Université Pierre et Marie Curie - CEA Cadarache

THÈSE

pour obtenir le grade de
docteur de l'Université Paris 6

Présentée par

Benjamin AUDER

Classification et modélisation de sorties fonctionnelles de codes de calcul.

Application aux calculs thermo-hydrauliques accidentels dans les réacteurs à eau pressurisés (REP)

Soutenue le 5 mai 2011 devant le jury composé de

M. Gérard	BIAU	Directeur de thèse
M. Michel	BRONIATOWSKI	Examinateur
M. Bertrand	IOOSS	Encadrant
M. Jean-Michel	MARIN	Président du jury
M. Michel	MARQUÈS	Encadrant
M. André	MAS	Rapporteur
M. Fabrice	ROSSI	Examinateur
M. Michel	VERLEYSEN	Rapporteur

Remerciements

À Gérard Biau pour son enthousiasme, le suivi de mes recherches et la confiance qu'il m'a accordée lorsque je semblais m'éloigner du sujet.

À Bertrand Iooss pour son encadrement dynamique et toute l'aide qu'il m'a apportée, me permettant notamment de participer à divers séminaires enrichissants.

À Michel Marquès pour le cahier des charges du package R et l'éclaircissement sur la physique des codes de calculs.

À l'équipe du LCFR qui m'a permis d'évoluer dans un environnement convivial et agréable.

À Jean-Michel Marin, Amandine Marrel et Fabrice Rossi pour leurs critiques et conseils avisés au sein du comité de thèse.

À André Mas et Michel Verleysen, rapporteurs, pour leur lecture attentive et critique, manifestant leur intérêt pour mes travaux.

Aux membres du jury pour avoir accepté de participer à ma soutenance, et pour tous les compliments qu'ils m'y ont adressés.

À ma famille pour le soutien qu'elle m'a apporté tout au long de la thèse, et pour l'organisation efficace du pot à la soutenance. Merci donc à Annie, Christophe, Élodie, Fanny, Gisèle, Jacqueline, Jean-Bernard, Madeleine, Michel, Joseph.

À Sophie pour les corrections apportées au manuscrit, l'aide sur l'orientation post-thèse et tous ses encouragements.

Aux "anciens" membres du LSTA pour l'ambiance sympathique au cours de mes quelques visites, dont Aurélie pour m'avoir supporté pendant la rédaction de l'article.

À Aurélien, Benjamin C., Benjamin G., Catherine, Cécile, Gérald, Layal, Matieyendou, Sarah, Thi Thu Nga (Tigrou), pour avoir assisté à la soutenance et ne pas s'être enfui pendant la longue série de questions.

Au Commissariat à l'Énergie Atomique pour le financement de ma thèse.

Aux intervenants du forum du site les-mathematiques.net, qui m'ont aidé à y voir plus clair sur certains points techniques.

Résumé

Compte-tenu de la complexité des systèmes industriels actuels et des progrès en calcul scientifique, les codes utilisés pour modéliser des phénomènes physiques en ingénierie nucléaire sont souvent coûteux en temps. Il est cependant nécessaire de réaliser des analyses statistiques sur certains événements, et ces analyses demandent de multiples applications du code pour être précises. C'est pourquoi le temps de simulation doit être réduit, en modélisant le code de calcul par une fonction de coût CPU négligeable. Cette modélisation s'effectue sur la base d'un échantillon de quelques centaines de résultats de calculs physiques.

Ce travail s'inscrit dans le cadre relativement peu étudié des codes de calcul à réponses fonctionnelles $1D$. Ces dernières modélisent l'évolution de paramètres physiques dans le temps, pour un état initial $x \in \mathbb{R}^p$. Différents types d'évolution peuvent se dégager ; c'est pourquoi les (entrées-)sorties sont d'abord divisées en K groupes, une méthode basée sur l'erreur de classification supervisée permettant de sélectionner ce dernier nombre automatiquement. Afin de contourner la difficulté liée aux réponses fonctionnelles, l'idée principale consiste à représenter ces dernières en dimension réduite d pour effectuer la régression dans le cadre vectoriel. Pour cela nous proposons une alternative non linéaire à la décomposition sur une base, accompagnée de sa justification théorique. Nous montrons que l'application $\hat{\phi}$ ainsi construite permet d'approximer une large classe de codes, et est complémentaire de l'approche classique (utilisant une base de fonctions) sur les jeux de données CEA.

Table des matières

Remerciements i

Résumé iii

Notations xi

0 Introduction 1
- 0.1 Contexte industriel . . . 1
- 0.2 Formulation du problème . . . 4
- 0.3 Exemple jouet et méthodologie . . . 6
- 0.4 État de l'art . . . 8
  - 0.4.1 Famille $\{f_j\}$ adaptée aux données . . . 8
  - 0.4.2 Modélisation via une base de fonctions . . . 10
- 0.5 Plan de la thèse . . . 13

1 Représentation des données dans un graphe 16
- 1.1 Construction . . . 17
  - 1.1.1 Définitions . . . 17
  - 1.1.2 Graphes de voisinage basiques . . . 18
    - 1.1.2.1 Graphe complet . . . 18
    - 1.1.2.2 ϵ-graphe . . . 19
    - 1.1.2.3 Graphe des k plus proches voisins . . . 19
  - 1.1.3 Graphes de voisinage évolués . . . 20
    - 1.1.3.1 Graphe de visibilité . . . 20
    - 1.1.3.2 Voisinages adaptatifs I . . . 22
    - 1.1.3.3 Voisinages adaptatifs II . . . 23
- 1.2 Distance dans le graphe . . . 25
  - 1.2.1 Distance euclidienne "Commute-Time" . . . 25
  - 1.2.2 Calcul du temps moyen d'aller-retour . . . 27
    - 1.2.2.1 Calcul par récurrence . . . 27
    - 1.2.2.2 Calcul via le temps d'atteinte . . . 27
    - 1.2.2.3 Calcul avec le laplacien . . . 28
- 1.3 Détermination de l'échelle locale . . . 30
  - 1.3.1 Bibliographie . . . 30
  - 1.3.2 Comparaisons . . . 31
    - 1.3.2.1 Données artificielles . . . 32
    - 1.3.2.2 Données réelles . . . 34
- 1.4 Conclusion . . . 36

1.5 Perspectives 36

2 Clustering des entrées-sorties du code **37**
2.1 Comparaison des courbes 38
2.2 Clustering simultané entrées - sorties 39
2.3 Classification non supervisée fonctionnelle : état de l'art 42
2.3.1 Algorithme des k-means 42
2.3.2 Méthodes non probabilistes 43
2.3.3 Méthodes bayésiennes 44
2.3.4 Bilan 45
2.4 Clustering dans un graphe : état de l'art 47
2.4.1 Algorithmes sans marche aléatoire 47
2.4.2 Algorithmes utilisant la marche aléatoire $\mathcal{M}$ 49
2.4.3 Classification ascendante hiérarchique 50
2.5 Détermination du nombre de classes 52
2.5.1 Méthodes basées sur la structure des données 53
2.5.2 Méthodes basées sur la stabilité des partitions 54
2.5.3 Approche retenue 56
2.5.4 Tests 60
2.5.4.1 Données artificielles 60
2.5.4.2 Données simulées (CATHARE) 62
2.6 Conclusion 66
2.7 Perspectives 66

3 Représentation discrète d'une variété **67**
3.1 Estimation de la dimension 68
3.1.1 Définitions alternatives 68
3.1.2 Approches existantes 70
3.1.2.1 Méthodes basées sur l'ACP 71
3.1.2.2 Méthodes basées sur la densité 72
3.1.2.3 Méthodes géométriques 73
3.1.3 Régularisation d'une méthode géométrique 75
3.1.4 Détection stochastique des slivers 77
3.1.5 Tests 78
3.2 Réduction de la dimension 80
3.2.1 Méthodes linéaires 81
3.2.1.1 Base des composantes principales 82
3.2.1.2 Quelques extensions de l'ACP 84
3.2.1.3 Autres bases de fonctions 85
3.2.2 Méthodes non linéaires globales 86
3.2.2.1 Approches spectrales 86
3.2.2.2 Approches probabilistes 89
3.2.2.3 Réseaux de neurones 90
3.2.2.4 Optimisation coûteuse 91
3.2.3 Méthodes non linéaires locales 92
3.2.3.1 Alignement incrémental de cartes ACP locales 93
3.2.3.2 Apprentissage des coordonnées riemanniennes normales . . . 94

3.2.3.3 Réduction de dimension multi-cartes . . . 96
3.2.4 Comparaisons des représentations . . . 97
3.2.4.1 Swisshole et boîte ouverte . . . 97
3.2.4.2 Jeux de données CATHARE . . . 100
3.3 Conclusion . . . 102
3.4 Perspectives . . . 102

4 Régression et apprentissage de variété 103
4.1 Méthodes de régression . . . 103
4.1.1 Régression linéaire . . . 104
4.1.2 Approximation polynomiale . . . 104
4.1.3 Méthodes à noyaux . . . 105
4.1.3.1 Support Vector Regression . . . 106
4.1.3.2 Krigeage . . . 106
4.1.4 Aggrégation d'estimateurs : boosting . . . 108
4.1.5 Réseaux de neurones . . . 108
4.1.6 Régression par "poursuite de projection" . . . 109
4.1.6.1 Régression sur composantes principales . . . 109
4.1.6.2 Projection sur les "structures latentes" . . . 110
4.1.6.3 Projection Pursuit Regression . . . 111
4.2 Reconstruction locale des courbes . . . 113
4.2.1 Inversion de LPcaML . . . 113
4.2.2 Inversion de RML . . . 114
4.2.2.1 Voisin de l'origine z_0 . . . 115
4.2.2.2 Sommet éloigné dans le graphe . . . 115
4.2.3 Tests . . . 115
4.2.3.1 Données artificielles . . . 116
4.2.3.2 Données réelles . . . 118
4.3 Régression fonctionnelle directe . . . 121
4.3.1 Estimateur des k plus proches voisins fonctionnel . . . 122
4.3.2 Régression fonctionnelle par ACP locale . . . 122
4.3.3 "Functional projection pursuit" . . . 123
4.4 Conclusion . . . 124
4.5 Perspectives . . . 124

5 Évaluation du métamodèle 125
5.1 Récapitulatif du modèle . . . 125
5.2 Validation via l'erreur de prédiction . . . 126
5.2.1 Données générées artificiellement . . . 126
5.2.2 Données réelles (CATHARE) . . . 128
5.3 Tests sur données artificielles . . . 129
5.3.1 Trois clusters non linéaires . . . 129
5.3.1.1 Avec clustering . . . 130
5.3.1.2 Sans clustering . . . 132
5.3.2 Fonctions sinusoïdales amorties . . . 133
5.3.2.1 Dimension 3 . . . 134
5.3.2.2 Dimension 5 . . . 135

5.4 Tests sur données réelles simulées . . . 136
5.4.1 CATHARE I . . . 136
5.4.1.1 Avec clustering . . . 137
5.4.1.2 Sans clustering . . . 139
5.4.2 CATHARE II . . . 141
5.5 Bilan . . . 145

6 Conclusion **146**

A Structure des sorties du code **148**
A.1 Variétés riemanniennes . . . 148
A.1.1 Variété . . . 149
A.1.2 Espace tangent . . . 150
A.1.3 Visualisation d'un espace tangent fonctionnel . . . 151
A.1.4 Fibré tangent . . . 152
A.1.5 Métriques riemanniennes . . . 154
A.2 Paramétrage "naturel" d'une variété riemannienne . . . 155
A.2.1 Connexions . . . 155
A.2.2 Géodésiques . . . 157
A.2.3 Coordonnées riemanniennes normales . . . 158
A.2.4 Courbure . . . 159

B k-means fonctionnel projeté **161**
B.1 Introduction . . . 163
B.1.1 The CATHARE code . . . 163
B.1.2 Clustering . . . 165
B.2 Finite-dimensional projection for clustering . . . 167
B.3 Basis selection . . . 171
B.4 Experimental results and analysis . . . 176
B.4.1 Synthetic control chart time series . . . 176
B.4.2 Industrial code examples . . . 181
B.5 Conclusion . . . 186
B.6 Proofs . . . 187
B.6.1 Proof of Lemma B.2.1 . . . 187
B.6.2 Proof of Theorem B.2.1 . . . 188
References . . . 189

C Criblage et métamodélisation fonctionnelle **192**
C.1 Introduction . . . 193
C.2 The thermal-hydraulic transient simulation . . . 195
C.2.1 The industrial problem . . . 195
C.2.2 The CATHARE2 model . . . 196
C.2.3 Definition of uncertainty sources . . . 197
C.3 Screening with generalized sensitivity indices . . . 198
C.3.1 Generalized sensitivity indices method . . . 198
C.3.2 Results for the response T_{bas} . . . 200
C.4 The functional metamodeling . . . 202
C.4.1 Methods for dimensionality reduction . . . 203

C.4.2 Application to the PTS 205
C.5 Conclusion 207
C.6 Acknowledgments 207
References 207

Références **211**

Notations

ϕ : fonction représentative du code de calcul étudié. Dans cette thèse, les entrées sont vectorielles et les sorties fonctionnelles.

$\hat{\phi}$: estimation du code de calcul que ϕ représente, appelée *métamodèle* ou simplement *modèle*. Plus généralement, $\hat{u}$ désigne l'estimation d'une certaine quantité u.

$\mathcal{E}_E$: espace des entrées du code de calcul (sous-ensemble de $\mathbb{R}^p$; en général un produit d'intervalles compacts).

p : dimension de l'espace d'entrée du code de calcul (éventuellement après réduction de la dimension).

D : nombre de points de discrétisation pour une fonction en sortie.

$\mathcal{U}$: variété sur laquelle sont échantillonnées les courbes ; $\mathcal{U} \subset \mathcal{C}([a,b])$, identifié à $\mathbb{R}^D$ (discrétisation).

k : nombre de voisins en un point du graphe décrivant les données (chapitre 1).

$\mathcal{V}(i)$: indices des voisins de l'élément u_i (dans un graphe, et/ou pour une certaine distance). Selon le contexte, $\mathcal{V}(i)$ peut aussi désigner directement les voisins.

K : nombre de clusters (entrées, sorties, ou les deux ensembles ; voir le chapitre 2).

d : dimension de l'espace de représentation des sorties du code de calcul.

$\mathcal{E}_R$: espace de représentation des courbes (sous-ensemble de $\mathbb{R}^d$).

r : fonction de représentation, calculant les coordonnées réduites d'une courbe (voir le chapitre 3).

g : fonction de régression reliant les entrées x_i aux coordonnées réduites z_i au sein d'un cluster.

R : fonction de reconstruction, recomposant une courbe à partir de ses coordonnées réduites (voir le chapitre 4).

n : nombre de simulations préliminaires réalisées. Pour $i = 1, \ldots, n$:

$$\begin{cases} x_i : \text{entrées du code, vecteurs de } \mathcal{E}_E \subset \mathbb{R}^p\,. \\ y_i : \text{sorties du code, fonctions continues de } [a,b] \text{ dans } \mathbb{R}\,. \\ z_i : \text{représentations des sorties du code, vecteurs de } \mathcal{E}_R \subset \mathbb{R}^d\,. \end{cases}$$

T_uV : espace tangent en un point u d'une variété différentiable V.

$\|\,.\,\|$: norme euclidienne ou L^2, suivant la nature de l'objet.

$\leftarrow$: "est remplacé par" ; raccourci utilisé pour la description d'algorithmes.

$\sqcup$: union d'ensembles disjoints.

$|X|$: cardinal de l'ensemble (fini) X.

$\mathbf{1}^m$: vecteur de taille m ne contenant que des 1.

${}^tA$: transposée de la matrice A.
Convention : si A est une matrice représentant des vecteurs ou fonctions discrétisées, ceux-ci sont toujours placés en lignes. En revanche, si u représente un seul vecteur (ou fonction discrétisée) on place les données en colonne.

I, $I_{m\times q}$: matrice identité (resp. "pseudo-identité"), contenant des 1 sur la grande diagonale et nulle partout ailleurs.

$\mathcal{C}^k(A)$: espace des fonctions de classe $\mathcal{C}^k$ définies sur un ensemble A, à valeurs dans $\mathbb{R}$. $\mathcal{C}^0(A)$ est noté $\mathcal{C}(A)$.

$\lfloor\, u \,\rfloor$, $\lceil\, u \,\rceil$: plus grand entier inférieur (resp. plus petit entier supérieur) à $u \in \mathbb{R}$.

$d(u,v)$, $d_{i,j}$: distance euclidienne ou L^2 entre u et v (resp. u_i et u_j).

$\rho(u,v)$, $\rho_{i,j}$: distance géodésique entre u et v (resp. u_i et u_j).

Chapitre 0

Introduction

Plan de l'introduction :

1. **Présentation du contexte industriel.**
2. **Formulation mathématique du problème posé.**
3. **Exemple jouet permettant de motiver la méthodologie.**
4. **État de l'art des métamodèles fonctionnels.**

0.1 Contexte industriel

Cette thèse effectuée au CEA Cadarache s'inscrit dans le cadre d'un projet sur la durée de vie des cuves de réacteurs nucléaires. L'objectif est de garantir que ces cuves résistent aux éventuelles défaillances matérielles. Afin d'estimer la fiabilité de la cuve sous les différents scénarios d'accidents pouvant survenir, des codes de calcul les simulant sont développés à partir de systèmes d'équations physiques. Ces codes (i.e. programmes informatiques) permettent d'évaluer certaines quantités clés sans avoir recours à l'expérimentation réelle, qui est bien sûr impossible. Deux méthodologies peuvent être distinguées, la seconde étant choisie pour cette thèse :

- l'établissement de marges déterministes sur chaque paramètre pris en compte ; un petit nombre d'évaluations pessimistes sont alors réalisées ;
- l'estimation de densités de probabilité sur chaque paramètre d'entrée, permettant le calcul de marges probabilistes en sortie du code.

Cette dernière approche est plus souple bien que plus délicate à mettre en œuvre. En particulier, le calcul exact des marges est en général impraticable compte-tenu de la complexité des systèmes d'équations modélisés. C'est pourquoi les distributions de probabilités en sortie du code sont estimées par des techniques du type Monte-Carlo. Ces dernières ont besoin de nombreuses applications du code pour fournir des résultats précis.

Nous nous intéressons plus particulièrement à la simulation numérique du choc thermique pressurisé sur une cuve de réacteur nucléaire (illustrée sur la figure 1), qui est un des types

d'accidents envisagés. La démarche sera ensuite généralisée à d'autres scénarios. L'étude de l'intégrité de la cuve d'un Réacteur à Eau Pressurisée (REP) en cas de Choc Thermique Pressurisé (CTP) comporte deux grandes étapes [101; 222] :

1. une analyse thermohydraulique dont l'objectif est la détermination des évolutions temporelles de la température, de la pression et du coefficient de transfert thermique dans l'espace annulaire de la cuve, paramètres influençant le chargement thermique et mécanique sur la paroi de la cuve.

2. une analyse mécanique pour déterminer si le transitoire de choc thermique peut être la source d'une propagation brutale de défauts présents sur la paroi de la cuve, et causer la rupture de celle-ci. Cette analyse est réalisée à partir des courbes obtenues à l'étape précédente.

La phase d'analyse thermohydraulique est simulée par le code de calcul CATHARE (Code Avancé de THermohydraulique pour les Accidents de Réacteurs à Eau, CEA). Cependant, celui-ci est très lent (plusieurs heures par calcul) relativement au code mécanique (quelques minutes), empêchant une exploitation optimale de ce dernier code. De plus cette lenteur exclut a priori toute analyse directe du type Monte-Carlo, nécessaire à l'évaluation des marges d'intérêt.

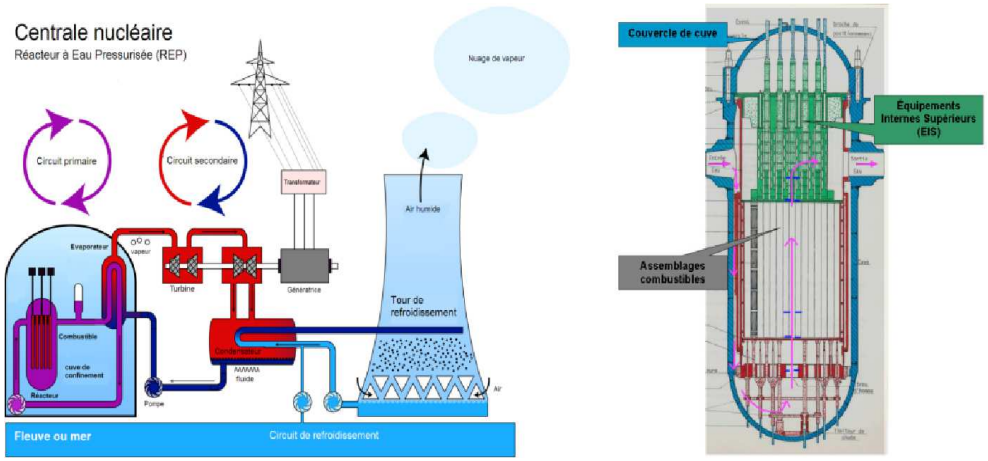

Figure 1: Schéma global d'un réacteur nucléaire et vue la cuve.

Afin de pallier ce problème, une méthode usuelle que nous adoptons ici est de modéliser le premier code par un algorithme très rapide ($\lesssim 1$ seconde), suffisamment précis pour que l'on puisse bénéficier de la substitution. Le code initial étant déjà un modèle d'un phénomène physique, le nouvel algorithme est appelé *métamodèle*, cette terminologie indiquant un niveau d'approximation supérieur (voir la figure 2, illustrant un schéma d'Amandine Marrel légèrement modifié). L'étape de modélisation physique étant absente de ce travail, le terme *modèle* y est également employé – sans ambiguité possible – pour désigner un métamodèle en référence au code physique. Concernant la seconde phase, mécanique, une méthodologie basée sur les surfaces de réponse est présentée dans la thèse de Deheeger [91].

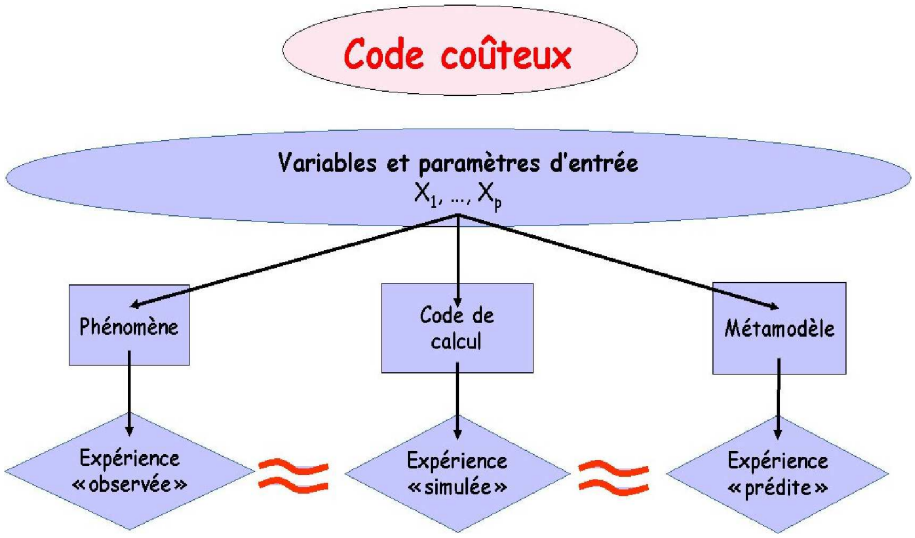

Figure 2: Trois niveaux d'approximation d'un phénomène physique.

Jusqu'à présent, la plupart des études ont été réalisées pour des entrées vectorielles et sorties scalaires [293; 61]. Nous étudions ici le cas des sorties fonctionnelles $1D$ − du code de calcul CATHARE en particulier, bien que les outils développés soit assez généraux. Le code à modéliser dans cette thèse est assimilable à la fonction suivante :

$$\begin{array}{rccc} \phi: & \mathcal{E}_E & \to & \mathcal{C}([a,b]) \\ & x & \mapsto & \phi(x)\,, \end{array}$$

avec $\mathcal{E}_E$ l'espace de variation des paramètres d'entrée, $a, b \in \mathbb{R}, a < b$, $p \in \mathbb{N}^*$ et $\mathcal{C}([a,b])$ l'ensemble des fonctions continues de $[a,b]$ dans $\mathbb{R}$; en effet l'évolution des paramètres physiques se fait de manière continue à l'échelle humaine. Quelques évolutions de températures typiques sont visibles sur la figure 3. Les entrées sont des paramètres décrivant au mieux l'état du système à l'instant initial. Pour des raisons qui seront dévoilées au début du chapitre 2, ϕ *est supposée continue par morceaux*. Cela signifie qu'il existe une partition de $\mathcal{E}_E$ en m ensembles disjoints $E_1, \ldots, E_m$ telle que ϕ soit continue sur chaque E_j. Elle n'est en revanche pas nécessairement continue aux points frontière.

L'objectif de cette thèse est de proposer un algorithme rapide approximant la fonction ϕ.

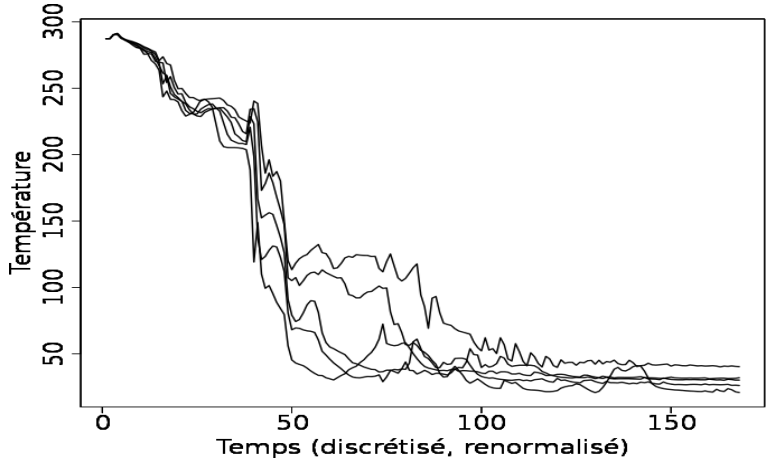

Figure 3: Quelques profils de température en sortie du code CATHARE.

Les données fonctionnelles sont présentes dans bien d'autres domaines que celui de l'industrie nucléaire. En économie on utilise des séries temporelles représentant l'évolution d'une variable d'intérêt au cours du temps. Celles-ci sont assimilables à des fonctions, dont on souhaite prédire l'évolution future. En biologie on parle de données longitudinales lorsque l'on dispose de plusieurs mesures effectuées sur un sujet au cours du temps. Ces données sont assimilables à des fonctions même s'il y a souvent peu de points de discrétisation (d'où l'efficacité des méthodes paramétriques dans ce domaine). Dans l'industrie agro-alimentaire l'évolution des récoltes au cours du temps peut aussi être vue comme une fonction, etc. D'autres cas seront succintement mentionnés au cours de l'état de l'art.

Dans tout ce qui suit nous étudions le cas d'une seule réponse fonctionnelle, mais il y a en réalité plusieurs courbes en sortie du code de calcul CATHARE. Il serait intéressant de construire un modèle multi-sorties tenant compte des corrélations inter-fonctions, mais cette voie n'a pas été explorée. Pour des raisons de simplicité toute application $f : [a, b] \to \mathbb{R}$ intervenant au cours de ce travail est supposée échantillonnée sur une grille régulière $a = t_1 < \cdots < t_D = b$. Cette grille est commune à toutes les courbes et suffisamment fine pour que l'erreur entre l'interpolation sur $f(t_1), \ldots, f(t_D)$ et f soit négligeable.

Cette introduction se divise désormais en trois volets : une fois le problème d'optimisation formalisé, un exemple jouet est présenté afin d'introduire les étapes de la construction du métamodèle. Le plan de la thèse est ensuite annoncé après un état de l'art sélectif des méthodes de régression à sorties fonctionnelles.

0.2 Formulation du problème

Il est nécessaire de formuler précisément le problème avant d'aller plus loin dans la construction du modèle. Les données initiales sont n couples $(x_i, y_i = \phi(x_i)) \in \mathcal{E}_E \times \mathcal{C}([a, b])$, obtenus à l'aide de quelques simulations coûteuses du code CATHARE ϕ ; n varie en général entre 50 et 1000. L'objectif est de construire une fonction

$$\hat{\phi} : \mathcal{E}_E \to \mathcal{C}([a, b])$$

capable de prédire correctement les sorties y pour des entrées x présentes ou non dans les données d'apprentissage. Afin de simuler empiriquement la validation de cet objectif on doit disposer de deux bases de données disjointes :

- une *base d'apprentissage* $BA = (x_i, y_i = \phi(x_i)) \in \mathcal{E}_E \times \mathcal{C}([a,b]), i = 1, \ldots, n_A$,
- une *base de test* $BT = (x'_i, y'_i = \phi(x'_i)) \in \mathcal{E}_E \times \mathcal{C}([a,b]), i = 1, \ldots, n_T$;

seule la première est utilisée pour la construction de $\hat{\phi}$. En pratique les résultats de simulation initiaux sont divisés en deux groupes : $n = n_A + n_T$, souvent avec $n_A \simeq 2n_T$ (Hastie et al. [157]). L'adéquation aux données initiales est ainsi estimée sur BA, et la capacité de généralisation sur BT.

La division du jeu de données pose toutefois un nouveau problème : comment choisir les entrées de la base d'apprentissage ? Celles-ci doivent être les plus différentes possible des entrées de la base de test, sans en être trop éloignées. Plusieurs méthodes sont envisageables pour l'extraction d'un tel plan d'expérience ; Iooss et al. [168] en présentent une basée sur la minimisation d'une mesure de discrépance. On peut aussi choisir n_A couples au hasard parmi les n, construire puis valider un modèle et recommencer cette opération un certain nombre de fois (validation croisée).

En supposant les données correctement séparées en deux groupes, d'apprentissage et de test, l'objectif peut être reformulé : on cherche $\hat{\phi}$ qui minimise

$$f(u_1, \ldots, u_{n_T}) \,,$$

sous les contraintes

$$\forall i = 1, \ldots, n_A \,,\, \hat{\phi}(x_i) = y_i \,,$$

avec $u_i = \|\hat{\phi}(x'_i) - y'_i\| \in [0, +\infty[, i = 1 \ldots n_T$.

La notation $\|.\|$ désigne la norme utilisée sur l'espace des fonctions continues $\mathcal{C}([a,b])$. La fonction f doit alors être choisie pour réaliser une norme sur $[0, +\infty[^{n_T}$. Les choix usuels pour ces normes sont les suivants.

- $f(u_1, \ldots, u_{n_T}) = \sum_{i=1}^{n_T} u_i$ (norme 1, ou "de Manhattan") ;
- $f(u_1, \ldots, u_{n_T}) = (\sum_{i=1}^{n_T} u_i^2)^{\frac{1}{2}}$ (norme 2, euclidienne) ;
- $f(u_1, \ldots, u_{n_T}) = \max_{i=1,\ldots,n_T} u_i$ (norme ∞).

Les contraintes traduisent l'interpolation exacte sur la base d'apprentissage, supposant l'absence de bruit ; dans un cas réel bruité on peut les relâcher légèrement. Le problème d'optimisation est alors établi, mais impossible à résoudre directement par des méthodes existantes : la fonction à optimiser a une image elle-même fonctionnelle. De plus ce problème est mal posé, dans le sens qu'il existe une infinité de solutions (les vecteurs $x_i, i = 1, \ldots, n$ étant supposés distincts) n'ayant pas toutes un intérêt physique.

Au lieu de chercher à résoudre ce problème directement, un modèle est choisi puis ajusté pour s'approcher au mieux du vrai code sur les n entrées x_i.

Cela revient à restreindre $\hat{\phi}$ à une certaine classe de fonctions, qui ne sera cependant pas explicitée. Vijayakumar et Ogawa [327] se placent ainsi dans un RKHS (espace de Hilbert à noyau reproduisant ; d'autres choix sont bien sûr possibles) et proposent trois méthodologies afin de déterminer l'application $\hat{\phi}$ optimale.

Un exemple simple est maintenant présenté afin d'annoncer les différentes étapes de la construction du modèle.

0.3 Exemple jouet et méthodologie

Soient $\alpha, \beta \in [-7, 7]$. Définissons la fonction

$$\begin{array}{rccl} f_{\alpha,\beta} : & [0, 2\pi] & \to & \mathbb{R} \\ & x & \mapsto & \arctan\alpha \cos x + \arctan\beta \sin x\,, \end{array}$$

et supposons que le code de calcul à modéliser associe le couple de paramètres (α, β) à $f_{\alpha,\beta}$. Un certain nombre de simulations doivent être lancées avant de construire un modèle. Supposant ne pas connaître l'expression de $f_{\alpha,\beta}$, on choisit souvent de faire varier les paramètres uniformément dans leurs intervalles de variation : évaluons alors le code sur la grille $]-7, 7[^2$ de pas 0.75. Cela représente $19 \times 19 = 361$ calculs, dont les sorties sont représentées figure 4. En couleur se distinguent quatre modes principaux, correspondant aux combinaisons extrêmes $(\alpha, \beta) = (\pm 7, \pm 7)$; en effet la fonction arctan converge assez vite vers $\pm\frac{\pi}{2}$.

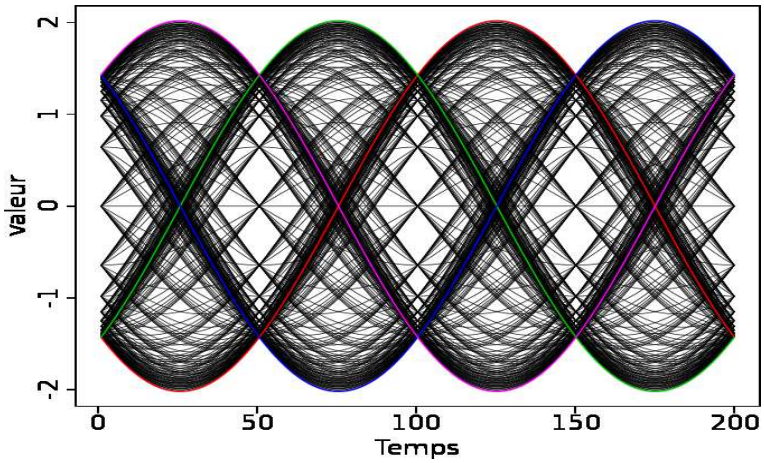

Figure 4: 361 courbes en sortie ; 4 comportements typiques en couleurs.

Ces quatre modes distincts suggèrent de diviser la construction du modèle en quatre petits modèles : un pour chaque comportement. Il faut déterminer les groupes de courbes à partir de l'échantillon obtenu − et éventuellement vérifier la cohérence avec les entrées. C'est un problème de classification non supervisée, encore appelé *clustering*. Supposons cette étape réalisée, et focalisons-nous sur un des quatre groupes (la méthode de clustering sera présentée ultérieurement). Comment associer alors une entrée (α, β) de dimension deux à

une sortie de dimension théorique infinie (discrétisée sur cent points dans cet exemple) ?

Le cas d'une sortie scalaire a été abondamment étudié dans la littérature (voir le livre de Hastie et al. [157] pour un aperçu global des méthodes). Une première idée consiste alors à effectuer l'apprentissage point par point de discrétisation. En effet au niveau algorithmique une fonction est représentée par un vecteur de valeurs discrètes $(f(t_1), \ldots, f(t_D))$. Cette approche est très coûteuse et ne prend pas en compte les corrélations entre points de discrétisation proches. C'est pourquoi nous cherchons plutôt à réduire la dimension des sorties pour se ramener au cas connu.

Revenant à l'exemple d'intérêt, même sans connaître $f_{\alpha,\beta}$ l'allure des courbes suggère d'utiliser une base de fonctions sinusoïdales pour les représenter. En ne gardant dans la représentation que les coefficients significatifs, la dimension est effectivement réduite. En décomposant sur la base de Fourier tronquée $(1, \cos, \sin)$, on s'aperçoit que le coefficient de la fonction constante est nul. Nous conservons donc seulement deux coefficients, et remplaçons provisoirement les sorties $f_{\alpha,\beta}$ par $(c_\alpha, c_\beta) = (\langle f_{\alpha,\beta}, \cos\rangle, \langle f_{\alpha,\beta}, \sin\rangle)$. Il suffit alors d'apprendre la relation

$$\begin{array}{rccc} g: & \mathbb{R}^2 & \to & \mathbb{R}^2 \\ & (\alpha, \beta) & \mapsto & (c_\alpha, c_\beta)\,. \end{array}$$

Dans l'exemple présenté cela revient à apprendre la fonction arctangente, problème facilement résolu. De nouvelles courbes peuvent alors être prédites : $\text{prédiction}(x) = g(x)\begin{pmatrix}\cos\\ \sin\end{pmatrix}$.

> De cet exemple introductif se dégagent deux axes majeurs pour notre travail : la classification des courbes, et la réduction de leur dimension. Les chapitres 2 et 3 seront consacrés à ces problématiques respectives.

La méthodologie de construction du modèle est la suivante :

1. clustering des n courbes en sortie (et éventuellement des entrées si nécessaire) ;
2. classification supervisée des entrées dans chaque cluster : $c(x_i)$ désigne les indices du groupe contenant x_i.
3. pour chaque cluster $C = \{i_1, \ldots, i_m\}$:
   (a) réduire la dimension : $\forall i \in C\,,\ y_i \leftarrow z_i \in \mathbb{R}^d$;
   (b) apprendre une fonction de régression g_C : $\forall i \in C\,,\ g_C(x_i) \simeq z_i$;

Pour une nouvelle entrée x, une courbe est alors prédite par

$$\hat{\phi}(x) = R_{c(x)}(g_{c(x)}(x))\,,$$

où $R_C : \mathbb{R}^d \to \mathcal{C}([a,b])$ est une fonction de reconstruction associée à la réduction de dimension dans le cluster C. Les étapes 1 et 3-a peuvent être fusionnées dans certains cas. Nous

choisissons cependant de les séparer, pour plus de clarté et de modularité.

Nous avons jugé visuellement qu'il y avait quatre groupes de courbes, mais comment formaliser cette étape ? Le chapitre 2 s'intéresse à la classification automatique en un certain nombre de groupes, tandis que le chapitre 3 propose quelques méthodes générales de réduction de la dimension (cette dernière pouvant être estimée).

Avant d'annoncer les autres chapitres, une bibliographie sélective autour des métamodèles fonctionnels est présentée.

0.4 État de l'art

Les données dont nous disposons étant très complexes (formes irrégulières, beaucoup de points de discrétisation), nous ne parlerons pas des nombreuses méthodes imposant un modèle paramétrique sur les courbes. La majeure partie des modèles non paramétriques trouvés dans la littérature se mettent sous la forme

$$\hat{\phi}(x,t) = \mu(x,t) + \sum_{j=1}^{m} c_j(x)\, f_j(t)\,,$$

avec $x \in \mathcal{E}_E$ (état initial) et $t \in [a,b]$ (temps), μ étant une tendance moyenne simple, souvent choisie indépendante de x. Les f_j sont des fonctions continues de $[a,b]$ dans $\mathbb{R}$, $m \in \mathbb{N}^*$. Deux grandes familles de modèles se distinguent alors : celles où ces dernières applications sont optimisées pour s'adapter aux données, et celles qui utilisent une base de fonctions a priori moins adaptée.

0.4.1 Famille $\{f_j\}$ adaptée aux données

Bien qu'il soit probablement possible d'optimiser les fonctions f_j de différentes manières, la régression linéaire offre un cadre simple pour déterminer ces dernières. Cette technique de régression est introduite dans un premier temps pour le cas vectoriel, avant de mentionner les travaux l'adaptant au contexte fonctionnel.

En notant $X \in \mathbb{R}^{n\times(p+1)}$ la matrice des entrées en lignes avec des 1 dans la première colonne et $Y \in \mathbb{R}^{n\times q}$ la matrice des vecteurs réponses en lignes ($q \in \mathbb{N}^*$ correspondant à la dimension en sortie), la régression linéaire revient à écrire $Y \simeq X\beta$, avec β de taille $(p+1)\times q$. On cherche donc la matrice $\hat{\beta} \in \mathbb{R}^{(p+1)\times q}$ minimisant $\|Y - X\beta\|$. La solution suivante est obtenue quand ${}^tXX$ est inversible.

$$\hat{\beta} = ({}^tXX)^{-1}\,{}^tXY\,.$$

On peut étendre la méthode en effectuant une transformation préalable sur X (le plus souvent polynomiale) : la ligne $X_i = (x_{i0}, \ldots, x_{ip})$ est remplacée par $f(X_i) = (f_0(X_i), \ldots, f_s(X_i))$ avec $i = 1, \ldots, n$, $s \in \mathbb{N}^*$, $f_0 = 1$ et $f_1, \ldots, f_s$ des applications (au moins) continues. La solution s'écrit toujours de la même manière :

$$\hat{\beta} = ({}^tf(X)f(X))^{-1}\,{}^tf(X)Y\,,$$

avec le modèle correspondant

$$\hat{\phi}(x) = {}^t(1\ x)\hat{\beta}\,.$$

Des algorithmes efficaces de résolution (applicables notamment lorsque ${}^tf(X)f(X)$ n'est pas inversible) sont présentés aux chapitres 2 et 15 du livre de Press et al. [255].

Plusieurs auteurs ont appliqué cette méthode de régression au cas fonctionnel, essentiellement pour la norme L^2. En posant Y le vecteur colonne des n fonctions en sortie, on reprend le même problème : minimiser

$$\|Y - X\beta\|\,,$$

avec β vecteur colonne de $p+1$ fonctions continues. Il y a alors plusieurs manières de résoudre le problème, guidées par le fait qu'on apprécie d'obtenir des fonctions plutôt lisses dans β. Ainsi, Ramsay et Silverman [258] proposent au chapitre 13 de résoudre le problème directement pour chaque point de discrétisation puis de lisser le résultat obtenu. Li et al. [210] utilisent des B-splines pénalisées (appelées P-splines) à cet effet. Alternativement, Ramsay et Silverman suggèrent de se placer dans le sous espace engendré par quelques fonctions lisses orthonormées : on effectue alors la régression vectorielle sur les coefficients, la régularité étant assurée par la décomposition sur une base. Enfin, une approche par régularisation est envisagée, cherchant à minimiser $\|Y - X\beta\| + \lambda PEN(\beta)$ où PEN est un opérateur différentiel, par exemple du type $f \mapsto \left(\int |f^{(m)}|^2\right)$.

Shen et Faraway [289] utilisent le modèle de régression linéaire ponctuel pour estimer le vecteur de fonctions β, et posent deux hypothèses :

$$\begin{aligned} H_0 &: \phi(x,t) = X_1\hat{\beta}_1(t) + \varepsilon(t)\,, \text{ et} \\ H_1 &: \phi(x,t) = X_1\hat{\beta}_1(t) + X_2\hat{\beta}_2(t) + \varepsilon(t)\,, \end{aligned}$$

où X_1 désigne la matrice des n entrées à laquelle on retire ℓ_1 colonnes, $0 \leq \ell_1 < p$. X_2 est une matrice obtenue à partir de X, dans laquelle on ne conserve que ℓ_2 colonnes parmi les ℓ_1 colonnes précitées, $1 \leq \ell_2 \leq \ell_1$. L'objectif est de déterminer si les paramètres présents dans X_2 sont pertinents pour la régression. Un test s'inspirant du F-test classique [9] est proposé :

$$\mathcal{F} \propto \frac{rss_0 - rss_1}{rss_1}\,.$$

Plus cette dernière quantité est grande, plus il y a de différences entre les deux modèles et donc plus X_2 est importante. rss pour "residual sum of squares" désigne $\sum_{i=1}^n \|y_i - \hat{y}_i\|^2$, avec $\hat{y}_i$ l'estimation de y_i pour le modèle considéré. Ce test permet de valider le nombre de variables d'entrée à incorporer dans le modèle final. Cela est intéressant dans le cas où l'on a beaucoup de paramètres d'entrée, même si le modèle reste le même que le précédent sans la pénalisation. Les auteurs présentent une application possible en ergonomie automobile, où les angles formés par certaines parties du corps (le coude par exemple) sont enregistrés en tant que sorties fonctionnelles.

En résumé, les modèles présentés jusqu'alors s'écrivent

$$\hat{\phi}(x,t) = \mu(t) + \sum_{j=1}^{p} x_j \, f_j(t) \, ,$$

avec $x_j = j^{eme}$ coordonnée de l'entrée $x \in \mathcal{E}_E$. Aucun apprentissage n'est effectué sur les coefficients $c_j(x) = x_j$ (projection sur la j^{eme} composante), seules les courbes f_j sont optimisées.

Cette modélisation est erronée dans le cas probable où les sorties du codes ne varient pas linéairement avec les entrées. Elle conserve tout de même certains avantages :

- des intervalles de confiance en prédiction peuvent être facilement calculés ; voir par exemple Faraway [119] ;
- le modèle est facile à interpréter : la variation d'un paramètre d'entrée agit linéairement sur une courbe f_j.

De plus l'idée d'optimiser les fonctions f_j sera reprise ultérieurement, dans d'autres méthodes où les coefficients varieront plus finement.

0.4.2 Modélisation via une base de fonctions

Les modèles présentés dans cette section s'articulent autour de la réduction de dimension suivie d'une étape de régression. Plus précisément, la réduction de dimension s'effectue par décomposition sur une base de fonctions (les f_j), en général orthonormée, dont on ne garde que les d premiers coefficients. Contrairement au développement du paragraphe précédent les coefficients sont estimés par apprentissage statistique. Cela donne de meilleurs résultats en pratique malgré le fait que les applications f_j ne soient pas optimisées, ou très sommairement. Quelques auteurs ont suggéré d'utiliser la validation croisée afin de déterminer d [165; 224], problème qui reste délicat.

Li et al. [209] construisent un métamodèle de la forme

$$\hat{\phi}(x,t) = \mu(t) + \sum_{j=1}^{d} c_j(x) f_j(t) \, ,$$

où les coefficients c_j ainsi que les fonctions f_j sont optimisés conjointement. Cette optimisation s'effectue en recherchant les projections dans l'espace de départ les plus influentes sur les composantes des courbes y_i (relativement aux f_j), l'influence d'une entrée étant mesurée par analyse de sensibilité (globale) [297; 274]. Une application à l'évolution du climat en Chine est présentée. Yoo et Cook [348] introduisent une approche visant à pallier les limitations de la précédente, notamment au niveau du réglage des paramètres. Dans cette thèse nous préférons nous concentrer sur un autre type de réduction de dimension, plus général, qui ne tient pas compte des entrées dans un premier temps.

Govaerts et Noël [147] proposent de décomposer les sorties du code sur une base de B-splines cubiques [85], puis d'apprendre les d relations entrées-coefficients par une régression

linéaire multivariée comme au paragraphe précédent. Le modèle s'écrit alors

$$\hat{\phi}(x,t) = \mu(t) + \sum_{j=1}^{d} \left(\beta^{(j)} + \langle x, \beta_j \rangle\right) f_j(t)\,,$$

avec $\{f_j\}_{j=1,\dots,d}$ base de B-splines tronquée, $x \in \mathcal{E}_E$ et $\beta^{(j)} \in \mathbb{R}, \beta_j \in \mathbb{R}^p$ pour $j = 1, \dots, d$. La méthode est appliquée à un cas industriel modélisant les propriétés d'un métal en fusion en fonction de sa composition.

Chiou et al. [68] utilisent une idée qui sera exploitée également dans d'autres méthodes : la réponse fonctionnelle est écrite comme somme d'une partie moyenne dépendant essentiellement du temps, et d'une partie corrective dépendant du temps et des variables d'entrée. Le prédicteur est de la forme

$$\hat{\phi}(x,t) = \beta_0(\langle \alpha_0, x \rangle)\mu(t) + \sum_{j=1}^{d} \beta_j(\langle \alpha_j, x \rangle) f_j(t)\,,$$

avec pour f_j les composantes principales fonctionnelles [258], $\beta_0\mu$ désignant la moyenne des données modulée par une fonction d'une combinaison linéaire des composantes du vecteur d'entrée. Si l'on excepte le premier terme (sorte de centrage sophistiqué), les applications β_j lient des projections dans l'espace des entrées aux coefficients de décomposition des courbes. Plus exactement, β_j effectue un lissage non paramétrique basé sur les points $(\langle \alpha_j, x_i \rangle, \langle f_j, y_i \rangle), i = 1, \dots, n$. Les α_j et β_j peuvent être optimisées exactement comme dans l'algorithme PPR (voir le paragraphe 4.1.6.3), ou via la méthode développée par Chiou et al. [69; 67]. Ce modèle est utilisé par les auteurs pour décrire l'évolution de de la population d'une certaine espèce de mouche au cours du temps, en fonction de l'environnement. On peut en fait le considérer comme un cas particulier du métamodèle que nous proposons, en restreignant le nombre de termes à 1 dans l'algorithme PPR. Nous ne modélisons cependant pas la partie moyenne (la distribution des courbes en sortie pouvant être multimodale).

Bayarri et al. [20] proposent d'utiliser une base de fonctions ondelettes pour réduire la dimension. Ces fonctions représentent un bon compromis lorsque l'on veut modéliser à la fois un comportement global et des variations localisées dans le temps (ou l'espace en $2D$ etc). L'apprentissage des coefficients s'effectue par la méthode du krigeage [301], dont les paramètres sont estimés par analyse bayésienne : des algorithmes MCMC (Markov Chain Monte-Carlo) permettent d'obtenir les valeurs moyennes a posteriori à partir des lois a priori sur chaque (hyper)paramètre. Un code industriel modélisant l'évolution des suspensions d'une voiture est présenté pour tester la méthode.

Marrel [223] s'est intéressée pendant sa thèse à un code de calcul comportant une sortie spatiale $2D$. Cette dernière est décomposée sur une base d'ondelettes (comme précédemment en dimension un). Ensuite, la modélisation des coefficients est hybride : les d_1 premiers sont approximés par des processus gaussiens, tandis que les d_2 suivants sont estimés par régression linéaire simple ; enfin, les d_3 derniers coefficients sont modélisés par des constantes. Une version modifiée de l'algorithme de Welch et al. [339] est implémentée afin d'estimer les hyperparamètres par maximum de vraisemblance. La méthode est appliquée pour le calcul d'indices de sensibilité, notamment sur un code modélisant la migration de polluants sur un

ancien site de stockage de déchets radioactifs.

Monestiez et Nerini [238] étudient le problème d'apprentissage statistique à sorties fonctionnelles de manière théorique. Ils imposent la forme suivante au modèle :

$$\hat{\phi}(x,t) = \mu(t) + \sum_{i=1}^{n} B_i^{(x)}(y_i)(t)\,,$$

avec $B_i^{(x)}(y_i)(t) = \langle \beta_i^{(x)}(.,t), y_i \rangle$ pour $t \in [a,b]$. Il s'agit d'une généralisation de l'estimateur linéaire $\mu + \sum_{i=1}^{n} \lambda_i(x) y_i$ avec λ_i fonctions polynomiales par exemples. En effet en posant $\beta_i^{(x)}(.,t) = \lambda_i(x)\delta_t$ (pic delta de Dirac), on retrouve ce dernier estimateur. Après un certain nombre de manipulations algébriques un système de $n+1$ équations fonctionnelles est obtenu. Celles-ci ne pouvant être résolues directement, une base de fonctions est utilisée et le système est alors à résoudre dans $\mathbb{R}^d$. La méthode revient finalement à écrire $\hat{\phi}(x,t) = \mu(t) + \sum_{j=1}^{d} c_j(x) f_j(t)$. $\{f_j\}_{j=1,\ldots,d}$ est une base orthonormée quelconque et les coefficients $c_j(x)$ sont estimés par cokrigeage (variante du krigeage où l'on apprend plusieurs sorties en même temps, tenant compte de leurs inter-corrélations). Une application au suivi de la température de l'océan grâce à des sondes disposées sur des éléphants de mer est proposée (en choisissant la base des polynômes de Legendre).

En conclusion ces modèles sont plus généraux que ceux du paragraphe précédent. En effet les fonctions f_j bien que non optimisées ne sont pas limitées en nombre, permettant d'approcher finement de nombreux types de codes. Nous utilisons un modèle décomposant les sorties y_i sur la base ACP fonctionnelle (voir le paragraphe 3.2.1.1) et approximant les coefficients par Projection Pursuit Regression [131] comme outil de comparaison systématique pour les tests, ces deux étapes étant assez performantes et rapides. On peut cependant se demander si la décomposition sur une base constitue une bonne représentation des données, et si on ne peut pas en trouver de meilleure. Nous présenterons des alternatives pour tenter de répondre à cette question au chapitre 3.

Finalement, une approche possible est d'exprimer $\hat{\phi}$ par une combinaison linéaire des n fonctions y_i obtenues en sortie du code. On peut s'inspirer de l'estimateur de Nadaraya-Watson [241; 336] comme suit :

$$\hat{\phi}(x,t) = C \sum_{i=1}^{n} K_{NW}(\|x - x_i\|)\, y_i(t)\,,$$

avec $C = \left(\sum_{i=1}^{n} K_{NW}(\|x - x_i\|)\right)^{-1}$. x_i désigne à nouveau une des n entrées de la base d'apprentissage. K_{NW} est une fonction continue de $[0, +\infty[$ dans lui-même, non nulle en 0. Cette méthode ne semble pas avoir été utilisée en pratique ; selon nos tests elle présente tout de même un intérêt dans certains cas, sous une version modifiée. Notons qu'elle ne se met pas sous la forme générale indiquée ci-dessus.

0.5 Plan de la thèse

Étant donnés n couples entrées-sorties (x_i, y_i), l'objectif est de construire une fonction $\hat{\phi} : \mathcal{E}_E \to \mathcal{C}([a,b])$ vérifiant $\hat{\phi}(x_i) \simeq y_i$. Cette fonction est utilisée ensuite pour prédire le comportement du code sur de nouvelles entrées. Les chapitres de cette thèse correspondent assez naturellement aux étapes de construction de $\hat{\phi}$ décrites au paragraphe 0.3.

> À chaque chapitre correspond des algorithmes implémentés dans le langage R, constituant un package disponible sur CRAN : modelcf [14]. Cet outil est utilisable afin d'aider à répondre à l'objectif industriel initial.

Le chapitre 1 développe une représentation des données par le biais d'un graphe. Cette dernière est utile tout au long des deux parties suivantes.

1. Classification non supervisée des n couples (x_i, y_i) en K clusters $C_1, \dots, C_K$ (ensembles d'indices), suivie de la classification supervisée des entrées dans chaque groupe : $c(x_i)$ désigne le cluster contenant x_i.

Le chapitre 2 est consacré à la recherche d'une méthode générale de détermination automatique du nombre de groupes K (et du partitionnement associé). La division des données en K sous-ensembles disjoints facilite la construction du modèle final : l'apprentissage se fait alors sur des groupes de fonctions plus spécialisées. Remarquons que l'étape de clustering est très rarement mentionnée ou utilisée en lien avec l'approximation d'un code de calcul. Cette étape peut alors être considérée comme une contribution au domaine des métamodèles.

2. Réduction de la dimension dans chaque cluster $C = \{i_1, \dots, i_m\}$:

$$\forall i \in C\,,\ y_i \leftarrow z_i \in \mathcal{E}_R\,.$$

Le chapitre 3 présente quelques techniques de réduction de dimension alternatives à l'ACP fonctionnelle. Les sorties du code de calcul sont supposées échantillonnées sur une variété fonctionnelle lisse (voir l'annexe A à ce sujet), et le but est de paramétrer cette variété via la recherche des z_i. La quasi totalité des métamodèles du type choisi dans ce travail utilise une base de fonctions orthonormée. Concernant la modélisation des codes de calcul, cette étape peut donc également être considérée comme une contribution.

3. Pour une nouvelle entrée x, une courbe est alors prédite par

$$\hat{\phi}(x) = R_{c(x)}(g_{c(x)}(x))\,,$$

où $g_C : \forall i \in C\,,\ g_C(x_i) \simeq z_i$ est une fonction de régression (vectorielle) transformant les entrées en les coordonnées réduites des sorties, dans le cluster C. L'application $R_C : \mathcal{E}_R \to \mathcal{C}([a,b])$ reconstitue une courbe à partir de sa représentation d-dimensionnelle dans ce même groupe. Nous choisissons au chapitre 4 une fonction de régression g_C suffisamment générale, performante et rapide. L'application de reconstruction est quant à elle déterminée à partir des méthodes de réduction de dimension sélectionnées.

Finalement, des tests divers sont effectués au chapitre 5. Ceux-ci portent sur le code de

calcul CATHARE pour la validation du modèle dans le cadre industriel, mais également sur des jeux de données supplémentaires pour appliquer le modèle dans d'autres contextes.

Rappelons que l'étape de modélisation au cœur de cette thèse s'inscrit dans un cadre plus général d'analyse d'incertitudes sur le code de calcul. En particulier, deux types d'études sont fréquemment menées :

- l'analyse de sensibilité, qui vise à rechercher les variables d'entrées les plus influentes par le calcul de grandeurs caractéristiques bien choisies ;
- la propagation d'incertitudes, qui vise à évaluer l'impact sur l'incertitude en sortie connaissant l'incertitude en entrée.

Sobol [297] présente les indices du même nom en analyse de sensibilité ; ceux ci sont les plus fréquemment utilisés. Concernant la propagation d'incertitudes, on trouve une bonne introduction accompagnée de références dans le livre de De Rocquigny et al. [88]. La note technique de Marques et Auder [221] présente ce type de calculs utilisant un métamodèle, dans le contexte de l'approximation du code CATHARE.

La problématique des sorties fonctionnelles en analyse de sensibilité a été abordée par Campbell et al. [54], qui proposent de décomposer les fonctions sur une base avant de calculer les indices de Sobol sur les coefficients. Cette approche est affinée par Lamboni et al. [197] qui utilisent l'ACP fonctionnelle pour définir des indices de sensibilité généralisés tenant compte de tous les coefficients. Marrel et al. [224] s'intéressent à l'analyse de sensibilité d'un modèle à entrées vectorielles et sorties fonctionnelles $2D$. Ces dernières sorties sont décomposés sur une base d'ondelettes, puis les coefficients sont modélisés par des processus gaussiens [223] pour simuler un grand nombre de cartes $2D$. Les indices de Sobol peuvent ainsi être calculés ponctuellement, fournissant des visualisations $2D$ de l'influence de chaque paramètre. Des courbes d'indices de sensibilité peuvent être tracées d'une manière similaire pour des fonctions $1D$, mais nous ne les présentons pas ici.

En résumé, la thèse s'organise comme suit, les numéros étant ceux des chapitres.

1. Représentation des données dans un graphe.
2. Clustering des entrées-sorties.
3. Réduction de la dimension des sorties.
4. Régression : entrées $\rightarrow$ représentations réduites,
   apprentissage de la fonction de reconstruction.
5. Applications : modélisation et analyse de sensibilité sur le code CATHARE.

Note pour la lecture :

- les encadrés en noir résument les points importants aux endroits clé de la thèse (tout comme dans cette introduction).
- les encadrés en bleu donnent des renseignements sur le contenu des parties d'un chapitre, indiquant notamment les contributions personnelles ;

Cette thèse comporte de nombreux éléments bibliographiques, dont un certain nombre ne sont pas directement utilisés. Ils sont toutefois présentés (en général brièvement) afin de donner un aperçu assez complet du sujet – sans rechercher l'exhaustivité.

Une partie de ces méthodes a été implémentée à des fins de comparaisons avec les algorithmes retenus dans le package R.

Chapitre 1

Représentation des données dans un graphe

Afin de classer les entrées-sorties puis de représenter efficacement les courbes y_i par des vecteurs de $\mathbb{R}^d$, plusieurs options sont possibles. En se concentrant sur la partie fonctionnelle des données on peut effectuer une simple analyse en composantes principales, puis projeter les courbes pour obtenir des représentations vectorielles que l'on classerait ensuite. Cela renverse l'ordre chronologique proposé, en plus d'être inadéquat dans certains cas comme nous le verrons. Afin de respecter l'ordre fixé, deux grand axes sont alors envisageables :

- utiliser une technique de clustering directement dans l'espace fonctionnel (voir le paragraphe 2.3) ;
- utiliser une représentation alternative des − (di)similarités entre − fonctions, puis classer les courbes dans cette nouvelle structure.

Nous privilégions cette dernière approche, principalement pour que le modèle final reste le plus général possible. Plus précisément, les (entrées−)sorties constituent les sommets d'un graphe dont la construction est détaillée ci-après. Les graphes de voisinages aux paragraphes 1.1.2.3 et 1.1.3, ainsi que la distance introduite en 1.2.1 sont utilisés tout au long des deux chapitres suivants.

La notion de graphe comme objet mathématique est introduite dans un premier temps. Nous présentons ensuite des algorithmes permettant de définir sa structure. Dans un second temps, une marche aléatoire sur le graphe représentant les données permettra d'introduire une nouvelle distance appropriée à la tâche de clustering. Le processus de sélection des arêtes et de leurs poids sera finalement détaillé.

À l'exception de la section 1.3, ce chapitre est en grande partie bibliographique.

Plan du chapitre :

1. **Construction du graphe de voisinage.**
2. **Introduction d'une distance sur ce graphe.**
3. **Détermination de l'échelle d'observation des données.**

1.1 Construction

La construction du graphe est développée ici, des définitions de base à la sélection des arêtes. Le lecteur est renvoyé aux livres de Bondy et Murty [36] et Diestel [102] pour une introduction complète de la théorie des graphes.

1.1.1 Définitions

Un graphe est un couple $G = (V, E)$ où V est un ensemble fini d'éléments appelés sommets. L'ensemble des arêtes E contient des couples (u, v) avec $u, v \in V$, modélisant les relations entre les sommets. Les ensembles V et E sont en général clairs selon le contexte, et ne sont donc pas systématiquement rappelés. Nous aurons besoin des quelques notions suivantes.

Définition 1.1.1 *Un graphe est* orienté *si on fait la distinction entre les arêtes* (u, v) *et* (v, u) *avec* $u, v \in V$.

Ces dernières arêtes peuvent alors être étiquetées différemment ou ne pas exister simultanément. Dans le cas non orienté on écrit (u, v) ou (v, u) suivant que l'on choisit de partir de u ou de v ; ces deux couples désignent la même arête, qui est notée $\{u, v\}$ lorsqu'il n'y a pas lieu de choisir un sommet de départ.

Définition 1.1.2 *Un* chemin *dans un graphe désigne une succession d'arêtes* $a_1 = (u_1, v_1), \ldots, a_m = (u_m, v_m)$ *avec* $\forall i = 1, \ldots, m-1, v_i = u_{i+1}$. *Si de plus* $v_m = u_1$ *on parle d'un* cycle.

Définition 1.1.3 *Un graphe non orienté est dit* connexe *si pour toute paire de sommets* $\{u, v\}$ *il existe un chemin de* u *vers* v.

Le cas des graphes orientés est plus complexe ; on distingue trois niveaux de connexité.

Définition 1.1.4 *Un graphe orienté* $G = (V, E)$ *est dit*

- faiblement connexe *si le graphe non orienté obtenu en transformant chaque arête* (u, v) *en* $\{u, v\}$ *est connexe ;*
- connexe *si pour toute paire de sommets* $\{u, v\}$ *il existe un chemin de* u *vers* v ***ou*** *de* v *vers* u *;*
- fortement connexe *si pour toute paire de sommets* $\{u, v\}$ *il existe un chemin de* u *vers* v ***et*** *de* v *vers* u *;*

La figure 1.1 montre un graphe satisfaisant la seconde définition (et donc la première, mais pas la troisième). Nous utilisons essentiellement la première définition aux chapitres 2 et 3, bien que les graphes considérés soient en général orientés. En effet, les voisinages sont presque toujours rendus symétriques afin que les trois définitions soient équivalentes. Un algorithme recherchant les parties connexes est alors très simple à écrire. Dans le cas où la symétrie n'est pas imposée, la forte connexité peut être testée par la recherche des plus courts chemins dans le graphe.

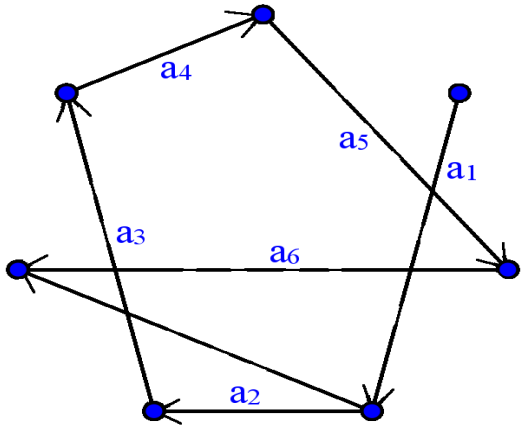

Figure 1.1: Graphe connexe, non fortement connexe.

1.1.2 Graphes de voisinage basiques

Nous supposons à présent disposer de n points $u_1, \ldots, u_n$ dans $\mathcal{E}_E$ ou $\mathcal{C}([a,b])$ (ou une fusion de ces deux espaces) ; l'arête (u_i, u_j) sera systématiquement abrégée en (i,j). Plusieurs types de graphes peuvent être construits, en fonction des caractéristiques du problème.

1.1.2.1 Graphe complet

La méthode la plus simple consiste à connecter tous les sommets deux à deux. On parle alors de graphe complet, illustré sur la figure 1.2.

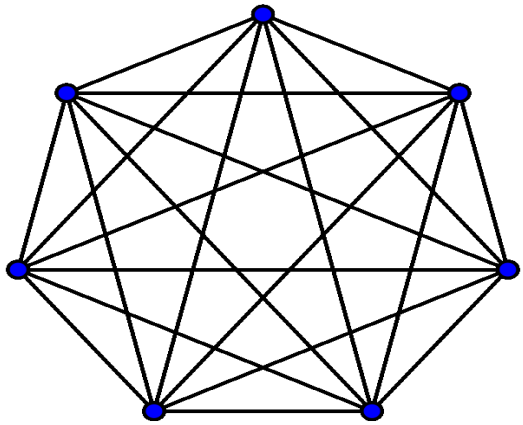

Figure 1.2: Graphe complet à 7 sommets (K_7).

Ceci est clairement inadapté pour la réduction de dimension, car la topologie des données n'est pas mise en évidence. En revanche, c'est une représentation possible pour l'étape de clustering. Les arêtes (i,j) peuvent par exemple avoir pour poids $e^{\frac{-\|u_i-u_j\|^2}{\sigma^2}}$ où σ est un réel positif. Cette quantité correspond à la similarité standard entre les sommets utilisée dans divers algorithmes. Le paramètre global σ contrôle l'échelle des similarités, et est difficile voire impossible à régler. En effet on ne peut choisir un unique σ pour des données de densité variable. La section 1.3 propose quelques stratégies pour adapter σ localement, et ainsi pallier dans une certaine mesure ce dernier problème.

1.1.2.2 ϵ-graphe

Contrairement au cas précédent, seuls les sommets à distance inférieure à ϵ de u_i seront reliés à ce dernier. Le problème d'échelle évoqué en fin de paragraphe demeure, ϵ devant idéalement varier localement, auquel cas sélectionner simplement le nombre de voisins sera plus approprié comme nous le verrons ensuite. Si ϵ est trop petit, le graphe résultant va comporter beaucoup de composantes connexes. En revanche, trop augmenter ϵ ramènerait au cas précédent. En pratique, pour des données suffisamment homogènes, on peut estimer une bonne valeur de cette constante à l'aide de l'écart type du nuage de points par exemple.

La figure 1.3 montre un exemple d'ϵ-graphe dans le carré unité, avec 50 points et $\epsilon = 0.15$.

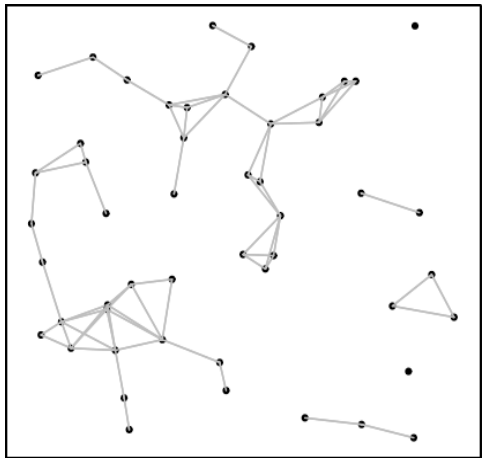

Figure 1.3: ϵ-graphe sur un échantillon uniforme.

1.1.2.3 Graphe des k plus proches voisins

Cette troisième approche vise à pallier les inconvénients des deux dernières. Au lieu de fixer un rayon autour de chaque élément, on sélectionne les k voisins les plus proche au sens de la norme euclidienne (dans cette thèse et en général). Ainsi les voisinages s'adaptent naturellement à la topologie locale. Cependant, le choix de k est délicat, ce dernier devant en principe varier en fonction du point. Nous examinons cette possibilité dans la section suivante. L'heuristique $k = \lceil \sqrt{n} \rceil$ donne des résultats acceptables dans la plupart des cas pratiques, c'est pourquoi nous ne choisissons pas toujours la méthode adaptative présentée ultérieurement.

Afin de rendre les voisinages symétriques (u_i voisin de $u_j \Rightarrow u_j$ voisin de u_i), il est possible de retirer toutes les connexions assymétriques : $(i, j) = (j, i)$ existe si et seulement si u_j est parmi les k plus proches voisins de u_i et réciproquement. Les arêtes (i, j) et (j, i) sont identifiées car le sens de parcours sur le graphe n'est alors plus pertinent. Le graphe résultant est dit des "k plus proches voisins mutuels" (*mutual k nearest neighbors* en anglais). Il permet de faciliter les opérations ultérieures dans le graphe, notamment la recherche des parties connexes. De plus, ce graphe étant plus déconnecté que l'original, il est bien indiqué à des fins de classification. Les figures 1.4 et 1.5 montrent respectivement le graphe des 5 plus proches voisins simples et mutuels, pour un jeu de 50 points tirés uniformément dans le carré $[0, 1]^2$.

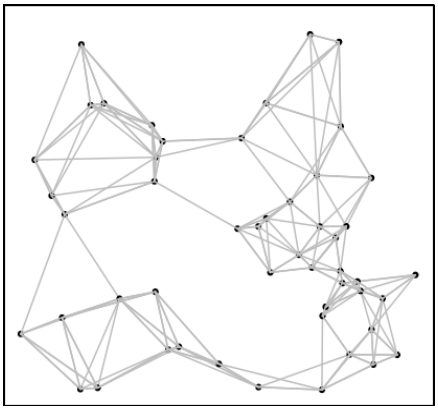

Figure 1.4: Graphe des 5 plus proches voisins.

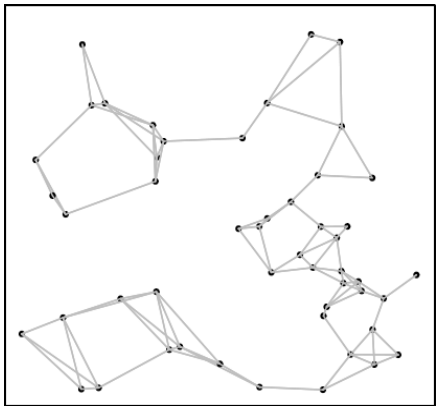

Figure 1.5: Graphe des 5 plus proches voisins mutuels.

1.1.3 Graphes de voisinage évolués

On peut raffiner la construction du graphe, sur la base de celui des k plus proches voisins vu précédemment. L'idée consiste à faire varier k localement pour mieux s'adapter à la géométrie des données, u_i ayant ainsi k_i voisins. Trois approches sont présentées ici : la première est basée sur la notion de visibilité, et les deux autres déterminent k_i en fonction de l'erreur réalisée par approximation linéaire locale.

Les graphes de ce paragraphe visent à représenter la topologie des données. Ils sont utilisés en lien avec les algorithme du chapitre 3.

1.1.3.1 Graphe de visibilité

Lin et al. [213] choisissent une méthode en accord avec leur algorithme de réduction de dimension (voir le paragraphe 3.2.3.2), basée sur la notion de visibilité, elle-même utilisant les angles entre éléments dans $\mathbb{R}^D$. La méthode est directement transposable à $\mathcal{C}([a,b])$, mais nous écrivons cette fois $\mathbb{R}^D$ (espace des fonctions discrétisées) pour rester conforme à l'article. L'angle entre f et h dans $\mathbb{R}^D$ étant pris non orienté, il est compris entre 0 et π et déterminé par son cosinus :

$$\begin{aligned}\cos(f,g) &= \frac{\langle f,h\rangle}{\|f\|\|h\|} \\ &= \frac{\sum_{i=1}^{D} f_i h_i}{\sqrt{\sum_{i=1}^{D} f_i^2}\sqrt{\sum_{i=1}^{D} h_i^2}}.\end{aligned}$$

Deux paramètres entiers peuvent être entrés par l'utilisateur, k_{min} et k_{max} – ce dernier indiquant la taille maximale d'un voisinage – avec $k_{min} \leq k_{max}$. k_{min} (nombre minimal de sommets dans un voisinage) est optionnel et fixé à un dans la version initiale de l'algorithme ; il peut cependant être intéressant d'imposer un seuil minimal sur le nombre de voisins, d'où son introduction. Le graphe est alors construit en deux temps, à partir de celui des k_{max} plus proches voisins :

1. "élagage de visibilité" dans chaque voisinage ;

2. suppression des arêtes de "court-circuit".

L'étape 1 correspond à la suppression des voisins "occultés" par d'autres sommets plus proches, comme illustré sur la figure 1.6. Dit d'une manière imagée, deux voisins de u ne peuvent coexister dans le champ de vision d'un personnage qui se trouverait sur u dans le graphe.

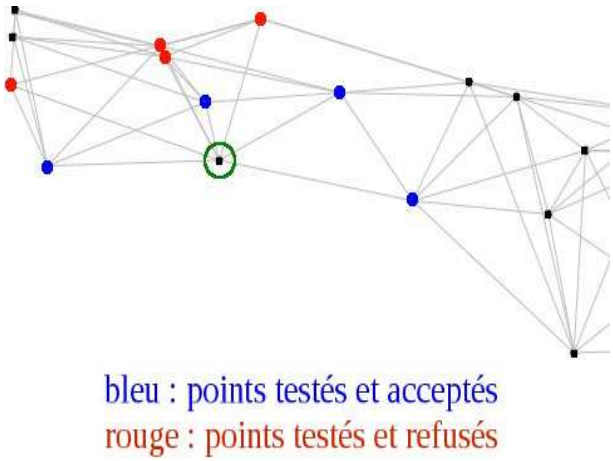

Figure 1.6: Suppression des sommets en rouge, "éclipsés" par ceux en bleu.

Mathématiquement, cette étape s'exécute comme suit pour un sommet u_i ayant pour voisins (classés par distance à u_i croissante) $v_1, \ldots, v_{k_{max}}$:

1. initialiser la liste de voisins $\mathcal{V}$ à $\{v_1\}$, et le compteur j à 2 ;
2. calculer $m = \min_{v \in \mathcal{V}} \cos \widehat{u_i v v_j}$ où $\widehat{\alpha\beta\gamma}$ désigne l'angle non orienté formé par les vecteurs $\overrightarrow{\beta\alpha}$ et $\overrightarrow{\beta\gamma}$;
3. si m est positif, ajouter v_j à $\mathcal{V}$;
4. $j \leftarrow j + 1$, revenir en 2.

Les (sous-)étapes 2 et 3 permettent de refuser les angles obtus, qui correspondent en effet à des "éclipses". Ensuite, supprimer les arêtes de "court-circuit" signifie rechercher et enlever les arêtes (relativement) trop longues, et donc reliant très probablement deux sommets qui ne devraient pas être connectés. Une heuristique est implémentée pour accomplir cette tâche, celle-ci ne pouvant pas vraiment être automatisée de façon satisfaisante.

La figure 1.7 montre un exemple de graphe au fil des trois stades (avant l'étape 1 à gauche, après cette dernière au centre, et après la seconde étape à droite), pour $k_{min} = 1$ et $k_{max} = 7$. Le graphe représente 50 points échantillonnés sur le swissroll (un rectangle enroulé dans $\mathbb{R}^3$, par analogie avec le gâteau), comme le montre la figure 1.8. Nous constatons qu'à l'étape intermédiaire deux sommets extrémaux sont reliés ; cela semble ne pas avoir de sens, mais il faut se rappeler que pour un faible échantillonnage ces points sont considérés proches sur le swissroll initial. Il est donc nécessaire d'échantillonner les données telles que les plus proches voisins soient dans le plan tangent.

Cette procédure économise significativement le nombre de voisins, malgré l'apparition de nouvelles connexions. Ces dernières paraissent suspectes, mais la représentation $2D$ fausse la visualisation en ne rendant pas compte des angles réels. Remarquons finalement que le graphe initial semble indiquer des groupes de données distincts, tandis que le résultat après l'algorithme reflète plutôt la topologie globale. C'est en effet exactement l'ambition de ce dernier.

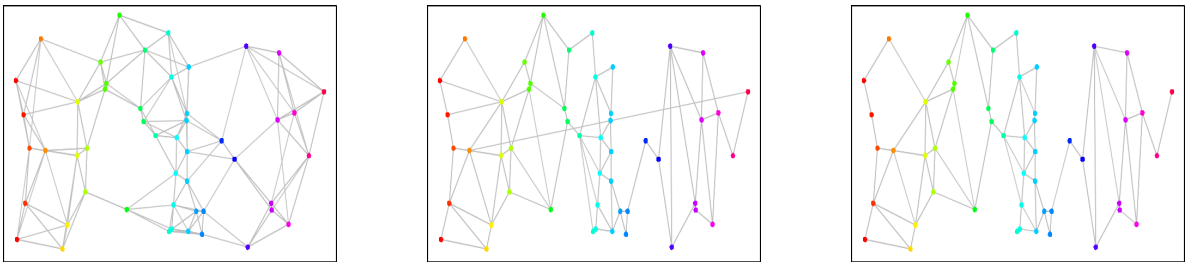

Figure 1.7: Graphe des 5 plus proches voisins à gauche, après la première étape de l'algorithme au milieu et après la seconde à droite, représenté dans l'espace des deux paramètres du swissroll (angle et hauteur).

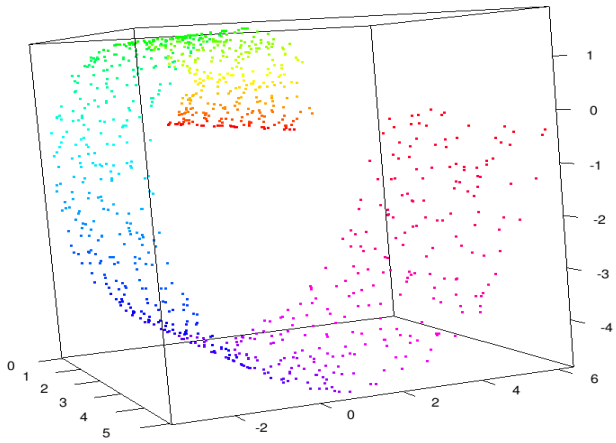

Figure 1.8: Swissroll $2D$ dans $\mathbb{R}^3$.

1.1.3.2 Voisinages adaptatifs I

Les méthodes de ce paragraphe et du suivant cherchent à évaluer la qualité des approximations linéaires locales, pour en déduire un nombre de voisins raisonnable relativement à un certain seuil d'erreur. Contrairement à la précédente, elles nécessitent de connaître (ou d'avoir estimé) la dimension d de la variété. En contrepartie, les voisinages obtenus possèdent une justification théorique (ils contiennent les points dont l'erreur d'approximation par le plan tangent est négligeable).

La première approche, par Zhan et al. [353], consiste à ajouter des points dans un voisinage jusqu'à ce que ceux-ci ne soient plus assez bien approximés relativement à un seuil $\tau \in]0, 1[$. Plus précisément, l'algorithme est le suivant.

1. Pour i allant de 1 à n :

(a) initialiser $\mathcal{V}(i)$ aux d plus proches voisins de u_i ;

(b) pour j parcourant les $k_{max} - d$ voisins restants,

i. calculer les valeurs singulières $\sigma_1 \geq \cdots \geq \sigma_m$ de la matrice formée par $\mathcal{V}(i)$ et u_j en lignes après soustraction de u_i (centrage), avec $m = |\mathcal{V}(i)| + 1$;

ii. si $\frac{\sum_{\ell=1}^{d} \sigma_\ell}{\sum_{\ell=1}^{m} \sigma_\ell} \geq \tau$ (les d premières composantes principales approximent bien les données), alors ajouter u_j à $\mathcal{V}(i)$.

2. Retourner les voisinages $\mathcal{V}(i)$.

On choisit en général $\tau = 0.95$, correspondant à 95% de variance expliquée. La figure 1.9 présente la transformation du graphe des 2 plus proches voisins − non connexe − sur le swissroll ($d = 2$, $k_{max} = 7$). Les arêtes supprimées ne sont pas les mêmes qu'à la méthode précédente, et quelques connexions "occultées" demeurent. L'allure globale du graphe résultant est cependant similaire.

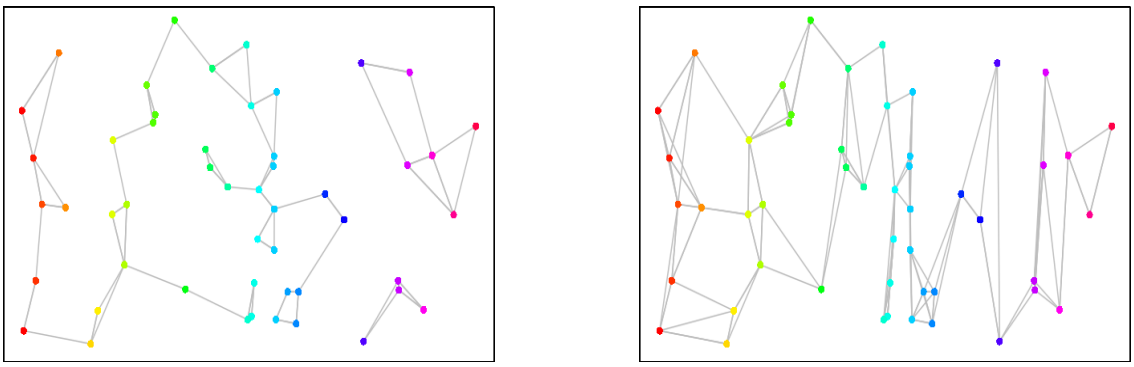

Figure 1.9: Graphe initial ($k = 2$) à gauche et le résultat après l'algorithme à droite, représenté dans l'espace des deux paramètres du swissroll (angle et hauteur).

1.1.3.3 Voisinages adaptatifs II

Nous présentons ici la méthode utilisée par Wang et al. [334] dans l'algorithme LTSA (Local Tangent Space Alignement) mentionné au paragraphe 3.2.2.1. Localement en u_i, la surface est approximée par un opérateur L, puis on réduit le nombre de points dans le voisinage $\mathcal{V}(u_i)$ jusqu'à ce que tous les éléments de ce dernier soient bien approximés par L (c'est le contraire qui est effectuée au paragraphe précédent). Ainsi, plus les données locales sont linéaires plus le nombre de voisins est élevé (supposant un échantillonnage uniforme).

Plus précisément, l'utilisateur doit à nouveau fournir deux paramètres, k_{max} et η. Le premier est une borne supérieure pour le nombre de voisins comme précédemment, tandis que le second est un seuil pour la validation d'un nombre de voisins, entre 0 et 1. Plus ce dernier est proche de 0 moins on prend de points dans un voisinage (en moyenne), et on tend à choisir k_{max} points quand il se rapproche de 1. On procède en deux phases :

1. diminution du nombre de voisins (à partir de k_{max}) jusqu'à obtenir une bonne approximation linéaire locale ;

2. expansion tant que l'approximation reste correcte.

L'étape 1 se déroule en deux temps pour un sommet u_i, k étant initialisé à k_{max} :

1. déterminer une base locale Q_0 (ACP, voir le paragraphe 3.2.1.1) à l'aide des k plus proches voisins ;
2. si la qualité de l'approximation par Q_0 est suffisamment bonne (erreur de reconstitution inférieure à $\eta\|\Theta_0\|$ où Θ_0 est la matrice des coefficients des k courbes), alors u_i a k voisins, stop ;
3. $k \leftarrow k-1$, retour en 1.

On s'arrête ainsi quand l'approximation linéaire est correcte. L'étape suivante d'expansion se base sur un théorème de majoration de l'erreur en fonction de η, et permet d'ajouter quelques points sous des conditions moins fortes. Nous ne la détaillons pas ici.

On reprend le même exemple que ci-dessus, l'évolution du graphe au fil des étapes étant présentée sur la figure 1.10, η étant choisi égal à 0.05. Nous constatons que la dernière étape ne modifie pas le graphe ; celle-ci est rendue optionnelle dans le package R final. Le résultat à l'étape intermédiaire est semblable au graphe de visibilité.

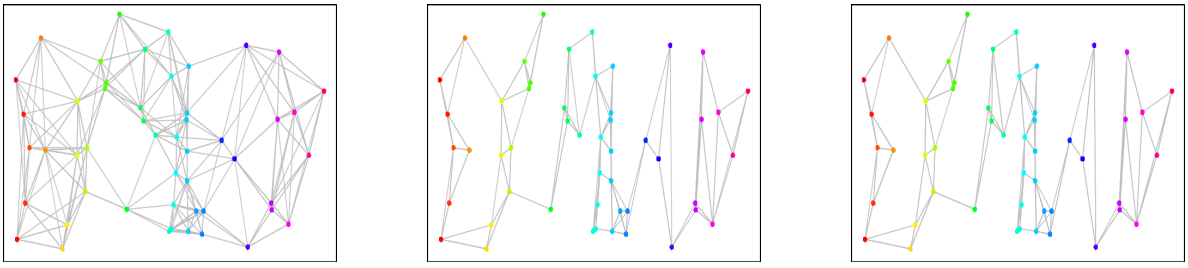

Figure 1.10: De gauche à droite : le graphe initial ($k = 7$), après la première étape, et après la seconde étape. Le tout est représenté dans l'espace des deux paramètres du swissroll (angle et hauteur).

La première approche de ce paragraphe est systématiquement utilisée par l'algorithme RML (Riemannian Manifold Learning, [213; 212]), et les deux suivantes peuvent servir à l'amélioration d'autres techniques de réduction de dimension [334]. Ces trois méthodes sont en revanche probablement inadaptées pour la phase de clustering, mais une étude plus approfondie pourrait s'avérer nécessaire pour parvenir peut-être à une méthode hybride.

Concernant ce chapitre et le suivant, nous choisissons le graphe des k plus proches voisins mutuels avec k de l'ordre de $\sqrt{n}$.

Ce dernier est bien adapté pour la classification car il comporte peu d'arêtes (moins de k à chaque sommet), donc il y a assez peu de risques de relier deux sommets séparant deux groupes différents. En effet, la distance définie ci-dessous puis les similarités ne seront utiles qu'au chapitre de classification non supervisée.

1.2 Distance dans le graphe

L'objectif de cette section est la définition d'une distance sur le graphe, utilisée comme substitut à la distance euclidienne pour faciliter la classification.

Une fois le graphe de voisinage construit, on pourrait se contenter d'utiliser les distances euclidiennes locales puis d'appliquer l'algorithme de clustering hiérarchique par exemple (voir le paragraphe 2.4.3). L'intérêt d'utiliser un graphe serait cependant limité. On choisit donc plutôt de définir une nouvelle distance en adéquation avec la problématique de clustering. Celle-ci nécessite des similarités entre éléments d'un même voisinage, qu'il va falloir évaluer (ce sera l'objet de la dernière section).

Définition 1.2.1 *Une fonction de* similarité s *sur un ensemble* E *est une application continue à valeurs dans* $[0, M]$, $M > 0$, *vérifiant*

$$s(u, v) = M \Leftrightarrow u = v$$

On définit aussi la grandeur duale : une fonction $\delta : E \to [0, +\infty[$ vérifiant $\delta(u, v) = 0 \Leftrightarrow u = v$ constitue une *dissimilarité*. Ces deux définitions sont très générales, car les résultats obtenus via ces quantités s'évaluent souvent de manière subjective. Ainsi il n'y a pas de raison d'imposer plus de contraintes à ce stade. On fait généralement en sorte que la (dis)similarité augmente (resp. diminue) lorsque la distance euclidienne diminue, mais ce n'est pas obligatoire. On peut par exemple comparer les courbes selon leurs formes [162].

Dans la suite les similarités entre les sommets i et j dans le graphe sont notées s_{ij}. On se place dans une composante connexe du graphe, les distances inter-composantes étant supposées infinies dans un premier temps (les classes de connexité peuvent être fusionnées à l'aide du clustering hiérarchique par exemple). Nous travaillons ainsi sur un cas particulier de graphe valué (positivement), orienté mais dont la forte connexité équivaut à la connexité faible. La matrice S des poids s_{ij} n'est pas symétrique en général. Ce n'est pas grave pour la définition de la distance qui va suivre, les problèmes posés étant de nature plutôt algorithmique.

Remarque : en réduisant les courbes à leurs inter-similarités via la matrice S, on passe en quelque sorte de la dimension infinie à la dimension n ; cela a une grande importance théorique (nette perte d'information), même si en pratique n est de l'ordre de quelques centaines, donc encore une haute dimension. Les tests montrent que l'information emmagasinée dans le graphe est suffisante pour bien classer les données dans nos applications.

Une marche aléatoire est à présent définie sur le (sous-)graphe, celle-ci permettant l'introduction d'une distance entre les sommets du graphe (distance euclidienne "commute-time"). Finalement quelques méthodes de calcul de cette distance sont présentées, avant de s'attacher à la détermination des poids des arêtes.

1.2.1 Distance euclidienne "Commute-Time"

L'objet de ce paragraphe est la définition d'une distance sur le graphe, mettant bien en évidence les groupes d'éléments distincts. Nous nous basons pour cela sur un parcours aléatoire

du graphe – symbolisé par $\mathcal{M}$ – guidé par les similarités locales. Avant toutes choses, rappelons que le calcul des distances repose sur le graphe de voisinage, dont la construction est loin d'être triviale. La recherche du graphe de voisinage "optimal" est en fait un problème non résolu à ce jour, aussi nous nous contenterons d'heuristiques (indiquées aux paragraphes précédents).

