Modélisation dynamique par réseaux de neurones et machines à vecteurs supports: contribution à la maîtrise des émissions polluantes de véhicules automobiles.

Marc Lucea

UNIVERSITE PIERRE & MARIE CURIE

THESE DE DOCTORAT DE L'UNIVERSITE PARIS 6

Spécialité :

Electronique

Présentée par :

Marc Lucea

Pour obtenir le grade de

DOCTEUR de L'UNIVERSITE PARIS 6

Sujet de la thèse :

Modélisation dynamique par réseaux de neurones et machines à vecteurs supports : contribution à la maîtrise des émissions polluantes de véhicules automobiles

Date de soutenance : 22 septembre 2006

Devant le jury composé de :

Bernadette Dorizzi,
Professeur, Institut national des télécommunications, rapporteur

Gérard Dreyfus,
Professeur, Ecole supérieure de physique et de chimie industrielles, directeur de thèse.

Patrick Gallinari,
Professeur, Université Pierre et Marie Curie, examinateur.

Mickaël Mallet,
Ingénieur, Direction de l'électronique avancée de Renault, examinateur.

Yacine Oussar,
Maître de conférences, Ecole supérieure de physique et de chimie industrielles, examinateur.

Manuel Samuelides,
Professeur, Ecole nationale supérieure de l'aéronautique et de l'espace, rapporteur.

Tables des matières

INTRODUCTION 1

1 Problématique générale 1

2 La Direction de la Recherche de Renault 2

3 Plan du mémoire 3

CHAPITRE 1 : Modélisation de processus 5

1 Introduction 5

2 Processus et modèles 5

2.1 Processus : définition 5

2.2 Modèle de processus 6

2.3 Modèle postulé et prédicteur optimal 7

2.4 Quelques hypothèses usuelles 8

2.4.1 Hypothèse purement déterministe 8

2.4.2 Hypothèse bruit de sortie 9

2.4.3 Hypothèse NARMAX 9

3 Modélisation par apprentissage : introduction à la théorie statistique de l'apprentissage 10

3.1 Risque empirique et risque moyen 11

3.2 Convergence uniforme en probabilité du risque empirique 12

3.3 Détermination de majorants du risque moyen 15

3.4 Minimisation du risque structurel 16

CHAPITRE 2 : Outils d'optimisation 19

1 Introduction 19

2 Optimisation paramétrique 19

2.1 Position du problème 19

2.2 Modèles linéaires vis-à-vis des paramètres 20

2.3 Modèles non linéaires vis-à-vis des paramètres 20

2.3.1 Méthodes du premier ordre 21

2.3.2 Méthodes du second ordre 22

2.3.3 L'algorithme de BFGS 23

2.3.4 L'algorithme de Levenberg-Marquardt 24

3 Optimisation lagrangienne - Le problème dual 25

3.1 Conditions théoriques d'optimalité 26

3.1.1 Problème sans contraintes 26

3.1.2 Problème avec contraintes d'égalité 27

3.1.3 Problème avec contraintes d'inégalité 28

3.2 Le problème dual 30

3.2.1 Conditions de point-selle 30

3.2.2 Expression du problème dual 31

4 Conclusion 31

CHAPITRE 3 : Les réseaux de neurones 34

1 Introduction 34

2 Définitions et notations 35

2.1 Réseau de neurones formels 35

2.2 Propriété d'approximation universelle et forme canonique 36

2.2.1 Le Perceptron à une couche cachée 36

2.2.2 Forme canonique des réseaux récurrents 37

3 Estimation des paramètres - Apprentissage 39

3.1 Apprentissage de réseaux statiques 39

3.2 Apprentissage de réseaux récurrents 41

3.2.1 Apprentissage dirigé / prédicteur non bouclé 42

3.2.2 Apprentissage semi-dirigé / prédicteur bouclé 43

3.3 Sélection de modèles 47

3.3.1 Minima locaux et initialisation des paramètres 47

3.3.2 Surajustement et dilemme biais-variance 48

3.4 Quelques méthodes de régularisation 48

3.4.1 Modération des paramètres 49

3.4.2 Arrêt prématuré 49

4 Modélisation de la température après turbine d'un véhicule 50

4.1 Contexte et motivation 50

4.2 Choix d'une hypothèse et élaboration d'une base d'apprentissage 52

4.3 Résultats et discussion 54

4.3.1 Elaboration du modèle neuronal récurrent 54

4.3.2 Comparaison à un modèle semi-physique existant 57

4.3.2.1 Description du modèle : équations physiques 57

4.3.2.2 Corrections apportées : modèle « boîte grise » 62

4.3.2.3 Résultats et comparaison 63

5 Modélisation des émissions polluantes d'un véhicule de série 64

5.1 Contexte et motivation 64

5.2 Dispositifs expérimentaux et élaboration de la base de données 66

5.2.1 Mesure des émissions polluantes 66

5.2.2 Choix des variables du modèle et élaboration de la base de données 68

5.3 Elaboration du modèle neuronal 70

5.4 Résultats et discussion 72

5.5 Conclusion *75*

6 Système de commande neuronale prédictive en boucle ouverte 75

6.1 Introduction *75*

6.2 Description du système de commande *76*

6.3 Optimisation de la fonction de coût *78*

6.4 Illustration sur un processus simulé *79*

6.4.1 Description du processus et élaboration d'un modèle neuronal récurrent 79

6.4.2 Commande prédictive en boucle ouverte du processus simulé 81

6.5 Conclusion *83*

CHAPITRE 4 : Les machines à vecteurs supports et autres méthodes de noyaux pour la régression 87

1 Introduction 87

2 Espaces de Hilbert à noyaux reproduisants 88

2.1 Définition et propriétés *88*

2.1.1 Définition 88

2.1.2 Propriétés 88

2.1.3 Exemple : noyaux linéaires 89

2.1.4 Cas des noyaux non linéaires 89

2.2 Construction d'un RKHS *90*

2.3 Lien avec la minimisation du risque structurel *91*

2.4 Résumé : apprentissage et noyaux *92*

3 Least Squares SVM (LSSVM) 93

3.1 Description de la méthode et résolution 93

3.1.1 Recherche de la solution par utilisation du théorème de représentation 94

3.1.2 Recherche de la solution par exploitation des conditions de points selle 96

3.1.3 Recherche de la solution par résolution des équations de Karush, Kuhn et Tucker 97

3.2 Variante parcimonieuse : Sparse least squares SVM (sLSSVM) 98

4 Les Machines à Vecteurs Supports (SVM) pour la régression 100

4.1 Formulation du problème dans le cas linéaire 100

4.2 Extension au cas non linéaire 102

4.3 Choix des hyperparamètres 103

4.4 Choix de la fonction de coût 104

4.4.1 Fonctions de coût et modèles de bruit associés 104

4.4.2 Problème d'optimisation pour une fonction de coût générale 106

5 Quelques autres méthodes de noyaux 107

5.1 Analyse en Composantes Principales par noyaux (K-PCA) 107

5.1.1 Principe général 107

5.1.2 Détermination du nombre de composantes principales 109

5.1.3 Coordonnées d'un point dans la nouvelle base et reconstruction 109

5.2 Moindres carrés partiels par noyaux (K-PLS) 110

6 Apprentissage de modèles récurrents : approche analytique 112

7 Apprentissage de modèles récurrents : approche algorithmique 115

7.1 Approche maître-élève pour les modèles de noyaux 116

7.1.1 Cas des modèles statiques 117

7.1.2 Cas des modèles dynamiques 119

7.1.3 Conclusion sur l'approche maître-élève. Critère retenu 120

7.2 Description de l'algorithme d'apprentissage semi dirigé 120

7.2.1 Principe 120

7.2.2 Différentes variantes **122**

7.3 Résultats et conclusion ***124***

7.3.1 Algorithme semi dirigé : approche maître-élève **124**

7.3.2 Algorithme semi dirigé : processus dynamique simulé **130**

8 Conclusion **133**

CONCLUSION **138**

ANNEXE 1 : calcul exact de la matrice Hessienne d'une fonction réalisée par un réseau récurrent **141**

ANNEXE 2 : brevet relatif à l'estimation de la température après turbine **163**

Remerciements

Je tiens à adresser mes plus vifs remerciements au Professeur Gérard DREYFUS, mon directeur de thèse, pour sa rigueur scientifique, ses conseils avisés, et pour tout le temps qu'il a su m'accorder, dans un agenda pourtant débordant. Votre connaissance pointue des problématiques de recherche actuelles, et votre conscience des contraintes industrielles m'ont permis de faire aboutir ce travail. Merci également pour m'avoir encouragé dans les moments les plus difficiles, sur le plan scientifique autant qu'humain : sans votre soutien, ce mémoire de thèse n'aurait sans doute pas vu le jour.

Je suis également très reconnaissant à Yacine OUSSAR d'avoir accepté d'encadrer mon travail au Laboratoire d'Electronique de l'ESPCI. Son enthousiasme sans faille et les nombreuses discussions que nous avons eues ont permis d'éclairer de nombreux problèmes, allant du bug d'un algorithme d'apprentissage au choix d'un appareil photo numérique. Je tiens donc à lui manifester ici toute ma gratitude.

Merci à Mickael MALLET, qui a encadré mon travail chez RENAULT, pour avoir accepté de se pencher sur des problèmes parfois très éloignés de son domaine de recherche, et pour l'aide qu'il a su m'apporter, tant vis-à-vis des questions de « culture générale automobile », que vis-à-vis des « procédures internes Renault ». Merci également à Caroline NETTER et Laurent LAUMESFELD, qui ont un temps été mes encadrants RENAULT, avant d'être réclamés dans d'autres services.

Madame Bernadette DORIZZI, Professeur à l'Institut National des Télécommunications d'Evry, et Monsieur Manuel SAMUELIDES, Professeur à l'Ecole Supérieure d'Aéronautique de Toulouse, ont consacré une partie de leur temps à lire ce mémoire et à rédiger un rapport sur ces travaux, et je tiens à les en remercier chaleureusement. Merci également à Monsieur Patrick GALLINARI, Professeur à l'Université Pierre et Marie Curie, qui a accepté de participer à mon jury de soutenance.

Je tiens par ailleurs à remercier l'ensemble des membres du Laboratoire d'Electronique de l'ESPCI : Ginette BOSSAVIT, Rémi DUBOIS, Arthur DUPRAT, Nicole GENDREY, Aurélie GOULON, Olivier PARODI, Pierre ROUSSEL et François VIALATTE. Merci à tous pour votre accueil, votre présence et votre aide, tout au long de ces trois années.

Luc BOURGEOIS m'a accueilli dans son équipe, à la Direction de la Recherche de RENAULT, et je tiens à lui manifester ma reconnaissance pour la confiance qu'il m'a témoignée. Josselin VISCONTI, Yann COLETTE et Sébastien MICHEL ont participé aux travaux menés autour de l'estimation des émissions polluantes : merci à tous pour l'aide que vous m'avez apportée dans la découverte de ce sujet. Un grand merci aussi à Pascal BARRILLON et Sébastien ROMIEUX, qui m'ont initié à la problématique de l'estimation de la température après turbine : les nombreuses informations que vous m'avez apportées en vous appuyant sur votre modèle semi physique ont grandement simplifié l'élaboration du modèle neuronal dont il est question dans cette thèse.

En tentant de remonter « aux sources », je voudrais remercier ma mère Nicole et mon père Jacques, qui m'ont permis d'étudier, autant matériellement que psychologiquement. Cette démarche de questionnement et de recherche, c'est en grande partie à votre éducation que je la dois.

Je tiens enfin à exprimer ici toute ma reconnaissance à Hélène DAVID, ma compagne, qui m'a supporté, réconforté, motivé, entouré,… et surtout aimé, à travers toutes les difficultés que j'ai rencontrées durant cette thèse, et malgré mon caractère qui me surprend moi-même si souvent.

Introduction

1 Problématique générale

La complexité croissante des systèmes employés dans l'industrie automobile, en termes de fonctions réalisées et de méthodes de mise en œuvre, mais aussi en terme de norme d'homologation, amène à envisager des outils toujours plus innovants lors de la conception d'un véhicule. On observe d'ailleurs depuis quelques années une forte augmentation du nombre de brevets déposés, en particulier dans le domaine des systèmes électroniques, dont l'importance ne cesse de croître au sein d'un véhicule automobile moderne. Cette complexité croissante des fonctions réalisées requiert une précision de description accrue pour les dispositifs impliqués, notamment pour les systèmes complexes où une approche analytique est difficilement envisageable. Aux impératifs de précision de la description, qui imposent souvent de prendre en considération les non-linéarités des processus, s'ajoute donc la complexité d'analyse des phénomènes physiques à l'origine des observations que l'on souhaite modéliser.

Les développements qu'ont connus ces dernières années les techniques de modélisation non linéaires par apprentissage (notamment les réseaux de neurones formels ou les machines à vecteurs supports), alliés à la croissance de la capacité des ordinateurs et des calculateurs embarqués dans les véhicules automobiles, justifient donc l'intérêt porté par Renault à ces outils.

C'est dans cette optique qu'a été envisagée une étude portant sur les méthodes de modélisation non linéaire par apprentissage, dont l'objectif était d'en tester les secteurs d'applications possibles dans le contexte automobile, et d'en évaluer les difficultés de mise en œuvre ainsi que les gains attendus. Cette étude a fait l'objet d'une collaboration, sous forme d'un contrat de thèse CIFRE, avec le Laboratoire d'Electronique de l'Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris (ESPCI), dirigé par le Professeur Gérard Dreyfus.

2 La Direction de la Recherche de Renault

Ce mémoire de thèse résume les travaux effectués durant trois ans entre le Laboratoire d'Electronique de l'ESPCI et le groupe Contrôle des Systèmes de la Direction de la Recherche de Renault. L'une des missions de ce groupe, constitué d'une trentaine de personnes encadrées par Luc Bourgeois, est de développer, sous forme de projets de recherche, des solutions innovantes en rupture avec l'existant, et d'en évaluer les gains en termes de coût nécessaire au développement et à la mise en œuvre, de délai et de prestation accomplie. L'un des enjeux actuels majeurs de l'industrie automobile est en effet de parvenir à réduire les temps et les coûts de développement d'un véhicule, tout en proposant au client des prestations innovantes : un équilibre doit donc être trouvé entre l'intérêt pratique et commercial d'une solution, et l'effort nécessaire à sa réalisation.

Nous nous sommes principalement intéressés, durant cette thèse, à la modélisation de certains phénomènes physiques qu'il est nécessaire de contrôler au sein d'un véhicule. On distingue deux types de modèles, dans le domaine automobile, suivant l'utilisation qui en est faite :

- *modèles embarqués* : ce type de modèle a pour vocation d'être implanté physiquement sur un calculateur de véhicule. Cette caractéristique impose de tenir compte des conditions particulières d'utilisation, vis-à-vis des capacités de mémoire et de la puissance du calculateur. On prêtera d'autre part une attention particulière à la stabilité et à la robustesse du modèle, afin d'exclure tout dysfonctionnement dans des conditions de roulage, d'autant plus que le modèle en question intervient sur des organes sensibles mettant en jeu la sûreté de fonctionnement du véhicule. À titre d'exemple, citons les capteurs logiciels, utilisés en remplacement de capteurs physiques, dont l'un des intérêts pratiques est de réduire les coûts de production du véhicule.
- *modèles débarqués* : l'utilisation de modèles peut également intervenir dans la phase de conception du véhicule, comme outil de prédiction d'une caractéristique particulière, afin d'aider au dimensionnement ou à la mise au point du véhicule. Ces modèles débarqués sont implantés sur des ordinateurs de bureau, dont les capacités sont généralement bien supérieures à celles des calculateurs embarqués, de sorte qu'elles ne constituent pas un facteur limitant. Bien que la stabilité et la robustesse des modèles conservent une importance, elle ne revêtent alors pas le même caractère crucial que lors d'une utilisation en embarqué. L'utilisation de modèles de prédiction

permet, d'un point de vue pratique, de s'affranchir des essais correspondants, et donc de réduire le temps et les coûts de développement d'un véhicule.

De manière générale, les techniques de modélisation par apprentissage permettent d'aborder la modélisation de phénomènes physiques dont la description est ardue, élargissant ainsi le champ des possibles en matière de modélisation, mais également de s'affranchir d'une description physique détaillée pour des processus connus, réduisant ainsi le temps de développement d'un modèle particulier. En contrepartie, l'élaboration de tels modèles par apprentissage requiert la réalisation de mesures sur ledit processus, ce qui implique des coûts qui sont parfois loin d'être négligeables. Notre objectif a donc été d'identifier certains problèmes correspondant à la première approche, c'est-à-dire pour lesquels la réalisation de modèles de connaissance est soit inenvisageable soit particulièrement ardue.

3 Plan du mémoire

Le premier chapitre de ce mémoire s'attachera à rappeler les concepts de base relatifs à la modélisation de processus par apprentissage. Nous y introduirons les notions essentielles que nous serons amenés à employer par la suite. Dans le deuxième chapitre, nous décrirons les principaux outils d'optimisation nécessaires à l'élaboration de modèles par apprentissage. Le troisième chapitre regroupera l'ensemble des travaux menés, au cours de cette thèse, sur le thème des réseaux de neurones. Après avoir rappelé la méthodologie d'élaboration de modèles neuronaux, en particulier dans le cas récurrent, nous présenterons les résultats obtenus sur deux applications industrielles : l'estimation de la température en un point particulier de la ligne d'échappement, et l'estimation des émissions de différents polluants en sortie d'échappement. Ces deux applications participent à la maîtrise des émissions polluantes, soit durant l'utilisation habituelle d'un véhicule, car la connaissance de cette température est indispensable à la mise en œuvre des stratégies de dépollution actives, soit au stade de la mise au point du moteur, qui sera facilitée par l'utilisation d'un modèle de prédiction des débits de polluants en fonction des réglages du moteur. Nous décrirons également un système de commande optimale en boucle ouverte, associé à un modèle neuronal, et destiné à réduire les variations rapides de la sortie d'un processus : ce système est susceptible d'être utilisé pour contrôler les à-coups de couple d'un véhicule, consécutifs à une variation rapide de l'enfoncement de la pédale. Une méthode de calcul exact de la matrice Hessienne, dans le cas de modèles décrits par des équations récurrentes, sera alors introduite pour permettre l'utilisation de ce système de commande dans le cas de processus dynamiques.

