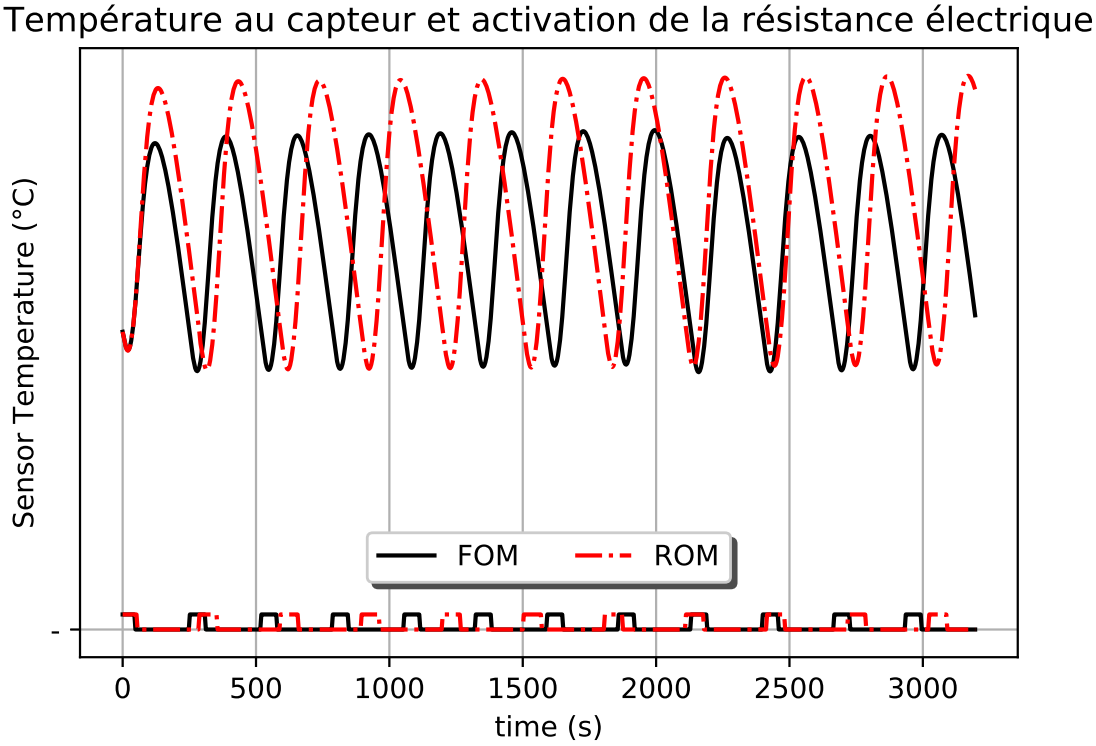


FIGURE 3.19 Évolution de la température au capteur prédite par le modèle réduit (ligne rouge pointillée) et données HF (ligne noire continue).

Nous n'avons donc pas pu construire un modèle réduit satisfaisant dans le cas totalement non-intrusif. Notons tout de même que la dynamique associée à chaque sous-système a été apprise avec une précision plutôt satisfaisante, comme le montre le tableau 3.5.

Conclusion

Dans ce chapitre, nous avons présenté une extension de la méthode d'identification, introduite au **Chapitre 2**, au cas de systèmes contrôlés par commutation. Trois niveaux de non-intrusion ont été considérés afin de généraliser au mieux la méthode ainsi que d'afficher ses limites. Pour chaque cas, une méthodologie a été proposée faisant intervenir les méthodes POD et DMD ainsi que différents algorithmes issue du *Machine Learning* tel que les réseau de neurones (ANN), les machines à vecteurs de supports (SVC) ou bien encore le partitionnement spectral (SC). Cela permet alors de construire un modèle réduit fidèle aux données. Cette approche a été appliquée au cas

d'un commutateur thermique. Les résultats numériques ont permis de conforter le choix des algorithmes proposés. Le dernier cas totalement non-intrusif s'est, en revanche, révélé être trop restrictif pour apprendre un modèle réduit fidèle au solveur HF, bien que la dynamique associée à chaque sous-système ait été correctement identifiée. Une application à la méthodologie proposée dans ce chapitre, est l'optimisation de la règle de contrôle. Cette application est présentée en **Annexe B**. Enfin, un inconvénient au modèle réduit appris, est qu'il requiert à chaque pas de temps, la reconstruction de la solution, afin d'ajuster le contrôle. Cela peut être évité, comme nous le verrons au chapitre suivant.

Chapitre 4

Réduction par échantillonnage des données et identification d'une EDP d'évolution linéaire

Résumé Ce chapitre est consacré à la réduction et l'identification d'une EDP d'évolution linéaire par échantillonnage des données. L'échantillonnage devra prendre en compte la structure des données générées par le solveur HF. Nous proposerons ainsi l'utilisation de la *méthode d'interpolation empirique* (EIM). Cette approche sera comparée numériquement sur un problème de diffusion avec des conditions aux limites dynamiques, ainsi que sur un problème d'onde avec également des conditions aux limites dynamiques.

Mots-clés. Data-driven model ; System identification ; EIM ; Reduced Order Model ; DMD ; Data analysis ; Column Sub-set Selection Problem ; Sampling methods.

Introduction

Nous avons présenté au **Chapitre 2** une méthode permettant l'identification d'une EDP d'évolution linéaire. Afin de parer au caractère mal-posé du problème d'identification, nous avions dû réduire la dimension. La méthode POD s'est alors avérée bien adaptée pour ce genre de problème. Cependant dans certains cas, cette approche de réduction présente certains inconvénients.

Le premier inconvénient est la nécessité d'une connaissance fine de la solution u à chaque instant. Cela permet ainsi de construire les modes POD sur lesquels sera

décomposée la solution en tout temps. Cette approche est donc restreinte à deux types de données : les données synthétiques et les données expérimentales abondantes.

Par ailleurs, dans de nombreuses applications, la prédiction de la solution sur tout le domaine n'est pas nécessaire. On s'intéresse à sa restriction à un sous-domaine ou bien à ses évaluations ponctuelles. Par exemple, pour le commutateur thermique, traité au chapitre précédent, nous devions évaluer à chaque instant uniquement la température au capteur. Lorsque l'on utilise une réduction POD, il faut d'abord reconstruire la solution sur tout le domaine à l'aide des modes POD. Or la reconstruction de la solution sur tout le domaine est de l'ordre de l'espace discret, ce qui est souvent coûteux.

Nous proposons dans ce chapitre, une seconde approche permettant de traiter le caractère mal-posé du problème d'identification. La réduction se fera en échantillonnant la matrice des données autrement dit, en sélectionnant des lignes de la matrice $\mathbf{U}$. Bien évidemment, le choix des lignes sera déterminant, et représente le cœur de ce travail. Nous proposons un algorithme basé sur la méthode d'interpolation empirique (EIM, [13]). Cette méthode est connue pour fournir des points dits *points magiques*, représentant de bons points d'interpolation pour une fonction quelconque ([117]). Sargsyan & al ont utilisé les *points magiques* afin de classifier la dynamique et ainsi de proposer un modèle réduit le plus fidèle possible ([150]). Notons que dans leurs travaux, le modèle réduit est obtenu à l'aide d'une méthode POD, la sélection de points n'est utilisée que pour classifier la dynamique.

Le problème d'échantillonnage de la matrice des données $\mathbf{U}$ est similaire au placement optimal de capteurs afin de reconstruire au mieux la solution. Cette problématique a été largement étudiée dans la littérature. On pourra citer, par exemple les travaux de Hammond & al, qui utilisent l'algorithme EIM afin de place des capteurs pour mesurer au mieux les champs de pollution ([83]). Krause & al utilisent un Processus Gaussien pour modéliser la diffusion de la température dans une pièce et trouver ainsi la position optimale de capteurs ([100]). Kianfar & al utilisent quant à eux des algorithmes de clustering afin de regrouper des capteurs en fonction des profils mesurés et ne conservent que ceux apportant une information complémentaire. L'application de leurs travaux est le placement optimal de capteurs sur une autoroute afin d'améliorer la prédiction du temps de trajet ([98]). Enfin Seeman ([156]), utilise l'algorithme *Random Forest* (RF) afin de classifier l'état du système en fonction des données de 96 capteurs. Les capteurs choisis sont ceux dont la prédiction d'état est la meilleure.

Une autre approche plus algébrique, consisterait à sélectionner des lignes de la matrice $\mathbf{U}$ de façon à minimiser l'erreur d'approximation faible rang $\|\mathbf{U} - \mathbf{C}\,\mathbf{U}_S\|$, où $\mathbf{C}$ est une matrice permettant de reconstruire $\mathbf{U}$ et $\mathbf{U}_S$ la matrice constituer des lignes

sélectionnées. Ce problème est connu sous le nom de *Column Sub-set Selection Problem* (CSSP).

Nous présenterons tout d'abord l'identification d'une EDP d'évolution à partir de données échantillonnées. Une analyse numérique ainsi qu'une comparaison avec la sélection de variables en régression multilinéaire sera présentée. Dans une deuxième partie, nous présenterons l'utilisation de la méthode EIM pour procéder à l'échantillonnage de la matrice de données $\mathbf{U}$. À la troisième section, nous construirons un opérateur d'interpolation permettant de reconstruire la solution sur tout le domaine. Enfin, dans la partie numérique, nous comparerons le modèle réduit appris à l'aide de la méthode EIM et d'autres algorithmes d'échantillonnage, sur un problème de diffusion et un problème d'ondes avec des conditions aux limites dynamiques.

4.1 Identification de la dynamique par échantillonnage

4.1.1 Cas standard

Dans cette section, nous chercherons à identifier le modèle

$$\partial_t u(\boldsymbol{x}, t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x}, t) = 0, \tag{4.1}$$

à partir de la matrice de données $\mathbf{U}$ obtenue par le solveur HF. Pour ce faire, on considère les deux ensembles d'entraînement suivant :

$$\mathcal{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^{N_t-1} \\ | & | & & | \end{bmatrix} \quad \text{et} \quad \mathcal{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^2 & \mathbf{u}^3 & \cdots & \mathbf{u}^{N_t} \\ | & | & & | \end{bmatrix}. \tag{4.2}$$

Cependant, comme nous l'avons vu au **Chapitre 2**, travailler avec ces ensembles d'entraînement n'est pas la bonne approche. Nous avions alors utilisé la méthode POD pour réduire la dimension des données d'entraînement. Ainsi à chaque vecteur $\mathbf{u}^n \in \mathbb{R}^{N_x}$ on associait une représentation basse dimension $\mathbf{a}^n \in \mathbb{R}^K$.

Dans ce chapitre nous souhaitons réduire les données tout en conservant leur interprétabilité. C'est pourquoi, nous échantillonnerons la matrice $\mathbf{U}$ en sélectionnant des lignes de celle-ci. Introduisons alors l'application

$$\sigma : \{1, \cdots, K\} \to \{1, \cdots, N_x\} \quad \text{avec } K \ll N_t.$$

avec comme précédemment, K la dimension du modèle réduit, N_x la dimension du modèle complet et N_t le nombre de pas de temps.

Les nouveaux ensembles d'entraînement sont alors :

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1_{\sigma(1:K)} & \mathbf{u}^2_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t-1}_{\sigma(1:K)} \\ | & | & & | \end{bmatrix} \quad \text{et} \quad Y = \begin{bmatrix} | & | & & | \\ \mathbf{u}^2_{\sigma(1:K)} & \mathbf{u}^3_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t}_{\sigma(1:K)} \\ | & | & & | \end{bmatrix}. \tag{4.3}$$

Et l'identification de la dynamique sera finalement obtenue en résolvant le problème d'optimisation :

$$\arg\min_A \; J(A) := \frac{1}{2} \left\| AX - Y \right\|_F^2,$$

dont l'unique solution est $A^* := YX^\dagger$. On a alors appris un modèle réduit de la forme :

$$\mathbf{u}_r^{n+1} = A\,\mathbf{u}_r^n, \tag{4.4}$$

avec $\mathbf{u}_r^n$ la prédiction de la solution u, i.e. $\left(\mathbf{u}_r^n\right)_k \approx \mathbf{u}^n_{\sigma(k)}$.

4.1.2 Extension de la méthode

Lorsque le modèle EDP est plus complexe que le modèle (4.1), il est toujours possible d'apprendre un modèle réduit en réorganisant les ensembles d'entraînement. Le tableau ci-dessous résume les ensembles d'entraînement à utiliser selon la forme du modèle EDP.

Forme de l'EDP	**Ensemble d'entrée X**	**Ensemble d'arrivée Y**
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = f(\boldsymbol{x})$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^1_{\sigma(1:K)} & \mathbf{u}^2_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t-1}_{\sigma(1:K)} \\ \vert & \vert & & \vert \\ 1 & 1 & \cdots & 1 \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^2_{\sigma(1:K)} & \mathbf{u}^3_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t}_{\sigma(1:K)} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_t u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{\theta}(t) \rangle$ (avec apprentissage de $\boldsymbol{\theta}(t)$)	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^1_{\sigma(1:K)} & \mathbf{u}^2_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t-1}_{\sigma(1:K)} \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^2_{\sigma(1:K)} & \mathbf{u}^3_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t}_{\sigma(1:K)} \\ \vert & \vert & & \vert \\ \boldsymbol{\theta}^2 & \boldsymbol{\theta}^3 & \cdots & \boldsymbol{\theta}^{N_t} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_{tt} u(\boldsymbol{x},t) + \mathcal{L}(\boldsymbol{x})\, u(\boldsymbol{x},t) = 0$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^1_{\sigma(1:K)} & \mathbf{u}^2_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t-2}_{\sigma(1:K)} \\ \vert & \vert & & \vert \\ \mathbf{u}^2_{\sigma(1:K)} & \mathbf{u}^3_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t-1}_{\sigma(1:K)} \\ \vert & \vert & & \vert \end{bmatrix}$	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^3_{\sigma(1:K)} & \mathbf{u}^4_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t}_{\sigma(1:K)} \\ \vert & \vert & & \vert \end{bmatrix}$
$\partial_t u(\boldsymbol{x},t) + \langle \boldsymbol{\mathcal{L}}(\boldsymbol{x})\, u(\boldsymbol{x},t), \boldsymbol{\theta}(t) \rangle = \langle \boldsymbol{g}, \boldsymbol{\theta}(t) \rangle$ avec $\boldsymbol{\theta}(t) = \boldsymbol{\theta}\Big(\ell\big(u(\cdot,t)\big), \boldsymbol{\theta}(t^-)\Big)$	$\big[\boldsymbol{\Theta} \otimes X_1\big]_{:,\pi}$, avec X_1 et π introduit au chapitre 2	$\begin{bmatrix} \vert & \vert & & \vert \\ \mathbf{u}^2_{\sigma(1:K)} & \mathbf{u}^3_{\sigma(1:K)} & \cdots & \mathbf{u}^{N_t}_{\sigma(1:K)} \\ \vert & \vert & & \vert \end{bmatrix}$

TABLE 4.1 Ensembles d'entraînement à utiliser selon la forme du modèle EDP.

4.1.3 Analogie avec la sélection de variables en régression multilinéaire

La qualité du modèle réduit (4.4) dépend du choix de l'application σ.
Reprenons le modèle (4.1), ce modèle est linéaire, il existe donc une matrice $\mathcal{A}$ telle que

$$\mathcal{A}\,\mathcal{X} = \mathcal{Y}, \qquad \text{avec } \mathcal{X}, \mathcal{Y} \quad \text{définis en (4.2).}$$

Ainsi, pour tout temps t^n et tout indice $k \in \{1, \cdots K\}$, on a la relation

$$\mathbf{u}^{n+1}_{\sigma(k)} = \mathcal{A}_{\sigma(k),1}\, \mathbf{u}^n_1 + \mathcal{A}_{\sigma(k),2}\, \mathbf{u}^n_2 + \cdots + \mathcal{A}_{\sigma(k),N_x}\, \mathbf{u}^n_{N_x}.$$

Ce problème peut donc être vu comme un problème de régression multilinéaire. La réduction de dimension, revient à approcher la quantité $\mathbf{u}^{n+1}_{\sigma(k)}$ par la somme pondérée

$$\beta_{k,1}\, \mathbf{u}^n_{\sigma(1)} + \beta_{k,2}\, \mathbf{u}^n_{\sigma(2)} + \cdots + \beta_{k,K}\, \mathbf{u}^n_{\sigma(K)}.$$

Il est souvent utile de selectionner les prédicteurs, on parle de selection de variables([94]). Cette dernière est d'ailleurs souvent effectuée en calculant la p-value de chaque quantité $\mathbf{u}_i^n$, c'est-à-dire la probabilité que l'on puisse retrouver la quantité $\mathbf{u}_{\sigma(k)}^{n+1}$ en omettant la dépendance de la donnée $\mathbf{u}_i^n$. Dans notre cas, N_x est supposé très grand, il ne serait donc pas judicieux de calculer la p-value associée à chaque $\mathbf{u}_i^n$ afin de choisir lesquelles sont négligeables. On choisira donc par la suite, d'autres méthodes de sélection de variables.

4.1.4 Analyse numérique du modèle réduit appris par échantillonnage

Considérons dans cette section un opérateur d'interpolation $\mathscr{I} : \mathbb{R}^K \to \mathbb{R}^{N_x}$, permettant de reconstruire la solution $\mathbf{u}^n$. On a alors

$$\left(\mathscr{I}\mathbf{u}^n_{\sigma(1:K)}\right)_i = \sum_{l=1}^{K} c_l^i \, \mathbf{u}^n_{\sigma(l)} \approx \mathbf{u}_i^n, \qquad \forall 1 \leq n \leq N_t, \quad \forall 1 \leq i \leq N_x.$$

Soit l'opérateur d'interpolation $\mathscr{I}^*$ minimisant l'erreur de réduction pour la norme de Frobenius, i.e.

$$\mathscr{I}^* = \arg\min_{\mathscr{I}} \mathcal{E}_{interp}(\sigma, \mathbf{U}, \mathscr{I}) := \left\| \mathbf{U} - \begin{bmatrix} | & & | \\ \mathscr{I}\mathbf{u}^1_{\sigma(1:K)} & \cdots & \mathscr{I}\mathbf{u}^{N_t}_{\sigma(1:K)} \\ | & & | \end{bmatrix} \right\|_F$$

De plus, on définira l'erreur de reconstruction associée à l'opérateur d'interpolation $\mathscr{I}^*$ par :

$$\mathcal{E}^*_{interp}(\sigma, \mathbf{U}) := \min_{\mathscr{I}} \mathcal{E}_{interp}(\sigma, \mathbf{U}, \mathscr{I}).$$

Au **Chapitre 2** nous avions défini la matrice $\tilde{A}$ comme étant la représentation basse dimension de la matrice $\mathcal{A}$ dans l'espace POD. Dans le même esprit, on définira ici la matrice $\tilde{A}$ par :

$$\tilde{A}_{k,l} = \sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i} \, c_l^i, \tag{4.5}$$

avec c_l^i les coefficients correspondant à l'opérateur d'interpolation $\mathscr{I}^*$.

Afin de quantifier et comprendre l'importance du choix de l'application σ, nous établissons le résultat suivant :

Proposition 4. *Soit $\tilde{A}$ la représentation basse dimension définie en* (4.5)*, on a alors*

$$\|\tilde{A}\,X - Y\|_F \leq \mathcal{E}^*_{interp}(\sigma, \mathbf{U})\,\rho(\mathcal{A}).$$

Démonstration. Sous l'hypothèse qu'il existe une matrice $\mathcal{A}$ telle que

$$\mathcal{A}\,\mathcal{X} = \mathcal{Y},$$

nous avons pour tout indice $\sigma(k)$ et pour tout temps t^n :

$$\mathbf{u}^{n+1}_{\sigma(k)} = \sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i}\,\mathbf{u}^n_i$$

Par ailleurs, en utilisant l'opérateur d'interpolation $\mathscr{I}^*$, on a :

$$\mathbf{u}^n_i = \sum_{l=1}^{K} c^i_l\,\mathbf{u}^n_{\sigma(l)} + \left(\mathbf{u}^n_i - \left(\mathscr{I}^*\mathbf{u}^n_{\sigma(1:K)}\right)_i\right)$$

On peut alors écrire

$$\begin{aligned}
\left(\tilde{A}\,X - Y\right)_{k,n} &= \sum_{l=1}^{K} \tilde{A}_{k,l}\,\mathbf{u}^n_{\sigma(l)} - \sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i}\left(\sum_{l=1}^{K} c^i_l\,\mathbf{u}^n_{\sigma(l)} + \left(\mathbf{u}^n_i - \left(\mathscr{I}^*\mathbf{u}^n_{\sigma(1:K)}\right)_i\right)\right) \\
&= \sum_{l=1}^{K}\left(\tilde{A}_{k,l} - \sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i}\,c^i_l\right)\mathbf{u}^n_{\sigma(l)} - \sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i}\left(\mathbf{u}^n_i - \left(\mathscr{I}^*\mathbf{u}^n_{\sigma(1:K)}\right)_i\right) \\
&= -\sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i}\left(\mathbf{u}^n_i - \left(\mathscr{I}^*\mathbf{u}^n_{\sigma(1:K)}\right)_i\right)
\end{aligned}$$

où l'on a utilisé la définition (4.5) de $\tilde{A}$ pour la dernière égalité. On obtient finalement que

$$\begin{aligned}
\|\tilde{A}\,X - Y\|^2_F &= \sum_{k,n}\left|\sum_{i=1}^{N_x} \mathcal{A}_{\sigma(k),i}\left(\mathbf{u}^n_i - \left(\mathscr{I}^*\mathbf{u}^n_{\sigma(1:K)}\right)_i\right)\right|^2 \\
&\leq \sum_{n=1}^{N_t-1} \|\mathcal{A}\|^2_2\,\|\mathbf{u}^n - \mathscr{I}^*\mathbf{u}^n_{\sigma(1:K)}\|^2_2 \\
&= \left(\mathcal{E}^*_{interp}(\sigma, \mathbf{U})\right)^2 \rho(\mathcal{A})^2
\end{aligned}$$

ce qui conclut la démonstration. □

Remarque 16. *La proposition ci-dessus est semblable à la* **Proposition 1** *établie au* **Chapitre 2**. *À la différence qu'ici l'erreur de réduction est donnée par l'estimation* $\mathcal{E}^*_{interp}(\sigma, \mathbf{U})$.

Pour la matrice apprise $\hat{A}$, que l'on rappelle est définie par la *formule DMD* :

$$\hat{A} = Y X^{\dagger}.$$

On a le résultat suivant, semblable à celui obtenu à la **Proposition 2** :

Théorème 4. *Soit* $\hat{A}$ *la matrice du modèle réduit appris, on a alors*

$$\|\hat{A}\, X - Y\|_F \leq \mathcal{E}^*_{interp}(\sigma, \mathbf{U})\, \rho(\mathcal{A}) \Big(1 + \kappa_F(X)\Big).$$

Démonstration. La preuve s'obtient de manière analogue à celle présentée au **Chapitre 2**. □

Ainsi l'erreur de prédiction commise par la méthode DMD, dépend de l'erreur d'interpolation $\mathcal{E}^*_{interp}(\sigma, \mathbf{U})$.

4.2 Échantillonnage des données par la méthode EIM

Il nous faut à présent construire une application σ de telle sorte que l'erreur de reconstruction $\mathcal{E}^*_{interp}(\sigma, \mathbf{U})$ soit minimale.
On souhaiterait ainsi construire l'application σ^* définie par

$$\sigma^* := \arg\min_{\sigma} \mathcal{E}^*_{interp}(\sigma, \mathbf{U}). \tag{4.6}$$

Bien que la fonctionnelle $\mathcal{E}^*_{interp}(\sigma, \mathbf{U})$ puisse être calculée, le problème d'optimisation combinatoire (4.6) semble très coûteux à résoudre. En effet, l'évaluation de la fonctionnelle en chaque σ est de l'ordre de N_x, et le nombre d'applications σ admissibles vaut $\frac{N_x!}{K!\,(N_x-K)!}$.

On propose alors une variante de la méthode EIM permettant de construire de façon itérative l'application σ. Par ailleurs, l'algorithme construit également un opérateur d'interpolation, que l'on notera $\mathbf{Q}$, conservant les lignes de la matrice $\mathbf{U}$ sélectionnées. Cette propriété est importante, car elle permet de s'assurer que les mesures capteurs sont conservées après reconstruction.

4.2.1 Présentation d'une variante de la méthode EIM

La méthode *Empirical Interpolation Method* (EIM,[13]) nous permet de construire une application σ et un opérateur de reconstruction $\mathbf{Q}$. Cependant nous souhaiterions avoir la relation

$$\mathbf{U} \approx \mathbf{Q}\,\mathbf{U}_{\sigma,\cdot}$$

ce qui n'est pas le cas avec la méthode EIM.
Pour obtenir cette relation, De Vuyst & al ont modifié la construction de l'opérateur $\mathbf{Q}$, dans leur variante nommée *Tensorized Empirical Interpolation Method* (TEIM,[49]). L'opérateur d'interpolation ainsi construit $\mathbf{Q}$ satisfait de plus,

$$\mathcal{P}\mathbf{Q} = \mathbf{I}_K \text{ avec } \mathcal{P}_{k,i} = \delta_{\sigma(k),i},$$

ce qui permet d'assurer que les lignes sélectionnées sont conservées lors de la reconstruction, autrement dit

$$\mathbf{U}_{\sigma,\cdot} = \left(\mathbf{Q}\,\tilde{\mathbf{U}}\right)_{\sigma,\cdot}.$$

Finalement, afin de réduire le temps de calcul et d'obtenir une meilleure identification des lignes à sélectionner, l'approche présentée ici sera légèrement différente. On sélectionnera les indices de lignes en travaillant avec leur représentation basse dimension, que l'on notera $\mathbf{w}^n$. On utilisera alors la méthode POD afin de construire K modes représentatifs de l'ensemble des colonnes de $\mathbf{U}$, ces modes seront notés $\mathbf{v}_k$. Projeter chaque ligne dans l'espace POD permet de hiérarchiser l'information, et de rendre l'algorithme plus robuste.

Remarque 17. *Dans la méthode Tensorized Empirical Interpolation Method, l'opérateur* $\mathbf{Q}$ *est construit à l'aide de combinaisons linéaires des vecteurs* $\mathbf{u}^n$*. En revanche, en utilisant les représentations basses dimensions* $\mathbf{w}^n$*, l'opérateur* $\mathbf{Q}$ *est construit à partir de combinaisons linéaires de modes POD spatiaux de* $\mathbf{U}$*.*

4.2.2 Algorithme : Approximation faible rang par EIM

La méthode ci-dessus est résumée dans l'algorithme suivant :

Algorithme 3 : Approximation faible rang par EIM

Entrée : Une matrice $\mathbf{U}$ et un nombre de points K

Sortie : L'ensemble de indices de lignes sélectionner σ et l'approximation faible rang $\mathbf{U} \approx \mathbf{Q}\tilde{\mathbf{U}}$.

Réduction des données ;

$\mathbf{U}^T\mathbf{U} = \mathbf{V}\Lambda\mathbf{V}^T$,

$\mathbf{W} = \mathbf{U}\mathbf{V}_r$.

Initialisation

$j_1 \leftarrow \arg\max_j \|\mathbf{W}_{\cdot,j}\|_{\ell^\infty}, \qquad i_1 \leftarrow \arg\max_i |\mathbf{W}_{i,j_1}|, \qquad \mathbf{Q}^1_{i,1} \leftarrow \frac{\mathbf{W}_{i,j_1}}{\mathbf{W}_{i_1,j_1}}, \qquad \sigma \leftarrow \{i_1\}$;

Sélections des points;

while $k \leq K$ **do**

$j_k \leftarrow \arg\max_j \|\mathbf{W}_{\cdot,j} - \mathbf{Q}^{k-1}\mathbf{W}_{\sigma,j}\|_{\ell^\infty}, \qquad i_k \leftarrow \arg\max_i |\mathbf{W}_{i,j_1} - \mathbf{Q}^{k-1}_{i,\cdot}\mathbf{W}_{\sigma,j_1}|,$

$\mathbf{Q}^k_{i,k} \leftarrow \frac{\mathbf{W}_{i,j_k} - \mathbf{Q}^{k-1}\mathbf{W}_{\sigma,j_k}}{\mathbf{W}_{i_k,j_k} - \mathbf{Q}^{k-1}_{i_k,\cdot}\mathbf{W}_{\sigma,j_k}}, \qquad \sigma \leftarrow \{\sigma, i_k\}$;

  for $l < k$ **do**

    $\mathbf{Q}^k_{\cdot,l} \leftarrow \mathbf{Q}^{k-1}_{\cdot,l} - \mathbf{Q}^{k-1}_{i_k,l}\mathbf{Q}^k_{\cdot,k}$

$\mathbf{Q} \leftarrow \mathbf{Q}^K, \qquad \tilde{\mathbf{U}} \leftarrow \mathbf{U}_{\sigma,\cdot}$

4.3 Reconstruction de la solution à partir de données échantillonnées

Dans la **section 4.1**, nous avons présenté une approche permettant d'identifier la dynamique et d'apprendre un modèle réduit (4.4) à partir de données échantillonnées. Cependant ce modèle réduit prédit seulement l'évolution en temps de cet échantillon. Ainsi dans cette section nous souhaitons obtenir un opérateur d'interpolation permettant de reconstruire la solution sur tout le domaine. En utilisant la définition de l'opérateur d'interpolation présenté en **section 4.1**, cela revient à chercher l'opérateur d'interpolation $\mathscr{I} : \mathbb{R}^K \mapsto \mathbb{R}^{N_x}$ minimisant

$$\left\| \mathbf{U} - \begin{bmatrix} | & & | \\ \mathscr{I}\mathbf{u}^1_{\sigma(1:K)} & \cdots & \mathscr{I}\mathbf{u}^{N_t}_{\sigma(1:K)} \\ | & & | \end{bmatrix} \right\|_\xi, \qquad \text{avec } \xi = F, \text{Max}$$

Une formulation matricielle de ce problème consiste à trouver une matrice $\mathbf{C}^*$ définie par

$$\mathbf{C}^* := \underset{\mathbf{C}\in\mathbb{R}^{N_x\times K}}{\arg\min} \|\mathbf{U} - \mathbf{C}\tilde{\mathbf{U}}\|_\xi, \qquad \text{avec } \tilde{\mathbf{U}}_{\cdot,n} = \mathbf{u}^n_{\sigma(1:K)}. \tag{4.7}$$

Dans le cas où $\xi = F$, i.e. pour la norme de Frobenius, on pourra expliciter la matrice $\mathbf{C}^*$. Lorsque $\xi = \text{Max}$, cette matrice ne pourra pas être explicitée, on utilisera alors **l'algorithme 3** pour construire une matrice $\mathbf{C}$.

4.3.1 Optimalité en norme de Frobenius

Réécrivons le problème d'optimisation (4.7) sous la forme,

$$\mathbf{C}^* := \underset{\mathbf{C}\in\mathbb{R}^{N_x\times K}}{\arg\min} \frac{1}{2}\left\|\mathbf{U} - \mathbf{C}\tilde{\mathbf{U}}\right\|_F^2$$

$\mathbf{C}^*$ est alors l'unique matrice définie par

$$\mathbf{C}^* = \mathbf{U}\tilde{\mathbf{U}}^\dagger. \tag{4.8}$$

D'après le **théorème 4**, l'erreur de prédiction commise par la méthode DMD, dépend de la quantité $\mathcal{E}^*_{interp}(\sigma, \mathbf{U})$. On peut ainsi donner une écriture analytique de cette quantité,

$$\mathcal{E}^*_{interp}(\sigma, \mathbf{U}) = \|\mathbf{U} - \mathbf{U}\tilde{\mathbf{U}}^\dagger\,\tilde{\mathbf{U}}\|_F.$$

Et estimer a priori le bon choix de l'application σ.

4.3.2 Sous-optimalité en norme Max

L'algorithme 3 génère, en plus de l'application σ, un opérateur d'interpolation $\mathbf{Q}$ vérifiant $\mathcal{P}\mathbf{Q} = \mathbf{I}_K$ avec $\mathcal{P}_{k,i} = \delta_{\sigma(k),i}$. Cette propriété assure que les lignes sélectionnées restent inchangées après reconstruction, ce qui n'est pas le cas pour l'interpolateur $\mathbf{C}^*$. Bien que la matrice $\mathbf{Q}$ ne soit pas solution du problème (4.7), elle satisfait la propriété de sous-optimalité :

$$\left\|\mathbf{U} - \mathbf{Q}\tilde{\mathbf{U}}\right\|_{\max} \leq \left(1 + \mathbf{\Lambda}_x\right) \min_{\mathbf{C}\in\mathbb{R}^{N_x\times K}} \left\|\mathbf{U} - \mathbf{C}\tilde{\mathbf{U}}\right\|_{\max}$$

où $\mathbf{\Lambda}_x$ est la constante de Lebesgue associé à l'opérateur d'interpolation $\mathbf{Q}$.

4.4 Applications numériques

Dans cette section, deux exemples d'applications numériques seront présentées et traitées à l'aide de la méthode d'échantillonnage des sonnées introduite dans ce chapitre. On considérera tout d'abord, une EDP parabolique dont la condition de Dirichlet est solution d'une EDO. Puis dans un second temps, on considérera un problème d'ondes dynamiques.

Par ailleurs, l'échantillonnage obtenu à l'aide de l'algorithme *Approximation faible rang par EIM* sera comparé à celui obtenu par différents autres algorithmes. On confrontera ainsi **l'algorithme 3** à l'algorithme des k-médoïdes ([129], [132]), aux algorithmes NOCS et RandCGSS issus de la communauté CSSP, et à un échantillonnage uniforme. Afin de comparer les algorithmes, on définira l'erreur relative de reconstruction pour chaque point du domaine par,

$$err_{local} := \frac{\|\mathbf{u}_i - \left(\mathbf{C}^* \, \mathbf{u}_\sigma\right)_i\|_{\ell^2}}{\|\mathbf{U}\|_F},$$

avec $\mathbf{C}^*$ définie en (4.8).
On définira de plus le taux de reconstruction par,

$$\tau := 100 \, \frac{\|\mathbf{U} - \mathbf{U}_r\|_F}{\|\mathbf{U} - \mathbf{C}^* \, \mathbf{U}_{\sigma,\cdot}\|_F},$$

où $\mathbf{U}_r$ est l'approximation de rang faible obtenue par une décomposition en valeurs singulières (SVD).

4.4.1 Problème de diffusion avec condition aux limites dynamiques

4.4.1.1 Présentation du modèle et génération des données

Considérons tout d'abord le cas d'une équation de diffusion avec une condition de Dirichlet évoluant au cours du temps selon une EDO. Mathématiquement, le modèle

sera défini par :

$$\begin{cases} \partial_t u(x,t) - \kappa \partial_{xx} u(x,t) = 0, & \text{dans } [0,1] \times (0,T), \\ u(0,t) = 0, \quad u(1,t) = \boldsymbol{\theta}_1(t), & \forall t \in (0,T), \\ \dot{\boldsymbol{\theta}} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{\theta}, & \\ u(x,0) = u_0(x), & \text{dans } [0,1] \end{cases} \tag{4.9}$$

Comme fait aux chapitres précédents, les données seront stockées dans les matrices $\mathbf{U}$ et $\boldsymbol{\Theta}$. Le tableau ci-dessous résume les paramètres physiques et numériques utilisés.

Paramètres	**Valeurs**
Conductivité thermique (κ)	$0.01\ W \cdot m^{-1} \cdot K^{-1}$
Temps final (T)	$4\pi\ sec$
Pas d'espace (δx)	2×10^{-3}
Pas de temps (δt)	5×10^{-3}

TABLE 4.2 Paramètres physique du problème

La figure ci-dessous représente la solution u calculée par le solveur HF à plusieurs temps.

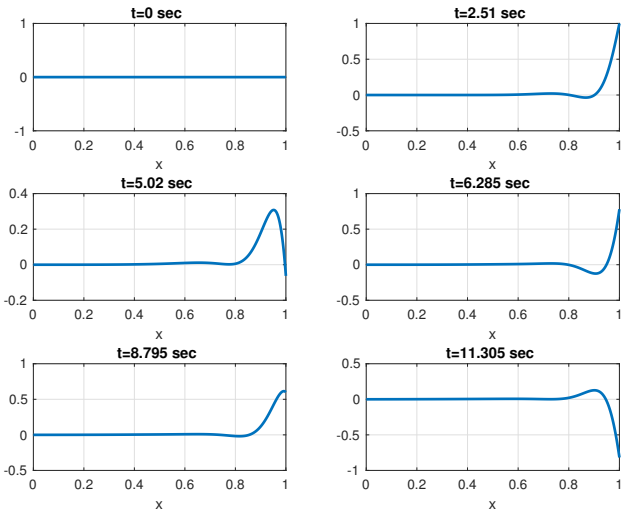
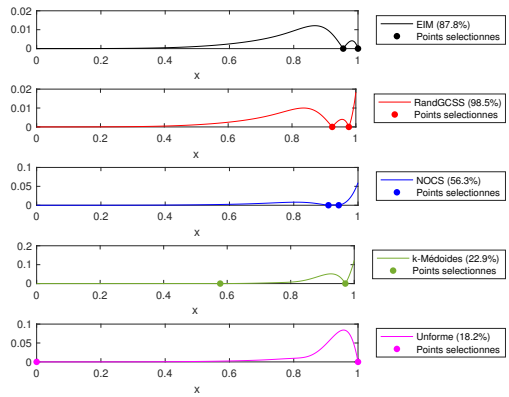


FIGURE 4.1 Solution u calculée par le solveur HF à différents pas de temps

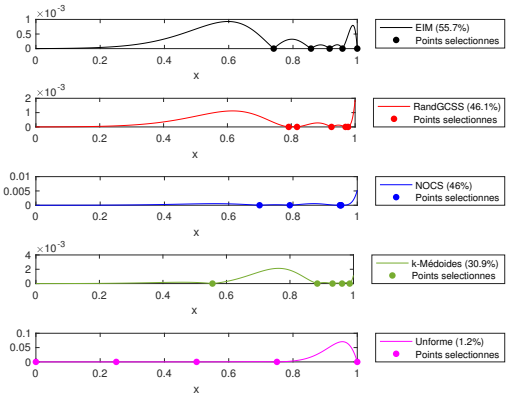
On peut ainsi remarquer que le problème est fortement dynamique sur la partie droite du domaine et quasi-statique sur la partie gauche.