Une marche aléatoire désigne un parcours (potentiellement infini) sur les sommets du graphe, chaque transition d'un sommet à un autre ne dépendant que de l'état courant. Il s'agit d'un cas particulier de chaîne de Markov du premier ordre ; voir par exemple les livres de Kemeny et Snell [181] et Norris [245] à ce sujet. Notons π_{ij} la probabilité de passer dans l'état (sommet) j en partant de i pour $i, j \in \{1, \ldots, n\}$. Les π_{ij} sont assez naturellement déterminés comme suit.

$$\forall i, j \in \{1, \ldots, n\}\, \pi_{ij} = \frac{s_{ij}}{\sum_{j=1,\ldots,n} s_{ij}}\,,$$

Afin de comparer les données u_i, il est alors intéressant de regarder le temps moyen nécessaire à la marche aléatoire $\mathcal{M}$ définie par les transitions π_{ij} pour effectuer un aller-retour entre deux sommets. En effet, plus ces sommets sont similaires plus il y a de chances qu'ils soient reliés par plusieurs courts chemins $i = i_1 \to \cdots \to i_m = j$ (resp. $i_1 = j, i_m = i$) avec $\pi_{i_\ell i_{\ell+1}}$ supérieur à un certain seuil. Plus il y a de tels courts chemins (empruntés avec des probabilités non négligeables), plus le temps nécessaire pour relier les deux sommets se réduit. À ce stade on peut émettre quelques objections :

- les probabilités de transition ne tiennent pas compte de la densité des régions traversées (seulement des distances relatives locales) ;
- ce temps moyen d'aller-retour vérifie-t-il l'inégalité triangulaire ? (seule propriété non évidente pour en faire une distance).

Le premier point soulevé ci-dessus est résumé sur la figure 1.11 où les probabilités de transition sont les mêmes aux voisinages de A et B. Cependant, s'il est facile de passer d'une région peu dense à une région dense, il est difficile de sortir de cette dernière : en effet les probabilités de transitions sont nettement plus fortes si l'on reste dans cette zone. Ainsi les temps d'aller-retours mesurés entre une région dense et peu dense seront bien plus élevés que ceux observés au sein d'un même cluster. Par un raisonnement similaire on voit que le temps moyen d'aller retour entre deux clusters bien séparés sera encore plus grand. Quant à l'inégalité triangulaire, elle est prouvée par exemple dans l'article de Göbel et Jagers [144] ; en fait les temps d'atteinte eux-mêmes vérifient l'inégalité triangulaire malgré leur dissymétrie. Ainsi le temps moyen d'aller-retour (commute-time) est une distance sur les sommets du graphe. Comme nous allons le voir, il s'agit du carré d'une distance euclidienne dans un espace de dimension n.

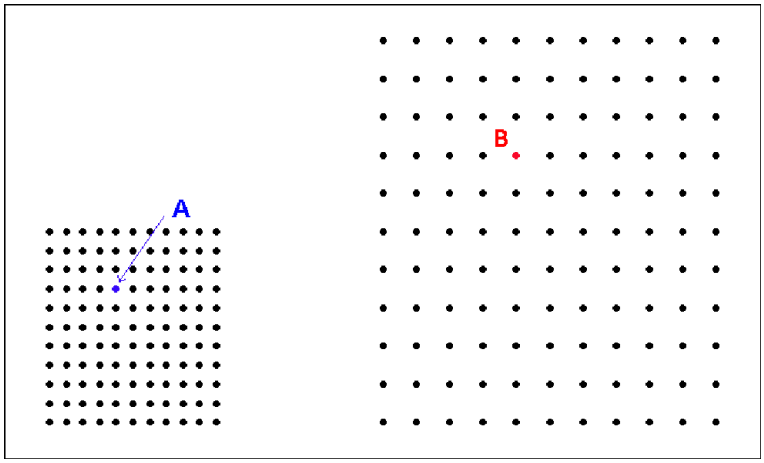

Figure 1.11: Probabilités de transition indépendantes de la densité.

1.2.2 Calcul du temps moyen d'aller-retour

On peut calculer le temps d'aller-retour de deux manières : en utilisant des relations de récurrences provenant de la théorie des marches aléatoires sur les chaînes de Markov, ou via le laplacien du graphe défini ci-après.

1.2.2.1 Calcul par récurrence

Notons $ct(i,j)$ le temps d'aller-retour entre i et j, et décomposons le en $ct(i,j) = a(j|i) + a(i|j)$ où $a(j|i)$ désigne le nombre moyen d'étapes nécessaires pour atteindre j depuis i. Pour aller de i à j, il faut d'abord faire un pas vers un voisin v de i, choisi avec la probabilité π_{iv}. Ce pas réalisé coûte un temps, auquel s'ajoute le temps nécessaire pour aller de v à j. La formule suivante s'obtient naturellement :

$$\begin{cases} a(i|i) = & 0, \\ a(j|i) = & 1 + \sum_{v=1}^{n} \pi_{iv} a(j|v). \end{cases}$$

Pour une démonstration formelle voir Fouss et al. [126], annexe A. Cette récurrence multiple peut être rapidement résolue de manière approchée ; c'est intéressant pour effectuer les calculs quand n est trop grand pour opérer directement sur une matrice de taille n^2.

Sarkar et Moore [280] approximent les temps d'atteinte $a(j|i)$ en se restreignant à des chemins de longueur au plus T. Cela a le double avantage d'éviter d'obtenir une trop petite valeur lorsqu'un sommet a beaucoup de voisins [43] et d'alléger les calculs. La valeur exacte de la colonne constituée des $a(j|i)^T$ à j fixé peut ainsi être calculée en temps $O(kT)$ par programmation dynamique (Bellman [23] ; Kaufmann et Cruon [178], chapitre 15). Si T est de l'ordre de $\sqrt{n}$ le calcul total s'effectue alors en temps $O(n^2)$, gagnant un facteur n par rapport à la complexité de la méthode exacte ci-après. Étant donné que les degrés du graphe (des voisins mutuels) sont majorés par k dans notre application et que n ne dépasse pas 1000, on préfère utiliser les méthodes introduites aux paragraphes suivants.

1.2.2.2 Calcul via le temps d'atteinte

Lorsque le graphe est orienté, on peut déterminer le temps moyen d'aller-retour via une expression algébrique du temps d'un aller. Dans un graphe apériodique la quantité suivante

peut toujours être définie, pour $i, j \in \{1, \ldots, n\}$:

$$Z_{ij} = \sum_{t=0}^{\infty} (p_{ij}^t - \pi_j) \,,$$

où p_{ij}^t désigne la probabilité d'atteindre j en partant de i en exactement t étapes. Si le graphe est périodique une possibilité pour évaluer tout de même Z_{ij} est de recourir à une moyenne de Cesaro. La matrice Z de taille $n \times n$ dont le coefficient (i, j) est égal à Z_{ij} est appelée *matrice fondamentale* associée au graphe G (voir Aldous et Fill [7], chapitre 2). Celle-ci peut s'exprimer de manière plus compacte par

$$Z = (I_n - P + \mathbf{1}^t\pi) \,,$$

où π est l'unique vecteur de probabilités stationnaires pour la marche aléatoire définie par les π_{ij} ($^t\pi P = P$), P étant la matrice des probabilités de transition.

Le lemme 12 du chapitre précité indique que l'on peut calculer $a(j|i)$ (avec les notations précédentes) comme suit.

$$a(j|i) = \frac{Z_{ij} - Z_{jj}}{\pi_j} \,,$$

avec $\pi = (\pi_1, \ldots, \pi_n)$. Ainsi, au coût de la recherche de l'inverse d'une matrice on peut déterminer analytiquement les temps d'atteinte moyens, puis les temps moyens d'aller-retour. À ce sujet, voir aussi l'article de Chen et Zhang [58]. Chen et al. [59] implémentent un algorithme du type k-means utilisant les temps d'atteinte, avec pour argument que ces derniers sont mieux adaptés que les temps d'aller-retour malgré leur dissymétrie. En effet certaines applications peuvent être naturellement dissymétriques.

Liben-Nowell et Kleinberg [211] remarquent que $a(j|i)$ est souvent très petit quand j a une probabilité stationnaire élevée. C'est pourquoi ils proposent de pondérer les dissimilarités de la façon suivante : $a(j|i) \leftarrow a(j|i)\pi_j$. On peut aussi appliquer cette modification pour le calcul du temps moyen d'aller retour :

$$ct(i, j) = a(j|i)\pi_j + a(i|j)\pi_i = 2Z_{ij} - Z_{ii} - Z_{jj} \,.$$

1.2.2.3 Calcul avec le laplacien

La formule concluant ce paragraphe n'est valable que si S est symétrique, et donc pour un graphe non orienté (par exemple si $\sigma_i = \sigma =$ constante avec une similarité gaussienne). Le laplacien du graphe G est la matrice de taille $n \times n$ définie à partir de la matrice S des similarités inter-sommets.

$$L = D - S \,,$$

avec D matrice diagonale des degrés, $d_{ii} = \sum_{j=1}^{n} s_{ij}$. Le laplacien a une interprétation en terme d'approximation discrète de l'opérateur (continu) de Laplace-Beltrami, agissant sur une variété [22]. On remarque que pour un graphe connexe L est de rang $n - 1$, le vecteur propre $\mathbf{1}$ vérifiant $L\mathbf{1} = 0$ et $^t\mathbf{1}L = 0$ (pour une introduction plus complète et d'autres propriétés voir par exemple le livre de Chung [71], chapitre 1).

Notons L^+ le pseudo inverse [6] du laplacien défini comme suit.

$$L^+ = V\Lambda^{-1}{}^tU\,,$$

avec la convention $\frac{1}{0} = 0$, $U\Lambda{}^tV$ étant la décomposition en valeurs singulières de L. Ce pseudo-inverse est comme son nom l'indique "presque" l'inverse de L ; remarquons cependant qu'en général $LL^+ \neq L^+L$. Il existe un lien entre les éléments du pseudo-inverse et les temps moyen d'atteinte définis précédemment. On peut deviner cette relation en observant que l'on a "presque" (valable partout sauf sur la diagonale qui est nulle) :

$$A = I + PA\,,$$

avec $A = (A_1, \ldots, A_n)$ où A_i est le vecteur colonne des $a(j|i)$, $j = 1, \ldots, n$, P étant la matrice des probabilités de transitions (éventuellement nulles). Comme $DP = S$ on obtient $A = (D - S)^+D$, faisant apparaître le pseudo-inverse du laplacien. La formule exacte est la suivante

$$a(j|i) = \sum_{v=1}^{n}(l^+_{iv} - l^+_{ij} - l^+_{jv} + l^+_{jj})d_{vv}\,,$$

avec l^+_{uv} les élements de la matrice L^+. On en déduit l'expression suivante du temps moyen d'aller-retour :

$$ct(i,j) = Vol_G(l^+_{ii} + l^+_{jj} - 2l^+_{ij})\,,$$

avec le volume du graphe $Vol_G = \sum_{i=1}^{n} d_{ii}$. Voir l'annexe E de Fouss et al. [126] pour une démonstration correcte.

Nous retenons ce temps d'aller-retour pour notre application, d'autres mesures de proximité étant présentées dans l'article de Chebotarev et Shamis [56] par exemple. On l'abrège désormais en ECTD pour "Euclidian Commute-Time Distance", par légère anticipation.

Remarque : suivant les mêmes motivations que Liben-Nowell et Kleinberg [211], Brand [43] préfère utiliser la similarité cosinus définie par $s_{cos}(i,j) = \frac{l^+_{ij}}{\sqrt{l^+_{ii}l^+_{jj}}}$. Nous n'avons pas de problèmes de sommets à degrés élevés, donc ne nous orientons pas dans cette direction.

Dans un graphe non orienté la distance ECT entre i et j s'écrit

$$d_{CT}(i,j) = \sqrt{Vol_G(l^+_{ii} + l^+_{jj} - 2l^+_{ij})}\,.$$

Si le graphe est orienté, alors le calcul peut s'effectuer comme suit

$$d_{CT}(i,j) = \frac{Z_{ij} - Z_{ii}}{\pi_i} + \frac{Z_{ij} - Z_{jj}}{\pi_j}\,.$$

Nous utilisons l'une ou l'autre de ces expressions dans le programme R, suivant la symétrie de S. La complexité globale de tout le processus de clustering est ainsi déterminée par un calcul d'inversion de matrice, en $O(n^3)$ opérations (des optimisations étant possibles, voir notamment Jessup et Sorensen [171] et Gu et Eisenstat [148]).

1.3 Détermination de l'échelle locale

En ce qui concerne la marche aléatoire, seule l'échelle relative compte lorsqu'on calcule les (dis)similarités inter-sommets. Ainsi, la formule donnant les similarités locales peut varier d'un sommet à un autre. Nous choisissons alors de fixer les similarités à $s_{ij} = e^{\frac{-\|u_i - u_j\|^2}{\sigma_i^2}}$ (gaussienne), alternativement notées $s(i,j)$, σ_i étant défini ci-dessous. En effet cette forme reste assez générale pour la plupart des applications.

Localement en u_i, si σ_i est trop petit toutes les similarités seront très proches de 0. À l'opposé, si ce dernier est trop grand, les similarités calculées seront très proches de 1. Dans les deux cas ces grandeurs ne seraient pas significatives. Une idée consiste alors à maximiser l'écart entre les deux similarités $s(i,j_1)$ et $s(i,j_k)$ où j_1 (resp. j_k) est le voisin le plus proche (resp. le plus éloigné) de u_i. Au bout de quelques lignes de calcul, on voit que cela revient à choisir

$$\sigma_i^2 = \frac{\|u_i - u_{j_k}\|^2 - \|u_i - u_{j_1}\|^2}{\log \|u_i - u_{j_k}\|^2 - \log \|u_i - u_{j_1}\|^2}$$

C'est cette valeur que nous utilisons dans tous les algorithmes. Dans le même ordre d'idée, van der Maaten et Hinton [322] suggèrent de choisir σ_i tel que l'entropie de la distribution locale des probabilités de transition soit égale à une valeur fixée. Bien que la recherche de σ_i s'effectue rapidement par dichotomie, la valeur de l'entropie est assez arbitraire. Nous passons maintenant en revue quelques autres similarités utilisées dans la littérature, avant de les comparer à celle qui vient d'être présentée.

1.3.1 Bibliographie

Liben-Nowell et Kleinberg [211] présentent et comparent plusieurs types de similarités, dont le nombre de voisins communs, noté s_{com}, et la similarité introduite par Katz [177] définie comme suit.

$$s_{chem}(i,j) = \sum_{t=1}^{\infty} \beta^t L^t(i,j)\,,$$

où $L^t(i,j)$ désigne le nombre de chemins de longueur t reliant i à j, β étant un réel positif d'autant plus petit qu'on privilégie l'influence des courts chemins. Cette dernière est avec l'heuristique des voisins communs une des méthodes donnant les meilleurs résultats dans l'article de Liben-Nowell et Kleinberg [211] (sur un test consistant à prédir l'existence de futurs liens dans un réseau social). Elle possède de plus une expression analytique simple : $s_{chem} = (I - \beta A)^{-1} - I$ où A est la matrice d'adjacence du graphe. Attention tout de même au rayon de convergence de la série, qui décroît rapidement avec n dans le pire cas ; en effet les coefficients de A^m sont de l'ordre de $(n-1)^m$ dans le cas d'un graphe complet, ce qui impose de choisir $\beta \simeq \frac{1}{n-1}$ (ou inférieur). Notons que s_{com} et s_{chem} ne réutilisent pas les distances euclidiennes (celles-ci ont servi à la construction du graphe).

Fischer et Buhmann [122] proposent d'estimer directement la distance entre u_i et u_j par la longueur minimale d'un segment maximal sur un chemin allant de i à j dans le graphe :

$$d_{path} = \min_{L:i\to j} \left(\max_{(t,t+1)\in L} \|u_{L[t+1]} - u_{L[t]}\| \right)\,,$$

avec L une suite de sommets formant un chemin d'extrémités i, j. Cela n'est pas sans rappeler l'heuristique single-linkage du clustering hiérarchique, connue pour être peu robuste au bruit. De plus la recherche d'une telle distance est un problème difficile, non résoluble en temps polynomial. Pour pallier le premier défaut, Chapelle et Zien [55] utilisent une version "adoucie" du maximum sur un chemin, dans le cadre de la définition d'un noyau servant à la classification semi-supervisée des données. La valeur obtenue est alors à mi-chemin entre la longueur maximale d'un segment et la somme des longueurs. La difficulté du réglage du paramètre (global) effectuant le compromis nous dissuade d'utiliser cette dernière méthode.

Zelnik-Manor et Perona [351] posent $s_{ij} = e^{\frac{-\|u_i-u_j\|^2}{\delta_i\delta_j}}$ où δ_ℓ désigne la distance de u_ℓ à son k^{eme} plus proche voisin. Une variante est implémentée dans le package R *kernlab* [175], en prenant pour δ_ℓ la valeur médiane des distances aux 5 plus proches voisins de u_ℓ. Ces deux similarités ont l'avantage d'être symétriques, mais σ n'est pas constant localement : $\sigma_{ij} \neq \sigma_{ij'}$ en général pour $j \neq j'$, ce qui peut poser des problèmes. De plus le produit $\delta_i\delta_j$ peut ne pas être adapté dans le cas où δ_i et δ_j ne sont pas du même ordre de grandeur, entraînant une assez grande sensibilité aux variations de densité. Enfin, Zhang et al. [354] utilisent un paramètre σ global, qu'ils pondèrent par le nombre de voisins communs dans l'intersection des boules de rayon ϵ centrées en u_i et u_j :

$$s_\epsilon(i,j) = e^{\frac{-\|u_i-u_j\|^2}{\sigma^2(1+CNN_{i,j}(\epsilon))}},$$

$CNN_{i,j}(\epsilon)$ désignant le nombre de points dans $B(u_i,\epsilon)\cap B(j,\epsilon)$. Le problème de cet approche réside dans le calibrage des paramètres σ et ϵ, pouvant se révéler inadaptés quand l'échelle varie.

1.3.2 Comparaisons

Compte-tenu de la difficulté à optimiser les similarités basées sur les chemins, nous comparons uniquement les quantités s_{com}, s_{chem} (avec $\beta = \frac{1}{n-1}$) et s_ϵ (avec ϵ et σ respectivement fixés à la distance moyenne aux 5 plus proches voisins, et à la distance moyenne entre deux points). La similarité présentée en introduction est notée s_{loc}.

Conformément à notre problématique de clustering, et contrairement à Liben-Nowell et Kleinberg [211], nous choisissons d'évaluer une similarité s comme suit pour un partitionnement $C = \bigsqcup_{i=1}^K C_i$.

$$V(s) = \frac{\sum_{i=1}^K \sum_{j,\ell\in C_i} s'_{j\ell}}{\sum_{j,\ell=1,\dots,n} s'_{j\ell}},$$

c'est-à-dire le ratio du volume des similarités intra-clusters par la somme de toutes les similarités, toutes celles-ci étant normalisées : $s'_{j\ell} = \frac{s_{j\ell}}{\sum_{q=1}^n s_{jq}}$. Plus cette dernière quantité est élevée plus les points d'un même cluster se voient attribuer des liens forts, au détriment des liaisons inter-clusters. Afin d'établir une comparaison honnête nous utilisons systématiquement le graphe des k plus proches voisins mutuels avec $k = \lceil\sqrt{n}\rceil$ pour définir les voisinages.

1.3.2.1 Données artificielles

La figure 1.12 présente le premier jeu de données, consistant en deux demi-cercles imbriqués. La transformation des coordonnées $(x_{i,1}, x_{i,2})$ en la fonction $y_i(t) = x_{i,1} \cos t + x_{i,2} \arctan t$ définie sur $[0, 2\pi]$ permet de calculer les similarités sur des données fonctionnelles. Nous ajoutons un bruit directement sur les courbes, car dans notre application les paramètres d'entrée sont fixés (par un plan d'expérience), tandis que les sorties peuvent comporter des erreurs d'approximation liées à la résolution d'équations différentielles non linéaires.

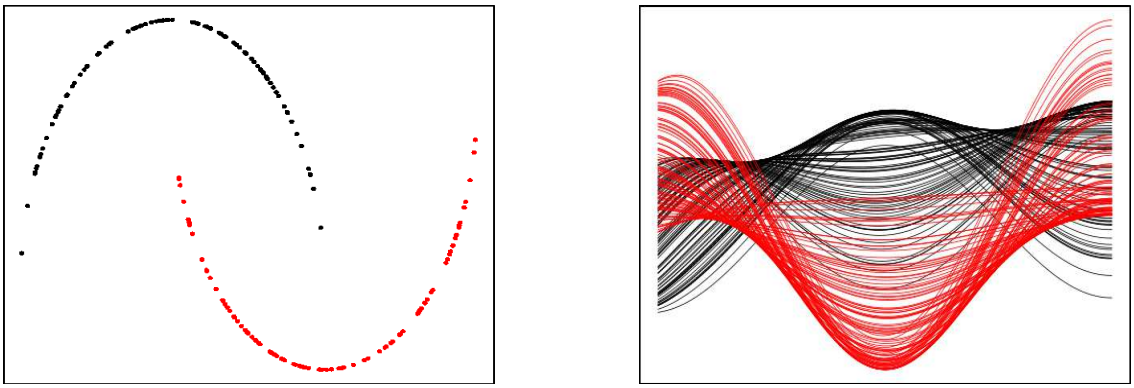

Figure 1.12: Jeu de données "two moons" non bruité à gauche, les fonctions correspondantes à droite.

Bruit	s_{com}	s_{chem}	s_{ϵ}	s_{loc}
1	0.96	0.97	0.97	0.97
2	0.86	0.89	0.86	0.89
3	0.78	0.83	0.74	0.82
4	0.66	0.74	0.61	0.72
5	0.61	0.70	0.58	0.69
6	0.57	0.65	0.54	0.63
7	0.55	0.62	0.53	0.61
8	0.53	0.59	0.52	0.58
9	0.52	0.57	0.51	0.56
10	0.51	0.56	0.51	0.55

Table 1.1: Qualité des similarités en fonction du bruit

Le tableau 1.1 présente les résultats obtenus, moyennés sur 10 exécutions. On constate la bonne robustesse au bruit (gaussien indépendant d'écart-type variant de 1 à 10) de la similarité de Katz [177], suivie de très près par la méthode introduite en début de paragraphe. Lorsque le bruit est faible et que les points se séparent bien visuellement, assez logiquement toutes les méthodes semblent se valoir. C'est pourquoi nous n'affichons pas les résultats en présence d'un bruit d'écart-type inférieur à 1.

Le second exemple est composé de trois gaussiennes $2D$ (de centres respectifs $(0, 0)$, $(1, 1)$ et $(2, 0)$), transformées en courbes (figure 1.13) comme indiqué ci-dessus. Celles-ci présentent la particularité d'avoir des écarts-types différents en abscisse et en ordonnée.

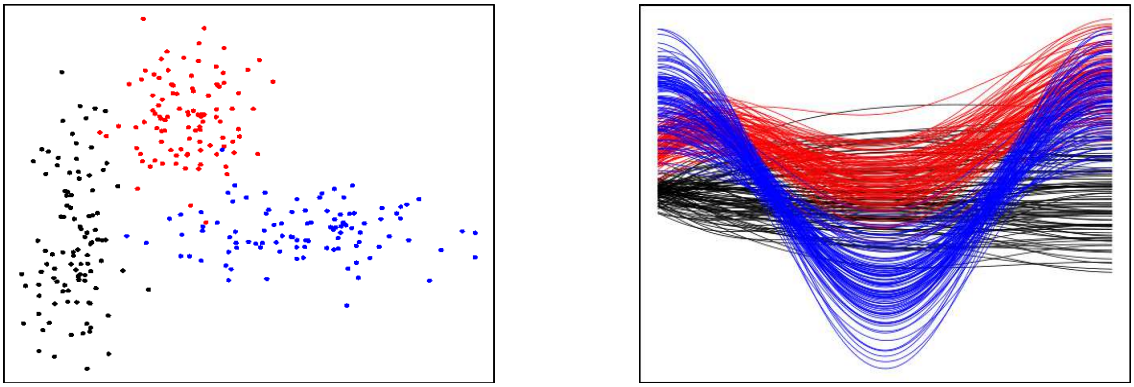

Figure 1.13: Trois gaussiennes non isotropiques à gauche, les fonctions correspondantes à droite.

Bruit	s_{com}	s_{chem}	s_{ϵ}	s_{loc}
1	0.91	0.93	0.92	0.93
2	0.79	0.85	0.77	0.85
3	0.63	0.73	0.56	0.71
4	0.54	0.65	0.46	0.62
5	0.45	0.56	0.41	0.54
6	0.40	0.50	0.37	0.49
7	0.38	0.46	0.35	0.44
8	0.36	0.43	0.35	0.42
9	0.35	0.41	0.34	0.40
10	0.34	0.39	0.34	0.38

Table 1.2: Qualité des similarités en fonction du bruit

Le tableau 1.2 présente les résultats obtenus, toujours moyennés sur 10 répétitions (protocole suivi également pour les tests suivants). Le bilan est similaire au précédent : s_{chem} donne les meilleurs résultats, suivie de près par s_{loc}, toutes deux assez nettement meilleures que les deux autres similarités.

L'exemple suivant est construit en faisant varier linéairement les paramètres de la fonction suivante définie sur $[1, 5]$:

$$f_{\alpha,\beta,\gamma}(t) = \left(\frac{\sin \alpha x}{x} + e^{-\beta x}\right) \cos \gamma t \,,$$

Le paramètre β varie uniformément dans trois plages différentes : $[1, 2]$, $[0, 1]$ et $[0.4, 1]$. On le note respectivement β_1, β_2 et β_3 suivant le groupe auquel il appartient. On pose alors $\alpha_{1,2,3} = 3\beta_{1,2,3}$, et $\gamma_1 = \sqrt{4 - \beta_1^2}$, $\gamma_2 = \sqrt{1 - \beta_2^2}$, $\gamma_3 = 3\sqrt{1 - \beta_3^2} + 3$, obtenant ainsi trois clusters. La figure 1.14 montre l'allure des courbes dans chaque cluster (non bruitées, régulièrement échantillonnées).

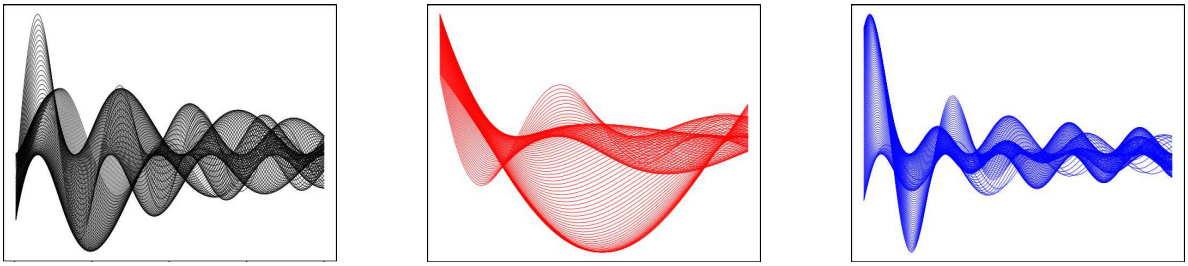

Figure 1.14: Clusters fonctionnels 1, 2 et 3 (non bruités, régulièrement échantillonnés) de gauche à droite.

Bruit	s_{com}	s_{chem}	s_{ϵ}	s_{loc}
0.25	0.94	0.97	0.97	0.96
0.50	0.68	0.78	0.62	0.76
0.75	0.52	0.61	0.44	0.59
1.00	0.45	0.53	0.39	0.50
1.25	0.41	0.48	0.36	0.45
1.50	0.37	0.43	0.34	0.41
1.75	0.35	0.40	0.34	0.39
2.00	0.35	0.39	0.34	0.38
2.25	0.34	0.37	0.33	0.36
2.50	0.34	0.36	0.33	0.36

Table 1.3: Qualité des similarités en fonction du bruit

Les similarités moyennes obtenus sont indiquées dans le tableau 1.3. Le même classement s'observe à nouveau, s_{chem} étant clairement meilleur que s_{loc} pour un bruit intermédiaire variant de 0.75 à 1.5 ; l'écart reste cependant raisonnable.

1.3.2.2 Données réelles

Les données de ce paragraphe sont téléchargeables sur le site UCI Machine Learning Repository.

Le premier jeu de données [100] consiste en 15 groupes de 24 paires de courbes, chacune comportant l'évolution de deux angles codant des gestuelles du language des signes. La figure 1.15 montre les deux courbes échantillonnées pour deux clusters différents, tandis que la figure 1.16 présente le jeu de données complet.

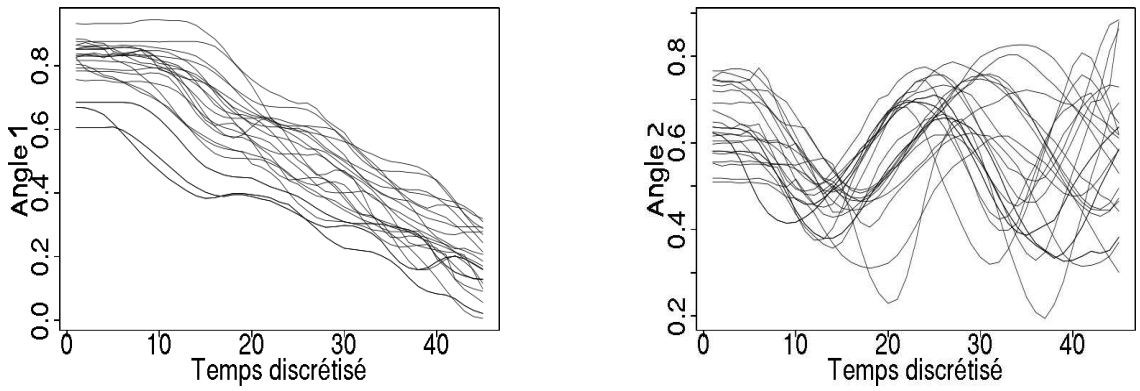

Figure 1.15: Groupe 12 du jeu de données LIBRAS - composante 1 à gauche, 2 à droite.

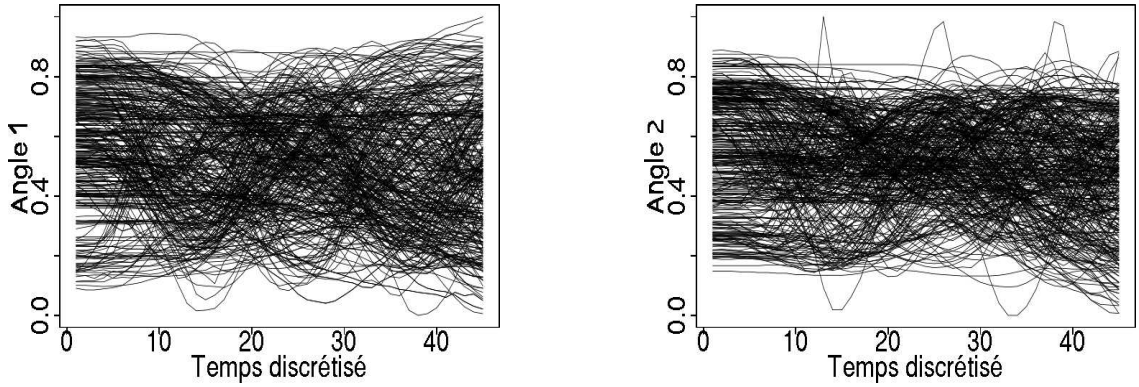

Figure 1.16: Jeu de données LIBRAS - composante 1 à gauche, 2 à droite.

	s_{com}	s_{chem}	s_ϵ	s_{loc}
Courbes 1	0.33	0.39	0.41	0.43
Courbes 2	0.45	0.49	0.42	0.52
Courbes 1 et 2	0.45	0.54	0.70	0.63

Table 1.4: Qualité des similarités en fonction des courbes sélectionnées

Comme le tableau 1.4 l'indique, s_{com} est clairement le moins bon choix sur cet exemple, et s_{chem} le second plus mauvais. Le classement des deux autres n'est pas très clair : si s_{loc} surclasse nettement s_ϵ lorsqu'une seule composante fonctionnelle est utilisée, cette dernière se révèle clairement plus robuste lorsque la sortie fonctionnelle est pleinement exploitée.

Le second jeu de données testé [146] contient des mesures relevées lors de la traversée d'un signal (sonar) dans un cylindre métallique ou rocheux. Celles-ci sont effectuées régulièrement autour de ces objets, et sont assimilables à des courbes continues. La figure 1.17 présente les deux clusters, le second (roche, à droite) étant légèrement décalé sur la gauche par rapport au premier.

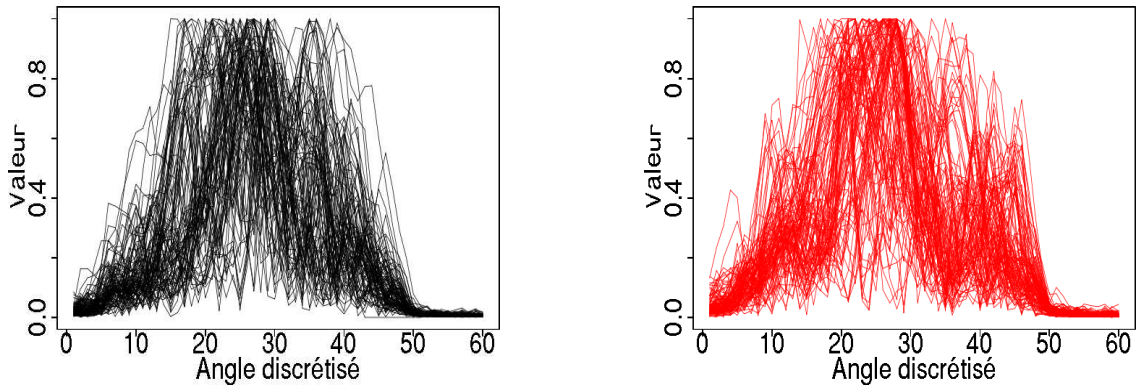

Figure 1.17: Les deux clusters du jeu de données SONAR.

s_{com}	s_{chem}	s_ϵ	s_{loc}
0.65	0.68	0.80	0.74

Table 1.5: Qualité des similarités

Les résultats sont présentés dans le tableau 1.5. Encore une fois on observe que s_ϵ est la plus performante, suivie par s_{loc}. Compte-tenu des résultats, la similarité s_{loc} constitue un bon compromis entre cas artificiels et réels. Ainsi, cette dernière est utilisée pour les tests au cours des chapitres suivants.

1.4 Conclusion

Le graphe construit dans cette partie va permettre d'élaborer des algorithmes assez sophistiqués pour classer les données et rechercher leur topologie au cours des chapitres suivants, dans le but d'élaborer un métamodèle s'adaptant efficacement à divers jeux de données. L'originalité du graphe choisi réside dans la façon de déterminer l'échelle locale σ_i, indiquée au paragraphe 1.3.

1.5 Perspectives

La construction du graphe de visibilité au paragraphe 1.1.3.1 est guidée par des considérations géométriques, mais a priori sans garanties topologiques. Ce dernier graphe permet toutefois à l'algorithme RML d'apprendre la topologie des données de façon satisfaisante (voir le paragraphe 3.2.3.2). Afin d'assurer la convergence − dans un sens à préciser − de ce graphe vers la vraie géométrie des y_i il peut être intéressant de le combiner avec les approches par ACP locales.

Les (dis)similarités dans le graphe sont mesurées à l'aide d'un unique réel. Cela peut sembler être un résumé assez pauvre en information (même si les résultats sont corrects concernant notre application). Il est en effet envisageable de définir des (dis)similarités vectorielles, puis d'établir une relation d'ordre floue [37; 349]. En effet une relation d'ordre classique comme celle du dictionnaire reviendrait à n'utiliser qu'un seul réel, les cas d'égalité permettant de tester les autres composantes étant de probabilité nulle.

Chapitre 2

Clustering des entrées-sorties du code

L'objectif de ce chapitre est de proposer un algorithme robuste et efficace de classification non supervisée pour des données mixtes vectorielles et fonctionnelles. On souhaite déterminer des groupes d'entrées-sorties $(x_i, y_i), i = 1, \ldots, n$ aux caractéristiques similaires, les plus différentes possibles des autres groupes. "Non supervisée" signifie qu'on ne connaît ni le nombre de groupes à trouver, ni leurs compositions comme c'est le cas dans l'application visée. Les termes groupes, clusters ou classes sont employés indifféremment, et la classification non supervisée est souvent désignée par le terme anglais "clustering".

Cette recherche de classes d'entrées-sorties est motivée par le fait qu'un code peut comporter plusieurs types de réponses différentes, comme schématisé sur la figure 2.1. On comprend alors pourquoi ϕ a été supposée continue par morceaux seulement. Le but de l'étape de classification consiste à retrouver ces "morceaux", pour ensuite effectuer la régression plus facilement sur chaque domaine de continuité.

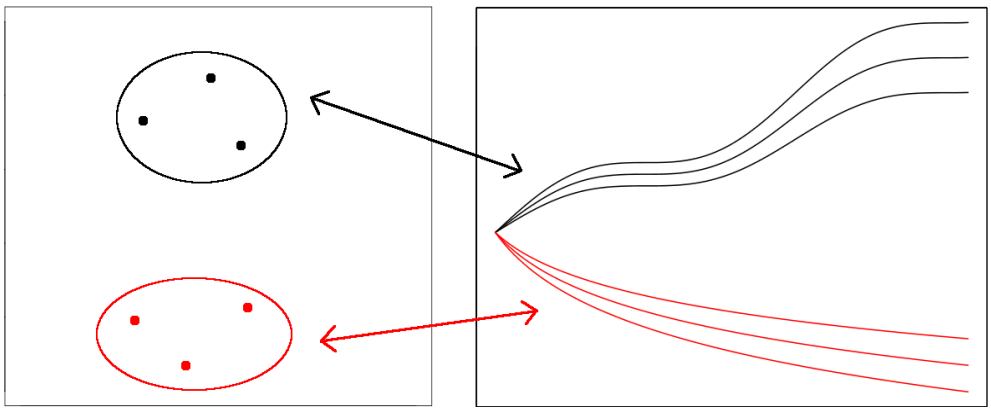

Figure 2.1: Plusieurs types de sorties ⇒ plusieurs clusters d'entrées-sorties.

Les données à disposition comportant des entrées vectorielles et sorties fonctionnelles, il peut sembler intéressant de classer en même temps entrées et sorties. Nous commençons par discuter cette possibilité. Une technique de regroupement classique est présentée ensuite, qui sera utilisée par plusieurs méthodes dans un bref état de l'art du clustering fonctionnel, ainsi que dans des approches plus générales. Finalement, la détermination du nombre de groupes est abordée, aboutissant à la présentation de l'algorithme utilisé.

Remarquons qu'il n'y a pas de définition officielle de ce qu'est un cluster ; toute cette étude peut alors sembler assez subjective. Nous proposons toutefois une approche assez objective vis-à-vis de l'application finale au paragraphe 2.5.3. Le nombre $K \in \mathbb{N}^*$ de clusters à découvrir est toujours supposé connu − sauf dans la section discutant son estimation.

Les sections 2.3, 2.4 et 2.5 sont en grande partie bibliographiques, afin d'effectuer une présentation relativement complète du sujet. En dehors des paragraphes 2.3.1, 2.3.4, 2.4.2, 2.4.3 et 2.5.3, leur lecture n'est pas essentielle pour la compréhension de la méthode mise en œuvre.

Plan du chapitre :

1. **Détermination de la mesure de comparaison inter-courbes.**
2. **Réflexion sur la classification simultanée des entrées et sorties.**
3. **État de l'art du clustering fonctionnel.**
4. **État de l'art du clustering dans un graphe.**
5. **Détermination du nombre de classes.**

2.1 Comparaison des courbes

Dans l'espace euclidien $\mathbb{R}^q$ les vecteurs sont en général comparés par la norme un, deux ou infinie. Celles-ci étant équivalentes (i.e. vérifiant des inégalités de proportionnalité) il n'y a pas de problème théorique majeur. Dans l'espace des fonctions continues $\mathcal{C}([a,b])$ les trois normes correspondantes L^1, L^2 et L^∞ ne sont plus équivalentes ; de plus un certain nombre d'autres normes voire semi-normes apparaissent dignes d'intérêt.

Pour comparer deux vecteurs nous ne pouvons qu'utiliser les différences des valeurs de leurs composantes, éventuellement modulées. C'est pourquoi un grand nombre de distances se met sous la forme

$$\delta(u, v \in \mathbb{R}^q) = \sum_{j=1}^{q} \left(\alpha_j \|u_j - v_j\|^\beta\right)^{\frac{1}{\beta}} \text{ (différences ponctuelles pondérées)},$$

où β et les α_j sont des réels strictement positifs. Dans un espace fonctionnel cette approche reste envisageable et mène à l'expression suivante

$$\delta(f, h \in \mathcal{C}([a,b])) = \int_a^b \left(\alpha(t)\|f(t) - h(t)\|^\beta\right)^{\frac{1}{\beta}},$$

avec $\alpha : [a,b] \to]0, +\infty[$. Le cas particulier $\alpha = 1$ mène aux distances L^q. D'autres points de vue sont néanmoins possibles. En restant au niveau local nous pouvons par exemple faire intervenir les dérivées (quand elles existent). Au niveau global, la forme générale de la fonction mérite attention.

Mas et Pumo [226] tirent partie du pouvoir discriminant des dérivées pour construire un modèle linéaire à entrées fonctionnelles. Ferraty et Vieu [120] choisissent la semi-distance basée sur les dérivées secondes $\delta(f,h) = \left(\int |f''(t) - h''(t)|^2 dt\right)^{\frac{1}{2}}$ afin de résoudre un problème de classification supervisée fonctionnelle. Abraham et al. [2] utilisent la norme suivante (écrite de façon simplifiée pour le contexte de la thèse) avec $\alpha > 0$.

$$\|f\|_\alpha = \|f\|_\infty + \sup_{s \neq t} \frac{|f(s) - f(t)|}{|s - t|}.$$

Celle-ci permet d'assurer que certains sous-ensembles de fonctions sont totalement bornés. Heckman et Zamar [162] proposent de mesurer les similarités de forme entre deux fonctions en se basant sur le coefficient de corrélation sur les rangs.

Toutefois, dans notre cas la norme L^2 ($\|f\|_2 = \left(\int |f(t)|^2 dt\right)^{\frac{1}{2}}$) est en général suffisante. En effet si les courbes sont suffisamment densément échantillonnées, on peut toujours relier deux fonctions similaires en passant par des courbes telles que les différences soient petites en norme L^2. La figure 2.2 illustre ce phénomène. De plus, utiliser une (dis)similarité qui n'est pas une distance entraînerait des discontinuités au sein des groupes, ce qui irait à l'encontre de l'hypothèse effectuée sur ϕ. Notons que dans le cas où l'échantillonnage n'est pas suffisant, il y a probablement trop peu d'information pour construire des modèles de régression spécialisés.

Sauf mention contraire, la norme L^2 est systématiquement utilisée.

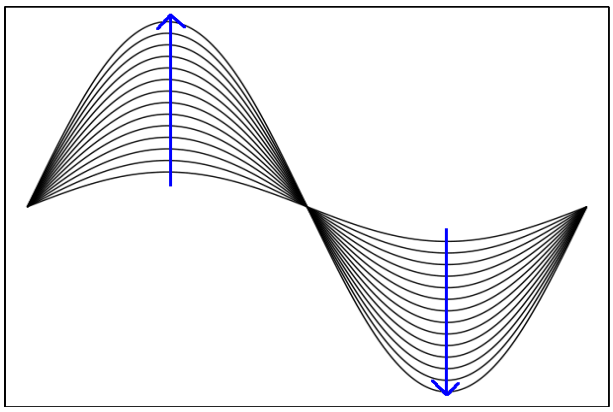

Figure 2.2: Courbes similaires reliées par un "chemin" indiqué en bleu.

2.2 Clustering simultané entrées - sorties

Les couples (x_i, y_i) étant dans l'espace produit $\mathcal{E}_E \times \mathcal{C}([a,b])$, il faut avant toutes choses disposer d'une fonction de comparaison sur ce dernier. Une première idée consiste à combiner des distances connues sur chaque espace :

$$d(u,u') = d^{(p)}(x,x') + \alpha\, d^{(\infty)}(y,y')$$

avec $d^{(p)}$ et $d^{(\infty)}$ distances respectivement dans $\mathcal{E}_E$ et $\mathcal{C}([a,b])$, $x^{(')} \in \mathcal{E}_E$, $y^{(')} \in \mathcal{C}([a,b])$. Le choix de α est cependant très délicat, conditionnant toute l'exécution de l'algorithme de

clustering. De plus, il se peut que la "bonne" valeur du paramètre varie d'une région de l'espace à une autre.

On peut alors envisager de mettre les distances en entrées et sorties à la même échelle en divisant $d^{(p)}(x, x')$ par la moyenne $\bar{d}^{(p)}(x_i, x_j), i, j = 1, \ldots, n$ (respectivement $d^{(\infty)}(y, y')$ par $\bar{d}^{(\infty)}(y_i, y_j), i, j = 1, \ldots, n$), avec $\alpha = 1$. Cependant les entrées et sorties n'ont aucune raison d'être réparties de la même façon autour de leur moyenne. Une autre approche consiste alors à standardiser les données : les centrer puis diviser par l'écart-type le long de chaque dimension. Cela nécessite probablement de réduire la dimension des fonctions, en les décomposant sur une base par exemple. Non seulement on perd un peu d'information, mais en plus la division par l'écart-type nivelle l'importance des variables : une composante d'entrée qui n'est que du bruit peut fausser toutes les comparaisons $d^{(p)}(x, x')$. Notons cependant qu'en ce qui concerne notre application un prétraitement est effectué sur les entrées, pouvant donc toutes être supposées significatives. Néanmoins, cette approche n'a pas donné de bons résultats, le nombre de fonctions de base étant difficile à sélectionner.

Peu d'auteurs ont abordé le problème de la classification simultanée entrées-sorties. Wang et al. [333] considèrent un système comportant des entrées x_i et sorties u_i vectorielles, pour lequel ils supposent l'existence de règles floues "$\simeq x \mapsto \simeq u$". Ces relations sont plus faciles à déterminer après une étape de clustering. La méthode appliquée est la suivante :

1. diviser l'espace d'arrivée en q pavés de tailles égales;
2. déterminer le nombre et la composition des sous-clusters induits en entrée, supposés gaussiens;
3. optimiser les paramètres dans chaque cluster (centre, écart-type).

Le degré d'appartenance d'une entrée $x \in \mathcal{E}_E$ à un cluster C de centre $c \in \mathcal{E}_E$ et d'écarts types $\sigma_j, j = 1, \ldots, p$ le long de chaque dimension s'écrit

$$d(x, C) = \exp\left(-\frac{1}{2}\sum_{j=1}^{p}\left(\frac{x_j - c_j}{\sigma_j}\right)^2\right).$$

Utilisant ce degré comme une fonction noyau les auteurs appliquent une formule de régression du type k plus proches voisins. Les paramètres sont alors optimisés à l'étape trois pour que les prédictions soient les meilleures possibles sur un ensemble de test. Bien qu'efficace sur les applications visées par ce dernier article, cette méthode n'est pas utilisable dans notre cas pour deux raisons :

- la classification initiale des sorties est trop sommaire ; on s'attend en effet à avoir une structure assez complexe en les y_i ;
- rien n'indique que les clusters induits en entrées soient de formes gaussiennes.

Shi et Wang [291] s'intéressent au problème de classification et prédiction de données

fonctionnelles. En simplifiant quelque peu leur modèle très général à K clusters on écrit

$$\mathbb{P}(y_i \in C_j) = \frac{\pi_j(x_i) f_j(y_i|\theta_j)}{\sum_{\ell=1}^{K} \pi_\ell(x_i) f(y_i|\theta_\ell)},$$

pour le cluster de sorties C_j et son jeu de paramètres associés θ_j. La fonction f_j est la densité de probabilité en sortie conditionnelle aux paramètres du cluster C_j, choisie gaussienne dans l'article. Les réels $\pi_j(x), j = 1, \ldots, K$ correspondent aux probabilités que l'entrée x soit dans le cluster numéro j, et sont définis comme suit

$$\pi_j(x) = \frac{\exp(\langle x, \gamma_j \rangle)}{\sum_{\ell=1}^{K} \exp(\langle x, \gamma_\ell \rangle)},$$

où les vecteurs $\gamma_j \in \mathbb{R}^p, j = 1, \ldots, K$ sont des paramètres relatifs aux clusters en entrée. La fonction y_i est classée dans le groupe C_{j_0} avec $j_0 = \arg \max_{j=1,\ldots,K} \mathbb{P}(y_i \in C_j)$. Ainsi pour qu'une courbe y_i soit classée dans le cluster C_j, il faut non seulement qu'elle ressemble aux autres courbes du cluster (valeur élevée de la fonction de densité), mais aussi que l'entrée x_i soit orientée dans la direction du paramètre vectoriel γ_j. Le clustering tient alors compte des entrées et sorties simultanément, au prix d'une paramétrisation intensive. Dans le cas général rien ne garantit que les clusters en entrée possèdent des directions vectorielles propres, ni que les courbes en sortie suivent des lois gaussiennes. C'est pourquoi nous abandonnons l'idée de classer simultanément entrées et sorties pour gagner en généralité. En contrepartie nous y perdons un peu en spécificité, mais le modèle développé devant être le plus générique possible cette solution semble acceptable.

La méthodologie suivante est adoptée :

1. classer les sorties y_i en $K' \leq K$ groupes C_j ; cela induit un partitionnement des entrées ;
2. affiner la classification des entrées x_i dans chaque cluster si nécessaire : $C_j \leftarrow C_{j,1} \sqcup \cdots \sqcup C_{j,m_j}$; $\sum_{j=1}^{K'} m_j = K$.

En plus de la contrainte indiquée précédemment, cette approche est motivée par l'hypothèse que les entrées sont réparties quasi uniformément dans leur espace (en général un produit d'intervalles compacts de $\mathbb{R}$). En effet dans ce cas les seules données que l'on a intérêt à classer sont les sorties. Les entrées servent ensuite de contrôle : si un cluster contient deux zones très différentes dans l'espace des entrées, on le divise en deux par exemple. Ceci est illustré sur la figure 2.3 pour un modèle très simple :

$$\phi(x, t) = sin(x)^{\frac{1}{3}} t,$$

avec $x, t \in [0, 4\pi]$. Les fonctions $\phi(x)$ sont des droites, la racine cubique imposant aux pentes d'être soit proches de 1, soit proches de -1. Il y a donc a priori deux clusters en sortie correspondant aux zones $x \in [0, \pi[\cup [2\pi, 3\pi[$ et $x \in [\pi, 2\pi[\cup [3\pi, 4\pi]$. Celles-ci étant composées de deux intervalles disjoints on peut rediviser chaque groupe en deux conformément aux entrées. Cette étape de subdivision est similaire au second point dans l'article de Wang et al. [333]. En revanche la première étape est améliorée, et la troisième inutile car nous privilégions des méthodes non paramétriques pour plus de souplesse.

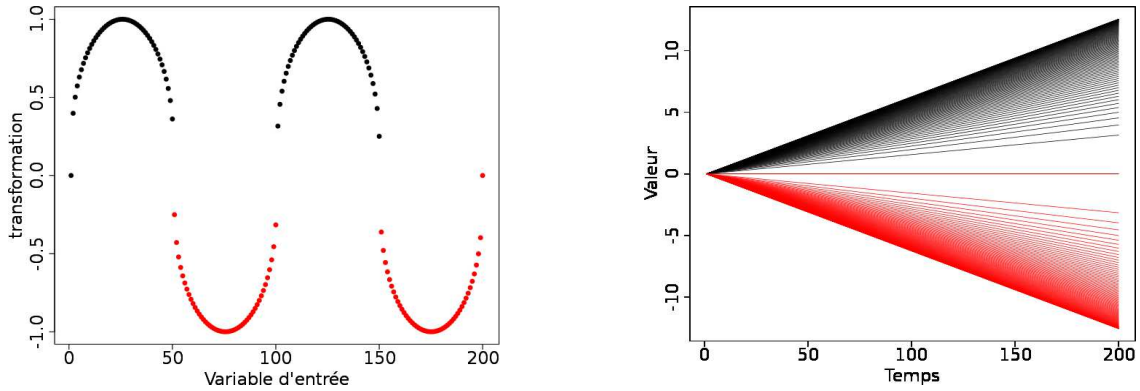

Figure 2.3: Division des clusters par rapport aux entrées, induite de celle en sortie.

2.3 Classification non supervisée fonctionnelle : état de l'art

Nous commençons par un état de l'art de quelques méthodes assez représentatives. La plupart des travaux réalisés sur le clustering de courbes organisent les groupes en une fonction centrale, les autres éléments pouvant s'en écarter légèrement. Plus précisément les courbes y d'un cluster C s'écrivent

$$y = \mu + e\,,$$

avec μ fonction moyenne, et e relativement petite perturbation. Cette écriture est toutefois très générale. Nous distinguons les approches non probabilistes (sans estimations de lois), présentées dans un premier temps, des approches bayésiennes. L'algorithme des k-means étant omniprésent dans presque tous les travaux de clustering de courbes, nous lui consacrons un paragraphe.

2.3.1 Algorithme des k-means

Cette technique de clustering remontant aux articles de Steinhaus [302] et Lloyd [217] est généralement utilisée pour n vecteurs $u_1, \ldots, u_n$ de $\mathbb{R}^q$. L'objectif est de trouver K éléments couramment appelés centres ou centroïdes, qui résument le mieux possible l'information contenue dans l'ensemble $\{u_i\}_{i=1,\ldots,n}$. Un vecteur est alors classé dans le cluster du centre le plus proche. Cet algorithme est très fréquemment utilisé du fait de sa simplicité et de sa rapidité.

Pour déterminer les K centroïdes on cherche assez naturellement à minimiser la quantité suivante appelée distorsion (empirique)

$$W_n(c_1, \ldots, c_K \in \mathbb{R}^q) = \frac{1}{n} \sum_{i=1}^{n} \min_{j=1,\ldots,K} \|u_i - c_j\|^2$$

Cependant, déterminer un jeu de K centres c_i minimisant (globalement) W_n est un problème difficile, impossible à résoudre en temps raisonnable avec un algorithme. Nous avons donc recours à une approximation consistant à améliorer itérativement des centres initialisés en

général aléatoirement. Le pseudo-code est le suivant :

1. initialiser K centres $c_1, \ldots, c_K \in \mathbb{R}^q$ (aléatoirement ou suivant une heuristique) ;
2. tant que la position des centres des clusters ne s'est pas stabilisée, faire :
   (a) (ré)initialiser chaque cluster C_j à vide,
   (b) assigner chaque vecteur u_i au groupe C_j tel que $c_j = \arg \min_{j=1,\ldots,K} \|u_i - c_j\|$,
   (c) (re)calculer les centres de gravité c_j des clusters C_j nouvellement formés.

La variance intra-cluster décroît de manière significative à chaque itération (Saporta [279], chapitre 12), entraînant une convergence rapide de l'algorithme en pratique. Il est cependant impossible de garantir la convergence vers un minimum global [25] ; le comportement de l'algorithme est en effet très sensible aux conditions initiales. Diverses heuristiques d'initialisation existent pour pallier ce défaut. Elles n'offrent pas de garanties théoriques mais sont assez satisfaisantes en pratique : Bradley et Fayyad [40] en présentent une assez efficace basée sur le bootstrap.

Compte-tenu de la fonction minimisée, l'algorithme fonctionne bien dans le cas où les vrais clusters sont convexes et "suffisamment sphériques" (il faut que les groupes soient contenus dans les régions de Voronoï formées sur les centres des classes). Si les données sont réparties suivant des formes géométriques complexes, le k-means ne trouvera pas les bons groupes. Une façon de contourner ce problème est de passer par un noyau et d'appliquer l'algorithme dans l'espace de représentations, dans le même esprit que le "kernel trick" des Support Vector Machines (voir Girolami [142] et Dhillon et al. [99]). On peut considérer l'algorithme des cartes auto-organisatrices [186; 187] comme une régularisation de celui des k-means, l'astuce mentionnée ci-dessus pouvant aussi s'appliquer. La version originale de la méthode des k-means est cependant très utilisée, car elle évite le réglage de divers paramètres (ceux du noyau et la décroissance de la température) tout en étant très rapide.

Les méthodes des deux paragraphes suivants constituent un résumé sélectif des travaux réalisés sur le clustering fonctionnel ; nous les présentons brièvement. Quelques unes d'entre elles comportent des idées intéressantes pour la suite de notre travail.

2.3.2 Méthodes non probabilistes

Tarpey et Kinateder [307] présentent l'algorithme des k-means fonctionnel. Celui-ci est en tout point semblable au k-means vectoriel, en remplaçant la norme euclidienne par la distance L^2. En supposant que les données y_i sont des réalisations d'un processus gaussien, les auteurs montrent qu'il suffit d'effectuer le clustering dans l'espace $\mathbb{R}^K$ des K premiers coefficients de projection des courbes sur la base de Karhunen-Loève. En pratique toutefois cette dernière base ne peut qu'être approchée (par exemple avec l'ACP fonctionnelle présentée en 3.2.1.1), et rien ne garantit la normalité des données.

Abraham et al. [1], Rossi et al. [266] et García-Escudero et Gordaliza [137] choisissent tous de décomposer les courbes sur une base de B-splines cubiques puis d'appliquer un algorithme du type k-means. La décomposition sur une base permet de traiter d'éventuels jeux

de données où les courbes ne seraient pas échantillonnées avec les mêmes pas de temps, et constitue une étape de lissage parfois nécessaire.

Rossi et al. [266] proposent une étape de normalisation des fonctions avant de les projeter sur la base, contrastant avec notre suggestion initiale de réduire d'abord la dimension.

García-Escudero et Gordaliza [137] effectuent un k-means classique sur les $m = \lfloor \alpha n \rfloor$ projections les plus représentatives des données, avec $0 < \alpha \leq 1$. Malgré des résultats d'existence et de convergence asymptotiques, un problème majeur de cette approche est le choix du paramètre α.

Auder et Fischer [15] décomposent également les courbes sur une base orthonormée (Fourier, ondelettes et ACP fonctionnelle), puis classent les coefficients à l'aide du k-means. L'intérêt théorique de ce travail réside dans un résultat de convergence (lorsque le nombre de courbes ainsi que le nombre de fonctions de base tendent vers l'infini), qui vient compléter celui obtenu par Abraham et al. [1]. Du point de vue pratique les distorsions empiriques obtenues pour plusieurs bases orthonormées sont comparées, menant à la conclusion qu'en général la base de Karhunen-Loève approchée et tronquée est indiquée. Cet article est reporté en annexe B car son contenu n'est pas exploité dans la thèse.

Serban et Wasserman [285] présentent une méthode complète de classification non supervisée fonctionnelle. Les courbes sont décomposées sur la base de Fourier puis la classification est opérée par un algorithme k-means sur les coefficients. Outre une étape initiale de lissage, l'originalité de la méthode réside dans l'ajout d'un test statistique simple pour éliminer les fonctions constantes n'apportant pas d'information, et une étape de vérification du clustering. Cette dernière étape compare les clusters obtenus en lissant les courbes ou non avant d'estimer leurs coefficients.

Chiou et Li [66] étendent les centroïdes du k-means fonctionnel à des sous-espaces correspondant à des ACP fonctionnelles locales (d'une manière similaire à la version de Bradley et Mangasarian [41] en dimension finie). Au lieu d'assigner une courbe y au centre le plus proche, on cherche le sous-espace (de centre c_j) qui approxime le mieux y au sens de la norme L^2. Il s'agit donc d'une généralisation du k-means fonctionnel présenté par Tarpey et Kinateder [307].

2.3.3 Méthodes bayésiennes

Contrairement aux méthodes précédentes basées sur la minimisation de la distorsion, celles de cette section sont plus axées sur la modélisation des courbes. Les fonctions dans les applications visées sont souvent assez mal échantillonnées, sur peu de points, ce qui explique cette phase de modélisation. Toutes les approches que nous allons présenter ici supposent les données issues d'un modèle de mélange paramétrique, et essayent d'en déterminer la structure optimale.

Gaffney et Smyth [135; 134] et James et Sugar [170] supposent tous que les données sont des tirages indépendants d'un certain générateur de fonctions aléatoires, répartissant les courbes dans K groupes, chacun ayant une probabilité d'occurrence $\pi_j > 0, j = 1, \ldots, K$ ($\sum_{j=1}^{K} \pi_j = 1$). La formule suivante appelée vraisemblance classifiante donne la densité de

probabilité globale du jeu des n courbes, en fonction des paramètres des clusters $\theta_j \in \mathbb{R}^q$

$$\mathbb{P}(y_1, \ldots, y_n | \theta_1, \ldots, \theta_K) = \prod_{i=1}^{n} \sum_{j=1}^{K} \pi_j f_j(y_i | \theta_j) \, .$$

Après affectation dans le cluster numéro j on suppose ainsi que les courbes y_i suivent la même loi paramétrique de densité $f_j(.|\theta_j)$. C'est principalement dans l'écriture de cette densité de probabilité que les trois méthodes diffèrent. Celles-ci imposent un modèle paramétrique, ponctuel ou bien en décomposition sur une base.

Dans sa thèse, Zhou [358] présente un modèle de classification de courbes hiérarchique entièrement bayésien. C'est-à-dire qu'une fois des lois a priori déterminées sur certains paramètres du modèle, tous les autres sont estimés par des calculs de probabilités conditionnelles, y compris le nombre de clusters K (nous consacrons par la suite un paragraphe entier au choix de K, et ne développons donc pas plus son estimation ici). Le modèle en question se base sur une description ponctuelle des courbes : $\forall i \in 1, \ldots, n$, $y_{ij} = y_i(t_j), j = 1, \ldots, D$, adaptée aux applications visées (peu de points d'échantillonnage). Un algorithme MCMC permet d'ajuster tous les paramètres itérativement, suivant les niveaux hiérarchiques.

Considérant l'approximation polynomiale inadéquate pour de grands degrés (> 3), Ma et al. [218] utilisent des splines de lissage pour représenter les centres des clusters μ_j. Ils écrivent $y_i = \mu_j + b_i + \varepsilon_i$ avec ε_i gaussienne indépendante, étendant l'approche de Gaffney et Smyth [135]. Le terme de biais b_i modélise dans une certaine mesure l'écart entre une courbe et le centre de son cluster en tenant compte des corrélations temporelles.

Les méthodes présentées jusqu'alors fixent un mode de représentation des fonctions, puis optimisent des paramètres vectoriels. Shubhankar et Bani [292] emploient une approche plus souple, consistant à optimiser simultanément la base de représentation des fonctions ainsi que les coefficients. Ces derniers sont modélisés par une loi normale − quand ils sont non nuls. Les auteurs choisissent une base d'ondelettes car celle-ci fournit des représentations parcimonieuses pour des espaces de fonctions très généraux (par exemple les espaces de Besov [98]). Un modèle d'inférence hiérarchique est alors développé puis les paramètres sont optimisés via un algorithme MCMC.

2.3.4 Bilan

Les méthodes du type k-means ont l'avantage d'être rapides et non paramétriques comparées aux approches bayésiennes (qui ont recours à des algorithmes E-M ou MCMC). Cependant, elles sont plus sensibles au bruit et aux outliers. En effet les méthodes probabilistes tiennent naturellement compte de ces phénomènes via des lois à supports non bornés.

Dans le cadre de cette thèse l'approche de Shubhankar et Bani [292] (en quelque sorte une spécialisation du travail de Zhou [358]) semble la plus pertinente car elle permet de traiter des données complexes échantillonnées sur de nombreux points de discrétisation. Elle reste cependant paramétrique, imposant des formes aux clusters. Les méthodes du type k-means contraignent aussi les formes des clusters via la fonction à optimiser (distorsion). C'est ce point que nous allons améliorer, au prix d'une interprétabilité plus faible. En effet

toutes les approches présentées aux deux derniers paragraphes ont l'avantage de présenter un résumé visuel des clusters via leurs centres. Nous ne pouvons pas conserver cet avantage en permettant des clusters de formes arbitraires.

Les méthodes de clustering de courbes présentées jusqu'alors sont assez peu sophistiquées comparées aux divers algorithmes de data-mining dans $\mathbb{R}^q$. En effet les auteurs préfèrent se focaliser sur la nature fonctionnelle des données, qui est déjà très complexe. Dans la seconde partie de ce chapitre nous proposons une méthode non spécifique au cadre fonctionnel (hormis dans le choix de la mesure de comparaison inter-courbes), mais qui se révèle plus appropriée dans le cas général dès lors que le taux d'échantillonnage des courbes est suffisant. Compte-tenu de la difficulté à régler les nombreux paramètres a priori des méthodes bayésiennes, nous utiliserons l'algorithme de Chiou et Li [66] − qui généralise le k-means fonctionnel − lorsque n est trop petit pour chercher des clusters de formes arbitraires. Voici les étapes de cet algorithme.

Notations :
$\mu^{(c)}$ *: centre du cluster* C.
$B^{(c)}$ *: base locale décrivant le cluster* C *(matrice de* K_C *fonctions en lignes,* $K_C \in \mathbb{N}^*$ *pouvant varier d'un groupe à l'autre.)*

1. Effectuer un k-means sur les courbes y_i pour initialiser les centres $\mu^{(c)}$.
2. Estimer les bases locales $B^{(c)}$ à l'aide de l'ACP fonctionnelle.
3. Pour chaque courbe y_i :
   (a) recalculer $\mu^{(c)}$ et $B^{(c)}$ pour le cluster contenant y_i, en enlevant y_i ;
   (b) mémoriser le numéro du groupe dont la structure $(\mu^{(c)}, B^{(c)})$ approxime le mieux y_i ;
   (c) rétablir $\mu^{(c)}$ et $B^{(c)}$ pour le cluster contenant y_i.
4. Mettre à jour la composition des classes à l'aide des numéros retenus ; si la composition a changé, estimer les centres $\mu^{(c)}$ à nouveau puis revenir en 2.

Notons pour conclure sur l'état de l'art des méthodes fonctionnelles qu'un certain nombre d'études ont été réalisées pour la classification de courbes décalées dans le temps [133; 258; 215]. Les courbes dont nous disposons étant en principe synchronisées, nous n'aurons pas besoin de ces techniques.

Motivations pour une méthode "shape free"

On peut facilement imaginer des clusters de formes circulaires, toriques ou plus généralement non convexes dans l'espace $\mathbb{R}^q$; la figure 2.4 en présente un exemple, emprunté à l'article de Karypis et al. [176]. Ces clusters peuvent être transposés dans un espace de courbes en supposant que les coordonnées q-dimensionnelles sont des coefficients de décomposition sur une base orthonormée. Sans indications physiques préalables, nous faisons ainsi l'hypothèse que les clusters (fonctionnels) ne sont pas forcément organisés autour d'un centre. Cela exclut l'écriture $y_i = \mu_j + e$ où e est un terme d'écart de moyenne nulle : les

groupes sont décrits uniquement via les différences entres les fonctions elles-mêmes, ce qui reste cohérent avec la vague définition donnée en introduction au chapitre.

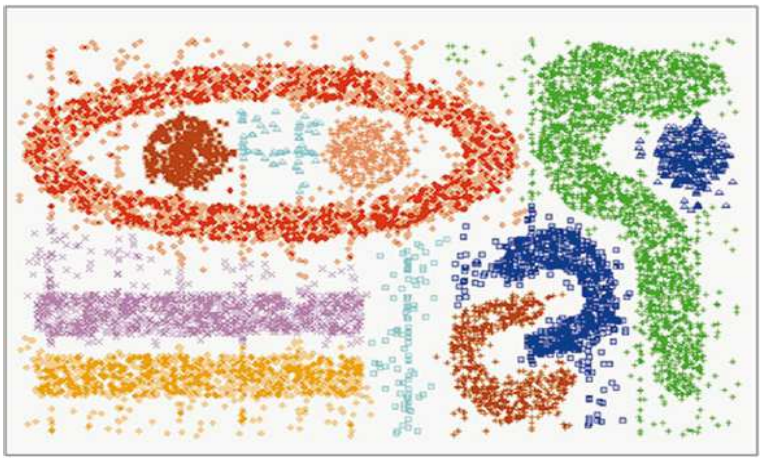

Figure 2.4: Clusters de formes arbitraires dans $\mathbb{R}^2$.

Au lieu des données y_i le point de départ est alors la matrice de taille $n \times n$ des distances inter-courbes : $d_{ij} = d(y_i, y_j)$, "oubliant" en apparence la nature fonctionnelle des données. Celle-ci peut cependant être prise en compte dans le calcul de la distance, ou plus généralement dans le calcul d'une mesure de comparaison quelconque entre courbes (similarité, dissimilarité). La suite de ce chapitre s'intéresse essentiellement au clustering à partir d'une matrice de (dis)similarités, naturellement représentée par un graphe.

2.4 Clustering dans un graphe : état de l'art

Nous présentons d'abord quelques techniques existantes non directement reliées à une marche aléatoire dans un graphe, puis des méthodes utilisant la marche aléatoire $\mathcal{M}$ définie en 1.2.1 (en particulier deux qui utilisent la distance ECT). Il existe beaucoup d'algorithmes de clustering dans un graphe : ceux présentés ici constituent un état de l'art sélectif sans prétention d'exhaustivité. Gan et al. [136] passent en revue un nombre considérable d'approches, dont certaines (comme les algorithmes évolutionnaires au chapitre 10) sont absentes de la bibliographie qui va suivre. En général on recherche une partition minimisant les liens inter-clusters tout en maximisant les connexions intra-clusters. Sauf mention contraire le graphe représentant les données, noté G, est toujours supposé connexe.

2.4.1 Algorithmes sans marche aléatoire

Mentionnons tout d'abord le clustering spectral, qui occupe une place un peu à part car il a été étudié par plusieurs auteurs. Celui-ci semble avoir été décrit pour la première fois dans l'article de Shi et Malik [290]. Partant de la matrice des similarités S, l'objectif est d'approximer un problème de coupe minimale normalisée. On associe à un partitionnement en K classes $C_1, \dots, C_K$ la mesure suivante :

$$Ncut = \sum_{i=1}^{K} \frac{cut(i)}{Vol_{C_i}},$$

dans laquelle les numérateurs $cut(i) = \sum_{i \in C_i, j \in \{1,\dots,n\} C_i} s_{ij}$ indiquent à quel degré un cluster est connecté au reste du graphe, pondérés par les importances des groupes (Vol_{C_i}). Cette dernière pondération est nécessaire pour ne pas obtenir une solution triviale (isolant un sommet par exemple), mais rend le problème impossible à résoudre en temps raisonnable. La partition optimale ($Ncut$ minimal) peut cependant être approchée en appliquant un algorithme de clustering simple sur les rangées de la matrice M formée par les vecteurs propres de $D^{-1}L$ (ou $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$, voir von Luxburg [329]) en colonnes. Shi et Malik [290] partitionnent les données en deux classes selon la seconde colonne de M et itèrent le procédé jusqu'à obtenir le nombre de groupes désiré. Qiu et Hancock [256] retrouvent cette dernière méthode en exprimant la distance ECT d'une nouvelle manière. Ng et al. [244] préfère appliquer le k-means avec $K > 2$.

L'approximation réalisée par le clustering spectral peut se justifier à l'aide de la théorie des perturbations matricielles (Ng et al. [244], von Luxburg [329]). Finalement, bien que cet algorithme comporte peu de garanties théoriques il est plus général que les k-means et souvent efficace en pratique, tout en étant relativement simple (à décrire et à coder).

Méthodes utilisant un arbre couvrant

Zahn [350] présente une méthode de clustering basée sur l'arbre couvrant de poids minimal (en terme de distances) du graphe G. Un arbre couvrant T est un sous-graphe contenant tous les sommets de G, connexe et sans cycles. Ainsi, en particulier, l'unicité d'un chemin entre deux sommets fixés est assurée dans T. Le poids d'un arbre couvrant est défini comme la somme des poids des arêtes, ceux-ci correspondant aux distances entre les sommets. Quelques stratégies sont proposées au cas par cas, supprimant des arêtes de poids élevé dans T. Les parties connexes résultantes sont les groupes recherchés.

L'approche de Wu et Leahy [344] est elle aussi basée sur un sous-graphe de G. Plus précisément, un arbre T^* équivalent au graphe G est construit, via l'algorithme de Gomory et Hu [145] ou de Gusfield [150]. "Équivalent" signifie ici que les sommes des poids des arêtes le long de l'unique chemin joignant deux sommets s et t dans T^* est égale à la valeur d'un flot maximal entre ces deux sommets dans G. (voir Ahuja et al. [4] à ce sujet). Il ne reste alors plus qu'à supprimer les arcs de T^* jusqu'à obtenir le nombre de parties connexes désirées.

Flake et al. [124] utilisent un artifice modifiant le graphe G : un sommet fictif est ajouté, connecté à tous les sommets de G avec un poids $\alpha > 0$. Comme dans la méthode précédente un arbre équivalent T^* au graphe complété est alors construit ; les parties connexes de T^* après suppression du sommet fictif sont les clusters du graphe initial.

Méthodes basées sur des coupes minimales

Gdalyahu et al. [138] cherchent à déterminer la coupe minimale (non normalisée) du graphe G. Ce problème peut être résolu très efficacement pour $K = 2$ [304; 45], mais devient NP-complet pour $K \geq 3$ (comme Dahlhaus et al. [81] le démontrent). C'est pourquoi les auteurs utilisent un algorithme stochastique basé sur un opérateur de contraction, consistant

à sélectionner puis fusionner deux clusters avec une probabilité proportionnelle à la somme des poids des arêtes les connectant.

Stein et Niggemann [300] visent quant à eux à maximiser la quantité suivante

$$\Lambda(\mathcal{C}) = \sum_{i=1}^{K} |C_i| \lambda_i \,,$$

où $\mathcal{C}$ désigne une partition de G en K clusters $C_1, \ldots, C_K$, λ_i étant le poids minimal d'un ensemble d'arêtes devant être enlevées à C_i pour le déconnecter. Le terme $\Lambda(\mathcal{C})$ mesure la connectivité partielle (pour chaque cluster) pondérée (par les tailles des classes) du graphe G. On cherche ainsi des groupes de tailles similaires − comme dans le clustering spectral −, aux connexions internes fortes, sans se préoccuper des capacités des arêtes aux interfaces cluster-cluster.

Autres méthodes

Fischer et Züller [123] estiment les distances géodésiques dans le graphe afin d'obtenir de nouvelles dissimilarités $d_{ij}, i, j = 1, \ldots, n$, mieux adaptées pour des groupes de formes quelconques. L'objectif est de minimiser la moyenne des distances intra-clusters. À cette fin, une matrice $M \in \{0,1\}^{n \times K}$ modélisant les classes est introduite ; $M_{ij} = 1$ si et seulement si i appartient au cluster j ($\sum_{i=1}^{n} M_{ij} = 1$ pour tout $j = 1, \ldots, K$). Une heuristique est alors implémentée afin de minimiser

$$\mathcal{H}(M, \delta) = \sum_{j=1}^{K} \sum_{i=1}^{n} M_{ij} \delta_{ij} \,,$$

où δ_{ij} représente la distance moyenne entre i et les points du cluster j.

Elghazel et al. [113] présentent une approche originale basée sur le coloriage de graphe. Les arêtes sont étiquetées par les distances entre les sommets et non par les similarités, puis les sommets reçoivent chacun une couleur représentant une classe. On cherche à éloigner le plus possible deux couleurs différentes sans se préoccuper des similarités entre les éléments d'une même couleur, ce qui est une nouvelle façon de voir les choses.

Enfin, Kyoung et Choi [194] choisissent de décomposer G en un certain nombre de sous-graphes réguliers disjoints ; un graphe est dit régulier quand tous ses sommets ont le même degré (défini comme la somme des poids des arêtes incidentes). Ces sous-graphes sont ensuite remaniés jusqu'à l'obtention des clusters finaux.

2.4.2 Algorithmes utilisant la marche aléatoire $\mathcal{M}$

Peu d'algorithmes de clustering utilisent la distance commute-time présentée en 1.2.1, son introduction en data-mining semblant relativement récente. Nous en mentionnons deux ici.

Harel et Koren [154] définissent les "probabilités de fuites" $\mathbb{P}_{escape}$: partant de i, $\mathbb{P}_{escape}(i, j)$ est égale à la probabilité d'atteindre j avant de revenir en i. Ces probabilités peuvent

se calculer facilement par récurrence et modélisent en quelque sorte les similarités (non symétriques) sur le graphe. En effet si i et j sont dans deux clusters différents il est beaucoup plus probable de revenir sur le sommet initial avant d'atteindre l'autre. Afin de rendre la probabilité de fuite symétrique et de minimiser la charge de calcul les auteurs définissent la "circular escape" :

$$CE^T(i,j) = \mathbb{P}^T_{escape}(i,j).\mathbb{P}^T_{escape}(j,i)\,,$$

où $\mathbb{P}^T_{escape}$ est défini comme $\mathbb{P}_{escape}$ dans un sous-graphe local de taille T aux voisinages de i et j. Contrairement à la distance ECT, la quantité CE^T est une similarité : elle peut donc servir à nouveau à définir une nouvelle marche aléatoire, qui elle-même permettra de calculer de nouvelles valeurs de CE^T, etc. La procédure de classification consiste alors en les deux étapes suivantes :

1. itérer le calcul des CE^T plusieurs fois (à T fixé), en déduire les similarités finales entre les sommets du graphe ;
2. utiliser l'algorithme de clustering hiérarchique dans le graphe résultant.

Yen et al. [346] remarquent que la dernière expression obtenue pour le temps d'aller-retour en fonction du pseudo-inverse du laplacien peut se réécrire de la façon suivante :

$$ct(i,j) = Vol_G{}^t(e_i - e_j)L^+(e_i - e_j)$$

avec e_i vecteur colonne de taille n contenant un 1 en i^{eme} position et 0 ailleurs. La matrice L^+ étant symétrique et semi-définie positive, on voit ainsi que $ct(i,j)$ est le carré d'une distance euclidienne sur les sommets du graphe. Les auteurs implémentent alors un algorithme des k-médoïdes (comme un k-means, en imposant aux centres d'appartenir aux observations) s'exécutant uniquement à partir des distances ECT calculées dans le graphe. Yen et al. [347] proposent d'effectuer une transformation sigmoïdale sur les éléments de L^+ avant de calculer les distances ECT. Celles-ci sont alors normalisées dans l'intervalle $[0,1]$, au prix de l'introduction d'un nouveau paramètre.

Toutes ces méthodes constituent des heuristiques de partitionnement optimal d'un graphe (souvent en retirant certaines arêtes). L'approche de Harel et Koren [154] semble très aboutie mais nécessite le réglage de paramètres additionnels. Nous privilégions donc l'utilisation de la distance ECT sur le graphe G. Finalement, l'algorithme de clustering hiérarchique est présenté. Celui-ci est bien adapté à la détermination du nombre de classes comme nous le verrons.

L'étape de clustering s'effectue à l'aide de l'algorithme de classification hiérarchique (ascendante), basé sur les distances ECT.

2.4.3 Classification ascendante hiérarchique

Le principe de cette méthode est le suivant : partant de n vecteurs $u_1, \ldots, u_n \in \mathbb{R}^q$ considérés comme n petits clusters $C_1, \ldots, C_n$, $n-1$ fusions successives sont appliquées. Supposant un compteur c initialisé à n, une fusion correspond à ces quatre étapes :

1. choisir $i, j \in \{1, \ldots, c\}$, $i < j$;

2. remplacer C_i par $C_i \cup C_j$;
3. pour ℓ allant de j à $c-1$, remplacer C_ℓ par $C_{\ell+1}$;
4. décrémenter c : $c \leftarrow c-1$.

Ainsi tous les éléments se retrouvent dans C_1 après la dernière fusion. En remontant le processus K clusters sont obtenus, $K = 1, \ldots, n$. Une heuristique relativement efficace pour déterminer la valeur finale de K (utilisée par Harel et Koren [154]) consiste à attribuer une note à chaque fusion effectuée, égale au produit des cardinaux des classes avant regroupement. Partant du groupe C_1 final, tant que les notes calculées sont supérieures à un certain seuil on revient à l'étape précédente.

Afin de sélectionner les fusions à effectuer, on définit une distance d_C entre deux clusters A et B. Cette distance est minimisée à chaque étape ; par exemple,

- la distance "single-linkage" :

$$d_C(A,B) = \min_{u\in A}\min_{u'\in B} d(u,u')$$

- la distance "complete-linkage" :

$$d_C(A,B) = \max_{u\in A}\max_{u\in B} d(u,u')$$

Celles-ci peuvent être vues comme deux extrêmes, la première échouant souvent en présence d'un bruit même léger ou d'un mauvais échantillonnage (effet de chaînage). La seconde distance correspond à une heuristique cherchant à minimiser le diamètre maximal d'une classe [19; 153] ; on peut la voir comme une version combinatoire des k-means. Karypis et al. [176] proposent une distance sophistiquée tenant compte de la "dynamique" des données, basée sur la proximité entre deux clusters candidats à la fusion et leur connectivité interne. Celle-ci réalise un compromis entre la prise en compte de la proximité relative inter et intra clusters, mais nécessite l'ajustement empirique d'au moins un paramètre. Fiorio et Mas [121] proposent d'estimer l'éloignement entre deux groupes (de pixels, dans leur application) via un test statistique comparant les moyennes des classes. Nous utilisons une mesure intermédiaire entre les single et complete link : la distance de Ward [335; 306]. Quelques définitions sont nécessaires avant sa présentation.

L'inertie totale du nuage de point $u_1, \ldots, u_n$ se définit comme suit :

$$I_{tot} = \sum_{i=1}^{n} d^2(u_i, \gamma),$$

γ étant le centre de gravité de tous les vecteurs u_i. Les inerties intra-classes et inter-classes pour un regroupement partiel au niveau c ont pour définitions respectives

$$I_{intra} = \sum_{j=1}^{c}\sum_{u\in C_j} d^2(u,\gamma_j)\,,$$

où γ_j est le centre de gravité du cluster C_j, et

$$I_{inter} = \sum_{j=1}^{c} |C_j| d^2(\gamma_j, \gamma) \, .$$

Au fur et à mesure des regroupements effectués, l'inertie intra-classes augmente tandis que l'inertie inter-classes diminue. Nous disposons du théorème suivant.

Théorème 2.4.1 *Pour tout partitionnement de $\{u_i\}_{i=1,\dots,n}$ en c classes la somme des inerties intra et inter classes est constante, égale à l'inertie totale :*

$$I_{tot} = I_{intra} + I_{inter} \, .$$

La *distance de Ward* entre deux groupes C_i et C_j s'écrit alors :

$$d_C(C_i, C_j) = \sqrt{\frac{|C_i||C_j|}{|C_i| + |C_j|}} d(\gamma_i, \gamma_j) \, .$$

Cette définition est motivée par le résultat suivant, démontré après quelques lignes de calculs.

Proposition 2.4.1 *Lorsque l'on fusionne C_i et C_j, l'inertie inter-classes est diminuée de ΔI, avec*

$$\Delta I = \frac{|C_i||C_j|}{|C_i| + |C_j|} d^2(\gamma_i, \gamma_j) \, .$$

Minimiser la distance de Ward revient donc à rechercher la paire de clusters dont la fusion minimise la variation d'inertie, favorisant les partitions "stables".

Celle-ci donne de très bons résultats quand d est la distance euclidienne et peut être choisie systématiquement en pratique. L'algorithme de clustering hiérarchique reste cependant plus sensible au bruit et à la qualité de l'échantillon que le k-means. Ce dernier défaut est pallié en fusionnant les clusters de trop petite taille en fin d'algorithme, le rendant ainsi moins sensible aux outliers que le k-means. Le bruit reste cependant un réel problème pour d'assez petits échantillons. Le clustering hiérarchique peut être implémenté avec une complexité en temps de $O(n^2)$, ce qui n'est pas rédhibitoire pour nos applications où n ne dépasse guère 1000.

Avec cet algorithme, l'accent est mis sur la capacité d'apprendre des clusters de n'importe quelle forme au détriment de l'interprétabilité des résultats. Ce n'est pas très grave car l'étape de classification s'effectuant en interne du code du modèle, l'utilisateur final n'a en principe pas besoin de connaître les groupes de courbes.

2.5 Détermination du nombre de classes

Il n'y a pas de méthode universelle donnant toujours le bon nombre de classes ; diverses heuristiques existent, chacune spécifique à un ou plusieurs types de données. Nous supposons classer n vecteurs $u_1, \dots, u_n$ de $\mathbb{R}^q$, même si la plupart des algorithmes présentés s'applique

dans des espaces plus généraux. Deux grandes familles de méthodes se distinguent : celles cherchant à évaluer une partition par une formule directe, et celles utilisant de multiples résultats de l'algorithme (dans l'esprit du bootstrap [39; 84]). La valeur de K optimale (selon l'algorithme) est notée $\hat{K}$.

2.5.1 Méthodes basées sur la structure des données

Plusieurs critères sont basés sur la distorsion W_n, dont la définition est rappelée ici

$$W_n(c_1, \ldots, c_K) = \frac{1}{n} \sum_{i=1}^{n} \min_{j=1,\ldots,K} \|u_i - c_j\|^2 ,$$

où $c_1, \ldots, c_K$ sont K centres représentatifs des données. On recherche alors un "point anguleux" dans le graphe d'évolution de W_n en fonction de K, $\hat{K}$ étant égal à l'abscisse à partir de laquelle la distorsion ne diminue plus significativement [275]. Cependant, même en présence de clusters de la bonne forme (convexes, "quasi-sphériques") l'interprétation de la courbe d'évolution de W_n peut être délicate, comme la figure 2.5 le montre ($K = 4$ ou $K = 6$?). C'est pourquoi on préfère généralement tracer des quantités plus complexes (liées à W_n) en fonction de K [183; 70; 173; 288; 184; 12], ou bien effectuer une transformation de la courbe $K \mapsto W_n$ [314; 305; 345].

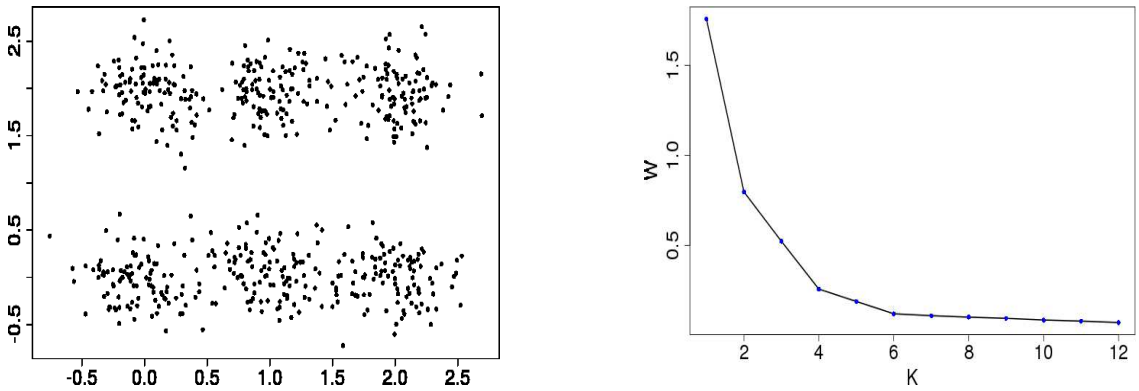

Figure 2.5: Six clusters gaussiens à gauche, W_n en fonction de K à droite.

Le cadre bayésien permet de modéliser des données plus générales, et est adapté lorsque les groupes sont de formes non convexes/sphériques mais connues ou estimées. Zhou [358] propose d'estimer K via des algorithmes MCMC spécifiques [261; 303], qui sont intégrés à un procédé de clustering. Restant dans un cadre probabiliste, il existe de nombreux critères basés sur la théorie de l'information : en résumé, un modèle probabiliste est ajusté sur les données, puis ses paramètres optimaux sont estimés pour plusieurs valeurs de K. Une formule du type suivant est minimisée

$$f(K) = -\alpha \log L(\hat{\theta}, K) + PEN(K, d_{lib}) ,$$

avec $\alpha > 0$ et $\hat{\theta}$ les paramètres estimés du modèle, $\log L$ désignant la log-vraisemblance. PEN est une fonction de pénalité positive, croissante avec K et d_{lib} (nombre de paramètres libres) pour ne pas favoriser les modèles trop complexes. Le chapitre 6 du livre de McLachlan et Peel [229] ainsi que le rapport de Oliveira-Brochado et Martins [246] présentent des

récapitulatifs de plusieurs techniques bayésiennes d'estimation de K. Ces critères nécessitent la connaissance (ou l'estimation) du modèle génératif des données. En pratique, faute de ces informations on se place souvent dans le cas gaussien, contraignant la forme des classes.

Clusters de formes arbitraires

Kyrgyzov et al. [195] utilisent une formule du type "− log-vraisemblance ($\log L$) + pénalité", où $\log L$ est calculé pour le modèle dans un espace de représentation abstrait. Le passage par cet espace permet de s'affranchir de la contrainte de convexité / sphéricité sur les clusters. La méthode est particulièrement adaptée au kernel k-means [142; 99], car la classification s'effectue alors dans l'espace de représentation.

Concernant l'algorithme du clustering spectral, plusieurs heuristiques existent pour déterminer le nombre de classes. Zheng et Lin [357] utilisent la marche aléatoire $\mathcal{M}$ définie en 1.2.1 afin de renforcer les similarités, avant d'appliquer la méthode classique du "saut spectral" choisissant $\hat{K} = \arg \max_{K=1,\dots,K_{max}} |\lambda_K - \lambda_{K+1}|$. λ_K désigne la K^{eme} valeur propre du laplacien. Sanguinetti et al. [278] observent que les rangées de la matrice formée par les vecteurs propres en colonnes se partitionnent (idéalement) selon des sous-espaces deux à deux orthogonaux. Les clusters obtenus sont de formes allongées suivant des directions radiales. Une version des k-means adaptée à cette situation est alors proposée.

