Thèse de doctorat de l'UTT

Chafic SAIDE

Filtrage adaptatif à l'aide de méthodes à noyau. Application au contrôle d'un palier magnétique actif

Spécialité :
Optimisation et Sûreté des Systèmes

2013TROY0018

Année 2013

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Chafic SAIDE

le 19 septembre 2013

Filtrage adaptatif à l'aide de méthodes à noyau. Application au contrôle d'un palier magnétique actif

JURY

M. S. CANU	PROFESSEUR DES UNIVERSITES	Président
M. R. EL ACHKAR	ASSOCIATE PROFESSOR	Directeur de thèse
M. P. HONEINE	MAITRE DE CONFERENCES	Examinateur
M. C. JAUFFRET	PROFESSEUR DES UNIVERSITES	Rapporteur
M. P. LARZABAL	PROFESSEUR DES UNIVERSITES	Rapporteur
M. R. LENGELLÉ	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Remerciements

Expression de gratitude à l'égard de toutes les personnes sans qui la présente étude n'aurait pas été possible. Je voudrais les prier d'accueillir ici toutes mes pensées émues qui viennent du fond de mon coeur, en acceptant mes remerciements.

Mes remerciements s'adressent en premier lieu à la personne en qui je salue la patience dont il a fait preuve à mon égard, Régis Lengellé, professeur des universités et directeur de l'école doctorale de l'UTT. Sans doute, cette thèse n'aurait eu lieu sans son encadrement, me favorisant l'opportunité et la liberté de mener à bout et à bien ce travail, malgré ses nombreuses charges. La complémentarité de sa compétence et son exigence m'ont beaucoup épaulé, grâce à ses remarques avisées et son partage de connaissances scientifiques.

Je remercie Paul Honeiné, Maître de Conférences à l'UTT, qui a toujours représenté le dynamo de la thèse, faisant preuve de trop de patience. Je lui suis reconnaissant de la disponibilité surtout dans les moments critiques, sans oublier toutefois son encadrement et son encouragement me faisant surmonter les difficultés observées dans les différentes étapes de cette thèse. Ses conseils et réflexions me remontaient le moral à divers moments, tout en m'apportant une compréhension plus approfondie de divers aspects du sujet.

Ce travail n'aurait jamais pu aboutir sans les conseils et les nombreuses discussions élaborées avec Roger Achkar, professeur à l'Université Américaine de Sciences et de Technologie (AUST) - Beirut. Ma gratitude s'adresse à lui pour l'intérêt, le soutien et la compréhension dont il a toujours fait preuve.

Un salut pour l'âme de Chaiban Nasr, sa coordination et ses discussions avec Régis Lengellé ont été le fil déclencheur de l'idée de la thèse.

Je tiens à remercier Ali Charara, Professeur des universités et directeur du Laboratoire HeuDiaSyC à l'UTC, pour la confiance qu'il m'a accordée, en nous accueillant au sein du laboratoire. Merci surtout pour les discussions que nous avons eues, apportant des réponses à des questions critiques.

Ma gratitude s'adresse aussi à Jérôme Demiras, Maître de Conférences à l'UTC, en qui je salue l'attitude proprement scientifique et professionnelle vis-à-vis des essais et des réponses à toutes mes questions concernant le palier magnétique actif.

Je souhaite inclure dans mes remerciements les personnes qui m'ont soutenu et assisté pendant mon séjour à l'UTT : Pascal Denis, Isabelle Leclercq, Thérèse Kazarian, Marie-José Rousselet et Veronique Bance. J'apprécie leur collaboration efficace et leur disponibilité continue, favorisant la bonne ambiance dont je garderai un très bon souvenir.

Mes affectueuses pensées adressées à Maya Kallas et Elias Khoury pour leur disponibilité permanente et leur support quotidien. Je n'oublie jamais les moments d'amitié que nous avons vécus ensemble.

Enfin, un grand merci à ma Thérèse pour le soutien moral quotidien qu'elle m'a porté et pour sa compréhension et sa patience malgé toutes les difficulités tout au long du trajet qui étaient parfois difficiles à gérer.

Nul ne peut atteindre l'aube sans passer par le chemin de la nuit.

Khalil Gibran

Table des matières

Table des figures

Liste des tableaux

Glossaire des notations et abréviations

Notation	Signification
$\mathcal{H}$	Espace de Hilbert
$\langle \cdot, \cdot \rangle_{\mathcal{H}}$	Produit scalaire dans l'espace $\mathcal{H}$
$\| \cdot \|_{\mathcal{H}}$	Norme associée au produit scalaire $\langle \cdot, \cdot \rangle_{\mathcal{H}}$
$\| \cdot \|_F$	Norme matricielle de Frobenius
ϕ	Fonction de mapping associée à l'espace de représentation $\mathcal{H}$
$\psi_n(\cdot)$	Modèle à l'instant n
$\boldsymbol{\alpha}_n$	Vecteur des coefficients du modèle à l'instant n
$\mathbb{R}$	Ensemble des nombres réels
$\mathbb{N}$	Ensemble des entiers naturels
$\mathcal{U}$	Espace d'entrées
$\mathcal{A}$	Ensemble d'apprentissage
$\mathcal{D}_n$	Dictionnaire à l'instant n
$\mathcal{D}_n^A$	Dictionnaire adapté à l'instant n
$\boldsymbol{K}_n$	Matrice de Gram des éléments du dictionnaire à l'instant n
p	Nombre de collecteurs (manifolds) dans l'algorithme KAPA
μ_0	Seuil de cohérence du dictionnaire
m	Taille du dictionnaire
$\boldsymbol{\lambda}$	Vecteur des multiplicateurs de Lagrange
$\boldsymbol{\Lambda}$	Matrice diagonale des multiplicateurs de Lagrange
$\mathbf{u}_{w_i}$	$i^{\text{ème}}$ élément du dictionnaire
$\mathbf{u}_{w_i}^A$	$i^{\text{ème}}$ élément du dictionnaire après adaptation
$\mathbf{g}_{w_i}$	Gradient de l'erreur quadratique par rapport au $i^{\text{ème}}$ élément du dictionnaire
ν_n	Pas du gradient pour adapter les éléments du dictionnaire à l'instant n
ν_0	Pas du gradient référence pour l'heuristique d'adaptation
N	Taille de l'échantillon
β	Degré du noyau polynomial
σ	Bande passante d'un noyau radial

Abréviation	Signification
RKHS	Espace de Hilbert à noyau reproduisant (Reproducing Kernel Hilbert Space)
SVM	Machines à Supports Vecteurs (Support Vector Machines)
SVR	Régression à Supports Vecteurs (Support Vector Regression)
KRLS	Algorithme de moindres carrés récursif à noyau
KAPA	Algorithme de projection affine à noyau
KNLMS	Algorithme de moindres carrés normalisé à noyau
RBF	Fonction noyau radiale (Radial Basis Function)
NMSE	Erreur quadratique moyenne normalisée (Normalized Mean Squared Error)
MISO	Modèle à entrées multiples et sortie unique (Multiple Inputs single Output)
MIMO	Modèle à entrées et sorties multiples (Multiple Inputs Multiples Outputs)
AKRLS	Algorithme KRLS avec adaptation du dictionnaire
AKAPA	Algorithme KAPA avec adaptation du dictionnaire
AKNLMS	Algorithme KNLMS avec adaptation du dictionnaire
MOKRLS	Algorithme KRLS pour modèle MIMO
MOKAPA	Algorithme KAPA pour modèle MIMO
MOKNLMS	Algorithme KNLMS pour modèle MIMO
PMA	Palier Magnétique Actif

Résumé

L'estimation fonctionnelle basée sur les espaces de Hilbert à noyau reproduisant a récemment conduit à des avancées majeures dans le domaine de l'identification des systèmes non linéaires. Toutefois, l'ordre des modèles identifiés est égal au nombre de couples entrée-sortie, ce qui rend cette méthode inadéquate pour une identification en ligne. Pour surmonter cet inconvénient, plusieurs méthodes de parcimonie ont été proposées afin de contrôler l'ordre du modèle. Parmi ces méthodes, le critère de cohérence qui est d'une grande importance, vu sa simplicité et son faible coût calculatoire. Par conséquent, le modèle est alors défini à partir d'un *dictionnaire* de faible taille.

Le dictionnaire est donc formé par les fonctions noyau les plus pertinentes qui forment le modèle. Une fonction noyau introduite dans le dictionnaire y demeure même si la non-stationnarité du système étudié rend sa contribution faible dans l'estimation de la sortie courante. Il apparaît alors opportun d'adapter les éléments du dictionnaire pour obtenir un dictionnaire amélioré afin de réduire l'erreur quadratique instantanée résultante et/ou mieux contrôler l'ordre du modèle.

D'autre part, les algorithmes adaptatifs utilisant le critère de cohérence sont développés pour identifier des modèles à sortie unique. La généralisation de ces algorithmes pour couvrir les cas des modèles à sorties multiples est une nécessité. Pour ces modèles, le dictionnaire est commun pour toutes les sorties et parsuite les mêmes heuristiques développées pour les modèles à sortie unique seront applicables dans ce cas.

La première partie de ce manuscrit traite le sujet des algorithmes adaptatifs utilisant un critère de parcimonie et spécifiquement le critère de cohérence. L'adaptation des éléments du dictionnaire en utilisant une méthode à gradient stochastique est abordée pour deux familles de fonctions noyau. Les heuristiques proposées varient suivant la fonction noyau utlisée et la faisabilité de cette approche est prouvée par des simulations variées. Cette partie a un autre objectif qui est la dérivation des algorithmes adaptatifs utilisant le critère de cohérence pour identifier des modèles à sorties multiples ainsi que l'adaptation des éléments du dictionnaire commun à toutes les sorties.

La deuxième partie introduit d'une manière abrégée le palier magnétique actif (PMA) avec une exploration des aspects pour contrôler son fonctionnement. La proposition de contrôler un PMA par un algorithme adaptatif à noyau est présentée pour remplacer une méthode utilisant les réseaux de neurones à couches multiples. Cette approche est testée à travers des simulations et des essais réels.

En résumé, nous examinons des heuristiques pour l'adaptation des éléments d'un dictionnaire afin de contrôler davantage l'ordre du modèle estimé et/ou minimiser l'erreur instantanée. De même,

les algorithmes adaptatifs sont généralisés pour identifier des systèmes à sorties multiples. Plusieurs expérimentations validant la justesse des méthodes proposées sont conduites sur des benchmarks réels et artificiels.

Abstract

Function approximation methods based on reproducing kernel Hilbert spaces (RKHS) are of great importance in kernel-based regression methods and led to major advances in the identification of nonlinear systems. However, the order of the identified model is equal to the number of observations (input-output pairs), which makes this approach inappropriate for online identification. To overcome this drawback, several sparsification methods have been proposed to control the order of the model. Among these sparsification methods, the coherence criterion is of great importance due to its simplicity and reduced calculation cost. It has been shown possible to select a subset of the most relevant passed input vectors to form a *dictionary* to identify the model.

A kernel function, once introduced into the dictionary, remains unchanged even if the non-stationarity of the studied system makes it less influent in estimating the current output of the model. This observation leads to the idea of adapting the elements of the dictionary to obtain an improved one with an objective to minimize the resulting instantaneous mean square error and/or to control the order of the model.

Furthermore, adaptive algorithms using the coherence criterion are developed to identify single output models. The generalization of these algorithms to cover the case of multiple outputs models is a necessity. Since all outputs of the model share the same dictionary, the same heuristics developed for the single-output models must be handled to become applicable in the case of multiple outputs models.

The first part of this thesis deals with adaptive algorithms using a sparsification criterion and specifically the coherence criterion. The adaptation of the elements of the dictionary using a stochastic gradient method is presented for two categories of kernel functions. The proposed heuristics vary depending on the used kernel function and the feasibility of this approach is proved by various simulations. Another topic is covered in this part which is the implementation of adaptive algorithms using the coherence criterion to identify Multiple-Outputs models as well as the adaptation of the dictionary elements.

The second part introduces briefly the active magnetic bearing (AMB) with an exploration of the aspects to control it. A proposed method to control an AMB by an adaptive algorithm using kernel methods is presented to replace an existing method using multi-layer perceptrons neural networks. This approach is tested through simulations.

In summary, we consider heuristics for adapting elements of a dictionary to further control the order of the estimated model and/or minimize the instantaneous error. Similarly, adaptive algorithms are generalized to identify systems with multiple outputs. Several experiments to validate the accuracy of the proposed methods are conducted on different benchmarks.

Introduction

La modélisation des systèmes non linéaires et non-stationnaires a été récemment largement étudiée. Face à ces systèmes, la simplicité des méthodes d'identification linéaires s'estompe dans le cas des systèmes non stationnaires en laissant la place à des méthodes adaptatives comme par exemple les méthodes des filtres de Volterra [Sch80, Wie66] et les réseaux neuronaux [Hay08]. L'une de ces méthodes adaptatives, qui a connu beaucoup d'attention, est la méthode d'approximation fonctionnelle basée sur les espaces de Hilbert à noyau reproduisant (RKHS) $\mathcal{H}$. Cette aproche constitue un prolongement des méthodes linéaires simples en remplaçant le calcul de certains produits scalaires par un noyau de Mercer. Les méthodes adaptatives ont pour but de suivre l'évolution des systèmes non-stationnaires avec le temps en se basant simplement sur des observations (des entrées avec des sorties désirées du modèle). Ce processus s'appelle l'*apprentissage supervisé*.

En apprentissage supervisé, l'existence de couples d'entrée-sortie est obligatoire. Généralement l'*espace d'entrée* $\mathcal{U}$ est un sous espace Euclidien de $\mathbb{R}^l$, et l'*espace de sortie* est un sous ensemble de $\mathbb{R}$. Lorsque l'ensemble de sorties est dénombrable, on parle dans ce cas d'un problème de "*classification*", dans le cas contraire, le problème sera un problème de "*régression*". En supposant qu'une relation entre les entrées et la sortie existe, l'objectif d'un algorithme d'apprentissage est de trouver cette relation.

La théorie des noyaux reproduisants a permis le développement de plusieurs algorithmes adaptatifs. La capacité adaptative de ces algorithmes est basée sur le principe d'apprentissage par correction de l'erreur entre la sortie désirée et la sortie actuelle du modèle estimé. L'idée de base des méthodes à noyau consiste à projeter non linéairement les données de l'espace d'entrée dans un espace vectoriel de dimension élevée, appelé "*espace transformé*", dans lequel les méthodes linéaires peuvent être appliquées. L'avantage des méthodes à noyau est que le calcul des grandeurs à évaluer se fait dans l'espace initial indépendamment de la dimension de l'espace transformé [Aro50].

Les exigences des calculs pour l'application des algorithmes adaptatifs utilisant les méthodes à noyau sont basées sur des matrices dont la taille augmente avec le nombre d'observations ; ce qui rend ces méthodes inadéquates pour les applications en ligne [KSW04]. Plusieurs approches ont été proposées pour remédier à cet inconvénient. L'idée principale derrière ces approches consiste à introduire un nouvel échantillon au modèle s'il contribue de manière significative à la réduction de l'erreur d'approximation. Ces méthodes sont appelées critères de *parcimonie* ou de *sparsification*. Les échantillons obtenus après l'application d'un critère de parcimonie forment ce qu'on appelle un *dictionnaire* dont la taille n'est autre que l'ordre du modèle obtenu. Le critère de cohérence est l'un des critère de parcimonie les plus importants vu sa simplicité et son coût calculatoire réduit. De plus, l'application de ce critère montre que la taille du dictionnaire reste finie avec le temps [RBH09] d'où la

possibilité de l'adoption des algorithmes adaptatifs utilisant ce critère pour les applications en ligne.

Un élément, une fois introduit dans le dictionnaire, reste inchangé même si la non-stationarité du système minimise sa contribution dans l'estimation du modèle. De ce fait, l'idée de l'adaptation des éléments du dictionnaire d'une façon instantanée semble être une solution pour réduire la taille du dictionnaire, en conséquence,l'ordre du modèle et/ou la réduction de l'erreur quadratique instantanée.

L'objectif de ce manuscrit est de proposer des heuristiques d'adaptation deséléments du dictionnaire obtenu en appliquant le critère de cohérence. Ces heuristiques, basées sur le gradient stochastique de l'erreur quadratique instantanée, sont dérivées pour des familles différentes de fonctions noyau telles que les noyaux radiaux et les noyaux polynômiaux. Les algorithmes adaptatifs utilisant le critère de cohérence sont utilisés pour identifier des modèles à sortie unique (scalaire). Une généralisation de ces algorithmes semble utile pour aborder l'identification des modèles à sorties multiples. De plus, une adaptation des éléments du dictionnaire en utilisant les mêmes heuristiques appliquées dans le cas des modèles à sortie unique est proposée. Pour l'application des algorithmes dérivés, on propose le contrôle d'un palier magnétique actif (PMA) qui est un système fortement non-linéaire. Des simulations et des essais en temps réel sont menés pour faire une comparaison entre les algorithmes adaptatifs utilisant les méthodes à noyau avec les autres méthodes de contrôle déjà utilisées.

Plan du manuscrit

Le manuscrit est composé de deux parties. La première partie qui s'étend du chapitre 1 au Chapitre 3, traite les algorithmes adaptatifs à noyau avec adaptation du dictionnaire, y inclus l'identification des modèles à sorties multiples. La deuxième partie qui est formée du Chapitre 4, introduit le palier magnétique actif avec les méthodes proposées pour le contrôler en conduisant des simulationsVoici la description abrégée du contenu des chapitres.

Chapitre 1 : Méthodes à noyau, critères de parcimonie et algorithmes adaptatifs

Ce chapitre est consacré aux méthodes à noyau pour l'identificaton des modèles dont l'ordre croît avec le temps et on s'intéresse particulièrement au théorème de représentation. Les principaux critères de parcimonie sont aussi explorés ainsi que les algorithmes d'apprentissage en ligne. On décrit deux algorithmes, le premier est celui des moindres carrés récursif à noyau (KRLS) et le second est la projection affine à noyau (KAPA).

Chapitre 2 : Adaptation du dictionnaire pour les algorithmes de prédiction en ligne

Les heuristiques de l'adaptation des éléments du dictionnaire obtenu en appliquant le critère de cohérence sont explicitement décrites dans ce chapitre. L'idée essentielle pivote autour de l'efficacité de

l'adaptation et sur le fait que chaque type de fonctions noyau nécessite une heuristique différente, en particulier la famille des noyaux radiaux et celle du noyau polynômial. Des simulations diversifiées sont faites pour confirmer les propositions adoptées.

Chapitre 3 : Algorithmes adaptatifs en ligne pour les modèles multi-sorties

La généralisation des algorithmes adaptatifs utilisant le critère de cohérence pour identifier des systèmes à sorties multiples (MIMO) est exploitée dans ce chapitre. En particulier, Les algorithme KAPA et KRLS sont ciblés. Et puisque toutes les sorties partagent le même dictionnare, l'adaptation des éléments de ce dernier en utilisant les mêmes heuristiques développées dans le Chapitre 2, est appliquée en minimisant la norne du vecteur des erreurs instantanées.

Chapitre 4 : Contrôle d'un palier magnétique actif

On introduit dans ce chapitre le palier magnétique actif (PMA) d'une façon abrégée avec les méthodes utilisées pour le contrôler, en particulier, la méthode des réseaux neurones multi-couches étudiée dans [ANDMC06]. Une méthode utilisant les méthodes à noyau est proposée pour le contrôle du PMA. L'efficacité de la méthode proposée est vérifiée par des simulations.

Annexes A et B

Dans les annexes A et B, les calculs détaillés pour l'obtention des algorithmes adaptatifs KRLS et KAPA sont présentés..

Produits de la recherche

Lors de la préparation de cette thèse, plusieurs publications ont vu le jour. Voici dans ce qui suit une liste de ces publications citées dans l'ordre chronologique de leur apparition.

1. Saide, C. ; Lengelle, R. ; Honeine, P. ; Richard, C. ; Achkar, R., Dictionary adaptation for online prediction of time series data with kernels, *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pp.604-607, 5-8 Aug. 2012.

2. Saide, C. ; Lengelle, R. ; Honeine, P. ; Achkar, R., Online Kernel Adaptive Algorithms With Dictionary Adaptation for MIMO Models, *Signal Processing Letters, IEEE*, vol.20, no.5, pp.535,538, May 2013.

3. Saide, C. ; Honeine, P. ; Lengelle, R. ; Richard, C. ; Achkar, R., Adaptation en ligne d'un dictionnaire pour les méthodes à noyau, In *Actes du XXIVème Colloque GRETSI sur le Traitement du Signal et des Images, Brest, France, September 2013*.

Première partie

Algorithmes adaptatifs à noyau avec adaptation du dictionnaire

CHAPITRE 1

Méthodes à noyau, critères de parcimonie et algorithmes adaptatifs

Sommaire

L'objectif de ce premier chapitre est de faire une présentation des méthodes à noyau et des critères de parcimonie. Nous allons définir ce qu'est qu'un noyau, ses caractéristiques ainsi que celles des espaces associés aux noyaux. Les notions de critère de cohérence ainsi que de *dictionnaire* seront détaillées en décrivant l'intérêt qu'ils portent pour l'identification en ligne de systèmes non linéaires. Le théorème de représentation sera étudié conjointement avec le critère de cohérence. Enfin, quelques algorithmes adaptatifs à noyau, utilisés et développés dans notre étude, seront présentés.

1.1 Introduction

Les modèles linéaires sont aisés à étudier par le simple fait qu'ils peuvent être entièrement déterminés par la connaissance de leur réponse impulsionnelle. Les modèles non linéaires sont d'une importance primordiale pour résoudre certains problèmes de la vie réelle, en particulier dans les domaines du traitement des signaux, de l'ingénierie biomédicale ou encore de l'analyse des séries chronologiques, voir par exemple [Sch80, Wie66]. L'une des possibilités pour obtenir un modèle non linéaire consiste à transformer les données initiales dans un espace convenable dans lequel les méthodes linéaires peuvent être appliquées. En général, cela conduit à une difficulté liée à la très grande dimension du nouvel espace de travail. Dans le cadre non linéaire, les réseaux neurones artificiels comme le perceptron multicouches [Hay08] ont fait l'objet d'une littérature abondante. Le problème essentiel qui se pose lors de l'utilisation de ce type de méthodes est de déterminer l'architecture appropriée, a priori, pendant le processus d'apprentissage ou encore a posteriori. Ceci résulte du fait que la fonction coût utilisée ne contient pas, en général, de terme de contrôle de la complexité de la solution.

Au cours des dernières années, des méthodes d'identification dites à noyau, ont été développées [Aro50, Vap95]. Ces méthodes à noyau fournissent des solutions non linéaires avec un coût calculatoire réduit. L'idée de base est de projeter non linéairement les données dans un espace de dimension élevée et d'effectuer un traitement linéaire dans ce nouvel espace. L'avantage des méthodes à noyau est que, dans la mesure où les traitements linéaires reposent sur les calculs de produits scalaires, effectuer la projection des données dans un espace de Hilbert à noyau reproduisant permet de calculer les grandeurs à évaluer dans l'espace initial, les calculs devenant alors indépendants de la dimension de l'espace dans lequel les données ont été projetées.

Dans ce chapitre, on présente la théorie des noyaux reproduisants et les outils mathématiques qui accompagnent cette théorie. Dans le cadre de la sélection de modèle, l'intérêt du critère de cohérence est présenté pour l'identification en ligne des systèmes.

1.2 Notions de base

L'idée essentielle des méthodes à noyau consiste à projeter les données de l'espace des entrées $\mathcal{U}$ dans un espace vectoriel, de dimension plus grande, à l'aide d'une fonction non linéaire telle que :

$$\begin{array}{rcl} \phi : \mathcal{U} & \mapsto & \mathcal{H} \\ \boldsymbol{u} & \rightarrow & \phi(\boldsymbol{u}) \end{array}$$

Après cette projection, des algorithmes linéaires peuvent être appliqués dans $\mathcal{H}$.

Définition 1.1. *Un noyau est une fonction $\kappa : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}$ telle que pour tout $\boldsymbol{u}$, $\boldsymbol{u'} \in \mathcal{U}$:*

$$\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{u'}) \rangle_{\mathcal{H}} \tag{1.1}$$

où ϕ est une transformation de $\mathcal{U}$ dans un espace vectoriel $\mathcal{H}$ muni d'un produit scalaire $\langle \cdot, \cdot \rangle_{\mathcal{H}}$.

La fonction noyau κ ne peut être choisie arbitrairement mais elle doit vérifier certaines conditions, détaillées dans le paragraphe suivant.

1.3 Qualification des fonctions noyaux

Cette section est consacrée aux caractéristiques que doit posséder une fonction noyau, afin qu'il puisse vérifier les propriétés d'un produit scalaire et aux définitions utiles pour la suite de ce document.

Proposition 1.1. *Une fonction noyau est une fonction symétrique si :*

$$\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \kappa(\boldsymbol{u'}, \boldsymbol{u}) \quad \forall \boldsymbol{u}, \boldsymbol{u'} \in \mathcal{U} \tag{1.2}$$

Cependant, la condition citée ne garantit pas l'existence d'un espace $\mathcal{H}$ associé à κ.

Définition 1.2. *(Produit scalaire - Espace préhilbertien) Un espace $\mathcal{H}$ est muni d'un produit scalaire s'il existe une forme bilinéaire $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ symétrique et à valeur réelle telle que pour tout $f \in \mathcal{H}$, on a $\langle f, f \rangle_{\mathcal{H}} \geq 0$, avec l'égalité obtenue uniquement pour $f = 0$. On dit que $\mathcal{H}$ est un espace préhilbertien.*

Un produit scalaire vérifie les propriétés suivantes pour $f, g, h \in \mathcal{H}$ et $a \in \mathbb{R}$:
- $\langle f, g \rangle_{\mathcal{H}} = \langle g, f \rangle_{\mathcal{H}}$
- $\langle f + g, h \rangle_{\mathcal{H}} = \langle f, h \rangle_{\mathcal{H}} + \langle g, h \rangle_{\mathcal{H}}$
- $\langle af, g \rangle_{\mathcal{H}} = a \langle f, g \rangle_{\mathcal{H}}$
- $\langle f, f \rangle_{\mathcal{H}} = 0 \Leftrightarrow f = 0$

Définition 1.3. *(Espace de Hilbert) Un espace $\mathcal{H}$ est muni d'un produit scalaire $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ est un espace de Hilbert s'il est complet pour la norme associée au produit scalaire $\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}}$ (ce qui signifie que toutes les séries de Cauchy convergent dans $\mathcal{H}$).*

Par exemple, l'espace $\mathbb{R}^n$, l'ensemble des polynômes de degré inférieur ou égal à n et l'espace des fonctions de carré intégrable sur l'intervalle $[a, b]$, $\mathcal{L}^2([a, b])$, sont des espaces de Hilbert. L'espace des fonctions intégrables $\mathcal{L}^1([a, b])$, ainsi que l'espace $\mathcal{L}^\infty([a, b])$ des fonctions bornées sur l'intervalle $[a, b]$, ne sont pas des espaces de Hilbert.

On peut munir un espace de Hilbert $\mathcal{H}$ d'une base orthonormée permettant de représenter les éléments de $\mathcal{H}$ à partir de leurs coordonnées. Le noyau associé est supposé normalisé, si tel n'est pas le cas, il convient de le normaliser.

Définition 1.4. *(Espace de Hilbert à noyau reproduisant - RKHS) On appelle espace de Hilbert à noyau reproduisant un espace de Hilbert $\mathcal{H}$ de fonctions définies sur $\mathcal{U}$, à valeurs réelles, tel que pour tout $\boldsymbol{u} \in \mathcal{U}$, les fonctions d'évaluation $F_{\boldsymbol{u}}$ définies par $F_{\boldsymbol{u}}(f) = f(\boldsymbol{u})$, $\forall f \in \mathcal{H}$, sont bornées :*

$$\forall \mathbf{u} \in \mathcal{U}, \exists a_{\mathbf{u}} \text{ tel que } |f(\mathbf{u})| \leq a_{\mathbf{u}} \|f\|_{\mathcal{H}} \tag{1.3}$$

L'espace $\mathbb{R}^n$, l'ensemble des polynômes de degré inférieur ou égal à n sont des espaces de Hilbert à noyau reproduisant tandis que l'espace $\mathcal{L}^2([a,b])$ ne l'est pas.

Définition 1.5. *(Matrice de Gram) On appelle matrice de Gram de dimension $(n \times n)$, la matrice semi-définie positive dont le terme général peut être écrit $K_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$ avec*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K_{ij} \geq 0 \tag{1.4}$$

pour toute séquence de nombres réels $\{a_i\}_{i=1}^n$.

Définition 1.6. *(Fonction définie positive) Une fonction $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ qui, pour tout $n \in \mathbb{N}$ et pour tous $\boldsymbol{u}_1 \cdots \boldsymbol{u}_n \in \mathcal{U}$, donne une matrice de Gram semi-définie positive est dite définie positive.*

Il faut noter que tout produit scalaire est défini positif.

En se basant sur les définitions précédentes, le théorème suivant fournit une condition nécessaire et suffisante d'admissibilité d'une fonction noyau.

Théorème 1.1. *(Théorème de Moore-Aronszajn [Aro50]) A toute fonction κ semi-définie positive sur $\mathcal{U} \times \mathcal{U}$, il correspond un RKHS unique de fonctions à valeurs réelles définies sur $\mathcal{U}$, et réciproquement.*

Démonstration. Dans le but de construire l'espace de Hilbert à noyau reproduisant associé à κ, considérons la transformation ϕ suivante :

$$\begin{aligned} \phi : \mathcal{U} &\mapsto \mathcal{H} \\ \mathbf{u} &\to \kappa(\mathbf{u}, .) \end{aligned}$$

où $\phi(\mathbf{u}) = \kappa(\mathbf{u}, .)$ désigne une fonction semi-définie positive sur $\mathcal{U}$, obtenue en fixant le premier argument de κ à $\mathbf{u}$, et $\mathcal{H}$ l'espace engendré par les fonctions $\kappa(\mathbf{u}, \cdot)$ avec $\mathbf{u} \in \mathcal{U}$. Etant donné deux fonctions de $\mathcal{H}$

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{u}_i, \cdot), \qquad g(\cdot) = \sum_{j=1}^{n'} \beta_j \kappa(\mathbf{u'}_j, \cdot) \tag{1.5}$$

avec $n, n' \in \mathbb{N}$, $\alpha_i, \beta_j \in \mathbb{R}$ et $\mathbf{u}_i, \mathbf{u'}_j \in \mathcal{U}$, on considère la forme bilinéaire suivante :

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u'}_j) \tag{1.6}$$

dont on cherche à démontrer qu'il s'agit bien d'un produit scalaire dans $\mathcal{H}$. On note en premier lieu qu'en

utilisant les définitions de f, g et la symétrie de κ, on peut mettre cette expression sous la forme :

$$\langle f, g\rangle_{\mathcal{H}} = \sum_{j=1}^{n'} \beta_j f(\mathbf{u'}_j) = \sum_{i=1}^{n} \alpha_i g(\mathbf{u}_i) \tag{1.7}$$

De (1.7), il est clair que $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ est à valeur réelle, symétrique et bilinéaire. Par ailleurs, comme κ est semi-définie positive, on a :

$$\langle f, f\rangle_{\mathcal{H}} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \kappa(\mathbf{u}_i, \mathbf{u}_j) \geq 0 \qquad \forall f \in \mathcal{H} \tag{1.8}$$

Par conséquent, pour des fonctions $f_1 \cdots f_N$ et des coefficients $\gamma_1 \cdots \gamma_N \in \mathbb{R}$, on a

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \gamma_i \gamma_j \langle f_i, f_j\rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{N} \gamma_i f_i, \sum_{j=1}^{N} \gamma_j f_j \right\rangle_{\mathcal{H}} \geq 0 \tag{1.9}$$

ce qui montre bien que $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ est semi-défini positif. Il reste de prouver la propriété

$$\langle f, f\rangle_{\mathcal{H}} = 0 \Leftrightarrow f = 0 \quad \forall f \in \mathcal{H} \tag{1.10}$$

pour démontrer que (1.6) est un produit scalaire. Or d'après (1.7) toute fonction $f \in \mathcal{H}$ vérifie :

$$\langle f, \kappa(\mathbf{u}, \cdot)\rangle_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{u}, \mathbf{u}_i) = f(\mathbf{u}) \tag{1.11}$$

en particulier

$$\langle \kappa(\mathbf{u}, \cdot), \kappa(\mathbf{u'}, \cdot)\rangle_{\mathcal{H}} = \kappa(\mathbf{u}, \mathbf{u'}) \tag{1.12}$$

Des équations, (1.23), (1.11) et (1.12), on déduit

$$f(\mathbf{u})^2 = \langle f, \kappa(\mathbf{u}, \cdot)\rangle_{\mathcal{H}}^2 \leq \langle f, f\rangle_{\mathcal{H}}\ \kappa(\mathbf{u}, \mathbf{u}) \tag{1.13}$$

qui prouve la propriété (1.10). Ainsi $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ est bien un produit scalaire sur $\mathcal{H}$. Afin de transformer $\mathcal{H}$ en un espace de Hilbert, il suffit de le compléter conformément à [Aro50] de manière à ce que toute suite de Cauchy y converge. En vertu de (1.11), $\mathcal{H}$ est un espace de Hilbert à noyau reproduisant. D'après (1.12), la fonction κ représente un produit scalaire dans $\mathcal{H}$. Elle est appelée noyau reproduisant de $\mathcal{H}$.

La réciproque du théorème peut être démontrée en s'appuyant sur le théorème de représentation de Riesz [RS80]. D'après ce théorème, il existe pour tout $\mathbf{u} \in \mathcal{U}$, une fonction $\kappa(\mathbf{u}, \cdot) \in \mathcal{H}$ qui vérifie (1.11). Or, il résulte de (1.12) que la fonction $\kappa(\mathbf{u}, \mathbf{u'})$ définie de $\mathcal{U} \times \mathcal{U}$ dans $\mathbb{R}$ est symétrique. Elle vérifie

également

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\kappa(\mathbf{u}_i,\mathbf{u}_j)=\left\langle\sum_{i=1}^{n}\alpha_i\kappa(\mathbf{u}_i,\cdot),\sum_{j=1}^{n}\alpha_j\kappa(\mathbf{u}_j,\cdot)\right\rangle_{\mathcal{H}}=\left\|\sum_{i=1}^{n}\alpha_i\kappa(\mathbf{u}_i,\cdot)\right\|_{\mathcal{H}}^2\geq 0 \tag{1.14}$$

pour tout $\alpha_1,\cdots\alpha_n\in\mathbb{R}$. Par conséquent, κ est semi-définie positive. □

L'espace RKHS associé à κ est appelé espace caractéristique canonique et $\kappa(\mathbf{u},\cdot)$ sa fonction de mapping canonique. Il faut noter que les noyaux peuvent être également complexes, en l'occurrence l'espace associé est un espace Hilbert complexe à noyau reproduisant [BCR84]. La seule condition que doit vérifier l'espace des observations $\mathcal{U}$ est qu'il doit être non vide. Par ailleurs, il est possible d'appliquer certains noyaux sur des données non vectorielles [Hau99, Wat99, GLF02, LSST+02, Ver02, CFV05], par exemple des chaînes de caractères ou des graphes. Cette propriété très importante reflète l'importance des méthodes à noyau.

1.4 Théorème de Mercer

Dans cette section on va introduire le théorème de Mercer qui permet de s'assurer de la construction de l'espace induit par une fonction noyau semi-définie positive. Considérons un espace d'entrées fini $\mathcal{U}=\{\mathbf{u}_1\cdots\mathbf{u}_n\}$, et supposons que $\kappa(\mathbf{u}_i,\mathbf{u}_j)$ est une fonction symétrique dans $\mathcal{U}$. Soit **K** la matrice telle que :

$$\mathbf{K}=\big(\kappa(\mathbf{u}_i,\mathbf{u}_j)\big)_{i,j=1\cdots n} \tag{1.15}$$

puisque **K** est symétrique, il existe une matrice orthogonale **V** telle que $\mathbf{K}=\mathbf{V}\Lambda\mathbf{V}^t$, avec Λ une matrice diagonale qui contient toutes les valeurs propres λ_h de **K**, et les vecteurs propres $\mathbf{v}_h$ qui sont les colonnes de **V**.

D'après la théorie de Hilbert-Schmidt [CH53], toute fonction continue et symétrique $\kappa(\mathbf{u},\mathbf{u'})$ admet une décomposition de la forme :

$$\kappa(\mathbf{u},\mathbf{u'})=\sum_{i=0}^{\infty}\lambda_i\mathbf{v}_i(\mathbf{u})\mathbf{v}_i(\mathbf{u'}) \tag{1.16}$$

En supposant que toutes les valeurs propres sont non-négatives et en adoptant par exemple une transformation ϕ telle que :

$$\phi:\mathbf{u}_i\rightarrow\big(\sqrt{\lambda_h}\mathbf{v}_{hi}\big)_{h=1\cdots n}\in\mathbb{R}^n,\ i=1\cdots n \tag{1.17}$$

On a alors

$$\langle\phi(\mathbf{u}_i),\phi(\mathbf{u}_j)\rangle_{\mathcal{H}}=\sum_{h=1}^{n}\lambda_h\mathbf{v}_{hi}\mathbf{v}_{hj}=(\mathbf{V}\Lambda\mathbf{V}^t)_{ij}=\kappa(\mathbf{u}_i,\mathbf{u}_j) \tag{1.18}$$

ce qui implique que $\kappa(\mathbf{u},\mathbf{u'})$ est bien une fonction noyau correspondant à l'espace transformé.

La condition nécessaire est que les valeurs propres de K soient toutes non-négatives. Or, si l'on a

une valeur propre négative λ_s associée au vecteur propre v_s, le point

$$\mathbf{z} = \sum_{i=1}^{n} \mathbf{v}_{si}\phi(\mathbf{u}_i) = \sqrt{\Lambda}\mathbf{V}^t\mathbf{v}_s \tag{1.19}$$

dans l'espace transformé aura une norme carrée

$$\|\mathbf{z}\|^2 = \langle \mathbf{z}, \mathbf{z} \rangle = \mathbf{v}_s^t\mathbf{V}\sqrt{\Lambda}\sqrt{\Lambda}\mathbf{V}^t\mathbf{v}_s = \mathbf{v}_s^t\mathbf{V}\Lambda\mathbf{V}^t\mathbf{v}_s = \mathbf{v}_s^t K\mathbf{v}_s = \lambda_s < 0 \tag{1.20}$$

ce qui est contraire à la géométrie de l'espace considéré.

Théorème 1.2. *(Théorème de Mercer)[Mer09, CST00] Soit $\mathcal{U}$ un sous ensemble compact de $\mathbb{R}^n$. Supposons que κ est une fonction symétrique continue de telle façon que l'opérateur intégral $T_\kappa : \mathcal{L}^2(\mathcal{U}) \to \mathcal{L}^2(\mathcal{U})$,*

$$(T_\kappa f)(\cdot) = \int_{\mathcal{U}} \kappa(\boldsymbol{u}, \cdot) f(\boldsymbol{u}) d\boldsymbol{u} \tag{1.21}$$

est positif :

$$\int_{\mathcal{U}} \int_{\mathcal{U}} \kappa(\boldsymbol{u}, \boldsymbol{u'}) f(\boldsymbol{u}) f(\boldsymbol{u'}) d\boldsymbol{u} d\boldsymbol{u'} \geq 0 \tag{1.22}$$

$\forall f \in \mathcal{L}^2(\mathcal{U})$. On peut donc développer $\kappa(\boldsymbol{u}, \boldsymbol{u'})$ en une série uniformément convergente en fonction des fonctions propres de T_κ, $\phi_i \in \mathcal{L}^2(\mathcal{U})$, étant normalisée ($\|\phi_i\|_{\mathcal{L}^2} = 1$), avec les valeurs propres associées positives $\lambda_i \geq 0$.

1.5 Construction d'un noyau

On a vu que la condition nécessaire et suffisante pour qu'une fonction soit un noyau reproduisant est qu'elle soit semi-définie positive. Dans cette section nous présentons quelques aspects de l'ingénierie des noyaux. On peut trouver plus d'exemples et de propriétés dans [Vap95, STC04].

Théorème 1.3. *Soient κ_1 et κ_2 deux noyaux reproduisants de $\mathcal{U} \times \mathcal{U}$ dans $\mathbb{R}$. La fonction $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ est un noyau reproduisant s'il est défini par l'une des expressions ci-dessous pour tout $\boldsymbol{u}, \boldsymbol{u'} \in \mathcal{U}$.*

- $\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \alpha_1\kappa_1(\boldsymbol{u}, \boldsymbol{u'}) + \alpha_2\kappa_2(\boldsymbol{u}, \boldsymbol{u'}) \quad \forall \alpha_1, \alpha_2 \in \mathbb{R}_+$.
- $\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \kappa_1(\boldsymbol{u}, \boldsymbol{u'}) + c \quad \forall c \in \mathbb{R}_+$.
- $\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \kappa_1(\boldsymbol{u}, \boldsymbol{u'})\kappa_2(\boldsymbol{u}, \boldsymbol{u'})$.
- $\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \kappa_1(\boldsymbol{u}, \boldsymbol{u'})^p \quad \forall p \in \mathbb{N}_+$.
- $\kappa(\boldsymbol{u}, \boldsymbol{u'}) = exp(\kappa_1(\boldsymbol{u}, \boldsymbol{u'})/2\sigma^2) \quad \forall \sigma \in \mathbb{R}$.
- $\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \frac{\kappa_1(\boldsymbol{u}, \boldsymbol{u'})}{\sqrt{\kappa_1(\boldsymbol{u}, \boldsymbol{u})\kappa_1(\boldsymbol{u'}, \boldsymbol{u'})}}$.

Proposition 1.2. *Une fonction noyau κ doit satisfaire l'inégalité de Cauchy-Schwartz :*

$$\kappa(\boldsymbol{u},\boldsymbol{u'})^2 = \langle\phi(\boldsymbol{u}),\phi(\boldsymbol{u'})\rangle_{\mathcal{H}}^2 \leq \|\phi(\boldsymbol{u})\|_{\mathcal{H}}^2\|\phi(\boldsymbol{u'})\|_{\mathcal{H}}^2 = \langle\phi(\boldsymbol{u}),\phi(\boldsymbol{u})\rangle_{\mathcal{H}}\langle\phi(\boldsymbol{u'}),\phi(\boldsymbol{u'})\rangle_{\mathcal{H}} = \kappa(\boldsymbol{u},\boldsymbol{u})\kappa(\boldsymbol{u'},\boldsymbol{u'}) \tag{1.23}$$

1.5.1 Noyau normalisé

La dernière proposition du théorème (1.3) exprime la normalisation d'une fonction noyau. La validité du noyau normalisé est révélée dans le corollaire ci-dessous.

Corollaire 1.1. *(Transformation isogone) Etant donné κ_1 un noyau et f une fonction à valeurs réelles, la fonction :*

$$\kappa(\boldsymbol{u},\boldsymbol{u'}) = f(\boldsymbol{u})f(\boldsymbol{u'})\kappa_1(\boldsymbol{u},\boldsymbol{u'}) \tag{1.24}$$

est une fonction noyau.

On parle dans ce cas de noyau (quasi)-isogone. Un noyau isogone conserve les angles entre les vecteurs transformés. En particulier, le noyau normalisé est un noyau isogone avec $f(\mathbf{u}) = 1/\sqrt{\kappa(\mathbf{u},\mathbf{u})}$.

1.5.2 Classes de noyaux

IL existe deux grandes familles de noyaux reproduisants : les noyaux radiaux et les noyaux projectifs.

Le noyau Gaussien est un noyau radial donné par $\kappa(\mathbf{u},\mathbf{u'}) = \exp(-\|\mathbf{u}-\mathbf{u'}\|^2/2\sigma^2)$, où σ est sa bande passante. Il est normalisé, c'est à dire $\kappa(\mathbf{u},\mathbf{u}) = 1$. Le noyau Laplacien $\kappa(\mathbf{u},\mathbf{u'}) = \exp(-\|\mathbf{u}-\mathbf{u'}\|/\sigma)$ est aussi un noyau radial normalisé.

Le noyau projectif le plus connu est le noyau polynomial $\kappa(\mathbf{u},\mathbf{u'}) = \langle\mathbf{u},\mathbf{u'}\rangle^p$ et le noyau polynomial complet $\kappa(\mathbf{u},\mathbf{u'}) = (\langle\mathbf{u},\mathbf{u'}\rangle + c)^p$ avec $c \in \mathbb{R}_+$ et $p \in \mathbb{N}_+$. Cette famille inclut le noyau linéaire $\kappa(\mathbf{u},\mathbf{u'}) = \langle\mathbf{u},\mathbf{u'}\rangle$. Ces deux catégories sont très vastes, mais nous restreignons volontairement celles-ci aux noyaux qui seront utilisés dans cette thèse.

1.6 Modèles non linéaires par méthodes à noyau

Les méthodes à noyau permettent d'étendre aisément les traitements linéaires au cas non linéaire, grâce à ce que l'on appelle *astuce du noyau* (en anglais kernel trick) [ABR64] et au théorème de représentation [Wah90, SHSW00].

1.6.1 Astuce du noyau

Le corollaire suivant est une propriété primordiale des noyaux reproduisants et des méthodes à noyau.

Corollaire 1.2. *(Astuce du noyau) Tout noyau reproduisant κ d'un espace de Hilbert $\mathcal{H}$ peut être écrit sous forme d'un produit scalaire dans cet espace, c'est à dire :*

$$\kappa(\boldsymbol{u}, \boldsymbol{u'}) = \langle \kappa(\boldsymbol{u}, \cdot), \kappa(\boldsymbol{u'}, \cdot) \rangle_{\mathcal{H}} \tag{1.25}$$

$\forall \boldsymbol{u}, \boldsymbol{u'} \in \mathcal{U}$.

Par conséquent, le noyau $\kappa(\mathbf{u}, \mathbf{u'})$ fournit le produit scalaire dans $\mathcal{H}$ des images $\kappa(\mathbf{u}, \cdot)$ et $\kappa(\mathbf{u'}, \cdot)$ de n'importe quelle paire de vecteurs d'entrée $\mathbf{u}, \mathbf{u'} \in \mathcal{U}$, sans qu'il soit nécessaire d'expliciter ces images. Cet astuce est très important car il transforme les méthodes linéaires de traitement d'information en méthodes non-linéaires sous condition que les calculs puissent s'exprimer seulement en fonction du produit scalaire des observations. En d'autres termes, il suffit de remplacer le produit scalaire $\langle \mathbf{u}, \mathbf{u'} \rangle$, qui est un noyau linéaire, par un noyau non linéaire $\kappa(\mathbf{u}, \mathbf{u'})$. Les algorithmes utilisés restent inchangés. Le coût calculatoire dû à l'évaluation des noyaux reste négligeable [Hon07].

A titre indicatif, on prend par exemple la fonction du noyau Gaussien, dans ce cas, l'espace $\mathcal{H}$ est de dimension infinie. Par conséquent, sans la mise en oeuvre de l'astuce du noyau, la détermination du produit scalaire dans cet espace sera impossible.

1.6.2 Théorème de représentation

Le corollaire ci-dessus permet d'utiliser des algorithmes linéaires dans le cas des modèles non-linéaires. En pratique, pour la mise en oeuvre de ce corollaire, il faut le lier au théorème de représentation, ce dernier étant introduit dans les travaux de Kimeldorf et Wahaba dans le domaine de la théorie de l'approximation [KW71, Wah90]. De même, ce théorème a été utilisé dans l'estimation des fonctions, l'identification des systèmes et dans les machines à supports vecteurs (SVM) [SS03, SV99]. La formulation suivante du théorème de représentation ainsi que sa démonstration pour différentes fonctions coût sont dues à Schölkopf et *coll* [SHSW00].