Dans le quatrième chapitre, nous nous intéresserons aux méthodes de modélisation par noyaux, dont font partie les machines à vecteurs supports, et tenterons de les adapter à la modélisation de processus dynamiques, d'abord par un traitement analytique (pour l'une particulière de ces méthodes), avant de proposer une approche itérative du problème d'apprentissage, inspirée de l'algorithme d'apprentissage semi dirigé utilisé pour les réseaux de neurones récurrents.

Modélisation de processus

1 Introduction

Nous allons aborder dans ce chapitre la question de la modélisation de processus. Cette notion sera décrite dans un contexte général, et nous insisterons sur les modalités de mise en œuvre dans le cas particulier de la modélisation par apprentissage.

Dans une première partie, nous définirons les notions de processus et de modèles, avant d'introduire dans une seconde partie les principaux concepts relatifs à la théorie statistique de l'apprentissage.

2 Processus et modèles

2.1 Processus : définition

Les processus auxquels nous nous intéresserons sont des systèmes physiques, qui évoluent au cours du temps sous l'effet de différentes actions, et sur lesquels il est possible de réaliser certaines observations ou mesures. Il existe de très nombreux types de processus : industriels, financiers, sociaux,… On caractérise un processus par :

- *ses sorties* : il s'agit des grandeurs d'intérêt, que l'on suppose mesurables, et qui sont sensibles aux actions auxquelles est soumis le processus
- *ses variables d'entrées* : il s'agit de l'ensemble des actions ayant une influence sur les sorties du processus. On distingue deux types d'entrées : celles qui sont mesurables et contrôlées par l'opérateur, que nous appellerons entrées de commande, et celles sur lesquelles il n'est pas possible d'agir, et qui sont alors considérées comme des perturbations, regroupées sous le terme de bruit.

Un modèle d'un processus est dit déterministe si les valeurs des sorties sont déterminées de manière certaine lorsque les entrées le sont. Il est en revanche dit stochastique si ces relations obéissent à des lois de probabilité, de telle sorte qu'il n'est pas possible de connaître de manière certaine l'évolution du modèle de ce processus au cours du temps. Dans la pratique, on est amené à modéliser de manière stochastique un processus déterministe, quand la nature des perturbations qui l'affectent est inconnue, c'est-à-dire quand l'incertitude sur ses entrées de commande est trop importante.

2.2 Modèle de processus

Un modèle d'un processus est une description mathématique de son fonctionnement, qui permet de rendre compte des relations existant entre ses entrées et ses sorties. Cette description peut être fondée sur :

- une analyse physique des phénomènes entrant en jeu dans le processus : on parle alors de *modèle de connaissance*. Ce type de modèle présente l'intérêt d'expliquer la nature des phénomènes qui interviennent dans le processus, et d'en comprendre l'importance relative par une analyse détaillée. Dans un contexte industriel, une évolution technique du processus pourra alors être intégrée au modèle existant, sans en changer toute la structure ;
- une analyse statistique des mesures de ses entrées et sorties : on parle alors de *modèle boîte noire,* ou de *modèle de comportement,* dont les paramètres peuvent être estimés *par apprentissage*. On utilise principalement ce type de modèles lorsque la nature des phénomènes intervenant dans le processus est inconnue, ou non descriptible de façon analytique. Leur inconvénient principal réside dans le fait qu'ils n'expliquent en rien le comportement du processus, et que toute évolution du processus nécessite une nouvelle modélisation complète.

Quel que soit le type de modèle envisagé, celui-ci est caractérisé par un domaine de validité, qui définit le domaine des entrées et sorties dans lequel les relations établies sont satisfaisantes. Ce domaine de validité est associé soit aux approximations ou lois physiques utilisées (dans le cas d'un modèle de connaissance), soit aux incertitudes statistiques (qui dépendent des mesures employées pour son élaboration) et au choix de la représentation adoptée, dans le cas d'un modèle boîte noire.

Suivant la description mathématique obtenue, on distinguera également deux types de modèles :

- *modèle statique* : les équations mathématiques qui régissent le modèle sont algébriques.
- *modèle dynamique* : les équations mathématiques qui régissent le modèle sont des équations différentielles ou des équations aux différences récurrentes.

Nous nous intéresserons principalement aux modèles dynamiques dans la suite de ce mémoire. Une autre distinction porte sur l'utilisation faite du modèle. On parlera de :

- *modèle de simulation* : quand le modèle a pour but de fonctionner indépendamment du processus, c'est-à-dire sans qu'aucune information (mesures) ne soit nécessaire à sa

mise en œuvre. Ces modèles sont d'une grande utilité lorsque l'on cherche à valider des hypothèses sur le processus, ou à prédire son comportement en l'absence de mesures (nous en verrons des illustrations aux Chapitres 3 et 4).

- *modèle de prédiction* : un modèle de prédiction a pour but d'estimer les sorties à venir du processus, connaissant celles-ci aux instants précédents. Dans la pratique, ce type de modèle est utilisé lorsque l'on cherche à connaître par anticipation les sorties du processus considéré, notamment pour mettre en œuvre des stratégies dépendant de ces valeurs à venir.

2.3 Modèle postulé et prédicteur optimal

Lors de la conception d'un modèle, notamment dans le cas d'une modélisation « boîte noire », il convient de faire des hypothèses concernant la forme mathématique de la fonction ou des fonctions qui constituent ce modèle ; ces choix sont en partie dictés par les informations dont on dispose sur la nature du processus (statique ou dynamique, linéaire ou non,…), notamment quant à la manière dont celui-ci est affecté par le bruit (bruit d'état, bruit de mesure,…).
L'ensemble de ces hypothèses et choix préliminaires conduit à postuler un modèle, dont la forme la plus générale, pour un processus déterministe à temps discret non linéaire et sans retard, peut s'écrire de deux façons, suivant le choix de représentation que l'on fait :

- *représentation entrée-sortie* :

$$\mathbf{y}^p\left(k+1\right)=\mathbf{\Phi}\left(\mathbf{y}^p\left(k\right),...,\mathbf{y}^p\left(k-N_s\right),\mathbf{u}\left(k\right),...,\mathbf{u}\left(k-N_e\right)\right) \tag{1}$$

où $\mathbf{y}^p\left(k\right)$ représente le vecteur des sorties du processus à l'instant discret k, où $\mathbf{u}\left(k\right)$ est le vecteur de ses entrées (de commande ou de perturbation) au même instant, où N_s et N_e sont deux entiers caractérisant le modèle, et où $\mathbf{\Phi}$ est une fonction de ces grandeurs.

- *représentation d'état* :

$$\begin{cases}\mathbf{x}^p\left(k+1\right)=\mathbf{\Phi}\left(\mathbf{x}^p\left(k\right),\mathbf{u}\left(k\right)\right)\\ \mathbf{y}^p\left(k+1\right)=\mathbf{\Psi}\left(\mathbf{x}^p\left(k+1\right),\mathbf{u}\left(k\right)\right)\end{cases} \tag{2}$$

où $\mathbf{x}^p\left(k\right)$ est le vecteur d'état du modèle à l'instant discret k, et où $\mathbf{\Phi}$ et $\mathbf{\Psi}$ sont deux fonctions potentiellement non linéaires de leurs arguments. Par définition, l'ordre du modèle est la dimension du vecteur d'état.

Les hypothèses faites sur la nature du processus doivent apparaître dans le modèle postulé. À titre d'exemple, on peut prendre, pour une représentation entrée-sortie :

- N_s =0 si le processus est supposé statique,
- $\mathbf{\Phi}$ fonction linéaire si le processus est supposé linéaire en ses différentes variables,
- un vecteur $\mathbf{u}(k)$ constitué des seules entrées de commande si l'on considère que le processus n'est pas soumis à des perturbations mesurables.

Une fois choisie l'hypothèse sur la forme du modèle, il convient d'établir l'expression du *prédicteur optimal* (ou *modèle optimal*) associé, c'est-à-dire l'expression mathématique permettant de prédire le plus précisément possible les sorties du processus à l'instant $k+d$ connaissant celles-ci à l'instant k, et en supposant que l'hypothèse employée est exacte. En considérant la sortie du processus à l'instant $k+d$ comme une réalisation particulière d'une variable aléatoire $\mathbf{Y}^p(k+d)$, on cherche donc à déterminer, pour le prédicteur optimal, une formule mathématique (qui fait intervenir des grandeurs $\mathbf{\Phi}$, $\mathbf{\Psi}$, N_s et N_e identiques à celles du modèle postulé) vérifiant : $E\left[\mathbf{Y}^p(k+d)\right] = y(k+d)$. L'hypothèse concernant la manière dont le bruit affecte le processus est alors le facteur principal à prendre en considération pour aboutir à une expression du prédicteur optimal. Nous allons illustrer cela par quelques exemples dans le paragraphe suivant.

2.4 Quelques hypothèses usuelles

2.4.1 Hypothèse purement déterministe

Cette hypothèse consiste à supposer que les sorties du processus s'expriment comme fonction des entrées de commande, et ne sont soumises à aucune perturbation non mesurable. On écrit alors :

$$\mathbf{y}^p(k) = \mathbf{\Phi}\left(\mathbf{y}^p(k-1),...,\mathbf{y}^p(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e)\right) \tag{3}$$

Le prédicteur optimal associé à l'hypothèse (3), dont on suppose qu'elle est exacte (c'est-à-dire que les grandeurs $\mathbf{\Phi}$, N_s et N_e permettent une représentation fidèle du comportement du processus), s'écrit alors :

$$\mathbf{y}(k) = \mathbf{\Phi}\left(\mathbf{y}^p(k-1),...,\mathbf{y}^p(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e)\right) \tag{4}$$

Les sorties estimées par le prédicteur (4) sont telles qu'elles coïncident parfaitement avec la mesure des sorties du processus. De ce fait, on peut remplacer à chaque instant les mesures

par leurs prédictions, de sorte que l'expression suivante constitue également un prédicteur optimal :

$$\mathbf{y}(k) = \mathbf{\Phi}\left(\mathbf{y}(k-1),...,\mathbf{y}(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e)\right) \quad (5)$$

La différence principale entre les expressions (4) et (5) réside en ce que l'estimation des paramètres de la fonction **Φ** nécessite des algorithmes d'apprentissage différents : la formule (4) correspond à un prédicteur à 1 pas (modèle de prédiction), tandis que la formule (5) correspond à un prédicteur bouclé (modèle de simulation). Nous présenterons, dans la partie consacrée à l'apprentissage des réseaux de neurones, les algorithmes correspondants.

2.4.2 Hypothèse bruit de sortie

On considère ici que les sorties du processus sont soumises à un bruit additif, qui n'affecte pas l'évolution ultérieure du processus. Cette hypothèse se traduit par un modèle de la forme :

$$\begin{cases} \mathbf{x}^p(k) = \mathbf{\Phi}\left(\mathbf{x}^p(k-1),...,\mathbf{x}^p(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e)\right) \\ \mathbf{Y}^p(k) = \mathbf{x}^p(k) + \mathbf{W}(k) \end{cases} \quad (6)$$

où $\mathbf{W}(k)$ est une variable aléatoire vectorielle, qui représente le bruit et les perturbations non déterministes, que l'on suppose de moyenne nulle et d'écart-type fini.

Le prédicteur optimal associé à l'hypothèse (6) a l'expression suivante :

$$\mathbf{y}(k) = \mathbf{\Phi}\left(\mathbf{y}(k-1),...,\mathbf{y}(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e)\right) \quad (7)$$

avec pour conditions initiales $\mathbf{y}(i) = \mathbf{x}^p(i)$, $i = 1,..,N_s$. En effet, on a, d'après (7), $\mathbf{y}(k) = \mathbf{x}^p(k)\ \forall k$, et, d'après (6), $E\left(\mathbf{Y}^p(k)\right) = \mathbf{x}^p(k)$, donc $E\left(\mathbf{Y}^p(k)\right) = \mathbf{y}(k)$. Ce résultat est intuitif : pour minimiser l'erreur de prédiction, il convient en effet d'utiliser comme arguments de la fonction **Φ** non pas les mesures des sorties, qui sont entachées de bruit, mais les prédictions de celles-ci, que l'on suppose exactes du fait qu'elles emploient les mêmes grandeurs **Φ**, N_s et N_e. En pratique, on est donc amené à réaliser l'apprentissage d'un prédicteur bouclé, comme décrit par la formule (7).

2.4.3 Hypothèse NARMAX

L'hypothèse NARMAX (Non linéaire Auto Régressif à Moyenne Ajustable et à entrées eXogènes) consiste à considérer que le bruit qui affecte le processus a une influence sur les sorties à l'instant *k*, avec un horizon temporel N_w. On écrit donc :

$$\mathbf{Y}^p(k) = \mathbf{\Phi}\left(\mathbf{y}^p(k-1),...,\mathbf{y}^p(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e),\mathbf{w}(k-1),...,\mathbf{w}(k-N_w)\right) + \mathbf{W}(k) \quad (8)$$

où $\mathbf{W}(k)$ est une variable aléatoire vectorielle, que l'on suppose de moyenne nulle et d'écart-type fini.

Pour minimiser l'erreur de prédiction, il faut définir un prédicteur tel que, à chaque instant, on ait : $\mathbf{y}^p(k) - \mathbf{y}(k) = \mathbf{w}(k)$. L'expression de ce prédicteur optimal est donc la suivante :

$$\mathbf{y}(k) = \mathbf{\Phi}\left(\mathbf{y}^p(k-1),...,\mathbf{y}^p(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e),\mathbf{w}(k-1),...,\mathbf{w}(k-N_w)\right)$$

Il est toutefois impossible de réaliser un tel prédicteur, du fait que les valeurs du bruit $\mathbf{w}(k-1),...,\mathbf{w}(k-N_w)$ sont inconnues. On les estime donc au moyen de l'erreur de prédiction aux mêmes instants, soit $\mathbf{e}(k-n) = \mathbf{y}^p(k-n) - \mathbf{y}(k-n)$ (pour n = 1, …, N_w). L'expression du prédicteur optimal devient alors :

$$\mathbf{y}(k) = \mathbf{\Phi}\left(\mathbf{y}^p(k-1),...,\mathbf{y}^p(k-N_s),\mathbf{u}(k-1),...,\mathbf{u}(k-N_e),\mathbf{e}(k-1),...,\mathbf{e}(k-N_w)\right) \quad (9)$$

avec pour conditions initiales $\mathbf{y}(i) = E\left(\mathbf{Y}^p(i)\right), i = 1,...,\max(N_s, N_w)$. On a alors, d'après (8), $\mathbf{e}(i) = \mathbf{y}^p(i) - \mathbf{y}(i) = \mathbf{w}(i)$, pour $i = 1,...,N_w$, donc $\mathbf{y}(k) = \mathbf{y}^p(k) - \mathbf{w}(k)$, $\forall k \geq N_s + 1$, d'après (9). L'erreur de prédiction est égale à tout instant à la réalisation du bruit, soit $\mathbf{e}(k) = \mathbf{y}^p(k) - \mathbf{y}(k) = \mathbf{w}(k)$, de sorte que le prédicteur (9) est bien optimal : $E\left(\mathbf{Y}(k)\right) = E\left(\mathbf{Y}^p(k)\right)$.

L'apprentissage d'un modèle, sur la base de ce prédicteur optimal (9), s'effectue donc de manière non bouclé. Son utilisation correspond à celle d'un modèle de prédiction, qui requiert la connaissance des erreurs (donc des mesures des sorties du processus) aux instants précédents.

3 Modélisation par apprentissage : introduction à la théorie statistique de l'apprentissage

Après avoir introduit les notions de modèle et de processus, nous allons à présent décrire quelques résultats importants, issus de la théorie statistique de l'apprentissage [1][2]; ils s'appliquent à l'ensemble des modèles boîtes noires, qui nous intéresseront dans la suite de ce mémoire.

La théorie statistique de l'apprentissage se situe à la frontière de plusieurs disciplines, incluant évidemment la statistique, mais aussi la théorie de l'information, l'analyse fonctionnelle ou la mécanique statistique. Elle aborde la question de l'apprentissage de systèmes complexes sur

la base d'un ensemble de données, en définissant des bornes supérieures à des quantités caractérisant la qualité du modèle, bornes qu'il est proposé d'optimiser pour réaliser l'apprentissage au sein d'une classe de fonctions particulières. Cette approche est innovante par rapport aux statistiques classiques, en ce sens qu'elle entend intégrer la notion de capacité de généralisation du modèle durant la phase d'élaboration de celui-ci.

3.1 Risque empirique et risque moyen

Soient $\mathbf{x} \in X$ et $y \in Y$, où X et Y sont deux ensembles mesurables, deux variables aléatoires associées à une distribution de probabilité conjointe $P(\mathbf{x}, y)$. On suppose disposer d'une base de données $D_N = \{(\mathbf{x}_i, y_i) \in X \times Y\}_{i=1}^{N}$, constituée de N réalisations de ces variables.

Le problème d'apprentissage est le suivant : étant donné D_N, trouver un bon estimateur f : X→Y, au sein d'un classe de fonctions donnée, qui associe à tout $\mathbf{x}$ une valeur $f(\mathbf{x})$ « proche » de y, notamment pour des valeurs qui ne sont pas présentes dans la base d'apprentissage. Idéalement, on souhaite donc minimiser la quantité suivante, appelée risque moyen (« expected risk »), définie, pour une fonction f donnée, par :

$$I[f] = \int_{X,Y} V(y, f(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy \tag{10}$$

où $V(y, f(\mathbf{x}))$ est une « fonction de perte » (loss function) qui reflète l'erreur commise en approchant y par $f(\mathbf{x})$. Au sein d'une classe de fonctions $\mathcal{F}$, la solution f_o du problème est alors définie par :

$$f_o = \arg\min_{\mathcal{F}} I[f] \tag{11}$$

On est donc amené à utiliser un principe d'induction, du fait que la distribution de probabilité $P(\mathbf{x},y)$ est inconnue. À titre d'exemple, on peut minimiser une quantité appelée risque empirique, définie par :

$$I_{emp}[f, N] = \frac{1}{N} \sum_{i=1}^{N} V(y_i, f(\mathbf{x}_i)) \tag{12}$$

En minimisant une telle quantité (principe appelé ERM, pour « Empirical Risk Minimization »), on s'expose au problème de surajustement, car aucune information relative à la distribution de probabilité sous-jacente n'est prise en considération. D'autre part, le problème de la minimisation de $I_{\text{emp}}[f,N]$ est mal posé, en ce sens qu'il peut y avoir plusieurs fonctions f (éventuellement en nombre infini) pour lesquelles cette quantité est minimale.