Biau et al. [30] utilisent la définition d'un cluster C dans $\mathbb{R}^q$ donnée par Hartigan [155] : C est une composante connexe de l'ensemble $\{u \in X / f(u) \geq t\}$, f désignant la densité de probabilité des données. Ils montrent que les parties connexes de l'ϵ_n-graphe (avec $\epsilon_n \to 0$) contenant les sommets de densité (estimée) supérieure à t convergent vers les classes de niveau t. Il suffit donc de compter les composantes connexes.

Finalement, Halkidi et Vazirgiannis [151] proposent une méthode pouvant s'utiliser en conjonction avec n'importe quel algorithme ; en effet celle-ci consiste à noter une partition en fonction de critères statistiques calculés sur les clusters. La différence avec les premiers critères évoqués vient du fait que les quantités mesurées ici sont essentiellement locales, permettant d'évaluer des groupes de toutes formes.

Toutes les approches de cette première partie fondent leur résultats sur l'organisation des clusters pour une partition donnée. Nous allons voir à présent quelques méthodes jugeant non pas de la qualité d'une partition directement, mais faisant appel plusieurs fois à l'algorithme de classification non supervisé sur diverses données.

2.5.2 Méthodes basées sur la stabilité des partitions

L'idée principale de ces approches consiste à appliquer l'algorithme de clustering sur un certain nombre de sous-ensembles des données initiales, avant de comparer les partitions obtenues. Une autre façon de faire consiste à perturber le jeu de données initial (Möller et Radke [236]) au lieu de sous-échantillonner : cela a l'avantage de perdre moins d'informations, mais introduit un problème supplémentaire qui est celui de l'estimation des variances des bruits à appliquer.

Levine et Domany [208] appliquent l'algorithme sur m sous-échantillons à $\alpha\, n$ éléments ($\alpha \in]0,1[$), obtenant m partitionnements de sous-ensembles des données initiales. Les n données sont elles aussi classées, les groupes obtenus servant de référence. La stabilité des clusters est alors estimée par la moyenne suivante :

$$\mathcal{I} = \overline{\delta(T_{ij}, T_{ij}^{ref})}\,,$$

avec $\delta(u,v) = \mathbf{1}_{u=v}$. T_{ij} vaut 1 si i et j sont dans la même classe lors d'un des m partitionnements, 0 dans le cas contraire. Ainsi $\mathcal{I}$ exprime l'adéquation moyenne des clusters des sous-échantillons aux groupes de référence. Le nombre de groupes est finalement estimé par la plus grande valeur de K telle que $\mathcal{I}$ soit au-delà d'un certain seuil (0.8 à 0.9).

Avant de poursuivre il nous faut préciser la notion de comparaison entre deux partitionnements P et P', dont la définition est restée floue au paragraphe précédent. Un exemple simpliste consiste à compter les paires d'éléments présents dans les deux partitions, qui se retrouvent dans le même cluster dans les deux cas :

$$\mathcal{I} = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} \delta(T_{ij}^{(P)}, T_{ij}^{(P')})\,,$$

avec les notations précédentes, i et j désignant deux éléments à classer. Cependant, de nombreux indices plus sophistiqués existent, un des premiers ayant été présenté par Rand [259]. Meilă [230] présente un bilan de plusieurs indices et en propose un basé sur la variation d'information, qui a la particularité d'être une métrique sur l'espace des partitionnements. Lorsque ce ne sera pas précisé on pourra supposer le critère simple donné ci-dessus.

Ben-Hur et al. [24] répètent N fois la procédure suivante, pour chaque valeur de K :

1. tirer deux sous-échantillons des données $\mathcal{S}$ et $\mathcal{S}'$, chacun de taille αn ($\alpha \in]0,1[$) ;
2. partitionner $\mathcal{S}$ et $\mathcal{S}'$ en P et P' respectivement ;
3. calculer un indice de similarité entre les deux partitionnements sur l'intersection $\mathcal{S} \cap \mathcal{S}'$.

On obtient en sortie K_{max} distributions empiriques des similarités, une pour chaque valeur de K. Les auteurs estiment ensuite visuellement $\hat{K}$ en traçant les histogrammes. Giurcăneanu et Tăbuş [143] améliorent ce dernier algorithme en introduisant une procédure de détermination automatique de $\hat{K}$, à partir des répartitions empiriques.

La méthode de Monti et al. [239] est très générale, basée sur les probabilités de voir deux éléments dans le même cluster. Elle peut se résumer comme suit. Pour chaque valeur de K initialiser $M^{(K)}$ de taille $n \times n$ à zéro, puis pour h allant de 1 à N faire :

1. rééchantillonner, ou sous-échantillonner les données par une méthode quelconque pour obtenir un ensemble $D^{(h)}$;
2. appliquer l'algorithme de clustering sur $D^{(h)}$, obtenant la partition $P^{(h)}$;
3. calculer la matrice de composition $C^{(h)}$ de taille $n \times n$; $C_{ij}^{(h)} = 1$ si i et j sont présents dans $D^{(h)}$ et classés dans le même cluster, 0 sinon ;

4. $M^{(K)} \leftarrow M^{(K)} + C^{(h)}$,

aboutissant à K_{max} matrices $M^{(K)}$, $M_{ij}^{(K)}$ représentant la probabilité que i et j soient dans le même cluster lors d'un partitionnement en K classes. $\hat{K}$ est estimé à la plus grande valeur de K telle que la distribution des éléments de $M^{(K)}$ soit "clairement" multimodale. Möller et Radke [237] choisissent en quelque sorte de spécialiser ce dernier algorithme. Ils utilisent le fuzzy k-means, avec une technique de rééchantillonnage de tous les éléments basées sur les k plus proches voisins.

Avec classification supervisée

En plus des étapes principales des méthodes précédentes, celles de ce paragraphe ont la particularité d'utiliser un algorithme de classification supervisée basé sur les étiquettes des classes. Dans notre application des entrées doivent être classées aussi, d'où l'intérêt de telles approches.

La méthode développée par Tibshirani et al. [313] est adaptée pour des clusterings de type k-means ; elle pourrait cependant être étendue au cas du kernel k-means, permettant des classes de toutes formes. Dans un premier temps celle-ci divise les données en un ensemble d'entraînement E et un ensemble de test T, puis applique l'algorithme de clustering (basé sur K centres) aux deux ensembles. Les K centroïdes de la partition de E sont alors utilisés pour définir de nouvelles classes dans T (règle du 1-plus-proche-voisin en classification supervisée). Les deux partitions de T sont finalement comparées, l'opération pouvant éventuellement être répétée N fois. La technique proposée par Roth et al. [267] peut être vue comme une généralisation de cette dernière, avec des algorithmes de clustering et de classification quelconques. La principale différence réside dans la façon de comparer les deux partitions de T : celle-ci s'effectue cluster par cluster, après un réétiquetage des éléments via la méthode hongroise par exemple (Papadimitriou et Steiglitz [247], chapitre 11). De plus, Roth et al. [267] effectuent une étape de normalisation finale afin de limiter l'influence de K. Nous n'implémentons cependant pas cette normalisation pour les tests finaux, car celle-ci ne permet pas de valider s'il y a ou non des clusters dans le jeu de données.

Dudoit et Fridlyand [109] développent le même type d'algorithme, divisant les données N fois en deux ensembles E et T. Les étiquettes des groupes de E sont apprises par classification supervisée, utilisée pour prédire les clusters de T. Finalement les deux partitions de T (obtenues par clustering et prédiction) sont comparées − sans réétiquetage. La différence majeure avec les méthodes précédentes se trouve dans la façon d'aggréger les similarités au cours des N boucles. Les auteurs décomposent $N = B \times B_0$, puis estiment B_0 fois la valeur médiane de la similarité entre les deux partitions de test. Ces valeurs sont ensuite moyennées, puis un test statistique final permet de renvoyer $\hat{K}$.

2.5.3 Approche retenue

Compte-tenu de la présence des entrées x_i, nous avons la possibilité d'effectuer la classification supervisée directement des x_i vers les labels des clusters ; la difficulté liée à l'apprentissage supervisé fonctionnel est ainsi contournée. S'inspirant largement des méthodes précitées, nous procédons en deux étapes : en premier lieu la stabilité des clusters est

estimée, puis on vérifie que les entrées peuvent être classées suivant les groupes trouvés. Les arguments de l'algorithme sont notés u_i pour la raison indiquée après le pseudo-code ci-dessous.

Concernant la première phase, une idée n'appelant qu'une fois l'algorithme de clustering (noté AC) consiste à

1. perturber le jeu de données U en U' comme dans Möller et Radke [237] ;
2. appliquer AC à U', obtenant la partition P' ;
3. comparer P' à P^{ref} (partition de référence, calculée une seule fois pour chaque valeur de K) par la méthode basée sur la variation d'information (VI) de Meilă [230].

Il est cependant trop difficile d'ajuster l'amplitude de la perturbation en 1) automatiquement. On choisit donc plutôt de sous-échantillonner deux fois, avant de comparer les partitions :

`Trouve_K_Et_Clusters`$(U = u_1, \ldots, u_n)$

1. Initialisations I :
   (a) `last`$P^{ref} \leftarrow \mathbf{1}^n$.
   (b) `last`$K \leftarrow 1$.
   (c) $K \leftarrow 2$.
2. Appliquer AC à U, obtenant la partition P^{ref}, servant de référence.
3. Si P^{ref} est identique à `last`P^{ref}, incrémenter K et passer à la boucle suivante.
4. Initialisations II :
   (a) `last`$K \leftarrow K$;
   (b) $\mathcal{I} \leftarrow 0$ (indice de stabilité) ;
5. Répéter N_s fois :
   (a) sous-échantillonner le jeu de données U en $U^{(1)}$ et $U^{(2)}$ de mêmes tailles ;
   (b) appliquer AC à $U^{(1)}$ et $U^{(2)}$, obtenant les partitions P_1 et P_2 ;
   (c) notant B l'intersection des indices présents dans P_1 et P_2, calculer la variation d'information entre $P_1[B]$ et $P_2[B]$ [230] ;
   (d) $\mathcal{I} \leftarrow \mathcal{I} + \frac{VI}{N_s}$;
6. Si $\mathcal{I}$ est en dessous d'un certain seuil $\tau_s \in]0, 1[$, **stop : return** `last`P^{ref}**.**
7. $\mathcal{I} \leftarrow 0$ (indice de capacité de prédiction).
8. Répéter N_c fois (validation croisée) :
   (a) diviser U en ensembles d'entraînement E et de test T (dans une proportion 70%-30% par exemple) ; les étiquettes de T dans P^{ref} sont notés $\mathcal{T}$;
   (b) appliquer AC à E, obtenant une partition P_E ;
   (c) appliquer l'algorithme de classification supervisé à E avec pour sorties les labels des clusters dans P_E, puis prédire les étiquettes des éléments de T (ensemble $\mathcal{T}'$) ;
   (d) réétiqueter $\mathcal{T}$ ou $\mathcal{T}'$ par la méthode hongroise (voir le chapitre 1) ;
   (e) $\tau \leftarrow \frac{1}{|T|} \sum_{i=1}^{|T|} \mathbf{1}_{\mathcal{T}[i]=\mathcal{T}'[i]}$ (taux de ressemblance basique) ;
   (f) $\mathcal{I} \leftarrow \mathcal{I} + \frac{\tau}{N_c}$;
9. Si $\mathcal{I}$ est en dessous d'un certain seuil $\tau_c \in]0, 1[$, **stop : return** `last`P^{ref}**.**
10. `last`$P^{ref} \leftarrow P^{ref}$.
11. Revenir en 2.

Après que cette procédure a été lancée sur les sorties y_i (avec classification selon les entrées x_i), elle est appliquée à nouveau dans chaque cluster d'entrées x_i (avec classification selon ces mêmes entrées x_i). Cela permet de tenir compte de l'exemple mentionné en début de chapitre (section 2.2). Finalement, la recherche de la partition associée à $\hat{K}$ s'effectue comme suit.

1. Appliquer `Trouve_K_Et_Clusters` aux n sorties y_i, obtenant une première partition en K_{init} clusters.
2. Pour chaque cluster C_j, appliquer `Trouve_K_Et_Clusters` aux n_j entrées x_i correspondantes.
3. Retourner la partition constituée de la concaténation des sous-partitions obtenues en 2 ; $\hat{K}$ est égal au nombre de classes de cette dernière partition.

Notons qu'il n'est pas possible de classer directement les données en $\hat{K}$ groupes. La classification supervisée est effectuée à l'aide d'un arbre de classification (éventuellement avec boosting) pour les applications physiques ; en effet, on s'attend à ce que les entrées se regroupent "à droite" et/ou "à gauche" de certaines valeurs (les lois des paramètres en entrées étant unimodales). Pour les autres applications on utilise l'algorithme des k plus proches voisins sur les entrées standardisées, afin de rester le plus général possible. k est alors estimé par validation croisée.

Remarque : l'algorithme de clustering pourrait ne pas être le même pour les entrées et les sorties, mais nous n'avons pas développé de méthode spécifique à la structure des groupes en entrée. Ainsi, la même méthode est utilisée dans les deux espaces.

Vers une définition d'un cluster

Dans l'algorithme de classification ascendante hiérarchique, un partitionnement en $m+1$ classes est obtenu par division d'un des m groupes de la partition précédente. Cette propriété assure la difficulté croissante des tests de capacité de prédiction, justifiant dans une certaine mesure l'approche utilisée. Les tests de classification supervisée sont présents pour valider les clusters dans le contexte du métamodèle : en effet si les classes ne peuvent être prédites correctement, toute cette étape s'avère inutile et même dangereuse. La stabilité doit quant à elle être vérifiée pour assurer la robustesse du clustering aux (petites) perturbations du jeu de données. Les définitions suivantes découlent naturellement des algorithmes choisis et du contexte de la thèse.

Définition 2.5.1 *Un* cluster C *est un ensemble d'éléments de* $\{(x_i, y_i)_{i=1,\dots,n}\}$ *tel que l'erreur de validation croisée effectuée par la classification supervisée binaire* $c : \mathcal{E}_E \to \{0, 1\}$ *(idéalement égale à l'indicatrice de* C*) n'excède pas un certain seuil* τ_c*.*

Définition 2.5.2 *Un cluster* C *est dit* stable *si ses éléments se retrouvent en général (au moins* $100\tau_s\%$ *des cas) dans le même groupe lors de l'application de l'algorithme de clustering sur des sous-échantillons des données initiales.*

2.5.4 Tests

L'algorithme décrit au paragraphe précédent (avec AC = clustering hiérarchique, utilisant la distance euclidienne ou ECT) est comparé uniquement à celui de Roth et al. [267], la méthode de Dudoit et Fridlyand [109] nécessitant le choix d'un test statistique, et les autres n'étant pas prévues pour classer des entrées et sorties "simultanément".

2.5.4.1 Données artificielles

Nous appliquons d'abord l'algorithme à trois jeux de données artificiels en deux dimensions (pour pouvoir visualiser les classes). Le premier comporte 6 clusters gaussiens, tandis que les deux autres contiennent des groupes de forme arbitraire.

Test 1

Le premier jeu de données utilisé pour les tests est un mélange de six variables aléatoires gaussiennes $2D$, de centres $(2, 10), (2, 5), (7, 8), (10, 5), (13, 10)$ et $(15, 5)$. La figure 2.6 montre les données échantillonnées (m points par classe). Celles-ci représentent les entrées x_i. Nous obtenons les sorties y_i correspondantes par une simple transformation : $y_i(t) = x_{i,1} \cos t + x_{i,2} \arctan t$.

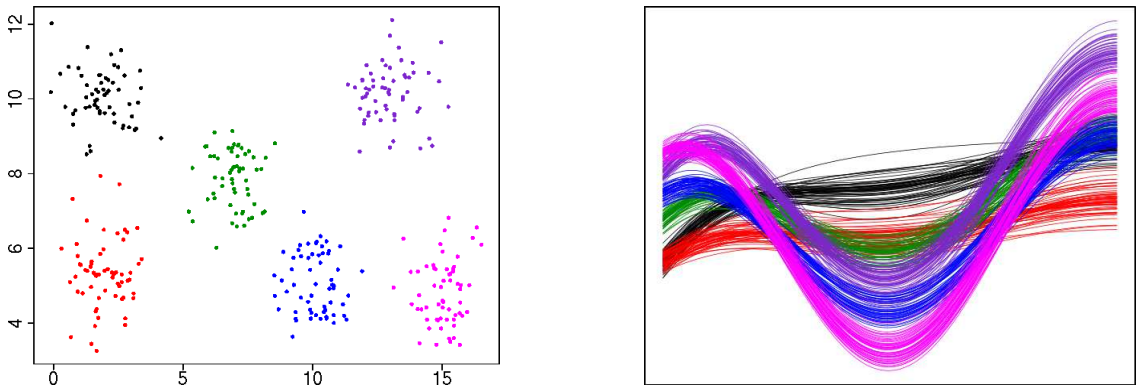

Figure 2.6: Six clusters gaussiens ($m = 60$) à gauche, les courbes correspondantes à droite.

Afin d'estimer l'influence de la taille de l'échantillon sur la stabilité (et la correction) du résultat, nous faisons varier m entre 10 et 200, avec les seuils τ_s et τ_c à 0.8 et 10 répétitions pour chaque estimation de $\mathcal{I}$ (l'algorithme étant relancé quelques dizaines de fois). La taille minimale d'un cluster est fixée à $\frac{m}{2}$, et l'ensemble de test a pour taille $0.3n$. La méthode présentée précédemment est notée SC (pour "stabilité et classification"), et celle de Roth et al. [267] RO. Le clustering hiérarchique utilisant la distance de Ward est indiqué par CH, celui utilisant la distance ECT est noté CTH. Le calcul de cette dernière distance s'effectue avec un nombre de plus proches voisins déterminé par une simple heuristique ($k = \lceil \sqrt{n} \rceil$).

Le tableau 2.1 montre les valeurs $V(s)$ obtenues, "$x|y$" signifiant que le nombre de clusters oscille entre les valeurs x et y, sans préférence marquée au fil des échantillons. Lorsqu'un nombre de clusters se démarque, la valeur est indiquée sans les autres moins probables. n désigne le nombre de points dans l'échantillon, égal à $6m$.

n	RO-CH	RO-CTH	SC-CH	SC-CTH
60	6	1	6|7	1
120	6	1	6	1|6
240	6	1|6	6	6
360	6	1|6	6	6
600	6	6	6	6
900	6	6	6	6
1200	6	6	6	6

Table 2.1: Nombre de clusters en fonction de n.

Les six groupes ne sont pas toujours découverts (bien qu'ils soient en général validés au moins quatre fois sur cinq). La figure 2.7 montre par exemple le partitionnement intermédiaire en quatre groupes obtenu par clustering hiérarchique. Les clusters colorés fusionnent lorsque deux classes seulement sont validées.

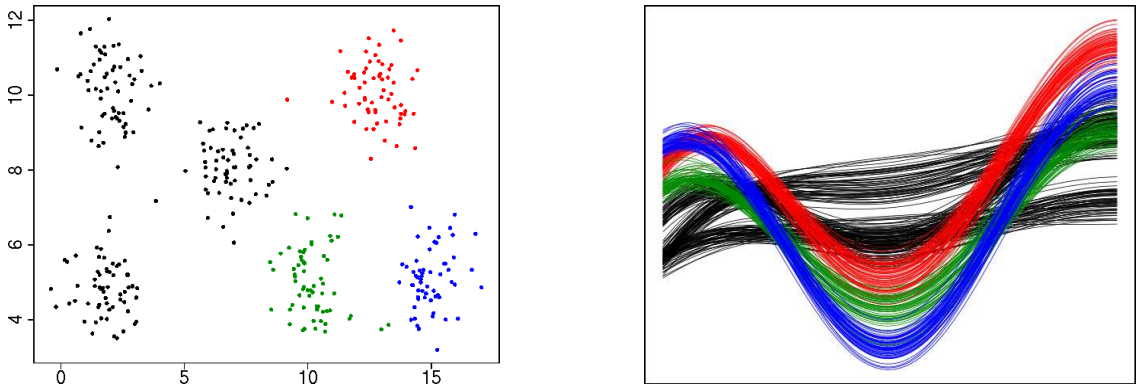

Figure 2.7: Quatre sur-clusters ($m = 60$) à gauche, les courbes correspondantes à droite.

L'instabilité observée sur les premières lignes provient du faible nombre de points par groupe. La distance ECT étant assez sensible aux perturbations, il faut un nombre de points relativement élevé ($\gtrsim 300$) pour valider plusieurs clusters. On constate que les deux méthodes produisent des résultats similaires sur cet exemple, et qu'il n'est pas nécessaire ici d'utiliser la distance ECT ; de plus l'évaluation de cette dernière est plus coûteuse.

Test 2

Le second jeu de données est construit en faisant varier les paramètres sur deux arcs de cercles, comme dans le premier test du chapitre précédent. Le bruit blanc ajouté ici a pour écart-type 0.1 seulement, car le résultat est trop difficile à visualiser sinon. Le tableau 2.2 montre les résultats obtenus, avec les mêmes conventions qu'au test du paragraphe précédent. La seule différence est le seuil, fixé ici à 0.95 pour τ_s et τ_c. Trop de groupes sont validés si on le choisit plus bas. Du point de vue du métamodèle global ce n'est pas grave, l'important étant que la régression s'effectue bien dans chaque classe. À des fins de visualisation nous préférons cependant en obtenir deux. La figure 2.8 montre l'allure des courbes dans chaque cluster.

n	RO-CH	RO-CTH	SC-CH	SC-CTH
60	1	1|2	1	1
120	1|2	2	1	2
240	1	2	1	2
360	1	2	1	2
600	1	2	1	2
900	1|2	2	1	2
1200	1	2	1	2

Table 2.2: Nombre de clusters en fonction de n.

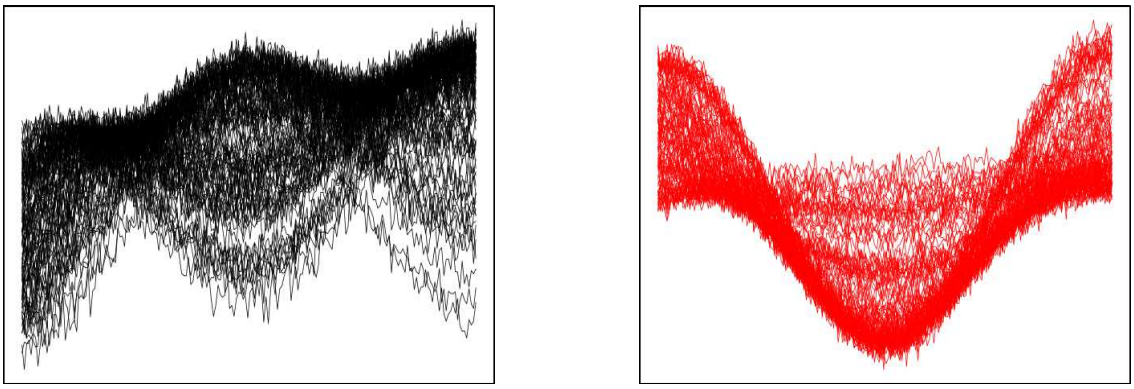

Figure 2.8: Clusters 1 et 2 du jeu de données "two moons" légèrement bruité.

Même pour plus de 1000 points, un seul groupe est découvert en utilisant la distance euclidienne (si l'on excepte les résultats étranges pour $n = 120$ et $n = 900$, difficiles à expliquer mais ne remettant pas en cause la conclusion). La distance ECT est donc très utile dans ce cas. Finalement, les deux méthodes donnent une fois encore des résultats similaires.

2.5.4.2 Données simulées (CATHARE)

Les deux jeux de données de cette section ont été créés en effectuant un certain nombre de simulation très coûteuses en temps. Ainsi nous ne pouvons pas faire varier n.

Test 1

Ce premier exemple − noté CATHARE I − comporte 100 courbes discrétisées sur 168 points, avec 4 paramètres d'entrée correspondants. En regardant la zone autour de $t = 100$ sur la figure 2.9, on s'aperçoit qu'environ une moitié des courbes décroît nettement tandis que l'autre croît tout aussi clairement (illustré par les flèches rouges sur l'autre partie de la figure). Cela semble déterminer deux groupes, comme le test suivant le confirmera.

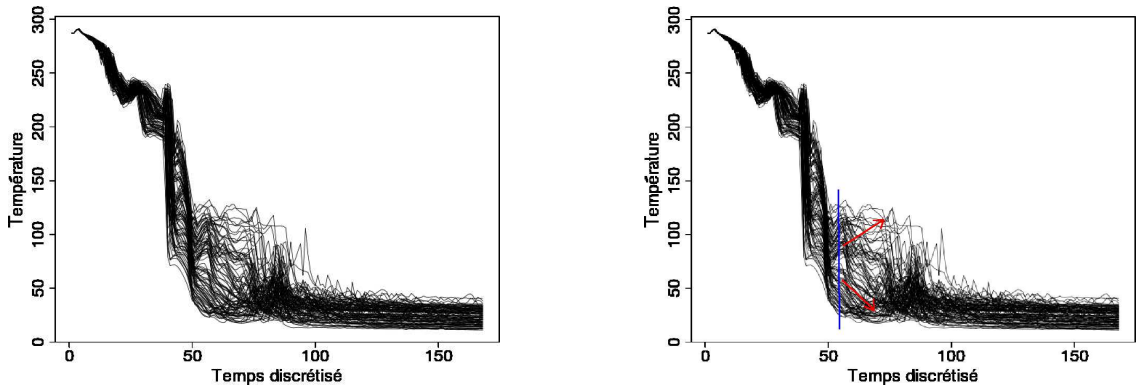

Figure 2.9: Les 100 courbes du jeu de données CATHARE I.

Le tableau 2.3 présente les résultats obtenus pour des seuils variant de 0.65 à 0.95 comme précédemment, la taille minimale d'un cluster étant arbitrairement fixée à 10. En supposant que le nombre de clusters visé est deux, les valeurs obtenues ne concordent pas vraiment.

Seuil	RO-CH	RO-CTH	SC-CH	SC-CTH
0.65	4|6	6	6	6
0.7	5	5	5	5|6
0.75	4|5	1	5	1|5
0.8	4	1	4|5	1
0.85	1	1	3|4	1
0.9	1	1	1	1
0.95	1	1	1	1

Table 2.3: Nombre de clusters en fonction du seuil.

Il suffit cependant d'imposer aux groupes de comporter au moins 30 éléments pour avoir les résultats visibles dans le tableau 2.4, légèrement en faveur de la méthode introduite au paragraphe précédent (SC).

Seuil	RO-CH	RO-CTH	SC-CH	SC-CTH
0.65	2	1	2	2
0.7	2	1	2	2
0.75	1	1	2	2
0.8	1	1	2	1
0.85	1	1	1	1
0.9	1	1	1	1
0.95	1	1	1	1

Table 2.4: Nombre de clusters en fonction du seuil.

La méthode SC est assez nettement plus performante dans ce cas, validant deux groupes jusqu'à un seuil de 0.8, alors que celui-ci doit être inférieur à 0.7 pour les découvrir avec RO. Notons toutefois que la taille minimale d'un groupe indiquée joue beaucoup sur le résultat final. En pratique nous imposons un cardinal d'au moins 30 (voire 50) pour chaque classe, afin que la régression ultérieure se déroule bien. Les trois à six groupes validés ci-dessus ne se trouveront donc pas dans le modèle final. La figure 2.10 montre les deux clusters indiqués

au début, dans l'espace fonctionnel à droite, et les composantes 3 et 4 des entrées correspondantes à gauche. Comme on peut le constater, les points après cette projection sont presque linéairement séparables.

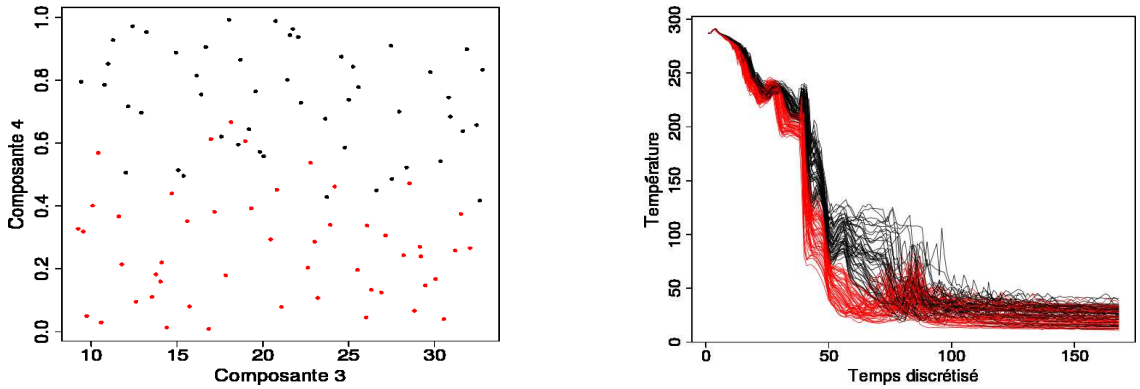

Figure 2.10: Composantes 3 et 4 des entrées à gauche, courbes correspondantes en sortie à droite.

On remarque que l'utilisation du clustering hiérarchique avec la distance de Ward entraîne la mauvaise classification d'une courbe (ajoutée au groupe du bas). Ainsi la distance ECT permet d'obtenir de meilleures partitions, au prix d'un temps d'exécution plus lent et d'une plus grande instabilité empêchant parfois la validation du bon nombre de clusters. En pratique il est intéressant d'essayer les deux méthodes.

Test 2

Ce second exemple – noté CATHARE II – comporte 600 courbes discrétisées sur 414 points, avec 11 paramètres d'entrée correspondants. Une analyse visuelle ne permet pas de mettre en évidence des groupes de courbes distincts, aussi nous laissons l'algorithme choisir.

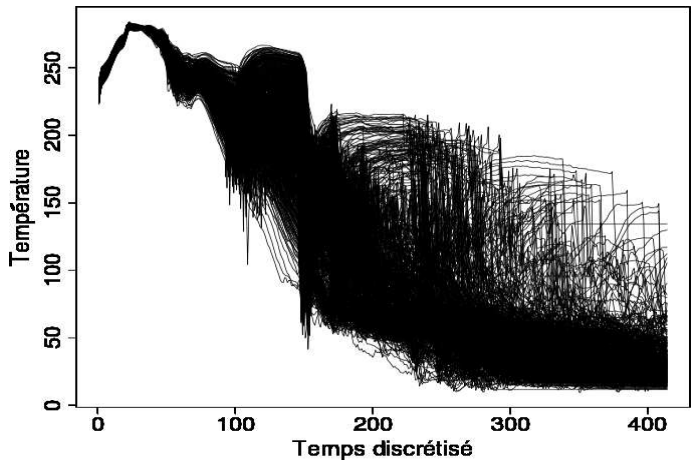

Figure 2.11: Les 600 courbes du jeu de données CATHARE II.

Le tableau 2.5 présente les résultats obtenus pour des seuils variant de 0.65 à 0.95 comme précédemment, la taille minimale d'un cluster étant arbitrairement fixée à 30.

Seuil	RO-CH	RO-CTH	ME-CH	ME-CTH
0.65	1	9	1	9
0.7	1	3|4	1	6|7
0.75	1	3	1	3|4
0.8	1	3	1	3|4
0.85	1	3	1	2
0.9	1	1	1	1
0.95	1	1	1	1

Table 2.5: Nombre de clusters en fonction du seuil.

Quelques clusters sont validés à des seuils relativement élevés en utilisant la distance ECT, mettant ainsi l'intuition visuelle (aucun groupe) en défaut. Quatre groupes trouvés sont présentés sur la figure 2.12, trois puis deux s'obtenant en fusionnant d'abord le cluster en rouge avec celui en noir, et enfin le bleu avec le vert. Les clusters rouge et noir sont très similaires, et ne diffèrent en fait que dans la forme des courbes aux alentours de l'abscisse 120 ; ces derniers ont été déterminés par classification sur les entrées.

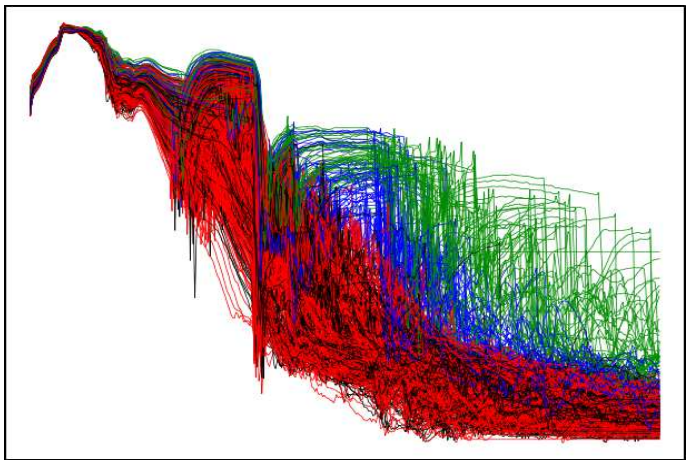

Figure 2.12: Quatre clusters des courbes CATHARE II.

Il n'est pas évident de départager les méthodes RO et SC sur ce dernier exemple. Cependant, la seconde est globalement au moins aussi performante que la première au travers les tests. Nous retenons $\tau_s = 0.8$ et $\tau_c = 0.8$ par défaut dans le package R, cette valeur étant suffisamment conservative pour ne pas surestimer le nombre de classes, tout en permettant d'en découvrir quelques-unes si le jeu de données en recèle.

2.6 Conclusion

Après avoir passé en revue un certain nombre d'algorithmes de clustering, nous avons choisi d'abandonner certaines spécificités du cadre fonctionnel pour effectuer la classification dans un graphe. Celle-ci utilise la distance ECT définie au chapitre précédent afin de renforcer l'information contenue dans les (dis)similarités. Le clustering hiérarchique fournit alors un cadre assez naturel de détermination du nombre de groupes comme indiqué au paragraphe 2.5.3. Il en découle une nouvelle définition d'un cluster, en accord avec nos objectifs.

2.7 Perspectives

Le graphe est un outil assez puissant pour la classification non supervisée, mais en l'utilisant la nature fonctionnelle des données est "oubliée". Il faudrait alors chercher un compromis entre les méthodes de clustering fonctionnelles et celles basées sur un graphe, par exemple en utilisant une (dis)similarité tenant compte de l'adéquation des formes de deux fonctions ainsi que de leur distance, comme suggéré via une (dis)similarité vectorielle à la section 1.5.

Concernant l'étape de clustering (en entrée ou sortie), une possibilité serait d'écarter les points isolés entre les groupes en prétraitement de l'algorithme, un peu dans l'esprit de la procédure de Harel et Koren [154]. Il y a cependant un problème d'échelle à résoudre − similaire à celui indiqué en 3.1.2 − afin d'éloigner les classes et d'isoler le bruit, et éventuellement pour calculer les distances ECT en limitant/annulant les effets des sommets trop éloignés.

Chapitre 3

Représentation discrète d'une variété

Comme il a été signalé en introduction, nous cherchons à représenter les courbes en sortie du code de calcul par des vecteurs de $\mathbb{R}^d$, avec d le plus petit possible. Nous faisons l'hypothèse que les courbes se trouvent sur une variété fonctionnelle riemannienne (voir la section A.1 pour un rappel de la définition). Le cas particulier d'un sous-espace vectoriel de $\mathcal{C}([a,b])$ est ainsi pris en compte. Avant toutes choses il est nécessaire d'estimer la dimension de cette variété (en préférant un majorant à un minorant), pour connaître le nombre de coefficients à apprendre ultérieurement. Dans une seconde partie nous présentons quelques familles d'algorithmes de réduction de dimension, concluant le chapitre avec les méthodes utilisées dans la thèse.

Précisons les motivations derrière ces deux étapes en rappelant l'écriture du métamodèle.

$$\hat{\phi}(x,t) = R(g_1(x), \dots, g_d(x)) \,,$$

où les g_j sont des fonctions de régression $1D$ apprenant les composantes représentant la sortie y au sein d'un cluster (non indiqué ici). R est une fonction de reconstruction des coordonnées réduites vers l'espace fonctionnel $\mathcal{C}([a,b])$, déterminée par la façon dont la dimension est réduite. L'estimation de R est d'autant plus précise que son nombre de paramètres est réduit. De plus, quand d n'est pas trop grand le modèle peut être visualisé par le biais des composantes g_j. Par exemple, quand d vaut deux on obtient une correspondance des entrées vers une "carte de fonctions". Pour ces deux raisons, le meilleur compromis entre un modèle simple (d petit) et précis (d plus élevé) est recherché. En faisant l'hypothèse que les sorties du codes ont une structure de variété, cela revient à estimer sa dimension (plus petit entier permettant de la décrire "raisonnablement") avant de chercher une fonction de représentation r, telle que $g_{(j)}(x) \simeq r_{(j)}(y)$.

Avant de poursuivre, rappelons que toute variété riemannienne est isométrique à une sous-variété d'un espace euclidien $\mathbb{R}^m$ (Nash [242] ; voir aussi Han et Hong [152]). Cela permet d'identifier $\mathcal{C}([a,b])$ à $\mathbb{R}^D$ par léger abus d'écriture : en effet la valeur minimale de m pourrait être plus grande que D pour que $\mathbb{R}^m$ "contienne" $\mathcal{U}$ (non discrétisée). Les produits scalaires $\langle f, h\rangle = \int f(t)h(t)dt$ sont donc identifiés aux produits scalaires euclidiens dans $\mathbb{R}^D$, et les développements algorithmiques de ce chapitre sont effectués quasi-systématiquement avec $\mathbb{R}^D$.

3.1 Estimation de la dimension

Rappelons tout d'abord la définition de la dimension d'une variété.

Définition 3.1.1 *Une variété V est dite de dimension $d < \infty$ si tout élément de V possède un voisinage homéomorphe à un ouvert de $\mathbb{R}^d$.*

Après avoir indiqué d'autres définitions intéressantes de la dimension, nous présentons quelques techniques existantes avant d'introduire celle retenue pour le métamodèle. La variété à estimer est supposée différentiable, munie de la métrique induite par le produit scalaire $\langle f, h\rangle = \int fh$ dans $\mathcal{C}([a, b])$ et bornée, éventuellement à bords. Cette dernière hypothèse n'est pas restrictive car il existe toujours une variété bornée contenant un nombre fini de points donnés. La différentiabilité permet de justifier l'approximation des distances géodésiques par les plus courts chemins dans un graphe (expliquée au paragraphe A.1.5). Il existe en fait des variétés topologiques n'admettant aucune structure différentielle [182], mais un nombre fini de points peut toujours être interpolé par une variété lisse.

Les paragraphes 3.1.1 et 3.1.2 sont bibliographiques, introduisant respectivement une vision alternative de la dimension et des méthodes d'estimation de celle-ci. Deux nouveaux algorithmes sont présentés aux paragraphes 3.1.3 et 3.1.4.

Plan de cette première partie :

1. **Définitions alternatives de la dimension.**
2. **État de l'art de l'estimation de la dimension.**
3. **Généralisation de la méthode de Lin et al. [213].**
4. **Approche basée sur les volumes des simplexes.**

3.1.1 Définitions alternatives

Il existe deux définitions alternatives majeures de la dimension d'une variété. La première s'énonce comme suit.

Définition 3.1.2 *La* dimension de recouvrement *(ou* dimension topologique*) d'un espace topologique normal E est égale au plus petit entier d vérifiant la propriété suivante :*
tout recouvrement ouvert $\mathcal{R}$ possède un sous-recouvrement $s\mathcal{R}$ tel qu'aucun point de E ne soit dans plus de $d + 1$ ouverts de $s\mathcal{R}$.

Elle possède l'avantage d'être définie pour des espaces topologiques n'ayant pas forcément une structure de variété ; il est toutefois nécessaire que deux fermés disjoints ne soient pas "infiniment proches" pour que l'espace E soit dit *normal*. Par exemple, $EP \cap -EP$ avec EP épigraphe de la fonction $]0, +\infty[\to]0, +\infty[, x \mapsto \frac{1}{x}$ ne satisfait pas les conditions de la définition. Pour une variété normale on peut montrer que la dimension de recouvrement est égale à la dimension définie en début de chapitre (voir Hurewicz et Wallman [167], exemple III.4 et théorème IV.2). Cette définition n'est pas utilisée directement mais donne une autre

vision de la dimension d'une variété.

La seconde dimension alternative peut prendre des valeurs non entières, et est de ce fait surtout adaptée aux objets fractals, potentiellement plus complexes que ceux auxquels nous nous intéressons. Elle a cependant été appliquée à des exemples réels, avec pour argument que ceux-ci peuvent comporter du bruit et donc ne s'alignent pas exactement sur une variété. Il s'agit en fait d'une famille de mesures de dimensions (Lee et Verleysen [201], chapitre 3), toutes obtenues à partir de la définition suivante.

$$C_q(\mu, \epsilon) = \int [\mu(\bar{B}_\epsilon(y))]^{q-1} d\mu(y)\,,$$

μ étant une mesure de probabilité borélienne à support dans la variété, et $\bar{B}_\epsilon(y)$ la boule fermée de rayon ϵ centrée en y. Les $q-$dimensions supérieure et inférieure de μ sont alors respectivement définies comme suit :

$$D_q^-(\mu) = \liminf_{\epsilon \to 0} \frac{\log C_q(\mu, \epsilon)}{(q-1)\log \epsilon}\,,$$

$$D_q^+(\mu) = \limsup_{\epsilon \to 0} \frac{\log C_q(\mu, \epsilon)}{(q-1)\log \epsilon}\,.$$

Lorsque les deux limites sont égales (ce qui est toujours le cas pour une variété différentielle), on appelle la fonction $q \mapsto D_q(\mu)$ le spectre dimensionnel de μ.

Remarque : la dimension de corrélation (la plus utilisée, correspondant à $q = 2$) ne peut être estimée avec précision que si elle est inférieure à $2 \log_{10} n$ [295; 111]. Une inégalité du même ordre de grandeur s'applique pour les autres familles de méthodes, reléguées à l'estimation des faibles dimensions. Cela n'est guère étonnant : l'espace étant de plus en plus vide à mesure que la dimension augmente [271], il faut un nombre exponentiel de points pour estimer la dimension localement, et ce quelle que soit l'approche envisagée.

Enfin, notons qu'il peut sembler plus simple de définir la dimension comme le nombre minimal de paramètres nécessaires pour décrire une variété. En plus d'être difficile à appliquer telle quelle cette "définition" indique que la courbe (fractale) de Peano [248] est de dimension 1 puisque paramétrée par $t \in [0, 1]$. Or celle-ci est surjective sur le carré unité, donc de dimension topologique 2 (considérant la définition ci-dessus avec les ouverts induits par la distance euclidienne dans $\mathbb{R}^2$; en considérant les ouverts induits par ceux de $[0, 1]$ la dimension est 1, mais cette structure présente peu d'intérêt, en plus de nécessiter la connaissance du paramétrage). Il existe plusieurs autres exemples de ce type, notamment les courbes de Hilbert, Sierpinski, Moore et Lebesgue (voir le livre de Sagan [273]) ; cette dernière est même différentiable presque partout. Plus généralement, le théorème de Hahn-Mazurkiewicz (Wilder [340], chapitre 3) indique que toute variété compacte et localement connexe est image d'un chemin continu à support dans $[0, 1]$. Cela rend inapplicable la définition par le nombre de paramètres. Il faut alors imposer des contraintes sur l'application de carte (bijective continue de réciproque continue), éliminant la possibilité d'une space-filling curve (ou autre construction du même type), et on en revient à la définition classique de la

dimension.

3.1.2 Approches existantes

Considérant la définition initiale de la dimension d'une variété (via les homéomorphismes), l'idée la plus naturelle consiste à évaluer les dimensions des espaces tangents en effectuant des analyses en composantes principales (ACP) locales. Dans un second temps nous présentons des méthodes basées sur l'évaluation de la densité, et finalement quelques approches géométriques. La principale difficulté dans l'estimation de la dimension réside dans le choix de l'échelle à laquelle on observe les données, comme le montre la figure 3.1 empruntée à Balázs Kégl. Cette dernière figure suggère d'estimer la dimension en regardant comment le nombre de points évolue dans un voisinage ; c'est ce que propose Brand [42], en cherchant le maximum de la fonction $\epsilon \mapsto \frac{d}{d \log n(\epsilon)} \log \epsilon$, $n(\epsilon)$ désignant le nombre de points dans une boule de rayon ϵ. Les méthodes estimant la dimension fractale ne contournent pas le problème de l'échelle, qui reste à l'appréciation de l'utilisateur. Elles ne sont donc pas présentées ici, le lecteur intéressé pouvant consulter les articles de Camastra et Vinciarelli [53], Kégl [179] et Hein et Audibert [163]. Notons que ces derniers proposent une heuristique de détermination automatique de l'échelle minimale d'observation des données. Celle-ci est cependant fixée pour toute la variété. La plupart des autres approches s'adaptent à la topologie des points en demandant d'entrer le nombre de voisins par élément ; en général ceci constitue un bon compromis.

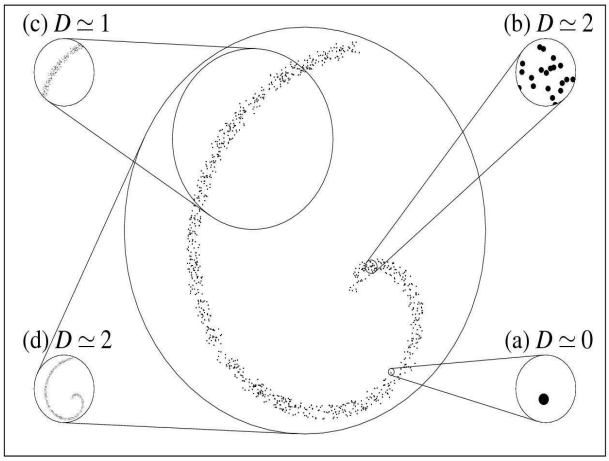

Figure 3.1: Différentes échelles mènent à différentes estimations de d.

À moins de connaitre ou d'estimer le modèle ayant généré les données (ceci permettant de "débruiter" la variété en éliminant la composante liée au bruit, ou au moins de pondérer les tirages par leurs probabilités d'appartenance à la variété), il est impossible de dire si l'on est en présence d'une variété de dimension $d' > d$ très aplatie, ou si la dimension est réellement d. C'est pourquoi nous supposons l'échantillonnage exempt de bruit, et testons la robustesse des méthodes ensuite. Il existe des approches géométriques visant à construire un modèle des données [11; 127; 64; 17; 35], mais elles nécessitent une tolérance sur l'erreur réalisée ou supposent simplement l'échantillon non bruité.

On pourrait aussi choisir d'estimer la dimension globale des données en approximant toute la variété par un espace vectoriel [46; 38]. Cela peut nettement surestimer la vraie

dimension dans certains cas, mais être adapté à d'autres. Nous préférons rechercher ici la dimension intrinsèque de la variété.

3.1.2.1 Méthodes basées sur l'ACP

En considérant les courbes y_i comme des réalisations d'un processus stochastique, Mas [225] étudie le comportement asymptotique de l'estimateur empirique de l'opérateur d'autocovariance local. Les fonctions propres de ce dernier constituent les directions des dérivées (fonctionnelles) en une courbe de la variété. Localement en $y_i \in \mathcal{U}$, ces dérivées s'approximent naturellement par $y_j - y_i$ où les y_j sont les fonctions voisines de y_i dans l'ensemble d'apprentissage (au sens de la norme L^2). La démonstration topologique est effectuée au paragraphe A.1.3. Toute la difficulté réside alors dans le choix du nombre de voisins k_i en y_i : celui-ci dépend de la courbure locale (définie au paragraphe A.2.4), qui est tout aussi difficile à estimer que la dimension. Sauf mention contraire k_i est alors supposé constant, fixé à environ $\sqrt{n}$ (les heuristiques du paragraphe 1.1.3 peuvent s'appliquer, mais ne sont pas mentionnées ici pour plus de clarté).

Nous anticipons légèrement sur la suite en utilisant alors l'ACP fonctionnelle (présentée en 3.2.1.1), permettant d'approximer l'espace tangent. Le calcul de celle-ci se base sur un théorème de décomposition matricielle sur les fonctions $y_j - y_i$ discrétisées, rangées en lignes dans la matrice A : $A = U\Sigma {}^t V$ avec U, V orthonormales et Σ "diagonale". La base locale (discrétisée) est estimée par les premières colonnes de la matrice V, les éléments diagonaux de Σ (valeurs singulières) étant notés $\sigma_1 \geq \cdots \geq \sigma_\ell \geq \ldots$. Voir le livre de Ramsay et Silverman [258] pour plus de détails concernant l'ACP fonctionnelle.

Fukunaga et Olsen [132] effectuent une ACP en chaque "région locale", puis réalisent la moyenne des résultats obtenus pour retourner $\hat{d}$. La technique d'estimation locale centrée en $u \in V$ est la suivante :

1. former la matrice A dont la j^{eme} ligne contient $v_j - u$, où $v_1, \ldots, v_k$ sont les k plus proches voisins de u ;
2. retourner $\hat{d}_u = \arg\max\{j = 1, \ldots, k / \frac{\sigma_j}{\sigma_1} \geq \tau\}$ (τ étant un seuil utilisateur).

Le paramètre τ est très difficile à régler ; en pratique on peut choisir des valeurs entre 0.01 et 0.05, avec plus ou moins de succès. Les centres des régions locales sont déterminés de façon à être séparés d'une distance d'au moins R_s (second paramètre d'estimation ardue).

Bruske et Sommer [51] effectuent une classification non supervisée des données via l'algorithme des k-means, puis appliquent la même méthode avec les centres des classes au lieu d'utiliser une heuristique sur les distances. Cela permet d'avoir des régions locales plus linéaires, et donc des valeurs singulières mieux séparées, bien que les problèmes de réglage des paramètres demeurent. Notons qu'il s'agit d'une des rares méthodes s'adaptant à la courbure de la variété (supposant le nombre de groupes choisi de manière optimale).

Fan et al. [116] appliquent la même méthodologie en approximant un recouvrement des données (via les voisinages du type k plus proches voisins) de taille minimale. Le problème du choix du nombre de clusters est à nouveau reporté dans le choix du nombre de voisins, mais l'heuristique est meilleure que celle proposée par Fukunaga et Olsen [132].

Broomhead et al. [47] remarquent que les valeurs singulières croissent linéairement avec la taille d'un voisinage si et seulement si elles correspondent à des dimensions effectives de la variété. Dans le cas contraire, la dépendance est au moins quadratique. Ils proposent alors de calculer les valeurs singulières σ_ϵ de la matrice des données dans un voisinage de taille ϵ pour plusieurs valeurs croissantes de ce dernier paramètre, et d'estimer la dimension par le nombre de dépendances linéaires observées. La figure 3.2 montre un comportement typique pour une variété à deux dimensions dans $\mathbb{R}^4$ (l'échelle en ordonnée variant d'une courbe à l'autre). Hundley et Kirby [166] améliorent la détection de la linéarité en effectuant une étape de normalisation préliminaire, remplaçant les données par les premières colonnes de la matrice U dans la décomposition en valeurs singulières.

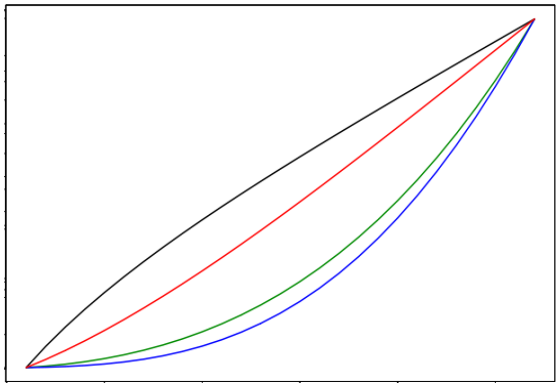

Figure 3.2: Allure des courbes de valeurs singulières ; les deux premières en noir et rouge, les deux suivantes en vert et bleu.

3.1.2.2 Méthodes basées sur la densité

Toutes les approches de ce paragraphe se basent sur la relation suivante

$$V_{k,d} f(x) \simeq \frac{k}{n},$$

où $V_{k,d}$ est le volume de la boule de rayon ϵ_k en dimension d, ϵ_k étant la distance de x au k^{eme} plus proche voisin parmi les x_i. f désigne la densité de probabilité des données sur la variété. Lorsque d augmente, un nombre exponentiellement croissant de points doit être présent au voisinage de x, comme mentionné au paragraphe 3.1.1.

Approches locales

Trunk [317] fut un des premiers à chercher à estimer la dimension d'une variété, par le biais de tests statistiques. En effet dans un voisinage suffisamment petit sur la variété les données "ressemblent" à $\mathbb{R}^d$, et donc la densité de ces dernières doit être approximativement uniforme dans l'espace tangent. On teste alors l'hypothèse "les données sont réparties uniformément dans $\mathbb{R}^d$", avec d croissant.

Levina et Bickel [207] considèrent la densité des données quasi constante sur la sphère $S(u, \epsilon)$ centrée en $u \in V$ de rayon ϵ suffisamment petit. Ils supposent alors que le nombre de points échantillonnées à l'intérieur de la sphère suit un processus de Poisson. En reliant

le taux de ce processus à la dimension des données on obtient l'estimateur local suivant :

$$\hat{d} = \left(\frac{1}{k-1} \sum_{j=1}^{k-1} \log \frac{\epsilon_k}{r_j} \right)^{-1} ,$$

qui est ensuite moyenné sur toutes les données, voire sur plusieurs valeurs de k. En pratique k est fixé à une faible valeur, et un assez fort biais peut être observé. MacKay et Ghahramani [219] proposent de corriger ce biais en effectuant la moyenne des inverses des dimensions locales, sous l'hypothèse très discutable de l'indépendance locale des données. Gupta et Huang [149] régularisent l'estimateur de Levina et Bickel via la divergence de Kullback-leibler [193], permettant ainsi de réduire la variance de l'estimateur. Farahmand et al. [118] déterminent les vitesses de convergence pour un estimateur simplifié, supposant toujours la densité des données quasi constante dans un voisinage.

Approches globales

Pettis et al. [253] obtiennent l'expression suivante à partir de la loi de probabilité f_k de la distance d'un point au k^{eme} voisin.

$$\log G_{k,d} + \log \bar{\epsilon}_k = \frac{1}{d} \log k + C(n) .$$

$C(n)$ est une quantité indépendante de k, $\bar{\epsilon}_k$ désigne la moyenne empirique des distances au k^{eme} voisin et $G_{k,d}$ est un terme provenant de l'expression de f_k. Un algorithme itératif est alors implémenté : on initialise $G_{k,d}$ à zéro avant d'estimer d via la pente de la droite de régression linéaire, puis on évalue $G_{k,d}$ et on reboucle.

Raginski et Lazebnik [257] déterminent la dimension en examinant l'erreur L^q − *q-distorsion* − commise par un jeu de centroïdes (déterminés par exemple via l'algorithme des k-means) approximant les données. En effet, en supposant la densité de répartition des éléments sur la variété suffisamment régulière la distorsion $W_{n,K}$ est proportionnelle à $K^{-\frac{1}{d}}$, où K est le nombre de clusters, supposé "grand". Cela suggère d'estimer d via la limite suivante.

$$\hat{d} = - \lim_{K \to \infty} \frac{\log K}{W_{n,K}} ,$$

$W_{n,K}$ étant approchée par son équivalent empirique.

3.1.2.3 Méthodes géométriques

Ces approches s'appuient sur l'élément géométrique de base suivant.

Définition 3.1.3 *Un* (m-)simplexe *est l'enveloppe convexe d'un ensemble de $m+1$ points utilisés pour former un repère affine dans $\mathbb{R}^m$ (et, par extension, dans un espace vectoriel quelconque).*

On identifie par léger abus de langage un simplexe aux $m+1$ points qui le définissent. Ainsi un simplexe de dimension deux (resp. trois) est un triangle (resp. tétraèdre). Plus

généralement il s'agit du polyèdre de volume non nul le plus simple, dans le sens que le nombre de points nécessaires à sa description est minimal. Des m-simplexes peuvent être formés dans un espace euclidien uniquement si m est inférieur strict à la dimension de l'espace. Les deux approches présentées ici font le raisonnement inverse : si l'on ne peut pas trouver de simplexes à $m+1$ points, alors la dimension est inférieure stricte à m.

Lin et al. [213] construisent le graphe de visibilité comme indiqué au paragraphe 1.1.3.1: on supprime les arêtes des sommets "éclipsés" dans le graphe des k plus proches voisins. Cela permet de supprimer – dans une certaine mesure ; voir le paragraphe suivant – les "slivers" (simplexes de volume négligeable), et la dimension est alors naturellement estimée via le nombre de sommets du plus grand simplexe trouvé. La figure 3.3 illustre le graphe obtenu pour un swissroll comportant cinquante points ; aucun sous-graphe complet (clique) d'ordre quatre ne se forme, conformément aux attentes.

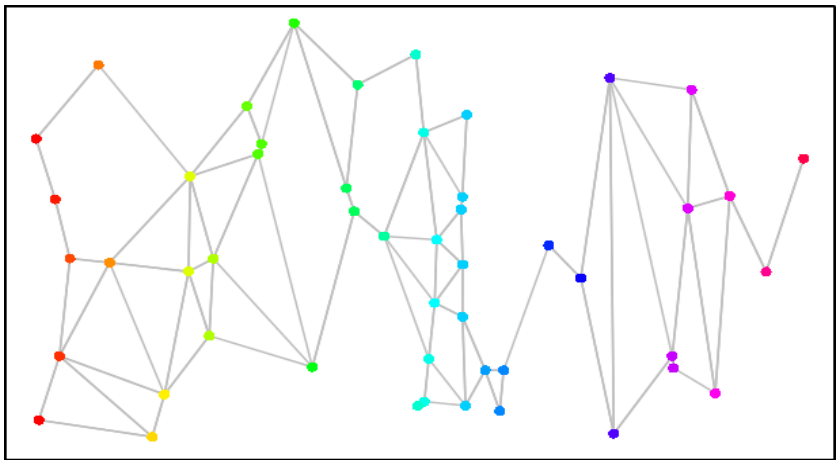

Figure 3.3: Graphe de visiblité du swissroll à 50 points.

Cette dernière méthode pose toutefois un problème de taille : la recherche exhaustive de tous les simplexes d'un graphe a un coût exponentiel. En effet en notant k le cardinal moyen d'un voisinage il y a $O(kn)$ 1-simplexes (segments), qui forment chacun potentiellement $O(k^2n)$ 2-simplexes avec leurs voisins communs. Par récurrence immédiate on s'aperçoit qu'il y a de l'ordre de $O(k^dn)$ d-simplexes au total, chacun devant être stocké ou retrouvé à chaque étape (modulo les possibles astuces d'implémentation mineures). Pour $k=8$, $d=10$ et $n=100$ cela représente plus de cent milliards de simplexes à stocker/trouver ; il n'est donc clairement pas envisageable d'appliquer la méthode telle quelle pour des dimensions moyennes. C'est pourquoi nous proposons un algorithme du type Monte-Carlo pour contourner cette difficulté au paragraphe 3.1.3.

La seconde approche de ce paragraphe est basée sur l'observation suivante : un ensemble de $d+1$ points de $\mathbb{R}^D$ $(D>d)$ interconnectés dans un sous-espace vectoriel de dimension $d-1$ est de volume nul en temps qu'objet de dimension d. Par exemple, un quadrilatère dont les sommets diagonalement opposés sont connectés (la projection d'un tétraèdre) dans $\mathbb{R}^2$ a un volume nul dans $\mathbb{R}^3$. En revanche, il est hautement probable que $d+1$ points pris au hasard dans $\mathbb{R}^d$ forment un simplexe de volume non négligeable. Si l'on dispose de suffisamment de points dans un voisinage de la variété pour que celui-ci "ressemble" assez à $\mathbb{R}^d$, alors on voit qu'il suffit d'augmenter m jusqu'à ne plus pouvoir former de m-simplexes

de dimension non négligeable. d peut ainsi être estimé via la dernière valeur de m.

Le cœur de l'algorithme de Cheng et Chiu [62] consiste en le choix d'un certain nombre de points au hasard dans un voisinage, puis en la recherche de simplexes de volume négligeable (appelés "slivers" [63]) parmi ces points ; si on en trouve, alors un compteur est incrémenté. En fin de boucle, si trop de slivers ont été formés la dimension courante est retournée. Les seuils déterminant si un volume est négligeable ou non sont assez arbitraires, ce qui rend l'application de la méthode difficile malgré des résultats théoriques de convergence (supposant les constantes bien choisies).

Les plus robustes aux perturbations et au mauvais échantillonnage sont les méthodes basées sur l'ACP, mais ce sont aussi celles qui donnent les moins bons résultats. Il faut donc chercher un compromis, qui nous semble trouvé dans ces méthodes géométriques ne se basant pas sur une estimation de la densité (contrairement aux approches du paragraphe précédent, et, indirectement, à celles basées sur les fractales). Deux variantes sont proposées dans les paragraphes qui suivent.

3.1.3 Régularisation d'une méthode géométrique

La condition imposée dans le graphe de visibilité impose aux angles internes des simplexes d'être tous aigus. En effet si un angle $\widehat{uu'u''}$ est fortement obtus, alors les vecteurs $\overrightarrow{uu'}$ et $\overrightarrow{u'u''}$ sont quasi colinéaires, et le déterminant intervenant dans une première formulation du volume d'un simplexe [48] est nécessairement proche de zéro :

$$V_m(u_1, \ldots, u_{m+1}) = \frac{1}{m!} \begin{vmatrix} u_2 - u_1 \\ \cdots \\ u_{m+1} - u_1 \end{vmatrix}.$$

La ligne $u_j - u_1$ dans le déterminant est la représentation m-dimensionnelle de cet élément (éventuellement via une ACP sur les $m+1$ points). Il subsiste néanmoins une possibilité pour qu'un tel volume soit négligeable. En effet il suffit qu'un des côtés soit de longueur presque nulle pour qu'intuitivement le volume le soit aussi. Nous vérifions cette intuition en écrivant la relation liant le volume d'un simplexe formé sur les points $u_1, \ldots, u_{d+1} \in \mathbb{R}^D$:

$$V_m(u_1, \ldots, u_{m+1}) = \frac{1}{2^{m/2} m!} \begin{vmatrix} 0 & \delta_{1,2}^2 & \delta_{1,3}^2 & \ldots & \delta_{1,n+1}^2 & 1 \\ \delta_{2,1} & 0 & \delta_{2,3} & \ldots & \delta_{2,n+1}^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \delta_{n+1,1} & \delta_{n+1,2} & \delta_{n+1,3} & \ldots & 0 & 1 \\ 1 & 1 & 1 & \ldots & 1 & 0 \end{vmatrix},$$

où $\delta_{ij}^2 = \|u_i - u_j\|^2$ (voir les livres de Sommerville [298], chapitre VIII et Blumenthal [34], chapitre IV). En effet si deux points du simplexes sont très proches, leurs distances à tous les autres points sont similaires et ils sont ainsi quasi colinéaires. Cette situation est toutefois plutôt improbable si la variété est échantillonnée assez uniformément.

De cette analyse succinte nous retenons qu'il existe tout un intervalle d'angles (grossièrement, de $\frac{\pi}{2}$ à $\frac{2\pi}{3}$, soit des cosinus variant de 0 à -0.5) interdits dans le graphe de visibilité et pouvant potentiellement composer des simplexes de volume non négligeable. Par exemple,

tous les angles locaux du réseau hexagonal (figure 3.4) sont égaux à $\frac{2\pi}{3}$, n'autorisant ainsi pour voisins immédiats que les trois sommets les plus proches et les trois diagonalement opposés. Cela ne permet pas la construction d'un triangle, et la dimension estimée vaut alors 1. Cette estimation ne change pas si l'on perturbe légèrement le jeu de données. La méthode initiale ne possède donc pas de garanties théoriques, bien qu'elle peut donner de bons résultats en pratique. Nous proposons ici une version probabiliste et régularisée, permettant de s'affranchir de la complexité exponentielle et apportant plus de souplesse dans le choix des simplexes ; elle n'offre cependant pas plus de garanties.

Figure 3.4: Pavage hexagonal ; aucun triangle ne se forme dans le graphe de visibilité.

On part du constat suivant : tout m-simplexe ($m > 0$) est réunion disjointe de deux simplexes non vides, et réciproquement l'union d'un m'-simplexe et d'un m''-simplexe disjoints (en connectant tous les sommets) constitue un m-simplexe avec $m = m' + m''$. Cela découle de la définition pour ce dernier point, et il suffit d'isoler un sommet pour vérifier le premier. Une version modifiée de l'algorithme du clustering hiérarchique est alors utilisée pour que les fusions s'effectuent aléatoirement entre simplexes complètement interconnectés. Plus précisément, les fusions sont biaisées pour s'effectuer entre simplexes proches, et on s'arrête lorsque plus aucun regroupement n'est possible. Ceci est effectué par la routine `rand-hclust` dans le programme suivant.

1. Initialiser *simps* de taille n à $(0, 0, ..., 0)$.
2. Pour t variant de 0 à 0.9 :
   (a) construire le graphe de voisinage avec une tolérance t sur les angles (cosinus $\geq -t$) ;
   (b) en déduire la matrice des "distances" D (infinies entre deux sommets non voisins);
   (c) $ps \leftarrow (1-t)^\alpha$;
   (d) répéter N fois :
   $simps \leftarrow simps + ps$. `rand-hclust`$(D)$.
3. Normaliser *simps* par N fois la somme des poids.

4. Retourner $d-1$ où d est le plus grand indice vérifiant $simps[d] \geq 1$.

Conformément aux remarques nous ne cherchons pas de preuve de convergence, celle-ci ne pouvant a priori s'obtenir que pour une certaine classe d'échantillonnages uniformes. En pratique cette nouvelle version n'est pas très fiable, car le paramètre α s'avère impossible à régler autrement qu'au cas par cas. Ce dernier contrôle l'importance relative des simplexes trouvés pour différentes tolérances, $\alpha = \infty$ correspondant à la proposition initiale de Lin et al. [213] ; $(1-t)^\alpha$ pourrait être remplacé par une fonction décroissante de t quelconque.

3.1.4 Détection stochastique des slivers

Tout comme Cheng et Chiu [62], nous nous basons sur l'observation indiquée au début du paragraphe 3.1.2.3. Cependant, au lieu de devoir choisir un seuil assez arbitraire sur le volume minimal d'un simplexe nous préférons utiliser un test statistique. Cela reporte a priori le problème dans le choix du seuil sur la p-value, mais cette valeur se révèle en général plus facile à choisir.

Dans l'algorithme initial, le volume V_m d'un m-simplexe est considéré négligeable si l'inégalité suivante est satisfaite.

$$V_m \leq \frac{\sigma^m L^m}{m!},$$

où L désigne la longueur de la plus grande arête du simplexe. Toute la difficulté consiste à déterminer la bonne échelle $\sigma \in]0,1[$, cette dernière valeur pouvant en plus dépendre de m plus finement que via l'exponentiation. C'est pourquoi on choisit plutôt d'adapter le test statistique via un algorithme Monte-Carlo. La distribution des volumes des m-simplexes pris aléatoirement dans des voisinages est d'abord évaluée, puis comparée à la distribution empirique observée pour de tels simplexes dans $\mathbb{R}^{m+1}$ (les longueurs des arêtes suivant la loi constatée sur les plus courtes distances du jeu de données initial). On peut par exemple utiliser le test de Kolmogorov-Smirnov, ou estimer les densités de probabilité puis utiliser une distance L^q. Voici l'algorithme en pseudo-code :

1. Estimer la densité de probabilité f des distances intra-voisinages.
2. Pour d allant de 2 à d_{max} :
   (a) échantillonner N volumes de simplexes aléatoires dont les longueurs des côtés suivent la distribution f ;
   (b) estimer la densité de probabilité v des volumes des $d-$simplexes engendrés par $d+1$ points pris aléatoirement dans un voisinage ;
   (c) si les densités f et v sont trop différentes (sur une échelle de 0 à 1), retourner $d-1$.

Les simplexes n'étant pas contraints, il est possible cette fois de prouver la convergence vers la dimension d de la variété. Nous indiquons ici les grandes lignes de la démonstration, qui seront détaillés afin d'obtenir les vitesses dans un travail ultérieur. Commençons par rappeler que le volume d'un simplexe dont les longueurs des arêtes sont majorées par η

est au plus de l'ordre de η^d. Fixons alors $u \in \mathcal{U}$, son espace tangent étant noté $T_u\mathcal{U}$. Il suffit de remarquer que pour $\epsilon > 0$ assez petit, l'écart entre $u' \in B(u, \epsilon) \cap \mathcal{U}$ et son projeté orthogonal sur $T_u\mathcal{U}$ est au plus quadratique en ϵ. Ainsi, la variation volumique d'un simplexe $S = (s_1, \ldots, s_d)$ (dont les longueurs des côtés sont de l'ordre de ϵ) projeté sur $T_u\mathcal{U}$ est un $O(\epsilon^{2d})$. Cette variation est négligeable devant les volumes des simplexes (initial et projeté), donc les distributions vont tendre à être les mêmes.

3.1.5 Tests

Les paramètres sont réglés en fonction de n (nombre d'échantillons), aux valeurs heuristiques donnant de bons résultats en moyenne. Ainsi le nombre de voisins est fixé à $\lceil\sqrt{n}\rceil$, le seuil étant choisi à 1% pour l'algorithme de Bruske et Sommer [51] basé sur l'ACP (d_{clust}), et à 0.05 pour la méthode du paragraphe précédent utilisant les slivers (d_{sliv}). Nous comparons ces deux dernières à l'algorithme de Levina et Bickel [207], noté d_{levi}.

Le premier exemple est une sphère fonctionnelle de dimension d, c'est à dire une transformation linéaire des composantes d'une sphère de $\mathbb{R}^{d+1}$. À $(x_1, \ldots, x_{d+1}) \in S^d$ nous faisons correspondre la fonction

$$t \mapsto \sum_{i=1, i\, impair}^{d+1} x_i \cos\left(\frac{i+1}{2}t\right) + \sum_{i=2, i\, pair}^{d+1} x_i \sin\left(\frac{i}{2}t\right) ,$$

qui est discrétisée sur 200 points. Les résultats en faisant varier le nombre d'échantillons de 100 à 1000 sont visibles dans le tableau 3.1, "$x|y$" signifiant que le nombre de dimensions oscille entre les valeurs x et y. d_{clust} surestime toujours la dimension, tandis que d_{sliv} trouve la valeur correcte à partir de $n = 500$, d_{levi} n'effectuant pas d'erreurs sur ce premier test.

n	d_{clust}	d_{levi}	d_{sliv}
100	6|7	5	6
200	6|7	5	6
500	6	5	5
1000	6	5	5

Table 3.1: Dimension estimée de la 5-sphère fonctionnelle suivant le nombre de courbes.

Le second exemple consiste en un ruban ($2D$) auquel on applique 10 torsions (obtenant donc une surface fortement incurvée) avant de recoller les deux extrémités. On l'appelle le 10-ruban de Möbius, par analogie avec l'exemple classique à une torsion. Celui-ci est paramétré dans l'espace $\mathbb{R}^3$, et transformé en variété fonctionnelle comme pour la sphère, à l'aide des fonctions $\cos t, \sin t$ et $\sin 2t$. Les résultats en faisant varier le nombre d'échantillons de 100 à 1000 sont visibles dans le tableau 3.2. d_{clust} est toujours en dernière position, les deux autres méthodes arrivant ex-aequo (résultat correct à partir de $n = 500$). La figure 3.5 (empruntée à l'article de Hein et Audibert [163]) présente 16000 points échantillonnés sur le ruban dans $\mathbb{R}^3$.

n	d_{clust}	d_{levi}	d_{sliv}
100	3	3	3
200	3	3	3
500	3	2	2
1000	3	2	2

Table 3.2: Dimension estimée du 10-ruban de Möbius fonctionnel.

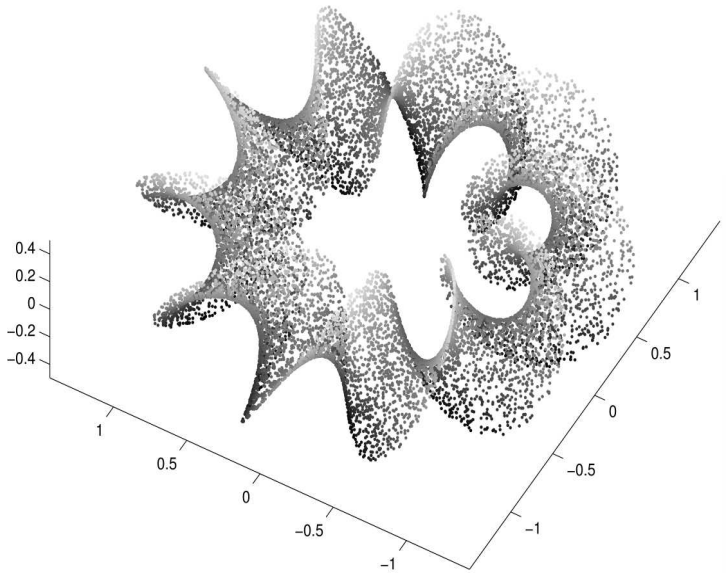

Figure 3.5: 16000 points sur le 10-ruban de Möbius.

Les dimensions estimées sur ces deux premiers exemples sont plutôt satisfaisantes, sauf pour d_{clust}, qui n'est alors pas meilleur qu'une ACP globale.

Nous testons finalement les méthodes sur le jeu de données "ISOMAP faces", consistant en 698 images 64×64 en niveaux de gris, stockées sous forme de vecteurs de longueur 4096. La figure 3.6 montre un échantillon des visages du jeu de données, qui comportent en principe trois degrés de liberté. Il est cependant admis qu'une estimation légèrement supérieure reste correcte dans ce cas.

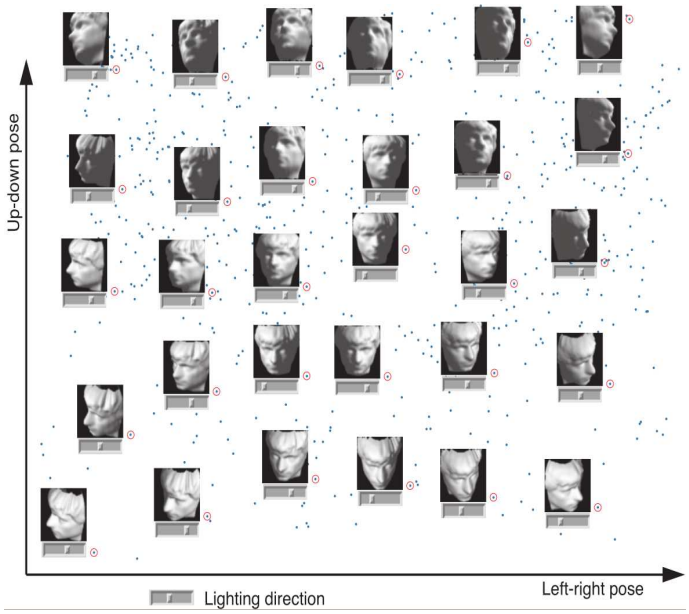

Figure 3.6: Aperçu du jeu de données "ISOMAP faces".

Les résultats obtenus via les méthodes d_{clust}, d_{levi} et d_{sliv} sont respectivement 11, 5 et 8. Ainsi, d_{sliv} semble en général légèrement surestimer la dimension, mais c'est préférable à une sous-estimation dans notre application. De plus, cette méthode elle est très adaptée à la façon dont la dimension est réduite. Nous préférons cependant utiliser l'algorithme de Levina et Bickel [207], le métamodèle n'étant pas significativement amélioré en utilisant le résultat de d_{sliv}. Ce choix est motivé par la lenteur actuelle de l'estimation de d_{sliv}, qui nécessite de nombreuses évaluations de densités.

d_{clust}, d_{sliv} et d_{levi} sont implémentés dans le package R, le dernier estimateur étant choisi par défaut pour les raisons indiquées ci-dessus.

Ouverture

Nous avons vu au paragraphe 1.1.3 comment estimer un graphe de voisinage adapté, connaissant d. Sont également à notre disposition plusieurs méthodes d'estimation de d sachant k. Ce dernier doit pouvoir varier localement, mais en fait chaque algorithme est adaptable à ce cas. Les tous premiers tests de couplage des deux estimateurs n'ont pas été concluants, mais cette voie mérite d'être explorée. Costa et Hero [75] développent une méthode difficile à classer dans les catégories des paragraphes précédents, basée sur les arbres couvrants. Celle-ci peut constituer une future piste de recherche.

3.2 Réduction de la dimension

Comme indiqué en introduction du chapitre, il est nécessaire de réduire la dimension des sorties pour que l'apprentissage statistique puisse se dérouler de façon satisfaisante. Dans le but de représenter la variété $\mathcal{U}$ connue uniquement par le biais des n échantillons $y_1, \ldots, y_n$ en dimension réduite d, deux axes principaux se dégagent :

- chercher n vecteurs $z_1, \ldots, z_n \in \mathbb{R}^d$ représentant la topologie des y_i, en imposant des contraintes locales et/ou globales sur les représentations ;
- paramétrer la variété entière en cherchant une fonction $\mathbb{R}^d \to \mathcal{U}$, interpolant (modulo un éventuel relâchement de contraintes) les points $y_1, \ldots, y_n$.

La réponse au second problème est bien sûr plus intéressante, mais ce dernier est aussi beaucoup plus délicat. C'est pourquoi nous commençons par rechercher n représentations réduites, avant de paramétrer la variété par inversion au chapitre 4.

Deux grandes familles se distinguent alors : les approches linéaires, très utilisées car offrant des fonctions de codage/décodage particulièrement simples tout en possédant une capacité d'approximation en général suffisantes, et les méthodes non linéaires auxquelles nous nous intéresserons plus particulièrement. Ces dernières ont pour objectif d'optimiser les représentations afin qu'elles reflètent au mieux (via la distance euclidienne et/ou géodésique dans $\mathbb{R}^d$) la topologie initiale des données. Si ce but est atteint la régression $x_i \mapsto y_i$ peut s'en trouver nettement facilitée, la dimension en sortie étant réduite "naturellement". Apprendre moins de coefficients permet en outre de diminuer les risques de surapprentissage. En contrepartie, la détermination des représentations et de leurs inverses (codage/décodage) est une tâche bien plus délicate que dans le cas linéaire.

Nous passons en revue quelques méthodes de réduction de dimension linéaires (essentiellement l'ACP, qui nous est souvent utile au cours de ce chapitre), puis abordons une bibliographie sélective des méthodes de réduction de dimension convergeant finalement vers une paraissant très appropriée par rapport à l'application finale.

Cette section est presque exclusivement bibliographique. Plusieurs approches non utilisées sont brièvement présentées, mais ne seront pas mises en œuvre et peuvent être omises par le lecteur. Seules les deux méthodes du paragraphe 3.2.3 sont codées dans le package R.

Plan de cette seconde partie :

1. **Bibliographie sélective des approches linéaires.**
2. **État de l'art des méthodes non linéaires globales.**
3. **Quelques méthodes non linéaires locales.**
4. **Comparaisons des représentations obtenues par les différentes approches.**

3.2.1 Méthodes linéaires

"Linéaire" signifie que la représentation d'une courbe est une application linéaire de $\mathcal{C}([a,b])$ dans $\mathbb{R}^d$. L'évaluation d'une fonction en d points distincts : $r : y \mapsto (y(t_1), \ldots, y(t_d))$ constitue un exemple simple d'une telle application. Il est cependant préférable d'utiliser une base orthonormée, clairement plus intéressante lorsque $d \ll D$. La réduction de dimension

s'écrit alors

$$r : \begin{array}{ccc} \mathcal{C}([a,b]) & \longrightarrow & \mathbb{R}^d \,, \\ y & \mapsto & (\langle y, f_1\rangle, \ldots, \langle y, f_d\rangle) \,. \end{array}$$

3.2.1.1 Base des composantes principales

L'objectif de ce paragraphe est la définition d'une famille de fonctions orthonormées ayant de bonnes propriétés d'approximation, utiles pour représenter une courbe dans $\mathcal{C}([a,b])$. Pour cela nous commençons par introduire la notion d'analyse en composantes principales (ACP) en dimension finie.

ACP en dimension finie

Étant donnée une famille F de m vecteurs de $\mathbb{R}^q$, l'ACP recherche les directions de plus grandes variations au sein de F, en leur imposant d'être mutuellement orthogonales. Les composantes principales $\zeta_j \in \mathbb{R}^q$ sont définies comme suit

$$\begin{array}{ll} \forall j = 1, \ldots, \min(m,q) \,, & \zeta_j \text{ maximise la variance projetée } \mathrm{Var}(\{\langle \zeta_j, F_i\rangle\}_{i=1,\ldots,m}) \\ & \text{sous les contraintes d'orthonormalité :} \\ & \|\zeta_j\| = 1 \text{ et } \forall \ell = 1, \ldots, j-1 \,,\ \langle \zeta_j, \zeta_\ell\rangle = 0 \,. \end{array}$$

Elle peuvent s'obtenir numériquement à l'aide du théorème suivant par exemple.

Théorème 3.2.1 *Une matrice réelle $M \in \mathbb{R}^{m\times q}$ admet une unique décomposition du type*

$$M = U\, D\, {}^t\!V \,,$$

où U et V sont des matrices orthogonales de tailles m et q respectivement. D est une matrice de taille $m \times q$ nulle sauf sur la diagonale, celle-ci étant ordonnée décroissante. On appelle cette écriture de M décomposition en valeurs singulières (SVD).

Considérant F comme une matrice $m \times q$ contenant les vecteurs en lignes, la SVD s'écrit $F = UD{}^t\!V$ et les j premières composantes principales sont les j premières colonnes de V. Pour plus de détails sur l'algorithme de calcul de la SVD, voir par exemple l'article de Demmel et Kahan [96] et le rapport de Larsen [198]. Diverses propriétés de l'ACP sont étudiées dans l'article de Rao [260], mais nous n'aurons besoin que de la suivante :

Théorème 3.2.2 *Les composantes principales $\zeta_j, j = 1, \ldots, \min(m,q)$ forment une base orthonormée de Vect$(F_1, \ldots, F_m)$. Pour toute famille orthonormée $e_1, \ldots, e_q$ de $\mathbb{R}^q$:*

$$\forall\, \ell = 1, \ldots, \min(m,q) \,,\ \sum_{i=1}^{m} \left\| F_i - \sum_{j=1}^{\ell} \langle F_i, \zeta_j\rangle \zeta_j \right\|_2 \leq \sum_{i=1}^{m} \left\| F_i - \sum_{j=1}^{\ell} \langle F_i, e_j\rangle e_j \right\|_2 \,.$$

C'est-à-dire que l'approximation au sens de la norme euclidienne des vecteurs F_i est meilleure (en moyenne) pour la base des composantes principales, lorsqu'on se limite à $\ell \leq \min(m,q)$ dimensions. Pour une présentation complète, voir le livre de Jolliffe [172].

ACP en dimension infinie

L'ACP peut être étendue au cas d'une famille $F = F_1, \ldots, F_m$ libre (i.e. engendrant un espace de dimension m) de fonctions réelles continues à support compact : il suffit de résoudre le même problème d'optimisation, en calculant les produits scalaires $\langle f, h\rangle = \int f(t)h(t)dt$. De plus, la même propriété reste valable en remplaçant $\mathbb{R}^q$ par $\mathcal{C}([a,b])$ et q par $+\infty$.

Théorème 3.2.3 *Les composantes principales fonctionnelles $\zeta_j, j = 1, \ldots, m$ forment une base orthonormée de Vect$(F_1, \ldots, F_m)$. Pour toute famille orthonormée $e_1, \ldots, e_m$ de $\mathcal{C}([a,b])$,*

$$\forall\, \ell = 1, \ldots, m\,,\ \sum_{i=1}^{m} \left\| F_i - \sum_{j=1}^{\ell} \langle F_i, \zeta_j\rangle \zeta_j \right\|_{L^2} \leq \sum_{i=1}^{m} \left\| F_i - \sum_{j=1}^{\ell} \langle F_i, e_j\rangle e_j \right\|_{L^2} .$$

Ainsi l'approximation au sens de la norme L^2 des courbes F_i est meilleure (en moyenne) pour la base des composantes principales, lorsqu'on se limite à $\ell \leq m$ dimensions.

Ramsay et Silverman [258] présentent au chapitre 6 de leur livre l'analyse en composantes principales fonctionnelles (ACPF), et indiquent des méthodes de résolution (SVD sur la matrice discrétisée, décomposition sur une nouvelle base de fonctions, régularisation). Nous n'entrons pas dans les détails de calcul et supposons dorénavant disposer de cette ACP fonctionnelle (initialement introduite par Dauxois et al. [82]).

Notons qu'avec une ACPF sur les sorties du code de calcul on obtient seulement une famille orthonormée à n éléments, qui n'est pas une base de $\mathcal{C}([a,b])$. Cependant, les quelques premières composantes principales suffisent souvent à bien représenter chaque courbe y_i. En pratique on retient les $d \ll n$ premières composantes pour approximer toute courbe du même type que les y_i. Ceci peut être justifié d'une autre manière en supposant que les courbes y_i sont des réalisations d'un processus stochastique P : les composantes principales $\zeta_j, j = 1, \ldots, n$ constituent alors une bonne approximation de la représentation optimale de P en termes de fonctions orthogonales (voir par exemple Ghanem et Spanos [139], chapitre 2 pour plus de détails).

> L'ACP fonctionnelle fournit une bonne approximation de la meilleure représentation possible des sorties du code par une famille orthogonale.

La figure 3.8 présente les trois premières composantes principales d'un jeu de 100 courbes CATHARE d'évolution de la température, auxquelles on a retranché leur moyenne empirique (figure 3.7).

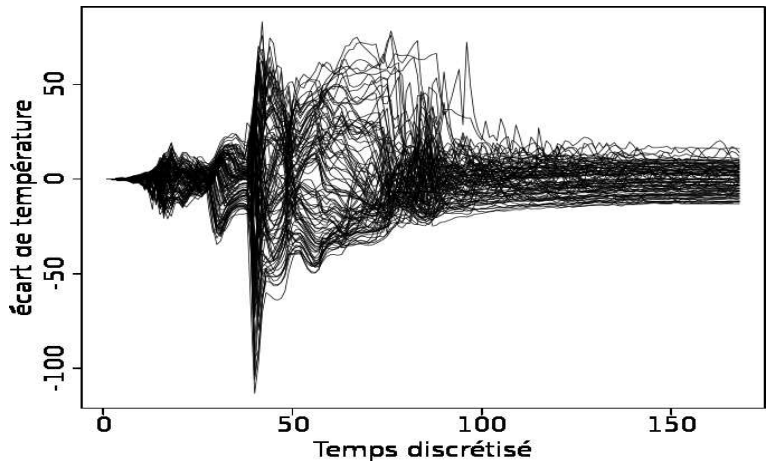

Figure 3.7: 100 courbes CATHARE centrées.

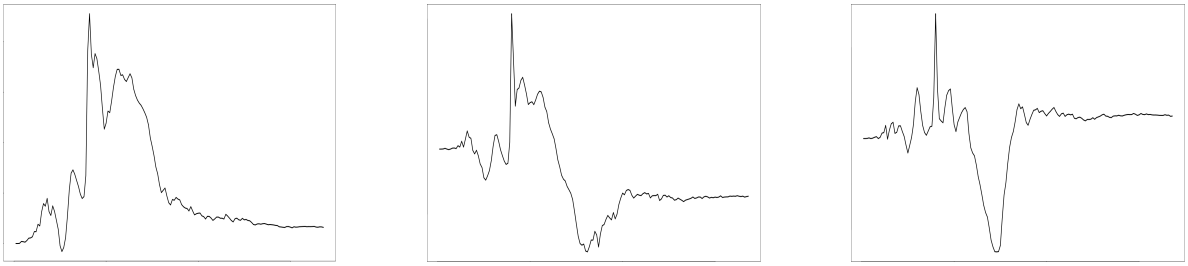

Figure 3.8: Les trois premières composantes principales, de gauche à droite.

3.2.1.2 Quelques extensions de l'ACP

Girard et al. [141] remarquent que l'ACP revient à chercher des directions de projection a_j maximisant la variance projetée, la reconstruction s'opérant via la somme pondérée (par les coefficients de décomposition) des fonctions a_j. Une extension naturelle consiste alors à utiliser un index différent de la variance projetée, plus guidé par des considérations topologiques ; les auteurs en proposent un évaluant la conservation des voisinages (un autre index plus facile à optimiser est introduit par Girard et Iovleff [140]). Une reconstruction non linéaire utilisant des splines de lissages est également présentée comme alternative à la reconstruction usuelle.

On peut voir l'ACP comme la recherche d'une représentation optimale d'un nuage de données par des combinaisons linéaires d'éléments se trouvant sur d droites vectorielles du type $t \mapsto ta_j$. L'analyse en courbes principales s'obtient en remplaçant les droites vectorielles par des courbes (contraintes à ne pas trop tourner pour éviter une adéquation totale aux données, en général non souhaitée). Plusieurs intuitions différentes ont mené à la construction de différents types de courbes, notamment celles passant par les moyennes locales des données [156; 311; 97; 92], et de longueur ou courbure bornée [180; 277]. Ce type d'approche est plutôt réservé à la visualisation/description de densités de probabilité en dimension ≤ 3, sauf dans les articles de Delicado et Huerta [93] et LeBlanc et Tibshirani [200] qui proposent respectivement une construction récursive afin de définir les courbes en dimensions supérieures, et une méthode utilisant des splines de régression ponctuelles (donc inadaptées à notre cas d'application).

Enfin, l'ACP peut être effectuée dans un espace abstrait de représentation via une fonction noyau [283; 272]. Cependant, les représentations obtenues ne sont pas dictées par des considérations géométriques (Lee et Verleysen [201], paragraphe 4.4.1) ; elles peuvent donc être difficiles à exploiter.