Théorème 1.4. *(Théorème de représentation) Soient $\mathcal{U}$ l'espace des observations, $\mathcal{A} = \{(\boldsymbol{u}_1, y_1) \cdots (\boldsymbol{u}_n, y_n)\}$ un ensemble d'apprentissage donné tel que $\boldsymbol{u}_i \in \mathcal{U}$ représentent les vecteurs d'entrée et $y_i \in \mathbb{R}$ sont les sorties correspondantes, $\mathcal{C}$ une fonction coût arbitraire et $g(\cdot)$ une fonction monotone croissante sur $\mathbb{R}_+$. Soit $\mathcal{H}$ l'espace de Hilbert induit par le noyau κ semi-defini positif sur $\mathcal{U}$. Toute fonction $\psi^* \in \mathcal{H}$ minimisant la fonctionnelle de risque régularisée*

$$\frac{1}{n} \sum_{i=1}^{n} \mathcal{C}(\psi(\boldsymbol{u}_i), y_i) + \zeta g(\|\psi\|_{\mathcal{H}}^2) \tag{1.26}$$

peut s'écrire sous la forme

$$\psi^*(\cdot) = \sum_{j=1}^{n} \alpha_j^* \kappa(\boldsymbol{u}_j, \cdot) \tag{1.27}$$

Démonstration. Soit $\mathcal{H}_n$ le sous-espace de $\mathcal{H}$ engendré par les fonctions $\{\kappa(\mathbf{u}_1,\cdot),\cdots,\kappa(\mathbf{u}_n,\cdot)\}$, c'est-à-dire :

$$\mathcal{H}_n = \left\{\psi \in \mathcal{H} : \psi(\cdot) = \sum_{j=1}^{n} \alpha_j \kappa(\mathbf{u}_j,\cdot), \quad \alpha_1,\ldots,\alpha_n \in \mathbb{R}\right\}$$

Toute fonction ψ de $\mathcal{H}$ admet une et une seule décomposition en deux contributions, l'une appartenant à l'espace $\mathcal{H}_n$ et l'autre qui lui est orthogonale. On peut en effet écrire

$$\psi = \psi^* + \psi^\perp$$

avec $\psi^* \in \mathcal{H}$ et $\psi^\perp$ la composante orthogonale telle que $\langle \psi^\perp, \kappa(\mathbf{u}_i,\cdot)\rangle_{\mathcal{H}} = 0$ pour tout $i = 1,\cdots,n$. La propriété reproduisante permet d'évaluer ψ en $\mathbf{u}_i$ selon l'expression

$$\psi(\mathbf{u}_i) = \langle \psi, \kappa(\mathbf{u}_i,\cdot)\rangle_{\mathcal{H}} = \sum_{j=1}^{n} \alpha_j \langle \kappa(\mathbf{u}_j,\cdot), \kappa(\mathbf{u}_i,\cdot)\rangle_{\mathcal{H}} + \langle \psi^\perp, \kappa(\mathbf{u}_i,\cdot)\rangle_{\mathcal{H}}$$

Le dernier terme s'annule par orthogonalité et on obtient

$$\psi(\mathbf{u}_i) = \sum_{j=1}^{n} \alpha_j \kappa(\mathbf{u}_j, \mathbf{u}_i)$$

La fonctionnelle du risque (1.26) ne dépend pas de la composante orthogonale $\psi^\perp$ vu que les évaluations de ψ en chaque point de l'ensemble d'apprentissage ne dépendent que des coefficients $\{\alpha_1,\ldots,\alpha_n\}$. En minimisant la fonctionnelle du risque, on obtient la classe des fonctions équivalentes dans $\mathcal{H}$ telle que deux fonctions ψ et ϕ appartiennent à la même classe si et seulement si $\psi(\mathbf{u}_i) = \phi(\mathbf{u}_i)$ pour tout $i = 1,\cdots,n$. Il reste à déterminer $\psi^\perp$ pour une classe donnée de fonctions équivalentes afin de minimiser le terme régularisant. En appliquant le théorème de Pythagore à ψ dans $\mathcal{H}$

$$\|\psi\|^2_{\mathcal{H}} = \|\psi^*\|^2_{\mathcal{H}} + \|\psi^\perp\|^2_{\mathcal{H}}$$

le terme régularisant $g(\|\psi\|^2_{\mathcal{H}})$ dans (1.26) s'écrit :

$$g(\|\psi\|^2_{\mathcal{H}}) = g\big(\|\sum_{j=1}^{n} \alpha_j \kappa(\mathbf{u}_j,\cdot)\|^2_{\mathcal{H}}\big) + \|\psi^\perp\|^2_{\mathcal{H}}$$

Comme la fonction $g(\cdot)$ est monotone croissante, la fonction qui minimise l'expression ci-dessus, pour une classe donnée de fonctions équivalentes, doit vérifier $\|\psi^\perp\|^2_{\mathcal{H}} = 0$. □

L'importance de ce théorème réside dans l'existence d'une solution unique à une fonctionnelle de coût régularisée, celle-ci pouvant s'exprimer comme un développement en série finie de fonctions noyau. La minimisation de la fonction coût (1.26) se ramène à un problème d'optimisation à n inconnues, celui

de la détermination des coefficients optimaux $\alpha_1^*, \ldots, \alpha_n^* \in \mathbb{R}$.

1.7 Parcimonie et notion de dictionnaire

Le théorème de représentation, appliqué à un ensemble d'apprentissage, conduit à trouver une solution unique exprimée sous forme d'une série de fonctions noyau. La taille de l'ensemble d'apprentissage est souvent très grande, voire infinie dans le cadre d'une acquisition en temps réel. Dans le cas d'identification en ligne d'un système dynamique non-stationnaire à l'aide des méthodes adaptatives, le développement de la solution conduit à une série dont la taille croît infiniment avec le temps. De ce qui précède, on déduit l'impossibilité de l'utilisation du modèle à noyau dans le cas des applications en ligne et même dans le cas où l'ensemble d'apprentissage est de dimension très grande. Pour surmonter cet obstacle, plusieurs méthodes ont été développées en décomposant le problème en plusieurs sous-problèmes, comme par exemple la technique de segmentation (*chunking*) [Vap82], la décomposition [OFG97, OG99], l'approche de contraction (*shrinking*) ou la sélection de sous-ensembles optimaux [Joa99]. L'actualisation d'un ou de deux coefficients du modèle à chaque itération est proposée respectivement par l'Adatron à noyau basé sur l'algorithme de descente de gradient stochastique [FCC98], et l'approche d'optimisation par minimisation séquentielle [Pla99]. La plupart les méthodes précédentes ont été présentées dans le cadre des SVM, dont la fonction coût (incluant un terme de régularisation) favorise la parcimonie de la solution.

Le principe d'élagage (*pruning*) a été proposé dans plusieurs travaux relatifs aux méthodes à noyau. Le contrôle de complexité varie alors entre l'élimination des fonctions noyau les plus anciennes [VVVS06], l'élagage aléatoire [CCBG07], l'élagage de la fonction avec le plus petit coefficient [DSsS05] et l'élimination des fonctions noyau à faible contribution dans le modèle. De même, Burges a introduit dans [Bur96] une technique avancée pour construire un modèle de faible ordre en appliquant une règle de décision simplifiée pour les SVM.

Toutes les approches précédentes exigent la résolution d'un problème d'optimisation menant à la manipulation et l'inversion d'une matrice de taille identique à celle de la base d'apprentissage. Pour surmonter ce problème, plusieurs techniques de *sparsification* (parcimonie) ont été utilisées pour approcher la matrice de Gram par une matrice de rang inférieur. La sparsification est le processus qui consiste à sélectionner, parmi l'ensemble d'apprentissage, le sous-ensemble *utile* qui entraîne la réduction du modèle. Deux stratégies sont principalement utilisées, l'*élimination* et la *construction* [LPH10]. L'élimination consiste à prendre en considération tous les échantillons d'apprentissage et ensuite, en résolvant le problème d'optimisation, à éliminer les échantillons dont les coefficients deviennent nuls. Des exemples sur l'élimination se trouvent dans [Vap95, EPP00, SLV00, Tip01]. Dans la méthode de construction, les échantillons qui sont pris en considération dans l'identification du modèle sont sélectionnés à chaque itération de l'algorithme d'optimisation. Ces algorithmes utilisent plusieurs techniques de sélection de type glouton (*greedy*) comme dans [SWL03, SB01, QnCR05].

Notion de Dictionnaire

Considérons $\mathcal{U}$ un compact de $\mathbb{R}^l$ et $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ un noyau reproduisant associé à l'espace de Hilbert $\mathcal{H}$ de produit scalaire $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. En raison de la propriété reproduisante, toute fonction $\psi(\cdot)$ de $\mathcal{H}$ peut être évaluée en tout $\mathbf{u}_i \in \mathcal{U}$ tel que $\psi(\mathbf{u}_i) = \langle \psi(\cdot), \kappa(\mathbf{u}_i, \cdot) \rangle_{\mathcal{H}}$ et la fonction noyau $\kappa(\mathbf{u}_i, \cdot)$ est telle que pour tout $\mathbf{u}_j \in \mathcal{U}$ on a $\kappa(\mathbf{u}_i, \mathbf{u}_j)$.

On s'intéresse à trouver la fonction $\psi(\cdot)$ qui minimise une fonctionnelle de risque égale à l'erreur quadratique moyenne qui est un coût lié à la différence entre la sortie désirée d_i et la réponse du modèle $\psi(\mathbf{u}_i)$,

$$\frac{1}{n} \sum_{i=1}^{n} (d_i - \psi(\mathbf{u}_i))^2 + \zeta \|\psi\|_{\mathcal{H}}^2 \tag{1.28}$$

avec ζ le paramètre de régularisation. Selon le théorème de représentation, la fonction optimale $\psi^*(\cdot)$ appartient à l'espace engendré par les fonctions noyau des données d'apprentissage, soit

$$\psi^*(\cdot) = \sum_{j=1}^{n} \alpha_j \kappa(\mathbf{u}_j, \cdot) \tag{1.29}$$

Ce modèle optimal, comme on le sait déjà, n'est pas adapté aux applications en-ligne car l'ordre du développement croît infiniment avec n. La solution de ce problème sera l'application d'une méthode de parcimonie pour réduire l'ordre du modèle en sélectionnant m ($m < n$) fonctions noyau parmi celles disponibles pour définir le modèle, qui devient :

$$\psi_n^*(\cdot) = \sum_{j=1}^{m} \alpha_j \kappa(\mathbf{u}_{w_j}, \cdot) \tag{1.30}$$

où l'indice n indique le temps et $\{\mathbf{u}_{w_j}\}_{j=1\cdots m} \subset \{\mathbf{u}_i\}_{i=1\cdots n}$.

Définition 1.7. *On appelle dictionnaire d'ordre m à l'instant n, désigné par $\mathcal{D}_n = \{\kappa(\boldsymbol{u}_{w_1}, \cdot), \cdots, \kappa(\boldsymbol{u}_{w_m}, \cdot)\}$, l'ensemble des m fonctions noyau les plus pertinentes utilisées pour élaborer le modèle.*

Compte tenu du fait que la solution obtenue est sous-optimale (car elle appartient au sous-espace engendré pas les m fonctions noyau), le but est alors de contrôler le choix des éléments du dictionnaire. La question qui se pose maintenant est de définir la procédure permettant d'aboutir à cet objectif dans le cas d'apprentissage en-ligne.

Tout d'abord il faut choisir un critère de parcimonie. A l'instant $n+1$, et à l'arrivée d'une nouvelle observation $\mathbf{u}_{n+1}$, une décision doit être faite s'il faut introduire la fonction $\kappa(\mathbf{u}_{n+1}, \cdot)$ au dictionnaire ou non. On a deux cas possibles :

1. si $\kappa(\mathbf{u}_{n+1}, \cdot)$ ne satisfait pas le critère de parcimonie, elle ne sera pas introduite dans le dictionnaire. Le dictionnaire reste inchangé.

2. si $\kappa(\mathbf{u}_{n+1}, \cdot)$ satisfait le critère de parcimonie, elle sera ajoutée au dictionnaire suivant l'une des deux stratégies :
   – dans le cas d'une sparcification par élimination, l'ordre du dictionnaire reste égal à m en éliminant une autre fonction noyau du dictionnaire (par exemple celle la plus ancienne, ou la moins influente dans le modèle, ou encore celle qui a le coefficient le plus petit).
   – dans le cas d'une sparcification par construction, l'ordre du dictionnaire est augmenté d'une unité (l'ordre devient $m + 1$).

Dans la suite on développe les critères de parcimonie les plus connus et leurs propriétés. Nous arrivons naturellement à celui que nous utilisons dans les chapitres suivants, le critère de cohérence.

1.8 Critères de parcimonie

De ce qui précède, on peut déduire que les critères de parcimonie sont d'une grande importance vu qu'ils permettent l'application du théorème de représentation pour l'identification en-ligne des systèmes. De plus, le choix des fonctions noyau qui doivent être introduites dans le dictionnaire est crucial car la qualité du modèle prédit est en jeu. Dans ce paragraphe on donne les principes et les caractéristiques de quelques critères de parcimonie.

Soit $\mathcal{U}$ un compact de $\mathbb{R}^l$ et $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ un noyau reproduisant de norme unité associé à l'espace de Hilbert $\mathcal{H}$ et soit, à l'instant n, $\mathcal{D}_n = \{\kappa(\mathbf{u}_{w_1}, \cdot), \cdots, \kappa(\mathbf{u}_{w_m}, \cdot)\}$ le dictionnaire d'ordre m dont les fonctions noyau sont linéairement indépendantes et forment ainsi une base d'un sous espace $\mathcal{H}_m$ de dimension m de $\mathcal{H}$ ($\mathcal{H}_m \subset \mathcal{H}$).

L'objectif est de trouver une fonction $\psi(\cdot)$ qui minimise une fonctionnelle de risque de la forme :

$$\frac{1}{n}\sum_{i=1}^{n} \mathcal{C}(\psi(\mathbf{u}_i), y_i) + \zeta\|\psi\|^2_{\mathcal{H}} \tag{1.31}$$

avec $\mathcal{C}$ une fonction coût arbitraire, y_i la réponse désirée du modèle et ζ un paramètre de régularisation. D'après le théorème de représentation et grâce à l'application d'un critère de parcimonie, et pour une entrée $\mathbf{u}_n \in \mathcal{U}$ à l'instant n, la fonction cherchée est de la forme :

$$\psi_n(\mathbf{u}_n) = \sum_{j=1}^{m} \alpha_j \kappa(\mathbf{u}_{w_j}, \mathbf{u}_n) \tag{1.32}$$

Le vecteur optimal des coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^t$ est obtenu par la résolution du le problème d'optimisation de la fonctionnelle du risque (1.31) en utilisant un algorithme adaptatif.

A l'instant $n + 1$, une nouvelle entrée $\mathbf{u}_{n+1}$ est présentée à l'entrée du modèle. Le critère de parcimonie permet de déterminer quand $\mathbf{u}_{n+1}$ doit être introduite ou non dans le dictionnaire.

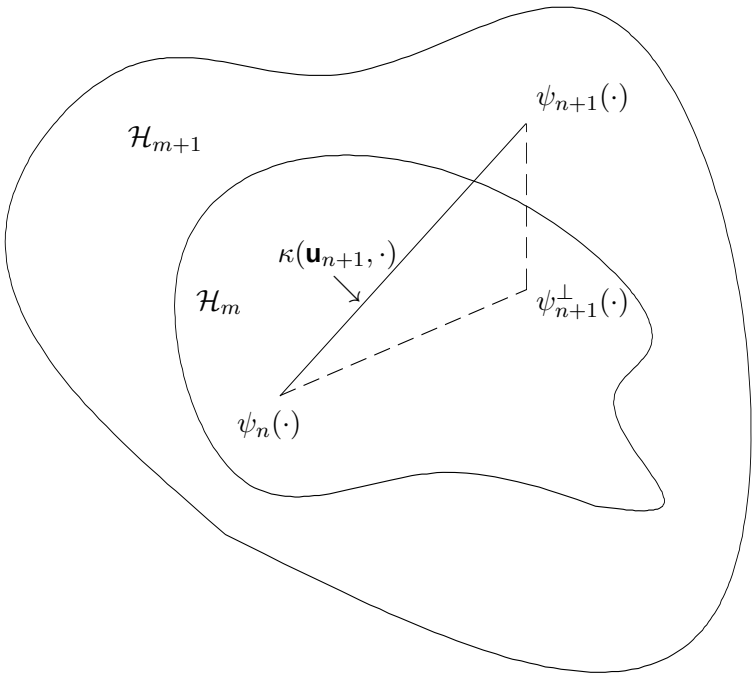

FIGURE *1.1: Illustration de la méthode des projections orthogonales*

1.8.1 Critère de projection orthogonale

La méthode des projections orthogonales a été introduite par Dodd et coll. [DH02, DKH03, DMH03, DNH+05]. L'idée de base de ce critère est la suivante. A l'instant $n+1$ et lors de l'arrivée d'une nouvelle observation $\mathbf{u}_{n+1}$, on effectue la projection de la fonction $\psi_{n+1}(\cdot)$ sur $\mathcal{H}_m$ engendré par les m fonctions noyau $\{\kappa(\mathbf{u}_{w_i},\cdot)\}_{i=1\cdots m}$ comme indiqué dans la figure (1.1). Soient $\psi^{\perp}_{n+1}(\cdot)$ la projection orthogonale de $\psi_{n+1}(\cdot)$ sur $\mathcal{H}_m$ et γ une constante positive fixée qui détermine le niveau de sparsification du dictionnaire. Si la condition suivante est vérifiée

$$\|\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\|^2_{\mathcal{H}} > \gamma \tag{1.33}$$

alors la nouvelle fonction noyau $\kappa(\mathbf{u}_{n+1},\cdot)$ est introduite dans le dictionnaire $\mathcal{D}_{n+1}$ et on aura $\kappa(\mathbf{u}_{w_{m+1}},\cdot) = \kappa(\mathbf{u}_{n+1},\cdot)$ avec un modèle $\psi_{n+1}(\cdot)$. Si la condition (1.33) n'est pas vérifiée, la fonction $\kappa(\mathbf{u}_{n+1},\cdot)$ n'est pas introduite dans le dictionnaire et ce dernier reste inchangé c'est à dire $\mathcal{D}_{n+1} = \mathcal{D}_n$ et le modèle est $\psi^{\perp}_{n+1}(\cdot)$.

Pour vérifier si la condition (1.33) est satisfaite ou non, on doit tout d'abord déterminer $\psi^{\perp}_{n+1}(\cdot)$. Puisque $\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot) \perp \mathcal{H}_m$, on a :

$$\langle \kappa(\mathbf{u}_{w_i},\cdot), \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} = 0 \qquad i = 1\cdots m \tag{1.34}$$

D'après la propriété $\psi_{n+1}(\mathbf{u}) = \langle \psi_{n+1}(\cdot), \kappa(\mathbf{u}, \cdot) \rangle_{\mathcal{H}}$ on a :

$$\psi_{n+1}(\mathbf{u}_{w_i}) - \psi_{n+1}^{\perp}(\mathbf{u}_{w_i}) = 0 \qquad i = 1 \cdots m \tag{1.35}$$

On a donc

$$\psi_{n+1}(\mathbf{u}_{w_i}) = \psi_{n+1}^{\perp}(\mathbf{u}_{w_i}) \qquad i = 1 \cdots m \tag{1.36}$$

Or $\psi_{n+1}(\cdot)$ est obtenue lors de l'introduction d'un nouvel élément dans le dictionnaire qui devient de taille $m+1$, donc :

$$\psi_{n+1}(\cdot) = \sum_{j=1}^{m+1} \alpha_j \kappa(\mathbf{u}_{w_j}, \cdot) \tag{1.37}$$

à noter que le vecteur des coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{m+1})^t$ est obtenu en optimisant la fonctionnelle de coût, et comme $\psi_{n+1}^{\perp}(\cdot) \in \mathcal{H}_m$, elle peut être écrite sous la forme :

$$\psi_{n+1}^{\perp}(\cdot) = \sum_{l=1}^{m} \beta_l \kappa(\mathbf{u}_{w_l}, \cdot) \tag{1.38}$$

avec $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_m)^t$ le vecteur des m paramètres inconnus à déterminer. De (1.36) on déduit :

$$\sum_{l=1}^{m} \beta_l \kappa(\mathbf{u}_{w_l}, \mathbf{u}_{w_i}) = \sum_{j=1}^{m+1} \alpha_j \kappa(\mathbf{u}_{w_j}, \mathbf{u}_{w_i}) \qquad i = 1 \cdots m \tag{1.39}$$

En posant $\sum_{j=1}^{m+1} \alpha_j \kappa(\mathbf{u}_{w_j}, \mathbf{u}_{w_i}) = c_i$, on a :

$$\sum_{l=1}^{m} \beta_l \kappa(\mathbf{u}_{w_l}, \mathbf{u}_{w_i}) = c_i \qquad i = 1 \cdots m \tag{1.40}$$

qui est un système de m équations à m inconnus. En posant $\boldsymbol{c} = (c_1, \ldots, c_m)^t$, la solution du problème est :

$$\boldsymbol{\beta} = \mathbf{K}_n^{-1} \boldsymbol{c} \tag{1.41}$$

où $\mathbf{K}_n$ est la matrice de Gram des éléments du dictionnaire $\mathcal{D}_n$

$$\mathbf{K}_n = \begin{pmatrix} \kappa(\mathbf{u}_{w_1}, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{w_m}, \mathbf{u}_{w_1}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{u}_{w_1}, \mathbf{u}_{w_m}) & \cdots & \kappa(\mathbf{u}_{w_m}, \mathbf{u}_{w_m}) \end{pmatrix}$$

sachant que $\boldsymbol{c} = \mathbf{Q}_{n+1}\boldsymbol{\alpha}$ où $\mathbf{Q}_{n+1}$ est une matrice $m \times (m+1)$ telle que :

$$\mathbf{Q}_{n+1} = \begin{pmatrix} \kappa(\mathbf{u}_{w_1}, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{w_{m+1}}, \mathbf{u}_{w_1}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{u}_{w_1}, \mathbf{u}_{w_m}) & \cdots & \kappa(\mathbf{u}_{w_{m+1}}, \mathbf{u}_{w_m}) \end{pmatrix}$$

Après la détermination de $\psi^{\perp}_{n+1}(\cdot)$, il reste à évaluer $\|\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\|^2$ pour décider si la fonction noyau $\kappa(\mathbf{u}_{w_{m+1}}, \cdot)$ doit être introduite dans le dictionnaire ou non.

$$\begin{aligned} \|\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\|^2 &= \langle \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot), \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} \\ &= \langle \psi_{n+1}(\cdot), \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} - \langle\langle \psi^{\perp}_{n+1}(\cdot), \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} \end{aligned}$$

Or $\left(\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\right) \perp \psi^{\perp}_{n+1}(\cdot)$, donc $\langle \psi^{\perp}_{n+1}(\cdot), \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} = 0$, par conséquent

$$\begin{aligned} \|\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\|^2 &= \langle \psi_{n+1}(\cdot), \psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} \\ &= \langle \psi_{n+1}(\cdot), \psi_{n+1}(\cdot)\rangle_{\mathcal{H}} - \langle \psi_{n+1}(\cdot), \psi^{\perp}_{n+1}(\cdot)\rangle_{\mathcal{H}} \end{aligned} \tag{1.42}$$

En combinant les expressions (1.37), (1.38) et (1.42), on obtient

$$\begin{aligned} \|\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\|^2 &= \left\langle \sum_{i=1}^{m+1} \alpha_i \kappa(\mathbf{u}_{w_i}, \cdot), \sum_{j=1}^{m+1} \alpha_j \kappa(\mathbf{u}_{w_j}, \cdot) \right\rangle_{\mathcal{H}} - \left\langle \sum_{i=1}^{m+1} \alpha_i \kappa(\mathbf{u}_{w_i}, \cdot), \sum_{j=1}^{m} \beta_j \kappa(\mathbf{u}_{w_j}, \cdot) \right\rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{m+1}\sum_{j=1}^{m+1} \alpha_i \alpha_j \kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_j}) - \sum_{i=1}^{m+1}\sum_{j=1}^{m} \alpha_i \beta_j \kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_j}) \end{aligned} \tag{1.43}$$

La réécriture de (1.43) conduit à :

$$\|\psi_{n+1}(\cdot) - \psi^{\perp}_{n+1}(\cdot)\|^2 = \boldsymbol{\alpha}^t \begin{bmatrix} \mathbf{K}_n & \mathbf{k}_n(\mathbf{u}_{n+1}) \\ \mathbf{k}_n^t(\mathbf{u}_{n+1}) & \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{n+1}) \end{bmatrix} \boldsymbol{\alpha} - \boldsymbol{\alpha}^t \begin{bmatrix} \mathbf{K}_n \\ \mathbf{k}_n^t(\mathbf{u}_{n+1}) \end{bmatrix} \boldsymbol{\beta} \tag{1.44}$$

où $\mathbf{k}_n(\mathbf{u}_{n+1}) = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$. A chaque instant et en comparant l'expression (1.44) à une valeur positive γ, on decide si on doit introduire ou non l'observation en question dans le dictionnaire. Notons que, lors de l'introduction d'un nouveau élément dans le dictionnaire, pour trouver le vecteur des coefficients $\boldsymbol{\beta}$ il faut déterminer l'inverse de la matrice $\mathbf{K}_n$ après concaténation. Ceci peut être fait itérativement en utilisant le lemme d'inversion matricielle suivant :

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^t & c \end{bmatrix}^{-1} = \frac{1}{d} \begin{bmatrix} d\boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{b}^t\boldsymbol{A}^{-1} & \boldsymbol{A}^{-1}\boldsymbol{b} \\ -\boldsymbol{b}^t\boldsymbol{A}^{-1} & 1 \end{bmatrix} \tag{1.45}$$

où $d = c - \boldsymbol{b}^t\boldsymbol{A}^{-1}\boldsymbol{b}$.

1.8.2 Critère de dépendance linéaire

Ce critère a été développé par Engel et coll [EMM02, EMM04]. L'idée de base de ce critère est que, à un instant donné, lors de la présentation d'une nouvelle observation à l'entrée, si la fonction noyau correspondante à cette observation est *approximativement* linéairement dépendante des fonctions noyau du dictionnaire, elle n'est pas introduite dans ce dernier. Dans ce cas les coefficients du modèle seront mis à jour en optimisant la fonctionnelle de coût. Si la fonction noyau de la nouvelle observation n'est pas *approximativement* linéairement dépendante des fonctions du dictionnaire, elle est introduite dans le dictionnaire.

A l'instant $n+1$, pour que la nouvelle fonction noyau $\kappa(\mathbf{u}_{n+1},\cdot)$ ne soit pas ajoutée au dictionnaire $\mathcal{D}_n$, il faut que la condition d'approximation de dépendance linéaire (Approximate Linear Dependence - ALD) soit satisfaite :

$$\delta_{n+1} \stackrel{def}{=} \min_{\boldsymbol{\alpha}} \left\| \sum_{j=1}^{m} \alpha_j \kappa(\mathbf{u}_{w_j},\cdot) - \kappa(\mathbf{u}_{n+1},\cdot) \right\|^2 \leq \tau \tag{1.46}$$

où τ est un paramètre de précision qui détermine le degré de la parcimonie du dictionnaire. Si la condition (1.46) est satisfaite, ceci implique que $\kappa(\mathbf{u}_{n+1},\cdot)$ peut être approchée par une combinaison linéaire des éléments du dictionnaire $\mathcal{D}_n$ avec une erreur quadratique limitée par τ. Le vecteur optimal des coefficients $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_m)^t$ est obtenu en minimisant le terme de gauche de l'inégalité (1.46).

$$\begin{aligned} \delta_{n+1} &= \min_{\boldsymbol{\alpha}} \left\{ \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j\kappa(\mathbf{u}_{w_i},\mathbf{u}_{w_j}) - 2\sum_{i=1}^{m}\alpha_i\kappa(\mathbf{u}_{w_i},\mathbf{u}_{n+1}) + \kappa(\mathbf{u}_{n+1},\mathbf{u}_{n+1}) \right\} \\ &= \min_{\boldsymbol{\alpha}} \left\{ \boldsymbol{\alpha}^t\mathbf{K}_n\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^t\mathbf{k}_n(\mathbf{u}_{n+1}) + \kappa_{n+1,n+1} \right\} \end{aligned} \tag{1.47}$$

où $\mathbf{K}_n$ est la matrice de Gram des m éléments du dictionnaire $\mathcal{D}_n$ dont l'élément (i,j) est $\kappa(\mathbf{u}_{w_i},\mathbf{u}_{w_j})$, $\mathbf{k}_n(\mathbf{u}_{n+1})$ est un vecteur tel que $\mathbf{k}_n(\mathbf{u}_{n+1}) = (\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_1}),\cdots,\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_m}))^t$ et $\kappa_{n+1,n+1} = \kappa(\mathbf{u}_{n+1},\mathbf{u}_{n+1})$. Par la résolution de (1.47) on obtient le vecteur $\boldsymbol{\alpha}$ et la condition pour l'approximation de dépendance linéaire :

$$\boldsymbol{\alpha} = \mathbf{K}_n^{-1}\mathbf{k}_n(\mathbf{u}_{n+1}) \tag{1.48}$$

et

$$\delta_{n+1} = \kappa_{n+1,n+1} - \mathbf{k}_n^t(\mathbf{u}_{n+1})\boldsymbol{\alpha} \tag{1.49}$$

Si $\delta_{n+1} > \tau$, le dictionnaire doit être augmenté en y introduisant la fonction $\kappa(\mathbf{u}_{n+1},\cdot)$ qui sera désigné par $\kappa(\mathbf{u}_{w_{m+1}},\cdot)$ et l'ordre du dictionnaire devient $m+1$. En d'autres termes $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{\kappa(\mathbf{u}_{n+1},\cdot)\}$. En utilisant le dictionnaire étendu, $\kappa(\mathbf{u}_{n+1},\cdot)$ sera représentée.Cette approche produit un modèle réduit avec une faible erreur d'approximation mais l'évaluation du critère nécessite l'inversion de la matrice de Gram du dictionnaire, ce qui peut augmenter la complexité calculatoire, bien que cette dernière peut être réduite en inversant cette matrice itérativement à chaque fois qu'un élément est ajouté au dictionnaire. A noter qu'en choisissant τ suffisamment petit, on peut réduire l'erreur d'approximation

de la dépendance linéaire d'une façon significative, au détriment d'une augmentation du coût de calcul liée à la croissance de la taille du dictionnaire.

1.8.3 Critère de cohérence

La cohérence est une grandeur qui caractérise aussi la parcimonie d'un dictionnaire. Elle désigne la plus grande corrélation entre les éléments du dictionnaire, ou mutuellement entre les éléments de deux dictionnaires [Hon07].

L'idée initiale a été introduite dans [MZ93] et a été reprise dans [DH01, DE02]. Ce critère a été utilisé en traitement du signal dans [GMS03, TGMS03, GV06]. La cohérence a été explicitement utilisée dans le cadre du filtrage adaptatif et de l'identification des systèmes dans [HR07, HRB07, RH07, Hon07, RBH09].

Soit un dictionnaire $\mathcal{D}_n = \{\kappa(\mathbf{u}_{w_1}, \cdot), \cdots, \kappa(\mathbf{u}_{w_m}, \cdot)\}$ d'ordre m, la cohérence du dictionnaire est définie par :

$$\begin{aligned} \mu &= \max_{i \neq j} |\langle \kappa(\mathbf{u}_{w_i}, \cdot), \kappa(\mathbf{u}_{w_j}, \cdot) \rangle_{\mathcal{H}}| \qquad i, j = 1, \cdots, m \\ &= \max_{i \neq j} |\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_j})| \qquad i, j = 1, \cdots, m \end{aligned} \tag{1.50}$$

Cette grandeur correspond au plus grand élément, en valeur absolue, hors diagonale, de la matrice de Gram du dictionnaire. Géométriquement, la cohérence représente le plus petit angle entre deux fonctions noyau (associées aux éléments du dictionnaire) dans $\mathcal{H}$. De ce qui précède, la cohérence est nulle pour un ensemble de fonctions orthonormées et vaut 1 pour un ensemble qui contient au moins deux fonctions identiques. Le dictionnaire est dit incohérent si μ est faible. Il faut signaler qu'un dictionnaire de cohérence μ est formé de fonctions noyau qui vérifient le critère d'approximation linéaire avec [Hon07]

$$\tau = 1 - \sqrt{\frac{(m-1)\mu_0^2}{1-(m-2)\mu_0}} \tag{1.51}$$

Le critère de parcimonie basé sur la cohérence est appelé critère de cohérence. En choisissant un seuil μ_0 pour la cohérence tel que $\mu_0 \in [0, 1[$, à l'instant $n+1$ quand une nouvelle observation $\mathbf{u}_{n+1}$ se présente à l'entrée du modèle, la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ est introduite dans le dictionnaire si la condition suivante est vérifiée :

$$\max_{i=1,\cdots,m} |\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{n+1})| \leq \mu_0 \tag{1.52}$$

De la condition (1.52) on peut déduire que le choix du seuil μ_0 détermine la cohérence du dictionnaire et le niveau de parcimonie du modèle résultant. En conséquence la taille du dictionnaire est directement liée à la valeur de μ_0.

Une conséquence importante du critère de cohérence est que son utilisation conduit au fait que la taille du dictionnaire reste finie lorsque le temps tend vers l'infini, ce qui est une propriété tout à fait souhaitable et que nous démontrons ici.

Proposition 1.3. *Soit $\mathcal{U}$ un sous-espace compact d'un espace de Banach, et soit $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ un noyau reproduisant. Pour toute série temporelle $\{\boldsymbol{u}_i\}_{i=1}^{\infty}$ et $\mu_0 \in [0, 1[$, le dictionnaire résultant en appliquant le critère de cohérence (1.52) est de taille finie.*

Démonstration. Le sous-espace $\mathcal{U}$ étant compact et la continuité de la fonction noyau produit un ensemble $\{\kappa(\mathbf{u}_i, \cdot)\}_{\mathbf{u}\in\mathcal{U}}$. Il existe alors un ensemble de boules de rayons non-nuls, définies selon la norme $\mathcal{L}^2$ pouvant couvrir ces fonctions noyau. Pour toute paire de fonctions noyau du dictionnaire de cohérence μ_0 on a :

$$\begin{aligned} \|\kappa(\mathbf{u}_{w_i}, \cdot) - \kappa(\mathbf{u}_{w_j}, \cdot)\|^2_{\mathcal{H}} &= \kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i}) - 2\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i}) + \kappa(\mathbf{u}_{w_j}, \mathbf{u}_{w_j}) \\ &= 2 - 2\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i}) \\ &\geq 2(1 - \mu_0) \end{aligned} \tag{1.53}$$

De cette borne (1.53) on peut déduire que le nombre de boules est fini et donc que la taille du dictionnaire reste nécessairement finie quand le temps n tend vers l'infini. □

Ce résultat important justifie l'utilisation de ce critère pour l'identification en-ligne des systèmes. Dans ce qui suit, on va utiliser ce critère avec les algorithmes d'identification en-ligne des modèles non-linéaires en raison de son faible coût calculatoire et de la propriété précédente.

1.9 Algorithmes d'identification adaptatifs non-linéaires

L'importance de l'utilisation des méthodes à noyau pour identifier des systèmes non-linéaires réside dans le fait que le modèle estimé sera écrit sous forme d'une combinaison linéaire de fonctions noyau. Le modèle est mis à jour à chaque itération en optimisant une fonction coût choisie préalablement et, en général, liée à l'erreur entre la réponse désirée et la sortie du modèle. La figure (1.2) illustre l'identification des systèmes en utilisant des algorithmes adaptatifs, où $\mathbf{u}_n$ est l'entrée à l'instant n, y_n la sortie du modèle, d_n la sortie désirée et $e_n = d_n - y_n$ représente l'erreur entre la sortie désirée et la réponse du modèle.

A chaque instant n, la procédure d'apprentissage est réalisée en deux phases consécutives. En premier lieu on applique le critère de parcimonie pour décider si la nouvelle observation doit être introduite dans le dictionnaire. Dans la deuxième phase, on procède à la mise à jour des coefficients du modèle en minimisant la fonctionnelle de coût.

Une grande variété d'algorithmes adaptatifs existe [Say03]. Comme étudié souvent dans la littérature, on distingue ici les deux catégories : les algorithmes des moindres carrés récursifs et les algorithmes de gradient stochastique. Dans la suite, on présente les versions à noyau, comme préconisé dans [Hon07, LPH10]

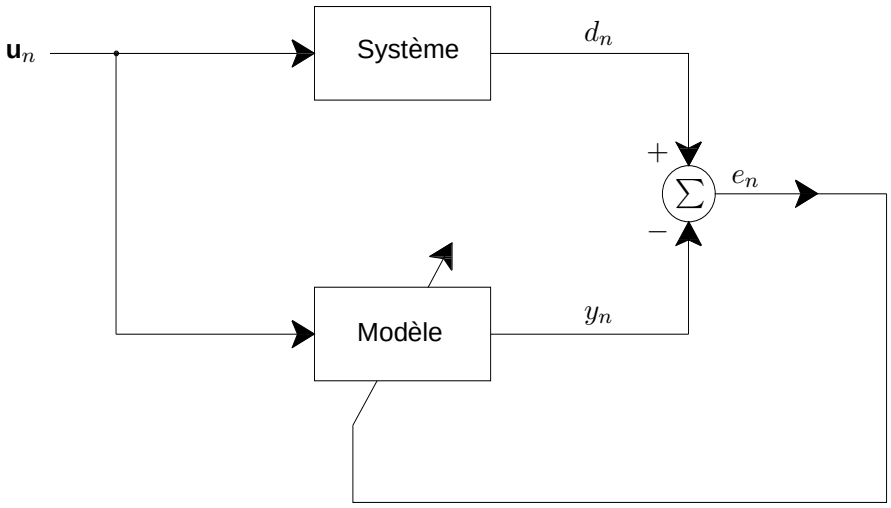

FIGURE *1.2: Illustration pour l'identification des systèmes en utilisant des algorithmes adaptatifs.*

1.9.1 Algorithme de moindres carrés récursif à noyau (KRLS)

Cet algorithme, qui appartient à la première catégorie des algorithmes adaptatifs (algorithmes des moindres carrés récursifs), cherche à minimiser la somme des erreurs quadratiques instantanées avec la possibilité de négliger les erreurs correspondantes aux données les plus anciennes (pondération). L'algorithme KRLS (Kernel Recursive Least Squares) a été largement étudié dans [EMM04, HRB07, LPH10].

Le problème original d'optimisation s'écrit :

$$\min_{\psi\in\mathcal{H}} \sum_{i=1}^{n} (d_i - \psi(\mathbf{u}_i))^2 + \zeta\|\psi\|^2_{\mathcal{H}} \tag{1.54}$$

où ζ est un paramètre positif de régularisation, d_i la réponse désirée du modèle à l'instant i et $\psi(\mathbf{u}_i)$ désigne la sortie correspondante du modèle à l'observation $\mathbf{u}_i$. En introduisant le facteur d'oubli $\theta \in]0, 1]$ et pour une résolution récursive du problème, la nouvelle formulation est :

$$\min_{\psi\in\mathcal{H}} \sum_{i=1}^{n} \theta^{n-i}(d_i - \psi(\mathbf{u}_i))^2 + \zeta\theta^n\|\psi\|^2_{\mathcal{H}} \tag{1.55}$$

La résolution d'un tel problème d'optimisation pour l'identification en ligne d'un modèle est impossible lorsque $n \to \infty$. Mais en appliquant un critère de parcimonie (tel que le critère de cohérence) pour obtenir un dictionnaire $\mathcal{D}_n = \{\kappa(\mathbf{u}_{w_1}, \cdot), \cdots, \kappa(\mathbf{u}_{w_m}, \cdot)\}$ et en s'inspirant du théorème de représentation, le modèle à l'instant n est :

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_{n,j}\kappa(\mathbf{u}_{w_j}, \cdot) \tag{1.56}$$

où $\boldsymbol{\alpha}_n = (\alpha_{n,1}, \cdots, \alpha_{n,m})^t$ est le vecteur des coefficients du modèle à l'instant n. En introduisant (1.56) dans (1.55), la solution optimale du problème dual pour trouver $\boldsymbol{\alpha}_n$ est :

$$\boldsymbol{\alpha}_n = \arg\min_{\boldsymbol{\alpha}} \big(\mathbf{d}_n - \mathbf{H}_n\boldsymbol{\alpha}\big)^t \boldsymbol{\Theta}_n \big(\mathbf{d}_n - \mathbf{H}_n\boldsymbol{\alpha}\big) + \zeta\theta^n \boldsymbol{\alpha}^t \mathbf{K}_n \boldsymbol{\alpha} \tag{1.57}$$

avec $\mathbf{K}_n$ la matrice de Gram des éléments du dictionnaire de taille $m \times m$ dont le $(i,j)^{\text{ème}}$ terme est $\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_j})$, $\boldsymbol{\Theta}_n$ une matrice diagonale de taille $n \times n$ dont le $(i,i)^{\text{ème}}$ élément est θ^{n-i}, $\mathbf{H}_n$ une matrice de taille $n \times m$ dont le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_i, \mathbf{u}_{w_j})$ et $\mathbf{d}_n$ est un vecteur de taille $n \times 1$ qui représente les réponses désirées jusqu'à l'instant n, $\mathbf{d}_n = (d_1, \cdots, d_n)^t$.

En posant $\mathbf{P}_n = \big(\mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{H}_n + \zeta\theta^n \mathbf{K}_n\big)^{-1}$ et supposant que $\mathbf{P}_n$ existe, la solution du problème (1.57) est :

$$\boldsymbol{\alpha}_n = \mathbf{P}_n \mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{d}_n \tag{1.58}$$

A l'instant $n+1$, une nouvelle observation $\mathbf{u}_{n+1}$ se présente à l'entrée du modèle et l'algorithme se déroule de la façon suivante, selon les deux cas ci-dessous :

* Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ ne vérifie pas le critère de parcimonie, le dictionnaire reste inchangé ($\mathcal{D}_{n+1} = \mathcal{D}_n$), en conséquence $\mathbf{K}_n$ reste inchangée, mais la matrice diagonale $\boldsymbol{\Theta}_n$ augmente de taille pour devenir $\boldsymbol{\Theta}_{n+1}$ de taille $n+1 \times n+1$ et dont le $(i,i)^{\text{ème}}$ élément est θ^{n+1-i}, le vecteur $\mathbf{d}_{n+1}$ est donné par $\mathbf{d}_{n+1} = (d_1, \cdots, d_{n+1})^t$ et la matrice $\mathbf{H}_n$ est *augmentée* d'une ligne :

$$\mathbf{H}_{n+1} = \begin{bmatrix} \mathbf{H}_n \\ \mathbf{h}_{n+1}^t \end{bmatrix}$$

avec $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$. Dans ce cas, la mise à jour des coefficients du modèle est réalisée selon :

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \mathbf{P}_{n+1}\mathbf{h}_{n+1}(d_{n+1} - \mathbf{h}_{n+1}^t \boldsymbol{\alpha}_n) \tag{1.59}$$

où

$$\mathbf{P}_{n+1} = \theta^{-1}\left[\mathbf{P}_n - \frac{\theta^{-1}\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1 + \theta^{-1}\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}\right] \tag{1.60}$$

Il est important de noter que le terme $(d_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n)$ correspond à l'erreur a priori à l'instant $n+1$, ce qui implique que $\psi_{n+1}(\mathbf{u}_{n+1}) = \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n$ n'est autre que la réponse du modèle à l'instant $n+1$.

* Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ vérifie le critère de parcimonie, donc elle doit être introduite dans le dictionnaire dont la taille augmente d'une unité pour devenir $\mathcal{D}_{n+1} = \{\kappa(\mathbf{u}_{w_1}, \cdot), \cdots, \kappa(\mathbf{u}_{w_{m+1}}, \cdot)\}$ où $\kappa(\mathbf{u}_{w_{m+1}}, \cdot) = \kappa(\mathbf{u}_{n+1}, \cdot)$. L'incrémentation de l'ordre du dictionnaire conduit à :

$$\mathbf{H}_{n+1} = \begin{bmatrix} \mathbf{H}_n & \mathbf{0}_n \\ \mathbf{h}_{n+1}^t & h_0 \end{bmatrix} \qquad \mathbf{K}_{n+1} = \begin{bmatrix} \mathbf{K}_n & \mathbf{h}_{n+1} \\ \mathbf{h}_{n+1}^t & h_0 \end{bmatrix} \qquad \mathbf{d}_{n+1} = \begin{bmatrix} \mathbf{d}_n \\ d_{n+1} \end{bmatrix}$$

où $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$, $h_0 = \kappa(\mathbf{u}_{w_{m+1}}, \mathbf{u}_{w_{m+1}})$ et $\mathbf{0}_n$ est un vecteur colonne de n zéros. La mise à jour du modèle se fait en deux phases. La première phase consiste à trouver $\tilde{\mathbf{P}}_{n+1}$ et $\tilde{\boldsymbol{\alpha}}_{n+1}$ exactement comme dans (1.59) et (1.60).

$$\tilde{\boldsymbol{\alpha}}_{n+1} = \boldsymbol{\alpha}_n + \tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}(d_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n) \tag{1.61}$$

et

$$\tilde{\mathbf{P}}_{n+1} = \theta^{-1}\left[\mathbf{P}_n - \frac{\theta^{-1}\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1 + \theta^{-1}\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}\right] \tag{1.62}$$

Dans la deuxième phase, le modèle est mis à jour selon la procédure suivante :

$$\boldsymbol{\alpha}_{n+1} = \begin{bmatrix}\tilde{\boldsymbol{\alpha}}_{n+1}\\ 0\end{bmatrix} + \frac{h_0 d_{n+1} - (h_0 + \zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{\alpha}}_{n+1}}{(h_0 + \zeta\theta^{n+1})(h_0 - \mathbf{h}_{n+1}^t\mathbf{q})}\begin{bmatrix}-\mathbf{q}\\ 1\end{bmatrix} \tag{1.63}$$

$$\mathbf{P}_{n+1} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1} & \mathbf{0}_n\\ \mathbf{0}_n^t & 0\end{bmatrix} + \frac{1}{s}\begin{bmatrix}-\mathbf{q}\\ 1\end{bmatrix}[-\mathbf{q}^t \qquad 1] \tag{1.64}$$

où

$$\begin{aligned}\mathbf{q} &= (h_0 + \zeta\theta^{n+1})\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}\\ s &= (h_0 + \zeta\theta^{n+1})(h_0 - \mathbf{h}_{n+1}^t\mathbf{q})\end{aligned}$$

Pour plus de détails du calcul, se référer à l'annexe A.

Avant de terminer la présentation de cet algorithme récursif, il faut signaler, pour des raisons de simplification des expressions, que si l'on néglige le coefficient de régularisation ($\zeta = 0$), les expressions (1.59) et (1.60) deviennent :

$$\begin{aligned}\boldsymbol{\alpha}_{n+1} &= \boldsymbol{\alpha}_n + \frac{\mathbf{P}_n\mathbf{h}_{n+1}}{1 + \mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}(d_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n) &(1.65)\\ \mathbf{P}_{n+1} &= \mathbf{P}_n - \frac{\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1 + \mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}} &(1.66)\end{aligned}$$

ainsi que (1.63) et (1.64) qui deviennent :

$$\begin{aligned}\boldsymbol{\alpha}_{n+1} &= \begin{bmatrix}\tilde{\boldsymbol{\alpha}}_{n+1}\\ 0\end{bmatrix} + \frac{d_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n}{1 - \mathbf{h}_{n+1}^t\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}}\begin{bmatrix}\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}\\ 1/h_0\end{bmatrix} &(1.67)\\ \mathbf{P}_{n+1} &= \begin{bmatrix}\tilde{\mathbf{P}}_{n+1} & \mathbf{0}_n\\ \mathbf{0}_n^t & 0\end{bmatrix} + \frac{1}{1 - \mathbf{h}_{n+1}^t\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}} \times \begin{bmatrix}-\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}\\ 1/h_0\end{bmatrix}\begin{bmatrix}-(\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1})^t & 1/h_0\end{bmatrix} &(1.68)\end{aligned}$$

1.9.2 Algorithme de projection affine à noyau (KAPA)

Cet algorithme fait partie de la catégorie des algorithmes de gradient stochastique. La minimisation de l'erreur quadratique est toujours notre objectif. Le problème d'optimisation, en ne faisant pas apparaître le terme de régularisation, est :

$$\min_{\psi\in\mathcal{H}} \sum_{i=1}^{n} (d_i - \psi(\mathbf{u}_i))^2 \tag{1.69}$$

d_i est la réponse désirée du modèle à l'instant i et $\psi(\mathbf{u}_i)$ est la réponse du modèle à la $i^{\text{ème}}$ observation $\mathbf{u}_i$. En appliquant un critère de parcimonie, à l'instant n nous disposons d'un dictionnaire $\mathcal{D}_n = \{\kappa(\mathbf{u}_{w_j}, \cdot)\}_{j=1,\cdots,m}$ d'ordre m, le modèle à l'instant n est donc :

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_{n,j}\kappa(\mathbf{u}_{w_j}, \cdot) \tag{1.70}$$

où $\boldsymbol{\alpha}_n = (\alpha_{n,1}, \cdots, \alpha_{n,m})^t$ est le vecteur des coefficients optimaux du modèle à l'instant n. La combinaison de (1.70) et (1.69) mène au problème suivant :

$$\boldsymbol{\alpha}_n = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{d}_n - \mathbf{H}_n\boldsymbol{\alpha}\|^2 \tag{1.71}$$

où $\mathbf{d}_n = (d_1, \cdots, d_n)^t$ est le vecteur des réponses désirées jusqu'à l'instant n et $\mathbf{H}_n$ est une matrice $n \times m$ dont le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_i, \mathbf{u}_{w_j})$. En supposant que $(\mathbf{H}_n^t\mathbf{H}_n)^{-1}$ existe, la solution du problème (1.71) est donnée par :

$$\boldsymbol{\alpha}_n = (\mathbf{H}_n^t\mathbf{H}_n)^{-1}\mathbf{H}_n^t\mathbf{d}_n \tag{1.72}$$

L'idée de base de l'algorithme de projection affine (Affine Projection Algoritm - APA) est de prendre en considération les p dernières observations seulement $\{\mathbf{u}_n, \cdots, \mathbf{u}_{n-p+1}\}$, c.à.d. en utilisant une fenêtre glissante de largeur p [Say03]. Ici, p désigne le nombre de collecteurs (manifolds). Dans ce cas, $\mathbf{d}_n$ devient un vecteur colonne de p éléments, $\mathbf{d}_n = (d_n, \cdots, d_{n-p+1})^t$, et $\mathbf{H}_n$ une matrice de taille $p \times m$ dont l'élément (i,j) est $\kappa(\mathbf{u}_{n-i+1}, \mathbf{u}_{w_j})$, donnée par :

$$\mathbf{H}_n = \begin{pmatrix} \kappa(\mathbf{u}_n, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_n, \mathbf{u}_{w_m}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{u}_{n-p+1}, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{n-p+1}, \mathbf{u}_{w_m}) \end{pmatrix}$$

Le vecteur des coefficients optimaux $\boldsymbol{\alpha}_{n+1}$ est déterminé à l'instant $n+1$, en minimisant $\|\boldsymbol{\alpha}_{n+1} -$

$\boldsymbol{\alpha}_n\|^2$ sous la contrainte de nullité des erreurs a posteriori pour les p dernières observations.

$$\min_{\boldsymbol{\alpha}_{n+1}} \|\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n\|^2 \tag{1.73}$$

$$\text{sous contrainte } \mathbf{d}_{n+1} = \mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1} \tag{1.74}$$

où $\mathbf{d}_{n+1} = (d_{n+1}, \cdots, d_{n-p+2})^t$, $\boldsymbol{\alpha}_{n+1} = (\alpha_{n+1,1}, \cdots, \alpha_{n+1,m})^t$ et $\mathbf{H}_{n+1}$ est toujours une matrice $p \times m$ dont l'élément (i, j) est $\kappa(\mathbf{u}_{n-i+2}, \mathbf{u}_{w_j})$. Géométriquement, $\boldsymbol{\alpha}_{n+1}$ est obtenu par la projection de $\boldsymbol{\alpha}_n$ sur l'intersection des p sous-espaces affines $\mathcal{S}_i$ définis par :

$$\mathcal{S}_i = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \mathbf{h}_{n-i+2}^t\boldsymbol{\alpha} - d_{n-i+2} = 0\}_{i=1,\cdots,p} \tag{1.75}$$

avec $\mathbf{h}_{n-i+2} = (\kappa(\mathbf{u}_{n-i+2}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n-i+2}, \mathbf{u}_{w_m}))^t$.