3.2 Convergence uniforme en probabilité du risque empirique

La question posée est alors de savoir sous quelles conditions la minimisation du risque empirique permet de converger en probabilité vers $I[f_o]$, convergence qui s'exprime de la façon suivante :

$$\lim_{N\to\infty} I_{emp}\left[\hat{f}_N, N\right] = \lim_{N\to\infty} I\left[\hat{f}_N\right] = I\left[f_o\right] \tag{13}$$

où $\hat{f}_N$ est la fonction pour laquelle le risque empirique (12) est minimal.

Il a pu être établi [3] que l'équation (13) est vérifiée si et seulement si la condition suivante est remplie :

$$\forall \varepsilon>0,\ \lim_{N\to\infty} P\left\{\sup_{f\in\mathcal{F}}\left|I\left[f\right]-I_{emp}\left[f,N\right]\right|>\varepsilon\right\}=0 \tag{14}$$

Cette équation (14) donne une condition nécessaire et suffisante pour que la solution minimisant le risque empirique converge uniformément en probabilité vers la solution idéale du problème. Cette condition n'a que peu d'intérêt pratique, puisqu'elle s'exprime dans la limite où le nombre d'exemples d'apprentissage tend vers l'infini.

La théorie statistique de l'apprentissage s'attache à déterminer des conditions pratiques pour assurer la convergence uniforme en probabilité du risque empirique vers le risque moyen, et aborde cette question au travers des trois quantités suivantes, définies pour un ensemble de fonctions $V\left(y, f\left(\mathbf{x}\right)\right), f\in\mathcal{F}$:

- VC-entropie $H^{\mathcal{F}}\left(\varepsilon, N\right)$:

$$H^{\mathcal{F}}\left(\varepsilon, N\right) = \int_{X\times Y} \ln\left(\mathcal{N}^{\mathcal{F}}\left(\varepsilon, D_N\right)\right)\prod_{i=1}^{N} P\left(\mathbf{x}_i, y_i\right) d\mathbf{x}_i dy_i \tag{15}$$

  où $\mathcal{N}^{\mathcal{F}}\left(\varepsilon, D_N\right)$ est le nombre minimal d'éléments de $\mathcal{F}$ permettant d'approcher tout autre élément à ε près, au sens d'une mesure sur cet ensemble $\mathcal{F}$, relativement à l'ensemble de données D_N, définie de la façon suivante : $\forall\left(f, f'\right)\in\mathcal{F}^2, d\left(f, f'\right) = \max_{1\le i\le N}\left|V\left(y_i, f\left(\mathbf{x}_i\right)\right)-V\left(y_i, f'\left(\mathbf{x}_i\right)\right)\right|$.

- VC-entropie « recuite » $H_{rec}^{\mathcal{F}}\left(\varepsilon, N\right)$:

$$H_{rec}^{\mathcal{F}}\left(\varepsilon, N\right) = \ln\int_{X\times Y} \mathcal{N}^{\mathcal{F}}\left(\varepsilon, D_N\right)\prod_{i=1}^{N} P\left(\mathbf{x}_i, y_i\right) d\mathbf{x}_i dy_i \tag{16}$$

- Fonction de croissance $G^{\mathcal{F}}\left(\varepsilon, N\right)$:

$$G^{\mathcal{F}}(\varepsilon,N)=\ln\left(\sup_{D_N\in(X\times Y)^N}\mathcal{N}^{\mathcal{F}}(\varepsilon,D_N)\right) \tag{17}$$

Il est clair que ces trois quantités sont fonctions du nombre d'exemples N et de ε, et que :

$$H^{\mathcal{F}}(\varepsilon,N)\le H_{rec}^{\mathcal{F}}(\varepsilon,N)\le G^{\mathcal{F}}(\varepsilon,N)$$

Etant donné un ensemble D_N fixé, on associe à chaque élément $f\in\mathcal{F}$ un ensemble de valeurs $\{f(x_1),\ldots,f(x_N)\}$, qui caractérise « l'état » de cette fonction f. Deux « états » seront considérés comme différents si la distance $d(f,f')$, qui dépend de l'ensemble de données D_N et de la fonction V utilisée pour définir le risque empirique, est supérieure à ε. La quantité $\mathcal{N}^{\mathcal{F}}(\varepsilon,D_N)$ peut alors s'interpréter comme étant le nombre « d'états » différents accessibles par les fonctions de l'ensemble $\mathcal{F}$, étant donné D_N.

Le logarithme de cette quantité, soit $\ln\left(\mathcal{N}^{\mathcal{F}}(\varepsilon,D_N)\right)$, s'appelle en physique statistique l'entropie partielle, pour une « configuration » particulière D_N de l'ensemble de données. Dans l'expression de la VC-entropie $H^{\mathcal{F}}(\varepsilon,N)$, une sommation de ces entropies partielles sur l'ensemble des « configurations » possibles (soit $\int_{X\times Y}d\mathbf{x}_i dy_i$), pondérées par la probabilité de réalisation de chacune (soit $\prod_{i=1}^{N}P(\mathbf{x}_i,y_i)$ en supposant les variables aléatoires correspondantes indépendantes et identiquement distribuées), permet d'aboutir à l'entropie totale de l'ensemble de fonctions.

Pour le calcul de la VC-entropie recuite $H_{rec}^{\mathcal{F}}(\varepsilon,N)$, la sommation sur l'ensemble des « configurations » possibles est effectuée avant de passer au logarithme, ce qui revient à exprimer l'entropie totale comme le logarithme de la somme du « nombre total de complexions » plutôt que comme la somme des entropies partielles associées à chaque « configuration ».

Dans l'expression de la fonction de croissance $G^{\mathcal{F}}(\varepsilon,N)$, le « nombre total de complexions » est cette fois exprimé indépendamment de toute fonction de distribution $P(\mathbf{x}_i,y_i)$, en majorant le nombre d'états différents accessibles pour les fonctions de l'ensemble $\mathcal{F}$, étant donné D_N, par la valeur maximale prise par $\mathcal{N}^{\mathcal{F}}(\varepsilon,D_N)$ sur l'ensemble $X\times Y$.

Les résultats principaux de la théorie statistique de l'apprentissage donnent des conditions de convergence uniforme en probabilité, comme définie par la formule (14), que l'on peut exprimer comme suit [2] :

- Pour une distribution de probabilité conjointe $P(\mathbf{x}, y)$ donnée, une condition nécessaire et suffisante de convergence est que :

$$\forall \varepsilon > 0, \ \lim_{N \to \infty} \frac{H^{\mathcal{F}}(\varepsilon, N)}{N} = 0$$

- Pour une distribution de probabilité conjointe $P(\mathbf{x}, y)$ donnée, une condition suffisante pour un taux de convergence asymptotique rapide est que :

$$\forall \varepsilon > 0, \ \lim_{N \to \infty} \frac{H_{red}^{\mathcal{F}}(\varepsilon, N)}{N} = 0$$

- Indépendamment de la distribution de probabilité $P(\mathbf{x}, y)$ sous-jacente, une condition suffisante pour un taux de convergence rapide est que :

$$\forall \varepsilon > 0, \ \lim_{N \to \infty} \frac{G^{\mathcal{F}}(\varepsilon, N)}{N} = 0$$

Le troisième résultat est d'autant plus important qu'il s'applique pour toute distribution de probabilité conjointe $P(\mathbf{x}, y)$, et notamment dans le cas qui nous intéresse où celle-ci est inconnue.

Pour poursuivre l'analogie avec la physique statistique, ces résultats expriment que lorsque l'incertitude sur les fonctions de la classe $\mathcal{F}$ relativement aux N exemples de la base de données D_N (quantifiée par l'une ou l'autre des trois quantités précédentes) croit moins vite que l'information apportée par l'ajout d'exemples à la base de données (quand N tend vers l'infini), alors la fonction qui minimise le risque empirique (au sens de la fonction de coût V utilisée pour caractériser la différence entre deux « états ») tendra à commettre une erreur égale à celle de la solution idéale f_0. L'ajout d'un exemple supplémentaire à la base de données peut en effet s'interpréter comme une augmentation de la dimensionnalité du vecteur caractérisant l'état d'une fonction f sur l'ensemble de données D_N, soit $\{f(x_1), \ldots, f(x_N), f(x_{N+1})\}$: par ajout d'un « descripteur » supplémentaire, on favorise l'apparition d'états différents accessibles, et donc l'augmentation de $\mathcal{N}^{\mathcal{F}}(\varepsilon, D_N)$ et consécutivement de l'entropie.

Il est néanmoins délicat dans la pratique de calculer la fonction de croissance $G^{\mathcal{F}}(\varepsilon, N)$, ce qui nous amène à considérer un majorant de cette quantité appelé VC-dimension, qui caractérise également la complexité de l'ensemble de fonctions $\mathcal{F}$.

3.3 Détermination de majorants du risque moyen

Dans le cas de fonctions indicatrices (la fonction indicatrice standard est $\theta(\mathbf{x})=1$ si $\mathbf{x}>0$, $\theta(\mathbf{x})=0$ sinon), employées dans le contexte de la classification, on appelle VC-dimension d'une classe de fonctions définie sur un ensemble $\mathcal{F}$ le nombre maximal h d'exemples pouvant être séparés en deux classes par l'une des fonctions de cet ensemble, pour toutes les 2^h dichotomies possibles.

La VC-dimension d'un ensemble de fonctions $\{V(y, f(\mathbf{x})), f \in \mathcal{F}\}$, dans le cas général, est alors définie comme étant la VC-dimension de l'ensemble des fonctions indicatrices $\{\theta(V(y, f(\mathbf{x})) - \alpha), \alpha \in [A, B], f \in \mathcal{F}\}$, où θ est la fonction indicatrice standard, et où A et B sont respectivement un minorant et un majorant de la fonction $V(y, f(\mathbf{x}))$ sur l'ensemble $\mathcal{F}$.

Une propriété remarquable de cette VC-dimension est qu'elle permet de majorer l'écart entre risque empirique et risque moyen, comme énoncé dans le théorème suivant, formulé initialement par V. Vapnik et A. Chervonenkis [4]:

avec une probabilité 1-η, les inégalités suivantes sont vérifiées pour tout élément $f \in \mathcal{F}$

$$I_{emp}[f, N] - (B - A)\sqrt{\frac{h\ln\left(\frac{2eN}{h}\right) - \ln\left(\frac{\eta}{4}\right)}{N}} \leq I[f] \leq I_{emp}[f, N] + (B - A)\sqrt{\frac{h\ln\left(\frac{2eN}{h}\right) - \ln\left(\frac{\eta}{4}\right)}{N}} \quad (18)$$

où h est la VC-dimension de l'ensemble de fonctions $\{V(y, f(\mathbf{x})), f \in \mathcal{F}\}$, et où A et B sont deux grandeurs bornant la fonction de coût V sur $\mathcal{F}$: $A \leq V(y, f(\mathbf{x})) \leq B$ pour tout $f \in \mathcal{F}$.

Ce théorème permet d'établir des majorants à l'écart entre risque empirique et risque moyen, et présente la caractéristique remarquable, très intéressante d'un point de vue pratique, de s'appliquer à des ensembles de données de taille N finie (l'exploitation de ces majorants impose toutefois d'avoir N supérieur à la VC-dimension h de l'ensemble de fonctions choisi).

On déduit aisément de la formule (18) le résultat suivant :

$$I\left[\hat{f}_N\right] - 2(B-A)\sqrt{\frac{h\ln\left(\frac{2eN}{h}\right) - \ln\left(\frac{\eta}{4}\right)}{N}} \le I\left[f_o\right] \le I\left[\hat{f}_N\right] + 2(B-A)\sqrt{\frac{h\ln\left(\frac{2eN}{h}\right) - \ln\left(\frac{\eta}{4}\right)}{N}} \quad (19)$$

où $\hat{f}_N$ est la fonction de $\mathcal{F}$ minimisant le risque empirique (12).

Pour tenter de s'approcher au mieux du risque moyen, il convient donc de minimiser non pas simplement le risque empirique, mais également le second terme des inégalités ci-dessus. Une minimisation importante du risque empirique, par le choix d'une classe de fonctions $\mathcal{F}$ suffisamment complexes (c'est-à-dire ayant une VC-dimension grande), s'accompagnera nécessairement d'une augmentation de ce second terme, ce qui constitue une nouvelle illustration du dilemme biais / variance.

3.4 Minimisation du risque structurel

Sur la base des résultats énoncés au paragraphe précédent, nous allons à présent introduire la notion de minimisation du risque structurel. Le principe de cette méthode est de définir une séquence d'espaces de fonctions candidates (appelés espaces hypothèse) qui sont imbriqués $H_1 \subset H_2 \subset \ldots \subset H_m$ et ont des capacités croissantes $h_1 \le h_2 \le \ldots \le h_m$, puis de déterminer la fonction minimisant le risque empirique dans chacun des H_i, notée $\hat{f}_{i,N}$:

$$\hat{f}_{i,N} = \arg\min_{f \in H_i} I_{emp}\left[f, N\right] \quad (20)$$

Par construction des H_i, on s'attend à ce que l'erreur empirique diminue quand i augmente, c'est-à-dire quand la capacité de l'espace hypothèse croit. Ce phénomène de diminution du risque empirique s'accompagne toutefois à nouveau d'une augmentation du second terme des inégalités (19), qui caractérise la capacité de l'espace de fonctions considéré. On choisira donc comme solution du problème la fonction $\hat{f}_{i^*,N}$ qui minimisera l'ensemble du membre de droite de l'inégalité (19), soit :

$$\hat{f}_{i^*,N} = \arg\min_{f \in H_i} I_{emp}\left[f, N\right] + \Phi\left(\sqrt{\frac{h}{N}}, \eta\right) \quad (21)$$

où Φ est une fonction croissante de ses deux arguments (cf. équation (19)).

On peut alors montrer [5] que pour cette solution $\hat{f}_{i^*,N}$ les inégalités (18) et (19) restent valables avec une probabilité $(1-\eta)^m \approx 1 - m\eta$, en remplaçant h par h_{i^*}, f_o par f_{i^*}, et $\hat{f}_N$ par $\hat{f}_{i^*,N}$ (ceci afin que ces inégalités s'appliquent à l'ensemble de la séquence d'espaces imbriqués H_i).

En définitive, le principe de minimisation du risque structurel consiste en la recherche d'une fonction dont le risque empirique s'approche du risque moyen, par optimisation d'un critère qui intègre également la capacité de l'ensemble des fonctions candidates, définies au sein d'une séquence d'espaces hypothèse imbriqués.

Bibliographie

[1] V. Vapnik. *Statistical Learning Theory*. John Wiley, 1998.

[2] T. Evgeniou, M. Pontil, T. Poggio. *A unified framework for regularization networks and support vector machines*. Technical Report No. 1654, Massachusetts Institute of Technology, 1999.

[3] V.N. Vapnik. *Estimation of dependences based on empirical data.* Nauka, Moscow, 1979 (traduction anglaise: Springer Verlag, Berlin, 1982).

[4] V.N. Vapnik, A.Y. Chervonenkis. *On the uniform convergence of relatives frequencies of events to their probabilities.* Th. Prob. And its Applications, 17(2), pp. 264-280, 1971.

[5] A. N. Tikhonov, V.Y. Arsenin. *Solutions of ill-posed problems.* W.H. Winston, Washington, D.C., 1977.

Outils d'optimisation

1 Introduction

Quel que soit le type de modèle paramétré envisagé, il est nécessaire, lors de la phase d'élaboration de celui-ci, d'estimer les valeurs de ses paramètres. Cette étape, appelée identification par les automaticiens, requiert l'utilisation de techniques d'optimisation, qui peuvent différer suivant la nature du modèle [1]. Nous distinguerons deux types d'optimisation :

- optimisation paramétrique : cette approche consiste à déterminer les paramètres optimaux par minimisation directe d'une fonction de coût vis-à-vis des paramètres du modèle. Suivant l'approximation choisie, ce type d'optimisation fait appel au calcul du gradient et/ou de la matrice Hessienne de la fonction de coût. C'est le type d'optimisation qui sera mise en œuvre lors de l'apprentissage de réseaux de neurones.
- optimisation lagrangienne : certains problèmes d'optimisation s'expriment sous forme d'une fonction de coût à minimiser, soumise à un certain nombre de contraintes. L'introduction d'une fonction de Lagrange permet alors de résoudre ce type de problème d'optimisation bien plus efficacement que par une approche paramétrique. Nous utiliserons notamment ce formalisme pour l'apprentissage de machines à vecteurs supports.

2 Optimisation paramétrique

2.1 Position du problème

Soit $\left\{\mathbf{x}_n, y_n^p\right\}_{1\le n\le N}$ une séquence de N mesures des entrées et sorties du processus à modéliser, où $\mathbf{x}_n$ et y_n^p représentent respectivement le vecteur regroupant toutes les variables du modèle et la sortie du processus à l'instant discret n. Le problème d'apprentissage revient alors à déterminer, parmi un ensemble de fonctions candidates $\psi\left(\mathbf{x},\boldsymbol{\theta}\right)$, paramétrées par le vecteur $\boldsymbol{\theta}$, celle qui minimise une fonction de coût prédéfinie, reflétant l'écart entre les sorties $y_n = \psi\left(\mathbf{x}_n,\boldsymbol{\theta}\right)$ estimées par le modèle et les mesures y_n^p de cette sortie. On utilise fréquemment la fonction de coût des moindres carrés, définie par :

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\left(y_n^p - y_n(\boldsymbol{\theta})\right)^2 \tag{1}$$

2.2 Modèles linéaires vis-à-vis des paramètres

Dans le cas particulier où la fonction $\psi(\mathbf{x},\boldsymbol{\theta})$ est linéaire par rapport aux paramètres, représentés par le vecteur $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_m)^T$, le modèle peut être décrit par une équation du type :

$$y_n = \psi(\mathbf{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{\varphi}(\mathbf{x}_n) \text{ pour } n = 1, \ldots, N \tag{2}$$

où $\boldsymbol{\varphi}(.)$ est une fonction vectorielle, de dimension m, du vecteur des variables du modèle. Dans le domaine de l'apprentissage, les fonctions φ_i (i = 1, …, m) sont appelées « descripteurs » (traduction du terme anglais *features*). Ces fonctions ne contiennent pas de paramètres ajustables pendant l'apprentissage. Elles peuvent néanmoins mettre en œuvre des hyper-paramètres, dont la détermination par le concepteur peut constituer un problème non trivial.