3.2.1.3 Autres bases de fonctions

Il existe de nombreuses familles de fonctions dignes d'intérêt, dont les polynômes orthogonaux [107; 188], les bases de splines [85; 284], la base de Fourier et les bases d'ondelettes [231; 90; 220]. Walter [332] effectue une présentation unifiée de plusieurs familles de fonctions orthogonales. Toutes celles-ci sont non adaptatives, dans le sens que leurs formes ne s'adaptent pas aux données. On peut toutefois sélectionner certaines fonctions de ces bases pour approximer efficacement un signal donné, notamment avec les ondelettes [83].

Quelques différences entres ces bases (en comparaison avec l'ACP) sont visibles sur les figures 3.9, 3.10, 3.11 et 3.12, représentant une courbe du jeu de données CATHARE avec seulement cinq fonctions. La courbe d'origine est tracée en noir, ses approximations en rouge. À cette faible résolution, aucune base ne permet de retrouver les petites oscillations observées entre les abscisses 100 et 160. Il est cependant clair que la base ACP est plus avantageuse en terme de nombre de coefficients à utiliser (comme prévu par la théorie).

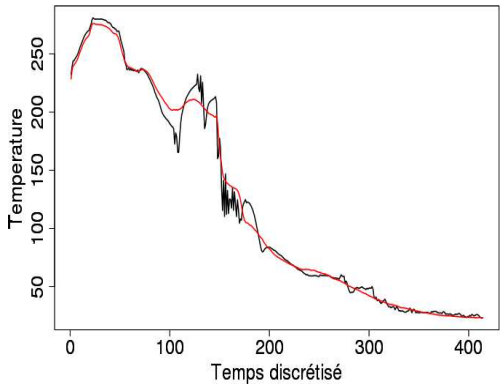

Figure 3.9: Approximation par la base ACP.

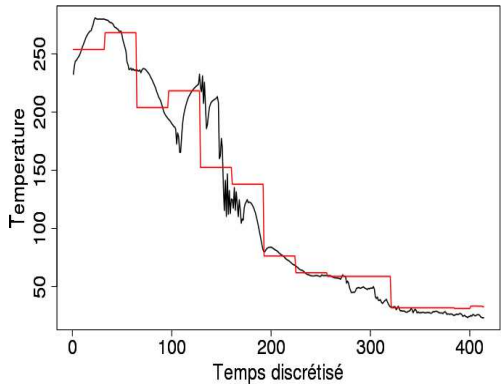

Figure 3.10: Approximation par la base de Haar (ondelettes).

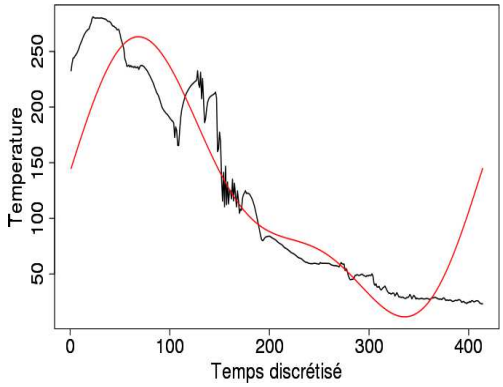

Figure 3.11: Approximation par la base de Fourier.

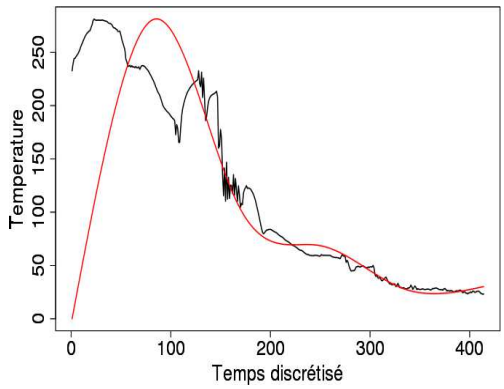

Figure 3.12: Approximation par une base de B-splines.

3.2.2 Méthodes non linéaires globales

Dans un premier temps sont passées en revue quelques techniques se ramenant à la recherche de vecteurs propres d'une certaine matrice basée sur les données. Nous verrons ensuite des méthodes probabilistes (supposant connu un modèle du jeu de données), neuronales, puis des techniques ne disposant pas d'algorithmes d'optimisation efficaces (algébriques). Ces dernières fournissent néanmoins souvent de meilleurs résultats que leur homologues spectrales.

Toutes les méthodes de cette première partie recherchent une paramétrisation globale des données, et ne sont donc applicables qu'à des variétés non fermées ("à bords mais ne contenant pas les bords"). En effet en visualisant une variété fermée comme une sphère, il est clair qu'il n'existe aucun paramétrage (au moins) continu de celle-ci. Cette dernière visualisation se justifie dans une certaine mesure, la conjecture de Poincaré généralisée indiquant que toute variété homotopiquement équivalente à − i.e. "s'obtenant par déformations depuis" − la d-sphère unité est aussi homéomorphe à cette dernière. La conjecture est à présent complètement démontrée : Smale [294] a prouvé le cas $d > 4$, Freedman [128] le cas $d = 4$ et on peut lire la démonstration de Perelman [249, 250, 251] du cas $d = 3$ dans le livre de Morgan et Tian [240]. De plus les variétés homotopiquement équivalentes à une sphère forment une classe probablement assez large pour les applications qui nous préoccupent. Attention cependant, il se peut qu'une variété soit homéomorphe à la sphère unité sans lui être difféomorphe ; on parle alors de sphère exotique [233; 234], mais il y a peu de risques d'être dans ce cas (de toutes manières indétectable avec un nombre fini d'échantillons).

3.2.2.1 Approches spectrales

Les algorithmes de ce paragraphe et des suivants s'attachent à optimiser la disposition des représentations en tenant compte de tous les points simultanément. Plus précisément, on se ramène ici à un problème de vecteurs/valeurs propres, que l'on sait résoudre assez rapidement (si n ne dépasse pas quelques milliers) malgré sa complexité intrinsèque. Il est parfois possible d'estimer la projection d'une nouvelle courbe $y \in \mathcal{U}$ via ces méthodes [26; 199], mais il n'existe en général pas de formule permettant de reconstruire un élément $y \in \mathcal{U}$ à partir de $z \in \mathbb{R}^d$. C'est pourquoi ce type d'approche est plutôt utilisé pour la visualisation et l'indexation de données.

Conservation des propriétés globales

Par "propriétés globales" nous entendons certaines caractéristiques des données, sans pondération liée à la distance entre les points. La première méthode appartenant à cette famille décrite historiquement est linéaire (MDS pour Multi Dimensional Scaling [316; 76]), mais figure dans ce paragraphe car elle est à la base de plusieurs approches non linéaires. L'idée consiste à conserver au mieux les produits scalaires (après centrage des données) dans l'espace de représentation. Notant S la matrice de Gram des produits scalaires, on écrit

$$S = Y\,{}^tY = X\,{}^tX = U\Lambda\,{}^tU\,,$$

où Y (resp. X) est la matrice des données initiales (resp. réduites) en lignes, $U\Lambda\,{}^tU$ représentant la décomposition spectrale de S. Par identification, on en déduit X en conservant les d

colonnes de $U\Lambda^{\frac{1}{2}}$ correspondant aux d plus grandes valeurs propres. Une extension naturelle consiste à calculer les produits scalaires dans un espace abstrait de haute dimension [283] avant d'appliquer la même méthode. Une variante (toujours linéaire) existe en utilisant les distances euclidiennes au lieu des produits scalaires : MDS métrique.

Cette dernière variante a inspiré Tenenbaum et al. [309] pour leur méthode ISOMAP (ISOmetric MAPping), appliquant la méthodologie ci-dessus à la matrice des distances géodésiques approchées [28]. En effet ces dernières (évaluées via un graphe de voisinage) définissent la topologie naturelle d'une variété, et sont donc plus adaptées. Comme son nom le suggère, ISOMAP cherche à représenter les données de manière isométrique. Or il n'est pas du tout garanti qu'une variété de dimension d soit isométrique à (un ouvert de) $\mathbb{R}^d$: la sphère privée d'un point est difféomorphe à $\mathbb{R}^2$ mais toutes les distances ne peuvent être conservées. de Silva et Tenenbaum [89] développent une extension de ce dernier algorithme visant à mieux préserver les angles.

Conservation des propriétés locales

Contrairement aux approches du paragraphe précédent, l'accent est mis ici sur la préservation des relations de voisinages au détriment des angles, distances ou autres caractéristiques entre sommets éloignés dans le graphe. Cela permet plus de souplesse dans la représentation, pouvant coder une classe de variétés plus générale.

Belkin et Niyogi [21] proposent de minimiser le terme d'erreur suivant

$$E = \sum_{i,j=1}^{n} s_{ij} \|z_i - z_j\|^2 ,$$

où s_{ij} désigne la similarité entre y_i et y_j, souvent choisie gaussienne ($s_{ij} = e^{\frac{-\|y_i - y_j\|^2}{\sigma^2}}$), z_i étant la représentation d-dimensionnelle de y_i. E peut s'écrire à l'aide du laplacien du graphe comme suit :

$$E = \mathrm{Tr}(ZL^tZ) ,$$

avec $L = D - S$ où D est la matrice diagonale des degrés et $S_{i,j} = s_{ij}$, Z contenant les z_i en lignes. En ajoutant des contraintes de normalisation évitant les solutions triviales, la matrice Z optimale correspond aux d premiers vecteurs propres généralisés v_j vérifiant $Lv_j = \lambda Dv_j$, en colonnes. La représentation via les temps moyens d'aller-retour approchés [272; 256] s'obtient d'une manière très similaire : z_i est alors la i^{eme} colonne tronquée de $\Lambda^{\frac{1}{2}t}U$ où Λ est la matrice diagonale des valeurs propres de L, U contenant les vecteurs propres en colonnes.

Roweis et Saul [269] partent d'une intuition différente : au lieu de confier l'optimisation des représentations aux seules similarités, ils cherchent à conserver explicitement la structure de voisinage. Pour cela chaque y_i est exprimé en fonction de ses k voisins : $y_i = \sum_{j \in \mathcal{V}(i)} \omega_{ij} y_j$. Les poids ω_{ij} sont ensuite fixés, et l'optimisation s'effectue sur les z_i. L'expression suivante doit alors être minimisée.

$$E = \sum_{i=1}^{n} \left\| z_i - \sum_{j \in \mathcal{V}(i)} \omega_{ij} z_j \right\|^2 .$$

Ce dernier problème se ramène à la recherche des vecteurs propres d'une matrice creuse. Un point s'exprimant linéairement par rapport à ses voisins, la méthode est baptisée LLE pour Locally Linear Embedding. Donoho et Grimes [106] appliquent une méthodologie proche de celle de LLE en étendant le cadre théorique de LE. Ils approximent une forme quadratique $\mathcal{H}$ sur la variété basée sur les hessiennes estimées en chaque point y_i, les coordonnées z_i s'obtenant ensuite via le noyau de $\mathcal{H}$.

He et Niyogi [160] et He et al. [161] approximent respectivement les algorithmes LE et LLE linéairement, en ajoutant la contrainte $z_i = Ay_i$. Une transformation linéaire préservant les voisinages mieux que l'ACP est ainsi obtenue. Sha et Saul [286] optimisent aussi une transformation linéaire, mais sur les représentations z_i obtenues par LE ou LLE, afin de conserver au mieux les angles locaux (conformal eigenmaps).

Alignement de cartes locales

Les méthodes de ce paragraphe s'appuient sur la définition d'une variété : en effet, le paramétrage naturel d'un tel objet consiste en un jeu de cartes locales, et tout le problème consiste à transformer ces cartes en un unique système de coordonnées (ce n'est pas toujours possible en théorie, mais une telle paramétrisation existe souvent en pratique). L'idée ici consiste à appliquer une transformation affine sur chaque carte locale pour obtenir les coordonnées globales, en respectant certaines contraintes.

Teh et Roweis [308] proposent d'utiliser la fonction de coût de l'algorithme LLE pour guider l'alignement des cartes, obtenant une nouvelle méthode appelée LLC (Locally Linear Coordination) pouvant s'appliquer à un ensemble de cartes locales déjà calculées. De Ridder et Franc [87] développent une version plus robuste de ce dernier algorithme, basée sur un mélange de t-distributions.

Zhang et Zha [355] s'appuient sur le développement de Taylor à l'ordre un de la fonction de représentation de la variété :

$$z_j = z_i + J_i^t(y_j - y_i) + o(\|y_i - y_j\|)\,,$$

J_i représentant la matrice jacobienne de la fonction en y_i. Les auteurs procèdent en deux temps afin d'approcher cette dernière matrice :

1. estimation des coordonnées locales θ_i des données centrées dans chaque voisinage, via des ACP (locales) ;
2. minimisation du terme d'erreur $\|\delta_{z_i} - L_i\theta_i\|$ où δ_{z_i} est la matrice des $z_j - z_i, j \in \mathcal{V}(i)$ en lignes, et L_i une transformation linéaire.

Ce dernier terme mesure l'écart entre les vraies coordonnées globales et leurs approximations linéaires locales. Cette méthode permet de reconstruire un élément y à partir de sa représentation z : en effet $\delta_{z_i} - L_i\theta_i \simeq 0$ avec $\delta_{z_i} = z_i - z$, i étant l'indice du plus proche voisin dans l'espace des représentations. Cela suppose toutefois que les voisinages sont conservés, ce qui n'est pas explicitement garanti par la résolution spectrale.

Plusieurs variantes de ce dernier algorithme ont été proposées, notamment une version adaptative déterminant le nombre de voisins et la courbure locale en fonction des données [334], et une permettant de recoller des coordonnées globales pour deux parties disjointes de la variété [356]. Enfin, une version incrémentale et "interpolée" (n'utilisant pas tous les points de l'ensemble d'apprentissage, donc intéressant pour de très grandes bases de données) est introduite par Liu et al. [216].

3.2.2.2 Approches probabilistes

Ces méthodes s'appuient sur un modèle probabiliste des données, en général un mélange de gaussiennes. Ces gaussiennes s'interprètent comme des ACP locales, dont les coordonnées sont ensuite alignées tout en respectant des contraintes sur les zones en chevauchement.

Roweis et al. [268] proposent un algorithme E-M optimisant simultanément les représentations d-dimensionnelles et les paramètres du mélange de gaussiennes en grande dimension, tout en satisfaisant des contraintes d'alignement (les directions principales de deux voisinages proches doivent être semblables) et de cohérence : la représentation z d'un point y doit vérifier $\mathbb{P}(z|M_1, y) \simeq \mathbb{P}(z|M_2, y)$ dès lors que les modèles locaux M_1 et M_2 expliquent y avec une probabilité non négligeable.

Brand [42] relâche la contrainte d'alignement imposée dans l'algorithme précédent, ce qui lui permet d'obtenir les représentations optimales algébriquement. La méthode procède en deux temps : les paramètres du modèle probabiliste sont d'abord estimés, puis des transformations affines locales sont optimisées sous des contraintes de cohérence dans les zones de chevauchement (comme ci-dessus). Le modèle probabiliste peut être inversé pour reconstruire la variété à partir de ses représentations réduites ; toutefois − comme indiqué par l'auteur − cela ne permet pas de retrouver exactement les y_i, à moins de mettre en place un coûteux algorithme d'optimisation.

La méthode proposée par Hinton et Roweis [164] se démarque des deux précédentes. En effet, si le modèle sous-jacent est encore un mélange de gaussiennes, on ne cherche plus à transformer les coordonnées locales mais plutôt à conserver les probabilités d'appartenance dans l'espace de représentation. Une possibilité consiste alors à minimiser la divergence de Kullback-Leibler entre les deux distributions, sur la variété et dans l'espace de représentation. van der Maaten et Hinton [322] simplifient l'expression à minimiser et remplacent les distributions gaussiennes par des t-distributions.

Extensions des cartes auto-organisatrices (SOM)

Nous concluons ce paragraphe avec deux méthodes visant à pallier les limitations de l'algorithme SOM (Self Organizing Maps, [186; 187]).

Le modèle introduit par Bishop et al. [32] présente l'originalité de chercher directement une application R (dite de "reconstruction" dans cette thèse) de $\mathcal{E}_E$ dans $\mathbb{R}^D \equiv \mathcal{C}([a, b])$. Pour cela une grille régulière à m nœuds est formée dans $\mathcal{E}_E$, chacun de ces éléments ayant une image sur $\mathcal{U} \subset \mathcal{C}([a, b])$. La variété est modélisée par un mélange de gaussiennes multivariées centrées en ces images, tous les paramètres étant estimés par un algorithme E-M.

Lee et al. [204] choisissent d'utiliser plutôt une grille déterminée par les données. Ainsi, un graphe de voisinage est d'abord construit et les distances géodésiques $\rho_{i,j}$ estimées. Les coordonnées z_i sont alors toutes initialisées à zéro, puis mises à jour d'une manière similaire à l'entraînement des réseaux de neurones : $z_i \leftarrow z_i + \alpha v_\lambda(i,\ell)(q - z_i)$, q étant tiré aléatoirement dans $\mathcal{E}_R$ et v_λ dépendant des relations de voisinage dans le graphe. Cet algorithme est difficile à automatiser car, comme pour les réseaux de neurones, il faut régler l'évolution temporelle des paramètres d'apprentissage.

3.2.2.3 Réseaux de neurones

Les méthodes de ce paragraphe utilisent un réseau de neurones à plusieurs couches cachées, afin de réduire la dimension dans un premier temps puis de reconstruire la variété ensuite. Celui-ci est représenté schématiquement sur la figure 3.13, empruntée à l'article de Demers et Cottrell [95].

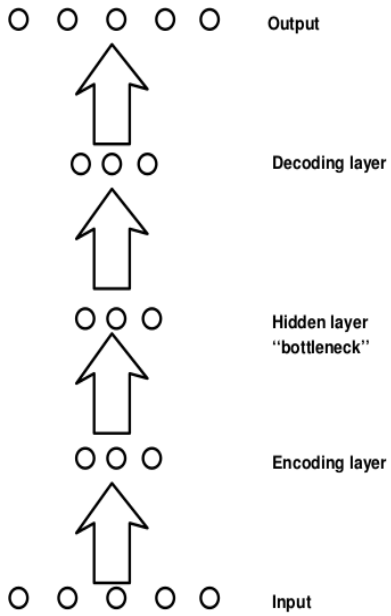

Figure 3.13: Réseau de neurones encodeur/décodeur.

Kramer [189] suggère d'ajouter des connections directes des entrées vers la couche centrale pour faciliter l'apprentissage de la composante linéaire de la onction de représentation. Un critère basé sur le nombre de contraintes est proposé pour déterminer le nombre de neurones sur les couches de codage (deux et quatre), respectivement noté N_1 et N_2. Ces paramètres sont cruciaux : s'ils sont trop petits le réseau ne sera pas assez complexe pour réaliser la fonction, mais s'ils sont trop grands on risque d'observer un effet de surapprentissage. Demers et Cottrell [95] ajoutent un terme de régularisation dans la fonction objectif, permettant de supprimer les neurones inutiles dans la couche centrale ; la dimension est ainsi estimée simultanément.

Dong et McAvoy [105] présentent un algorithme original en deux étapes : la dimension est d'abord réduite à l'aide des courbes principales de Hastie et Stuetzle [156] itérées sur les

résidus successifs. Cela fixe les valeurs d'activation de la couche centrale du réseau de la figure 3.13, et il suffit alors d'entraîner deux réseaux de neurones à trois couches. Remarquons que c'est la seule méthode de ce paragraphe déterminant des représentations guidés par la géométrie des données.

Les réseaux de neurones ont l'avantage de fournir une fonction de reconstruction (de $\mathbb{R}^d$ vers la variété). Cependant, plusieurs paramètres sensibles rendent l'apprentissage automatique difficile, notamment le nombre de neurones sur les couches cachées et l'évolution du taux de renforcement des connexions.

3.2.2.4 Optimisation coûteuse

Les approches de ce paragraphe se réduisent à l'optimisation d'un critère sur les données, mais contrairement au groupe introduisant cette section, elles ne se ramènent pas (en tout cas, pas uniquement) à la recherche de vecteurs propres d'une matrice bien définie. C'est pourquoi elles sont limitées à quelques centaines voire quelques milliers de points.

Sammon [276] propose de minimiser l'expression suivante :

$$E = \sum_{1 \leq i < j \leq n} \frac{(d_{ij} - d_{ij}^{(R)})^2}{d_{ij}},$$

où $d_{ij}^{(R)}$ désigne la distance euclidienne entre les éléments d'indices i et j dans l'espace de représentation. L'optimisation s'effectuant lentement, Agrafiotis [3] propose une approximation de la solution minimisante s'exécutant en temps linéaire. Dans la version originale de l'algorithme on impose $d_{ij}^{(R)} \simeq d_{ij}$, ce qui peut induire des erreurs malgré le terme de normalisation comme le montre la figure 3.14, d_{ij} n'étant pas la distance naturelle sur $\mathcal{U}$. C'est pourquoi Lee et Verleysen [201] (paragraphe 4.3.3) suggèrent plutôt d'utiliser la distance géodésique ρ_{ij}, améliorant sensiblement les performances de la méthode.

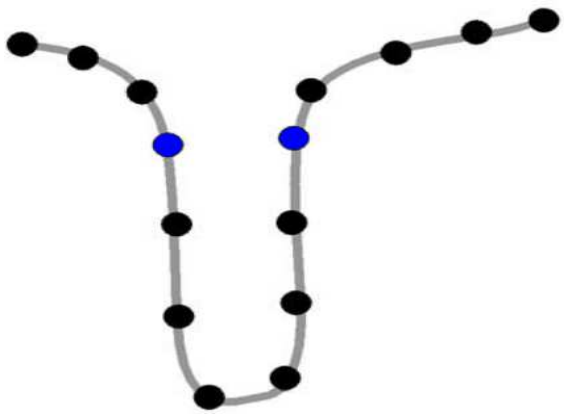

Figure 3.14: Deux points éloignés pour la distance géodésique, proches pour la distance euclidienne.

Demartines et Hérault [94] choisissent de multiplier les carrés $(d_{ij} - d_{ij}^{(R)})^2$ par $F_\lambda(d_{ij}^{(R)})$, fonction décroissante bornée de la distance dans l'espace de représentation. Comparativement à la normalisation par d_{ij}, cela autorise plus facilement une petite distance euclidienne

dans l'espace à être grande dans l'espace de représentation. Lee et al. [203] préfèrent cependant utiliser les distances géodésiques approchées sur $\mathcal{U}$ à l'aide d'un graphe de voisinage. Dans la version de ces derniers auteurs, le paramètre λ est adapté à la densité locale des données en considérant les k plus proches voisins.

Brucher et al. [49] proposent de minimiser une fonction d'erreur assez sophistiquée (légèrement simplifiée ici) :

$$E = \sum_{1 \leq i < j \leq n} |d_{ij} - d_{ij}^{(R)}| \sqrt{\frac{\tau^2 + (d_{ij} - d_{ij}^{(R)})^2}{\tau^2}} \frac{d_{ij}}{\sigma + d_{ij}} .$$

Le premier terme dans l'expression ci-dessus mène à une optimisation plus robuste qu'avec la norme 2, le second étant présent pour des raisons techniques (τ étant égal à un certain quantile des distances inter-points). Le troisième terme permet de limiter l'influence des très petites distances, assimilées à du bruit. $\sigma > 0$ est donc fixé à un seuil déterminé arbitrairement ou via un quantile des $\{d_{ij}\}$. Les auteurs introduisent une application de reconstruction affine par morceaux (de $\mathcal{E}_R$ dans $\mathcal{U}$), dont la construction est générale et peut s'appliquer à d'autres coordonnées réduites. Son optimisation est cependant difficile.

Weinberger et al. [338; 337] cherchent à conserver les distances intra-voisinages tout en maximisant les distances plus grandes, favorisant l'étalement des représentations. Ces contraintes sont écrites à l'aide des produits scalaires, permettant d'en déduire une matrice de Gram dont on souhaite maximiser la trace. Ce type d'optimisation est coûteux en temps [324], mais une fois la matrice obtenue on en déduit les coordonnées réduites par décomposition spectrale comme dans la méthode MDS. Shaw et Jebara [287] remarquent qu'il est avantageux de chercher également à minimiser le volume de l'enveloppe convexe des représentations, et proposent ainsi une variante de ce dernier algorithme.

Aucune des approches vues jusqu'alors ne contrôle géométriquement les représentations des courbes. Le dernier paragraphe ci-dessous visent à pallier cela.

Le lecteur est renvoyé au livre de Lee et Verleysen [201] pour plus de détails sur les méthodes évoquées jusqu'ici (beaucoup de comparaisons y sont effectuées), ainsi qu'au rapport de van der Maaten et al. [323] et à l'article de Tsai [318] pour une présentation des principales approches avec des tests sur divers jeux de données. Fodor [125] effectue un bilan des techniques de réduction de dimension, notamment les méthodes linéaires dont nous n'avons pas trop parlé ici. Enfin, van der Maaten [321] implémente de nombreuses techniques dans une toolbox MATLAB, un programme de démonstration de l'utilisation de MATLAB pour la réduction de dimension étant proposé par Wittman [341].

3.2.3 Méthodes non linéaires locales

Au lieu de voir le problème de réduction de dimension comme une minimisation globale, les approches de cette dernière partie résolvent une suite de petits problèmes locaux, chacun estimant quelques coordonnées globales z_i qui serviront aux problèmes suivants. On exprime ainsi explicitement les relations locales entre les y_i et z_i, facilitant largement la reconstruction (voir le chapitre 4). De plus ces méthodes ont l'avantage d'être nettement

moins paramétriques que les réseaux de neurones (les seuls à fournir automatiquement la fonction R). Enfin, un avantage considérable lié à la nature locale de ces procédures est le faible coût algorithmique (stratégie "divide and conquer" ; voir par exemple le livre de Cormen et al. [74], chapitre 4).

Ces approches présentent l'inconvénient potentiel d'estimer des z_i éloignés de la réalisation d'un critère global, les erreurs pouvant s'accumuler. Cela ne pose pas de problèmes pour la reconstruction, la cohérence locale des données suffisant ; en revanche ce phénomène peut être gênant lors de la phase d'apprentissage statistique des entrées x_i vers les représentations z_i. Un meilleur compromis entre approches locales et globales serait alors souhaitable. Toutefois, des résultats assez satisfaisants sont obtenus via les deux méthodes présentées ici.

3.2.3.1 Alignement incrémental de cartes ACP locales

Cette méthode introduite par Zhan et al. [352] cherche à aligner des cartes locales obtenues à l'aide de l'ACP. Elle est appelée LPcaML pour "Local PCA Manifold Learning". Contrairement aux approches similaires du paragraphe 3.2.2.1 les cartes sont calculées de proche en proche. L'algorithme s'effectue en deux temps :

1. déterminer un ensemble de voisinages recouvrant les y_i, constituant les domaines des cartes locales ;
2. fixer les coordonnées z_i d'un certain voisinage à celles de l'ACP locale, puis appliquer itérativement la procédure décrite ci-après.

Une idée pour déterminer l'ensemble de voisinages initial consiste à minimiser le nombre de cartes utilisées (limitant la complexité algorithmique et l'éventuel risque d'erreur), avec une contrainte de chevauchement : chaque domaine doit partager une fraction α de ses éléments avec l'union de tous les autres. Ce problème est impossible à résoudre en temps polynomial, aussi les auteurs ont recours à un algorithme d'approximation glouton. Ce dernier n'est pas décrit ici car il n'apporte rien à la compréhension.

Quelques notations sont nécessaires avant de poursuivre. $\mathcal{V}_1, \ldots, \mathcal{V}_m$ désigne la suite des indices des voisinages, chacun de cardinal k, recouvrant $\{1, \ldots, n\}$ et vérifiant la propriété suivante :

$$\forall j = 1, \ldots, m \ \mathcal{V}_j \cap (\cup_{\ell<j} \mathcal{V}_\ell) \ .$$

Les données étant supposées localement linéaires, les coordonnées C_j des éléments d'indices $\mathcal{V}_j$ s'obtiennent via des analyses en composantes principales. La matrice des coordonnées globales correspondantes est notée Z_j.

L'hypothèse principale (comme pour les méthodes d'alignement spectrales) consiste à dire que Z_j s'obtient à partir de C_j par une transformation affine T_j. Ceci se justifie par la linéarité des données, locale sur $\mathcal{U}$ et globale dans $\mathcal{E}_R$. $\tilde{C}_j$ désignant la matrice C_j concaténée à une colonne de 1, on écrit alors

$$Z_j = \tilde{C}_j T_j \ ,$$

avec T_j de taille $(d+1) \times d$.

Après avoir fixé arbitrairement Z_1 à C_1, le but est d'optimiser les transformations T_j afin obtenir un système de coordonnées globales $\{z_i\}$ cohérent. Une contrainte naturelle pour parvenir à cet objectif consiste à imposer

$$Z_j^{inter} \simeq \tilde{C}_j^{inter} T_{j+1}$$

où Z_j^{inter} (resp. $\tilde{C}_j^{inter}$) désigne la matrice obtenue à partir de Z_j (resp. $\tilde{C}_j$) en ne conservant que les lignes d'indices présents parmi ceux de $\mathcal{V}_{j+1} \cap (\cup_{\ell<j} \mathcal{V}_\ell)$. Si cette dernière intersection est de cardinal supérieur à d le problème est bien posé, permettant d'obtenir T_{j+1} par moindres carrés et de l'appliquer à $\tilde{C}_{j+1}$, etc. La figure 3.15 illustre cette procédure.

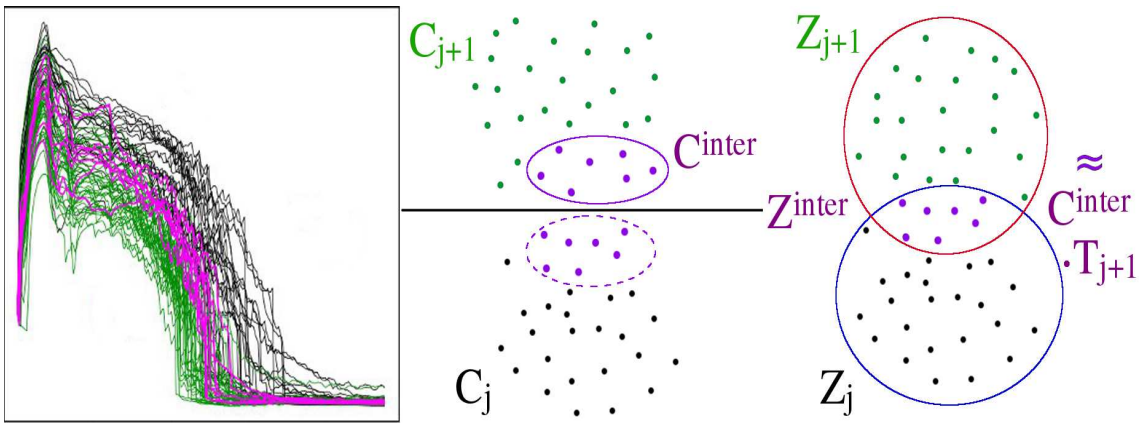

Figure 3.15: Voisinages fonctionnels j et $j+1$ à gauche, avec leur intersection en violet. Représentations par ACP locales au centre. Coordonnées globales à droite.

Grâce à l'utilisation de l'ACP ces méthodes ont l'avantage d'être assez robustes au bruit et aux outliers. Il est cependant difficile de régler automatiquement la taille (fixe) d'un voisinage. Celle-ci pourrait varier en fonction du point (k_i en y_i), mais la dernière remarque impose une valeur minimale de k_i à environ $\frac{d+1}{\alpha}$, ce qui est déjà relativement élevé considérant que l'on ne dispose que de quelques centaines de points.

3.2.3.2 Apprentissage des coordonnées riemanniennes normales

Les deux approches de ce dernier paragraphe cherche à approximer un système de coordonnées bien défini, appelé les *coordonnées riemanniennes normales*, souvent abrégées en "coordonnées normales". La compréhension de ces dernières nécessite l'introduction de l'application exponentielle exp définie localement de l'espace tangent $T_u\mathcal{U}$ dans un voisinage $\mathcal{V}(u)$ de la façon suivante (voir les références données dans l'annexe A pour plus de détails).

$$\forall v \in T_u\mathcal{U}\,,\; exp(v) = \gamma_v(1)\,,$$

où γ_v est l'unique géodésique passant par u en 0 avec une vélocité v. exp réalise un difféomorphisme d'un voisinage de 0 dans $T_u\mathcal{U}$ vers $\mathcal{V}(u)$. On peut ainsi définir les coordonnées riemanniennes normales dans $\mathcal{V}(u)$ comme les coordonnées des images réciproques par exp dans une base orthonormée de $T_u\mathcal{U}$ (par exemple la base canonique $(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_d})$). Les coordonnées normales possèdent l'intéressante propriété de conserver les distances radiales :

$$\forall w, w' \in \mathrm{Im}(\gamma_v) \cap \mathcal{V}(u)\,,\; \rho(v, v') = d(exp^{-1}(w), exp^{-1}(w'))\,.$$

Cela n'est vrai que localement, bien que pour un certain nombre de variétés on puisse largement étendre le voisinage autour de $u \in \mathcal{U}$. C'est d'ailleurs ce que tentent de faire les méthodes de ce paragraphe, en cherchant un système global de coordonnées normales.

Estimation des directions géodésiques

Comme le titre l'indique, cette première approche détermine les coordonnées normales en suivant la géodésique dans la bonne direction depuis le centre de la carte. Plus exactement, Brun et al. [50] déterminent d'abord un point origine y_0 parmi les y_i, puis calculent les coordonnées de ses proches voisins par une ACP locale dans $\mathcal{V}(y_0)$. Ils évaluent ensuite toutes les distances géodésiques (par l'algorithme de Dijkstra dans le graphe de voisinage) de $\mathcal{V}(y_0)$ vers les y_i, afin d'interpoler les fonctions

$$\begin{array}{rccc} f_i : & \mathcal{V}(y_0) & \to & \mathbb{R}^+ \\ & y & \mapsto & \rho^2(y, y_i)\,, \end{array}$$

permettant d'approcher leurs dérivées partielles en $y = y_0$. Ces dernières correspondent aux directions δ_i (dans $T_{y_0}\mathcal{U}$) des géodésiques passant par y_0 et y_i. Les coordonnées normales de y_i sont alors naturellement approximées par $\rho(y_0, y_i)\frac{\delta_i}{\|\delta_i\|}$. Notons que celles-ci ne sont pas calculées de façon incrémentale.

À cause de l'interpolation dans $\mathcal{V}(y_0)$ cette méthode nécessite des données densément échantillonnées (2000 points dans les exemples des auteurs), et n'est donc pas facilement applicable au cas test de cette thèse. C'est pourquoi nous préférons l'approche suivante dans le package R.

Riemannian Manifold Learning (RML)

L'objectif de l'algorithme de Lin et al. [213] est encore une fois d'approximer les coordonnées riemanniennes normales, mais sans passer par l'évaluation des dérivées de la fonction distance. L'idée consiste plutôt à les déterminer de proche en proche, celles-ci étant guidées par des contraintes locales assez naturelles, visant à conserver la forme des voisinages. L'algorithme RML peut se décrire comme suit, une fois le graphe de visibilité construit comme indiqué au paragraphe 1.1.3.1. Afin de ne pas alourdir les notations les cardinaux des voisinages sont pris égaux à k, mais ceux-ci dépendent du point.

1. **Détermination de l'origine.** Pour i allant de 1 à n chercher la distance géodésique maximale de y_i à un autre élément ; choisir y_0 minimisant cette distance maximale.
2. **Estimation des coordonnées autour de l'origine.** Poser $z_0 = 0$, puis, notant $y_{i_1}, \ldots, y_{i_k}$ les voisins de y_0 : $z_{i_j} = \frac{\|y_0 - y_{i_j}\|}{\|a_{i_j}\|} a_{i_j}$, a_{i_j} étant le vecteur des coordonnées ACP locales de y_{i_j}.
3. **Estimation des autres coordonnées.** Pour y_i hors du voisinage de l'origine, poser y_p le prédécesseur de y_i sur un plus court chemin de y_0 à y_i dans le graphe. Notant $y_{i_1}, \ldots, y_{i_k}$ les voisins de y_p, on cherche alors à conserver les angles :

$$\forall j = 1, \ldots, k \;\; \cos \widehat{z_{i_j} z_p z_i} \simeq \cos \widehat{y_{i_j} y_p y_i}\,,$$

sous la contrainte $\|z_i - z_p\| = \|y_i - y_p\|$. Tout ceci est illustré sur la figure 3.16.

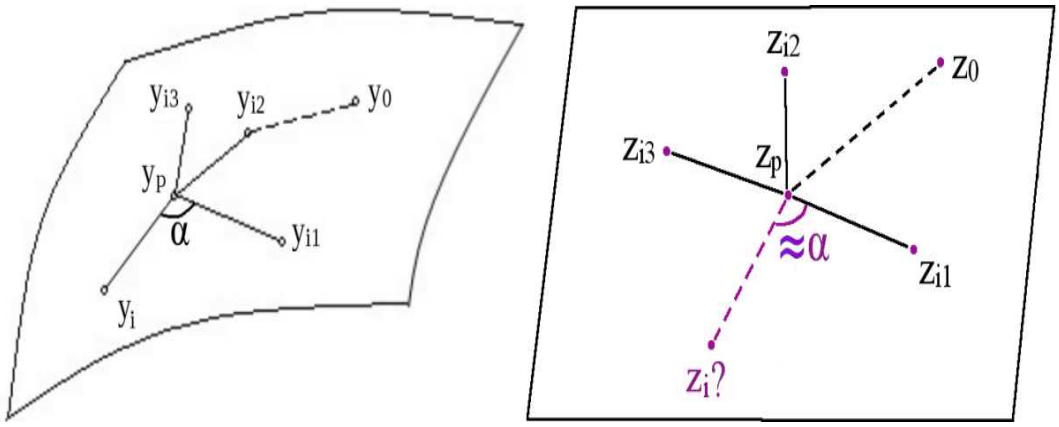

Figure 3.16: Angle α formé par y_i, son prédécesseur y_p et un voisin de ce dernier à gauche ; détermination de z_i dans $\mathcal{E}_R$ à droite.

Notons que l'algorithme de Dijsktra peut retourner en plus des distances géodésiques approchées le degré d'éloignement d'un point par rapport à l'origine, c'est-à-dire le nombre d'arêtes n_i sur le plus court chemin allant de y_0 à y_i. Ce dernier problème d'optimisation sous contraintes a pour seule inconnue z_i si le graphe est parcouru en largeur relativement aux valeurs n_i ; il se ramène alors à la résolution numérique d'une équation à une variable assez simple (Lin et al. [213]). Une tentative d'amélioration de l'estimation des distances géodésiques en autorisant le prédécesseur d'un point à varier est décrite par Lin et Zha [212]. Nous ne l'avons cependant pas implémentée, celle-ci ne semblant pas jouer un rôle important dans les performances de l'algorithme. En effet on utilise surtout les distances géodésiques locales, ces dernières étant approximées par les distances euclidiennes dans tous les cas.

3.2.3.3 Réduction de dimension multi-cartes

Toutes les méthodes précédentes échouent lorsque la variété $\mathcal{U}$ est fermée (donc quand elle "ressemble" à une sphère). Il semble alors naturel de chercher non pas une unique carte, mais un atlas − si possible de cardinal minimal. Il serait alors nécessaire d'ajouter une étape de classification supervisée dans le modèle final, similaire à celle de l'étape de clustering.

La méthodologie pourrait être la suivante.

1. Déterminer une origine y_0 comme dans l'algorithme RML.
2. Pour chaque couple d'éléments y_i, y_j contenus dans un voisinage sur $\mathcal{U}$, évaluer si les géodésiques approchées s'éloignent en remontant vers y_0.
3. Tous les y_i vérifiant le test ci-dessus avec un de leurs voisins sont candidats pour être l'origine y'_0 d'une seconde carte ; on en sélectionne alors un (via une procédure à préciser).
4. Déterminer les domaines des deux cartes en cherchant les cellules de Voronoï de centres y_0 et y'_0 selon les distances géodésiques sur $\mathcal{U}$. Appliquer alors un algorithme de

réduction de dimension dans chaque cellule puis répéter l'opération si nécessaire.

Kambhatla et Leen [174] proposent au contraire d'utiliser un nombre conséquent de voisinages obtenus par quantification vectorielle (k-means par exemple). Cela peut être difficile à appliquer dans notre cas en vue de l'étape de régression, mais mérite néanmoins l'attention. Cette voie n'a pas été explorée, et il n'est pas garanti qu'elle apporte quelque chose au cas du code CATHARE.

3.2.4 Comparaisons des représentations

Compte-tenu des contraintes sur la méthode à utiliser (capacité de reconstruction), et par souci de limiter le nombre de paramètres nous ne testons que les approches LPcaML et RML. Les deux premiers tests correspondent à ceux utilisés par Lee et Verleysen [201], avec une modification mineure : le swisshole (swissroll privé de sa partie centrale) remplace le swissroll. L'autre jeu de données consiste en une boite ouverte, difficile à représenter en $2D$ (même pour un humain). Ces deux structures de $\mathbb{R}^3$ sont représentées respectivement sur les figures 3.17 et 3.20.

Tout comme aux chapitres précédents nous appliquons une transformation aux coordonnées tridimensionnelles pour obtenir des fonctions (se rapprochant de l'objet d'étude dans cette thèse).

$$f_{\alpha,\beta,\gamma}(t) = \alpha \cos t + \beta \sin t + \gamma \sin 2t \,.$$

Celles-ci servent d'entrées aux algorithmes une fois discrétisées (sur 200 points). La taille d'un voisinage dans LPcaML est fixée à $\lceil \sqrt{n} \rceil$ où n est le nombre d'échantillons. α et les paramètres de RML sont fixés aux valeurs par défaut du package R.

Il existe des heuristiques de comparaisons pour estimer les qualités des représentations: Lee et Verleysen [202] en rappellent un certain nombre et en présente une basée sur les rangs des voisins des éléments y_i. Nous préférons cependant nous contenter d'évaluations visuelles, suffisantes à ce stade.

3.2.4.1 Swisshole et boîte ouverte

Ces deux benchmarks sont illustrés respectivement sur les figures 3.17 et 3.20, les données en $3D$ à gauche et les fonctions correspondantes à droite (non bruitées). Les représentations obtenues par l'algorithme LPcaML sur le swisshole sont visibles (avec et sans bruit) sur la figure 3.18, tandis que ceux de RML se trouvent sur la figure 3.19. Enfin, les figures 3.21 et 3.22 montrent respectivement les représentations obtenues avec LPcaML et RML sur la boîte ouverte. Comme au chapitre précédent le bruit blanc appliqué est d'écart-type 0.1. Tous les résultats sont affichés avec $n = 600$, les jeux de données étant uniformément échantillonnés.

A - Swisshole

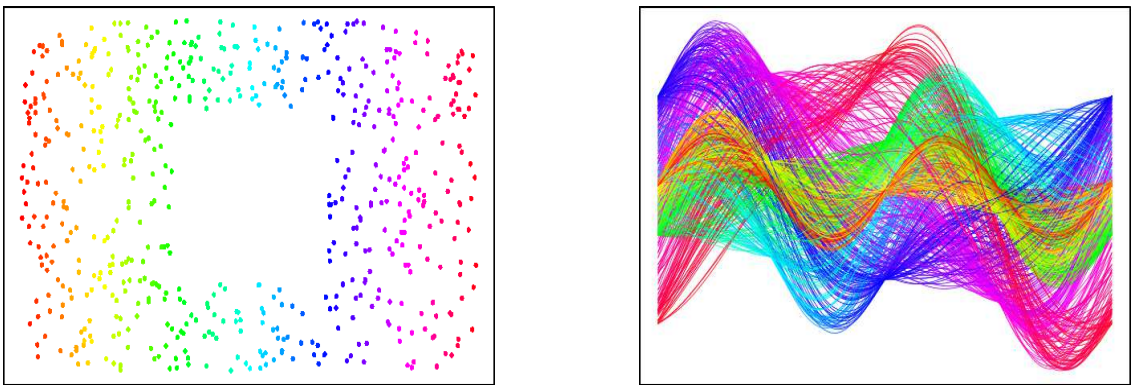

Figure 3.17: Swisshole $3D$ à gauche, swisshole fonctionnel à droite.

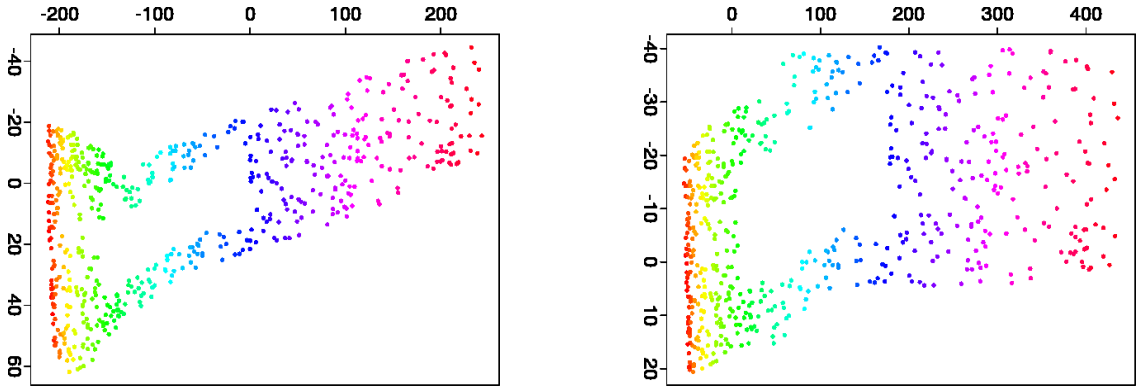

Figure 3.18: Représentation obtenue par LPcaML sur le swisshole fonctionnel, sans bruit à gauche, avec bruit à droite.

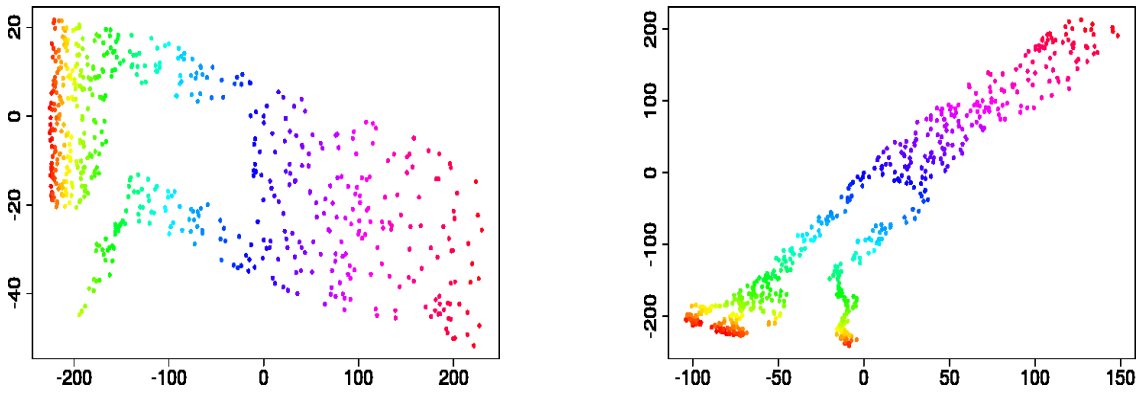

Figure 3.19: Représentation obtenue par RML sur le swisshole fonctionnel, sans bruit à gauche, avec bruit à droite.

Les deux méthodes semblent comprimer les distances à gauche et les dilater à droite. Cela provient du paramétrage du swisshole : $(\alpha \cos \alpha, \alpha \sin \alpha, \beta)$ avec $\alpha \in [0, 3\pi]$, $\beta \in [0, 4.5]$, qui est "effacé" sur la figure 3.17, cette dernière représentant β versus α. RML produit un résultat étrange sur une petite bande dans le coin inférieur gauche, très probablement lié aux spécificités de la méthode. Cela confirme sa plus grande sensibilité à la qualité de

l'échantillon par rapport à LPcaML, dont le résultat est satisfaisant sur ce premier exemple, malgré de légères erreurs de directions.

B - Boîte ouverte

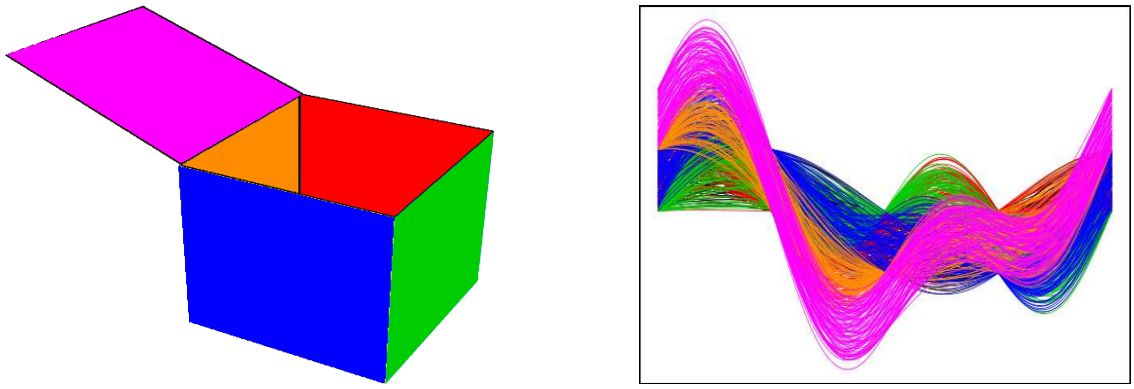

Figure 3.20: Boîte ouverte $3D$ à gauche, fonctionnelle à droite.

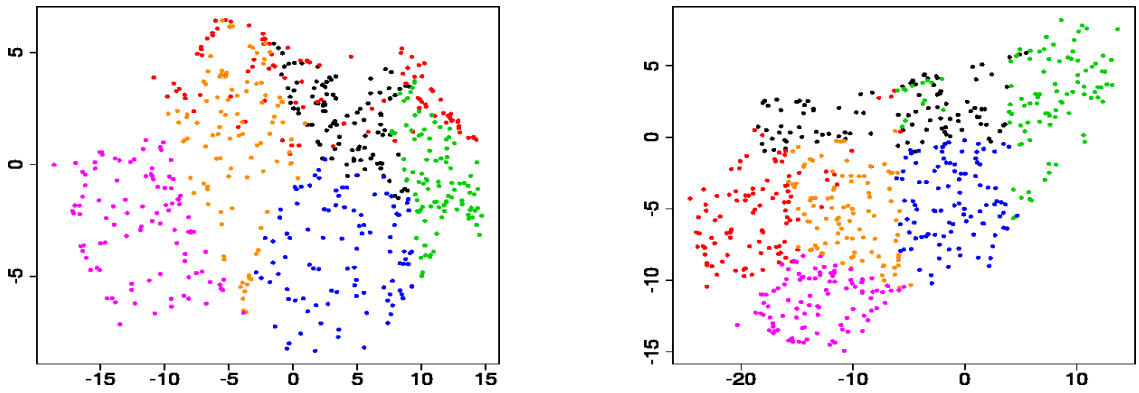

Figure 3.21: Représentation obtenue par LPcaML sur la boîte ouverte, sans bruit à gauche, avec bruit à droite.

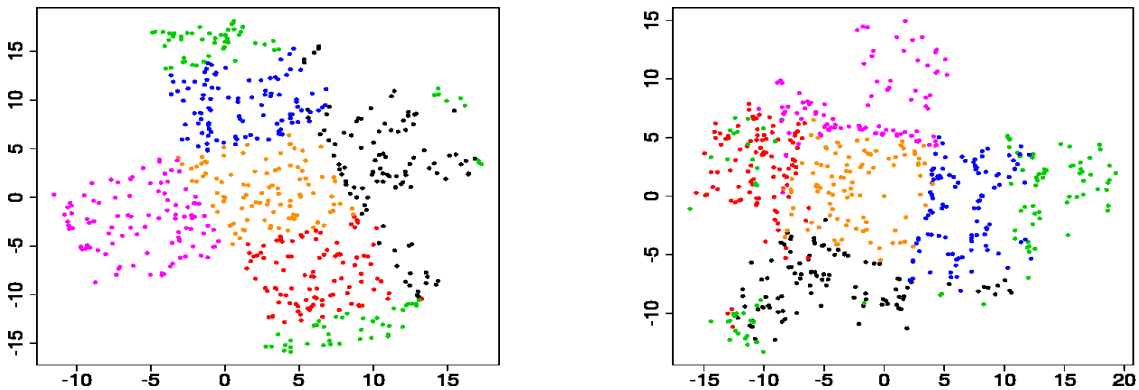

Figure 3.22: Représentation obtenue par RML sur la boîte ouverte, sans bruit à gauche, avec bruit à droite.

LPcaML sacrifie systématiquement la représentation d'une face : celle en rouge à gauche sur la figure 3.21, et celle en vert à droite, ces dernières se trouvant superposées aux autres

et déformées. C'est compréhensible compte-tenu de l'impossibilité à projeter la boîte en $2D$ sans distorsions. Hormis cette observation les résultats respectent la disposition spatiale des faces, qui ne sont pas trop déformées. Avec la méthode RML la boîte est correctement dépliée et les distances au sein de chaque face semblent conservées, sauf pour la verte qui se retrouve répartie en trois endroits sur la carte Ceci est une conséquence directe de la façon dont l'algorithme procède, la face verte étant atteinte depuis trois géodésiques différentes, passant par trois faces distinctes. Des déformations relativement importantes apparaissent en présence de bruit, la méthode n'étant pas prévue pour ça, mais l'aspect global est cohérent – sauf encore une fois pour la face en vert.

3.2.4.2 Jeux de données CATHARE

Les deux jeux de données de ces derniers tests sont les courbes en sortie du code CATHARE, déjà présentées au chapitre précédent et rappelées ici sur la figure 3.23. Le premier comporte 100 fonctions et se divise naturellement en deux clusters indiqués en rouge et noir. Le second contient 600 courbes, et est présenté divisé arbitrairement en quatre classes (via un k-means). Les représentations $2D$ obtenues respectivement avec les méthodes ACP, LPcaML et RML sont indiquées sur les figure 3.24, 3.25 et 3.26 . Les trois semblent cohérentes, séparant les classes de façon satisfaisante malgré la forme repliée du groupe en rouge pour CATHARE II. Ainsi les approches non linéaires ne semblent pas améliorer clairement les représentations ; leurs atouts seront présentés au chapitre applicatif final.

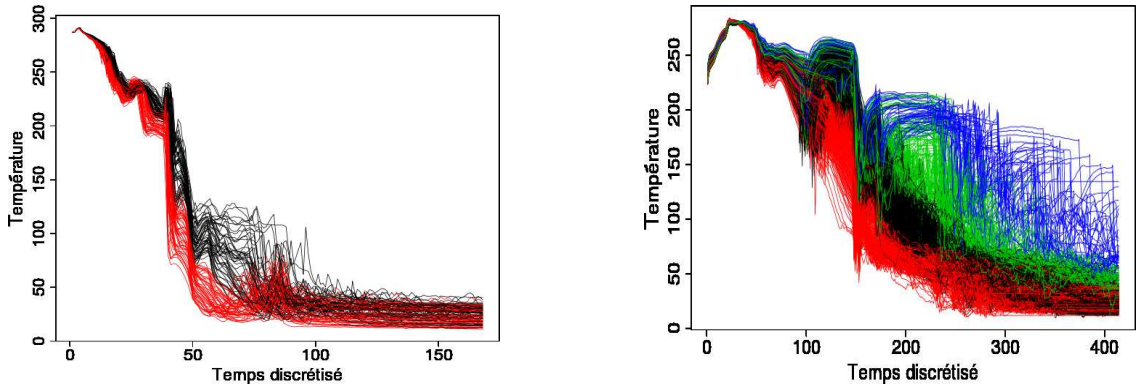

Figure 3.23: Courbes en sortie des jeux de données CATHARE I (à gauche) et CATHARE II (à droite), avec respectivement 2 et 4 clusters.

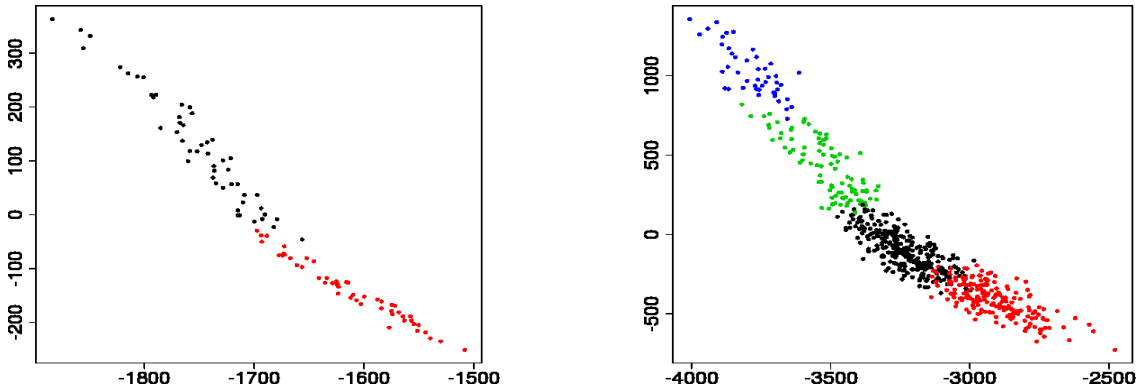

Figure 3.24: Représentation obtenue par ACP sur CATHARE I à gauche, CATHARE II à droite.

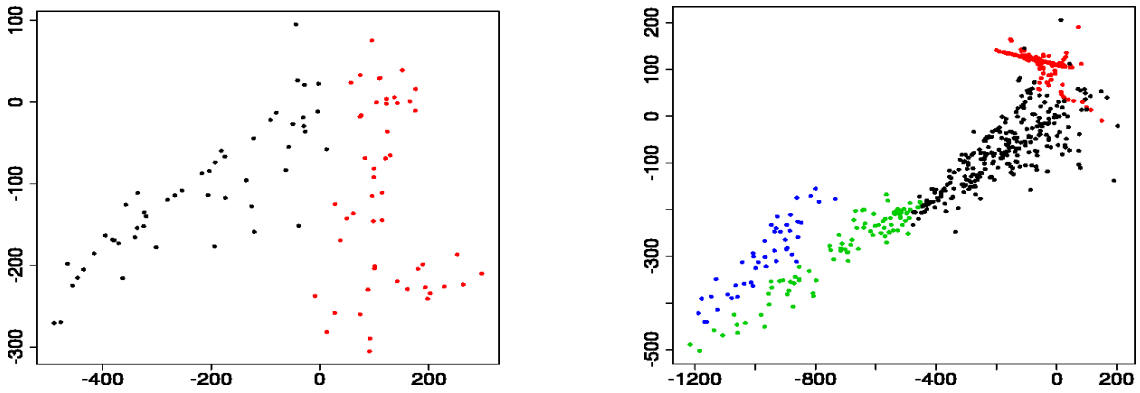

Figure 3.25: Représentation obtenue par LPcaML sur CATHARE I à gauche, CATHARE II à droite.

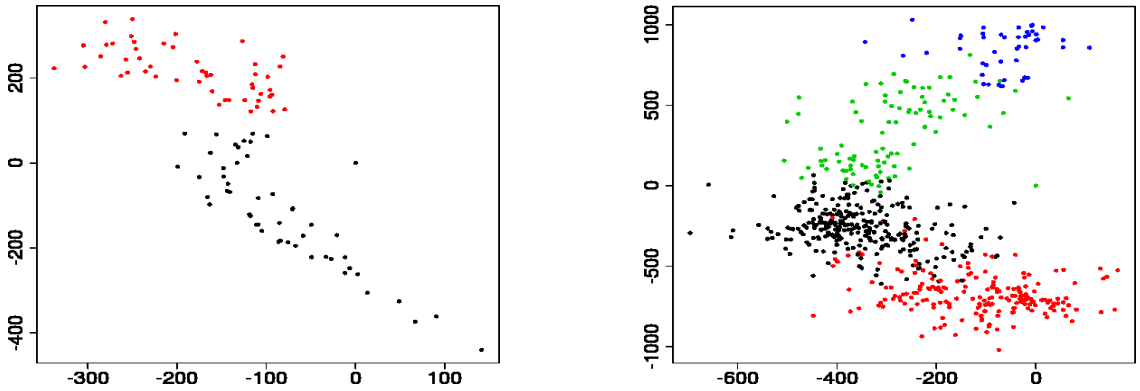

Figure 3.26: Représentation obtenue par RML sur CATHARE I à gauche, CATHARE II à droite.

3.3 Conclusion

Nous avons passé en revue plusieurs types de méthodes estimant la dimension d'une variété avant de nous arrêter sur d_{levi}. L'algorithme d_{sliv} a été proposé comme alternative valable, bien que surestimant légèrement la dimension sur les cas testés. Une fois la dimension d estimée, l'objectif est de trouver des représentations d-dimensionnelles pour chaque courbe y_i. C'est pourquoi nous avons effectué une bibliographie d'un certain nombre de méthodes aux motivations diverses, avant d'en choisir deux semblant bien adaptées au contexte. En effet, il faut pouvoir reconstruire une courbe suffisamment facilement à partir de sa représentation tout en ayant peu de paramètres à régler.

3.4 Perspectives

Toutes les approches introduites dans ce chapitre (à l'exception notable de celles estimant la dimension fractale) supposent les données y_i appartenant à une variété $\mathcal{U}$. Il peut cependant exister des paramètres d'entrées menant à un ensemble de fonctions solutions fractales (comme l'attracteur étrange de Lorenz, illustré sur la figure 3.27). Il y a peu de risques qu'en choisissant une courbe pour chaque entrée l'ensemble de ces dernières constitue un objet fractal, mais il n'y a pas forcément plus de raisons de penser que ce dernier constitue exactement une variété. L'analyse de la structure des sorties est un problème lié à la résolution des équations différentielles partielles, intéressant bien que sortant du cadre de la thèse.

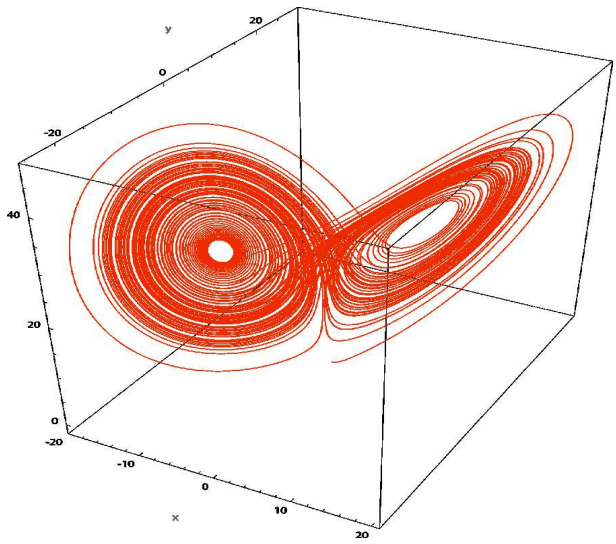

Figure 3.27: Attracteur de Lorenz (source : Wikipedia).

Supposant conserver l'approximation des sorties par une variété, il peut être intéressant d'ajouter un paramètre de décalage $h_i \in [-\tau, \tau]$ sur chaque courbe, définissant $\tilde{y}_i = y_i(. - h_i)$ (supposant pouvoir prolonger les fonctions sur $[a - \tau, b + \tau]$). On peut alors chercher les paramètres h_i qui fournissent les meilleures représentations z_i (relativement aux $\tilde{y}_i$), dans l'esprit de l'alignement des courbes effectué en parallèle du clustering par Gaffney [133] et Liu et Yang [215]. Les valeurs h_i sont ensuite à apprendre lors de l'étape de régression entrées $\rightarrow$ sorties réduites.

Chapitre 4

Régression et apprentissage de variété

Une fois la classification effectuée et la dimension réduite, il reste à relier les entrées x_i aux vecteurs z_i, avant de déterminer une application transformant un élément $z \in \mathcal{E}_R$ en une courbe. Ainsi, le modèle sera enfin fonctionnel.

Ce chapitre propose dans un premier temps un aperçu bibliographique des méthodes de régression multivariée, avec leurs avantages et inconvénients, avant d'en sélectionner une offrant un bon compromis vitesse/qualité. Des fonctions de reconstructions associées aux algorithmes terminant le chapitre 3 seront ensuite présentées, avant de conclure par deux procédures apprenant directement la relation liant les x_i aux y_i par apprentissage statistique local.

La section 4.1 est purement bibliographique et peut être omise sans gêner la compréhension, à l'exception du paragraphe 4.1.6.3 décrivant l'approche retenue. Les deux sections suivantes (4.2 et 4.3) présentent des algorithmes implémentés et utilisés dans le package R.

Plan du chapitre :

1. **Bibliographie des méthodes de régression vectorielle.**
2. **Reconstruction des courbes à partir de leurs représentations d-dimensionnelles.**
3. **Quelques approches de régression fonctionnelle directe.**

4.1 Méthodes de régression

L'objectif de cette section est l'estimation d'une fonction $g : \mathcal{E}_E \to \mathcal{E}_R$ vérifiant $g(x_i) \simeq z_i$ (dans un certain cluster dont l'indice n'est pas mentionné). Idéalement l'interpolation est exacte ($g(x_i) = z_i$), mais comme signalé dans l'introduction de la thèse ce problème est mal posé, dans le sens où il existe une infinité d'applications g n'effectuant aucune erreur sur les x_i. Beaucoup de telles fonctions n'ont pas de bonnes capacités de généralisation. C'est pourquoi on se limite systématiquement à une certaine classe de modèles pour lesquels l'optimum vérifie des conditions de régularité. À ce sujet, voir par exemple les livres de Tikhonov [315] et Alber et Ryazantseva [5].

Notons que les coefficients $z_{i1}, \ldots, z_{id}$ associés à une entrée x_i peuvent être corrélés. L'idéal serait d'utiliser une méthode tenant compte de cela en s'inspirant par exemple des travaux de Chen [60], Vazquez et Walter [326], Matías [228] et Liu et al. [214], mais cette voie n'a pas été explorée. On suppose alors une seule sortie scalaire dans tout ce qui suit (notée z ou z_i). Quelques approches sont brièvement présentées (sans chercher à être exhaustif) avant l'introduction plus détaillée de la méthode ayant servi dans les tests finaux. La plupart d'entre elles sont introduites dans l'ouvrage de Hastie et al. [157].

4.1.1 Régression linéaire

Cette technique a déjà été présentée en 0.4.1 au chapitre d'introduction ; les détails ne sont donc pas rappelés ici. Il s'agit d'une des méthodes les plus simples, modélisant la sortie z_i par une combinaison linéaire fixe des entrées x_{ij}, $j = 0, \ldots, p$ avec $x_{i0} = 1$. On cherche ainsi à minimiser le terme d'erreur $\|z_i - (1\ x_i)\beta\|$ en moyenne, avec $\beta \in \mathbb{R}^{p+1}$.

La régression linéaire peut se généraliser de plusieurs manières différentes, une des plus simples consistant à écrire $z_i \simeq f((1\ x_i)\beta)$, f étant une fonction lisse à déterminer [243; 104]. Cette approche peut être visualisée comme un perceptron [263; 235] utilisé en régression avec une transformation quelconque en sortie.

Les modèles additifs généralisés (Hastie et Tibshirani [158]) constituent une autre extension courante de la régression linéaire. Ils se mettent sous la forme

$$\hat{z} = f\left(\mu + \sum_{j=1}^{p} f_j(x_j)\right),$$

avec μ en général égal à la moyenne des données, f et les f_j étant des fonctions à estimer.

Enfin, il est possible de régulariser le modèle initial en minimisant $E + \lambda N(\beta)$ où E est l'erreur quadratique effectuée par la régression linéaire simple. Le paramètre $\lambda \geq 0$ contrôle le degré de régularisation, et peut être déterminé par validation croisée. $N(\beta)$ désigne la norme de β : $N(\beta) = \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j\|$ correspond à la à la méthode LASSO [312], tandis que $N(\beta) = \|\beta\|_2^2 = \sum_{j=1}^{p} |\beta_j\|^2$ mène à la régression "ridge". Notons qu'aucune des méthodes de ce paragraphe ne tiennent compte des interactions entre les variables d'entrée.

4.1.2 Approximation polynomiale

Les fonctions polynômes multivariées sont denses dans $\mathcal{C}(\mathcal{E}_E)$ (supposant $\mathcal{E}_E$ est égal à un produit d'intervalles compacts), donc permettent en théorie d'approcher la relation liant les entrées aux coordonnées réduites. En pratique, on choisit un polynôme $P \in \mathbb{R}[X_1, \ldots, X_p]$ puis on ajuste ses coefficients via les relations $P(x_{i1}, \ldots, x_{ip}) \simeq z_i$. La résolution est similaire à celle de la régression linéaire (Press et al. [255], chapitre 15). Deux critiques peuvent être émises : le modèle ainsi constitué est paramétrique, donc difficile à automatiser, et présente un phénomène d'oscillation désagréable quand le degré est élevé (Runge [270] ; de Boor [85], chapitres 1 et 2).

Les fonctions splines sont des polynômes par morceaux. Elles visent à éliminer le dernier

défaut mentionné tout en limitant l'erreur réalisée en cas de mauvaise paramétrisation. La définition d'une fonction spline $1D$ de domaine $[\alpha, \beta]$ est donnée par exemple dans le livre de Schumaker [284].

Définition 4.1.1 *Soit q un entier positif, et $t_1, \ldots, t_m$ une suite croissante de m points dans $]\alpha, \beta[$, on appelle* spline polynomiale *d'ordre q ayant pour nœuds simples les points $t_1, \ldots, t_m$ toute fonction s de $[\alpha, \beta]$ dans $\mathbb{R}$ telle que :*

- *s est continûment dérivable jusqu'à l'ordre $\max(0, q-1)$;*
- *la restriction de s aux intervalles inter-nœuds $[\alpha, t_1], \ldots, [t_j, t_{j+1}], \ldots, [t_m, \beta]$ coïncide avec un polynôme de degré inférieur ou égal à q.*

Il est naturel de choisir pour les nœuds t_j les points à interpoler, mais on peut aussi chercher une spline de lissage et relâcher cette contrainte. Une spline telle que décrite ci-dessus est l'unique fonction f minimisant l'expression suivante

$$E = \frac{1}{n}\sum_{i=1}^{m}(z_i - f(t_i))^2 + \lambda \int_{\alpha}^{\beta} (f^{(q-1)}(t))^2 dt \,,$$

$\lambda > 0$ étant un paramètre de régularisation.

Friedman [130] introduit la méthode MARS pour "Multivariate Adaptive Regression Splines", les applications f_j s'écrivant comme des produits tensoriels de splines univariées d'un des types suivants :

$$x \mapsto 1,\ x^q \text{ ou } (x-t)_+^q \,.$$

Ces dernières sont adaptées aux données à l'aide d'une procédure récursive similaire à celle utilisée pour les arbres de régression. Il existe d'autres façons de généraliser les splines en dimension supérieure, notamment via les fonctions radiales de base [281]. Enfin, Wahba [330] effectue une présentation théorique des splines (de lissage), reformulant en particulier le problème d'optimisation dans un espace hilbertien à noyau reproduisant.

4.1.3 Méthodes à noyaux

Les deux approches de ce paragraphe utilisent le formalisme des espaces de Hilbert fonctionnels $\mathcal{H} \subset \mathcal{F}(\mathcal{X}, \mathbb{R})$ ($\mathcal{X}$ étant un ensemble quelconque) à noyau reproduisant (RKHS), c'est-à-dire pour lesquels il existe une application dite "noyau" $N : \mathcal{X}^2 \to \mathbb{R}$ continue, symétrique et définie positive telle que les propriétés suivantes soient vérifiées.

1. $\forall x \in \mathcal{X}\ \ N(x, .) \in \mathcal{H}$;
2. $\forall f \in \mathcal{H}\ \forall x \in \mathcal{X}\ \ f(x) = \langle f, N(x, .)\rangle$.

N "définie positive" signifie que pour tous $x^{(1)}, \ldots, x^{(m)} \in \mathcal{X}$ la matrice de terme général $N(x^{(i)}, x^{(j)})$ est définie positive.

Aronszajn [13] montre qu'à $\mathcal{X}$ fixé il y a une correspondance bijective entre les fonctions noyau N et les RKHS. Ainsi, pour travailler dans un espace hilbertien fonctionnel "dual"

il suffit de se donner un noyau N. La fonction $x \mapsto N(x, .)$ étant clairement injective, une représentation alternative de $\mathcal{X}$ est obtenue via les applications $N(x, .)$ − que l'on note N_x. Cela permet de transporter un problème d'optimisation dans un espace mieux adapté (sous réserve de bien choisir le noyau).

4.1.3.1 Support Vector Regression

Cette approche (SVR) se base sur le modèle linéaire $z_i \simeq \langle \omega, x_i \rangle + \beta$, avec $\omega \in \mathbb{R}^p$. Celui-ci étant trop limité, on choisit de calculer les produits scalaires dans un RKHS $\mathcal{H}$ après avoir défini un noyau N sur $\mathcal{E}_E$:

$$z_i \simeq \langle N_\omega, N_{x_i} \rangle + \beta \,.$$

La régularité de la fonction objectif est contrôlée par la norme de N_ω. Il est alors naturel de chercher à minimiser cette norme sous des contraintes de (quasi) interpolation aux points x_i. Ce problème d'optimisation est résolu en utilisant exclusivement les produits scalaires $N(x, x')$, N étant en général choisi dans une bibliothèque de noyaux admissibles (voir par exemple les chapitres 2 et 13 du livre de Schölkopf et Smola [282]). L'estimateur obtenu est de la forme

$$\hat{z} = \sum_{i=1}^{n} N(x, x_i)\alpha_i + \beta \,.$$

Cette dernière formulation est assez intuitive : on prédit z en effectuant une somme pondérée de quantités associées aux images des x_i proches (en supposant N décroissant en fonction de la distance entre ses deux arguments). Pour une introduction plus complète, voir par exemple le livre de Cristianini et Shawe-Taylor [78] et le tutoriel de Smola et Schölkopf [296].

4.1.3.2 Krigeage

Cette méthode − historiquement introduite par Krige [190] et Matheron [227] − s'appuie sur une intuition différente : les sorties z_i sont supposées être les réalisations d'un champ aléatoire (spatial) indexé par les entrées $\in \mathcal{E}_E$. Plus précisément, on écrit

$$z = f(x) + Z_x \,,$$

avec f fonction de régression déterministe sur les (x_i, z_i), en général choisie simple (constante ou régression linéaire), et Z un processus du second ordre à moyenne nulle. En effectuant quelques hypothèses simplificatrices sur la covariance de Z, on peut estimer ses paramètres par maximum de vraisemblance [339; 223], puis calculer $\hat{z}$ pour une nouvelle entrée.

$\hat{z} - f(x)$ s'exprime en fait linéairement par rapport aux sorties z_i, et est égal à l'espérance de la projection de la variable aléatoire Z_x sur l'espace engendré par les Z_{x_i} (voir la thèse de Vazquez [325]).

$$\hat{z} = f(x) + \sum_{i=1}^{n} \lambda_i(x) z_i \,.$$

D'une manière similaire aux paramètres α_i d'un SVR, les coefficients des z_i s'écrivent en fonction des valeurs $\mathrm{Cov}(Z_x, Z_{x'}) = N(x, x')$ uniquement, N étant un noyau sur $\mathcal{E}_E$. L'espace de représentation est alors l'espace de Hilbert des variables aléatoires Z_x. On peut pousser l'analogie plus loin en montrant que le modèle des processus gaussiens − cas particulier du

krigeage – est équivalent à la variante des SVR appelée "Kernel Ridge Regression" (Cristianini et Shawe-Taylor [78], chapitre 6).

D'un certain point de vue le krigeage est moins général que les machines à vecteurs support, ces dernières ne recherchant pas un prédicteur linéaire en les sorties y_i. Cette spécialisation dispose cependant d'un cadre théorique bien étudié pour l'optimisation des paramètres (maximisation de la vraisemblance), et possède l'avantage de fournir des intervalles de confiance sur les sorties. Toutefois, ces derniers peuvent varier beaucoup d'un modèle de covariance à un autre, comme illustré sur la figure 4.1 empruntée à Tom Minka (lien). Nous n'avons pas sélectionné ces deux dernières approches à cause de la difficulté à sélectionner automatiquement le bon noyau, en plus de la lenteur relative de l'optimisation des hyperparamètres en comparaison avec d'autres méthodes. Concernant la sélection du noyau mentionnons cependant les travaux de Amari et Wu [10], qui s'appuient sur des considérations topologiques pour modifier une fonction radiale de base, élargissant la marge d'un SVM. Ali et Smith-Miles [8] complètent cette dernière procédure en proposant une méthode heuristique pour sélectionner un noyau parmi quatre familles, sous la forme d'un arbre de classification.

Remarque : Fang et al. [117] proposent d'utiliser le (co)krigeage ponctuel afin de construire un métamodèle sans passer par l'étape de réduction de dimension. En plus de comporter des redondances liées à la nature fonctionnelle des données, le modèle s'avère alors très long à construire. C'est pourquoi nous n'avons pas choisi cette voie.

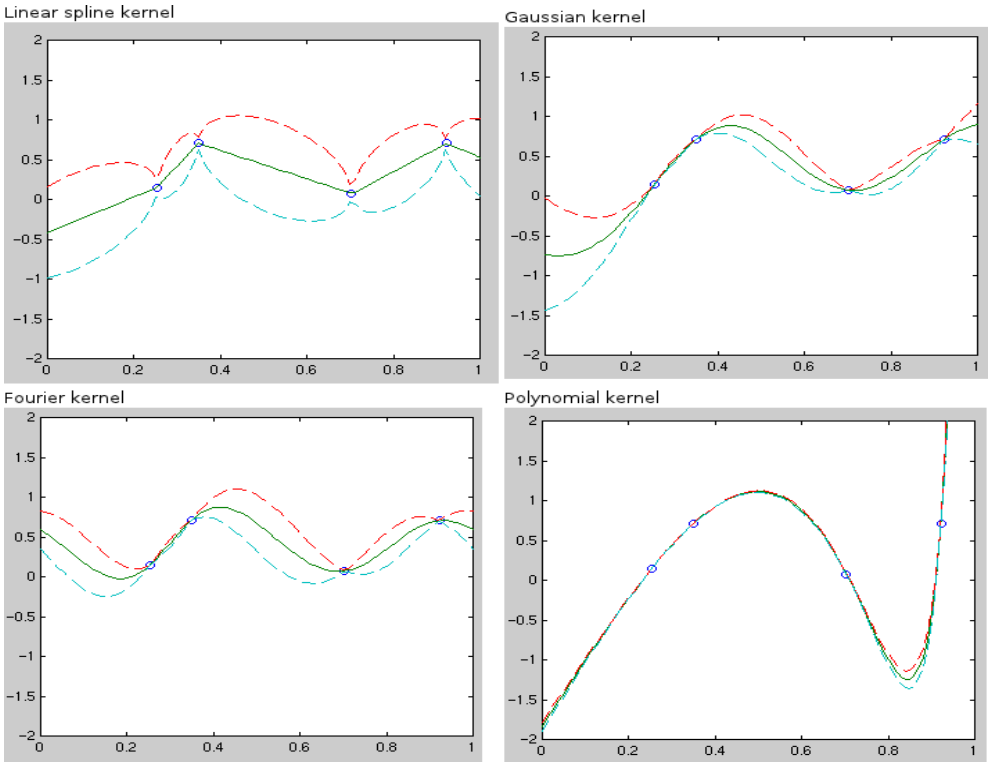

Figure 4.1: Différents intervalles de confiance (et prédictions) suivant le noyau – apprentissage sur 4 points.

4.1.4 Aggrégation d'estimateurs : boosting

L'idée du boosting consiste, comme le titre le suggère, à fusionner les prédictions d'un certain nombre d'estimateurs. Compte-tenu du coût algorithmique potentiellement élevé, ce genre d'approche est plutôt utilisé avec des prédicteurs simples, notamment les arbres de régression [44]. La méthodologie est en générale la suivante : un premier estimateur est entraîné sur un certain sous-ensemble du jeu de données, puis un second sur les résidus, puis un troisième, etc. On continue éventuellement jusqu'à observer un phénomène de surapprentissage, avant de supprimer quelques prédicteurs superflus. Ainsi, Elith et al. [114] utilisent quelques milliers d'arbres de régression afin de prédire diverses quantités liées à l'écosystème. Suivant le même schéma d'apprentissage, Tutz et Binder [320] étudient le boosting d'estimateurs du type ridge regression.

Il est également possible de se limiter à un petit groupe d'estimateurs plus sophistiqués, en appliquant une méthodologie différente. Au lieu d'entraîner des estimateurs sur les résidus successifs de tous les exemples, ces approches divisent les données en quelques sous-ensemble sur lesquels des algorithmes plus spécialisés sont appliqués. Jacobs et al. [169] utilisent $m > 1$ réseaux de neurones experts reliés à un sélecteur qui choisit le meilleur réseau en fonction de l'entrée présentée. Avnimelech et Intrator [18] entraîne d'abord un premier estimateur sur toutes les données, puis un second sur les exemples difficiles à apprendre, et un troisième reprenant les exemples mal appris par les deux premiers. Ainsi les prédicteurs successivement appliqués sont de plus en plus spécialisés.

On peut considérer le clustering comme une aggrégation d'estimateurs au sens du paragraphe précédent. Concernant l'apprentissage itéré sur les résidus, nous avons effectué une tentative peu concluante constituée de deux étapes (avec deux schémas de réduction de dimension différents). L'idée est tout de même à approfondir, constituant une possible future piste de recherche pour notre application.

4.1.5 Réseaux de neurones

Un réseau de neurones est, comme son nom l'indique et par analogie avec les neurones biologiques, un ensemble de cellules interconnectées transformant un signal. On peut le représenter comme sur la figure 4.2 (empruntée ici à Kjell Magne Fauske), l'information transitant de gauche à droite.

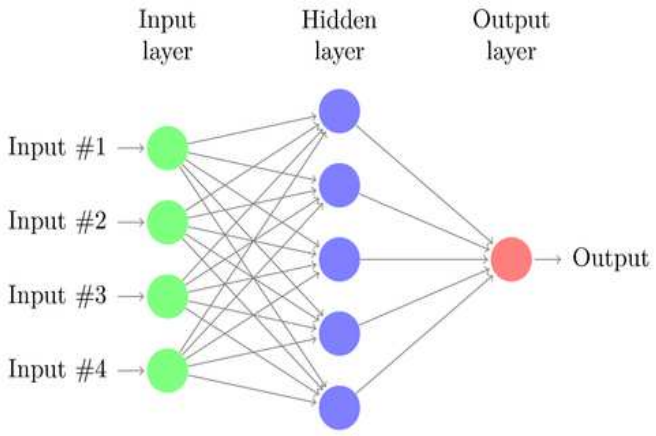

Figure 4.2: Schéma d'un réseau de neurones à trois couches

Mathématiquement, la fonction réalisée par un réseau à trois couches s'écrit :

$$g(x) = \sum_{j=1}^{m} \beta_j \sigma(\langle x, \alpha_j \rangle + \theta_j) \,,$$

avec $\alpha_1, \ldots, \alpha_m \in \mathbb{R}^p$ les vecteurs des poids des connections des composantes des entrées vers les m neurones de la couche centrale (dite "cachée"), et $\beta_1, \ldots, \beta_m \in \mathbb{R}$ les poids des connections de la couche cachée vers la sortie. σ est une fonction non linéaire dite "d'activation" (par analogie biologique), en général choisie sigmoïdale. Une transformation finale est envisageable sur le résultat, mais Cybenko [79; 80] a montré qu'un tel réseau à trois niveaux est suffisant pour approximer n'importe quelle fonction continue (et bornée).

Les principales difficultés d'automatisation des réseaux de neurones résident dans la détermination du nombre de neurones sur la couche cachée, et l'apprentissage des paramètres du réseau (α, β, θ). Ils sont cependant appliqués dans plusieurs domaines, dont la reconnaissance de forme [31] et la commande de systèmes dynamiques [108]. Pour une introduction plus complète, voir par exemple les livres de Blayo et Verleysen [33] et Krose et van der Smagt [191].

4.1.6 Régression par "poursuite de projection"

Les approches qui vont suivre ont en commun d'effectuer la régression sur des sous-espaces de dimension 1 en entrée, obtenus par projection le long d'axes "intéressants". La somme de ces prédicteurs constitue alors une estimation de la sortie $z = g(x)$. Ces techniques diffèrent principalement dans la façon dont les axes sont choisis. Ce type de régression peut être vu comme une extension des modèles additifs généralisés [158].

4.1.6.1 Régression sur composantes principales

Les composantes de l'ACP sur la matrice X des entrées étant notées $\zeta_1, \ldots, \zeta_m$ (avec $m = \min(p, n)$), la régression sur ces composantes consiste à apprendre la relation $\langle x_i, \zeta_1 \rangle \mapsto z_i$, puis $\langle x_i, \zeta_2 \rangle \mapsto z_i - \langle x_i, \zeta_1 \rangle$, etc. En pratique on s'arrête lorsque le résidu est inférieur à un seuil donné, ou bien quand la variance expliquée par les composantes principales est

suffisamment proche de 100%. L'estimateur final s'écrit ($q \leq m$) :

$$g(x) = \sum_{j=1}^{q} f_j(\langle x, \zeta_j \rangle) \,,$$

les applications f_j pouvant être déterminées avec des splines de lissage par exemple.

Remarque : les réseaux de neurones codent des fonctions similaires plus simples ($f_j \equiv \beta_j \sigma$), mais ne contraignent pas les projections dans l'espace de départ.

4.1.6.2 Projection sur les "structures latentes"

Cette méthode − couramment appelée PLS pour "Partial Least Squares" − fut introduite par Wold [342; 343], puis enrichie par un certain nombre de variantes de différents auteurs [86; 110; 27; 328]. Celle-ci s'étend naturellement au cas multi-sorties (régression "PLS2"), mais nous présentons ici une variante inspirée par la version appelée "PLS1", opérant sur des couples $(x_i, z_i) \in \mathcal{E}_E \times \mathbb{R}$. L'idée principale consiste à choisir les directions de projections de façon à ce qu'elles soient un maximum corrélées avec la sortie. Contrairement à l'approche précédente, les directions dans l'espace $\mathcal{E}_E$ sont ainsi optimisées pour être prédictives. L'algorithme (adapté pour suivre le schéma de cette sous-section) peut se décrire comme suit.

1. Initialiser les résidus d'approximation η_i aux valeurs z_i, et un compteur c à 1.
2. Chercher ζ_c direction de projection (normalisée) dans l'espace d'entrée pour maximiser

$$\mathrm{Cov}(X\zeta_c, \eta) = \frac{1}{n} \sum_{i=1}^{n} (\langle x_i, \zeta_c \rangle - \mu_e)(\eta_i - \mu_\eta) \,,$$

   où μ_e (resp. μ_η) désigne la moyenne des projections en entrée (resp. la moyenne des résidus).
3. Apprendre la relation $f_c : \langle x_i, \zeta_c \rangle \mapsto \eta_i$ via une spline de lissage, ou toute autre méthode.
4. Actualiser les résidus : $\eta_i \leftarrow \eta_i - f_c(\langle x_i, \zeta_c \rangle)$. Si η n'est pas assez proche de zéro, incrémenter c et revenir en 2.

La maximisation de la covariance à l'étape 2) peut s'effectuer algébriquement en recherchant les vecteurs propres d'une matrice.

Pour plus d'informations sur la régression PLS, voir le livre de Tenenhaus [310]. Les régressions MARS et PLS constituent des alternatives probablement à explorer dans un travail ultérieur, car elles remplissent le cahier des charges pour être incluses dans un métamodèle (rapides, comportant peu de paramètres tout en ayant de bonnes capacités d'approximation). Dans cette thèse nous choisissons la méthode suivante, ayant elle aussi les caractéristiques requises.

4.1.6.3 Projection Pursuit Regression

Cette méthode (PPR) recherche des directions de projection "optimales" dans l'espace d'entrée − tout comme la précédente −, mais pas dans l'espace des sorties. En contrepartie, les projections dans $\mathcal{E}_E$ ne sont pas contraintes. Le modèle s'écrit donc

$$g(x) = \sum_{j=1}^{q} f_j(\langle x, \zeta_j \rangle)\,,$$

et on cherche à optimiser à la fois les directions ζ_j et les fonctions f_j. L'algorithme de Friedman et Stuetzle [131] se déroule comme suit.

1. Initialiser les résidus scalaires $r_i \leftarrow z_i$, et un compteur $c \leftarrow 1$.
2. Rechercher la direction de projection ζ_c minimisant l'erreur quadratique moyenne (normalisée) E :
$$E = \frac{1}{\sum_{i=1}^{n} r_i^2} \sum_{i=1}^{n} (r_i - f_c(\langle x_i, \zeta_c \rangle))^2\,,$$
f_c effectuant un lissage $1D$.
3. Si E est en dessous d'un certain seuil τ, retourner le modèle à c termes obtenu.
4. Mettre à jour les résidus : $r_i \leftarrow r_i - f_c(\langle x_i, \zeta_c \rangle)$, puis incrémenter c et revenir en 2.

À l'étape 2) pour une direction de projection ζ_c donnée, f_c applique un lissage plus fort que celui généralement réalisé par les fonctions splines [331]. En effet, l'objectif n'étant pas de décrire correctement la relation $\langle x_i, \zeta_c \rangle \mapsto r_i$ à l'aide d'une seule courbe, il vaut mieux éviter une trop bonne adéquation aux données et laisser les composantes suivantes du modèle expliquer ce qui reste. Étant donnée la suite $\alpha_1, \ldots, \alpha_n$ des projections $\langle x_i, \zeta_c \rangle$ ordonnées, le lissage se déroule comme suit :

1. remplacer chaque α_i par la médiane de $\alpha_{i-1}, \alpha_i, \alpha_{i+1}$ (méthode "median of three" de Tukey [319], éliminant les outliers isolés) ;
2. estimer la variabilité des résidus par ajustement linéaire local [72], puis lisser ces variances avec une largeur de bande fixe ;
3. effectuer un lissage linéaire local de la séquence des (α_i, r_i), avec une largeur de bande variable déterminée par l'étape 2.