Il est important de remarquer que m indique l'ordre du modèle (qui n'est autre que la taille du dictionnaire) et que la valeur de m peut changer selon le résultat de l'application du critère de parcimonie. En effet, lors de la présence d'une nouvelle observation $\mathbf{u}_{n+1}$, celui-ci conduit à décider si la fonction noyau correspondante à cette observation doit être introduite ou non dans le dictionnaire selon :

* Si $\kappa(\mathbf{u}_{n+1}, \cdot)$ ne vérifie pas le critère de parcimonie, elle est bien représentée par les fonctions existantes du dictionnaire et ne doit pas être introduite dans ce dernier. L'ordre du modèle reste inchangé et la mise à jour des coefficients du modèle est faite comme suit :

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\mathbf{H}_{n+1}^t(\varepsilon\mathbf{I} + \mathbf{H}_{n+1}\mathbf{H}_{n+1}^t)^{-1}(\mathbf{d}_{n+1} - \mathbf{H}_{n+1}\boldsymbol{\alpha}_n) \tag{1.76}$$

  Dans l'expression ci-dessus, η est appelé paramètre de contrôle du pas de convergence et $\varepsilon\mathbf{I}$ est le terme de régularisation.

* Si $\kappa(\mathbf{u}_{n+1}, \cdot)$ vérifie le critère de parcimonie, on l'introduit dans le dictionnaire dont la taille augmente d'une unité ($m = m + 1$) et $\kappa(\mathbf{u}_{w_{m+1}}, \cdot) = \kappa(\mathbf{u}_{n+1}, \cdot)$. La ligne $(\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_{m+1}}))$ est concaténée à la matrice $\mathbf{H}_n$ qui devient de taille $p \times m + 1$. La dimension du vecteur des coefficients du modèle augmente d'une unité et sa mise à jour devient :

$$\boldsymbol{\alpha}_{n+1} = \begin{bmatrix}\boldsymbol{\alpha}_n \\ 0\end{bmatrix} + \eta\mathbf{H}_{n+1}^t(\varepsilon\mathbf{I} + \mathbf{H}_{n+1}\mathbf{H}_{n+1}^t)^{-1}(\mathbf{d}_{n+1} - \mathbf{H}_{n+1}\begin{bmatrix}\boldsymbol{\alpha}_n \\ 0\end{bmatrix}) \tag{1.77}$$

Pour les détails du calcul se référer à l'annexe B.

1.9.3 Algorithme de moindres carrés normalisé à noyau (KNLMS)

La version instantanée de l'algorithme KAPA n'est autre que l'algorithme de moindre carrés normalisés à noyau (Kernel Normalized Least Squares Algorithm - KNLMS). En considérant un nombre de collecteurs $p = 1$ et suivant la satisfaction ou non du critère de parcimonie, on obtient les mises à jour à

l'instant $n+1$:

* Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ n'est pas introduite dans le dictionnaire on a :

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \frac{\eta}{\varepsilon + ||\mathbf{h}_{n+1}||^2}(d_{n+1} - \mathbf{h}_{n+1}^t \boldsymbol{\alpha}_n)\mathbf{h}_{n+1} \tag{1.78}$$

  avec $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_{m+1}}))^t$.

* Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ est introduite dans le dictionnaire, la taille de ce dernier est incrémentée d'une unité ($m = m+1$). La mise à jour devient :

$$\boldsymbol{\alpha}_{n+1} = \begin{bmatrix} \boldsymbol{\alpha}_n \\ 0 \end{bmatrix} + \frac{\eta}{\varepsilon + ||\mathbf{h}_{n+1}||^2} \left(d_{n+1} - \mathbf{h}_{n+1}^t \begin{bmatrix} \boldsymbol{\alpha}_n \\ 0 \end{bmatrix} \right) \mathbf{h}_{n+1} \tag{1.79}$$

CHAPITRE 2

Adaptation du dictionnaire pour les algorithmes de prédiction en ligne

Sommaire

2.1 Introduction

L'identification en-ligne des systèmes non-linéaires reste toujours un sujet important de recherche. Parmi les méthodes les plus connues, les filtres de Volterra et de Wiener [Wie66, Sch06] ou encore des réseaux de neurones [Hay08] ont fait l'objet d'un nombre important de travaux. Chacune de ces méthodes possède ses atouts et ses limitations. Par exemple, dans le cas des filtres de Volterra, le nombre de paramètres à estimer dépend de l'ordre du filtre et de la complexité de celui-ci. Ceci implique potentiellement une grande complexité en termes de nombre de paramètres [RLCS03]. Les Filtres de Wiener ont été définis pour des signaux stationnaires et sont mathématiquement exigeants [Ogu07]. Le point faible des réseaux de neurones multicouches réside dans le choix de la structure du réseau. Par ailleurs, le critère de performance n'est pas convexe en les paramètres du réseau.

Dans le chapitre précédent, on a exposé l'identification des modèles non-linéaires à l'aide des méthodes à noyau. Grâce au théorème de représentation, l'estimation du modèle est obtenue sous la forme d'un développement linéaire de fonctions noyau. Pour les applications en-ligne, l'ordre du modèle est réduit en appliquant un critère de parcimonie. Le critère de cohérence est d'une importance majeure vu sa simplicité calculatoire et son influence sur la réduction de l'ordre du modèle. En conséquence, la fonction noyau d'une nouvelle observation est introduite dans le dictionnaire si elle vérifie la condition de cohérence définie dans (1.52). Une conséquence directe du critère de cohérence est que la taille du dictionnaire reste finie avec le temps. Cependant, lorsqu'on introduit un élément dans le dictionnaire, cet élément reste en permanence, sans changement, même s'il devient moins influent dans l'estimation du modèle. Notre objectif est d'adapter les éléments du dictionnaire, à chaque itération, sans enfreindre la contrainte de cohérence de celui-ci. Le but de l'adaptation est de réduire davantage l'erreur d'approximation et/ou l'ordre du modèle.

On présente tout d'abord l'idée de base de l'adaptation des éléments du dictionnaire. Le principe de l'adaptation est développé sous forme d'une décente de gradient pour différentes classes de fonctions noyau tout en respectant le critère de cohérence qui sera adopté comme critère de parcimonie. L'heuristique adoptée pour le choix du pas de l'algorithme de gradient utilisé pour adapter les éléments du dictionnaire sera exposée de manière détaillée. Les performances obtenues seront présentées sur la base de simulations et de modélisation de signaux réels montrant l'efficacité des techniques proposées.

2.2 Adaptation du dictionnaire

Considérons un problème d'identification en-ligne et soient $\mathbf{u}_n \in \mathcal{U}$ le vecteur d'entrée du modèle à l'instant n et d_n est la sortie désirée correspondante. $\kappa : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}$ est une fonction noyau associée à un RKHS $\mathcal{H}$. L'algorithme des moindres carrés consiste à déterminer la fonction $\psi(\cdot)$ qui minimise une fonction coût qui prend en compte la moyenne des erreurs quadratiques instantanées.

$$\min_{\psi \in \mathcal{H}} \frac{1}{n} \sum_{j=1}^{n} |d_j - \psi(\boldsymbol{u}_j)|^2 + \zeta \|\psi\|^2 \tag{2.1}$$

où $\zeta\|\psi\|^2$ est un terme de régularisation. Le théorème de représentation et la monotonicité de $\|\psi\|^2$ sur $[0, +\infty[$ mènent à une solution optimale $\psi^*(\cdot)$ du problème :

$$\psi^*(.) \equiv \psi_n(.) = \sum_{j=1}^{n} \alpha_j \kappa(., \mathbf{u}_j) \tag{2.2}$$

Les $\alpha_j \in \mathbb{R}$ sont les coefficients du modèle. Puisqu'il s'agit d'une identification en-ligne, l'adaptation du modèle devient de plus en plus difficile en raison de l'augmentation du nombre des observations avec le temps. Une solution est d'utiliser le critère de cohérence. Soit $\mathcal{D}_n = \{\kappa(., \mathbf{u}_{w_j})\}_{j=1}^{m}$ le dictionnaire, de taille m, des fonctions noyau pertinentes sélectionnées parmi toutes les observations jusqu'à l'instant n.

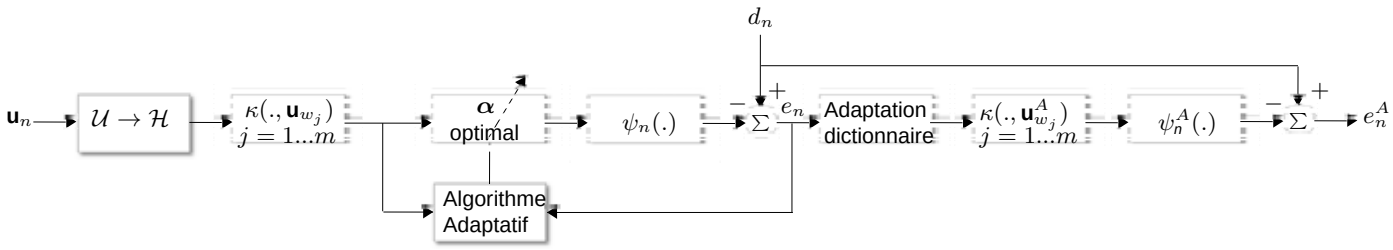

FIGURE *2.1: Algorithme global d'adaptation du dictionnaire.*

Le modèle (2.2) devient :

$$\psi_n(.) = \sum_{j=1}^{m} \alpha_{n,j}\kappa(.,\mathbf{u}_{w_j}) \tag{2.3}$$

$\boldsymbol{\alpha}_n = (\alpha_{n,1}, \cdots, \alpha_{n,m})^t$ le vecteur des coefficients optimaux du modèle à l'instant n.

A l'instant $n+1$, un nouveau vecteur d'entrée $\mathbf{u}_{n+1}$ se présente à l'entrée du modèle. Si la fonction $\kappa(.,\mathbf{u}_{n+1})$ vérifie le critère de cohérence (2.4) elle est introduite dans le dictionnaire dont la taille augmente d'une unité.

$$\max_{\mathbf{u}_{w_j}\in\mathcal{D}_n} |\langle \kappa(.,\mathbf{u}_{n+1}), \kappa(.,\mathbf{u}_{w_j})\rangle_{\mathcal{H}}| = \max_{\mathbf{u}_{w_j}\in\mathcal{D}_n} |\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_j})| \leq \mu_0 \tag{2.4}$$

Le seuil $\mu_0 \in [0,1[$ détermine le niveau de parcimonie du dictionnaire et il contrôle l'ordre du modèle. Il est important de noter que la fonction noyau utilisée dans (2.4) doit être de norme unité ; si ce n'est pas le cas, il faut faire une normalisation en substituant $\kappa(\mathbf{u}_i,\mathbf{u}_j)$ par $\kappa(\mathbf{u}_i,\mathbf{u}_j)/\sqrt{\kappa(\mathbf{u}_i,\mathbf{u}_i)\kappa(\mathbf{u}_j,\mathbf{u}_j)}$.

Définition 2.1. *Un dictionnaire $\mathcal{D}_n$ de taille m est dit μ_0-cohérent si toutes les fonctions noyau qui y appartiennent vérifient la condition :*

$$\max_{i\neq j} |\kappa(\boldsymbol{u}_{w_i},\boldsymbol{u}_{w_j})| \leq \mu_0 \tag{2.5}$$

2.2.1 Principe de l'adaptation du dictionnaire

L'idée de départ pour adapter le dictionnaire est motivée dans la suite : Une fonction noyau, une fois introduite dans le dictionnaire, reste sans aucun changement même si la non-stationarité du modèle à identifier rend la contribution de cette fonction noyau faible pour l'estimation de la sortie courante. Il apparait alors opportun d'adapter les éléments du dictionnaire $\mathcal{D}_n$ pour obtenir $\mathcal{D}^A_n$. A chaque instant n, le processus d'optimisation du (2.1) se fait en 2 phases. La première consiste à trouver le vecteur des coefficients optimaux du modèle $\boldsymbol{\alpha}_n$ en appliquant un algorithme de prédiction en-ligne. La deuxième phase a pour but d'adapter les éléments du dictionnaire sans que ce dernier ne perde sa cohérence. L'adaptation des éléments du dictionnaire a pour but la minimisation de l'erreur quadratique instantanée

e_n^2 où

$$e_n = d_n - \psi_n(\mathbf{u}_n) = d_n - \sum_{i=1}^{m} \alpha_{n,i}\, \kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) \tag{2.6}$$

tout en respectant la contrainte du cohérence du dictionnaire (2.5). L'algorithme d'adaptation du dictionnaire est représenté dans la figure (2.1).

Le $i^{\text{ème}}$ élément du dictionnaire est adapté suivant le principe suivant :

$$\mathbf{u}_{w_i}^A = \mathbf{u}_{w_i} - \nu_n \mathbf{g}_{w_i} \qquad \forall i = 1...m \tag{2.7}$$

$\mathbf{u}_{w_i}^A$ indique l'élément du dictionnaire après adaptation, $\mathbf{g}_{w_i}$ est le gradient de l'erreur quadratique instantanée par rapport à $\mathbf{u}_{w_i}$ et ν_n indique le pas du gradient utilisé pour adapter tous les éléments du dictionnaire. Bien sûr, le choix de ν_n n'est pas arbitraire mais il est soumis à des contraintes strictes permettant de préserver la cohérence du dictionnaire. Ces idées sont détaillées dans les sous-paragraphes suivants qui sont spécifiques au type de fonction noyau choisie.

En utilisant (2.6), le gradient par rapport à l'élément $\mathbf{u}_{w_i}$ du dictionnaire est :

$$\begin{aligned}
\mathbf{g}_{w_i} &= \nabla_{\mathbf{u}_{w_i}}(e_n^2) \\
&= -2\alpha_{n,i}\Big(d_n - \sum_{i=1}^{m} \alpha_{n,i}\, \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\Big) \nabla_{\mathbf{u}_{w_i}} \Big(\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\Big) \\
&= -2e_n\, \alpha_{n,i}\, \nabla_{\mathbf{u}_{w_i}} \Big(\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\Big)
\end{aligned} \tag{2.8}$$

La combinaison de (2.7) et (2.8) donne

$$\mathbf{u}_{w_i}^A = \mathbf{u}_{w_i} + 2\nu_n e_n\, \alpha_{n,i}\, \nabla_{\mathbf{u}_{w_i}} \Big(\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\Big) \qquad \forall i = 1...m \tag{2.9}$$

D'après (2.9), il est clair que l'adaptation des éléments du dictionnaire dépend de la fonction noyau choisie. Pour cette raison, on explore les deux types les plus répandus de fonctions noyau : les fonctions noyau radiales (Radial Basis Functions-RBF) et les fonctions noyau polynomiales.

2.2.2 Cas d'un noyau radial (RBF)

Cette catégorie de fonctions noyau comprend les deux fonctions les plus utilisées : le noyau Gaussien et le noyau Laplacien. Les fonctions noyau qui appartiennent à cette catégorie peuvent être exprimées sous la forme :

$$\kappa(\mathbf{u}_i, \mathbf{u}_j) = f(\|\mathbf{u}_i - \mathbf{u}_j\|^2), \tag{2.10}$$

où $f \in \mathcal{C}^\infty$ ($\mathcal{C}$ est l'espace de Banach des fonctions continues). Une condition suffisante pour que cette fonction noyau soit définie positive est sa monotonicité, c'est à dire :

$$(-1)^k f^{(k)}(r) \geq 0, \qquad \forall r \geq 0 \tag{2.11}$$

où $f^{(k)}$ indique la $k^{\text{ème}}$ dérivée de la fonction $f(.)$ [CS02]. En incluant (2.10) dans (2.8), on obtient :

$$\nabla_{\mathbf{u}_{w_i}} \kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) = -2(\mathbf{u}_n - \mathbf{u}_{w_i})\, f^{(1)}(\|\mathbf{u}_n - \mathbf{u}_{w_i}\|^2) \tag{2.12}$$

Donc, le gradient de l'erreur quadratique instantanée par rapport à $\mathbf{u}_{w_i}$ est

$$\mathbf{g}_{w_i} = 4e_n\, \alpha_{n,i}(\mathbf{u}_n - \mathbf{u}_{w_i})\, f^{(1)}(\|\mathbf{u}_n - \mathbf{u}_{w_i}\|^2) \tag{2.13}$$

La contrainte essentielle que l'on doit respecter lors de l'adaptation des éléments du dictionnaire est la cohérence de ce dernier. En d'autres termes, un dictionnaire μ_0-cohérent avant adaptation doit rester μ_0-cohérent après adaptation, c'est à dire :

$$\max_{i \neq j} |\kappa(\mathbf{u}^A_{w_i}, \mathbf{u}^A_{w_j})| \leq \mu_0 \qquad \forall i, j = 1...m \tag{2.14}$$

Cette contrainte fait apparaître explicitement le type de fonction noyau, c'est pourquoi nous détaillons dans la suite les expressions du gradient et de la contrainte de cohérence pour le noyau Gaussien, d'une part, et le noyau Laplacien, d'autre part.

2.2.2.1 Gradient et condition de cohérence pour un noyau Gaussien

La $i^{\text{ème}}$ fonction noyau du dictionnaire est :

$$\kappa(\cdot, \mathbf{u}_{w_i}) = \exp\Big(-\frac{\|\cdot - \mathbf{u}_{w_i}\|^2}{2\sigma^2}\Big)$$

où σ est la bande passante du noyau. A l'instant n, en prenant en considération la dérivée de cette fonction noyau par rapport à $\mathbf{u}_{w_i}$, le gradient de l'erreur quadratique instantanée définie dans (2.8) et (2.12) est :

$$\mathbf{g}_{w_i} = -2\frac{e_n\, \alpha_{n,i}}{\sigma^2}\, \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\, (\mathbf{u}_n - \mathbf{u}_{w_i}).$$

La condition de cohérence entre deux éléments quelconques du dictionnaire (2.14) s'écrit sous la forme :

$$\|\mathbf{u}^A_{w_i} - \mathbf{u}^A_{w_j}\|^2 \quad \geq \quad -2\sigma^2 \ln(\mu_0) \qquad \forall i \neq j = 1...m \tag{2.15}$$

où on a $-2\sigma^2 \ln(\mu_0) > 0$ car $\mu_0 \in [0, 1[$.

2.2.2.2 Gradient et condition de cohérence pour un noyau Laplacien

Pour Le noyau Laplacien, la $i^{\text{ème}}$ fonction noyau du dictionnaire est :

$$\kappa(\cdot, \mathbf{u}_{w_i}) = \exp\Big(-\frac{\|\cdot - \mathbf{u}_{w_i}\|}{2\sigma^2}\Big)$$

où σ est la bande passante du noyau. Le gradient instantané de l'erreur quadratique par rapport à $\mathbf{u}_{w_i}$ à l'instant n est :

$$\mathbf{g}_{w_i} = -\frac{e_n\, \alpha_{n,i}}{\sigma^2\, \|\mathbf{u}_n - \mathbf{u}_{w_i}\|}\, \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\, (\mathbf{u}_n - \mathbf{u}_{w_i})$$

De cette expression on peut déduire qu'il existe une condition restrictive $\|\mathbf{u}_n - \mathbf{u}_{w_i}\| \neq 0$. Ceci implique que lorsque l'on introduit un nouvel élément dans le dictionnaire, celui-ci ne peut pas être adapté.

La condition de cohérence sur les éléments du dictionnaire dans le cas d'un noyau Laplacien est :

$$\|\mathbf{u}^A_{w_i} - \mathbf{u}^A_{w_j}\|^2 \quad \geq \quad \big(-2\sigma^2 \ln(\mu_0)\big)^2 \tag{2.16}$$

2.2.2.3 Valeurs possibles pour le pas du gradient pour une fonction noyau radiale

Lors de l'adaptation des éléments du dictionnaire, il ne faut pas enfreindre la cohérence de ce dernier. Pour toute paire d'éléments du dictionnaire et en se basant sur (2.7) on a :

$$\mathbf{u}^A_{w_i} - \mathbf{u}^A_{w_j} = \mathbf{u}_{w_i} - \mathbf{u}_{w_j} - \nu_n(\mathbf{g}_{w_i} - \mathbf{g}_{w_j}) = \delta\mathbf{u} - \nu_n\delta\mathbf{g} \qquad \forall i, j = 1...m \tag{2.17}$$

où $\delta\mathbf{u} = \mathbf{u}_{w_i} - \mathbf{u}_{w_j}$ et $\delta\mathbf{g} = \mathbf{g}_{w_i} - \mathbf{g}_{w_j}$. D'une façon générale, pour une fonction noyau radiale, en combinant (2.10) et (2.17) on obtient :

$$f(\|\delta\mathbf{u} - \nu_n\, \delta\mathbf{g}\|^2) \leq \mu_0. \tag{2.18}$$

Le développement en série de Taylor du terme de gauche de cette inégalité autour de $\nu_n \sim 0$ donne :

$$f(\|\delta\mathbf{u} - \nu_n\delta\mathbf{g}\|^2) = f(\|\delta\mathbf{u}\|^2) - 2\nu_n(\delta\mathbf{u}^t\delta\mathbf{g} - \nu_n\|\delta\mathbf{g}\|^2)f^{(1)}(\|\delta\mathbf{u}\|^2) + \mathcal{O}(\nu_n)$$

En utilisant cette approximation, la condition (2.18) devient :

$$\begin{aligned} -\big(2\|\delta\mathbf{g}\|^2\nu_n^2 - 2\nu_n\delta\mathbf{u}^t\delta\mathbf{g}\big)f^{(1)}(\|\delta\mathbf{u}\|^2\big) + \mu_0 - f(\|\delta\mathbf{u}\|^2) \quad &\geq 0 \\ -2\|\delta\mathbf{g}\|^2 f^{(1)}\big(\|\delta\mathbf{u}\|^2\big)\nu_n^2 + 2\delta\mathbf{u}^t\delta\mathbf{g}f^{(1)}(\|\delta\mathbf{u}\|^2\big)\nu_n + \mu_0 - f(\|\delta\mathbf{u}\|^2) \quad &\geq 0 \end{aligned} \tag{2.19}$$

Le discriminant de l'équation du second degré associée est :

$$\Delta = \big(\delta\mathbf{u}^t\delta\mathbf{g}\, f^{(1)}(\|\delta\mathbf{u}\|^2)\big)^2 + 2\|\delta\mathbf{g}\|^2 f^{(1)}(\|\delta\mathbf{u}\|^2)(\mu_0 - f(\|\delta\mathbf{u}\|^2)$$

Si $\Delta < 0$, l'équation du second degré associée à (2.19) ne possède pas de racines et il n'y a pas de contraintes sur le choix du pas de gradient ν_n entre le $i^{\text{ème}}$ et le $j^{\text{ème}}$ élément du dictionnaire. Si $\Delta \geq 0$, alors l'équation du second degré associée à (2.19) possède deux racines $\nu_{i,j-}$ et $\nu_{i,j+}$ données par :

$$\nu_{i,j\pm} = \frac{-\delta\mathbf{u}^t\delta\mathbf{g}\, f^{(1)}(\|\delta\mathbf{u}\|^2) \pm \sqrt{\Delta}}{-\|\delta\mathbf{g}\|^2 f^{(1)}(\|\delta\mathbf{u}\|^2)}$$

Le domaine des valeurs possibles de ν_n est $]-\infty, \nu_{i,j-}] \cup [\nu_{i,j+}, +\infty[$, car le terme de gauche de (2.19) doit être positif :

	$\nu_{i,j-}$		$\nu_{i,j+}$	
$+$	$\vert$	$-$	$\vert$	$+$

Il convient de faire deux remarques importantes :

1. Les deux racines $\nu_{i,j-}$ et $\nu_{i,j+}$ sont de même signe vu que la fonction noyau satisfait la condition de validité (2.11) $f^{(1)}(\|\delta\mathbf{u}\|^2) \leq 0$ et que l'on a toujours $\mu_0 - f(\|\delta\mathbf{u}\|^2) \geq 0$ pour un dictionnaire cohérent à tout instant n.

2. Il est clair que, pour chaque paire d'éléments du dictionnaire ($\mathbf{u}_{w_i}$,$\mathbf{u}_{w_j}$), la valeur $\nu_n = 0$ appartient toujours à l'intervalle des valeurs acceptables $]-\infty, \nu_{i,j-}] \cup [\nu_{i,j+}, +\infty[$ puisque, dans ce cas, il n'y a pas adaptation du dictionnaire et qu'il est μ_0-cohérent.

A noter également que, si l'équation du second degré associée à (2.19) admet une racine double $\nu_{i,j-} = \nu_{i,j+}$, ν_n peut prendre n'importe quelle valeur entre $]-\infty, +\infty[$ y compris $\nu_n = \nu_{i,j-} = \nu_{i,j+}$. Il est aisé d'interpréter géométriquement le type d'intervalle des valeurs acceptables pour ν_n : en adaptant deux éléments du dictionnaire, les deux bornes ($\nu_{i,j-}, \nu_{i,j+}$) doivent être respectées pour éviter le chevauchement des régions d'influence des fonctions noyau. La figure (2.2) donne une illustration en dimension 2 de cette contrainte sur le choix de ν_n.

Nous présentons maintenant l'heuristique de choix d'un pas du gradient convenable pour adapter tous les éléments du dictionnaire. On choisit un pas de référence ν_0 de manière similaire à tous les algorithmes adaptatifs à pas fixe. Après le calcul de $m(m+1)/2$ intervalles $[(\nu_{i,j-}, \nu_{i,j+})]$ entre les m éléments du dictionnaire, ν_n est choisi selon la procédure suivante :

- Si $\max_{i,j} \nu_{i,j+} \leq 0 \Rightarrow \nu_n = \nu_0$
- Si $0 \leq \min_{i,j} \nu_{i,j-} \leq \nu_0 \Rightarrow \nu_n = \min_{i,j} \nu_{i,j-}$
- Si $0 \leq \nu_0 \leq \min_{i,j} \nu_{i,j-} \Rightarrow \nu_n = \nu_0$
- Si $0 \leq \min_{i,j} (\nu_{i,j-})^+ \leq \nu_0 \Rightarrow \nu_n = \min_{i,j} (\nu_{i,j-})^+$
- Si $0 \leq \nu_0 \leq \min_{i,j} (\nu_{i,j-})^+ \Rightarrow \nu_n = \nu_0$

La notation $(\nu_{i,j-})^+$ indique les valeurs positives des $\nu_{i,j-}$. Il faut attirer l'attention sur le fait qu'il faut choisir ν_0 petit. A défaut, les observations qui ont formé le dictionnaire risquent d'être dispersés sur des

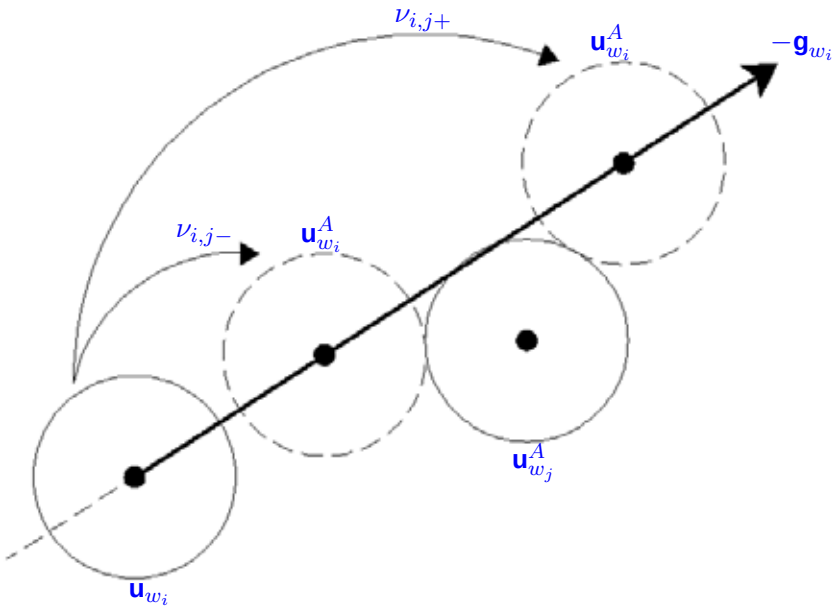

FIGURE *2.2: Illustration en 2D montrant la contrainte sur le choix de $\nu_n \leq \nu_{i,j-}$ ou $\nu_n \geq \nu_{i,j+}$ pour éviter le chevauchement entre les régions d'influence de $\mathbf{u}_{w_i}^A$ et $\mathbf{u}_{w_j}^A$.*

régions inutiles de l'espace des entrées et entraîner une augmentation de la taille du dictionnaire sans réduction de l'erreur quadratique.

Remarque. *On peut ne pas utiliser le développement en série de Taylor, mais on développe l'expression (2.15) (respectivement (2.16)) pour obtenir :*

$$\|(\mathbf{u}_{w_i} - \mathbf{u}_{w_j}) - \nu_{\mathsf{n}} (\mathbf{g}_{w_i} - \mathbf{g}_{w_j})\|^2 \geq (-2\sigma^2 \ln(\mu_0))$$
$$\|\delta\mathbf{u} - \nu_{\mathsf{n}}\, \delta\mathbf{g}\|^2 \geq (-2\sigma^2 \ln(\mu_0))$$

après développement on obtient :

$$\|\delta\mathbf{g}\|^2 \nu_{\mathsf{n}}^2 - 2\,\delta\mathbf{u}^t \delta\mathbf{g}\, \nu_{\mathsf{n}} + \|\delta\mathbf{u}\|^2 + 2\sigma^2 \ln(\mu_0) \geq 0 \tag{2.20}$$

qui est une inéquation quadratique en ν_n. Le discriminant réduit est :

$$\Delta = (\delta\mathbf{u}^t \delta\mathbf{g})^2 - \|\delta\mathbf{g}\|^2 (\|\delta\mathbf{u}\|^2 + 2\sigma^2 \ln(\mu_0)) \tag{2.21}$$

Dans le cas où $\Delta < 0$, il n'y a pas de conditions sur le choix du pas de gradient et si $\Delta \geq 0$, les racines seront :

$$\nu_{\mathsf{i,j}\pm} = \frac{(\delta\mathbf{u}^t \delta\mathbf{g})^2 \pm \sqrt{\Delta}}{\|\delta\mathbf{g}\|^2} \tag{2.22}$$

Le terme $\|\delta\mathbf{u}\|^2 + 2\sigma^2 \ln(\mu_0)$ est toujours positif pour un dictionnaire cohérent avant l'adaptation, ce qui entraîne que les deux racines sont de même signe. Les mêmes discussions faites dans le cas où on a

utilisé le développement de Taylor sont toujours valables ici, et l'algorithme utilisé pour le choix du pas de gradient afin d'adapter tous les éléments du dictionnaire est encore le même.

2.2.3 Cas d'un noyau polynomial

Le noyau polynomial est de la forme :

$$\kappa(\mathbf{u}_i, \mathbf{u}_j) = f(\mathbf{u}_i^t \mathbf{u}_j) = (1 + a\mathbf{u}_i^t \mathbf{u}_j)^\beta \tag{2.23}$$

où $\beta > 0$. La norme de ce noyau n'est pas égale à l'unité, d'où la nécessité le normaliser en remplaçant $\kappa(\mathbf{u}_i, \mathbf{u}_j)$ par

$$\frac{\kappa(\mathbf{u}_i, \mathbf{u}_j)}{\sqrt{\kappa(\mathbf{u}_i, \mathbf{u}_i)}\sqrt{\kappa(\mathbf{u}_j, \mathbf{u}_j)}}$$

Nous effectuons la même démarche que précédemment. Nous présentons d'abord le calcul du gradient de l'erreur quadratique instantanée par rapport aux éléments du dictionnaire.

2.2.3.1 Calcul du gradient d'un noyau polynomial

Puisque la fonction noyau a été normalisée, le modèle réduit (2.3) à l'instant n devient :

$$\psi_n(.) = \sum_{i=1}^{m} \alpha_{n,i} \frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)}\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}} \tag{2.24}$$

En posant

$$\hbar(\mathbf{u}_n, \mathbf{u}_{w_i}) = \frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)}\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}} = \frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{s(\mathbf{u}_n, \mathbf{u}_{w_i})}$$

l'erreur instantanée devient :

$$e_n = d_n - \sum_{i=1}^{m} \alpha_{n,i}\, \hbar(\mathbf{u}_n, \mathbf{u}_{w_i}) \tag{2.25}$$

D'après (2.25), le gradient de e_n^2 par rapport à l'élément $\mathbf{u}_{w_i}$ du dictionnaire est :

$$\mathbf{g}_{w_i} = 2e_n \frac{\partial e_n}{\partial \mathbf{u}_{w_i}} = -2e_n \alpha_{n,i} \frac{\partial \hbar(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} \tag{2.26}$$

Le calcul de $\frac{\partial \hbar(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}}$ se fait de la manière suivante :

$$\frac{\partial \hbar(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} = \frac{\frac{\partial \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} s(\mathbf{u}_n, \mathbf{u}_{w_i}) - \kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) \frac{\partial s(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}}}{\Big(s(\mathbf{u}_n, \mathbf{u}_{w_i})\Big)^2} \tag{2.27}$$

Sachant que $\kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) = (1 + a\mathbf{u}_n^t \mathbf{u}_{w_i})^\beta$,on a :

$$\begin{aligned}\frac{\partial \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} &= a\beta \mathbf{u}_n (1 + a\mathbf{u}_n^t \mathbf{u}_{w_i})^{\beta-1} \\ &= a\beta \mathbf{u}_n \frac{(1 + a\mathbf{u}_n^t \mathbf{u}_{w_i})^\beta}{1 + a\mathbf{u}_n^t \mathbf{u}_{w_i}} \\ &= a\beta \mathbf{u}_n \frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{1 + a\mathbf{u}_n^t \mathbf{u}_{w_i}}\end{aligned} \tag{2.28}$$

$s(\mathbf{u}_n, \mathbf{u}_{w_i}) = \sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)}\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}$ conduit à :

$$\begin{aligned}\frac{\partial s(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} &= \sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)} \frac{\frac{\partial \kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}}}{2\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}} \\ &= \frac{1}{2} \frac{\sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)}}{\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}} \frac{\partial \kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}}\end{aligned} \tag{2.29}$$

or, par similitude avec (2.28), (2.29) se met sous la forme :

$$\frac{\partial s(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} = \frac{1}{2} \frac{\sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)}}{\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}} a\,\beta\, \mathbf{u}_{w_i} \frac{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}{1 + a\mathbf{u}_{w_i}^t \mathbf{u}_{w_i}} \qquad = \frac{a\,\beta}{2} \mathbf{u}_{w_i} \frac{s(\mathbf{u}_n, \mathbf{u}_{w_i})}{1 + a\mathbf{u}_{w_i}^t \mathbf{u}_{w_i}} \tag{2.30}$$

(2.27),(2.28), et (2.30) nous donnent :

$$\frac{\partial h(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} = \frac{a\,\beta\, \mathbf{u}_n \frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{(1+a\mathbf{u}_n^t \mathbf{u}_{w_i})} s(\mathbf{u}_n, \mathbf{u}_{w_i}) - \kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) \frac{a\,\beta}{2} \mathbf{u}_{w_i} \frac{s(\mathbf{u}_n, \mathbf{u}_{w_i})}{(1+a\mathbf{u}_{w_i}^t \mathbf{u}_{w_i})}}{\Big(s(\mathbf{u}_n, \mathbf{u}_{w_i})\Big)^2} \tag{2.31}$$

en simplifiant :

$$\begin{aligned}\frac{\partial h(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} &= a\,\beta\, \frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{s(\mathbf{u}_n, \mathbf{u}_{w_i})} \left(\frac{\mathbf{u}_n}{1 + a\mathbf{u}_n^t \mathbf{u}_{w_i}} - \frac{1}{2} \frac{\mathbf{u}_{w_i}}{1 + a\mathbf{u}_{w_i}^t \mathbf{u}_{w_i}} \right) \\ &= a\,\beta\, \hbar(\mathbf{u}_n, \mathbf{u}_{w_i}) \left(\frac{\mathbf{u}_n}{1 + a\mathbf{u}_n^t \mathbf{u}_{w_i}} - \frac{1}{2} \frac{\mathbf{u}_{w_i}}{1 + a\mathbf{u}_{w_i}^t \mathbf{u}_{w_i}} \right)\end{aligned} \tag{2.32}$$

posons $\kappa^*(\mathbf{u}_i, \mathbf{u}_j) = (1 + a\mathbf{u}_i^t \mathbf{u}_j)$ c'est à dire $\kappa^*(\mathbf{u}_i, \mathbf{u}_j) = \kappa(\mathbf{u}_i, \mathbf{u}_j)$ pour $\beta = 1$, (2.32) devient :

$$\frac{\partial \hbar(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} = a\,\beta\, \hbar(\mathbf{u}_n, \mathbf{u}_{w_i}) \left(\frac{\mathbf{u}_n}{\kappa^*(\mathbf{u}_n, \mathbf{u}_{w_i})} - \frac{1}{2} \frac{\mathbf{u}_{w_i}}{\kappa^*(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})} \right) \tag{2.33}$$

en utilisant (2.33), l'équation (2.26) devient :

$$\mathbf{g}_{w_i} = -2\,a\,\beta\, e_n\, \alpha_{n,i}\, \hbar(\mathbf{u}_n, \mathbf{u}_{w_i}) \left(\frac{\mathbf{u}_n}{\kappa^*(\mathbf{u}_n, \mathbf{u}_{w_i})} - \frac{1}{2} \frac{\mathbf{u}_{w_i}}{\kappa^*(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})} \right) \tag{2.34}$$

ou encore

$$\mathbf{g}_{w_i} = -2\,a\,\beta\,e_n\,\alpha_{n,i}\,\frac{\kappa(\mathbf{u}_n,\mathbf{u}_{w_i})}{\sqrt{\kappa(\mathbf{u}_n,\mathbf{u}_n)}\sqrt{\kappa(\mathbf{u}_{w_i},\mathbf{u}_{w_i})}}\left(\frac{\mathbf{u}_n}{\kappa^*(\mathbf{u}_n,\mathbf{u}_{w_i})} - \frac{1}{2}\frac{\mathbf{u}_{w_i}}{\kappa^*(\mathbf{u}_{w_i},\mathbf{u}_{w_i})}\right) \tag{2.35}$$

2.2.3.2 Condition sur les valeurs possibles du pas du gradient

La condition de cohérence (2.14) appliquée à la fonction noyau polynomiale donne :

$$\frac{|(1+a\mathbf{u}_{w_i}^t\mathbf{u}_{w_j})^\beta|}{\sqrt{(1+a\mathbf{u}_{w_i}^t\mathbf{u}_{w_i})^\beta}\sqrt{(1+a\mathbf{u}_{w_j}^t\mathbf{u}_{w_j})^\beta}} \leq \mu_0 \tag{2.36}$$

Pour éliminer la condition sur la parité de β et sur le signe du produit scalaire $\mathbf{u}_{w_i}^t\mathbf{u}_{w_j}$, on élève au carré l'expression précédente qui devient :

$$\frac{(1+a\mathbf{u}_{w_i}^t\mathbf{u}_{w_j})^{2\beta}}{(1+a\mathbf{u}_{w_i}^t\mathbf{u}_{w_i})^\beta\,(1+a\mathbf{u}_{w_j}^t\mathbf{u}_{w_j})^\beta} \leq \mu_0^2$$

La condition sur la cohérence du dictionnaire à l'instant n devient :

$$\frac{(1+a\mathbf{u}_{w_i}^t\mathbf{u}_{w_j})^2}{(1+a\|\mathbf{u}_{w_i}\|^2)\,(1+a\|\mathbf{u}_{w_j}\|^2)} \leq (\mu_0)^{2/\beta} \tag{2.37}$$

Dans la suite, et comme c'est le cas dans la plupart des articles issus de la littérature, nous considérons que $a = 1$. Ceci n'affectera pas les conditions sur le choix du pas du gradient pour adapter le dictionnaire. En considérant (2.37), on déduit qu'il est difficile d'adapter simultanément tous les éléments du dictionnaire à chaque itération tout en respectant la contrainte de cohérence. En conséquence, nous proposons d'adapter un seul élément du dictionnaire à chaque itération. Le choix de cet élément sera présenté ultérieurement.

Supposons que l'élément à adapter soit $\mathbf{u}_{w_i}$, alors l'introduction de (2.7) dans (2.37) donne :

$$\begin{aligned}
&\frac{(1+\langle\mathbf{u}_{w_i}^A,\mathbf{u}_{w_j}\rangle)^2}{(1+\|\mathbf{u}_{w_i}^A\|^2)\,(1+\|\mathbf{u}_{w_j}\|^2)} \leq (\mu_0)^{2/\beta} \quad \forall j\neq i,\ j=1...m\\
&\frac{(1+\langle\mathbf{u}_{w_i}-\nu_n\mathbf{g}_{w_i},\mathbf{u}_{w_j}\rangle)^2}{(1+\|\mathbf{u}_{w_i}^A\|^2)\,(1+\|\mathbf{u}_{w_j}\|^2)} \leq (\mu_0)^{2/\beta}
\end{aligned} \tag{2.38}$$

or,

$$\langle\mathbf{u}_{w_i}-\nu_n\mathbf{g}_{w_i},\mathbf{u}_{w_j}\rangle = \langle\mathbf{u}_{w_i},\mathbf{u}_{w_j}\rangle - \nu_n\,\langle\mathbf{g}_{w_i},\mathbf{u}_{w_j}\rangle = \mathbf{u}_{w_i}^t\mathbf{u}_{w_j} - \nu_n\,\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}$$

L'expression du numérateur de (2.38), après développement est :

$$\begin{aligned}(1+\langle \mathbf{u}_{w_i}^A, \mathbf{u}_{w_j}\rangle)^2 &= (1+\mathbf{u}_{w_i}^t\mathbf{u}_{w_j})^2 - 2\,\nu_n\,\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}(1+\mathbf{u}_{w_i}^t\mathbf{u}_{w_j}) + \nu_n^2\,(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i})^2\\ &= \kappa^{*^2}(\mathbf{u}_{w_i},\mathbf{u}_{w_j}) - 2\,\nu_n\,\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\kappa^*(\mathbf{u}_{w_i},\mathbf{u}_{w_j}) + \nu_n^2\,(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i})^2\\ &= \kappa_{ij}^{*^2} - 2\,\nu_n\,\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\kappa_{ij}^* + \nu_n^2\,(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i})^2 \end{aligned} \tag{2.39}$$

où $\kappa_{ij}^* = \kappa^*(\mathbf{u}_{w_i},\mathbf{u}_{w_j}) = (1+\mathbf{u}_{w_i}^t\mathbf{u}_{w_j})$. D'autre part,

$$\|\mathbf{u}_{w_i}^A\|^2 = \|\mathbf{u}_{w_i}\|^2 - 2\nu_n\mathbf{u}_{w_i}^t\mathbf{g}_{w_i} + \nu_n^2\,\|\mathbf{g}_{w_i}\|^2$$

Le dénominateur de (2.38) prend la forme :

$$\begin{aligned}(1+\|\mathbf{u}_{w_i}^A\|^2)\,(1+\|\mathbf{u}_{w_j}\|^2) &= (1+\|\mathbf{u}_{w_j}\|^2)(1+\|\mathbf{u}_{w_i}\|^2 - 2\nu_n\mathbf{u}_{w_i}^t\mathbf{g}_{w_i} + \nu_n^2\,\|\mathbf{g}_{w_i}\|^2)\\ &= \kappa_{jj}^*(\kappa_{ii}^* - 2\nu_n\mathbf{u}_{w_i}^t\mathbf{g}_{w_i} + \nu_n^2\,\|\mathbf{g}_{w_i}\|^2)\end{aligned} \tag{2.40}$$

La combinaison de (2.38), (2.39) et (2.40) mène à :

$$\frac{\kappa_{ij}^{*^2} - 2\,\nu_n\,\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\kappa_{ij}^* + \nu_n^2\,\left(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\right)^2}{\kappa_{jj}^*\left(\kappa_{ii}^* - 2\nu_n\mathbf{u}_{w_i}^t\mathbf{g}_{w_i} + \nu_n^2\,\|\mathbf{g}_{w_i}\|^2\right)} \leq (\mu_0)^{2/\beta} \quad \forall j \neq i,\ j = 1...m \tag{2.41}$$

En posant $D = \kappa_{jj}^*\,(\mu_0)^{2/\beta}$, l'expression précédente mène à :

$$\left(\left(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\right)^2 - D\|\mathbf{g}_{w_i}\|^2\right)\nu_n^2 - 2\left(\kappa_{ij}^*\mathbf{u}_{w_j}^t\mathbf{g}_{w_i} - D\mathbf{u}_{w_i}^t\mathbf{g}_{w_i}\right)\nu_n + \left(\kappa_{ij}^{*^2} - D\kappa_{ii}^*\right) \leq 0 \tag{2.42}$$

de la forme :

$$A\nu_n^2 + 2B\nu_n + C \geq 0$$

avec

$$\begin{aligned}A &= D\|\mathbf{g}_{w_i}\|^2 - \left(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\right)^2\\ B &= \kappa_{ij}^*\mathbf{u}_{w_j}^t\mathbf{g}_{w_i} - D\mathbf{u}_{w_i}^t\mathbf{g}_{w_i}\\ C &= D\kappa_{ii}^* - \kappa_{ij}^{*^2}\end{aligned}$$

Le discriminant de l'équation du second degré correspondante est :

$$\begin{aligned}\Delta &= B^2 - AC\\ \Delta &= \left(k_{ij}^*\mathbf{u}_{w_j}^t\mathbf{g}_{w_i} - D\mathbf{u}_{w_i}^t\mathbf{g}_{w_i}\right)^2 - \left(D\|\mathbf{g}_{w_i}\|^2 - \left(\mathbf{u}_{w_j}^t\mathbf{g}_{w_i}\right)^2\right)\left(Dk_{ii}^* - k_{ij}^{*^2}\right)\end{aligned} \tag{2.43}$$

C est toujours ≥ 0 pour un dictionnaire cohérent, et ceci est dû au respect de la contrainte de cohérence

à tout instant n. On a :

$$\frac{k_{ij}}{\sqrt{k_{ii}\,k_{jj}}} \leq \mu_0 \Rightarrow \frac{k_{ij}^{*^2}}{k_{ii}^{*}k_{jj}^{*}} \leq (\mu_0)^{2/\beta} \Rightarrow C \geq 0$$

Le signe du discriminant ainsi que l'intervalle des valeurs acceptables de ν_n dépend donc du signe de A. Etudions maintenant les différentes possibilités suivant le signe de A :

* si $A < 0 \Rightarrow \Delta \geq 0$ car dans ce cas on a $-AC \geq 0$ (voir (2.43))
  - Dans le cas où $\Delta > 0$, les deux racines sont de signes opposés et on a :

$$\begin{aligned}\nu_{i,j}\pm &= \frac{-B \pm \sqrt{\Delta}}{A} \\ &= \frac{-\left(k_{ij}^{*}\mathbf{u}_{w_j}^{t}\mathbf{g}_{w_i} - D\mathbf{u}_{w_i}^{t}\mathbf{g}_{w_i}\right) \pm \sqrt{\Delta}}{D\|\mathbf{g}_{w_i}\|^2 - \left(\mathbf{u}_{w_j}^{t}\mathbf{g}_{w_i}\right)^2}\end{aligned}$$

    avec :

$$\begin{array}{ccccccc} & & \nu_{i,j-} & & \nu_{i,j+} & & \\ \hline - & & | & + & | & & - \end{array}$$

    $\nu_n \in [\nu_{i,j-}, \nu_{i,j+}]$ tel que $\nu_{i,j-} \leq 0 \leq \nu_{i,j+}$ car $0 \in$ toujours à l'intervalle des valeurs admissibles de ν_n.