4.4.1.2 Comparaison de l'échantillonnage des données

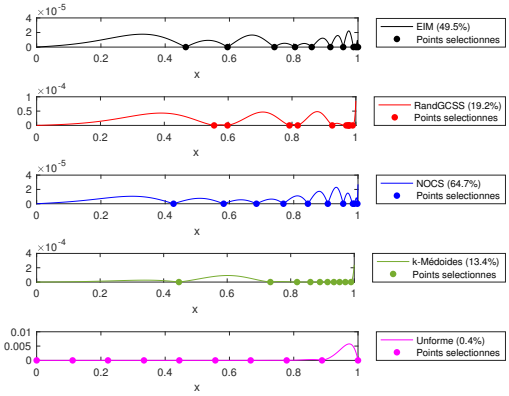
Dans ce paragraphe, on comparera différents algorithmes d' échantillonnages appliqués à la matrice $\mathbf{U}$. La ligne i de la matrice $\mathbf{U}$ contient la solution u_h calculée par le solveur HF au point du maillage x_i, pour tout temps t^n.
Au vu de la figure 4.1, la sélection devra se faire principalement aux points localisés à droite du domaine. Représentons alors l'erreur de reconstruction en chaque point du domaine, err_{local}, suivant les K points sélectionnés par les algorithmes d'échantillonnage. On affichera également, dans la légende, le taux de reconstruction τ.



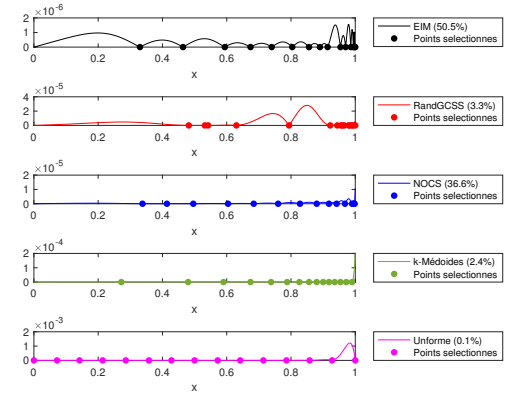
(a) $K = 2$



(b) $K = 5$



(c) $K = 10$



(d) $K = 15$

FIGURE 4.2 Erreur de reconstruction suivant la sélection de points

Lorsque $K = 2$, l'algorithme RandGCSS est quasi-optimal en faisant presque aussi bien que la réduction SVD. **L'algorithme 3** est un peu moins bon. En revanche, lorsque K augmente seul **l'algorithme 3** semble conserver un bon taux de reconstruction. Du fait de son échantillonnage aléatoire, l'algorithme NOCS ne donne pas toujours de bons résultats. Quant à l'algorithme RandGCSS et l'algorithme des k-médoides, ils semblent se dégrader avec l'augmentation de K. Enfin, comme attendu, l'échantillonnage uniforme est dans ce cas une très mauvaise approche, en effet la solution u est quasi-statique sur la partie gauche et fortement dynamique sur la partie droite.

Remarque 18. *Les mauvais résultats obtenus en utilisant l'algorithme des k-médoïdes s'expliquent par le partitionnement qu'il opère. En effet, en regroupant les vecteurs* $\mathbf{u}_i$ *selon leur similarité, on sélectionne les vecteurs sans prendre en compte la richesse de l'information contenue dans les autres vecteurs* $\mathbf{u}_j$ *sélectionnés, contrairement à* **l'algorithme 3**. *Pour mieux comprendre cela, affichons la dynamique aux 10 points sélectionnés par les k-médoides et* **l'algorithme 3**.

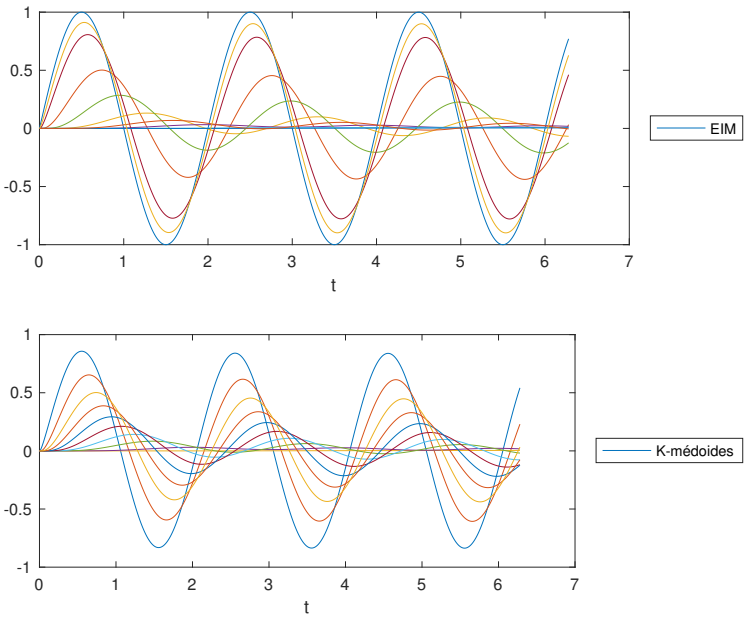


FIGURE 4.3 Dynamique aux points sélectionnés

On remarque que l'algorithme des k-médoides s'efforce de choisir le plus de profils possible entre la courbe bleue et la courbe mauve, tandis que **l'algorithme 3** *n'en choisit que quelques-uns, les autres "appartenant" à l'espace engendré par les vecteurs sélectionnés. Ainsi l'algorithme des k-médoides délaisse d'autres profils.*

4.4.1.3 Comparaison des modèles réduits appris

Une fois les points sélectionnés, nous pouvons à présent identifier le modèle (4.9). En reprenant l'extension présentée à la **section 2.4.2**, nous apprenons un modèle réduit de la forme :

$$\begin{cases} \mathbf{u}_r^{n+1} = \hat{A}\,\mathbf{u}_r^n + \hat{B}\,\mathbf{z}^n, \\ \mathbf{z}^{n+1} = \hat{C}\,\mathbf{z}^n, \\ \hat{\mathbf{u}}^n = \mathbf{C}^*\,\mathbf{u}_r^n. \end{cases} \tag{4.10}$$

Le tableau ci-dessous présente l'erreur de prédiction pour les modèles réduits suivant les points sélectionnés. On notera en rouge l'erreur la plus importante et en vert la plus faible.

	Choix de réduction	Erreur de prédiction sur l'ensemble d'entrainement	Erreur de prédiction sur l'ensemble de validation	Erreur de réduction
K=5	SVD	$\mathbf{5.6 \times 10^{-3}}$	$\mathbf{9.8 \times 10^{-3}}$	$\mathbf{5.3 \times 10^{-3}}$
	EIM	1.1×10^{-2}	2.1×10^{-2}	9.6×10^{-3}
	NOCS	1.5×10^{-2}	2.5×10^{-2}	1.3×10^{-2}
	RandGCSS	1.5×10^{-2}	3×10^{-2}	1.2×10^{-2}
	k-médoides	2.8×10^{-2}	1×10^{-2}	1.7×10^{-2}
	Uniforme	0.4740	1.2671	0.4303
K=10	SVD	$\mathbf{1.2 \times 10^{-4}}$	$\mathbf{2.1 \times 10^{-4}}$	$\mathbf{1.1 \times 10^{-4}}$
	EIM	2.6×10^{-4}	5×10^{-4}	2.2×10^{-4}
	NOCS	2×10^{-4}	3.3×10^{-4}	1.7×10^{-4}
	RandGCSS	1×10^{-3}	1.4×10^{-3}	6×10^{-4}
	k-médoides	1.2×10^{-3}	1.5×10^{-3}	8×10^{-4}
	Uniforme	2.5×10^{-2}	2.7×10^{-2}	2.7×10^{-2}
K=15	SVD	$\mathbf{6.3 \times 10^{-6}}$	$\mathbf{1.5 \times 10^{-5}}$	$\mathbf{5.9 \times 10^{-6}}$
	EIM	1.7×10^{-5}	5×10^{-5}	1.2×10^{-5}
	NOCS	2.3×10^{-5}	7.6×10^{-5}	2.1×10^{-5}
	RandGCSS	2.1×10^{-4}	3×10^{-4}	9.4×10^{-5}
	k-médoides	2.4×10^{-4}	7.7×10^{-5}	2.4×10^{-4}
	Uniforme	5.4×10^{-3}	2.1×10^{-3}	4.5×10^{-3}

TABLE 4.3 Erreur de prédiction

Lorsque la dimension est réduite à l'aide d'une décomposition en valeurs singulières, il est raisonnable d'obtenir la meilleure prédiction du modèle réduit puisque l'erreur de réduction est optimale (d'après le théorème d'Eckart-Young-Mirsky,[57]). Ainsi, on l'utilisera alors comme erreur de référence. Pour $K = 5$, quel que soit l'algorithme d'échantillonnage utilisé, le modèle réduit appris a une erreur de prédiction semblable à l'erreur de référence, excepté dans le cas d'un échantillonnage uniforme. Lorsque l'on prend $K = 10$, l'algorithme 3 et l'algorithme NOCS sont meilleurs que les autres. Cela se confirme pour $K = 15$.

4.4.1.4 Analyse du modèle réduit appris par EIM

Étudions à présent le modèle réduit appris (4.12) par EIM. Comme expliqué au **Chapitre 2**, la méthode DMD par empilement nous fournit la matrice suivante :

$$\left(\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{D} & \hat{C} \end{array}\right) = Y X^{\dagger}$$

Lorsque l'algorithme sélectionne le point $x = 1$, il y a doublon entre la condition de Dirichlet non homogène imposée par $\mathbf{z}_1$, et l'évaluation ponctuelle de la solution en ce point. Nous sommes alors dans un cas de multicolinéarité. Si ce problème n'a pas perturbé notre prédiction, l'interprétation de notre modèle réduit en est difficile. Pour comprendre cela, considérons les points sélectionnés par EIM pour $K = 5$. Le premier point sélectionné est alors celui correspondant à la condition de Dirichlet. La matrice obtenue par la méthode DMD est :

	$x_{\sigma(1)} = 1$	$x_{\sigma(2)}$	$x_{\sigma(3)}$	$x_{\sigma(4)}$	$x_{\sigma(5)}$		$\mathbf{z}_1$	$\mathbf{z}_2$
$x_{\sigma(1)} = 1$	**0.4999**	**0.0000**	**0.0000**	**−0.0000**	**−0.0000**	|	**0.4999**	**0.0157**
$x_{\sigma(2)}$	0.0058	0.9799	0.0091	−0.0036	0.0020	|	0.0058	−0.0008
$x_{\sigma(3)}$	−0.0001	0.0001	0.9885	0.0050	0.0072	|	−0.0001	0.0000
$x_{\sigma(4)}$	0.0000	0.0001	0.0030	0.9966	−0.0008	|	0.0000	−0.0000
$x_{\sigma(5)}$	−0.0004	0.0174	0.0133	−0.0011	0.9710	|	−0.0004	0.0001
	—	—	—	—	—	—	—	—
$\mathbf{z}_1$	**0.4999**	**0.0000**	**0.0000**	**−0.0000**	**−0.0000**	|	**0.4999**	**0.0157**
$\mathbf{z}_2$	−0.0079	0.0000	0.0000	−0.0000	−0.0000	|	−0.0079	0.9999

On remarque ainsi que les deux lignes en gras sont égales, et correspondent à l'évolution de $\mathbf{z}_1^{n+1}$. De plus, le rôle joué par $\mathbf{z}_1^n$ et $\left(\mathbf{u}_r^n\right)_1$ est le même. Cela explique pourquoi les coefficients en pourpre sont égaux.

Afin d'éviter ce problème, il suffit de vérifier qu'il n'y a pas de doublon dans les matrices d'entraînement. Auquel cas, deux stratégies sont alors envisageables, la première consiste à ne conserver que le vecteur $\mathbf{z}_1$. Cela revient à considérer la condition de Dirichlet comme un terme source. La seconde, consiste à ne conserver que la solution au point $x = 1$.

Condition de Dirichlet imposée par le vecteur $\mathbf{z}_1^n$:

Considérons à présent le nouveau modèle réduit appris, en retirant de l'ensemble d'entraînement la solution au point $x = 1$. La matrice calculée par la méthode DMD

est cette fois

$$
\begin{array}{c|cccc|cc}
 & x_{\sigma(2)} & x_{\sigma(3)} & x_{\sigma(4)} & x_{\sigma(5)} & \mathbf{z}_1 & \mathbf{z}_2 \\
x_{\sigma(2)} & 0.9799 & 0.0091 & -0.0036 & 0.0020 & 0.0117 & -0.0008 \\
x_{\sigma(3)} & 0.0001 & 0.9885 & 0.0050 & 0.0072 & -0.0002 & 0.0000 \\
x_{\sigma(4)} & 0.0001 & 0.0030 & 0.9966 & -0.0008 & 0.0000 & -0.0000 \\
x_{\sigma(5)} & 0.0174 & 0.0133 & -0.0011 & 0.9710 & -0.0007 & 0.0001 \\
 & — & — & — & — & — & — \\
\mathbf{z}_1 & 0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.9999 & 0.0157 \\
\mathbf{z}_2 & -0.0000 & -0.0000 & 0.0000 & 0.0000 & -0.0157 & 0.9999
\end{array}
$$

On remarque ainsi que cette fois les termes en pourpre, correspondant à l'action de $\mathbf{z}_1^n$ sur le système et valent le double de la matrice précédente. Autrement dit, les deux modèles réduits appris prédisent la même solution.

À présent, analysons les termes sources contenus dans la matrice $\hat{B}$. La première colonne correspond à la condition de Dirichlet non homogène au point $x = 1$, tandis que la seconde colonne correspond à la condition de Dirichlet homogène au point $x = 0$. La figure ci-dessous représente ces termes sources :

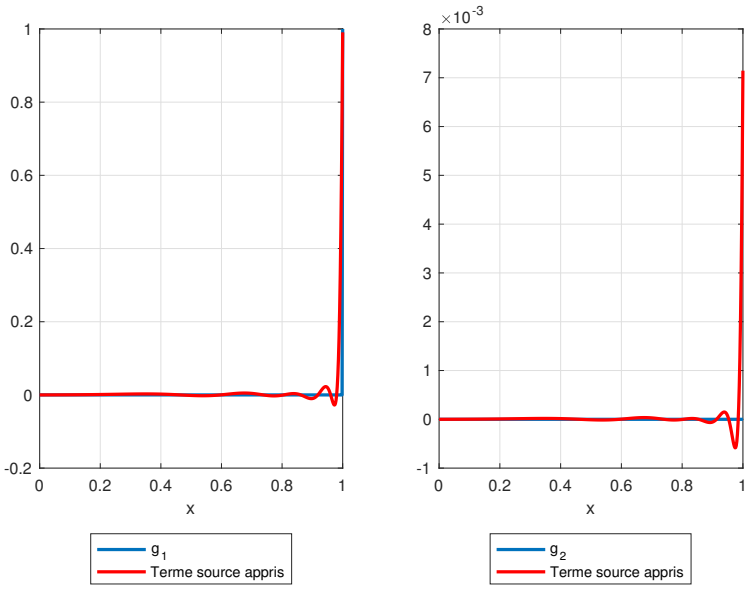


FIGURE 4.4 Termes sources identifiés par le modèle réduit

Le modèle réduit a donc bien identifié les conditions aux limites. De même, une analyse spectrale montre que les dynamiques de l'EDP et de l'EDO ont correctement été identifiées. La figure ci-dessous représente les valeurs propres des matrices $\hat{A}$ et $\hat{C}$.

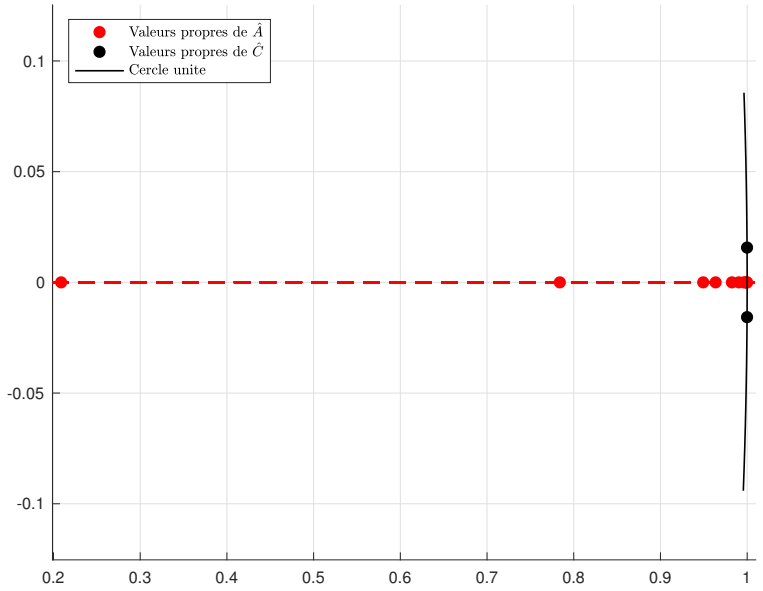


FIGURE 4.5 Valeurs propres des matrices $\hat{A}$ et $\hat{C}$

Condition de Dirichlet imposée par la solution u au point $x = 1$:
La matrice du modèle réduit obtenue dans ce cas est présentée ci-dessous :

$$
\begin{array}{c}
 \\
x_{\sigma(1)} = 1 \\
x_{\sigma(2)} \\
x_{\sigma(3)} \\
x_{\sigma(4)} \\
x_{\sigma(5)} \\
 \\
\mathbf{z}_2
\end{array}
\begin{array}{c}
\begin{array}{ccccc|c} x_{\sigma(1)} = 1 & x_{\sigma(2)} & x_{\sigma(3)} & x_{\sigma(4)} & x_{\sigma(5)} & \mathbf{z}_2 \end{array} \\
\left(\begin{array}{ccccc|c}
\mathbf{0.9999} & -0.0000 & 0.0000 & 0.0000 & -0.0000 & \mathbf{0.0157} \\
0.0117 & 0.9799 & 0.0091 & -0.0036 & 0.0020 & -0.0008 \\
-0.0002 & 0.0001 & 0.9885 & 0.0050 & 0.0072 & 0.0000 \\
0.0000 & 0.0001 & 0.0030 & 0.9966 & -0.0008 & -0.0000 \\
-0.0007 & 0.0174 & 0.0133 & -0.0011 & 0.9710 & 0.0001 \\
\hline
\mathbf{-0.0157} & -0.0000 & -0.0000 & -0.0000 & 0.0000 & \mathbf{0.9999}
\end{array}\right)
\end{array}
$$

La matrice obtenue est plus difficilement interprétable, en effet on retrouve les coefficients correspondant à l'EDO en bleu aux coins de la matrice. Cependant, au vu des trois matrices, toutes ces approches sont numériquement équivalentes dans notre cas.

4.4.2 Problème d'ondes avec condition aux limites dynamiques

4.4.2.1 Présentation du modèle et génération des données

À présent considérons un couplage EDP-EDO, avec une EDP hyperbolique et avec une EDO d'ordre 2 en temps. Ce couplage interviendra au niveau de la condition de Dirichlet mais aussi au niveau du terme source.
Le modèle que nous chercherons à identifier est le suivant :

$$\begin{cases} \partial_{tt}u(x,t) - c\partial_{xx}u(x,t) = g(x)\,\boldsymbol{\theta}_2(t) \text{ dans } [0,1]\times(0,T), \\ u(0,t) = 0, \quad u(1,t) = q_1\,\boldsymbol{\theta}_1(t), \qquad \forall t \in (0,T), \\ \\ \ddot{\boldsymbol{\theta}}(t) = \begin{pmatrix} -2\alpha_1 & 0 \\ 0 & -2\alpha_2 \end{pmatrix} \dot{\boldsymbol{\theta}}(t) + \begin{pmatrix} -(\alpha_1^2+\omega_1^2) & 0 \\ 0 & -(\alpha_2^2+\omega_2^2) \end{pmatrix} \boldsymbol{\theta}(t), \qquad \boldsymbol{\theta}(0) = q_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \\ g(x) = -q_3 \mathbb{1}_{[x_1,x_2]}, \\ u(x,0) = u_0(x), \qquad \text{dans } [0,1], \\ u_0(x) = q_4\Big(x^2 + 0.1\sin(12\pi x)\Big) \qquad \text{dans } [0,1]. \end{cases} \tag{4.11}$$

Ainsi l'onde sera excitée à la fois à son extrémité par une condition de Dirichlet dynamique, mais aussi par un terme source dynamique.
Le tableau ci-dessous, résume les paramètres physiques et numériques utilisés.

Paramètres	**Valeurs**
Intervalle du terme source g ($[x_1, x_2]$)	$[0.25, 0.75]$
Variance du terme source (τ)	0.05
Intensité de la condition de Dirichlet (q_1)	0.5
Intensité de la série temporelle (q_2)	0.1
Intensité du terme source (q_3)	1
Intensité de la condition initial (q_4)	$q_1 q_2$
Fréquence de l'activation du terme source (ω_1)	$2\,\pi$
Vitesse de dissipation du terme source (α_1)	0
Fréquence de l'activation du terme source (ω_2)	$12\,\pi$
Vitesse de dissipation du terme source (α_2)	0.15
Célérité de l'onde (c)	0.5
Temps final (T)	30 sec
Pas d'espace (δx)	5×10^{-3}
Pas de temps (δt)	$\frac{0.3}{c}\delta x$

TABLE 4.4 Paramètres physiques du problème

La figure ci-dessous représente, à différents instants, la solution u générée par le solveur HF.

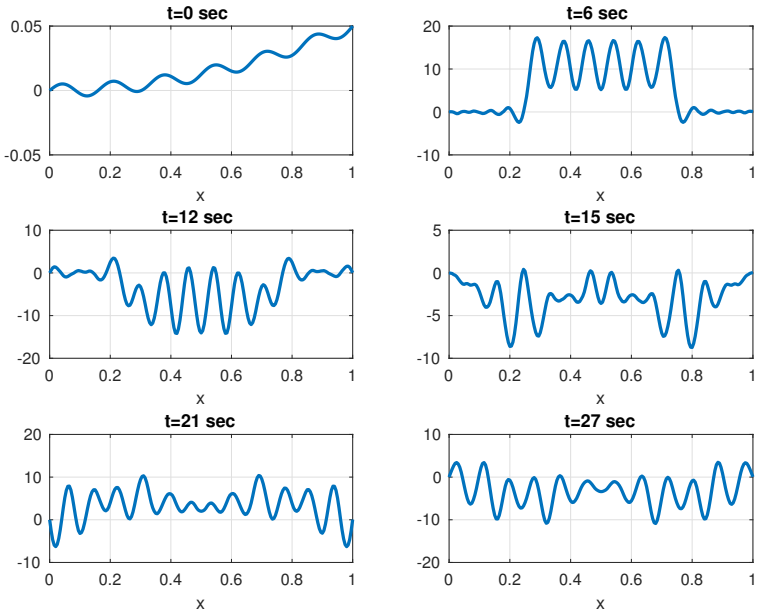
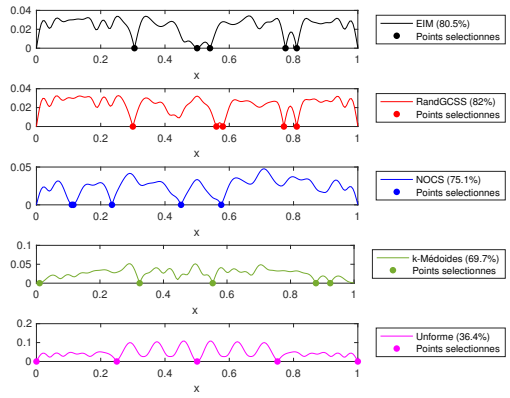


FIGURE 4.6 Solution, générée par le solveur HF, à différents pas de temps

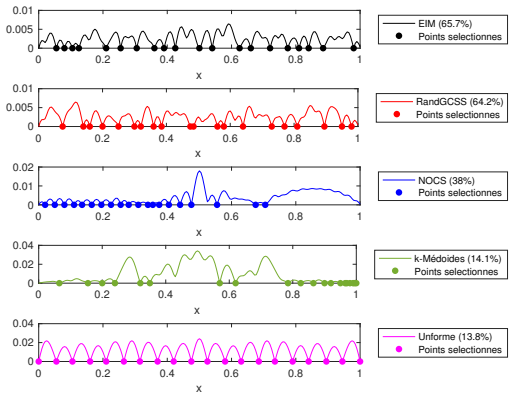
Comme attendu la solution u générée par le solveur HF est fortement dynamique, ainsi la matrice $\mathbf{U}$ contenant la solution sera très riche en information.

4.4.2.2 Comparaison de l'échantillonnage des données

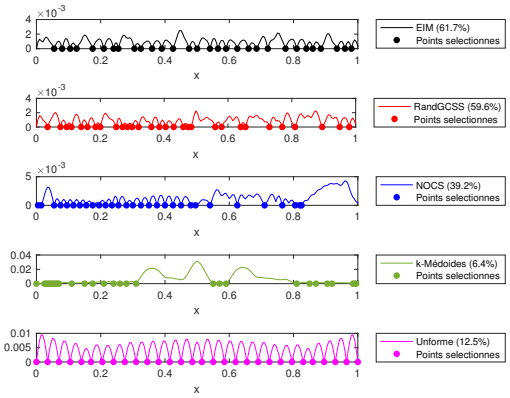
Comme précédemment, nous évaluons ici l'échantillonnage des données au travers de l'erreur de reconstruction. La figure ci-dessous présente les points sélectionnés par les différents algorithmes ainsi que le taux de reconstruction pour $K = 5, 20, 30$ et 50.



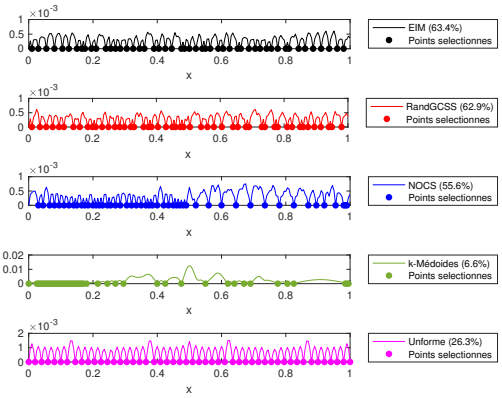
(a) $K = 5$



(b) $K = 20$



(c) $K = 30$

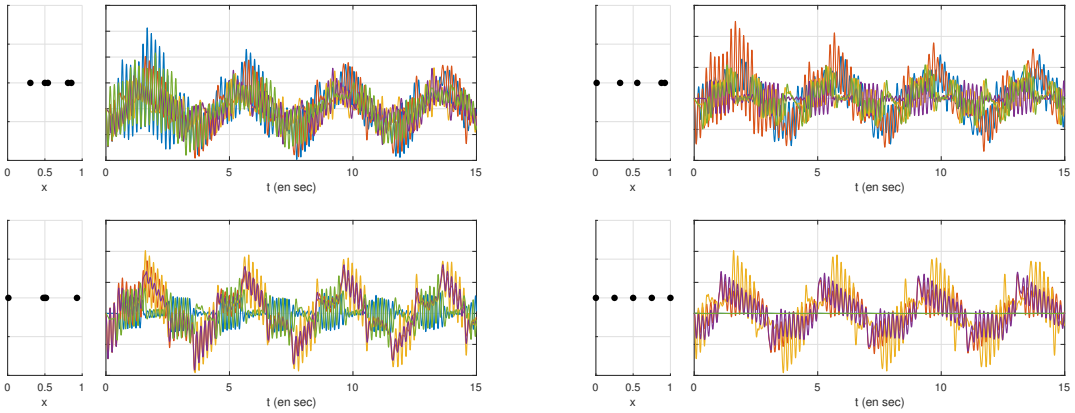


(d) $K = 50$

FIGURE 4.7 Erreur de reconstruction suivant la sélection de points

Seuls les algorithmes EIM et RandGCSS donnent de bons résultats quel que soit le nombre de points K sélectionnés. L'algorithme NOCS ne choisit pas toujours des points pertinents, il a tendance à sur-sélectionné les points à gauche du domaine. En ce qui concerne l'algorithme des k-médoides, sa performance se détériore sévèrement avec l'augmentation de K, comme c'était déjà le cas pour le problème de diffusion. Enfin l'échantillonnage uniforme ne semble pas adapté à ce problème.

Finissons par présenter les profils de dynamique retenus par les différents algorithmes, pour $K = 5$ avec les taux de reconstruction associés.



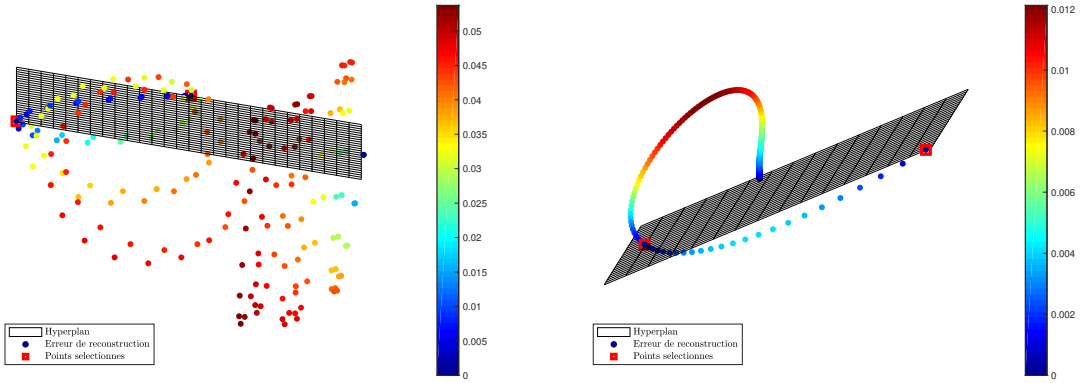
(a) (De haut en bas) EIM (80,5%) et NOCS (67,3 %)

(b) (De haut en bas) k-Médoides (69,7%) et Uniforme (36,4%)

FIGURE 4.8 Dynamique aux points sélectionnés pour $K = 5$

On pourrait penser qu'en choisissant de manière uniforme les points on retiendrait une information assez riche, ce qui n'est pas le cas pour notre problème d'ondes. On remarque que la sélection des points par les k-médoides et l'échantillonnage uniforme est assez similaire mais les taux de reconstruction bien différents (70% contre 36%), on a ici des points d'intérêt et non pas des régions d'intérêt. Notons que l'algorithme 3 donne le meilleur taux de reconstruction en sélectionnant principalement des points au centre du domaine.

Remarque 19. *Afin de comprendre la difficulté à reconstruire la solution à partir de points sélectionnés. Représentons pour chaque point du maillage x_i, les représentations basses dimensions $\mathbf{w}_{i,1:3}$, introduites à la* **section 4.2.1**. *Par ailleurs, à l'aide de l'algorithme 3 nous sélectionnerons deux points et afficherons l'hyperplan correspondant au sous-espace vectoriel engendré par les $\mathbf{w}_k$ associé à ces deux points. Enfin l'erreur de projection sur ce sous-espace vectoriel sera affichée. À titre de comparaison, la figure de droite présente le même résultat mais cette fois-ci avec les données issues du problème de diffusion* (4.9).



(a) Problème d'ondes (b) Problème de diffusion

FIGURE 4.9 Erreur d'interpolation pour $K = 2$

Ainsi il semble ne pas exister de sous-espace vectoriel permettant de les projeter avec une erreur faible. De plus il ne semble pas y avoir de structure dans les données, contrairement au problème de diffusion. Cela est dû à la richesse de l'information contenue dans $\mathbf{U}$.

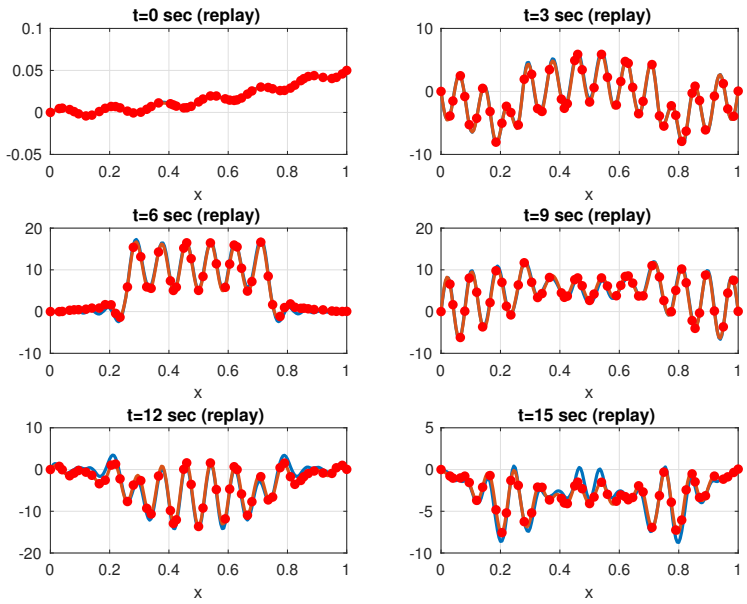
4.4.2.3 Apprentissage d'un modèle réduit par EIM

Le modèle réduit appris est de la forme :

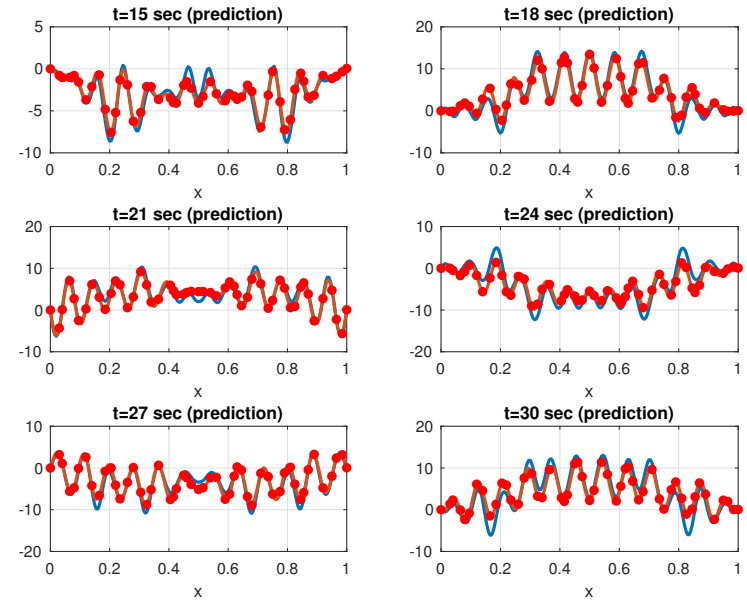
$$\begin{cases} \mathbf{u}_r^{n+1} = \hat{A}_1\,\mathbf{u}_r^{n-1} + \hat{A}_2\,\mathbf{u}_r^{n} + \hat{B}\,\mathbf{z}_2^{n}, \\ \mathbf{z}^{n+1} = \hat{C}\,\mathbf{z}^{n}, \\ \hat{\mathbf{u}}^{n} = \mathbf{C}\,\mathbf{u}_r^{n}. \end{cases} \tag{4.12}$$

Les tableaux ci-dessous présentent la comparaison entre la solution HF (en bleue) et la prédiction du modèle réduit (en rouge) pour $K = 50$ et $K = 100$. Les points rouges correspondent aux composantes du vecteur $\mathbf{u}_r^n$.

$$K = 50$$



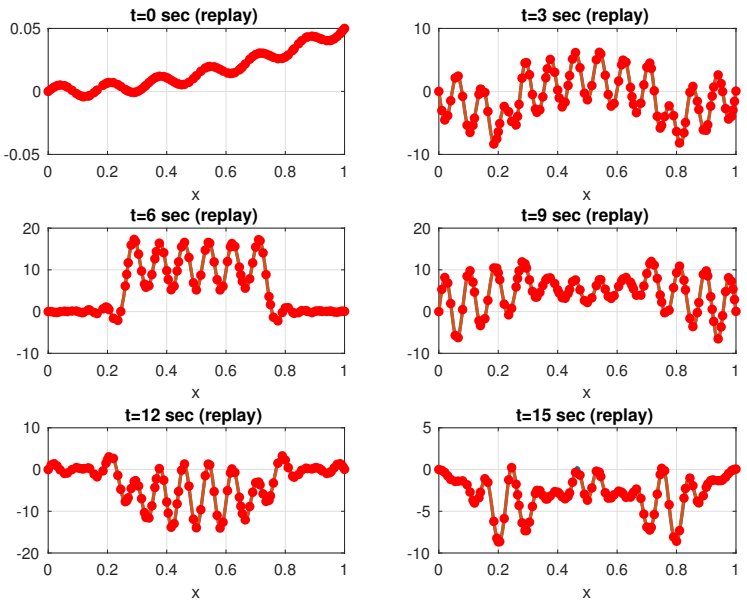
(a) Prédiction sur l'ensemble d'entrainement (entre 0 et 15 sec)



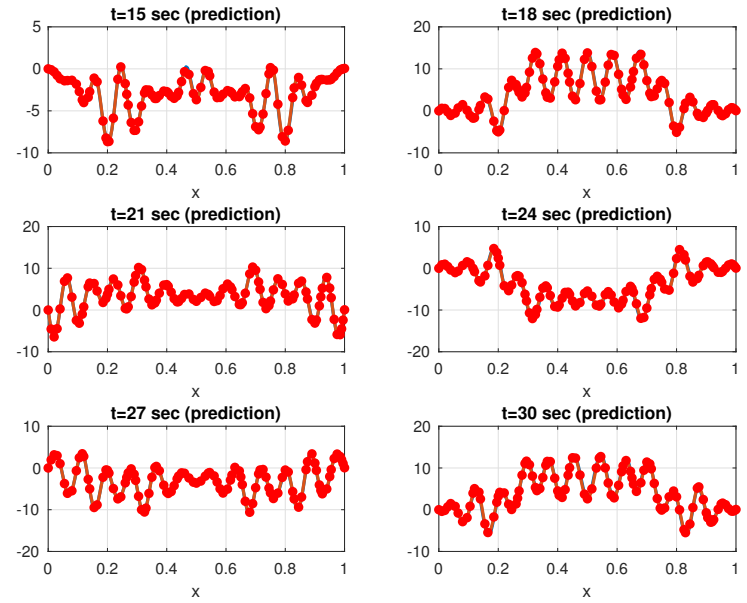
(b) Prédiction sur l'ensemble de validation (entre 15 et 30 sec)

FIGURE 4.10 Prédiction et solution HF pour $K = 50$ avec EIM

$K = 100$



(a) Prédiction sur l'ensemble d'entrainement (entre 0 et 15 sec)



(b) Prédiction sur l'ensemble de validation (entre 15 et 30 sec)

FIGURE 4.11 Prédiction et solution HF pour $K = 100$ avec EIM

Lorsque l'on sélectionne 50 points, le modèle réduit appris semble, en temps long, ne pas être fidèle au solveur HF. En revanche lorsque $K = 100$, il semble reproduire parfaitement le comportement de l'onde au cours du temps.