Il est possible de réajuster certaines fonctions de lissage à chaque nouvelle direction ζ_c ("backfitting" [129]), augmentant considérablement le coût algorithmique pour un meilleur modèle. Concernant la recherche d'une bonne direction de projection, les auteurs choisissent d'utiliser une version modifiée de la méthode de Rosenbrock [264], réinitialisant aléatoirement ζ_c lorsque le minimum local atteint n'est pas satisfaisant. Chen [57] montre que la vitesse de convergence de l'algorithme (en fonction de n) est indépendante de la dimension en entrée p, à condition que les composantes ζ_c optimales ne soient pas trop colinéaires.

La figure 4.3 illustre la fonction de Michalewicz [232] $2D$ pour $m = 1$:

$$f(u, v) = -\sin u \sin^{2m}\left(\frac{u^2}{\pi}\right) - \sin v \sin^{2m}\left(\frac{2v^2}{\pi}\right) ,$$

Le tableau 4.1 présente les Q_2 moyens obtenus (sur une base de test à 1000 éléments) en faisant varier le nombre de points n_A de 20 à 200 dans la base d'apprentissage, pour un nombre de termes T allant de 1 à 5. Rappelons que le Q_2 est une MSE normalisée, d'autant plus proche de 1 que le modèle est bon. Cet indicateur est défini précisément au paragraphe 5.2.1. On voit ainsi que 50 à 100 points suffisent sous réserve d'utiliser au moins deux à trois termes, ce qui est logique compte-tenu de la dimensionalité des données.

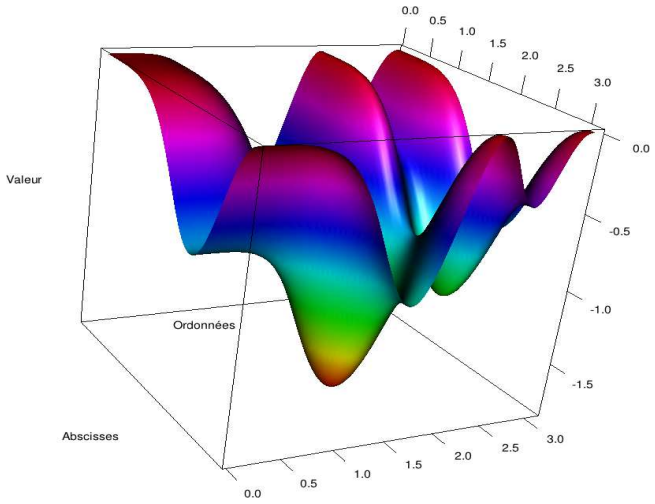

Figure 4.3: Fonction de Michalewicz $2D$, $m = 1$.

n_A \ T	1	2	3	4	5
20	0.10	0.15	0.25	0.30	0.33
50	0.20	0.50	0.70	0.75	0.80
100	0.30	0.70	0.85	0.90	0.95
200	0.35	0.85	0.95	0.97	0.99

Table 4.1: Qualité des prédictions en fonction de la taille de l'échantillon et du nombre de termes dans le modèle.

On peut comparer cet algorithme avec les réseaux de neurones de la façon suivante. Ces derniers n'optimisent que les directions ζ_j (si l'on excepte les coefficients multiplicatifs β_j), en augmentant le nombre de termes (neurones) dans le modèle si nécessaire. L'algorithme PPR adapte mieux les fonctions f_j aux données, permettant de réduire le nombre de termes. Les réseaux de neurones ont cependant l'avantage d'optimiser simultanément toutes les directions de projection. Pour une comparaison plus approfondie entre ces deux modèles, voir par exemple le rapport de Klinke et Grassmann [185].

L'algorithme PPR est utilisé par défaut dans le package R pour toutes les tâches de régression multivariée à sortie scalaire.

4.2 Reconstruction locale des courbes

En supposant les phases de clustering, réduction de dimension et apprentissage statistique réalisées, on est en mesure de prédire la représentation d-dimensionnelle d'une courbe à partir d'une entrée $x \in \mathcal{E}_E$. Afin de prédire $\phi(x)$, une dernière étape de régression est nécessaire. Celle-ci est habituellement nommée "apprentissage de variété", car déterminer la relation liant $z \in \mathcal{E}_R$ à $y \in \mathcal{U}$ revient à chercher (l'inverse d') une carte locale.

Plus spécifiquement, cette seconde section s'attache à reconstituer des courbes depuis les représentations d-dimensionnelles obtenues par les algorithmes LPcaML et RML des paragraphes 3.2.3.1 et 3.2.3.2. En effet ce sont ces méthodes qui ont été choisies dans le package R. Rappelons que LPcaML aligne des cartes obtenues par ACP locales de manière incrémentale, tandis que RML cherche à conserver les angles intra-voisinages sous une contrainte d'égalité des distances radiales. Nous appliquons ici les mêmes méthodologies, en inversant les rôles des variables.

Remarque : le processus de réduction de dimension suivi de la reconstruction ne s'assimile pas à de la compression d'information. En effet on utilise un maximum d'éléments des données initiales, notamment la matrice des sorties discrétisées si nécessaire.

4.2.1 Inversion de LPcaML

L'algorithme LPcaML est supposé avoir été appliqué à un ensemble de fonctions discrétisées $y_1, \ldots, y_n \in \mathbb{R}^D$. Nous disposons donc des éléments suivants en plus de la base d'apprentissage constituée des triplets (x_i, y_i, z_i) :

- m voisinages $\mathcal{V}_j$ de cardinaux k, se chevauchant et recouvrant les indices $\{1, \ldots, n\}$;
- m bases ACP locales, données sous la forme de md fonctions discrétisées où d est la dimension des sorties réduites ;
- m transformations affines (matricielles) T_j de tailles $(d+1) \times d$, reliant les coordonnées ACP locales aux z_i.

Une sortie réduite z venant d'être prédite (par l'algorithme PPR appliqué sur chaque composante), il s'agit d'estimer la courbe y codée par z. L'hypothèse suivante

Les voisins de z dans $\mathcal{E}_R$ correspondent aux voisins de y dans $\mathcal{C}([a,b])$.

est nécessaire pour appliquer la bonne transformation. Elle est toujours vérifiée si les représentations z_i ne se superposent pas. Notons $z_{i_1}, \ldots, z_{i_k}$ les k voisins de z dans l'ensemble d'entraînement, k étant précisé ultérieurement. Deux stratégies sont envisageables :

1. assigner z au voisinage $\mathcal{V}$ dont les indices sont majoritaires dans $\{i_1, \ldots, i_k\}$;
2. considérer les proportions d'indices présents dans chaque $\mathcal{V}_j$ comme des probabilités d'appartenance.

Si la seconde option est retenue, alors $\hat{y} = R(z)$ s'écrit comme une somme pondérée d'estimations calculées avec différents voisinages. Cette approche a été testée sans succès, probablement à cause de l'aspect assez subjectif de la détermination des poids. C'est pourquoi on choisit simplement la règle du k plus proche voisin afin d'estimer le voisinage $\mathcal{V}$ contenant z. k (le nombre d'éléments dans $\mathcal{V}$) est estimé par validation croisée pendant la classification supervisée.

Rappelons à présent le lien existant entre les coordonnées z_i (matrice Z) et celles de l'ACP locale c_i (matrice C) dans le voisinage $\mathcal{V}$:

$$Z = \tilde{C}T\,,$$

où $\tilde{C}$ désigne la matrice C concaténée à une colonne de 1, T étant une (matrice de) transformation affine, de taille $(d+1) \times d$. En écrivant $\tilde{C} = (\mathbf{1}^k\ C)$ (en colonnes) et $T = \begin{pmatrix} T_0 \\ T_c \end{pmatrix}$ (en lignes), on obtient pour $i \in \mathcal{V}$:

$$z_i - T_0 = {}^t c_i T_c\,.$$

Ainsi on estime d'abord les coordonnées ACP de z par

$$\hat{c} = (z - T_0)T_c^+\,,$$

T_c^+ désignant l'inverse généralisé de T_c (comme défini au paragraph 1.2.2.3). Il suffit ensuite d'étendre ces coefficients sur la base ACP locale pour obtenir $\hat{y}$.

Remarque : concernant le choix du paramètre α, il n'est pas possible d'utiliser la validation croisée directement sur la classification supervisée, car il faut tenir compte de la qualité des représentations, typiquement optimale pour α ni trop réduit ni trop élevé. L'automatisation de la détermination de cette variable nécessiterait une étude plus poussée.

4.2.2 Inversion de RML

Comme au paragraphe précédent on suppose les voisinages conservés dans l'espace de représentation $\mathcal{E}_R$. L'algorithme RML s'effectuant en deux étapes, on distingue deux situations pour la reconstruction :

1. z **est "proche" de** z_0 : les éléments de $\mathcal{V}_0$ sont majoritaires parmi les indices des voisins de z dans l'ensemble $\{z_i\}$;
2. z **est "loin" de** z_0 : les éléments de $\mathcal{V}_0$ sont minoritaires dans ce dernier ensemble,

$\mathcal{V}_0$ désignant l'ensemble des indices des courbes dont les coordonnées z_i ont été calculées par projection (normalisée) sur l'espace tangent en y_0. Ce sont donc les "voisins" de y_0. Rappelons que ce dernier élément constitue l'origine du graphe des plus courts chemins, et a pour coordonnées réduites $z_0 = (0, \ldots, 0)$. Comme pour LPcaML on pourrait obtenir $\hat{y}$ comme une somme pondérée des estimations des cas 1 et 2, mais ce n'est pas vraiment utile. Les paragraphes suivants détaillent respectivement les modalités 1 et 2.

4.2.2.1 Voisin de l'origine z_0

Si z_i est suffisamment proche de $z_0 = 0$, alors il est déterminé comme suit dans l'algorithme RML.

$$z_i = \alpha_i C_0(y_i - y_0)\,,$$

C_0 désignant les d premières fonctions de la base ACP(F) en lignes. Le coefficient α_i effectue la normalisation pour assurer $\|z_i\| = \|y_i - y_0\|$, et est donc égal à $\dfrac{\|y_i - y_0\|}{\|C_0(y_i - y_0)\|}$.

Si la plupart des voisins de z sont dans $\mathcal{V}_0$ on peut supposer qu'il s'écrit aussi sous la forme $\alpha C_0(y - y_0)$. L'unique problème consiste alors à estimer α^{-1} en fonction de z. Cette estimation peut être réalisée en apprenant la relation $z_i \mapsto \alpha_i^{-1}$, puis en appliquant le prédicteur à z. Les tests indiquent qu'un modèle linéaire généralisé polynomial de degré deux sans interactions est suffisant pour cette tâche. On en déduit alors $\hat{y}$.

Remarque : si l'approximation linéaire par C_0 est excellente tous les α_i sont proches de 1, et l'apprentissage statistique indiqué ci-dessus ne présente pas d'intérêt. En pratique cependant, les données ne sont pas très densément échantillonnées et on observe des valeurs s'éloignant significativement de 1 (jusqu'à une amplitude de 0.2 environ).

4.2.2.2 Sommet éloigné dans le graphe

Si z est trop loin de z_0, alors il n'a pas été déterminé à l'aide de l'espace tangent en y_0, mais par un problème d'optimisation sous contraintes (conservation des angles et des distances géodésiques). L'idée consiste naturellement à résoudre à nouveau ce problème en intervertissant les rôles des variables. On ne peut cependant pas définir un objet de dimension $D \gg d$ ("infinie") avec $\simeq d$ contraintes. C'est pourquoi on commence par réduire localement la dimension des y_i dont les indices $\mathcal{V}$ sont ceux des voisins de z_p. z_p est le plus proche voisin de z parmi les $\{z_i\}$. Cela s'effectue via une ACP (fonctionnelle), permettant de remplacer les $y_i, i \in \mathcal{V}$ par leurs représentations d-dimensionnelles c_i.

En notant $i_1, \ldots, i_k$ les indices des voisins de z_p, il ne reste ainsi plus qu'à chercher à conserver les angles :

$$\forall j = 1, \ldots, k \;\; \cos \widehat{c_{i_j} c_p c} \simeq \cos \widehat{z_{i_j} z_p z}\,,$$

sous la contrainte $\|c - c_p\| = \|z - z_p\|$. Les détails de la résolution sont indiqués dans l'article de Lin et al. [213]. y est ensuite estimé à partir du vecteur de coefficients c.

4.2.3 Tests

Afin d'évaluer la capacité de ces deux méthodes à reconstruire des courbes, nous commençons par réduire la dimension d'un jeu de n données y_i, obtenant n représentations z_i. Une procédure leave-one-out est alors implémentée. Concernant RML, $\hat{y}_i$ est estimé à partir de z_i qui a été virtuellement retiré des paramètres de reconstruction (les sorties de l'algorithme, non détaillées dans cette thèse). En effet une restitution directe serait exacte car lorsque z est très proche d'un des z_i on retourne simplement y_i. Avec toutes les données la classification supervisée serait systématiquement exacte pour la reconstruction via LPcaML, entraînant des erreurs d'autant plus petites que les voisinages sont réduits. C'est pourquoi nous retirons

l'exemple i des paramètres de classification pour l'estimation de y_i. L'ACP fonctionnelle n'a pas besoin de ces artifices, seule une base de fonctions globale étant stockée.

4.2.3.1 Données artificielles

Les deux jeux de données de ce premier paragraphe ne sont pas bruités, le problème étant déjà difficile sans ce brouillage. Les exemples réels du second paragraphe donneront une idée des performance en présence de courbes (probablement) bruitées.

Swisshole

Le premier benchmark a été utilisé lors les tests du chapitre précédent. Il consiste en un swissroll comportant un trou rectangulaire, d'où le terme "swisshole" pour le désigner. Ce dernier est rendu fonctionnel via $f_{\alpha,\beta,\gamma} : t \mapsto \alpha \cos t + \beta \sin t + \gamma \sin 2t$. La figure 4.4 rappelle ce jeu de données, et le tableau 4.2 indique les résultats obtenus pour $d = 2$ en terme de MSE (erreur quadratique moyenne), normalisée par le nombre de points de discrétisation, pour dix exécutions. Seules 200 courbes sont échantillonnées, pour se rapprocher des conditions des cas réels.

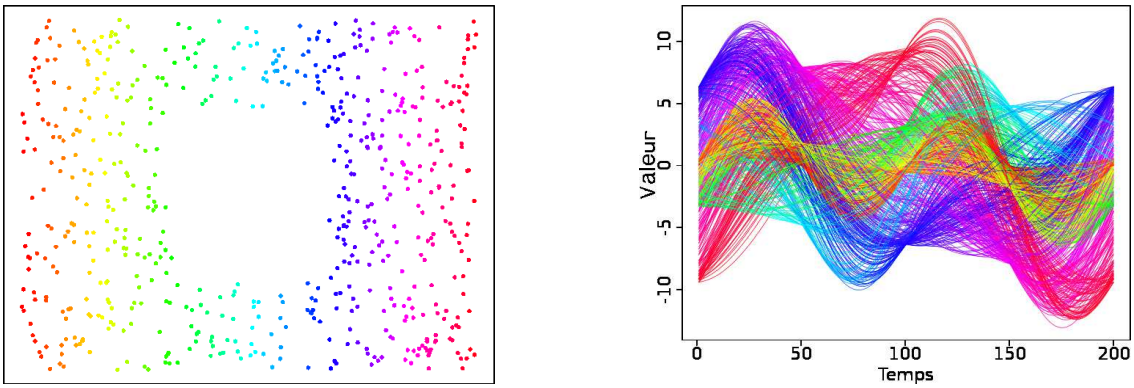

Figure 4.4: Swisshole $3D$ à gauche, swisshole fonctionnel à droite.

Méthode	ACPF	LPcaML	RML
MSE	2.71	1.42	3.34

Table 4.2: MSE globale sur le jeu de données swisshole pour les méthodes ACP, LPcaML et RML.

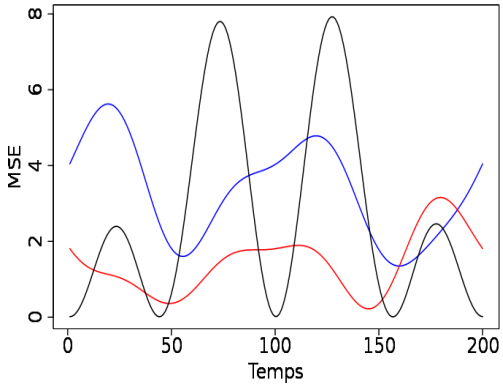

Figure 4.5: Courbes du MSE ponctuel, respectivement avec les méthodes ACP(F), LPcaML et RML, en noir, rouge et bleu.

La figure 4.5 illustre l'évolution de la MSE ponctuelle. Les courbes des méthodes non linéaires tendent à moins osciller que celle de l'ACP. Il est assez étonnant de voir l'ACP

fonctionnelle dépasser RML sur cet exemple. Cela est très probablement dû au faible échantillonnage. La méthode LPcaML (avec le réglage par défaut, c'est à dire qu'au moins la moitié d'une cellule est en chevauchement avec les autres) est assez efficace sur ce jeu de données, et pourrait l'être nettement plus si la classification supervisée s'effectuait avec moins d'erreurs.

Sinus cardinal composé

Le second exemple consiste en une composition d'un application sinusoïdale/exponentielle avec un sinus cardinal :

$$f_\alpha(t) = \frac{\sin(\alpha(e^t - 2.3\sin^2 t))}{e^t - 2.3\sin^2 t},$$

où $t \in [0, 1]$, $\alpha \in [30, 42]$. $n = 600$ fonctions du type f_α sont échantillonnées sur 400 points. La dimension intrinsèque vaut un, mais nous testons les méthodes pour $d = 1, \ldots, 3$. La figure 4.6 présente l'allure des courbes (régulièrement échantillonnées, pour faciliter la visualisation).

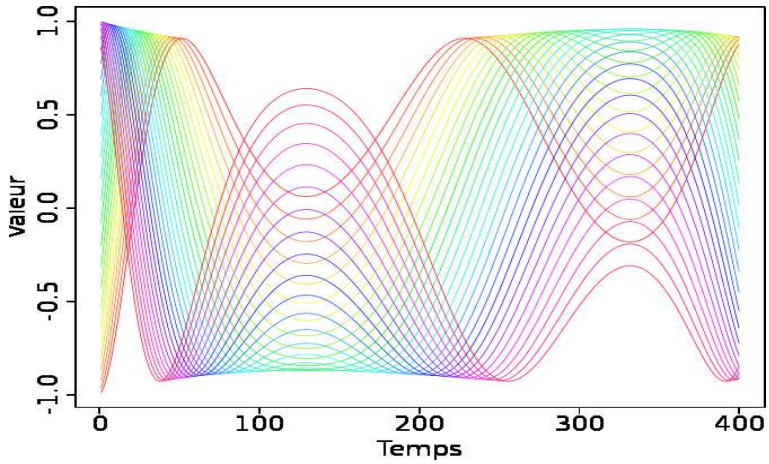

Figure 4.6: 30 courbes du type "sinus cardinal composé".

Le tableau 4.3 présente les MSE moyennes de $d = 1$ à $d = 3$. L'algorithme RML produit le meilleur codage/décodage des données en moyenne sur cet exemple, obtenant une MSE déjà assez basse pour $d = 1$. Les résultats affichés par LPcaML sont symptomatiques de la procédure de codage utilisée. En effet si l'erreur est inférieure à celle de l'ACP en cas d'une prédiction correcte de la cellule d'un z_i, elle croît rapidement à la moindre erreur de classification. Pour $d = 3$ certaines erreurs font ainsi monter la MSE au-delà de 1000, d'où le résultat reporté dans le tableau. Quelques rares configurations du jeu de données provoquent des erreurs similaires pour $d = 2$. Elles n'ont pas été observées dans le cas unidimensionnel. Il faudrait donc chercher une meilleure méthode pour classer les représentations, et/ou détecter les résultats incohérents en relançant la reconstruction. Enfin, l'analyse en composantes principales offre une nouvelle fois un bon compromis. Les courbes d'erreurs sont illustrées sur la figure figure 4.7.

Méthode	ACPF	LPcaML	RML
MSE $d = 1$	$1.42\,10^{-1}$	$1.17\,10^{-1}$	$4.20\,10^{-2}$
MSE $d = 2$	$8.56\,10^{-3}$	$2.63\,10^{-1}$	$6.91\,10^{-4}$
MSE $d = 3$	$1.49\,10^{-3}$	indéfini ($> 10^3$)	$2.13\,10^{-5}$

Table 4.3: MSE globale sur le jeu de données "sinus cardinal composé" pour les méthodes ACP, LPcaML et RML.

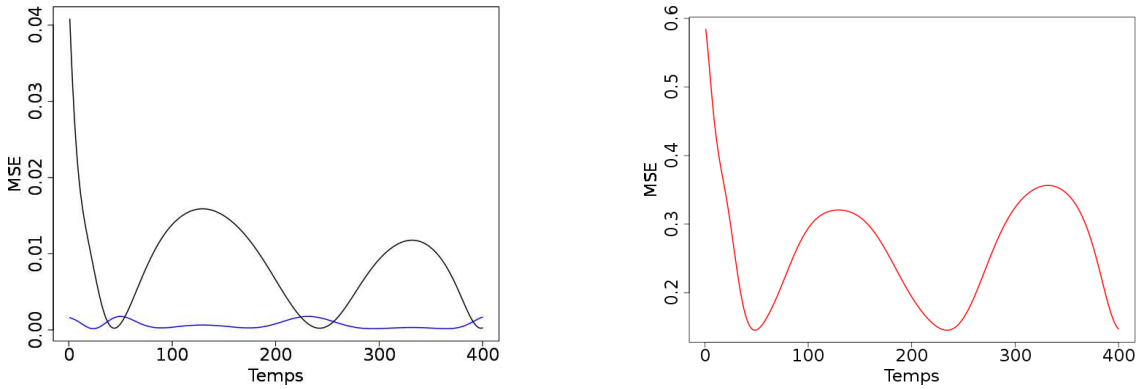

Figure 4.7: Courbes du MSE ponctuel pour $d = 2$, avec les méthodes ACP(F) et RML en noir et bleu à gauche ; avec la méthode LPcaML en rouge à droite.

4.2.3.2 Données réelles

Les jeux de données de ce second paragraphe correspondent aux courbes d'évolution de la température en sortie du code CATHARE, provenant de deux conditions expérimentales différentes.

CATHARE I

Cet ensemble d'entraînement constitué de 100 courbes discrétisées sur 168 points a déjà été présenté aux chapitres précédents. L'aspect des courbes est rappelé sur la figure 4.8. Les résultats pour $d = 4$ à $d = 6$ (l'algorithme de Levina et Bickel [207] estime d à 5) sont visibles dans le tableau 4.4, les courbes d'évolution du MSE étant présentées sur la figure 4.9.

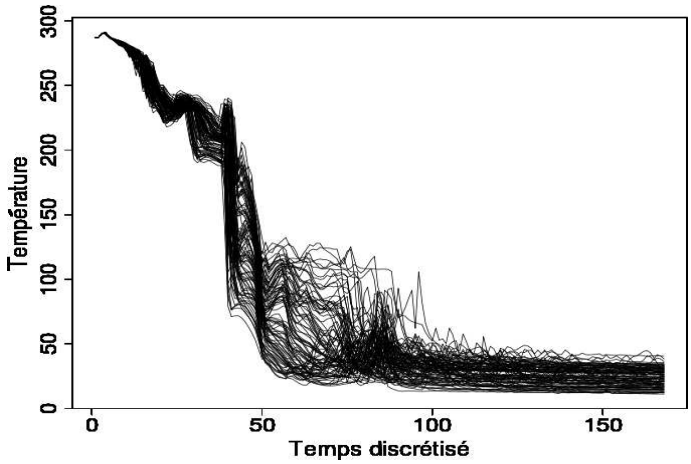

Figure 4.8: Les 100 courbes du jeu de données CATHARE I.

Méthode	ACPF	LPcaML	RML
MSE $d = 4$	26.8	28.2	48.4
MSE $d = 5$	20.3	41.1	36.2
MSE $d = 6$	15.9	60.7	34.6

Table 4.4: MSE globale sur le jeu de données CATHARE I pour les méthodes ACP, LPcaML et RML.

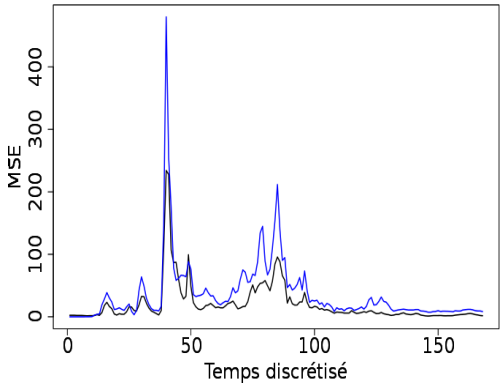

Figure 4.9: Courbes du MSE ponctuel pour $d = 5$, respectivement avec les méthodes ACP(F) et RML, en noir et bleu.

La méthode LPcaML s'avère cette fois trop instable pour que des résultats fiables puissent être reportés. Les valeurs indiquées correspondent à des minima observés sur quelques dizaines d'évaluations. Les statistiques approximatives plus proches de la réalité sont les suivantes :

- $d = 4$: MSE souvent aux alentours de 40, atteignant parfois plus de 70 ;
- $d = 5$: MSE souvent aux alentours de 60, atteignant parfois plus de 500 ;
- $d = 6$: MSE souvent aux alentours de 100, atteignant parfois plus de 1000.

C'est pourquoi la courbe du MSE ponctuel n'est pas tracée pour cette méthode. Comme en témoigne la figure 4.10, les courbes reconstituées par l'ACP fonctionnelle ne comportent pas d'aberrations mais sont vraisemblablement trop lissées. Celles produites par RML tendent à mieux reproduire les irrégularités, quitte à augmenter la MSE en cas d'erreur (notamment le pic vers l'abscisse 40).

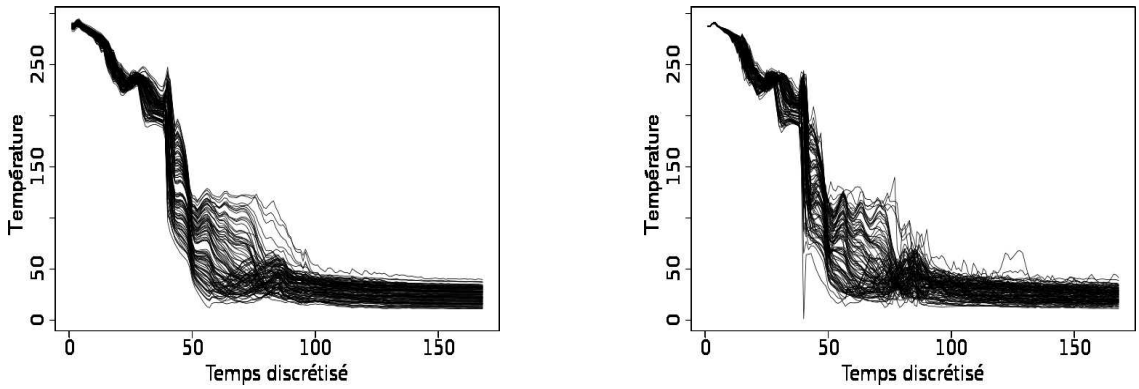

Figure 4.10: Reconstruction via la méthode ACPF à gauche et RML à droite, pour $d = 5$.

CATHARE II

Ce dernier jeu de données constitué de 600 courbes discrétisées sur 414 points a également été présenté aux chapitres précédents. L'aspect des courbes est rappelé sur la figure

4.11. Les résultats pour $d = 6$ à $d = 8$ (autour de l'estimation via l'algorithme de Levina et Bickel [207]) sont indiqués dans le tableau 4.5, la figure 4.12 illustrant les courbes d'évolution du MSE.

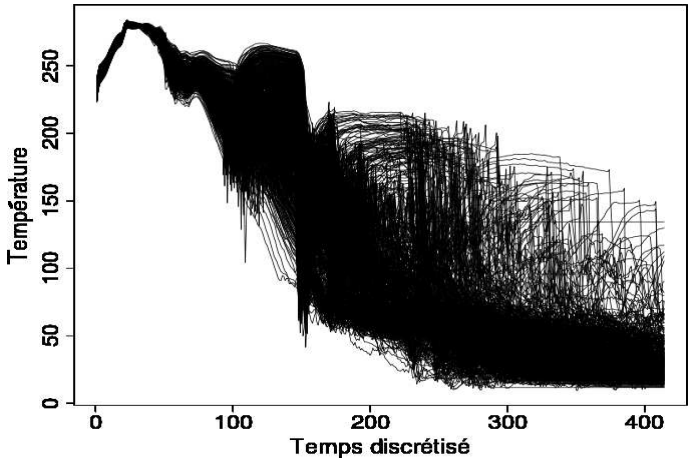

Figure 4.11: Les 600 courbes du jeu de données CATHARE II.

Méthode	ACPF	LPcaML	RML
MSE $d = 7$	63.1	$1.23\,10^3$	150
MSE $d = 8$	57.5	$2.70\,10^3$	145
MSE $d = 9$	52.6	168	142

Table 4.5: MSE globale sur le jeu de données CATHARE II pour les méthodes ACP, LPcaML et RML.

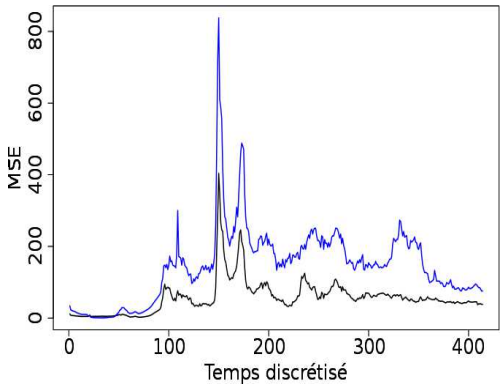

Figure 4.12: Courbes du MSE ponctuel pour $d = 8$, respectivement avec les méthodes ACP(F) et RML, en noir et bleu.

L'ACP fonctionnelle est cette fois encore clairement le meilleur choix en terme d'erreur quadratique moyenne, suivie par RML. Les résultats affichés pour LPcaML correspondent encore à des minima observés sur quelques dizaines d'essais (la variabilité provenant de la classification supervisée uniquement). Les statistiques (approximatives) suivantes sont plus proches de la réalité.

- $d = 7$: MSE stable à 1228 ;
- $d = 8$: MSE souvent aux alentours de 3000, atteignant parfois plus de 10000 ;
- $d = 9$: MSE souvent aux alentours de 200, atteignant parfois plus de 300.

L'amélioration subite à $d = 9$ est difficile à expliquer. Cette méthode devra de toutes façons être révisée pour être utilisée à des fins de régression.

Entre RML et l'ACP fonctionnelle les courbes reconstruites ont des allures très différentes, comme illustré sur les figures 4.13 et 4.14. Comme auparavant l'ACP lisse excessivement les courbes tandis que la méthode non linéaire tente de reproduire des irrégularités. Si le résultat de l'algorithme RML est presque parfait sur les deux premières courbes, les erreurs sont plus cher payées lorsqu'elle se trompe, comme l'indiquent les deux courbes de la dernière figure.

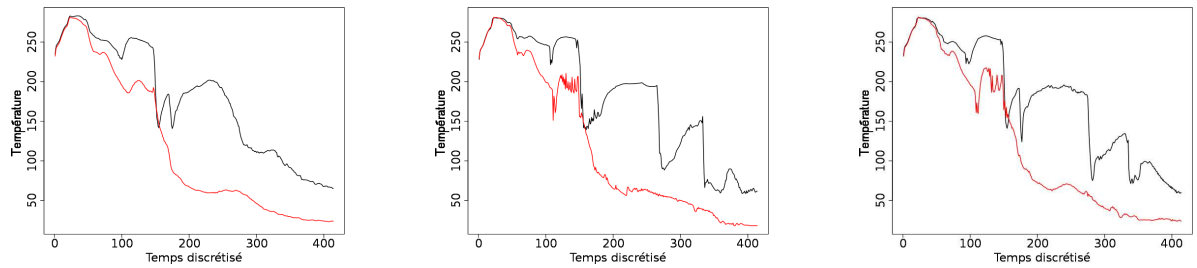

Figure 4.13: Deux courbes reconstruites avec les algorithmes ACPF (à gauche) et RML (à droite), les vraies fonctions étant au centre.

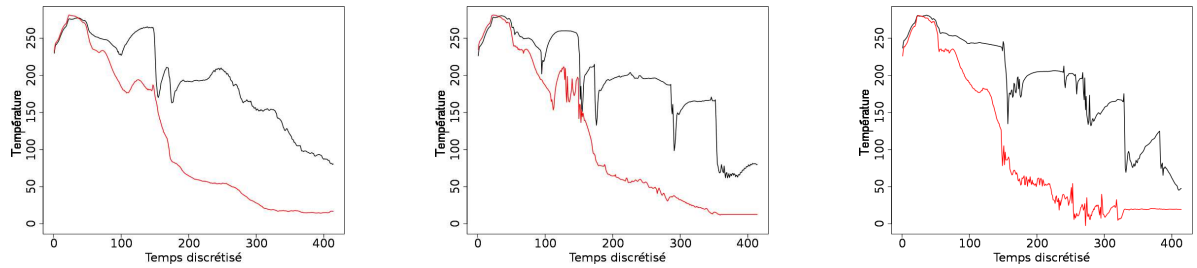

Figure 4.14: Deux courbes reconstruites avec les algorithmes ACPF (à gauche) et RML (à droite), les vraies fonctions étant au centre.

La section suivante se démarque du schéma de construction du métamodèle évoqué jusqu'alors. Elle présente des alternatives simples ne faisant pas intervenir la réduction de dimension (globale) et la reconstruction associée.

4.3 Régression fonctionnelle directe

Le but étant uniquement d'obtenir un modèle de régression, le passage par une carte globale des données via les z_i peut être contourné. En effet, lorsqu'une nouvelle entrée x est présentée, on peut considérer que seuls les éléments proches de x dans $\mathcal{E}_E$ ainsi que leurs images par ϕ vont jouer un rôle dans l'estimation de $\phi(x)$. Cela se justifie par la continuité de ϕ, au moins sur chaque cluster (hypothèse initiale). Les deux premiers estimateurs présentés ci-dessous effectuent donc un apprentissage local. Le troisième, en revanche, est plus proche d'une version fonctionnelle de l'algorithme PLS introduit au paragraphe 4.1.6.2.

4.3.1 Estimateur des k plus proches voisins fonctionnel

Évoquée dans introduction générale au paragraphe 0.4.1, cette méthode revient à estimer les sorties fonctionnelles comme suit.

$$\hat{\phi}(x) = \hat{y} = \frac{1}{C}\sum_{i=1}^{n} \kappa(x, x_i) y_i \,,$$

avec $C = \sum_{i=1}^{n} \kappa(x, x_i)$, κ étant éventuellement une fonction noyau comme au paragraphe 4.1.3. Si $\kappa(x, x_i)$ décroit suffisamment vite lorsque x_i s'éloigne de x, alors la prédiction correspond à une moyenne locale des voisins de y. Si les y_i sont densément échantillonnés, cette moyenne est en principe proche de la vraie courbe à estimer. Afin de s'adapter aux données nous choisissons

$$\kappa(x, x_i) = \begin{cases} e^{\frac{-\|x-x_i\|^2}{\sigma_i^2}} & \text{si } x_i \text{ est parmi les } k \text{ plus proches voisins de } x\,, \\ 0 & \text{sinon,} \end{cases}$$

k étant déterminé par validation croisée, et σ_i obtenu comme indiqué dans l'introduction du paragraphe 1.3. Notons que κ n'est alors pas un noyau ; en fait cette fonction n'est même pas définie sur un ensemble "continu" comme $\mathcal{E}_E$. On parle alors d'estimateur des k plus proches voisins pondéré, que l'on note kPPVF pour "k plus proches voisins fonctionnel".

Ce schéma de régression a reçu peu d'attention, contrairement à son homologue à entrées fonctionnelles et sortie scalaire [29; 120; 77; 52]. Cependant, les courbes que nous devons prédire ont la particularité d'être toutes égales à $t = 0$. En effet, la température (ou pression, ou coefficient d'échange) initiale est déterminée par les conditions de fonctionnement normales du réacteur, avant l'accident simulé. Cette méthode est donc particulièrement adaptée pour modéliser la toute première partie des courbes. Une variante est possible en s'inspirant de l'algorithme LLE de Roweis et Saul [269] :

1. chercher les poids ω_i minimisant $\|x - \sum_{i=1}^{k} \omega_i x_i\|^2$;
2. prédire $\hat{y} = \sum_{i=1}^{k} \omega_i y_i$.

cette dernière n'a pas été implémentée, la topologie des entrées n'ayant pas forcément de rapport avec celle des sorties (au moins par hypothèse).

4.3.2 Régression fonctionnelle par ACP locale

L'approche précédente détermine $\hat{y}$ par une moyenne locale des sorties. Une extension fréquente dans le cas scalaire consiste à utiliser plutôt des polynômes locaux pour modéliser $y_i = \phi(x_i)$ [73]. Elle n'est pas directement applicable ici, la sortie étant en dimension (théorique) infinie. On peut cependant réduire cette dimension localement, en utilisant l'ACP (fonctionnelle). Il ne reste alors plus qu'à apprendre la relation liant les entrées aux coefficients réduits des sorties, par exemple avec des polynômes. Nous préférons cependant utiliser la méthode PPR, par souci de cohérence. Le modèle peut alors s'écrire

$$\hat{\phi}(x) = g_{loc}(x) \,.\, C_{loc}\,,$$

avec pour g_{loc} la fonction de régression locale (PPR) autour de x, C_{loc} étant la matrice des fonctions de la base ACP correspondante en sortie. g_{loc} est entraînée sur les couples correspondant aux lignes des matrices (X_{loc} et $Y_{loc} \,.\, {}^tC_{loc}$), X_{loc} et Y_{loc} correspondant aux entrées (resp. sorties) voisines de x (resp. images des voisins de x). Les tailles des régions locales peuvent être déterminées par validation croisée. On note cette méthode ACPFL, pour ACP fonctionnelle locale.

En comparaison avec l'approche précédente celle-ci a − en principe − l'avantage d'être plus robuste au mauvais échantillonnage. Elle est cependant nettement plus coûteuse, chaque prédiction nécessitant la recherche d'une base ACP locale ainsi que l'apprentissage d'une fonction de régression. Cela n'est pas rédhibitoire en pratique tant que les voisinages ne sont pas trop importants (jusqu'à quelques centaines de points chacun). Enfin, une contrainte pèse sur cette méthode par rapport à la précédente : la dimension doit être estimée au préalable, comme indiqué à la section 3.1.

Remarque : Les algorithmes de ces deux derniers paragraphes nécessitent la recherche des voisins d'une nouvelle entrée x. Cela se fait très rapidement dans notre cas, mais pourrait être trop coûteux dans le cas de très grandes bases de données.

4.3.3 "Functional projection pursuit"

Abandonnant la régression locale, une dernière idée envisagée consiste à étendre une procédure du type PLS à des sorties fonctionnelles. Nous nous inspirons de l'algorithme nommé "PLS2" [310; 265], qui vise à régresser des projections dans l'espace (multidimensionnel) des sorties sur des projections dans l'espace $\mathcal{E}_E$. En notant X (resp. Y) la matrice des entrées en lignes (resp. le vecteur colonne des fonctions en sortie), l'algorithme peut se décrire comme suit. Celui-ci n'a pas été implémenté dans le package R mais le sera probablement dans une version ultérieure.

1. Initialiser les résidus d'approximation fonctionnels η_i aux courbes y_i, et un compteur c à 1.
2. Chercher $\zeta_c \in \mathbb{R}^p$ et $\xi_c \in \mathcal{C}([a,b])$ directions de projection respectivement dans les espaces d'entrée et sortie, de manière à maximiser

$$\mathrm{Cov}(X\zeta_c, Y\xi_c) = \frac{1}{n}\sum_{i=1}^{n}(\langle x_i, e_c\rangle - \mu_e)(\langle \eta_i, \xi_c\rangle - \mu_s)\,,$$

   où μ_e (resp. μ_s) désigne la moyenne des projections en entrée (resp. la moyenne des produits scalaires $\langle \eta_i, \xi_c\rangle$).
3. Apprendre la relation $f_c : \langle x_i, \zeta_c\rangle \mapsto \langle \eta_i, \xi_c\rangle$ via une spline de lissage par exemple.
4. Actualiser les résidus fonctionnels : $\eta_i \leftarrow \eta_i - f_c(\langle x_i, \zeta_c\rangle)\xi_c$. Si η n'est pas assez proche de zéro, incrémenter c et revenir en 2.

Preda et Saporta [254] montrent que les composantes PLS fonctionnelles ξ_c sont obtenues via les fonctions propres de l'opérateur d'Escoufier [115]. Ce dernier opérateur est approximé

par son équivalent empirique matriciel, dont on recherche les vecteurs propres.

Remarque : dans l'algorithme original la matrice X est également réduite au cours des itérations. Cela permet de garantir l'orthogonalité mutuelle des composantes ζ_c. Cette dernière propriété ne semble pas nécessaire pour notre application.

Les méthodes de régression fonctionnelle par moyenne locale pondérée et via l'ACP sont implémentées dans le package R, et testées au chapitre suivant.

4.4 Conclusion

Ce chapitre vient compléter les précédents en proposant une méthode de régression vectorielle au sein de chaque cluster, ainsi que des algorithmes de reconstruction des courbes à partir de leurs représentations. Cela permet de finaliser la construction du métamodèle, celui-ci étant résumé au début du chapitre suivant. Nous avons également discuté de quelques approches alternatives visant à régresser directement les courbes sur les variables d'entrée. Celles-ci se révèlent efficaces sur certains jeux de données artificiels comme le montrent les tests finaux.

4.5 Perspectives

Le modèle incluant la réduction de dimension étant à présent achevé, et résumé par les étapes ci-dessous

$$x \to g(x) \simeq r(y) \to R(g(x)) \simeq y\,,$$

il devient possible d'étudier la convergence vers le métamodèle optimal. Notons que les représentations n'ont pas besoin d'être elles-mêmes "optimales" : il faut et il suffit que les reconstructions à partir de ces dernières le soient. De plus on sait que l'algorithme PPR peut apprendre toute fonction continue.

Concernant la méthode du paragraphe 4.3.2 il est envisageable d'utiliser les travaux de Mas [225], qui donne la vitesse de convergence de l'opérateur de covariance local empirique vers l'opérateur théorique. En les combinant avec des résultats d'approximation des données par le plan tangent (par exemple avec un échantillonnage supposé uniforme), on devrait pouvoir obtenir la vitesse de convergence du modèle ACPFL.

Chapitre 5

Évaluation du métamodèle

Ce dernier chapitre vise à valider le modèle construit en testant sa capacité de prédiction. Après avoir résumé les différentes étapes du métamodèle, nous discutons de la méthodologie à suivre afin de valider ce dernier. Divers tests sur des données artificielles et réelles (CATHARE) sont finalement présentés.

Auder et al. [16] effectuent quelques tests similaires à ceux indiqués dans cette dernière partie. L'article n'est pas inséré dans cette thèse car les résultats ont été complétés et légèrement améliorés depuis. Il est cependant présent en annexe C, car il explique le rôle joué par le métamodèle au sein de la méthodologie développée dans le cadre industriel.

Plan du chapitre :

1. **Rappel de l'expression du métamodèle complet.**
2. **Définition de deux indicateurs d'erreur utilisés pour la validation.**
3. **Tests sur deux jeux de données artificielles.**
4. **Tests sur deux jeux de données réelles (CATHARE).**

5.1 Récapitulatif du modèle

En rassemblant les étapes étudiées par chaque chapitre, on obtient l'enchaînement suivant aboutissant à l'estimation de $\phi(x)$ pour $x \in \mathcal{E}_E$:

1. $x \mapsto c(x)$, numéro du cluster contenant x (étape optionnelle) ;
2. $x \mapsto g_{c(x)}(x) \simeq r(y)$ (approximation de la représentation de la courbe $y = \phi(x)$) ;
3. $g_{c(x)}(x) \mapsto R(g_{c(x)}(x))$: reconstruction de l'approximation de $r(y) \in \mathcal{E}_R$.

Ainsi

$$\hat{\phi}(x) = R(g_{c(x)}(x)) \,.$$

La figure 5.1 résume la méthodologie, les entrées étant schématisées au centre, les sorties à gauche et leur représentations à droite.

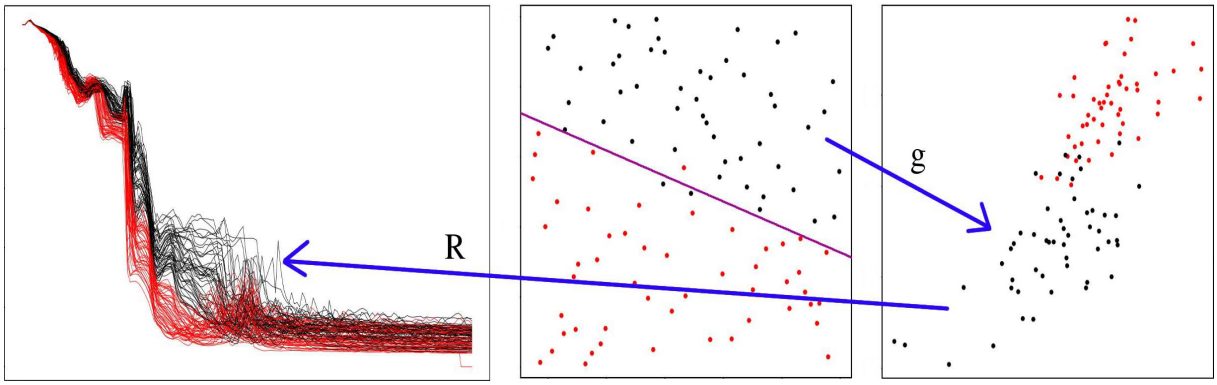

Figure 5.1: Illustration des étapes du métamodèle.

Mélange de modèles

Dans la philosophie du boosting, une extension immédiate consiste à effectuer une somme pondérée de plusieurs métamodèles comme suit :

$$\hat{\phi}(x,t) = \sum_{j=1}^{m} \alpha_j(t)\varphi_j(x,t)\,,$$

où φ_j désigne un modèle construit selon la méthodologie ci-dessus, les $\alpha_j : [a,b] \to [0,+\infty[$ étant des fonctions de pondération vérifiant la propriété de normalisation suivante.

$$\forall t \in [a,b] \ \sum_{j=1}^{m} \alpha_j(t) = 1\,.$$

En particulier, le métamodèle basé sur les k plus proches voisins présenté au paragraphe 4.3.1 peut être combiné avec un des autres pour améliorer les prédictions au voisinage de l'origine des temps. Ainsi, les courbes d'erreur du meilleur modèle pouvant être construit automatiquement correspondent au minimum ponctuel des courbes d'erreur des différents modèles testés.

5.2 Validation via l'erreur de prédiction

Nous distinguons deux cas dans cette section, suivant que l'on peut disposer d'autant d'échantillons que nécessaire ou non.

5.2.1 Données générées artificiellement

La méthodologie de validation est très simple dans cette situation : on commence par tirer un jeu de N couples entrées-sorties avec N "grand", puis on effectue l'apprentissage sur n nouvelles données (n de l'ordre de quelques centaines) avant de calculer des indicateurs d'erreur sur les N couples. Afin que ces indicateurs ne soient pas influencés par une configuration

de points très particulière, il est nécessaire de répéter plusieurs fois la phase d'entraînement puis de calcul d'erreurs (m répétitions).

Notons T (comme "test") l'ensemble de N éléments, et A_j le j^{eme} ensemble d'apprentissage à n éléments, $j = 1, \ldots, m$. S désignant un de ces deux ensembles de couples, S^x correspond aux entrées de S et S^y aux sorties ; $S^x[s]$ (resp. $S^y[s]$) désigne la s^{eme} entrée (resp. sortie) de l'ensemble S. Enfin, le métamodèle construit à l'aide de A_j est noté $\hat{\phi}_{A_j}$.

La MSE (Mean Square Error) moyenne, fonctionnelle, se calcule alors comme suit :

$$\forall \ell = 1, \ldots, D \;\; \text{MSE}[\ell] = \frac{1}{mN} \sum_{j=1}^{m} \sum_{s=1}^{N} (\hat{\phi}_{A_j}(T^x[s])[\ell] - T^y[s][\ell])^2 \,,$$

$f[\ell]$ désignant la valeur de la fonction $f : [a, b] \to \mathbb{R}$ au ℓ^{eme} point de discrétisation. Il s'agit simplement de l'écart quadratique moyen ponctuel entre une prédiction et la vraie courbe. Cet indicateur donne déjà une certaine information : les zones où il est (relativement) élevé correspondent aux parties difficiles à apprendre. Il est souhaitable, cependant, d'évaluer la performance "absolue" de l'estimateur $\hat{\phi}$. L'indicateur suivant (Q_2) répond dans une certaine mesure à cette attente.

Le Q_2 (moyen, fonctionnel) se calcule comme suit :

$$\forall \ell = 1, \ldots, D \;\; Q_2[\ell] = 1 - \frac{mN\text{MSE}}{\sum_{j=1}^{m} \sum_{s=1}^{N} (\overline{A_j^y}[\ell] - T^y[s][\ell])^2} \,,$$

$\overline{A_j^y}$ désignant la moyenne des fonctions en sortie de l'ensemble d'entraînement A_j. Le Q_2 mesure ainsi à quel point le métamodèle est meilleur (ou plus mauvais) que le prédicteur simpliste consistant à toujours retourner la moyenne des sorties A_j^y. Celui-ci est normalisé pour être dans $]-\infty, 1]$. Une valeur de 1 indique un apprentissage parfait (interpolation exacte sur la base de test), tandis qu'une valeur inférieure à zéro signifie que le modèle ne doit pas être utilisé.

Finalement, en calculant les MSE et Q_2 partiels sur chacun des m jeux de données, il est possible d'estimer les écarts-type de ces grandeurs et ainsi d'évaluer l'incertitude associée. Les deux écarts-type notés respectivement σ_{MSE} et σ_{Q_2} s'expriment comme suit, pour tout $\ell = 1, \ldots, D$.

$$\sigma^2_{\text{MSE}}[\ell] = \frac{1}{m} \sum_{j=1}^{m} (\text{MSE}^j[\ell] - \overline{\text{MSE}}[\ell])^2$$

$$\sigma^2_{Q_2}[\ell] = \frac{1}{m} \sum_{j=1}^{m} (Q_2^j[\ell] - \overline{Q_2}[\ell])^2 \,,$$

où MSE^j est la MSE calculée sur le j^{eme} ensemble d'entraînement seulement :

$$\text{MSE}_j[\ell] = \frac{1}{N} \sum_{s=1}^{N} (\hat{\phi}_{A_j}(T^x[s])[\ell] - T^y[s][\ell])^2 \,.$$

De même,

$$Q_2^j[\ell] = 1 - \frac{N\text{MSE}_j}{\sum_{s=1}^{N}(\overline{A_j^y}[\ell] - T^y[s][\ell])^2}\,.$$

$\overline{\text{MSE}}$ (resp. $\overline{Q_2}$) désigne la fonction (discrétisée) égale à la moyenne des MSE_j (resp. Q_2^j).

Les courbes d'erreur indiquées lors des tests sur les jeu de données générés artificiellement sont les MSE et Q_2 définies ci-dessus.

5.2.2 Données réelles (CATHARE)

Compte-tenu de la lenteur du code de calcul CATHARE (ou de tout autre code du même type), il n'est pas possible de constituer une grande base de tests comme au paragraphe précédent. En fait, si cela était réalisable il ne serait pas nécessaire de construire un métamodèle. Il est alors envisageable d'entraîner le modèle sur toutes les données à disposition, puis d'estimer sa capacité à reconstruire ces même données. L'objectif étant une utilisation en prédiction, on préfère appliquer la validation croisée comme évoquée à la section 0.2 :

1. retirer quelques couples entrée-sortie de la base de données complète, suivant la loi uniforme sur $\{1, \ldots, n\}$;
2. entraîner le métamodèle sur les échantillons restant ;
3. évaluer les erreurs réaliser sur les données retirées,

ce schéma étant répété autant de fois que la puissance de calcul le permet. Concernant les tests de ce chapitre cette dernière boucle est réalisée 100 fois. Notons qu'il existe une variante de la validation croisée dans laquelle tous les ensemble de tests sont disjoints. Cela correspond à un tirage sans remise dans l'ensemble $\{1, \ldots, n\}$ au fil des itérations. Nous préférons cependant les choisir aléatoirement, afin de tester potentiellement plus de combinaisons. Efron et Tibshirani [112] proposent d'utiliser le bootstrap afin d'améliorer la fiabilité de la validation croisée, mais leur méthode n'a pas été implémentée ici.

Introduisons tout d'abord quelques notations. A_j désigne les indices de l'ensemble d'apprentissage lors de la j^{eme} boucle de validation croisée, $j = 1, \ldots, m$, T_j étant la base de test correspondante : $A_j \bigsqcup T_j = \{1, \ldots, n\}$. Bien qu'ayant nettement moins d'éléments, T_j (de cardinal $q \ll N$) peut jouer le rôle de T du paragraphe précédent. Les indicateurs s'écrivent alors de la même manière, pour $\ell = 1, \ldots, D$:

$$\text{MSE}[\ell] = \frac{1}{mq}\sum_{j=1}^{m}\sum_{s=1}^{q}(\hat{\phi}_{A_j}(T_j^x[s])[\ell] - T_j^y[s][\ell])^2$$

$$Q_2[\ell] = 1 - \frac{mq\text{MSE}}{\sum_{j=1}^{m}\sum_{s=1}^{q}(\overline{A_j^y}[\ell] - T_j^y[s][\ell])^2}\,,$$

etc. En pratique on choisit q petit devant n, mais non négligeable ; en général 5 à 10% de l'ensemble des données.

Les courbes d'erreur indiquées lors des tests sur les jeu de données réels sont les MSE et Q_2 définies ci-dessus.

Vers une évaluation moins conventionnelle

Toutes les étapes de la construction du métamodèle se basant sur la norme L^2, il semble cohérent de mesurer les erreurs moyennes L^2 comme indiqué ci-dessus. Cependant, rappelons que ce modèle est couplé avec un code de calcul mécanique afin de déterminer les "facteurs de marge", à partir desquels les probabilités de défaillance sont directement estimées. Une possibilité serait alors de comparer non pas les écarts L^2 avec les courbes du jeu de test, mais d'évaluer directement les erreurs sur les facteurs de marges, depuis les courbes prédites et "réelles". Ces facteurs étant scalaires il suffit de sommer les carrés des écarts absolus sur les facteurs de marge pour obtenir un nouvel indicateur. Cette approche sera probablement utilisée en interne au laboratoire où la thèse a été effectuée.

Une autre piste serait d'évaluer les irrégularités reproduites par les différentes méthodes relativement à celles présentes dans le jeu de données initial. Par exemple, une dimension fractale similaire pourrait indiquer une méthode potentiellement plus efficace.

Tout le reste de ce chapitre est constitué de tests divers, les méthodes étant comparées à l'aide des indicateurs statistiques présentés ci-dessus.

5.3 Tests sur données artificielles

Deux jeux de données artificiels sont testés dans cette première sous-partie. Il est possible avec ceux-ci d'évaluer dans une certaine mesure l'influence du nombre d'échantillons sur la précision des prédictions. Nous choisissons $N = 10000$, soit une base de test de 10000 éléments. Les erreurs sont reportées en calculant les indicateurs introduits au paragraphe 5.2.1. Aucun résultat n'est indiqué pour la méthode LPcaML (voir 3.2.3.1), car les erreurs de classifications emmènent le Q_2 systématiquement en dessous de zéro.

Quelques notations sont utilisées tout au long de ce chapitre, afin d'éviter de trop nombreuses répétitions. Ainsi, mACPF, mRML, mkPPVF et mACPFL désignent les (méta)modèles (ou méthodes) basé(e)s respectivement sur l'ACP fonctionnelle (globale), l'algorithme de réduction de dimension RML, les k plus proches voisins fonctionnels et l'ACP fonctionnelle locale. Les paramètres de ces méthodes sont fixés à leurs valeurs par défaut.

5.3.1 Trois clusters non linéaires

Ce premier exemple a déjà été présenté au paragraphe 1.3.2.1 pour tester les qualités des similarités. Nous l'utilisons cette fois en tant que fonction qui à (α, β, γ) associe $f_{\alpha,\beta,\gamma}$ définie sur $[1, 5]$ comme suit.

$$f_{\alpha,\beta,\gamma}(t) = \left(\frac{\sin \alpha x}{x} + e^{-\beta x} \right) \cos \gamma t \,,$$

les paramètres α, β et γ variant comme indiqué en 1.3.2.1 afin d'obtenir trois clusters. La figure 5.2 montre l'allure des courbes dans chaque groupe (légèrement bruitées, uniformément échantillonnées). Un bruit blanc d'écart-type 0.02 est appliqué pour rendre les courbes plus réalistes.

La dimension est estimée à 2 par la méthode d_{levi} présentée au paragraphe 3.1.2.2. L'algorithme d_{sliv} (voir 3.1.4) trouve en revanche la bonne valeur ($d = 1$). Il est assez surprenant de constater que la méthode qui surestimait la dimension lors des tests au chapitre 3 trouve cette fois la bonne valeur, tandis que d_{levi} surestime (très légèrement) la vraie dimension. Cependant, d_{sliv} donne des résultats élevés sur les jeux de données réels de la partie suivante, mais l'estimateur n'est pas vraiment meilleur en augmentant la dimension. C'est pourquoi nous continuons d'utiliser d_{levi} pour la modélisation du code CATHARE.

Les écarts-type des Q_2 et MSE sont quasi nuls pour mkPPVF et mACPFL, et très faibles pour mACPF. Les prédictions de mRML sont en revanche assez variables, l'écart-type atteignant 0.3 dans les zones très mal modélisées. Celui-ci n'est cependant pas tracé, l'erreur étant déjà grande là où l'écart-type est élevé. Nous préférons donc nous concentrer sur les tendances moyennes.

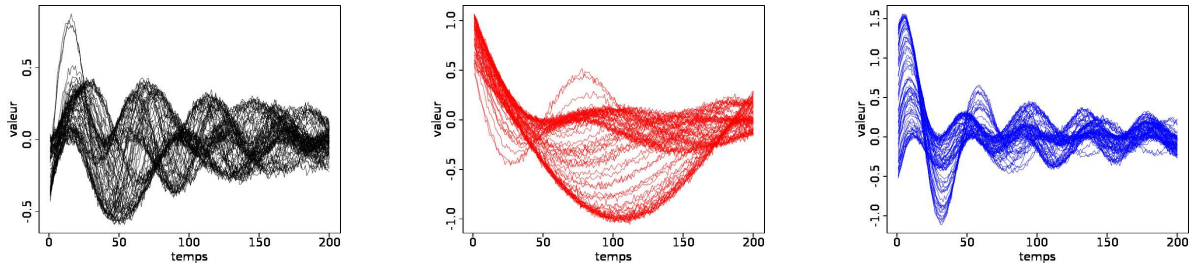

Figure 5.2: Clusters fonctionnels 1, 2 et 3 de gauche à droite, légèrement bruités.

5.3.1.1 Avec clustering

Lorsque la procédure de détermination automatique du nombre de clusters est utilisée, les trois groupes ne sont pas toujours validés (avec les seuils choisis à la fin du chapitre 2). Cela freine les performances du métamodèle, et devra faire l'objet d'améliorations ultérieures. Concernant ce test nous forçons le nombre de classes à trois, afin de visualiser le gain "idéal" apporté par le clustering. Les vrais résultats du modèle complètement automatique se situeraient quelque part entre ceux de ce paragraphe et ceux du suivant, répétant les tests sans l'étape de classification.

Dimension 1

Dans un premier temps, la dimension réelle est utilisée afin d'évaluer les performances des méthodes. La figure 5.3 présente l'évolution du Q_2 respectivement pour $n = 120$, $n = 240$ et $n = 360$. La figure 5.4 illustre l'évolution de la MSE pour les mêmes nombres de courbes dans la base d'apprentissage.

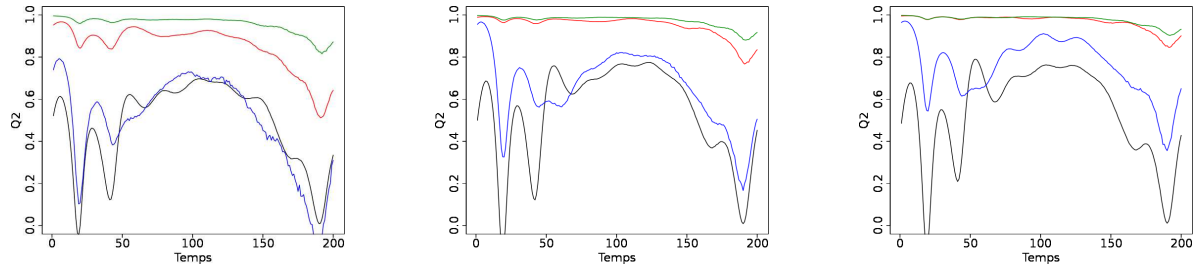

Figure 5.3: Q_2 pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

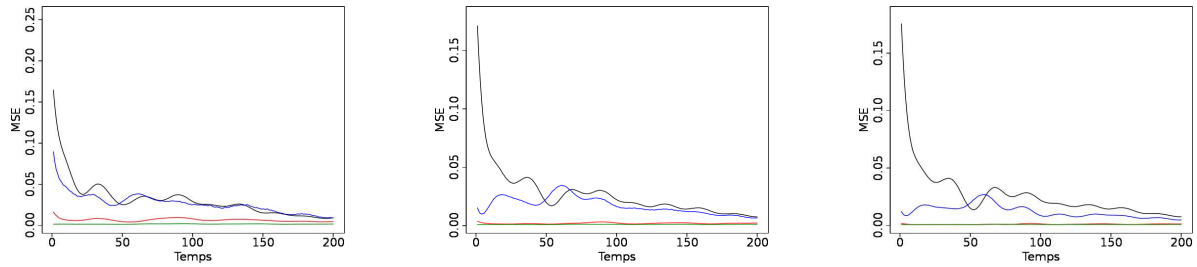

Figure 5.4: MSE pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

La méthode mkPPVL donne clairement les meilleurs résultats. Il semble en effet assez normal qu'une moyenne locale prédise bien les courbes dans le cas de la dimension 1. mACPFL est presque aussi performante pour $n \geq 240$, et reste nettement au-dessus de mACPF et mRML. Ces deux dernières donnent des résultats similaires, tout juste acceptables, pour $n \leq 240$. RML se démarque de la simple ACP fonctionnelle lorsque n atteint 360, avec un Q_2 moyen autour de 0.8. En absence de bruit mRML est en fait presque aussi performant que les deux meilleurs méthodes, avec un Q_2 très proche de 1 pour $n \geq 240$. Cette situation est cependant assez improbable.

Dimension 2

En choisissant cette fois $d = 2$ nous pouvons observer l'amélioration apportée aux prédictions sur les figures 5.5 (Q_2) et 5.6 (MSE). Il y a en effet un net progrès, en particulier pour mACPF et mRML, cette dernière étant une nouvelle fois meilleure que l'ACP fonctionnelle dès que n dépasse 240. Le Q_2 obtenu par mACPFL est amélioré d'environ 0.1 pour $n = 120$, atteignant presque 1. Cela montre qu'une (légère) surestimation de la dimension peut parfois être souhaitable. mkPPVF ne se basant pas sur la dimension, ses résultats indiqués ici sont les mêmes qu'au paragraphe précédent.

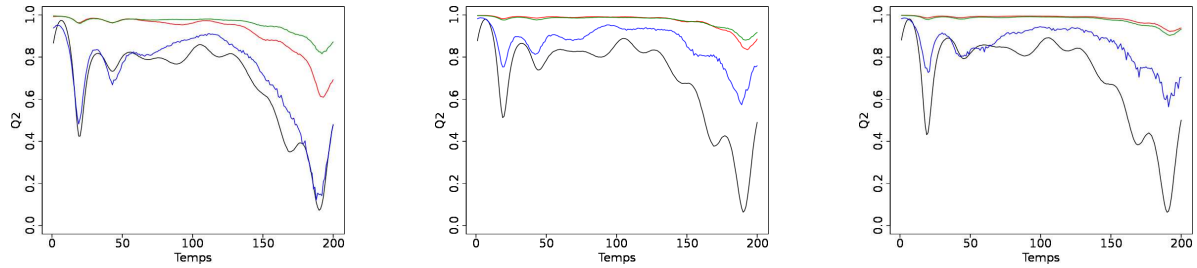

Figure 5.5: Q_2 pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

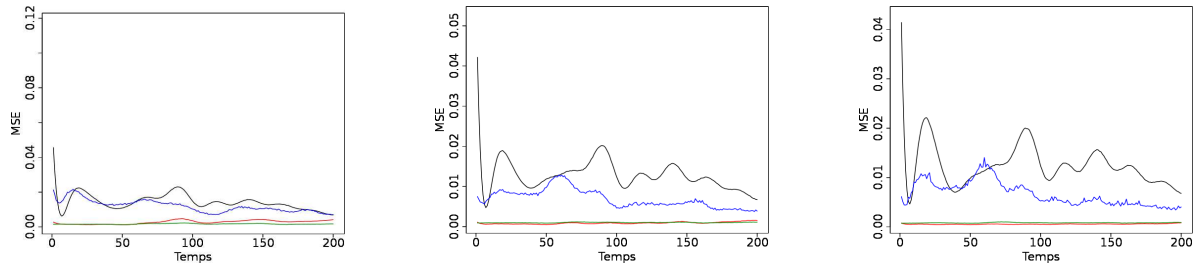

Figure 5.6: MSE pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

5.3.1.2 Sans clustering

Nous présentons finalement les résultats obtenus sans passer par l'étape de classification, afin de montrer que cette dernière est nécessaire sur cet exemple, comme attendu compte-tenu de la structure du jeu de données.

Dimension 1

Les évolutions des Q_2 et MSE sont visibles respectivement sur les figures 5.7 et 5.8. Assez logiquement, les méthodes complètement locales (mkPPVF et mACPFL) donnent des résultats similaires aux précédents. Les performances des deux autres sont catastrophiques, le Q_2 moyen étant en dessous de 0.4. Il est en effet impossible d'obtenir une représentation topologiquement correcte des données en dimension 1.

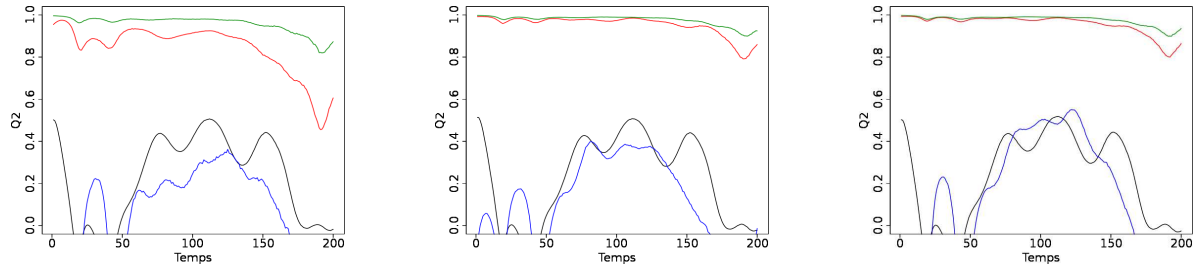

Figure 5.7: Q_2 pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

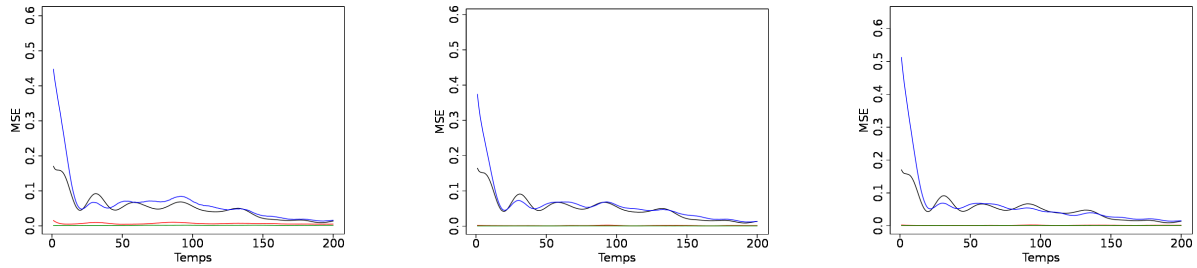

Figure 5.8: MSE pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

Dimension 2

Les figures 5.9 et 5.10 présentent les courbes d'évolution du Q_2 et de la MSE pour $d = 2$, toujours sans clustering. Il y a clairement une amélioration au niveau des méthodes mACPF et mRML, les deux autres prédisant encore les courbes de façon quasi parfaite. Les performances des deux premières méthodes restent cependant médiocres, notamment pour les abscisses inférieures à 50 et supérieures à 150 (soit tout de même la moitié de la plage de valeurs des 200 points de discrétisation).

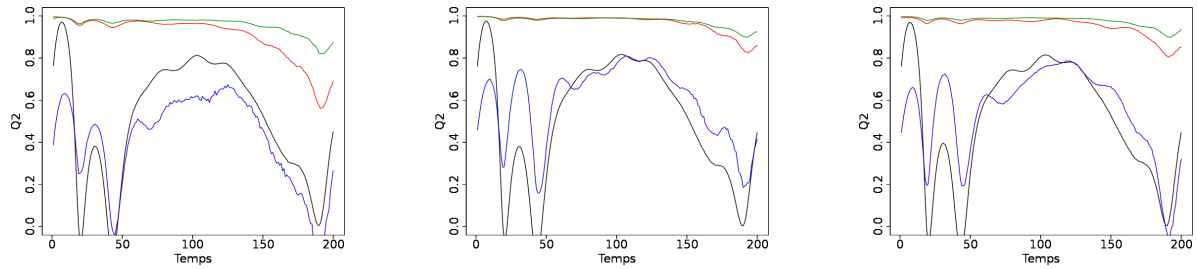

Figure 5.9: Q_2 pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

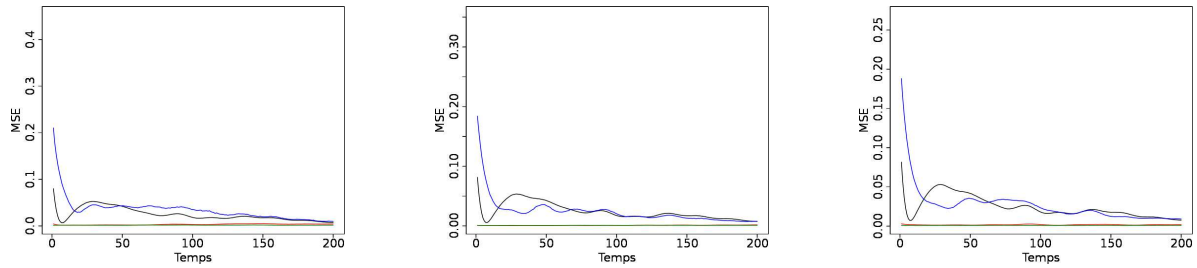

Figure 5.10: MSE pour $n = 120$, $n = 240$ et $n = 360$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

5.3.2 Fonctions sinusoïdales amorties

Ce second jeu de données générées artificiellement ne comporte pas de clusters, aussi nous n'activons pas la classification non supervisée dans ce cas. Voici l'expression des courbes en

sortie, définies sur $[0, 5]$.

$$f_{\alpha,\beta,\gamma,\delta,\epsilon}(t) = \sin((\alpha t \arctan(\beta t))^3)e^{\frac{-3t^2}{\gamma}} + \log(2 + \cos(\delta t)) + \arctan(\epsilon t)) \,,$$

avec $\alpha, \beta, \gamma, \delta$ et ϵ suivant des lois uniformes respectivement dans les intervalles $[0.5, 1.5]$, $[2, 7]$, $[0.2, 2.5]$, $[0.5, 1.5]$ et $[0, 1.5]$. Ces derniers constituent les paramètres d'entrée, et indiquent la dimension de la variété, égale à 5. Compte-tenu des possibles oscillations nous choisissons d'échantillonner les fonctions sur 500 points de discrétisation. La figure 5.11 présente un aperçu des courbes $f_{\alpha,\beta,\gamma,\delta,\epsilon}$, avec un bruit blanc gaussien d'écart-type 0.01. Si l'on met de côté mRML lorsque $d = 3$, les écarts-type sur le Q_2 sont tous très faibles. Ils ne sont donc pas représentés ici. Nous pouvons cependant retenir que mkPPVF est clairement la méthode donnant les prédictions les moins variables, suivie de près par mACPFL. L'estimation par ACP fonctionnelle globale est un peu plus variable, et celle utilisant RML l'est encore nettement plus.

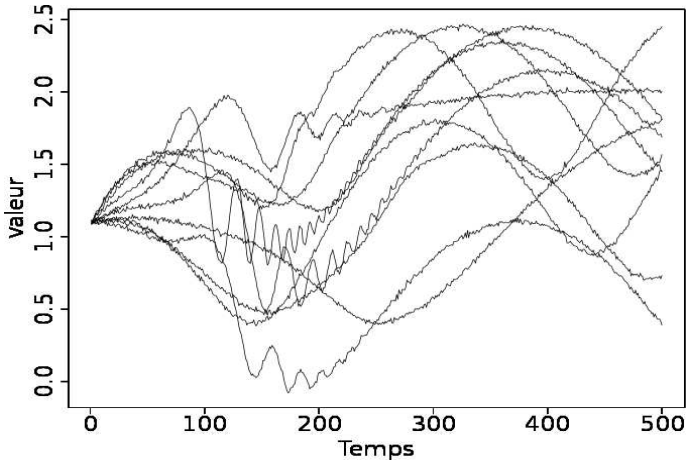

Figure 5.11: Dix courbes du type $f_{\alpha,\beta,\gamma,\delta,\epsilon}$, légèrement bruitées.

Concernant l'estimation de la dimension, d_{levi} sous-estime d en retournant 3, tandis que d_{sliv} trouve encore une fois la bonne valeur ($d = 5$), et ce dès $n = 200$. Il faut plus de 800 courbes pour que d_{levi} estime la dimension à 4, et probablement plusieurs dizaines de milliers de courbes pour voir $d = 5$ (nous avons essayé jusqu'à $n = 10000$). Ainsi, les résultats sont présentés en dimensions trois et cinq.

5.3.2.1 Dimension 3

Les figures 5.12 et 5.13 illustrent respectivement les courbes des Q_2 et MSE. Les méthodes utilisant l'ACP fonctionnelle (locale ou non) sont cette fois clairement meilleures que les autres en moyenne. Les performances de mkPPVF et mRML sont similaires à partir de l'abscisse 300. Cependant, le Q_2 obtenu via l'algorithme RML est médiocre entre les temps 0 et 300 (en dessous de 0.5). Notons que mkPPVF est la seule à fournir d'excellentes prédictions proche de l'origine des temps, toutes les fonctions partageant la même valeur à $t = 0$. C'est d'ailleurs la principale raison pour laquelle nous l'avons ajoutée au package R. Cette propriété se retrouve quelque peu avec mRML, qui est meilleure que mACPF(L) entre les abscisses 0 et 50 environ.

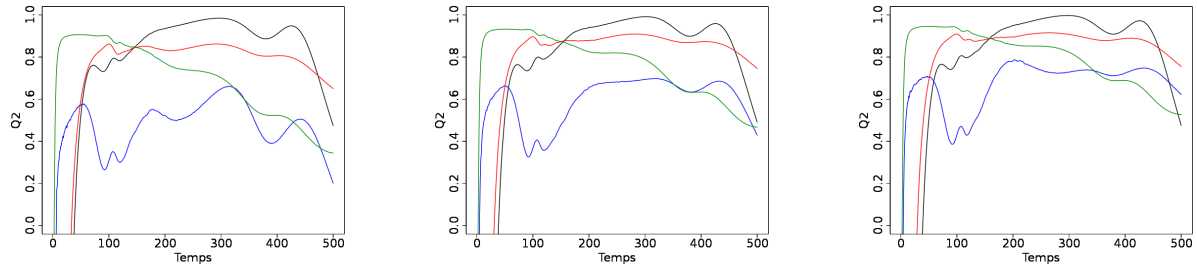

Figure 5.12: Q_2 pour $n = 200$, $n = 400$ et $n = 600$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

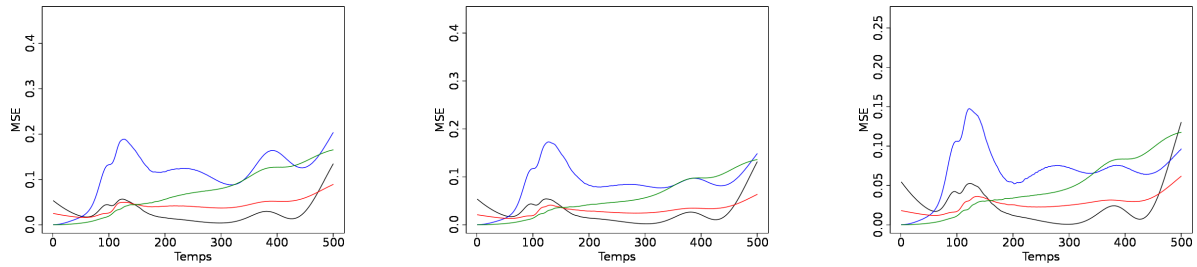

Figure 5.13: MSE pour $n = 200$, $n = 400$ et $n = 600$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

5.3.2.2 Dimension 5

Comme les figures 5.14 (Q_2) et 5.15 (MSE) l'attestent, La hiérarchie précédente est quelque peu altérée lorsque la vraie dimension est utilisée. En effet, à partir de $n = 400$ le Q_2 obtenu par mRML est en moyenne aussi bon que celui de mkPPVF. Ces deux méthodes se complètent très bien, suggérant un mélange de modèles comme indiqué à la section 5.1.

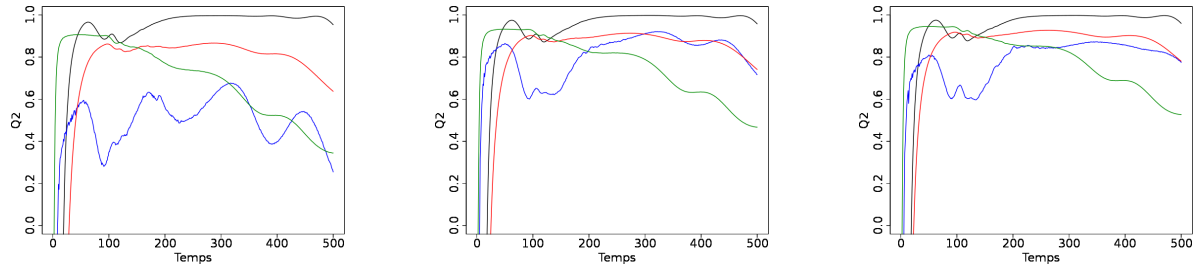

Figure 5.14: Q_2 pour $n = 200$, $n = 400$ et $n = 600$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

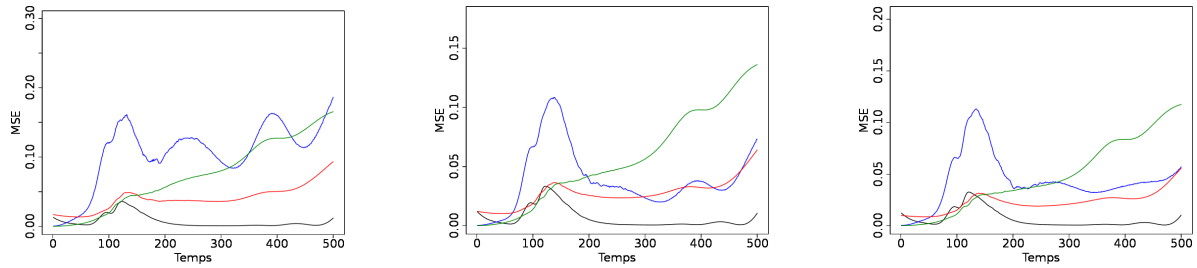

Figure 5.15: MSE pour $n = 200$, $n = 400$ et $n = 600$ de gauche à droite. Les courbes des méthodes mACPF, mRML, mkPPVF et mACPFL sont respectivement en noir, bleu, vert et rouge.

5.4 Tests sur données réelles simulées

Les deux jeux de données CATHARE illustrés au fil des chapitres sont à présent utilisés afin de tester la capacité de prédiction du métamodèle selon le processus de validation croisée indiqué au paragraphe 5.2.2, avec $q = 10$ pour CATHARE I et $q = 30$ pour CATHARE II. Comme précédemment, aucun résultat n'est indiqué pour la méthode LPcaML, les prédictions étant en moyenne trop mauvaises à cause des erreurs de classification (supervisée). Nous avons essayé d'utiliser un modèle mixte comprenant d'abord mACPF, avant d'apprendre les résidus via mRML. Ses performances ne dépassent pas celle utilisant uniquement l'ACP fonctionnelle, et les résultats ne sont pas indiqués ici.

5.4.1 CATHARE I

Ce benchmark est constitué de 100 courbes échantillonnées sur 168 points de discrétisation (rappelées sur la figure 5.16). La grille est commune à toutes les fonctions mais non régulière ; nous indiquons cependant seulement "temps discrétisé" comme titre des axes temporels, afin de ne pas alourdir les figures. Les entrées sont au nombre de quatre, et la dimension est estimée à cinq avec $dest_{levi}$. Peu d'améliorations sont visibles lorsque cette dernière est augmentée. Nous la fixons donc à cette valeur pour tous les tests de ce premier paragraphe. Notons qu'il n'est pas choquant d'estimer une dimension supérieure au nombre de variables d'entrée. En effet, un (possible) trop faible échantillonnage auquel s'ajoute un probable bruit numérique peut altérer la dimension théorique des données.

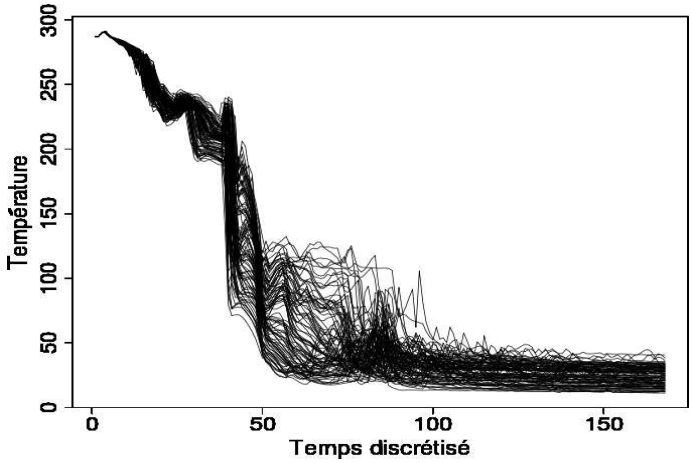

Figure 5.16: Les 100 courbes du jeu de données CATHARE I.

5.4.1.1 Avec clustering

Tout comme pour le premier exemple artificiel présenté plus haut, on choisit de fixer le nombre de groupes (à deux cette fois). Des résultats plus réalistes utilisant la méthodologie de détermination automatique du nombre de classes se situeraient quelques part entre ceux affichés dans ce paragraphe et ceux du suivant. Toutefois, imposer deux clusters permet de visualiser le gain ("maximal théorique") réalisé à l'aide de l'étape de classification. Les courbes d'erreur (respectivement Q_2 et MSE) indiquées sur les figures 5.17 et 5.19 sont obtenues de cette manière. Les écarts-type correspondants sont affichés respectivement sur les figures 5.18 et 5.20.

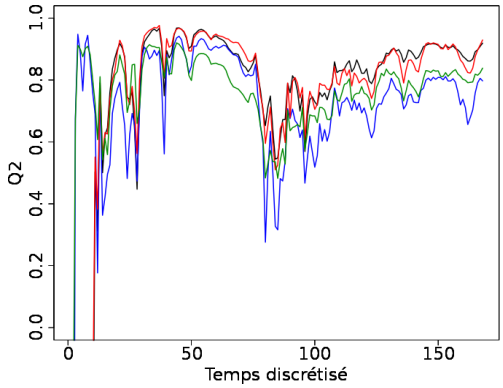

Figure 5.17: Q_2 empirique via les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

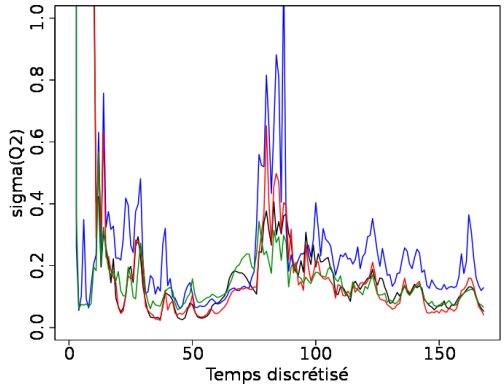

Figure 5.18: Écarts-type du Q_2 pour les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

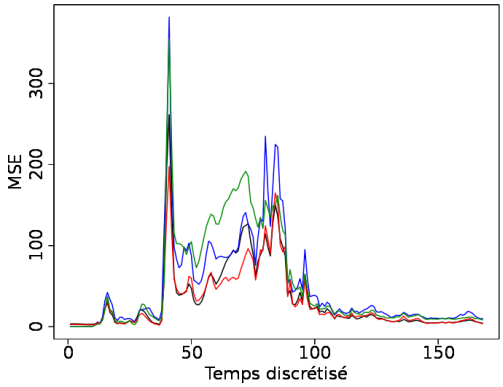

Figure 5.19: MSE empirique via les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

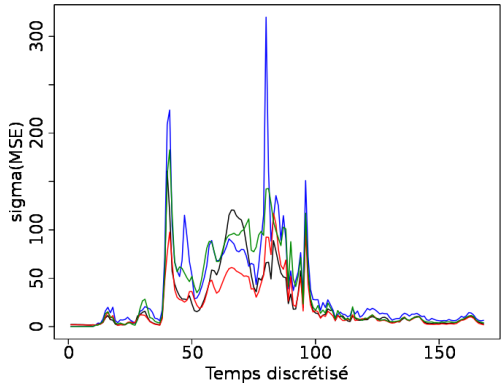

Figure 5.20: Écarts-type de la MSE pour mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

Comme l'allure des courbes et la MSE l'indiquent, la zone la plus difficile à prédire se situe entre les temps discrétisés (renormalisés) 40 et 100. Il est donc remarquable que le Q_2 soit au-delà de 0.8 pour toutes les méthodes entre 40 et 70 environ. Les principales différences entre les prédicteurs s'observent sur le Q_2 au tout début, proche de $t = 0$ où mRML et mkPPVF sont clairement meilleurs (bien que l'écart en terme de MSE soit négligeable), ainsi que dans la phase finale à partir de $t = 100$. Les approches basées sur l'ACP (locale ou non) reprennent alors le dessus. Assez logiquement, l'écart-type sur la MSE croît lorsque

cette dernière augmente. Ainsi, les courbes des figures 5.19 et 5.20 sont très similaires. Enfin, la hiérarchie observée sur les valeurs des écarts-type dans les exemples vus jusqu'alors se retrouve, la méthode utilisant RML donnant les prédictions les plus variables.

Quelques courbes prédites

Les figures 5.22, 5.23, 5.24 et 5.25 illustrent quelques estimations de courbes (les fonctions cibles étant visibles sur la figure 5.21), assez typiques de chacune des méthodes utilisées.

On constate que mRML et mACPFL estiment des courbes assez peu régulières, tandis que mkPPVF prédit des courbes trop lissées en effectuant les moyennes locales ; mACPF semble présenter un bon compromis. Au niveau de la qualité des prédictions la courbe rouge est bien estimée avec chacune des méthodes, tandis que celle en noir n'est prédite de manière acceptable qu'avec mRML. Elle constitue en effet presque un outlier dans le jeu de données, les oscillations présentes un peu après l'abscisse 100 ne se retrouvant pas sur les autres fonctions. Notons que la bosse visible juste avant l'abscisse 150 sur la courbe rouge n'est jamais reproduite. Celle-ci pourrait correspondre à une petite erreur dans la résolution des équations différentielles au sein du code CATHARE, car il est étrange que la température remonte à ce stade. Enfin, le sursaut sur la courbe mauve vers l'abscisse 45 ainsi que les irrégularités de la courbe bleue autour de l'abscisse 80 sont prédites (de façon atténuée) uniquement via mkPPVF.

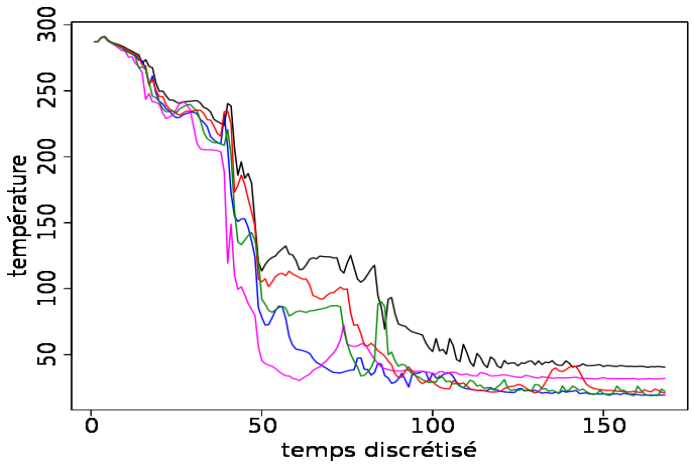

Figure 5.21: Cinq courbes en sortie de CATHARE I

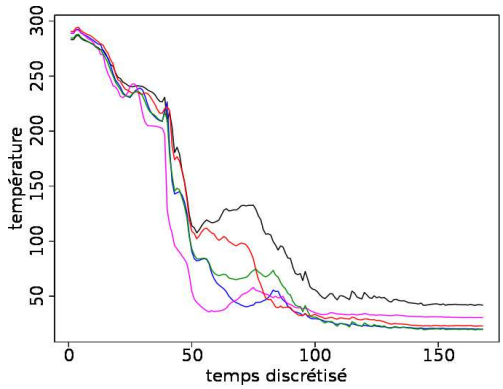

Figure 5.22: Prédictions obtenues par mACPF.