Sous forme matricielle, le problème d'optimisation des moindres carrés [2] s'écrit alors :

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}) = \frac{1}{2}\left(\mathbf{Y}_p - \mathbf{X}\boldsymbol{\theta}\right)^T \left(\mathbf{Y}_p - \mathbf{X}\boldsymbol{\theta}\right) \\ &\text{avec } \mathbf{X} = \begin{bmatrix} \boldsymbol{\varphi}(\mathbf{x}_1)^T \\ \vdots \\ \boldsymbol{\varphi}(\mathbf{x}_N)^T \end{bmatrix},\ \mathbf{Y}_p = \begin{bmatrix} y_1^p \ldots y_N^p \end{bmatrix}^T,\ \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \ldots \theta_m \end{bmatrix}^T \end{aligned} \tag{3}$$

La matrice $\mathbf{X}$ est appelée *matrice des observations*.

La solution $\boldsymbol{\theta}_{mc}$ de ce problème, dite *solution des moindres carrés*, s'obtient en imposant la nullité de la dérivée de J par rapport à $\boldsymbol{\theta}$:

$$\left[\mathbf{X}^T\mathbf{X}\right]\boldsymbol{\theta}_{mc} = \mathbf{X}^T\mathbf{Y}_p \Leftrightarrow \boldsymbol{\theta}_{mc} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{Y}_p \tag{4}$$

Cette dernière relation suppose que la matrice $\mathbf{X}^T\mathbf{X}$ soit inversible, ce qui est généralement vérifié lorsque le nombre d'exemples N est grand devant le nombre m de descripteurs du modèle.

2.3 Modèles non linéaires vis-à-vis des paramètres

La solution des moindres carrés, présentée dans le paragraphe précédent, constitue un moyen aisé de déterminer les paramètres optimaux d'un modèle linéaire. Dans le cas non linéaire [3],

une approche itérative doit être adoptée pour rechercher le minimum de la fonction de coût des moindres carrés (1).

2.3.1 Méthodes du premier ordre

Les méthodes dites du premier ordre sont fondées sur le calcul du gradient de la fonction de coût J par rapport au vecteur des paramètres $\boldsymbol{\theta} = \left(\theta_1, \theta_2, \ldots, \theta_m\right)^T$. Par définition, celui-ci s'écrit :

$$\nabla J = \left(\frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \ldots, \frac{\partial J}{\partial \theta_m} \right)^T \tag{5}$$

Le principe de la méthode est alors de mettre à jour le vecteur des paramètres $\boldsymbol{\theta}$ par la formule suivante :

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \mu_k \nabla J\left(\boldsymbol{\theta}^k\right) \tag{6}$$

où l'exposant k est le numéro de l'itération, et où le scalaire μ_k est appelé pas du gradient.
Suivant la valeur du pas du gradient μ_k, on distingue deux variantes :

- *Méthode du gradient à pas constant :* dans cette variante, le pas du gradient est fixé à une valeur constante $\mu_k = \mu$. La direction de descente est donc simplement l'opposé de celle du gradient, ce qui garantit une décroissance de la fonction de coût, à condition que le pas μ soit bien choisi. En effet, une grande valeur de cette quantité assure une décroissance rapide, mais peut provoquer un phénomène d'oscillation autour du minimum de J, tandis qu'une petite valeur de μ requiert un grand nombre d'itérations pour atteindre le minimum.
- *Méthode du gradient à pas asservi :* pour tenir compte de la décroissance de la norme du gradient au voisinage du minimum, de nombreuses heuristiques ont été proposées ; par exemple, il est possible d'adapter le pas du gradient par la formule suivante :

  $$\mu_k = \mu\left(1 + \left|\nabla J\right|\right) \tag{7}$$

  où μ est un paramètre constant (typiquement, $\mu = 10^{-3}$). Cette méthode conduit à une valeur importante du pas du gradient, et donc à une décroissance rapide, loin du minimum (c'est-à-dire pour de grandes valeurs de la norme du gradient), et garantit d'autre part la convergence au voisinage du minimum, compte tenu de la faible valeur de μ.

2.3.2 Méthodes du second ordre

Dans le paragraphe précédent, nous avons exposé différentes méthodes fondées sur la seule estimation du gradient de la fonction de coût. On peut également établir une procédure de minimisation itérative fondée sur un développement limité au second ordre au voisinage du vecteur $\boldsymbol{\theta}^k$ obtenu à l'itération k :

$$J\left(\boldsymbol{\theta}^{k+1}\right) \approx J\left(\boldsymbol{\theta}^k\right) + \nabla J^T\left(\boldsymbol{\theta}^k\right)\left(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right) + \frac{1}{2}\left(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right)^T \mathbf{H}\left(\boldsymbol{\theta}^k\right)\left(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right) \tag{8}$$

où $\mathbf{H}(\boldsymbol{\theta}^k)$ est la matrice Hessienne de la fonction de coût J, définie par :

$$\mathbf{H}\left(\boldsymbol{\theta}^k\right) = \begin{bmatrix} \dfrac{\partial^2 J}{\left(\partial \theta_1^k\right)^2} & \dfrac{\partial^2 J}{\partial \theta_1^k \partial \theta_2^k} & \cdots & \dfrac{\partial^2 J}{\partial \theta_1^k \partial \theta_m^k} \\ \vdots & & & \vdots \\ \dfrac{\partial^2 J}{\partial \theta_m^k \partial \theta_1^k} & \dfrac{\partial^2 J}{\partial \theta_m^k \partial \theta_2^k} & \cdots & \dfrac{\partial^2 J}{\left(\partial \theta_m^k\right)^2} \end{bmatrix} \tag{9}$$

Supposons que $\boldsymbol{\theta}^{k+1}$ soit un minimum de la fonction de coût. Le gradient de celle-ci est donc nul, ce qui s'écrit :

$$\mathbf{0} = \nabla J^T\left(\boldsymbol{\theta}^k\right) + \mathbf{H}\left(\boldsymbol{\theta}\right)\left(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right) \tag{10}$$

On obtient donc l'estimation itérative recherchée :

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \mathbf{H}^{-1}\left(\boldsymbol{\theta}^k\right)\nabla J^T\left(\boldsymbol{\theta}^k\right) \tag{11}$$

si la matrice hessienne est inversible.

Pour que la direction de mise à jour des paramètres corresponde effectivement à une direction de descente, il est nécessaire que la matrice Hessienne soit définie positive (ce qui assure par ailleurs son inversibilité, et le fait que l'inverse est également définie positive).

Cette méthode de recherche itérative du minimum, appelée méthode de Newton, présente de très bonnes propriétés de convergence au voisinage du minimum, mais nécessite l'inversion d'une matrice, ce qui peut s'avérer lourd en termes de temps de calcul. Cependant, cette augmentation du temps de calcul pour une itération (par comparaison à la formule de mise à jour des paramètres (6)) est compensée par la diminution du nombre d'itérations nécessaires.

D'un point de vue pratique, il est donc préférable de mettre en œuvre dans un premier temps des méthodes du premier ordre, de manière à s'approcher du minimum, avant d'utiliser cette

approximation du deuxième ordre. Une étude comparative de différentes méthodes d'optimisation de ce type est donnée dans [4].

Sur la base de cette approximation dite de Newton, plusieurs algorithmes ont été développés. Nous allons décrire deux d'entre eux de manière plus détaillée [5] :

- l'algorithme de BFGS
- l'algorithme de Levenberg-Marquardt

2.3.3 L'algorithme de BFGS

Cet algorithme (du nom de ses inventeurs Broyden, Fletcher, Glodfarb et Shanno) est fondé sur une approximation de la méthode de Newton exposée précédemment. La règle de mise à jour des paramètres est définie comme suit :

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \mu_{k+1}\mathbf{M}_{k+1}\nabla J\left(\boldsymbol{\theta}^{k}\right) \tag{12}$$

où le scalaire μ_{k+1} est appelé pas de descente, et où $\mathbf{M}_{k+1}$ est une approximation, calculée itérativement, de l'inverse de la matrice Hessienne, selon la formule suivante [6] :

$$\mathbf{M}_{k+1} = \mathbf{M}_{k} + \left(1 + \frac{\boldsymbol{\gamma}_k^T \mathbf{M}_k \boldsymbol{\gamma}_k}{\boldsymbol{\delta}_k^T \boldsymbol{\gamma}_k}\right)\frac{\boldsymbol{\delta}_k^T \boldsymbol{\delta}_k}{\boldsymbol{\delta}_k^T \boldsymbol{\gamma}_k} - \frac{\boldsymbol{\delta}_k \boldsymbol{\gamma}_k^T \mathbf{M}_k + \mathbf{M}_k \boldsymbol{\gamma}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\delta}_k^T \boldsymbol{\gamma}_k} \tag{13}$$

où $\boldsymbol{\gamma}_k = \nabla J\left(\boldsymbol{\theta}^{k}\right) - \nabla J\left(\boldsymbol{\theta}^{k-1}\right)$ et $\boldsymbol{\delta}_k = \boldsymbol{\theta}^{k} - \boldsymbol{\theta}^{k-1}$. La valeur initiale de la matrice $\mathbf{M}$ est généralement la matrice identité (valeur à laquelle $\mathbf{M}_{k+1}$ sera également réinitialisée au cours de l'algorithme si elle s'avère ne plus être définie positive).

La valeur du pas μ_k peut être choisie comme étant la valeur permettant la diminution la plus importante de la fonction de coût. On procède alors de la façon suivante :

- initialiser μ_k à une petite valeur (typiquement 10^{-3}) ;
- mettre à jour les paramètres conformément à (12), puis vérifier que la condition de descente est bien respectée (soit $J\left(\boldsymbol{\theta}^{k+1}\right) < J\left(\boldsymbol{\theta}^{k}\right)$) ;
    - si tel est le cas, rechercher une valeur du pas plus importante (par multiplication par un facteur supérieur à 1), et vérifier que cette valeur conduit à une fonction de coût moindre que précédemment ;
    - si la condition de descente n'est pas vérifiée, multiplier le pas par un facteur inférieur à 1, et tester à nouveau la condition de descente. Cette procédure d'ajustement du pas est arrêtée soit si la décroissance de la fonction de coût est inférieure à un seuil prédéfini, soit si le nombre de tentatives atteint une limite prédéfinie.

L'intérêt de cet algorithme de BFGS réside en ce qu'il permet de s'affranchir du calcul de l'inverse de la matrice Hessienne (qui peut lui-même s'avérer délicat dans certains cas), en estimant itérativement une approximation de cette matrice inverse suivant la formule (13). Cette méthode quasi-newtonienne n'est efficace qu'à proximité du minimum de la fonction de coût : il convient donc de combiner l'utilisation de cet algorithme avec, dans un premier temps, une approche du minimum par une méthode du premier ordre.

2.3.4 L'algorithme de Levenberg-Marquardt

Cet algorithme, qui appartient également à la classe des méthodes quasi-newtoniennes [7], obéit à la formule suivante de mise à jour des paramètres [8] :

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \left[\mathbf{H}\left(\boldsymbol{\theta}^{k}\right) + \mu_{k+1}\mathbf{I}\right]^{-1} \nabla J\left(\boldsymbol{\theta}^{k}\right) \tag{14}$$

où $\mathbf{H}\left(\boldsymbol{\theta}^{k}\right)$ est la matrice Hessienne de la fonction de coût J, $\mathbf{I}$ est la matrice identité, et où μ_{k+1} est un scalaire appelé pas. Pour de petites valeurs du pas μ_{k+1}, cette méthode s'approche de celle de Newton, tandis que pour de grandes valeurs du pas, la méthode tend vers celle du gradient simple. En choisissant judicieusement la valeur du pas au cours de l'algorithme, il est donc possible de s'affranchir de la mise en œuvre préalable d'une méthode de gradient simple pour s'approcher du minimum.

Le calcul de l'inverse de la matrice $\left[\mathbf{H}\left(\boldsymbol{\theta}^{k}\right) + \mu_{k+1}\mathbf{I}\right]$ peut s'effectuer par des méthodes d'inversion directe. Néanmoins, compte tenu de la fonction de coût envisagée (1), il est préférable de mettre en œuvre une méthode d'inversion itérative, fondée sur la propriété suivante : étant données quatre matrice **A**, **B**, **C** et **D**

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

Or, l'expression de la matrice Hessienne est la suivante :

$$\mathbf{H}\left(\boldsymbol{\theta}^{k}\right) = \sum_{n=1}^{N}\left(\frac{\partial e_n}{\partial \boldsymbol{\theta}^{k}}\right)\left(\frac{\partial e_n}{\partial \boldsymbol{\theta}^{k}}\right)^{T} + \sum_{n=1}^{N}\frac{\partial^2 e_n}{\partial \boldsymbol{\theta}^{k}\left(\partial \boldsymbol{\theta}^{k}\right)^{T}} e_n \tag{15}$$

où $e_n = y_n^p - y_n$ est l'erreur de prédiction.

En négligeant dans la relation (15) le second terme, qui est proportionnel à l'erreur, on aboutit à l'approximation suivante de la matrice Hessienne :

$$\mathbf{H}\left(\boldsymbol{\theta}^{k}\right) \approx \tilde{\mathbf{H}}\left(\boldsymbol{\theta}^{k}\right) = \sum_{n=1}^{N}\left(\frac{\partial e_n}{\partial \boldsymbol{\theta}^{k}}\right)\left(\frac{\partial e_n}{\partial \boldsymbol{\theta}^{k}}\right)^{T} \tag{16}$$

Cette matrice Hessienne approchée obéit à la formule de récurrence suivante :

$$\tilde{\mathbf{H}}^n = \tilde{\mathbf{H}}^{n-1} + \mathbf{X}^n \left(\mathbf{X}^n\right)^T \text{ où } \mathbf{X}^n = \frac{\partial e_n}{\partial \boldsymbol{\theta}^k} = \frac{\partial y_n}{\partial \boldsymbol{\theta}^k} \text{ pour } n = 1,\ldots,N$$

Par définition, et en fixant comme valeur initiale $\tilde{\mathbf{H}}^0 = \mu_k \mathbf{I}$, on a donc : $\tilde{\mathbf{H}}^N = \tilde{\mathbf{H}} + \mu_k \mathbf{I}$. L'utilisation du lemme d'inversion énoncé précédemment (avec $\mathbf{A} = \tilde{\mathbf{H}}^{n-1}$, $\mathbf{B} = \mathbf{X}^n$, $\mathbf{C} = \mathbf{I}$, $\mathbf{D} = \left(\mathbf{X}^n\right)^T$) permet alors d'écrire :

$$\left(\tilde{\mathbf{H}}^n\right)^{-1} = \left(\tilde{\mathbf{H}}^{n-1}\right)^{-1} - \frac{\left(\tilde{\mathbf{H}}^{n-1}\right)^{-1} \mathbf{X}^n \left(\mathbf{X}^n\right)^T \left(\tilde{\mathbf{H}}^{n-1}\right)^{-1}}{\mathbf{I} + \left(\mathbf{X}^n\right)^T \left(\tilde{\mathbf{H}}^{n-1}\right)^{-1} \mathbf{X}^n} \tag{17}$$

Il est alors possible de calculer itérativement l'inverse de la matrice $\tilde{\mathbf{H}}^N = \tilde{\mathbf{H}} + \mu_k \mathbf{I}$.

Notons que cette méthode de calcul approché de l'inverse de la matrice découle de l'approximation (16), qui n'est valable que pour de faibles valeurs de l'erreur de prédiction e_n, et donc pour des valeurs de $\boldsymbol{\theta}$ proches de la valeur optimale. Le domaine de validité de l'approximation Newtonienne, a priori étendu par l'ajout du terme $\mu_k\mathbf{I}$ dans la formule (14), est finalement restreint pour pouvoir calculer efficacement l'inverse de cette matrice Hessienne augmentée.

3 Optimisation lagrangienne - Le problème dual

On s'intéresse dans cette partie à la recherche du minimum d'une fonction de coût en présence de contraintes sous forme d'égalités et/ou d'inégalités. Le problème primal s'écrit donc :

$$\begin{aligned} &\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \\ &\text{sous } \begin{cases} \mathbf{g}(\boldsymbol{\theta}) = 0 \\ \mathbf{h}(\boldsymbol{\theta}) \le 0 \end{cases} \end{aligned} \tag{18}$$

où

- J est une fonction scalaire de la variable vectorielle $\boldsymbol{\theta}$,
- la fonction vectorielle $\mathbf{g}$ de la variable vectorielle $\boldsymbol{\theta}$ représente l'ensemble des contraintes d'égalité,
- la fonction vectorielle $\mathbf{h}$ représente l'ensemble des contraintes d'inégalité.

Sous forme développée, les contraintes s'écrivent :

$$\begin{cases} g_i(\boldsymbol{\theta}) = 0, & \text{pour } i=1,...,p \\ h_j(\boldsymbol{\theta}) \le 0, & \text{pour } j=1,...,m \end{cases}$$

Des méthodes algorithmiques (du type gradient projeté [9] ou fonction de pénalisation [10]) permettent d'aborder ce type de problèmes, mais la présence des contraintes entraîne une complexité très importante.

La reformulation du problème initial, par introduction d'une fonction de Lagrange, lève en partie ces difficultés, et permet en général d'aboutir un à nouveau problème d'optimisation, appelé problème dual, dont la résolution est plus simple.

Dans un premier temps, nous allons rappeler les conditions théoriques d'optimalité, avant d'introduire le problème dual associé au problème (18).

3.1 Conditions théoriques d'optimalité

L'expression des conditions théoriques d'optimalité fait appel au développement en série de Taylor de la fonction $J(\boldsymbol{\theta})$ au voisinage de son minimum $\boldsymbol{\theta}^*$, qui s'écrit au deuxième ordre :

$$J\left(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}\right) = J(\boldsymbol{\theta}^*) + \left(\nabla J\left(\boldsymbol{\theta}^*\right)\right)^T \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T \mathbf{H}(\boldsymbol{\theta}^*)\delta\boldsymbol{\theta} + \mathrm{o}\left(\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta}\right) \qquad (19)$$

où ∇J et $\mathbf{H}$ sont respectivement le gradient et la matrice hessienne de la fonction J au point $\boldsymbol{\theta}^*$.