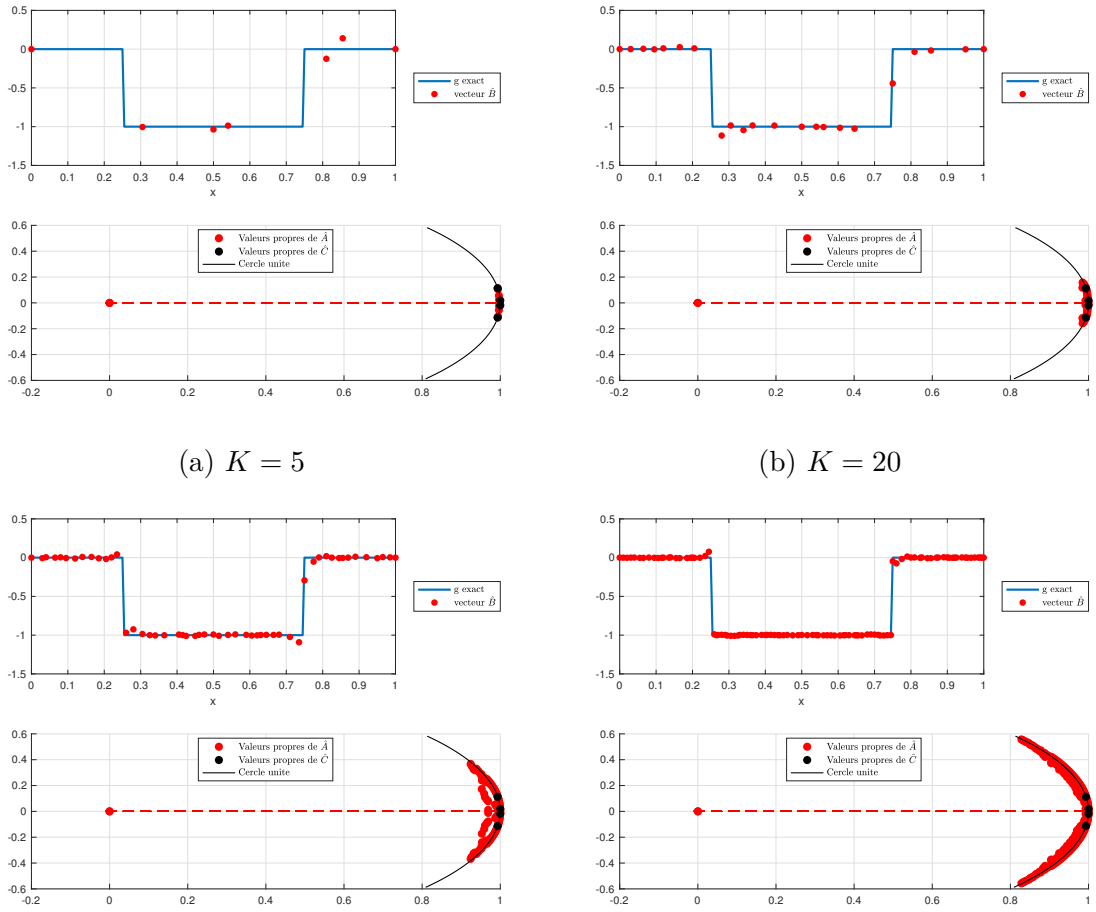
Remarque 20. *Les prédictions obtenues en utilisant les autres algorithmes d'échantillonnage sont présentées en* **Annexe C**.

4.4.2.4 Analyse du modèle réduit appris par EIM

Terminons par analyser le modèle réduit appris. Nous cherchons à identifier les matrices $\hat{A}_1$ et $\hat{A}_2$ correspondant à l'équation des ondes, le vecteur $\hat{B}$ correspondant à la valeur du terme source g aux points sélectionnés, et enfin la matrice $\hat{C}$ correspondant à l'EDO. Comme au **Chapitre 2**, nous considérerons la matrice $\hat{A}_{EDP}$ définie comme

$$\hat{A}_{EDP} := \left(\begin{array}{c|c} \mathbf{0}_K & \mathbf{I}_K \\ \hline \hat{A}_1 & \hat{A}_2 \end{array}\right),$$

afin d'étudier le spectre correspondant à l'identification de l'équation des ondes. La figure ci-dessous présente les résultats pour $K = 5, 20, 50$ et 100. En haut nous présenterons le terme source g du modèle(4.11) (ligne continue bleue) ainsi que sa valeur prédite par le modèle réduit aux points sélectionnés (points rouges), tandis qu'en bas nous présenterons les valeurs propres de la matrice $\hat{A}_{EDP}$ (en rouge), et de la matrice $\hat{C}$ (en noire) dans le plan complexe.



(a) $K = 5$ (b) $K = 20$

(c) $K = 50$ (d) $K = 100$

FIGURE 4.12 Identification du terme source et spectre de la matrice $\hat{A}_{EDP}$

On remarque, que le caractère discontinu du terme source semble être la raison de la difficulté d'identification. En effet, un nombre important de points doivent être sélectionnés afin de reproduire correctement la forme de g.
Ainsi pour $K = 5$ et $K = 20$, le spectre des matrices $\hat{A}$ et $\hat{C}$ semble avoir été correctement appris, le terme source a quant à lui mal été appris. Lorsque $K = 50$, il semble que la dimension élevée du modèle réduit nuise à la bonne identification de la dynamique tout en étant trop faible pour identifier correctement le terme source. Ainsi les valeurs propres de la matrice $\hat{A}_{EDP}$ ne sont pas entièrement inscrites sur le cercle unité, certaines sont strictement à l'intérieur, le modèle appris est donc diffusif. Cela explique pourquoi la prédiction à la figure 4.10 sous-estime de plus en plus l'onde. Enfin, lorsque $K = 100$ cette fois-ci le terme source est parfaitement identifié.
Ces résultats correspondent à ceux présentés aux figures 4.10 et 4.11. Une valeur propre

de la matrice $\hat{A}_{EDP}$ vaut zéro, cela correspond à la condition aux limites $u(0,t) = 0$ pour tout $t \in [0,T]$.

Conclusion

L'identification par échantillonnage des données a été présentée dans ce chapitre. Cela permet d'éviter une reconstruction coûteuse de la solution lorsque seule une évaluation ponctuelle ou restreinte à un sous-domaine est désirée. L'échantillonnage joue un rôle important, d'autant plus sur des modèles dont la dynamique n'évolue pas de la même façon sur tout le domaine. Ainsi un algorithme, basé sur la méthode EIM a été proposé. Cet algorithme permet, par ailleurs, de reconstruire la solution en s'assurant que son évaluation aux points sélectionnés soit conservée. Par ailleurs, deux algorithmes de la communauté CSSP, l'algorithme des k-Médoides, et un échantillonnage uniforme ont été considérés afin de comparer les performances. La méthodologie introduite dans ce chapitre a été confrontée à un problème de diffusion mais également un problème d'onde, avec des conditions aux limites dynamiques. Les résultats ont montré l'importance du choix des points sélectionnés, ainsi que le bien-fondé de l'utilisation de la méthode EIM.

La réduction des données par échantillonnage peut être facilement étendu au cas de solution dépendant aussi d'un paramètre. Ce qui n'est pas le cas d'une réduction POD, car la matrice des données $\mathbf{U}$ est alors un tenseur d'ordre 3. Le chapitre suivant présentera ainsi la réduction de tenseurs, ainsi que l'identification d'une EDP d'évolution paramétrée.

Chapitre 5

Réduction de tenseurs et identification d'une EDP d'évolution paramétrée à partir de données

Résumé Ce chapitre est consacré à l'identification d'une EDP d'évolution paramétrée. Ce type de modèle a de nombreuses applications notamment en optimisation de formes ou en quantification d'incertitudes. Cependant, dès lors que la solution de l'EDP dépend également d'un paramètre, la matrice des données est un tenseur d'ordre 3. Ce qui rend sa réduction difficile. Ainsi on proposera une méthode de réduction combinant la méthode *High Order Singular Value Decomposition* (HOSVD) et la méthode EIM. Par ailleurs, on proposera également une nouvelle méthode de réduction de tenseurs dont la représentation basse dimension correspond à la solution aux points et aux pas de temps sélectionnés. Une fois la matrice des données réduites, la variété solution sera apprise par l'algorithme *Kernel Ridge Regression* (KRR). Finalement, l'apprentissage d'un modèle réduit se fera par la méthode DMD. On présentera l'apprentissage d'un modèle réduit dans le cas d'un couplage paramétré : équation de la chaleur-Oscillateur de Van der Pol, ainsi que l'optimisation de formes d'un canal traversé par une onde.

Mots-clés. Data-driven model ; Parametrized PDEs ; Nonlinear regression ; KRR ; HOSVD ; Tensor Low-rank approximation ; Machine Learning.

Introduction

Jusqu'à présent, nous avons considéré des solutions u ne dépendant que de l'espace au travers de la variable $\boldsymbol{x}$, et du temps au travers de la variable t. La géométrie du problème, ou bien les paramètres physiques tels que la conductivité thermique, étaient fixés. Ainsi si l'on désirait prédire la solution u pour une autre géométrie ou bien pour des paramètres physiques différents, le modèle réduit ne serait alors plus adapté. L'idée est alors d'apprendre la solution u pour n'importe quel paramètre, que l'on notera $\boldsymbol{\mu}$, à partir de l'évaluation de u en plusieurs paramètres $\boldsymbol{\mu}_m$. Ce qui a donné lieu à l'introduction des méthodes des bases réduites ([86], [145], [121]). On génère alors dans une phase dite *OFFLINE*, la solution u pour différents paramètres $\boldsymbol{\mu}_m$, puis dans une phase dite *ONLINE* on calcul la solution u pour n'importe quel paramètre $\boldsymbol{\mu}$ par une méthode de Galerkin, en utilisant le sous-espace vectoriel engendré par les $u(\cdot, \cdot; \boldsymbol{\mu}_m)$. Théoriquement, il a été démontré que représenter avec précision la solution $u(\cdot, \cdot; \boldsymbol{\mu})$ à partir de solutions $u(\cdot, \cdot; \boldsymbol{\mu}_m)$ était possible dans de nombreux cas ([42]). Cela repose sur la petite épaisseur de Kolmogorov de la variété solution définie comme étant

$$\{u(\cdot, \cdot; \boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathcal{D}\},$$

avec $\mathcal{D}$ l'ensemble des paramètres admissibles. Grossièrement, cela signifie que bien souvent la variété solution n'occupe pas tout l'espace mais seulement une petite partie.

Dans notre cas, une méthode de Galerkin n'est pas envisageable car l'on ne connait pas l'EDP dont sont issues les données. On est alors confronté à un problème de régression non-linéaire, où l'on cherche à apprendre l'application qui à tout $\boldsymbol{\mu}$ associe la fonction $u(\cdot, \cdot; \boldsymbol{\mu})$. Comme nous travaillons avec des données simulées, nous sommes en réalité confronté au problème discret d'apprendre la matrice $\mathbf{U}(\boldsymbol{\mu})$ à partir des matrices $\mathbf{U}(\boldsymbol{\mu}_m)$. De nombreux auteurs se sont penchés sur ce problème en proposant l'utilisation d'algorithmes de régression non linéaires issus de la communauté du Machine Learning ([9], [149], [67], [172]). Une des difficultés est la grande dimension de chaque matrice $\mathbf{U}(\boldsymbol{\mu}_m)$. Afin de contourner ce problème, on calcul l'approximation faible rang de chaque matrice

$$\mathbf{U}(\boldsymbol{\mu}_m) \approx \boldsymbol{\Phi}_r \, \mathbf{A}(\boldsymbol{\mu}_m).$$

Cette approximation peut être réalisé par les méthodes SVD, POD, PCA, etc. Cependant, rien ne nous assure que la base stockée dans la matrice $\boldsymbol{\Phi}_r$ soit bien adaptée pour un autre paramètre $\boldsymbol{\mu}_p$. La construction d'une telle base adaptée à n'importe quel

paramètre $\boldsymbol{\mu}_m$ n'est pas trivial, car nous travaillons avec un tenseur d'ordre 3 (espace, temps, paramètre) ce qui rend toute approche algébrique impossible.

En statistiques, Tucker & al ont introduit la méthode *High Order Singular Value Decomposition* (HOSVD, [177]) aussi appelée *Multilinear Singular Value Decomposition* (m-mode SVD, [48]). Appliquée à notre problème, cette méthode consiste à empiler les matrices $\mathbf{U}(\boldsymbol{\mu}_m)$ et ainsi décomposer en valeurs singulières la matrice :

$$\mathbf{U}^{\mathbb{S}} := \begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) & \cdots & \mathbf{U}(\boldsymbol{\mu}_M) \end{bmatrix} \approx \boldsymbol{\Phi}_r \begin{bmatrix} \mathbf{A}(\boldsymbol{\mu}_1) & \cdots & \mathbf{A}(\boldsymbol{\mu}_M) \end{bmatrix}.$$

Une autre méthode plus récente est la méthode *Algebraic Proper Generalized Decomposition* (Algerbraic PGD, [54]). Elle consiste à chercher de manière itérative une approximation sous forme tensorisée.

Une des applications à la sélection de sous-ensemble de colonnes d'une matrice (CSSP), est la décomposition CUR d'une matrice ([120],[23]). Cette décomposition consiste à sélectionner un sous-ensemble de colonnes d'une matrice $\mathbf{A}$, que l'on stocke dans une matrice $\mathbf{C}$ puis à sélectionner un sous-ensemble de lignes que l'on stockera dans une matrice $\mathbf{R}$. Enfin on cherchera alors une matrice $\mathbf{U}$ de façon à obtenir la meilleure approximation

$$\mathbf{A} \approx \mathbf{C}\,\mathbf{U}\,\mathbf{R}.$$

Cette décomposition permet une grande économie en termes de stockage, tout en gardant une facile interprétation des matrices $\mathbf{C}$ et $\mathbf{R}$. Cette décomposition a été étendue au cas de tenseurs d'ordre 3, sous le nom de *décomposition CURT* ([162]). On propose dans ce travail, une méthode de réduction de tenseurs basée sur une décomposition HOSDV et l'algorithme EIM. On présentera également dans ce chapitre, une méthode de réduction de tenseurs dont la représentation basse dimension est interprétable.

Dans la première section, nous présenterons les deux méthodes de réduction, ainsi que leurs extensions au cas vectoriel. Dans la deuxième section, nous présenterons l'apprentissage de la variété solution. Enfin, l'apprentissage d'un modèle réduit paramétré sera introduit à la section 3. On présentera à la quatrième section, l'apprentissage d'un couplage paramétré : équation de la chaleur-Oscillateur de Van der Pol. Enfin, on utilisera la réduction interprétable afin d'optimiser la forme d'un canal traversé par une onde.

5.1 Réduction de solutions spatio-temporelles paramétrées

Considérons l'EDP d'évolution paramétrée suivante :

$$\partial_t u(\boldsymbol{x}, t; \boldsymbol{\mu}) + \mathcal{L}(\boldsymbol{\mu})\, u(\boldsymbol{x}, t; \boldsymbol{\mu}) = 0 \tag{5.1}$$

avec comme toujours $\mathcal{L}(\boldsymbol{\mu})$ un opérateur linéaire.
Dans ce chapitre le solveur HF, nous fournit M matrices que l'on notera $\mathbf{U}(\boldsymbol{\mu}_m)$, solution de l'équation (5.1), pour M paramètres $\boldsymbol{\mu}_m$. Nous souhaitons dans cette section, réduire la dimension de ces matrices. Dans un premier temps, nous chercherons à approcher $\mathbf{U}(\boldsymbol{\mu}_m)$ par :

$$\mathbf{U}(\boldsymbol{\mu}_m) \approx \mathbf{Q}\, \tilde{\mathbf{W}}(\boldsymbol{\mu}_m)\, \boldsymbol{\Psi}_r^T, \qquad \forall 1 \leq m \leq M,$$

avec $\mathbf{Q} \in \mathbb{R}^{N_x \times K}$, $\boldsymbol{\Psi}_r \in \mathbb{R}^{N_t \times K'}$, et $\tilde{\mathbf{W}}(\boldsymbol{\mu}_m)$ la représentation basse dimension de $\mathbf{U}(\boldsymbol{\mu}_m)$.
Dans un second temps, on cherchera l'approximation,

$$\mathbf{U}(\boldsymbol{\mu}_m) \approx \mathbf{Q}\, \tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}_m)\, \mathbf{R}, \qquad \forall 1 \leq m \leq M,$$

avec $\mathbf{Q} \in \mathbb{R}^{N_x \times K}$, $\mathbf{R} \in \mathbb{R}^{K' \times N_t}$, et $\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}_m) := \mathbf{U}_{\sigma_x, \sigma_t}(\boldsymbol{\mu}_m)$ où σ_x (resp σ_t) l'application permettant de sélectionner les lignes (resp les colonnes).
On appellera *approximation de tenseur par EIM* la première approche, et *approximation QUR* la seconde.

Remarque 21. *La matrice* $\mathbf{Q}$ *est commune aux deux approximations, et sera obtenue de manière analogue à celle construite par* **l'algorithme 3** *au* **Chapitre 4**.

5.1.1 Approximation de tenseur par EIM

L'idée étant ici d'adapter l'algorithme *approximation faible rang par EIM* lorsque l'on considère plusieurs matrices $\mathbf{U}(\boldsymbol{\mu}_m)$. Pour ce faire, considérons la matrice empilée :

$$\mathbf{U}^{\mathbb{S}} := \begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}(\boldsymbol{\mu}_M) \end{bmatrix} \in \mathbb{R}^{MN_x \times N_t},$$

et effectuons la décomposition en valeurs propres suivante :

$$\left(\mathbf{U}^{\mathbb{S}}\right)^T \mathbf{U}^{\mathbb{S}} = \boldsymbol{\Psi}\,\boldsymbol{\Lambda}\,\boldsymbol{\Psi}^T,$$

Enfin tronquons nos modes temporels à l'ordre K' comme suit,

$$\boldsymbol{\Psi}_r = \boldsymbol{\Psi}_{\cdot,1:K'}.$$

En considérant la matrice empilée $\mathbf{U}^{\mathbb{S}}$, on a ainsi construit K' modes temporels adaptés à tout paramètre $\boldsymbol{\mu}_m$. Par définition de la projection orthogonale, les représentations basses dimensions sont données par :

$$\begin{bmatrix} \mathbf{W}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{W}(\boldsymbol{\mu}_M) \end{bmatrix} := \begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}(\boldsymbol{\mu}_M) \end{bmatrix} \boldsymbol{\Psi}_r$$

Ainsi nous ne travaillerons, non plus avec les matrices $\mathbf{U}(\boldsymbol{\mu}_m)$ mais avec leurs représentations basses dimensions $\mathbf{W}(\boldsymbol{\mu}_m)$. Il suffit maintenant de sélectionner K lignes communes aux matrices $\left(\mathbf{W}(\boldsymbol{\mu}_m)\right)_{1\leq m\leq M}$, puis de construire un opérateur d'interpolation $\mathbf{Q}$ de sorte que $\mathbf{Q}\,\tilde{\mathbf{W}}(\boldsymbol{\mu}_m)$ soit une bonne approximation de $\mathbf{W}(\boldsymbol{\mu}_m)$, avec $\tilde{\mathbf{W}}(\boldsymbol{\mu}_m) := \mathbf{W}_{\sigma,\cdot}(\boldsymbol{\mu}_m)$, pour tout m. On procédera comme dans **l'algorithme 3**.

L'algorithme *approximation de tenseur par EIM* est résumé ci-dessous.

Algorithme 4 : Approximation de tenseur par EIM

Entrée : M matrices $\mathbf{U}(\boldsymbol{\mu}_m)$ de taille $N_x \times N_t$, un nombre de points K, et un rang de réduction en temps K'.

Sortie : L'ensemble des indices de lignes sélectionnées σ , ainsi que l'approximation $\mathbf{U}(\boldsymbol{\mu}_m) \approx \mathbf{Q}\,\tilde{\mathbf{W}}(\boldsymbol{\mu}_m)\,\boldsymbol{\Psi}_r^T$ pour tout m.

Empilement des données ;

$$\mathbf{U}^{\mathbb{S}} \leftarrow \begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}(\boldsymbol{\mu}_M) \end{bmatrix} ;$$

Réduction des données ;

$$\left(\mathbf{U}^{\mathbb{S}}\right)^T \mathbf{U}^{\mathbb{S}} \leftarrow \boldsymbol{\Psi}\Lambda\boldsymbol{\Psi}^T, \qquad \boldsymbol{\Psi}_r \leftarrow \boldsymbol{\Psi}_{:,1:K'}, \qquad \begin{bmatrix} \mathbf{W}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{W}(\boldsymbol{\mu}_M) \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}(\boldsymbol{\mu}_M) \end{bmatrix} \boldsymbol{\Psi}_r.$$

Sélections des lignes;

$j_1 \leftarrow \arg\max_j \max_m \|\mathbf{W}_{\cdot,j}(\boldsymbol{\mu}_m)\|_{\ell^\infty}, \quad m_1 \leftarrow \arg\max_m \|\mathbf{W}_{\cdot,j_1}(\boldsymbol{\mu}_m)\|_{\ell^\infty},$

$i_1 \leftarrow \arg\max_i |\mathbf{W}_{i,j_1}(\boldsymbol{\mu}_{m_1})|$

$\mathbf{Q}^1_{i,1} \leftarrow \frac{\mathbf{W}_{i,j_1}(\boldsymbol{\mu}_{m_1})}{\mathbf{W}_{i_1,j_1}(\boldsymbol{\mu}_{m_1})}, \qquad \sigma \leftarrow \{i_1\}$;

while $k \leq K$ **do**

$j_k \leftarrow \arg\max_j \max_m \|\mathbf{W}_{\cdot,j}(\boldsymbol{\mu}_m) - \mathbf{Q}^{k-1}\mathbf{W}_{\sigma,j}(\boldsymbol{\mu}_m)\|_{\ell^\infty},$

$m_k \leftarrow \arg\max_m \|\mathbf{W}_{\cdot,j_k}(\boldsymbol{\mu}_m) - \mathbf{Q}^{k-1}\mathbf{W}_{\sigma,j_k}(\boldsymbol{\mu}_m)\|_{\ell^\infty},$

$i_k \leftarrow \arg\max_i |\mathbf{W}_{i,j_k}(\boldsymbol{\mu}_{m_k}) - \mathbf{Q}^{k-1}_{i,\cdot}\mathbf{W}_{\sigma,j_k}(\boldsymbol{\mu}_{m_k})|$

$\mathbf{Q}^k_{i,k} \leftarrow \frac{\mathbf{W}_{i,j_k}(\boldsymbol{\mu}_{m_k}) - \mathbf{Q}^{k-1}\mathbf{W}_{\sigma,j_k}(\boldsymbol{\mu}_{m_k})}{\mathbf{W}_{i_k,j_k}(\boldsymbol{\mu}_{m_k}) - \mathbf{Q}^{k-1}_{i_k,\cdot}\mathbf{W}_{\sigma,j_k}(\boldsymbol{\mu}_{m_k})}, \qquad \sigma \leftarrow \{\sigma, i_k\}$;

for $l < k$ **do**

$\mathbf{Q}^k_{\cdot,l} \leftarrow \mathbf{Q}^{k-1}_{\cdot,l} - \mathbf{Q}^{k-1}_{i_k,l}\mathbf{Q}^k_{\cdot,k}$

$\mathbf{Q} \leftarrow \mathbf{Q}^K, \qquad \tilde{\mathbf{W}}(\boldsymbol{\mu}_m) \leftarrow \mathbf{W}_{\sigma,:}(\boldsymbol{\mu}_m).$

Lorsque la solution u de l'équation (5.1) est vectorielle, quelques modifications sont nécessaires. Nous les présenterons au paragraphe ci-dessous.

5.1.2 Extension de l'approximation de tenseur par EIM au cas vectoriel

Lorsque la solution de l'équation (5.1) n'est plus scalaire mais vectorielle chaque $\mathbf{U}(\boldsymbol{\mu}_m)$ est un tenseur d'ordre trois. Ainsi quelques modifications s'imposent, la matrice empilée sera de la forme :

$$\mathbf{U}^{\mathbb{S}} := \begin{bmatrix} \mathbf{U}_{\cdot,\cdot,1}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}_{\cdot,\cdot,N_c}(\boldsymbol{\mu}_1) \\ \mathbf{U}_{\cdot,\cdot,1}(\boldsymbol{\mu}_2) \\ \vdots \\ \vdots \\ \mathbf{U}_{\cdot,\cdot,1}(\boldsymbol{\mu}_M) \\ \vdots \\ \mathbf{U}_{\cdot,\cdot,N_c}(\boldsymbol{\mu}_M) \end{bmatrix} \in \mathbb{R}^{N_x M N_c \times N_t}$$

avec N_c le nombre de composantes du vecteur u. Une fois la base temporelle $\boldsymbol{\Psi}_r$ construite, les représentations basses dimensions s'obtiennent comme précédemment par :

$$\begin{bmatrix} \mathbf{W}_{\cdot,\cdot,1}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{W}_{\cdot,\cdot,N_c}(\boldsymbol{\mu}_1) \\ \mathbf{W}_{\cdot,\cdot,1}(\boldsymbol{\mu}_2) \\ \vdots \\ \vdots \\ \mathbf{W}_{\cdot,\cdot,1}(\boldsymbol{\mu}_M) \\ \vdots \\ \mathbf{W}_{\cdot,\cdot,N_c}(\boldsymbol{\mu}_M) \end{bmatrix} := \begin{bmatrix} \mathbf{U}_{\cdot,\cdot,1}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}_{\cdot,\cdot,N_c}(\boldsymbol{\mu}_1) \\ \mathbf{U}_{\cdot,\cdot,1}(\boldsymbol{\mu}_2) \\ \vdots \\ \vdots \\ \mathbf{U}_{\cdot,\cdot,1}(\boldsymbol{\mu}_M) \\ \vdots \\ \mathbf{U}_{\cdot,\cdot,N_c}(\boldsymbol{\mu}_M) \end{bmatrix} \boldsymbol{\Psi}_r.$$

Enfin, nous procéderons différemment pour la sélection des lignes. Ci-dessous, présentons la phase d'initialisation, la récurrence pouvant en être facilement déduite.

$$j_1 \leftarrow \arg\max_j \max_m \max_c \|\mathbf{W}_{\cdot,j,c}(\boldsymbol{\mu}_m)\|_{\ell^\infty}, \quad m_1 \leftarrow \arg\max_m \max_c \|\mathbf{W}_{\cdot,j_1,c}(\boldsymbol{\mu}_m)\|_{\ell^\infty},$$
$$c_1 \leftarrow \arg\max_c \|\mathbf{W}_{\cdot,j_1,c}(\boldsymbol{\mu}_{m_1})\|_{\ell^\infty}, \quad i_1 \leftarrow \arg\max_i |\mathbf{W}_{i,j_1,c_1}(\boldsymbol{\mu}_{m_1})|$$
$$\mathbf{Q}^1_{i,1} \leftarrow \frac{\mathbf{W}_{i,j_1,c_1}(\boldsymbol{\mu}_{m_1})}{\mathbf{W}_{i_1,j_1,c_1}(\boldsymbol{\mu}_{m_1})}, \qquad \sigma \leftarrow \{i_1\}.$$

5.1.3 Approximation QUR

Un inconvénient de l'algorithme *approximation de tenseur par EIM* présenté ci-dessus, est la difficulté d'interprétation des représentations basses dimensions. Si la réduction en espace se fait en sélectionnant des lignes, ce n'est pas le cas pour la réduction en temps. Dans cette section, on souhaite alors construire des représentations basses dimensions interprétables.

Pour ce faire, on sélectionnera pour chaque matrice $\mathbf{U}(\boldsymbol{\mu}_M)$, K lignes et K' colonnes identiques à l'aide de deux applications σ_x et σ_t. La représentation basse dimension de chaque matrice sera alors la matrice restreinte à ses indices par σ_x et σ_t. Enfin, on construira deux opérateurs d'interpolations $\mathbf{Q}$ et $\mathbf{R}$ permettant de reconstruire les matrices $\mathbf{U}(\boldsymbol{\mu}_M)$.

En pratique, on utilisera l'algorithme *approximation de tenseur par EIM* afin de générer l'opérateur d'interpolation $\mathbf{Q}$ ainsi que l'application σ_x. Puis, nous sélectionnerons les colonnes en réappliquant l'algorithme *approximation de tenseur par EIM*, sans effectuer de réduction, à la matrice empilée :

$$\begin{bmatrix} \mathbf{U}^T_{\sigma_x,\cdot}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}^T_{\sigma_x,\cdot}(\boldsymbol{\mu}_M) \end{bmatrix}$$

Cette fois-ci l'algorithme nous fournit, un opérateur d'interpolation $\mathbf{Q}'$ et une application σ_t. Finalement en posant,

$$\mathbf{R} := \left(\mathbf{Q}'\right)^T \qquad \text{et} \qquad \tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}_m) := \mathbf{U}_{\sigma_x,\sigma_t}(\boldsymbol{\mu}_m), \quad \forall 1 \leq m \leq M,$$

on retrouve *l'approximation QUR*.

5.2 Apprentissage de la variété solution

Nous faisons le choix, ici, de présenter cette section en utilisant *l'approximation de tenseur par EIM*. Notons qu'en utilisant *l'approximation QUR*, on procéderait de manière similaire.

Nous disposons à présent d'une représentation basse dimension $\tilde{\mathbf{W}}(\boldsymbol{\mu}_m)$ pour chaque matrice de données $\mathbf{U}(\boldsymbol{\mu}_m)$.
Le but de cette section est alors de prédire la matrice $\tilde{\mathbf{W}}(\boldsymbol{\mu})$ pour tout paramètre $\boldsymbol{\mu} \in \mathcal{D}$, avec $\mathcal{D}$ l'ensemble des paramètres admissibles. Pour cela nous disposons de l'ensemble d'entrainement :

$$\Big\{\big(\boldsymbol{\mu}_1, \tilde{\mathbf{W}}(\boldsymbol{\mu}_1)\big), \big(\boldsymbol{\mu}_2, \tilde{\mathbf{W}}(\boldsymbol{\mu}_2)\big), \cdots, \big(\boldsymbol{\mu}_M, \tilde{\mathbf{W}}(\boldsymbol{\mu}_M)\big)\Big\}.$$

Les deux principales difficultés pour prédire $\tilde{\mathbf{W}}(\boldsymbol{\mu})$ sont :

- l'éventuel relation non-linéaire entre $\boldsymbol{\mu}$ et $\tilde{\mathbf{W}}(\boldsymbol{\mu})$.
- le peu de données d'entraînement, M étant souvent petit.

Une régression linéaire ne sera pas adaptée à ce problème. On utilisera alors des algorithmes plus complexes tels que : les *Forêts aléatoires* (RF, [112]), les *Réseaux de neurones* (ANN) ou encore l'algorithme *Kernel Ridge Regression* (KRR, [180]). De plus, le peu de données d'entrainement, restreint considérablement le choix des algorithmes. On cherche ainsi un algorithme assez complexe, mais avec peu de paramètres à optimiser. L'algorithme *Kernel Ridge Regression* semble être le mieux adapté pour ce genre de problème.

5.3 Apprentissage d'un modèle réduit paramétré

Le but de cette section est d'apprendre un modèle réduit fidèle au modèle (5.1). Nous aurons une phase dite *OFFLINE* où l'on s'autorisera un coût de calcul important. Et une phase dite *ONLINE* où l'on souhaitera réduire autant que possible le coût de calcul afin d'obtenir une prédiction instantanée.

5.3.1 Phase *OFFLINE* : Réduction des données et entraînement de l'algorithme de régression

Dans cette phase, nous disposons d'un panel de paramètres $\mathcal{D}_h := \{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_M\}$ ainsi que d'un panel de solutions $\mathbf{U}(\boldsymbol{\mu}_m)$, que l'on notera $\mathcal{M}_h := \{\mathbf{U}(\boldsymbol{\mu}_1), \cdots, \mathbf{U}(\boldsymbol{\mu}_M)\}$. À l'aide de **l'algorithme 4**, nous construisons la représentation basse dimension de

chaque élément de $\mathcal{M}_h$, ainsi que les matrices $\mathbf{Q}$ et $\mathbf{\Psi}_r$ permettant de reconstruire les solutions, i.e.

$$\mathbf{U}(\boldsymbol{\mu}_m) \approx \mathbf{Q}\,\tilde{\mathbf{W}}(\boldsymbol{\mu}_m)\,\mathbf{\Psi}_r^T, \quad \forall \boldsymbol{\mu}_m \in \mathcal{D}_h.$$

Enfin, nous entrainons l'algorithme KRR avec $\mathcal{D}_h$ comme ensemble d'entrée d'entrainement et $\{\tilde{\mathbf{W}}(\boldsymbol{\mu}_1), \cdots, \tilde{\mathbf{W}}(\boldsymbol{\mu}_M)\}$ comme ensemble de sortie d'entrainement.

5.3.2 Phase *ONLINE* : Apprentissage d'un modèle réduit pour un paramètre donné

À présent nous souhaitons apprendre un modèle réduit, pour un paramètre $\boldsymbol{\mu}$ donné. Cette phase se décompose en deux étapes :

- la première consiste à prédire la représentation basse dimension de la matrice de données $\mathbf{U}(\boldsymbol{\mu})$
- tandis que la seconde consiste à apprendre un modèle réduit de la même manière qu'au **Chapitre 4**.

À l'aide de notre algorithme KRR entrainé, nous prédisons la matrice $\hat{\tilde{\mathbf{W}}}(\boldsymbol{\mu})$. Afin de construire les ensembles d'entrainements pour l'apprentissage de notre modèle réduit, nous reconstruisons la partie temporelle à l'aide de la matrice $\mathbf{\Psi}_r$. On obtient alors :

$$\hat{\mathbf{U}}_{\sigma,\cdot}(\boldsymbol{\mu}) := \hat{\tilde{\mathbf{W}}}(\boldsymbol{\mu})\,\mathbf{\Psi}_r^T.$$

Les ensembles d'entrainements seront alors définis par :

$$X(\boldsymbol{\mu}) = \begin{bmatrix} | & | & & | \\ \hat{\mathbf{u}}_\sigma^1(\boldsymbol{\mu}) & \hat{\mathbf{u}}_\sigma^2(\boldsymbol{\mu}) & \cdots & \hat{\mathbf{u}}_\sigma^{N_t-1}(\boldsymbol{\mu}) \\ | & | & & | \end{bmatrix} \quad \text{et} \quad Y(\boldsymbol{\mu}) = \begin{bmatrix} | & | & & | \\ \hat{\mathbf{u}}_\sigma^2(\boldsymbol{\mu}) & \hat{\mathbf{u}}_\sigma^3(\boldsymbol{\mu}) & \cdots & \hat{\mathbf{u}}_\sigma^{N_t}(\boldsymbol{\mu}) \\ | & | & & | \end{bmatrix} \tag{5.2}$$

et en posant $A(\boldsymbol{\mu}) := Y(\boldsymbol{\mu})X^\dagger(\boldsymbol{\mu})$, on retrouve un modèle réduit de la forme

$$\hat{\mathbf{u}}_\sigma^{n+1}(\boldsymbol{\mu}) = A(\boldsymbol{\mu})\,\hat{\mathbf{u}}_\sigma^n(\boldsymbol{\mu}), \quad \text{et } \hat{\mathbf{u}}^{n+1}(\boldsymbol{\mu}) = \mathbf{Q}\,\hat{\mathbf{u}}_\sigma^{n+1}(\boldsymbol{\mu}),$$

fidèle au modèle (5.1).

Remarque 22. *On a donc appris un modèle réduit interprétable, comme au* **Chapitre 4**. *En ce qui concerne le coût de calcul pour la phase ONLINE, la prédiction par KRR est peu coûteuse et ne dépend que de M, K et K' . Les opérations coûteuses*

sont le calcul de la matrice $\hat{\mathbf{U}}_{\sigma,\cdot}(\boldsymbol{\mu})$ *et la prédiction de la matrice* $A(\boldsymbol{\mu})$*, cependant ces opérations restent tout de même linéaires par rapport à* N_t*.*

5.4 Applications numériques

Nous présenterons tout d'abord l'apprentissage d'un modèle réduit dans le cas d'un couplage paramétré équation de la chaleur-Oscillateur de Van der Pol, en utilisant l'approximation de tenseur par EIM. L'*approximation QUR* sera ensuite utilisée afin d'optimiser la forme d'un canal traversé par une onde. Notons que dans ce deuxième exemple, nous ne chercherons pas à identifier le modèle EDP.

5.4.1 Couplage paramétré : Équation de la chaleur-Oscillateur de Van der Pol

5.4.1.1 Présentation du modèle et génération des données

On considère ici l'équation de la chaleur dont la condition aux limites, de type Neumann, est dynamique. De plus, cette équation comporte un terme source dynamique. L'évolution en temps de la condition aux limites et du terme source est guidée par une EDO non linéaire, à savoir l'oscillateur de Van der Pol. Ce couplage sera par ailleurs dépendant d'un paramètre $\mu \in \mathcal{D}$, qui interviendra au niveau de la condition aux limites.

Le modèle que l'on souhaite identifier est définie par :

$$\begin{cases} \partial_t u(\boldsymbol{x}, t; \mu) - \nabla \cdot \Big(\kappa(\boldsymbol{x}) u(\boldsymbol{x}, t; \mu)\Big) = g(\boldsymbol{x})\theta(t), & \text{dans } \Omega \times (0, T) \\ \kappa(\boldsymbol{x})\partial_{\boldsymbol{n}} u(\boldsymbol{x}, t; \mu) = -\phi_{out}(\mu)\, \dot{\theta}(t), & \text{sur } \partial\Omega \times (0, T) \\ u(\boldsymbol{x}, 0; \mu) = u_0(\boldsymbol{x}), & \text{dans } \Omega \\ \ddot{\theta}(t) = \epsilon\, \omega\Big(1 - \theta(t)^2\Big)\, \dot{\theta}(t) + \omega\theta(t), & \forall t \in (0, T) \end{cases} \tag{5.3}$$

Ce système dépend du paramètre μ à travers le flux externe ϕ_{out}, modélisé par la `sigmoide` suivante :

$$\phi_{out}(\mu) = \frac{10}{1 + e^{-\lambda\mu}},$$

Le terme source g est défini par :

$$g(\boldsymbol{x}) = 3 \sin\Big(16\pi(1 - \boldsymbol{x}^1)\boldsymbol{x}^1(1 - \boldsymbol{x}^2)\boldsymbol{x}^2\Big), \tag{5.4}$$

où l'on note $\boldsymbol{x} = \left(\boldsymbol{x}^1, \boldsymbol{x}^2\right)$.
La conductivité thermique κ vaudra κ_D sur tout le domaine. Enfin la condition initiale sera définie par une gaussienne centrée au point $\boldsymbol{x}_C$.
L'ensemble des paramètres admissibles $\mathcal{D}$, sera discrétisé en sélectionnant aléatoirement M éléments de cet espace. Et on notera $\mathcal{D}_h$ l'ensemble des paramètres sélectionnés. On utilisera le logiciel FreeFem++ afin de générer les solutions $u(\cdot, \cdot; \mu)$, pour tout $\mu \in \mathcal{D}_h$, et la routine `ode45` du logiciel MATLAB pour la solution θ. Les solutions $u(\cdot, \cdot; \mu)$ seront stockées dans M matrices $\mathbf{U}(\boldsymbol{\mu})$, et on notera $\mathcal{M}_h$ la variété solution discrète, i.e.

$$\mathcal{M}_h := \{\mathbf{U}(\mu), \quad \forall \mu \in \mathcal{D}_h\}.$$

L'EDO ne dépendant pas du paramètre, on stockera θ et sa dérivée $\dot{\theta}$ dans une seule matrice $\boldsymbol{\Theta}$. Le tableau ci-dessous résume les paramètres physiques et numériques utilisés pour la génération des données.