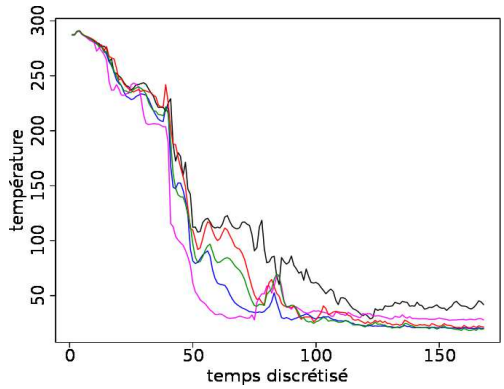

Figure 5.23: Prédictions obtenues par mRML.

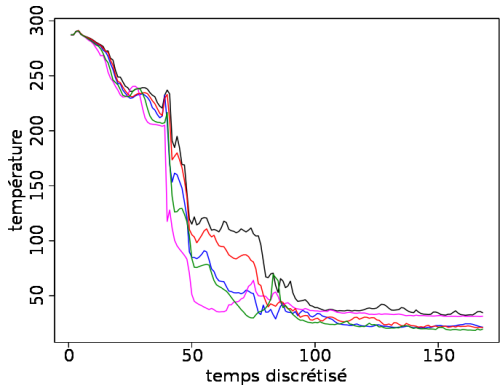

Figure 5.24: Prédictions obtenues par mkPPVF.

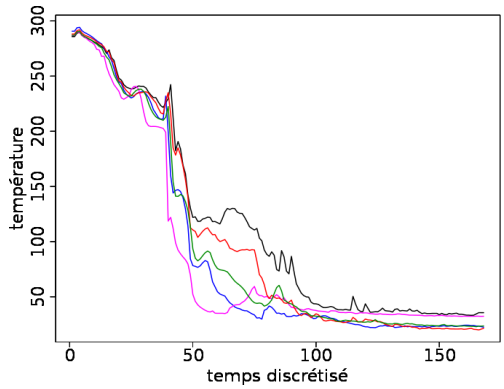

Figure 5.25: Prédictions obtenues par mACPFL.

5.4.1.2 Sans clustering

Il est possible de désactiver l'étape de classification non supervisée au sein du package R. Les courbes d'erreur (respectivement Q_2 et MSE) indiquées sur les figures 5.26 et 5.28 sont obtenues de cette façon. Les écarts-type correspondants sont affichés respectivement sur les figures 5.27 et 5.29. Ces derniers sont légèrement supérieurs à leurs homologues avec l'étape de classification initiale. L'erreur réalisée par mRML est assez nettement augmentée, indiquant que l'étape de clustering est utile au moins à cette méthode. Cependant, en moyenne, seule une très légère diminution du Q_2 s'observe.

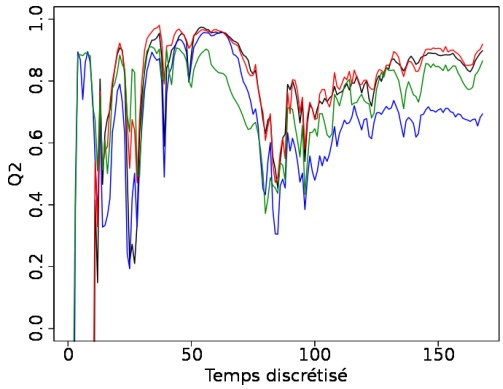

Figure 5.26: Q_2 empirique via les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

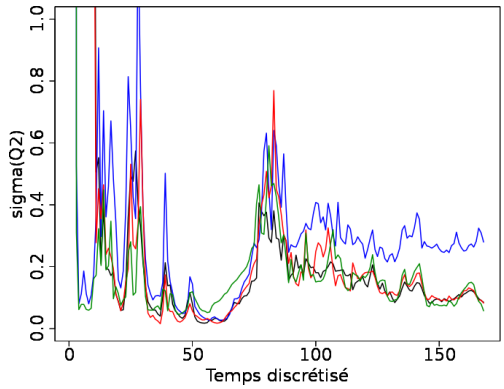

Figure 5.27: Écarts-type du Q_2 pour les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

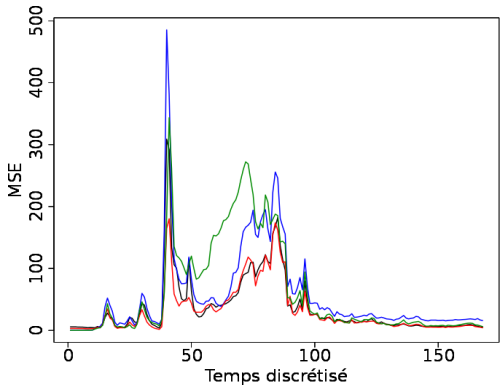

Figure 5.28: MSE empirique via les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

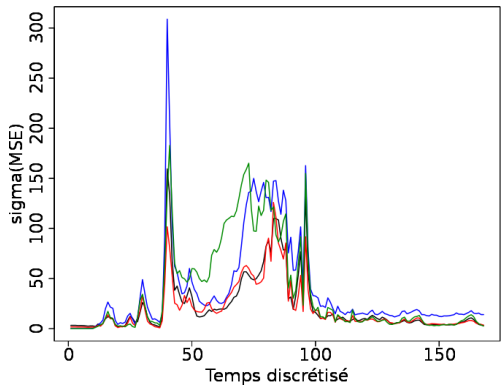

Figure 5.29: Écarts-type de la MSE pour mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

Quelques courbes prédites

Les figures 5.30, 5.31, 5.32 et 5.33 illustrent les estimations des courbes indiquées sur la figure 5.21. Le bilan est globalement le même que précédemment, avec quelques changements notables. En effet, les oscillations autour de l'abscisse 50 sur la courbe rouge prédite par mRML disparaissent, tandis que la courbe noire se retrouve nettement moins bien reproduite par cette même méthode. Malgré une baisse des performances moyennes, la suppression de l'étape de clustering permet aux modèles ACPF et ACPFL de retrouver le pic sur la courbe en vert à l'abscisse 80. Quelques réglages supplémentaires pourraient donc être nécessaires afin de mieux profiter de la classification non supervisée.

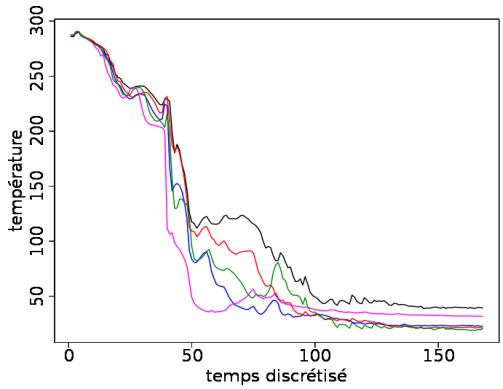

Figure 5.30: Prédictions obtenues par mACPF.

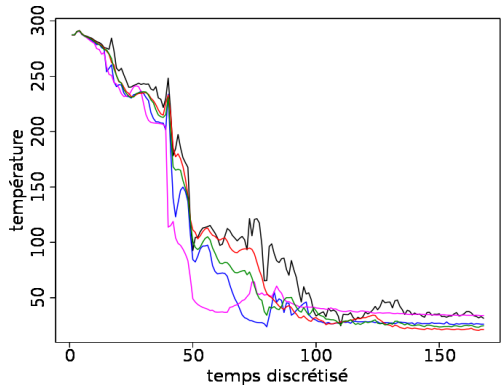

Figure 5.31: Prédictions obtenues par mRML.

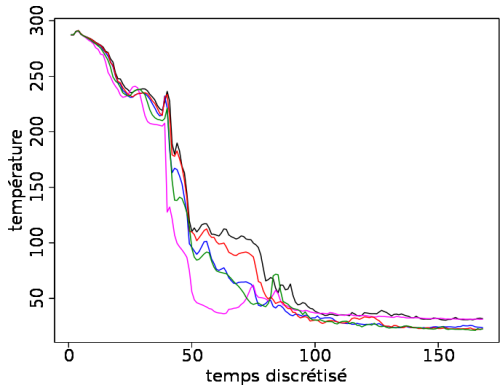

Figure 5.32: Prédictions obtenues par mkPPVF.

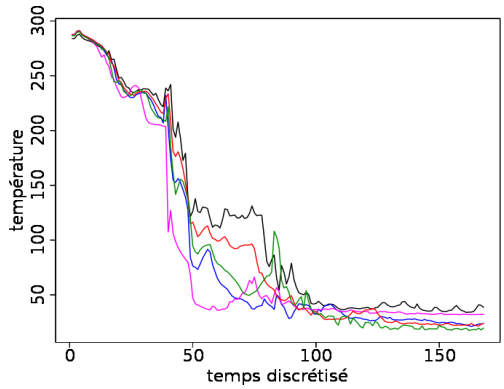

Figure 5.33: Prédictions obtenues par mACPFL.

5.4.2 CATHARE II

Ce dernier jeu de données a aussi été utilisé au cours des chapitres précédents. Il est constitué de 600 courbes échantillonnées sur 414 points de discrétisation, rappelées sur la figure 5.34. Les entrées sont en dimension onze ($p = 11$), et la dimension estimée par l'algorithme d_{levi} vaut 8. C'est cette valeur que nous utilisons. Le clustering apporte cette fois un gain très marginal sur un petit nombre de points de discrétisation, la qualité de la prédiction étant nettement dégradée ailleurs. C'est pourquoi seuls les résultats sans classification non supervisée sont indiqués.

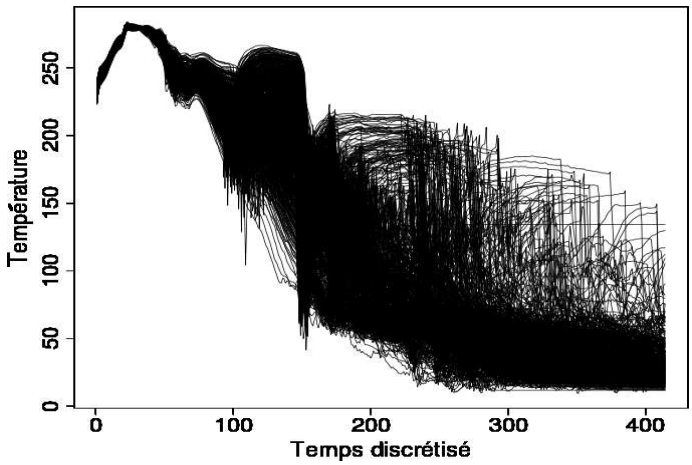

Figure 5.34: Les 600 courbes du jeu de données CATHARE II.

Les figures 5.35 et 5.37 montrent respectivement les Q_2 et MSE, tandis que les figures 5.36 et 5.38 indiquent les écarts-type correspondants. La première chose à noter est la dégradation des performances de la méthode mACPFL, qui se retrouve derrière toutes les autres, en plus d'être celle donnant les estimation les plus variables. On constate également que mRML est la seule à être compétitive avec l'ACP globale, notamment autour de l'abscisse 200. De plus, l'écart-type de cette dernière se réduit jusqu'à presque égaler celui obtenu via mACPF, tandis que celui observé pour la MSE sur la méthode mkPPCV augmente considérablement.

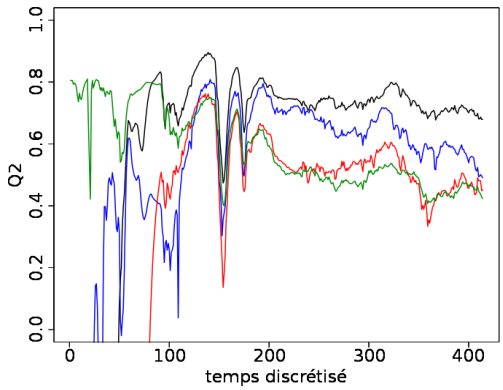

Figure 5.35: Q_2 empirique via les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

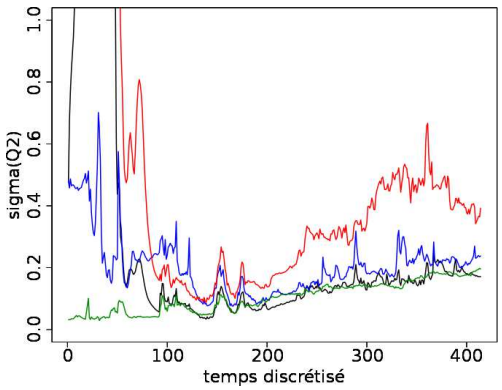

Figure 5.36: Écarts-type du Q_2 pour les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

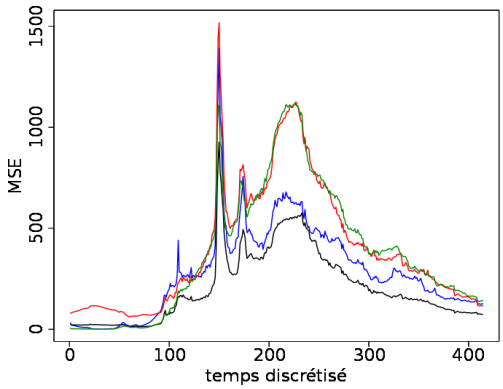

Figure 5.37: MSE empirique via les modèles mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

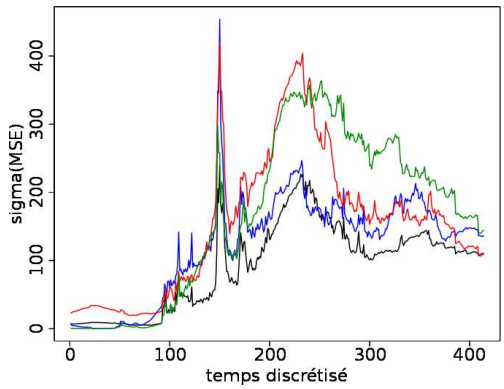

Figure 5.38: Écarts-type de la MSE pour mACPF, mRML, mkPPVF et mACPFL, respectivement en noir, bleu, vert et rouge.

Quelques courbes prédites

Les figures 5.40,5.41,5.42 et 5.43 illustrent quelques estimations de courbes (les fonctions cibles étant visibles sur la figure 5.39). Tout d'abord, d'une manière générale, les prédictions obtenues par mACPF et mkPPVF sont nettement trop lissées. Cela s'explique par l'aspect lisse des fonctions de la base ACP(F) pour la première, et par les moyennes locales effaçant les irrégularités pour kPPVF. Il est cependant étonnant de constater qu'aucune méthode ne parvient à estimer les irrégularités sur la courbe bleu, celle-ci se retrouvant systématiquement beaucoup trop lissée. La fonction en vert subit presque le même sort, quelques légers sauts étant prédits avec mACPFL seulement. Ce dernier modèle est, avec mRML, le seul permettant d'estimer les sauts de la courbe en noir à partir de l'abscisse 250. Ceux-ci sont toutefois légèrement décalés, et globalement trop décroissants. Enfin, la température le long de la courbe en mauve diminue trop brusquement via les méthodes linéaires (locale et globale). Celle-ci est assez bien prédite par les deux autres modèles.

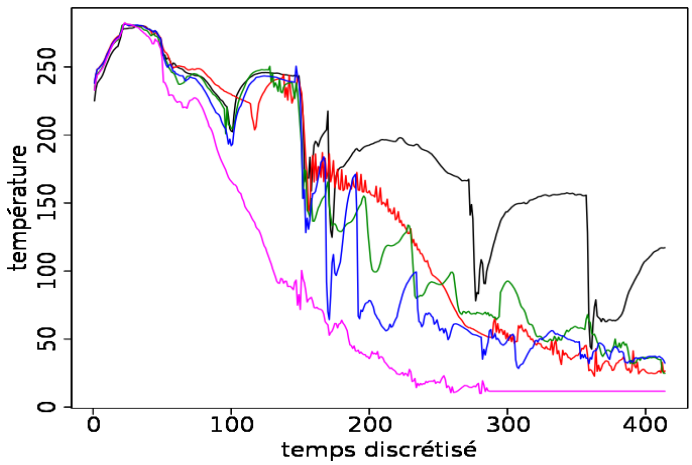

Figure 5.39: Cinq courbes en sortie de CATHARE II

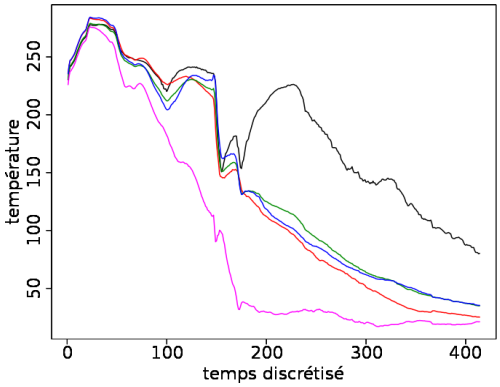

Figure 5.40: Prédictions obtenues par mACPF.

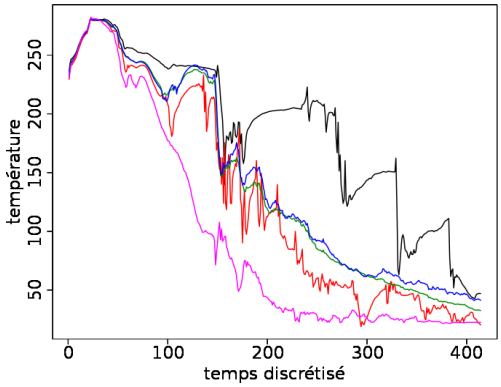

Figure 5.41: Prédictions obtenues par mRML.

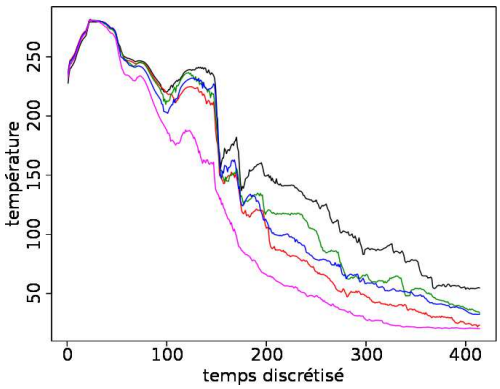

Figure 5.42: Prédictions obtenues par mkPPVF.

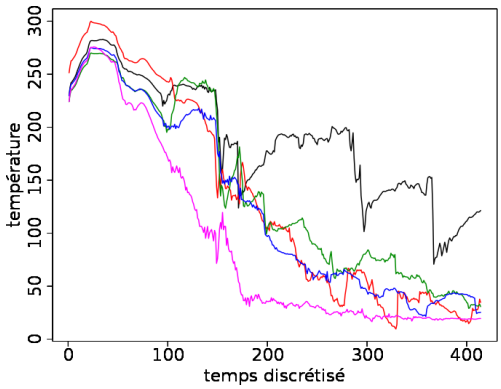

Figure 5.43: Prédictions obtenues par mACPFL.

5.5 Bilan

L'étape de clustering automatique est efficace sur certains jeux de données, bien que des optimisations soient nécessaires pour en tirer pleinement profit. Elle est cependant assez peu utile sur les données réelles dont nous disposons actuellement, celles-ci ne s'organisant pas forcément en classes conformément à l'hypothèse. Il se pourrait également qu'il y ait trop peu de données significatives pour bien distinguer différents groupes.

Une analyse plus poussée montre que les réductions de dimension linéaire et non linéaire sont complémentaires (elles mènent à des modèles prédisant bien certains ensembles de courbes quasi disjoints). Celle-ci n'est pas présentée ici afin de ne pas trop alourdir ce dernier chapitre. L'apprentissage de type "boosting" sur les résidus n'a cependant pas donné les résultats escomptés. Une étude plus approfondie serait nécessaire pour réussir à fusionner les deux types de méthodes.

Globalement, en ce qui concerne les jeux de données CATHARE on note de bonnes performances des modèles mACPF et mRML, la méthode mACPF l'emportant en moyenne. En effet l'erreur effectuée par mRML est accentuée par les irrégularités prédites aux mauvais endroits. Cette dernière est moins efficace sur les jeux de données artificiels, pour lesquels mACPFL est en moyenne la meilleure. Il semble que kPPVF soit d'autant moins précis que la dimension intrinsèque est élevée, contrairement à mACPF qui conserve de bonnes performances dans tous les cas. mRML, quant à lui, est trop sensible au bruit pour afficher des résultats supérieurs à ceux des approches linéaires.

La régression PPR a été choisie pour son adaptivité, son petit nombre de paramètres et sa vitesse. D'autres algorithmes pourraient cependant fournir de meilleurs résultats, notamment celui indiqué au paragraphe 4.3.3, qui présente les mêmes avantages en plus d'effectuer la réduction de dimension − linéaire − simultanément. Celui-ci pourrait ne pas apporter beaucoup au cas général d'un espace de sorties trop complexe, mais améliorer les résultats sur les jeux de données CATHARE, ceux-ci étant assez bien modélisés par des approches linéaires.

Enfin, aucun temps d'exécution n'a été reporté car l'accent n'a pas été mis sur les optimisations algorithmiques, réservées à une version ultérieure du package R. Dans la version actuelle, certains calculs utilisant beaucoup la validation croisée ont mis jusqu'à 15 heures pour s'exécuter (sur un processeur cadencé à 2.33GHz, utilisant 4Go de RAM). Toutefois, la plupart se sont terminés en moins d'une heure.

Chapitre 6

Conclusion

Un schéma de régression pour entrées vectorielles et sorties fonctionnelles $1D$ a été présenté. Celui-ci comporte une étape de clustering et prend relativement peu de paramètres, l'objectif étant de déterminer ces derniers en fonction des données. La réduction de dimension des sorties s'effectue non linéairement, permettant de modéliser efficacement une large gamme de codes de calcul. Comme le chapitre d'applications le montre, ses performances sont toutefois perfectibles sur les jeux de données réels, bien qu'il soit nettement complémentaire des approches avec réduction de dimension linéaire.

Dans la mesure où elle n'avait pas vraiment été utilisée en lien avec les métamodèles, l'étape de classification non supervisée constitue une contribution nouvelle. Celle-ci peut être utile a priori si des tendances distinctes s'observent, voire a posteriori pour guider l'échantillonnage en entrée avant de refaire des calculs. Une extension future devrait consister en la prise en compte de la nature fonctionnelle des données dans l'étape de clustering, sans toutefois aller vers un modèle paramétrique. On pourrait par exemple utiliser une distance mixte faisant intervenir les dérivées, celles-ci étant évaluées à l'aide de splines de lissage.

Bien que le sujet ne soit pas nouveau, la réduction de dimension correspond également à une tentative nouvelle dans le domaine des métamodèles. Concernant cette étape, il semble envisageable de tirer partie des spécificités de $\mathcal{C}([a,b])$. En effet, en partant de la base des fonctions principales ζ_j obtenues via une ACP il est possible de modifier légèrement les applications $t \mapsto t\zeta_j$ dans l'esprit des courbes principales [156], afin mieux s'adapter aux données. Une composante peut alors être visualisée comme un chemin continu dans l'espace des courbes, voire interprétée physiquement. A contrario, les techniques de réduction de dimension présentées au chapitre 3 agissent un peu comme des boîtes noires, bien que certaines propriétés locales soient conservées pour celles que nous avons retenu.

Quelques perspectives

Il pourrait être utile d'effectuer des opérations préliminaires sur les courbes (recalage, débruitage, etc.) avant d'appliquer l'algorithme. Certaines sont présentées dans le livre de Ramsay et Silverman [258]. Cela nécessite cependant l'introduction de nouveaux paramètres. Par exemple, y_i peut être remplacée par $t \mapsto y_i(t - \alpha_i)$ avec $\alpha_i \in \mathbb{R}$, à supposer que les domaines de définition des courbes puissent être étendus.

Les sorties fonctionnelles présentent des discontinuités visuelles (liées au pas de discrétisation du code de calcul CATHARE). Bien que la continuité soit assurée en théorie, il semble intéressant de juxtaposer plusieurs modèles pour gagner en qualité de prédiction. La principale difficulté réside dans la détermination des points de discontinuité, ceux-ci variant légèrement d'une courbe à l'autre. De plus, les modèles peuvent éventuellement être adaptés aux domaines de continuité suivant les caractéristiques des données.

Le code de calcul CATHARE comportant trois réponses fonctionnelles corrélées, il est envisageable d'en tirer parti pour améliorer le modèle. En supposant la dimension réduite via une base non adaptée aux données (ou bien tenant compte des trois types de réponse), on pourrait appliquer une des méthodes évoquées en introduction de la section 4.1 pour prédire les triplets constitués des coefficients sur chaque composante. Une recherche plus approfondie serait nécessaire pour parvenir à prendre en compte toutes les dépendances, et éventuellement étendre le modèle à une base adaptée voire à une méthode de réduction de dimension non linéaire.

Enfin, en complément des pistes de recherche indiquées à la fin de chaque chapitre il faudrait étudier l'analyse de sensibilité multi-sorties sur les représentations réduites des courbes. Les indices de sensibilités basés sur l'entropie (définis par Krzykacz-Hausmann [192]) pourraient convenir à cette tâche car leur définition se généralise naturellement au cas multidimensionnel. Il resterait alors à déterminer les vitesses de convergence et à comparer les indices obtenus pour différentes techniques de réduction de dimension.

Annexe A

Structure des sorties du code

Nous avons fait l'hypothèse d'une structure de variété sur les fonctions en sortie du code. C'est pourquoi nous nous attachons ici à formaliser cette notion et montrons que les approximations discrètes effectuées sont justifiées. Une première partie est consacrée à des définitions nécessaires pour définir une métrique sur une variété, puis des applications autour de cette métrique seront abordées.

Sauf mention contraire, les figures de ce chapitre sont empruntées aux livres de John M. Lee [206; 205].

A.1 Variétés riemanniennes

Cette section décrit la structure des courbes telle que nous la supposons en sortie, selon le modèle décrit historiquement par Riemann [262]. Bien que très générale, celle-ci constitue une hypothèse pouvant se révéler fausse. Nous l'utilisons néanmoins, faute de pouvoir accéder à la vraie structure. Elle est basée sur la constatation qu'une surface bidimensionnelle en $3D$ (par exemple) peut ne pas être représentée optimalement par une base orthonormée, comme le montre la figure A.1. Cela est bien sûr généralisable à d'autres espaces en dimension supérieure.

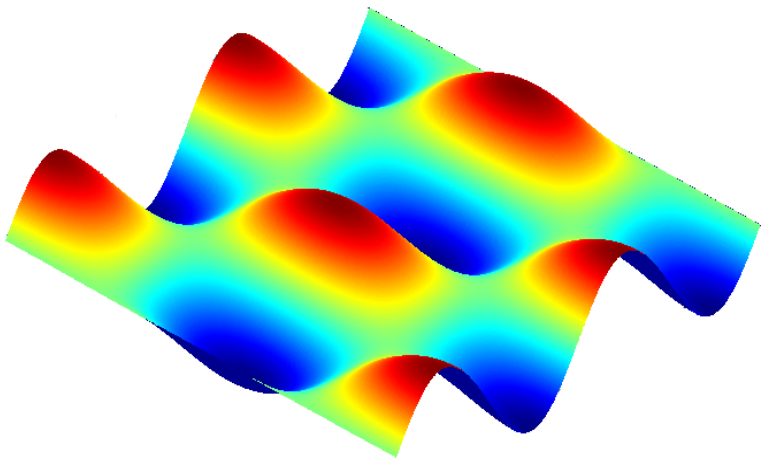

Figure A.1: Surface "non linéaire" dans $\mathbb{R}^3$ (personnelle).

A.1.1 Variété

Définition A.1.1 *Un* espace topologique *consiste en un couple $(E, \mathcal{O})$ où $\mathcal{O}$ est un ensemble de parties de E, appelées "ouverts", et vérifiant les trois axiomes suivants :*

- *l'ensemble vide est dans $\mathcal{O}$;*
- *une réunion quelconque d'ouverts est dans $\mathcal{O}$;*
- *une intersection finie d'ouverts est dans $\mathcal{O}$.*

$\mathcal{O}$ est appelée *topologie* de E. Il y en a en général beaucoup, parmi lesquelles on peut citer la topologie discrète ($\mathcal{O}$ contient tous les singletons de E), et grossière ($\mathcal{O} = \{, E\}$). Les cas intéressants se situent entre ces deux extrêmes. Nous allons spécialiser petit à petit cette définition pour obtenir une structure sur laquelle on peut travailler.

Définition A.1.2 *Une* variété *(topologique) est un espace topologique $(E, \mathcal{O})$ vérifiant les propriétés suivantes :*

- *$(E, \mathcal{O})$ est séparé : $\forall u, u' \in E / u \neq u'$, $\exists O, O' \in \mathcal{O} / u \in O, u' \in O', O \cap O' =$;*
- *$(E, \mathcal{O})$ est à base dénombrable : il existe un ensemble dénombrable d'ouverts $(\mathcal{O}_n)_{n\in\mathbb{N}}$ engendrant $\mathcal{O}$ par réunions quelconques ;*
- *$(E, \mathcal{O})$ est localement homéomorphe à $\mathbb{R}^d$ (muni de sa topologie usuelle) pour un certain $d \in \mathbb{N}$, éventuellement dépendant du point.*

Les deux premiers points sont assez techniques et vérifiés dans la plupart des cas pratiques. Mentionnons cependant le contre-exemple de $(\mathbb{N}, \mathcal{F})$ où $\mathcal{F}$ est engendré par l'ensemble vide et les complémentaires d'ensembles finis (filtre de Fréchet) ; un tel espace n'est pas séparé car tout voisinage d'un point contient nécessairement un "voisinage de l'infini". Concernant la seconde propriété, $\mathbb{R}$ muni de la topologie discrète par exemple n'est clairement pas à base dénombrable. Le troisième point est vérifié en utilisant la définition topologique d'une application continue : les images réciproques d'ouverts de l'espace d'arrivée doivent être des ouverts de l'espace de départ. Dans le cas d'une variété connexe on peut montrer que d est indépendant du point (via le théorème d'invariance du domaine, voir par exemple le livre de Hatcher [159], paragraphe 2.B), et on peut parler de dimension (globale) de la variété. Supposant que la phase de clustering a déterminé les parties connexes de la variété entière, on peut supposer travailler sur une composante connexe (sous-variété).

Remarque : $(\mathcal{C}([a, b]), \|.\|_{L_2})$ est un espace topologique séparé, à base dénombrable d'ouverts (on peut considérer par exemple les $B\left(P, \frac{1}{k}\right)$ avec P polynôme à coefficients rationnels, $k \in \mathbb{N}^*$). En revanche $\mathcal{C}([a, b]$ est un espace vectoriel de dimension infini (la famille $\{t \mapsto t^k\}_{k\in\mathbb{N}}$ est libre), donc aucun de ses ouverts ne peut être homéomorphe à un $\mathbb{R}^d$.

L'espace V d'une variété $(V, \mathcal{O})$ est défini par l'ensemble $\mathcal{H}$ des homéomorphismes d'un ouvert de V dans un ouvert de $\mathbb{R}^d$, tel que l'indique la définition précédente. Nous avons en particulier $\cup_{\theta\in\mathcal{H}} \mathrm{Im}(\theta^{-1}) = V$: l'union des images réciproques (domaines) des homéomorphismes recouvre V. Toute famille d'homéomorphisme (nécessairement un sous-ensemble de

$\mathcal{H}$) vérifiant cette condition est appelée un *atlas* de la variété, ses éléments étant nommés de façon imagée des *cartes*. Étant donnés deux homéomorphismes $\theta, \theta' \in \mathcal{H}$ définis respectivement sur U, U' avec $U \cap U' \neq$, l'application de changement de carte (ou changement de coordonnées) associée est

$$\begin{array}{rcl} h(U \cap U') & \rightarrow & h'(U \cap U') \\ x & \mapsto & \theta' \circ \theta^{-1}(x) \end{array}$$

Définition A.1.3 *Une variété V est dite* différentielle *(ou* différentiable*) de classe $\mathcal{C}^k$ s'il existe un atlas tel que toutes les applications de changement de carte soient $\mathcal{C}^k$.*

Les changements de cartes étant bijectifs, on en déduit que ce sont des $\mathcal{C}^k$-difféomorphismes dans le cas de la définition. Nous n'avons jusqu'alors pas de possibilités d'effectuer des calculs de distances ou volumes (entre autres) sur la variété. Les sections suivantes s'attachent à pallier ce manque.

A.1.2 Espace tangent

Tout ce qui suit est développé plus en détail dans la plupart des ouvrages modernes traitant de géométrie différentielle (par exemple ceux de Spivak [299]; Lafontaine [196]; Chern et al. [65]; Lee [205]). Pour une sous-variété de $\mathbb{R}^m$, l'espace tangent est facile à décrire (informellement) et correspond à l'intuition physique : c'est l'ensemble des vecteurs vitesse issus de u pour un point se délaçant sur V.

Définition A.1.4 *Soit V une sous-variété différentielle de $\mathbb{R}^m$, de dimension d ($d \leq m$). L'espace tangent $T_u V$ en un point $u \in V$ correspond à l'union des vecteurs tangents en u aux courbes (de classe $\mathcal{C}^1$) tracées sur la variété et contenant u.*

La figure A.2 illustre une courbe γ tracée sur V, ainsi qu'un vecteur tangent en $\gamma(t_0)$.

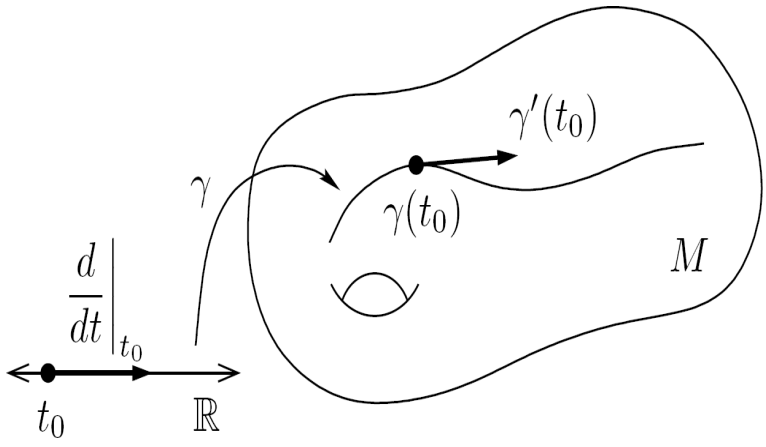

Figure A.2: Vecteur tangent sur une sous-variété de $\mathbb{R}^3$.

On peut montrer que pour tout $u \in V$ ce dernier ensemble de vecteurs est un sous-espace vectoriel de $\mathbb{R}^m$, de dimension d. Soit v un des ses éléments. On lui associe l'opérateur de dérivation ξ_v comme suit, à u fixé, $k \geq 1$.

$$\begin{array}{rccl} \xi_v : & \mathcal{C}^k(V, \mathbb{R}) & \rightarrow & \mathbb{R} \\ & f & \mapsto & \frac{d}{dt} f(u + tv) \end{array}$$

Ce dernier opérateur est linéaire et vérifie la relation suivante

$$\forall f, h \in \mathcal{C}^k(V, \mathbb{R}) \, , \, \xi_v(fh) = f(u)\xi_v(h) + h(u)\xi_v(f) \, .$$

L'ensemble des opérateurs vérifiant cette dernière relation forme clairement un espace vectoriel. En fait, l'application qui à v associe ξ_v est un isomorphisme (voir par exemple Lee [205], proposition 3.2) ; on identifie ainsi chaque vecteur tangent à une dérivation. C'est (entre autres) via la relation ci-dessus que l'on peut définir l'espace tangent en $u \in V$ pour une variété (différentielle) V quelconque.

Définition A.1.5 *Soit V une variété différentielle. L'*espace tangent *en un point $u \in V$ est noté T_uV, et constitué de l'ensemble des opérateurs linéaires ξ vérifiant la condition suivante*

$$\forall f, h \in \mathcal{C}^k(V, \mathbb{R}) \, , \, \xi(fh) = f(u)\xi(h) + h(u)\xi(f) \, .$$

Ses éléments sont appelés vecteurs tangents *(en référence à l'intuition géométrique).*

On peut montrer que l'espace vectoriel formé par les vecteurs tangents en u est de dimension d, isomorphe à l'espace des dérivations partielles en $\theta(u) \in \mathbb{R}^m$, (U, θ) étant une carte locale avec $u \in U$. Ainsi un vecteur tangent en u est identifiable à une application de la forme $\sum_{i=1}^{d} v_i \frac{\partial}{\partial x_i}\Big|_{h(p)}$. Malgré le formalisme abstrait des définitions, cette dernière identification permet d'effectuer certaines opérations sur V "comme si" on était dans $\mathbb{R}^m$. Mesurer la "distance" entre deux points d'une variété nécessite cependant l'ajout d'une structure supplémentaire, comme nous le voyons dans la section A.1.5.Pour l'instant, attardons nous un peu sur la visualisation de l'espace tangent dans le cas d'un espace de courbes.

A.1.3 Visualisation d'un espace tangent fonctionnel

Nous revenons dans ce paragraphe au cas de la thèse, à savoir un ensemble de fonctions dans $\mathcal{C}([a, b])$ (l'évolution d'un paramètre physique étant continue). La variété (de dimension d) est ainsi notée $\mathcal{U}$, et supposée de classe $\mathbb{C}^k$. Il est facile de voir que si $\mathcal{U}$ forme un espace vectoriel, l'espace tangent en un point est identifié à $\mathcal{U}$ elle-même. Mais dans le cas d'un espace vectoriel on utiliserait une base orthonormée et n'aurait pas besoin de tous ces outils géométriques. L'objectif de ce paragraphe est donc la réalisation d'une identification valable dans le cas général d'une variété de $\mathcal{C}([a, b])$. Soit $u \in V$, et (U, θ) une carte locale en u. On cherche à identifier l'espace tangent $T_u\mathcal{U}$ en u à un sous-espace vectoriel de $\mathcal{C}([a, b])$, conformément à l'intuition.

Soient $\alpha < \beta \in \mathbb{R}, t_0 \in]\alpha, \beta[$ et $\gamma :]\alpha, \beta[\to \mathcal{U} \subset \mathcal{C}([a, b])$ une courbe $\mathcal{C}^1$ passant par u en t_0 : $\gamma(t_0) = u$. On peut lui associer la fonction

$$f_\gamma = \lim_{t \to 0} \frac{\gamma(t_0 + t) - \gamma(t_0)}{t}$$

Si γ_1 et γ_2 (courbes $\mathcal{C}^1$ passant par u en t_0) sont égales dans un voisinage de t_0 $-$ à un reparamétrage affine près $-$, alors f_{γ_1} et f_{γ_2} sont colinéaires. Cela définit une classe d'équivalence sur les courbes $\mathcal{C}^1$ passant par u en t_0 pour la relation "représenter la même fonction f à une constante multiplicative près". Choisissons alors de désigner toute la classe

d'équivalence par la notation $\bar{\gamma}$, et d'écrire $\bar{\gamma}_f$ pour l'unique classe représentant f.

Soit F l'ensemble des courbes $f_{\bar{\gamma}}$; c'est un espace vectoriel ($\bar{\gamma}_{f+\lambda h} = \bar{\gamma}_f + \lambda\bar{\gamma}_h$), candidat pour être identifié à l'espace tangent. On peut alors définir l'application

$$\begin{array}{rccc} \xi: & F & \to & \mathbb{R}^d \\ & f & \mapsto & (\varphi \circ \bar{\gamma}_f)'(t_0)\,. \end{array}$$

θ étant bijective, les images $\theta(\bar{\gamma})$ tracent tous les chemins $\mathcal{C}^1$ (à reparamétrage affine près) passant par $\theta(u)$ dans $\mathrm{Im}(\theta) \subset \mathbb{R}^d$. $\mathrm{Im}(\xi)$ décrit alors l'espace tangent en $\theta(u)$ dans $\mathbb{R}^d$. On voit que ξ est linéaire en écrivant la "chain rule" de l'application composée en t_0 :

$$\forall f \in F\,,\; D_{t_0}(\theta \circ \bar{\gamma}_f) = D_{\bar{\gamma}(t_0)}(\theta) \circ D_{t_0}(\bar{\gamma})\,.$$

En effet $D_{\bar{\gamma}(t_0)}(\theta)$ est constante car $\bar{\gamma}(t_0) = u$, et $\gamma \mapsto D_{t_0}(\gamma)$ est linéaire en γ (par définition de la différentielle dans des espaces de Banach). F est alors de dimension au moins d.

Définissons finalement l'application $\Gamma : F \to T_u\mathcal{U}, f \mapsto \psi_f$ définie comme suit.

$$\begin{array}{rccc} \psi_f: & \mathcal{C}^k(\mathcal{C}([a,b])) & \to & \mathbb{R} \\ & \zeta & \mapsto & (\zeta \circ \bar{\gamma}_f)'(t_0)\,, \end{array}$$

avec $T_u\mathcal{U} \subset \mathcal{L}(\mathcal{C}^k(\mathcal{C}([a,b])), \mathbb{R})$. En développant l'écriture de la dérivée comme précédemment on voit que Γ est linéaire. Il reste à montrer qu'elle est aussi injective. Soit alors $f \in F$ telle que $\forall \zeta \in \mathcal{C}^k(\mathcal{C}([a,b])), (\zeta \circ \bar{\gamma}_f)'(t_0) = 0$. En choisissant la famille $\{\zeta_t\}_{t\in[a,b]}$ des évaluation ponctuelles en t on voit que $\bar{\gamma}_f$ est localement constante en t_0. Ainsi $f = 0$.
$\mathrm{Im}(\Gamma)$ est alors de dimension au moins d, donc exactement $d = \dim(T_u\mathcal{U})$. Ainsi $\mathrm{Im}(\Gamma) = T_u\mathcal{U}$. On en déduit que F est de dimension d, isomorphe à $T_u\mathcal{U}$ via Γ.

On identifie ainsi $\psi_f \in T_u\mathcal{U}$ à $f \in \mathcal{C}([a,b])$, celle-ci étant une dérivée fonctionnelle conforme à l'intuition. D'après cette identification, afin d'évaluer la dimension de l'espace tangent en $y_0 \in \mathcal{U}$ il est suffisant de chercher une base de l'ensemble des fonctions du type f_γ. Ne disposant que d'un nombre fini d'échantillons, ces courbes sont simplement approximées par les différences du type $y_i - y_0$. Notons que cette identification est en fait générale à toute variété incluse dans un espace vectoriel, mais nous n'avons pas besoin de ce résultat.

A.1.4 Fibré tangent

Notre objectif est de pouvoir calculer des distances (de plus courts chemins) le long de la variété. Cela permettra de représenter les courbes y_i dans un graphe, puis d'appliquer un algorithme de réduction de dimension. Pour cela, il faut passer par la géométrie riemannienne, abordée par exemple dans les ouvrages de do Carmo [103], Lee [206] et Petersen [252]. Quelques définitions préliminaires sont nécessaires, données dans le cas général d'une variété V non nécessairement incluse dans $\mathcal{C}([a,b])$.

Définition A.1.6 *Le* fibré tangent TV *d'une variété différentielle* V *est l'union disjointe des*

espaces tangents en chaque point.

$$TV = \bigsqcup_{u \in V} T_u V \, .$$

Un élément de cet espace est noté (u, X) *avec* $u \in V, X \in T_u V$.

En général on ne mentionne pas le point u, réalisant l'identification légèrement abusive $(u, X) \equiv X$ pour un élément de TV.

Nous disposons alors de la projection naturelle suivante.

$$\begin{array}{rccc} \pi : & TV & \to & V \\ & (u, X) & \mapsto & u \, . \end{array}$$

(ou $X \mapsto u$ en notation abrégée).

Le fibré tangent TV possède une topologie naturelle. En effet, considérons une carte locale (U, θ) de V (U est l'ensemble de départ de θ, $\theta_1, \ldots, \theta_d$ sont ses fonctions coordonnées), et définissons l'application

$$\begin{array}{rccc} \tilde{\theta} : & \pi^{-1}(U) & \to & \mathbb{R}^{2d} \\ & \left(u, \sum_{i=1}^{d} v_i \frac{\partial}{\partial x_i}\Big|_{\theta(u)} \right) & \mapsto & (\theta_1(u), \ldots, \theta_d(u), v_1, \ldots, v_d) \end{array}$$

(via l'identification mentionnée à la fin du paragraphe précédent). Celle-ci est bijective, et on peut alors choisir comme ouverts de TV les images réciproques des ouverts de $\mathbb{R}^{2d}$ par $\tilde{\theta}$. Il est alors légitime de parler d'une application continue faisant intervenir l'espace TV. On peut même montrer que TV a une structure de variété différentielle de dimension $2d$ (où d est la dimension de V) ; voir par exemple Lee [205], lemme 5.5 pour une démonstration.

Définition A.1.7 *Un* champ de vecteurs C *(ou section continue du fibré tangent) est une application continue de* V *(ou d'un de ses ouverts) dans* TV *vérifiant la condition suivante.*

$$\pi \circ C = Id_V \, .$$

Cette dernière contrainte assure que l'image d'un point $u \in V$ par un champ de vecteur est bien dans $T_u V$. On écrit $C_u \equiv C(u)$ afin de ne pas surcharger les notations. La figure A.3 en présente une illustration en $3D$.

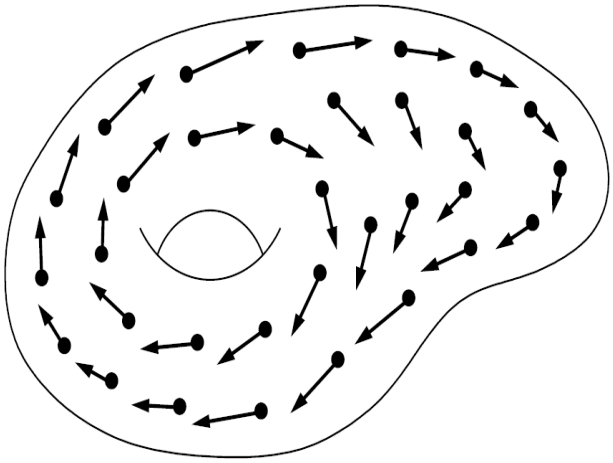

Figure A.3: Champ de vecteurs sur une sous-variété de $\mathbb{R}^3$.

Remarque : on peut définir de manière plus générale un *espace fibré* comme un couple d'espaces topologiques (E, B) (E étant appelé l'espace total et B la base), muni d'une projection continue $\pi : E \to B$ telle que toutes les images réciproques $\pi^{-1}(u)$ pour $u \in B$ soient homéomorphes à un espace fixe F appelé fibre. Cela généralise la notion de graphe d'une application $X \to Y$. En effet celle-ci peut être vue comme une section de $E = X \times Y$, avec $B = X$ et $F = Y$, toutes les images réciproques par π étant égales à Y. Un champ de vecteurs peut alors être pensé comme une "application généralisée", l'espace d'arrivée "variant continûment" d'un point à un autre.

A.1.5 Métriques riemanniennes

Les définitions du paragraphe précédent permettant l'introduction de l'outil de base en géométrie riemannienne.

Définition A.1.8 *Soit V une variété de classe $\mathcal{C}^k$. On appelle* métrique riemannienne *une application ω qui à $u \in V$ associe un produit scalaire (forme bilinéaire définie positive) dans l'espace tangent T_uV, et qui vérifie la condition de régularité suivante.*

Pour tous champs de vecteurs X, Y, $u \mapsto \omega_u(X_u, Y_u)$ est de classe $\mathcal{C}^k$.

Nous disposons ainsi d'un produit scalaire sur les espaces tangents de V, dépendant continûment du point. Bien qu'abstrait, celui-ci permet de définir une distance sur la variété. Définissons d'abord la longueur d'un chemin $\gamma : [\alpha, \beta] \to V$ de classe $\mathcal{C}^1$ sur une variété différentielle V :

$$L_\alpha^\beta(\gamma) = \int_\alpha^\beta \sqrt{\omega_{\gamma(t)}(\gamma'(t), \gamma'(t))}dt = \int_\alpha^\beta \|\gamma'(t)\|dt\,.$$

Si V est connexe par arcs (ce que l'on suppose dorénavant), tout couple de points peut être relié par un chemin $\mathcal{C}^1$, et la définition suivante est valide.

$$\forall u, v \in V\ ,\ \rho(u, v) = \inf\{L_\alpha^\beta(\gamma), \gamma(\alpha) = u, \gamma(\beta) = v\}\,.$$

Ainsi V devient un espace métrique ; cette propriété a été utilisée implicitement tout au long des trois premiers chapitres.

Dans le cas de $\mathcal{C}([a,b])$, on peut identifier $\gamma'(t)$ à f_γ comme au paragraphe A.1.3. Un produit scalaire possible (noté $\langle .,. \rangle$) est alors celui de la norme L_2 ; c'est en fait celui que nous utilisons systématiquement au cours de cette thèse :

$$\omega_{\gamma(t)}(\gamma'(t),\gamma'(t)) = \|f_\gamma\|^2 = \langle f_\gamma, f_\gamma \rangle = \int_a^b f_\gamma^2(t)dt\,.$$

On voit alors que la définition de la longueur d'un chemin est cohérente avec l'intuition, donnant cette distance comme la somme des "chemins élémentaires". En effet si l'on dispose de suffisamment d'échantillons de courbes $f_i = \gamma(t_i)$ sur le tracé du chemin γ, une telle distance peut être approximée par la somme suivante

$$L_\alpha^\beta(\gamma) \simeq \sum_{i=0}^{m} \|f_{i+1} - f_i\|$$

avec $t_0 = \alpha$, $t_{m+1} = \beta$. Cette approximation a été utilisée aux chapitres 1, 2 et 3, et est étudiée dans le rapport de Bernstein et al. [28].

Notons que la borne inférieure peut ne pas être atteinte ; en effet il y a une différence subtile entre la définition d'une géodésique que nous verrons au paragraphe A.2.2 et celle de la distance $\rho(u,v)$ entre deux points $u, v \in V$ (même si cette dernière est appelée "distance géodésique" par abus de langage). Nous avons cependant opté pour la notation désignant la distance (mesurée le long d'une) géodésique, car lorsque le plus court chemin est dans V il coïncide avec une géodésique.

A.2 Paramétrage "naturel" d'une variété riemannienne

Cette seconde section s'attache à définir la notion de géodésique sur une variété, permettant de décrire les plus courts chemins entre deux points u et v. De manière analogue au cas euclidien il semble naturel de chercher à annuler les "dérivées secondes". Ces dernières nécessitent la notion de connexion (linéaire). On pourra alors définir les coordonnées riemanniennes normales qui ont été utilisées au chapitre 3, avant d'évoquer la courbure, objet incontournable en apprentissage de variété.

A.2.1 Connexions

Nous pouvons définir les dérivées directionnelles en un point via l'identification réalisée au paragraphe A.1.3, car celles-ci nécessite le calcul de différences entre objets de l'espace vectoriel $\mathcal{C}([a,b])$. En revanche, les vecteurs tangents $\gamma'(t_0)$ et $\gamma'(t)$ ne vivant pas dans le même espace, on ne peut les soustraire pour écrire "$\gamma''(t_0) = \lim_{t\to 0} \frac{\gamma'(t_0+t)-\gamma'(t_0)}{t}$". Afin de contourner cette difficulté nous allons "connecter" les espaces tangents depuis $u \in V$ jusqu'à $v \in \mathcal{V}(u)$, via un outil assez logiquement appelé une *connexion*.

Un *fibré vectoriel* désignant un espace fibré dont la base a une structure d'espace vectoriel, on peut commencer par la définition générale suivante.

Définition A.2.1 *Soit (E,V) un fibré vectoriel sur une variété V. Notons $\mathcal{E}(V)$ (resp.*

$\mathcal{T}(V)$) l'espace des sections continues de (E, V) (resp. des champs de vecteurs sur V). Une connexion (générale) sur V est une application

$$\nabla : \mathcal{T}(V) \times \mathcal{E}(V) \to \mathcal{E}(V)\,,$$

notée $\nabla(X, Y) = \nabla_X Y$ et vérifiant les propriétés suivantes.

- *∇ est $\mathcal{C}^k(V)$-linéaire en X : pour $f, h \in \mathcal{C}^k(V)$, $\nabla_{fX_1+hX_2}Y = f\nabla_{X_1}Y + h\nabla_{X_2}Y$;*
- *∇ est $\mathbb{R}$-linaire en Y : pour $\alpha, \beta \in \mathbb{R}$, $\nabla_X(\alpha Y_1 + \beta Y_2) = \alpha\nabla_X Y_1 + \beta\nabla_X Y_2$;*
- *∇ satisfait la règle produit : $\nabla_X(fY) = f\nabla_X Y + (Xf)Y$ pour $f \in \mathcal{C}^k(V)$.*

$\nabla_X Y$ est appelée "dérivée (covariante) de Y dans la direction X".

Une *connexion linéaire* sur V est une application $\nabla : \mathcal{T}(V) \times \mathcal{T}(V) \to \mathcal{T}(V)$ vérifiant les conditions de la définition précédente. Cette dernière s'interprète comme la dérivée d'un champ de vecteur Y selon les directions du champ X, et permet ainsi la définition de la dérivée seconde le long d'une courbe tracée sur la variété.

Soit $\gamma : I \to V$ un chemin (au moins) $\mathcal{C}^1$. Notons alors $\mathcal{T}(\gamma)$ l'ensemble des champs de vecteurs sur $\mathrm{Im}(\gamma)$. Un élément de $\mathcal{T}(\gamma)$ est dit *extensible* lorsqu'il existe un champ de vecteurs $\tilde{X}$ défini sur un ouvert contenant $\mathrm{Im}(\gamma)$, coïncidant avec X sur sa restriction à $\mathrm{Im}(\gamma)$. La figure A.4 illustre deux champs de vecteurs le long d'un chemin, l'un n'étant pas extensible. Étant donnée une connexion linéaire ∇ sur V, il existe un unique opérateur $D : \mathcal{T}(\gamma) \to \mathcal{T}(\gamma)$ vérifiant les propriétés suivantes (Lee [206], lemme 4.9) :

- $\forall \alpha, \beta \in \mathbb{R}\, \forall X, Y \in \mathcal{T}(\gamma)\ D(\alpha X + \beta Y) = \alpha DX + \beta DY$ (linéarité) ;
- $\forall f \in \mathcal{C}^k(I)\, \forall X \in \mathcal{T}(\gamma)\ D(fX) = f'X + fDX$ (règle produit) ;
- $\forall X \in \mathcal{T}(\gamma)$ extensible, $DX(t) = \nabla_{\gamma'}\tilde{X}(t)$ avec $\tilde{X}$ extension de X (compatibilité),

$\gamma' \in \mathcal{T}(\gamma)$ désignant le champ de vecteurs formé par les dérivées le long de γ. Pour $X \in \mathcal{T}(\gamma)$, DX désigne ainsi le champ des vecteurs dérivée seconde le long du chemin γ, relativement à une connexion linéaire ∇.

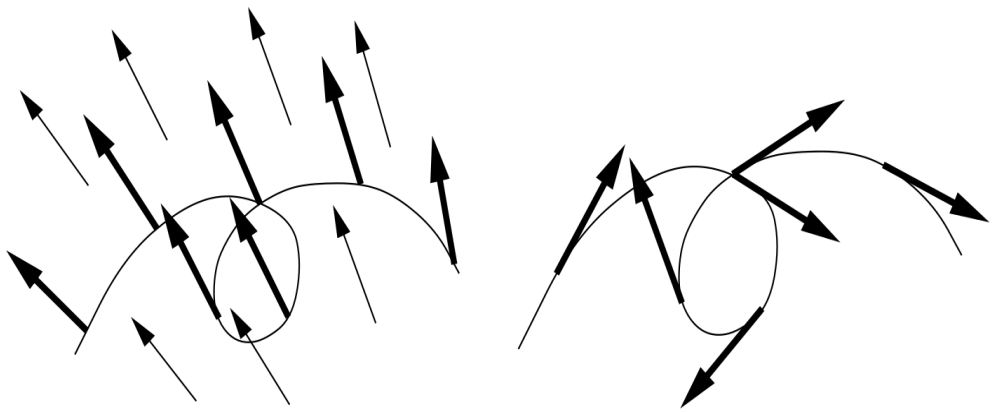

Figure A.4: Champs de vecteurs extensible à gauche, non extensible à droite.

A.2.2 Géodésiques

À l'aide de la notion de dérivée seconde le long d'une courbe, les géodésiques vont pouvoir être définies. Étant donné ∇ une connexion linéaire sur V et $\gamma : I \to V$ un chemin $\mathcal{C}^1$, la dérivée seconde de γ au point $t \in I$ s'assimile à $D\gamma'(t)$, c'est-à-dire la valeur en t de l'image du champ de vecteurs γ' par l'opérateur D.

Définition A.2.2 *Une* géodésique *sur la variété V est (l'image d') un chemin γ de classe $\mathcal{C}^1$ vérifiant $D\gamma' = 0$.*

Conformément à l'intuition (obtenue en considérant une surface de dimension 2 dans $\mathbb{R}^3$), il existe une unique géodésique passant par un point donné avec une vitesse fixée (Lee [206], théorème 4.10). Notons cependant que la définition d'une géodésique dépend de la connexion linéaire choisie. Dans le cadre des variétés riemanniennes, une définition universelle découle de l'unicité d'une connexion linéaire ∇ vérifiant deux propriétés additionnelles. Avant de les présenter, une définition supplémentaire est nécessaire.

Définition A.2.3 *Le crochet de Lie $[.,.]$ sur une variété différentielle V vérifie :*

$$\forall X, Y \in \mathcal{T}(V)\ \forall f \in \mathcal{C}^k(V)\ \ [X,Y].f = X(Y(f)) - Y(X(f))\,,$$

en considérant $Z(f)$ comme la fonction qui à $u \in V$ associe $Z_u(f)$ pour $Z \in \{X, Y\}$.

Nous pouvons alors introduire la *connexion de Levi-Civita*, définie comme étant l'unique connexion linéaire vérifiant en outre les deux critères suivants pour tous champs de vecteurs $X, Y, Z \in \mathcal{T}(V)$.

- $\nabla_X \langle Y, Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle$ (compatibilité avec la métrique).
- $\nabla_X Y - \nabla_Y X = [X, Y]$ (symétrie, absence de torsion).

Dans la première propriété ci-dessus, les produits scalaires entre deux champs de vecteurs s'interprètent naturellement comme des fonctions (lisses) de V dans $\mathbb{R}$. L'expression $\nabla_X \langle Y, Z\rangle$ désigne quant à elle la dérivée de la fonction $\langle Y, Z\rangle$ dans les directions vectorielles données par X. Ainsi, lorsque l'on parlait de géodésiques sur la variété $\mathcal{U}$, c'est toujours celles relatives à cette connexion particulière qui étaient sous-entendues. Ce sera également le cas dans toute la suite, la notation ∇ étant conservée.

Remarque : toute géodésique est une courbe localement minimisante. C'est à dire qu'à $u \in V$ et $v \in T_u V$ fixés, il existe un voisinage $\mathcal{V}$ de u tel que :

$$\forall \tilde{\alpha} = \gamma(\alpha), \tilde{\beta} = \gamma(\beta) \in \mathrm{Im}(\gamma) \cap \mathcal{V}\,,\ \ L_\alpha^\beta(\gamma_v) = \rho(\alpha, \beta)\,.$$

En revanche, il peut ne pas exister de géodésique vérifiant l'égalité ci-dessus pour α et β quelconques. De plus, un trajet le long d'une géodésique n'est pas nécessairement globalement minimisant (bien qu'il le soit localement). On peut penser par exemple aux méridiens sur un globe terrestre.

A.2.3 Coordonnées riemanniennes normales

L'introduction des géodésiques permet de définir une application locale de l'espace tangent T_uV dans un voisinage de u sur V. Cette application définit des coordonnées "canoniques" aux propriétés particulières, appelées *coordonnées riemanniennes normales*.

Pour $(u, v) \in TV$, notons γ_v l'unique géodésique passant par u avec la vélocité v (le point u étant sous-entendu). Notons également $\mathcal{F}$ l'ensemble des éléments (u, v) du fibré tangent TV tels que γ_v soit définie sur un intervalle contenant $[0, 1]$. L'*application exponentielle* se définit alors comme suit.

$$\begin{array}{rccc} \exp : & \mathcal{F} & \to & V \\ & v & \mapsto & \gamma_v(1)\,. \end{array}$$

exp envoie un vecteur v sur le point obtenu en suivant la géodésique issue de u à la vitesse v pendant un temps 1. Dans un espace euclidien, on peut visualiser cela comme une projection du plan tangent sur la variété (figure A.5, empruntée à wikimedia ici). Il est en général plus intéressant de fixer le point u et de considérer l'application exponentielle restreinte à l'espace tangent en u : $\exp_u : \mathcal{F} \cap T_uV \to \mathcal{V}(u)$, $\mathcal{V}(u)$ désignant un voisinage ouvert de u dans V.

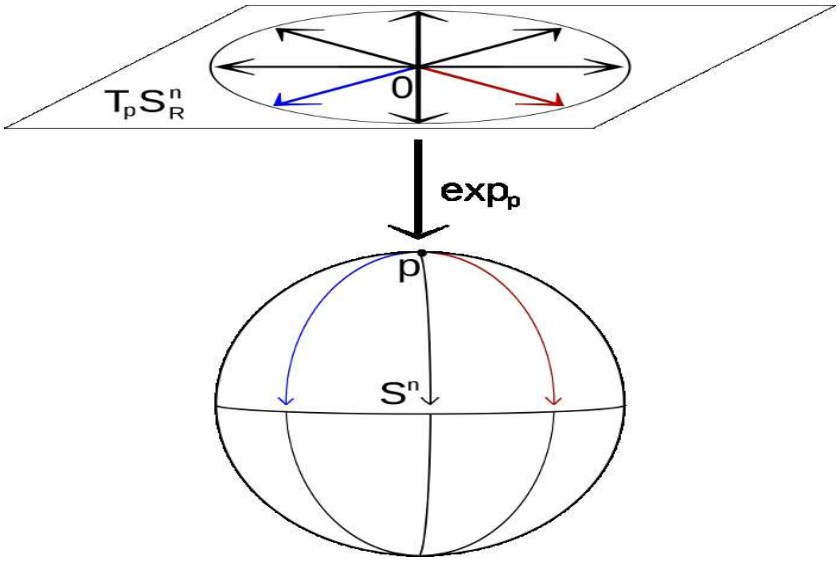

Figure A.5: Visualisation de l'application exponentielle dans $\mathbb{R}^3$.

La fonction exp (restreinte ou non) vérifie quelques propriétés intéressantes en principe indiquées dans tous les ouvrages de géométrie riemannienne. Nous nous concentrons ici sur la suivante (Lee [206], lemme 5.10) :

Proposition A.2.1 *L'application exponentielle exp_u réalise un difféomorphisme d'un voisinage de l'origine dans T_uV vers un voisinage de u dans V.*

Cette dernière propriété permet de définir des coordonnées locales naturelles. En effet, en notant $\{E_1, \ldots, E_d\}$ une base de l'espace tangent T_uV, nous disposons de l'isomorphisme suivant.

$$\begin{array}{rccc} i : & \mathbb{R}^d & \to & T_uV \\ & (s_1, \ldots, s_d) & \mapsto & \sum_{j=1}^d s_j E_j\,. \end{array}$$

Il suffit alors d'écrire la composition $i^{-1} \circ \exp^{-1}$ (restreinte aux bons espaces) pour obtenir une carte locale en u.

Définition A.2.4 *Les coordonnées riemanniennes normales sont définies localement en $u \in V$ par la composée $i^{-1} \circ exp^{-1}$ indiquée ci-dessus.*

En général on choisit la base canonique sur T_uV, c'est à dire les dérivations du type $\frac{\partial}{\partial x_i}$. Le lecteur est reporté la bibliographie pour une présentation complète.

Ces coordonnées sont approximées par les algorithmes présentés au paragraphe 3.2.3.2.

A.2.4 Courbure

La courbure d'une variété de dimension 1 s'exprime naturellement en chaque point comme l'inverse du rayon du cercle osculateur, comme la figure A.6 le montre.

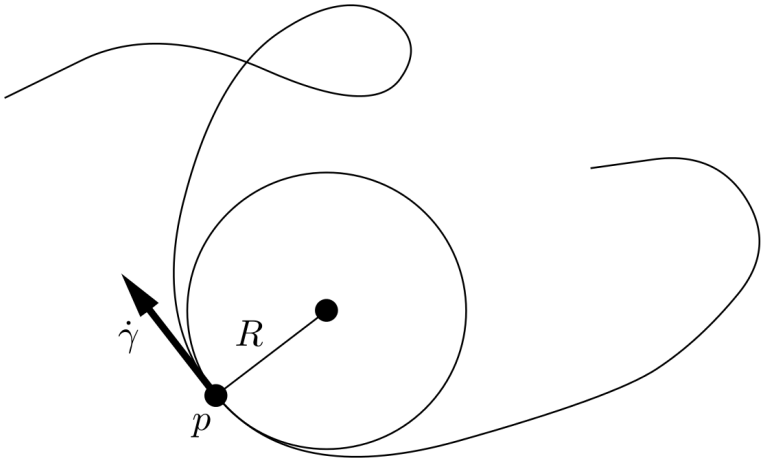

Figure A.6: Courbure en dimension 1 via le cercle osculateur.

En dimension 2, on peut se ramener au cas précédent en considérant l'espace tangent T_uV en un point u. Chaque géodésique issue de u avec une vitesse $v \in T_uV$ possède une courbure scalaire en u, car elle s'identifie à une variété de dimension 1. On calcule ensuite les valeurs minimale et maximale de ces courbures, respectivement notées κ_1 et κ_2. La courbure de Gauss (ou courbure scalaire) est alors obtenue en effectuant le produit de ces dernières quantités :

$$\kappa = \kappa_1.\kappa_2 \,.$$

Cette valeur définit complètement la courbure d'une variété de dimension 2 (Lee [206], lemme 8.7). La figure A.7 illustre des géodésiques légèrement incurvées issues du même point avec des vitesses dans le plan tangent.

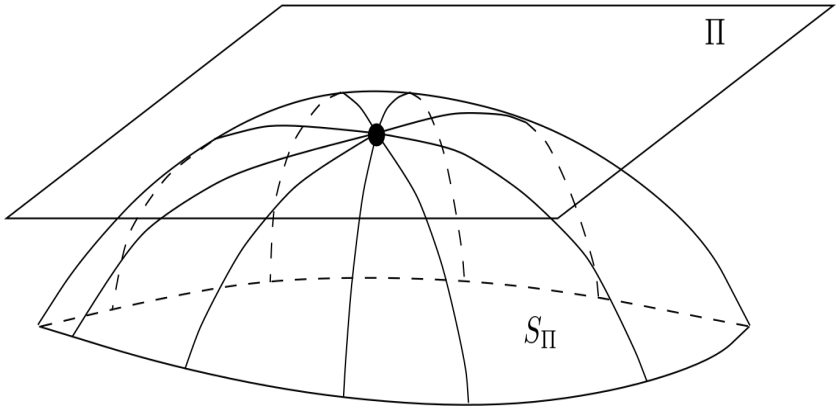

Figure A.7: Courbure de Gauss en dimension 2 via les géodésiques.

En dimension supérieure à deux, cependant, la notion de courbure s'avère plus délicate à décrire. ∇ désignant connexion de Levi-Civita sur la variété riemannienne V, on introduit l'opérateur ("tenseur") suivant :

$$\begin{array}{rccl} C: & \mathcal{T}(V)^3 & \to & \mathcal{T}(V) \\ & (X,Y,Z) & \mapsto & \nabla_X\nabla_Y Z - \nabla_Y\nabla_X Z - \nabla_{[X,Y]}Z\,. \end{array}$$

Ce dernier étant identiquement nul pour toute variété localement isométrique à l'espace euclidien $\mathbb{R}^d$, on choisit le critère "$C = 0$" pour tester si une variété est "plate" ou non. On peut alors montrer qu'il y a équivalence entre $C = 0$ et "être localement isométrique à $\mathbb{R}^d$" (Lee [206], théorème 7.3). Cet opérateur mesure donc à quel point une variété n'est pas localement isométrique à $\mathbb{R}^d$; il est très utilisé en théorie de la relativité, notamment pour exprimer la courbure de l'espace-temps. Si l'on fixe X et Y, on peut le considérer comme un endomorphisme de l'espace des champs de vecteurs. C est ainsi nommé *endomorphisme de courbure*.

Cet objet véhicule toute l'information pertinente sur la courbure d'une variété, mais reste particulièrement abstrait. C'est pourquoi on peut considérer sa trace (par rapport à X et Y), obtenant le *tenseur de courbure de Ricci*. Si l'on reprend la trace de ce dernier tenseur on obtient la courbure scalaire, fonction de V dans $\mathbb{R}$. Pour des démonstrations formelles, l'expression initiale de C est néanmoins utile.

Alternativement, il est possible d'étendre la définition du cas de la dimension 2 en considérant des sous-espaces vectoriels de T_uV. Plus exactement, pour $u \in V$ et Π un sous-espace vectoriel de dimension 2 dans T_uV, l'application exponentielle restreinte à (un voisinage de l'origine dans) $T_uV \cap \Pi$ définit une sous-variété de V (de dimension 2). On peut alors calculer sa courbure de Gauss comme ci-dessus, à l'aide de la métrique induite. L'ensemble de toutes ces courbures pour u décrivant la variété définit entièrement l'endomorphisme de courbure (Lee [206], lemme 8.9).

Annexe B

k-means fonctionnel projeté

Ce premier article porte sur la convergence des centres lors du k-means fonctionnel effectué sur les coefficients de projection obtenues via une base orthonormale. Il a été rédigé avec Aurélie Fischer (UPMC Paris 6), et est accepté pour publication dans le Journal of Statistical Computation and Simulation.

Résumé de l'article

Ce papier aborde la classification de courbes non supervisée dans le contexte de l'industrie nucléaire. Au Commissariat à l'Energie Atomique (CEA) de Cadarache, le code de calcul thermohydraulique CATHARE est utilisé afin d'évaluer la fiabilité de la cuve du réacteur. Les entrées du code sont des vecteurs de paramètres physiques, et les sorties des courbes d'évolution de certaines quantités comme la température. Le code CATHARE étant particulièrement complexe et coûteux en temps CPU, on choisit de l'approximer par un méta-modèle. La construction de ce dernier comporte une étape de clustering, qui est l'objet de cet article. Cette étape est effectuée ici via l'algorithme des k-means sur les coefficients de projection des courbes dans un espace de dimension réduite. Nous étudions les propriétés des centroïdes empiriques optimaux, en comparaison avec les "vrais" centres dans l'espace de fonctions. Plusieurs bases de fonctions sont envisagées, et un algorithme est implémenté pour sélectionner la meilleure par rapport à un critère de distorsion (empirique). L'approche est illustrée sur un exemple synthétique avant de l'appliquer aux données industrielles.

Mots-clé : clustering, k-means, base de fonctions, projection, paquet d'ondelettes.

Projection-based curve clustering

Benjamin AUDER
DEN/DER/SESI/LCFR
CEA Cadarache
13108 SAINT PAUL LEZ DURANCE Cedex, FRANCE

Aurélie FISCHER*
Laboratoire de Statistique Théorique et Appliquée
Université Pierre et Marie Curie – Paris VI
Boîte 158, couloir 15–16, 2^e étage
4 place Jussieu, 75252 PARIS Cedex 05, FRANCE

Abstract – This paper focuses on unsupervised curve classification in the context of nuclear industry. At the Commissariat à l'Energie Atomique (CEA), Cadarache (France), the thermal-hydraulic computer code CATHARE is used to study the reliability of reactor vessels. The code inputs are physical parameters and the outputs are time evolution curves of a few other physical quantities. As the CATHARE code is quite complex and CPU-time consuming, it has to be approximated by a regression model. This regression process involves a clustering step. In the present paper, CATHARE output curves are clustered using a k-means scheme, with a projection onto a lower dimensional space. We study the properties of the empirically optimal cluster centers found by the clustering method based on projections, compared to the "true" ones. The choice of the projection basis is discussed, and an algorithm is implemented to select the best projection basis among a library of orthonormal bases. The approach is illustrated on a simulated example and then applied to the industrial problem.

Index Terms — Clustering, basis expansion, projections, k-means, wavelet packets.

*Corresponding author. E-mail: aurelie.fischer@upmc.fr

B.1 Introduction

B.1.1 The CATHARE code

A major concern in nuclear industry is the life span of reactor vessels. To go on using the current nuclear reactors, their reliability has to be proved. For this purpose, complex computer codes are developed to simulate the behavior of the vessel under different sequences of accidents. At the Commissariat à l'Energie Atomique (CEA), Cadarache (France), one of the main types of accident under study is the pressurized thermal shock. This is a problem due to the combined stresses from a rapid temperature and pressure change. More specifically, as a reactor vessel gets older, the potential for failure by cracking when it is cooled rapidly at high pressure increases greatly. The analysis of pressurized thermal shock is made of two main steps. First, a thermal-hydraulic analysis is done to determine the temporal evolutions of temperature, pressure and thermal exchange coefficient in the vessel annular space, since these features have an influence on the mechanical and thermal charge on the vessel inner surface. Some evolution curves $x_1(t), \ldots, x_n(t)$ corresponding to the thermal exchange coefficient are depicted in Figure B.1 ($n = 66$). Each curve $x_i(t)$ is obtained as the simulation result for a certain vector of input physical parameters. The curves of temperature, pressure and thermal exchange coefficient obtained during this first step are then used as limit conditions in the second part of the analysis, which is a mechanical investigation aiming at checking if some defects on the vessel annular space could propagate and gain importance to such an extent that this would cause a break of the vessel inner surface. For further details on the reliability of reactor vessels, we refer the reader to Auder, De Crecy, Iooss, and Marquès [5].

The simulation step relies on a computer code called CATHARE (Code Avancé de THermohydraulique pour les Accidents des Réacteurs à Eau, in English Code for Analysis of THermalhylaudrics during an Accident of Reactor and safety Evaluation). The CATHARE code is a system code for pressurized water reactors safety analysis, accident management, definition of plant operating procedures and for research and development. The project is a result of a joint effort of the reactor vendor AREVA, the CEA, EDF (Electricité de France) and the IRSN (Institut de Radioprotection et de Sûreté Nucléaire). The first delivered version V1.3L was available in 1997. The CEA team CATHARE located in Grenoble (France) is in charge of the development, the assessment and the maintenance of the code. (See http://www-cathare.cea.fr.)

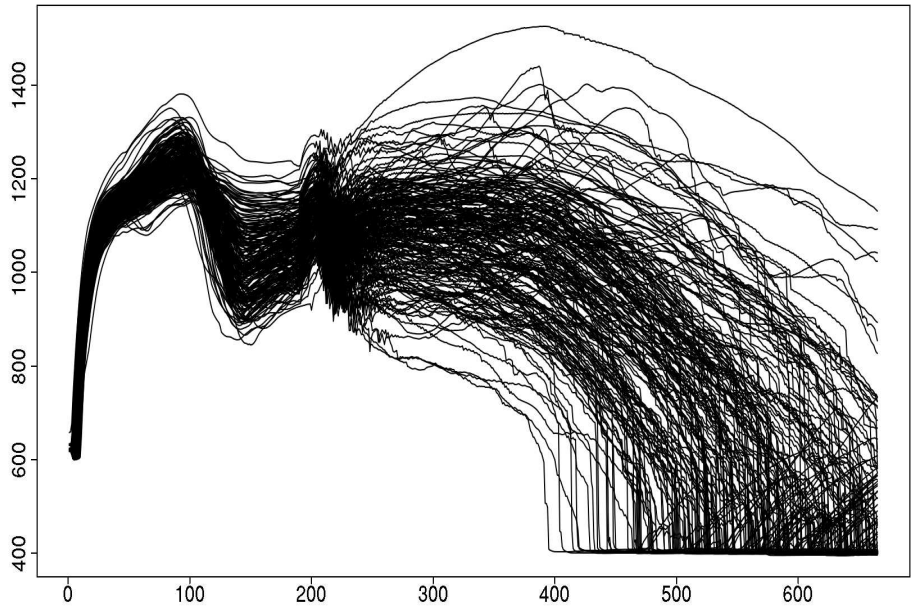

Figure B.1: 66 evolution curves of thermal exchange coefficient in a nuclear vessel.

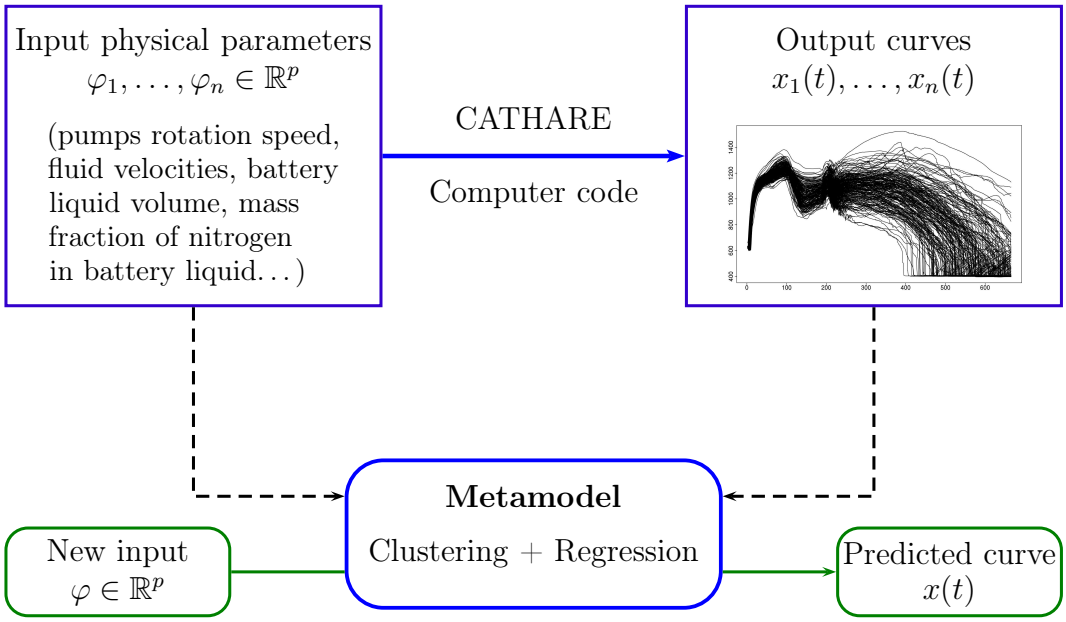

Figure B.2: Flowchart of the CATHARE code.

The CATHARE code allows to simulate the evolution of temperature, pressure and thermal exchange coefficient, given the physical parameters as inputs. However, this code is so slow (about 6 to 10 hours for one run) that it cannot be used directly for reliability calculations. To bypass this obstacle, the strategy drawn up by the CEA is to build a so-called metamodel which is a fast approximation of the original code, precise enough to carry out statistical computations. The term "metamodel" indicates that a computer code approximating a physical process has already been developed, and now this code is modeled in turn. Here, the purpose is the construction of a regression model based on a few hundreds CATHARE code outputs, obtained during one week of computation on a supercomputer in 2007. The inputs were sampled randomly by latin hypercube methods (see, e.g., McKay,

Conover, and Beckman [28] and Loh [26]), so that we have no control over the inputs in the learning sample. As different kinds of behavior for temperature, pressure and thermal exchange coefficient may be observed depending on the physical parameters, a preliminary unsupervised classification of CATHARE code output curves is essential. Once the curves have been clustered in meaningful classes, the regression model can be adjusted for each group of outputs separately. The clustering step is the object of the present paper.

B.1.2 Clustering

Clustering is the problem of partitioning data into a finite number of groups (denoted hereafter by k), or clusters, so that the data items inside each of them are very similar among themselves and as different as possible from the elements of the other clusters (Duda, Hart, and Stork [14, Chapter 10]). In our industrial context, the data is made of evolution curves of temperature, pressure or thermal exchange coefficient. Using a probabilistic point of view, these curves can be seen as independent draws $X_1(t), \ldots, X_n(t)$ with the same distribution as a generic random variable $X(t)$ taking values in a functional space $(E, \|\cdot\|)$ — typically, the Hilbert space of square integrable functions.

A widely used clustering method is the so-called k-means clustering, which consists in partitioning the random observations $X_1, \ldots, X_n \in E$ into k classes by minimizing the empirical distortion

$$W_{\infty,n}(\mathbf{c}) = \frac{1}{n} \sum_{i=1}^{n} \min_{\ell=1,\ldots,k} \|X_i - c_\ell\|^2,$$

over all possible cluster centers $\mathbf{c} = (c_1, \ldots, c_k) \in E^k$. Here, μ_n denotes the empirical measure associated with the sample $X_1, \ldots, X_n$, i.e.,

$$\mu_n(A) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{X_i \in A\}}$$

for every Borel subset A of E. In other words, we look for a Voronoi partition of E. The Voronoi partition $C_1, \ldots, C_k$ associated with $\mathbf{c} = (c_1, \ldots, c_k) \in E^k$ is defined by letting an element $x \in E$ belong to C_ℓ if it is closer (with respect to the norm $\|\cdot\|$) to c_ℓ than to any other c_j (ties are broken arbitrarily). The ℓ-th cluster is made of the observations X_i assigned to c_ℓ, or equivalently, falling in the Voronoi cell C_ℓ. In this framework, the accuracy of the clustering scheme is assessed by the distortion or mean squared error

$$W_\infty(\mathbf{c}) = \mathbb{E}\Big[\min_{\ell=1,\ldots,k} \|X - c_\ell\|^2 \Big],$$

where $\mathbb{E}$ stands for expectation with respect to the distribution of X. This clustering method is in line with the more general theory of quantization. More specifically, it corresponds to the empirical version of nearest neighbor quantization (Linder [24], Gersho and Gray [18], Graf and Luschgy [21]). However, the problem of finding a minimizer of the criterion $W_{\infty,n}(\mathbf{c})$ is in general NP-hard, and there is no efficient algorithm to find the optimal solution in reasonable time. That is why several iterative algorithms have been developed to give approximate solutions. One of the first to be described historically is Lloyd's algorithm (Lloyd [25]).

The challenge is to adapt the k-means clustering method to our setting. The main difficulty here is the high dimensionality of the data, which casts the problem into the general class of functional statistics. For a comprehensive introduction to this topic, see Ramsay and Silverman [32] and Ferraty and Vieu [15]. A possible approach to reduce the infinite dimension of the observations $X_1, \ldots, X_n$ consists in projecting them onto a lower-dimensional subspace. In this context, Abraham, Cornillon, Matzner-Løber, and Molinari [1] project the curves on a B-spline basis, and get clusters with the k-means algorithm applied to the coefficients. These authors argue that projecting onto a smooth spline basis plays the role of a denoising procedure, given that the observed curves could contain measurement errors, and also allows to deal with curves which were not measured at the same time. James and Sugar [23] use a B-spline basis to model the centers of the clusters and write each curve in cluster ℓ as a main effect defined by spline coefficients plus an error term. This allows for some deviations around a model curve specific to cluster ℓ. The authors add a Gaussian error term to model the individual variations among one cluster. This way, the main effect is enriched, and the model can take into account more complex behaviors. The method in Gaffney [17] is similar, also with a B-spline basis. Another option is to use a Self-Organizing Map algorithm on the coefficients (Rossi, Conan-Guez, and El Golli [34]), again obtained by projecting the functions onto a B-spline basis. These bases are often used because they are easy to implement, and require a relatively minimal number of parametric assumptions. Besides, Biau, Devroye, and Lugosi [6] examine the theoretical performance of clustering with random projections based on the Johnson-Lindenstrauss Lemma, which represent a sound alternative to orthonormal projections thanks to their distance-preserving properties. Chiou and Li [7] propose a method which generalizes the k-means algorithm to some extent, by considering covariance structures via functional principal component analysis. In the approach of these authors, each curve is decomposed on an adaptive local basis (valid for the elements in the cluster), and the clusters are determined according to the full approximation onto each basis. In the wavelet-based method for functional data clustering developed in Antoniadis, Brossat, Cugliari, and Poggi [3], a smooth curve is reduced to a finite number of representative features, by considering the contribution of each wavelet coefficient to the global energy of the curve.

In the present contribution, we propose to investigate the problem of clustering output curves $X_1, \ldots, X_n$ of the CATHARE code, assuming that they arise from a random variable X taking its values in some subset of the space of square integrable functions. As a general strategy, we reduce the infinite dimension of X by considering only the first d coefficients of the expansion on a Hilbertian basis, and then perform clustering in $\mathbb{R}^d$. We study the theoretical properties of this clustering method with projection. A bound expressing what is lost when replacing the empirically optimal cluster centers by the centers obtained by projection is offered (Section 2). Since the result may depend on the basis choice, several projection bases are used in practice, and we look for the best one minimizing a given criterion. To this end, an algorithm based on Coifman and Wickerhauser [9] is implemented, searching for an optimal basis among a library of wavelet packet bases available in the **R** package *wmtsa*, and this "optimal basis" is compared with the Fourier basis, the Haar basis, and the functional principal component basis (Section 3). Finally, this algorithm is applied to a simulated example and to our industrial problem (Section 4). Proofs are postponed to Section 5.

B.2 Finite-dimensional projection for clustering

As mentioned earlier, we are concerned with square integrable functions. Since all results can be adapted to $L^2([a,b])$ by an appropriate rescaling, we consider for the sake of simplicity the space $L^2([0,1])$. As an infinite-dimensional separable Hilbert space, $L^2([0,1])$ is isomorphic via the choice of a Hilbertian basis to the space ℓ^2 of square-summable sequences. We focus more particularly on functions in $L^2([0,1])$ whose coefficients in the expansion on a given Hilbertian basis belong to the subset $\mathcal{S}$ of ℓ^2 given by

$$\mathcal{S} = \Big\{ \mathbf{x} = (x_j)_{j\geq 1} \in \ell^2 : \sum_{j=1}^{+\infty} \varphi_j x_j^2 \leq R^2 \Big\}, \tag{B.1}$$

where $R > 0$ and $(\varphi_j)_{j\geq 1}$ is a nonnegative increasing sequence such that

$$\lim_{j\to+\infty} \varphi_j = +\infty.$$

It is worth pointing out that $\mathcal{S}$ is closely linked with the basis choice, even if the basis does not appear explicitly in the definition. To illustrate this important fact, three examples are discussed below.

Example B.2.1 (Sobolev ellipsoids) *For $\beta \in \mathbb{N}^*$ and $L > 0$, the periodic Sobolev class $W^{per}(\beta, L)$ is the space of all functions $f \in [0,1] \to \mathbb{R}$ such that $f^{(\beta-1)}$ is absolutely continuous, $\int_0^1 (f^{(\beta)}(t))^2 dt \leq L^2$ and $f^{(\ell)}(0) = f^{(\ell)}(1)$ for $\ell = 0, \ldots, \beta-1$. Let $(\psi_j)_{j\geq 1}$ denote the trigonometric basis. Then a function $f = \sum_{j=1}^{+\infty} x_j \psi_j$ is in $W^{per}(\beta, L)$ if and only if the sequence $\mathbf{x} = (x_j)_{j\geq 1}$ of its Fourier coefficients belongs to*

$$\mathcal{S} = \Big\{ \mathbf{x} \in \ell^2 : \sum_{j=1}^{+\infty} \varphi_j x_j^2 \leq R^2 \Big\},$$

where

$$\varphi_j = \begin{cases} j^{2\beta} & \text{for even } j \\ (j-1)^{2\beta} & \text{for odd } j \end{cases}$$

and $R = \dfrac{L}{\pi^\beta}$.

For the proof of this result and further details about Sobolev classes, we refer the reader to the book of Tsybakov [36]. Note that the set $\mathcal{S}$ could also be defined by $\varphi_j = j^r e^{\alpha j}$ with $\alpha > 0$ and $r \geq -\alpha$ (Tsybakov [35]).

Example B.2.2 (Reproducing Kernel Hilbert Spaces) *Let $K : [0,1] \times [0,1] \to \mathbb{R}$ be a Mercer kernel, i.e., K is continuous, symmetric and positive definite. Recall that a kernel K is said to be positive definite if for all finite sets $\{x_1, \ldots, x_m\}$, the matrix A defined by $a_{ij} = K(x_i, x_j)$ for $1 \leq i, j \leq m$ is positive definite. For example, the Gaussian kernel $K(x,y) = \exp(-\frac{(x-y)^2}{\sigma^2})$ and the kernel $K(x,y) = (c^2 + (x-y)^2)^{-a}$ with $a > 0$ are Mercer kernels. For $x \in [0,1]$, let $K_x : y \mapsto K(x,y)$. Then, Moore-Aronszajn's Theorem (Aronszajn [4]) states that there exists a unique Hilbert space $(\mathcal{H}_K, \langle\cdot,\cdot\rangle)$ of functions on $[0,1]$ such that:*

1. *For all $x \in [0,1]$, $K_x \in \mathcal{H}_K$.*

2. *The span of the set* $\{K_x, x \in [0,1]\}$ *is dense in* $\mathcal{H}_K$.

3. *For all* $f \in \mathcal{H}_K$ *and* $x \in [0,1]$, $f(x) = \langle K_x, f \rangle$.