3.1.1 Problème sans contraintes

Le problème d'optimisation sans contraintes associé à (18) s'exprime de la façon suivante :

$$\min_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right)$$

<u>*Condition nécessaire*</u> *:* au point $\boldsymbol{\theta}^*$, la fonction $J(\boldsymbol{\theta})$ doit être stationnaire. Le terme du premier ordre dans le développement (19) doit donc nécessairement être nul pour que toute variation $\delta\boldsymbol{\theta}$ entraîne une augmentation de la fonction $J(\boldsymbol{\theta})$, soit :

$$\nabla J\left(\boldsymbol{\theta}^*\right) = \mathbf{0}$$

<u>*Condition suffisante*</u> *:* supposons la condition précédente satisfaite. Pour que $\boldsymbol{\theta}^*$ soit un minimum, il suffit alors que, pour tout $\delta\boldsymbol{\theta}$, la fonction $J(\boldsymbol{\theta})$ vérifie :

$$J\left(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}\right) \geq J\left(\boldsymbol{\theta}^*\right) + \frac{1}{2}\delta\boldsymbol{\theta}^T \mathbf{H}\left(\boldsymbol{\theta}^*\right)\delta\boldsymbol{\theta} + \mathrm{o}\left(\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta}\right)$$

La matrice Hessienne doit donc être semi-définie positive, condition que l'on exprime symboliquement par :

$$\mathbf{H}\left(\boldsymbol{\theta}^*\right) \geq 0$$

En résumé, le point $\boldsymbol{\theta}^*$ est un minimum de la fonction J si et seulement si il vérifie :

$$\begin{cases} \nabla J\left(\boldsymbol{\theta}^*\right) = 0 \\ \mathbf{H}\left(\boldsymbol{\theta}^*\right) \geq 0 \end{cases} \tag{20}$$

3.1.2 Problème avec contraintes d'égalité

La prise en considération des contraintes d'égalité du problème (18) a pour effet de restreindre le domaine admissible de $\boldsymbol{\theta}$, défini par :

$$g_i\left(\boldsymbol{\theta}\right) = 0 \text{ pour } i = 1,..., p$$

Notons que, pour que ce problème admette une solution, il faut que $p < n$.

Condition nécessaire : au voisinage d'un minimum local $\boldsymbol{\theta}^*$, on a :

$$J\left(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}\right) \approx J\left(\boldsymbol{\theta}^*\right) + \left(\nabla J\left(\boldsymbol{\theta}^*\right)\right)^T \delta\boldsymbol{\theta} \geq J\left(\boldsymbol{\theta}^*\right)$$

$$g_i\left(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}\right) = g_i\left(\boldsymbol{\theta}^*\right) + \left(\nabla g_i\left(\boldsymbol{\theta}^*\right)\right)^T \delta\boldsymbol{\theta} + \mathrm{o}\left(\delta\boldsymbol{\theta}\right) = 0 \text{ pour } i = 1,..., p$$

La variation $\delta\boldsymbol{\theta}$ doit donc vérifier :

$$\left(\nabla g_i\left(\boldsymbol{\theta}^*\right)\right)^T \delta\boldsymbol{\theta} = 0 \text{ pour } i = 1,..., p$$

En $\boldsymbol{\theta}^*$, toute variation admissible $\delta\boldsymbol{\theta}$ doit donc être orthogonale aux p vecteurs gradients des contraintes d'égalité ∇g_i. Contrairement au cas sans contraintes, la condition nécessaire d'optimalité $\left(\nabla J\left(\boldsymbol{\theta}^*\right)\right)^T \delta\boldsymbol{\theta} = 0$ n'implique plus que $\nabla J\left(\boldsymbol{\theta}^*\right) = 0$, mais seulement que $\delta\boldsymbol{\theta}$ soit orthogonal à $\nabla J\left(\boldsymbol{\theta}^*\right)$. On en déduit que le vecteur $\nabla J\left(\boldsymbol{\theta}^*\right)$ doit appartenir au sous-espace engendré par les p vecteurs $\nabla g_i\left(\boldsymbol{\theta}^*\right)$, soit :

$$\exists\left\{\lambda_i, i = 1,..., p\right\} \left| \nabla J\left(\boldsymbol{\theta}^*\right) + \sum_{i=1}^{p} \lambda_i \nabla g_i\left(\boldsymbol{\theta}^*\right) = \mathbf{0} \right.$$

Cette condition correspond à la condition nécessaire d'optimalité du problème sans contraintes suivant :

$$\min_{\boldsymbol{\theta},\boldsymbol{\lambda}} \mathrm{L}\left(\boldsymbol{\theta},\boldsymbol{\lambda}\right) = J\left(\boldsymbol{\theta}\right) + \boldsymbol{\lambda}^T \mathbf{g}\left(\boldsymbol{\theta}\right)$$

La fonction $\mathrm{L}\left(\boldsymbol{\theta},\boldsymbol{\lambda}\right)$ est appelée Lagrangien et les λ_i les paramètres de Lagrange. La condition nécessaire d'optimalité revient finalement à trouver les couples $\left(\boldsymbol{\theta}^*,\boldsymbol{\lambda}^*\right)$ vérifiant :

$$\begin{cases} \dfrac{\partial L}{\partial \boldsymbol{\theta}}\left(\boldsymbol{\theta}^*,\boldsymbol{\lambda}^*\right) = \mathbf{0} \\ \dfrac{\partial L}{\partial \boldsymbol{\lambda}}\left(\boldsymbol{\theta}^*,\boldsymbol{\lambda}^*\right) = \mathbf{g}\left(\boldsymbol{\theta}^*\right) = \mathbf{0} \end{cases}$$

Les *n* premières équations du système ci-dessus déterminent des solutions $\boldsymbol{\theta}(\boldsymbol{\lambda})$ fonction de $\boldsymbol{\lambda}$ et les *p* dernières déterminent $\boldsymbol{\lambda}^*$, et par suite $\boldsymbol{\theta}^* = \boldsymbol{\theta}(\boldsymbol{\lambda}^*)$.

Condition suffisante : un développement au deuxième ordre de la fonction de Lagrange introduite précédemment s'écrit, au voisinage d'un minimum $\boldsymbol{\theta}^*$:

$$L(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}, \boldsymbol{\lambda}) \geq L(\boldsymbol{\theta}^*, \boldsymbol{\lambda}) + \left(\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*, \boldsymbol{\lambda})\right)^T \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\lambda}}} \delta\boldsymbol{\theta} + \mathrm{o}\left(\delta\boldsymbol{\theta}^T \delta\boldsymbol{\theta}\right)$$

De plus, au point $\boldsymbol{\theta}^*$, nous avons :

$$\begin{cases} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*, \boldsymbol{\lambda}) = \mathbf{0} \\ J(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}) = L(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}, \boldsymbol{\lambda}) \end{cases}$$

Ainsi : $J(\boldsymbol{\theta}^* + \delta\boldsymbol{\theta}) \geq J(\boldsymbol{\theta}^*) + \frac{1}{2}\delta\boldsymbol{\theta}^T \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\lambda}}} \delta\boldsymbol{\theta}$

La quantité $\frac{1}{2}\delta\boldsymbol{\theta}^T \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\lambda}}} \delta\boldsymbol{\theta}$ doit donc être non-négative pour toute variation $\delta\boldsymbol{\theta}$ telle que $\left(\nabla g_i(\boldsymbol{\theta}^*)\right)^T \delta\boldsymbol{\theta} = 0$ pour $i = 1,..., p$. On peut montrer que cette condition est réalisée si les racines ω_i du polynôme suivant sont toutes positives ou nulles :

$$P(\omega) = \begin{vmatrix} \omega\mathbf{I} - \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\theta}}} & \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\lambda}}} \\ \mathbf{H}_{L_{\boldsymbol{\lambda},\boldsymbol{\theta}}} & \mathbf{H}_{L_{\boldsymbol{\lambda},\boldsymbol{\lambda}}} \end{vmatrix} = 0$$

3.1.3 Problème avec contraintes d'inégalité

Prenons à présent en considération les contraintes d'inégalité du problème (18). L'ensemble des contraintes admissibles est alors défini par :

$$h_i(\boldsymbol{\theta}) \leq 0 \text{ pour } i = 1,..., q$$

Comme les contraintes ci-dessus ne sont pas nécessairement simultanément saturées, il est possible d'avoir $q>n$. On peut alors se ramener à un problème avec contraintes d'égalité par introduction de variables supplémentaires ξ_i (i=1,…,q) vérifiant :

$$h_i(\boldsymbol{\theta}) + \xi_i^2 = 0 \text{ pour } i = 1,..., q$$

Condition nécessaire : en appliquant les résultats du paragraphe précédent, on montre donc que la condition nécessaire d'optimalité du problème initial (faisant intervenir des contraintes inégalités) correspond à celle du problème d'optimisation sans contraintes suivant :

$$\min_{\boldsymbol{\theta},\boldsymbol{\mu},\boldsymbol{\xi}} L(\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\xi}) = J(\boldsymbol{\theta}) + \sum_{i=1}^{q} \mu_i \left(h_i(\boldsymbol{\theta}) + \xi_i^2\right)$$

où $L(\boldsymbol{\theta},\boldsymbol{\mu},\boldsymbol{\xi})$ est la fonction de Lagrange, et les paramètres μ_i sont les paramètres de Lagrange (également appelés dans ce cas paramètres de Kuhn et Tucker [11]). La condition nécessaire d'optimalité s'écrit alors :

$$\begin{cases} \dfrac{\partial L(\boldsymbol{\theta},\boldsymbol{\mu},\boldsymbol{\xi})}{\partial \boldsymbol{\theta}} = \dfrac{\partial J}{\partial \boldsymbol{\theta}} + \sum_{i=1}^{q} \mu_i \dfrac{\partial h_i}{\partial \boldsymbol{\theta}} = 0 \\ \dfrac{\partial L(\boldsymbol{\theta},\boldsymbol{\mu},\boldsymbol{\xi})}{\partial \xi_i} = 2\mu_i \xi_i = 0 \text{ pour } i = 1,...,q \\ \dfrac{\partial L}{\partial \mu_i} = h_i(\boldsymbol{\theta}) + \xi_i^2 = 0 \text{ pour } i = 1,...,q \end{cases}$$

La dernière relation ré-exprime les contraintes. D'autre part, la première relation peut être obtenue en écrivant la condition de stationnarité du Lagrangien simplifié $L(\boldsymbol{\theta},\boldsymbol{\mu}) = J(\boldsymbol{\theta}) + \sum_{i=1}^{q} \mu_i h_i(\boldsymbol{\theta})$. Pour tenir compte de la deuxième relation, il suffit alors d'imposer μ_i tel que $\mu_i h_i(\boldsymbol{\theta}) = 0$, ce qui revient à prendre :

$$\mu_i = 0 \text{ si } h_i(\boldsymbol{\theta}) \neq 0 \text{ (contrainte non saturée)}$$
$$\mu_i \neq 0 \text{ si } h_i(\boldsymbol{\theta}) = 0 \text{ (contrainte saturée)}$$

Le théorème de Kuhn et Tucker (1951) [12] résume ces conditions nécessaires d'optimalité, dans le cas d'un problème d'optimisation avec contraintes d'égalité et d'inégalité :

> Une condition nécessaire pour que $\boldsymbol{\theta}^*$ soit un optimum local du problème d'optimisation (18) est qu'il existe des $\mu_i \geq 0$ et des λ_i tels que :
>
> $$\begin{cases} \nabla J(\boldsymbol{\theta}) + \sum_{i=1}^{q} \mu_i \nabla h_i(\boldsymbol{\theta}) + \sum_{i=1}^{p} \lambda_i \nabla g_i(\boldsymbol{\theta}) = 0 \\ \mu_i h_i(\boldsymbol{\theta}) = 0 \text{ pour } i = 1,...q \end{cases}$$

<u>*Condition suffisante*</u> *:* à partir des résultats précédents, on montre qu'une condition suffisante pour que $\boldsymbol{\theta}^*$ soit un minimum est que :

$$\mu_i \geq 0 \text{ pour } i = 1,...,q$$
$$\omega_j \geq 0 \quad \text{tq} \quad P(\omega_j) = \begin{vmatrix} \omega_j \mathbf{I} - \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\theta}}} & \mathbf{H}_{L_{\boldsymbol{\theta},\boldsymbol{\lambda}}} \\ \mathbf{H}_{L_{\boldsymbol{\lambda},\boldsymbol{\theta}}} & \mathbf{H}_{L_{\boldsymbol{\lambda},\boldsymbol{\lambda}}} \end{vmatrix} = 0$$

On peut par ailleurs montrer que si les fonctions J, h_i et g_i du problème (18) sont convexes, les conditions de Kuhn et Tucker sont nécessaires et suffisantes :

$\boldsymbol{\theta}^*$ est un optimum global de (18) si et seulement si il existe des $\mu_i^* \geq 0$ (i=1,…,q) et des λ_i^* (i=1, …,p) quelconques vérifiant :

$$\begin{cases} \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\right) = 0 \\ \mu_i^* h_i\left(\boldsymbol{\theta}^*\right) = 0 \text{ pour } i = 1,...,q \end{cases}$$

où $L\left(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right) = J\left(\boldsymbol{\theta}\right) + \sum_{i=1}^{p} \lambda_i g_i\left(\boldsymbol{\theta}\right) + \sum_{i=1}^{q} \mu_i h_i\left(\boldsymbol{\theta}\right)$

3.2 Le problème dual

Considérons à nouveau le problème d'optimisation (18), que nous appellerons désormais problème primal :

$$\min_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right)$$
$$\text{sous } \begin{cases} \mathbf{g}\left(\boldsymbol{\theta}\right) = 0 \\ \mathbf{h}\left(\boldsymbol{\theta}\right) \leq 0 \end{cases}$$

Nous avons introduit au paragraphe précédent une fonction de coût modifiée, appelée fonction de Lagrange, et définie par :

$$L\left(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right) = J\left(\boldsymbol{\theta}\right) + \boldsymbol{\lambda}^T \mathbf{g}\left(\boldsymbol{\theta}\right) + \boldsymbol{\mu}^T \mathbf{h}\left(\boldsymbol{\theta}\right)$$

Les caractéristiques de cette fonction de Lagrange vont à présent nous permettre de transformer le problème primal en un nouveau problème d'optimisation, appelé problème dual.

3.2.1 Conditions de point-selle

Par définition, $\left(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}\right)$ est un point-selle de $L\left(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right)$ s'il vérifie :

$$L\left(\bar{\boldsymbol{\theta}}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right) \leq L\left(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}\right) \leq L\left(\boldsymbol{\theta}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}\right) \quad \forall\, \boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}$$

On peut montrer [13] que si $\left(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}\right)$ est un point-selle de $L\left(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right)$, alors $\bar{\boldsymbol{\theta}}$ est un optimum global du problème d'optimisation (18). Ce résultat est très général et n'impose aucune hypothèse de convexité ou de différentiabilité des fonctions (pour certains problèmes, il peut néanmoins ne pas exister de point-selle).

De plus, dans le cas où les fonctions J, g_i et h_i sont convexes (et à condition que (18) admette une solution optimale $\boldsymbol{\theta}^*$), l'existence d'un point-selle en la solution $\left(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\right)$ est assurée.

Cette propriété permet donc de définir le point-selle en termes de minimisation et de maximisation sans contraintes :

$$L\left(\boldsymbol{\theta}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*\right)=J\left(\boldsymbol{\theta}^*\right)=\max_{\boldsymbol{\lambda},\boldsymbol{\mu}}\ \min_{\boldsymbol{\theta}}\ L\left(\boldsymbol{\theta},\boldsymbol{\lambda},\boldsymbol{\mu}\right) \tag{21}$$

3.2.2 Expression du problème dual

L'introduction de cette notion de point-selle, caractéristique de la fonction de Lagrange dans la plupart des problèmes d'optimisation, permet de transformer le problème d'optimisation primal. La recherche de la solution de ce problème peut en effet se ramener à la recherche du point-selle de la fonction de Lagrange.

Soit $L^D\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right)$ la fonction duale définie par : $L^D\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right)=\min_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta},\boldsymbol{\lambda},\boldsymbol{\mu}\right)$. La valeur correspondante du vecteur $\boldsymbol{\theta}$ est fonction des paramètres $\boldsymbol{\lambda}$ et $\boldsymbol{\mu}$: $\boldsymbol{\theta}_{\min}=l\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right)$, et cette solution n'existe a priori pas pour toutes les valeurs de $\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right)$. Soit $\Delta=\left\{\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right),\boldsymbol{\mu}\geq 0 \mid L^D\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right) \text{ existe}\right\}$. Le problème d'optimisation dual est alors :

$$\max_{(\boldsymbol{\lambda},\boldsymbol{\mu})\in\Delta} L^D\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right) \tag{22}$$

Du fait de l'existence d'un point-selle, la solution $\left(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*\right)$ du problème (22) est également solution du problème (18) :

$$\max_{(\boldsymbol{\lambda},\boldsymbol{\mu})\in\Delta} L^D\left(\boldsymbol{\lambda},\boldsymbol{\mu}\right)=J\left(\boldsymbol{\theta}^*\right)$$

La solution $\boldsymbol{\theta}^*$ est finalement donnée par :

$$\boldsymbol{\theta}^*=l\left(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*\right) \tag{23}$$

4 Conclusion

Les outils d'optimisation présentés dans ce chapitre vont nous permettre d'aborder efficacement la question de l'apprentissage de modèles de type réseau de neurones ou machines à vecteurs supports. Pour une optimisation paramétrique, nous retiendrons les méthodes quasi-newtoniennes du second ordre, et notamment les algorithmes de Levenberg-Marquardt et de BFGS, qui présentent des propriétés de convergence améliorées par comparaison avec les méthodes de gradient (du premier ordre).

Le formalisme adopté pour transformer un problème d'optimisation sous contraintes en un problème dual sera employé dans la partie consacrée aux méthodes de noyau. Il sera alors

possible d'obtenir un problème d'optimisation dual dont la résolution est simplifiée, auquel sera associée une solution dont l'existence et l'expression seront justifiées par les considérations développées dans ce chapitre.

Bibliographie

[1] R. Fletcher. *Practical methods of optimization.* John Wiley & Sons, New-York, 1987.

[2] A. Björck. *Numerical methods for least squares problems.* SIAM, Philadelphia, 1996.

[3] D. Bertsekas. *Nonlinear programming.* Athena Scientific, Belmont, MA, 1999.

[4] R. Battiti. *First and second order methods for learning : between steepest descent methods and Newton's method.* Neural Computation, Vol. 4, No. 2, pp. 141-166, 1992.

[5] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[6] R.Fletcher. *A New Approach to Variable Metric Algorithms*. Computer Journal, Vol.13, pp. 317-322, 1970.

[7] J.E. Dennis, J.J. More. *Quasi-Newton methods, motivation and theory.* SIAM Review, Vol. 19, No. 1, pp. 46-89, 1977.