  - Dans le cas où $\Delta = 0$, on a une racine double $\nu_{i,j-} = \nu_{i,j+} = \frac{-B}{A}$. La contrainte de positivité du polynôme n'est jamais vérifiée ici, mais le polynôme peut être égal a zéro pour cette racine double qui n'est autre que $\nu_n = \nu_{i,j-} = \nu_{i,j+} = 0$.
    Preuve :
    Pour que Δ soit égal à zéro, il faut que $B^2 = AC$. sachant que $A < 0$ et $C \geq 0$, nous obtenons une condition nécessaire : il faut que $C = 0$ et donc que $B = 0$. En conséquence,

$$A\nu_n^2 = 0 \text{ et } A < 0 \Rightarrow \nu_n = 0$$

    $C = 0$ indique que $\frac{k_{ij}}{\sqrt{k_{ii}\,k_{jj}}} = \mu_0$, c'est à dire que les deux zones d'influence des fonctions noyaux des éléments $\mathbf{u}_{w_i}$ et $\mathbf{u}_{w_j}$ se touchent. $B = 0$ implique que $k_{ij}^{*}\mathbf{u}_{w_j}^{t}\mathbf{g}_{w_i} = D\mathbf{u}_{w_i}^{t}\mathbf{g}_{w_i} \Rightarrow \mathbf{u}_{w_j}^{t}\mathbf{g}_{w_i} = k_{ij}^{*}k_{ii}^{*}\mathbf{u}_{w_i}^{t}\mathbf{g}_{w_i} \Rightarrow \mathbf{u}_{w_j}^{t}\mathbf{g}_{w_i} = C^{te} * \mathbf{u}_{w_i}^{t}\mathbf{g}_{w_i}$. donc les deux vecteurs $\mathbf{u}_{w_i}$ et $\mathbf{u}_{w_j}$ sont colinéaires.

* si $A > 0$
  - si $\Delta > 0$, on a deux racines qui sont de même signe car $AC \geq 0$ avec :

$$\begin{array}{ccccccc} & & \nu_{i,j-} & & \nu_{i,j+} & & \\ \hline + & & | & - & | & & + \end{array}$$

    et $\nu_n \in]-\infty, \nu_{i,j-}] \cup [\nu_{i,j+}, +\infty[$. Là encore, $\nu_n = 0$ est toujours dans l'intervalle des valeurs acceptables.

  - si $\Delta < 0$, il n'existe pas de racines de l'équation du second degré mais le signe du polynome

est toujours positif, donc il n'existe pas de contraintes pour le choix de ν_n, qui peut appartenir à $]-\infty,+\infty[$.

– si $\Delta = 0 \Rightarrow$ on a une racine double $\nu_{i,j-} = \nu_{i,j+} = \frac{-B}{A}$. Dans ce cas encore il n'existe pas de contraintes sur le choix de ν_n.

* si $A = 0$, on a $\nu_{i,j} \geq \frac{-C}{2B}$. Donc il faut choisir $\nu_n \geq 0$ si $\frac{-C}{2B} \leq 0$ ou $\nu_n \geq \frac{-C}{2B}$ si $\frac{-C}{2B} \geq 0$.

Ce calcul doit être fait pour un $\mathbf{u}_{w_i}$ fixé avec les $m-1$ éléments restants $\mathbf{u}_{w_j}$ $i \neq j = 1...m$. On obtient ainsi $m-1$ valeurs possibles pour ν_n que l'on appelle $\{\nu_{ij_n}\}_{i\neq j=1\cdots m}$. La valeur de ν_n choisie pour adapter l'élément $\mathbf{u}_{w_i}$ du dictionnaire sera $\nu_n = \min\{\nu_{ij_n}\}_{i\neq j=1\cdots m}$.

A chaque instant, avant l'arrivée d'une nouvelle observation, on adapte un seul élément du dictionnaire, cet élément étant choisi préalablement. L'heuristique de choix du pas de gradient ν_n d'adaptation de cet élément est présentée ici. On fixe un pas de gradient initial $\nu_0 \geq 0$. Pour un élément $\mathbf{u}_{w_i}$ fixé et tout autre élément $\mathbf{u}_{w_j}$, on a l'un des cas ci-dessous :

1. si $A > 0$ et $\Delta > 0$
   – si $\nu_{i,j-} \leq \nu_{i,j+} \leq 0 \Rightarrow \nu_{ij_n} = \nu_0$
   – si $0 \leq \nu_{i,j-} \leq \nu_0 \leq \nu_{i,j+} \Rightarrow \nu_{ij_n} = \nu_{i,j-}$
   – si $0 \leq \nu_{i,j-} \leq \nu_{i,j+} \leq \nu_0 \Rightarrow \nu_{ij_n} = \nu_0$ ou bien $\nu_{ij_n} = \nu_{i,j-}$
   – si $0 \leq \nu_0 \leq \nu_{i,j-} \leq \nu_{i,j+} \Rightarrow \nu_{ij_n} = \nu_0$ ou bien $\nu_{ij_n} = \nu_{i,j-}$
2. si $A > 0$ et $\Delta \leq 0$
   – on prend toujours $\nu_{ij_n} = \nu_0$
3. si $A < 0$ et $\Delta > 0$
   dans ce cas on a toujours $\nu_{i,j-} \leq 0 \leq \nu_{i,j+}$
   – si $\nu_{i,j+} \leq \nu_0 \Rightarrow \nu_{ij_n} = \nu_{i,j+}$
   – si $\nu_0 \leq \nu_{i,j+} \Rightarrow \nu_{ij_n} = \nu_0$ ou bien $\nu_{ij_n} = \nu_{i,j+}$
4. si $A < 0$ et $\Delta = 0$
   – on prend toujours $\nu_{ij_n} = 0$
5. si $A = 0$
   – si $\frac{-C}{2B} \leq 0 \Rightarrow \nu_{ij_n} = \nu_0$
   – si $0 \leq \frac{-C}{2B} \leq \nu_0 \Rightarrow \nu_{ij_n} = \frac{-C}{2B}$
   – si $\nu_0 \leq \frac{-C}{2B} \Rightarrow \nu_{ij_n} = \nu_0$

2.2.3.3 Choix de l'élément à adapter

On sait d'après ce qui précède comment choisir le pas du gradient pour adapter un seul élément du dictionnaire à chaque itération. Le problème est maintenant de procéder au choix de l'élément à adapter. Plusieurs approches sont possibles, nous étudions deux possibilités reposant sur des idées très simples.

La première approche consiste à choisir l'élément qui possède le plus petit coefficient $\alpha_{n,i}$ (en valeur absolue) dans le modèle $\psi_n(\cdot)$ défini en (2.24). La deuxième consiste à calculer les gradients de l'erreur quadratique instantanée par rapport à chaque élément du dictionnaire et à retenir celui qui donne la plus grande norme du gradient. L'avantage de la première méthode est sa simplicité, la deuxième étant plus coûteuse du point de vue calculatoire. Cette dernière présente toutefois la possibilité d'aller dans la direction locale de la plus grande réduction de l'erreur. Les résultats des simulations pour les deux approches sont présentés dans les paragraphes suivants.

2.3 Expérimentations

Dans cette section, nous montrons l'efficacité de l'adaptation du dictionnaire. Nous effectuons des simulations pour comparer les courbes d'apprentissage sans et avec adaptation. Les algorithmes adaptatifs utilisés sont l'algorithme de moindres carrés récursifs à noyau (KRLS), l'algorithme de projection affine à noyau (KAPA) et l'algorithme de moindres carrés normalisés à noyau (KNLMS), avec des séries temporelles types utilisées dans plusieurs références et avec des séries réelles. La taille finale du dictionnaire, tout comme l'erreur quadratique moyenne normalisée (Normalized Mean Squared Error - NMSE), sont systématiquement utilisées comme critères de performance pour réaliser les comparaisons. Rappelons que l'erreur quadratique moyenne normalisée est calculée en utilisant les derniers 500 échantillons du signal utilisé selon la relation :

$$\text{NMSE} = E\left\{\frac{\sum_{n=N-500}^{N}\left(d_n - \psi_n(\mathbf{u}_n)\right)^2}{\sum_{n=N-500}^{N} d_n^2}\right\} \tag{2.44}$$

où N représente la taille de l'échantillon.

2.3.1 Prédiction de la série temporelle de Henon

Cette première expérimentation a pour but de prédire la série temporelle de Henon :

$$d_n = 1 - \gamma_1 d_{n-1}^2 + \gamma_2 d_{n-2}$$

où $d_0 = -0.3$, $d_1 = 0$, $\gamma_1 = 1.4$, et $\gamma_2 = 0.3$. Le modèle considéré est de la forme $\psi_n(d_{n-1}, d_{n-2})$, étudié sur une suite de 2000 échantillons temporels. Les résultats obtenus avec adaptation sont comparés à ceux obtenus dans [HRB07, Hon07] sans adaptation. Pour cela, on a utilisé l'algorithme KRLS

et le noyau Gaussien avec une bande passante $\sigma = 0.35$ et les mêmes paramètres de réglage. Le même seuil de cohérence est utilisé avec $\mu_0 = 0.6$. Pour l'adaptation on a choisi, après une recherche grossière permettant avoir de bons résultats, le pas du gradient de référence $\nu_0 = 0.05$. Pour les résultats de simulation, voir figures (2.3) et (2.4).

La taille du dictionnaire a diminué de 17 éléments sans adaptation à 16 éléments avec adaptation. Malgré une faible réduction de cette taille, on observe un gain important sur l'erreur quadratique moyenne normalisée. Celle-ci chute de 0.01036 sans adaptation à 0.001628 avec adaptation ce qui correspond à une diminution de 84.29%.

Cette première simulation atteste du fait que l'on peut diminuer la taille du dictionnaire et réduire l'erreur quadratique en même temps.

2.3.2 Prédiction de la série logistique

L'objectif de cette simulation est de prédire la série logistique (Logistic map) qui est un modèle déterministe chaotique non linéaire de la forme :

$$d_{n+1} = \rho d_n(1 - d_n)$$

avec la condition initiale $d_0 = 0.2027$ et $\rho = 4$. Un bruit blanc Gaussien d'écart-type 0.25 est ajouté à la sortie désirée. Rappelons que dans le cas de l'utilisation des fonctions noyaux radiales, on adapte tous les éléments du dictionnaire avec le même pas du gradient. L'algorithme KAPA est utilisé avec un seuil de cohérence fixé à $\mu_0 = 0.3$. Une série de taille $N = 3000$ échantillons est utilisée et l'erreur quadratique instantanée est moyennée pour les derniers 500 échantillons de la série.

Dans un premier temps, le noyau Gaussien de bande passante $\sigma = 0.418$ est utilisé en conjonction avec un pas de gradient de référence $\nu_0 = 0.2$. La taille finale du dictionnaire est $m = 2$ éléments avec et sans adaptation. L'erreur quadratique moyenne normalisée passe de 0.0060538 sans adaptation à 0.0016778 avec adaptation ce qui correspond à une diminution de 72.285%. Les courbes d'apprentissage sont montrées sur la figure (2.5).

Le deuxième essai est réalisé avec le noyau Laplacien de bande passante $\sigma = 0.5$ et le pas de gradient de référence utilisé est $\nu_0 = 0.01$. On aboutit à la même taille finale du dictionnaire $m = 2$. Sans adaptation, l'erreur quadratique moyenne normalisée est de 0.01601, mais avec adaptation cette erreur devient 0.009911 ce qui correspond à une diminution de 38.09%. La figure (2.6) montre les résultats obtenus.

Le choix des paramètres n'est pas aisé. Nous avons procédé par essai et erreur avec différentes valeurs de la bande passante des noyaux tout en changeant le seuil de cohérence et le pas référence du gradient. Par exemple, si l'on fixe la bande passante du noyau Laplacien à $\sigma = 0.418$, la taille du dictionnaire devient $m = 3$ avec adaptation au lieu de $m = 2$, mais le gain en erreur quadratique moyenne est important. En effet, l'erreur est de 0.036957, sans adaptation, et devient 0.0076908 avec adaptation. Donc pour une augmentation de la taille finale du dictionnaire de 50% l'erreur quadratique

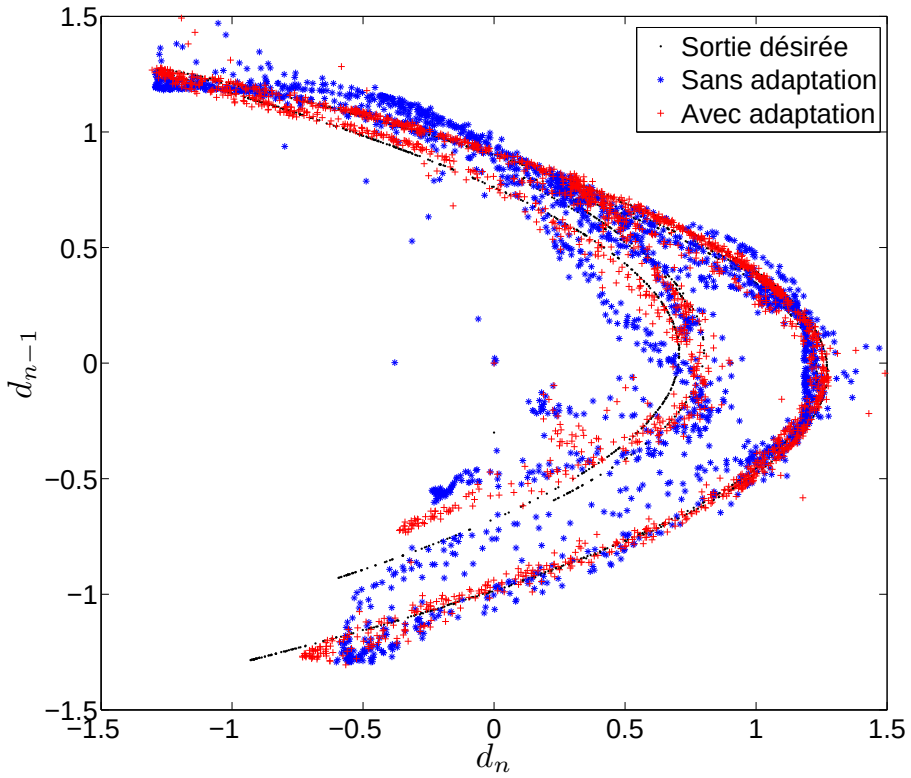

FIGURE *2.3: Modélisation avec KRLS de la série de Hénon avec un noyau Gaussien* $\sigma = 0.35$, $\mu_0 = 0.6$ *et* $\nu_0 = 0.05$

moyenne décroît de 79.19%. Ce gain est visible sur la figure (2.7).

2.3.3 Simulations avec l'algorithme KAPA pour un signal de référence issu de la littérature

Le signal de référence (benchmark) utilisé dans cette partie a été étudié dans [RBH09, Hon07] pour tester le critère de cohérence. Il est de la forme :

$$\begin{cases} s_n = 1.1 \exp(-|s_{n-1}|) + u_n \\ d_n = s_n^2 \end{cases}$$

où u_n et d_n sont respectivement l'entrée et la sortie désirée du modèle à l'instant n. Les données ont été générées à partir de la condition initiale $s_0 = 0.5$. Les entrées u_n suivent une distribution Gaussienne de moyenne nulle et d'écart type 0.25. La sortie du modèle est noyée dans un bruit blanc Gaussien de moyenne nulle et de variance unité. L'algorithme KAPA est utilisé dans ces simulations avec les paramètres suivants : nombre de collecteurs $p = 3$, paramètre du contrôle de pas $\eta = 0.01$ et facteur de régularisation $\varepsilon = 0.07$. Pour les détails relatifs à ces paramètres voir l'annexe B. A noter que les valeurs des paramètres sont identiques à celles utilisées dans [RBH09], ceci pour pouvoir comparer les résultats.

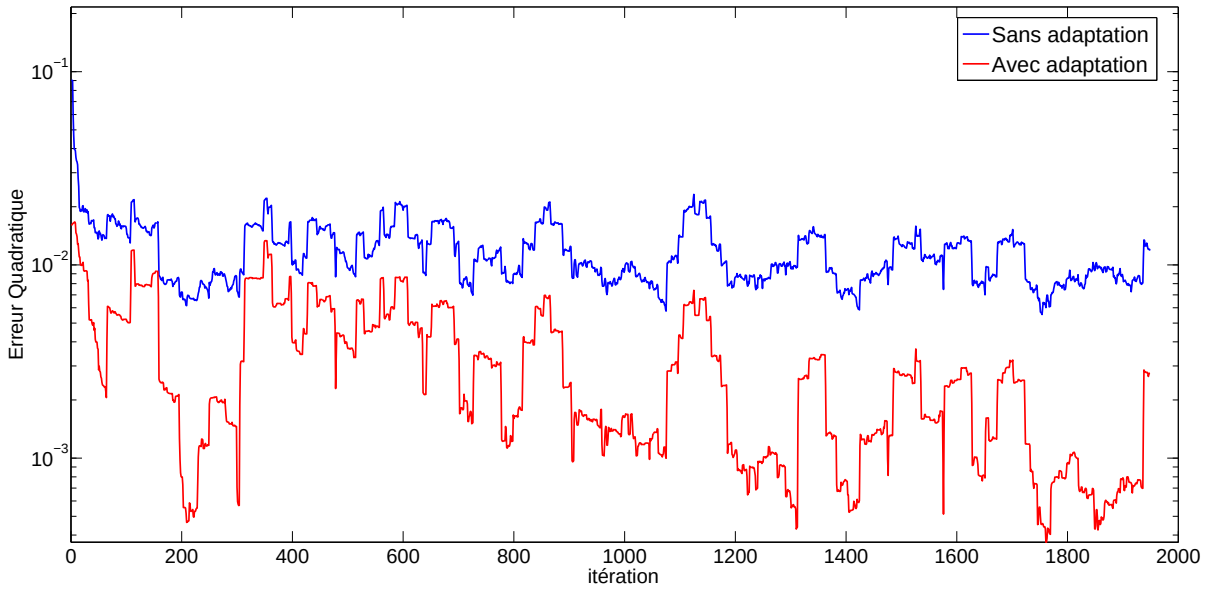

FIGURE *2.4: Erreur quadratique pour la modélisation avec KRLS de la série de Hénon avec un noyau Gaussien* $\sigma = 0.35$ - $\mu_0 = 0.6$ *et* $\nu_0 = 0.05$

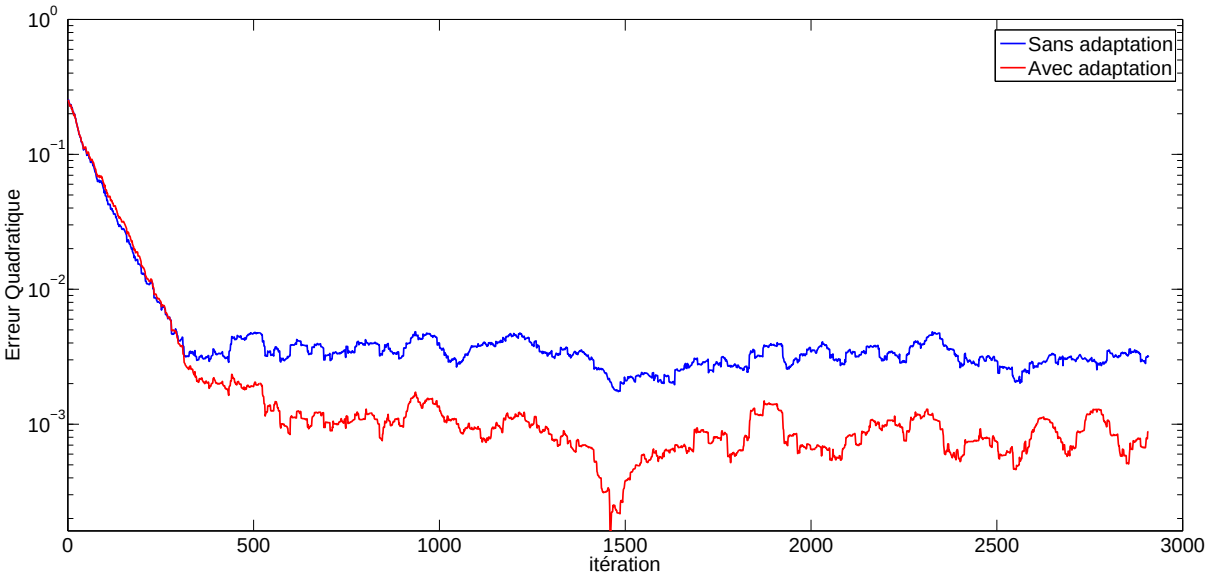

FIGURE *2.5: Prédiction de la série logistique avec un noyau Gaussien* $\sigma = 0.418$, $\mu_0 = 0.3$ *et* $\nu_0 = 0.2$

Un ensemble de 200 séries chronologiques de 3000 échantillons chacune est utilisé pour comparer les résultats de simulation de l'algorithme KAPA sans adaptation avec celles obtenues en utilisant le même algorithme mais avec adaptation (on utilisera l'acronyme AKAPA pour indiquer l'algorithme KAPA avec adaptation du dictionnaire). Les critères de performance sont la taille moyenne finale du dictionnaire ($\overline{m}$) et le NMSE dont les espérances ont été estimées en moyennant sur les 200 séries temporelles utilisées.

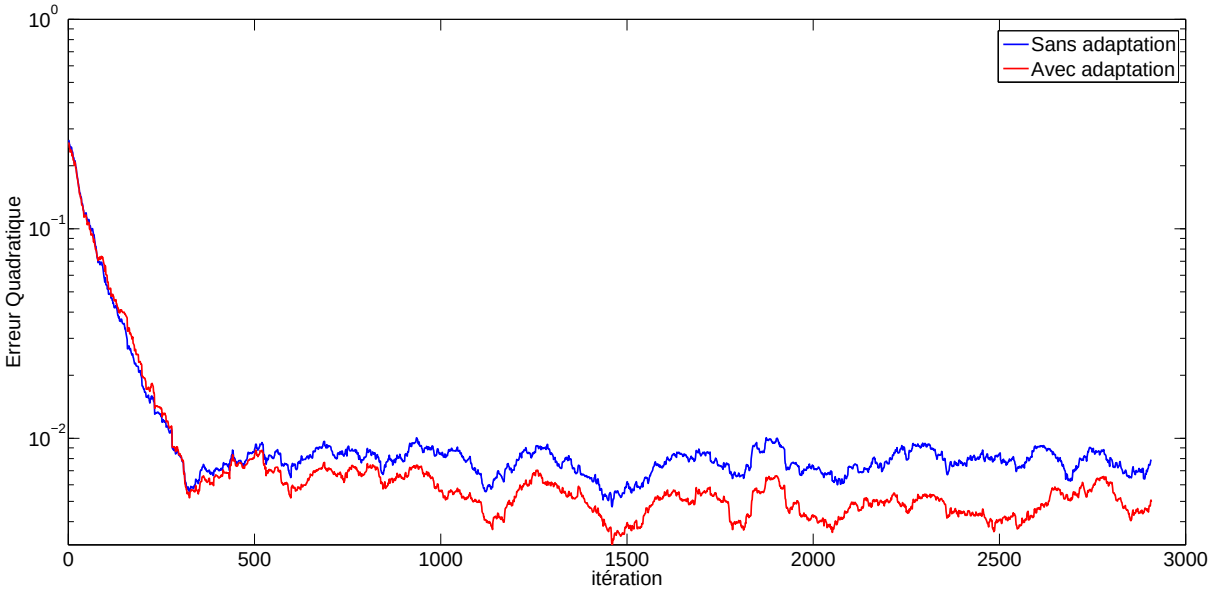

FIGURE *2.6: Prédiction de la série logistique avec un noyau Laplacien* $\sigma = 0.5$, $\mu_0 = 0.3$ *et* $\nu_0 = 0.01$

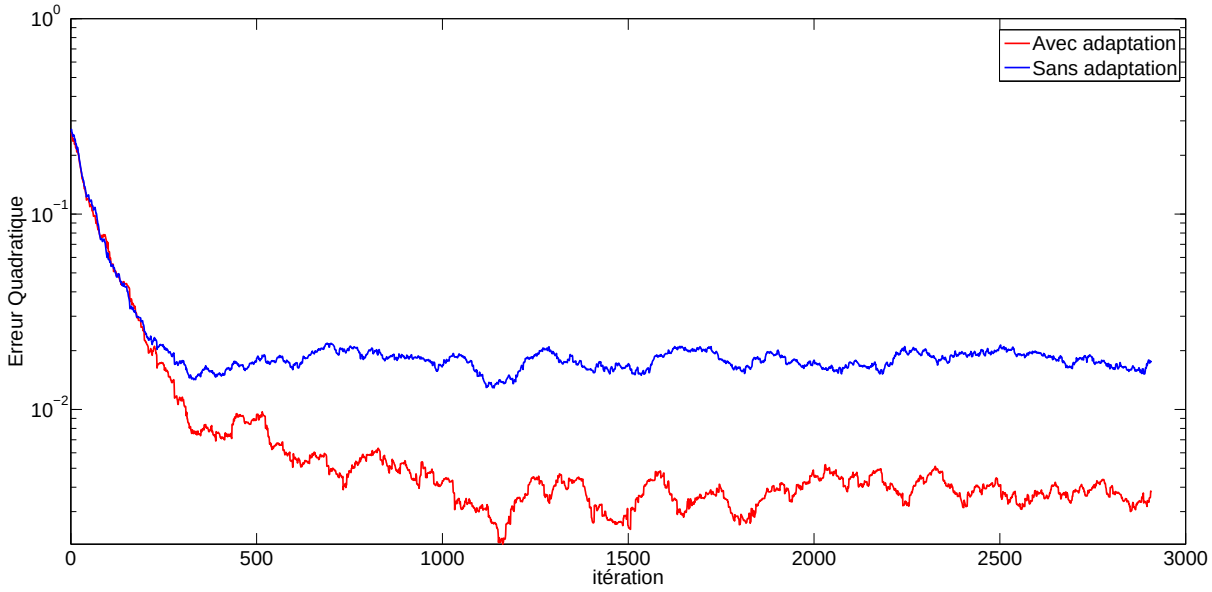

FIGURE *2.7: Prédiction de la série logistique avec un noyau Laplacien* $\sigma = 0.418$, $\mu_0 = 0.3$ *et* $\nu_0 = 0.01$

Dans les paragraphes suivants on va exploiter les simulations obtenues lors de l'utilisation des fonctions noyau RBF et polynômiales.

TABLE *2.1: Configurations et performance de l'algorithme AKAPA pour noyaux RBF, avec* $p = 3$*,* $\eta = 0.01$*, and* $\varepsilon = 0.07$

	Noyau Gaussien ($\sigma = 0.418$)			Noyau Laplacien ($\sigma = 0.35$)		
Algorithme	Paramètres	$\overline{m}$	NMSE	Paramètres	$\overline{m}$	NMSE
KAPA	μ_0=0.3	2.655	0.1597	μ_0=0.3	5.135	0.1521
AKAPA	μ_0=0.2, ν_0=0.02	2.63	0.1099	μ_0=0.24, ν_0=0.05	5.29	0.1385
AKAPA	μ_0=0.3, ν_0=0.04	3.29	0.0903	μ_0=0.3, ν_0=0.015	5.665	0.1259
AKAPA	μ_0=0.2, ν_0=0.033	2.865	0.0940	μ_0=0.25, ν_0=0.001	4.91	0.1268

2.3.3.1 Simulations avec les noyaux RBF

Le modèle à prédire est de la forme $\psi_n(u_n)$. Les simulations avec une fonction noyau Gaussienne de bande passante $\sigma = 0.418$ sont montrées en figure (2.8) et celles avec une fonction Laplacienne de bande passante $\sigma = 0.33$ sont montrées en figure (2.9). Le choix des paramètres μ_0 et ν_0 a été fait après un nombre important d'essais, ceci dans le but de retenir les meilleures combinaisons afin de montrer clairement les performances atteignables, tandis que le choix des bandes passantes des fonctions noyau (Gaussienne et Laplacienne) a été repris de la littérature pour faciliter la comparaison. Le tableau (2.1) résume les paramètres utilisés et les résultats obtenus. On peut faire les conclusions suivantes :

- Pour le même seuil de cohérence μ_0 et après adaptation, la taille moyenne du dictionnaire $\overline{m}$ a augmenté de 23.92% pour une diminution du NMSE de 43.45% pour le noyau Gaussien (respectivement 10.32% et 17.24% pour le noyau Laplacien), voir lignes 1 et 3 du tableau (2.1).
- Pour la même taille moyenne du dictionnaire $\overline{m}$ en changeant le seuil de cohérence et après adaptation, on obtient une baisse de 31.18% du NMSE pour le noyau Gaussien (respectivement 8.95% pour le noyau Laplacien), voir lignes 1 et 2 du tableau (2.1).
- Pour un NMSE approximativement constant et en jouant sur les paramètres de cohérence et sur le pas du gradient, on peut diminuer la taille moyenne du dictionnaire $\overline{m}$ de 12.91% pour le noyau Gaussien (respectivement 13.33 pour le noyau Laplacien). Voir lignes 3 et 4 du tableau (2.1).

2.3.3.2 Simulations avec le noyau polynomial

Le modèle à identifier dans ce cas est de la forme $\psi_n(u_n, u_{n-1})$. Rappelons que dans le cas du noyau polynomial, on adapte, à chaque itération, un seul élément du dictionnaire. Rappelons aussi que, dans notre étude, cet élément peut être celui qui correspond à la plus petite valeur, en valeur absolue, des coefficients $\{\alpha_{n,i}\}_{i=1\cdots m}$ du modèle $\psi_n(\cdot)$ (choix 1) ou celui qui donne la plus grande norme du gradient de l'erreur quadratique instantanée (choix 2). Les résultats des simulations pour ces deux choix sont montrés dans les figures (2.10) et (2.11) et le tableau (2.2) donne une récapitulation des résultats

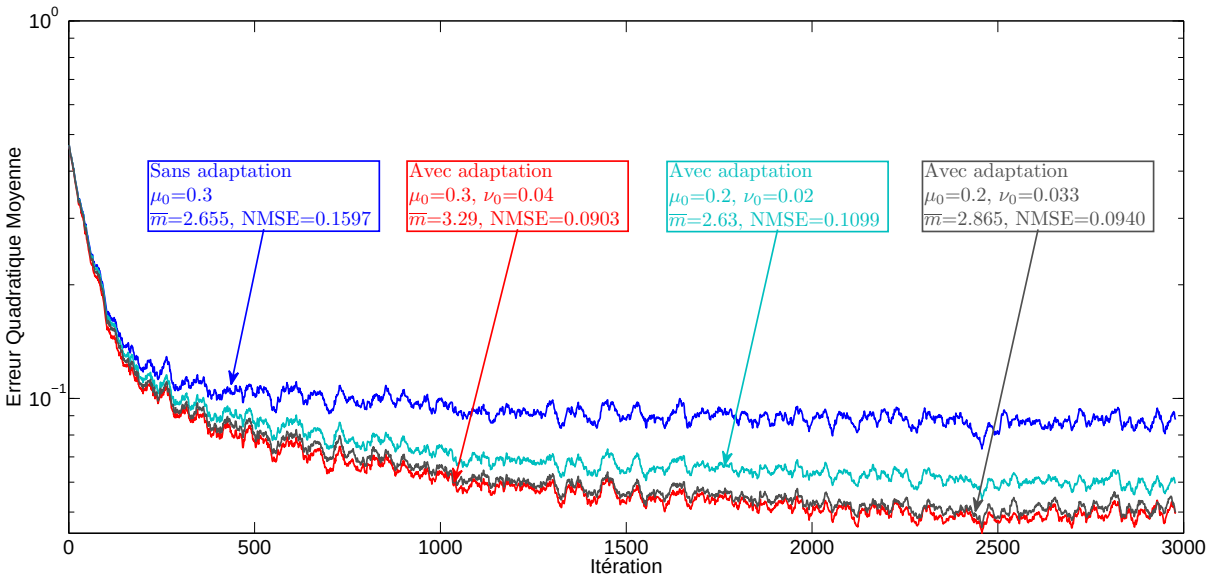

FIGURE *2.8: Comparaison avec et sans adaptation pour un noyau Gaussien avec* $\sigma = 0.418$ *- KAPA/AKAPA*

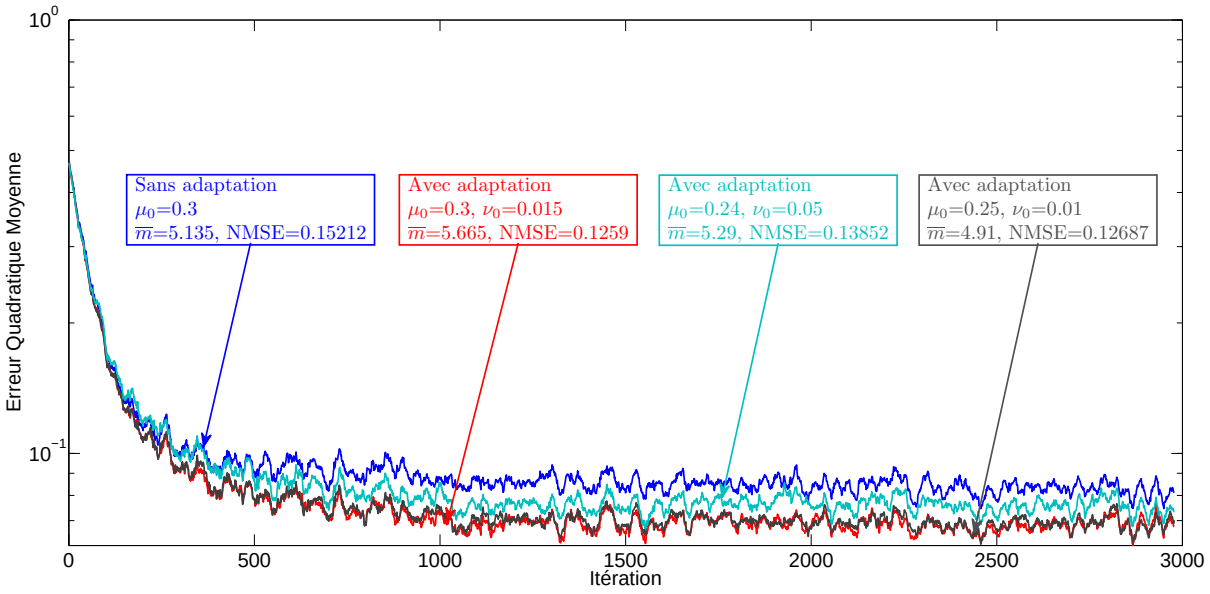

FIGURE *2.9: Comparaison avec et sans adaptation pour un noyau Laplacien avec* $\sigma = 0.35$ *- KAPA/AKAPA*

obtenus. L'analyse des résultats de simulation nous permet de faire les conclusions suivantes :

– Pour le même seuil de cohérence, et après adaptation, une augmentation de $\overline{m}$ de 76.46% mène à une diminution de 50.85% du NMSE dans le cas du choix 1 (respectivement 76.23% et 74.246% dans le cas du choix 2). Voir lignes 1 et 3 du tableau (2.2).

TABLE *2.2: Configuration et performance de l'algorithme AKAPA pour le noyau polynomial avec* $p = 3$, $\eta = 0.01$ *et* $\varepsilon = 0.07$

	Noyau polynomial du troisième ordre $\beta = 3$ Adaptation de l'élément qui correspond au à la plus petite valeur de $\lvert\alpha_{n,i}\rvert$			Noyau polynomial du deuxième ordre $\beta = 2$ Adaptation de l'élément avec le plus grand gradient en valeur absolue de e_n^2		
Algorithme	Paramètres	$\overline{m}$	NMSE	Paramètres	$\overline{m}$	NMSE
KAPA	μ_0=0.3	2.315	0.21308	μ_0=0.3	1.62	0.27649
AKAPA	μ_0=0.2, ν_0=0.02	2.32	0.12203	μ_0=0.01, ν_0=0.005	1.73	0.09078
AKAPA	μ_0=0.3, ν_0=0.75	4.085	0.10473	μ_0=0.3, ν_0=0.05	2.855	0.071207
AKAPA	μ_0=0.1, ν_0=0.03	2.055	0.10584	μ_0=0.25, ν_0=0.073	2.495	0.070121

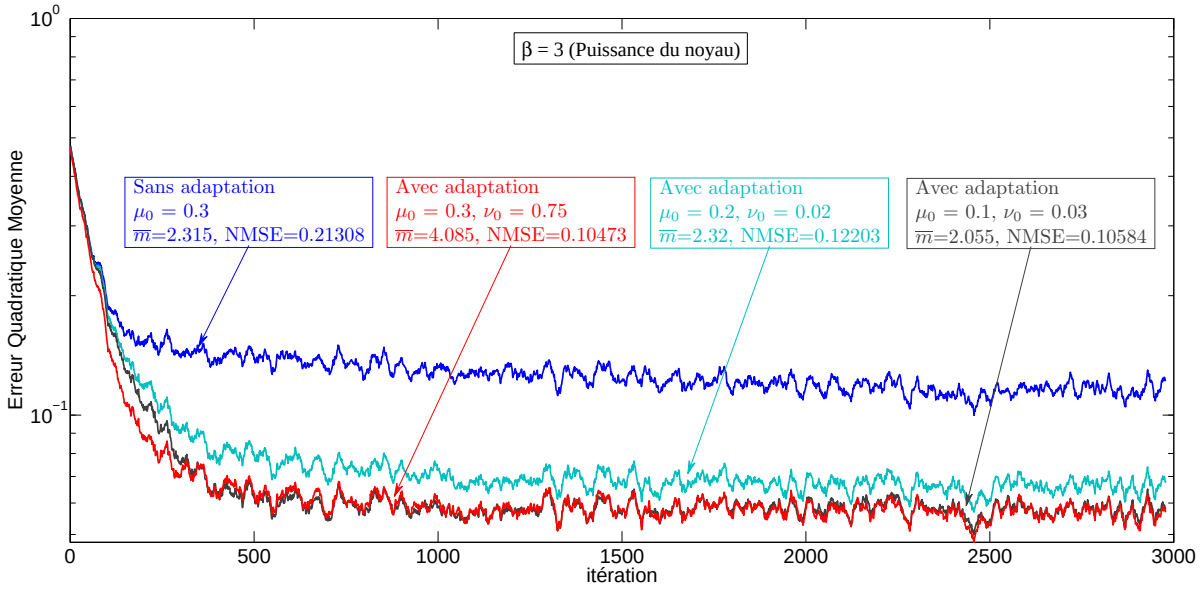

FIGURE *2.10: Comparaison avec et sans adaptation pour un noyau Polynomial avec* $\beta = 3$ *et adaptation de l'élément qui correspond au* $\min_i \lvert\alpha_{i,n}\rvert$ *- KAPA/AKAPA*

– Pour la même taille moyenne du dictionnaire ($\overline{m}$), et en changeant le seuil de cohérence, après adaptation on obtient une réduction du NMSE de 42.73% pour le choix 1 et de 67.17% pour le choix 2. Voir lignes 1 et 2 du tableau (2.2).

– Pour un NMSE comparable, en faisant varier le seuil de cohérence et le pas du gradient, la taille moyenne du dictionnaire $\overline{m}$ a diminué de 12.92% dans le cas du choix 1 et de 13.38% dans le cas du choix 2. Voir lignes 3 et 4 dans le tableau (2.2).

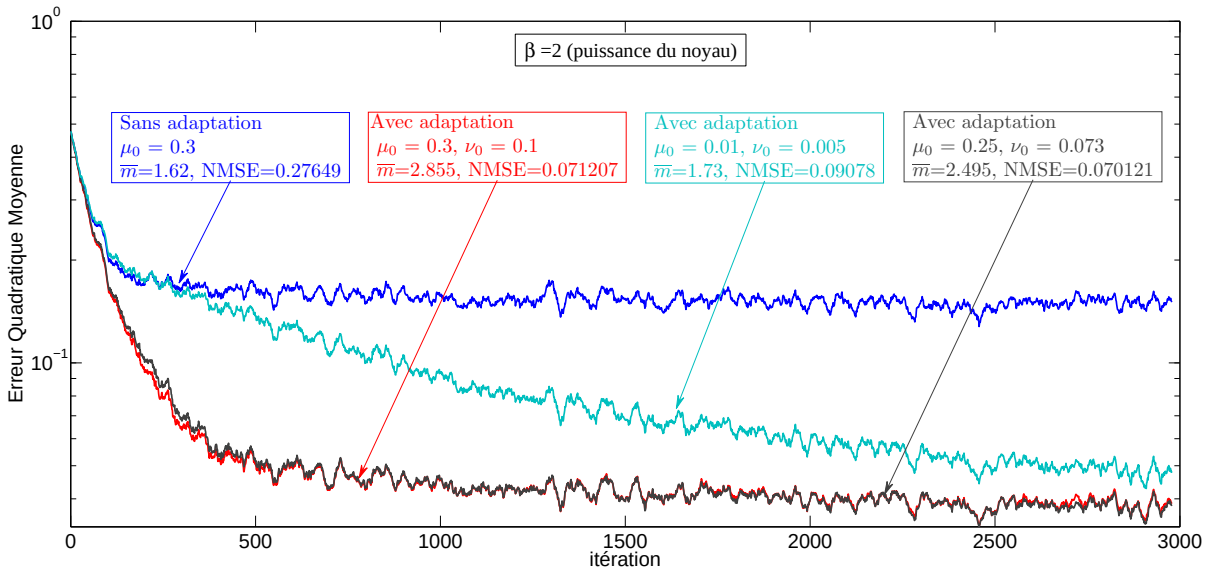

FIGURE *2.11: Comparaison avec et sans adaptation pour un noyau Polynomial avec $\beta = 2$ et adaptation de l'élément qui correspond au plus grand gradient de e_n^2 en valeur absolue - KAPA/AKAPA*

2.3.4 Simulations avec les algorithmes KNLMS et KRLS pour un signal de référence issu de la littérature

Le deuxième signal de référence que l'on va tester avec les algorithmes KNLMS et tKRLS a été utilisé dans [DKH03, RBH09, Hon07]. Son expression est :

$$d_n = (0.8 - 0.5e^{-d_{n-1}^2})d_{n-1} - (0.3 - 0.9e^{-d_{n-1}^2})d_{n-2} + 0.1\sin(d_{n-1}\pi)$$

Les conditions initiales sont (0.1 ;0.1). La sortie est noyée dans un bruit blanc de moyenne nulle et d'écart-type 0.1. Le modèle à prédire est de la forme $\psi_n(d_{n-1}, d_{n-2})$. On a généré 200 séries temporelles de 3000 échantillons chacune pour les simulations. Les mêmes critères de performances que ceux utilisés précédemment ont été utilisés dans ces essais, à savoir la taille finale moyenne du dictionnaire $\overline{m}$ et l'erreur quadratique moyenne normalisée NMSE. Nous avons utilisé la même configuration que dans [RBH09], avec $\eta = 0.09$, $\varepsilon = 0.03$. L'algorithme KNLMS est testé en premier lieu avec le noyau Gaussien de bande passante $\sigma = 0.37$. Les résultats des simulations sont montrés en figure (2.12). Le deuxième essai est réalisé avec le noyau Laplacien de bande passante $\sigma = 0.5$. Les courbes d'apprentissage sont présentées en figure (2.13). L'acronyme AKNLMS est utilisé pour indiquer l'algorithme KNLMS avec adaptation du dictionnaire. Un résumé des résultats est donné dans le tableau (2.3). Les remarques que l'on peut faire à partir de l'examen du tableau (2.3) sont les suivantes :

– Pour un même seuil de cohérence et après adaptation, le noyau Gaussien montre une diminution de la taille moyenne du dictionnaire de 15.51% ainsi qu'une réduction du NMSE de 73.47%. Pour le noyau Laplacien, la taille moyenne du dictionnaire a augmenté de 8.32% et le NMSE est réduit

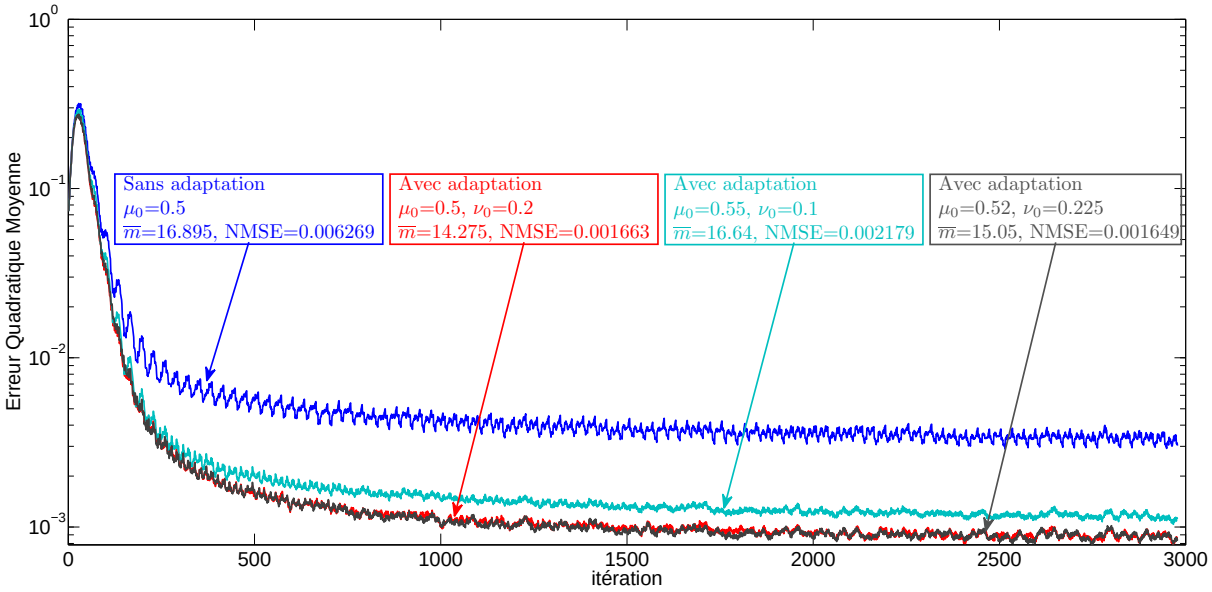

FIGURE *2.12: Comparaison avec et sans adaptation pour un noyau Gaussien avec* $\sigma = 0.37$ *- KNLMS/AKNLMS*

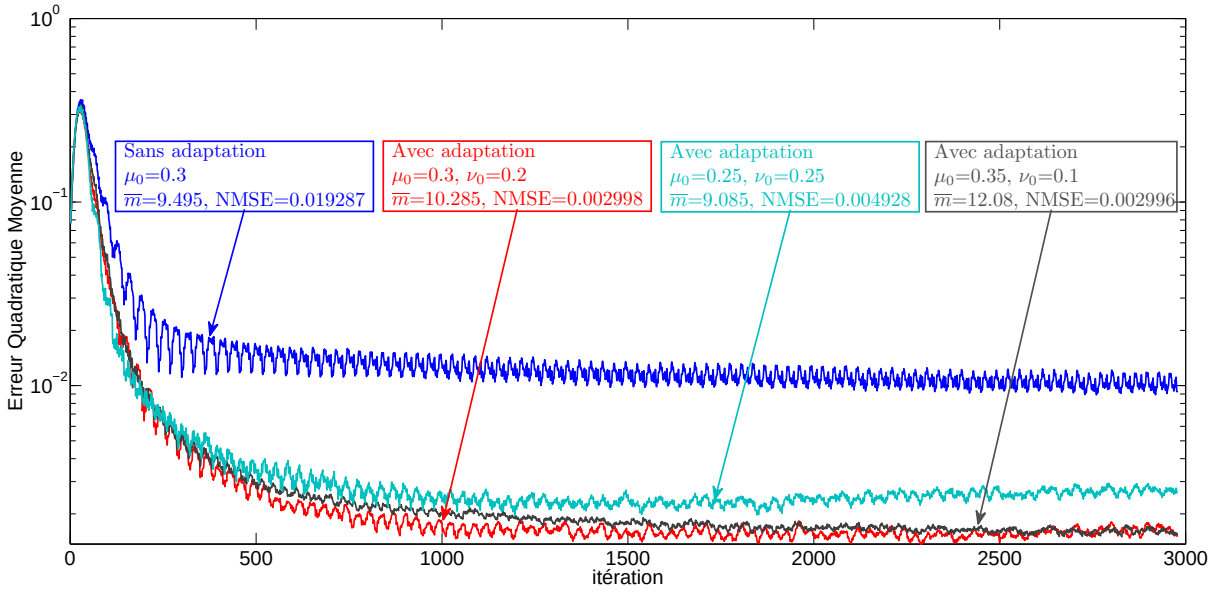

FIGURE *2.13: Comparaison avec et sans adaptation pour un noyau Laplacien avec* $\sigma = 0.5$ *- KNLMS/AKNLMS*

de 84.45%. Voir les lignes 1 et 3 du tableau (2.3).

– Pour approximativement la même taille finale moyenne du dictionnaire, le NMSE est réduit de 65.24% pour le noyau Gaussien et de 74.45% pour le noyau Laplacien, voir les lignes 1 et 3 du tableau (2.3).