The Hilbert space $\mathcal{H}_K$ *is said to be the reproducing kernel Hilbert space (for short, RKHS) associated with the kernel* K. *Next, the operator* $\mathcal{K}$ *defined by*

$$\mathcal{K}f : y \mapsto \int_0^1 K(x,y)f(x)dx$$

is self-adjoint, positive and compact. Consequently, there exists a complete orthonormal system $(\psi_j)_{j\geq 1}$ *of* $L^2([0,1])$ *such that* $\mathcal{K}\psi_j = \lambda_j\psi_j$, *where the set of eigenvalues* $\{\lambda_j, j \geq 1\}$ *is either finite or a sequence tending to* 0 *at infinity. Moreover, the* λ_j *are nonnegative. Suppose that* $\mathcal{K}$ *is not of finite rank — so that* $\{\lambda_j, j \geq 1\}$ *is infinite — and that the eigenvalues are sorted in decreasing order, that is* $\lambda_j \geq \lambda_j + 1$ *for all* $j \geq 1$. *Clearly, there is no loss of generality in assuming that* $\lambda_j > 0$ *for all* $j \geq 1$. *Indeed, if not,* $L^2([0,1])$ *is replaced by the linear subspace spanned by the eigenvectors corresponding to non-zero eigenvalues.*

According to Mercer's theorem, K *has the representation*

$$K(x,y) = \sum_{j=1}^{+\infty} \lambda_j\psi_j(x)\psi_j(y),$$

where the convergence is absolute and uniform (Cucker and Smale [10, Chapter III, Theorem 1]). Moreover, $\mathcal{H}_K$ *may be characterized through the eigenvalues of the operator* $\mathcal{K}$ *by*

$$\mathcal{H}_\mathcal{K} = \Big\{ f \in L^2([0,1]) : f = \sum_{j=1}^{+\infty} x_j\psi_j, \sum_{j=1}^{+\infty} \frac{x_j^2}{\lambda_j} < \infty \Big\},$$

with the inner product

$$\Big\langle \sum_{j=1}^{+\infty} x_j\psi_j, \sum_{j=1}^{+\infty} y_j\psi_j \Big\rangle = \sum_{j=1}^{+\infty} \frac{x_j y_j}{\lambda_j}$$

(Cucker and Smale [10, Chapter III, Theorem 4]). Then, letting

$$\mathcal{S} = \Big\{ \mathbf{x} \in \ell^2, \sum_{j=1}^{+\infty} \frac{x_j^2}{\lambda_j} \leq R^2 \Big\},$$

the set $\mathcal{S}$ *is of the desired form (B.1), with* $\varphi_j = 1/\lambda_j$.

Example B.2.3 (Besov ellipsoids and wavelets) *Let* $\alpha > 0$. *For* $f \in L^2([0,1])$, *the Besov semi-norm* $|f|_{B_2^\alpha}(L^2)$ *is defined by*

$$|f|_{B_2^\alpha}(L^2) = \Big(\sum_{j=0}^{+\infty} [2^{j\alpha}\omega_r(f, 2^{-j}, [0,1])_2]^2 \Big)^{1/2}$$

where $\omega_r(f,t,[0,1])_2$ *denotes the modulus of smoothness of* f, *as defined for instance in DeVore and Lorentz [12], and* $r = \lfloor \alpha \rfloor + 1$. *Let* $\Lambda(j)$ *be an index set at resolution level* j *and* $(x_{j,\ell})_{j\geq 0, \ell \in \Lambda(j)}$ *the coefficients of the expansion of* f *in a suitable wavelet basis. Then, for* f

such that $|f|_{B_2^\alpha}(L^2) \leq \rho$, *the coefficients* $x_{j,\ell}$ *satisfy*

$$\sum_{j=0}^{+\infty} \sum_{\ell \in \Lambda(j)} 2^{2j\alpha} x_{j,\ell}^2 \leq \rho^2 C^2,$$

where $C > 0$ *depends only on the basis. We refer to Donoho and Johnstone [13] for more details.*

Let us now come back to the general setting

$$\mathcal{S} = \Big\{ \mathbf{x} = (x_j)_{j \geq 1} \in \ell^2 : \sum_{j=1}^{+\infty} \varphi_j x_j^2 \leq R^2 \Big\},$$

and consider the problem of clustering the sample $X_1, \ldots, X_n$ with values in $\mathcal{S}$. Some notation and assumptions are in order. First, we will suppose that $\mathbb{P}\{\|X\| \leq R\} = 1$. Notice that the fact that X takes its values in $\mathcal{S}$ is in general not enough to imply $\mathbb{P}\{\|X\| \leq R\} = 1$. Secondly, let j_0 be the smallest integer j such that $\varphi_j > 0$. To avoid technical difficulties, we require in the sequel $d \geq j_0$. For all $d \geq 1$, we will denote by Π_d the orthogonal projection on $\mathbb{R}^d$ and let $\mathcal{S}_d = \Pi_d(\mathcal{S})$. Lastly, observe that $\mathcal{S}_d$ identifies with the ellipsoid

$$\Big\{ \mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d : \sum_{j=1}^{d} \varphi_j x_j^2 \leq R^2 \Big\}.$$

As explained in the introduction, the criterion to minimize is

$$W_{\infty,n}(\mathbf{c}) = \frac{1}{n} \sum_{i=1}^{n} \min_{\ell=1,\ldots,k} \|X_i - c_\ell\|^2,$$

and the performance of the clustering obtained with the centers $\mathbf{c} = (c_1, \ldots, c_k) \in \mathcal{S}^k$ is measured by the distortion

$$W_\infty(\mathbf{c}) = \mathbb{E}\Big[\min_{\ell=1,\ldots,k} \|X - c_\ell\|^2 \Big].$$

The quantity

$$W_\infty^* = \inf_{\mathbf{c} \in \mathcal{S}^k} W_\infty(\mathbf{c})$$

represents the optimal risk we can achieve. With the intention of performing clustering in the projection space $\mathcal{S}^d$, we also introduce the "finite-dimensional" distortion

$$W_d(\mathbf{c}) = \mathbb{E}\Big[\min_{\ell=1,\ldots,k} \|\Pi_d(X) - \Pi_d(c_\ell)\|^2 \Big]$$

and its empirical counterpart

$$W_{d,n}(\mathbf{c}) = \frac{1}{n} \sum_{i=1}^{n} \min_{\ell=1,\ldots,k} \|\Pi_d(X_i) - \Pi_d(c_\ell)\|^2,$$

as well as

$$W_d^* = \inf_{\mathbf{c}\in\mathcal{S}^k} W_d(\mathbf{c}).$$

Let us observe that, as the support of the empirical measure μ_n contains at most n points, there exists an element $\hat{\mathbf{c}}_{d,n}$ which is a minimizer of $W_{d,n}(\mathbf{c})$ on $\mathcal{S}^k$. Moreover, in view of its definition, $W_{d,n}(\mathbf{c})$ only depends on the centers projection $\Pi_d(\mathbf{c})$ (one has $W_{d,n}(\mathbf{c}) = W_{d,n}(\Pi_d(\mathbf{c}))$ for all $\mathbf{c}$) and we can thus assume that $\hat{\mathbf{c}}_{d,n} \in (\mathcal{S}_d)^k$. Notice also that for all $c \in \mathcal{S}$,

$$\|\Pi_d(X) - \Pi_d(c)\|^2 \leq \|X - c\|^2$$

(the projection Π_d is 1-Lipschitz), which implies that

$$W_d(\mathbf{c}) \leq W_\infty(\mathbf{c})$$

for all $\mathbf{c}$.

The following lemma provides an upper bound for the maximal deviation

$$\sup_{\mathbf{c}\in\mathcal{S}^k} [W_\infty(\mathbf{c}) - W_d(\mathbf{c})].$$

Lemma B.2.1 *We have*

$$\sup_{\mathbf{c}\in\mathcal{S}^k} [W_\infty(\mathbf{c}) - W_d(\mathbf{c})] \leq \frac{4R^2}{\varphi_d}.$$

We are now in a position to state the main result of this section.

Theorem B.2.1 *Let $\hat{\mathbf{c}}_{d,n} \in (\mathcal{S}_d)^k$ be a minimizer of $W_{d,n}(\mathbf{c})$. Then,*

$$\mathbb{E}[W_\infty(\hat{\mathbf{c}}_{d,n})] - W_\infty^* \leq \mathbb{E}[W_d(\hat{\mathbf{c}}_{d,n})] - W_d^* + \frac{8R^2}{\varphi_d}. \tag{B.2}$$

Theorem B.2.1 expresses the fact that the expected excess clustering risk in the infinite dimensional space is bounded by the corresponding "finite-dimensional risk" plus an additional term representing the price to pay when projecting onto $\mathcal{S}_d$. Yet, the first term in the right-hand side of inequality (B.2) above is known to tend to 0 when n goes to infinity. More precisely, as $\mathbb{P}\{\|\Pi_d(X)\| \leq R\} = 1$, we have

$$\mathbb{E}[W_d(\hat{\mathbf{c}}_{d,n})] - W_d^* \leq \frac{Ck}{\sqrt{n}},$$

where $C = 12R^2$ (Biau, Devroye, and Lugosi [6]). In our setting, to keep the same rate of convergence $O(1/\sqrt{n})$ in spite of the extra term $8R^2/\varphi_d$, φ_d must be of the order $\sqrt{n}$. For Sobolev ellipsoids (Example B.2.1), where $\varphi_j \geq (j-1)^{2\beta}$, this means a dimension d of the order $n^{1/4\beta}$. When $\varphi_j = j^r e^{\alpha j}$, the rate of convergence is $O(1/\sqrt{n})$ as long as d is chosen of the order $\ln n/(2\alpha)$. In the RKHS context (Example B.2.2), consider the case of eigenvalues $\{\lambda_j, j \geq 1\}$ with polynomial or exponential-polynomial decay, which covers a broad range of kernels (Williamson, Smola, and Schölkopf [38]). If $\lambda_j = O(j^{-(\alpha+1)}), \alpha > 0$, then $1/\varphi_d = O(d^{-(\alpha+1)})$, and d must be of the order $n^{1/(2\alpha+2)}$, whereas $\lambda_j = O(e^{-\alpha j^p}), \alpha, p > 0$, leads to a projection dimension d of the order $(\ln n/(2\alpha))^{1/p}$. Obviously, the upper bound (B.2) is

better for large φ_d, and consequently large d. Nevertheless, from a computational point of view, the projection dimension should not be chosen too large.

Remark B.2.1 *Throughout, we assumed that* $\mathbb{P}\{\|X\| \leq R\} = 1$. *This requirement, called the peak power constraint, is standard in the clustering and signal processing literature. We do not consider in this paper the case where this assumption is not satisfied, which is feasible but leads to technical complications (see Merhav and Ziv [29], Biau, Devroye, and Lugosi [6] for results in this direction). Besides, the number of clusters is assumed to be fixed throughout the paper. Several methods for estimating k have been proposed in the literature (see, e.g., Milligan and Cooper [31] and Gordon [20]).*

As already mentioned, the subset of coefficients $\mathcal{S}$ is intimately connected to the underlining Hilbertian basis. As a consequence, all the results presented strongly depend on the orthonormal system considered. Therefore, the choice of a proper basis is crucial and is discussed in the next section.

B.3 Basis selection

Wavelet packet best basis algorithm In this section, we describe an algorithm searching for the best projection basis among a "library". If $\{\psi_\alpha, \alpha \in I\} \subset L^2([0,1])$ is a collection of elements in $L^2([0,1])$ which span $L^2([0,1])$ and allow to build several different bases by choosing various subsets $\{\psi_\alpha, \alpha \in I_\beta\} \subset L^2([0,1])$, the collection of bases built this way is called a library of bases. Here, I is some index set, and β runs over some other index set.

More specifically, we focus on the best basis algorithm of Coifman and Wickerhauser [9] (see also Wickerhauser [37]), which yields an optimal basis among a library of wavelet packets. Wavelets are functions which cut up a signal into different frequency components to study each component with a resolution matched to its scale. Unlike the Fourier basis, wavelets are localized both in time and frequency. Hence, they have advantages over traditional Fourier methods when the signal contains discontinuities as well as noise. For detailed expositions of the mathematical aspects of wavelets, see the books of Daubechies [11], Mallat [27] and Meyer [30].

Let the sequence of functions $(\psi_\nu)_{\nu \geq 0}$ be defined by

$$\begin{aligned}
\psi_0(t) &= H\psi_0(t), \quad \int_{\mathbb{R}} \psi_0(t)dt = 1, \\
\psi_{2\nu}(t) &= H\psi_\nu(t) = \sqrt{2}\sum_{p\in\mathbb{Z}} h(p)\psi_\nu(2t-p), \\
\psi_{2\nu+1}(t) &= G\psi_\nu(t) = \sqrt{2}\sum_{p\in\mathbb{Z}} g(p)\psi_\nu(2t-p),
\end{aligned}$$

where H and G are orthogonal quadrature filters, i.e., convolution-decimation operators satisfying some algebraic properties (see, e.g., Wickerhauser [37]). Let Λ_ν denote the closed linear span of the translates $\psi_\nu(\cdot - p), p \in \mathbb{Z}$, of ψ_ν, and

$$\sigma^s \Lambda_\nu = \{2^{-s/2} x(2^{-s}t), x \in \Lambda_\nu\}.$$

To every such subspace of $L^2(\mathbb{R})$ corresponds a dyadic interval

$$I_{s\nu} = \left[\frac{\nu}{2^s}, \frac{\nu+1}{2^s}\right[.$$

For all (s, ν), these subspaces give an orthogonal decomposition

$$\sigma^s \Lambda_\nu = \sigma_{s+1}\Lambda_{2\nu} \oplus \sigma_{s+1}\Lambda_{2\nu+1}.$$

Observe that for $\nu = 0, \ldots, 2^s - 1$, the $I_{s\nu}$ are dyadic subintervals of $[0, 1[$.

The next proposition provides a library of orthonormal bases built with functions of the form $\psi_{s\nu p} = 2^{-s/2}\psi_\nu(2^{-s}t - p)$, called wavelet packets of scale index s, frequency index ν and position index p.

Proposition B.3.1 (Wickerhauser [37]) *If $s \leq L$ for some finite maximum L, H and G are orthogonal quadrature filters and $\mathcal{I}$ is a collection of disjoint dyadic intervals whose union is $\mathbb{R}^+$, then $\mathcal{B}_\mathcal{I} = \{\psi_{s\nu p}, p \in \mathbb{Z}, I_{s\nu} \in \mathcal{I}\}$ is an orthonormal basis for $L^2(\mathbb{R})$. Moreover, if $\mathcal{I}$ is a disjoint dyadic cover of $[0, 1[$, then $\mathcal{B}_\mathcal{I}$ is an orthonormal basis of Λ_0.*

This construction yields orthonormal bases of $L^2(\mathbb{R})$. Some changes must be made to obtain bases of $L^2([0, 1])$. Roughly, they consist in considering not all scales and shifts, and adapting the wavelets which overlap the boundary of $[0, 1]$ (see for instance Cohen, Daubechies, and Vial [8]).

The library can be seen as a binary tree whose nodes are the spaces $\sigma^s\Lambda_\nu$ (Figure B.3 and B.4). An orthonormal basis is given by the leaves of some subtree. Figure B.5 and B.6 show two examples of bases which can be obtained in this way.

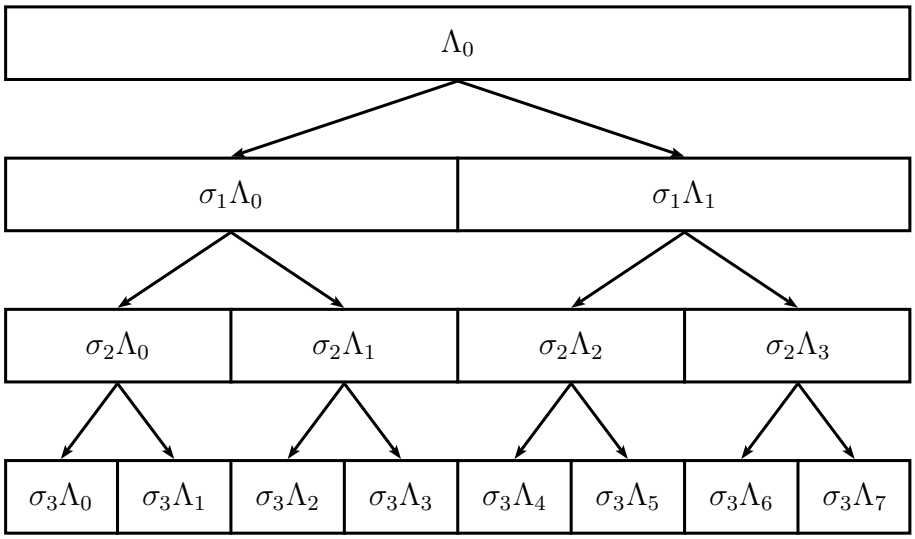

Figure B.3: Tree structure of wavelet packet bases.

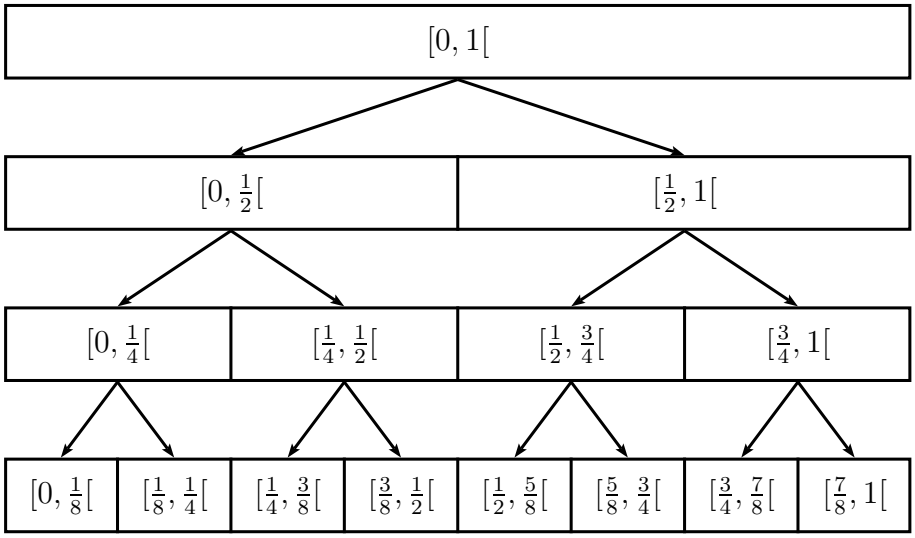

Figure B.4: Correspondence with dyadic covers of [0,1[.

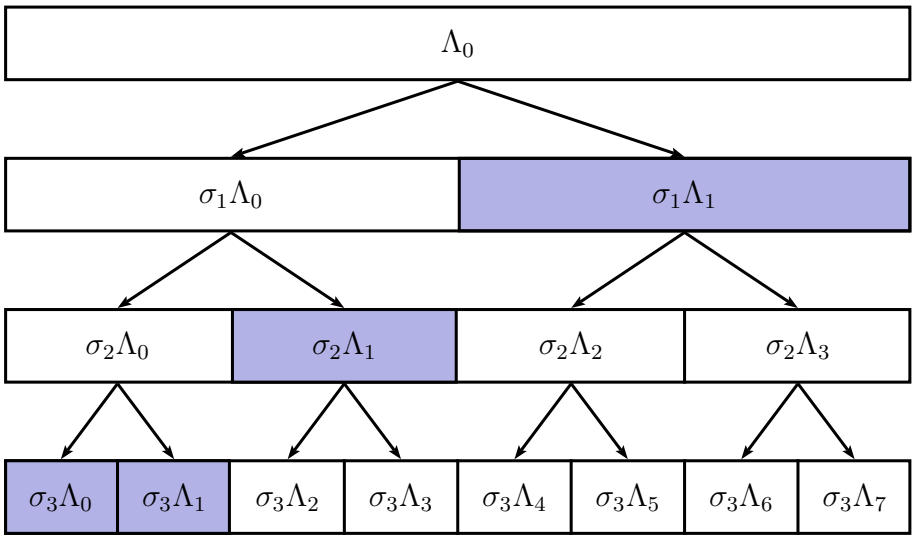

Figure B.5: The wavelet basis.

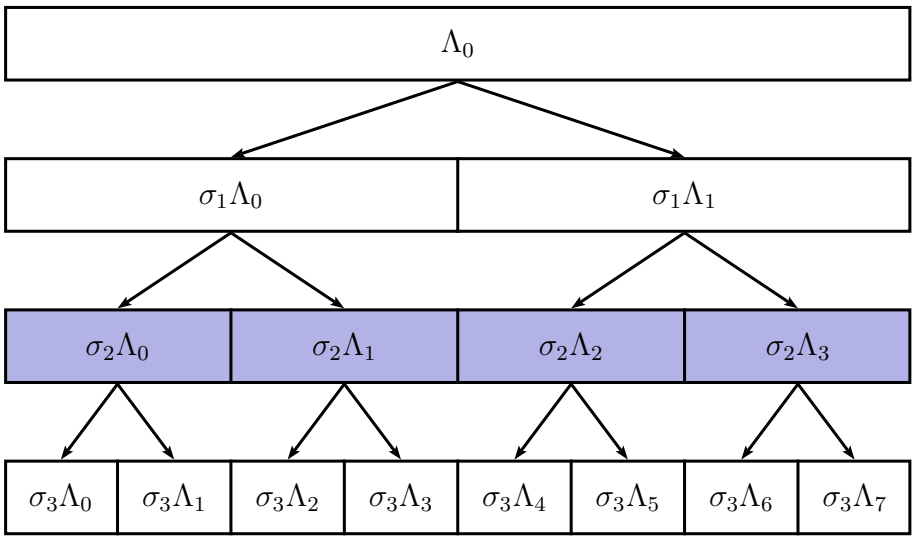

Figure B.6: An example of fixed level wavelet packet basis.

To define an optimal basis, a notion of information cost is needed. Coifman and Wickerhauser [9] propose to use Shannon entropy. In our context, the basis choice will be done with respect to some reference curve x_0 which has to be representative of the data. We compute, for each basis in the library, the Shannon entropy of the coefficients of x_0 in this basis, and select the basis minimizing this entropy. The construction of this best basis, relying on the binary tree structure of the library, is achieved by comparing, at each node, starting from the bottom of the tree, parents with their two children. We decided to take for x_0 the median curve in the sample in the L^2 norm sense. Indeed, it is more likely to present characteristic behaviors of the other curves than the mean, because the mean curve is a smooth average representative, which is probably too easy to approximate with a few basis functions. However, we observed that these two functions surprisingly give rise to almost the same basis. Both choices are thus possible. The mean curve is useful if we know that some noise has to be removed, whereas the median curve seems a better choice to reflect the small-scale irregularities.

We have implemented the algorithm in **R**, and it has been run through all filters available in the **R** package *wmtsa*. These filters belong to four families, extremal phase family (Daubechies wavelets), including Haar basis, least asymmetric family (Symmlets), "best localized" wavelets and Coiflets. For example, the least asymmetric family contains ten different filters "s2", "s4", "s6", "s8", "s10", "s12", "s14", "s16", "s18", "s20". Finally, we keep the clustering result obtained with the basis minimizing the distortion among the various filters. In the sequel, this basis will be called Best-Entropy basis.

In the applications, the performance of the Best-Entropy basis will be compared with the Haar wavelet basis, the Fourier basis and the functional principal component analysis basis. For the sake of completeness, we recall here the definition of these bases.

The Haar wavelet basis Let $\phi(t) = \mathbf{1}_{[0,1]}(t)$ and $\psi(t) = \mathbf{1}_{[0,1/2[}(t) - \mathbf{1}_{[1/2,1]}(t)$. Then, the family $\{\phi, \psi_{j,\ell}\}$, where

$$\psi_{j,\ell}(t) = 2^{j/2}\psi(2^j t - \ell), j \geq 0, 0 \leq \ell \leq 2^j - 1,$$

constitutes a Hilbertian basis of $L^2([0,1])$, called Haar basis.

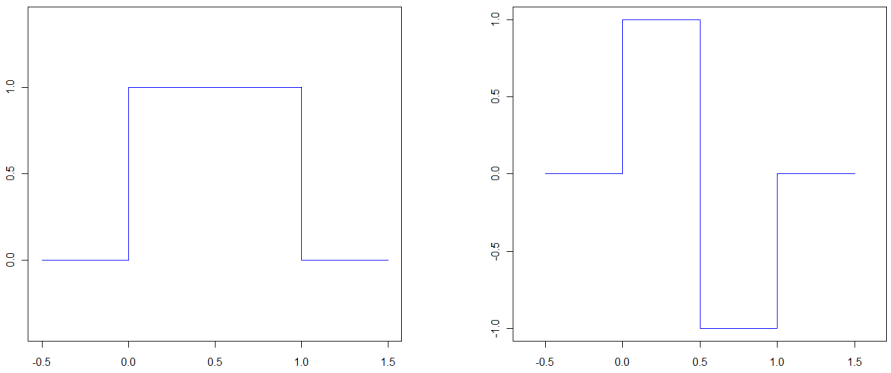

Figure B.7: Haar scaling function ϕ and mother wavelet function ψ.

The Fourier basis The Fourier basis on $[0,1]$ is the complete orthonormal system of $L^2([0,1])$ built with the trigonometric functions

$$\psi_1(t) = 1, \quad \psi_{2j}(t) = \sqrt{2}\cos(2\pi jt), \quad \psi_{2j+1}(t) = \sqrt{2}\sin(2\pi jt), \quad j \geq 1.$$

Functional principal component analysis Principal component analysis for functional data (for short, functional PCA) is the generalization of the usual principal component analysis for vector data. The Euclidean inner product is replaced by the inner product in $L^2([0,1])$. More precisely, functional PCA consists in writing $X_i(t)$ under the form

$$X_i(t) = \mathbb{E}X(t) + \sum_{j=1}^{+\infty} x_{ij}\psi_j(t),$$

where the $(x_{ij})_{j\geq 1}$ and the functions $(\psi_j)_{j\geq 1}$ are defined as follows. At the first step, the function ψ_1 is chosen to maximize

$$\frac{1}{n}\sum_{i=1}^{n} x_{i1}^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\int \psi_1(t)X_i(t)dt\right]^2$$

subject to

$$\int \psi_1(t)^2 dt = 1.$$

Then, each ψ_j is computed by maximizing

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2$$

subject to

$$\int \psi_j(t)^2 dt = 1$$

and to the orthogonality constraints

$$\int \psi_\ell(t)\psi_j(t)dt = 0,\ 1 \leq \ell \leq j-1.$$

Functional PCA can be characterized in terms of the eigenanalysis of covariance operators. If $(\lambda_j)_{j\geq 1}$ denotes the eigenvalues and $(\psi_j)_{j\geq 1}$ the eigenfunctions of the operator $\mathcal{C}$ defined by $\mathcal{C}(f)(s) = \int_0^1 C(s,t)f(t)dt$, where $C(s,t) = \mathrm{cov}(X(s), X(t))$, then

$$X_i(t) = \mathbb{E}X(t) + \sum_{j=1}^{+\infty} x_{ij}\psi_j(t),$$

where the x_j are uncorrelated centered random variables with variance $\mathbb{E}[x_{ij}^2] = \lambda_j$. There are similarities with the context of Example B.2.2, but here the kernel depends on X. In practice, the decomposition can easily be computed with discrete matrix operations, replacing $C(s,t)$ by the covariance matrix of the X_i. This basis has some nice properties. In particular, considering a fixed number of coefficients, it minimizes among all orthogonal bases the average squared L^2 distance between the original curve and its linear representation (see, e.g., the book of Ghanem and Spanos [19]). For more details on functional PCA, we refer the reader to Ramsay and Silverman [32].

Observe that since the functional PCA basis is a stochastic basis and the Best-Entropy basis algorithm also uses the data, rigorously the sample should be divided into two subsamples, one to build the basis, and the other for clustering.

B.4 Experimental results and analysis

We evaluated the performance of the clustering method with projection, using the various bases described in the previous section, for two different kinds of curves. First, a simulated example where the right clusters are known is discussed, to illustrate the efficiency of the method. Then, we focus on our industrial problem and cluster output curves of a "black box" computer code.

Observe that, although the curves considered in Section B.2 and Section B.3 were true functions, in practice, we have to deal with curves sampled on a finite number of discretization points. Therefore, a preprocessing step based on spline interpolation is necessary.

B.4.1 Synthetic control chart time series

Control chart time series are used for monitoring process environments, to achieve appropriate control and to produce high quality products. Different types of series can be encountered, but only one, a kind of white noise, indicates a normal working. All the other types of series must be detected, because they correspond to abnormal behavior of the process.

The data set contains a few hundreds to a few thousands curves generated by the process described in Alcock and Manolopoulos [2], discretized on 128 time points. There are six types of curves: normal, cyclic, increasing trend, decreasing trend, upward shift and downward shift, which are represented in Figure B.8. The equations which generated the data are indicated below.

(A) *Normal pattern*: $y(t) = m + rs$ where $m = 30$, $s = 2$ and $r \sim \mathcal{U}(-3, 3)$.

(B) *Cyclic pattern*: $y(t) = m + rs + a \sin \frac{2\pi t}{T}$ where $a, T \sim \mathcal{U}(10, 15)$.

(C) *Increasing shift*: $y(t) = m + rs + gt$ with $g \sim \mathcal{U}(0.2, 0.5)$.

(D) *Decreasing shift*: $y(t) = m + rs - gt$.

(E) *Upward shift*: $y(t) = m + rs + hx$ where $x \sim \mathcal{U}(7.5, 20)$, $h = \mathbf{1}_{[t_0, D]}$, $t_0 \sim \mathcal{U}\left(\frac{D}{3}, \frac{2D}{3}\right)$, and D is the number of discretization points.

(F) *Downward shift*: $y(t) = m + rs - hx$.

The two main advantages using this synthetic data set is that we can simulate as many curves as we wish and we know the right clusters.

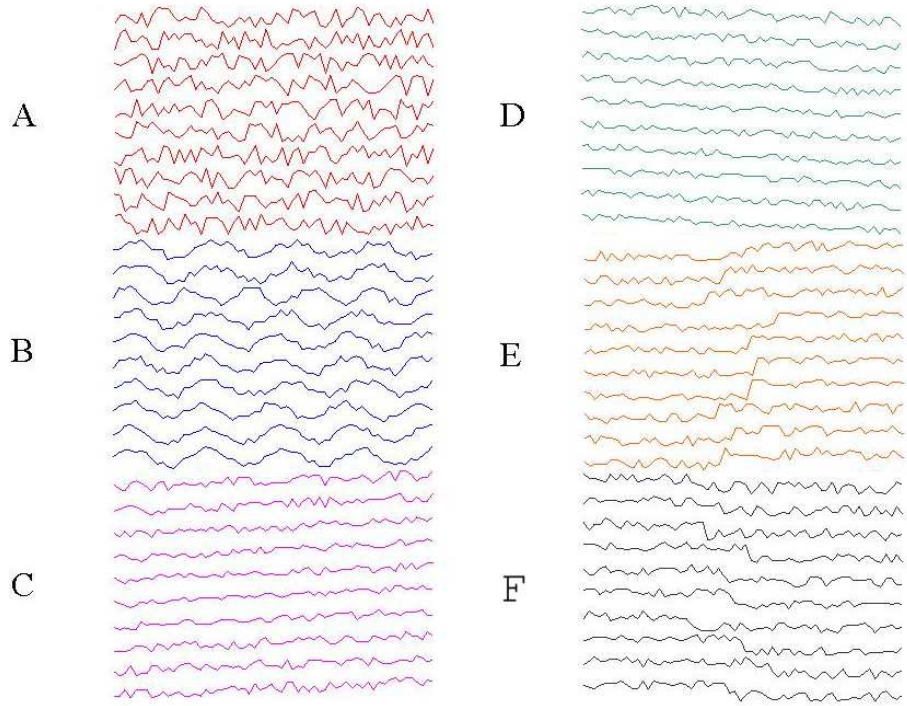

Figure B.8: 10 example curves for each of the 6 types of control chart.

The k-means algorithm on the projected coefficients has been run for all four bases (Best-Entropy, Haar, functional PCA and Fourier basis), with varying sample size n and projection dimension d. Since the result of a k-means algorithm may depend on the choice of the initial centers, the algorithm is restarted 100 times. The maximum number of iterations per run is set to 500. This program tries to globally minimize the projected empirical distortion $W_{d,n}(\mathbf{c})$. To evaluate its performance, we compute an approximation $W(d, n)$ of the distortion $W_\infty(\hat{\mathbf{c}}_{d,n})$ using a set of 18000 sample curves. This is possible in this simulated

example, since we can generate as many curves as we want. The distortion $W(d, n)$ is computed for d varying from 2 to 50 and n ranging from 100 to 3100. We restricted ourselves to the case $d \leq 50$, since there are only 128 discretization points. Moreover, as pointed out earlier, d must not be too large for computational complexity reasons. Indeed, a projection dimension $d = 50$ is already high for a practical use.

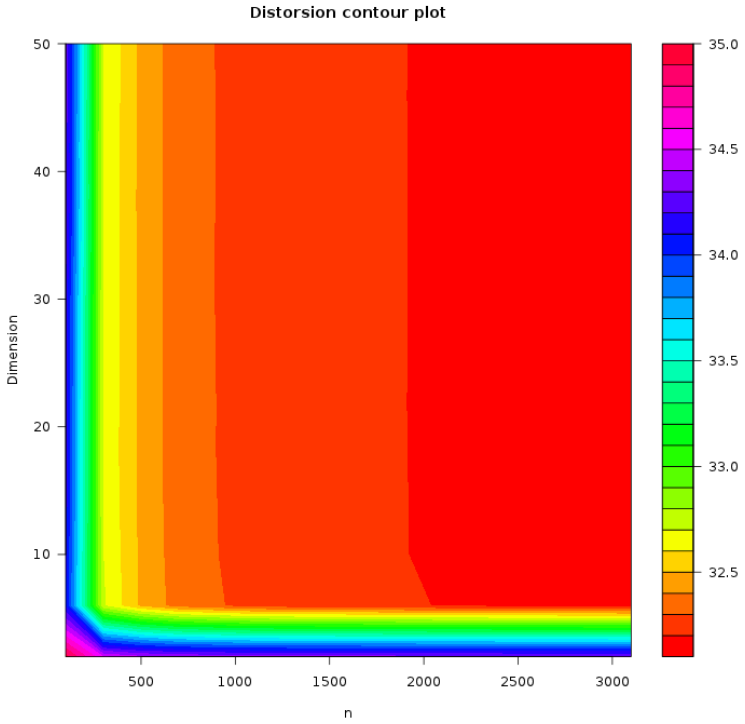

Figure B.9: Contour plot of $W(d, n)$ for the functional PCA basis.

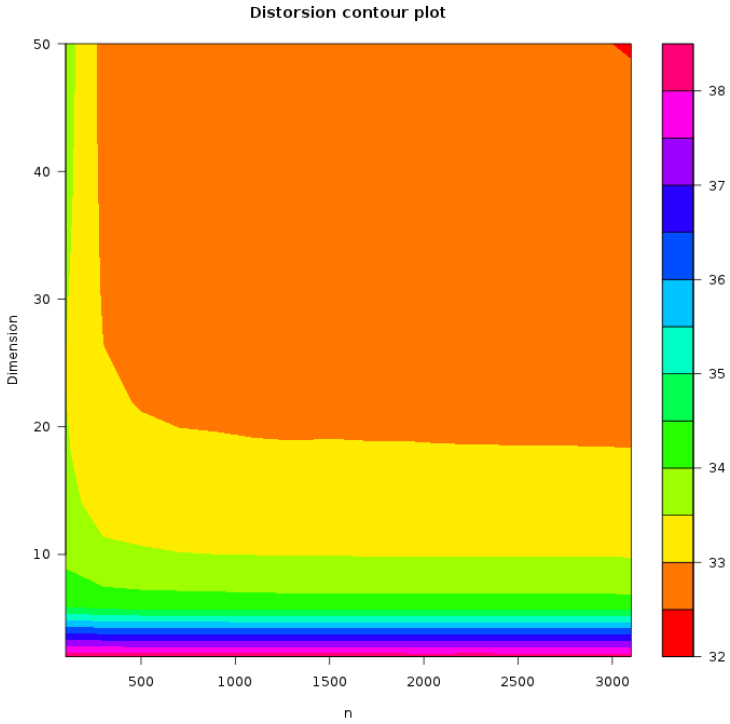

Figure B.10: Contour plot of $W(d, n)$ for the Haar basis.

Figures B.9 and B.10 show the contours plots corresponding to the evolution of $W(d, n)$ as a function of d and n, for the functional PCA and the Haar basis. We remark that the norm of the gradient of $W(d, n)$ vanishes when d and n are close to their maximal values. Hence, as expected according to Theorem B.2.1, $W(d, n)$ is decreasing in d and n. When d or n is too small (for instance $d = 2$ or $n = 100$), the clustering results are inaccurate. Besides, they are not stable with respect to the n observations chosen. However, for larger values of these parameters, the partitions obtained quickly become satisfactory. The choice $n \geq 300$ together with $d \geq 6$ generally provides good results and is reasonably low for many applications.

Figure B.11 shows the curve corresponding to the evolution of $W(d, n)$ as a function of d for $n = 500$, for all four bases, whereas Figure B.12 represents the evolution of $W(d, n)$ versus n for $d = 10$. According to Section 2, φ_d must be of the order $\sqrt{n}$. For $n = 500$, $\sqrt{n}$ is about 22. Considering that φ_d and d are approximately of the same order, a projection dimension close to 22 should thus be suitable. Indeed, we see via Figure B.11 that $W(d, n)$ does not decrease much more after this value.

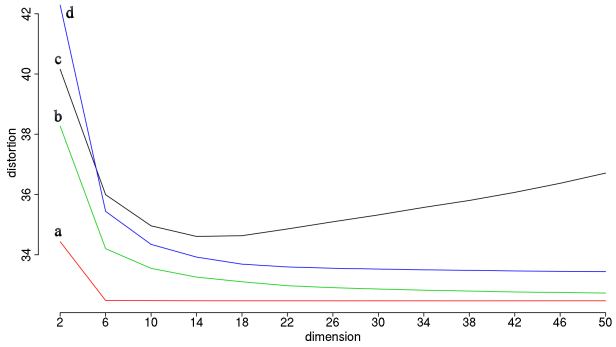

Figure B.11: Evolution of $W(d, n)$ for $n = 500$ and d ranging from 2 to 50, for (a) functional PCA basis, (b) Haar basis, (c) Fourier basis and (d) Best-Entropy basis.

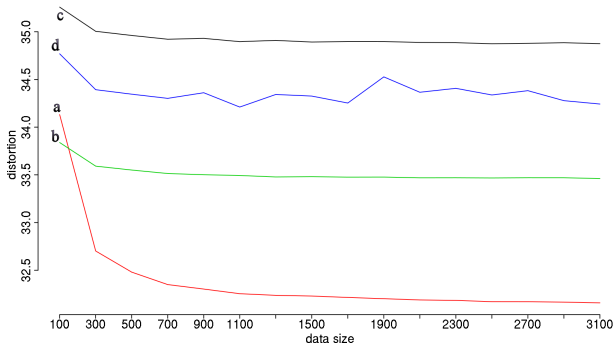

Figure B.12: Evolution of $W(d, n)$ for $d = 10$ and n ranging from 100 to 3100, for (a) functional PCA basis, (b) Haar basis, (c) Fourier basis and (d) Best-Entropy basis.

The evolution of the distortion for the Fourier basis looks quite odd: it shows a first decreasing step before increasing again. However, this increasing can be explained in the following way. The centers chosen first are wrong, but seem to give a better distortion than the "real" clustering. When the dimension grows large enough, these first wrong centers no longer represent a local minimum, and the k-means algorithm moves slowly toward the "right" clustering, although losing a bit in distortion. This interpretation is confirmed if we look at the clusters corresponding to each distortion. Furthermore, when the dimension is high relatively to the number of discretization points, some basis functions which oscillate a lot may not be sampled correctly. As a result, the coefficients estimated by approximating the inner products can become very inaccurate as d increases. For very high d, these computed coefficients confound with some noise. Consequently, data becoming more noisy without adding any information, the distortion will increase. The other bases tend to oscillate too, so that they would probably show the same behavior if d were increased above 50. Besides, the small fluctuations observed for the Best-Entropy basis indicate that this basis is not suitable for clustering of control chart time series. The functional PCA basis always gives the lowest distortion. However, the distortions obtained for the three other bases are quite similar, with a preference for the Haar basis over the Best-Entropy wavelet basis, the Fourier basis being the worst choice. As an example, Table B.1 gives the values of $W(d, n)$ for $n = 1100$ and $d = 30$.

Basis	Fourier	Functional PCA	Haar	Best-Entropy
Distortion	35.3	32.3	32.8	33.4

Table B.1: Distortion $W(d,n)$ for $n = 1100$ and $d = 30$.

Figure B.13 represents the 6 clusters for the Fourier basis, for $n = 300$ and $d = 10$, whereas Figure B.14 shows them for $d = 30$. The classes obtained with the algorithm are shown in colors, and the real clusters are indicated in the caption. For relatively small values of d, the normal and cyclic patterns are merged into one big cluster, and one cluster corresponding to increasing (or decreasing) shift pattern is split in two. For large enough d, the normal and cyclic designs are well detected, and the overall clustering is correct despite some mixing increasing-upward shift or decreasing-downward shift. We also tested the algorithm for smaller values of the number k of classes. As expected, for the particular choice $k = 3$, clusters A and B are merged into one single group, and the same occurs for the cluster pairs $\{C, E\}$ and $\{D, F\}$.

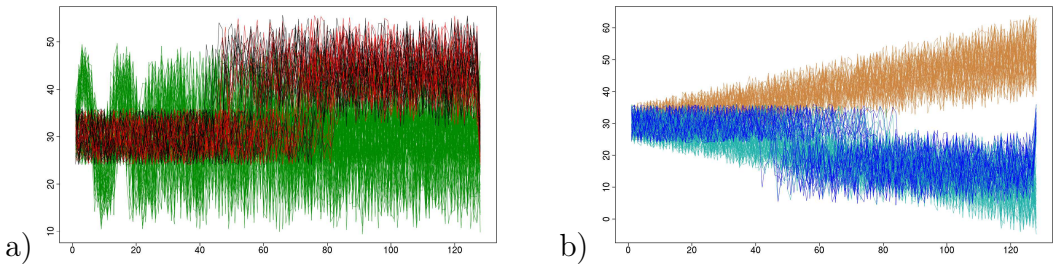

Figure B.13: (a) Clusters A and B in green, cluster E in red and black. (b) Clusters C, D and F in brown, light blue and blue respectively. (Fourier basis, $n = 300$, $d = 10$.)

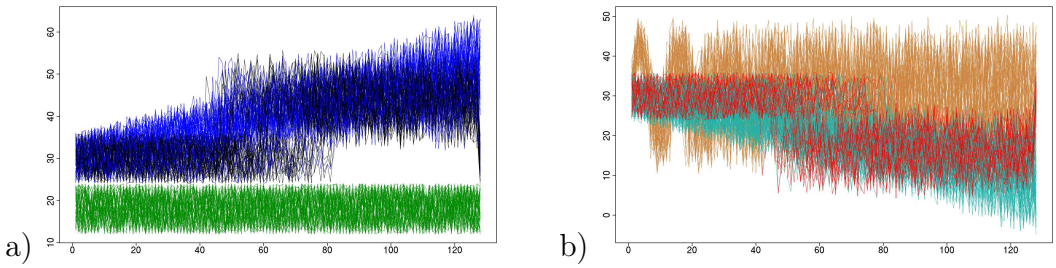

Figure B.14: (a) Clusters A, C and E in green, blue, black. (b) Clusters B, D and F in brown, light blue and red. (Fourier basis, $n = 300$, $d = 30$.)

B.4.2 Industrial code examples

Let us now turn to the industrial issue which motivated our study. As explained in the introduction, the computer codes used in nuclear engineering have become very complex and costly in CPU-time consumption. That is why we try to approximate them with a cheap function substituted to the code. In order to build a regression model, a preliminary analysis of the different types of outputs is essential. This leads to data clustering, applied here to

a computer code with functional outputs. Two different kinds of outputs are presented, the temperature evolution with a data set containing 100 curves, and the thermal exchange coefficient evolution with a data set of 200 curves.

Temperature curves The data is made of 100 CATHARE code outputs representing the evolution of the temperature in the vessel annular space (Figure B.15). Here, the sample size is fixed to $n_0 = 100$. However, the discretization can be controlled to some extent with spline interpolation. In this case, 256 discretization points are used.

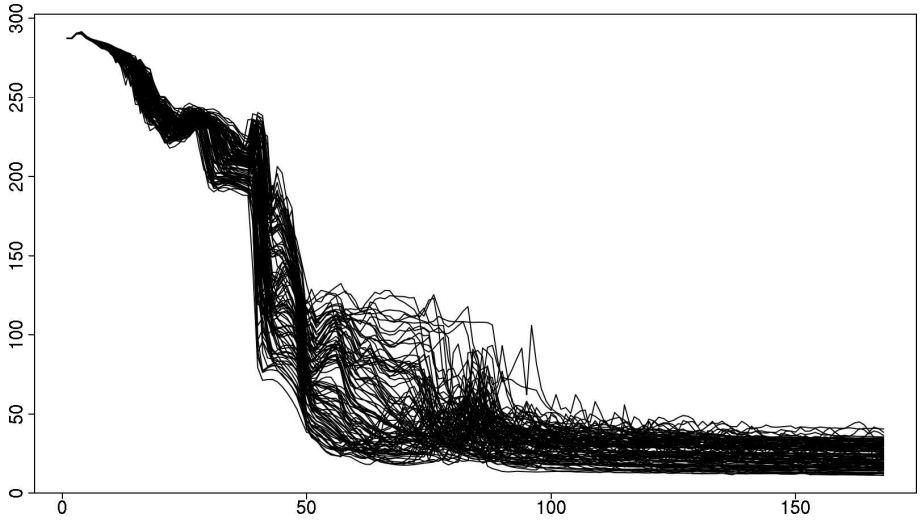

Figure B.15: The 100 temperature curves.

Observe that all curves converge in the long-time limit to the same value, corresponding to the temperature of the cold water injected. These curves have been clustered, for physical reasons pertaining to nuclear engineering, in two groups. More precisely, there is a critical set of physical parameters beyond which the thermal shock is more violent and the temperature changes more rapidly (see Auder, De Crecy, Iooss, and Marquès [5] for more details). The algorithm on the projected coefficients has been run for the Best-Entropy, Haar, functional PCA and Fourier bases, with varying dimension, with the same settings as before. Since we consider real-life data, it is not possible to compute an approximation of $W_\infty(\hat{\mathbf{c}}_{d,n})$ as in the simulated example. Hence, the distortion $W(d, n_0)$ is simply the output $W_{d,n_0}(\hat{\mathbf{c}}_{d,n_0})$ of the clustering algorithm, with fixed $n_0 = 100$. This distortion is computed for d varying from 2 to 50. Figure B.16 shows the curve corresponding to the evolution of $W(d, n)$ as a function of d. As expected, it is decreasing in d.

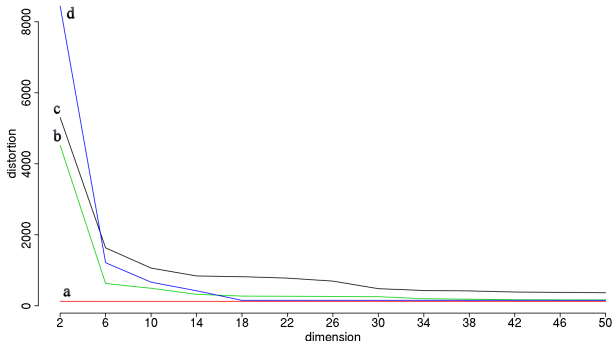

Figure B.16: Evolution of $W(d, n_0)$ for the 100 temperature curves, d ranging from 2 to 50, for (a) functional PCA basis, (b) Haar basis, (c) Fourier basis and (d) Best-Entropy basis.

Basis	Fourier	Functional PCA	Haar	Best-Entropy
Distortion	480.8	125.3	254.9	151.5

Table B.2: Distortion values for $d = 30$.

We note that until $d = 16$, the Haar basis provides lower distortion, but for larger values of d, the Best-Entropy basis is better. As before, the Fourier basis is the worst and the functional PCA basis is the best. This can also be checked from Table B.2, which presents the distortion obtained for each basis. Although the functional PCA basis gives the best result in terms of distortion, we see that using any of the other three bases is not that bad. Indeed, the same partitioning is found every time (Figure B.17).

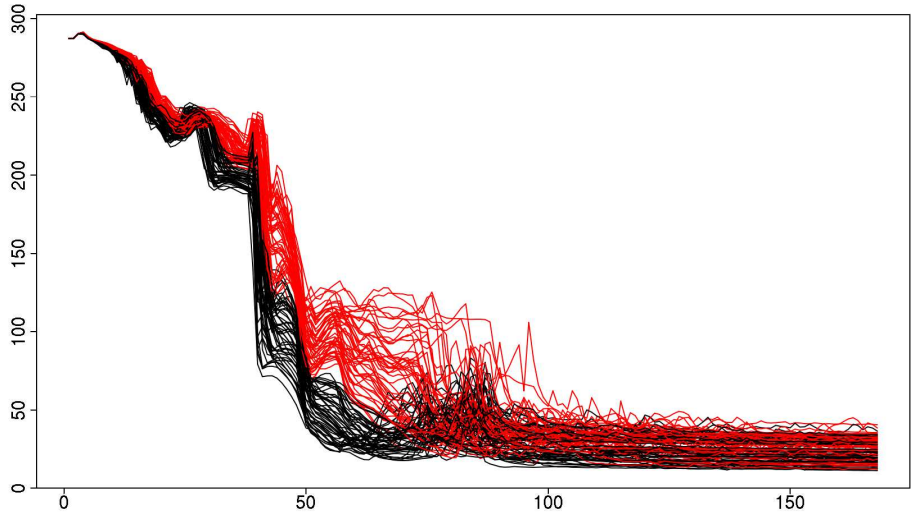

Figure B.17: The temperature curves divided in two groups.

Finally, Figure B.18 shows the two centers representing the classes obtained for $d = 30$ with the Fourier basis, functional PCA basis, Haar basis and Best-Entropy basis. The two curves obtained with the functional PCA basis characterize with an especially good accuracy the shape of the data items in the corresponding clusters.

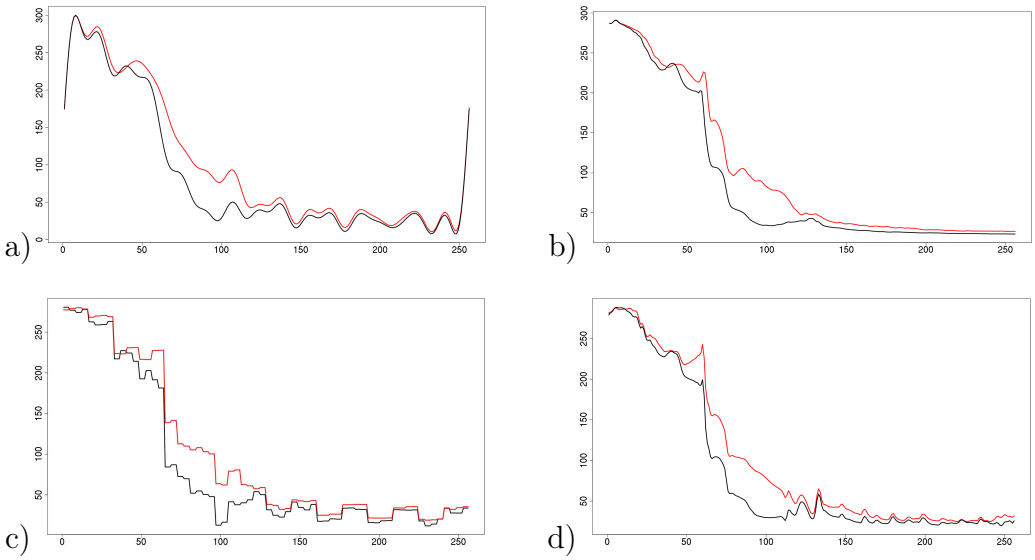

Figure B.18: The two centers for $d = 30$ for (a) Fourier, (b) functional PCA, (c) Haar and (d) Best-Entropy basis.

Thermal exchange coefficient curves Figure B.19 shows all 200 CATHARE code outputs. Here, the number of discretization points is set to 1024. The data has been partitioned in three groups. As for the temperature curves, $W(d, n_0)$ is computed for d varying from 2 to 50 ($n_0 = 200$).

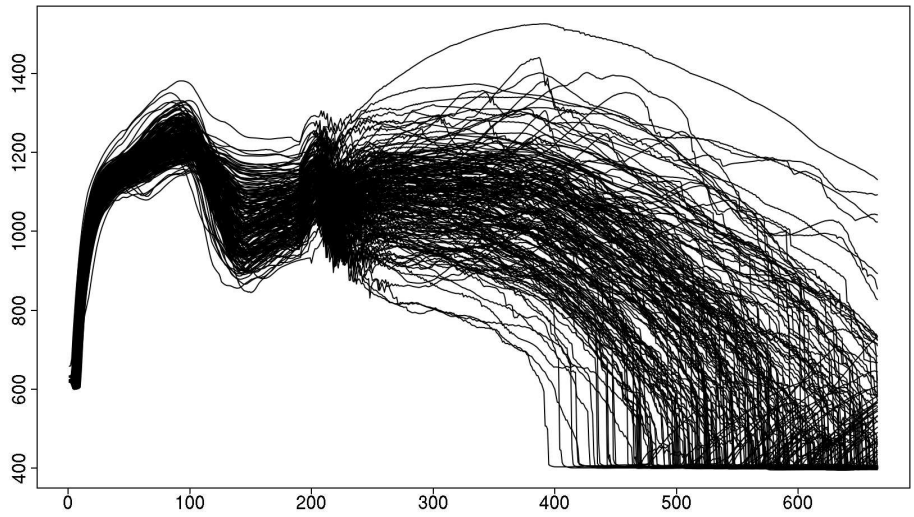

Figure B.19: The 200 thermal exchange coefficient curves.

We can see via Figure B.20 that $W(d, n_0)$ is decreasing in d again. The functional PCA basis is still the best choice, with a fast convergence (stabilization from $d = 10$). Interestingly, the Fourier basis shows smaller distortion values than the two wavelets basis in this case, as Table B.3 indicates.

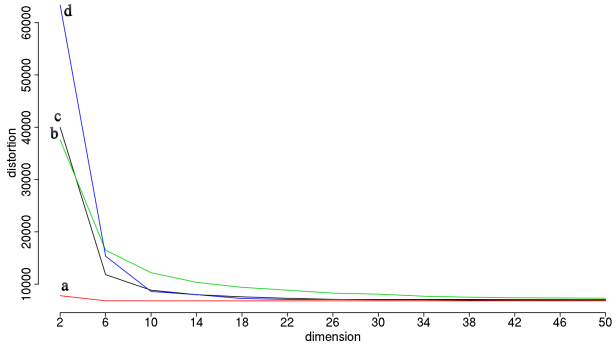

Figure B.20: Evolution of $W(d, n_0)$ for the 200 thermal exchange coefficient curves, d ranging from 2 to 50 for (a) functional PCA basis, (b) Haar basis, (c) Fourier basis and (d) Best-Entropy basis.

Basis	$d = 6$	$d = 10$	$d = 18$	$d = 26$	$d = 34$	$d = 42$	$d = 50$
Fourier	11810	8843.8	7570.0	7097.0	7013.2	6893.3	6881.2
Functional PCA	6815.8	6802.0	6801.4	6801.2	6801.1	6801.1	6801.0
Haar	16491	12185	9385.2	8279.3	7688.0	7390.6	7313.9
Best-Entropy	15338	8596.2	7244.5	7082.1	7082.1	7082.1	6801.0

Table B.3: Distortion values for the thermal exchange coefficient curves.

All the partitions obtained are very similar. A typical example is given by Figure B.21. However, as for the temperature curves, it is interesting to look at the curves selected as centers. Figure B.22 shows the three centers obtained with the Fourier basis, functional PCA basis, Haar basis and Best-Entropy basis, for $d = 30$.

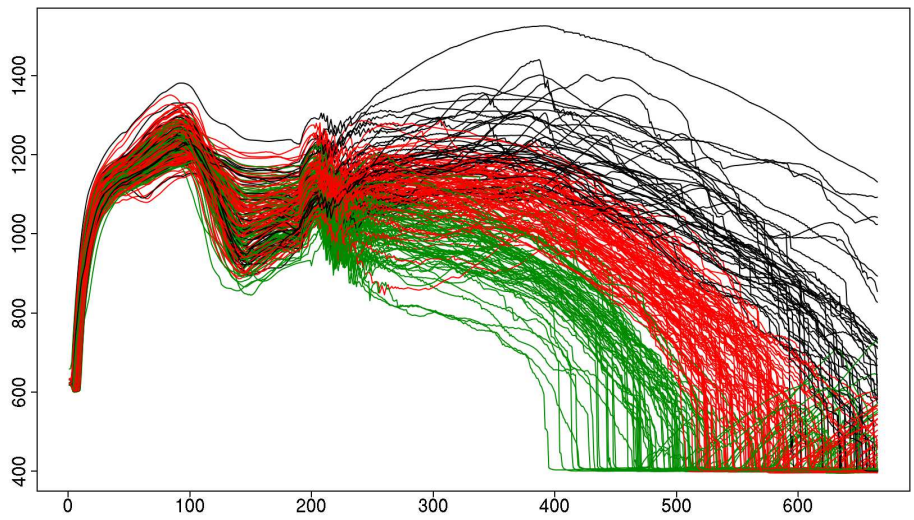

Figure B.21: Three clusters obtained with $d = 14$, functional PCA basis.

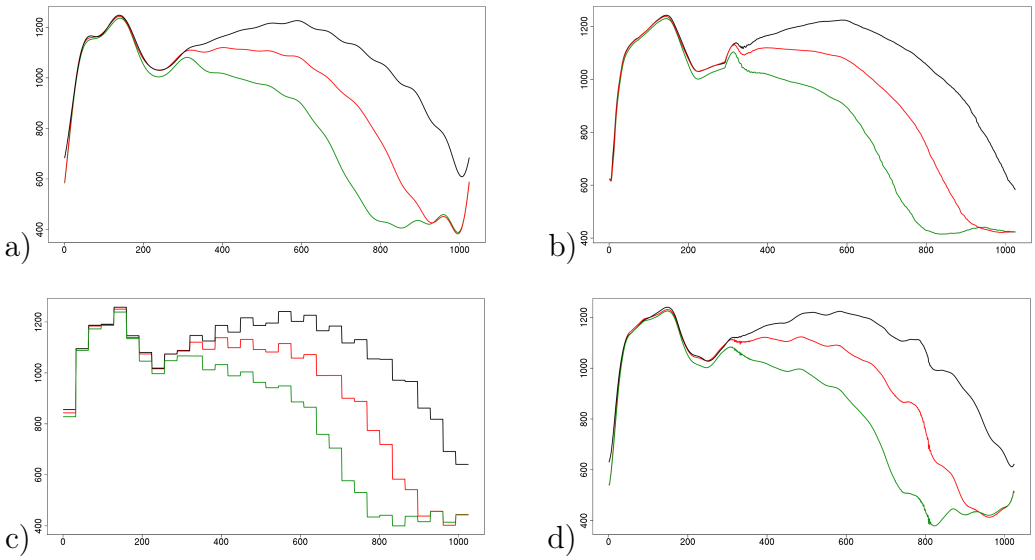

Figure B.22: The three centers with (a) Fourier, (b) functional PCA, (c) Haar and (d) Best-Entropy basis, for $d = 30$.

B.5 Conclusion

These clusters allow to build accurate models for the industrial application. The partitioning method presented in this article has been integrated in our metamodel written in **R**. More specifically, given an array of n input vectors corresponding to n output curves, the purpose is to learn a function $\phi : z \mapsto x$ mapping an input vector to a continuous curve. In order to improve the accuracy of this task, we begin with a clustering step and then look for a regression model in each cluster separately. The metamodel lets the user choose between several clustering techniques, either assuming some clusters shapes (like this projected k-means) or trying to discover them in data (like the ascendant hierarchical clustering). The latter are attractive as they do not make assumptions about the results, but they generally need a relatively good sampling of the data. Consequently, the k-means-like techniques are useful in many of our industrial applications, where only a few samples are available. Moreover, these methods provide easily interpretable clusters. In each cluster, after a dimension reduction step, which can either be achieved through the decomposition on an orthonormal basis (linear), or any manifold learning algorithm (nonlinear, with the assumption that the outputs lie on a functional manifold), a statistical learning method is applied to predict representation within this cluster. The mainly used method at this stage is the Projection Pursuit Regression algorithm (see Friedman, Jacobson, and Stuetzle [16]). Finally, a simple k-nearest neighbors classifier gives the most probable cluster for a new input, the corresponding regression function is applied, and the curve can be reconstruct from its predicted representation.

For the moment, our metamodel with the clustering method presented have successfully been used on two different scenarios involving the CATHARE code (minor or major break, for each we get temperature, pressure and thermal exchange coefficient curves).

Another research track could consist in considering other types of distances between curves. Distances involving derivatives might be hard to estimate on the thermal exchange

coefficient dataset, because several curves are varying rapidly over short period of time, contrasting for instance with the Tecator dataset (http://lib.stat.cmu.edu/datasets/tecator), on which such distances proved successful (Ferraty and Vieu [15, Chapter 8], Rossi and Villa [33]). However, further investigations are needed to know if a smoothing step before clustering based on m-order derivatives would lead to improved results. As the "true" classes are unknown, such a procedure can only be validated within a cross validation framework involving the full metamodel. Experiments with the L^1 distance or some mixed distances related to functions shapes (Heckman and Zamar [22]) could also be studied in future research.

B.6 Proofs

B.6.1 Proof of Lemma B.2.1

If we define the remainder R_d by $R_d(\mathbf{x}) = \mathbf{x} - \Pi_d(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{S}$, then for $(\mathbf{x}, \mathbf{y}) \in \mathcal{S}^2$,

$$\begin{aligned}
\|R_d(\mathbf{x}-\mathbf{y})\|^2 &\leq 2\|R_d(\mathbf{x})\|^2 + 2\|R_d(\mathbf{y})\|^2 \\
&= 2\sum_{j=d}^{+\infty} x_j^2 + 2\sum_{j=d}^{+\infty} y_j^2 \\
&= 2\sum_{j=d}^{+\infty} \frac{\varphi_j x_j^2}{\varphi_j} + 2\sum_{j=d}^{+\infty} \frac{\varphi_j y_j^2}{\varphi_j} \\
&\qquad (\varphi_j > 0 \text{ for all } j \geq d, \text{ since } d > j_0) \\
&\leq 2\sum_{j=d}^{+\infty} \frac{\varphi_j x_j^2}{\varphi_d} + 2\sum_{j=d}^{+\infty} \frac{\varphi_j y_j^2}{\varphi_d} \\
&\leq \frac{4R^2}{\varphi_d}.
\end{aligned}$$

Thus, for $\mathbf{c} \in \mathcal{S}^k$,

$$\begin{aligned}
W_\infty(\mathbf{c}) - W_d(\mathbf{c}) &= \mathbb{E}\Big[\min_{\ell=1,\dots,k} \|X - c_\ell\|^2 - \min_{\ell=1,\dots,k} \|\Pi_d(X) - \Pi_d(c_\ell)\|^2\Big] \\
&= \mathbb{E}\Big[\min_{\ell=1,\dots,k} \|\Pi_d(X) + R_d(X) - \Pi_d(c_\ell) - R_d(c_\ell)\|^2 \\
&\quad - \min_{\ell=1,\dots,k} \|\Pi_d(X) - \Pi_d(c_\ell)\|^2\Big] \\
&= \mathbb{E}\Big[\min_{\ell=1,\dots,k} \big(\|\Pi_d(X) - \Pi_d(c_\ell)\|^2 + \|R_d(X) - R_d(c_\ell)\|^2\big) \\
&\quad - \min_{\ell=1,\dots,k} \|\Pi_d(X) - \Pi_d(c_\ell)\|^2\Big] \\
&\qquad \text{(since } \Pi_d \text{ is the orthogonal projection on } \mathbb{R}^d) \\
&\leq \mathbb{E}\Big[\max_{\ell=1,\dots,k} \|R_d(X) - R_d(c_\ell)\|^2\Big] \\
&\leq \frac{4R^2}{\varphi_d}.
\end{aligned}$$

Hence,

$$\sup_{\mathbf{c}\in\mathcal{S}^k}[W_\infty(\mathbf{c}) - W_d(\mathbf{c})] \leq \frac{4R^2}{\varphi_d},$$

as desired.

B.6.2 Proof of Theorem B.2.1

We have

$$W_\infty(\hat{\mathbf{c}}_{d,n}) - W_\infty^* = W_\infty(\hat{\mathbf{c}}_{d,n}) - W_d(\hat{\mathbf{c}}_{d,n}) + W_d(\hat{\mathbf{c}}_{d,n}) - W_d^* + W_d^* - W_\infty^*.$$

According to Lemma B.2.1, on the one hand,

$$\begin{aligned} W_\infty(\hat{\mathbf{c}}_{d,n}) - W_d(\hat{\mathbf{c}}_{d,n}) &\leq \sup_{\mathbf{c}\in\mathcal{S}^k}[W_\infty(\mathbf{c}) - W_d(\mathbf{c})] \\ &\leq \frac{4R^2}{\varphi_d}, \end{aligned}$$

and on the other hand,

$$\begin{aligned} W_d^* - W_\infty^* &= \inf_{\mathbf{c}\in\mathcal{S}^k} W_d(\mathbf{c}) - \inf_{\mathbf{c}\in\mathcal{S}^k} W_\infty(\mathbf{c}) \\ &\leq \sup_{\mathbf{c}\in\mathcal{S}^k}[W_\infty(\mathbf{c}) - W_d(\mathbf{c})] \\ &\leq \frac{4R^2}{\varphi_d}, \end{aligned}$$

and the theorem is proved.

References

[1] C. Abraham, P. A. Cornillon, E. Matzner-Løber, and N. Molinari. Unsupervised Curve Clustering using B-Splines. *Scandinavian Journal of Statistics*, 30:581–595, 2003.

[2] R. J. Alcock and Y. Manolopoulos. Time-series similarity queries employing a feature-based approach. In *7th Hellenic Conference on Informatics*, Ioannina, Greece, 1999.

[3] A. Antoniadis, X. Brossat, J. Cugliari, and J-M. Poggi. Clustering Functional Data using Wavelets. In *COMPSTAT 2010*. Compstat, in press, 2010.

[4] N. Aronszajn. Theory of reproducing kernel. *Transactions of American Mathematical Society*, 68(3):337–404, 1950.

[5] B. Auder, A. De Crecy, B. Iooss, and M. Marquès. Screening and metamodeling of computer experiments with functional outputs. Application to thermal-hydraulic computations. *Journal of Reliability Engineering and System Safety*, Submitted, 2011. URL http://hal.archives-ouvertes.fr/docs/00/52/54/91/PDF/ress_samo10_BA.pdf.

[6] G. Biau, L. Devroye, and G. Lugosi. On the performance of clustering in Hilbert spaces. *IEEE Transactions on Information Theory*, 2008.

[7] J. M. Chiou and P. L. Li. *Functional and Operatorial Statistics*, chapter Functional clustering of longitudinal data. Physica-Verlag. Springer, 2008. 103–107.

[8] A. Cohen, I. Daubechies, and P. Vial. Wavelets on the interval and fast wavelet transforms. *Applied and Computational Harmonic Analysis*, 1:54–81, 1993.

[9] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE transactions on information theory*, 38:713–718, 1992.

[10] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2002.

[11] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[12] R. A. DeVore and G. G. Lorentz. *Constructive Approximation*. Springer-Verlag, Berlin, 1993.

[13] D. L. Donoho and I. M. Johnstone. Minimax estimation via wavelet shrinkage. *Annals of Statistics*, 26:879–921, 1998.

[14] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[15] F. Ferraty and P. Vieu. *Nonparametric Functional Data Analysis: Theory and Practice*. Springer Series in Statistics. Springer-Verlag, New York, 2006.

[16] J. H. Friedman, M. Jacobson, and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–846, 1981.

[17] S. Gaffney. *Probabilistic Curve-Aligned Clustering and Prediction with Mixture Models*. PhD thesis, Department of Computer Science, University of California, Irvine, 2004.

[18] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[19] R. G. Ghanem and P. D. Spanos. *Stochastic finite elements: a spectral approach*. Springer-Verlag, New York, 1991.

[20] A. D. Gordon. *Classification*, volume 82 of *Monographs on Statistics and Applied Probability*. Chapman Hall/CRC, second edition, 1999.

[21] S. Graf and H. Luschgy. *Foundations of quantization for probability distributions*. Lecture Notes in Mathematics. Springer-Verlag, Berlin, 2000.

[22] N.E. Heckman and R.H. Zamar. Comparing the shapes of regression functions. *Biometrika*, 87(1):135–144, 2000.

[23] G. M. James and C. A. Sugar. Clustering for Sparsely Sampled Functional Data. *Journal of the American Statistical Association*, 98:397–408, 2003.

[24] T. Linder. Learning-Theoretic Methods in Vector Quantization. In L. Györfi, editor, *Principles of Nonparametric Learning*, CISM lecture Notes. Springer, 2002.

[25] S. P. Lloyd. Least Squares Quantization in PCM. *Bell Telephone Laboratories, 1957 (non published), IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[26] W. L. Loh. On Latin Hypercube Sampling. *The Annals of Statistics*, 24:2058–2080, 1996.

[27] S. Mallat. *A wavelet tour of signal processing, The Sparse Way*. Academic Press, third edition, 2008.

[28] M. D. McKay, W. J. Conover, and R. J. Beckman. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21:239–245, 1979.

[29] N. Merhav and J. Ziv. On the amount of statistical side information required for lossy data compression. *IEEE Transactions on Information Theory*, 43(4), July 1997.

[30] Y. Meyer. *Wavelet and Operators*. Cambridge University Press, Cambridge, 1992.

[31] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–79, 1985.

[32] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, second edition, 2006.

[33] F. Rossi and N. Villa. Support vector machine for functional data classification. *Neurocomputing*, 69:730–742, 2006.

[34] F. Rossi, B. Conan-Guez, and A. El Golli. Clustering Functional Data with the SOM algorithm. In *ESANN'2004 proceedings - European Symposium on Artificial Neural Networks*, pages 305–312, Bruges (Belgium), 28-30 April 2004.

[35] A. B. Tsybakov. On the best rate of adaptive estimation in some inverse problems. *Comptes-rendus de l'Académie des Sciences Paris*, 2000.

[36] A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Series in Statistics. Springer, 2009.

[37] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A. K. Peters, Wellesley, 1994.

[38] R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization Performance of Regularization Networks and Support Vector machines via Entropy Numbers of Compact Operators. *IEEE Transactions on Information Theory*, 47(6):2516–2532, September 2001.

Annexe C

Criblage et métamodélisation fonctionnelle

Ce second article présente une méthodologie complète d'analyse de données industrielles, depuis la sélection des variables d'entrée jusqu'à la (méta)modélisation du code de calcul. Il a été rédigé avec Agnès De Crecy (CEA), Bertrand Iooss (EDF) et Michel Marquès (CEA), et est soumis Journal of Reliability Engineering and System Safety.

Résumé de l'article

Afin de réaliser des analyses d'incertitudes, de sensibilité et d'optimisation sur une variable scalaire en sortie d'un code coûteux en temps CPU, une méthodologie couramment adoptée consiste à d'abord identifier les entrées les plus influentes (par criblage) avant de remplacer le code par une fonction de coût négligeable appelée métamodèle. Ce papier étend la méthodologie au cas d'une sortie fonctionnelle, par exemple quand la réponse du modèle est une courbe ($1D$). Le criblage est alors basé sur l'analyse de la variance et l'ACP fonctionnelle sur les sorties. Le métamodèle fonctionnel consiste en une étape de classification non supervisée (optionnelle), puis une étape de réduction de dimension suivie de la construction d'un métamodèle vectoriel classique. Une application industrielle traitant des incertitudes lors d'un choc thermique pressurisé dans un réacteur nucléaire vient illustrer toutes ces étapes.

Mots-clé : sortie fonctionnelle, transitoire de choc thermique pressurisé, analyse en composantes principales, analyse d'incertitudes et de sensibilité.

Screening and metamodeling of computer experiments with functional outputs. Application to thermal-hydraulic computations

Benjamin AUDER*, Agnès DE CRECY†, Bertrand IOOSS‡ and Michel MARQUÈS*

Submitted to: *Reliability Engineering and System Safety*
for the special SAMO 2010 issue

*CEA, DEN, Centre de Cadarache, F-13108, Saint-Paul-lez-Durance, France
†CEA, DEN, Centre de Grenoble, F-38054, Grenoble, France
‡EDF R&D, 6 Quai Watier, F-78401, Chatou, France

Corresponding author: B. Iooss ; Email: bertrand.iooss@edf.fr
Phone: +33-1-30877969 ; Fax: +33-1-30878213

Abstract – To perform uncertainty, sensitivity or optimization analysis on scalar variables calculated by a cpu time expensive computer code, a widely accepted methodology consists in first identifying the most influential uncertain inputs (by screening techniques), and then in replacing the cpu time expensive model by a cpu inexpensive mathematical function, called a metamodel. This paper extends this methodology to the functional output case, for instance when the model output variables are curves. The screening approach is based on the analysis of variance and principal component analysis of output curves. The functional metamodeling consists in a curve classification step, a dimension reduction step, then a classical metamodeling step. An industrial nuclear reactor application (dealing with uncertainties in the pressurized thermal shock analysis) illustrates all these steps.
Keywords: Functional output, Pressurized thermal shock transient, Principal component analysis, Uncertainty and sensitivity analysis

C.1 Introduction

The uncertainty analysis and sensitivity analysis (UASA) process is one of the key step for the development and the use of predictive complex computer models (de Rocquigny et al. [6], Saltelli et al. [27]). On one hand, this process aims at quantifying the impact of the input data and parameters uncertainties on the model predictions. On the other hand, it investigates how the model outputs respond to variations of the model inputs. For example in the nuclear engineering domain, it has been applied to waste storage safety studies (Helton et al. [11]), radionuclide transport modeling in the aquifer (Volkova et al. [31]), safety passive system evaluation (Marquès et al. [19]) and simulation of large break loss of coolant accident scenario (de Crécy et al. [4]).

In practice, when dealing with UASA methods, four main problems can arise:

1. Physical models can involve complex and irregular phenomena sometimes with strong interactions between physical variables. This problem is resolved by using variance-based measures (Sobol [29]; Saltelli et al. [27]);

2. Computing variance-based measures can be infeasible for cpu time consuming code. Metamodel-based techniques (Sacks et al. [25]; Fang et al. [9]; Iooss et al. [12]) solve this problem and provide a deep exploration of the model behavior. A metamodel is a mathematical emulator function, approximating a few simulation results performed with the computer code and giving acceptable predictions with a negligible cpu time cost;

3. The two preceeding tools (variance-based sensitivity analysis and metamodel's method) are applicable for low-dimensional problems while numerical models can take as inputs a large number of uncertain variables (typically several tens or hundreds). One simple solution is to apply, in a preliminary step, a screening technique which allows to rapidly indentify the main influent input variables (Saltelli et al. [27]; Kleijnen [14]);

4. Another problem of high dimensionality arises when numerical models produce functional output variables, for instance spatially or temporally dependent. Until recent years, this problem has received only little attention in the UASA framework. However, three recent works have brought some first solutions:

   - Campbell et al. [2] use a principal component analysis of output temporal curves, then compute sensitivity indices of each input on each principal component coefficient;
   - Lamboni et al. [16] develop the multivariate global sensitivity analysis method. It allows to aggregate the different sensitivity indices of the principal component coefficients in a unique index, called the generalized sensitivity index. Each generalized sensitivity index explains the influence of the corresponding input on the overall output curve variability.
   - Dealing with complex and cpu time consuming computer codes, Marrel et al. [20] propose to build a functional metamodel (based on a wavelet decomposition technique and the Gaussian process metamodel, see Bayarri et al. [1]). Then, Monte Carlo techniques are applied on the functional metamodel to obtain variance-based sensitivity indices.

In this paper, we present an overall UASA methodology, from screening to metamodeling, applicable to output curves of cpu time consuming computer models.

This methodology is motivated by an industrial application concerning the nuclear safety. Such study requires the numerical simulation of the so-called pressurized thermal shock analysis using qualified computer codes. This quantitative analysis aims at calculating the vessel failure probability of a nuclear pressurized reactor. In a general context, structural reliability aims at determining the failure probability of a system by modeling its input variables as random variables. The system can be simulated by a numerical model (often seen as a black box function), which can be written for instance $Y = f(X)$ where $Y \in \mathbb{R}$ is the monodimensional output variable, $f(\cdot)$ is the model function and $X = (X^1, \ldots, X^p) \in \mathbb{R}^p$ are the p-dimensional input variables. A reliability problem can consist in evaluating the probability that the output Y exceeds a given treshold Y_s. One problem is that, as the

model function (i.e. the numerical computer code) becomes more and more complex, cpu time of the model increases dramatically for each run. To solve this problem, authors from structural reliability domain have largely contributed to the development and promotion of advanced Monte Carlo and FORM/SORM methods (Madsen et al. [18]). These techniques are now widely used in other physical domains, from hydraulics to aerospace engineering (see some examples in de Rocquigny et al. [6]). For our purpose (scenario of a nuclear reactor pressurized thermal shock), past and recent works have proposed some advanced methods in order to compute the failure probability with a small number of computer code runs, additionally to a high level and conservative confidence interval (de Rocquigny [5], Munoz-Zuniga et al. [22]).

At present, one major challenge in such calculations is to propagate input uncertainties in thermal-hydraulic models, by using for example Monte Carlo methods. However, the four previously described difficulties turn this problem to an extremely difficult one. First, sophisticated phenomena are implemented in the computer code, rending the behaviour of the output variables particularly complex and trivial method (as linear approximation) not applicable. Second, the thermal-hydraulic models require several hours to perform one simulation in order to simulate the full transient, while several thousands simulations are required by the Monte Carlo methods. Thus, metamodels are necessary to solve this difficulty. Third, the uncertainty sources are numerous: several tens of uncertain inputs have to be taken into account and screening techniques are required. Finally, the model output variables of interest are several time-dependent curves (temperature, primary pressure, exchange coefficient) while well-known UASA are defined for scalar output variables.

In the following section, we present more deeply the industrial problem, the associated numerical model and the variables of interest. In the third section, a functional screening technique, based on variance decomposition and generalized sensitivity indices, is described and applied. In the fourth section, the functional metamodeling method is detailed. Finally, a conclusion summarizes our results and gives some perspectives for this work.

C.2 The thermal-hydraulic transient simulation

In the generic methodology of uncertainty treatment (de Rocquigny et al. [6]), the first step is devoted to the specification of the problem. In particular, the objectives of the studies, the computer code which will be used, the studied scenario, the input uncertain variables and the output variables of interest have to be defined.

C.2.1 The industrial problem

Evaluating nuclear reactor pressure vessel (NRPV) performance during transient and accidental conditions is a major issue required by the regulatory authorities for the safety demonstration of the nuclear power plant. Indeed, during the normal operation of a nuclear power plant, the NRPV walls are exposed to neutron radiation, resulting in localized embrittlement of the vessel steel and weld materials in the area of the reactor core. Therefore, the knowledge of the behaviour of the pressure vessel subjected to highly hypothetic accidental conditions, as a Pressurized Thermal Shock (PTS) transient, is one of the most important inputs in the nuclear power plant lifetime program (see Fig. C.1).

In our industrial context, the transient critical for the PTS consists of a Small Break LOCA (Loss of Coolant Accident), located at a hot leg of the studied PWR (see Fig. C.1).

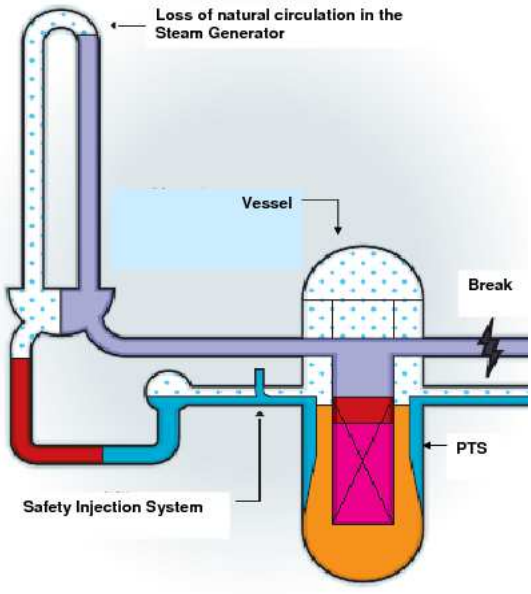

Figure C.1: Scheme of a break in the hot leg of the primary circuit leading to a Pressurized Thermal Shock (PTS). Adapted from an IRSN document.

Consecutive water loss and depressurization lead to the initiation of the safety injection system. The cold water injected by this system may enter in contact with the hot walls of the vessel and may provoke a sudden temperature decrease (thermal shock) associated with severe related thermal stresses. The temperature decrease depends on the way the injected cold water mixes with the hot fluid and on the level of heat transfer between the more or less mixed cold water and the vessel. The safety issue corresponding to the PTS is the vessel rupture. Fluid temperature close to the vessel walls and heat transfer coefficient are, with the pressure inside the vessel, the three determinant variables in the prediction of PTS consequences. They are all the three time dependent and calculated by a thermal-hydraulic code. As many of the input variables of the thermal-hydraulic code are uncertain, it will be a challenging task to evaluate the uncertainties on these three time dependent responses. In the following of the paper, we will focus on only one of the responses of interest: the fluid temperature (named T_{bas}).

C.2.2 The CATHARE2 model

CATHARE is the French best-estimate code for thermal-hydraulic nuclear reactor safety studies. It is the result of a more than thirty years joint development effort by CEA (the research institution), EDF (the utility), AREVA (the vendor) and IRSN (the safety authority). The code is able to model any kind of experimental facility or PWR (Pressurized Water Reactors, the present Western type reactors), or is usable for other reactors (already built such as RBMK reactors, or still underway such as the future reactors of Generation IV). It is also used as a plant analyzer, in a full scope training simulator providing real time calculations for the plant operators.

The code has a modular structure, which allows all the quoted above modelings. The main module is the 1-D module, but there is also 0-D and 3-D volumes and all the modules can be connected to walls or heat exchangers. Many sub-modules are available to calculate for example the neutronics, the fuel thermo-mechanics, pump characteristics, accumulators, etc.

All modules use the two-fluid model to describe steam-water flows. Four non-condensable gases may also be transported. The thermal and mechanical non-equilibrium are described, as well as all kinds of two-phase flow patterns.

A stringent process of validation is performed. First of all, quite analytical experiments are used in order to develop and qualify the physical models of the code. After that, a verification process on integral effect tests is carried out to verify the overall code performance.

C.2.3 Definition of uncertainty sources

This section describes how the list of the input variables and the quantification of their uncertainty via a probability density function have been performed. First of all, two existing lists were used as a basis. The first list is the one established by CEA for the BEMUSE benchmark (de Crécy et al. [4]), devoted to the Large Break LOCA. The second list is aimed at defining the work program of determination by CEA of the uncertainties of the physical models of CATHARE, both for Small and Large Break LOCA. In both lists, the output variable of interest is the maximum clad temperature. The PTS is a transient which has some specificities, of course with respect to the Large Break LOCA, but also to the already studied Small Break LOCA, owing to the size and the position of the break. In addition to that, the output variables of interest in a PTS are related to the conditions of pressure, liquid temperature, flow rate and level in the cold leg and not to the maximum clad temperature. Consequently, a rather detailed analysis of the PTS scenario has also been performed by experts and has led to the definition of $p = 31$ input variables, which are of two types.

The first type of variables is those related to the system behavior of the reactor. They are:

- The initial conditions (e.g. secondary circuit pressure);
- The boundary conditions (e.g. the residual power);
- The flow rate at the break (all the parameters involved in the critical flow rate prediction, e.g. liquid-interface heat transfer by flashing);
- The conditions upstream from the break (e.g. interfacial friction in stratified flow in the hot leg);
- Interfacial friction (e.g. in the core);
- Bypass phenomenon in the vessel upper head.

The second type of variables is those related to the safety injections and accumulators. They are:

- The temperature and the flow rate of the injections;
- The features of the accumulators (e.g. initial pressure);
- The condensation phenomena (e.g. liquid-interface heat transfer in stratified flow downstream from the injections);
- The thermal stratification in the cold leg.

The quantification of the uncertainty of the variables is performed for the physical models either by using a statistical method of data analysis, the name of which is CIRCE (giving normal or log-normal distribution types), or by expert judgment (giving also normal or log-normal distributions). For the initial and boundary conditions, reactor data are generally used, giving uniform or normal laws. CIRCE (de Crécy [3]) can be used only if there are experiments with a sufficient numbers of measures, which is the case of a rather large number of variables of both existing lists: BEMUSE and CATHARE. By lack of time, CIRCE was not used for the "new" parameters, specific to the PTS. A CIRCE study is nevertheless underway for two parameters related to the condensation effects resulting from the injection of cold water in the cold leg.

At the end of this step, 31 random input variables have been defined with their probability density function. The following step consists in performing a sensitivity analysis in order to retain only the most influent inputs on the output variables of interest.

C.3 Screening with generalized sensitivity indices

In order to reduce the initial list of 31 variables for the construction of metamodels and the propagation of uncertainties, it is necessary to implement a method that gives the sensitivity of the input variables on the responses (i.e. the output variables of interest) obtained with the code CATHARE2. For a scalar response, this approach is called the screening and numerous methods can be applied (Saltelli et al. [27], Dean & Lewis [7]). In this paper, the model response is a time curve and these classical approaches are not applicable.

A first sensitivity analysis, using a One-At-a-Time (OAT) method has provided a first idea of the influence of the input variables. OAT is a coarse sensitivity analysis process where each input value is moved once (for example from a minimal value to a maximal value), and its effect on the output value is analyzed. In our case, $p + 1 = 32$ simulation runs have been performed; then a subjective visual analysis between the 32 output curves has allowed to detect 14 influent inputs. For several reasons, OAT is judged as a bad practice for sensitivity analysis (see Annoni & Saltelli [26]), but engineers still use this approach in order to verify their model. In the following, we use a more rigorous method for the sensitivity analysis of inputs on an output curve.

C.3.1 Generalized sensitivity indices method

For a discrete time response, a sensitivity analysis could be performed separately at each calculation time step. Indeed in this case, the studied output is a scalar and the classical global sensitivity techniques are applicable. However, this technique has the disadvantage of introducing a high level of redundancy because of the strong relationship between responses from one time step to the next one. It may also miss important dynamic features of the response.

Recently, Lamboni et al. [15; 16] have developed the method of generalized sensitivity indices (GSI), which summarizes directly the effect of each input variable on the time series output of a computer code. This method can be divided in four steps:

1. n simulations are carried out with the computer code (CATHARE2 in our case) by varying the p input variables $(X^1, ..., X^p)$ randomly or according to a factorial experimental design. This gives n time responses. The set of outputs can be represented in

the form of a $n \times T$ matrix, where T is the number of time sampling points (in our case $T = 512$):

$$\mathbb{Y} = \begin{pmatrix} Y_1^1 & \cdots & Y_1^T \\ \vdots & \ddots & \vdots \\ Y_n^1 & \cdots & Y_n^T \end{pmatrix} . \tag{C.1}$$

Each column Y^t in $\mathbb{Y}$ represents the simulated values of the output variable at a given time, while each row of $\mathbb{Y}$ is a transient simulation obtained for a given set of values of the input variables.

2. The principal components analysis (PCA) is used to decompose the whole variability (or total inertia) in $\mathbb{Y}$. The total inertia is defined as $I(\mathbb{Y}_c) = \text{trace}({}^t\mathbb{Y}_c\mathbb{Y}_c)$, where $\mathbb{Y}_c$ is the matrix $\mathbb{Y}$ with each column centered around its mean. Thus ${}^t\mathbb{Y}_c\mathbb{Y}_c$ is the variance-covariance matrix of the columns of $\mathbb{Y}$. The PCA decomposition is based on the eigenvalues and eigenvectors of ${}^t\mathbb{Y}_c\mathbb{Y}_c$. Let $\lambda_1, \ldots, \lambda_T$ denote the obtained eigenvalues in decreasing order and let $\mathbb{L}$ denotes the $T \times T$ matrix of eigenvectors, where each column I_k is the eigenvector associated to λ_k. The $N \times T$ matrix $\mathbb{H}$ of principal components scores is obtained by: $\mathbb{H} = \mathbb{Y}_c\mathbb{L}$. The columns of $\mathbb{H}$, h_k for $k = 1, 2, \ldots, T$, are mutually orthogonal linear combinations of the $\mathbb{Y}_c$ columns. By construction, $\mathbb{H}$ has the same total inertia as $\mathbb{Y}_c$, but its inertia is mainly concentrated in the first principal components scores.

3. The third step in the method involves performing an ANOVA (ANalysis Of Variance) type decomposition of each principal components h_k:

$$SS(h_k) = SS_{1,k} + \ldots + SS_{i,k} + \ldots + SS_{p,k} + SS_{12,k} + \ldots + SS_{ij,k} + \ldots + SS_{p-1p,k} + \ldots + SS_{1..p,k} \tag{C.2}$$

where SS represents the sum of square of the effects. $SS_{i,k}$ is the main effect of input variable i on the principal component k, $SS_{ij,k}$ is the effect of the interaction between the input variables i and j on the principal component k, etc. Note that this decomposition is always possible if the factorial design is orthogonal and is unique if the factorial design is complete (Montgomery [21]). The sensitivity indices SI on the principal component h_k are defined by:

$$SI_W(h_k) = \frac{SS_{W,k}(h_k)}{SS(h_k)} \tag{C.3}$$

where $W = 1, \ldots, p$ for the first order sensitivity indices (main effects), $W = \{1, 2\}, \ldots, \{p-1, p\}$ for the second order effects, etc.

4. Finally the GSI are calculated by summing the $SI(h_k)$ of the principal components weighted by the inertia I_k associated with the component h_k:

$$GSI_W = \sum_{k=1}^{T} \frac{I_k}{I} SI_W(h_k) . \tag{C.4}$$

These GSI measure the contribution of the terms W (of first order, second order, etc.) to the total inertia of the time responses. For practical purposes, only the first

T_0 principal components ($T_0 \ll T$) are kept such that $\sum_{k=1}^{T_0} I_k = \frac{x}{100} I$ with x a given percentage (99% for example).