Paramètres	**Valeurs**
Conductivité thermique (κ_D)	1
Hyperparamètre de la `sigmoid` (λ)	6
Pulsation (ω)	2π
Coefficient de non-linéarité (ϵ)	1.5
Centre de la gaussienne ($\boldsymbol{x}_C$)	$(0.5, 0.5)$
Domaine (Ω)	$[0; 1]^2$
Temps final (T)	6 sec
Ensemble des paramètres admissibles ($\mathcal{D}$)	$[-1; 1]$
Pas de temps (δt)	5×10^{-3}
Pas d'espace (δx)	3.5×10^{-2}
Nombre de paramètres choisis (M)	100

Table 5.1 Paramètres physiques et numériques utilisés

La figure ci-dessous représente l'évolution de la solution θ ainsi que de sa dérivée $\dot{\theta}$ au cours du temps, générées par la routine `ode45`.

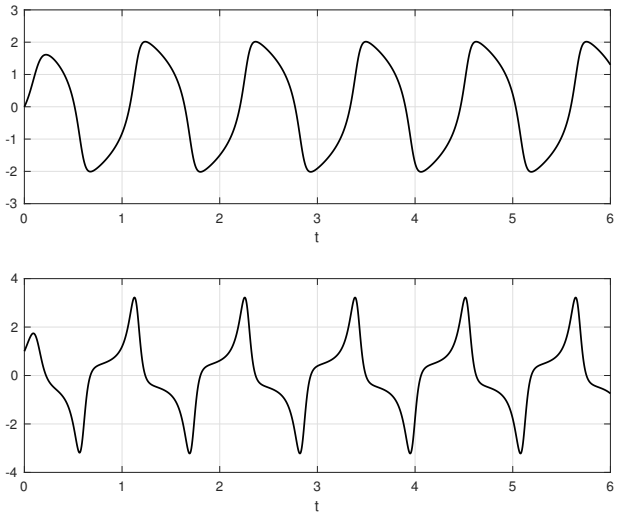
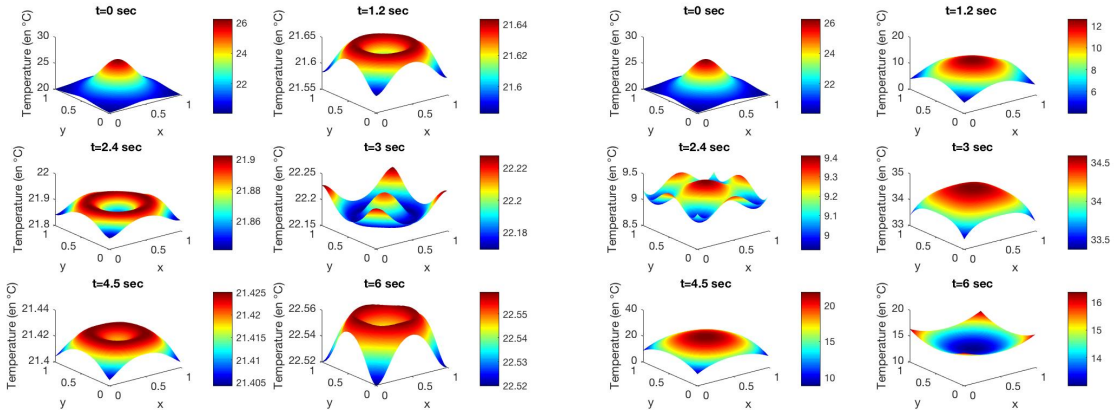


FIGURE 5.1 (Haut) Évolution de θ en fonction du temps. (Bas) Évolution de la dérivée $\dot{\theta}$ au cours du temps

On peut ainsi observer le caractère non linéaire de cette EDO. La figure ci-dessous représente quant à elle l'évolution au cours du temps de la solution u pour différents paramètres μ.



(a) $\mu_{33} = -0.9690$ (b) $\mu_{59} = 0.9982$

FIGURE 5.2 Évolution au cours du temps de la solution u pour différents paramètres μ

La solution u semble avoir un comportement différent en fonction du paramètre μ choisi. Cela est appuyé par la figure ci-dessous qui représente l'évolution au cours du temps de u au point $\boldsymbol{x} = (0.9, 0.9)$, pour tous les paramètres $\mu \in \mathcal{D}_h$.

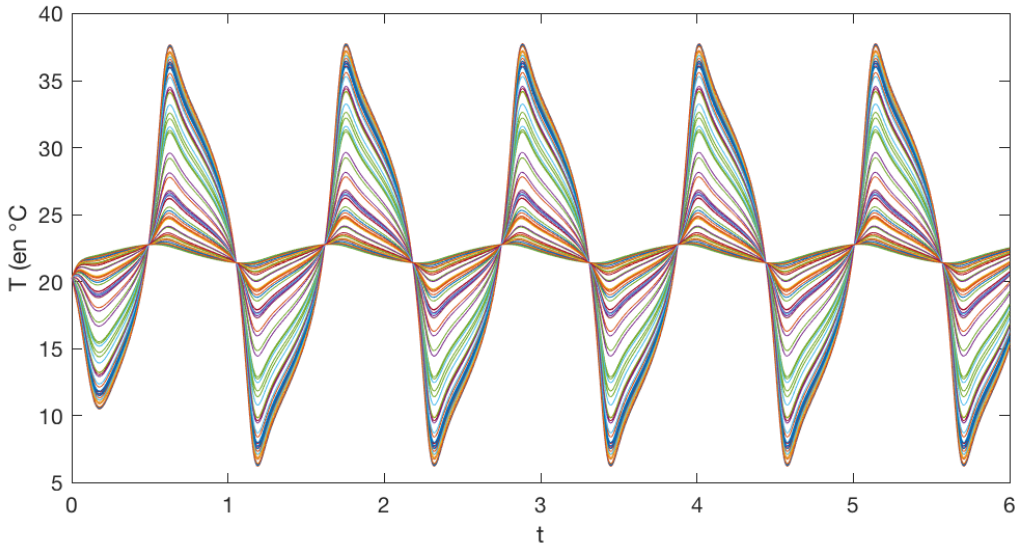


FIGURE 5.3 Évolution au cours du temps de la solution u au point $\boldsymbol{x} = (0.9, 0.9)$ pour tous les paramètres $\mu \in \mathcal{D}_h$

Au vu de la figure 5.3, il semble que l'on soit dans un cas bien adapté à l'utilisation des bases réduites. En effet, la solution u évolue différemment pour chaque paramètre, mais l'information en termes de profils dynamiques semble être assez pauvre.

Afin de valider, le bon déroulement de l'apprentissage et de l'identification, nous séparerons les données en deux ensembles : un pour l'entrainement et l'autre pour la validation. Pour la variable temporelle, nous utiliserons les $\frac{N_t}{2}$ premiers pas de temps pour l'entrainement et les pas de temps suivants pour la validation. En ce qui concerne les paramètres, nous séparerons aléatoirement l'ensemble $\mathcal{D}_h$ et utiliserons 50% de l'ensemble $\mathcal{M}_h$ pour l'entrainement et le reste pour la validation. La figure ci-dessous représente le flux ϕ_{out} sur les ensembles d'entrainement et de validation.

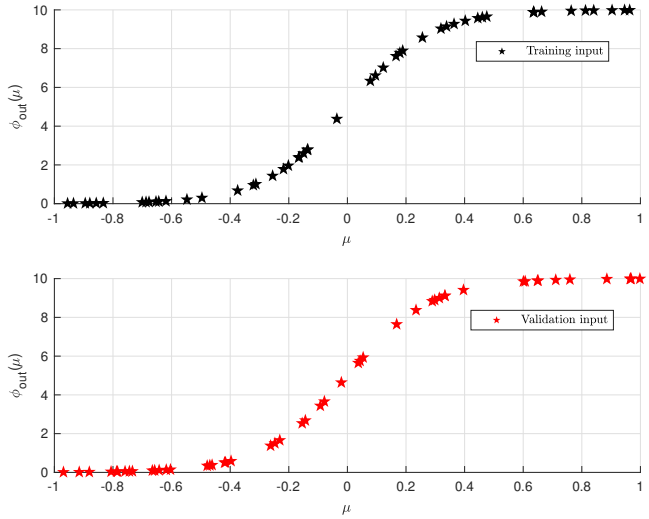


FIGURE 5.4 (Haut) ϕ_{out} sur l'ensemble d'entraînement. (Bas) ϕ_{out} sur l'ensemble de validation.

Nous utiliserons les notations $\mathcal{M}_h^{train}$, $\mathcal{D}_h^{train}$ pour les données d'entrainement. Et on notera M^{train} le cardinal de l'ensemble d'entrainement.

5.4.1.2 Réduction de la dimension par EIM

À présent, réduisons les données contenues dans les matrices $\mathbf{U}(\mu_m) \in \mathcal{M}_h^{train}$, utilisons pour cela, l'algorithme *approximation de tenseur par EIM* avec $K = K' = 10$. La figure ci-dessous représente l'erreur de réduction selon l'indice du paramètre $\mu_m \in \mathcal{D}_h^{train}$ ainsi que les points sélectionnés par l'algorithme.

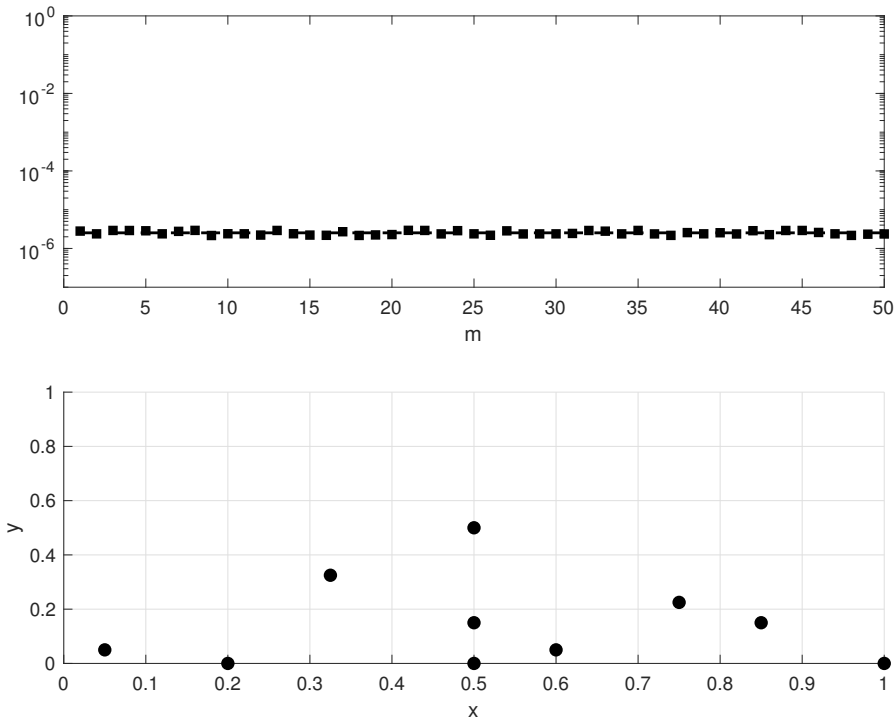


FIGURE 5.5 (Haut) Erreur de réduction pour chaque paramètre μ_m (carré noir) et erreur de réduction moyenne (ligne noire pointillée), (Bas) Points correspondant aux lignes sélectionnées (point noir)

Bien que le problème soit en 2D, on obtient une erreur de réduction de 2.53×10^{-6} en moyenne avec seulement 10 points sélectionnés.

Remarque 23. *En plus des points d'intérêt sélectionnés, l'algorithme approximation de tenseur par EIM nous fournit dix modes* $\mathbf{q}_k$ *permettant de reconstruire la solution* u *sur tout le domaine. La figure ci-dessous présente les* 10 *modes associés aux* 10 *points de la figure 5.5.*

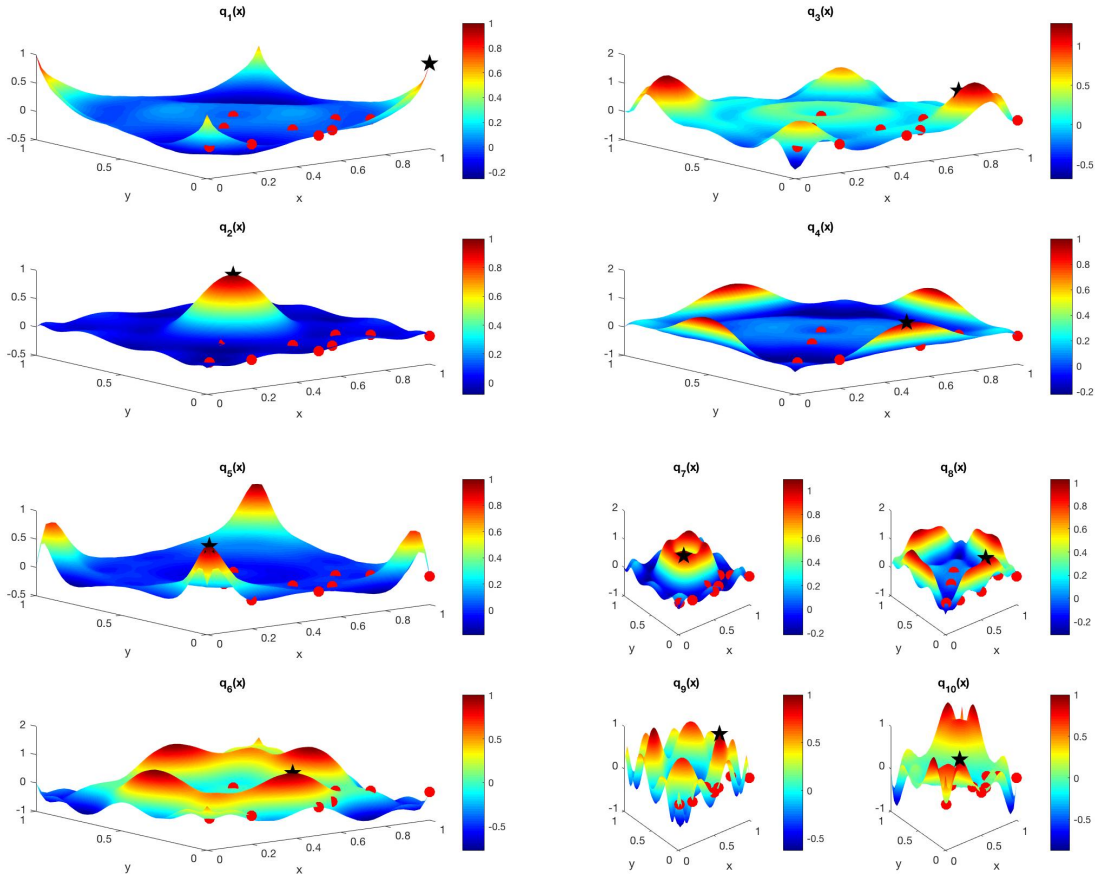


FIGURE 5.6 Les 10 modes $\mathbf{q}_k$ ainsi que leurs valeurs au point $\boldsymbol{x}_{\sigma(k)}$ (étoile noire) et aux autres points $\boldsymbol{x}_{\sigma(l)}$ (point rouge)

5.4.1.3 Apprentissage de la variété solution

À la section précédente, nous avons construit la représentation basse dimension $\tilde{\mathbf{W}}(\mu_m)$ pour chaque matrice $\mathbf{U}(\mu_m) \in \mathcal{M}_h^{train}$, à l'aide de l'algorithme *Approximation de tenseur par EIM*. Le but de cette section est de prédire $\tilde{\mathbf{W}}(\mu)$ pour un paramètre μ donné, à partir des ensembles d'entrainement d'entrée et de sortie :

$$X = \begin{bmatrix} \mu_1 & \cdots & \mu_{M^{train}} \end{bmatrix}, \qquad Y = \begin{bmatrix} \tilde{\mathbf{W}}_{\cdot,1}(\mu_1) & \cdots & \tilde{\mathbf{W}}_{\cdot,1}(\mu_{M^{train}}) \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{W}}_{\cdot,K'}(\mu_1) & \cdots & \tilde{\mathbf{W}}_{\cdot,K'}(\mu_{M^{train}}) \end{bmatrix}.$$

Ayant choisi comme paramètres de réduction $K = K' = 10$, les représentations basses dimensions sont chacune composée de 100 coefficients. La majorité des algorithmes de régression ne peuvent prédire que des quantités scalaires, ainsi on entrainera 100

algorithmes de régression afin de prédire chaque coefficient de la représentation basse dimension.

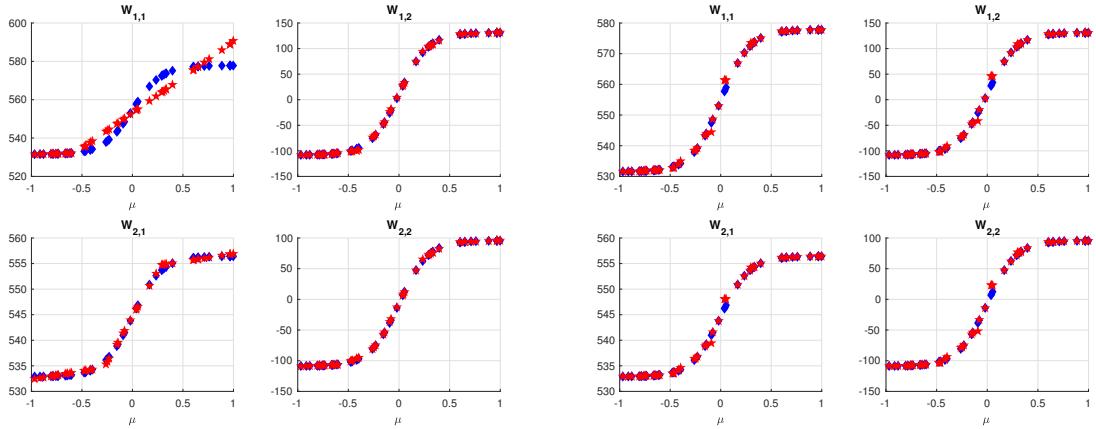
En ce qui concerne l'algorithme de régression, nous utiliserons l'algorithme *Kernel Ridge Regression* (KRR, [180]). Et nous le comparerons avec d'autres algorithmes de régression. Nous utiliserons les algorithmes de la librairie Scikit-Learn dont les appels sont donnés ci-dessous :

```
KRR = KernelRidge(alpha=0.00, kernel='rbf', gamma=0.8)
RF =  RandomForestRegressor(max_depth=30, random_state=1, n_estimators=50)
kNN = neighbors.KNeighborsRegressor(2, weights='distance')
GP = GaussianProcessRegressor()
ANN = MLPRegressor(solver="lbfgs", activation='relu')
ANN.loss = 'log_loss'
ANN.hidden_layer_sizes = (6,6)
```

On mesurera la performance de chaque algorithme par :

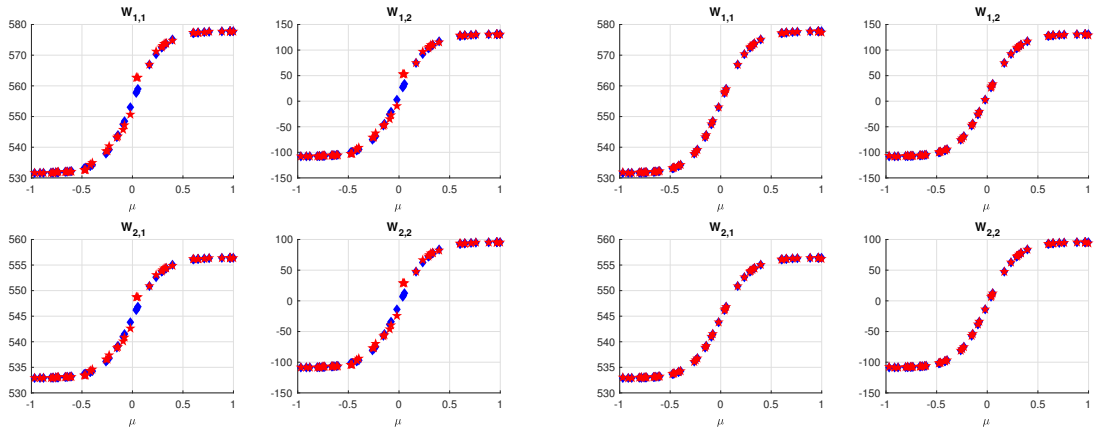
$$100\left[1-\frac{1}{\#\mathcal{D}_h^{val}}\sum_{\mu\in\mathcal{D}_h^{val}}\frac{\|\tilde{\mathbf{W}}^{exact}(\mu)-\tilde{\mathbf{W}}^{pred}(\mu)\|_F}{\|\tilde{\mathbf{W}}^{exact}(\mu)\|_F}\right]^+,$$

avec $\tilde{\mathbf{W}}^{exact}(\mu)$ la représentation basse dimension de la solution HF, $\tilde{\mathbf{W}}^{pred}(\mu)$ la prédiction par l'algorithme de régression, et $\mathcal{D}_h^{val}$ l'ensemble des paramètres de validation. Par ailleurs, le temps de prédiction sera aussi à prendre en compte étant donné que nous souhaitons obtenir un algorithme *ONLINE*. La figure ci-dessous représente la prédiction de la sous-matrice $\begin{pmatrix}\tilde{\mathbf{W}}_{1,1}(\mu) & \tilde{\mathbf{W}}_{1,2}(\mu)\\ \tilde{\mathbf{W}}_{2,1}(\mu) & \tilde{\mathbf{W}}_{2,2}(\mu)\end{pmatrix}$ pour chaque algorithme.



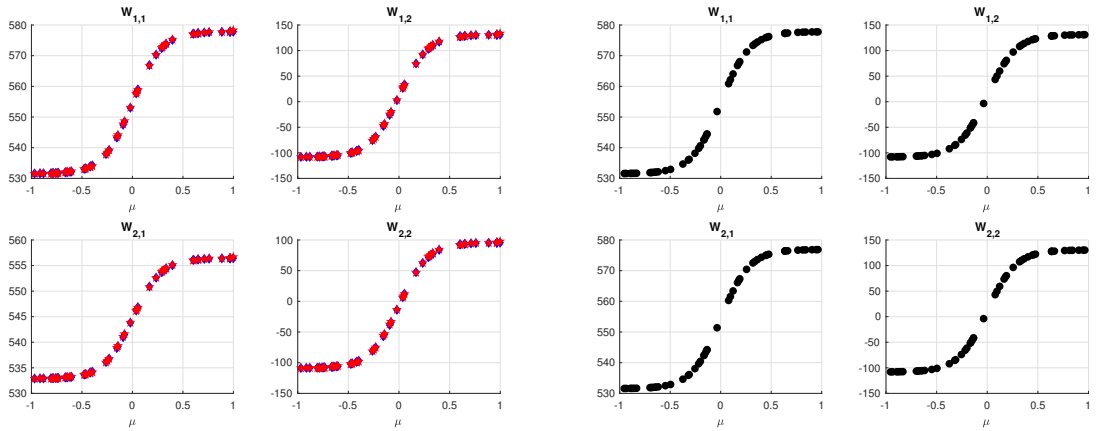
(a) ANN, score = 99.3%, temps = 0.013 sec

(b) kNN, score = 99.68%, temps = 0.051 sec



(c) RF, score = 99.47 %, temps = 0.046 sec

(d) KRR, score = 99.95%, temps = 0.049 sec



(e) GP, score = 99.89%, temps = 0.02 sec

(f) Données d'entrainement

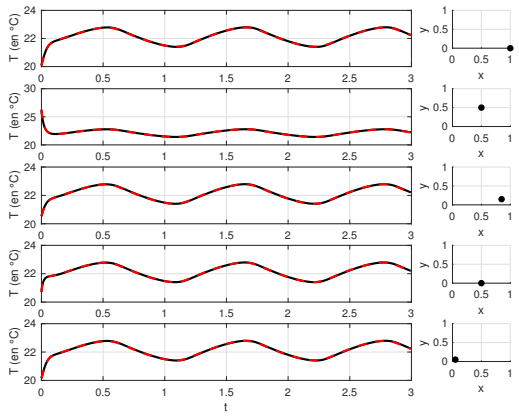
FIGURE 5.7 Prédiction (en rouge) et valeur exacte (en bleu) des quatre premiers coefficients de la matrice $\tilde{\mathbf{W}}(\mu)$ pour tout $\mu \in \mathcal{D}_h^{val}$. (En bas à droite) Données d'entrainement.

On remarque que les données sont structurées, on retrouve la forme sigmoïdale du flux externe ϕ_{out}. Le réseau de neurones semble avoir du mal à prédire correctement la sous-matrice, de plus son entrainement est coûteux. L'algorithme des k plus proches voisins montre quant à lui de bons résultats, cependant lorsque localement peu de données sont disponibles, comme c'est le cas autour de $\mu = 0$, cet algorithme montre ses limites. C'est par ailleurs aussi le cas pour l'algorithme des forêts aléatoires. Enfin, les deux meilleurs algorithmes en termes de performance sont les algorithmes KRR et GP.

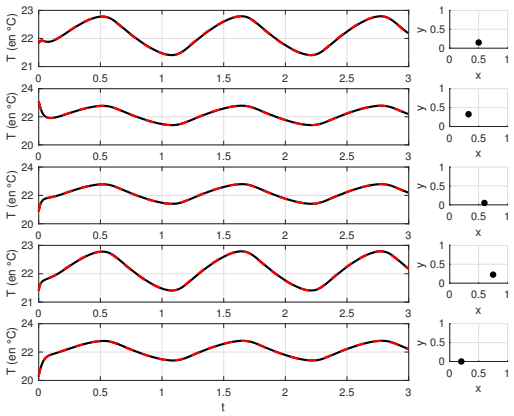
Il nous reste à présent à reconstruire la solution u aux points sélectionnés, à l'aide des modes temporels (comme fait à la **section 5.3.2**).

$$\hat{\mathbf{U}}_{\sigma,\cdot}(\mu) := \hat{\tilde{\mathbf{W}}}(\mu)\, \mathbf{\Psi}_r^T$$

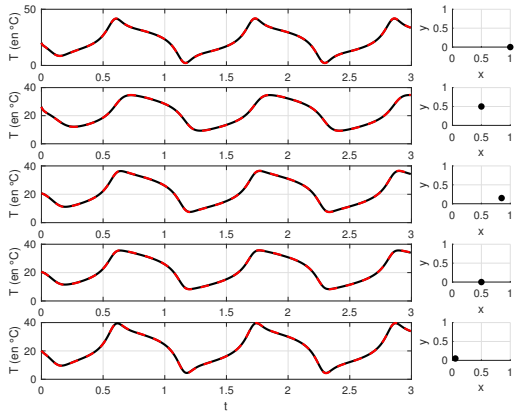
La figure ci-dessous représente les solutions HF et les solutions prédites pour deux paramètres aux 10 points sélectionnés.



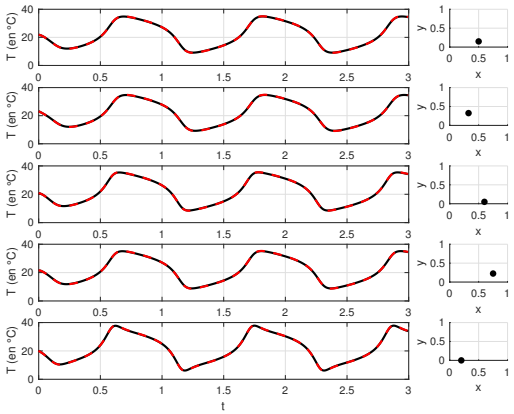
(a) $\mu = -0.9690$ (en $\boldsymbol{x}_{\sigma(1)}, ..., \boldsymbol{x}_{\sigma(5)}$)



(b) $\mu = -0.9690$ (en $\boldsymbol{x}_{\sigma(6)}, ..., \boldsymbol{x}_{\sigma(10)}$)



(c) $\mu = 0.9982$ (en $\boldsymbol{x}_{\sigma(1)}, ..., \boldsymbol{x}_{\sigma(5)}$)



(d) $\mu = 0.9982$ (en $\boldsymbol{x}_{\sigma(6)}, ..., \boldsymbol{x}_{\sigma(10)}$)

FIGURE 5.8 (Gauche) Comparaison entre la prédiction de la solution (ligne pointillé rouge) et la solution HF (ligne noir continue) aux points sélectionnés pour différents μ, (Droite) Position du point sur le domaine

Les deux courbes sont parfaitement alignées, on a donc bien prédit la solution aux points sélectionnés pour un paramètre donné. Cependant, pour prédire son évolution à des temps futurs, il reste encore à apprendre un modèle réduit fidèle au modèle (5.3).

5.4.1.4 Apprentissage d'un modèle réduit

Le modèle (5.3) étant issu d'un couplage EDP-EDO, il faudra d'une part identifier l'EDO et d'autre part identifier l'EDP. Commençons par identifier l'EDO. Une difficulté est qu'ici l'EDO est non linéaire, or nous avons jusqu'à présent travaillé avec des équations différentielles linéaires. Un moyen de contourner ce problème est de considérer

des observables non linéaires de θ, dans le même esprit que la méthode *Extended DMD* (EDMD, [184]).

Rappelons tout d'abord l'écriture de l'EDO guidant l'évolution de θ :

$$\ddot{\theta}(t) = \epsilon\,\omega\Big(1-\theta(t)^2\Big)\,\dot{\theta}(t) + \omega\theta(t), \qquad \forall t \in (0,T),$$

Comme nous l'avons vu au **Chapitre 2**, un moyen d'identifier un modèle dynamique d'ordre 2 est d'augmenter la dimension du modèle réduit en considérant sa dérivée. Considérons alors le vecteur $\boldsymbol{\theta}(t) = \begin{bmatrix}\theta(t)\\ \dot{\theta}(t)\end{bmatrix}$, on obtient l'EDO,

$$\ddot{\theta}(t) = \begin{bmatrix}\omega & \epsilon\,\omega\Big(1-\theta(t)^2\Big)\end{bmatrix}\,\boldsymbol{\theta}(t), \qquad \forall t \in (0,T).$$

Malheureusement ici la matrice dépend de l'état $\theta(t)$ et donc de t. Or la méthode DMD ne peut identifier qu'un opérateur stationnaire. Afin de contourner ce problème, ajoutons l'observable non linéaire $\theta(t)^2\dot{\theta}(t)$ au vecteur $\boldsymbol{\theta}(t)$, on obtient alors l'EDO :

$$\ddot{\theta}(t) = \begin{bmatrix}\omega & \epsilon\,\omega & -\epsilon\,\omega\end{bmatrix}\,\boldsymbol{\theta}(t), \qquad \forall t \in (0,T),$$

Finalement, on retrouve un opérateur stationnaire, facilement identifiable par la méthode DMD. En utilisant la formule de Taylor, on retrouve dans le cas discret

$$\theta^{n+1} \approx \mathbf{C}\,\boldsymbol{\theta}^n, \qquad \forall 1 \leq n \leq N_t - 1,$$

avec $\mathbf{C}$ ne dépendant pas de t^n.
Bien-sûr considérer cette observable non linéaire demande la connaissance a priori de l'EDO et est donc intrusive. C'est pourquoi nous considérerons l'ensemble des observables $\theta^\alpha(t)\dot{\theta}^\beta(t)$ avec $\alpha+\beta \leq 3$, afin d'entraîner notre modèle réduit.
Les ensembles d'entrainement seront alors :

$$X = \begin{bmatrix} 1 & \cdots & 1 \\ \theta^1 & \cdots & \theta^{N_t-1} \\ \dot{\theta}^1 & \cdots & \dot{\theta}^{N_t-1} \\ \vdots & \ddots & \vdots \\ (\dot{\theta}^1)^2\theta^1 & \cdots & (\dot{\theta}^{N_t-1})^2\theta^{N_t-1} \\ (\theta^1)^3 & \cdots & (\theta^{N_t-1})^3 \\ (\dot{\theta}^1)^3 & \cdots & (\dot{\theta}^{N_t-1})^3 \end{bmatrix}, \qquad Y = \begin{bmatrix} \theta^2 & \cdots & \theta^{N_t} \\ \dot{\theta}^2 & \cdots & \dot{\theta}^{N_t} \end{bmatrix},$$

et le modèle réduit fidèle à l'EDO sera :

$$\begin{bmatrix} \theta^{n+1} \\ \dot{\theta}^{n+1} \end{bmatrix} = \hat{\mathbf{C}} \left[(\theta^n)^\alpha (\dot{\theta}^n)^\beta \right]_{\alpha+\beta \leq 3}, \tag{5.5}$$

avec

$$\hat{\mathbf{C}} := Y X^\dagger.$$

La figure ci-dessous représente $\theta(t)$ et sa dérivée $\dot{\theta}(t)$ obtenues par la routine `ode45` et par le modèle réduit.

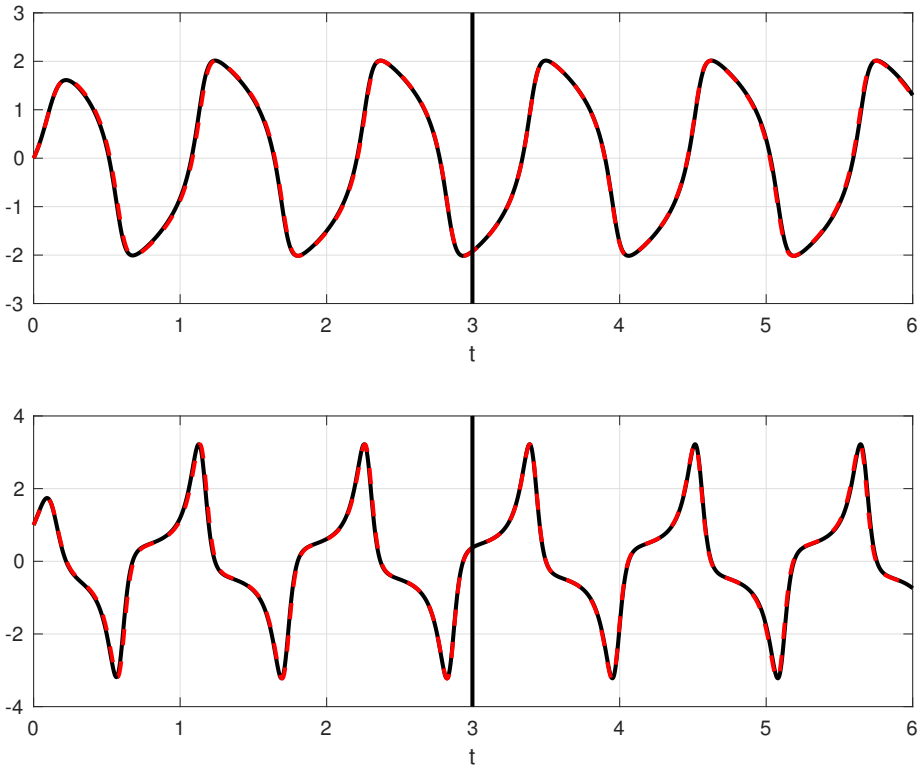


FIGURE 5.9 (Haut) Évolution de $\theta(t)$, obtenue par la routine `ode45` (ligne continue noire) et par la prédiction du modèle réduit (ligne pointillée rouge), pour les temps d'entrainement (avant la droite verticale noire) et les temps de validation, (Bas) Même chose pour $\dot{\theta}(t)$.

On a donc parfaitement identifié l'EDO.

À présent, on cherche à identifier l'EDP. La méthode est semblable à celle présentée au **Chapitre 4** . Ainsi les ensembles d'entrainement seront définis par :

$$X(\boldsymbol{\mu}) = \begin{bmatrix} | & | & & | \\ \hat{\mathbf{u}}_\sigma^1(\boldsymbol{\mu}) & \hat{\mathbf{u}}_\sigma^2(\boldsymbol{\mu}) & \cdots & \hat{\mathbf{u}}_\sigma^{N_t-1}(\boldsymbol{\mu}) \\ | & | & & | \\ \theta^1 & \theta^2 & \dots & \theta^{N_t-1} \\ \dot{\theta}^1 & \dot{\theta}^2 & \dots & \dot{\theta}^{N_t-1} \end{bmatrix} \quad \text{et} \quad Y(\boldsymbol{\mu}) = \begin{bmatrix} | & | & & | \\ \hat{\mathbf{u}}_\sigma^2(\boldsymbol{\mu}) & \hat{\mathbf{u}}_\sigma^3(\boldsymbol{\mu}) & \cdots & \hat{\mathbf{u}}_\sigma^{N_t}(\boldsymbol{\mu}) \\ | & | & & | \end{bmatrix},$$

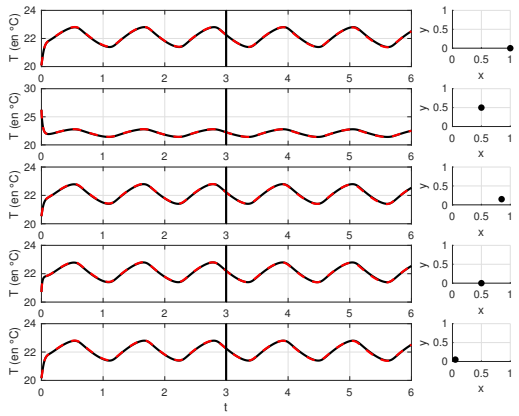
et le modèle réduit fidèle au modèle (5.3) sera de la forme :

$$\begin{cases} \hat{\mathbf{u}}_\sigma^{n+1}(\boldsymbol{\mu}) = \hat{\mathbf{A}}(\boldsymbol{\mu})\, \hat{\mathbf{u}}_\sigma^n(\boldsymbol{\mu}) + \hat{\mathbf{B}}(\boldsymbol{\mu}) \begin{bmatrix} \theta^n \\ \dot{\theta}^n \end{bmatrix}, \\ \begin{bmatrix} \theta^{n+1} \\ \dot{\theta}^{n+1} \end{bmatrix} = \hat{\mathbf{C}} \left[(\theta^n)^\alpha (\dot{\theta}^n)^\beta \right]_{\alpha+\beta\leq 3}, \\ \hat{\mathbf{u}}^{n+1}(\boldsymbol{\mu}) = \mathbf{Q}\, \hat{\mathbf{u}}_\sigma^{n+1}(\boldsymbol{\mu}), \end{cases} \tag{5.6}$$

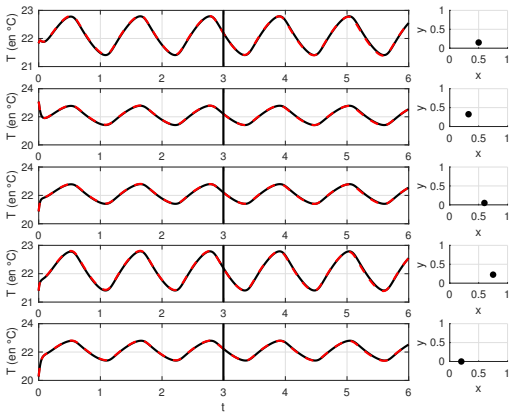
avec

$$\begin{bmatrix} \hat{\mathbf{A}}(\boldsymbol{\mu}) & \hat{\mathbf{B}}(\boldsymbol{\mu}) \end{bmatrix} = Y(\boldsymbol{\mu}) X^\dagger(\boldsymbol{\mu}).$$

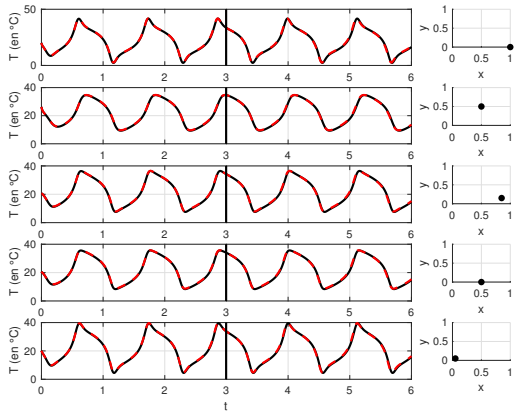
Comparons à présent l'évolution de la solution u aux points sélectionnés, fournie par le solveur HF et la prédiction par le modèle réduit (5.6). La figure ci-dessus représente la comparaison des solutions aux 10 points sélectionnés.