[8] J.J. Moré. *The Levenberg-Marquardt Algorithm: Implementation and Theory.* In Numerical Analysis, Lecture Notes in Mathematics 630, Springer Verlag, pp. 105-116, 1977.

[9] J.B. Rosen. *The gradient projection method for nonlinear programming, part 1: linear constraints.* SIAM Journal, Vol.8, pp. 181-217, 1960.

[10] A.V. Fiacco, G.P. McCormick. *Nonlinear programming: sequential unconstrained minimization techniques.* John Wiley, New-York, 1968.

[11] H.W. Kuhn, A.W. Tucker. *Linear inequalities and related systems*. Princeton University Press, Princeton, 1956.

[12] H.W. Kuhn, A.W. Tucker. *Nonlinear programming*. In Proc. of the 2nd Berkeley Symp. on Mathematical Statistics and Probability, University of California Press, Berkeley, pp. 481-492, 1951.

[13] M. Minoux. *Programmation mathématique : théorie et algorithmes.* Dunod, 1983.

Les réseaux de neurones

1 Introduction

Les réseaux de neurones constituent une famille de fonctions non linéaires paramétrées, utilisées dans de nombreux domaines (physique, chimie, biologie, finance, etc.), notamment pour la modélisation de processus et la synthèse de lois de commandes, deux problèmes que nous avons abordés dans le cadre de ce travail.

Les réseaux de neurones dynamiques peuvent être conçus soit en temps continu, soit en temps discret. Dans la mesure où la grande majorité des applications actuelles, notamment dans le secteur automobile, sont réalisées par des calculateurs, c'est la deuxième approche que nous avons adoptée. De façon générale, nous supposerons que les processus auxquels nous nous intéressons peuvent être décrits :

- dans le cas du choix d'une représentation d'état, par l'hypothèse suivante :

$$\begin{cases} \mathbf{x}^p(k+1) = \mathbf{\Phi}\left(\mathbf{x}^p(k), \mathbf{u}(k), \mathbf{w}(\mathbf{k})\right) \\ \mathbf{y}^p(k+1) = \mathbf{\Psi}\left(\mathbf{x}^p(k+1), \mathbf{w}(k+1)\right) \end{cases} \quad (1)$$

  où $\mathbf{x}^p(k+1)$ est le vecteur d'état à l'instant $k+1$, $\mathbf{u}(k)$ est le vecteur des entrées de commande, où $\mathbf{y}^p(k+1)$ est le vecteur des sorties du processus, et où $\mathbf{w}(k)$ est une réalisation d'un vecteur aléatoire ; $\mathbf{\Phi}$ et $\mathbf{\Psi}$ sont deux fonctions non linéaires.

- dans le cas du choix d'une représentation entrée-sortie, par l'hypothèse suivante :

$$\mathbf{y}^p(k+1) = \mathbf{\Phi}\left(\mathbf{y}^p(k), ..., \mathbf{y}^p(k-N_s), \mathbf{u}(k), ..., \mathbf{u}(k-N_e), \mathbf{w}(\mathbf{k}), \mathbf{w}(k-1), \mathbf{w}(k-N_w)\right) \quad (2)$$

  où $\mathbf{\Phi}$ est une fonction non linéaire, et où N_s, N_e et N_w sont trois entiers

Si l'on s'intéresse à la modélisation de ce processus par un réseau de neurones , l'objectif est alors d'ajuster, par apprentissage, les paramètres de fonctions permettant d'approcher $\mathbf{\Phi}$ et $\mathbf{\Psi}$ avec une précision suffisante. Nous verrons par la suite que, sous des hypothèses que nous indiquerons plus loin, les réseaux de neurones sont des approximateurs universels, c'est-à-dire qu'ils peuvent approcher uniformément n'importe quelle fonction continue et bornée avec une précision arbitraire.

Dans le cas de la commande de processus, l'objectif est de synthétiser une loi de commande permettant de satisfaire le critère de performance recherché, soit de manière directe (un réseau de neurones réalise la fonction de commande), soit de manière indirecte (un réseau de neurones intervient dans la conception de la loi de commande).

Dans un premier temps, nous rappèlerons les définitions et notations de base relatives aux réseaux de neurones, avant de décrire la méthodologie d'ajustement des paramètres de ces réseaux (phase d'apprentissage). Nous poursuivrons en exposant les résultats de modélisation portant sur deux processus dynamiques industriels : l'évolution de la température en un point particulier de la ligne d'échappement, et les débits de différents polluants réglementés. Pour conclure, nous présenterons un système de commande optimale en boucle ouverte, fondé sur un modèle neuronal récurrent, destiné à réduire les variations rapides de la sortie du processus à commander. L'élaboration de ce système de commande nous a conduits à concevoir une méthodologie générale pour le calcul exact de la matrice Hessienne d'une fonction de coût particulière, dans le contexte de modèles récurrents.

2 Définitions et notations

2.1 Réseau de neurones formels

Un réseau de neurones formels est une combinaison de fonctions élémentaires appelées « neurones formels », ou simplement neurones. Chacun des neurones formels réalise, à chaque instant k, une fonction non linéaire paramétrée de ses variables, qui, au sein du réseau, peuvent être soit les sorties d'autres neurones, soit des variables exogènes. Une représentation graphique d'un réseau de neurones est donnée sur la Figure 1.

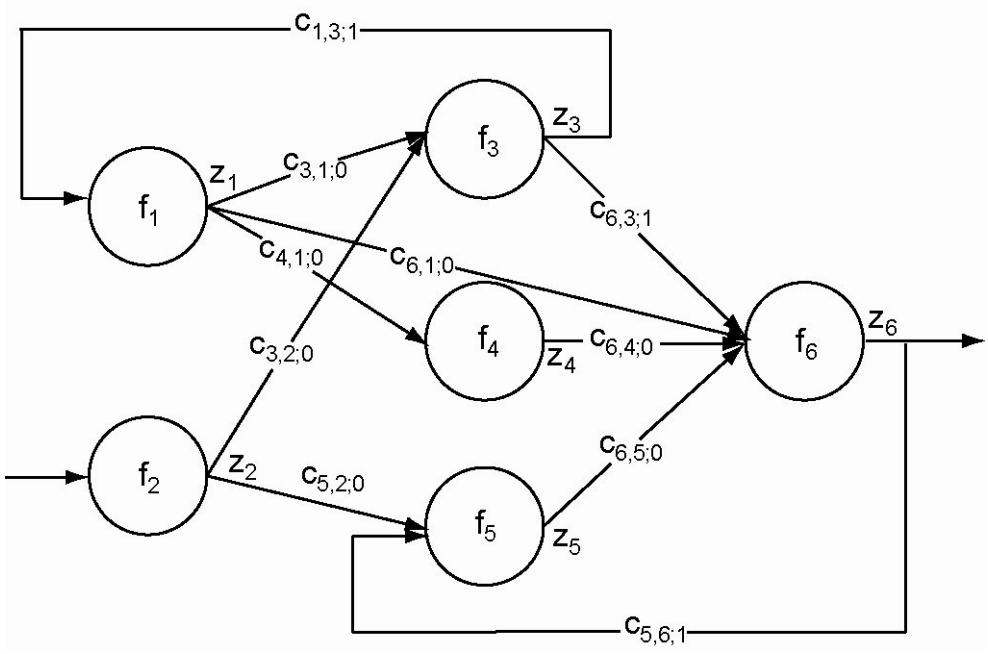


Figure 1 : réseau de neurones formels représenté sous forme d'un graphe orienté. À chaque arête, reliant la sortie z_i d'un neurone i à l'entrée d'un neurone j, est associé un paramètre $c_{j,i;\tau}$.

La fonction réalisée par le neurone formel *i* se décompose en deux opérations :

- calcul du potentiel du neurone : $v_i(k) = \sum_{j \in P(i)} \sum_{\tau=0}^{\tau_{ij}} c_{i,j;\tau} z_j(k-\tau)$, où $P(i)$ représente l'ensemble des neurones parents du neurone *i* (c'est-à-dire l'ensemble des neurones dont la sortie z_j est présente en entrée du neurone *i*), où $c_{i,j;\tau}$ est le paramètre associé à la connexion du neurone *j* vers le neurone *i*, et où τ_{ij} représente le retard maximal avec lequel la variable de sortie du neurone *j* intervient à l'entrée du neurone *i*.
- calcul de la sortie du neurone : $z_i(k) = f_i(v_i(k))$, où f_i est la fonction d'activation du neurone *i* (fonction qui peut également être paramétrée).

On distingue deux structures de réseau, en fonction du graphe de leurs connexions, c'est-à-dire du graphe dont les nœuds sont les neurones et les arêtes les « connexions » entre ceux-ci :

- *les réseaux de neurones statiques (ou acycliques, ou non bouclés)* : le graphe des connexions de ces réseaux est acyclique. Ils réalisent une fonction algébrique de leurs entrées. Dans le cas de signaux à temps discret, ces réseaux constituent donc des filtres transverses non linéaires.
- *les réseaux de neurones dynamiques (ou récurrents, ou bouclés)* : le graphe des connexions de ces réseaux est cyclique. Ces réseaux sont décrits par un système d'équations aux différences ; ils constituent des filtres récursifs non linéaires.

2.2 Propriété d'approximation universelle et forme canonique

Dans ce paragraphe, nous décrivons les structures de réseaux que nous avons choisi de mettre en œuvre.

2.2.1 Le Perceptron à une couche cachée

Aussi bien en modélisation qu'en commande de processus, l'objectif est d'approcher, en particulier à l'aide d'un réseau de neurones formels, une fonction à valeurs réelles que l'on suppose continue. Un résultat fondamental [1][2] prouve la possibilité d'approcher toute fonction continue sur un domaine borné par un réseau de neurones particulier appelé Perceptron. Graphiquement, ce réseau présente une structure organisée en couches (cf. Figure 2) : une « couche cachée » dont chaque neurone calcule une fonction non linéaire des variables du réseau, et un « neurone linéaire de sortie » qui calcule une combinaison linéaire des grandeurs calculées par les neurones « cachés ». Les hypothèses de ce théorème d'approximation universelle imposent pour la couche cachée des fonctions d'activation

continues, monotones croissantes, et bornées (nous utiliserons habituellement la fonction sigmoïdale tangente hyperbolique *tanh*). Notons la présence de connexions entre la première entrée du réseau (supposée constante et de valeur 1) et la couche de sortie : celles-ci ont pour but d'autoriser l'ajout d'un terme constant, appelé biais, aux sorties du réseau, pour tenir compte du fait que les valeurs prises par celles-ci ont a priori une valeur moyenne non nulle.

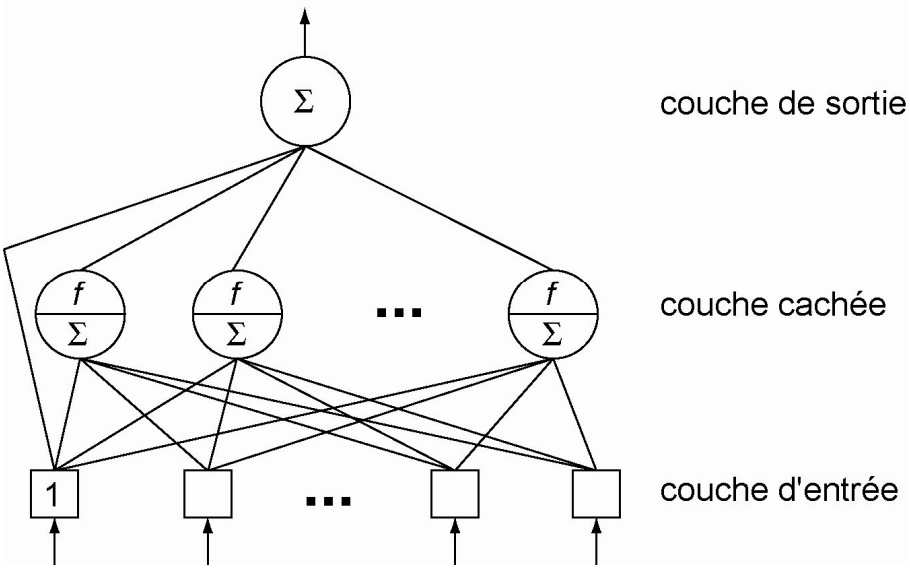


Figure 2 : structure d'un Perceptron à une couche cachée, vérifiant la propriété d'approximation universelle.

D'un point de vue pratique, cette propriété réduit grandement la recherche de la structure adaptée à la tache que l'on souhaite réaliser : une fois les variables du réseau déterminées, la complexité du modèle est simplement contrôlée par le nombre de neurones dans la couche cachée. Il n'est toutefois pas garanti que cette structure soit optimale, notamment du point de vue du nombre de paramètres ou de la capacité du modèle à généraliser : des connaissances physiques sur le processus à modéliser peuvent suggérer une structure particulière, mieux adaptée au problème posé.

2.2.2 Forme canonique des réseaux récurrents

Il a été montré [3] que tout réseau récurrent peut être mis sous une forme particulière, appelée forme canonique, qui est la représentation d'état minimale de la fonction réalisée par ce réseau. Cette forme canonique est constituée d'un graphe acyclique, et de connexions à retard unité reliant certaines sorties de ce graphe à ses entrées. La fonction réalisée par un réseau de neurones ayant cette structure particulière est décrite par les équations aux différences suivantes :

$$\begin{cases} \mathbf{x}(k+1) = \boldsymbol{\Phi}\left(\mathbf{x}(k), \mathbf{u}(k+1)\right) \\ \mathbf{y}(k+1) = \boldsymbol{\Psi}\left(\mathbf{x}(k+1), \mathbf{u}(k+1)\right) \end{cases} \tag{3}$$

où $\mathbf{x}(k)$ est le vecteur d'état à l'instant k, $\mathbf{u}(k)$ est le vecteur des variables de commande exogènes, $\mathbf{y}(k)$ le vecteur des sorties, $\boldsymbol{\Psi}$ et $\boldsymbol{\Phi}$ sont deux fonctions qui dépendent de la structure de la partie acyclique du réseau.

En choisissant pour cette partie acyclique du réseau une structure de type Perceptron, la propriété d'approximation universelle introduite au paragraphe précédent garantit la possibilité d'approcher avec la précision souhaitée tout processus qui peut être décrit, sous une représentation d'état, par des équations aux différences du type des équations (3). Ceci inclut une très grande classe de modèles dynamiques non linéaires déterministes.

Nous utiliserons cette structure particulière, représentée sur la Figure 3, pour la modélisation de processus dynamiques.

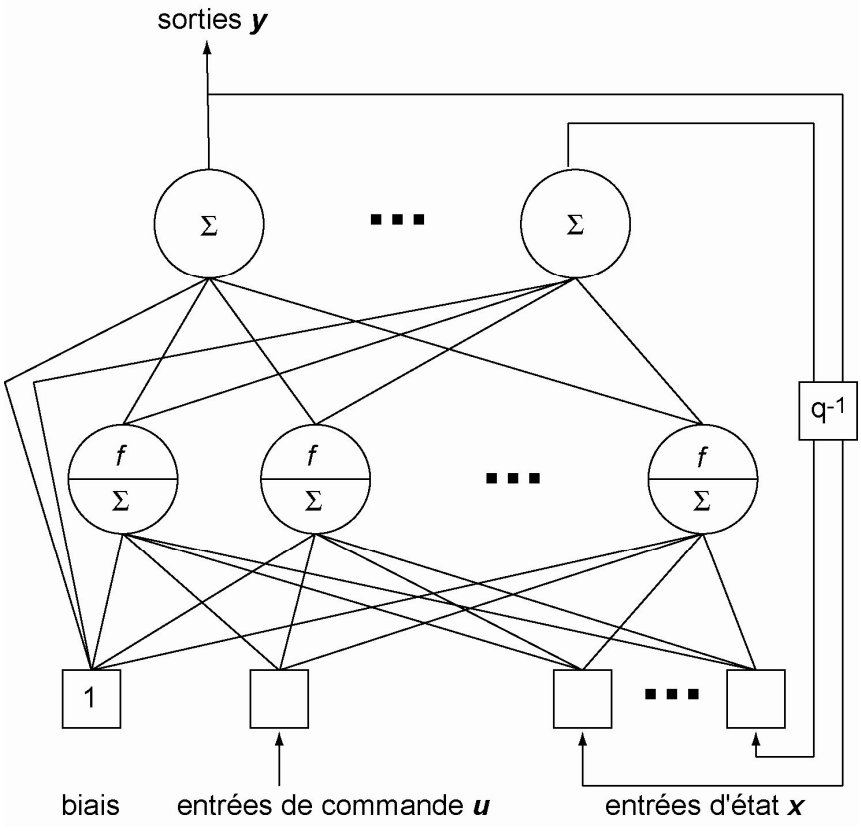


Figure 3 : forme canonique d'un réseau de neurones récurrent, dont la partie acyclique est de type Perceptron à une couche cachée.

3 Estimation des paramètres - Apprentissage

Après avoir introduit les notions de base concernant les réseaux de neurones, et la question du choix de leur structure, nous allons présenter dans cette partie les méthodes algorithmiques permettant d'estimer les paramètres de ces réseaux (phase dite d'apprentissage). Ces algorithmes font appel à des notions abordées dans le chapitre 2 de ce mémoire; nous décrirons les calculs spécifiques nécessaires à leur mise en œuvre, en distinguant les réseaux statiques des réseaux récurrents, et en nous limitant au cas de structures de type Perceptron à une couche cachée.

Rappelons que l'objectif de cette phase d'apprentissage est de déterminer, sur la base d'une séquence de données $D = \left\{ \mathbf{x}_k, y_k^p \right\}_{1 \le k \le N}$, la valeur des paramètres $\boldsymbol{\theta}$ du réseau de neurones permettant de minimiser la fonction de coût des moindres carrés $J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{k=1}^{N} \left(y_k^p - y_k(\boldsymbol{\theta}) \right)^2$, où :

- y_k^p est la sortie mesurée du processus à l'instant k,
- y_k est la sortie estimée par le réseau de neurones (que nous supposerons pour simplifier qu'il ne comporte qu'une seule sortie) au même instant,
- $\mathbf{x}_k = \left[x_k^1, x_k^2, ..., x_k^{N_i} \right]^T$ est le vecteur des N_i entrées du modèle,
- le vecteur $\boldsymbol{\theta}$ comprend l'ensemble des paramètres du réseau (les connexions entre couches c_{ij}, ainsi que les biais de sortie).