The dynamic coefficient of determination R_t^2 evaluates the quality of the approximations (truncations of the principal components and of the ANOVA decomposition) directly on the original time series (matrix $\mathbb{Y}$):

$$R_t^2 = \frac{\sum_{i=1}^{n} \left(\tilde{y}_i^t - \bar{y}^t\right)^2}{\sum_{i=1}^{n} \left(y_i^t - \bar{y}^t\right)^2} \quad \text{for } t = 1, \ldots, T \ , \tag{C.5}$$

where $\tilde{y}^t$ are the columns of the approximated matrix of outputs $\tilde{\mathbb{Y}}_c$ and $\bar{y}^t = \sum_{i=1}^{n} y_i^t$. A value of R_t^2 close to 1 indicates that most of the inertia of $\mathbb{Y}$ was captured while a low R_t^2 means that the obtained GSI should be interpreted with caution.

The complete decomposition (C.2) is possible only in the case of a full factorial design (untractable in our case because a full factorial design at 2 levels and 31 variables would require $2^{31} \simeq 215 \times 10^6$ simulations). As a consequence, we use a fractional factorial design (Montgomery [21]), which enables only studying a limited number of effects. We focus our study to a two levels fractional factorial design of resolution IV, which can estimate without bias the first order effects of $p = 31$ input variables with $n = 2^{p-q} = 2^6 = 64$ simulations ($q = 25$). The generated experimental design selects 64 points among the 2^{31} possible points of the two levels full factorial design.

C.3.2 Results for the response T_{bas}

A two levels (corresponding to the selected minimal and maximal values of the variables) experimental design, and then with only 64 points, was generated and the corresponding calculations performed with the CATHARE2 code. The results concern the response of interest T_{bas}, the temperature at the bottom of the cold leg as a function of time. The 64 times series are shown in the figure C.2. 95% of the inertia of these time series are captured by the nine first principal components (the three first principal components capture 82% of the inertia).

Figure C.3 shows at the top the correlation coefficients between the 3 first principal components (PC) and the outputs Y_t for $t = 1, \ldots, T$ (with $T = 512$) and at the bottom the sensitivity indices SI_W (only the six most important are shown) of the 3 first PC. We notice that the first PC is positively correlated with the output except at the beginning of the transient, while the second has a better correlation at the beginning of the transient. The first two PC are more sensitive to the variable 5 (residual power) and the third one to the variable 31 (stratification rate).

Figure C.4 shows the dynamic coefficient of determination versus time. The values of R_t^2, between 0.6 and 1, accredit a good confidence to the GSI estimates. This confidence is better in the first part of the transient where R_t^2 are highest. This result is interesting because with regards to the thermal shock impact, we are more interested by the temperatures in the first part of the transient. We can therefore conclude that the GSI estimates will be good indicators of the sensitivity of input variables to the thermal shock.

Table C.1 shows the GSI of the variables, listed in decreasing order. The sum of the indices (75%) represents the percentage of inertia (ie variability of the response) explained by the main effects. If we accept the variables having a GSI greater than or equal to 1%,

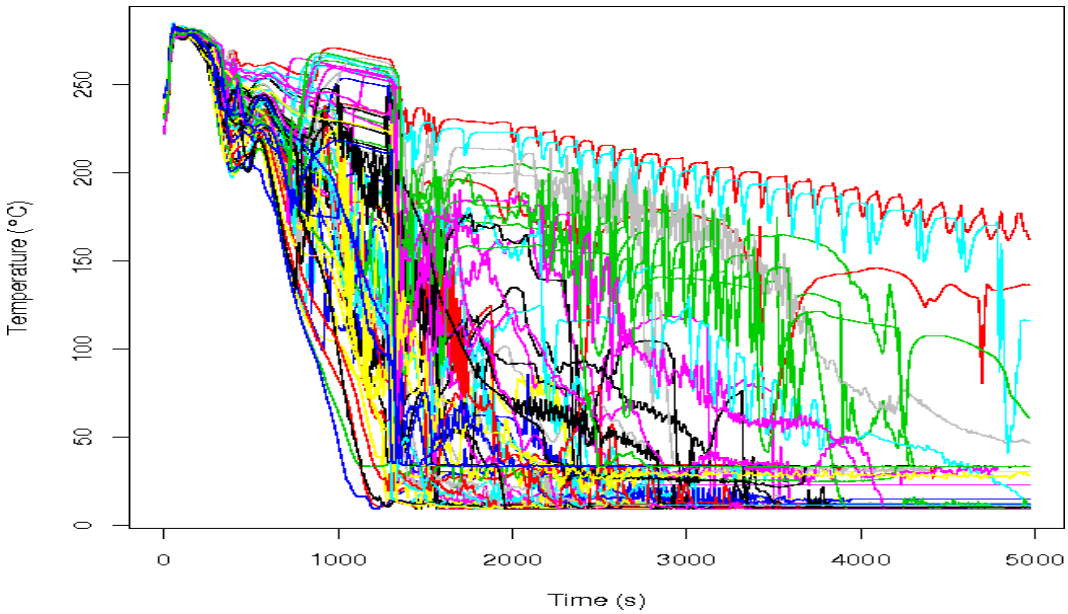

Figure C.2: 64 simulations of T_{bas} obtained from the results of the CATHARE2 code.

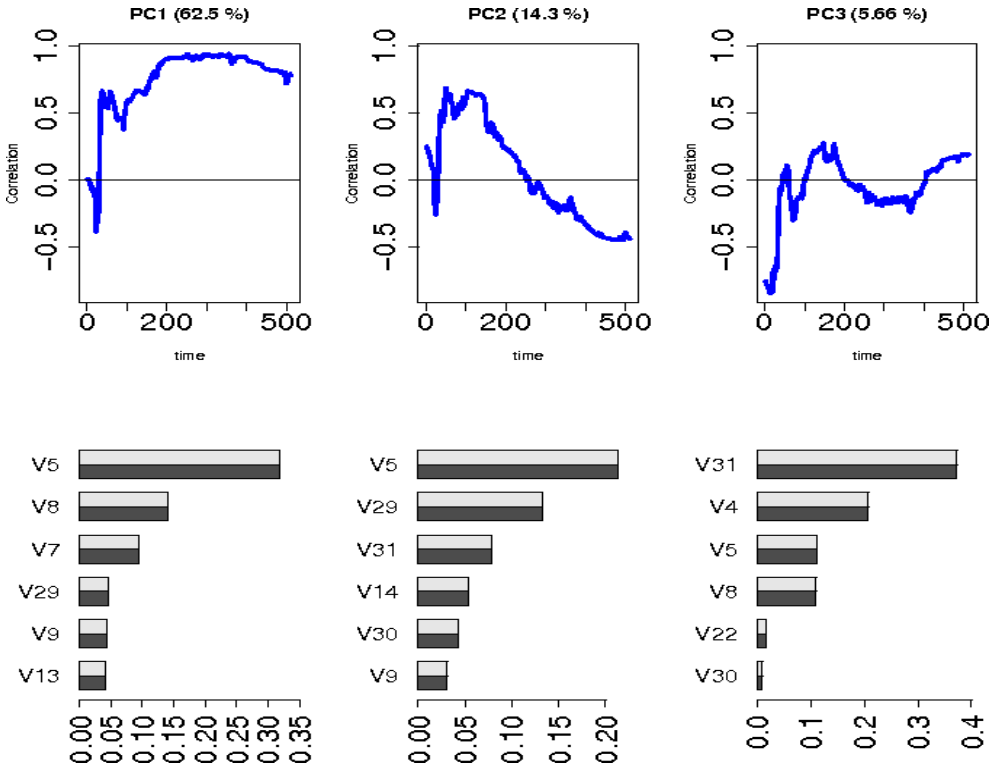

Figure C.3: Up: correlation coefficients between the 3 first principal components (PC) and the outputs Y_i; bottom: 6 larger sensitivity indices SI_W of the 3 first PC.

we obtain the 12 variables listed in Table C.1. When we compare these variables to the 13 variables identified as important in the OAT analysis (see beginning of section C.3), we find six variables in common: residual power ($V5$), steam generator emergency feedwater supply temperature ($V6$), injection system temperature ($V7$) and injection system flowrate ($V8$), accumulators temperature ($V9$) and nitrogen fraction in accumulators ($V13$). The other variables were not included in the OAT list: liquid-interface exchange in stratified flow ($V29$), rate of stratification ($V31$), singular pressure drop in dome bypass ($V14$), secondary

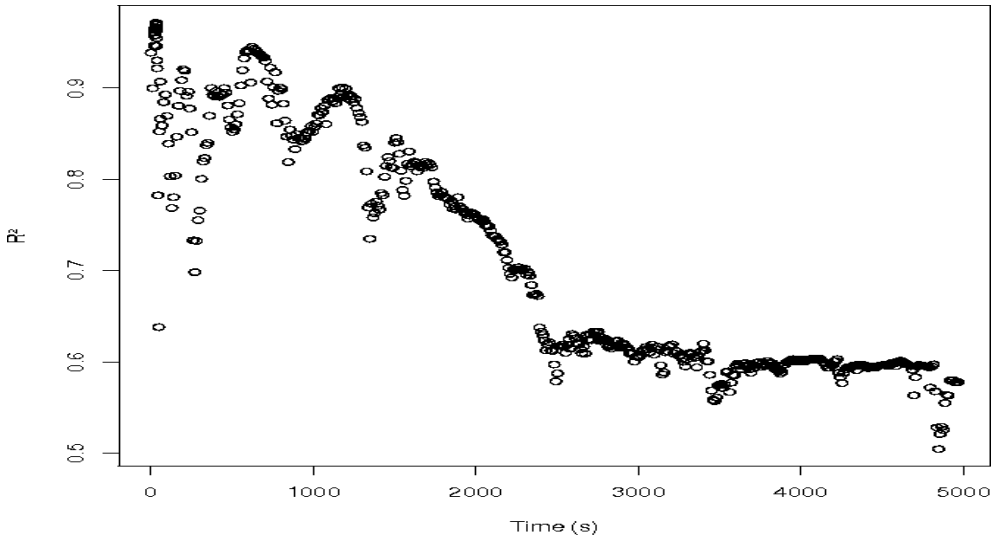

Figure C.4: Dynamic coefficient of determination R_i^2 versus time.

circuit pressure ($V4$), liquid-interface exchange for all flow types ($V30$) and accumulators initial pressure ($V11$). It shows that applying OAT can induce errors on the sensitivity analysis process and that the GSI approach can produce useful results.

Table C.1: First GSI of the inputs (decreasing order).

Input number	Input definition	$GSI(\%)$
$V5$	Residual power	25.45
$V8$	Injection System flowrate	10.29
$V7$	Injection System temperature	6.71
$V29$	Liquid-interface exchange in stratified flow	6.03
$V31$	Rate of stratification	3.97
$V9$	Accumulators temperature	3.35
$V13$	Nitrogen fraction in accumulators	3.27
$V6$	Steam generator emergency feedwater supply temperature	2.59
$V14$	Singular pressure drop in dome bypass	2.51
$V4$	Secondary circuit pressure	2.10
$V30$	Liquid-interface exchange for all flow types	1.74
$V11$	Initial pressure of accumulators	1.03

In the following, we supress from this list of 12 inputs one additional input (the nitrogen fraction in accumulators) and retain only 11 input variables. This is due to a posteriori modeling considerations.

C.4 The functional metamodeling

The section C.3 has introduced the GSI which accuracy depends on the number of simulations. Using a small number of simulations (64), this screening step has allowed us to find a restricted number of influent inputs on the output curves. In this section, a methodology is proposed in order to build a metamodel of the computer code. Not restricted by the cpu time, this metamodel will be able to be run as many times as necessary. It will serve us for a quantitative sensitivity analysis and subsequent reliability studies. Starting from n

input-output couples $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^p \times \mathbb{R}^T$, we first cluster the inputs-outputs into K groups, then reduce the dimension T to $d \ll T$ (for the reasons given in the paragraph 3.1), learn the relation between x_i and reduced representations (using classical metamodeling techniques), and finally determine a reconstruction function from $\mathbb{R}^d$ to $\mathbb{R}^T$.

The clustering step allows to distinguish several physical behaviors, which can be modeled in quite different ways. Our main ambition is to handle functional clusters of arbitrary shapes, and then reduce the dimension non-linearly to describe (almost) optimally any manifold. Ascendant hierarchical clustering (with Ward linkage) will be the algorithm of choice, because it can detect classes of arbitrary shapes and is well suited to estimate the number of clusters. This last task is done by estimating prediction accuracy using subsampling, like in Roth et al. [24]. We use the L_2 distance between curves to embed them into a k-nearest-neighbors graph, and then estimate the commute-time euclidian distances in this graph (Yen et al. [32]). This preprocessing step eases the clustering process by increasing long distances and decreasing short ones. The figure C.5 shows an example of two well separated clusters produced by the algorithm.

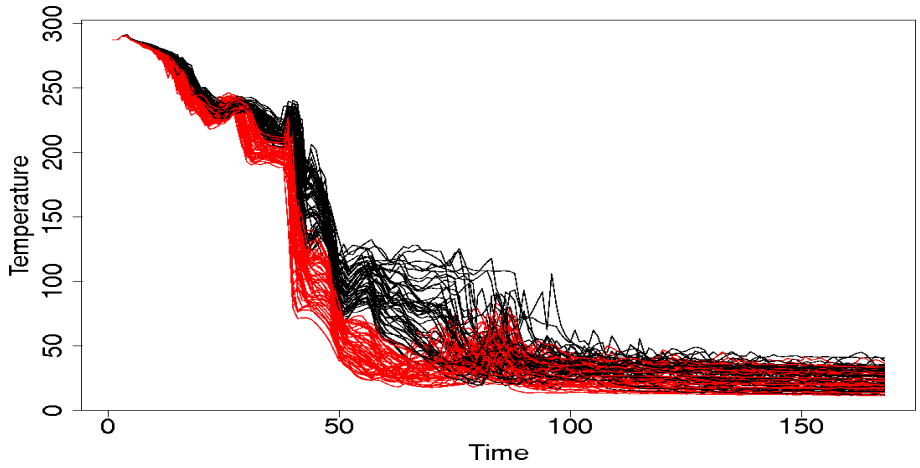

Figure C.5: Temperature versus time (CATHARE2 code), two clusters.

C.4.1 Methods for dimensionality reduction

Assume now that a group of $m < n$ inputs-outputs (X_i, Y_i) has to be analyzed for dimension reduction (the indexation goes from 1 to m, but represents a subset of $\{1, \ldots, n\}$). The classical method for the dimension reduction is a (functional) principal components analysis (Ramsay & Silverman [23]), which is perfectly suited for dimensionality reduction of linear subspaces embedded into $\mathbb{R}^T$. However, in some cases this method is inappropriate; for example a simple sphere in $\mathbb{R}^3$ cannot be represented in two dimensions using principal components. Functions in $\mathbb{R}^T$ could have an even more complex structure ; that is why we choose to use a manifold learning technique, namely Riemannian Manifold Learning (RML) (Lin et al. [17]). This method aims at approximating Riemannian normal coordinates (see for example do Carmo [8]). It performs the best in our practical tests versus other nonlinear techniques.

The RML consists at the beginning in building a graph representing the data; all the further computations are done inside this graph. It is basically a k-nearest-neighbors graph, in which we cut "invisible" edges as well as "shortcut" edges. The edges being weighted by the L_2 distances between curves, all shortest-paths are then computed. An origin curve Y_0

is determined by minimizing the sum of shortest paths from it. Now, the first step to obtain reduced coordinates consists in projecting Y_0-neighbor data to the (estimated) tangent plane at Y_0. This sums up to a least squares problem, after which coordinates are normalized to verify $\|Z - Z_0\| = \|Y - Y_0\|$, where Y is a neighbor of Y_0, with respective d-dimensional representations Z and Z_0. The figure C.6 illustrates this local step.

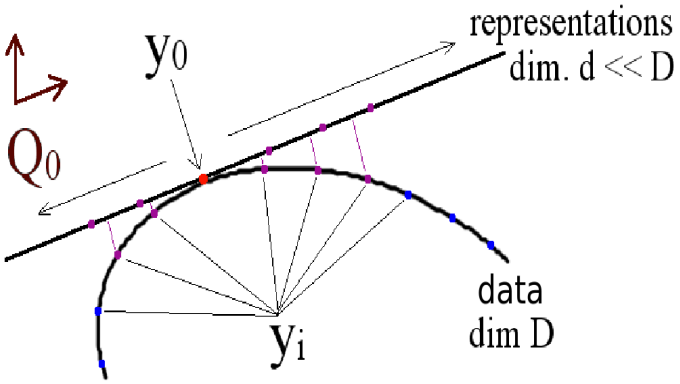

Figure C.6: Local step in the RML algorithm.

If Y is too far from Y_0 for the tangent plane approximation to hold, the basic idea is to preserve local angles with the neighbors of a predecessor on a shortest path from Y_0 to Y. If we write Y_r for a predecessor of Y, and $Y_r^{(1)}, \ldots, Y_r^{(q)}$ its neighbors with known reduced coordinates $Z_r^{(i)}$, we have to ensure that $\widehat{ZZ_rZ_r^{(i)}} \simeq \widehat{YY_rY_r^{(i)}}, i = 1, \ldots, q$ under the distance constraint $\|Z - Z_r\| = \|Y - Y_r\|$. This can be tranformed into a least-squares problem with quadratic constraints, leading to an equation solved numerically. The figure C.7 illustrates this process.

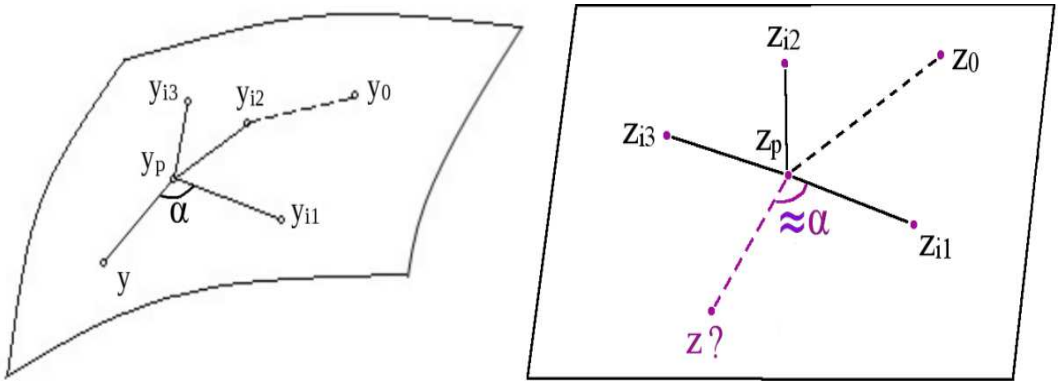

Figure C.7: Global step in the RML algorithm.

The reconstruction function is then easily done. We start from the training sample of functional outputs $Y_i \in \mathbb{R}^T$ and corresponding embeddings $Z_i \in \mathbb{R}^d$. Let us write $Z_{i_1}, \ldots, Z_{i_k}$ the k nearest neighbors of Z in the d-dimensional space. First, a local functional PCA is achieved around $Y_{i_1}, \ldots, Y_{i_k}$ to replace the curves by their d-dimensional principal components scores Y'_{i_j}. Then we solve the problem stated above with reversed roles, before re-expanding reduced coordinates Y' onto the orthonormal basis of (functional) principal components.

C.4.2 Application to the PTS

A special low discrepancy design of size $n = 600$ with the 11 influent input variables has been built. This design is an optimized Latin Hypercube Sample with a minimal wrap-around discrepancy, which has been proved to be especially efficient for the metamodel fitting process (Iooss et al. [13]). Therefore, 600 simulation runs have been performed in order to obtain the 600 output curves. Each curve is sampled on $T = 414$ points (see Fig. C.8 for an example of 100 curves). We compare three kind of metamodels: functional PCA, (functional) RML and a direct Nadaraya-Watson estimate which does not use any dimensionality reduction (see Tsybakov [30] for more information). We use the Projection Pursuit Regression method (see for example Friedman & Stuetzle [10]) as a metamodel to learn the reduced coefficients.

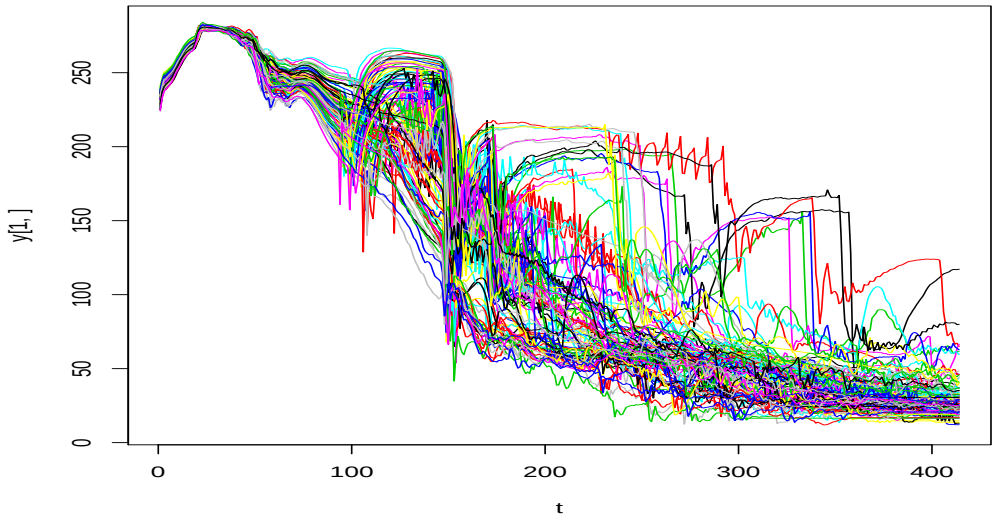

Figure C.8: 100 curves amonst the 600 output curves.

Figure C.9 shows the mean square error (MSE) and the predictivity coefficient Q_2 (coefficient of determination computed on test samples) pointwise errors. These measures have been computed using a 10-fold cross-validation technique (dividing the 600 simulation sets in 10 simulation sets). A Q_2 small or negative corresponds to a poor model, while a Q_2 close to 1 corresponds to a good model. The small Q_2 values for RML and PCA at the beginning of the transient (time index smaller than 100) are non significative because they are due to the small variability of the curves, turning the output variance to be close to 0 (which is the denominator in the Q_2 formula). The corresponding small values of the MSE confirm this behaviour. For Nadaraya-Watson curve, this effect is not present because this estimator computes a local mean ans the MSE at the beginning of the curves are extremely small. For the other parts of the curve (time index larger than 100), the Nadaraya-Watson estimate is clearly insufficient compared to the PCA and RML estimates. Globally, the PCA method performs best: the predictivity coefficients are larger than 70% (for time index larger than 100) which is clearly a good result with regard to the complexity and chaotic aspect of the CATHARE2 output curves.

To visualize more accurately the performances of the different methods, five metamodel-based predicted curves are given in Figure C.10. Several curves are better approximated by the RML method, but several others are really badly predicted by RML comparing to the PCA predictions. Indeed, RML reproduces the form of the curves irregularites, but these

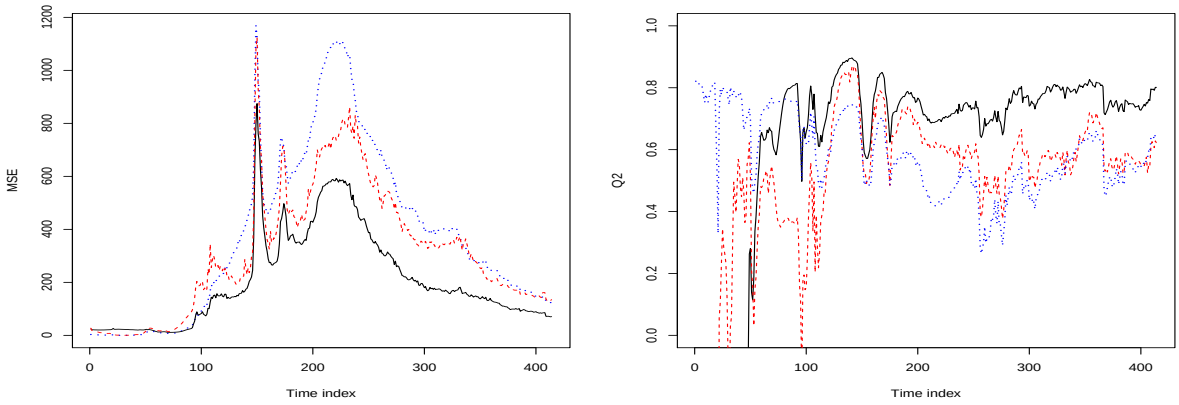

Figure C.9: Mean square error MSE (left) and predictivity coefficient Q_2 (right) error curves (solid line: PCA, dashed line: RML, dotted line: Nadaraya-Watson).

irregularities are sometimes misplaced. We have seen previously that PCA gives better results than RML in terms of MSE and Q_2 criteria. In fact, these L_2 criteria are not the most adapted ones to measure the adequacy of the predicted curves when we are interested by reproducing the irregularities and the jumps of the curves. As a conclusion, linear (PCA) and nonlinear (RML) techniques seem complementary. We are still working on improvements of the RML technique in order to improve its L_2 performance.

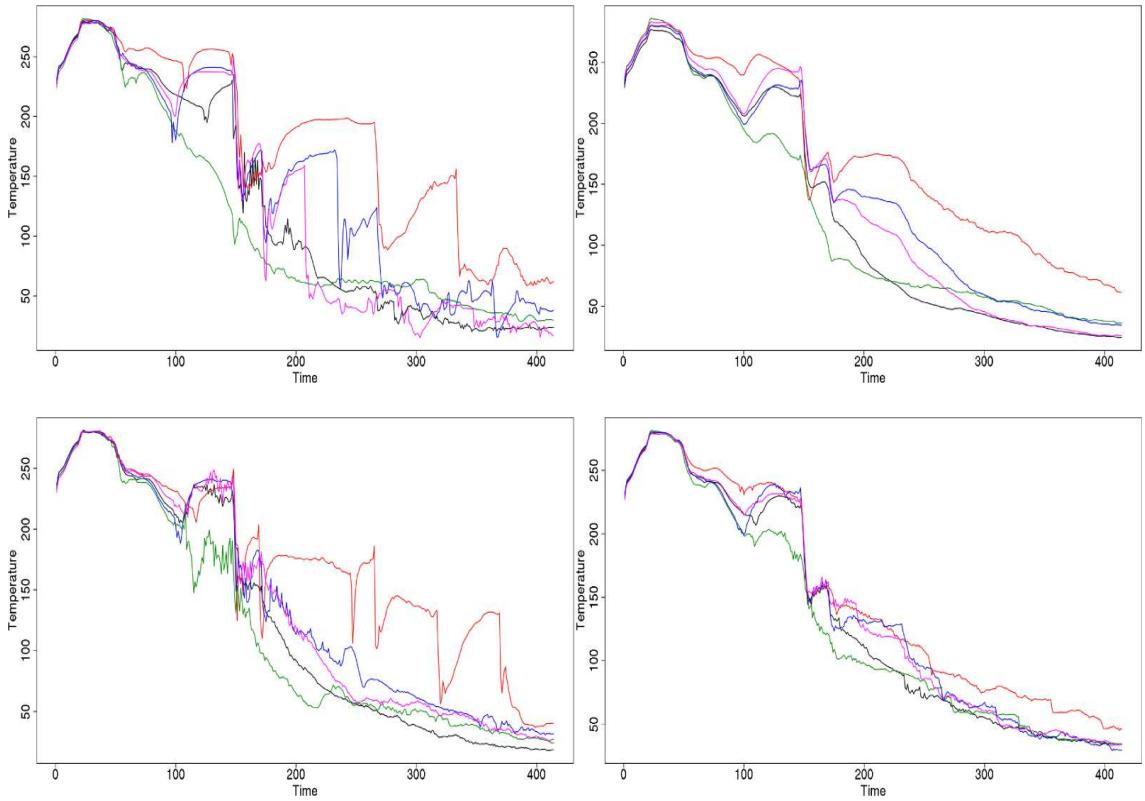

Figure C.10: Up left: 5 original (CATHARE2 simulation) curves. Up right: corresponding predicted curves with functional PCA. Bottom left: corresponding predicted curves with RML. Bottom right: corresponding predicted curves with Nadaraya-Watson method.

C.5 Conclusion

In order to improve the UASA of the PTS analysis using probabilistic methods, a methodology of metamodel fitting on the output curves of a thermal-hydraulic computer code has been proposed in this paper. Compared to other works on this subject which deals with scalar outputs (de Rocquigny et al. [6], Saltelli et al. [28]), considering the whole output curve renders the UASA more difficult. We have shown that the GSI approach of Lamboni et al [15] is particularly adapted to a screening process by using a fractional factorial design technique. For the metamodeling process, due to the extremely complex behaviour of the thermal-hydraulic output curves, a new methodology has been developed in several steps: curves clustering, dimensionality reduction, metamodel fitting and curves reconstruction. Even if the functional PCA and RML methods of dimensionality reduction have provided complementary and promising results, they need further improvements to precisely capture the irregularities of the output curves. One first possibility would be to mix these approaches in order to keep the best properties of each one.

The final goal of a functional metamodel development is to be able to take into account the thermal-hydraulic uncertainties when performing a reliability analysis of the PTS scenario. Indeed, such analysis needs several thousands of model runs in order to compute the vessel failure probability subject to this scenario. Using a best-estimate thermal-hydraulic computer code as CATHARE2 (required by such safety analyses), this large number of runs remains intractable. This paper puts a first shoulder to the wheel by proposing to replace the thermal-hydraulic computer code by a metamodel. Of course, the effects of the metamodel approximation will have to be estimated with precision. This will be done in future works.

Finally, our methodology can be applied to many other industrial fields and UASA applied problems, as in automobile, aerospace engineering, environmental sciences, etc., where the computer codes are used for prediction of temporal phenomena.

C.6 Acknowledgments

Part of this work has been backed by French National Research Agency (ANR) through COSINUS program (project COSTA BRAVA n^{o}ANR-09-COSI-015). The application concerning the PTS has been performed within the framework of a collaboration supported by CEA/DISN, EDF and AREVA-NP. We thank Gérard Biau for helful discussions. All the statistical parts of this work have been performed within the R environment. We are grateful to Hervé Monod and Matieyendou Lamboni who have provided the "planor" and"multisensi" R packages to compute respectively the fractional factorial design and the generalized sensitivity indices. We have also used the "modelcf" R package (author: B. Auder) which contains functional metamodel functions.

References

[1] M.J. Bayarri, J.O. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R.J. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh. Computer model validation with functional output. *The Annals of Statistics*, 35:1874–1906, 2007.

[2] K. Campbell, M.D. McKay, and B.J. Williams. Sensitivity analysis when model ouputs are functions. *Reliability Engineering and System Safety*, 91:1468–1472, 2006.

[3] A. de Crécy. Determination of the uncertainties of the constitutive relationships of the CATHARE 2 code. In *M&C 2001*, Salt Lake City, Utah, USA, september 2001.

[4] A. de Crécy, P. Bazin, H. Glaeser, T. Skorek, J. Joucla, P. Probst, K. Fujioka, B.D. Chung, D.Y. Oh, M. Kyncl, R. Pernica, J. Macek, R. Meca, R. Macian, F. D'Auria, A. Petruzzi, L. Batet, M. Perez, and F. Reventos. Uncertainty and sensitivity analysis of the LOFT L2-5 test: Results of the BEMUSE programme. *Nuclear Engineering and Design*, 12:3561–3578, 2008.

[5] E. de Rocquigny. La maîtrise des incertitudes dans un contexte industriel - 1ère partie : une approche méthodologique globale basée sur des exemples. *Journal de la Société Française de Statistique*, 147(3):33–71, 2006.

[6] E. de Rocquigny, N. Devictor, and S. Tarantola, editors. *Uncertainty in industrial practice*. Wiley, 2008.

[7] A. Dean and S. Lewis, editors. *Screening - Methods for experimentation in industry, drug discovery and genetics*. Springer, 2006.

[8] M. P. do Carmo. *Riemannian Geometry*. Birkhauser, 1992.

[9] K-T. Fang, R. Li, and A. Sudjianto. *Design and modeling for computer experiments*. Chapman & Hall/CRC, 2006.

[10] J. H. Friedman and W. Stuetzle. Projection Pursuit Regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[11] J.C. Helton, J.D. Johnson, C.J. Salaberry, and C.B. Storlie. Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering and System Safety*, 91:1175–1209, 2006.

[12] B. Iooss, F. Van Dorpe, and N. Devictor. Response surfaces and sensitivity analyses for an environmental model of dose calculations. *Reliability Engineering and System Safety*, 91:1241–1251, 2006.

[13] B. Iooss, L. Boussouf, V. Feuillard, and A. Marrel. Numerical studies of the metamodel fitting and validation processes. *International Journal of Advances in Systems and Measurements*, 3:11–21, 2010.

[14] J.P.C. Kleijnen. *Design and analysis of simulation experiments*. Springer, 2008.

[15] M. Lamboni, D. Makowski, S. Lehuger, B. Gabrielle, and H. Monod. Multivariate global sensitivity analysis for dynamic crop models. *Fields Crop Research*, 113:312–320, 2009.

[16] M. Lamboni, H. Monod, and D. Makowski. Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models. *Reliability Engineering and System Safety*, submitted, 2010.

[17] T. Lin, H. Zha, and S. U. Lee. Riemannian manifold learning for nonlinear dimensionality reduction. In *9th European Conference on Computer Vision*, pages 44–55, Graz, Austria, May 2006. Springer.

[18] H.O. Madsen, S. Krenk, and N.C. Lind, editors. *Methods of structural safety*. Prentice Hall, 1986.

[19] M. Marquès, J.F. Pignatel, P. Saignes, F. D'Auria, L. Burgazzi, C. Müller, R. Bolado-Lavin, C. Kirchsteiger, V. La Lumia, and I. Ivanov. Methodology for the reliability evaluation of a passive system and its integration into a probabilistic safety assesment. *Nuclear Engineering and Design*, 235:2612–2631, 2005.

[20] A. Marrel, B. Iooss, M. Jullien, B. Laurent, and E. Volkova. Global sensitivity analysis for models with spatially dependent outputs. *Environmetrics*, to appear, 2010.

[21] D.C. Montgomery. *Design and analysis of experiments*. John Wiley & Sons, 6th edition, 2004.

[22] M. Munoz-Zuniga, J. Garnier, E. Remy, and E. de Rocquigny. Adaptive directional stratification for controlled estimation of the probability of a rare event. *Reliability Engineering and System Safety*, submitted, 2010.

[23] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, 2005.

[24] V. Roth, T. Lange, M. Braun, and J. Buhmann. A resampling approach to cluster validation. In *International Conference on Computational Statistics*, volume 15, pages 123–128, Berlin, Germany, 2002. Springer.

[25] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–435, 1989.

[26] A. Saltelli and P. Annoni. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling and Software*, 25:1508–1517, 2010.

[27] A. Saltelli, K. Chan, and E.M. Scott, editors. *Sensitivity analysis*. Wiley Series in Probability and Statistics. Wiley, 2000.

[28] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Salsana, and S. Tarantola. *Global sensitivity analysis - The primer*. Wiley, 2008.

[29] I.M. Sobol. Sensitivity estimates for non linear mathematical models. *Mathematical Modelling and Computational Experiments*, 1:407–414, 1993.

[30] A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009.

[31] E. Volkova, B. Iooss, and F. Van Dorpe. Global sensitivity analysis for a numerical model of radionuclide migration from the RRC "Kurchatov Institute" radwaste disposal site. *Stochastic Environmental Research and Risk Assesment*, 22:17–31, 2008.

[32] L. Yen, D. Vanvyve, F. Wouters, F. Fouss, M. Verleysen, and M. Saerens. Clustering using a random-walk based distance measure. In *Symposium on Artificial Neural Networks*, volume 13, pages 317–324, Bruges, Belgium, 2005.

Références

[1] C. Abraham, P.A. Cornillon, E. Matzner-Løber, et N. Molinari. Unsupervised Curve Clustering using B-Splines. *Scandinavian Journal of Statistics*, 30(3) :581–595, 2003.

[2] C. Abraham, G. Biau, et B. Cadre. On the kernel rule for function classification. *Annals of the Institute of Statistical Mathematics*, 58(3) :619–633, 2006.

[3] D.K. Agrafiotis. Stochastic proximity embedding. *Journal of Computational Chemistry*, 24(10) :1215–1221, 2003.

[4] R.K. Ahuja, T.L. Magnanti, et J.B. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, 1993.

[5] Y. Alber et I. Ryazantseva. *Nonlinear Ill-posed Problems of Monotone Type*. Springer, 2006.

[6] A.E. Albert. *Regression and the Moore-Penrose Pseudo-Inverse*. Academic Press, 1972.

[7] D. Aldous et J.A. Fill. *Reversible Markov Chains and Random Walks on Graphs*. Non publié, 2002. URL http://www.stat.berkeley.edu/~aldous/RWG/book.html.

[8] S. Ali et K.A. Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing – Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04)*, 70(1–3) :173–186, 2006.

[9] M.P. Allen. *Understanding Regression Analysis*, chapter Testing hypotheses in nested regression models, pages 113–117. Springer, 1997.

[10] S. Amari et S. Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6) :783–789, 1999.

[11] N. Amenta et M. Bern. Surface Reconstruction by Voronoi Filtering. *Discrete and Computational Geometry*, 22 :481–504, 1999.

[12] O. Ammor, N. Raiss, et K. Slaoui. Détermination du nombre optimal de classes présentant un fort degré de chevauchement. *Revue MODULAD*, 37 :31–42, 2007.

[13] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68(3) :337–404, 1950.

[14] B. Auder. modelcf : MODELing physical computer Codes with Functional outputs using clustering and dimensionality reduction. R package on CRAN, 2011. URL http://cran.r-project.org/web/packages/modelcf/index.html.

[15] B. Auder et A. Fischer. Curve Clustering with Minimal Distortion Basis. *Computational Statistics and Data Analysis*, In press, 2011.

[16] B. Auder, A. De Crecy, B. Iooss, et M. Marquès. Screening and metamodeling of computer experiments with functional outputs. Application to thermal-hydraulic computations. *Journal of Reliability Engineering and System Safety*, Submitted, 2011.

[17] M. Aupetit. Learning Topology with the Generative Gaussian Graph and the EM Algorithm. In *Advances in Neural Information Processing Systems*, volume 18, pages 83–90, Vancouver, BC, Canada, 2006. MIT Press.

[18] R. Avnimelech et N. Intrator. Boosting Regression Estimators. *Neural Computation*, 11(2) :491–513, 1999.

[19] F.B. Baker et L.J. Hubert. A Graph-Theoretic Approach to Goodness-of-Fit in Complete-Link Hierarchical Clustering. *Journal of the American Statistical Association*, 71(356) :870–878, 1976.

[20] M.J. Bayarri, J.O. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R.J. Parthasarathy, R. Paulo, J. Sacks, et D. Walsh. Computer Model Validation with Functional Output. *Annals of Statistics*, 35(5) :1874–1906, 2007.

[21] M. Belkin et P. Niyogi. Laplacian Eigenmaps for dimensionality reduction and data representation. *Journal of Neural Computation*, 15(6) :1373–1396, 2003.

[22] M. Belkin et P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computational System Science*, 74(8) :1289–1308, 2008.

[23] R. Bellman. *Dynamic Programming*. Princeton University Press, Dover, 1957 (PUP) ; republié en 2003 (Dover).

[24] A. Ben-Hur, A. Elisseeff, et I. Guyon. A stability based method for discovering structure in clustered data. In *7th Pacific Sympsium On Biocomputing*, pages 6–17, Lihue, Hawaii, USA, 2002. Stanford University.

[25] Y. Bengio. Convergence Properties of the K-Means Algorithms. In *Advances in Neural Information Processing Systems*, volume 9, pages 585–592, Denver, Colorado, USA, 1995. MIT Press.

[26] Y. Bengio, J-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, et M. Ouimet. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In *Advances in Neural Information Processing Systems*, volume 16, pages 177–184, Vancouver, B.C., Canada, 2004. MIT Press.

[27] A. Berglund et S. Wold. INLR, implicit non-linear latent variable regression. *Journal of Chemometrics*, 11(2) :141–156, 1997.

[28] M. Bernstein, V. de Silva, J.C. Langford, et J.B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Stanford University, Palo Alto, CA, 2000.

[29] G. Biau, F. Bunea, et M.H. Wegkamp. Functional classification in Hilbert Spaces. *IEEE Transactions on Information Theory*, 51(6) :2163–2172, 2005.

[30] G. Biau, B. Cadre, et B. Pelletier. A graph-based estimator of the number of clusters. *ESAIM : Probability and Statistics*, 11 :272–280, 2007.

[31] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[32] C.M. Bishop, M. Svensén, et C.K.I. Williams. GTM : The Generative Topographic Mapping. *Neural Computation*, 10(1) :215–234, 1998.

[33] F. Blayo et M. Verleysen. *Les réseaux de neurones artificiels*. Presses Universitaires de France, 1996.

[34] L.M. Blumenthal. *Theory and applications of distance geometry*. Chelsea Publishing Company, 1970.

[35] J-D. Boissonnat, L.J. Guibas, et S.Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. In *23rd ACM Symposium on Computational Geometry*, pages 194–203, Gyeongju, South-Korea, 2007. ACM.

[36] J.A. Bondy et U.S.R. Murty. *Graph Theory*. Springer, 2008.

[37] B. Bouchon-Meunier. *La logique floue*. Presses Universitaires de France, 1993.

[38] C. Bouveyron, G. Celeux, et S. Girard. Intrinsic Dimension Estimation by Maximum Likelihood in Probabilistic PCA. In *73rd Annual Meeting of the Institute of Mathematical Statistics*, Gothenburg, Sweden, 2010.

[39] E. Bradley et R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994.

[40] P.S. Bradley et U.M. Fayyad. Refining Initial Points for K-Means Clustering. In *15th International Conference on Machine Learning*, pages 91–99, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[41] P.S. Bradley et O.L. Mangasarian. k-Plane Clustering. *Journal of Global Optimization*, 16 :23–32, 2000.

[42] M. Brand. Charting a Manifold. In *Advances in Neural Information Processing Systems*, volume 15, pages 961–968, Vancouver, BC, Canada, 2003. MIT Press.

[43] M. Brand. A Random Walks Perspective on Maximizing Satisfaction and Profit. In *SIAM International Data Mining Conference*, Newport Beach, California, USA, 2005.

[44] L. Breiman, J. Friedman, R. Olshen, et C. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.

[45] M. Brinkmeier. A simple and fast min-cut algorithm. In *15th International Symposium on Fundamentals of Computation Theory*, pages 317–328, Lübeck, Germany, 2005. Springer.

[46] D.S. Broomhead et M. Kirby. A new approach to dimensionality reduction : theory and algorithms. *SIAM Journal of Applied Mathematics*, 60(6) :2114–2142, 2000.

[47] D.S. Broomhead, R. Jones, et G.P. King. Topological dimension and local coordinates from time series data. *Journal of Physics A : Mathematical and General*, 20(9) : L563–L569, 1987.

[48] K.S. Brown. Simplex Volumes and the Cayley-Menger Determinant, 2000. URL http://www.mathpages.com/home/kmath664/kmath664.htm.

[49] M. Brucher, C. Heinrich, F. Heitz, et J-P. Armspach. A Metric Multidimensional Scaling-Based Nonlinear Manifold Learning Approach for Unsupervised Data Reduction. *EURASIP Journal on Advances in Signal Processing*, 2008, 2008. Article ID 862015.

[50] A. Brun, C.F. Westin, M. Herberthson, et H. Knutsson. Fast Manifold Learning Based on Riemannian Normal Coordinates. In *14th Scandinavian Conference on Image Analysis*, pages 920–929, Joensuu, Finland, 2005.

[51] J. Bruske et G. Sommer. Intrinsic Dimensionality Estimation With Optimally Topology Preserving Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), 1998.

[52] F. Burba, F. Ferraty, et P. Vieu. k-Nearest Neighbour method in functional nonparametric regression. *Journal of Nonparametric Statistics*, 21(4) :453–469, 2009.

[53] F. Camastra et A. Vinciarelli. Intrinsic Dimension Estimation of Data : An Approach Based on Grassberger–Procaccia's Algorithm. *Neural Processing Letters*, 14(1) :27–34, 2001.

[54] K. Campbell, M. D. McKay, et B. J. Williams. Sensitivity analysis when model outputs are functions. *Reliability Engineering & System Safety*, 91(10–11) :1468–1472, 2006.

[55] O. Chapelle et A. Zien. Semi-supervised classification by low density separation. In *10th International Workshop on Artificial Intelligence and Statistics*, pages 57–64, Barbados, 2005.

[56] P.Y. Chebotarev et E.V. Shamis. On Proximity Measures for Graph Vertices. *Automation and Remote Control*, 59(10) :1443–1459, 1998.

[57] H. Chen. Estimation of a Projection-Pursuit type Regression Model. *Annals of Statistics*, 19(1) :142–157, 1991.

[58] H. Chen et F. Zhang. The expected hitting times for finite Markov chains. *Linear Algebra and its Applications*, 428(11–12) :2730–2749, 2008.

[59] M. Chen, J. Liu, et X. Tang. Clustering via Random Walk Hitting Time on Directed Graphs. In *23rd national conference on Artificial intelligence*, volume 2, pages 616–621, Chicago, Illinois, USA, 2008. AAAI Press.

[60] S. Chen. Multi-Output Regression Using a Locally Regularised Orthogonal Least Square Algorithm. *IEEE Transactions on Vision, Image and Signal Processing*, 149 (4) :185–195, 2002.

[61] V.C.P. Chen, K-L. Tsui, R.R. Barton, et J.K. Allen. *Handbook of Statistics : Statistics in Industry*, chapter A review of design and modeling in computer experiments, pages 231–261. Elsevier Science, 2003.

[62] S-W. Cheng et M-K. Chiu. Dimension detection via slivers. In *20th ACM-SIAM Symposium on Discrete Algorithms*, pages 1001–1010. SIAM, 2009.

[63] S-W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, et S-H. Teng. Sliver exudation. In *15th Symposium on Computational Geometry*, pages 1–13. ACM Press, 1999.

[64] S-W. Cheng, T.K. Dey, et E.A. Ramos. Manifold reconstruction from point samples. In *16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1027, Vancouver, BC, Canada, 2005. SIAM.

[65] S-S. Chern, W-H. Chen, et K.S. Lam. *Lectures on Differential Geometry*. World Scientific Publishing, 2000.

[66] J-M. Chiou et P-L. Li. Functional clustering and identifying substructures of longitudinal data. *Journal of the Royal Statistical Society Series B*, 69(4) :679–699, 2007.

[67] J-M. Chiou, H-G. Müller, et J-L. Wang. Functional quasi-likelihood regression models with smooth random effects. *Journal of the Royal Statistical Society Series B*, 65(2) : 405–423, 2003.

[68] J-M. Chiou, H-G. Müller, et J-L. Wang. Functional Response Models. *Statistica Sinica*, 14 :675–693, 2004.

[69] J.M. Chiou et H.G. Müller. Quasi-likelihood regression with unknown link and variance functions. *Journal of the American Statistical Association*, 93(444) :1376–1387, 1998.

[70] C.H. Chou, M.C. Su, et E. Lai. A New Cluster Validity Measure for Clusters with Different Densities. In *6th International Conference on Intelligent Systems & Control*, pages 276–281, Salzburg, Austria, 2003. IASTED.

[71] F.R. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[72] W.S. Cleveland. Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, 74(368) :829–836, 1979.

[73] W.S. Cleveland et S.J. Devlin. Locally Weighted Regression : An Approach to Regression Analysis by Local Fitting. *Journal of the American Statistical Association*, 83(403) :596–610, 1988.

[74] T.H. Cormen, C.E. Leiserson, R.L. Rivest, et C. Stein. *Introduction to Algorithms*. MIT Press, 2009.

[75] J.A. Costa et A.O. Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Transactions on Signal Processing*, 52(8) :2210–2221, 2004.

[76] T.F. Cox et M.A.A. Cox. *Multidimensional Scaling*. Chapman & Hall/CRC, 2000.

[77] C. Crambes, L. Delsol, et A. Laksaci. Robust nonparametric estimation for functional data. *Journal of Nonparametric Statistics*, 20(7) :573–598, 2008.

[78] N. Cristianini et J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[79] G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Department of Computer Science, Tufts University, Medford, MA, 1988.

[80] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2 :303–314, 1989.

[81] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, et M. Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4) :864–894, 1994.

[82] J. Dauxois, A. Pousse, et Y. Romain. Asymptotic theory for the principal component analysis of a random vector function : some applications to statistical inference. *Journal of Multivariate Analysis*, 12(1) :136–154, 1982.

[83] G. Davis et A. Nosratinia. Wavelet-Based Image Coding : An Overview. *Applied and Computational Control, Signals, and Circuits*, 1(1), 1998.

[84] A.C. Davison et D.V. Hinkley. *Bootstrap Methods and Their Application*. Cambridge University Press, 1997.

[85] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[86] S. de Jong. SIMPLS : An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3) :251–263, 1993.

[87] D. De Ridder et V. Franc. Robust manifold learning. Technical Report CTU-CMP-2003-08, Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, 2003.

[88] E. De Rocquigny, N. Devictor, et S. Tarantola. *Uncertainty in industrial practice : A guide to quantitative uncertainty management*. Wiley, 2008.

[89] V. de Silva et J. Tenenbaum. Global versus Local Methods in Nonlinear Dimensionality Reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 705–712, Vancouver, BC, Canada, 2003. MIT Press.

[90] L. Debnath. *Wavelet Transforms and Their Applications*. Birkhäuser Boston, 2002.

[91] F. Deheeger. *Couplage mécano-fiabiliste : SMART – méthodologie d'apprentissage stochastique en fiabilité*. Thè de l'Université Blaise Pascal – Clermont II, 2008.

[92] P. Delicado. Another Look at Principal Curves and Surfaces. *Journal of Multivariate Analysis*, 77(1) :84–116, 2001.

[93] P. Delicado et M. Huerta. Principal Curves of Oriented Points : Theoretical and computational improvements. *Computational Statistics*, 18(2) :293–315, 2003.

[94] P. Demartines et J. Hérault. Curvilinear component analysis : a self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1) :148–154, 1997.

[95] D. Demers et G. Cottrell. Non-Linear Dimensionality Reduction. In *Advances in Neural Information Processing Systems*, volume 7, pages 580–587. Morgan Kaufmann, 1993.

[96] J. Demmel et W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM - scientific and statistical computing*, 11(5) :873–912, 1990.

[97] R. Der, U. Steinmetz, G. Balzuweit, et G. Schürmann. Nonlinear principal component analysis. Technical Report 4/98, Université de Leipzig, département d'informatique, 1998.

[98] R. DeVore et V. Popov. Interpolation of Besov spaces. *Transactions of the American Mathematical Society*, 305(1) :297–314, 1988.

[99] I.S. Dhillon, Y. Guan, et B. Kulis. Kernel k-means, Spectral Clustering and Normalized Cuts. In *10th International Conference on Knowledge Discovery and Data Mining*, pages 551–556, Seattle, WA, USA, 2004. ACM SIGKDD.

[100] D.B. Dias, S.M. Peres, et H.H. Bíscaro. LIBRAS dataset, 2009. URL http://archive.ics.uci.edu/ml/datasets/Libras+Movement.

[101] T.L. Dickson et S.N.M. Malik. An updated probabilistic fracture mechanics methodology for application to pressurized thermal shock. *International Journal of Pressure Vessels and Piping*, 78(2–3) :155–163, 2001.

[102] R. Diestel. *Graph Theory*. Springer, 2010.

[103] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[104] A.J. Dobson et A. Barnett. *An Introduction to Generalized Linear Models*. Chapman & Hall/CRC, 2008.

[105] D. Dong et T.J. McAvoy. Nonlinear principal component analysis – Based on principal curves and neural networks. *Computers & Chemical Engineering*, 20(1) :65–78, 1996.

[106] D.L. Donoho et C. Grimes. Hessian eigenmaps : Locally linear embedding techniques for high-dimensional data. *PNAS*, 100(10) :5591–5596, 2003.

[107] A. Drau. *Polynômes Orthogonaux Formels - Applications*. Springer, 1983.

[108] G. Dreyfus, J-M. Martinez, M. Samuelides, M. Gordon, F. Badran, et S. Thiria. *Apprentissage statistique : réseaux de neurones, cartes topologiques, machines à vecteurs supports*. Eyrolles, 2008.

[109] S. Dudoit et J. Fridlyand. A prediction-based resampling method to estimate the number of clusters in a dataset. *Journal of Genome Biology*, 3(7) :0036.1–0036.21, 2002.

[110] J-F. Durand et R. Sabatier. Additive Splines for Partial Least Squares Regression. *Journal of the American Statistical Association*, 92(440) :1546–1554, 1997.

[111] J-P. Eckmann et D. Ruelle. Fundamental limitations for estimating dimensions and Lyapunov exponents in dynamical systems. *Physica D : Nonlinear Phenomena*, 56 (2–3) :185–187, 1992.

[112] B. Efron et R. Tibshirani. Improvements on Cross-Validation : The .632+ Bootstrap Method. *Journal of the American Statistical Association*, 92(438) :548–560, 1997.

[113] H. Elghazel, H. Kheddouci, V. Deslandres, et A. Dussauchoy. A New Graph-Based Clustering Approach : Application to PMSI Data. In *3rd International Conference on Service Systems and Service Management*, pages 110–115, Université de Technologie de Troyes, France, 2006. IEEE.

[114] J. Elith, J.R. Leathwick, et T. Hastie. A working guide to boosted regression trees. *Journal of animal ecology*, 77(4) :802–813, 2008.

[115] Y. Escoufier. Échantillonnage dans une population de variables aléatoires réelles. *Publications de l'Institut de Statistique de l'Université de Paris*, 19(4) :1–47, 1970.

[116] M. Fan, N. Gu, H. Qiao, et B. Zhang. Intrinsic dimension estimation of data by principal component analysis. *Submitted*, 2010.

[117] K-T. Fang, R. Li, et A. Sudjianto. *Design and Modeling for Computer Experiments*. Chapman & Hall/CRC, 2006.

[118] A.M. Farahmand, C. Szepesv'ari, et J-Y. Audibert. Manifold-adaptive dimension estimation. In *24th International Conference on Machine Learning*, pages 265–272, Corvalis, Oregon, 2007. ACM.

[119] J.J. Faraway. Regression analysis for a functional response. *Technometrics*, 39(3) : 254–261, 1997.

[120] F. Ferraty et P. Vieu. *Nonparametric Functional Data Analysis*. Springer, 2006.

[121] C. Fiorio et A. Mas. A sharp concentration-based adaptive segmentation algorithm. *Lecture Notes in Computer Science*, In press, 2010.

[122] B. Fischer et J.M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4) :513–518, 2003.

[123] B. Fischer et T. Züller. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, chapter Path Based Pairwise Data Clustering with Application to Texture Segmentation, pages 235–250. Springer, 2001.

[124] G.W. Flake, R.E. Tarjan, et K. Tsioutsiouliklis. Graph Clustering and Minimum Cut Trees. *Internet Mathematics*, 4(1) :385–408, 2004.

[125] I. K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, LLNL, 2002.

[126] F. Fouss, A. Pirotte, J-M. Renders, et M. Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative filtering and subspace projection of the graph nodes. Technical report, ISYS at the Université catholique de Louvain, and Xerox Research Center Europe, 2005. URL www.isys.ucl.ac.be/staff/marco/Publications/Fouss_WIC2005.pdf.

[127] D. Freedman. Efficient Simplicial Reconstructions of Manifolds from Their Samples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10) :1349–1357, 2002.

[128] M.H. Freedman. The topology of four-dimensional manifolds. *Journal of Differential Geometry*, 17(3) :357–453, 1982.

[129] J. Friedman. SMART user's guide. Technical Report LCM001, Department of Statistics, Stanford University, 1984.

[130] J.H. Friedman. Multivariate Adaptive Regression Splines. *Annals of Statistics*, 19(1) : 1–67, 1991.

[131] J.H. Friedman et W. Stuetzle. Projection Pursuit Regression. *Journal of the American Statistical Association*, 76 :817–823, 1981.

[132] K. Fukunaga et D.R. Olsen. An Algorithm for Finding Intrinsic Dimensionality of Data. *IEEE Transactions on Computers*, 20(2) :176–183, 1971.

[133] S. Gaffney. *Probabilistic Curve-Aligned Clustering and Prediction with Mixture Models*. PhD thesis, University of California, Irvine, 2004.

[134] S. Gaffney et P. Smyth. Curve Clustering with Random Effects Regression Mixtures. In *9th International Workshop on Artificial Intelligence and Statistics*, Hyatt Hotel, Key West, Florida, 2003. C. Bishop & B. Frey.

[135] S. Gaffney et P. Smyth. Trajectory clustering with mixtures of regression models. In *5th International Conference on Knowledge Discovery and Data Mining archive*, pages 63–72, San Diego, California, United States, 1999. ACM New York, NY, USA.

[136] G. Gan, C. Ma, et J. Wu. *Data Clustering : Theory, Algorithms, and Applications*. SIAM, 2007.

[137] L.A. García-Escudero et A. Gordaliza. A Proposal for Robust Curve Clustering. *Journal of Classification*, 22(2) :185–201, 2005.

[138] Y. Gdalyahu, D. Weinshall, et M. Werman. Stochastic image segmentation by typical cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 12, pages 596–601, Fort Collins, CO, USA, 1999. IEEE.

[139] R.G. Ghanem et P.D. Spanos. *Stochastic Finite Elements : A Spectral Approach*. Dover Publications, 2003.

[140] S. Girard et S. Iovleff. Auto-Associative Models and Generalized Principal Component Analysis. *Journal of Multivariate Analysis*, 93(1) :21–39, 2005.

[141] S. Girard, B. Chalmond, et J-M. Dinten. Une ACP non linéaire basée sur l'approximation par variété. *Revue de Statistique Appliquée*, 46(3) :5–19, 1998.

[142] M. Girolami. Mercer Kernel-Based Clustering in Feature Space. *IEEE Transactions on Neural Networks*, 13(3) :780–784, 2002.

[143] C.D. Giurcăneanu et I. Tăbuş. Cluster Structure Inference Based on Clustering Stability with Applications to Microarray Data Analysis. *EURASIP Journal on Applied Signal Processing*, 2004 :64–80, 2004.

[144] F. Göbel et A.A. Jagers. Random Walks on Graphs. *Stochastic Processes and their Applications*, 2(4) :311–336, 1974.

[145] R.E. Gomory et T.C. Hu. Multi-Terminal Network Flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4) :551–570, 1961.

[146] R.P. Gorman et T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1(1) :75–89, 1988.

[147] B. Govaerts et J. Noël. Analysing the Results of a Design Experiment when the Response is a Curve : Methodology and Application in Metal Injection Moulding. *Quality and Reliability Engineering International*, 21(5) :509–520, 2005.

[148] M. Gu et S.C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal svd. *SIAM Journal on Matrix Analysis and Applications*, 16(1) :79–92, 1995.

[149] M.D. Gupta et T. Huang. Regularized Maximum Likelihood for Intrinsic Dimension Estimation. In *Conference on Uncertainty in Artificial Intelligence*, volume 26, Catalina Island, California, USA, 2010.

[150] D. Gusfield. Very Simple Methods for All Pairs Network Flow Analysis. *SIAM Journal on Computing*, 19(1) :143–155, 1990.

[151] M. Halkidi et M. Vazirgiannis. A density-based cluster validity approach using multi-representatives. *Pattern Recognition Letters*, 29(6) :773–786, 2008.

[152] Q. Han et J-X. Hong. *Isometric Embedding of Riemannian Manifolds in Euclidean Spaces*. American Mathematical Society, 2006.

[153] P. Hansen et M. Delattre. Complete-Link Cluster Analysis by Graph Coloring. *Journal of the American Statistical Association*, 73(362) :397–403, 1978.

[154] D. Harel et Y. Koren. Clustering spatial data using random walks. In *7th International Conference on Knowledge Discovery and Data mining*, volume 7, pages 281–286, San Francisco, California, 2001. ACM.

[155] J.A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.

[156] T. Hastie et W. Stuetzle. Principal Curves. *Journal of the American Statistical Association*, 84(406) :502–516, 1989.

[157] T. Hastie, R. Tibshirani, et J. Friedman. *The Elements of Statistical Learning Data Mining, Inference and Prediction*. Springer, 2009.

[158] T.J. Hastie et R.J. Tibshirani. *Generalized Additive Models*. Chapman & Hall/CRC, 1990.

[159] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[160] X. He et P. Niyogi. Locality Preserving Projections. In *Advances in Neural Information Processing Systems*, volume 16, Vancouver, BC, Canada, 2004. MIT Press.

[161] X. He, D. Cai, S. Yan, et H-J. Zhang. Neighborhood Preserving Embedding. In *2nd IEEE International Conference on Computer Vision*, pages 1208–1213, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[162] N.E. Heckman et R.H. Zamar. Comparing the shapes of regression functions. *Biometrika*, 87(1) :135–144, 2000.

[163] M. Hein et J-Y. Audibert. Intrinsic Dimensionality Estimation of Submanifolds in euclidian spaces. In *22nd International conference on Machine learning*, pages 289–296, Bonn, Germany, 2005.

[164] G. Hinton et S. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840, Vancouver, BC, Canada, 2003. MIT Press.

[165] J.Z. Huang, C.O. Wu, et L. Zhou. Varying-coefficient models and basis function approximations for the analysis of repeated measurements. *Biometrika*, 89(1) :111–128, 2002.

[166] D. Hundley et M. Kirby. Estimation of Topological Dimension. In *3rd SIAM International Conference on Data Mining*, pages 194–202, San Francisco, CA, USA, 2003. SIAM.

[167] W. Hurewicz et H. Wallman. *Dimension Theory*. Princeton University Press, 1948, révisé en 1996.

[168] B. Iooss, L. Boussouf, V. Feuillard, et A. Marrel. Numerical studies of the metamodel fitting and validation processes. *International Journal of Advances in Systems and Measurements*, 3(1–2) :11–21, 2010.

[169] R. Jacobs, M.I. Jordan, S.J. Nowlan, et G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1) :79–87, 1991.

[170] G.M. James et C.A. Sugar. Clustering for sparsely sampled functional data. *Journal of the American Statistical Association*, 98(462) :397–408, 2003.

[171] E.R. Jessup et D.C. Sorensen. A Parallel Algorithm for Computing the Singular Value Decomposition of a Matrix. *SIAM Journal on Matrix Analysis and Applications*, 15 (2) :530–548, 1994.

[172] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[173] Y. Jung, H. Park, D.-Z. Du, et B.L. Drake. A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering. *Journal of Global Optimization*, 25(1) : 91–111, 2003.

[174] N. Kambhatla et T.K. Leen. Dimension Reduction by Local Principal Component Analysis. *Neural Computation*, 9(7) :1493–1516, 1997.

[175] A. Karatzoglou, A. Smola, et K. Hornik. kernlab : Kernel-based Machine Learning Lab. R package on CRAN, 2010. URL http://cran.r-project.org/web/packages/kernlab/index.html.

[176] G. Karypis, E-H. Han, et V. Kumar. CHAMELEON : A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Transactions on Computers*, 32(8) :68–75, 1999.

[177] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1) : 39–43, 1953.

[178] A. Kaufmann et R. Cruon. *Programmation dynamique*. Dunod, 1991.

[179] B. Kégl. Intrinsic dimension estimation using packing numbers. In *Advances in Neural Information Processing Systems*, volume 14, pages 681–688. MIT Press, 2002.

[180] B. Kégl, A. Krzyzak, T. Linder, et K. Zeger. Learning and Design of Principal Curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3) :281–297, 2000.

[181] J.G. Kemeny et J.L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[182] M. Kervaire. A manifold which does not admit any differentiable structure. *Commentarii Mathematici Helvetici*, 34(1) :257–270, 1960.

[183] D-J. Kim, Y-W. Park, et K-J. Park. A Novel Validity Index for Determination of the Optimal Number of Clusters. *IEICE transactions on information and systems*, 84(2) : 281–285, 2001.

[184] M. Kim et R.S. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15) :2353–2363, 2005.

[185] S. Klinke et J. Grassmann. Projection Pursuit Regression and Neural Networks. Technical report, Humboldt University of Berlin, Germany, 1998. URL http://edoc.hu-berlin.de/series/sfb-373-papers/1998-17/PDF/17.pdf.

[186] T. Kohonen. Self-organization of topologically correct feature maps. *Biological Cybernetics*, 43(1) :59–69, 1982.

[187] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

[188] T.H. Koornwinder, R.S.C. Wong, R. Koekoek, et R.F. Swarttouw. *NIST Handbook of Mathematical Functions*, chapter 18 – Orthogonal Polynomials. Cambridge University Press, 2010.

[189] M.A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *American Institute of Chemical Engineers Journal*, 37(2) :233–243, 1991.

[190] D.G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of Chemistry, Metallurgy and Mining Society of South Africa*, 52 :119–139, 1951.

[191] B. Krose et P. van der Smagt. *An Introduction to Neural Networks*. Internet, 1996. URL http://lia.univ-avignon.fr/fileadmin/documents/Users/Intranet/chercheurs/torres/li

[192] B. Krzykacz-Hausmann. Epistemic sensitivity analysis based on the concept of entropy. In *Proceedings of SAMO 2001*, pages 31–35, Madrid, Espagne, 2001. CIEMAT.

[193] S. Kullback et R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1) :79–86, 1951.

[194] K.J. Kyoung et S. Choi. Clustering with r-regular graphs. *Pattern Recognition*, 42(9) : 2020–2028, 2009.

[195] I.O. Kyrgyzov, O.O. Kyrgyzov, H. Maître, et M. Campedel. Kernel MDL to Determine the Number of Clusters. In *5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 203–217, Leipzig, Germany, 2007. Springer-Verlag.

[196] J. Lafontaine. *Introduction aux variétés différentielles*. Grenoble Sciences, 2002.

[197] M. Lamboni, H. Monod, et D. Makowski. Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models. *Reliability Engineering & System Safety*, submitted, 2010.

[198] R.M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical report, Aarhus University, department of Computer Science, 1998.

[199] M.H. Law et A.K. Jain. Incremental Nonlinear Dimensionality Reduction by Manifold Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3) : 377–391, 2006.

[200] M. LeBlanc et R. Tibshirani. Adaptive principal surfaces. *Journal of the American Statistical Association*, 89(425) :53–64, 1994.

[201] J.A. Lee et M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[202] J.A. Lee et M. Verleysen. Quality assessment of dimensionality reduction : Rank-based criteria. *Neurocomputing*, 72(3) :1431–1443, 2009.

[203] J.A. Lee, A. Lendasse, et M. Verleysen. Curvilinear distances analysis versus isomap. In *10th European Symposium on Artificial Neural Networks*, pages 185–192, Bruges, Belgium, 2002.

[204] J.A. Lee, C. Archambeau, et M. Verleysen. Locally linear embedding versus Isotop. In *11th European Symposium on Artificial Neural Networks*, pages 527–534, Bruges, Belgium, 2003. d-side.

[205] J.M. Lee. *Introduction to Smooth Manifolds*. Springer, 2002.

[206] J.M. Lee. *Riemannian Manifolds : An Introduction to Curvature*. Springer, 1997.

[207] E. Levina et P.J. Bickel. Maximum Likelihood Estimation of Intrinsic Dimension. In *Advances in Neural Information Processing Systems*, volume 17, pages 777–784, Cambridge, MA, 2005. MIT Press.

[208] E. Levine et E. Domany. Resampling Method For Unsupervised Estimation Of Cluster Validity. *Journal of Neural Computation*, 13(11) :2573–2593, 2001.

[209] K-C. Li, Y. Aragon, K. Shedden, et C. Thomas-Agnan. Dimension reduction for multivariate response data. *Journal of the American Statistical Association*, 97(461) : 99–109, 2003.

[210] R. Li, A. Sudjianto, et Z. Zhang. Modeling computer experiments with functional response. In *SAE 2005 World Congress & Exhibition*, pages 1661–1666, Detroit, MI, USA, 2005. Society of Automative Engineers.

[211] D. Liben-Nowell et J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7) :1019–1031, 2007.

[212] T. Lin et H. Zha. Riemannian Manifold Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5) :796–809, 2008.

[213] T. Lin, H. Zha, et S. Lee. Riemannian Manifold Learning for Nonlinear Dimensionality Reduction. In *9th European Conference on Computer Vision*, volume 3951, pages 44–55, Graz, Autriche, 2006. Springer.

[214] G. Liu, Z. Lin, et Y. Yu. Multi-output regression on the output manifold. *Pattern Recognition*, 42(11) :2737–2743, 2009.

[215] X. Liu et M.C.K. Yang. Simultaneous curve registration and clustering for functional data. *Computational Statistics and Data Analysis*, 53(4) :1361–1376, 2009.

[216] X. Liu, J. Yin, Z. Feng, et J. Dong. Incremental Manifold Learning Via Tangent Space Alignment. In *2nd Workshop on Artificial Neural Networks in Pattern Recognition*, volume 4087, pages 107–121, Ulm , Allemagne, 2006. Springer.

[217] S.P. Lloyd. Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28(2) :129–137, 1957, publié en 1982.

[218] P. Ma, C.I. Castillo-Davis, W. Zhong, et J.S. Liu. A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, 34(4) :1261–1269, 2006.

[219] D.J. MacKay et Z. Ghahramani. Comments on "Maximum Likelihood Estimation of Intrinsic Dimension" by E. Levina and P. Bickel (2004), 2005. URL http://www.inference.phy.cam.ac.uk/mackay/dimension/.

[220] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2008.

[221] M. Marques et B. Auder. Construction de métamodèles pour la propagation d'incertitudes dans les calculs thermo-hydrauliques de chocs thermiques pressurisés sur la cuve d'un REP de 900 MWe. Technical report, CEA/DEN/CAD/DER/SESI/LCFR 22/10/08, 2008.

[222] M. Marques et J.F. Pignatel. Propagation d'incertitudes dans les calculs thermo-hydrauliques de chocs thermiques pressurisés sur la cuve du réacteur. Technical report, CEA/DEN/CAD/DER/SESI/LCFR DR5 20/09/07, 2007.

[223] A. Marrel. *Mise en Oeuvre et Utilisation du Métamodèle Processus Gaussien pour l'Analyse de Sensibilité de Modèles Numériques*. Thèse de l'Institut National des Sciences Appliquées de Toulouse, 2008.

[224] A. Marrel, B. Iooss, M. Jullien, B. Laurent, et E. Volkova. Global sensitivity analysis for models with spatially dependent outputs. *Environmetrics*, In press, 2010.

[225] A. Mas. Local Functional Principal Component Analysis. *Complex Analysis and Operator Theory*, 2(1) :135–167, 2008.

[226] A. Mas et B. Pumo. Functional linear regression with derivatives. *Journal of Nonparametric Statistics*, 21(1) :19–40, 2009.

[227] G. Matheron. Principles of Geostatistics. *Economic Geology*, 58 :1246–1268, 1963.

[228] José M. Matías. Multi-output Nonparametric Regression. In *12th Portuguese Conference on Artificial Intelligence*, volume 3808, pages 288–292, Covilhã, Portugal, 2005. Springer.

[229] G. McLachlan et D. Peel. *Finite Mixture Models*. John Wiley & Sons, 2000.

[230] M. Meilă. Comparing Clusterings. Technical Report Statistics TR. 418, University of Washington, 2002.

[231] Yves Meyer. *Wavelets : Algorithms & Applications*. SIAM, 1993.

[232] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[233] J.W. Milnor. On manifolds homeomorphic to the 7-sphere. *Annals of Mathematics*, 64(2) :399–405, 1956.

[234] J.W. Milnor. Differentiable structures on spheres. *American Journal of Mathematics*, 81(4) :962–972, 1959.

[235] M.L. Minsky et S.A. Papert. *Perceptrons*. MIT Press, 1969.

[236] U. Möller et D. Radke. Performance of data resampling methods for robust class discovery based on clustering. *Journal of Intelligent Data Analysis*, 10(2) :139–162, 2006.

[237] U. Möller et D. Radke. A Cluster Validity Approach based on Nearest-Neighbor Resampling. In *18th International Conference on Pattern Recognition*, pages 892–895, Hong-Kong, China, 2006. IEEE Computer Society.

[238] P. Monestiez et D. Nerini. *Functional and Operatorial Statistics*, chapter A Cokriging Method for Spatial Functional Data with Applications in Oceanology, pages 237–242. Springer, 2008.

[239] S. Monti, P. Tamayo, J. Mesirov, et T. Golub. Consensus clustering – A resampling-based method for class discovery and visualization of gene expression microarray data. *Journal of Machine Learning : Special Issue – Methods in Functional Genomics*, 52 (1) :91–118, 2003.

[240] J. Morgan et G. Tian. *Ricci Flow and the Poincare Conjecture*. American Mathematical Society, 2007.

[241] E.A. Nadaraya. On Estimating Regression. *Theory of Probability and its Applications*, 9(1) :141–142, 1964.

[242] J.F. Nash. The imbedding problem for Riemannian manifolds. *Annals of Mathematics*, 63(1) :20–63, 1956.

[243] J. Nelder et R. Wedderburn. Generalized Linear Models. *Journal of the Royal Statistical Society Series A*, 135(3) :370–384, 1972.

[244] A.Y. Ng, M. Jordan, et Y. Weiss. On Spectral Clustering : Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, pages 849–856, Vancouver, BC, Canada, 2002. MIT Press.

[245] J. Norris. *Markov Chains*. Cambridge University Press, 1997.

[246] A. Oliveira-Brochado et F. Vitorino Martins. Assessing the Number of Components in Mixture Models : a Review. Technical Report 194, Faculdade de Economia da Universidade do Porto, Portugal, 2005.

[247] C.H. Papadimitriou et K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Dover, 1998.

[248] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1) :157–160, 1890.

[249] G. Perelman. The entropy formula for the Ricci flow and its geometric applications, 2002. arXiv :math.DG/0211159.

[250] G. Perelman. Ricci flow with surgery on three-manifolds, 2003. arXiv :math.DG/0303109.

[251] G. Perelman. Finite extinction time for the solutions to the Ricci flow on certain three-manifolds, 2003. arXiv :math.DG/0307245.

[252] P. Petersen. *Riemannian Geometry*. Springer, 2006.

[253] K.W. Pettis, T.A. Bailey, A.K. Jain, et R.C. Dubes. An Intrinsic Dimensionality Estimator from Near-Neighbor Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1) :25–37, 1979.

[254] C. Preda et G. Saporta. Régression PLS sur un processus stochastique. *Revue de statistique appliquée*, 50(2) :27–45, 2002.

[255] W.H. Press, S.A. Teukolsky, W.T. Vetterling, et B.P. Flannery. *Numerical Recipes : The Art of Scientific Computing*. Cambridge University Press, 2007.

[256] H. Qiu et E.R. Hancock. Clustering and Embedding Using Commute Times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11) :1873–1890, 2007.

[257] M. Raginski et S. Lazebnik. Estimation of intrinsic dimensionality using high-rate vector quantization. In *Advances in Neural Information Processing Systems*, volume 17, pages 1105–1112. MIT Press, 2005.

[258] J.O. Ramsay et B.W. Silverman. *Functional Data Analysis*. Springer, 2005.

[259] W.M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336) :846–850, 1971.

[260] C.R. Rao. The use and interpretation of principal component analysis in applied research. *Sankhya : The Indian Journal of Statistics, Series A*, 26(4) :329–358, 1964.

[261] S. Richardson et P. Green. On bayesian analysis of mixtures with unknown number of components (with discussions). *Journal of the Royal Statistical Society Series B*, 59 (4) :731–792, 1997.

[262] B. Riemann. On the Hypotheses Which Lie at the Bases of Geometry. *Nature*, 8 : 14–17, 36–37, 1873. translated by W.K. Clifford.

[263] F. Rosenblatt. The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6) :386–408, 1958.

[264] H.H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3) :175–184, 1960.

[265] R. Rosipal et N. Kramer. *Subspace, Latent Structure and Feature Selection*, chapter Overview and recent advances in partial least squares, pages 34–51. Springer-Verlag, 2006. Statistical and Optimization Perspectives Workshop.

[266] F. Rossi, B. Conan-Guez, et A. El Golli. Clustering Functional Data with the SOM algorithm. In *12th European Symposium on Artificial Neural Networks*, pages 305–312, Bruges, Belgium, 2004. d-side.

[267] V. Roth, T. Lange, M. Braun, et J. Buhmann. A resampling approach to cluster validation. In *15th International Conference on Computational Statistics*, pages 123–128, Berlin, Germany, 2002. Springer.

[268] S. Roweis, L. Saul, et G. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems*, volume 13, pages 889–896, Vancouver, BC, Canada, 2001. MIT Press.

[269] S.T. Roweis et L.K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500) :2323–2326, 2000.

[270] C. Runge. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeitschrift für Mathematik und Physik*, 46 :224–243, 1901.

[271] M. Rupp, P. Schneider, et G. Schneider. Distance phenomena in high-dimensional chemical descriptor spaces : Consequences for similarity-based approaches. *Journal of Computational Chemistry*, 30(14) :2285–2296, 2009.

[272] M. Saerens, F. Fouss, L. Yen, et P. Dupont. The Principal Components Analysis of a Graph, and its Relationships to Spectral Clustering. In *15th European Conference on Machine Learning*, volume 3201, pages 371–383, Pisa, Italy, 2004.

[273] H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.

[274] A. Saltelli, K. Chan, et E.M. Scott, editors. *Sensitivity Analysis*. Wiley, 2000.

[275] S. Salvador et P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. Technical report, Florida Institute of Technology, Melbourne, 2003.

[276] J. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5) :401–409, 1969.

[277] S. Sandilya et S.R. Kulkarni. Principal Curves With Bounded Turn. *IEEE Transactions on Information Theory*, 48(10) :2789–2793, 2002.

[278] G. Sanguinetti, J. Laidler, et N.D. Lawrence. Automatic Determination of the Number of Clusters Using Spectral Algorithms. In *15th Workshop on Machine Learning for Signal Processing*, pages 55–60, Mystic, Connecticut, USA, 2005. IEEE.

[279] G. Saporta. *Probabilités, Analyse des données et Statistique*. Technip, 2006.

[280] P. Sarkar et A.W. Moore. A Tractable Approach to Finding Closest Truncated-commute-time Neighbors in Large Graphs. In *23rd Annual Conference on Uncertainty in Artificial Intelligence*, pages 335–343, Vancouver, BC, Canada, 2007. AUAI Press.

[281] S.A. Sarra. Integrated Multiquadric Radial Basis Function Approximation Methods. *Computers and Mathematics with Applications*, 51(8) :1283–1296, 2006.

[282] B. Schölkopf et A.J. Smola. *Learning with Kernels*. MIT Press, 2002.

[283] B. Schölkopf, A. Smola, et K-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10 :1299–1319, 1998.

[284] L.L. Schumaker. *Spline Functions : Basic Theory*. Cambridge University Press, 2007.

[285] N. Serban et L. Wasserman. CATS : Clustering after transformation and smoothing. *Journal of the American Statistical Association*, 100(471) :990–999, 2005.

[286] F. Sha et L.K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *22nd International Conference on Machine Learning*, pages 785–792, New York, NY, USA, 2005. ACM.

[287] B. Shaw et T. Jebara. Minimum Volume Embedding. In *11th International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 2007.

[288] J. Shen, S.I. Chang, E.S. Lee, Y. Deng, et S.J. Brown. Determination of cluster number in clustering microarray data. *Applied Mathematics and Computation*, 169(2) :1172–1185, 2005.

[289] Q. Shen et J. Faraway. An F Test for Linear Models with Functional Responses. *Statistica Sinica*, 14(4) :1239–1257, 2004.

[290] J. Shi et J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :888–905, 2000.

[291] J.Q. Shi et B. Wang. Curve prediction and clustering with mixtures of gaussian process functional regression models. *Journal of Statistics and Computing*, 18(3) :267–283, 2008.

[292] R. Shubhankar et M. Bani. Functional clustering by Bayesian wavelet methods. *Journal of the Royal Statistical Society Series B*, 68(2) :305–332, 2006.

[293] T.W. Simpson, J.D. Poplinski, P.N. Koch, et J.K. Allen. Metamodels for Computer-based Engineering Design : Survey and recommendations. *Engineering with Computers*, 17(2) :129–150, 2001.

[294] S. Smale. Generalized Poincaré's conjecture in dimensions greater than four. *Annals of Mathematics*, 74 :391–406, 1961.

[295] L.A. Smith. Intrinsic Limits on Dimension Calculations. *Physics Letters A*, 133(6) : 283–288, 1988.

[296] A.J. Smola et B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3) :199–222, 2003.

[297] I.M. Sobol. Sensitivity analysis for non-linear mathematical models. *Mathematical Modeling & Computational Experiment*, 1 :407–414, 1993.

[298] D.M.Y. Sommerville. *An Introduction to the Geometry of n Dimensions*. Dover, 1930, publié en 1958.

[299] M. Spivak. *A Comprehensive Introduction to Differential Geometry (5 volumes)*. Publish or Perish, 1999.

[300] B. Stein et O. Niggemann. On the nature of structure and its identification. In *25th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 122–134, Ascona, Suisse, 1999. Springer.

[301] M.L. Stein. *Interpolation of Spatial Data : Some Theory for Kriging*. Springer, 1999.

[302] H. Steinhaus. Sur la division des corps matériels en parties. *Bulletin of the Polish Academy of Sciences*, 4(12) :801–804, 1956.

[303] M. Stephens. Bayesian analysis of mixture models with an unknown number of components – an alternative to reversible jump methods. *Annals of Statistics*, 28(1) :40–74, 2000.

[304] M. Stoer et F. Wagner. A simple min-cut algorithm. *Journal of the Association for Computing Machinery*, 44(4) :585–591, 1997.

[305] C. Sugar et G.M. James. Finding the number of clusters in a dataset : An information-theoretic approach. *Journal of the American Statistical Association*, 98 :750–763, 2003.

[306] G.J. Szekely1 et M.L. Rizzo. Hierarchical Clustering via Joint Between-Within Distances : Extending Ward's Minimum Variance Method. *Journal of Classification*, 22 (2) :151–183, 2005.

[307] T. Tarpey et K.K.J. Kinateder. Clustering Functional Data. *Journal of Classification*, 20(1) :93–114, 2003.

[308] Y.W. Teh et S. Roweis. Automatic Alignment of Local Representations. In *Advances in Neural Information Processing Systems*, volume 15, pages 841–848, Vancouver, BC, Canada, 2003. MIT Press.

[309] J.B. Tenenbaum, V. Silva, et J.C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500) :2319–2323, 2000.

[310] M. Tenenhaus. *La régression PLS : Théorie et pratique*. Technip, 1998.

[311] R. Tibshirani. Principal curves revisited. *Statistics and Computing*, 2(4) :183–190, 1992.

[312] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, 58(1) :267–288, 1996.

[313] R. Tibshirani, G. Walther, D. Botstein, et P. Brown. Cluster Validation By Prediction Strength. Technical report, Stanford University, 2001.

[314] R. Tibshirani, G. Walther, et T. Hastie. Estimating the number of clusters in a data set via the gap statistic series b. *Journal of the Royal Statistical Society*, 63(2) :411–423, 2002.

[315] A-I.N. Tikhonov. *Solutions of Ill Posed Problems*. Vh Winston, 1977.

[316] W.S. Torgerson. *Theory & Methods of Scaling*. Wiley, 1958.

[317] G.V. Trunk. Statistical estimation of the intrinsic dimensionality of data collections. *Journal of Information and Control*, 12(5) :508–525, 1968.

[318] F.S. Tsai. Comparative Study of Dimensionality Reduction Techniques for Data Visualization. *Journal of Artificial Intelligence*, 3(3) :119–134, 2010.

[319] J.W. Tukey. *EDA ; Exploratory Data Analysis*. Addison-Wesley, 1977.

[320] G. Tutz et H. Binder. Boosting ridge regression. *Computational Statistics & Data Analysis*, 51(12) :6044–6059, 2007.

[321] L. van der Maaten. Matlab Toolbox for Dimensionality Reduction, 2010. URL http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html.

[322] L. van der Maaten et G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9 :2579–2605, 2008.

[323] L.J.P. van der Maaten, E.O. Postma, et H.J. van den Herik. Dimensionality Reduction : A Comparative Review, 2008.

[324] L. Vandenberghe et S. Boyd. Semidefinite programming. *SIAM Review*, 38(1) :49–95, 1996.

[325] E. Vazquez. *Modélisation comportementale de systèmes non-linéaires multivariables par méthodes à noyaux et applications*. Thèse de l'Université d'Orsay – Paris XI, 2005.

[326] E. Vazquez et E. Walter. Multi-Output Support Vector Regression. Technical report, Laboratoire des Signaux et Systèmes ; CNRS - Supélec - Université Paris-Sud, 2003.

[327] S. Vijayakumar et H. Ogawa. RKHS-based functional analysis for exact incremental learning. *Neurocomputing*, 29(1–3) :85–113, 1999.

[328] M. Vivien et R. Sabatier. Generalized orthogonal multiple co-inertia analysis(–PLS) : new multiblock component and regression methods. *Journal of Chemometrics*, 17(5) : 287–301, 2003.

[329] U. von Luxburg. A Tutorial on Spectral Clustering. Technical Report TR-149, Max Planck Institute for Biological Cybernetics, 2006.

[330] G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

[331] G. Wahba et S. Wold. A completely automatic french curve : fitting spline functions by cross validation. *Communications in Statistics – Theory and Methods*, 4(1) :1–17, 1975.

[332] G.G. Walter. *Wavelets and Other Orthogonal Systems with Applications*. CRC Press, 1994.

[333] D. Wang, X-J. Zeng, et J.A. Keane. An Input-Output Clustering Method for Fuzzy System Identification. In *16th Fuzzy Systems Conference*, pages 1–6, Imperial College, London, UK, 2007. IEEE.

[334] J. Wang, Z. Zhang, et H. Zha. Adaptive Manifold Learning. In *Advances in Neural Information Processing Systems*, volume 17, pages 1473–1480, Vancouver, BC, Canada, 2005. MIT Press.

[335] J.H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301) :236–244, 1963.

[336] G.S. Watson. Smooth Regression Analysis. *Sankhya : The Indian Journal of Statistics, Series A*, 26(4) :359–372, 1964.

[337] K.Q. Weinberger et L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1) :77–90, 2006.

[338] K.Q. Weinberger, F. Sha, et L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *21st International Conference on Machine Learning*, pages 839–846, Banff, Canada, 2004. ACM Press.

[339] W.J. Welch, R.J. Buck, J. Sacks, H.P. Wynn, T.J. Mitchell, et M.D. Morris. Screening, predicting, and computer experiments. *Technometrics*, 34(1) :15–25, 1992.

[340] R.L. Wilder. *Topology of manifolds*. American Mathematical Society, 1949.

[341] T. Wittman. MANI fold Learning Matlab Demo, 2005. URL http://www.math.ucla.edu/~wittman/mani/.

[342] H. Wold. *Multivariate Analysis*, chapter Estimation of principal components and related models by iterative least squares, pages 391–420. Academic Press, 1966.

[343] H. Wold. Soft Modeling by Latent Variables ; the Nonlinear Iterative Partial Least Squares Approach. *Perspectives in Probability and Statistics. Papers in honour of M. S. Bartlett*, 1975.

[344] Z. Wu et R. Leahy. An optimal graph theoretic approach to data clustering : theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11) :1101–1113, 1993.

[345] M. Yan et K. Ye. Determining the Number of Clusters Using the Weighted Gap Statistic. *Biometrics*, 63(4) :1031–1037, 2007.

[346] L. Yen, D. Vanvyve, F. Wouters, F. Fouss, M. Verleysen, et M. Saerens. Clustering using a random-walk based distance measure. In *13th Symposium on Artificial Neural Networks*, pages 317–324, Bruges, Belgium, 2005.

[347] L. Yen, F. Fouss, C. Decaestecker, P. Francq, et M. Saerens. Graph nodes clustering with the sigmoid commute-time kernel : A comparative study. *Data & Knowledge Engineering*, 68(3) :338–361, 2009.

[348] J.K. Yoo et R.D. Cook. Response dimension reduction for the conditional mean in multivariate regression. *Computational Statistics and Data Analysis*, 53(2) :334–343, 2008.

[349] L.A. Zadeh. Generalized theory of uncertainty (GTU) - Principal concepts and ideas. *Computational Statistics and Data Analysis*, 51 :15–46, 2006.

[350] C.T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, 20(1) :68–86, 1971.

[351] L. Zelnik-Manor et P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, volume 16, pages 1601–1608, Vancouver, BC, Canada, 2004. MIT Press.

[352] Y. Zhan, J. Yin, G. Zhang, et E. Zhu. Incremental Manifold Learning Algorithm Using PCA on Overlapping Local Neighborhoods for Dimensionality Reduction. In *3rd International Symposium on Advances in Computation and Intelligence*, volume 5370, pages 406–415, Wuhan, China, 2008. Springer-Verlag.

[353] Y. Zhan, J. Yin, X. Liu, et G. Zhang. Adaptive Neighborhood Select Based on Local Linearity for Nonlinear Dimensionality Reduction. In *4th International Symposium on Advances in Computation and Intelligence*, volume 5821, pages 337–348, Huangshi, China, 2009. Springer.

[354] X. Zhang, J. Li, et H. Yu. Local density adaptive similarity measurement for spectral clustering. *Pattern Recognition Letters*, 32(2) :352–358, 2011.

[355] Z. Zhang et H. Zha. Principal Manifolds and Nonlinear Dimensionality Reduction via Tangent Space Alignment. *SIAM Journal of Scientific Computing*, 26(1) :313–338, 2005.

[356] Z. Zhang et H. Zha. A Domain Decomposition Method for Fast Manifold Learning. In *Advances in Neural Information Processing Systems*, volume 18, Vancouver, BC, Canada, 2006. MIT Press.

[357] X. Zheng et X. Lin. Automatic determination of intrinsic cluster number family in spectral clustering using random walk on graph. In *5th International Conference on Image Processing*, pages 3471–3474, Singapore, Malaysia, 2004. IEEE.

[358] C. Zhou. *A Bayesian Model for Curve Clustering with Application to Gene Expression Data Analysis*. PhD thesis, University of Washington, 2003.