TABLE *2.3: Configurations et performance de l'algorithme AKNLMS pour noyaux RBF, avec* $\eta = 0.09$*, and* $\varepsilon = 0.03$

	Noyau Gaussien ($\sigma = 0.37$)			Noyau Laplacien ($\sigma = 0.5$)		
Algorithme	Paramètres	$\overline{m}$	NMSE	Paramètres	$\overline{m}$	NMSE
KNLMS	μ_0=0.5	16.895	0.006269	μ_0=0.3	9.495	0.019287
AKNLMS	μ_0=0.55, ν_0=0.1	16.64	0.002179	μ_0=0.25, ν_0=0.25	9.085	0.004928
AKNLMS	μ_0=0.5, ν_0=0.2	14.275	0.001663	μ_0=0.3, ν_0=0.2	10.285	0.002998
AKNLMS	μ_0=0.52, ν_0=0.225	15.05	0.001649	μ_0=0.35, ν_0=0.1	12.08	0.002996

– Pour un NMSE à peu près identique, la taille moyenne du dictionnaire est réduite de 5.15% pour le noyau Gaussien et de 14.86% pour le noyau Laplacien, voir les lignes 3 et 4 du tableau (2.3).

Avant de terminer les simulations menées dans ce paragraphe, une comparaison entre les algorithmes KRLS et KNLMS avec et sans adaptation semble nécessaire. On a choisi le noyau Gaussien et le même jeu de données que précédemment pour effectuer cette comparaison. La sélection de la bande passante du noyau a été fixée à $\sigma = 0.5$ et ceci après un grand nombre d'essais. On a retenu un seuil de cohérence $\mu_0 = 0.5$. Notre objectif est d'aboutir approximativement au même NMSE sans adaptation pour les deux algorithmes et encore au même NMSE après adaptation. Les résultats des simulations sont présentés en figure (2.14). Pour un pas de gradient de référence $\nu_0 = 0.2$, on a obtenu une réduction de la taille moyenne du dictionnaire de 21.40% pour l'algorithme KRLS et pour un pas $\nu_0 = 0.085$, on a pu réduire la taille moyenne du dictionnaire de 14.50% avec l'algorithme KNLMS.

De ce qui précède, nous pouvons déduire que la taille finale du dictionnaire m peut être réduite en faisant une bonne sélection du seuil de cohérence μ_0 et du pas du gradient de référence ν_0. Parallèlement, le coût de calcul de l'adaptation sera raisonnable car il est proportionnel à la taille m.

2.3.5 Prédiction des tâches solaires (sunspots) en utilisant le KAPA

Nous appliquons maintenant notre heuristique d'adaptation à une série temporelle réelle pour prédire le nombre mensuel de tâches solaires (sunspots) entre Janvier 1749 et Février 2012. Les données ont été obtenues du site officiel de la NASA [Hat12], il s'agit de 3158 échantillons.

Pour cette série on a utilisé l'algorithme KAPA avec un noyau Gaussien de bande passante $\sigma = 0.418$. Le modèle utilisé est de la forme $\psi_n(u_{n-1}, u_{n-2}, u_{n-3})$. Les paramètres de l'algorithme sont $\eta = 0.1$, $\varepsilon = 0.07$, et $p = 3$. L'erreur NMSE est calculée sur les derniers 300 échantillons de la série. Le but est de comparer le NMSE en réglant les algorithmes (seuil de cohérence et pas du gradient) afin d'obtenir la même taille finale du dictionnaire $m \approx 536$. Avec adaptation, en ajustant un seuil de cohérence $\mu_0 = 0.1$ et un pas de gradient de référence $\nu_0 = 0.0175$ nous avons obtenu un NMSE de 0.0033112. Sans adaptation, il a fallu retenir $\mu_0 = 0.12415$ pour obtenir un NMSE de 0.016839 qui correspond à un gain de performance de 80.336%. Les résultats sont montrés en figure (2.15).

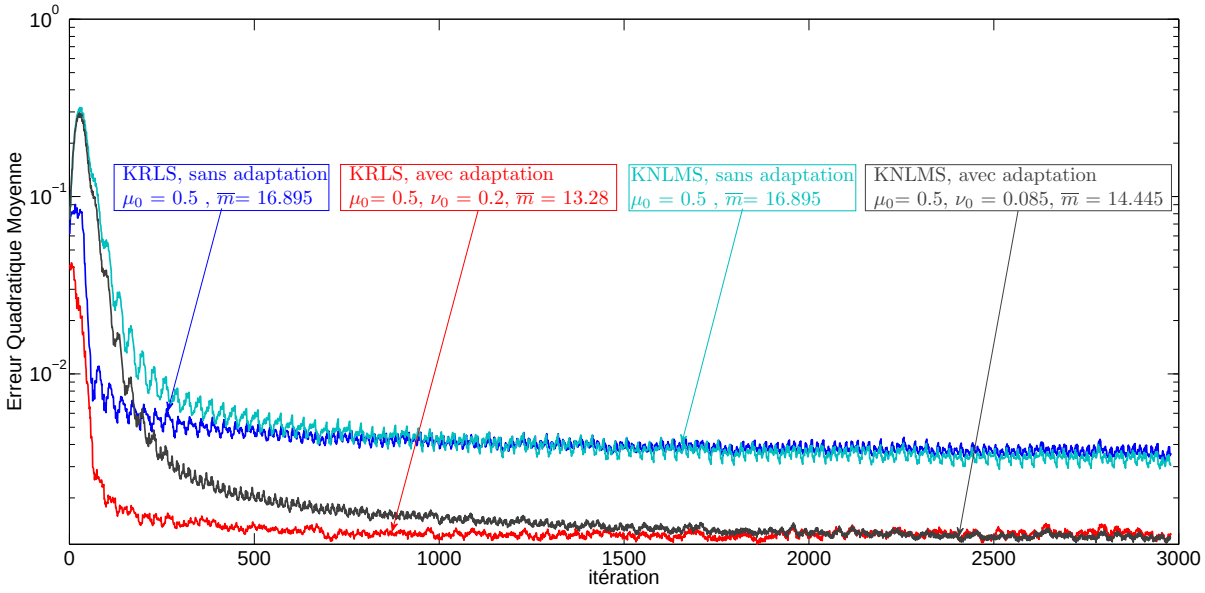

FIGURE *2.14: Comparaison AKNLMS et AKRLS en utilisant un noyau Gaussien de bande passante* $\sigma = 0.37$

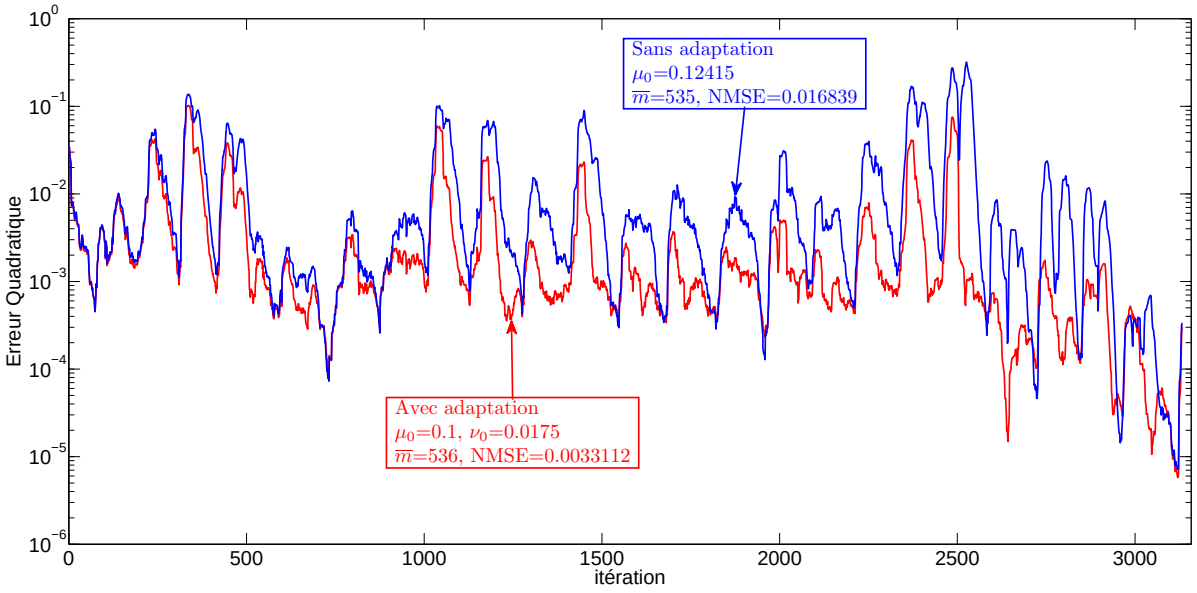

FIGURE *2.15: Prédiction des tâches solaires en utilisant un noyau Gaussien de bande passante* $\sigma = 0.418$ *avec l'algorithme KAPA*

CHAPITRE 3

Algorithmes adaptatifs en ligne pour les modèles multi-sorties

Sommaire

La majorité des approches utilisées pour l'identification des systèmes se focalise sur le cas des systèmes à sortie unique qui forment un cas particulier des problèmes réels. On considère alors qu'un système à sorties multiples n'est autre que plusieurs systèmes à sortie unique disposés en parallèle. La spécificité des modèles utilisés dans ces travaux (modèles à base de dictionnaire) fait qu'un modèle unique à sorties multiples implique que toutes les sorties partagent le même dictionnaire. Compte tenu des algorithmes d'adaptation proposés dans le chapitre précédent, nous étudions, dans ce chapitre, l'incidence de cette contrainte sur l'adaptation du dictionnaire et sur les performances obtenues. La cohérence sera à nouveau utilisée comme critère de parcimonie pour les mêmes raisons que précédemment, à savoir sa simplicité et son faible coût calculatoire.

3.1 Introduction

L'identification des systèmes basée sur l'espace de Hilbert à noyau reproduisant est d'une grande importance et a été introduite dans régression à support vecteur (SVR) [SS03, TSY11]. Les modèles étudiés sont essentiellement à entrées multiples et à sortie unique (MISO : Multiple Inputs Single Output). En réalité, plusieurs applications nécessitent des modèles à entrées multiples et sorties multiples (MIMO : Multiple Inputs Multiple Outputs) [SBT12, AHI08]. L'identification de ces derniers repose principalement sur les réseaux de neurones [Jan04, Hay08]. L'inconvénient de ces méthodes est leur complexité qui croît avec le nombre de sorties. D'autres méthodes d'identification MIMO non linéaires existent tels que les algorithmes MIMO adaptatifs et la multiregression adaptative dans RKHS [SBT12].

Les algorithmes d'identification en-ligne comme le KRLS et le KAPA, permettent d'identifier des systèmes à sortie unique (scalaire, à chaque instant). Utilisant la réduction du coût calculatoire de ces algorithmes [LZS04, LCW+11], il devient opportun d'envisager l'adaptation de ces algorithmes à des modèles à sorties multiples.

Comme il a été dit en introduction de ce chapitre, il est important de constater que, pour identifier un modèle à plusieurs sorties, on peut utiliser plusieurs modèles à sortie unique disposés en parallèle, chacun d'eux étant soumis aux mêmes entrées. Cette approche est très complexe surtout dans le cas où chaque sortie (modèle) possède son propre dictionnaire et son propre seuil de cohérence.

Dans ce chapitre on propose l'adaptation des algorithmes précédents au cas MIMO en utilisant un dictionnaire commun. Nous montrons que ce dernier peut être adapté à chaque itération comme dans le cas d'une sortie unique et avec le même coût calculatoire.

3.2 Présentation du problème

Avant de présenter en détails les algorithmes adaptatifs pour les modèles MIMO, une explication des notations doit être faite. Le modèle que l'on va étudier est à l entrées et k sorties. Considérons comme entrées l séries chronologiques $\{u_{(n)}^{(r)}\}_{r=1\cdots l}$. L'expression "entrées multiples" signifie que, à chaque instant n, l'entrée du modèle est un vecteur $\mathbf{u}_n \in \mathbb{R}^l$ avec $\mathbf{u}_n = (u_n^{(1)} \cdots u_n^{(l)})^t$ et dont chaque composante $u_n^{(r)}$ est la $n^{\text{ème}}$ valeur de la série $u_{(n)}^{(r)}$ ($r = 1 \cdots l$). Soient, à l'instant n, $y_n^{(j)}$ la $j^{\text{ème}}$ sortie du modèle et $d_n^{(j)}$ la $j^{\text{ème}}$ sortie désirée avec $j = 1 \cdots k$. On a $e_n^{(j)} = d_n^{(j)} - y_n^{(j)}$ l'erreur instantanée pour cette $j^{\text{ème}}$ sortie. Le schéma qui symbolise le modèle MIMO est illustré en figure (3.1).

Considérons la fonction numérique $\psi_n^{(j)}(\cdot)$ qui correspond à la $j^{\text{ème}}$ sortie du modèle et qui est la solution du problème d'optimisation suivant :

$$\min_{\psi^{(j)} \in \mathcal{H}} \sum_{i=1}^{n} (d_i^{(j)} - \psi^{(j)}(\mathbf{u}_i))^2 + \zeta \|\psi^{(j)}\|_{\mathcal{H}}^2 \qquad j = 1 \cdots k \tag{3.1}$$

où ζ est un paramètre de régularisation. En s'inspirant du théorème de représentation et

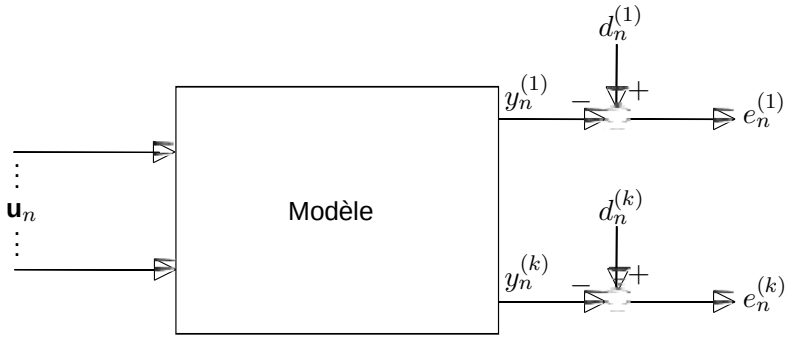

FIGURE *3.1: Modèle à multiples entrées et multiples sorties (MIMO).*

suite à l'application d'un critère de parcimonie qui conduit, à l'instant n, au dictionnaire $\mathcal{D}_n = \{\kappa(\cdot, \mathbf{u}_{w_1}), \cdots, \kappa(\cdot, \mathbf{u}_{w_m})\}$ d'ordre m ($m \ll n$), la $j^{\text{ème}}$ sortie du modèle est de la forme :

$$y_n^{(j)} = \psi_n^{(j)*}(\mathbf{u}_n) = \sum_{i=1}^{m} \alpha_{n,i}^{(j)} \kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) \qquad j = 1 \cdots k \tag{3.2}$$

où $\boldsymbol{\alpha}_n^{(j)} = (\alpha_{n,1}^{(j)}, \ldots, \alpha_{n,m}^{(j)})^t$ est le vecteur optimal des coefficients de la $j^{\text{ème}}$ sortie du modèle. Le dictionnaire $\mathcal{D}_n$ est le même pour le calcul de toutes les sorties. Pour contrôler l'ordre du modèle on va utiliser le critère de cohérence. Après avoir présenté les algorithmes adaptatifs pour le cas des sorties multiples, nous présentons l'adaptation des éléments du dictionnaire en nous inspirant des résultats du chapitre 2.

3.3 Algorithme KAPA pour sorties multiples (MOKAPA)

Le premier algorithme que nous présentons est l'algorithme de projection affine déjà étudié dans le cas linéaire [Say03] et dans sa version à noyau [LP08, ST08, RBH09]. Dans cette section, nous adaptons l'algorithme KAPA au cas des sorties multiples. Celui-ci est désigné par MOKAPA (Multiple Outputs Kernel Affine Projection Algorithm).

3.3.1 Le problème d'optimisation

L'idée de base de l'algorithme KAPA est de ne prendre en considération que les p dernières observations. p est appelé le nombre de collecteurs (manifolds) et $p \leq n$. En ne faisant pas apparaître le terme de régularisation, la solution du problème d'optimisation, à l'instant n, pour chaque sortie du modèle (3.1) et en accord avec (3.2) donne :

$$\begin{aligned} \boldsymbol{d}_n^{(1)} &= \boldsymbol{H}_n \boldsymbol{\alpha}_n^{(1)} \\ &\vdots \\ \boldsymbol{d}_n^{(k)} &= \boldsymbol{H}_n \boldsymbol{\alpha}_n^{(k)} \end{aligned}$$

où $\boldsymbol{d}_n^{(j)} = (d_n^{(j)} \ \cdots \ d_{n-p+1}^{(j)})^t$ ($j = 1 \cdots k$) est le vecteur des p dernières sorties désirées pour la $j^{\text{ème}}$ sortie du modèle et $\boldsymbol{H}_n$ est la matrice $p \times m$ définie par :

$$\mathbf{H}_n = \begin{pmatrix} \kappa(\mathbf{u}_n, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_n, \mathbf{u}_{w_m}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{u}_{n-p+1}, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{n-p+1}, \mathbf{u}_{w_m}) \end{pmatrix}$$

Puisqu'on a un dictionnaire unique, cela implique que $\boldsymbol{H}_n$ est la même pour toutes les sorties du modèle. Soient $\boldsymbol{A}_n = (\boldsymbol{\alpha}_n^{(1)} \ \cdots \ \boldsymbol{\alpha}_n^{(k)})$ la matrice des coefficients optimaux du modèle dont la taille est $m \times k$, $\boldsymbol{D}_n = (\boldsymbol{d}_n^{(1)} \ \cdots \ \boldsymbol{d}_n^{(k)})$ la matrice des sorties désirées du modèle de taille $p \times k$ et $\boldsymbol{E}_n = (\boldsymbol{e}_n^{(1)} \ \cdots \ \boldsymbol{e}_n^{(k)})$ la matrice des erreurs dont la taille est $p \times k$. On a donc :

$$\boldsymbol{A}_n = \begin{pmatrix} \alpha_{n,1}^{(1)} & \cdots & \alpha_{n,1}^{(k)} \\ \vdots & \ddots & \vdots \\ \alpha_{n,m}^{(1)} & \cdots & \alpha_{n,m}^{(k)} \end{pmatrix} \quad \boldsymbol{D}_n = \begin{pmatrix} d_n^{(1)} & \cdots & d_n^{(k)} \\ \vdots & \ddots & \vdots \\ d_{n-p+1}^{(1)} & \cdots & d_{n-p+1}^{(k)} \end{pmatrix} \quad \boldsymbol{E}_n = \begin{pmatrix} e_n^{(1)} & \cdots & e_n^{(k)} \\ \vdots & \ddots & \vdots \\ e_{n-p+1}^{(1)} & \cdots & e_{n-p+1}^{(k)} \end{pmatrix}$$

Par similitude au cas d'une sortie unique, le problème d'optimisation est :

$$\boldsymbol{A}_n = \arg\min_{\boldsymbol{A}} \|\boldsymbol{D}_n - \boldsymbol{H}_n \boldsymbol{A}\|_F^2 \tag{3.3}$$

Notons que $\|\cdot\|_F^2$ est la norme de Frobenius. La solution de ce problème d'optimisation est donnée par la résolution de :

$$\boldsymbol{D}_n = \boldsymbol{H}_n \boldsymbol{A}_n \tag{3.4}$$

Si $(\mathbf{H}_n^t \mathbf{H}_n)^{-1}$ existe, la solution du problème est :

$$\boldsymbol{A}_n = (\mathbf{H}_n^t \mathbf{H}_n)^{-1} \mathbf{H}_n^t \boldsymbol{D}_n \tag{3.5}$$

Avec une approche adaptative, la matrice $\boldsymbol{A}_{n+1}$ est déterminée à l'instant $n+1$ à partir de l'estimation précédente $\boldsymbol{A}_n$. Comme dans le cas de la sortie unique, le principe de fluctuation minimale est appliqué par la minimisation de $\|\boldsymbol{A}_{n+1} - \boldsymbol{A}_n\|_F^2$. En se basant sur ce qui précède, le problème devient :

$$\min_{\boldsymbol{A}_{n+1}} \|\boldsymbol{A}_{n+1} - \boldsymbol{A}_n\|_F^2 \tag{3.6}$$

$$\text{sous contrainte } \boldsymbol{D}_{n+1} = \mathbf{H}_{n+1} \boldsymbol{A}_{n+1} \tag{3.7}$$

Dans ce qui suit, nous dérivons la solution de ce problème tout en utilisant le critère de cohérence pour maîtriser la taille du dictionnaire à chaque instant.

3.3.2 Obtention de l'algorithme MOKAPA avec critère de cohérence

A l'instant $n+1$, l'objectif est toujours de trouver la matrice des coefficients optimaux du modèle $\boldsymbol{A}_{n+1}$. Quand un nouveau vecteur $\mathbf{u}_{n+1}$ se présente à l'entrée du modèle, un des deux cas suivant se produit :

– Si $\max_{i=1,\dots,m} |\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_i})| > \mu_0$
Si c'est le cas, ceci signifie que la nouvelle fonction noyau $\kappa(\cdot, \mathbf{u}_{n+1})$ peut être représentée avec une faible erreur d'approximation par les fonctions noyau du dictionnaire actuel $\mathcal{D}_n$. Cette fonction n'est pas introduite dans le dictionnaire et l'ordre de ce dernier reste égal à m. La solution du problème (3.6) sous contrainte (3.7) est trouvée en minimisant le Lagrangien :

$$J(\boldsymbol{A}, \boldsymbol{\Lambda}) = \|\boldsymbol{A} - \boldsymbol{A}_n\|_F^2 + \mathbf{1}_p^t \left(\boldsymbol{\Lambda} \odot (\boldsymbol{D}_{n+1} - \boldsymbol{H}_{n+1}\boldsymbol{A})\right)\mathbf{1}_k \tag{3.8}$$

où $\odot$ est le produit de Hadamard, $\boldsymbol{\Lambda}$ est la matrice diagonale des multiplicateurs de Lagrange et $\mathbf{1}_k = (1, \cdots, 1)^t$ un vecteur de taille $k \times 1$. En dérivant le Lagrangien par rapport à $\boldsymbol{A}$ et $\boldsymbol{\Lambda}$ et en annulant ses dérivées en $\boldsymbol{A}_{n+1}$, et $\boldsymbol{\Lambda}_{n+1}$ on obtient les expressions suivantes :

$$\begin{aligned} \frac{\partial J(\boldsymbol{A}, \boldsymbol{\Lambda})}{\partial \boldsymbol{A}} = 0 \qquad &\Rightarrow \qquad 2(\boldsymbol{A}_{n+1} - \boldsymbol{A}_n) = \boldsymbol{H}_{n+1}^t \boldsymbol{\Lambda}_{n+1} \\ \frac{\partial J(\boldsymbol{A}, \boldsymbol{\Lambda})}{\partial \boldsymbol{\Lambda}} = 0 \qquad &\Rightarrow \qquad \boldsymbol{D}_{n+1} = \boldsymbol{H}_{n+1}\boldsymbol{A}_{n+1} \end{aligned}$$

Si $\mathbf{H}_{n+1}^t \mathbf{H}_{n+1}$ est non singulière, on obtient :

$$\boldsymbol{\Lambda}_{n+1} = 2(\boldsymbol{H}_{n+1}\boldsymbol{H}_{n+1}^t)^{-1}\boldsymbol{H}_{n+1}(\boldsymbol{A}_{n+1} - \boldsymbol{A}_n) = 2(\boldsymbol{H}_{n+1}\boldsymbol{H}_{n+1}^t)^{-1}(\boldsymbol{D}_{n+1} - \boldsymbol{H}_{n+1}\boldsymbol{A}_n)$$

La matrice des coefficients du modèle est mise à jour de la manière suivante :

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \boldsymbol{H}_{n+1}^t(\varepsilon\mathbf{I} + \boldsymbol{H}_{n+1}\boldsymbol{H}_{n+1}^t)^{-1}(\boldsymbol{D}_{n+1} - \boldsymbol{H}_{n+1}\boldsymbol{A}_n) \tag{3.9}$$

Le terme $\varepsilon\mathbf{I}$ est un facteur de régularisation avec $\mathbf{I}$ est une matrice identité de taille $p \times p$.

– Si $\max_{i=1,\dots,m} |\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_i})| \leq \mu_0$
Dans ce cas la fonction $\kappa(\cdot, \mathbf{u}_{n+1})$ ne peut pas être représentée correctement par les fonctions noyau du dictionnaire, d'où la nécessité de l'introduire dans le dictionnaire dont la taille augmente d'une unité. Le nouveau dictionnaire devient $\mathcal{D}_{n+1} = \{\kappa(\cdot, \mathbf{u}_{w_1}), \cdots, \kappa(\cdot, \mathbf{u}_{w_{m+1}})\}$ où $\kappa(\cdot, \mathbf{u}_{w_{m+1}}) = \kappa(\cdot, \mathbf{u}_{n+1})$. La matrice $\boldsymbol{H}_{n+1}$ devient de taille $p \times (m+1)$ suite à la concaténation de la colonne $(\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_{m+1}}) \cdots \kappa(\mathbf{u}_{n-p+2}, \mathbf{u}_{w_{m+1}}))^t$. De même la matrice $\boldsymbol{A}_{n+1}$ augmente de taille pour devenir $(m+1) \times k$. La matrice $\boldsymbol{D}_{n+1}$ reste toujours de taille $p \times k$ et le problème

d'optimisation (3.6) devient :

$$\min_{\boldsymbol{A}_{n+1}} \left\| \boldsymbol{A}_{n+1} - \begin{bmatrix} \boldsymbol{A}_n \\ \boldsymbol{0}^t \end{bmatrix} \right\|_F^2 \tag{3.10}$$

$$\text{sous contrainte } \boldsymbol{D}_{n+1} = \mathbf{H}_{n+1}\boldsymbol{A}_{n+1} \tag{3.11}$$

où $\boldsymbol{0} = (0, \cdots, 0)^t$ est le vecteur nul de taille $k \times 1$.

La procédure utilisée précédemment peut être appliquée ici aussi pour aboutir à la mise à jour de la matrice des coefficients du modèle à l'instant $n+1$:

$$\boldsymbol{A}_{n+1} = \begin{bmatrix} \boldsymbol{A}_n \\ \boldsymbol{0}^t \end{bmatrix} + \boldsymbol{H}_{n+1}^t(\varepsilon \mathbf{I} + \boldsymbol{H}_{n+1}\boldsymbol{H}_{n+1}^t)^{-1}\Big(\boldsymbol{D}_{n+1} - \boldsymbol{H}_{n+1}\begin{bmatrix} \boldsymbol{A}_n \\ \boldsymbol{0}^t \end{bmatrix}\Big) \tag{3.12}$$

On appelle les expressions (3.9) et (3.12) par l'algorithme de projection affine à noyau à multiples sorties MOKAPA.

3.4 L'algorithme KNLMS pour multiples sorties (MOKNLMS)

L'algorithme de moindres carrés normalisé à noyau (KNLMS) appartient à la famille des algorithmes à gradient stochastique et il est la version instantanée de l'algorithme KAPA. La version à multiples sorties de cet algorithme sera nommée (Multiple Outputs Kernel Normalised Least Mean Squared algorithm : MOKNLMS) et elle est dérivée de l'algorithme MOKAPA en prenant le nombre de collecteurs $p = 1$. Les mises à jour à l'instant $n+1$ de la matrice des coefficients du modèle sont obtenues de la façon suivante :

– Si $\max_{i=1,\ldots,m} |\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_i})| > \mu_0$

Dans ce cas la fonction $\kappa(\cdot, \mathbf{u}_{n+1})$ n'est pas introduite dans le dictionnaire qui reste de taille m. La mise à jour se fait de la façon suivante :

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \frac{\mathbf{h}_{n+1}}{\varepsilon + ||\mathbf{h}_{n+1}||^2}(\boldsymbol{\delta}_{n+1}^t - \mathbf{h}_{n+1}^t\boldsymbol{A}_n) \tag{3.13}$$

avec $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$ de taille $m \times 1$ et $\boldsymbol{\delta}_{n+1} = (d_{n+1}^{(1)}, \cdots, d_{n+1}^{(k)})^t$ de taille $k \times 1$.

– Si $\max_{i=1,\ldots,m} |\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_i})| \leq \mu_0$

Dans ce cas la fonction $\kappa(\cdot, \mathbf{u}_{n+1})$ est introduite dans le dictionnaire qui devient de taille $m+1$. L'augmentation de l'ordre du modèle impose la mise à jour comme suit :

$$\boldsymbol{A}_{n+1} = \begin{bmatrix} \boldsymbol{A}_n \\ \boldsymbol{0}^t \end{bmatrix} + \frac{\mathbf{h}_{n+1}}{\varepsilon + ||\mathbf{h}_{n+1}||^2}\Big(\boldsymbol{\delta}_{n+1}^t - \mathbf{h}_{n+1}^t\begin{bmatrix} \boldsymbol{A}_n \\ \boldsymbol{0}^t \end{bmatrix}\Big) \tag{3.14}$$

$\mathbf{h}_{n+1}$ devient de taille $m+1 \times 1$ tel que : $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m+1}))^t$ et

$\mathbf{0} = (0, \cdots, 0)^t$ un vecteur de taille $k \times 1$.

3.5 L'algorithme KRLS pour multiples sorties (MOKRLS)

Cette section est dédiée à la présentation de l'algorithme de moindres carrés récursif à noyau pour les modèles à plusieurs sorties (Multiple Outputs Kernel Recursive Least Squares algorithm). Cet algorithme est une extension de l'algorithme KRLS, étudié pour une simple sortie scalaire dans [EMM04, Hon07, HRB07, LPH10], revisité ici pour des modèles à sorties multiples.

3.5.1 Le problème d'optimisation

Les algorithmes MOKNLMS et MOKAPA présentés ci-dessus se basent sur une approximation instantanné pour le premier et à fenêtre glissante pour le second. L'algorithme étudié ici, par analogie avec le RLS en sortie scalaire, permet une meilleure approximation, en considérant toutes les observations précédentes jusqu'à l'instant actuel.

L'introduction d'un facteur d'oubli permet de négliger les observations les plus anciennes pour permettre l'identification de systèmes non-stationnaires. Après l'introduction de ce facteur d'oubli $\theta \in]0, 1]$, la $j^{\text{ème}}$ sortie du modèle est obtenue par la résolution du problème d'optimisation suivant :

$$\min_{\psi^{(j)} \in \mathcal{H}} \sum_{i=1}^{n} \theta^{n-i} (d_i^{(j)} - \psi^{(j)}(\mathbf{u}_i))^2 + \zeta \theta^n \|\psi^{(j)}\|_{\mathcal{H}}^2 \qquad j = 1 \cdots k \tag{3.15}$$

Pour simplifier les calculs, un facteur d'oubli identique est utilisé pour toutes les sorties. A l'instant n, en se basant sur le théorème de représentation, la $j^{\text{ème}}$ sortie du modèle est toujours obtenue par (3.2) où $\boldsymbol{\alpha}_n^{(j)} = (\alpha_{n,1}^{(j)}, \ldots, \alpha_{n,m}^{(j)})^t$ est le vecteur optimal des coefficients de cette $j^{\text{ème}}$ sortie du modèle. Soit $\boldsymbol{d}_n^{(j)} = (d_1^{(j)} \ \cdots \ d_n^{(j)})^t$ ($j = 1 \cdots k$) le vecteur des sorties désirées du modèle j jusqu'à l'instant n.

En se basant sur (3.15), le problème dual se met sous la forme :

$$\boldsymbol{A}_n = \arg\min_{\boldsymbol{A}} \big(\boldsymbol{D}_n - \mathbf{H}_n \boldsymbol{A}\big)^t \boldsymbol{\Theta}_n \big(\boldsymbol{D}_n - \mathbf{H}_n \boldsymbol{A}\big) + \zeta \theta^n \boldsymbol{A}^t \mathbf{K}_n \boldsymbol{A} \tag{3.16}$$

où $\boldsymbol{A}_n = (\boldsymbol{\alpha}_n^{(1)} \ \cdots \ \boldsymbol{\alpha}_n^{(k)})$ est la matrice des coefficients optimaux du modèle (de taille $m \times k$), $\boldsymbol{D}_n = (\boldsymbol{d}_n^{(1)} \ \cdots \ \boldsymbol{d}_n^{(k)})$ la matrice des sorties désirées du modèle de dimension $n \times k$, $\mathbf{K}_n$ est la matrice de Gram des éléments du dictionnaire de taille $m \times m$, $\boldsymbol{\Theta}_n$ est une matrice diagonale carrée de taille $n \times n$ dont le $(i,i)^{\text{ème}}$ élément est θ^{n-i} et $\mathbf{H}_n$ une matrice de taille $n \times m$ dont le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_i, \mathbf{u}_{w_j})$.

La solution de (3.16) est :

$$\begin{aligned} \mathbf{H}_n^t \boldsymbol{\Theta}_n \big(\boldsymbol{D}_n - \mathbf{H}_n \boldsymbol{A}_n\big) &= \zeta \theta^n \mathbf{K}_n \boldsymbol{A}_n \\ \Rightarrow \big(\mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{H}_n + \zeta \theta^n \mathbf{K}_n\big) \boldsymbol{A}_n &= \mathbf{H}_n^t \boldsymbol{\Theta}_n \boldsymbol{D}_n \end{aligned}$$

En posant $\mathbf{P}_n = \big(\mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{H}_n + \zeta \theta^n \mathbf{K}_n\big)^{-1}$ et supposant que $\mathbf{P}_n$ existe, la solution du problème est :

$$\boldsymbol{A}_n = \mathbf{P}_n \mathbf{H}_n^t \boldsymbol{\Theta}_n \boldsymbol{D}_n \tag{3.17}$$

La recherche d'une solution récursive implique que l'on exprime la solution du problème à l'instant $n+1$ à l'aide de celle trouvée à l'instant n. Ceci est donné dans la suite.

3.5.2 Algorithme MOKRLS avec le critère de cohérence

L'ordre du modèle est toujours contrôlé par le critère de cohérence en choisissant a priori le seuil μ_0. A l'instant $n+1$, lorsqu'une nouvelle observation $\mathbf{u}_{n+1}$ se présente à l'entrée du modèle, l'application du critère de cohérence conduit aux deux cas suivants :

– Si $\max_{i=1,\ldots,m} |\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_i})| > \mu_0$

Dans ce cas, la contribution de la fonction noyau $\kappa(\cdot, \mathbf{u}_{n+1})$ à tout modèle est faible et elle n'est pas introduite dans le dictionnaire $\mathcal{D}_n$. L'ordre de ce dernier reste m et la matrice de Gram reste de taille $m \times m$ à l'instant $n+1$, avec $\mathbf{K}_{n+1} = \mathbf{K}_n$. La matrice $\boldsymbol{\Theta}_{n+1}$ est de taille $(n+1)\times(n+1)$ et dont le $(i,i)^{\text{ème}}$ élément est θ^{n+1-i}. $\mathbf{H}_{n+1}$ devient de taille $(n+1) \times m$ et la taille de $\boldsymbol{D}_{n+1}$ devient $(n+1) \times k$.

$$\mathbf{H}_{n+1} = \begin{bmatrix} \mathbf{H}_n \\ \mathbf{h}_{n+1}^t \end{bmatrix} \qquad \boldsymbol{\Theta}_{n+1} = \begin{bmatrix} \theta \boldsymbol{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1 \end{bmatrix} \qquad \boldsymbol{D}_{n+1} = \begin{bmatrix} \boldsymbol{D}_n \\ \boldsymbol{\delta}_{n+1}^t \end{bmatrix}$$

où $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$ et $\boldsymbol{\delta}_{n+1} = (d_{n+1}^{(1)}, \ \cdots \ d_{n+1}^{(k)})^t$. La solution du problème (3.16) à l'instant $n+1$ est :

$$\boldsymbol{A}_{n+1} = \mathbf{P}_{n+1} \mathbf{H}_{n+1}^t \boldsymbol{\Theta}_{n+1} \boldsymbol{D}_{n+1} \tag{3.18}$$

$\mathbf{P}_{n+1}$ est une matrice carrée de taille $m \times m$ et peut être écrite sous la forme :

$$\begin{aligned} \mathbf{P}_{n+1} &= \big(\mathbf{H}_{n+1}^t \boldsymbol{\Theta}_{n+1} \mathbf{H}_{n+1} + \zeta \theta^{n+1} \mathbf{K}_{n+1}\big)^{-1} \\ &= \left(\begin{bmatrix} \mathbf{H}_n \\ \mathbf{h}_{n+1}^t \end{bmatrix}^t \begin{bmatrix} \theta \boldsymbol{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_n \\ \mathbf{h}_{n+1}^t \end{bmatrix} + \zeta \theta^{n+1} \mathbf{K}_n \right)^{-1} \\ &= \big(\theta \mathbf{P}_n^{-1} + \mathbf{h}_{n+1} \mathbf{h}_{n+1}^t\big)^{-1} \end{aligned} \tag{3.19}$$

En se basant sur (A.10), (A.11)et (A.17) de l'annexe A et sachant que :

$$\begin{aligned}\mathbf{H}_{n+1}^t\boldsymbol{\Theta}_{n+1}\boldsymbol{D}_{n+1} &= \begin{bmatrix}\mathbf{H}_n\\ \mathbf{h}_{n+1}^t\end{bmatrix}^t\begin{bmatrix}\theta\boldsymbol{\Theta}_n & \mathbf{0}_n\\ \mathbf{0}_n^t & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{D}_n\\ \boldsymbol{\delta}_{n+1}^t\end{bmatrix}\\ &= \theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\boldsymbol{D}_n + \mathbf{h}_{n+1}\boldsymbol{\delta}_{n+1}^t\end{aligned} \tag{3.20}$$

Les équations (3.18) et (3.20) entrainent

$$\begin{aligned}\boldsymbol{A}_{n+1} &= \mathbf{P}_{n+1}\big(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\boldsymbol{D}_n + \mathbf{h}_{n+1}\boldsymbol{\delta}_{n+1}\big)\\ &= \left[\mathbf{P}_n - \frac{\theta^{-1}\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1+\theta^{-1}\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}\right]\mathbf{H}_n^t\boldsymbol{\Theta}_n\boldsymbol{D}_n + \mathbf{P}_{n+1}\mathbf{h}_{n+1}\boldsymbol{\delta}_{n+1}\end{aligned} \tag{3.21}$$

L'expression précédente mène à :

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \mathbf{P}_{n+1}\mathbf{h}_{n+1}(\boldsymbol{\delta}_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{A}_n) \tag{3.22}$$

où $\mathbf{P}_{n+1}$ est calculée d'après l'expression (A.11) :

$$\mathbf{P}_{n+1} = \theta^{-1}\left[\mathbf{P}_n - \frac{\theta^{-1}\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1+\theta^{-1}\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}\right] \tag{3.23}$$

– Si $\max_{i=1,\dots,m}|\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_i})| \leq \mu_0$

Dans ce cas $\kappa(\cdot,\mathbf{u}_{n+1})$ doit être introduite dans le dictionnaire $\mathcal{D}_{n+1}$. L'ordre de ce dernier devient $m+1$ avec $\kappa(\cdot,\mathbf{u}_{w_{m+1}}) = \kappa(\cdot,\mathbf{u}_{n+1})$. la matrice de Gram $\mathbf{K}_{n+1}$ devient de taille $(m+1)\times(m+1)$. L'élément (i,j) de $\mathbf{K}$ est $\kappa(\mathbf{u}_{w_i},\mathbf{u}_{w_j})$, la matrice $\boldsymbol{\Theta}_{n+1}$ devient de taille $(n+1)\times(n+1)$ et son $(i,i)^{\text{ème}}$ élément est θ^{n+1-i}, $\mathbf{H}_{n+1}$ devient de taille $(n+1)\times(m+1)$ et la taille de $\boldsymbol{D}_{n+1}$ devient $(n+1)\times k$. Ces matrices sont données par :

$$\mathbf{H}_{n+1} = \begin{bmatrix}\mathbf{H}_n & \mathbf{0}_n\\ \mathbf{h}_{n+1}^t & h_0\end{bmatrix} \qquad \mathbf{K}_{n+1} = \begin{bmatrix}\mathbf{K}_n & \mathbf{h}_{n+1}\\ \mathbf{h}_{n+1}^t & h_0\end{bmatrix} \qquad \boldsymbol{\Theta}_{n+1} = \begin{bmatrix}\theta\boldsymbol{\Theta}_n & \mathbf{0}_n\\ \mathbf{0}_n^t & 1\end{bmatrix} \qquad \boldsymbol{D}_{n+1} = \begin{bmatrix}\boldsymbol{D}_n\\ \boldsymbol{\delta}_{n+1}^t\end{bmatrix}$$

où $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_1}),\cdots,\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_m}))^t$, $\boldsymbol{\delta}_{n+1} = (d_{n+1}^{(1)},\ \cdots\ d_{n+1}^{(k)})^t$, $h_0 = \kappa(\mathbf{u}_{w_{m+1}},\mathbf{u}_{w_{m+1}})$ et $\mathbf{0}_n$ est un vecteur colonne de n zéros. L'expression (3.18) s'applique toujours pour déterminer la matrice des coefficients optimaux du modèle et $\mathbf{P}_{n+1}$ reste identique au cas d'une sortie unique car toutes les sorties partagent le même dictionnaire. $\mathbf{P}_{n+1}$ est donnée par (A.17)

$$\mathbf{P}_{n+1} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m\\ \mathbf{0}_m^t & 0\end{bmatrix} + \frac{1}{s}\begin{bmatrix}\mathbf{q}\mathbf{q}^t & -\mathbf{q}\\ \mathbf{q}^t & 1\end{bmatrix} \tag{3.24}$$

où $\mathbf{0}_m$ est un vecteur colonne de m zéros et

$$\begin{aligned}\tilde{\mathbf{P}}_{n+1} &= (\theta \mathbf{P}_n^{-1} + \mathbf{h}_{n+1}\mathbf{h}_{n+1}^t)^{-1} \\ \mathbf{q} &= (h_0 + \zeta\theta^{n+1})\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1} \\ s &= (h_0 + \zeta\theta^{n+1})(h_0 - \mathbf{h}_{n+1}^t\mathbf{q})\end{aligned}$$

$\tilde{\mathbf{P}}_{n+1}$ existe dans la solution $\tilde{\boldsymbol{A}}_{n+1} = \tilde{\mathbf{P}}_{n+1}\tilde{\mathbf{H}}_{n+1}^t\boldsymbol{\Theta}_{n+1}\boldsymbol{D}_{n+1}$ du problème :

$$\tilde{\boldsymbol{A}}_{n+1} = \arg\min_{\boldsymbol{A}} \big(\boldsymbol{D}_{n+1} - \tilde{\mathbf{H}}_{n+1}\boldsymbol{A}\big)^t\boldsymbol{\Theta}_{n+1}\big(\boldsymbol{D}_{n+1} - \tilde{\mathbf{H}}_{n+1}\boldsymbol{A}\big) + \zeta\theta^n\boldsymbol{A}^t\mathbf{K}_{n+1}\boldsymbol{A}$$

où $\tilde{\mathbf{H}}_{n+1} = \begin{bmatrix}\mathbf{H}_n^t & \mathbf{h}_{n+1}\end{bmatrix}^t$. La mise à jour de la matrice des coefficients optimaux du modèle se fait donc en deux phases. La première phase consiste de trouver $\tilde{\boldsymbol{A}}_{n+1}$ et $\tilde{\mathbf{P}}_{n+1}$ de la même manière que dans le cas où on n'a pas introduit un nouvel élément dans le dictionnaire, c.à.d.

$$\tilde{\mathbf{P}}_{n+1} = \theta^{-1}\left[\mathbf{P}_n - \frac{\theta^{-1}\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1 + \theta^{-1}\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}\right]$$

et

$$\tilde{\boldsymbol{A}}_{n+1} = \boldsymbol{A}_n + \mathbf{P}_{n+1}\mathbf{h}_{n+1}(\boldsymbol{\delta}_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{A}_n)$$

Dans la deuxième phase, $\boldsymbol{A}_{n+1}$ est calculée d'après l'expression (3.18) :

$$\begin{aligned}\boldsymbol{A}_{n+1} &= \mathbf{P}_{n+1}\mathbf{H}_{n+1}^t\boldsymbol{\Theta}_{n+1}\boldsymbol{D}_{n+1} \\ &= \begin{bmatrix}\tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m \\ \mathbf{0}_m^t & 0\end{bmatrix}\begin{bmatrix}\mathbf{H}_n^t & \mathbf{h}_{n+1} \\ \mathbf{0}_n^t & h_0\end{bmatrix}\begin{bmatrix}\theta\boldsymbol{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{D}_n \\ \boldsymbol{\delta}_{n+1}^t\end{bmatrix} \\ &\quad + \frac{1}{s}\begin{bmatrix}\mathbf{q}\mathbf{q}^t & -\mathbf{q} \\ -\mathbf{q}^t & 1\end{bmatrix}\begin{bmatrix}\mathbf{H}_n^t & \mathbf{h}_{n+1} \\ \mathbf{0}_n^t & h_0\end{bmatrix}\begin{bmatrix}\theta\boldsymbol{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{D}_n \\ \boldsymbol{\delta}_{n+1}^t\end{bmatrix} \\ &= \boldsymbol{A}_{n+1}^{(1)} + \boldsymbol{A}_{n+1}^{(2)}\end{aligned} \tag{3.25}$$

Or, et en se basant sur (3.20), le développement de $\boldsymbol{A}_{n+1}^{(1)}$ donne :

$$\boldsymbol{A}_{n+1}^{(1)} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1}(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\mathbf{d}_n + \mathbf{h}_{n+1}\boldsymbol{\delta}_{n+1}^t \\ \mathbf{0}_k^t\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1}\mathbf{H}_{n+1}^t\boldsymbol{\Theta}_{n+1}\boldsymbol{D}_{n+1} \\ \mathbf{0}_k^t\end{bmatrix} = \begin{bmatrix}\tilde{\boldsymbol{A}}_{n+1} \\ \mathbf{0}_k^t\end{bmatrix} \tag{3.26}$$

En développant $\boldsymbol{A}_{n+1}^{(2)}$ on obtient :

$$\begin{aligned}\boldsymbol{A}_{n+1}^{(2)} &= \frac{1}{s}\begin{bmatrix}\mathbf{q}\mathbf{q}^t(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\boldsymbol{D}_n + \mathbf{h}_{n+1}\boldsymbol{\delta}_{n+1}^t) - h_0\mathbf{q}\boldsymbol{\delta}_{n+1}^t \\ -\mathbf{q}^t(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\boldsymbol{D}_n + \mathbf{h}_{n+1}\boldsymbol{\delta}_{n+1}^t) + h_0\boldsymbol{\delta}_{n+1}^t\end{bmatrix} \\ &= \frac{1}{s}\begin{bmatrix}\mathbf{q}\mathbf{q}^t(\mathbf{H}_{n+1}^t\boldsymbol{\Theta}_{n+1}\boldsymbol{D}_{n+1}) - h_0\mathbf{q}\boldsymbol{\delta}_{n+1}^t \\ -\mathbf{q}^t(\mathbf{H}_{n+1}^t\boldsymbol{\Theta}_{n+1}\boldsymbol{D}_{n+1}) + h_0\boldsymbol{\delta}_{n+1}^t\end{bmatrix} \\ &= \frac{1}{s}\begin{bmatrix}-\mathbf{q}(h_0\boldsymbol{\delta}_{n+1}^t - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{A}}_{n+1}) \\ h_0\boldsymbol{\delta}_{n+1}^t - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{A}}_{n+1}\end{bmatrix} \\ &= \frac{1}{s}\begin{bmatrix}-\mathbf{q} \\ 1\end{bmatrix}(h_0\boldsymbol{\delta}_{n+1}^t - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{A}}_{n+1}) \end{aligned} \tag{3.27}$$

En combinant (3.26) et (3.27), on obtient :

$$\boldsymbol{A}_{n+1} = \begin{bmatrix}\tilde{\boldsymbol{A}}_{n+1} \\ \mathbf{0}_k^t\end{bmatrix} + \frac{1}{(h_0+\zeta\theta^{n+1})(h_0 - \mathbf{h}_{n+1}^t\mathbf{q})}\begin{bmatrix}-\mathbf{q} \\ 1\end{bmatrix}(h_0\boldsymbol{\delta}_{n+1}^t - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{A}}_{n+1}) \tag{3.28}$$

Pour simplifier les expressions trouvées ci-dessus, on peut négliger le paramètre de régularisation $\zeta = 0$ et prendre un facteur d'oubli $\theta = 1$. Dans ce cas, les expressions (3.22) et (3.23) deviennent :

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \frac{\mathbf{P}_n\mathbf{h}_{n+1}}{1+\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}(\boldsymbol{\delta}_{n+1}^t - \mathbf{h}_{n+1}^t\boldsymbol{A}_n) \tag{3.29}$$

$$\mathbf{P}_{n+1} = \mathbf{P}_n - \frac{\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1+\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}} \tag{3.30}$$

et les expressions (3.24) et (3.28) deviennent :

$$\mathbf{P}_{n+1} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m \\ \mathbf{0}_m^t & 0\end{bmatrix} + \frac{1}{1-\mathbf{h}_{n+1}^t\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}} \times \begin{bmatrix}-\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1} \\ 1/h_0\end{bmatrix}\begin{bmatrix}-(\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1})^t & 1/h_0\end{bmatrix} \tag{3.31}$$

$$\boldsymbol{A}_{n+1} = \begin{bmatrix}\tilde{\boldsymbol{A}}_{n+1} \\ \mathbf{0}_k^t\end{bmatrix} + \frac{1}{1-\mathbf{h}_{n+1}^t\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}}\begin{bmatrix}\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1} \\ 1/h_0\end{bmatrix}(\boldsymbol{\delta}_{n+1}^t - \mathbf{h}_{n+1}^t\tilde{\boldsymbol{A}}_{n+1}) \tag{3.32}$$

3.6 Adaptation du dictionnaire dans le cas multiples sorties

L'adaptation du dictionnaire dans le cas de sortie unique a prouvé son efficacité pour diminuer l'ordre du modèle et/ou l'erreur quadratique instantanée, comme étudié dans le chapitre 2. La méthode d'adaptation est basée sur le gradient stochastique de l'erreur quadratique instantanée par rapport aux éléments du dictionnaire. Dans le cas de la sortie unique, et à l'instant n, on a une seule erreur instantanée qui est la différence entre la sortie désirée du modèle et la sortie actuelle du modèle. Dans ce chapitre, on traite le modèle à sorties multiples où désormais on a k erreurs instantanées de la forme $e_n^{(j)} = d_n^{(j)} - y_n^{(j)}$ avec $j = 1 \cdots k$. Donc, à l'instant n, on a un vecteur d'erreurs instantanées tel que $\boldsymbol{\xi}_n = (e_n^{(1)}, \cdots, e_n^{(k)})^t$ et

il est de taille $k \times 1$. L'idée est donc l'adaptation du dictionnaire en utilisant le gradient de la norme ℓ_2 de $\boldsymbol{\xi}_n$ par rapport aux éléments du dictionnaire dans le but de réduire l'erreur quadratique.