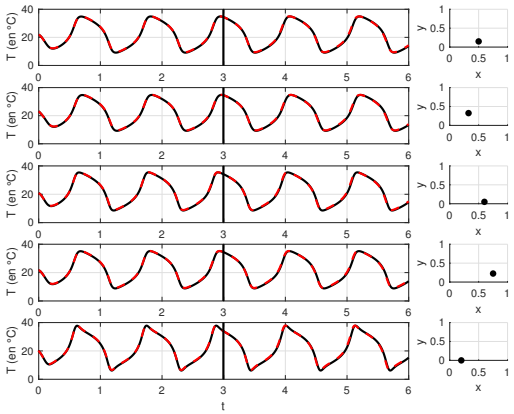
(a) $\mu = -0.8808$ (en $\boldsymbol{x}_{\sigma(1)}, ..., \boldsymbol{x}_{\sigma(5)}$)



(b) $\mu = -0.8808$ (en $\boldsymbol{x}_{\sigma(6)}, ..., \boldsymbol{x}_{\sigma(10)}$)



(c) $\mu = 0.9982$ (en $\boldsymbol{x}_{\sigma(1)}, ..., \boldsymbol{x}_{\sigma(5)}$)



(d) $\mu = 0.9982$ (en $\boldsymbol{x}_{\sigma(6)}, ..., \boldsymbol{x}_{\sigma(10)}$)

FIGURE 5.10 (Gauche) Comparaison entre la prédiction de la solution u (ligne pointillé rouge) et celle fournie par le solveur HF (ligne noir continue) aux points sélectionnés pour la réduction (Droite) Position du point sur le domaine

Là encore la prédiction de la solution u colle parfaitement à la solution fournie par le solveur HF.

Remarque 24. *Le but était d'apprendre un modèle réduit tel que l'on puisse prédire la solution u instantanément. Le tableau ci-dessous résume les temps de calcul des différentes étapes pour la phase ONLINE.*

Étape	**Temps de calcul**
Prédiction par KRR	0.049 sec
Apprentissage du modèle réduit	0.14 sec
Prédiction de la solution u sur $(0, T)$	0.07 sec
Reconstruction globale de la solution u	0.07 sec
Total	0.33 sec
Solveur HF	34 sec

TABLE 5.2 Comparaison des temps de calcul pour la phase *ONLINE* et pour le solveur HF.

On a donc réduit le temps de calcul par 100*.*

On aurait pu envisager une autre approche, consistant à apprendre M^{train} modèles réduits pour chacun des paramètres $\mu_m \in \mathcal{D}_h^{train}$. Puis utiliser un algorithme de régression afin de prédire pour un paramètre donné, le modèle réduit. Pour notre problème, les données d'entrainement ne sont pas structurées et le problème de régression complexe. La prédiction par cette approche explose totalement. Les résultats numériquement sont présentés en **Annexe D**.

Cela illustre le fait qu'en plus du choix de l'algorithme de régression, une attention particulière doit être portée aux données que l'on souhaite apprendre. En effet, apprendre la variété solution a du sens et est motivée par la finesse de son épaisseur de Kolmogorov.

5.4.2 Optimisation de formes non-intrusive à l'aide d'une approximation QUR d'un canal traversé par une onde

5.4.2.1 Présentation du problème

On souhaite dans cette section, optimiser la forme d'un canal traversé par une onde afin de limiter son impact sur celle-ci. Par ailleurs, l'onde sera constamment excitée au bord du domaine, ce que l'on modélisera par une condition de Dirichlet dynamique. La forme du canal sera quant à elle, définie par deux fonctions permettant de décrire la partie supérieure et inférieure du canal. Ce système peut ainsi être modélisé par

([130]) :

$$\begin{cases} \partial_{tt} u(x,t;\boldsymbol{\mu}) - c\,\partial_x\Big(s(x;\boldsymbol{\mu})\partial_x u(x,t;\boldsymbol{\mu})\Big) = 0, & \text{dans } [0;1]\times(0,T), \\ u(0,t;\boldsymbol{\mu}) = \boldsymbol{\theta}_1(t), \quad u(1,t;\boldsymbol{\mu}) = 0 & \forall t \in \times(0,T), \\ u(x,0;\boldsymbol{\mu}) = 0, & \text{dans } [0;1] \\ \dot{\boldsymbol{\theta}}(t) = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \boldsymbol{\theta}(t), \qquad \boldsymbol{\theta}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, & \forall t \in \times(0,T), \\ s(x;\boldsymbol{\mu}) = \boldsymbol{g}_1(x;\boldsymbol{\mu}) - \boldsymbol{g}_2(x;\boldsymbol{\mu}), & \text{dans } [0;1] \end{cases} \tag{5.7}$$

La fonction s représente la hauteur du canal, ainsi plus s sera petit, plus l'onde oscillera. Les fonctions $\boldsymbol{g}_1$ et $\boldsymbol{g}_2$ correspondent quant à elles à la partie supérieure et inférieure du canal, et sont définies par :

$$\boldsymbol{g}_1(x;\boldsymbol{\mu}) = \frac{\epsilon}{2} + \frac{2}{\tau\sqrt{100\pi}} e^{-\frac{(x-\boldsymbol{\mu}_1)^2}{2\tau^2}},$$

et

$$\boldsymbol{g}_2(x;\boldsymbol{\mu}) = -\frac{\epsilon}{2} - \frac{2}{\tau\sqrt{100\pi}} e^{-\frac{(x-1-\boldsymbol{\mu}_2)^2}{2\tau^2}}.$$

Ainsi le paramètre $\boldsymbol{\mu} \in \mathcal{D}$ définit les endroits où le canal sera élargi. De plus, le paramètre ϵ permet d'assurer une hauteur minimale en tout point du domaine, de sorte que l'on ait :

$$s(x;\boldsymbol{\mu}) \geq \epsilon, \qquad \forall x, \boldsymbol{\mu}.$$

Dans ce travail, nous souhaitons optimiser la forme du canal de façon à ce que l'onde soit la moins perturbée possible, tout en limitant la hauteur du canal. Pour ce faire, nous introduisons la fonction coût J définie par :

$$J(\boldsymbol{\mu}) = \alpha\, D(\boldsymbol{\mu}) + (1-\alpha)\; L(\boldsymbol{\mu}), \qquad \text{avec } \alpha \in [0;1], \tag{5.8}$$

avec

$$D(\boldsymbol{\mu}) = \frac{\|u(\cdot,\cdot;\boldsymbol{\mu}) - u_f\|_{L^2(0,T;L^2([0;1]))}}{\|u_f\|_{L^2(0,T;L^2([0;1]))}}$$

et où la fonction u_f correspond à une onde libre i.e. $s = 1$ sur tout le domaine, et est solution du système :

$$\begin{cases} \partial_{tt} u_f(x,t) - c\,\partial_{xx} u_f(x,t) = 0, & \text{dans } [0;1] \times (0,T), \\ u_f(0,t) = \boldsymbol{\theta}_1(t), \quad u_f(1,t) = 0 & \forall t \in \times(0,T), \\ u_f(x,0) = 0, & \text{dans } [0;1] \\ \dot{\boldsymbol{\theta}}(t) = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \boldsymbol{\theta}(t), \qquad \boldsymbol{\theta}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, & \forall t \in \times(0,T), \end{cases} \tag{5.9}$$

Tandis que la fonction L est définie par :

$$L(\boldsymbol{\mu}) := \int_0^1 s(x;\boldsymbol{\mu}) dx, \tag{5.10}$$

et correspond ainsi à la somme des hauteurs du canal.
Il nous reste alors à trouver le paramètre optimal $\boldsymbol{\mu}^*$ défini par :

$$\boldsymbol{\mu}^* := \underset{\boldsymbol{\mu} \in \mathcal{D}}{\arg\min}\ J(\boldsymbol{\mu}).$$

Cependant, comme tout au long de cette thèse, nous nous plaçons dans un cadre non-intrusif où l'on ne dispose que des sorties du solveur HF. Il nous faudra ainsi résoudre ce problème d'optimisation sans la connaissance du modèle (5.7). On apprendra alors à partir de données, cette fonctionnelle afin de pouvoir l'évaluer rapidement, et ainsi trouver $\boldsymbol{\mu}^*$ à l'aide d'un algorithme d'optimisation. La section ci-dessous présente les paramètres physiques et numériques ayant servi à la génération de données pour le solveur HF.

5.4.2.2 Génération des données

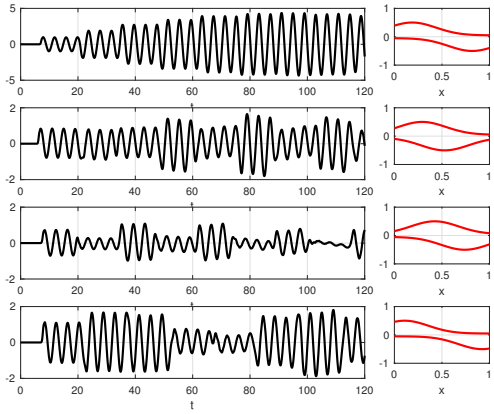
On échantillonnera l'ensemble des paramètres admissibles $\mathcal{D}$, par une grille régulière contenant M nœuds, que l'on notera $\mathcal{D}_h$. Ainsi on calculera, à l'aide du solveur HF, la solution du modèle (5.7) pour chaque nœud de cette grille, que l'on stockera dans la matrice $\mathbf{U}(\boldsymbol{\mu}_m)$. On stockera également les fonctions $\boldsymbol{g}_1(\cdot;\boldsymbol{\mu}_m)$ et $\boldsymbol{g}_2(\cdot;\boldsymbol{\mu}_m)$ dans une matrice $\mathbf{G}(\boldsymbol{\mu}_m)$. Enfin, l'onde libre sera stockée dans la matrice $\mathbf{U}^f$.

Le tableau ci-dessous présente les paramètres physiques et numériques utilisés pour la génération des données.

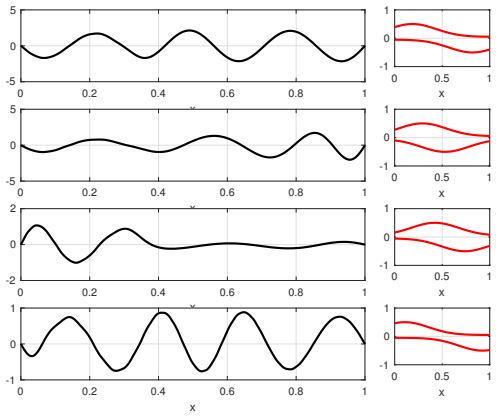
Paramètres	**Valeurs**
Célérité de l'onde (c)	0.1
Pulsation (ω)	$\frac{\pi}{2}$
Hauteur minimale du canal (ϵ)	0.1
Variance du canal (τ)	0.25
Temps final (T)	120 sec
Ensemble admissible ($\mathcal{D}$)	$[0; 0.5]^2$
Pas d'espace (δx)	5×10^{-3}
Pas de temps (δt)	1.5×10^{-2}
Nombre de nœuds de $\mathcal{D}_h$ (M)	400

TABLE 5.3 Paramètres physiques et numériques utilisés

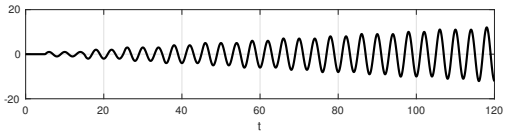
La figure, à gauche, ci-dessous représente l'évolution de l'onde au cours du temps au point $x = 0.5$. Tandis que la figure, à droite, représente l'onde à $t = 60$ seconde sur tout le domaine.



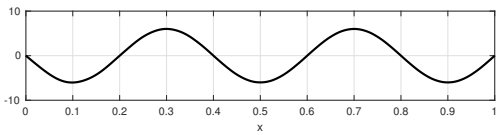
(a) Évolution de l'onde au cours du temps selon la géométrie du canal (rouge)



(b) Profils d'onde à $t = 60$ sec selon la géométrie du canal (rouge)



(c) Évolution de l'onde libre au cours du temps



(d) Profil de l'onde libre à $t = 60$ sec

On peut ainsi remarquer sur la figure (5.11a) qu'une légère variation de la forme du canal affecte sévèrement le comportement de l'onde au cours du temps. Ainsi contrairement au problème thermique précédent, il semble que la variété des solutions ait une épaisseur de Kolmogorov bien moins fine. En ce qui concerne notre problème d'optimisation, on remarque que l'onde libre continue de croitre au cours du temps avec la même périodicité (5.11c). On souhaiterait alors trouver une géométrie de canal permettant ce comportement.

5.4.2.3 Discrétisation de la fonction coût

Travaillant à partir de données fournies par le solveur HF nous souhaiterions discrétiser la fonction coût J. Le premier terme de la fonction coût, définie par (5.8),

sera alors approché par :

$$D(\boldsymbol{\mu}) \approx \frac{\|\mathbf{U}(\boldsymbol{\mu}) - \mathbf{U}^f\|_F}{\|\mathbf{U}^f\|_F} \tag{5.11}$$

Tandis que le second terme, défini par (5.10), sera approché à l'aide d'une intégration numérique, i.e.

$$L(\boldsymbol{\mu}) \approx \sum_{i=1}^{N_x} \omega_i \, s(x_i; \boldsymbol{\mu})$$

Afin d'alléger les notations on notera également J, la fonction coût discrète, D la distance discrète, et L la hauteur discrète. Évaluer le premier terme de cette fonctionnelle pour un paramètre $\boldsymbol{\mu}$ donné, nécessite la prédiction d'une matrice de taille $N_x \times N_t$, mais également de mesurer l'erreur par rapport à la matrice cible $\mathbf{U}^f$, ce qui là aussi dépend de N_x et N_t. Quant au second terme, il nécessite l'évaluation de la hauteur en N_x points, puis son intégration numérique dépendant là aussi de N_x. L'optimisation nécessitant un grand nombre d'évaluations de la fonction coût, cette approche n'est pas viable. Il nous faut ainsi réduire les données.

5.4.2.4 Réduction des données

L'approximation QUR introduite à la **Section 5.1.3** permet de sélectionner des lignes et des colonnes d'intérêts, ainsi que de construire deux opérateurs d'interpolation $\mathbf{Q}$ et $\mathbf{R}$. Considérons la matrice augmentée :

$$\begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}(\boldsymbol{\mu}_M) \\ \mathbf{U}^f \end{bmatrix},$$

ainsi l'approximation QUR de la matrice ci-dessus est :

$$\begin{bmatrix} \mathbf{U}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{U}(\boldsymbol{\mu}_M) \\ \mathbf{U}^f \end{bmatrix} \approx \mathbf{Q} \begin{bmatrix} \tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}_1) \\ \vdots \\ \tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}_M) \\ \tilde{\tilde{\mathbf{U}}}^f \end{bmatrix} \mathbf{R},$$

avec $\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}_m) := \mathbf{U}_{\sigma_x,\sigma_t}(\boldsymbol{\mu}_m)$ la matrice réduite aux lignes et colonnes sélectionnées. En substituant cette décomposition au premier terme de la fonction coût J, on obtient :

$$\frac{\|\mathbf{U}(\boldsymbol{\mu}) - \mathbf{U}^f\|_F}{\|\mathbf{U}^f\|_F} \approx \frac{\|\mathbf{Q}\left(\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}) - \tilde{\tilde{\mathbf{U}}}^f\right)\mathbf{R}\|_F}{\|\mathbf{Q}\,\tilde{\tilde{\mathbf{U}}}^f\,\mathbf{R}\|_F}$$

Cela motive ainsi de calculer le premier terme de le fonction coût en le substituant par :

$$\frac{\|\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu}) - \tilde{\tilde{\mathbf{U}}}^f\|_F}{\|\tilde{\tilde{\mathbf{U}}}^f\|_F}.$$

On est ainsi amené à prédire une matrice de taille $K \times K'$, et l'évaluation de J ne dépend alors plus de N_x et de N_t. L'interprétation de cette substitution est de comparer deux à deux l'onde libre et l'onde parcourant le canal, à des nœuds du maillage et des pas de temps pertinents. Cette approche est dans le même esprit que les travaux de Le Guennec & al qui ont utilisé la décomposition CUR combinée avec un algorithme de régression afin d'en déduire un modèle réduit non-intrusif pour des problèmes paramétrés ([108]).

Vérifions à présent la bonne sélection des points du maillage et pas de temps par l'algorithme QUR. Pour cela présentons l'erreur de reconstruction pour chaque paramètre $\boldsymbol{\mu} \in \mathcal{D}_h$, ainsi que les points et pas de temps sélectionnés. Nous choisirons $K = 20$ et $K' = 30$.

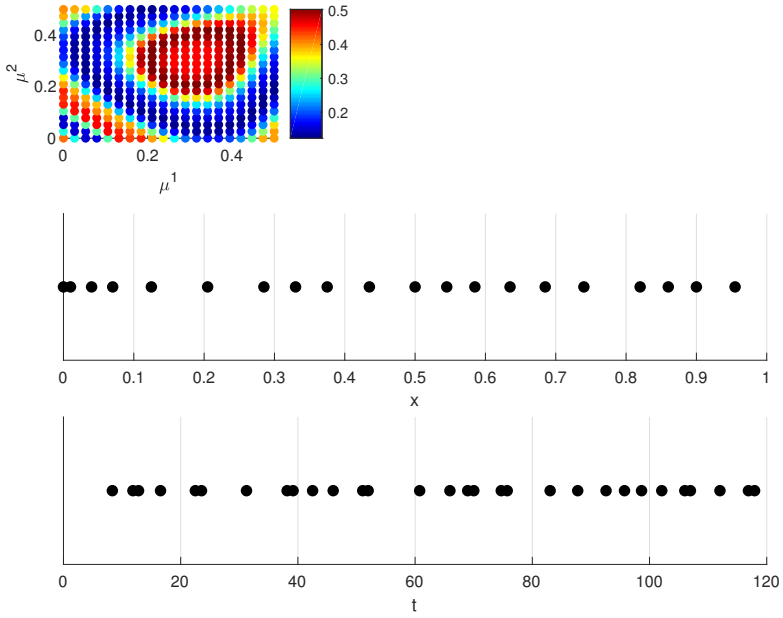


FIGURE 5.12 (Haut) Erreur de reconstruction pour chaque paramètre $\boldsymbol{\mu} \in \mathcal{D}_h$, (Bas) Points du maillage et pas de temps sélectionnés

Voyons ainsi si l'on peut, malgré l'erreur de réduction non négligeable, tout de même évaluer précisément la fonctionnelle J pour un paramètre $\boldsymbol{\mu}$ donné.

5.4.2.5 Apprentissage de la fonction coût J

Dans cette section, nous souhaitons apprendre la fonction coût J, afin de prédire rapidement sa valeur pour un $\boldsymbol{\mu}$ donné. Pour ce faire, nous chercherons d'abord à apprendre la fonction L, puis dans un second temps à apprendre la fonction D en utilisant l'approximation QUR présentée plus haut.

Apprentissage de la hauteur L

La fonction L est définie par :

$$L(\boldsymbol{\mu}) = \sum_{i=1}^{N_x} \omega_i \, s(x_i; \boldsymbol{\mu})$$

avec $s(x_i; \boldsymbol{\mu}) := \boldsymbol{g}_1(x_i; \boldsymbol{\mu}) - \boldsymbol{g}_2(x_i; \boldsymbol{\mu})$. Bien que s soit connue, évaluer telle quelle la fonction L dépend de N_x ce qui n'est pas souhaitable. Nous allons donc réduire la dimension de s.

En utilisant l'algorithme *Approximation faible rang par EIM* introduit au **Chapitre 4**, nous sélectionnerons K points du maillage à l'aide de l'application σ, ainsi que K modes $\mathbf{q}_k$. Ainsi on obtient l'approximation :

$$s(x_i;\boldsymbol{\mu}) \approx \sum_{k=1}^{K} s(x_{\sigma(k)};\boldsymbol{\mu})\,\mathbf{q}_k(x_i),$$

de sorte que

$$L(\boldsymbol{\mu}) = \sum_{i=1}^{N_x} \omega_i\, s(x_i;\boldsymbol{\mu}) \approx \sum_{k=1}^{K} s(x_{\sigma(k)};\boldsymbol{\mu}) \sum_{i=1}^{N_x} \omega_i\, \mathbf{q}_k(x_i).$$

Bien que l'intégration numérique des modes $\mathbf{q}_k$ dépend encore de N_x, on peut la réaliser *OFFLINE* car elle ne dépend pas du paramètre. Il nous suffira donc d'effectuer un produit scalaire de taille K dans la phase *ONLINE*.

Enfin, bien que s soit connue, il est plus simple d'apprendre directement sa valeur aux points sélectionnés par un algorithme de régression. Ainsi nous l'apprendrons avec l'algorithme KRR.

Apprentissage de la distance D

En considérant l'approximation QUR présentée précédemment, il suffit de prédire la matrice $\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu})$. Nous utiliserons l'algorithme KRR.

5.4.2.6 Validation de la fonction coût apprise $\hat{J}$

À présent vérifions que la méthode permet de prédire avec précision la fonction coût. Pour ce faire, considérons 1000 paramètres tirés aléatoirement dans $\mathcal{D}$.

Validation de la hauteur apprise $\hat{L}$

Nous vérifierons que les valeurs de s aux points du maillage sélectionnés ont été correctement prédites, que la connaissance à ces points suffit pour correctement reconstruire s, et finalement que la fonction $\hat{L}$ apprise est fidèle à la fonction L

La figure ci-dessous présente les résultats pour $K = 5$.

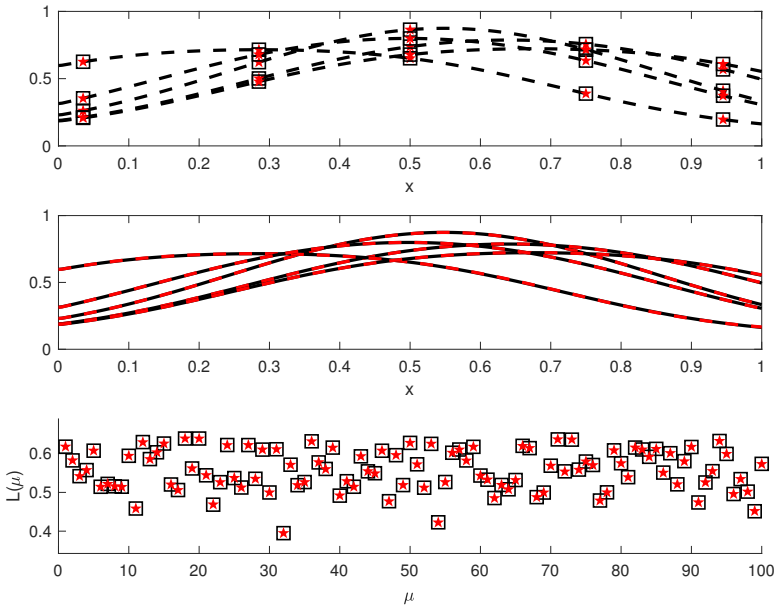


FIGURE 5.13 (Haut) Valeurs exactes aux points sélectionnés de s (carré noir) et valeurs prédites aux points sélectionnés de s (étoile rouge), valeur de s sur tout le domaine (pointillé noir), (Milieu) Valeurs exactes de s (ligne noire), et reconstruction de la prédiction de s sur tout le domaine (ligne pointillée rouge), (Bas) Valeurs exactes de L pour 100 paramètres de validations (carré noir), valeurs prédites de L (étoile rouge).

Au vu des deux premières figures, on remarque donc que d'une part l'algorithme KRR prédit avec une grande précision la valeur de s aux points sélectionnés. D'autre part que la sélection de ces points ainsi que l'opérateur d'interpolation, permet de reconstruire avec une grande précision la fonction s. Cela permet de prédire de façon précise la fonction L comme le montre la figure d'en bas.

Le tableau ci-dessous résume les erreurs relatives, sur l'ensemble de validation, pour les trois étapes.

Étape	**Erreur moyenne**	**Erreur maximal**
Prédiction des $s(x_{\sigma_x(k)}; \boldsymbol{\mu})$	2.86×10^{-5}	8.58×10^{-5}
Reconstruction de la prédiction de s	1.8×10^{-3}	6.4×10^{-3}
Prédiction de L	4.09×10^{-4}	1.6×10^{-3}

TABLE 5.4 Erreurs de prédiction sur l'ensemble de validation

On a donc parfaitement prédit la hauteur du canal en fonction du paramètre $\boldsymbol{\mu}$.

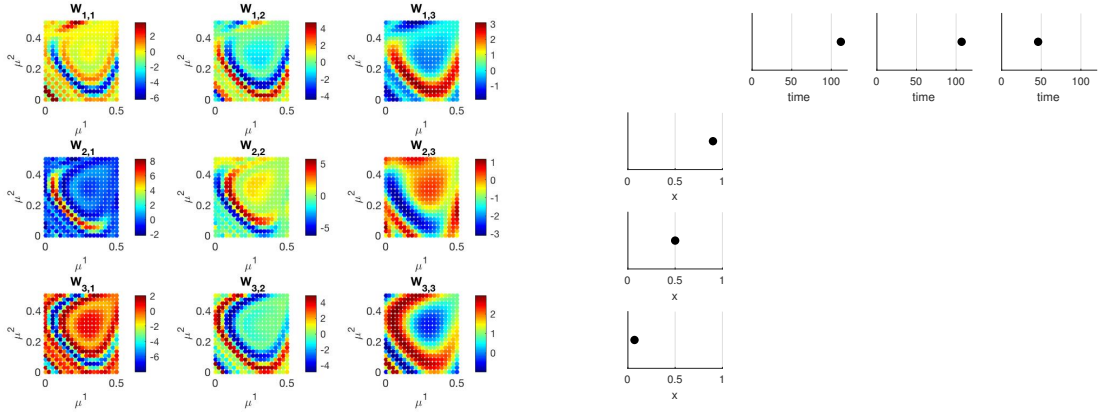
Validation de la distance apprise $\hat{D}$

À présent vérifions l'apprentissage du premier terme de J. Nous utiliserons l'algorithme KRR en comparant différents noyaux. Nous validerons ensuite sur le même ensemble de validation que précédemment.

Présentons tout d'abord la sous-matrice de l'ensemble d'entrainement fournit à nos algorithmes :

$$\begin{pmatrix} \tilde{\tilde{\mathbf{U}}}_{1,1}(\boldsymbol{\mu}) & \tilde{\tilde{\mathbf{U}}}_{1,2}(\boldsymbol{\mu}) & \tilde{\tilde{\mathbf{U}}}_{1,3}(\boldsymbol{\mu}) \\ \tilde{\tilde{\mathbf{U}}}_{2,1}(\boldsymbol{\mu}) & \tilde{\tilde{\mathbf{U}}}_{2,2}(\boldsymbol{\mu}) & \tilde{\tilde{\mathbf{U}}}_{2,3}(\boldsymbol{\mu}) \\ \tilde{\tilde{\mathbf{U}}}_{3,1}(\boldsymbol{\mu}) & \tilde{\tilde{\mathbf{U}}}_{3,2}(\boldsymbol{\mu}) & \tilde{\tilde{\mathbf{U}}}_{3,3}(\boldsymbol{\mu}) \end{pmatrix}.$$

pour tout $\boldsymbol{\mu} \in \mathcal{D}_h$. La figure ci-dessous présente les résultats :

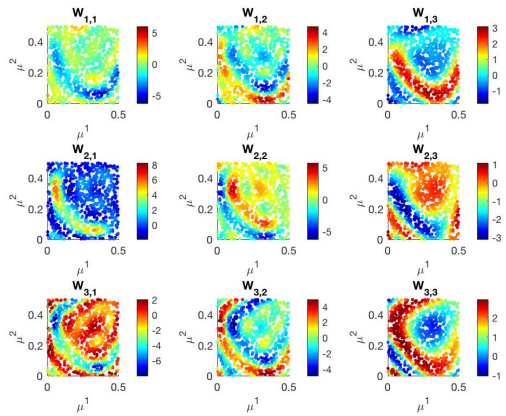


(a) Données d'entrainement (b) Points correspondants

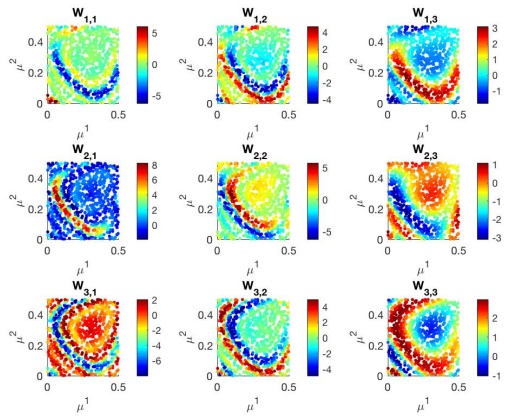
FIGURE 5.14 (Droite) Données d'entrainements pour chaque paramètre $\boldsymbol{\mu} \in \mathcal{D}_h$, (Gauche) Trois premiers points et pas de temps sélectionnés.

On remarque ainsi toute la difficulté de prédiction. En effet, bien que les données soient structurées, la fonction qui, à tout $\boldsymbol{\mu}$ associe la matrice $\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu})$ est complexe, et donc difficile à apprendre. De plus, nous n'avons comme ensemble d'entrainement que 400 évaluations de cette fonction.

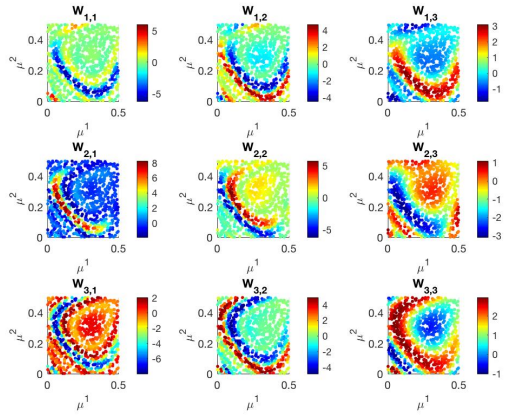
La figure ci-dessous présente les prédictions de la matrice $\tilde{\tilde{\mathbf{U}}}(\boldsymbol{\mu})$, obtenues par les différents algorithmes, sur l'ensemble de validation :



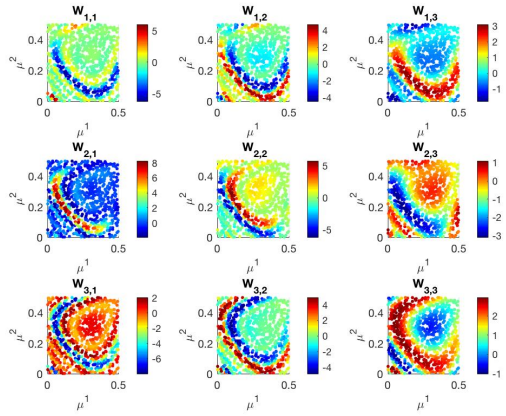
(a) KRR-kernel='rbf', score = 57.27%



(b) KRR-kernel='périodique', score = 85.29%



(c) KRR-kernel='Matérn', score = 96.53%



(d) Coefficients exacts

FIGURE 5.15 Prédictions des 6 premiers coefficients en fonction du paramètre $\boldsymbol{\mu}$ sur les données de validation, et valeur exacte (en bas à droite)

Comme attendu le problème de régression est cette fois plus complexe. Lorsque l'on utilise un noyau 'rbf', la prédiction est mauvaise. En effet en choisissant ce noyau on suppose la fonction infiniment différentiable, et au vu des données d'entrainement cela ne semble pas être le cas, la fonction que l'on souhaite apprendre semble être assez singulière. Choisir un noyau périodique revient à chercher une fonction périodique. On peut penser que nos données provenant d'un problème d'onde, une certaine périodicité se retrouve dans la fonction qui à tout $\boldsymbol{\mu}$ associe la matrice $\widetilde{\widetilde{\mathbf{U}}}(\boldsymbol{\mu})$. Les résultats sont d'ailleurs plutôt bons. Enfin le noyau de Matérn est une généralisation du noyau 'rbf', et permet de contrôler la régularité de la fonction que l'on souhaite apprendre au travers d'un hyperparamètre ν ([71]). Nous avons ici choisit $\nu = \frac{5}{2}$, ce qui revient à chercher une fonction deux fois différentiable. Ce choix semble être le bon au vu du

score de prédiction. Dans toute la suite de ce travail, nous utiliseront le noyau de Matérn.

Validation de la fonction coût apprise $\hat{J}$:

Nous pouvons à présent prédire la fonction coût, à l'aide la procédure :

(1) **Calcul de la hauteur L**

- Prédiction par KRR-'rbf' de s aux 5 points sélectionnés :

$$\hat{s}(x_{\sigma_x(k)}; \boldsymbol{\mu}) \approx s(x_{\sigma_x(k)}; \boldsymbol{\mu}).$$

- Approximation de L :

$$\hat{L}(\boldsymbol{\mu}) := \sum_{k=1}^{5} \hat{s}(x_{\sigma_x(k)}; \boldsymbol{\mu}) \sum_{i=1}^{N_x} \omega_i \, \mathbf{q}_k(x_i) \approx L(\boldsymbol{\mu}).$$

(2) **Calcul de la distance D**

- Prédiction par KRR-'Matérn' de l'onde aux points et pas de temps sélectionnés :

$$\hat{\tilde{\mathbf{U}}}(\boldsymbol{\mu}) \approx \tilde{\mathbf{U}}(\boldsymbol{\mu}).$$

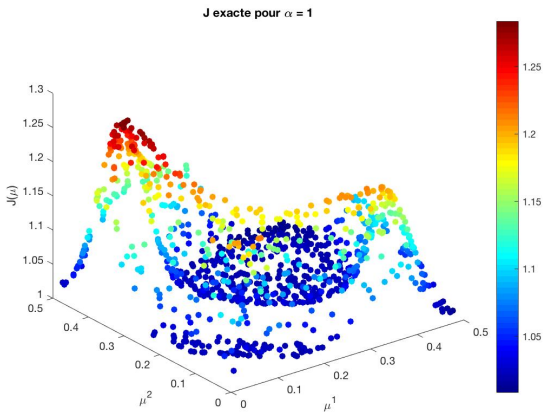
- Approximation de D :

$$\hat{D}(\boldsymbol{\mu}) := \frac{\|\hat{\tilde{\mathbf{U}}}(\boldsymbol{\mu}) - \tilde{\mathbf{U}}^f\|_F}{\|\tilde{\mathbf{U}}^f\|_F} \approx \frac{\|\|\mathbf{U}(\boldsymbol{\mu}) - \mathbf{U}^f\|_F}{\|\mathbf{U}^f\|_F}.$$

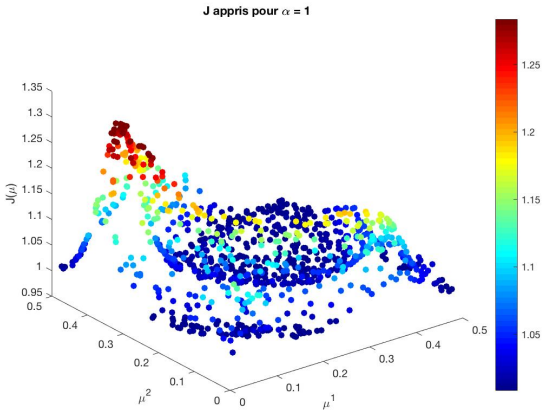
(3) **Assemblage des termes**

$$\hat{J}(\boldsymbol{\mu}) := \alpha \, \hat{D}(\boldsymbol{\mu}) + \Big(1 - \alpha\Big) \, \hat{L}(\boldsymbol{\mu}) \approx J(\boldsymbol{\mu}).$$

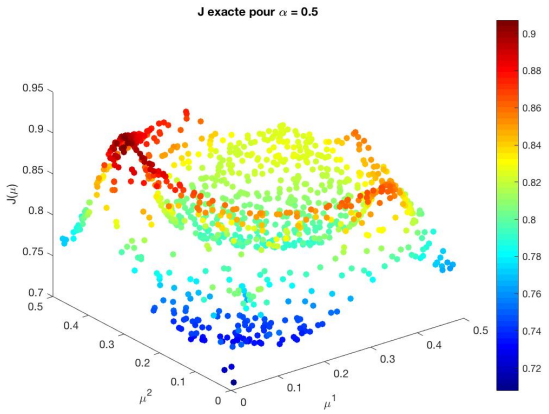
La figure ci-dessous représente la fonction coût apprise ainsi que la fonction exacte sur l'ensemble de validation.



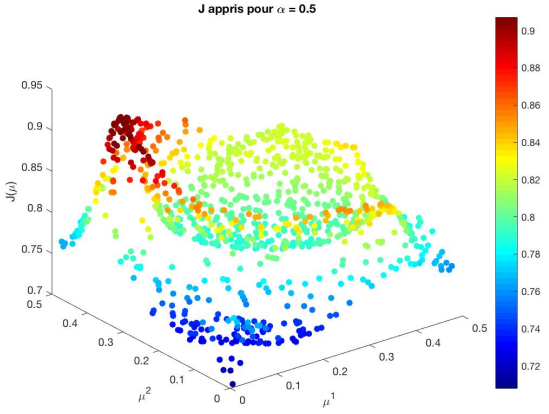
(a) fonction J exacte pour $\alpha = 1$



(b) fonction apprise $\hat{J}$ pour $\alpha = 1$



(c) fonction J exacte pour $\alpha = 0.5$



(d) fonction apprise $\hat{J}$ pour $\alpha = 0.5$

La fonction apprise $\hat{J}$ semble être plutôt fidèle à la fonction coût exacte.