Comme la fonction réalisée par le réseau de neurones est généralement non linéaire vis-à-vis des paramètres, il est nécessaire de mettre en œuvre des algorithmes d'optimisation itératifs, fondés sur le calcul, à chaque itération, du gradient de la fonction de coût.

3.1 Apprentissage de réseaux statiques

Dans le cas d'un réseau statique, ayant une structure de Perceptron à une couche cachée (cf. Figure 1), la fonction réalisée par le réseau à l'instant discret k s'écrit :

$$y_k = \sum_{i=1}^{N_{cc}} c_i \Phi \left(\sum_{j=1}^{N_i} c_{i,j} x_k^j \right) + b \qquad (4)$$

où N_{cc} représente le nombre de neurones dans la couche cachée, où c_i est le paramètre associé à la connexion du i-ème neurone de la couche cachée vers la sortie, où Φ est la fonction sigmoïde réalisée par les neurones de la couche cachée, où b est le biais, et où $c_{i,j}$ est le

paramètre associé à la connexion de la *j*-ième entrée (parmi les N_i) vers le *i*-ème neurone de la couche cachée.

Pour une fonction de ce type (4), nous allons présenter la manière de calculer le gradient de la fonction de coût qui s'écrit :

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = -\sum_{k=1}^{N} \left(y_k^p - y_k \right) \frac{\partial y_k}{\partial \boldsymbol{\theta}} \tag{5}$$

La méthode dite de rétropropagation [4] (que nous présenterons en détail dans le cas récurrent au prochain paragraphe) est largement utilisée pour calculer le gradient de cette fonction de coût. Elle consiste à effectuer le calcul des dérivées partielles de la fonction neuronale vis-à-vis des paramètres du réseau en partant de la couche de sortie, puis de se servir de ces quantités pour calculer les dérivées partielles dans la couche cachée, avant de remonter vers la couche d'entrée. Cette méthode présente l'avantage d'être économique en termes de nombre d'opérations à effectuer.

Dans certaines situations, il est néanmoins préférable de calculer le gradient de la fonction de coût dans le sens direct. C'est le cas pour l'apprentissage de réseaux de neurones récurrents, dont l'apprentissage est effectué par l'algorithme de Levenberg-Marquardt, associé à une fonction de coût quadratique : le nombre d'opérations à effectuer est alors moindre dans le sens direct que par rétropropagation. Bien que cette situation ne s'applique pas au cas des réseaux statiques, nous allons présenter le calcul direct du gradient, dont l'extension au cas récurrent ne pose pas de problèmes particuliers.

Pour calculer chacune des composantes du gradient de la fonction de coût, on est ramené au calcul de la dérivée de la sortie du réseau par rapport au paramètre considéré (relation (5)). On distingue alors :

- le biais de sortie *b* :

$$\frac{\partial y_k}{\partial b} = 1 \tag{6}$$

- les paramètres de connexions couche cachée / sortie c_i :

$$\frac{\partial y_k}{\partial c_i} = \Phi\left(\sum_{j=1}^{N_i} c_{i,j} x_k^j \right) \text{ pour } i = 1, \ldots, N_{cc} \tag{7}$$

- les paramètres de connexions couche d'entrée / couche cachée $c_{i,j}$:

$$\frac{\partial y_k}{\partial c_{i,j}} = c_i x_k^j \Phi'\left(\sum_{n=1}^{N_i} c_{i,n} x_k^n \right) \tag{8}$$

où $\Phi^{'}$ est la dérivée de la fonction sigmoïde utilisée. Rappelons que pour la fonction tangente hyperbolique : $\tanh^{'}(x) = 1 - (\tanh(x))^2$

Après avoir ainsi calculé chacune des composantes du vecteur gradient, il est possible de mettre en œuvre des algorithmes d'optimisation itératifs, tels que celui de BFGS ou de Levenberg-Marquardt.

3.2 Apprentissage de réseaux récurrents

Nous décrivons à présent l'apprentissage de réseaux de neurones récurrents. On considère les réseaux correspondant à une représentation d'état et mis sous forme canonique (comme celle décrite par la Figure 3). On présente le cas d'un modèle mono-sortie. La fonction réalisée par un réseau de ce type peut s'écrire :

$$y_k = \sum_{i=1}^{N_{cc}} c_i \Phi\left(\sum_{j=1}^{N_e+N_s} c_{i,j} x_k^j \right) + b \qquad (9)$$

où N_e représente le nombre d'entrées exogènes et où N_s représente le nombre d'entrées d'états. Le caractère récurrent du réseau se traduit par le fait que le vecteur des entrées du modèle $\mathbf{x}_k = \left[x_k^1, x_k^2, ..., x_k^{N_i} \right]^T$ (où N_i=N_e+N_s) comprend les N_s sorties d'état à l'instant précédent.

Deux situations sont possibles :

- *les variables d'état du processus que l'on souhaite modéliser sont mesurables* : on est alors ramené au problème de l'apprentissage d'un réseau à plusieurs sorties. Chacune de ces grandeurs, qu'il s'agisse de la sortie du modèle ou des différentes variables d'états, doit être prise en considération dans une fonction de coût généralisée, qui est la somme des fonctions de coût des moindres carrés associées à ces différentes grandeurs, et qui s'écrit :

$$J_{multi}(\theta) = \frac{1}{2} \sum_{k=1}^{N} \left(y_k^p - y_k(\theta) \right)^2 + \frac{1}{2} \sum_{m=N_e+1}^{N_e+N_s} \sum_{k=1}^{N} \left(x_k^{m,p} - x_k^m(\theta) \right)^2$$

  où $x_k^{m,p}$ est la mesure de la m-ième variable d'état à l'instant discret k, dont l'estimation est x_k^m. Le vecteur θ représente l'ensemble des paramètres du réseau.

- *les variables d'état du processus que l'on souhaite modéliser sont non mesurables* : il est possible dans ce cas de laisser les variables d'état du réseau de neurones évoluer librement, c'est-à-dire sans leur imposer de valeurs désirées. Les valeurs des variables

d'état perdent alors toute signification physique, au sens où elles ne reproduisent a priori le comportement d'aucune des variables d'états physiques du système. En revanche, elles assurent la présence de récurrences au sein du réseau, pour la prise en considération des phénomènes dynamiques. Seule l'initialisation de ces variables d'état, en début d'apprentissage, a une influence sur les valeurs que prendront ces variables d'état aux instants suivants. On est alors ramené au problème de l'apprentissage d'un réseau récurrent à une seule sortie.

Comme indiqué dans le Chapitre 1 de ce mémoire, la méthode d'apprentissage d'un modèle neuronal récurrent comme celui décrit par l'équation (9) dépend de l'hypothèse concernant la manière selon laquelle le bruit affecte le processus, et du type de prédicteur que l'on souhaite obtenir.

Deux types d'apprentissage sont envisageables :

- *apprentissage dirigé (teacher-forced* [5]*)*: les variables d'états du réseau sont fixées à leurs valeurs mesurées tout au long de la séquence, quelles que soient leurs valeurs estimées par le modèle. Ce type d'apprentissage ne peut naturellement être mis en œuvre qu'à condition que les états soient mesurables. En revanche, il peut être utilisé pour un réseau récurrent décrit par une représentation entrée-sortie, pour lequel les variables d'état sont les sorties retardées, mesurables par définition. Ce type d'apprentissage est bien adapté à l'élaboration de prédicteurs non bouclés, permettant de prédire la sortie à l'instant à venir connaissant les mesures de la sortie et des variables d'état à l'instant courant (ou les mesures de la sortie aux instants courant et précédents dans le cas d'une représentation entrée-sortie).
- *apprentissage semi-dirigé* : lors d'un tel apprentissage, le caractère récurrent du réseau de neurones est pris en considération durant la phase d'apprentissage. Au lieu de fixer les variables d'état du réseau à leurs valeurs mesurées, on leur attribue les valeurs estimées par le modèle à l'instant précédent (qui dépendent des paramètres du réseau). Comme nous l'avons vus dans le chapitre 1, cette méthode d'apprentissage est optimale dans le cas où le bruit agissant sur le processus est un bruit de sortie.

3.2.1 Apprentissage dirigé / prédicteur non bouclé

Pour réaliser un apprentissage de ce type, il faut disposer des mesures des variables d'état du modèle. Il convient alors de construire une séquence d'apprentissage de la manière suivante :

le vecteur des variables du réseau à l'instant discret k est constitué des N_e entrées exogènes, ainsi que des mesures des N_s variables d'états : $\mathbf{x}_k = \left[x_k^1, \ldots, x_k^{N_e}, x_k^{1,p}, \ldots, x_k^{N_s,p} \right]^T$.

Notons que, dans le cas d'une représentation entrée-sortie, le vecteur des variables du réseau fait intervenir les mesures retardées de la sortie ; il s'écrit alors $\mathbf{x}_k = \left[x_k^1, \ldots, x_k^{N_e}, y_{k-1}^p, \ldots, y_{k-m}^p \right]^T$.

Une fois construite cette séquence d'apprentissage $\left\{ \mathbf{x}_k, y_k^p \right\}_{1 \le k \le N}$, la mise en œuvre d'algorithmes d'optimisation itératifs se fait de la même manière que dans le cas d'un réseau statique, notamment en ce qui concerne le calcul du gradient de la fonction de coût.

3.2.2 Apprentissage semi-dirigé / prédicteur bouclé

Lors d'un apprentissage semi-dirigé, l'objectif est d'obtenir le prédicteur optimal dans l'hypothèse selon laquelle le bruit qui affecte le processus est un bruit additif sur la grandeur de sortie. À cet effet, on doit prendre en considération le caractère récurrent du modèle, en affectant à ses variables d'état, à l'instant k, leurs valeurs estimées à l'instant précédent. Du fait que les algorithmes d'optimisation utilisés sont nécessairement itératifs, le vecteur des variables du réseau évolue au cours de l'apprentissage.

Il est commode, dans cette situation, de faire subir au modèle neuronal récurrent un « dépliement temporel » (cf. Figure 4).

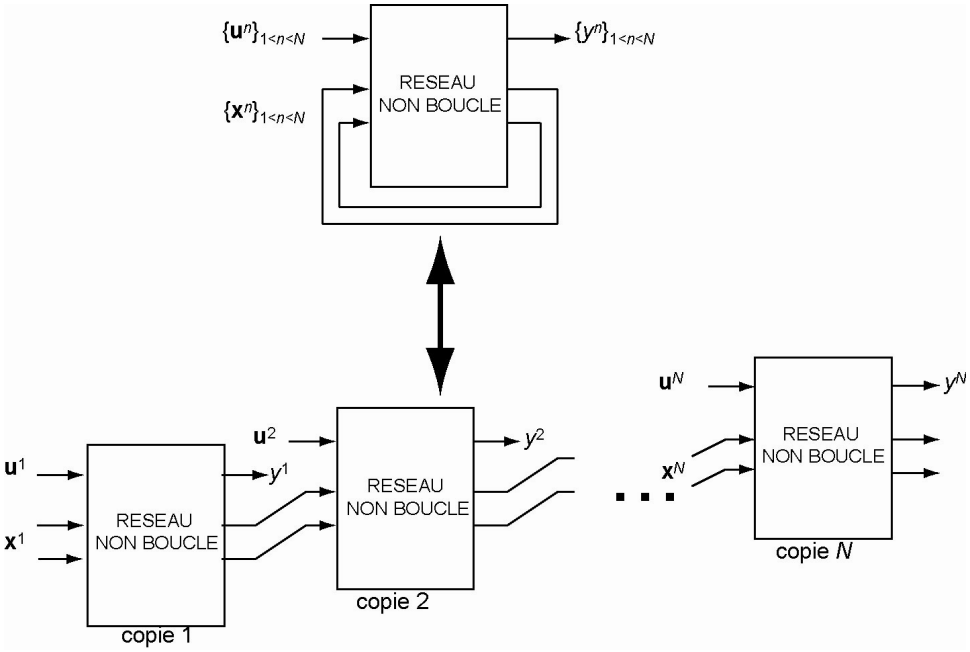


Figure 4 : dépliement temporel d'un réseau récurrent

Sur l'ensemble de la séquence de données de taille N, le réseau récurrent est remplacé par une série de N réseaux acycliques, appelés copies, ayant chacun les mêmes paramètres, et connectés entre eux via les sorties et entrées d'état. Il apparaît ainsi clairement que les valeurs prises à un instant k par les variables d'état, qui dépendent elles-mêmes des paramètres du réseau, ont une influence sur toutes les valeurs ultérieures de la sortie du réseau. Cette caractéristique doit donc être prise en considération durant l'apprentissage, notamment en ce qui concerne le calcul du gradient de la fonction de coût.

En se référant aux Figure 3 et Figure 4, nous adopterons les notations suivantes :

- *numérotation des neurones* : à chaque neurone de la couche d'entrée est attribué un numéro compris entre 1 et $N_i = N_e + N_s$. Les N_e premiers correspondant aux entrées exogènes, et les N_s suivant aux entrées d'états. Les neurones de la couche cachée porteront des numéros variant de $(N_e + N_s + 1)$ à $(N_e + N_s + N_{cc})$, et ceux de la couche de sortie des numéros compris entre $(N_e + N_s + N_{cc} + 1)$ à $(N_e + 2N_s + N_{cc})$. La sortie du modèle sera affectée de l'indice $(N_e + N_s + N_{cc} + 1)$; si la grandeur de sortie fait partie des variables d'état, cette sortie sera donc identique à l'une des N_S sorties d'état.
- *notation des sorties de chaque neurone* : nous désignerons par x_k^n la sortie du neurone indexé par n (variant de 1 à $N_e + 2N_s + N_{cc}$), dans la copie k, c'est-à-dire à l'instant discret k. De plus, l'activité des neurones de la couche cachée sera notée $v_k^n = \sum_{i=1}^{N_e+N_s} c_{n,i} x_k^i$ (pour n variant de $N_e + N_s + 1$ à $N_e + N_s + N_{cc}$), de telle sorte que $x_k^n = \Phi\left(v_k^n\right)$, où Φ est généralement la fonction tangente hyperbolique.

Rappelons que la fonction de coût s'écrit :

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{k=1}^{N}\left(y_k^p - y_k(\boldsymbol{\theta})\right)^2 \equiv \frac{1}{2}\sum_{k=1}^{N}\left(e_k\right)^2 \equiv \frac{1}{2}\sum_{k=1}^{N}\left(y_k^p - x_k^{N_e+N_s+N_{cc}+1}(\boldsymbol{\theta})\right)^2 \qquad (10)$$

Le calcul du gradient de cette fonction de coût dans le cas récurrent peut se calculer par rétropropagation [6], en deux étapes : dans un premier temps, on calcule la dérivée de la fonction de coût par rapport aux sorties et entrées d'état du réseau, puis, connaissant ces quantités, on calcule sa dérivée par rapport aux paramètres du réseau. Bien que dans chacune des copies du dépliement temporel, le vecteur $\boldsymbol{\theta}$ ait la même valeur, il est impératif de distinguer

- le gradient de la fonction de coût par rapport à ce vecteur $\boldsymbol{\theta}$ sur l'ensemble des copies,

- le gradient de la fonction de coût par rapport à $\boldsymbol{\theta}^{(n)}$ dans une copie n donnée. En effet, pour deux copies d'indice n et n' différents, les vecteurs $\frac{\partial J}{\partial \boldsymbol{\theta}^{(n)}}$ et $\frac{\partial J}{\partial \boldsymbol{\theta}^{(n')}}$ ne prennent pas la même valeur, bien que $\boldsymbol{\theta}^{(n)}$ et $\boldsymbol{\theta}^{(n')}$ soient égaux. Le gradient total peut ainsi s'écrire :

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = \sum_{n=1}^{N} \frac{\partial J}{\partial \boldsymbol{\theta}^{(n)}}$$

<u>Calcul de la dérivée de la fonction de coût par rapport aux sorties et entrées d'état</u> :

Il convient, pour effectuer ce calcul, de distinguer trois cas, suivant que l'on s'intéresse à la dernière copie du dépliement temporel, à la première, ou à une copie intermédiaire.

<u>Pour la copie N</u> :

Pour la sortie ($k = N_e+N_s+N_{cc}+1$):

$$\frac{\partial J}{\partial x_N^k} = \frac{\partial J}{\partial y_N} = -\left(y_N^p - y_N\left(\boldsymbol{\theta}\right)\right) = -e_N$$

Pour les autres sorties d'état ($k = N_e+N_s+N_{cc}+2,\ \ldots,\ N_e+2N_s+N_{cc}$):

$$\frac{\partial J}{\partial x_N^k} = 0$$

Pour les entrées d'état ($k = N_e+1,\ \ldots,\ N_e+N_s$):

$$\frac{\partial J}{\partial x_N^k} = \frac{\partial J}{\partial y_N}\frac{\partial y_N}{\partial x_N^k} = -e_N \sum_{j=N_e+N_s+1}^{N_e+N_s+N_{cc}} \left(c_{\alpha,j}\Phi'\left(v_N^j\right)c_{j,k}\right)$$

où $\alpha = N_e+N_s+N_{cc}+1$ et où Φ' est la dérivée de la fonction d'activation des neurones de la couche cachée.

<u>Pour les copies intermédiaires ($n = N$-1,…,2)</u> :

Pour la sortie ($k = N_e+N_s+N_{cc}+1$):

$$\frac{\partial J}{\partial x_n^k} = \frac{\partial J}{\partial y_n} = -e_n + \frac{\partial J}{\partial x_{n+1}^{k-N_s-N_{cc}}}$$

Pour les autres sorties d'état ($k = N_e+N_s+N_{cc}+2,\ \ldots,\ N_e+2N_s+N_{cc}$):

$$\frac{\partial J}{\partial x_n^k} = \frac{\partial J}{\partial x_{n+1}^{k-N_s-N_{cc}}}$$

Pour les entrées d'état ($k = N_e+1,\ \ldots,\ N_e+N_s$):

$$\frac{\partial J}{\partial x_n^k} = \sum_{i=N_e+N_s+N_{cc}+1}^{N_e+2N_s+N_{cc}} \frac{\partial J}{\partial x_n^i}\frac{\partial x_n^i}{\partial x_n^k} = \sum_{j=N_e+N_s+1}^{N_e+N_s+N_{cc}} \Phi'\left(v_n^j\right)c_{j,k}\left[\sum_{i=N_e+N_s+N_{cc}+1}^{N_e+2N_s+N_{cc}} \frac{\partial J}{\partial x_n^i}c_{i,j}\right]$$

Pour la copie n=1 :

Pour la sortie ($k = N_e+N_s+N_{cc}+1$):

$$\frac{\partial J}{\partial x_1^k} = -e_1 + \frac{\partial J}{\partial x_2^{k-N_s-N_{cc}}}$$

Pour les autres sorties d'état ($k = N_e+N_s+N_{cc}+2, \ldots, N_e+2N_s+N_{cc}$):

$$\frac{\partial J}{\partial x_1^k} = \frac{\partial J}{\partial x_2^{k-N_s-N_{cc}}}$$

Pour les entrées d'état ($k = N_e+1, \ldots, N_e+N_s$):

le calcul de $\frac{\partial J}{\partial x_1^k}$ est inutile

Calcul de la dérivée de la fonction de coût par rapport aux paramètres du réseau :

Nous pouvons à présent exprimer la dérivée de la fonction de coût par rapport à chacun des paramètres du réseau, en distinguant le rôle de ces paramètres.