Le modèle à multiples sorties, on a un seul dictionnaire et les éléments de ce dernier sont choisis tout en respectant un critère de cohérence. Quelque soit l'heuristique d'adaptation, le dictionnaire doit rester cohérent. Un élément du dictionnaire est toujours adapté suivant le principe (2.7) :

$$\mathbf{u}_{w_i}^A = \mathbf{u}_{w_i} - \nu_n \mathbf{g}_{w_i} \qquad \forall i = 1...m$$

ν_n est le pas du gradient et la seule différence ici c'est que $\mathbf{g}_{w_i}$ est le gradient de la norme ℓ_2 du vecteur d'erreurs instantannées $\boldsymbol{\xi}_n$ par rapport à l'élément $\mathbf{u}_{w_i}$:

$$\mathbf{g}_{w_i} = \nabla_{\mathbf{u}_{w_i}} \|\boldsymbol{\xi}_n\|^2 \tag{3.33}$$

Puisque $\|\boldsymbol{\xi}_n\|^2 = (d_n^{(1)} - y_n^{(1)})^2 + \cdots + (d_n^{(k)} - y_n^{(k)})^2$ et en se référant à (3.2), (3.33) devient :

$$\begin{aligned} \mathbf{g}_{w_i} &= -2\Big(e_n^{(1)} \frac{\partial y_n^{(1)}}{\partial \mathbf{u}_{w_i}} + e_n^{(2)} \frac{\partial y_n^{(2)}}{\partial \mathbf{u}_{w_i}} + \cdots + e_n^{(k)} \frac{\partial y_n^{(k)}}{\partial \mathbf{u}_{w_i}}\Big) \\ &= -2\Big(e_n^{(1)} \alpha_{n,i}^{(1)} + \cdots + e_n^{(k)} \alpha_{n,i}^{(k)}\Big) \frac{\partial \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} \end{aligned} \tag{3.34}$$

L'heuristique d'adaptation diffère avec le type de la fonction noyau. Par exemple, dans le cas d'une fonction noyau RBF on adapte tous les éléments du dictionnaire à chaque itération, mais si on a une fonction noyau polynomiale, à chaque itération on adapte un seul élément du dictionnaire.

3.6.1 L'expression du gradient dans le cas d'un noyau radial (RBF)

3.6.1.1 Noyau Gaussien

La fonction du noyau Gaussien est de la forme :

$$\kappa(\mathbf{u}_i, \mathbf{u}_j) = \exp\Big(-\frac{\|\mathbf{u}_i - \mathbf{u}_j\|^2}{2\sigma^2}\Big)$$

avec σ est la bande passante du noyau. A l'instant n, pour une entrée $\mathbf{u}_n$, on a :

$$\frac{\partial \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} = \frac{1}{\sigma^2}(\mathbf{u}_n - \mathbf{u}_{w_i})\kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) \tag{3.35}$$

Parsuite, l'expression du gradient (3.34) devient :

$$\mathbf{g}_{w_i} = \frac{-2}{\sigma^2}\Big(e_n^{(1)} \alpha_{n,i}^{(1)} + \cdots + e_n^{(k)} \alpha_{n,i}^{(k)}\Big) \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})(\mathbf{u}_n - \mathbf{u}_{w_i}).$$

3.6.1.2 Noyau Laplacien

La fonction du noyau Laplacien est de la forme :

$$\kappa(\mathbf{u}_i, \mathbf{u}_j) = \exp\Big(-\frac{\|\mathbf{u}_i - \mathbf{u}_j\|}{2\sigma^2}\Big)$$

avec σ est la bande passante du noyau. A l'instant n, pour une entrée $\mathbf{u}_n$, on a :

$$\frac{\partial \kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\partial \mathbf{u}_{w_i}} = \frac{1}{2\sigma^2 \|\mathbf{u}_n - \mathbf{u}_{w_i}\|}(\mathbf{u}_n - \mathbf{u}_{w_i})\kappa(\mathbf{u}_n, \mathbf{u}_{w_i}) \tag{3.36}$$

Ceci conduit à un gradient de la forme :

$$\mathbf{g}_{w_i} = -\frac{\left(e_n^{(1)}\alpha_{n,i}^{(1)} + \cdots + e_n^{(k)}\alpha_{n,i}^{(k)}\right)}{2\sigma^2 \|\mathbf{u}_n - \mathbf{u}_{w_i}\|}\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})\,(\mathbf{u}_n - \mathbf{u}_{w_i})$$

3.6.2 L'expression du gradient dans le cas d'un noyau polynomial

La fonction noyau polynomial est de la forme :

$$\kappa(\mathbf{u}_i, \mathbf{u}_j) = f(\mathbf{u}_i^t \mathbf{u}_j) = (1 + a\mathbf{u}_i^t \mathbf{u}_j)^\beta \tag{3.37}$$

avec a une constante et $\beta > 0$ est la puissance du noyau. Rappelons que cette fonction noyau doit être normalisée comme suit :

$$\frac{\kappa(\mathbf{u}_i, \mathbf{u}_j)}{\sqrt{\kappa(\mathbf{u}_i, \mathbf{u}_i)}\sqrt{\kappa(\mathbf{u}_j, \mathbf{u}_j)}}$$

en utilisant le même calcul fait dans le paragraphe (2.2.3.1) du chapitre 2, on obtient :

$$\mathbf{g}_{w_i} = -2\,a\,\beta\left(e_n^{(1)}\alpha_{n,i}^{(1)} + \cdots + e_n^{(k)}\alpha_{n,i}^{(k)}\right)\frac{\kappa(\mathbf{u}_n, \mathbf{u}_{w_i})}{\sqrt{\kappa(\mathbf{u}_n, \mathbf{u}_n)}\sqrt{\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}}\left(\frac{\mathbf{u}_n}{\kappa^*(\mathbf{u}_n, \mathbf{u}_{w_i})} - \frac{1}{2}\frac{\mathbf{u}_{w_i}}{\kappa^*(\mathbf{u}_{w_i}, \mathbf{u}_{w_i})}\right) \tag{3.38}$$

avec $\kappa^*(\mathbf{u}_i, \mathbf{u}_j) = (1 + a\mathbf{u}_i^t \mathbf{u}_j)$ c'est à dire $\kappa^*(\mathbf{u}_i, \mathbf{u}_j) = \kappa(\mathbf{u}_i, \mathbf{u}_j)$ pour $\beta = 1$.

3.6.3 Les heuristiques pour l'adaptation du Dictionnaire

L'heuristique de l'adaptation du dictionnaire est basée sur le respect de la contrainte de la cohérence du dictionnaire. Puisqu'on a un seul dictionnaire pour toutes les sorties du modèle, les mêmes heuristiques trouvées dans le paragraphe (2.2.2.3) pour les fonctions noyau radiales et dans le paragraphe (2.2.3.2) pour les fonctions noyau polynomiales seront adoptées dans le cas du modèle à sorties multiples. La seule différence dans ce cas est l'expression du gradient de la norme ℓ_2 de $\boldsymbol{\xi}_n$.

TABLE *3.1: Performance MOKAPA avec* $\mu_0 = 0.3$ *et* $\nu_0 = 0.05$

	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6	Sortie 7	Sortie 8
NMSE (Sans adaptation)	0.21691	0.11548	0.62603	0.03242	1.00880	0.69114	1.03240	1.20270
NMSE (Avec adaptation)	0.10505	0.07601	0.25562	0.022164	0.50270	0.44475	0.52830	0.69226
Diminution	51.57%	34.18%	59.17%	31.63%	50.17%	35.65%	48.83%	42.44%

3.7 Expérimentations

Dans cette section nous réalisons des simulations pour identifier des modèles à plusieurs sorties, avec et sans adaptation du dictionnaire. Le critère de performance utilisé est toujours l'erreur quadratique moyenne normalisée qui est calculée en utilisant les derniers 500 échantillons de chaque sortie suivant la formule :

$$\text{NMSE}^{(j)} = \frac{\sum_{n=N-500}^{N} \left(d_n^{(j)} - y_n^{(j)}\right)^2}{\sum_{n=N-500}^{N} \left(d_n^{(j)}\right)^2} \qquad j = 1 \cdots k \tag{3.39}$$

La taille du dictionnaire reste toujours une grandeur à contrôler. Les simulations sont réalisées en utilisant les deux algorithmes MOKAPA et MOKRLS. Rappelons que l'algorithme MOKNLMS n'est autre que l'algorithme MOKAPA en considérant le paramètre (nombre de collecteurs) $p = 1$.

3.7.1 Prédiction de signaux EMG

La simulation dans cette partie consiste à identifier huit signaux d'électromyogramme pour actions physiques (EMG Physical Action Data Sets) pris de [BL13] (Subset 1 - normal - running). On peut trouver les détails concernant ces signaux sur le site (UCI Machine Learning Repository - EMG Physical Action Data Set Data Set). Ces huit signaux (séries temporelles) sont considérés comme sorties du modèle à identifier et on a considéré seulement les premiers 2000 échantillons de chaque série.

3.7.1.1 Résultats avec l'algorithme MOKAPA

L'algorithme MOKAPA est utilisé avec un noyau Gaussien de bande passante $\sigma = 0.42$. Le seuil de cohérence choisi est $\mu_0 = 0.3$. Les paramètres de l'algorithme sont : $p = 3$ (nombre de collecteurs), $\varepsilon = 0.09$ et le pas de référence pour l'adaptation est $\nu_0 = 0.05$. Tous ces paramètres ont été choisis après un très grand nombre d'essais rigoureux pour trouver la meilleure combinaison. Une autre contrainte qui a influencé la sélection des paramètres est le but d'obtenir d'un dictionnaire essentiellement de même taille avec et sans adaptation.

Avec adaptation, la taille du dictionnaire obtenu est de 148 éléments contre 151 éléments sans adaptation (pratiquement on peut considérer qu'on a la même taille car la réduction est de 1.99% par rapport à la taille sans adaptation). Les courbes d'apprentissage sont montrées dans la figure (3.2) et les gains résultant des erreurs quadratiques moyennes normalisées pour les huit sorties sont montrés dans la

TABLE *3.2: Performance MOKRLS avec $\mu_0 = 0.3$ et $\nu_0 = 0.05$*

	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6	Sortie 7	Sortie 8
NMSE (Sans adaptation)	0.39397	0.43833	0.26720	0.70404	0.077164	0.10241	0.07745	0.12580
NMSE (Avec adaptation)	0.19617	0.25243	0.05901	0.48677	0.03004	0.03999	0.029872	0.03654
Diminution	50.21%	42.41%	77.91%	30.86%	58.07%	60.95%	61.43%	70.96%

TABLE *3.3: Comparaison entre les critères ALD (seuil 0.856) et cohérence ($\mu_0 = 0.3$) en utilisant l'algorithme MOKRLS.*

	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6	Sortie 7	Sortie 8
NMSE (Cohérence)	0.39397	0.43833	0.26720	0.70404	0.077164	0.10241	0.07745	0.12580
NMSE (ALD)	0.38944	0.46527	0.25078	0.72381	0.0.07779	0.1110	0.08226	0.12773
différence	-1.15%	+6.15%	-6.15%	+2.80%	+0.81%	+8.39%	+6.21%	+1.53%

table (3.1). A noter que les courbes sont tracées en lissant avec une fenêtre de largeur 50 pour mieux visualiser les résultats.

3.7.1.2 Résultats avec l'algorithme MOKRLS

Pour l'algorithme MOKRLS, on a utilisé le noyau Gaussien avec la même valeur de bande passante $\sigma = 0.42$. Sans adaptation, l'ordre du modèle (taille du dictionnaire) est de 382 éléments, alors que cette taille devient 376 éléments avec adaptation. On peut toujours supposer qu'on a la même taille car la réduction est minime (1.57%). Les essais sont faits en négligeant le coefficient de régularisation $\zeta = 0$ et en prenant le paramètre d'oubli $\theta = 1$. On a choisi le même seuil de cohérence $\mu_0 = 0.3$ et le même pas du gradient référence pour l'adaptation $\nu_0 = 0.05$ que dans le cas de l'algorithme MOKAPA. Les courbes d'apprentissage sont exposées dans la figure (3.3) et les gains résultant des erreurs quadratiques moyennes normalisées pour les huit sorties sont montrés dans la table (3.2). De même, les courbes sont tracées après lissage avec une fenêtre de largeur 50 pour mieux visualiser les résultats.

L'algorithme MOKRLS est aussi testé dans le cas du critère de dépendance linéaire [EMM04] (voir le paragraphe (1.8.2) en utilisant un seuil 0.856 pour obtenir approximativement la même taille du dictionnaire (378 éléments). L'algorithme sans adaptation avec le critère ALD est aussi étudié et les courbes d'apprentissage sont montrées dans la figure (3.3). Le tableau (3.3) sert à faire une comparaison entre les erreurs quadratiques moyennes normalisées pour le critère de cohérence et pour le critère ALD. La comparaison montre les performances rapprochées des deux critères, ceci justifie l'adaptation du dictionnaire en se basant sur le critère de cohérence vu sa simplicité et son coût calculatoire réduit par rapport au critère ALD.

TABLE *3.4: Performance MOKAPA avec* $\mu_0 = 0.4$ *et* $\nu_0 = 0.125$ *pour la prédiction des signaux EEG*

	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6	Sortie 7
NMSE (Sans adaptation)	0.12725	0.06932	0.06853	0.11307	0.16237	0.09944	0.10357
NMSE (Avec adaptation)	0.0471	0.01404	0.03187	0.02817	0.04355	0.01504	0.04305
Diminution	62.90%	79.74%	53.50%	75.08%	73.18%	84.88%	58.43%

	Sortie 8	Sortie 9	Sortie 10	Sortie 11	Sortie 12	Sortie 13
NMSE (Sans adaptation)	0.15305	0.11212	0.14998	0.08821	0.08821	0.02205
NMSE (Avec adaptation)	0.05106	0.02720	0.03731	0.02585	0.02585	0.00646
Diminution	66.64%	75.74%	75.12%	70.69%	70.69%	70.69%

3.7.2 Prédiction de signaux EEG

La diversification des simulations en utilisant des fonctions noyau différentes nous conduit à changer le sujet de l'expérimentation. Dans cette section, on cherche à identifier treize signaux d'éléctroencephalogram (EEG - Planning Relax Data Set) pris de [BL13]. Les détails concernant ces signaux peuvent être trouvées sur le site (UCI Machine Learning Repository - Planning Relax Data Set). Ces treize séries temporelles sont considérées comme sorties du modèle à identifier et elles sont de 182 échantillons chacune.

L'algorithme MOKAPA est utilisé avec la configuration suivante : $p = 3$ et $\varepsilon = 0.09$. Un noyau polynomial de puissance $\beta = 3$ est adopté avec un seuil de cohérence de $\mu_0 = 0.4$. Sans adaptation l'ordre du modèle est 10 tandis qu'avec adaptation (pas du gradient référence $\nu_0 = 0.125$) la taille du dictionnaire augmente à 12 éléments, ceci implique une augmentation de 20%. Les courbes d'apprentissage sont montrées dans la figure (3.4) et les erreurs quadratiques moyennes normalisées avec leurs gains résultants sont montrées dans la table (3.4).

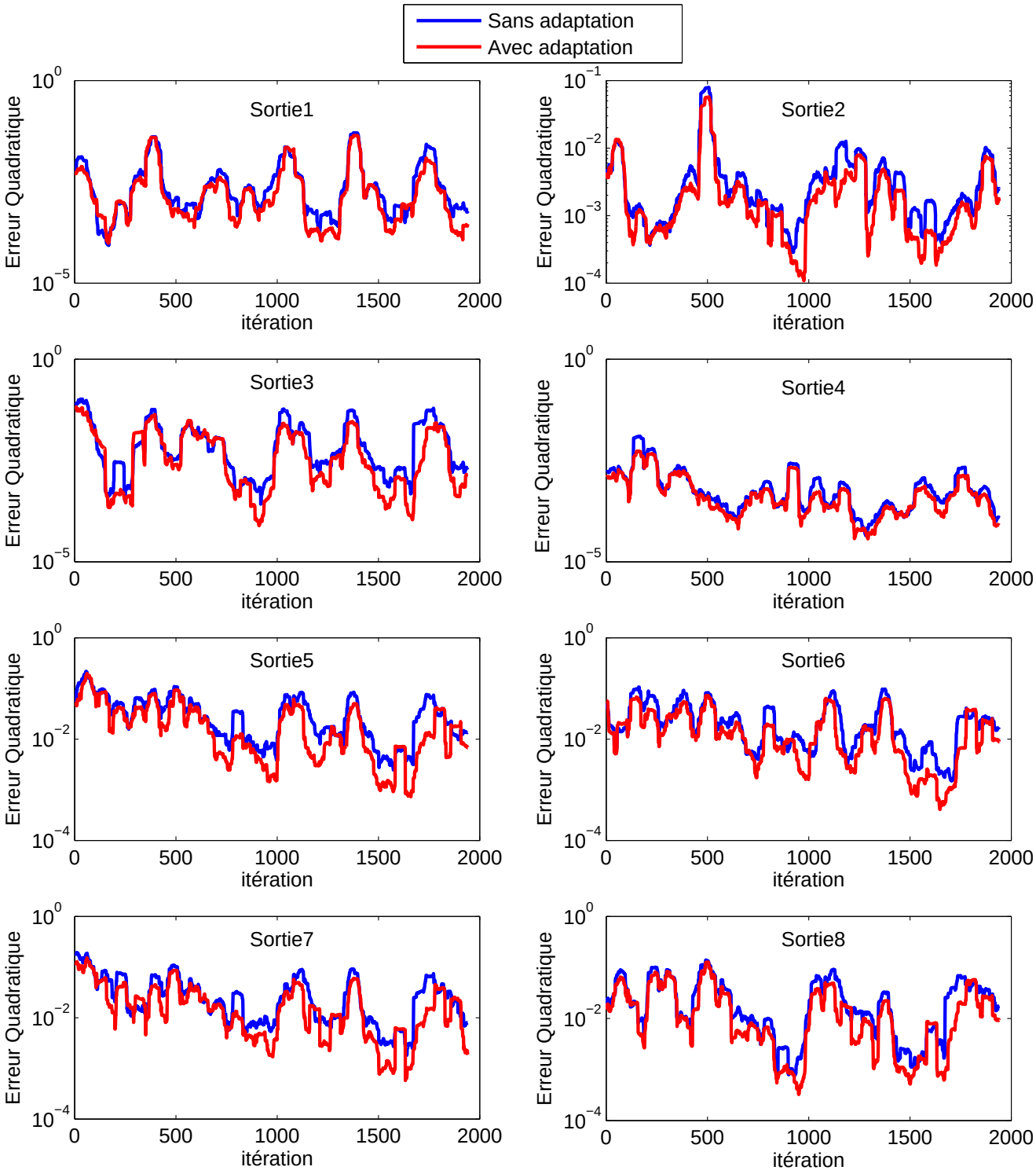

FIGURE *3.2: Comparaison avec et sans adaptation pour un noyau Gaussien avec* $\sigma = 0.42$ *- MOKAPA*

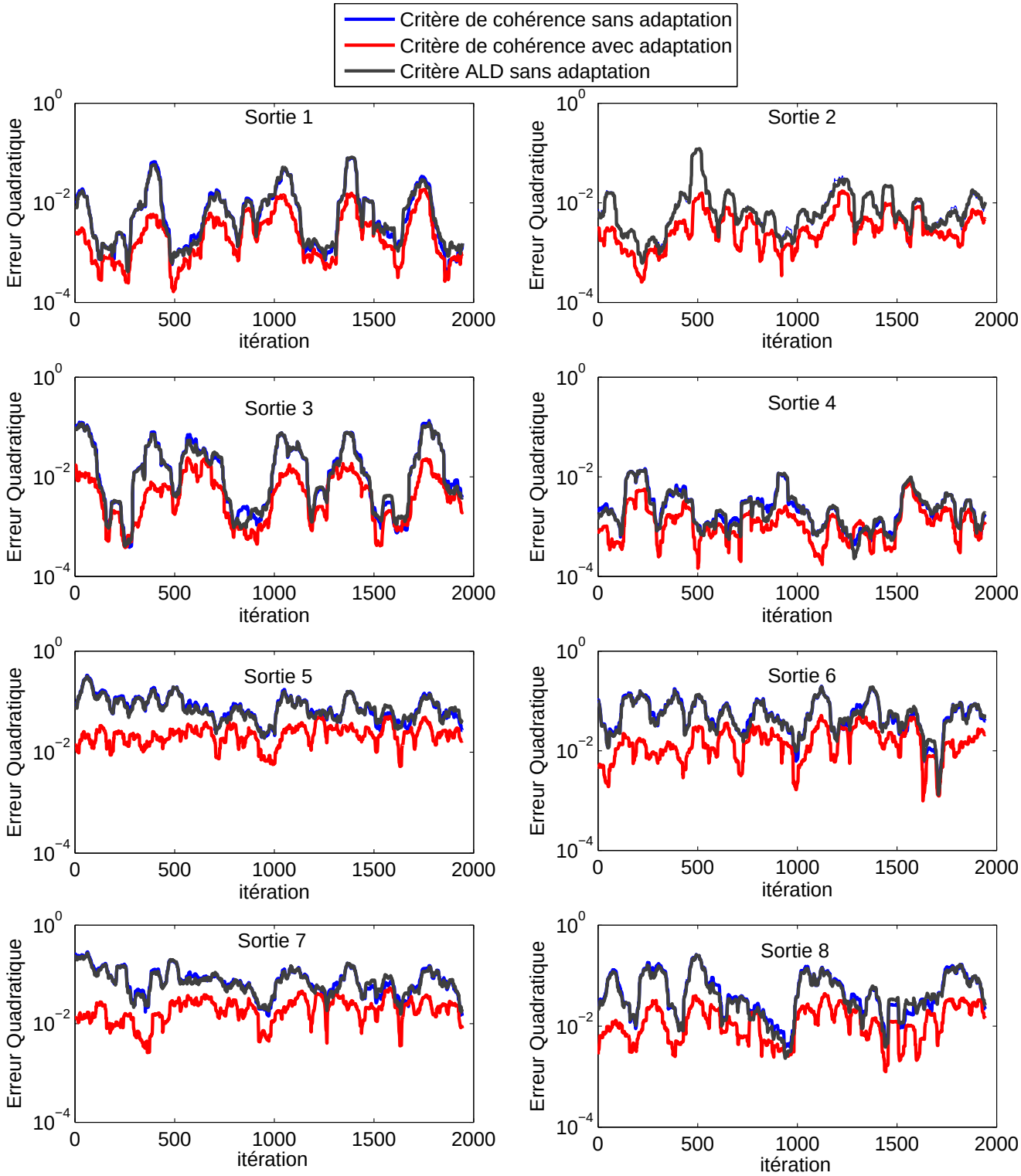

FIGURE *3.3: Comparaison pour le critère de cohérence, avec et sans adaptation, et le critère ALD sans adaptation, pour un noyau Gaussien avec* $\sigma = 0.42$ *- MOKRLS*

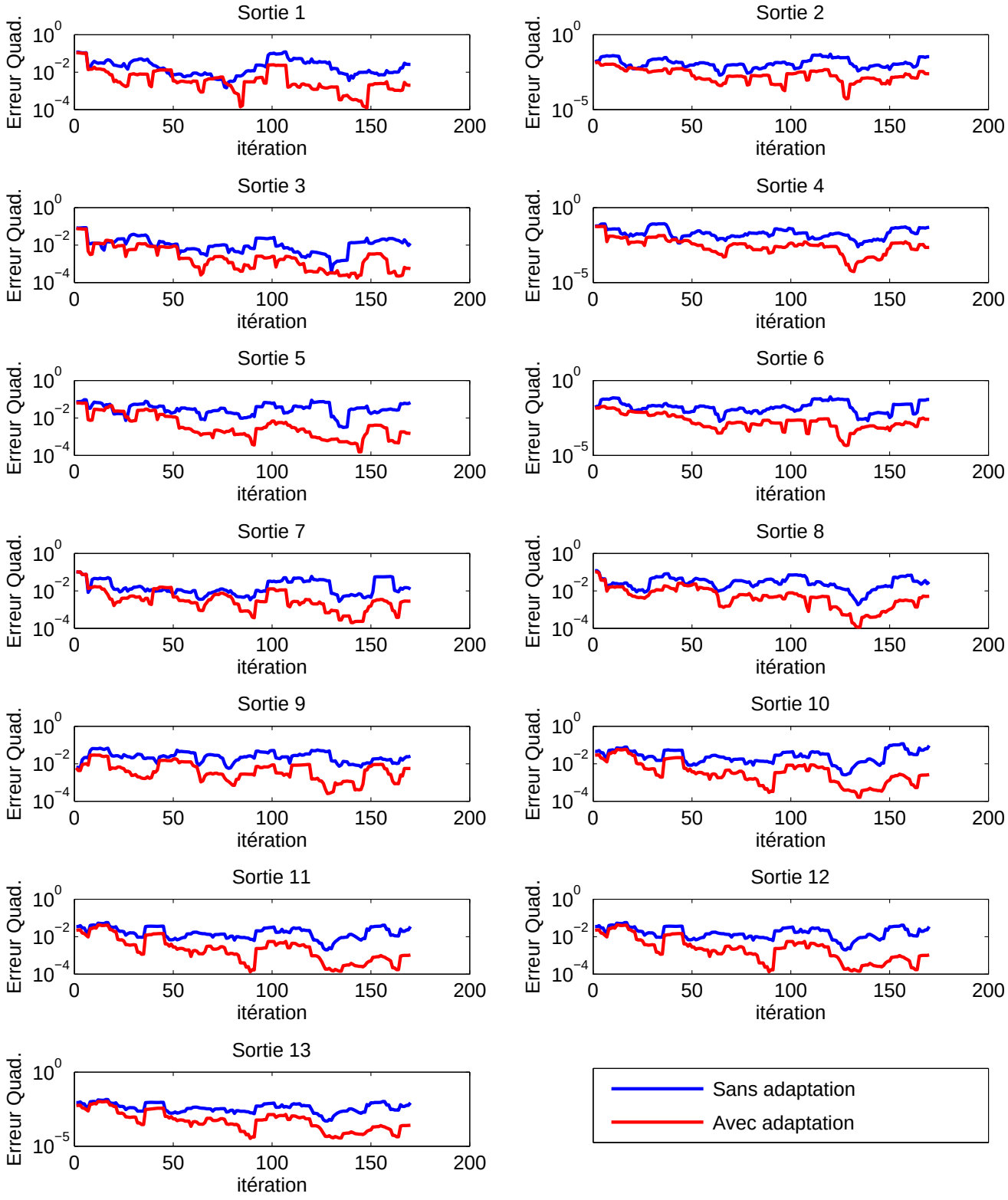

FIGURE *3.4: Comparaison avec et sans adaptation pour un noyau polynomial* $\beta = 3$, $\mu_0 = 0.4$ *et* $\nu_0 = 0.125$ - *MOKAPA*

Deuxième partie

Application : contrôle d’un palier magnétique actif

CHAPITRE 4

Contrôle d'un palier magnétique actif

Sommaire

Les machines tournantes soufrent toujours des problèmes techniques dont le remède n'est pas évident. L'origine de ces problèmes est principalement dûe aux contraintes liées aux paliers. Parmis ces problèmes il y a le frottement qui entraîne de l'échauffement, la limitation de la vitesse de rotation et la dégradation des paliers mécaniques. La lubrification est le seul moyen pour réduire ces effets nuisibles mais d'autres problèmes surgissent tels que l'étanchéité, par exemple.

Le palier magnétique actif (PMA) est une solution prometteuse pour surmonter les problèmes des machines tournantes surtout pour assurer une grande vitesse de rotation et ceci est dû à l'absence de frottement. Il est également très utile pour les machines tournantes placées dans des conditions spéciales telles que les milieux à température élevée et le vide [MY86]. Cependant, le contrôle d'un PMA reste toujours une tâche difficile car il s'agit d'assurer une bonne lévitation de la partie tourante (rotor) pour éviter le contact de cette dernière avec la partie fixe (stator), en présence de perturbations induites. Les forces électromagnétiques qui assurent la lévitation sont contrôlées en-ligne et instantanément. Etant des modèles non-linéaires à entrées multiples et sorties multiples (MIMO), les PMA peuvent être contrôlés par plusieurs méthodes dont chacune possède ses propres avantages et inconvénients. Dans ce chapitre on va utiliser les méthodes à noyau pour contrôler les PMA en utilisant le critère de cohérence comme critère de parcimonie, et on propose d'adapter le dictionnaire obtenu par les méthodes explorées dans le chapitre 2.

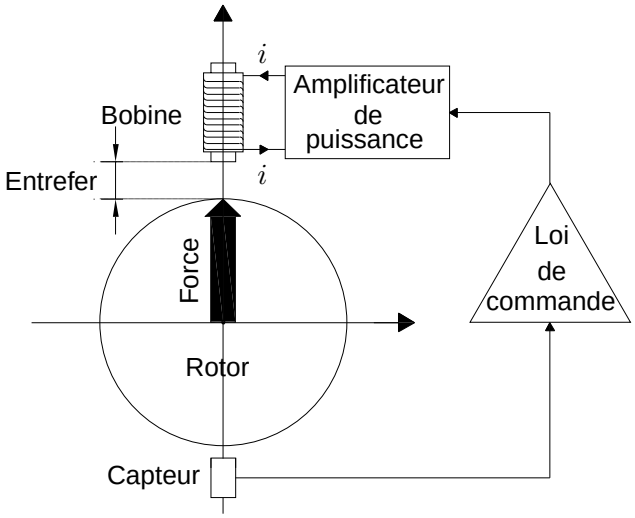

FIGURE *4.1: Représentation schématique d'un actionneur [Mir98].*

4.1 Introduction

Le contrôle d'un Palier Magnétique Actif (PMA) représente un défi pour obtenir une bonne sustentation de la partie tournante de la machine (rotor) pour éviter tout contact entre cette partie et la partie fixe (stator) [JD85] éliminant ainsi les frottements dûs au contact. La sustentation est assurée par des forces électromagnétiques créées par des électro-aimants et ceci en modulant le courant qui traverse leur bobinage. Plusieurs défauts et perturbations mènent à la non-coïncidence de l'axe géometrique du PMA avec l'axe d'inertie [ANDMC06, AN10]. Pour éliminer ces défauts, plusieurs types de contrôle ont été utilisés comme le contrôle en mode glissant [TCK04, SN98, ARMNAH09], la linéarisation entrée-sortie [CC92, CDMC96, SW95], et le contrôle à logique floue [NY91, CDMC96, MC96]. Plus récemment, les réseaux de neurones ont été utilisés pour contrôler un PMA [ANDMC06, AN10, HMA10].

Dans ce Chapitre, nous proposons une nouvelle méthode pour contrôler un PMA en utilisant les méthodes à noyau avec un algorithme adaptatif en ligne. Les avantages de la méthode proposée sont multiples : d'une part, elle ne comprend pas les phases préliminaires d'apprentissage, et d'autre part, les méthodes à noyau atteignent le minimum global du critère de performance, au contraire de celles à réseaux neurones qui, à titre d'exemple, risquent d'être piégées dans des minima locaux. Des algorithmes adaptatifs avec le critère de cohérence sont étudiés, avec et sans adaptation du dictionnaire, pour comparer l'amélioration des courbes d'apprentissage qui indiquent les performances du modèle estimé.

4.2 Le palier magnétique actif

4.2.1 Notions de base

Le contrôle d'un PMA est basé sur le principe d'un *Actionneur*. Cet actionneur est représenté par un électro-aimant fixé sur le stator de la machine et qui agit sur le rotor en exerçant une force électromagnétique sur ce dernier. Le stator et le rotor sont séparés par un intervalle appelé *entrefer* dont la valeur nominale est notée e_0. L'électro-aimant est constitué par une bobine associée à un amplificateur de puissance qui assure son alimentation par un courant modulant (ou une tension) afin d'exercer une force d'attraction sur le rotor. La position de ce dernier est mesurée par un capteur de position qui sert comme entrée pour commander l'électro-aimant. Voir figure (4.1) pour une illustration de cet actionneur. Puisqu'un électro-aimant exerce seulement une force d'attraction sur le rotor, il n'est pas généralement utilisé seul, mais comme élément d'un axe de contrôle [Bon08]. Voici dans la suite quelques définitions utiles.

Définition 4.1. *Axe de contrôle*
Un axe de contrôle est formé par deux actionneurs en vis-à-vis couplés pour exercer des forces positives et négatives par rapport à leur axe de symétrie dans le but de garantir un bon positionnement du rotor suivant la direction de cet axe de contrôle.

Définition 4.2. *Plan de contrôle, Centreur*
Un plan de contrôle est constitué de deux axes de contrôle perpendiculaires (voir l'illustration dans la figure 4.2). Ce groupe est appelé aussi centreur*, sa fonction étant de maintenir le rotor à une position donnée, soit le centre figuré par l'origine, soit de lui faire suivre une trajectoire généralement circulaire [Mir98]. En général, les commandes des deux axes de contrôle sont indépendantes, mais on peut toujours coupler les commandes pour mieux éliminer les perturbations dûes aux défauts géométriques de l'arbre de rotation [MC98].*

Définition 4.3. *Butée*
Une butée est une suspension magnétique axiale permettant de maintenir le rotor le long de son axe principal [Bon08]. On peut l'assimiler à un axe de contrôle suivant la direction de l'axe x.

Définition 4.4. *Broche*
Une broche est formée généralement de deux plans de contrôle et d'une butée dont le but est de contrôler cinq degrés de liberté. Les figures (4.3) et (4.4) donnent une illustration d'une broche.

4.2.2 Défauts et perturbations d'un PMA

La broche d'un PMA présente des défauts qui mènent à des perturbations dans le fonctionnement du système et l'idée fondamentale derrière le contrôle d'un PMA est d'éliminer ces perturbations. Voici les trois défauts les plus gênants d'un PMA.

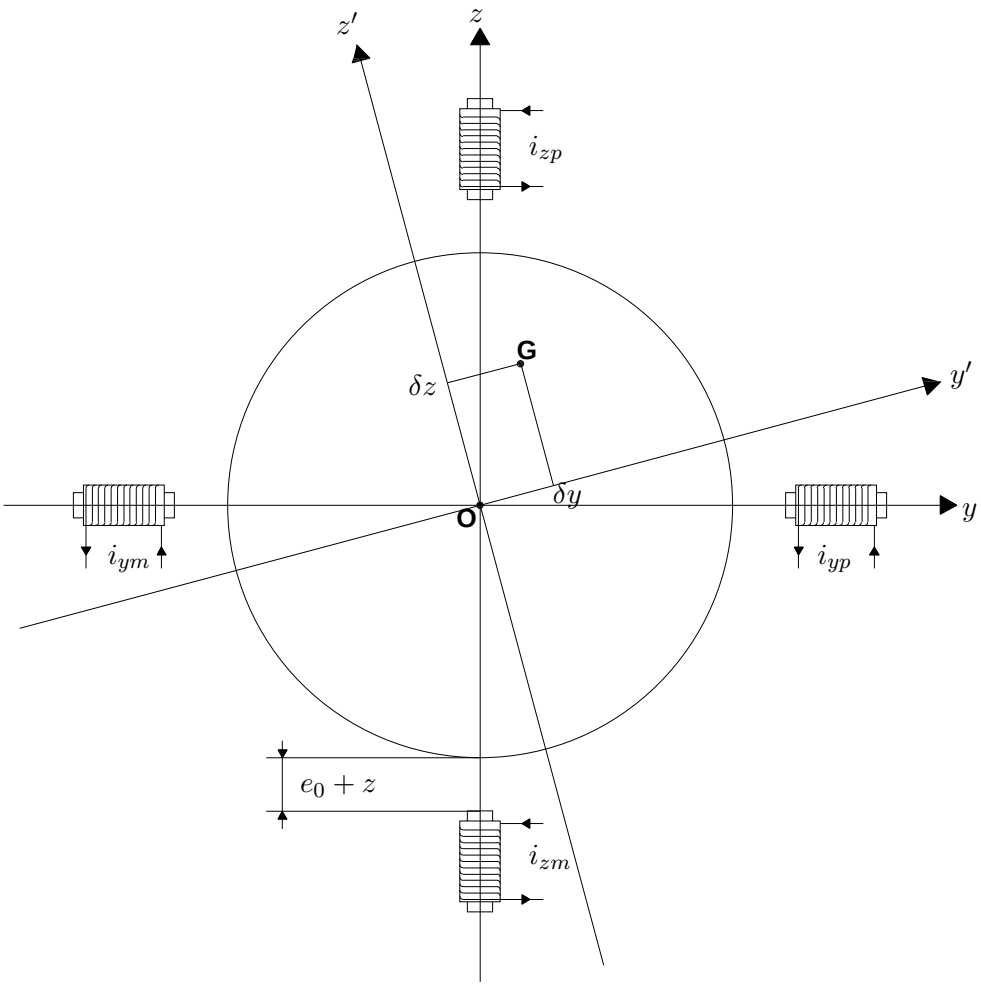

FIGURE *4.2: Illustration d'un plan de contrôle ($y - z$).*

Le balourd : Le *balourd* est un défaut qui résulte de la non-coïncidence de l'axe d'inertie et de l'axe géométrique du rotor. En rotation, le rotor tend à tourner autour de son axe d'inertie, ce qui introduit des perturbations dans les mesures de position, car les capteurs de positions mesurent celles de l'axe géométrique. Pour éliminer ces perturbations, il faut contrôler la rotation du rotor afin qu'il tourne autour de son axe géométrique.

Le faux rond : La mesure des positions a pour référence la surface du rotor qui peut présenter des irrégularités. En rotation, les mesures présentent des variations dûes à ces irrégulatités. Ceci se traduit par un bruit de mesure variable suivant la vitesse de rotation du rotor. Ce défaut devient de plus en plus moins influençant avec les progrès technologiques dans la fabrication des PMA.

La flexibilité du rotor : Le rotor présente des fréquences de résonnance qui sont amorties par l'environnement mais qui peuvent rendre le système instable si elles sont excitées par les actionneurs

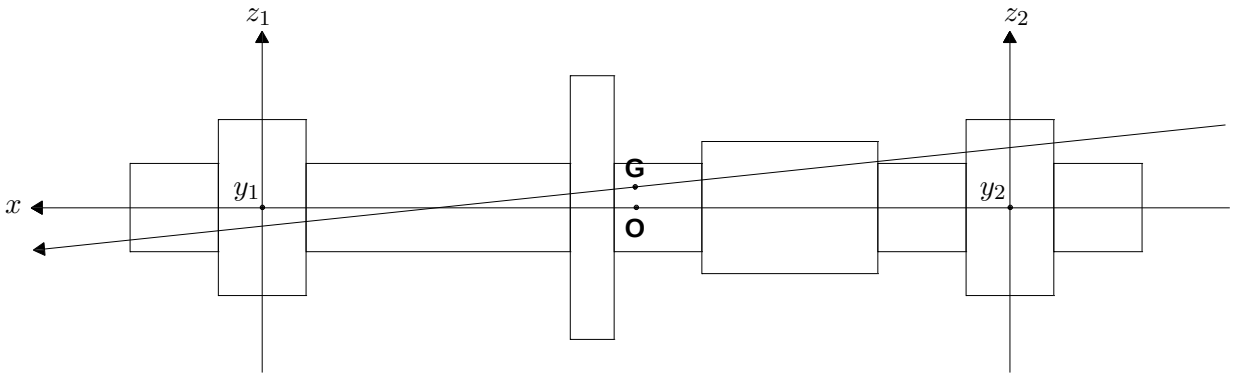

FIGURE *4.3: Illustration en 2D d'une broche du PMA.*

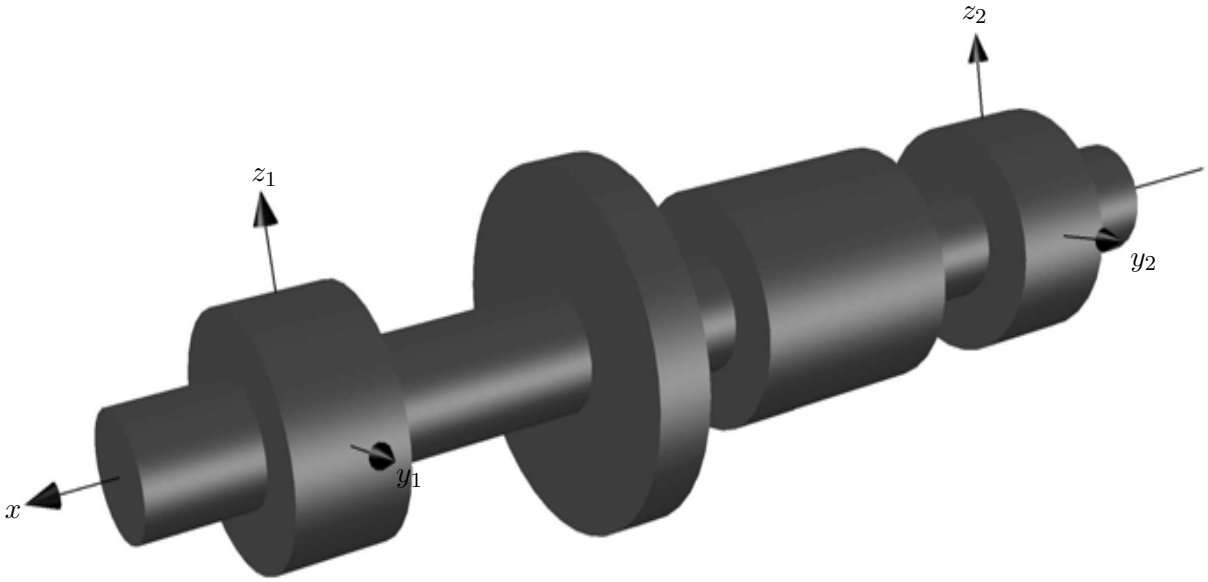

FIGURE *4.4: Illustration en 3D d'une broche du PMA.*

[Bon08]. Cette perturbation est importante et doit être contrôlée comme dans le cas d'un balourd.

4.2.3 Généralités sur le fonctionnement d'un PMA

Le principe de fonctionnement d'un PMA est basé sur les électro-aimants pour stabiliser le rotor dans la position désirée tout en éliminant les perturbations. Ces perturbations sont contrôlées suivant les deux plans de contrôle (y_1, z_1) et (y_2, z_2) ainsi que suivant la butée de l'axe x. Un courant injecté dans un électro-aimant créé une force électromagnétique proportionnelle au carré de ce courant et inversement proportionnelle à l'entrefer entre le rotor et le stator [BDMV12].

L'ensemble du PMA est complexe, de plus il est fortement non linéaire. De ce fait, la majorité des méthodes proposées pour contrôler un PMA sont basées sur des modèles simplifiés en faisant des approximations, comme par exemple, la supposition que les contrôles des axes d'un plan sont découplés et que l'induction mutuelle entre les électro-aimants n'est pas prise en considération.

Une section plane est représentée sous forme de disque tournant autour de l'axe x (voir schéma (4.2)) où **O** désigne le centre du repère référence (x, y, z) attaché au stator et **G** désigne le repère (x, y', z') attaché au rotor avec les coordonnées (0,δy,δz). Ce repère mobile tourne avec un angle ω par rapport au repère fixe. e_0 indique l'entrefer entre l'électro-aimant (fixé au stator) et le rotor. Les courants i_{yp} et i_{ym} (respectivement i_{zp} et i_{zm}) sont les courants de contrôle des électro-aimants suivant l'axe y (respectivement z). L'objectif du contrôle est de rendre $\delta y = 0$ et $\delta z = 0$.

Le système PMA est non linéaire ce qui exige l'utilisation des méthodes de contrôle non linéaires. Ceci implique plusieurs avantages comme par exemple l'utilisation non simultanée des actionneurs sur un même axe et ceci mène à la division par deux de l'énergie nécessaire pour sustenter le rotor suivant cet axe et par suite l'énergie totale nécessaire sera divisée par dix (sur les cinq axes) [CC92, CCL92, SW95, CDMC96, Ach08]. D'autres avantages pour le contrôle non linéaires sont cités dans [Ach08].

La démarche proposée par [CDMC96] pour commander un système PMA commence par le choix de la cinématique désirée du système qui mène à l'obtention des forces devant être appliquées, et parsuite aux courants nécessaires pour créer ces forces. Ces courants sont obtenues à partir du vecteur forces régularisé par un PID non linéaire. Les équations qui lient les forces aux courants sont données par :

$$
\begin{aligned}
F_{ix} &= \frac{1}{2}\frac{\upsilon\, i_{xp}^2}{(x-e_0)^2} - \frac{1}{2}\frac{\upsilon\, i_{xm}^2}{(x+e_0)^2} \\
F_{iy_1} &= \frac{\upsilon\, i_{y_1p}^2}{(-2e_0l_c + 2y_1l_c - d_{ca}y_1 + d_{ca}y_2)^2} - \frac{\upsilon\, i_{y_1m}^2}{(2e_0l_c + 2y_1l_c - d_{ca}y_1 + d_{ca}y_2)^2} \\
F_{iz_1} &= \frac{\upsilon\, i_{z_1p}^2}{(-2e_0l_c + 2z_1l_c - d_{ca}z_1 + d_{ca}z_2)^2} - \frac{\upsilon\, i_{z_1m}^2}{(2e_0l_c + 2z_1l_c - d_{ca}z_1 + d_{ca}z_2)^2} \\
F_{iy_2} &= \frac{\upsilon\, i_{y_2p}^2}{(-2e_0l_c + 2y_2l_c - d_{ca}y_2 + d_{ca}y_1)^2} - \frac{\upsilon\, i_{y_2m}^2}{(2e_0l_c + 2y_2l_c - d_{ca}y_2 + d_{ca}y_1)^2} \\
F_{iz_2} &= \frac{\upsilon\, i_{z_2p}^2}{(-2e_0l_c + 2z_2l_c - d_{ca}z_2 + d_{ca}z_1)^2} - \frac{\upsilon\, i_{z_2m}^2}{(2e_0l_c + 2z_2l_c - d_{ca}z_2 + d_{ca}z_1)^2}
\end{aligned}
\tag{4.1}
$$

$l_c =$ Distance entre le centre géométrique et les capteurs

$d_{ca} =$ Distance entre le capteur et l'actionneur correspondant

$\upsilon = \frac{\varsigma_0 \mathrm{N}^2 \mathrm{S}}{2}$

$\varsigma_0 =$ Perméabilité de l'air

$\mathrm{N} =$ Nombre de spires de l'électro-aimant

$\mathrm{S} =$ Surface vue par l'entrefer

L'obtention des équations (4.1) est détaillée dans [Mir98]. Ces cinq équations contiennent dix inconnues (deux courants pour chaque axe de contrôle) d'où la nécessité de fixer, sur chaque axe, un courant comme paramètre pour calculer l'autre. Puisqu'on alimente un seul électro-aimant sur chaque axe à chaque instant, le courant paramètre sera fixé à zéro. Le choix du courant paramètre sera comme suit : si la force désirée est positive (respectivement négative), c'est le courant qui crée une accélération négative (respectivement positive) qui est mis à zéro.