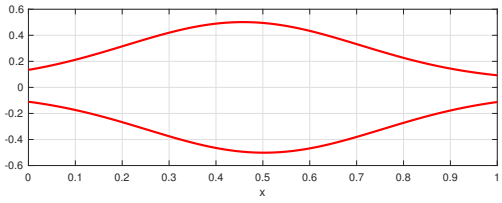
Remarque 25. *Aucune de ces procédures ne dépend de N_x et N_t, ce qui permet d'évaluer un grand nombre de fois cette fonctionnelle en peu de temps, et rend ainsi possible l'optimisation du canal. Le tableau ci-dessous présente les temps de calcul des différentes étapes.*

Étape	**Temps de calcul**
(1) Calcul de la hauteur L	0.0023 sec
(2) Calcul de la distance D	0.086 sec
(3) Assemblage des termes	$---$
Total	0.0883 sec

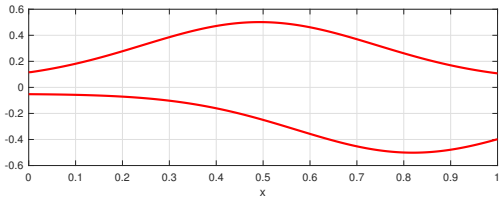
TABLE 5.5 Temps d'évaluation de J

5.4.2.7 Optimisation du canal

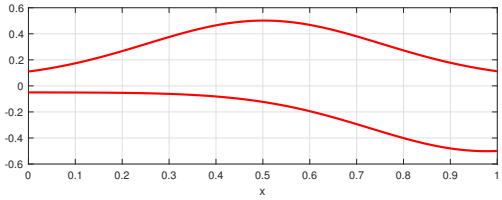
Il ne reste à présent plus qu'à optimiser la géométrie du canal, afin que l'onde qui le traverse ait un comportement proche de l'onde libre. Pour ce faire, nous utiliserons l'algorithme d'optimisation *Constrained Optimization By Linear Approximation* (COBYLA), et afin d'éviter les minima locaux, nous le réinitialiserons plusieurs fois avec différents paramètres initiaux. Nous utiliserons l'implémentation de cet algorithme de la bibliothèque Scipy ([96]). La figure ci-dessus présente les canaux correspondant aux paramètres $\boldsymbol{\mu}^*$ obtenus après minimisation.



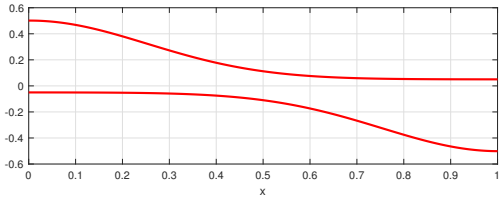
(a) Canal optimal pour $\alpha = 1$



(b) Canal optimal pour $\alpha = 0.97$



(c) Canal optimal pour $\alpha = 0.95$



(d) Canal optimal pour $\alpha = 0.5$

Lorsque $\alpha = 1$, on ne prend pas en compte la hauteur du canal. Ainsi le canal s'élargit autant que possible afin de ne pas perturber l'onde. Plus on diminue α et plus on prend en compte la hauteur globale du canal. Ce qui se traduit par un affinement du canal.

En utilisant l'approximation QUR combinée à l'algorithme KRR, nous avons pu optimiser la forme du canal pour un problème hyperbolique et dynamique. Notons que cette approche peut être appliquée dans le cas d'un modèle non-linéaire, aucune hypothèse de linéarité n'a été faite. De plus, cette approche est adaptable pour des données $\mathbf{U}(\boldsymbol{\mu})$ de très grandes dimensions : la phase *ONLINE* ne dépendant que de la dimension des espaces discrets.

Conclusion

Dans ce chapitre nous avons tout d'abord proposé une nouvelle méthode de réduction de tenseurs dans le cas de données provenant d'une EDP paramétrée. Cette réduction basée sur les méthodes HOSVD et EIM, permet d'une part d'obtenir une représentation basse dimension interprétable des données, mais également de construire des opérateurs d'interpolation en temps et en espace adaptés à tout paramètre. En utilisant l'algorithme KRR, nous avons ainsi pu apprendre un modèle réduit paramétré fidèle à un couplage EDP-EDO non-linéaire. Par ailleurs, l'approximation QUR, introduite dans ce chapitre permet de construire une représentation basse dimension correspondant à des évaluations ponctuelles de la solution. Cette réduction a ainsi permis l'optimisation de formes d'un canal.

Chapitre 6

Conclusion et ouverture

6.1 Bilan

Au **Chapitre 2**, nous avons présenté l'utilisation combinée des méthodes DMD et POD afin d'apprendre un modèle réduit précis et stable. Une analyse spectrale de ce modèle permet de valider l'identification de la dynamique. Enfin, une analyse numérique de la méthode a permis de quantifier théoriquement l'erreur d'apprentissage. Différentes extensions de cette méthode pour des EDP plus complexes ont été présentés, comme le cas d'EDP d'ordre 2 en temps. Le cas de systèmes contrôlés par commutation a pu être abordé, au **Chapitre 3**, à l'aide d'un réseau de neurones artificiels. L'utilisation de réseaux de neurones afin de modéliser une règle de contrôle est très prometteuse et cela a notamment permis d'assurer un meilleur contrôle de la température au capteur, comme présenté en **Annexe B**. Lorsque l'on souhaite prédire la solution en une région seulement, l'utilisation des méthodes DMD et EIM permet de s'abstenir d'une reconstruction coûteuse. La variante de la méthode EIM, introduite au **Chapitre 4** s'est avéré performante pour différents types d'EDP. Finalement, l'extension au cas d'EDP paramétrées, a été présentée au **Chapitre 5**. Une réduction de tenseurs basée, sur les méthodes HOSVD et EIM, a ainsi été introduite. Et a permis l'apprentissage d'un modèle réduit paramétré précis et stable. Enfin l'approximation QUR a permis de construire une représentation basse dimension interprétable. Ce qui a permis l'optimisation de formes d'un canal traversé par une onde.

6.2 Perspectives

Les travaux présentés dans cette thèse offrent de nombreuses perspectives de suites. Nous présenterons ainsi quelques travaux possibles.

6.2.1 Identification d'EDP d'évolution non-linéaire

Nous nous sommes restreint dans cette thèse à des EDP d'évolution linéaires. Lorsque les données proviennent d'une EDP d'évolution non-linéaire, certaines modifications des ensembles d'entrainement, comme celles réalisées dans le cas de l'oscillateur de Van der Pol au **Chapitre 5**, permettent d'apprendre un modèle réduit fidèle à l'EDP.

Considérons l'équation de la chaleur non-linéaire :

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \nabla \cdot \Big(\kappa(u(\boldsymbol{x},t),\boldsymbol{x})\, \nabla u(\boldsymbol{x},t) \Big),$$

avec une conductivité thermique définie par :

$$\kappa(u(\boldsymbol{x},t),\boldsymbol{x}) := \kappa_0 + \beta(u(\boldsymbol{x},t) - u_{ref}).$$

La formulation faible semi-discrétisée de ce problème est alors :

$$\begin{aligned} &\frac{1}{\delta t} \int_\Omega u^{n+1}(\boldsymbol{x})\, \psi(\boldsymbol{x}) d\boldsymbol{x} + \alpha \int_\Omega \nabla u^{n+1}(\boldsymbol{x}) \cdot \nabla \psi(\boldsymbol{x}) d\boldsymbol{x} \\ &+ \beta \int_\Omega u^{n+1}(\boldsymbol{x})\, \nabla u^{n+1}(\boldsymbol{x}) \cdot \nabla \psi(\boldsymbol{x}) d\boldsymbol{x} = \frac{1}{\delta t} \int_\Omega u^{n}(\boldsymbol{x})\, \psi(\boldsymbol{x}) d\boldsymbol{x}, \end{aligned}$$

avec $\alpha := \kappa_0 - \beta u_{ref}$. La méthode POD-Galerkin, nous donne alors le modèle réduit :

$$\left(\frac{1}{\delta t} \mathbf{M}_r + \mathbf{K}_r \right) \mathbf{a}^{n+1} + \tilde{\mathbf{K}}_r(\mathbf{a}^{n+1})\, \mathbf{a}^{n+1} = \frac{1}{\delta t} \mathbf{M}_r\, \mathbf{a}^{n},$$

avec

$$\tilde{\mathbf{K}}_{k,l}(\mathbf{a}^{n+1}) := \beta \sum_{r=1}^{K} \mathbf{a}_r^{n+1} \int_\Omega \phi_r(\boldsymbol{x})\, \nabla \phi_l(\boldsymbol{x}) \cdot \nabla \phi_k(\boldsymbol{x}) d\boldsymbol{x}$$

Ainsi afin de retrouver un système linéaire, il suffit de ne pas considérer les vecteurs $\mathbf{a}^n$ mais plutôt les vecteurs augmentés :

$$\begin{bmatrix} | \\ \mathbf{a}^n \\ | \\ \left(\mathbf{a}_1^n\right)^2 \\ \mathbf{a}_1^n\,\mathbf{a}_2^n \\ \vdots \\ \mathbf{a}_k^n\,\mathbf{a}_l^n \\ \vdots \\ \left(\mathbf{a}_K^n\right)^2 \end{bmatrix}.$$

En revanche, cela a augmenté la dimension de notre modèle réduit, on est alors passé de K coefficients à $K(K+1)$. Un moyen de limiter cette augmentation est tout d'abord de considérer seulement le produit $\mathbf{a}_k^n\,\mathbf{a}_l^n$ avec $k \leq n$. De plus, en remarquant que chaque coefficient $\mathbf{a}_k^n$ correspond à une valeur propre λ_k, on peut ne garder que les produits pour lesquels $\lambda_k\lambda_l > tol$, avec tol un seuil de tolérance.

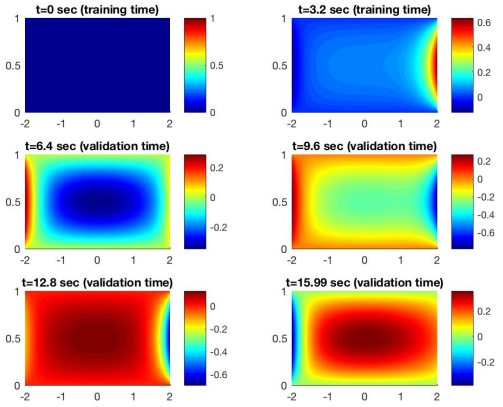
Remarque 26. *Cette extension est dans le même esprit que la méthode EDMD, à la différence qu'ici on ne considère pas des observables non-linéaires de la solution u, mais de sa représentation basse dimension. Cela permet ainsi de réduire considérablement la dimension des entrées.*

Lorsque la réduction est réalisée en échantillonnant la matrice des données, l'idée reste la même.

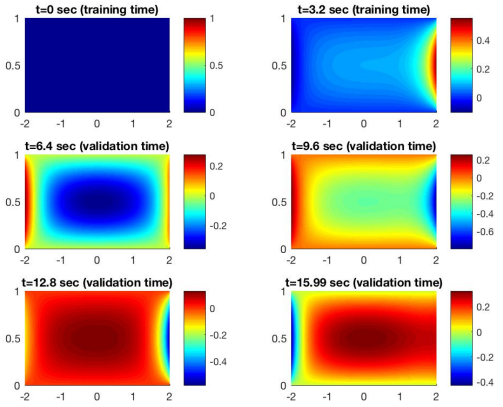
Présentons une application de cette extension. Considérons pour cela, un couplage entre l'équation de la chaleur non-linéaire et une EDO. La non-linéarité se situera au niveau de la conductivité thermique, que l'on définira par :

$$\kappa(u(\boldsymbol{x},t),\boldsymbol{x}) := 3 + 2\,u(\boldsymbol{x},t)$$

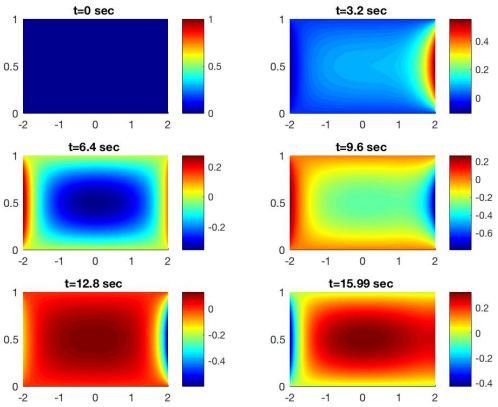
Après avoir généré les données, à l'aide du logiciel FreeFem++, nous échantillonnerons la matrice $\mathbf{U}$ à l'aide de l'algorithme *Approximation fiable rang par EIM*. On apprendra ensuite un modèle réduit linéaire comme présenté au **Chapitre 4**. Par ailleurs, on considérera des observables non-linéaire afin d'apprendre un modèle réduit non-linéaire. La figure ci-dessous représente l'évolution de la température à différents instants générée par le solveur HF, et les prédictions des modèles réduits linéaire et non-linéaire.



(a) Modèle réduit linéaire



(b) Modèle réduit non-linéaire



(c) Données HF

FIGURE 6.1 Prédictions de la température à différents instants et données HF

Bien que le modèle réduit linéaire prédise correctement le comportement global de la température, il ne parvient pas à parfaitement identifier la dynamique. Cela est encore plus marqué sur la figure ci-dessous, où l'on présente les comparaisons de l'évolution de la température aux points sélectionnés.

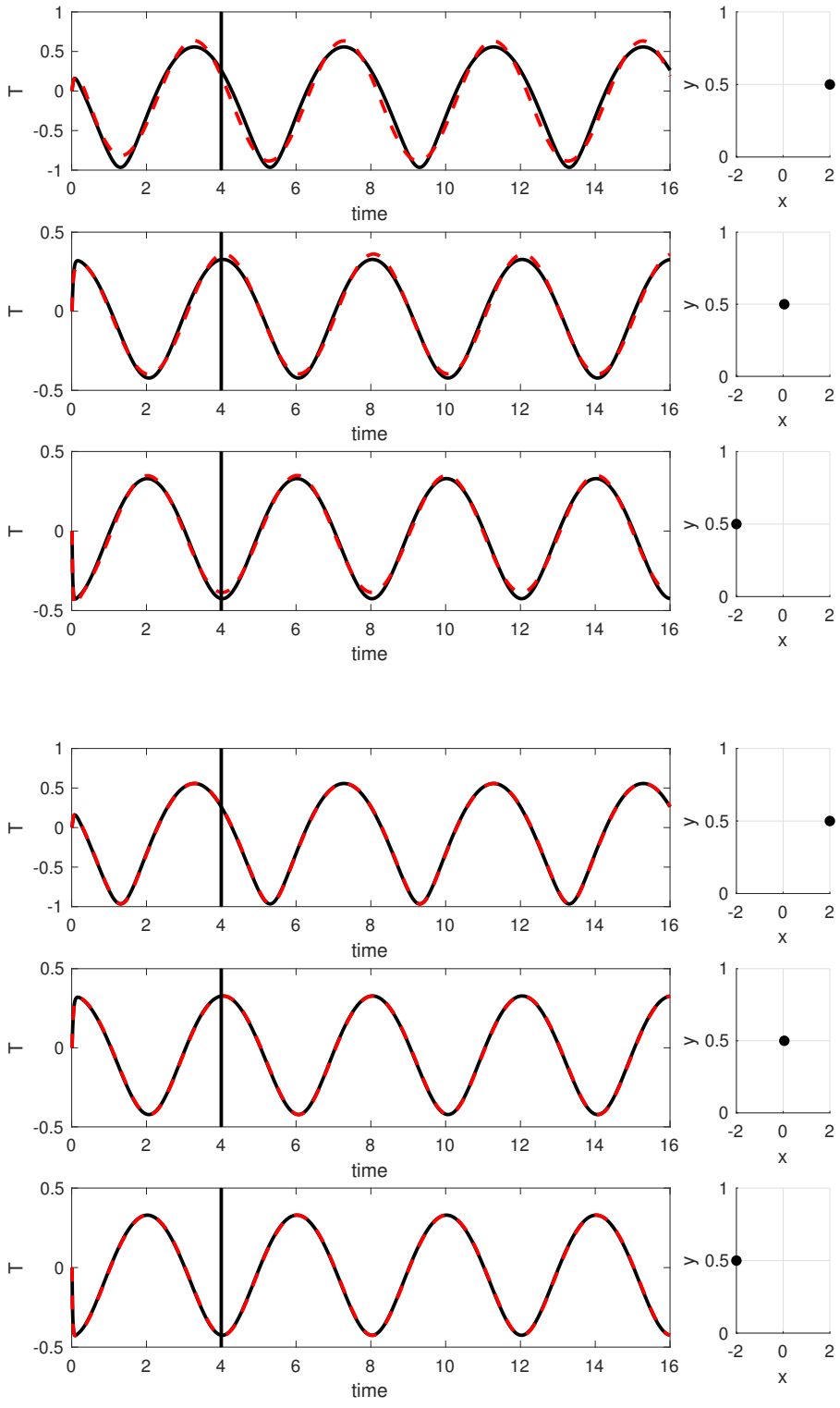


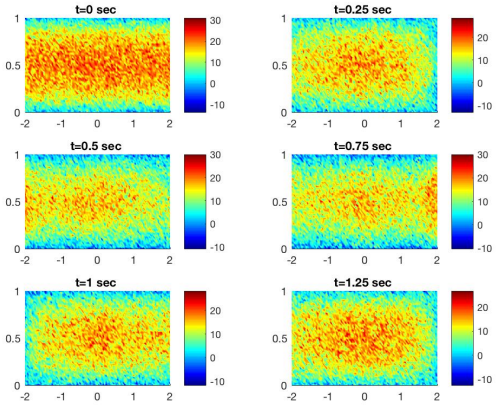
FIGURE 6.2 (Haut) Comparaison entre la prédiction de l'évolution de la température par le modèle réduit linéaire (ligne rouge pointillée) et solution HF (ligne noire continue), (Bas)Même chose pour le modèle réduit non-linéaire.

6.2.2 Données bruitées

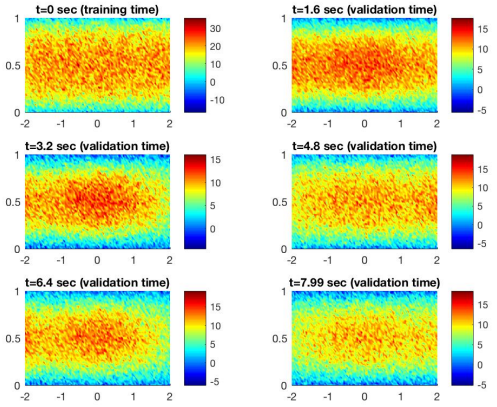
Dans toute cette thèse, nous avons uniquement considéré des données synthétiques. Lorsque les données sont issues de mesures expérimentales, un bruit blanc vient s'ajouter dû aux erreurs commises par le capteur.

Dans ce cas, la réduction par la méthode POD est bien adaptée car on cherche à minimiser en moyenne l'erreur de réduction. Ainsi, travailler avec la représentation basse dimension rend les algorithmes *Approximation faible rang par EIM* et *Réduction de tenseur par EIM*, introduit aux **Chapitres 4 et 5**, plus robustes au bruit.

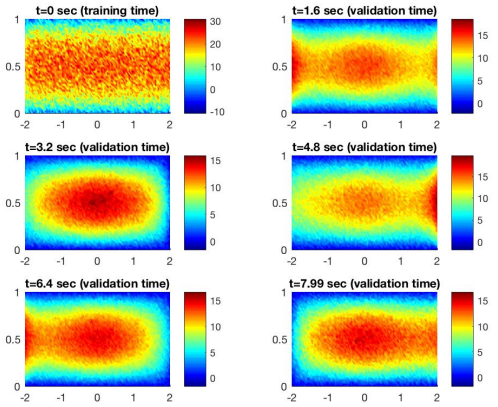
Considérons un couplage équation de la chaleur-EDO. Afin d'apprendre un modèle réduit, nous aurons accès à la matrice contenant les températures $\mathbf{U}^{noisy}$ perturbée par un bruit blanc de variance relative $0,2$. En ce qui concerne l'EDO nous supposerons avoir accès aux données non bruitées. Nous comparerons alors l'apprentissage en utilisant algorithme *Approximation faible rang par EIM*, en travaillant avec et sans la représentation basse dimension. La figure ci-dessous, présente la température générée par le solveur HF pour différents temps, la température bruitée utilisée pour l'entrainement, la prédiction du modèle réduit appris avec EIM sans réduction POD préalable, et la prédiction du modèle réduit appris avec EIM et réduction POD.



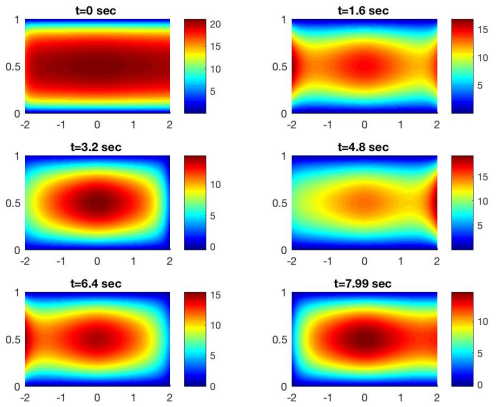
(a) Données d'entrainement bruitées



(b) Prédiction avec EIM sans réduction POD



(c) Prédiction avec EIM avec réduction POD



(d) Données HF

FIGURE 6.3 Comparaison des prédictions de la température à différents instants

On remarque donc la nécessité de réduire les données au préalable lorsque celles-ci sont bruitées. On a donc pu prédire efficacement l'évolution de la température. Par ailleurs, le modèle réduit appris semble être stable comme le montre la figure ci-dessous, où l'on représente l'évolution de la température aux trois premiers points sélectionnés. La courbe noire correspond aux données HF, la courbe pointillée rouge à la prédiction du modèle réduit, et les points bleus aux données d'entrainement.

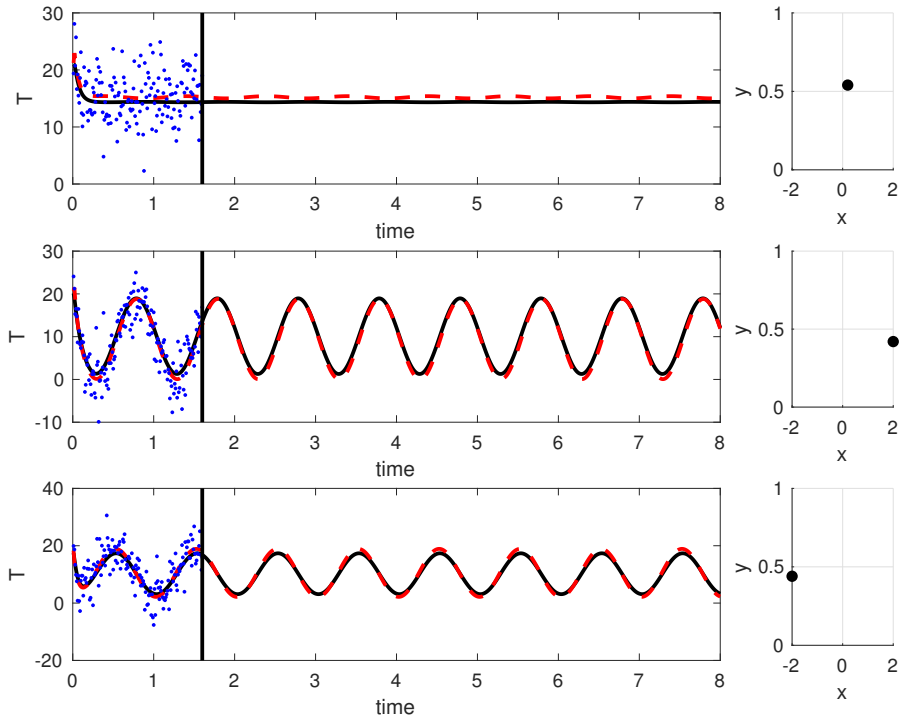


FIGURE 6.4 Évolution de la température avec en noir les données HF, en rouge la prédiction du modèle réduit, et en bleu les données d'entrainement.

Cependant dans certains cas, le bruit contenu dans les ensembles d'entrainement peut conduire à identifier une mauvaise physique. Ainsi, l'analyse spectrale présentée dans cette thèse, permet de détecter une anomalie lors de l'apprentissage. En revanche, cela ne donne pas la solution pour correctement identifier la dynamique. La variante de la méthode DMD introduite par De Vuyst & al ([181]), pourrait être une solution en imposant au modèle réduit de préserver certaines quantités. Un autre moyen d'imposer cela consiste à régulariser le problème d'apprentissage.

Deux formes de régularisation peuvent être utilisées :

- **Régularisation de Tikhonov :**
  Cela consiste à ajouter la pénalisation $\frac{\eta}{2}\|A\|_F^2$, dans notre problème d'optimisation. Étant donné que $\|A\|_F^2 = \sum_{k=1}^{K} |\lambda_k(A)|^2$, cela permet ainsi de contrôler le spectre de A.
- **Régularisation de Lasso :**
  Cela consiste à pénaliser cette fois la norme ℓ^1 de la matrice A empilée, i.e.

$\eta \sum_{k,l} |A_{k,l}|$. Cette fois-ci on cherche la matrice la plus creuse possible. Cela permet ainsi de minimiser le rôle joué par les modes hautes fréquences.

6.2.3 Identification de l'ordre de dérivée en temps à partir de données

Si l'apprentissage du modèle réduit n'a pas nécessité la connaissance du modèle, cela demande en revanche de connaître l'ordre de dérivée en temps afin d'adapter notre modèle réduit. On pourrait ainsi souhaiter automatiser cette étape en reconnaissant à partir des données le type de dérivée en temps. Un moyen d'y parvenir serait alors de sélectionner des points particuliers à l'aide de la méthode EIM, et ainsi prédire l'ordre de dérivée en fonction du profil de dynamique en ces points. Cette démarche a été utilisée par Sargsyan & al ([150]) afin de sélectionner la bonne non-linéarité de l'EDP.

6.2.4 Apprentissage en ligne

Lorsque le modèle réduit est utilisé dans un contexte de prédiction en temps réel, on souhaiterait ajuster notre prédiction en fonction des données assimilées. Pour cela, l'apprentissage par échantillonnage permet de comparer sans reconstruction les valeurs prédites aux valeurs réelles. De plus, la mise à jour des ensembles d'entrainement ne nécessite aucun calcul supplémentaire, contrairement à une réduction POD où il faudrait projeter les données assimilées. Enfin, lorsque le modèle réduit n'est plus fidèle aux données assimilées, la mise à jour du modèle réduit peut se faire en direct. Sa complexité étant en $\mathcal{O}\Big(\max(N_t, K^3)\Big)$.

Notons cependant que lorsque les ensembles d'entrainements contiennent des données correspondant à deux modèles EDP différents, le problème d'apprentissage est mal posé. Afin d'éviter l'explosion de la prédiction, on choisira d'ajouter une régularisation de Tikhonov, de manière à pénaliser le spectre de la matrice apprise (cf 6.2.2). Enfin, pour éviter de trop perturber le spectre de la matrice apprise, on pourra adapter cette régularisation en faisant dépendre l'hyper paramètre η de l'erreur de prédiction.

Reprenons le problème de diffusion avec conditions aux limites dynamiques, introduit au **Chapitre 4**, mais cette fois couplé avec l'oscillateur de Van der Pool. Le modèle

sera alors défini par :

$$\begin{cases} \partial_t u(x,t) - \kappa \partial_{xx} u(x,t) = 0, & \text{dans } [0,1] \times (0,T), \\ u(0,t) = 0, \quad u(1,t) = \theta(t), & \forall t \in (0,T), \\ \ddot{\theta}(t) = \epsilon\, \omega \Big(1 - \theta(t)^2\Big)\, \dot{\theta}(t) + \omega \theta(t), & \forall t \in (0,T) \\ u(x,0) = u_0(x), & \text{dans } [0,1] \end{cases} \tag{6.1}$$

On modifiera au cours du temps les termes sources et conditions aux limites de ce modèle. Ainsi les données avec lesquels nous avons entrainé notre modèle réduit, ne correspondent plus aux données assimilées en temps réel.

La figure ci-dessous représente l'évolution de l'erreur de prédiction. Les lignes noires verticales correspondent aux instants où le modèle EDP est modifié, tandis que la ligne verticale rouge correspond au début de l'assimilation de données. La courbe bleue correspond à l'erreur de prédiction du modèle réduit non mis à jour, et celle en rouge au modèle réduit mis à jour avec une régularisation de Tikhonov adaptative.

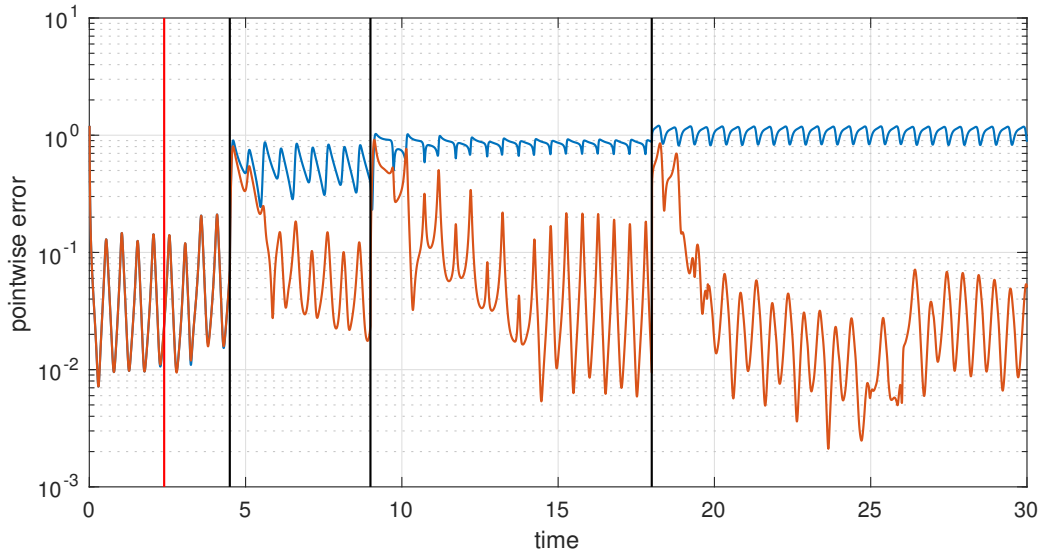


FIGURE 6.5 Évolution de l'erreur de prédiction des modèles réduits, mis à jour avec et non mis à jour, en bleu l'erreur du modèle non mis à jour, en rouge celle du modèle mis à jour.

On peut ainsi remarquer la capacité du modèle réduit à s'adapter aux données assimilées.

Présentons enfin l'évolution de la solution u aux 10 points sélectionnés ainsi que la prédiction du modèle réduit mis à jour.

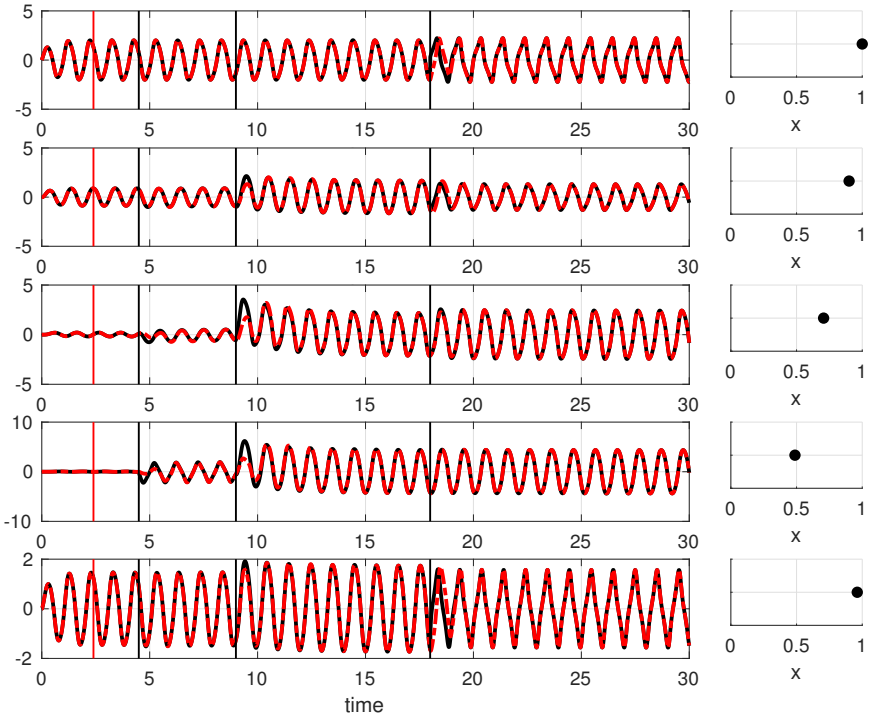


FIGURE 6.6 Évolution de la solution réelle (noire) et celle prédite par le modèle réduit mis à jour avec régularisation de Tikhonov (rouge), aux cinq premiers points sélectionnées

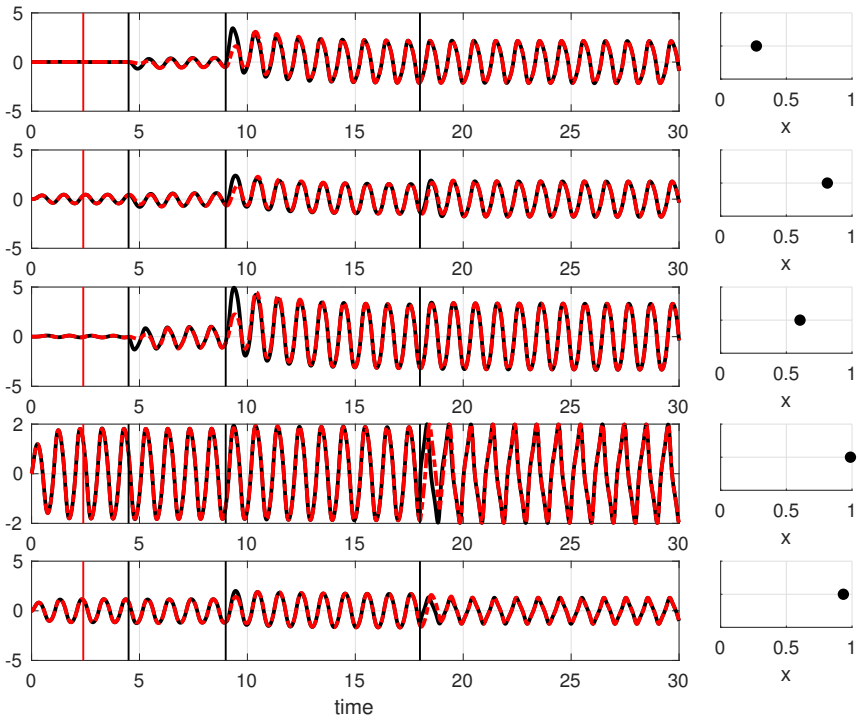


FIGURE 6.7 Évolution de la solution réelle (noire) et celle prédite par le modèle réduit mis à jour avec régularisation de Tikhonov (rouge), aux cinq derniers points sélectionnées

Bibliographie

[1] Abdel-Aal, R. and Al-Garni, A. (1997). Forecasting monthly electric energy consumption in eastern saudi arabia using univariate time-series analysis. *Energy*, 22(11) :1059–1069.

[2] Ahmad, M. A., Azuma, S.-I., Baba, I., and Sugie, T. (2014). Switching controller design for hybrid electric vehicles. *SICE Journal of Control, Measurement, and System Integration*, 7(5) :273–282.

[3] Alexanderian, A. (2015). A brief note on the karhunen-loeve expansion. *arXiv preprint arXiv :1509.07526*.

[4] Allaire, G. (2005). *Analyse numérique et optimisation : une introduction à la modélisation mathématique et à la simulation numérique*. Editions Ecole Polytechnique.

[5] Ames, W. F. (2014). *Numerical methods for partial differential equations*. Academic press.

[6] Ammar, A., Huerta, A., Chinesta, F., Cueto, E., and Leygue, A. (2014). Parametric solutions involving geometry : a step towards efficient shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268 :178–193.

[7] Artac, M., Jogan, M., and Leonardis, A. (2002). Incremental pca for on-line visual learning and recognition. In *Object recognition supported by user interaction for service robots*, volume 3, pages 781–784. IEEE.

[8] Arthur, D. and Vassilvitskii, S. (2007). k-means++ : The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

[9] Audouze, C., De Vuyst, F., and Nair, P. B. (2013). Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. *Numerical Methods for Partial Differential Equations*, 29(5) :1587–1628.

[10] Azoff, E. M. (1994). *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc.

[11] Bahouri, H., Chemin, J.-Y., and Danchin, R. (2011). *Fourier analysis and nonlinear partial differential equations*, volume 343. Springer Science & Business Media.

[12] baron Fourier, J. B. J. (1822). *Théorie analytique de la chaleur*. F. Didot.

[13] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An 'empirical interpolation'method : application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9) :667–672.

[14] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509–517.

[15] Bernardi, C. and Hecht, F. (2007). Quelques propriétés d'approximation des éléments finis de nédélec, application à l'analyse a posteriori. *Comptes Rendus Mathematique*, 344(7) :461–466.

[16] Bernardi, C. and Maday, Y. (1997). Spectral methods. *Handbook of numerical analysis*, 5 :209–485.

[17] Bhatia, N. et al. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv :1007.0085*.

[18] Binev, P., Cohen, A., Dahmen, W., DeVore, R., Petrova, G., and Wojtaszczyk, P. (2011). Convergence rates for greedy algorithms in reduced basis methods. *SIAM journal on mathematical analysis*, 43(3) :1457–1472.

[19] Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. journal of political economy 81 (may-june).

[20] Bollinger, G. (1981). Book review : Regression diagnostics : Identifying influential data and sources of collinearity.

[21] Boutsidis, C., Drineas, P., and Magdon-Ismail, M. (2014). Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2) :687–717.

[22] Boutsidis, C., Mahoney, M. W., and Drineas, P. (2009). An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977. SIAM.

[23] Boutsidis, C. and Woodruff, D. P. (2017). Optimal cur matrix decompositions. *SIAM Journal on Computing*, 46(2) :543–589.

[24] Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis : forecasting and control*. John Wiley & Sons.

[25] Brahim-Belhouari, S. and Bermak, A. (2004). Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4) :705–712.

[26] Brezis, H. (2010). *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media.

[27] Brezis, H. and Browder, F. (1998). Partial differential equations in the 20th century. *Advances in Mathematics*, 135(1) :76–144.

[28] Brezzi, F. and Fortin, M. (2012). *Mixed and hybrid finite element methods*, volume 15. Springer Science & Business Media.