L'inconvénient de cette méthode est que lorsque la force désirée tend vers zéro, le courant correspondant qui génère cette force tend vers zéro et donc la tension appliquée à l'électro-aimant, qui est proportionnelle à la dérivée de ce courant, tend vers l'infini. Dans un tel cas, il devient impossible aux circuits de contrôle de générer cette tension. La solution à ce problème est la suivante : on fixe une force limite F_{lim} choisie d'une façon arbitraire. Si la force désirée $F_{desiree}$ est supérieure à la force limite, on applique le calcul précédent mais dans le sens contraire, le courant paramètre est alors calculé à partir du polynôme d'interpolation suivant (la force désirée est supposée positive, le calcul du polynôme pour une force désirée négative est alors obtenu d'une façon symétrique) :

$$i_m = c_2 F_{desiree}^2 + c_1 F_{desiree} + c_0 \tag{4.2}$$

avec

$$c_2 = \sqrt{\frac{1}{8F_{lim}^3 F_k}} \qquad c_1 = -\sqrt{\frac{1}{2F_{lim}F_k}} \qquad c_0 = \sqrt{\frac{F_{lim}}{8F_k}} \quad \text{et} \quad F_k = \frac{\upsilon}{2}$$

4.2.4 Schéma bloc en boucle fermée pour le contrôle d'un PMA

Le schéma bloc en boucle fermée utilisé dans les simulations est montré dans la figure (4.5). Dans ce schéma, on a le vecteur $\mathbf{q}_d = (0,0,0,0,0)^t$ des cinq positions désirées du rotor par rapport aux deux plans de contrôle (y_1,z_1), (y_2,z_2) et l'axe x, le vecteur des positions obtenues à la sortie du modèle $\mathbf{q}_s$ et le vecteur **e** qui représente l'erreur entre les positions désirées et les positions instantanées. $\mathbf{F} = (F_{ix}, F_{iy_1}, F_{iz_1}, F_{iy_2}, F_{iz_2})^t$ est le vecteur des forces désirées pour contrôler le PMA. Les vecteurs précédents sont tous de taille 5×1 tandis que le vecteur $\mathbf{I} = (i_{xp}, i_{xm}, i_{y_1p}, i_{y_1m}, i_{z_1p}, i_{z_1m}, i_{y_2p}, i_{y_2m}, i_{z_2p}, i_{z_2m},)^t$ qui n'est autre que le vecteur des courants appliqués sur les électro-aimants, est de taille 10×1. Le contrôle unique dans ce schéma est réalisé en

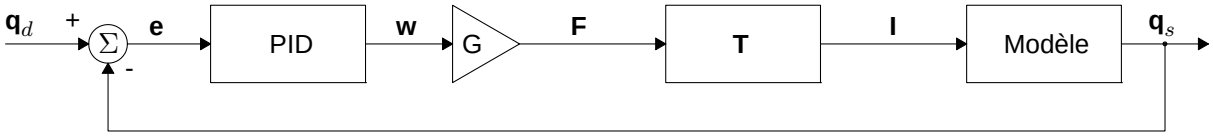

FIGURE *4.5: Schéma bloc en boucle fermée [Ach08].*

utilisant un bloc PID classique (proportional-integral-derivative controller) qui a comme entrée le vecteur **e** et comme sortie le vecteur **w**. Le bloc G représente un amplificateur de gain G.

Le but est de minimiser la norme du vecteur

$$\mathbf{w} = k_p\mathbf{e} + k_d\frac{d\mathbf{e}}{dt} + k_i\int \mathbf{e}dt$$

Les paramètres k_p, k_d et k_i indiquent les gains du PID. Le bloc **T** (Forces/Courants) représente la linéarisation entrée-état décrit dans (4.1) et le bloc Modèle n'est autre que le modèle du PMA. À signaler, que le modèle utilisé pour le PMA est celui dérivé dans [Ach08] avec toutes les approximations faites et toutes les linéarisations adoptées. Plus de détails sur le fonctionnement du PMA en boucle fermée, et toutes les équations qui gouvernent la cinématique du PMA se trouvent dans [Mir98, Ach08, Bon08].

4.2.5 Les caractéristiques de la broche utilisée pour les expérimentations

Dans le but de conduire des essais expérimentaux ainsi que pour faire des simulations, on a utilisé le banc d'essais appartenant au laboratoire Heudiasyc de l'UTC (Université de Technologie de Compiègne). Le banc de test est une broche classique dont le rotor est centré par dix actionneurs : deux plans de contrôle comportent chacun les deux axes y et z (huit actionneurs au total) ; l'axe x est contrôlé indépendamment par une butée axiale. Un capteur inductif, de mesure de position, est couplé avec chacune des bobines, légèrement décalé selon x d'une valeur d_{ca}. Chacune des dix bobines est indépendamment alimentée par une alimentation à découpage inversible en tension (mais pas en courant) de tension maximum 50 Volts et de courant maximum 6 Ampères. Elle est commandable en tension et en courant. Un moteur asynchrone est inclus dans la broche pour assurer l'entrainement en rotation du rotor avec possibilité de réglage des consignes manuellement. Les paramètres mécaniques et électriques de la broche, tels fournis par le constructeur, sont rapportés dans le tableau (4.1). Les positions initiales du rotor sont $x = y_1 = z_1 = y_2 = z_2 = 2\ 10^{-4}m$ qui correspondent au déplacement maximal du rotor tel qu'il est limité par le stator.

4.3 Simulations en utilisant un algorithme adaptatif à noyau

Dans cette section, une nouvelle méthode pour contrôler le palier magnétique actif sera exposée. L'objectif du contrôle est d'amener tous les axes de contrôle à des positions désirées ($x = y_1 = z_1 = y_2 = z_2 = 0$) à partir des positions initiales ($x = y_1 = z_1 = y_2 = z_2 = 2\ 10^{-4}m$) en l'absence de

TABLE *4.1: Paramètres constructeur de la broche expérimentale*

	Paramètres mécaniques	
M	$3.097kg$	masse du rotor
I_x	$8.589\ 10^{-4}kg.m^2$	Moment d'inertie du rotor selon l'axe x
I_y	$2.146\ 10^{-4}kg.m^2$	Moment d'inertie du rotor selon l'axe y
I_z	$2.146\ 10^{-4}kg.m^2$	Moment d'inertie du rotor selon l'axe z
e_0	$0.4mm$	entrefer nominal des actionneurs
d_{ca}	$18mm$	Distance entre les capteurs et l'actionneur correspondant
ω_{max}	$30000tr/mn$	Vitesse de rotation maximale
	Paramètres électriques	
R	0.2Ω	Résistance des actionneurs des plans de contrôle
L	$3mH$	Inductance des actionneurs des plans de contrôle
υ	$1.2\ 10^{-6}mH.m$	$\upsilon = \frac{\varsigma_0 N^2 S}{2}$ pour les actionneurs des plans de contrôle, avec ς_0 = Perméabilité de l'air, N = Nombre de spires de l'électro-aimant et S= Surface vue par l'entrefer
R_x	1.6Ω	Résistance des actionneurs de la butée principale
L_x	$5.8mH$	Inductance des actionneurs de la butée principale
υ_x	$2.32\ 10^{-6}mH.m$	υ des actionneurs de la butée principale
	Paramètres capteurs et entrées	
Capteurs		$\pm 10V$ correspond à un déplacement de $\pm 25mm$
Entrée en tension		0 à $10V$ correspond à une tension de -50 à $50V$
Entrée en courant		0 à $10V$ correspond à un courant de 0 à $6A$

rotation. La méthode proposée est basée sur les algorithmes adaptatifs utilisant les méthodes à noyau. Mais avant d'exposer cette méthode, il faut revoir la méthode proposée dans [ANDMC06] pour contrôler le PMA à l'aide des réseaux de neurones multicouches (MLP pour Multi-Layer Perceptrons).

Le bloc MLP proposé dans [ANDMC06, Ach08] est intégré dans le schéma en boucle fermée comme indique la figure (4.6). Ce bloc a pour entrée le vecteur **I** des dix courants plus un biais et a pour sortie un vecteur $\mathbf{\Phi}$ (de taille 5×1). Le modèle MLP contient trois couches : une couche pour l'entrée formée de onze neurones (dix courants + un biais), une couche cachée formée de P neurones et une couche de sortie formée de cinq neurones. Le vecteur de sortie $\mathbf{\Phi}$ est ajouté au vecteur $\mathbf{V}_h$ qui n'est autre que la sortie du bloc PID (accélération) amplifiée par le gain G pour donner le vecteur des cinq forces **F**. Le but de l'introduction du bloc MLP est de minimiser la norme de l'erreur **e** entre les positions désirées $\mathbf{q}_d$ et les positions actuelles $\mathbf{q}_s$. Minimiser la norme de **e** revient à minimiser le module du vecteur $\mathbf{V}_h = \mathbf{F} - \mathbf{\Phi}$.

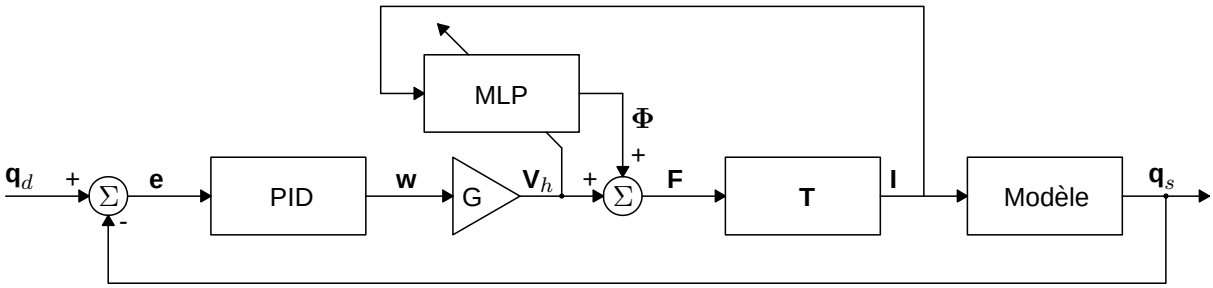

FIGURE *4.6: Schéma bloc en boucle fermée avec MLP [Ach08].*

Démonstration. [Ach08]

$$\begin{aligned} \mathbf{F} &= \mathbf{V}_h + \mathbf{\Phi} \quad \text{et} \quad \mathbf{I} = \mathbf{T}\,\mathbf{F} = \mathbf{T}\,(\mathbf{V}_h + \mathbf{\Phi}) \Rightarrow \mathbf{T}\,\mathbf{V}_h = \mathbf{I} - \mathbf{T}\,\mathbf{\Phi} \\ \text{or} \quad \mathbf{V}_h &= G\,\mathbf{w} = G(k_p\mathbf{e} + k_d\frac{d\mathbf{e}}{dt} + k_i\int \mathbf{e}dt) \\ &\Rightarrow G(k_p\mathbf{e} + k_d\frac{d\mathbf{e}}{dt} + k_i\int \mathbf{e}dt) = \mathbf{T}^{-1}\,\mathbf{I} - \mathbf{T}\,\mathbf{T}^{-1}\,\mathbf{\Phi} \\ &\Rightarrow G(k_p\mathbf{e} + k_d\frac{d\mathbf{e}}{dt} + k_i\int \mathbf{e}dt) = \mathbf{F} - \mathbf{\Phi} \end{aligned}$$

□

Le problème d'optimisation a donc pour but de minimiser la fonction objective $g = \frac{1}{2}\|\mathbf{V}_h\|^2$ et les poids des couches MLP sont ajustés en fonction du gradient instantané de g. À noter que les paramètres et les configurations utilisés pour les simulations sont les suivants [Ach08, ANDMC06] :

– Le nombre de neurones dans la couche cachée est choisi égal à cinq.

– Le facteur d'oubli est choisi égal à 0.1.

– Le taux d'apprentissage est choisi égal à 0.001.

– La fonction d'activation choisie est une sigmoïde à valeurs négatives.

– Les poids synaptiques sont choisis initialement de faibles valeurs et uniformement distribuées.

– Le nombre de répétitions des cycles d'apprentissage (époques) est égale à 15.

– À chaque instant n on a deux passages : direct et rétrograde et ceci pour améliorer l'apprentissage.

Les résultats de simulation obtenus en adoptant cette approche sont utilisés pour faire une comparaison avec ceux obtenus par la méthode proposée en utilisant les algorithmes adaptatifs à noyau.

Les méthodes d'identification par réseaux neurones ont leurs inconvénients comme par exemple la complexité du modèle qui augmente avec le nombre d'entrées et de sorties et avec le nombre ne neurones dans les couches cachées. De plus ces méthodes nécessitent des phases d'apprentissage

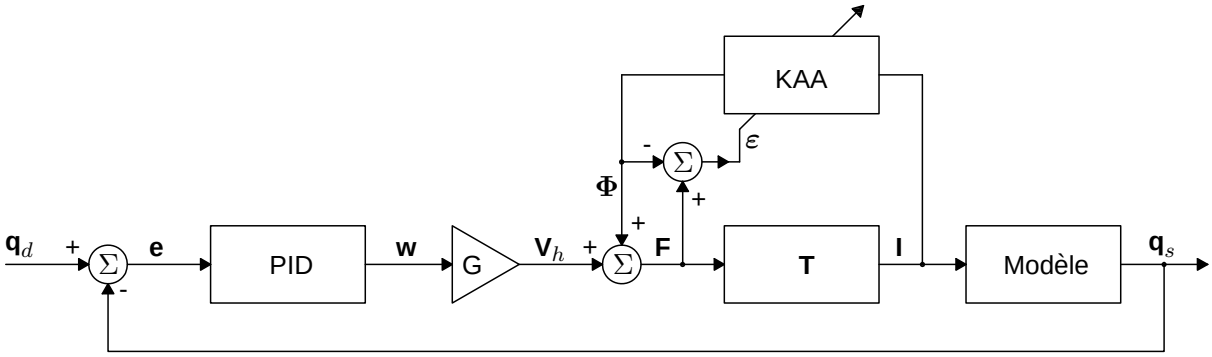

FIGURE *4.7: Schéma bloc en boucle fermée avec Algorithme Adaptatif à Noyau (KAA).*

préliminaires qui induisent des inconvénients dans l'identification en-ligne des modèles non-stationnaires. Ceci s'ajoute à leur inconvénient classique qu'ils risquent d'être piégés dans des minima locaux. De ce qui précède, surgit la nécessité d'essayer d'autres méthodes d'identification telles que les méthodes à noyau pour estimer des modèles MIMO fortement non-linéaires et non-stationnaires comme le palier magnétique actif.

Le schéma bloc proposé pour la substitution du bloc MLP par un algorithme adaptatif à noyau (Kernel Adaptive Algorithm - KAA) est exposé dans la figure (4.7). Les algorithmes adaptatifs à entrées et sorties multiples et utilisant les méthodes à noyau sont explorés dans le chapitre 2. Le bloc KAA proposé a pour entrée, comme dans le cas du bloc MLP, le vecteur des dix courants, et a pour sortie un vecteur $\mathbf{\Phi}$ de taille (5×1). Le but reste toujours la minimisation de la norme de l'erreur instantanée **e** résultant de la différence entre les positions actuelles et celles désirées. Comme c'est déjà démontré dans le cas MLP, la minimisation de la norme de **e** revient à minimiser la norme de vecteur $\mathbf{F} - \mathbf{\Phi}$. La fonction objective pour l'algorithme adaptatif à noyau est donc la minimisation de $\|\boldsymbol{\varepsilon}\|^2 = \|\mathbf{F} - \mathbf{\Phi}\|^2$ qui mène certainement à la minimisation de $\|\mathbf{e}\|^2$.

4.3.1 Algorithme MOKNLMS

4.3.1.1 Sans adaptation

Une première simulation est faite en utilisant l'algorithme MOKNLMS (MOKAPA avec $p = 1$). La prédiction est faite instantanément et en temps réel sans aucune phase d'apprentissage hors-ligne. La fonction noyau utilisée est le noyau Gaussien de bande passante 4.18 et la cohérence utilisée est $\mu_0 = 0.3$ avec $\varepsilon = 0.09$. Les paramètres k_p, k_d et k_i pour les cinq axes (respectivement x, y_1, z_1, y_2, z_2) sont adoptées comme dans [Ach08] tels que $k_p = (1000, 9400, 9400, 9400, 9400)$, $k_i = (400, 120000, 40000, 120000, 40000)$ et $k_d = (50, 600, 500, 600, 500)$. En fixant la période d'échantillonnage à $0.5ms$ avec approximativement 2000 échantillons, les simulations conduites donnent une taille finale de dictionnaire de $m = 3$. Les courbes de positions montrées dans la figure (4.8) décrivent comment les positions désirées sont atteintes à partir des positions initiales $(2\ 10^{-4}m)$. Le

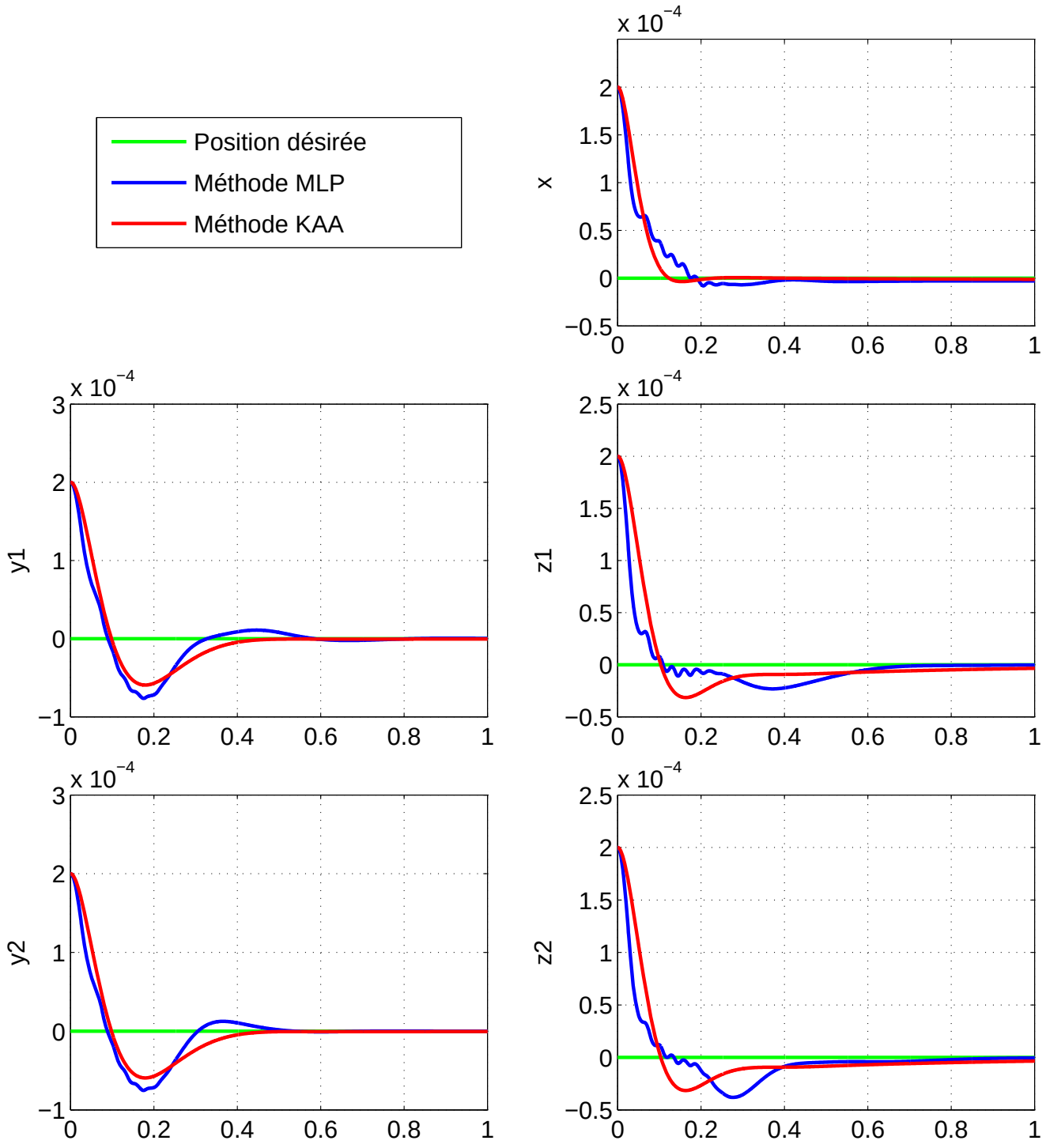

FIGURE *4.8: Courbes des positions des axes - comparaison entre MLP et KAA (MOKNLMS)*

TABLE *4.2: Tableau comparatif entre MLP et KAA (MOKNLMS).*

	Position x			Position y_1			Position z_1		
	MLP	KAA	Gain (%)	MLP	KAA	Gain (%)	MLP	KAA	Gain (%)
Temps de réponse	0.1266	0.0754	67.90	0.0654	0.0679	-3.68	0.0656	0.0719	-15.24
Temps de stabilisation	0.2381	0.1152	106.68	0.5355	0.4031	32.85	0.6094	0.5582	9.17

	Position y_2			Position z_2		
	MLP	KAA	Gain (%)	MLP	KAA	Gain (%)
Temps de réponse	0.0646	0.0679	-4.86	0.0668	0.0719	-7.09
Temps de stabilisation	0.4674	0.4031	15.95	0.5045	0.5582	-9.62

tableau (4.2) donne une comparaison entre les temps de réponse et de stabilisation pour les courbes de réponse dans les deux cas MLP et KAA (algorithme MOKNLMS).

4.3.1.2 Adaptation des éléments du dictionnaire

Dans cette section une adaptation des éléments du dictionnaire est faite en utilisant les mêmes paramètres pour le noyau Gaussien. La cohérence et le pas de gradient sont choisis de telle façon à obtenir la même taille du dictionnaire après adaptation ($m = 3$). Ces paramètres sont $\mu_0 = 0.05$ et $\nu_0 = 0.01$. La figure (4.9) montre les courbes de positions obtenues après adaptation comparées à celles obtenues sans adaptation. Le tableau exposé dans la figure (4.10) donne une comparaison entre les différentes caractéristiques des courbes de réponses dans le cas de l'utilisation d'un réseau de neurones (MLP) et dans le cas de l'utilisation de l'algorithme adaptatif MOKNLMS avec et sans adaptation des éléments du dictionnaire. Un autre schéma peut révéler l'éfficacité de l'adaptation c'est celui exposé dans la figure (4.11) qui compare la norme quadratique de l'erreur instantanée avant et après adaptation. Rappelons que l'erreur instantanée est la différence entre les positions désirées et les positions obtenues en appliquant l'algorithme d'apprentissage.

4.3.2 Algorithme MOKRLS

L'idée de base concernant l'utilisation d'un algorithme adaptatif en-ligne à la place du réseau de neurones (MLP) prouve sa éfficacité. Dans cette section et pour vérifier davantage les résultats des simulations, un autre algorithme adaptatif est utilisé. Il s'agit de l'algorithme de moindres carrés récursif à noyau à plusieurs sorties (MOKRLS). Un noyau Gaussien est utilisé avec une bande passante $\sigma = 0.6$ et les paramètres k_p, k_d et k_i pour les cinq axes (respectivement x, y_1, z_1, y_2, z_2) sont adoptées de la façon suivante $k_p = (1000, 9400, 9400, 9400, 9400)$, $k_i = (400, 120000, 40000, 120000, 40000)$ et

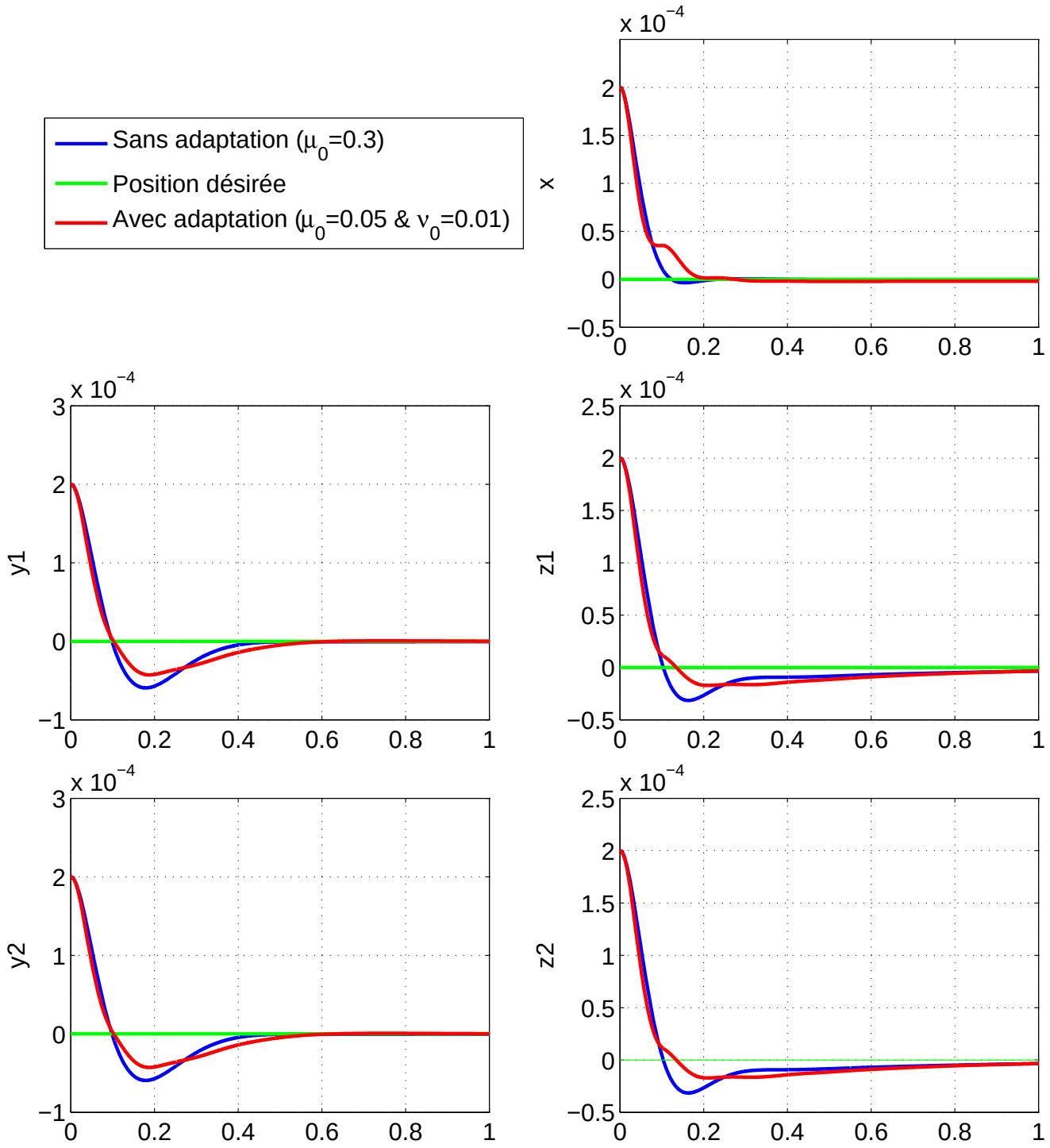

FIGURE *4.9: Courbes des positions des axes - comparaison avec et sans adaptation - MOKNLMS.*

MLP	Sans adaptation (m=3)	Avec adaptation (m=3)
position x: ---------- RiseTime: 0.1266 SettlingTime: 0.2381 SettlingMin: -8.0578e-006 SettlingMax: 1.7276e-005 Overshoot: 170.8096 Undershoot: 6.7217e+003 ---------- position y1: ---------- RiseTime: 0.0654 SettlingTime: 0.5355 SettlingMin: -7.6446e-005 SettlingMax: 1.9749e-005 Overshoot: 8.0443e+004 Undershoot: 3.0786e+004 ---------- position z1: ---------- RiseTime: 0.0656 SettlingTime: 0.6094 SettlingMin: -2.3216e-005 SettlingMax: 1.8904e-005 Overshoot: 8.0570e+003 Undershoot: 7.0271e+004 ---------- position y2: ---------- RiseTime: 0.0646 SettlingTime: 0.4674 SettlingMin: -7.5289e-005 SettlingMax: 1.9461e-005 Overshoot: 8.3789e+004 Undershoot: 2.2284e+005 ---------- position z2: ---------- RiseTime: 0.0668 SettlingTime: 0.5045 SettlingMin: -3.8096e-005 SettlingMax: 1.9537e-005 Overshoot: 7.9854e+003 Undershoot: 4.2448e+004 ----------	position x: ---------- RiseTime: 0.0754 SettlingTime: 0.1152 SettlingMin: -3.6153e-006 SettlingMax: 1.8680e-005 Overshoot: 190.7975 Undershoot: 1.6087e+004 ---------- position y1: ---------- RiseTime: 0.0679 SettlingTime: 0.4031 SettlingMin: -5.9403e-005 SettlingMax: 1.8905e-005 Overshoot: 1.5460e+004 Undershoot: 5.2387e+004 ---------- position z1: ---------- RiseTime: 0.0719 SettlingTime: 0.5582 SettlingMin: -3.1532e-005 SettlingMax: 1.6487e-005 Overshoot: 805.6276 Undershoot: 5.7443e+003 ---------- position y2: ---------- RiseTime: 0.0679 SettlingTime: 0.4031 SettlingMin: -5.9415e-005 SettlingMax: 1.8899e-005 Overshoot: 1.5452e+004 Undershoot: 5.2351e+004 ---------- position z2: ---------- RiseTime: 0.0719 SettlingTime: 0.5582 SettlingMin: -3.1546e-005 SettlingMax: 1.6478e-005 Overshoot: 805.9706 Undershoot: 5.7437e+003 ----------	position x: ---------- RiseTime: 0.1289 SettlingTime: 0.1901 SettlingMin: -2.1483e-006 SettlingMax: 1.8078e-005 Overshoot: 8.9939 Undershoot: 1.0147e+004 ---------- position y1: ---------- RiseTime: 0.0657 SettlingTime: 0.5161 SettlingMin: -4.2845e-005 SettlingMax: 1.9968e-005 Overshoot: 1.5001e+005 Undershoot: 3.2158e+004 ---------- position z1: ---------- RiseTime: 0.0725 SettlingTime: 0.6798 SettlingMin: -1.7190e-005 SettlingMax: 1.6761e-005 Overshoot: 405.2484 Undershoot: 5.8785e+003 ---------- position y2: ---------- RiseTime: 0.0657 SettlingTime: 0.5161 SettlingMin: -4.2854e-005 SettlingMax: 1.9957e-005 Overshoot: 1.5000e+005 Undershoot: 3.2161e+004 ---------- position z2: ---------- RiseTime: 0.0725 SettlingTime: 0.6798 SettlingMin: -1.7196e-005 SettlingMax: 1.6749e-005 Overshoot: 405.3906 Undershoot: 5.8780e+003 ----------

FIGURE *4.10: Tableau comparatif entre MLP - KAA avec et sans adaptation*

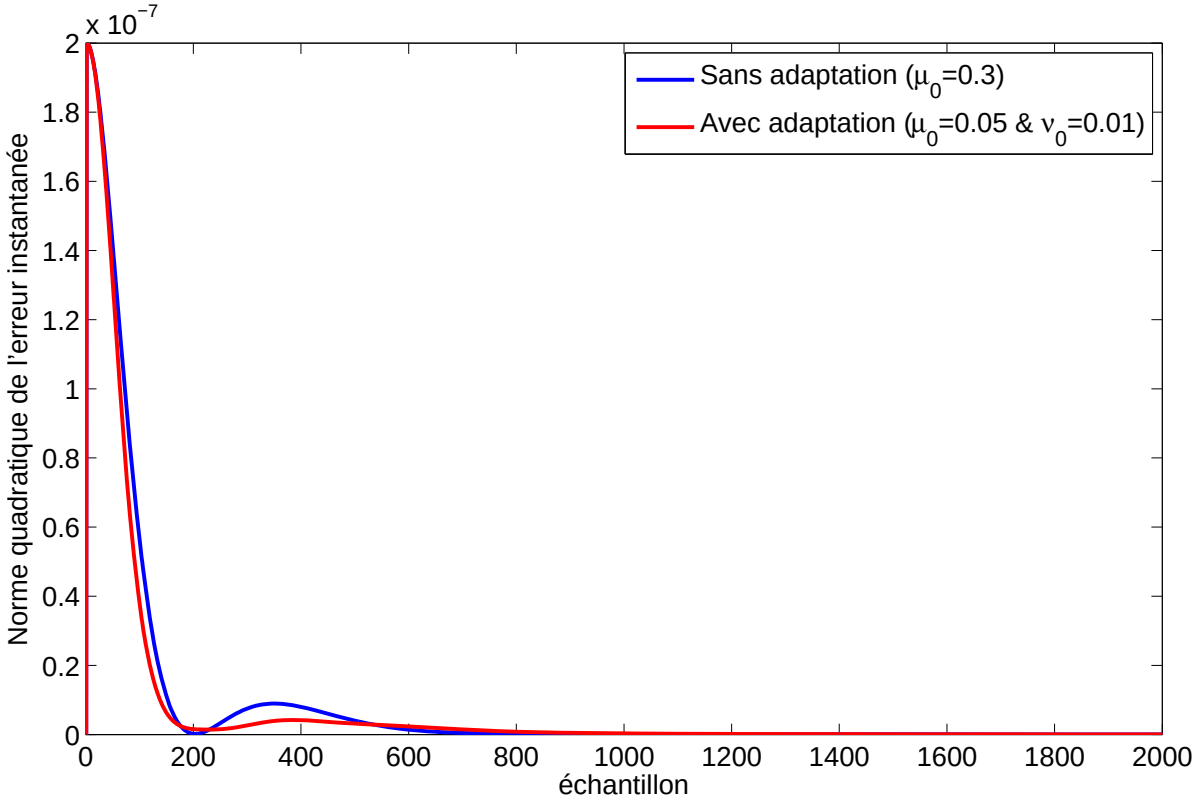

FIGURE *4.11: Courbes de la norme quadratique de l'erreur instantanée - comparaison avec et sans adaptation - MOKNLMS.*

$k_d = (100, 1200, 1000, 1200, 1000)$. La même période d'échantillonnage est considérée, $0.5ms$ avec approximativement 4000 échantillons. À signaler encore que les essais sont faits en négligeant le coefficient de régularisation ($\zeta = 0$) et en prenant le paramètre d'oubli $\theta = 1$.

En fixant la cohérence μ_0 à 0.6 et sans adaptation des éléments du dictionnaire, la taille de ce dernier est $m = 3$.

Pour adapter le dictionnaire, une sélection soignée conduit à fixer la cohérence μ_0 à 0.05 et le pas de gradient ν_0 à 0.01 et ceci pour obtenir la même taille du dictionnaire ($m = 3$).

La figure (4.12) donne les courbes des positions des axes, avec et sans adaptation du dictionnaire et le tableau (4.13) permet la comparaison des différentes caractéristiques de ces courbes de positions. Les courbes de la norme quadratique de l'erreur instantanée sont montrées dans la figure (4.14).

4.4 perspectives

Les résultats obtenus ne sont pas optimaux car il y a plusieurs paramètres à régler. Ces paramètres concernent soit l'algorithme d'apprentissage (cohérence, coefficients de régularisation, pas de gradient, ...) soit le système (gains du PID, fréquence d'échantillonnage,...). Donc, ce n'est pas facile de trouver la

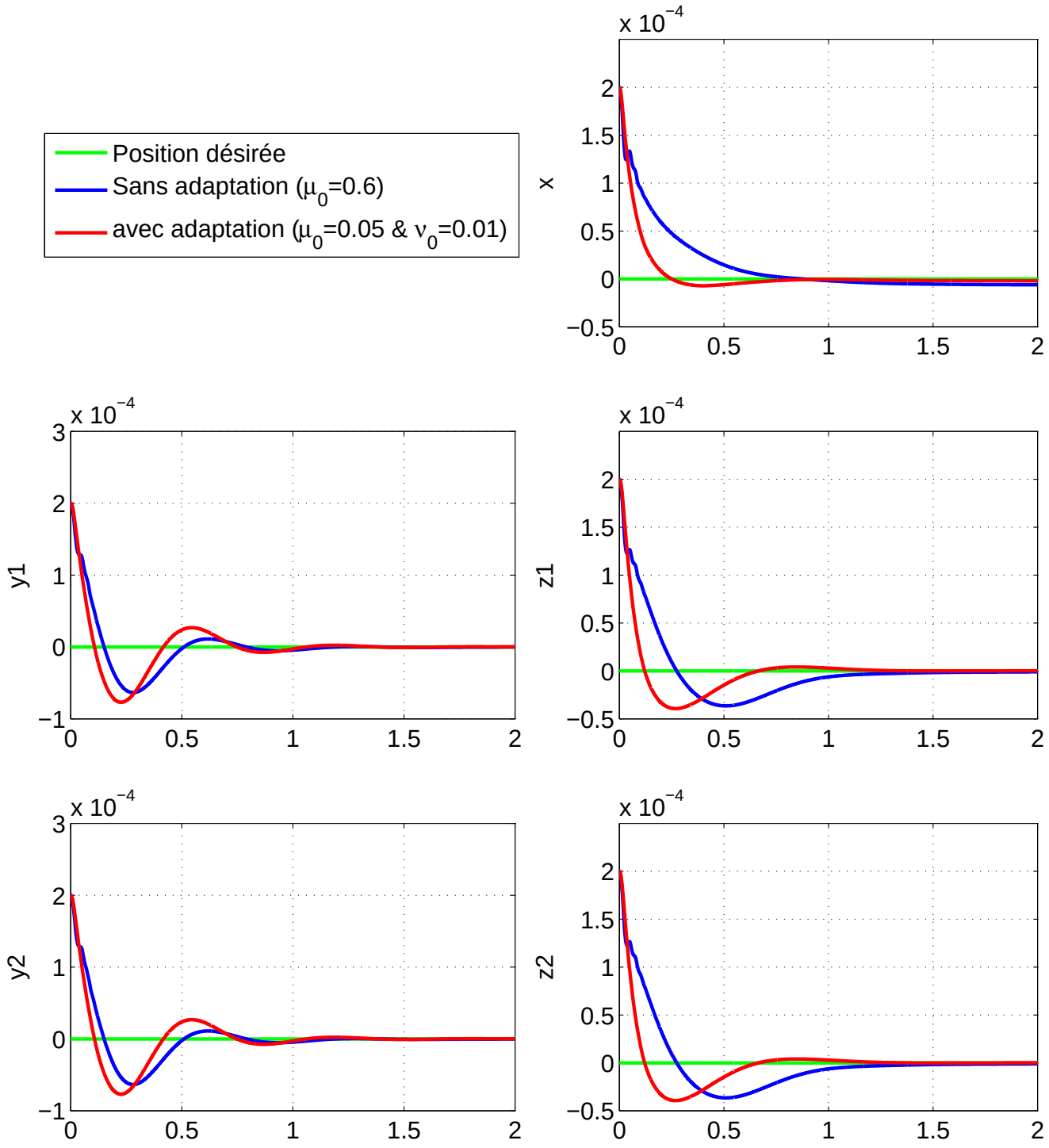

FIGURE *4.12: Courbes des positions des axes - comparaison avec et sans adaptation-MOKRLS.*

Sans adaptation (m=3)	Avec adaptation (m=3)
position x: ---------- RiseTime: 0.4898 SettlingTime: 1.0080 SettlingMin: -6.1300e-006 SettlingMax: 1.4478e-005 Overshoot: 0 Undershoot: 3.2626e+003 ---------- position y1: ---------- RiseTime: 0.1182 SettlingTime: 1.0128 SettlingMin: -6.3519e-005 SettlingMax: 1.9516e-005 Overshoot: 3.1626e+004 Undershoot: 9.9895e+004 ---------- position z1: ---------- RiseTime: 0.2145 SettlingTime: 1.0534 SettlingMin: -3.6519e-005 SettlingMax: 1.8970e-005 Overshoot: 3.4983e+003 Undershoot: 1.9707e+004 ---------- position y2: ---------- RiseTime: 0.1183 SettlingTime: 1.0128 SettlingMin: -6.3517e-005 SettlingMax: 1.9554e-005 Overshoot: 3.1577e+004 Undershoot: 9.9742e+004 ---------- position z2: ---------- RiseTime: 0.2146 SettlingTime: 1.0536 SettlingMin: -3.6532e-005 SettlingMax: 1.9011e-005 Overshoot: 3.4960e+003 Undershoot: 1.9687e+004 ----------	position x: ---------- RiseTime: 0.1467 SettlingTime: 0.4992 SettlingMin: -7.1847e-006 SettlingMax: 1.8033e-005 Overshoot: 257.8204 Undershoot: 9.9606e+003 ---------- position y1: ---------- RiseTime: 0.0776 SettlingTime: 0.9786 SettlingMin: -7.7001e-005 SettlingMax: 2.6757e-005 Overshoot: 2.7110e+006 Undershoot: 1.0438e+006 ---------- position z1: ---------- RiseTime: 0.0834 SettlingTime: 0.8755 SettlingMin: -3.9382e-005 SettlingMax: 1.9621e-005 Overshoot: 2.1268e+006 Undershoot: 4.1881e+005 ---------- position y2: ---------- RiseTime: 0.0776 SettlingTime: 0.9786 SettlingMin: -7.6996e-005 SettlingMax: 2.6757e-005 Overshoot: 2.6777e+006 Undershoot: 1.0309e+006 ---------- position z2: ---------- RiseTime: 0.0834 SettlingTime: 0.8757 SettlingMin: -3.9383e-005 SettlingMax: 1.9589e-005 Overshoot: 2.0354e+006 Undershoot: 4.0082e+005 ----------

FIGURE *4.13: Tableau comparatif pour MOKRLS avec et sans adaptation*

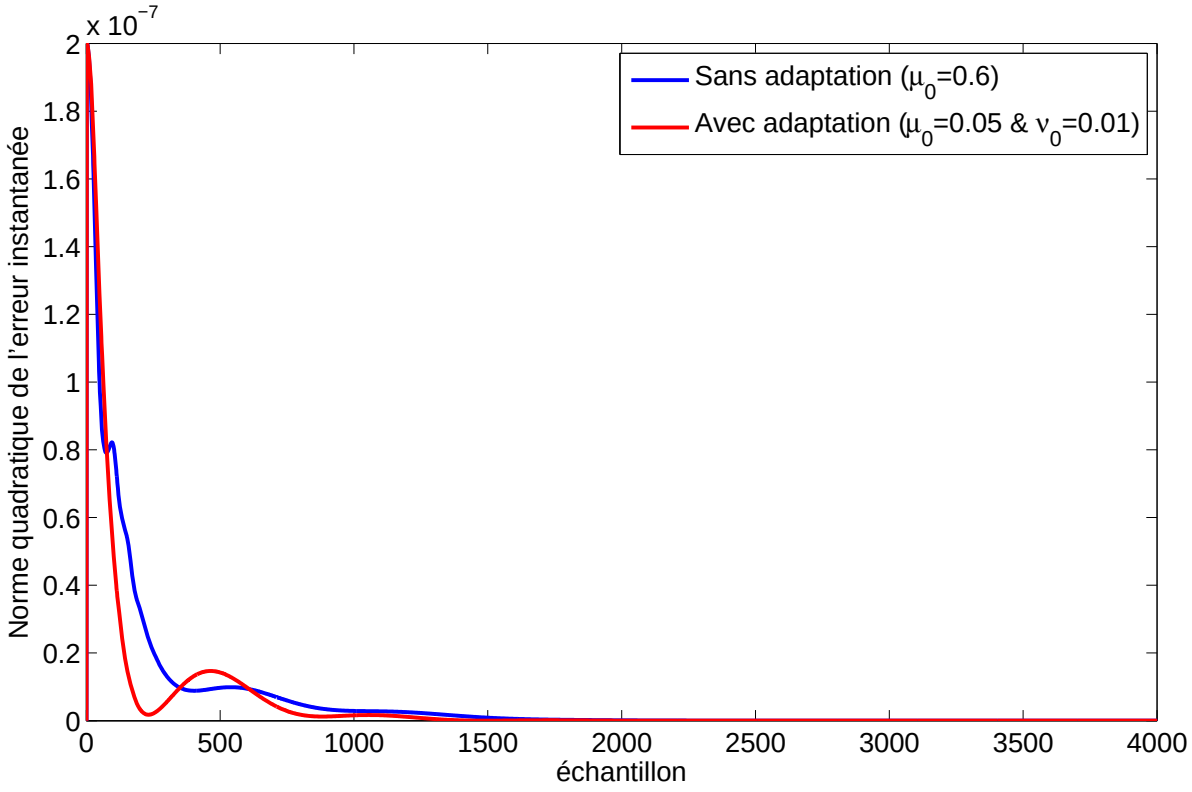

FIGURE *4.14: Courbes de la norme quadratique de l'erreur instantanée - comparaison avec et sans adaptation-MOKRLS.*

bonne combinaison de ces paramètres qui donne des résultats optimaux. Une perspective pour l'extension de ce travail sera peut être de trouver les relations qui lient les différents paramètres pour optimiser les résultats.

Conclusion générale et perspectives

Le contenu de ce mémoire de thêse tombe dans le cadre de l'identification en ligne des systèmes non linéaires par le moyen des algorithmes adaptatifs utilisant les méthodes à noyau. Nous avons exploré une technique d'amélioration du dictionnaire, obtenue par l'application du critère de cohérence, qui mène à l'estimation de la sortie d'un modèle. La technique de l'adaptation des éléments du dictionnaire est basée sur le gradient stochastique de l'erreur quadratique instantanée tout en respectant le critère de cohérence. Les heuristiques proposées, qui sont basées sur cette technique, dépendent du type de la fonction noyau adoptée. L'éfficacité de ces heuristiques était prouvée à travers des simulations diversifiées. En changeant le seuil de cohérence, les paramètres de l'algorithme d'adaptation et ceux du noyau et en choisissant le pas du gradient convenable, l'objectif d'améliorer le dictionnaire est atteint suivant la stratégie suivante :

– En fixant l'erreur quadratique à une certaine valeur acceptable, la taille du dictionnaire, et donc l'ordre du modèle, peut être réduite.

– À taille du dictionnaire comparable, l'erreur quadratique peut être réduite d'une façon considérable.

Un autre sujet traîté dans ce manuscrit qest l'adaptation des algorithmes adaptatifs utilisant le critère de cohérence pour l'identification des systèmes à sorties multiples. Les mêmes heuristiques utilisées pour l'adaptation des éléments du dictionnaire ont été appliquées mais en utilisant le gradient de la norme du vecteur des erreurs.

Comme application, on a choisi le palier magnétique actif qui a été décrit ainsi que les moyens pour le contrôler. Une méthode de contrôle basée sur les algorithmes adaptatifs à noyau a été proposée et des simulationspour prouver l'éfficacité de cette méthode.

Perspectives

Une des perspectives qui peuvent être envisagées c'est d'essayer de trouver une relation liant les différents paramètres de l'algorithme spécifiquement une relation qui lie le seuil de cohérence choisi avec le pas du gradient qu'il faut adopter. Une des extensions possibles de ce travail, sera de réduire la complexité des calculs qui est fortement liée au nombre d'éléments du dictionnaire - variant dans le temps - et de réfléchir à l'application d'autres critères de sparsification mesurant la dépendance linéaire.

Annexes

ANNEXE A

Algorithme de moindres carrés récursif à noyau (KRLS)

L'algorithme KRLS (Kernel Recursive Least squares) appartient à la catégorie des algorithmes des moindres carrés récursifs. Le problème d'optimisation à traiter est de la forme :

$$\min_{\psi\in\mathcal{H}} \sum_{i=1}^{n} (d_i - \psi(\mathbf{u}_i))^2 + \zeta\|\psi\|_{\mathcal{H}}^2 \tag{A.1}$$

où ζ est un paramètre positif de régularisation, d_i la réponse désirée du modèle à l'instant i et $\psi(\mathbf{u}_i)$ désigne la sortie correspondante du modèle à l'observation $\mathbf{u}_i$. Pour une résolution récursive du problème et en introduisant un facteur d'oubli $\theta \in]0, 1]$, la reformulation du problème sera de la forme :

$$\min_{\psi\in\mathcal{H}} \sum_{i=1}^{n} \theta^{n-i}(d_i - \psi(\mathbf{u}_i))^2 + \zeta\theta^n\|\psi\|_{\mathcal{H}}^2 \tag{A.2}$$

Le facteur d'oubli permet la négligence des anciennes données tout en accordant une importance aux données les plus récentes, ce qui permet une convenable évolution du modèle dans le cas de l'identification des systèmes non-stationnaires. En se basant sur le théorème de représentation, la solution optimale du problème à l'instant n s'écrit :

$$\psi_n(\cdot) = \sum_{j=1}^{n} \alpha_j \kappa(\mathbf{u}_j, \cdot) \tag{A.3}$$

Les $\{\alpha_j\}_{j=1,\cdots,n} \in \mathbb{R}$ représentent les coefficients du modèle. En combinant (A.3) et (A.2), on obtient le problème dual :

$$\min_{\boldsymbol{\alpha}} \big(\mathbf{d} - \mathbf{K}\boldsymbol{\alpha}\big)^t \boldsymbol{\Theta} \big(\mathbf{d} - \mathbf{K}\boldsymbol{\alpha}\big) + \zeta\theta^n \boldsymbol{\alpha}^t \mathbf{K}\boldsymbol{\alpha} \tag{A.4}$$

avec $\boldsymbol{\Theta}$ une matrice carrée diagonale dont le $(i,i)^{\text{ème}}$ élément est θ^{n-i} et $\mathbf{K}$ est la matrice de Gram de toutes les observations a priori. Elle est de taille $n \times n$ et dont le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_i, \mathbf{u}_j)$. La taille de la matrice $\mathbf{K}$ croît avec n ce qui rend la solution du problème impossible dans le cas de l'identification en-ligne des modèles. Pour surmonter ce problème, on doit appliquer un critère de parcimonie pour réduire l'ordre du modèle en considérant seulement les observations les plus pertinentes afin de construire le dictionnaire. Le critère de cohérence avec un seuil μ_0 est le plus favorable, vu sa simplicité et son faible

coût calculatoire par rapport aux autres critères.