[29] Buffa, A., Maday, Y., Patera, A. T., Prud'homme, C., and Turinici, G. (2012). A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM : Mathematical modelling and numerical analysis*, 46(3) :595–603.

[30] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2) :121–167.

[31] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (2006). *Spectral methods*. Springer.

[32] Carmeli, C., De Vito, E., and Toigo, A. (2005). Reproducing kernel hilbert spaces and mercer theorem. *arXiv preprint math/0504071*.

[33] Chakir, R. and Hammond, J. K. (2018). A non-intrusive reduced basis method for elastoplasticity problems in geotechnics. *Journal of Computational and Applied Mathematics*, 337 :1–17.

[34] Chakir, R., Joly, P., Maday, Y., and Parnaudeau, P. (2013). A non intrusive reduced basis method : application to computational fluid dynamics. In *2nd ECCOMAS Young Investigators Conference (YIC 2013)*.

[35] Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds. ; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3) :542–542.

[36] Chen, W., Hesthaven, J. S., Junqiang, B., Qiu, Y., Yang, Z., and Tihao, Y. (2018). Greedy nonintrusive reduced order model for fluid dynamics. *AIAA Journal*, 56(12) :4927–4943.

[37] Chinesta, F., Ammar, A., and Cueto, E. (2010). Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models. *Archives of Computational methods in Engineering*, 17(4) :327–350.

[38] Chinesta, F., Keunings, R., and Leygue, A. (2013). *The proper generalized decomposition for advanced numerical simulations : a primer*. Springer Science & Business Media.

[39] Chinesta, F., Ladeveze, P., and Cueto, E. (2011). A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4) :395.

[40] Clark, L. (2012). Google's artificial brain learns to find cat videos. *Wired UK, www. wired. com*.

[41] Cockburn, B., Karniadakis, G. E., and Shu, C.-W. (2012). *Discontinuous Galerkin methods : theory, computation and applications*, volume 11. Springer Science & Business Media.

[42] Cohen, A. and Devore, R. (2016). Kolmogorov widths under holomorphic mappings. *IMA Journal of Numerical Analysis*, 36(1) :1–12.

[43] Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1) :32–74.

[44] Courant, R. and Hilbert, D. (2008). *Methods of Mathematical Physics : Partial Differential Equations*. John Wiley & Sons.

[45] Cunningham, P. and Delany, S. J. (2007). k-nearest neighbour classifiers. *Multiple Classifier Systems*, 34(8) :1–17.

[46] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314.

[47] Dahmen, W., Plesken, C., and Welper, G. (2014). Double greedy algorithms : Reduced basis methods for transport dominated problems. *ESAIM : Mathematical Modelling and Numerical Analysis*, 48(3) :623–663.

[48] De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4) :1253–1278.

[49] De Vuyst, F. and Toumi, A. (2017). A mixed eim-svd tensor decomposition for bivariate functions. *arXiv preprint arXiv :1711.01821*.

[50] Deb, C., Zhang, F., Yang, J., Lee, S. E., and Shah, K. W. (2017). A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74 :902–924.

[51] Deshpande, A. and Vempala, S. (2006). Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 292–303. Springer.

[52] DeVore, R., Petrova, G., and Wojtaszczyk, P. (2013). Greedy algorithms for reduced bases in banach spaces. *Constructive Approximation*, 37(3) :455–466.

[53] Di Pietro, D. A. and Ern, A. (2011). *Mathematical aspects of discontinuous Galerkin methods*, volume 69. Springer Science & Business Media.

[54] Díez, P., Zlotnik, S., García-González, A., and Huerta, A. (2018). Algebraic pgd for tensor separation and compression : An algorithmic approach. *Comptes Rendus Mécanique*, 346(7) :501–514.

[55] Dolean, V., Jolivet, P., and Nataf, F. (2015). *An introduction to domain decomposition methods : algorithms, theory, and parallel implementation*, volume 144. SIAM.

[56] Drineas, P., Mahoney, M. W., and Muthukrishnan, S. (2006). Subspace sampling and relative-error matrix approximation : Column-based methods. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 316–326. Springer.

[57] Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3) :211–218.

[58] Ern, A. and Guermond, J.-L. (2002). *Eléments finis : théorie, applications, mise en oeuvre*, volume 36. Springer Science & Business Media.

[59] Euler, L. (1792). *Institutiones calculi integralis*, volume 1. Academia Imperialis Scientiarum.

[60] Evans, L. C. (2010). *Partial differential equations*. American Mathematical Society, Providence, R.I.

[61] Eymard, R., Gallouët, T., and Herbin, R. (2000). Finite volume methods. *Handbook of numerical analysis*, 7 :713–1018.

[62] Fahlaoui, T. and Vuyst, F. D. (2019). Nonintrusive data-based learning of a switched control heating system using pod, dmd and ann. *Comptes Rendus Mécanique*, 347(11) :793 – 805. Data-Based Engineering Science and Technology.

[63] Farahat, A. K., Elgohary, A., Ghodsi, A., and Kamel, M. S. (2015). Greedy column subset selection for large-scale data sets. *Knowledge and Information Systems*, 45(1) :1–34.

[64] Farrar, D. E. and Glauber, R. R. (1967). Multicollinearity in regression analysis : the problem revisited. *The Review of Economic and Statistics*, pages 92–107.

[65] Ferrara, A. and Paderno, J. (2006). Application of switching control for automatic pre-crash collision avoidance in cars. *Nonlinear Dynamics*, 46(3) :307–321.

[66] Fortin, M. and Brezzi, F. (1991). *Mixed and hybrid finite element methods*, volume 734. New York : Springer-Verlag.

[67] Freno, B. A. and Carlberg, K. T. (2019). Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations. *Computer Methods in Applied Mechanics and Engineering*, 348 :250–296.

[68] Friedman, A. (2008). *Partial differential equations of parabolic type*. Courier Dover Publications.

[69] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

[70] Gallagher, I. (2010). Autour des équations de navier-stokes. *Images des Mathématiques*, pages http–images.

[71] Genton, M. G. (2001). Classes of kernels for machine learning : a statistics perspective. *Journal of machine learning research*, 2(Dec) :299–312.

[72] Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.".

[73] Gilbarg, D. and Trudinger, N. S. (2015). *Elliptic partial differential equations of second order*. springer.

[74] Godlewski, E. and Raviart, P.-A. (1991). *Hyperbolic systems of conservation laws*. Ellipses.

[75] Godlewski, E. and Raviart, P.-A. (2013). *Numerical approximation of hyperbolic systems of conservation laws*, volume 118. Springer Science & Business Media.

[76] Gong, H., Maday, Y., Mula, O., and Taddei, T. (2019). Pbdw method for state estimation : error analysis for noisy data and nonlinear formulation. *arXiv preprint arXiv :1906.00810*.

[77] González, D., Masson, F., Poulhaon, F., Leygue, A., Cueto, E., and Chinesta, F. (2012). Proper generalized decomposition based dynamic data driven inverse identification. *Mathematics and Computers in Simulation*, 82(9) :1677–1695.

[78] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[79] Gottlieb, S. and Shu, C.-W. (1998). Total variation diminishing runge-kutta schemes. *Mathematics of computation of the American Mathematical Society*, 67(221) :73–85.

[80] Granger, C. W. J. and Newbold, P. (2014). *Forecasting economic time series*. Academic Press.

[81] Guo, M. and Hesthaven, J. S. (2019). Data-driven reduced order modeling for time-dependent problems. *Computer methods in applied mechanics and engineering*, 345 :75–99.

[82] Guruswami, V. and Sinop, A. K. (2012). Optimal column-based low-rank matrix reconstruction. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1207–1214. SIAM.

[83] Hammond, J. K., Chakir, R., Bourquin, F., and Maday, Y. (2019). Pbdw : A non-intrusive reduced basis data assimilation method and its application to an urban dispersion modeling framework. *Applied Mathematical Modelling*, 76 :1–25.

[84] Hecht, F. (2012). New development in freefem++. *Journal of numerical mathematics*, 20(3-4) :251–266.

[85] Hesthaven, J. S., Rozza, G., Stamm, B., et al. (2016a). *Certified reduced basis methods for parametrized partial differential equations*. Springer.

[86] Hesthaven, J. S., Rozza, G., Stamm, B., et al. (2016b). *Certified reduced basis methods for parametrized partial differential equations*. Springer.

[87] Hesthaven, J. S., Stamm, B., and Zhang, S. (2014). Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods. *ESAIM : Mathematical Modelling and Numerical Analysis*, 48(1) :259–283.

[88] Hesthaven, J. S. and Ubbiali, S. (2018). Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363 :55–78.

[89] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8) :1735–1780.

[90] Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1) :66–73.

[91] Jain, A. K. (2010). Data clustering : 50 years beyond k-means. *Pattern recognition letters*, 31(8) :651–666.

[92] Jain, A. K., Dubes, R. C., et al. (1988). *Algorithms for clustering data*, volume 6. Prentice hall Englewood Cliffs, NJ.

[93] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3) :264–323.

[94] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

[95] Jolliffe, I. (1986). *Principal Component Analysis*. Springer Verlag.

[96] Jones, E., Oliphant, T., Peterson, P., et al. (2001). Scipy : Open source scientific tools for python.

[97] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning : A survey. *Journal of artificial intelligence research*, 4 :237–285.

[98] Kianfar, J. and Edara, P. (2010). Optimizing freeway traffic sensor locations by clustering global-positioning-system-derived speed patterns. *IEEE Transactions on Intelligent Transportation Systems*, 11(3) :738–747.

[99] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning : A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160 :3–24.

[100] Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes : Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb) :235–284.

[101] Kutz, J. N., Brunton, S. L., Brunton, B. W., and Proctor, J. L. (2016). *Dynamic mode decomposition : data-driven modeling of complex systems*. SIAM.

[102] Ladevèze, P. (1985). Sur une famille d'algorithmes en mécanique des structures. *Comptes-rendus des séances de l'Académie des sciences. Série 2, Mécanique-physique, chimie, sciences de l'univers, sciences de la terre*, 300(2) :41–44.

[103] Ladevèze, P., Passieux, J.-C., and Néron, D. (2010). The latin multiscale computational method and the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21-22) :1287–1296.

[104] Lassila, T. and Rozza, G. (2010). Parametric free-form shape design with pde models and reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24) :1583–1592.

[105] Lax, P. D. (1957). Hyperbolic systems of conservation laws ii. *Communications on pure and applied mathematics*, 10(4) :537–566.

[106] Le Coënt, A. (2017). *Guaranteed control synthesis for switched space-time dynamical systems*. PhD thesis.

[107] Le Coënt, A., Fribourg, L., Markey, N., De Vuyst, F., and Chamoin, L. (2016). Distributed synthesis of state-dependent switching control. In *International Workshop on Reachability Problems*, pages 119–133. Springer.

[108] Le Guennec, Y., Brunet, J.-P., Daim, F.-Z., Chau, M., and Tourbier, Y. (2018). A parametric and non-intrusive reduced order model of car crash simulation. *Computer Methods in Applied Mechanics and Engineering*, 338 :186–207.

[109] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–48. Springer.

[110] Lee, G., Marinho, Z., Johnson, A. M., Gordon, G. J., Srinivasa, S. S., and Mason, M. T. (2017). Unsupervised learning for nonlinear piecewise smooth hybrid systems. *arXiv preprint arXiv :1710.00440*.

[111] LeVeque, R. J. et al. (2002). *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press.

[112] Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3) :18–22.

[113] Liberzon, D. (2003). *Switching in systems and control*. Springer Science & Business Media.

[114] Limache, A. C. and Idelsohn, S. R. (2007). On the development of finite volume methods for computational solid mechanics.

[115] Lin, H. and Antsaklis, P. J. (2009). Stability and stabilizability of switched linear systems : a survey of recent results. *IEEE Transactions on Automatic control*, 54(2) :308–322.

[116] Lions, J.-L. (1981). Some methods in the mathematical analysis of systems and their control(book). *Beijing, Science Press*.

[117] Maday, Y., Nguyen, N. C., Patera, A. T., and Pau, S. H. (2009). A general multipurpose interpolation procedure : the magic points. *Communications on Pure and Applied Analysis*, 8 :383.

[118] Maday, Y., Patera, A. T., Penn, J. D., and Yano, M. (2015). A parameterized-background data-weak approach to variational data assimilation : formulation, analysis, and application to acoustics. *International Journal for Numerical Methods in Engineering*, 102(5) :933–965.

[119] Magoulès, F. and Roux, F.-X. (2017). *Calcul scientifique parallèle-2e éd. : Cours, exemples avec openMP et MPI, exercices corrigés*. Dunod.

[120] Mahoney, M. W. and Drineas, P. (2009). Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3) :697–702.

[121] Manzoni, A., Quarteroni, A., and Rozza, G. (2012). Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5) :646–670.

[122] Maulik, R., Mohan, A., Lusch, B., Madireddy, S., and Balaprakash, P. (2019). Time-series learning of latent-space dynamics for reduced-order model closure. *arXiv preprint arXiv :1906.07815*.

[123] McCarthy, J. and Feigenbaum, E. A. (1990). In memoriam : Arthur samuel : Pioneer in machine learning. *AI Magazine*, 11(3) :10–10.

[124] Mercer, J. (1909). Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458) :415–446.

[125] Nédélec, J.-C. (1980). Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(3) :315–341.

[126] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering : Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

[127] Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press USA.

[128] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

[129] Novikov, A. (2019). PyClustering : Data mining library. *Journal of Open Source Software*, 4(36) :1230.

[130] Ochi, S. C. and Williams Jr, J. H. (1987). One-dimensional wave propagation in rods of variable cross section : A wkbj solution.

[131] Omohundro, S. M. (1989). *Five balltree construction algorithms*. International Computer Science Institute Berkeley.

[132] Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2) :3336–3341.

[133] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

[134] Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2) :263–269.

[135] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn : Machine learning in python. *Journal of machine learning research*, 12(Oct) :2825–2830.

[136] Peitz, S. and Klus, S. (2017). Koopman operator-based model reduction for switched-system control of pdes. *arXiv preprint arXiv :1710.06759*.

[137] Proctor, J. L., Brunton, S. L., and Kutz, J. N. (2016). Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1) :142–161.

[138] Prud'Homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G. (2001). Reliable Real-Time Solution of Parametrized Partial Differential Equations : Reduced-Basis Output Bound Methods. *Journal of Fluids Engineering*, 124(1) :70–80.

[139] Quarteroni, A., Rozza, G., and Manzoni, A. (2011). Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1) :3.

[140] Quarteroni, A. and Valli, A. (1999). *Domain decomposition methods for partial differential equations*. Number BOOK. Oxford University Press.

[141] Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

[142] Rasmussen, C. E. and Ghahramani, Z. (2002). Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems*, pages 881–888.

[143] Raviart, P.-A. and Thomas, J.-M. (1977). Primal hybrid finite element methods for 2nd order elliptic equations. *Mathematics of computation*, 31(138) :391–413.

[144] Rozza, G., Huynh, D. B. P., and Patera, A. T. (2007). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3) :1.

[145] Rozza, G., Lassila, T., and Manzoni, A. (2011). Reduced basis approximation for shape optimization in thermal flows with a parametrized polynomial geometric map. In *Spectral and high order methods for partial differential equations*, pages 307–315. Springer.

[146] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088) :533.

[147] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization : Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin.

[148] Samuel, A. L. (1962). Artificial intelligence : a frontier of automation. *The Annals of the American Academy of Political and Social Science*, 340(1) :10–20.

[149] Santo, N. D., Deparis, S., and Pegolotti, L. (2019). Data driven approximation of parametrized pdes by reduced basis and neural networks. *arXiv preprint arXiv :1904.01514*.

[150] Sargsyan, S., Brunton, S. L., and Kutz, J. N. (2015). Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries. *Physical Review E*, 92(3) :033304.

[151] Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656 :5–28.

[152] Schmidt, E. (1924). Über die anwendung der differenzenrechnung auf technische anheiz-und abkühlungsprobleme. In *Beiträge zur Technischen Mechanik und Technischen Physik*, pages 179–189. Springer.

[153] Schölkopf, B., Burges, C. J., Smola, A. J., et al. (1999). *Advances in kernel methods : support vector learning*. MIT press.

[154] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5) :1299–1319.

[155] Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT press.

[156] Semaan, R. (2017). Optimal sensor placement using machine learning. *Computers and Fluids*, 159 :167 – 176.

[157] Serre, D. (1996). Systèmes de lois de conservation i(hyperbolicité, entropies, ondes de choc). *Fondations Paris. 1996*.

[158] Sirovich, L. and Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3) :519–524.

[159] Slinker, B. and Glantz, S. (1985). Multiple regression for physiological data analysis : the problem of multicollinearity. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 249(1) :R1–R12.

[160] Smith, G. D., Smith, G. D., and Smith, G. D. S. (1985). *Numerical solution of partial differential equations : finite difference methods*. Oxford university press.

[161] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3) :199–222.

[162] Song, Z., Woodruff, D. P., and Zhong, P. (2019). Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2772–2789. Society for Industrial and Applied Mathematics.

[163] Speiser, D. and Williams, K. (2008). *Discovering the principles of mechanics 1600-1800 : essays by David Speiser*, volume 1. Springer Science & Business Media.

[164] Stewart, I. (2012). *Seventeen equations that changed the world*. Profile.

[165] Strang, G. and Fix, G. J. (1973). *An analysis of the finite element method*, volume 212. Prentice-hall Englewood Cliffs, NJ.

[166] Strikwerda, J. C. (2004). *Finite difference schemes and partial differential equations*, volume 88. Siam.

[167] Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep neuroevolution : Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv :1712.06567*.

[168] Suganuma, M., Shirakawa, S., and Nagao, T. (2017). A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497–504. ACM.

[169] Süli, E. (1991). Convergence of finite volume schemes for poisson's equation on nonuniform meshes. *SIAM Journal on Numerical Analysis*, 28(5) :1419–1430.

[170] Sun, Z. (2006). *Switched linear systems : control and design*. Springer Science & Business Media.

[171] Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 2. MIT press Cambridge.

[172] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K. (2019). Projection-based model reduction : Formulations for physics-based machine learning. *Computers & Fluids*, 179 :704–717.

[173] Szabó, B. and Babuška, I. (1991). *Finite element analysis*. John Wiley & Sons.

[174] Thomas, J. W. (2013). *Numerical partial differential equations : finite difference methods*, volume 22. Springer Science & Business Media.

[175] Toro, E. F. (2013). *Riemann solvers and numerical methods for fluid dynamics : a practical introduction*. Springer Science & Business Media.

[176] Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., and Kutz, J. N. (2013). On dynamic mode decomposition : theory and applications. *arXiv preprint arXiv :1312.0041*.

[177] Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3) :279–311.

[178] Volkwein, S. (2013). Proper orthogonal decomposition : Theory and reduced-order modelling. *Lecture Notes, University of Konstanz*, 4(4).

[179] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416.

[180] Vovk, V. (2013). Kernel ridge regression. In *Empirical inference*, pages 105–116. Springer.

[181] Vuyst, F. D. and Villon, P. (2019). Identification of nonlinear dynamical system equations using dynamic mode decomposition under invariant quantity constraints. *Comptes Rendus Mécanique*, 347(11) :882 – 890. Data-Based Engineering Science and Technology.

[182] Walsh, J. B. (1986). An introduction to stochastic partial differential equations. In *École d'Été de Probabilités de Saint Flour XIV-1984*, pages 265–439. Springer.

[183] Warmuth, M. K. and Kuzmin, D. (2007). Randomized pca algorithms with regret bounds that are logarithmic in the dimension. In *Advances in neural information processing systems*, pages 1481–1488.

[184] Williams, M. O., Kevrekidis, I. G., and Rowley, C. W. (2015). A data–driven approximation of the koopman operator : Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6) :1307–1346.

[185] Xiao, D., Fang, F., Buchan, A., Pain, C., Navon, I., and Muggeridge, A. (2015). Non-intrusive reduced order modelling of the navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293 :522–541.

[186] Xu, J. and Zou, Q. (2009). Analysis of linear and quadratic simplicial finite volume methods for elliptic equations. *Numerische Mathematik*, 111(3) :469–492.

[187] Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1) :1–130.

[188] Zienkiewicz, O. C., Taylor, R. L., Nithiarasu, P., and Zhu, J. (1977). *The finite element method*, volume 3. McGraw-hill London.

Annexe A

Analyse numérique de l'identification d'une EDP d'évolution non homogène

À la **section 2.3.4** du **Chapitre 2**, nous avions énoncé le **théorème 2** fournissant une estimation a priori de l'erreur commise par le modèle réduit appris, sous l'hypothèse que les données générées par le solveur haute-fidélité satisfaisaient la relation

$$\mathcal{A}\,\mathcal{X} = \mathcal{Y}.$$

Dans cette annexe, nous étendrons ce résultat au cas non homogène, introduit à la **section 2.4** , où les données générées par le solveur haute-fidélité satisfont la relation

$$\mathcal{A}\,\mathcal{X} + \mathcal{B}Z = \mathcal{Y},$$

où l'écriture $\mathcal{B}Z$ correspond à la décomposition en somme tensorisée espace-temps du terme source, et où Z contient l'historique des valeurs de $\boldsymbol{\theta}(t)$, i.e.

$$Z = \begin{bmatrix} | & | & & | \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-1} \\ | & | & & | \end{bmatrix},$$

et $\mathcal{B}$ les fonctions $\boldsymbol{g}$. Soient $\tilde{A}$ la représentation basse dimension de $\mathcal{A}$, i.e. $\tilde{A} := \boldsymbol{\Phi}_r^T \mathbf{M} \mathcal{A} \boldsymbol{\Phi}_r$, et $\tilde{B} := \boldsymbol{\Phi}_r^T \mathbf{M} \mathcal{B}$ la représentation basse dimension de $\mathcal{B}$. On a alors :

Corolaire 1. *Sous l'hypothèse*

$$\mathcal{A}\,\mathcal{X} + \mathcal{B}Z = \mathcal{Y},$$

on a

$$\|\tilde{A}\,X + \tilde{B}Z - Y\|_F \leq \Big(\sum_{k>K} \lambda_k\Big)^{\frac{1}{2}} \rho(\mathcal{A}).$$

Ce résultat est une conséquence directe de la **Proposition 1**. Là encore l'identification sera au mieux de l'ordre de l'erreur de réduction. Notons que ce résultat est bien plus général que dans le cas d'un terme source tensorisé, car ici $\tilde{B}$ est la projection de l'excitation exacte du système.

À présent supposons que le terme source s'écrit comme une somme tensorisée et que la partie temporelle est solution de l'EDO

$$\dot{\boldsymbol{\theta}}(t) = C\,\boldsymbol{\theta}(t),$$

Soient $\hat{A}, \hat{B}$, et $\hat{C}$ les trois matrices apprises par l'algorithme, qui on le rappelle sont définies par :

$$\begin{cases} \hat{B} = \Big(YX^\dagger XZ^\dagger - YZ^\dagger\Big)\Big(ZX^\dagger XZ^\dagger - \mathbf{I}_M\Big)^{-1}, \\ \hat{A} = YX^\dagger - \hat{B}ZX^\dagger, \\ \hat{C} = Z'Z^\dagger. \end{cases}$$

On a alors

Proposition 5.

$$\|\hat{A}\,X + \hat{B}\,Z - Y\|_F \leq \Big(\sum_{k>K} \lambda_k\Big)^{\frac{1}{2}} \rho(\mathcal{A}) \left(1 + \kappa_F(X)\Big(1 + C(X,Z)\Big(1 + \kappa_F(Z)\Big)\Big)\right),$$

$$\|\hat{C}\,Z - Z'\|_F \leq o(\delta t^2),$$

avec $C(X,Z) := \|\mathbf{I}_{N_t-1} - X^\dagger X\|_F \|\Big(\mathbf{I}_{N_t-1} - X^\dagger XZ^\dagger Z\Big)^{-1}\|_F$*, et* δt *le pas de temps.*

Démonstration. La majoration de $\|\hat{C}\,Z - Z'\|_F$ découle d'une approximation de l'EDO par une méthode des différences finies, et par le fait que s'il existe une matrice C_h telle que $Z' = C_h\,Z$, alors $\hat{C} := Z'Z^\dagger = C_h$. On a alors

$$\|\hat{C}\,Z - Z'\|_F \leq o(\delta t^2),$$

avec δt le pas de temps choisi. À présent majorons $\|\hat{A}\,X + \hat{B}\,Z - Y\|_F$. Reprenons le système matriciel (3.2) satisfait par les matrices $\hat{A}$ et $\hat{B}$ en multipliant la première ligne par X et la seconde par Z, à droite, on a alors :

$$\begin{cases} \hat{A}X + \hat{B}ZX^{\dagger}X = YX^{\dagger}X, \\ \hat{A}XZ^{\dagger}Z + \hat{B}Z = YZ^{\dagger}Z, \end{cases} \tag{A.1}$$

Par ailleurs, on a en utilisant le corolaire ci-dessous ainsi que la **Proposition 1**,

$$\tilde{A}X + \tilde{B}Z = Y - \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}},$$

en multipliant par $X^{\dagger}X$, à droite, on obtient

$$\tilde{A}X = YX^{\dagger}X - \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}X^{\dagger}X - \tilde{B}ZX^{\dagger}X,$$

et en multipliant cette fois par $Z^{\dagger}Z$, à droite, on obtient

$$\tilde{B}Z = YZ^{\dagger}Z - \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}Z^{\dagger}Z - \tilde{A}XZ^{\dagger}Z.$$

À présent en utilisant le système matriciel (A.1), et les deux équations ci-dessus on peut en déduire le système matriciel suivant

$$\begin{cases} \hat{A}X - \tilde{A}X + \Big(\hat{B}Z - \tilde{B}Z\Big)X^{\dagger}X = \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}X^{\dagger}X, \\ \hat{B}Z - \tilde{B}Z + \Big(\hat{A}X - \tilde{A}X\Big)Z^{\dagger}Z = \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}Z^{\dagger}Z, \end{cases} \tag{A.2}$$

ce qui nous donne

$$\begin{cases} \hat{A}X - \tilde{A}X = \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}\Big(\mathbf{I}_{N_t-1} + \Big(\mathbf{I}_{N_t-1} - X^{\dagger}X\Big)Z^{\dagger}Z\Big(\mathbf{I}_{N_t-1} - X^{\dagger}XZ^{\dagger}Z\Big)^{-1}\Big)X^{\dagger}X, \\ \hat{B}Z - \tilde{B}Z = \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}\Big(\mathbf{I}_{N_t-1} - X^{\dagger}X\Big)Z^{\dagger}Z\Big(\mathbf{I}_{N_t-1} - X^{\dagger}XZ^{\dagger}Z\Big)^{-1}, \end{cases} \tag{A.3}$$

le résultat s'obtient en remarquant que

$$\hat{A}X + \hat{B}Z - Y = \Big(\hat{A}X - \tilde{A}X\Big) + \Big(\hat{B}Z - \tilde{B}Z\Big) - \mathbf{\Phi}_r^T \mathbf{M}_{\mathcal{A}}\,\mathcal{E}_{\mathcal{X}}.$$

□

On peut alors donner le théorème suivant, qui quantifie la solution prédite par le modèle appris,

Théorème 5. *Soit $u_h(\boldsymbol{x}, t^n)$ la solution du modèle* (2.17)*, calculée par le solveur haute-fidélité, et $\hat{u}_h(\boldsymbol{x}, t^n)$ la solution calculée par le modèle réduit. On a alors*

$$\Big(\sum_{n=1}^{N_t}\|u_h(\cdot,t^n)-\hat{u}_h(\cdot,t^n)\|^2_{L^2(\Omega)}\Big)^{\frac{1}{2}} \leq \Big(\sum_{k>K}\lambda_k\Big)^{\frac{1}{2}}\Bigg\{1+\rho(\mathcal{A})\Bigg(1+\kappa_F(X)\Big(1+C(X,Z)\Big(1+\kappa_F(Z)\Big)\Big)\Bigg) \Big(\sum_{m=1}^{N_t-1}\|\hat{A}^{m-1}\|_F^2\Big)^{\frac{1}{2}}\Bigg\}+\|\hat{B}\|_F\Big(\sum_{m=1}^{N_t-1}\|\hat{C}^{m-1}\|_F^2\Big)^{\frac{1}{2}}o(\delta t^2)$$

Démonstration. On procède de la même manière que pour la démonstration du **Théorème 2**. Il suffit donc d'exprimer l'erreur $\|Y-\hat{Y}\|_F$ en fonction des erreurs $\|Y-\hat{A}X-\hat{B}Z\|_F$ et $\|Z'-\hat{C}Z\|_F$. On peut établir que

$$Y^n-\hat{Y}^n=\sum_{m=1}^{n}\hat{A}^{m-1}\Big(Y^{n-m+1}-\hat{A}X^{n-m+1}-\hat{B}Z^{n-m+1}\Big)+\hat{B}\sum_{m=1}^{n-1}\hat{C}^{m-1}\Big((Z')^{n-m}-\hat{C}Z^{n-m}\Big).$$

on a alors

$$\sum_{n=1}^{N_t-1}\|Y^n-\hat{Y}^n\|_F^2 \leq \sum_{n=1}^{N_t-1}\sum_{m=1}^{n}\|\hat{A}^{m-1}\Big(Y^{n-m+1}-\hat{A}X^{n-m+1}-\hat{B}Z^{n-m+1}\Big)\|_F^2 + \|\hat{B}\|_F^2\sum_{n=1}^{N_t-1}\sum_{m=1}^{n}\|\hat{C}^{m-1}\Big((Z')^{n-m}-\hat{C}Z^{n-m}\Big)\|_F^2$$

On a que

$$\sum_{n=1}^{N_t-1}\|Y^n-\hat{Y}^n\|_F^2 \leq \Big(\sum_{m=1}^{N_t-1}\|\hat{A}^{m-1}\|_F^2\Big)\|\hat{A}\,X+\hat{B}\,Z-Y\|_F^2+\|\hat{B}\|_F^2\Big(\sum_{m=1}^{N_t-1}\|\hat{C}^{m-1}\|_F^2\Big)\|\hat{C}\,Z-Z'\|_F^2.$$

Et en utilisant la **Proposition 5**, on a finalement,

$$\Big(\sum_{n=1}^{N_t-1}\|Y^n-\hat{Y}^n\|_F^2\Big)^{\frac{1}{2}} \leq \Big(\sum_{k>K}\lambda_k\Big)^{\frac{1}{2}}\rho(\mathcal{A})\Bigg(1+\kappa_F(X)\Big(1+C(X,Z)\Big(1+\kappa_F(Z)\Big)\Big)\Bigg)\Big(\sum_{m=1}^{N_t-1}\|\hat{A}^{m-1}\|_F^2\Big)^{\frac{1}{2}} + \|\hat{B}\|_F\Big(\sum_{m=1}^{N_t-1}\|\hat{C}^{m-1}\|_F^2\Big)^{\frac{1}{2}}o(\delta t^2),$$

ce qui conclut la preuve. □

Le premier terme du membre de droite est similaire à celui exprimé au **Théorème 2**. Le second terme quant à lui est dû au fait que nous supposons que le modèle satisfait par $\boldsymbol{\theta}(t)$ est continu, or nous n'avons qu'un nombre fini d'évaluations de cette

fonction. En pratique, le pas de temps δt est choisi petit et donc le premier terme de la majoration est dominant.

Donnons à présent un résultat sur le rayon spectral de la matrice $\hat{A}$ apprise.

Théorème 6. *Soit $\hat{A}$ la matrice apprise par l'algorithme, on a alors*

$$\rho(\hat{A}) \leq \rho(\mathcal{A})\Big(1 + \big(\sum_{k>K} \lambda_k\big)^{\frac{1}{2}} \big(\sigma_{min}(X)\big)^{-1} C'(X,Z)\Big),$$

avec $C'(X,Z) = \|\mathbf{I}_{N_t-1} - Z^\dagger Z\|_2 \| \Big(\mathbf{I}_K - XZ^\dagger ZX^\dagger\Big)^{-1} \|_2$.

Démonstration. On a précédemment établi que

$$\tilde{A}X + \tilde{B}Z - Y = -\mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\mathcal{E}_{\mathcal{X}}.$$

En utilisant la définition (3.2) de $\hat{A}$ et $\hat{B}$, on a

$$\begin{cases} \hat{A} - \tilde{A} + \big(\hat{B} - \tilde{B}\big) ZX^\dagger = \mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\, \mathcal{E}_{\mathcal{X}} X^\dagger, \\ \big(\hat{A} - \tilde{A}\big) XZ^\dagger + \hat{B} - \tilde{B} = \mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\, \mathcal{E}_{\mathcal{X}} Z^\dagger, \end{cases} \tag{A.4}$$

d'où l'on retrouve :

$$\hat{A} = \tilde{A} + \mathbf{\Phi}_r^T \mathbf{M}\mathcal{A}\, \mathcal{E}_{\mathcal{X}} \Big(\mathbf{I}_{N_t-1} - Z^\dagger Z\Big) X^\dagger \Big(\mathbf{I}_K - XZ^\dagger ZX^\dagger\Big)^{-1},$$

le résultat s'établit comme précédemment. □

Pour clore cette annexe, énonçons un résultat permettant de quantifier l'identification des termes sources $\boldsymbol{g}$. Afin d'alléger les notations, nous supposerons que la méthode numérique ayant permis de générer les données est assez fine, de telle sorte que la projection des $\boldsymbol{g}(\boldsymbol{x})$ dans l'espace discret soit exacte, autrement dit :

$$\boldsymbol{g}_m(\boldsymbol{x}) = \sum_{i=1}^{N_x} \psi_i(\boldsymbol{x}) \mathcal{B}_{i,m},$$

avec on le rappelle l'équation matricielle ayant servi à la génération des données

$$\mathcal{A}\,\mathcal{X} + \mathcal{B}\, Z = \mathcal{Y}.$$

On a ainsi

Théorème 7. *Soit $\hat{\boldsymbol{g}}(\boldsymbol{x})$ le vecteur de terme source appris par la méthode, on a alors :*

$$\Big(\sum_{m=1}^{M}\|\boldsymbol{g}_m-\hat{\boldsymbol{g}}_m\|^2_{L^2(\Omega)}\Big)^{\frac{1}{2}}\leq\ \rho(\mathcal{A})\Big(\sum_{k>K}\lambda_k\Big)^{\frac{1}{2}}\sigma^{-1}_{min}(Z)\,C''(X,Z)+\Big(\sum_{m=1}^{M}\|\boldsymbol{g}_m-\Pi_c^K\boldsymbol{g}_m\|^2_{L^2(\Omega)}\Big)^{\frac{1}{2}},$$

avec Π_c^K la projection sur l'espace POD continu, $\sigma_{min}(Z)$ la plus petite valeur singulière de Z et $C''(X,Z)=\|X^\dagger X-\mathbf{I}_{N_t}\|_F\|\Big(ZX^\dagger XZ^\dagger-\mathbf{I}_M\Big)^{-1}\|_F$.

Démonstration. En utilisant l'hypothèse

$$\boldsymbol{g}_m(\boldsymbol{x})=\sum_{i=1}^{N_x}\psi_i(\boldsymbol{x})\mathcal{B}_{i,m},$$

on a

$$\Big(\sum_{m=1}^{M}\|\boldsymbol{g}_m-\hat{\boldsymbol{g}}_m\|^2_{L^2(\Omega)}\Big)^{\frac{1}{2}}\leq\ \|\hat{B}-\tilde{B}\|_F+\Big(\sum_{m=1}^{M}\|\boldsymbol{g}_m-\Pi_c^K\boldsymbol{g}_m\|^2_{L^2(\Omega)}\Big)^{\frac{1}{2}}$$

Par ailleurs, d'après (A.4),

$$\hat{B}-\tilde{B}=\boldsymbol{\Phi}_r^T\mathbf{M}\mathcal{A}\mathcal{E}_{\mathcal{X}}\Big(X^\dagger XZ^\dagger-Z^\dagger\Big)\Big(ZX^\dagger XZ^\dagger-\mathbf{I}_M\Big)^{-1},$$

ce qui conclut la preuve. □

Annexe B

Optimisation de la règle de contrôle à l'aide d'un réseau de neurones

Reprenons l'exemple du commutateur thermique présenté au **Chapitre 3**. Nous choisirons ici d'identifier la dynamique avec 9 points sélectionnés sur une grille de façon uniforme, ainsi que le capteur. Par ailleurs, le but final est l'optimisation de la règle de contrôle, on ne cherchera alors pas un modèle réduit très précis, mais un modèle réduit donnant une idée du comportement de la température en fonction de la règle de contrôle choisie. Le processus d'optimisation requiert beaucoup de simulations, c'est donc sur la vitesse de prédiction que nous serons exigeant.

B.1 Présentation du problème

Les paramètres physiques ainsi que la géométrie utilisés dans ce paragraphe seront les même qu'au **Chapitre 3**. La figure B.2 les rappelle.

Paramètres	**Value**
Conductivité thermique dans la résistance (κ_R)	$0.5\ W \cdot m^{-1} \cdot K^{-1}$
Conductivité thermique dans la plaque (κ_P)	$0.2\ W \cdot m^{-1} \cdot K^{-1}$
Température minimale (T_{min})	$20°C$
Température maximale (T_{max})	$21°C$
Flux de la résistance électrique (q_R)	$4W \cdot m^{-2}$
Flux sortant (ϕ_{out})	$0.5\ W \cdot m^{-1}$
Pas de temps (δt)	4
Nombre de sommets du maillage (N_x)	4997
Temps final (T)	$53\,min$

TABLE B.1 Paramètres physiques du problème

Par ailleurs, la position des 9 points sélectionnés permettant d'apprendre la dynamique et du capteur guidant le contrôle est représenté à la figure B.1. Le caractère diffusif du problème laisse à penser qu'en répartissant les points de façon uniforme on capture suffisamment d'information pour pouvoir reproduire ce phénomène. Enfin nous nous limitons à 9 points sélectionnés afin d'apprendre un modèle réduit peu coûteux et ainsi réduire le temps d'optimisation du contrôle.

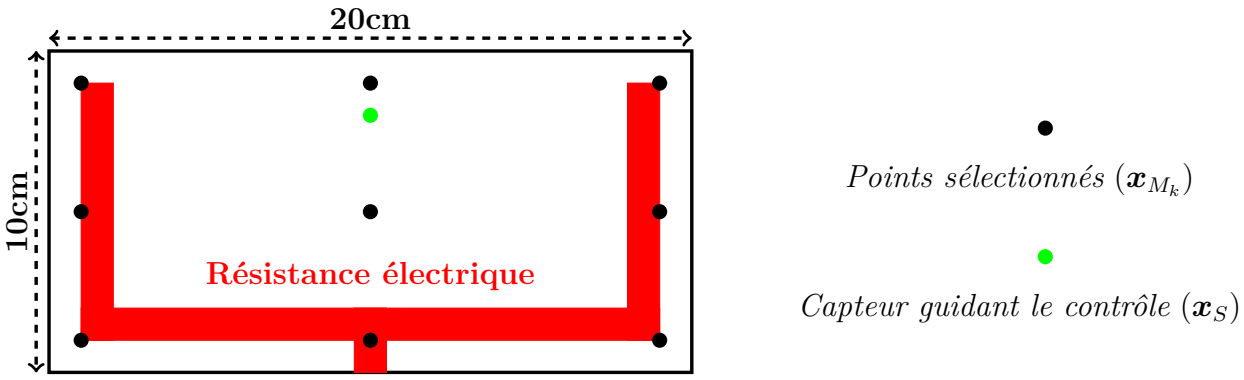


FIGURE B.1 Géométrie du problème et positions des points sélectionnés

Comme précédemment les données sont générées par un solveur dit *haute-fidélité*.