En supposant que $\mathcal{D}_n = \{\kappa(\mathbf{u}_{w_1}, \cdot), \cdots, \kappa(\mathbf{u}_{w_m}, \cdot)\}$ est le dictionnaire de taille m à l'instant n, Le modèle estimé sera :

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_{n,j} \kappa(\mathbf{u}_{w_j}, \cdot) \tag{A.5}$$

$\boldsymbol{\alpha}_n = (\alpha_{n,1}, \cdots, \alpha_{n,m})^t$ est le vecteur des coefficients du modèle à l'instant n. En introduisant (A.5) dans (A.2), et conformément à (A.4), le vecteur $\boldsymbol{\alpha}_n$ sera obtenu en résolvant le problème d'optimisation ci-dessous :

$$\boldsymbol{\alpha}_n = \arg\min_{\boldsymbol{\alpha}} \big(\mathbf{d}_n - \mathbf{H}_n\boldsymbol{\alpha}\big)^t \boldsymbol{\Theta}_n \big(\mathbf{d}_n - \mathbf{H}_n\boldsymbol{\alpha}\big) + \zeta\theta^n \boldsymbol{\alpha}^t \mathbf{K}_n \boldsymbol{\alpha} \tag{A.6}$$

avec $\mathbf{K}_n$ la matrice de Gram des éléments du dictionnaire de taille $m \times m$ dont le $(i,j)^{\text{ème}}$ terme est $\kappa(\mathbf{u}_{w_i}, \mathbf{u}_{w_j})$, $\boldsymbol{\Theta}_n$ une matrice diagonale carrée de taille $n \times n$ dont le $(i,i)^{\text{ème}}$ élément est θ^{n-i}, $\mathbf{H}_n$ une matrice de taille $n \times m$ dont le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_i, \mathbf{u}_{w_j})$ et $\mathbf{d}_n$ est un vecteur de taille $n \times 1$ qui représente les réponses désirées jusqu'à l'instant n, $\mathbf{d}_n = (d_1, \cdots, d_n)^t$. La solution de (A.6) en $\boldsymbol{\alpha}_n$ mène à :

$$\begin{aligned} \mathbf{H}_n^t \boldsymbol{\Theta}_n \big(\mathbf{d}_n - \mathbf{H}_n\boldsymbol{\alpha}_n\big) &= \zeta\theta^n \mathbf{K}_n \boldsymbol{\alpha}_n \\ \Rightarrow \big(\mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{H}_n + \zeta\theta^n \mathbf{K}_n\big)\boldsymbol{\alpha}_n &= \mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{d}_n \end{aligned}$$

En posant $\mathbf{P}_n = \big(\mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{H}_n + \zeta\theta^n \mathbf{K}_n\big)^{-1}$ et supposant que $\mathbf{P}_n$ existe, la solution du problème devient :

$$\boldsymbol{\alpha}_n = \mathbf{P}_n \mathbf{H}_n^t \boldsymbol{\Theta}_n \mathbf{d}_n \tag{A.7}$$

L'agorithme KRLS est récursif ceci consiste donc à déduire la solution du problème d'optimisation à l'instant $n+1$ à partir de la solution trouvée à l'instant n. A l'instant $n+1$, et pour une nouvelle observation $\mathbf{u}_{n+1}$ à l'entrée du modèle deux cas peuvent se produire. Si la fonction noyau correspondante à la nouvelle entrée $\kappa(\mathbf{u}_{n+1}, \cdot)$ ne vérifie pas le critère de parcimonie, elle ne sera pas introduite au dictionnaire. Mais si elle vérifie le critère de parcimonie, on l'introduit dans le dictionnaire qui augmente de taille d'une unité. Dans les deux cas, le modèle est mis à jour convenablement :

– Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ ne vérifie pas le critère de parcimonie, le dictionnaire reste inchangé ($\mathcal{D}_{n+1} = \mathcal{D}_n$), parsuite $\mathbf{K}_n$ reste inchangée, mais la matrice $\boldsymbol{\Theta}_n$ augmente de taille pour devenir $\boldsymbol{\Theta}_{n+1}$ de taille $(n+1) \times (n+1)$ avec le $(i,i)^{\text{ème}}$ élément est θ^{n+1-i}. on doit ajoute un élément au vecteur $\mathbf{d}_{n+1}$ et une ligne à la matrice $\mathbf{H}_n$ comme suit :

$$\mathbf{H}_{n+1} = \begin{bmatrix} \mathbf{H}_n \\ \mathbf{h}_{n+1}^t \end{bmatrix} \qquad \boldsymbol{\Theta}_{n+1} = \begin{bmatrix} \theta\boldsymbol{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1 \end{bmatrix} \qquad \mathbf{d}_{n+1} = \begin{bmatrix} \mathbf{d}_n \\ d_{n+1} \end{bmatrix}$$

avec $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$. Dans ce cas, la solution du problème à l'instant

$n+1$ se fait par analogie à (A.7) :

$$\boldsymbol{\alpha}_{n+1} = \mathbf{P}_{n+1}\mathbf{H}_{n+1}^{t}\boldsymbol{\Theta}_{n+1}\mathbf{d}_{n+1} \tag{A.8}$$

En supposant que $\mathbf{P}_{n+1}$ ne soit pas singulière,

$$\begin{aligned}\mathbf{P}_{n+1} &= \left(\mathbf{H}_{n+1}^{t}\boldsymbol{\Theta}_{n+1}\mathbf{H}_{n+1} + \zeta\theta^{n+1}\mathbf{K}_{n+1}\right)^{-1} \\ &= \left(\theta\mathbf{P}_{n}^{-1} + \mathbf{h}_{n+1}\mathbf{h}_{n+1}^{t}\right)^{-1}\end{aligned} \tag{A.9}$$

L'expression précédente est évaluée sans inversion en utilisant l'identité de Woodbury connue sous le nom de lemme d'inversion matricielle :

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1}$$

en posant $\mathbf{B} = \mathbf{D} = \mathbf{I}$ avec $\mathbf{I}$ est une matrice identité, l'expression précédente devient :

$$(\mathbf{A}+\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}(\mathbf{C}^{-1}+\mathbf{A}^{-1})^{-1}\mathbf{A}^{-1} \tag{A.10}$$

En appliquant (A.10) à (A.9) on obtient :

$$\begin{aligned}\mathbf{P}_{n+1} &= \theta^{-1}\mathbf{P}_{n} - \theta^{-2}\mathbf{P}_{n}((\mathbf{h}_{n+1}\mathbf{h}_{n+1}^{t})^{-1} + \theta^{-1}\mathbf{P}_{n})^{-1}\mathbf{P}_{n} \\ &= \theta^{-1}\left[\mathbf{P}_{n} - \frac{\theta^{-1}\mathbf{P}_{n}\mathbf{h}_{n+1}\mathbf{h}_{n+1}^{t}\mathbf{P}_{n}}{1+\theta^{-1}\mathbf{h}_{n+1}^{t}\mathbf{P}_{n}\mathbf{h}_{n+1}}\right] \qquad \text{(A.11)} \\ &= \frac{\theta^{-1}\mathbf{P}_{n}}{1+\theta^{-1}\mathbf{h}_{n+1}^{t}\mathbf{P}_{n}\mathbf{h}_{n+1}} \qquad \text{(A.12)}\end{aligned}$$

Notons que :

$$\mathbf{H}_{n+1}^{t}\boldsymbol{\Theta}_{n+1}\mathbf{d}_{n+1} = \theta\mathbf{H}_{n}^{t}\boldsymbol{\Theta}_{n}\mathbf{d}_{n} + \mathbf{h}_{n+1}d_{n+1} \tag{A.13}$$

Les expressions (A.13) et (A.8) donnent :

$$\begin{aligned}\boldsymbol{\alpha}_{n+1} &= \mathbf{P}_{n+1}\left(\theta\mathbf{H}_{n}^{t}\boldsymbol{\Theta}_{n}\mathbf{d}_{n} + \mathbf{h}_{n+1}d_{n+1}\right) \\ &= \left[\mathbf{P}_{n} - \frac{\theta^{-1}\mathbf{P}_{n}\mathbf{h}_{n+1}\mathbf{h}_{n+1}^{t}\mathbf{P}_{n}}{1+\theta^{-1}\mathbf{h}_{n+1}^{t}\mathbf{P}_{n}\mathbf{h}_{n+1}}\right]\mathbf{H}_{n}^{t}\boldsymbol{\Theta}_{n}\mathbf{d}_{n} + \mathbf{P}_{n+1}\mathbf{h}_{n+1}d_{n+1}\end{aligned} \tag{A.14}$$

En se reférant à (A.7) et (A.17), l'expression précédente devient :

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_{n} + \mathbf{P}_{n+1}\mathbf{h}_{n+1}(d_{n+1} - \mathbf{h}_{n+1}^{t}\boldsymbol{\alpha}_{n}) \tag{A.15}$$

Dans la mise à jour de $\boldsymbol{\alpha}_{n+1}$ on retrouve l'erreur a priori $e_{n+1} = d_{n+1} - \mathbf{h}_{n+1}^{t}\boldsymbol{\alpha}_{n}$ car $\psi_{n+1}(\mathbf{u}_{n+1}) = \mathbf{h}_{n+1}^{t}\boldsymbol{\alpha}_{n}$ n'est autre que la réponse du modèle à l'instant $n+1$.

– Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ vérifie le critère de parcimonie, donc elle doit être introduite dans le dictionnaire qui augmente de taille d'une unité pour devenir $\mathcal{D}_{n+1} = \{\kappa(\mathbf{u}_{w_1}, \cdot), \cdots, \kappa(\mathbf{u}_{w_{m+1}}, \cdot)\}$ avec $\kappa(\mathbf{u}_{w_{m+1}}, \cdot) = \kappa(\mathbf{u}_{n+1}, \cdot)$. L'incrémentation de l'ordre du dictionnaire mène à :

$$\mathbf{H}_{n+1} = \begin{bmatrix} \mathbf{H}_n & \mathbf{0}_n \\ \mathbf{h}_{n+1}^t & h_0 \end{bmatrix} \qquad \mathbf{K}_{n+1} = \begin{bmatrix} \mathbf{K}_n & \mathbf{h}_{n+1} \\ \mathbf{h}_{n+1}^t & h_0 \end{bmatrix} \qquad \mathbf{\Theta}_{n+1} = \begin{bmatrix} \theta\mathbf{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1 \end{bmatrix} \qquad \mathbf{d}_{n+1} = \begin{bmatrix} \mathbf{d}_n \\ d_{n+1} \end{bmatrix}$$

où $\mathbf{h}_{n+1} = (\kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n+1}, \mathbf{u}_{w_m}))^t$, $h_0 = \kappa(\mathbf{u}_{w_{m+1}}, \mathbf{u}_{w_{m+1}})$ et $\mathbf{0}_n$ est un vecteur colonne de n zéros. L'équation (A.8) reste toujours valide pour trouver les coeficients optimaux du modèle à l'instant $n+1$, mais $\mathbf{P}_{n+1}$ est devenue :

$$\begin{aligned} \mathbf{P}_{n+1} &= \left(\mathbf{H}_{n+1}^t \mathbf{\Theta}_{n+1} \mathbf{H}_{n+1} + \zeta\theta^{n+1}\mathbf{K}_{n+1}\right)^{-1} \\ &= \left[\begin{bmatrix} \mathbf{H}_n^t & \mathbf{h}_{n+1} \\ \mathbf{0}_n^t & h_0 \end{bmatrix} \begin{bmatrix} \theta\mathbf{\Theta}_n & \mathbf{0}_n \\ \mathbf{0}_n^t & 1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_n & \mathbf{0}_n \\ \mathbf{h}_{n+1}^t & h_0 \end{bmatrix} + \zeta\theta^{n+1} \begin{bmatrix} \mathbf{K}_n & \mathbf{h}_{n+1} \\ \mathbf{h}_{n+1}^t & h_0 \end{bmatrix} \right]^{-1} \\ &= \begin{bmatrix} \theta\mathbf{P}_n^{-1} + \mathbf{h}_{n+1}\mathbf{h}_{n+1}^t & (h_0 + \zeta\theta^{n+1})\mathbf{h}_{n+1} \\ (h_0 + \zeta\theta^{n+1})\mathbf{h}_{n+1}^t & (h_0 + \zeta\theta^{n+1})h_0 \end{bmatrix}^{-1} \end{aligned} \tag{A.16}$$

Pour calculer $\mathbf{P}_{n+1}$ on utilise l'identité ci-dessous pour assurer une inversion par bloc :

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{I} \end{bmatrix} \left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \begin{bmatrix} -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix}$$

En appliquant l'identité précédente à (A.11) on obtient :

$$\begin{aligned} \mathbf{P}_{n+1} &= \begin{bmatrix} \tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m \\ \mathbf{0}_m^t & 0 \end{bmatrix} + \frac{1}{s} \begin{bmatrix} -\mathbf{q} \\ 1 \end{bmatrix} \begin{bmatrix} -\mathbf{q}^t & 1 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m \\ \mathbf{0}_m^t & 0 \end{bmatrix} + \frac{1}{s} \begin{bmatrix} \mathbf{q}\mathbf{q}^t & -\mathbf{q} \\ -\mathbf{q}^t & 1 \end{bmatrix} \end{aligned} \tag{A.17}$$

où

$$\begin{aligned} \tilde{\mathbf{P}}_{n+1} &= (\theta\mathbf{P}_n^{-1} + \mathbf{h}_{n+1}\mathbf{h}_{n+1}^t)^{-1} \\ \mathbf{q} &= (h_0 + \zeta\theta^{n+1})\tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1} \\ s &= (h_0 + \zeta\theta^{n+1})(h_0 - \mathbf{h}_{n+1}^t\mathbf{q}) \end{aligned}$$

A souligner que $\tilde{\mathbf{P}}_{n+1}$ existe dans la solution $\tilde{\boldsymbol{\alpha}}_{n+1} = \tilde{\mathbf{P}}_{n+1}\tilde{\mathbf{H}}_{n+1}^t\mathbf{\Theta}_{n+1}\mathbf{d}_{n+1}$ du problème

$$\tilde{\boldsymbol{\alpha}}_{n+1} = \arg\min_{\boldsymbol{\alpha}} \left(\mathbf{d}_{n+1} - \tilde{\mathbf{H}}_{n+1}\boldsymbol{\alpha}\right)^t \mathbf{\Theta}_{n+1} \left(\mathbf{d}_{n+1} - \tilde{\mathbf{H}}_{n+1}\boldsymbol{\alpha}\right) + \zeta\theta^n \boldsymbol{\alpha}^t \mathbf{K}_{n+1}\boldsymbol{\alpha}$$

avec $\tilde{\mathbf{H}}_{n+1} = \begin{bmatrix}\mathbf{H}_n^t & \mathbf{h}_{n+1}\end{bmatrix}^t$. Donc dans une première phase, on détermine $\tilde{\mathbf{P}}_{n+1}$ et $\tilde{\boldsymbol{\alpha}}_{n+1}$ d'une façon similaire à celle appliquée dans le premier cas (où on n'a pas introduit un nouvel élément dans le dictionnaire).

$$\tilde{\boldsymbol{\alpha}}_{n+1} = \boldsymbol{\alpha}_n + \tilde{\mathbf{P}}_{n+1}\mathbf{h}_{n+1}(d_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n) \tag{A.18}$$

$$\tilde{\mathbf{P}}_{n+1} = \theta^{-1}\left[\mathbf{P}_n - \frac{\theta^{-1}\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1+\theta^{-1}\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}\right] \tag{A.19}$$

Dans une deuxième phase, la mise à jour des coefficients du problème optimal est obtenue en utilisant $\mathbf{P}_{n+1}$ trouvée dans (A.17) comme suit :

$$\begin{aligned}\boldsymbol{\alpha}_{n+1} &= \mathbf{P}_{n+1}\mathbf{H}_{n+1}^t\boldsymbol{\Theta}_{n+1}\mathbf{d}_{n+1}\\ &= \begin{bmatrix}\tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m\\ \mathbf{0}_m^t & 0\end{bmatrix}\begin{bmatrix}\mathbf{H}_n^t & \mathbf{h}_{n+1}\\ \mathbf{0}_n^t & h_0\end{bmatrix}\begin{bmatrix}\theta\boldsymbol{\Theta}_n & \mathbf{0}_n\\ \mathbf{0}_n^t & 1\end{bmatrix}\begin{bmatrix}\mathbf{d}_n\\ d_{n+1}\end{bmatrix}\\ &+ \frac{1}{s}\begin{bmatrix}\mathbf{q}\mathbf{q}^t & -\mathbf{q}\\ -\mathbf{q}^t & 1\end{bmatrix}\begin{bmatrix}\mathbf{H}_n^t & \mathbf{h}_{n+1}\\ \mathbf{0}_n^t & h_0\end{bmatrix}\begin{bmatrix}\theta\boldsymbol{\Theta}_n & \mathbf{0}_n\\ \mathbf{0}_n^t & 1\end{bmatrix}\begin{bmatrix}\mathbf{d}_n\\ d_{n+1}\end{bmatrix}\\ &= \boldsymbol{\alpha}_{n+1}^{(1)} + \boldsymbol{\alpha}_{n+1}^{(2)}\end{aligned} \tag{A.20}$$

Après développement

$$\boldsymbol{\alpha}_{n+1}^{(1)} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1}(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\mathbf{d}_n + \mathbf{h}_{n+1}d_{n+1})\\ 0\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{P}}_{n+1}\tilde{\mathbf{H}}_{n+1}^t\boldsymbol{\Theta}_{n+1}\mathbf{d}_{n+1}\\ 0\end{bmatrix} = \begin{bmatrix}\tilde{\boldsymbol{\alpha}}_{n+1}\\ 0\end{bmatrix} \tag{A.21}$$

$$\begin{aligned}\boldsymbol{\alpha}_{n+1}^{(2)} &= \frac{1}{s}\begin{bmatrix}\mathbf{q}\mathbf{q}^t(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\mathbf{d}_n + \mathbf{h}_{n+1}d_{n+1}) - h_0d_{n+1}\mathbf{q}\\ -\mathbf{q}^t(\theta\mathbf{H}_n^t\boldsymbol{\Theta}_n\mathbf{d}_n + \mathbf{h}_{n+1}d_{n+1}) + h_0d_{n+1}\end{bmatrix} = \frac{1}{s}\begin{bmatrix}-\mathbf{q}(h_0d_{n+1} - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{\alpha}}_{n+1})\\ h_0d_{n+1} - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{\alpha}}_{n+1}\end{bmatrix}\\ &= \frac{1}{s}(h_0d_{n+1} - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{\alpha}}_{n+1})\begin{bmatrix}-\mathbf{q}\\ 1\end{bmatrix}\end{aligned} \tag{A.22}$$

Les expressions (A.20), (A.21) et (A.22) conduisent à :

$$\boldsymbol{\alpha}_{n+1} = \begin{bmatrix}\tilde{\boldsymbol{\alpha}}_{n+1}\\ 0\end{bmatrix} + \frac{h_0d_{n+1} - (h_0+\zeta\theta^{n+1})\mathbf{h}_{n+1}^t\tilde{\boldsymbol{\alpha}}_{n+1}}{(h_0+\zeta\theta^{n+1})(h_0 - \mathbf{h}_{n+1}^t\mathbf{q})}\begin{bmatrix}-\mathbf{q}\\ 1\end{bmatrix} \tag{A.23}$$

Pour simplifier les expression trouvées ci-haut, on peut négliger le terme de régularisation dans la fonction coût initiale en assumant $\zeta = 0$. Les expressions (A.11) et (A.15) deviennent :

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \frac{\mathbf{P}_n\mathbf{h}_{n+1}}{1+\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}}(d_{n+1} - \mathbf{h}_{n+1}^t\boldsymbol{\alpha}_n) \tag{A.24}$$

$$\mathbf{P}_{n+1} = \mathbf{P}_n - \frac{\mathbf{P}_n\mathbf{h}_{n+1}\mathbf{h}_{n+1}^t\mathbf{P}_n}{1+\mathbf{h}_{n+1}^t\mathbf{P}_n\mathbf{h}_{n+1}} \tag{A.25}$$

et (A.17) et (A.23) deviennent :

$$\boldsymbol{\alpha}_{n+1} = \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_{n+1} \\ 0 \end{bmatrix} + \frac{d_{n+1} - \mathbf{h}_{n+1}^t \boldsymbol{\alpha}_n}{1 - \mathbf{h}_{n+1}^t \tilde{\mathbf{P}}_{n+1} \mathbf{h}_{n+1}} \begin{bmatrix} \tilde{\mathbf{P}}_{n+1} \mathbf{h}_{n+1} \\ 1/h_0 \end{bmatrix} \tag{A.26}$$

$$\mathbf{P}_{n+1} = \begin{bmatrix} \tilde{\mathbf{P}}_{n+1} & \mathbf{0}_m \\ \mathbf{0}_m^t & 0 \end{bmatrix} + \frac{1}{1 - \mathbf{h}_{n+1}^t \tilde{\mathbf{P}}_{n+1} \mathbf{h}_{n+1}} \times \begin{bmatrix} -\tilde{\mathbf{P}}_{n+1} \mathbf{h}_{n+1} \\ 1/h_0 \end{bmatrix} \begin{bmatrix} -(\tilde{\mathbf{P}}_{n+1} \mathbf{h}_{n+1})^t & 1/h_0 \end{bmatrix} \tag{A.27}$$

ANNEXE B

Algorithme de projection affine à noyau (KAPA)

La deuxième catégorie des algorithmes adaptatifs est celle à gradient stochastique. L'algorithme de projection affine à noyau (Kernel Affine Projection Algorithm - KAPA) fait partie de cette catégorie. L'objectif de l'agorithme est toujours la minimisation de l'erreur quadratique. La formulation du problème en négligeant le terme de régularisation est :

$$\min_{\psi \in \mathcal{H}} \sum_{i=1}^{n} (d_i - \psi(\mathbf{u}_i))^2 \tag{B.1}$$

d_i est la réponse désirée du modèle à l'instant i et $\psi(\mathbf{u}_i)$ est la réponse du modèle à la $i^{\text{ème}}$ observation $\mathbf{u}_i$. A l'instant n, si $\mathcal{D}_n = \{\kappa(\mathbf{u}_{w_j}, \cdot)\}_{j=1,\cdots,m}$ est un dictionnaire de taille m obtenu en appliquant un critère de parcimonie, le modèle à l'instant n sera :

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_{n,j} \kappa(\mathbf{u}_{w_j}, \cdot) \tag{B.2}$$

avec $\boldsymbol{\alpha}_n = (\alpha_{n,1}, \cdots, \alpha_{n,m})^t$ le vecteur des coefficients optimaux du modèle à l'instant n. La combinaison de (B.2) et (B.1) donne le problème dual suivant :

$$\boldsymbol{\alpha}_n = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{d}_n - \mathbf{H}_n \boldsymbol{\alpha}\|^2 \tag{B.3}$$

où $\mathbf{d}_n = (d_1, \cdots, d_n)^t$ le vecteur des réponses désirées jusqu'à l'instant n et $\mathbf{H}_n$ est une matrice $n \times m$ dont le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_i, \mathbf{u}_{w_j})$. En supposant que $(\mathbf{H}_n^t \mathbf{H}_n)^{-1}$ existe, la solution du problème (B.3) est :

$$\boldsymbol{\alpha}_n = (\mathbf{H}_n^t \mathbf{H}_n)^{-1} \mathbf{H}_n^t \mathbf{d}_n \tag{B.4}$$

La taille de la matrice $\mathbf{H}_n$ augmente avec n ce qui crée une charge calculatoire qui croît avec le temps. Le principe de l'algorithme est de prendre seulement en considération les p dernières observations ($p \leq n$) $\{\mathbf{u}_n, \cdots, \mathbf{u}_{n-p+1}\}$. En d'autres termes, il s'agit d'une fenêtre glissante de largeur p pour sélectionner les observations [Say03]. Le paramètre p est appelée nombre de collecteurs (manifolds). Le vecteur $\mathbf{d}_n$ est alors un vecteur colonne de taille $p \times 1$ tel que $\mathbf{d}_n = (d_n, \cdots, d_{n-p+1})^t$, et $\mathbf{H}_n$ une matrice $p \times m$ dont

le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_{n-i+1}, \mathbf{u}_{w_j})$ telle que :

$$\mathbf{H}_n = \begin{pmatrix} \kappa(\mathbf{u}_n, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_n, \mathbf{u}_{w_m}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{u}_{n-p+1}, \mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{n-p+1}, \mathbf{u}_{w_m}) \end{pmatrix}$$

La solution optimale du problème $\boldsymbol{\alpha}_{n+1}$ à l'instant $n+1$ est déterminée en utilisant la méthode à gradient stochastique. Pour cette raison, le principe de fluctuation minimale est appliqué en minimisant $\|\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n\|^2$. Des contraintes sur l'annulation de l'erreur a posteriori des p dernères observations, sont imposées à cette fonction objective. Ceci implique que le problème devient :

$$\min_{\boldsymbol{\alpha}_{n+1}} \|\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n\|^2 \tag{B.5}$$

$$\text{sous contrainte } \mathbf{d}_{n+1} = \mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1} \tag{B.6}$$

où $\mathbf{d}_{n+1} = (d_{n+1}, \cdots, d_{n-p+2})^t$, $\boldsymbol{\alpha}_{n+1} = (\alpha_{n+1,1}, \cdots, \alpha_{n+1,m})^t$ et $\mathbf{H}_{n+1}$ est toujours une matrice $p \times m$ avec le $(i,j)^{\text{ème}}$ élément est $\kappa(\mathbf{u}_{n-i+2}, \mathbf{u}_{w_j})$. Il s'agit d'une projection affine puisque, $\boldsymbol{\alpha}_{n+1}$ est obtenue par la projection de $\boldsymbol{\alpha}_n$ sur l'intersection des p sous-espaces affines $\mathcal{S}_i$ définis par :

$$\mathcal{S}_i = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \mathbf{h}_{n-i+2}^t \boldsymbol{\alpha} - d_{n-i+2} = 0\}_{i=1,\cdots,p} \tag{B.7}$$

avec $\mathbf{h}_{n-i+2} = (\kappa(\mathbf{u}_{n-i+2}, \mathbf{u}_{w_1}), \cdots, \kappa(\mathbf{u}_{n-i+2}, \mathbf{u}_{w_m}))^t$. L'ordre du modèle, qui n'est autre que la taille du dictionnaire m, est susceptible d'augmenter à chaque instant suivant l'introduction ou non d'un nouvel élément au dictionnaire. Lors de l'application du critère de parcimonie à l'instant $n+1$, deux cas peuvent se présenter :

– Si la fonction noyau $\kappa(\mathbf{u}_{n+1}, \cdot)$ correspondante à l'observation $\mathbf{u}_{n+1}$ ne vérifie pas le critère de parcimonie, alors elle ne sera pas introduite dans le dictionnaire. Le problème d'optimisation (B.5) sous contrainte (B.6) est obtenue en minimisant le Lagrangien :

$$\mathcal{L}(\boldsymbol{\alpha}_{n+1}, \boldsymbol{\lambda}) = \|\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n\|^2 + \boldsymbol{\lambda}^t(\mathbf{d}_{n+1} - \mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1}) \tag{B.8}$$

avec $\boldsymbol{\lambda}$ est le vecteur des multplicateurs de Lagrange. En dérivant $\mathcal{L}(\boldsymbol{\alpha}_{n+1}, \boldsymbol{\lambda})$ par rapport à $\boldsymbol{\alpha}_{n+1}$ et $\boldsymbol{\lambda}$ et annulant ces dérivées, on obtient :

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}_{n+1}, \boldsymbol{\lambda})}{\partial \boldsymbol{\alpha}_{n+1}} = 2(\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n) - \mathbf{H}_{n+1}^t \boldsymbol{\lambda} = \mathbf{0}_m \quad \Rightarrow \quad 2(\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n) = \mathbf{H}_{n+1}^t \boldsymbol{\lambda} \tag{B.9}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha}_{n+1}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = \mathbf{d}_{n+1} - \mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1} = \mathbf{0}_m \quad \Rightarrow \quad \mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1} = \mathbf{d}_{n+1} \tag{B.10}$$

l'équation (B.9) donne

$$2\mathbf{H}_{n+1}(\boldsymbol{\alpha}_{n+1}-\boldsymbol{\alpha}_n)=\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t\boldsymbol{\lambda}$$
$$2(\mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1}-\mathbf{H}_{n+1}\boldsymbol{\alpha}_n)=\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t\boldsymbol{\lambda}$$

de ce qui précède et en supposant que $\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t$ est non singulière on obtient :

$$\boldsymbol{\lambda}=2(\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t)^{-1}(\mathbf{d}_{n+1}-\mathbf{H}_{n+1}\boldsymbol{\alpha}_n) \tag{B.11}$$

En mettant l'expression de $\boldsymbol{\lambda}$ dans (B.9), la mise à jour des coefficients du modèle à l'instant $n+1$ est comme suit :

$$\boldsymbol{\alpha}_{n+1}=\boldsymbol{\alpha}_n+\eta\mathbf{H}_{n+1}^t(\varepsilon\mathbf{I}+\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t)^{-1}(\mathbf{d}_{n+1}-\mathbf{H}_{n+1}\boldsymbol{\alpha}_n) \tag{B.12}$$

où le pas η contrôle la convergence et le terme $\varepsilon\mathbf{I}$ est pour la régularisation ($\mathbf{I}$ est la matrice identité). L'évaluation de l'expression (B.12) nécessite l'inversion de la matrice $(\varepsilon\mathbf{I}+\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t)$ qui est de taille $p\times p$ qui a normalement un coût faible.

– Si $\kappa(\mathbf{u}_{n+1},\cdot)$ vérifie le critère de parcimonie, on l'introduit dans le dictionnaire dont la taille augmente d'une unité ($m=m+1$) et $\kappa(\mathbf{u}_{w_{m+1}},\cdot)=\kappa(\mathbf{u}_{n+1},\cdot)$. La matrice $\mathbf{H}_{n+1}$ est alors de taille $p\times(m+1)$ telle que :

$$\mathbf{H}_{n+1}=\begin{pmatrix}\kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{n+1},\mathbf{u}_{w_{m+1}})\\ \vdots & \ddots & \vdots\\ \kappa(\mathbf{u}_{n-p+2},\mathbf{u}_{w_1}) & \cdots & \kappa(\mathbf{u}_{n-p+2},\mathbf{u}_{w_{m+1}})\end{pmatrix}$$

et $\boldsymbol{\alpha}_{n+1}$ est un vecteur $(m+1)\times 1$ tel que $\boldsymbol{\alpha}_{n+1}=(\alpha_{n+1,1},\cdots,\alpha_{n+1,m+1})^t$. Le problème (B.5) et (B.6) se transforme en :

$$\min_{\boldsymbol{\alpha}_{n+1}}\left\|\boldsymbol{\alpha}_{n+1}-\begin{bmatrix}\boldsymbol{\alpha}_n\\0\end{bmatrix}\right\|^2 \tag{B.13}$$

$$\text{sous contrainte } \mathbf{d}_{n+1}=\mathbf{H}_{n+1}\boldsymbol{\alpha}_{n+1} \tag{B.14}$$

La même procédure utilisée dans le premier cas, mène à la solution du problème d'optimisation (B.13) sous contrainte (B.14) pour obtenir la mise à jour des coefficients du modèle :

$$\boldsymbol{\alpha}_{n+1}=\begin{bmatrix}\boldsymbol{\alpha}_n\\0\end{bmatrix}+\eta\mathbf{H}_{n+1}^t(\varepsilon\mathbf{I}+\mathbf{H}_{n+1}\mathbf{H}_{n+1}^t)^{-1}(\mathbf{d}_{n+1}-\mathbf{H}_{n+1}\begin{bmatrix}\boldsymbol{\alpha}_n\\0\end{bmatrix}) \tag{B.15}$$

Bibliographie

[ABR64] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25 :821–837, 1964. 18

[Ach08] R. El Achkar. *Contribution à l'étude et à la validation expérimentale de commandes neuronales d'un palier magnétique actif*. PhD thesis, mémoire de thèse de doctorat en Technologie de l'Information et des Systèmes, UTC, Compiègne, France, June 2008. viii, 90, 92, 93, 94, 95

[AHI08] A. Al-Hinai and M. Ibnkahla. Neural network nonlinear mimo channel identification and receiver design. In *Communications, 2008. ICC '08, Beijing, China, 19-23 May. IEEE International Conference on*, pages 835–839, 2008. 64

[AN10] R. Achkar and C. Nasr. Real time application of an amb using mlp : Study of robustness. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2010 Second International Conference on*, pages 9 –14, sept. 2010. 86

[ANDMC06] R. Achkar, C. Nasr, J. De Miras, and A. Charara. A new method of controlling active magnetic bearing through neural network. In *Computer Aided Control System Design, International Conference on Control Applications, International Symposium on Intelligent Control, 2006 IEEE*, pages 2848–2853, oct. 2006. 7, 86, 93, 94

[ARMNAH09] H. Abdul Rashid, A. Mohamad Noh, and M. Abdul Halim. control of an active magnetic bearing system using sliding mode techniques. In *3rd SEATUC Symposium Proceeding, 25th - 26th February 2009, Johor Bahru, Malaysia*, pages 337 –344, 2009. 86

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3) :337–404, 1950. 5, 12, 14, 15

[BCR84] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984. 16

[BDMV12] S. Bonnet, J. De Miras, and B. Vidolov. Commande non linéaire d'un centreur magnétique par inversion numérique de modèle de comportement. In *Septième Conférence Internationale Francophone d'Automatique*, pages pp. 419–424, Grenoble, France, July 2012. 6 pages. 89

[BL13] K. Bache and M. Lichman. UCI machine learning repository, 2013. 76, 78

[Bon08] S. Bonnet. *Approches numériques pour la commande des systèmes dynamiques*. PhD thesis, mémoire de thèse de doctorat en Contrôle des Systèmes, UTC, Compiègne, France, 2008. 87, 89, 92

[Bur96] C. J.C. Burges. Simplified support vector decision rules. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 71–77. Morgan Kaufmann, 1996. 21

[CC92] A. Charara and B. Caron. Magnetic bearing : Comparison between linear and nonlinear functioning. *in Proc. 3rd Int. Symp. Magn. Bearings*, pages 451 –463, 1992. 86, 90

[CCBG07] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Mach. Learn.*, 69(2-3) :143–167, December 2007. 21

[CCL92] A. Charara, B. Caron, and G. Lemarquand. Modelling and noninteractive control of an active magnetic bearing. *International Journal of applied Electromagnetics in Materials*, pages 359 –368, 1992. 90

[CDMC96] A. Charara, J. De Miras, and B. Caron. Nonlinear control of a magnetic levitation system without premagnetization. *Control Systems Technology, IEEE Transactions on*, 4(5) :513 –523, sep 1996. 86, 90

[CFV05] M. Cuturi, K. Fukumizu, and J.-P. Vert. Semigroup kernels on measures. *Journal of Machine Learning Research*, 6 :1169–1198, 2005. 16

[CH53] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience, 1953. 16

[CS02] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39 :1–49, 2002. 41

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1 edition, March 2000. 17

[DE02] D.L. Donoho and M. Elad. *Optimally Sparse Representation in General (non-orthogonal) Dictionaries Via L1 Minimization*. Technical report (Stanford University. Dept. of Statistics). Department of Statistics, Stanford University, 2002. 28

[DH01] D.L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *Information Theory, IEEE Transactions on*, 47(7) :2845 –2862, nov 2001. 28

[DH02] T.J. Dodd and R.F. Harrison. A theory of iterative sparse solutions to strict interpolation in reproducing kernel hilbert spaces. *Technical repport.*, 827, 2002. 24

[DKH03] T. J. Dodd, V. Kadirkamanathan, and R. F. Harrison. Function estimation in hilbert space using sequential projections. *Intell. Control Syst. Signal Process.*, pages 113–118, 2003. 24, 59

[DMH03] T. J. Dodd, B. Mitchinson, and R. F. Harrison. Sparse stochastic gradient descent learning in kernel models. *2nd Int. Conf. Computat. Intell., Robot. Autonomous Syst.*, 2003. 24

[DNH+05] T. J. Dodd, S. Nair, R. F. Harrison, V. Kadirkamanathan, and S. Phonphitakchai. Sparse,online learning in reproducing kernel hilbert spaces. *IEEE Trans. Signal Processing*, 2005. 24

[DSsS05] O. Dekel, S. Shalev-shwartz, and Y. Singer. The forgetron : A kernel-based perceptron on a fixed budget. In *In Advances in Neural Information Processing Systems 18*, pages 259–266. MIT Press, 2005. 21

[EMM02] Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *Proceedings of the 13th European Conference on Machine Learning*, ECML '02, pages 84–96, London, UK, UK, 2002. Springer-Verlag. 27

[EMM04] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *Signal Processing, IEEE Transactions on*, 52(8) :2275 – 2285, aug. 2004. 27, 30, 69, 77

[EPP00] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In *Advances in Computational Mathematics*, pages 1–50. MIT Press, 2000. 21

[FCC98] T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel-adatron algorithm : a fast and simple learning procedure for support vector machines. In *Machine Learning : Proceedings of the Fifteenth International Conference*. Morgan Kaufmann Publishers, 1998. 21

[GLF02] T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels for structured data. In *Inductive Logic Programming, 12th International Conference, ILP 2002, Sydney, Australia, July 9-11, 2002.*, pages 66–83. Springer–Verlag, 2002. 16

[GMS03] A. C. Gilbert, S. Muthukrishnan, and M. J. Strauss. Approximation of functions over redundant dictionaries using coherence. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 243–252, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. 28

[GV06] R. Gribonval and P. Vandergheynst. On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *Information Theory, IEEE Transactions on*, 52(1) :255 – 261, jan. 2006. 28

[Hat12] D. Hathaway. The sunspot cycle, http ://solarscience.msfc.nasa.gov/ SunspotCycle.shtml, March 2012. 61

[Hau99] D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999. 16

[Hay08] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, November 2008. 5, 12, 37, 64

[HMA10] Y. Harkouss, S. Mcheik, and R. Achkar. Accurate wavelet neural network for efficient controlling of an active magnetic bearing system. *Journal of Computer Science*, 6(12) :1457 –1464, 2010. 86

[Hon07] P. Honeine. *Méthodes à noyau pour l'analyse et la décision en environnement non-stationnaire*. PhD thesis, mémoire de thèse de doctorat en Optimisation et Sûreté des Systèmes, Ecole doctoral SSTO - UTT, Troyes, France, 2007. 19, 28, 29, 51, 53, 59, 69

[HR07] P. Honeine and C. Richard. Signal-dependent time-frequency representations for classification using a radially gaussian kernel and the alignment criterion. In *Statistical Signal Processing. SSP '07. IEEE/SP 14th Workshop on*, pages 735 –739, aug. 2007. 28

[HRB07] P. Honeine, C. Richard, and J. C. M. Bermudez. Modélisation parcimonieuse non linéaire en ligne par une méthode à noyau reproduisant et un critère de cohérence. In *Actes du XXI-ème Colloque GRETSI sur le Traitement du Signal et des Images*, Troyes, France, September 2007. 28, 30, 51, 69

[Jan04] A. Janczak. *Identification of Nonlinear Systems Using Neural Networks and Polynomial Models : A Block-Oriented Approach*. Lecture Notes in Control and Information Sciences. Springer, 2004. 64

[JD85] B. Jackson and Y. Destombes. The potential application of the active magnetic bearings to high power electrical machines. *Lse Eng. Bull.*, 15(2) :5 – 9, 1985. 86

[Joa99] T. Joachims. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, USA, 1999. MIT Press. 21

[KSW04] J. Kivinen, A.J. Smola, and R.C. Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8) :2165–2176, August 2004. 5

[KW71] G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J.˜Math. Anal. Applic.*, 33 :82–95, 1971. 19

[LCW+11] Y. Liu, W. Chen, H. Wang, Z. Gao, and P. Li. Adaptive control of nonlinear time-varying processes using selective recursive kernel learning method. *Industrial & Engineering Chemistry Research*, 50(5) :2773–2780, 2011. 64

[LP08] W. Liu and J.C. Príncipe. Kernel Affine Projection Algorithms. *EURASIP Journal on Advances in Signal Processing*, 2008(1), 2008. 65

[LPH10] W. Liu, J. C. Principe, and S. Haykin. *Kernel Adaptive Filtering : A Comprehensive Introduction*. Wiley Publishing, 1st edition, 2010. 21, 29, 30, 69

[LSST+02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, and B. Scholkopf. Text classification using string kernels. *Journal of Machine Learning Research*, 2 :563–569, 2002. 16

[LZS04] F. Liu, T. Zhang, and J. Sun. Adaptive MIMO Channel Estimation and Multiuser Detection Based on Kernel Iterative Inversion. *IEICE trans. on fundamentals of electronics, communications and computer sciences*, 87(3) :649–655, 2004. 64

[MC96] J. De Miras and A. Charara. Stabilisation non lineaire d'un palier magnétique actif par passivité et mode de glissement. *Algerian Journal of Technology*, 1(5) :177 –182, 1996. 86

[MC98] J. De Miras and A. Charara. Unbalance cancellation with rotating reference control for a horizontal shaft. In *"6th International Symposium on Magnetic Bearings"*, pages 673 –682, MIT, Cambridge, MS, USA, 1998. 87

[Mer09] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209 :415–446, 1909. 17

[Mir98] J. De Miras. *Contribution à l'élimination du balourd pour une machine tournante à Paliers Magnétiques Actifs par des techniques de commandes non linéaires*. PhD thesis, mémoire de thèse de doctorat en Contrôle des Systèmes, UTC, Compiègne, France, 1998. viii, 86, 87, 91, 92

[MY86] Fumio Matsumura and T. Yoshimoto. System modeling and control design of a horizontal-shaft magnetic-bearing system. *Magnetics, IEEE Transactions on*, 22(3) :196–203, 1986. 85

[MZ93] S.G. Mallat and Z. Zhifeng. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12) :3397 –3415, dec 1993. 28

[NY91] K. Noami and H. Yamaguchi. Robust control of magnetic bearing systems by means of sliding mode control. In *Symp. Applicat. Electromagnetism - ISEM91*, Sendai, Japan, Jan 1991. 86

[OFG97] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285, 1997. 21

[OG99] E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 271–283, Cambridge, MA, USA, 1999. MIT Press. 21

[Ogu07] T. Ogunfunmi. *Adaptive Nonlinear System Identification : The Volterra and Wiener Model Approaches*. Springer, 2007. 37

[Pla99] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press. 21

[QnCR05] J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6 :1939–1959, December 2005. 21

[RBH09] C. Richard, J.C.M. Bermudez, and P. Honeine. Online prediction of time series data with kernels. *Signal Processing, IEEE Transactions on*, 57(3) :1058 –1067, march 2009. 5, 28, 53, 59, 65

[RH07] C. Richard and P. Honeine. Filtrage adaptatif non linéaire par méthode à noyau. In *Journée signal, reconnaissance des formes et machines à noyaux, journées thématiques au GdR ISIS*, 8 juin 2007. 28

[RLCS03] C. Richard, R. Lengellé, I. Constantine, and L. Soufflet. Structures à noyau reproduisant pour le filtrage adaptatif. In *Actes du 19e Colloque GRETSI sur le Traitement du Signal et des Images*, 2003. 37

[RS80] M. Reed and B. Simon. *Method of modern mathematical physics. Vol.1 : Functional Analysis*. Academic Press, San Diego, 1980. 15

[Say03] A. Sayed. *Fundamentals of Adaptive Filtering*. Wiley, New York, 2003. 29, 33, 65, 115

[SB01] A. J. Smola and P. Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001. 21

[SBT12] K. Slavakis, P. Bouboulis, and S. Theodoridis. Adaptive Multiregression in Reproducing Kernel Hilbert Spaces : The Multiaccess MIMO Channel Case. *IEEE Trans. on Neural Networks and Learning Systems*, 23(2) :260 –276, feb. 2012. 64

[Sch80] M. Schetzen. *The Volterra and Wiener theories of nonlinear systems*. Wiley, 1980. 5, 12

[Sch06] M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Publishing Co., Inc., Melbourne, FL, USA, 2006. 37

[SHSW00] B. Schölkopf, R. Herbrich, A.J. Smola, and R.C. Williamson. A generalized representer theorem. Technical Report 81, NeuroCOLT, 2000. 18, 19

[SLV00] J.A.K. Suykens, L. Lukas, and J. Vandewalle. Sparse approximation using least squares support vector machines. In *IEEE International Symposium on Circuits and Systems ISCAS'2000, 2000*, pages 757–760, 2000. 21

[SN98] S. Sivrioglu and K. Nonami. Sliding mode control with time-varying hyperplane for amb systems. *Mechatronics, IEEE/ASME Transactions on*, 3(1) :51 –59, mar 1998. 86

[SS03] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, statistics and computing, 2003. 19, 64

[ST08] K. Slavakis and S. Theodoridis. Sliding window generalized kernel affine projection algorithm using projection mappings. *EURASIP J. Adv. Sig. Proc.*, 2008, 2008. 65

[STC04] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, June 2004. 17

[SV99] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers, 1999. 19

[SW95] R.D. Smith and W.F. Weldon. Nonlinear control of a rigid rotor magnetic bearing system : modeling and simulation with full state feedback. *Magnetics, IEEE Transactions on*, 31(2) :973 –980, march 1995. 86, 90

[SWL03] M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Workshop on AI and Statistics 9, January 3-6, 2003, Key West, Florida*, 2003. 21

[TCK04] N.C. Tsai, C. W. Chiang, and C. H. Kuo. Robust sliding mode control for axial AMB systems. In *Control Conference, 2004. 5th Asian*, volume 1, pages 64 –69 Vol.1, july 2004. 86

[TGMS03] J.A. Tropp, A.C. Gilbert, S. Muthukrishnan, and M.J. Strauss. Improved sparse approximation over quasiincoherent dictionaries. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I – 37–40 vol.1, sept. 2003. 28

[Tip01] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1 :211–244, September 2001. 21

[TSY11] S. Theodoridis, K. Slavakis, and I. Yamada. Adaptive learning in a world of projections. *Signal Processing Magazine, IEEE*, 28(1) :97 –123, jan. 2011. 64

[Vap82] V. Vapnik. *Estimation of dependencies based on empirical data*. Springer Series in Statistics. Springer-Verlag, New York, 1982. 21

[Vap95] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. 12, 17, 21

[Ver02] J. P. Vert. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 649–660. World Scientific, 2002. 16

[VVVS06] S. Van Vaerenbergh, J. Vía, and I. Santamaría. A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, Toulouse, France, May 2006. 21

[Wah90] G. Wahba. *Spline models for observational data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990. 18, 19

[Wat99] C. Watkins. Dynamic alignment kernels. In *Advances in Large Margin Classifiers*, pages 39–50. MIT Press, 1999. 16

[Wie66] N. Wiener. *Nonlinear Problems in Random Theory (Technology Press Research Monographs)*. The MIT Press, August 1966. 5, 12, 37

Chafic SAIDE

Doctorat : Optimisation et Sûreté des Systèmes

Année 2013

Filtrage adaptatif à l'aide de méthodes à noyau. Application au contrôle d'un palier magnétique actif

L'estimation fonctionnelle basée sur les espaces de Hilbert à noyau reproduisant demeure un sujet de recherche actif pour l'identification des systèmes non linéaires. L'ordre du modèle croit avec le nombre de couples entrée-sortie, ce qui rend cette méthode inadéquate pour une identification en ligne. Le critère de cohérence est une méthode de parcimonie pour contrôler l'ordre du modèle. Le modèle est donc défini à partir d'un dictionnaire de faible taille qui est formé par les fonctions noyau les plus pertinentes.

Une fonction noyau introduite dans le dictionnaire y demeure même si la non-stationnarité du système rend sa contribution faible dans l'estimation de la sortie courante. Il apparaît alors opportun d'adapter les éléments du dictionnaire pour réduire l'erreur quadratique instantanée et/ou mieux contrôler l'ordre du modèle.

La première partie traite le sujet des algorithmes adaptatifs utilisant le critère de cohérence. L'adaptation des éléments du dictionnaire en utilisant une méthode de gradient stochastique est abordée pour deux familles de fonctions noyau. Cette partie a un autre objectif qui est la dérivation des algorithmes adaptatifs utilisant le critère de cohérence pour identifier des modèles à sorties multiples.

La deuxième partie introduit d'une manière abrégée le palier magnétique actif (PMA). La proposition de contrôler un PMA par un algorithme adaptatif à noyau est présentée pour remplacer une méthode utilisant les réseaux de neurones à couches multiples.

Mots clés : séries chronologiques - filtres adaptatifs - noyaux (analyse fonctionnelle) - Hilbert, espaces de - apprentissage automatique - systèmes à entrées multiples et à sorties multiples – paliers magnétiques.

Adaptive Filtering using Kernel Methods. Application to the Control of an Active Magnetic Bearing

Function approximation methods based on reproducing kernel Hilbert spaces are of great importance in kernel-based regression. However, the order of the model is equal to the number of observations, which makes this method inappropriate for online identification. To overcome this drawback, many sparsification methods have been proposed to control the order of the model. The coherence criterion is one of these sparsification methods. It has been shown possible to select a subset of the most relevant passed input vectors to form a dictionary to identify the model.

A kernel function, once introduced into the dictionary, remains unchanged even if the non-stationarity of the system makes it less influent in estimating the output of the model. This observation leads to the idea of adapting the elements of the dictionary to obtain an improved one with an objective to minimize the resulting instantaneous mean square error and/or to control the order of the model.

The first part deals with adaptive algorithms using the coherence criterion. The adaptation of the elements of the dictionary using a stochastic gradient method is presented for two types of kernel functions. Another topic is covered in this part which is the implementation of adaptive algorithms using the coherence criterion to identify Multiple-Outputs models.

The second part introduces briefly the active magnetic bearing (AMB). A proposed method to control an AMB by an adaptive algorithm using kernel methods is presented to replace an existing method using neural networks.

Keywords: time series analysis - adaptive filters - kernel functions - Hilbert space - machine learning - multiple-input multiple-output systems – magnetic bearings.