B.2 Apprentissage d'un modèle réduit

L'algorithme 2 présenté au **Chapitre 3**, nous permet d'apprendre un modèle réduit de la forme :

$$\mathbf{u}_r^{n+1} = \sum_{m=1}^{2} \hat{\boldsymbol{\theta}}_m^n \Big(\big(\mathbf{u}_r^n\big)_{10}, \hat{\boldsymbol{\theta}}^{n-1} \Big) \big(\hat{A}_m \, \mathbf{u}_r^n + \hat{B}_m \big), \quad \text{avec} \quad \begin{cases} \big(\mathbf{u}_r^n\big)_k \approx \mathbf{u}_{M_k}^n, & \forall 1 \leq k \leq 9 \\ \big(\mathbf{u}_r^n\big)_k \approx \mathbf{u}_S^n, & \text{pour } k = 10. \end{cases} \tag{B.1}$$

Afin de réduire le nombre de paramètres à optimiser, on choisira ici un réseau de neurones avec une architecture $5 - 5 - 1$. La figure ci-dessous représente l'évolution de la température au capteur, localisé en $\boldsymbol{x}_S$, au cours du temps générée par le solveur haute-fidélité et par le modèle réduit (B.1).

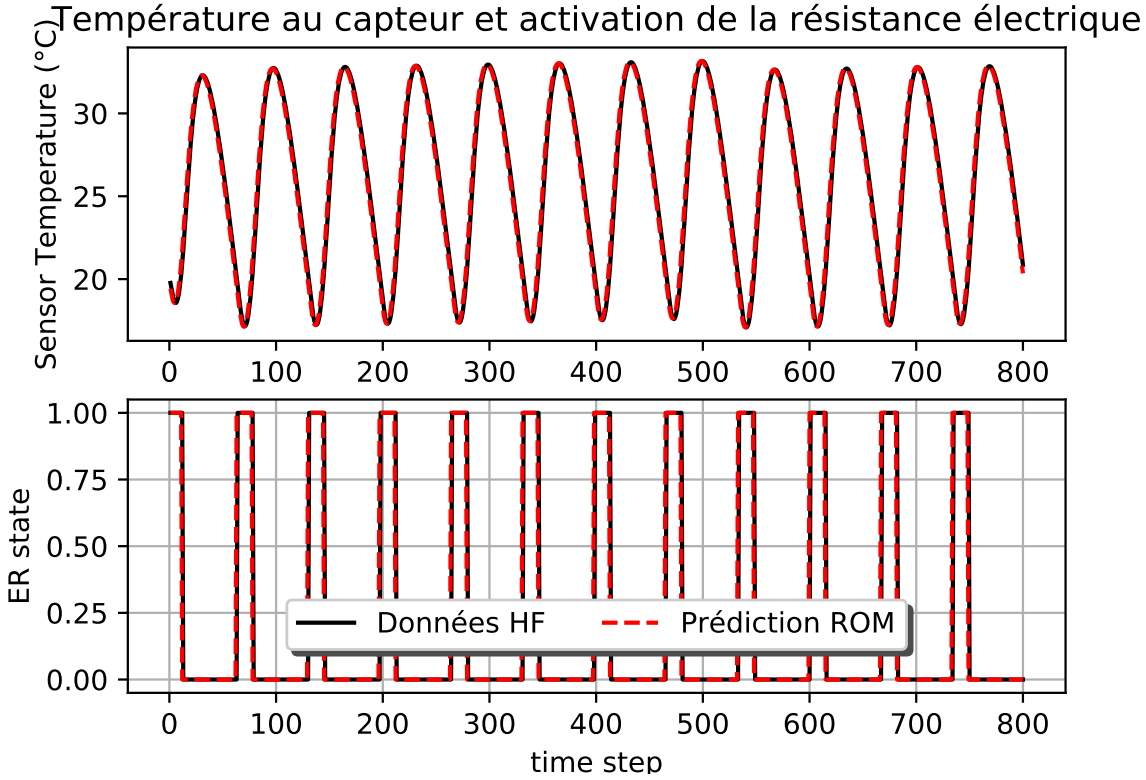


FIGURE B.2 Comparaison entre les données HF et le modèle réduit

On peut ainsi voir que la prédiction de la température au capteur est presque parfaite. Le pourcentage de réussite est de $97,3\%$, où l'on définit la performance comme étant

$$100\Big(1 - \frac{\|\mathbf{u}_S - \big(\mathbf{u}_r\big)_{10}\|_{\ell^2}}{\|\mathbf{u}_S\|_{\ell^2}}\Big).$$

B.3 Optimisation de la règle de contrôle

Dans ce paragraphe, nous souhaitons optimiser la règle de contrôle. En effet, la règle utilisée par le solveur HF, que l'on nommera règle initiale, fait varier la température entre $17,08°C$ et $33,17°C$, alors que l'on souhaitait initialement que celle-ci soit comprise entre $T_{min} = 20°C$ et $T_{max} = 21°C$. Ce défaut est dû au caractère diffusif de la chaleur, le contrôle initial attendant que la température soit supérieure à T_{max} ou inférieure à T_{min} pour changer de mode. Ainsi, en modifiant cette règle et quelque part en anticipant le caractère diffusif du problème nous pourrions obtenir une règle rendant un meilleur contrôle de la température.

Cette optimisation requiert trois choses :

- Un modèle paramétré pour le contrôle
- Un estimateur de performance pour le contrôle dépendant des paramètres
- Un algorithme d'optimisation

Nous avons vu au **Chapitre 3** que les réseaux de neurones artificiels permettaient d'apprendre efficacement la règle de contrôle. On choisira ici un réseau de neurones contenant deux sous-couches de 5 neurones chacune, afin d'avoir peu de paramètres (51) à optimiser tout en conservant une large possibilité de règle de contrôle.

En ce qui concerne l'estimateur de performance, nous choisirons la fonctionnelle $\mathcal{J}$ définie par :

$$\begin{aligned}\mathcal{J}(\boldsymbol{\mu}_{ANN}) = &\left\| \left(\mathbf{u}_r\right)_{10} - \frac{T_{min}+T_{max}}{2} \right\|_{\ell^2} + \left\| \left(\mathbf{u}_r\right)_{10} - \frac{T_{min}+T_{max}}{2} \right\|_{\ell^\infty} \\ &+ \lambda \sum_{n=0}^{N_t-1} |H^n - H^{n+1}|,\end{aligned} \tag{B.2}$$

avec $\boldsymbol{\mu}_{ANN}$ les paramètres du réseau de neurones, et H^n l'état de la résistance électrique au temps t^n.

Le premier terme de la fonctionnelle $\mathcal{J}$ permet de garder globalement la température entre T_{min} et T_{max}, le second terme permet quant à lui d'éviter qu'à certain temps la température ne monte ou ne descende trop. Enfin le dernier terme permet d'assurer un changement de mode lisse. Ce terme n'est rien d'autre que la variation totale, souvent utilisé en *volume finie* pour éviter que le schéma numérique n'engendre des oscillations parasites ([79]).

Le dernier point est l'algorithme d'optimisation. Il existe deux classes d'algortihmes d'optimisation ceux nécessitant le gradient de la fonctionnelle et ceux ne le nécessitant pas ([128]). Nous choisirons ici une méthode sans gradient. Par ailleurs, au vu du grand nombre de minima locaux, il sera préférable de choisir un algorithme ayant

une bonne exploration de l'espace de recherche. C'est pourquoi nous choisirons un algorithme génétique ([90]). L'algorithme ci-dessous présente la procédure d'optimisation :

Algorithme 5 : Algorithme génétique pour optimiser les paramètres du réseau de neurones

Entrée : le modèle réduit (B.1), ainsi que les paramètres `num_generations`, `num_parents`, `mutation_proba`, et `sol_per_pop`.

Sortie : Les paramètres optimaux $\boldsymbol{\mu}_{ANN}$.

Initialisation :

for $per \in \{1, \cdots, \texttt{sol\_per\_pop}\}$ **do**

$$\boldsymbol{\mu}_{ANN} \leftarrow \texttt{semi-train}\left(\begin{bmatrix} \mathbf{u}_S^2 & \mathbf{u}_S^3 & \cdots & \mathbf{u}_S^{N_t-1} \\ \boldsymbol{\theta}^1 & \boldsymbol{\theta}^2 & \cdots & \boldsymbol{\theta}^{N_t-2} \end{bmatrix} ; \boldsymbol{\Theta}_{\cdot,2:N_t-1} \right)$$

pop.`append`($\boldsymbol{\mu}_{ANN}$).

Évolution de la population :

for $iter \in \{1, \cdots, \texttt{num\_generations}\}$ **do**

$J \leftarrow$ `fitness_pop`(pop)

$X_{pop} \leftarrow$ `mlp_to_vector`(pop)

$pop\_parents \leftarrow$ `select_mating_pool`($X_{pop}, J,$ `num_parents`)

$pop\_crossover \leftarrow$ `crossover`($pop\_parents$)

$pop\_mutation \leftarrow$ `mutation`($pop\_crossover,$ `mutation_proba`)

$X_{pop} \leftarrow \begin{bmatrix} pop\_parents & pop\_mutation \end{bmatrix}$

$pop \leftarrow$ `vector_to_mlp`(X_{pop})

Nous considérerons plusieurs valeurs de λ dans la fonctionnelle $\mathcal{J}$ définie par (B.2), afin de construire des règles de contrôle plus ou moins lisse. La validation consistera a comparer les données produites par le solveur haute-fidélité aux donnée produite par le modèle réduit POD (avec 30 modes) du **Chapitre 3** et pour la règle de contrôle optimisée.

La figure ci-dessous présente les résultats pour différentes valeurs de λ.

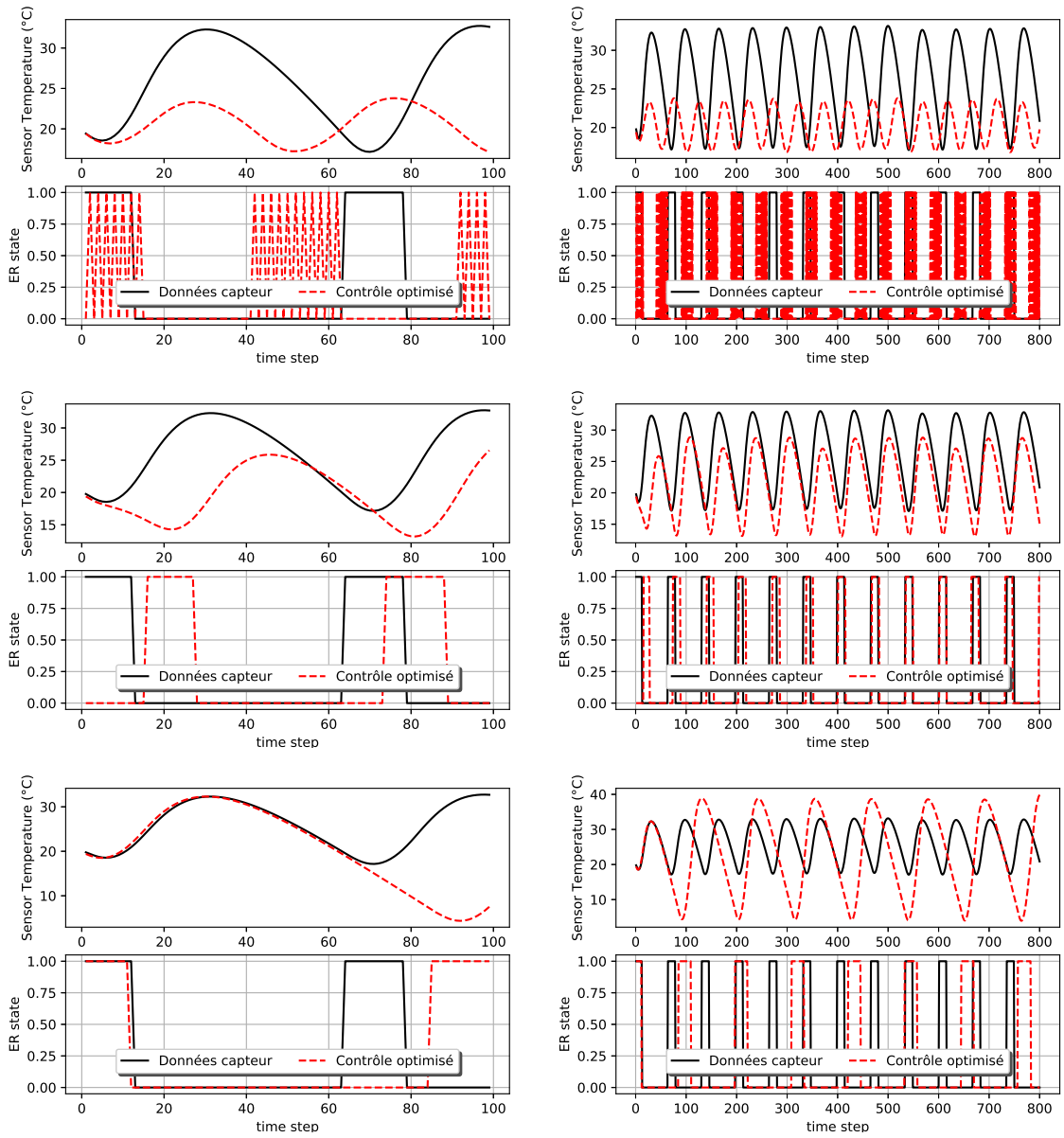


FIGURE B.3 (Gauche) Température et état de la résistance électrique, (droite) Validation en temps long, (De haut en bas) $\lambda = 0, 0.5, 8$

Lorsque le nombre de commutation n'est pas imposé (cas où $\lambda = 0$), le contrôle optimisé alterne entre une phase où à chaque pas de temps l'état de la résistance électrique change, et une phase où la résistance électrique est désactivée. Il semble qu'en faisant cela le contrôle essaie de compenser la puissance trop importante de la résistance électrique. Ainsi en changeant à chaque pas de temps l'état de la résistance électrique, on "créé" un flux permettant de maintenir la température au capteur supérieur à T_{min} sans pour autant dépasser T_{max} de beaucoup. Lorsque l'on pénalise le nombre

de commutation, cette stratégie n'est plus employable. Ainsi le seuil de température mesuré par le capteur est plus important. Et plus la pénalisation est importante, moins bon sera le contrôle.

Afin de quantifié l'amélioration du contrôle, définissons le taux de réussite par

$$100\left(1-\max\left(\frac{\left(T_{min}-\min_n \mathbf{u}_S^n\right)^+}{T_{min}},\frac{\left(\max_n \mathbf{u}_S^n-T_{max}\right)^+}{T_{max}}\right)\right).$$

Le tableau ci-dessous résume le score suivant la valeur λ choisie.

Type de contrôle	**Score**	**Nb de commutation**
Contrôle HF	$38,4\%$	23
$\lambda = 0$	$84,1\%$	365
$\lambda = \frac{1}{2}$	$62,6\%$	26
$\lambda = 8$	$9,9\%$	15

TABLE B.2 Paramètres physique du problème

Les résultats viennent conforter les observations de la figure B.3. Si le nombre de commutation n'est pas coûteux, alors le cas où $\lambda = 0$ donne de bien meilleur résultats que le contrôle initial. En revanche, si le changement d'état de la résistance électrique a un coût, il vaut mieux trouver un compromis, et le cas où $\lambda = \frac{1}{2}$ améliore nettement les performances du contrôle tout en requierant peu de commutation.

Ce paragraphe a ainsi pu mettre en lumière l'utilité du modèle introduit présenté au **Chapitre 3**. Combiné à deux algorithme de machine learning, à savoir les algorithmes génétiques et les réseaux de neurones, on a pu construire un contrôle ayant de meilleur capacités que celui initialement utilisé. On pourrait considérer une autre architecture pour notre réseau de neurones et ainsi engendré d'autres types de contrôles. L'utilisation d'algorithme génétique afin d'entraîner des réseaux de neurones est d'ailleurs en pleine extension afin d'éviter les minima locaux ([168], [167]).

Annexe C

Comparaison des algorithmes d'échantillonnage pour la prédiction d'une onde

Dans cette annexe nous présentons les prédictions du modèle (4.11) introduit à la **Section 4.4.2** du **Chapitre 4**, en utilisant les différents algorithmes d'échantillonnage. Nous avions vu que l'échantillonnage par EIM, permettait une prédiction correcte pour $K = 50$ et parfaite pour $K = 100$. Voyons à présent ce qu'il en est lorsque l'on utilise d'autres méthodes d'échantillonnage. La figure ci-dessous présente les résultats.

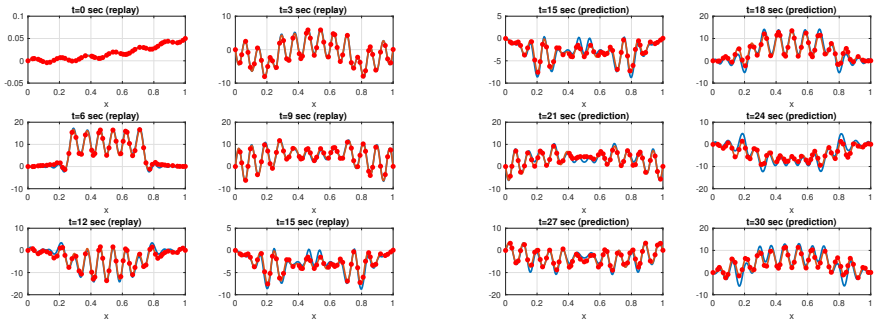
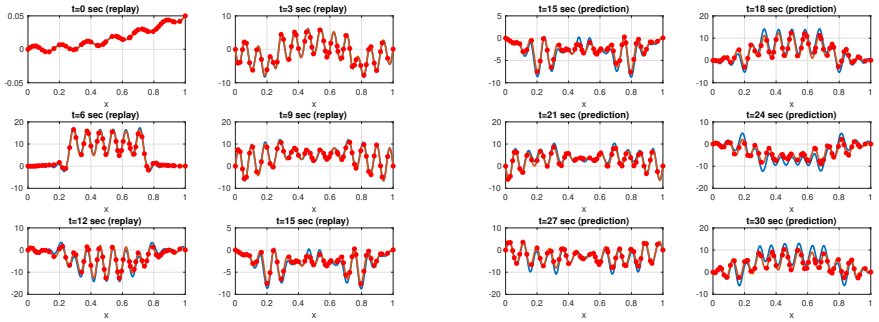
Réduction	**Prédiction sur l'ensemble d'entrainement (entre 0 et 15 sec)**	**Prédiction sur l'ensemble de validation (entre 15 et 30 sec)**
EIM		
RandGCSS		

FIGURE C.1 Prédiction et données HF pour $K = 50$

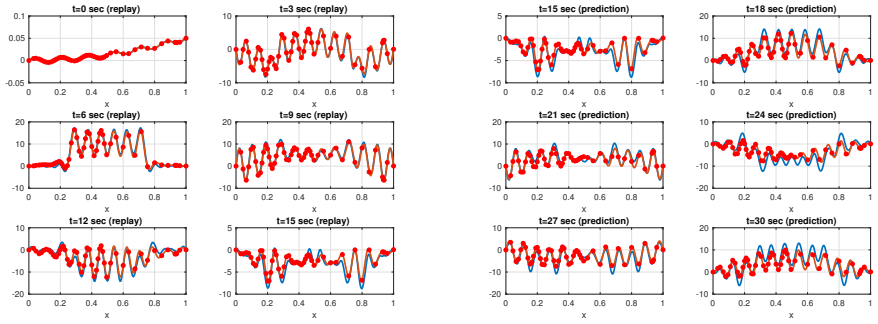
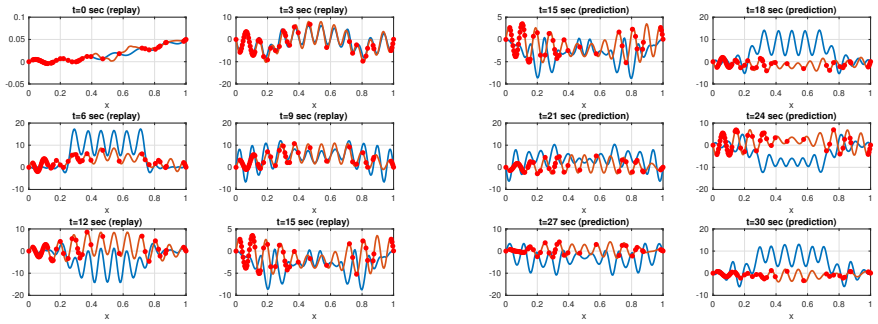
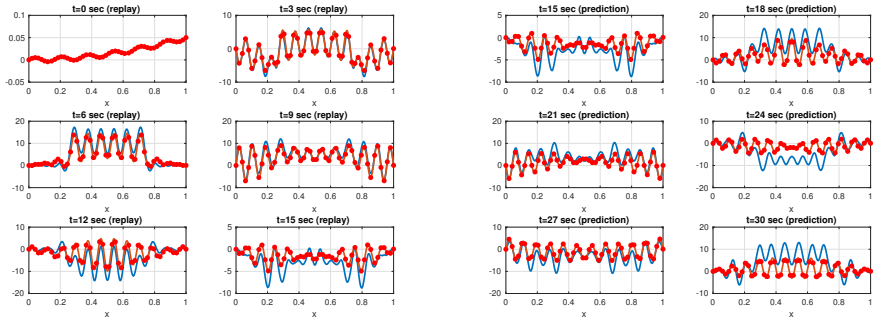
Réduction	Prédiction sur l'ensemble d'entrainement (entre 0 et 15 sec)	Prédiction sur l'ensemble de validation (entre 15 et 30 sec)
NOCS		
k-Médoides		
Uniforme		

FIGURE C.2 Prédiction et données HF pour $K = 50$ (suite)

L'algorithme RandGCSS semble présenter des résultats similaires à EIM. L'algorithme NOCS est quant à lui un peu moins bon. En revanche, sélection par l'algorithme des k-Médoides ne parvient même pas à reconstruire correctement la solution initiale, tandis que la prédiction à l'aide de l'échantillonnage uniforme semble se dégrader sévèrement au cours du temps. On voit ainsi l'importance de la bonne sélection des points.

Voyons à présent si augmenter la dimension du modèle réduit K augment la précision de la prédiction. Le tableau ci-dessous présente les résultats pour $K = 100$.

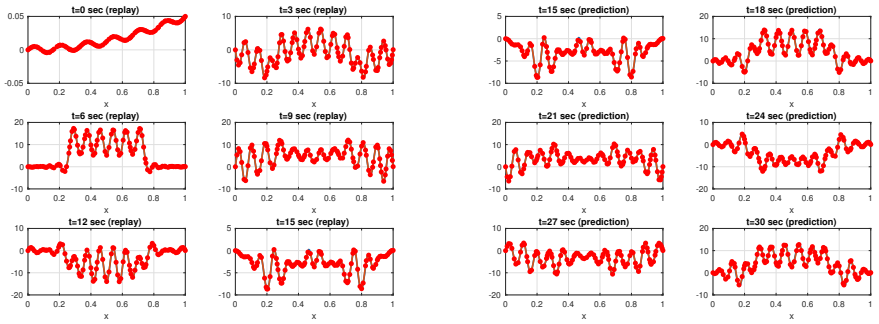
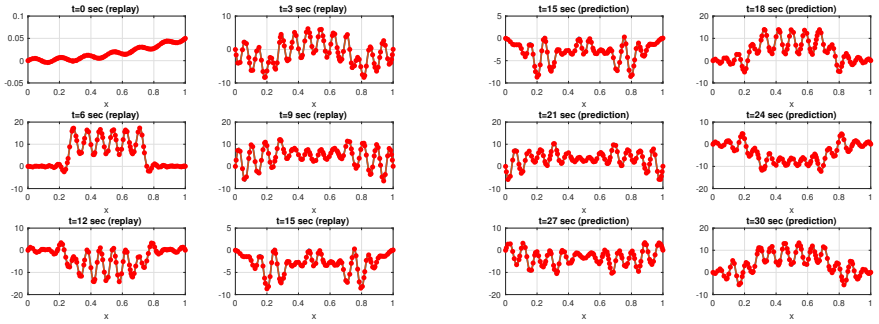
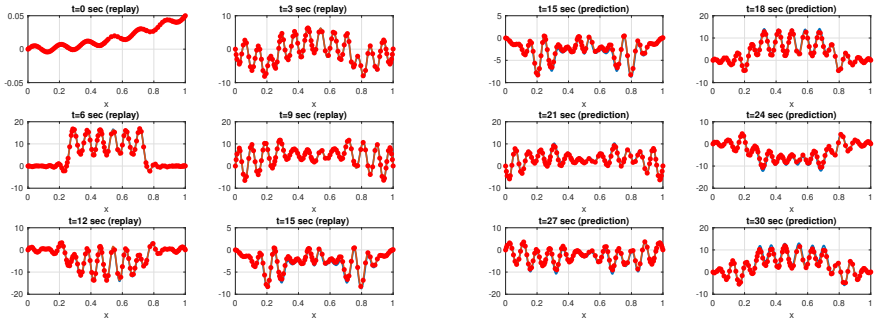
Réduction	**Prédiction sur l'ensemble d'entrainement (entre 0 et 15 sec)**	**Prédiction sur l'ensemble de validation (entre 15 et 30 sec)**
EIM		
RandGCSS		
NOCS		

FIGURE C.3 Prédiction et données HF pour $K = 100$

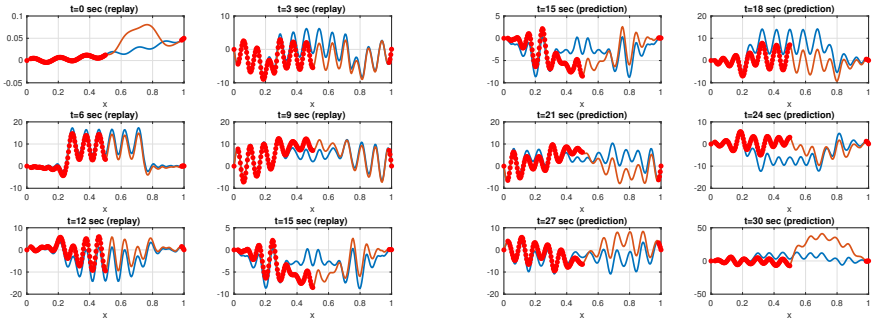
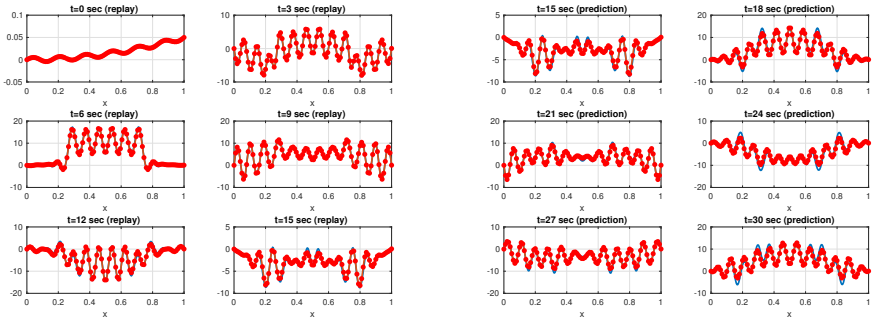
Réduction	Prédiction sur l'ensemble d'entrainement (entre 0 et 15 sec)	Prédiction sur l'ensemble de validation (entre 15 et 30 sec)
k-Médoides		
Uniforme		

FIGURE C.4 Prédiction et données HF pour $K = 100$ (suite)

Cette fois-ci la prédiction par EIM et RandGCSS colle parfaitement aux données HF, celle par NOCS est presque aussi bonne. En revanche la sélection par l'algorithme des k-Médoides ne permet pas une correct prédiction. En effet, l'algorithme sélectionne principalement les points à gauche du domaine délaissant ceux à droite et au milieu. Enfin la sélection par échantillonnage uniforme donne une prédiction plutôt bonne, elle semble cependant se dégrader au cours du temps.

Annexe D

Apprentissage d'un modèle réduit paramétré par une autre approche : Régression du modèle réduit

Nous avions appris la variété des solutions à l'aide de l'algorithme KRR, puis nous avions appris un modèle réduit pour un paramètre fixé à l'aide de la méthode DMD. Cette approche s'est avérée fructueuse. Mais qu'en est-il si nous inversons la procédure, et apprenons d'abord les modèles réduits pour tout $\mu \in \mathcal{D}_h^{train}$, puis à l'aide de l'algorithme KRR prédisons les matrice et vecteur $\hat{\mathbf{A}}(\mu)$ et $\hat{\mathbf{B}}(\mu)$ pour un paramètre μ donné ?

Pour répondre à cette question, reprenons les algorithmes de régression présentés au **Chapitre 5**, et apprenons la matrice

$$\left[\hat{\mathbf{A}}(\mu) \quad \hat{\mathbf{B}}(\mu)\right].$$

On reprendra la mesure de performance définie au **Chapitre 5**. La figure ci-dessous présente les données d'entrainement pour la sous-matrice

$$\begin{pmatrix} \hat{\mathbf{A}}_{1,1}(\mu) & \hat{\mathbf{A}}_{1,2}(\mu) & \hat{\mathbf{A}}_{1,3}(\mu) \\ \hat{\mathbf{A}}_{2,1}(\mu) & \hat{\mathbf{A}}_{2,2}(\mu) & \hat{\mathbf{A}}_{2,3}(\mu) \\ \hat{\mathbf{A}}_{3,1}(\mu) & \hat{\mathbf{A}}_{3,2}(\mu) & \hat{\mathbf{A}}_{3,3}(\mu) \end{pmatrix},$$

.

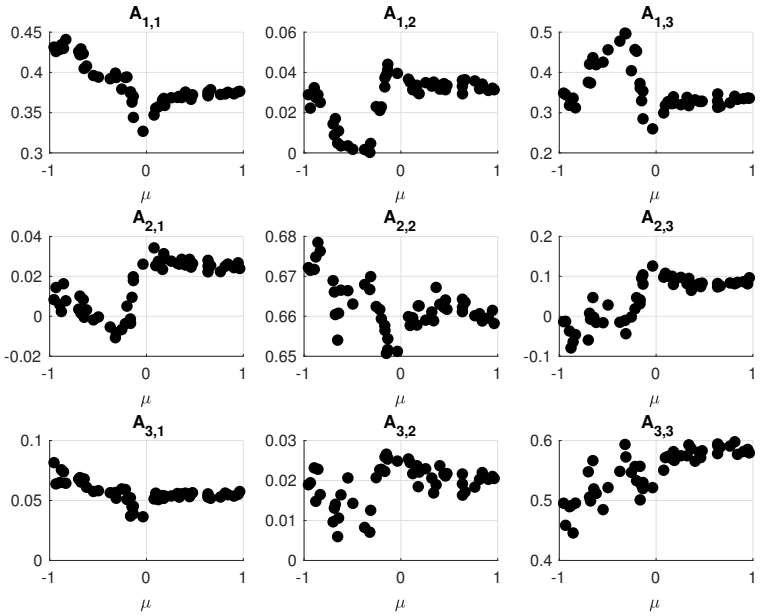
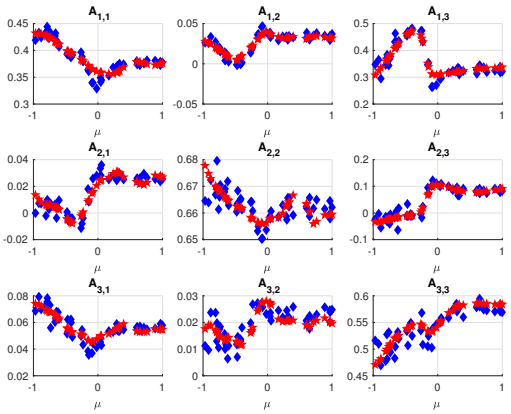


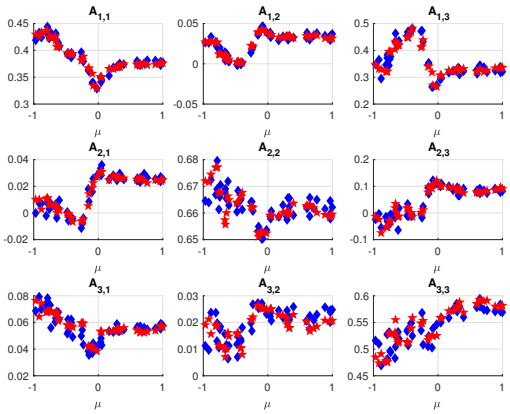
FIGURE D.1 Données d'entrainement.

On remarque que cette fois identifier une structure dans les données est beaucoup moins évident.

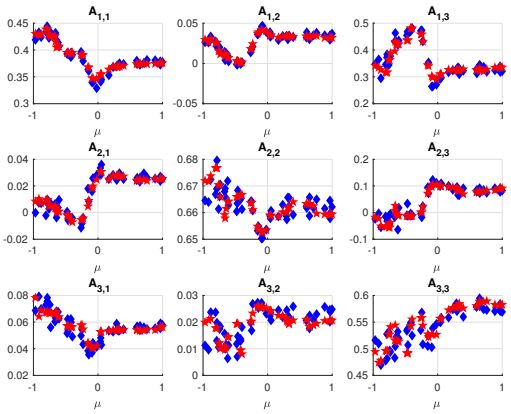
À présent essayons de prédire le modèle réduit en fonction d'un paramètre μ donné. La figure ci-dessous représente la prédiction de la sous-matrice par différents algorithmes :



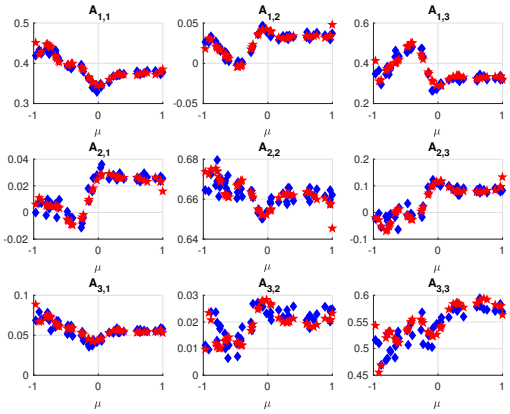
(a) ANN, score = 92.57%, temps = 0.031 sec



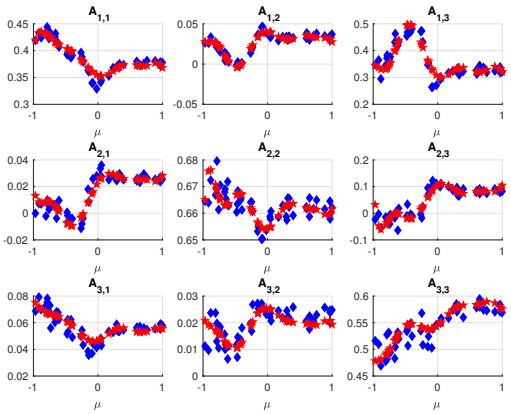
(b) kNN, score = 91.18%, temps = 0.055 sec



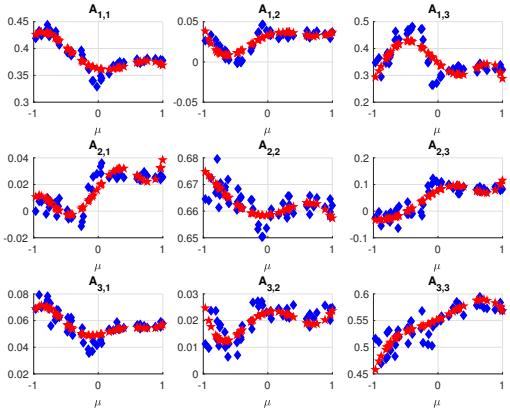
(c) RF, score = 91.34 %, temps = 0.057 sec



(d) KRR-'rbf', score = 91.45%, temps = 0.069 sec



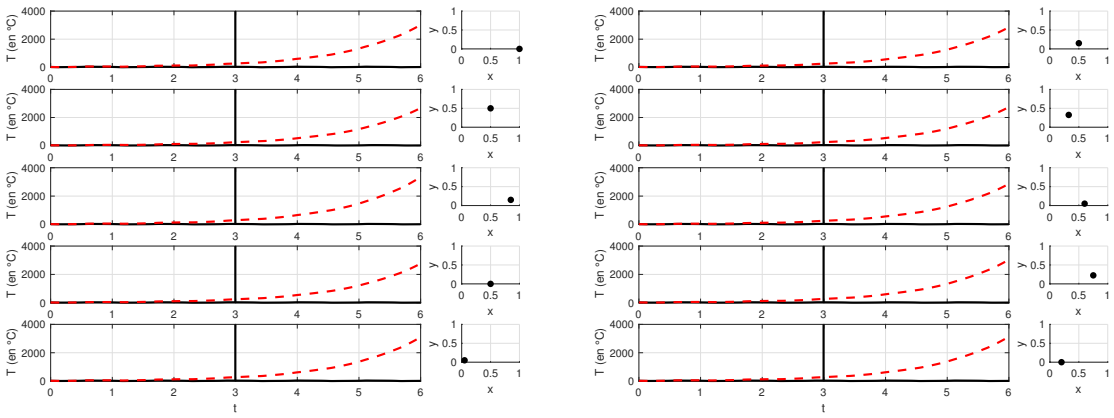
(e) GP, score = 91.89%, temps = 0.022 sec



(f) KRR-'poly', score = 90.51%, temps = 0.053 sec

FIGURE D.2 Prédictions (rouge) de la sous-matrice pour tous les paramètres $\mu \in \mathcal{D}_h^{val}$. Valeur exacte (bleue)

Comme attendu, la prédiction est cette fois-ci beaucoup plus difficile et les résultats ne dépassent pas les 93% de réussite. Cette approche ne semble donc pas être la bonne, pour s'en convaincre la figure ci-dessous représente la prédiction aux points sélectionnés obtenue à l'aide du modèle réduit paramétré appris par régression.



(a) $\mu = 0.9982$ (en $\boldsymbol{x}_{\sigma(1)}, ..., \boldsymbol{x}_{\sigma(5)}$) (b) $\mu = 0.9982$ (en $\boldsymbol{x}_{\sigma(6)}, ..., \boldsymbol{x}_{\sigma(10)}$)

FIGURE D.3 (Gauche) Comparaison entre la prédiction de la solution (ligne pointillé rouge) et la solution fournit par le solveur HF (ligne noir continue). (Droite) Position du point sur le domaine