Paramètres qui pondèrent les sorties des neurones cachés dans le calcul des potentiels des neurones de sortie du modèle :

Il s'agit donc des paramètres $c_{k,j}$ pour des valeurs de $k = N_e+N_s+N_{cc}+1, \ldots, N_e+2N_s+N_{cc}$ et $j= N_e+N_s+1,\ldots, N_e+N_s+N_{cc}$. On peut écrire :

$$\frac{\partial J}{\partial c_{k,j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \frac{\partial x_n^k}{\partial c_{k,j}^{(n)}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} x_n^j \qquad (11)$$

Paramètres qui pondèrent les variables (exogènes ou d'état) dans le calcul des potentiels des neurones cachés :

Il s'agit donc des paramètres $c_{k,j}$ pour $k = N_e+N_s+1,\ldots, N_e+N_s+N_{cc}$ et $j = 1,\ldots,N_e+N_s$. On peut écrire :

$$\frac{\partial J}{\partial c_{k,j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \frac{\partial x_n^k}{\partial c_{k,j}^{(n)}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \Phi'\left(v_n^k\right) x_n^j = \sum_{n=1}^{N} \Phi'\left(v_n^k\right) x_n^j \left[\sum_{i=N_e+N_s+N_{cc}+1}^{N_e+2N_s+N_{cc}} \frac{\partial J}{\partial x_n^i} c_{i,k}^{(n)} \right] \qquad (12)$$

Pour les biais de la couche de sortie :

Il s'agit des paramètres $c_{k,1}$ pour $k = N_e+N_s+N_{cc}+1,\ldots, N_e+2N_s+N_{cc}$. On peut écrire :

$$\frac{\partial J}{\partial c_{k,1}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \frac{\partial x_n^k}{\partial c_{k,1}^{(n)}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} x_n^1 = \sum_{n=1}^{N} \frac{\partial J}{\partial x_n^k} \qquad (13)$$

Mise en œuvre d'algorithmes d'optimisation :

Nous venons de décrire la procédure de calcul du gradient de la fonction de coût des moindres carrés *J* vis-à-vis des paramètres d'un réseau de neurones récurrent. Il est donc désormais possible de mettre en œuvre des algorithmes d'optimisation, de manière à atteindre le minimum de *J*. La non linéarité de la fonction réalisée par le réseau vis-à-vis de ses paramètres impose d'autre part l'utilisation d'algorithmes d'optimisation itératifs. On procède donc de la même manière que pour un réseau non bouclé, à ceci près que les paramètres des différentes copies doivent être identiques (poids partagés). L'apprentissage d'un réseau de neurones récurrent s'effectue alors de la façon suivante :

1. initialisation des paramètres du réseau
2. propagation à travers le réseau déplié temporellement
3. calcul du gradient de la fonction de coût (en tenant compte des poids partagés)
4. mise à jour des paramètres du réseau, puis retour à l'étape 2. pour une nouvelle itération

Nous utiliserons cet algorithme d'apprentissage dans la suite de ce mémoire.

3.3 Sélection de modèles

3.3.1 Minima locaux et initialisation des paramètres

Nous avons mentionné précédemment que la fonction de coût *J* n'est pas quadratique en $\boldsymbol{\theta}$. Ceci provient du fait que la sortie d'un réseau de neurones est non linéaire en ses paramètres : en raison du choix des fonctions d'activation de la couche cachée, qui sont généralement non linéaires (fonctions sigmoïdes), et éventuellement en raison du caractère récurrent des réseaux employés. Dans ces conditions, la fonction de coût ne présente plus un minimum absolu unique, mais plusieurs minima locaux, vers lesquels peuvent converger les algorithmes d'optimisation. Selon le choix des paramètres du réseau en début d'apprentissage, les résultats de l'apprentissage peuvent donc être très différents.

Lors de l'apprentissage d'un modèle neuronal, nous procéderons donc à un grand nombre d'apprentissages successifs, correspondant à des initialisations différentes des paramètres du réseau. De plus, afin d'accélérer le début de l'apprentissage, il est recommandé de choisir des valeurs initiales de ces paramètres suffisamment proches de zéro pour que les sigmoïdes des neurones cachés soient dans leur zone linéaire. La formule (12) du paragraphe précédent montre en effet que le gradient de la fonction de coût est proportionnel à la dérivée de la fonction sigmoïde Φ', qui est maximale en zéro, et tend vers zéro en $\pm\infty$ [7]. Pour s'assurer

que l'apprentissage débute correctement, nous choisirons donc les différentes initialisations à partir d'une distribution de probabilité centrée en 0, et d'écart type égal à l'inverse de la racine carrée du nombre d'entrées du réseau (biais non compris).

3.3.2 Surajustement et dilemme biais-variance

D'autre part, rien ne garantit que le modèle correspondant à un coût d'apprentissage minimum soit le mieux adapté au processus que l'on souhaite modéliser. On peut en effet se trouver dans une situation de surajustement, c'est-à-dire une situation dans laquelle le modèle rend très bien compte des mesures employées pour ajuster ses paramètres (base d'apprentissage), mais ne peut prédire correctement des mesures inconnues. En effet , en vertu de la propriété d'approximation universelle, il est possible d'approcher avec une grande précision les mesures d'apprentissage, si la complexité du modèle (contrôlée par le nombre de neurones cachés dans une structure de type Perceptron) est suffisamment grande. Si ces mesures sont entachées de bruit, le modèle aura en quelque sorte « appris le bruit », ce qui pénalisera sa capacité de généralisation. L'objectif est donc de trouver un compromis entre qualité de l'apprentissage, et capacité de généralisation, compromis connu également sous le nom de dilemme biais-variance [8], le biais caractérisant l'écart des estimations aux mesures, et la variance exprimant la sensibilité du modèle aux mesures utilisées pour son apprentissage.
Différentes méthodes pratiques de sélection ont été élaborées pour tenter de trouver un compromis satisfaisants à ce dilemme [9]. Citons notamment la validation croisée [10], et le leave-one-out réel ou virtuel [11].

3.4 Quelques méthodes de régularisation

Nous avons présenté précédemment les résultats nécessaires à l'élaboration de modèles neuronaux, dans les cas statique et dynamique. La démarche de conception consiste à définir une structure pour le réseau (configuration du réseau, nombre d'entrées, nombre de neurones cachés,…), puis à minimiser la fonction de coût des moindres carrés, pour le modèle choisi et la base d'apprentissage disponible. Nous venons également d'évoquer la question de la capacité de généralisation du réseau ainsi obtenu, que nous proposons d'évaluer en fin d'apprentissage, dans une phase dite de sélection de modèle.
Il est toutefois possible de prendre en considération le dilemme biais-variance dès la phase de conception du modèle, par des méthodes dites de régularisation [12], dont nous allons présenter deux exemples : une méthode de modération des paramètres (« weight decay ») et une méthode d'arrêt prématuré (« early stopping »).

3.4.1 Modération des paramètres

Une valeur élevée d'un paramètre d'un neurone rend la sortie de celui-ci très sensible à la valeur de la variable correspondante, donc au bruit de mesure sur cette variable. Dans la pratique, on observe qu'en situation de surajustement certains paramètres du réseau obtenu prennent des valeurs absolues élevées. De manière à éviter ce phénomène, une solution consiste à ajouter à la fonction de coût des moindres carrés un terme de régularisation qui pénalise les grandes valeurs absolues des paramètres [13] :

$$J'(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\alpha}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} = \frac{1}{2}\sum_{n=1}^{N}\left(y_n^p - y_n(\boldsymbol{\theta})\right)^2 + \frac{\alpha}{2}\sum_{n=1}^{N}(\theta_n)^2 \qquad (14)$$

où α est un scalaire positif contrôlant la modération des paramètres (de grandes valeurs de α favorisent les valeurs faibles, tandis que, pour de petites valeurs de α, le terme de régularisation aura une influence plus faible).

Notons que du point de vue de la théorie statistique de l'apprentissage (cf. chapitre 1 : Modélisation), la pénalisation des valeurs importantes des paramètres revient à limiter le nombre de fonctions candidates : le premier terme de la fonction de coût (14) correspond donc à une diminution du risque empirique, tandis que son second terme correspond à une limitation de la capacité de la classe de fonctions envisagée. Nous retrouverons ce principe de régularisation par pénalisation de la norme du vecteur des paramètres dans le chapitre 4, consacré aux machines à vecteurs supports.

Le vecteur gradient de cette nouvelle fonction de coût *J'* s'exprime alors de la façon suivante :

$$\frac{\partial J'}{\partial \boldsymbol{\theta}} = \frac{\partial J}{\partial \boldsymbol{\theta}} + \alpha\boldsymbol{\theta} \qquad (15)$$

Il suffit donc d'ajouter au gradient de la fonction de coût des moindres carrés, calculable par rétropropagation, le terme de régularisation $\alpha\boldsymbol{\theta}$ pour obtenir le gradient de *J'*. L'apprentissage d'un réseau de neurones par cette méthode de modération des paramètres s'effectue par ailleurs de la même manière que dans le cas classique.

3.4.2 Arrêt prématuré

La méthode d'arrêt prématuré est fondée sur un algorithme d'apprentissage classique, dont le critère d'arrêt est modifié : pour éviter que le modèle ne s'ajuste trop finement sur les données d'apprentissage, l'optimisation est interrompue en cours d'algorithme. Le critère d'arrêt habituellement retenu est alors l'augmentation de la fonction de coût sur une base de validation (qui n'intervient pas directement dans la détermination des paramètres du modèle) : on s'attend en effet à ce que l'erreur sur cette base de validation diminue en début

d'apprentissage (ce qui correspond à une diminution importante du facteur de biais), avant d'augmenter à partir du moment où le modèle commence à se surajuster à la base d'apprentissage (augmentation importante du terme de variance).

4 Modélisation de la température après turbine d'un véhicule

4.1 Contexte et motivation

On s'intéresse dans ce paragraphe à l'estimation de la température T_{apt} en un point particulier de la ligne d'échappement d'un véhicule de série (Renault Laguna), situé après la turbine. La Figure 5 présente de façon schématique les circuits d'admission et d'échappement du moteur diesel G9T600 (2.2 litre DCI), qui a été utilisé lors de la réalisation des mesures de cette température.

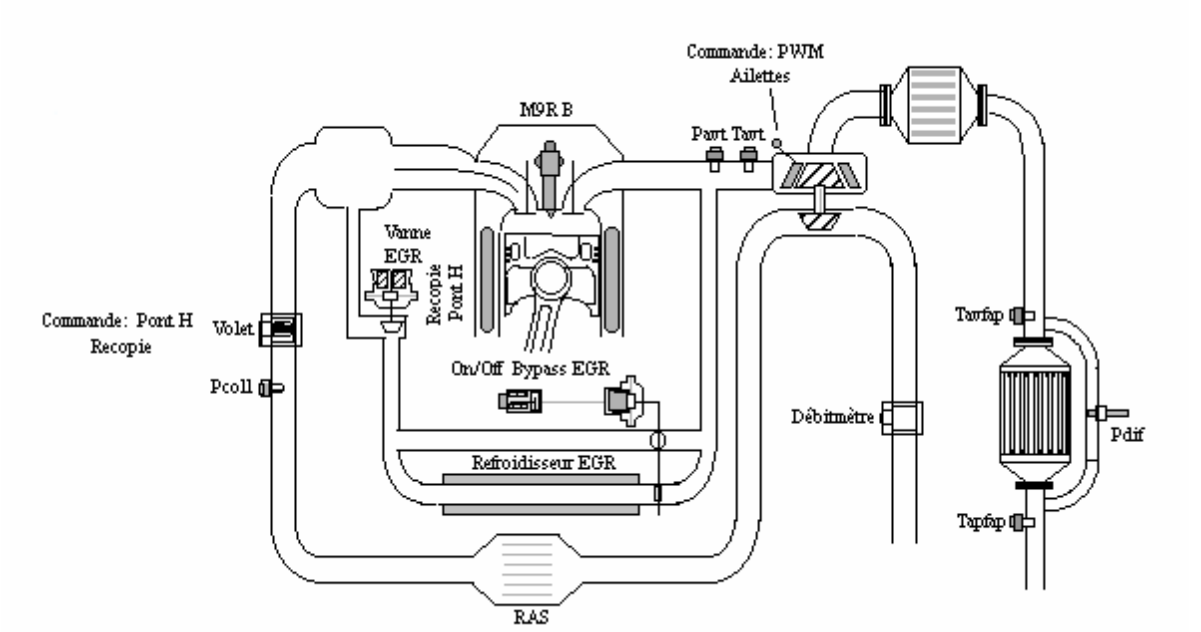


Figure 5 : circuits d'admission et d'échappement du moteur G9T600

Les détails de la ligne d'échappement sont représentés sur la Figure 6. On distingue les organes suivants :

- *la turbine* : cet élément, également appelé turbo, a pour fonction de détendre les gaz d'échappements, et de comprimer les gaz d'admission (liaison mécanique avec le compresseur)
- *le filtre à NOx* : le piège à oxydes d'azote assure une fonction de dépollution, en retenant la majorité de ces oxydes issus de la combustion

- *le filtre à particules* : il assure également une fonction de dépollution vis-à-vis des particules principalement présentes dans les combustions diesel
- *le calculateur d'injection* : cet élément gère l'ensemble du contrôle moteur et permet le traitement des signaux électriques provenant des capteurs (thermocouples) présents sur la ligne d'échappement, pour qu'elles soient prises en considération dans les stratégies de dépollution.

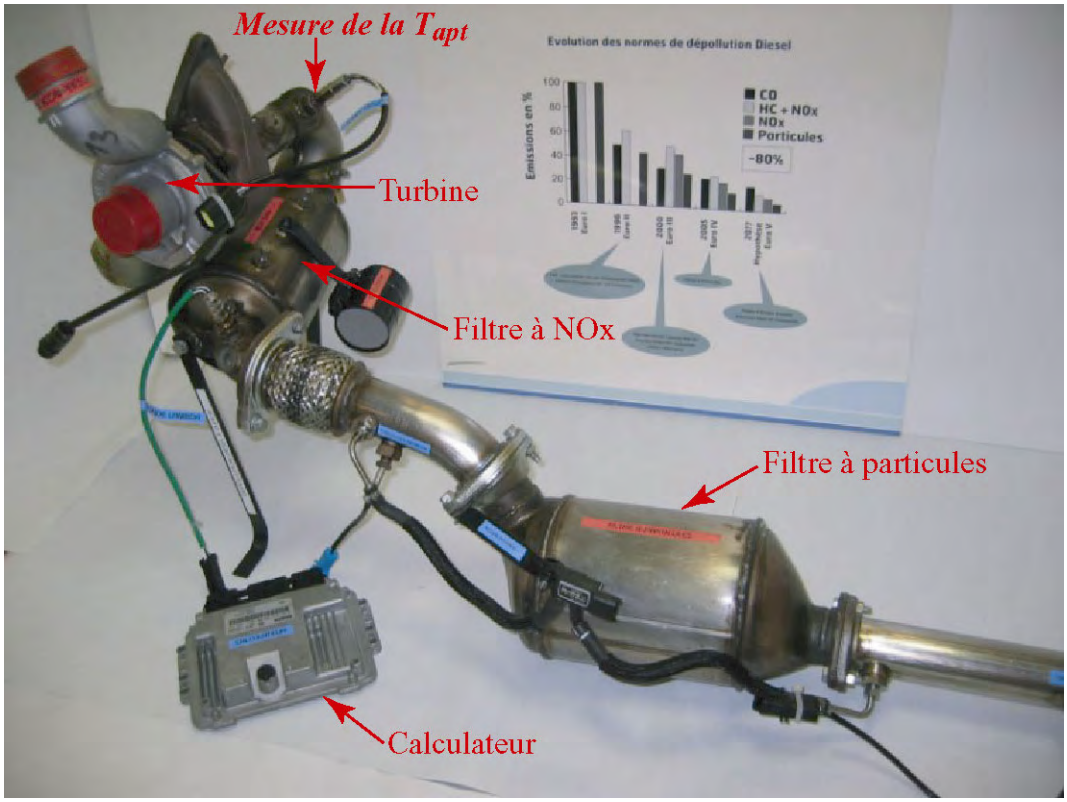


Figure 6 : détail de la ligne d'échappement et différents organes d'intérêt

Une mesure directe de cette température est envisageable (par exemple au moyen d'un thermocouple supplémentaire), mais l'on cherche ici à proposer une alternative plus économique, consistant à estimer celle-ci au moyen d'un modèle : on cherche à réaliser un « capteur virtuel ». L'utilisation d'une mesure directe soulève en effet, outre des questions de surcoût non négligeables, un problème relatif au nombre de mesures gérables par le calculateur embarqué actuel : d'autres mesures de températures sur la ligne d'échappement sont indispensables, et elles sont prioritaires pour la mise en œuvre les stratégies de contrôle des différents organes de la ligne d'échappements (filtre à particules, filtres à oxydes d'azote, pot catalytique), de sorte que, pour mesurer la T_{apt}, il conviendrait de modifier les caractéristiques du calculateur utilisé. Le surcoût qui en découle serait alors double : d'une

part en raison du prix d'un thermocouple, et d'autre part en raison du coût d'un calculateur embarqué plus évolué.

L'objectif à atteindre est donc d'estimer cette température T_{apt} avec une précision de l'ordre de 10°C, à partir des mesures des variables de contrôle du moteur accessibles sur le véhicule (nous reviendrons sur la liste des variables accessibles dans le paragraphe suivant). On s'autorise toutefois à commettre des erreurs supérieures à ce seuil de 10°C, mais de manière très localisée et temporaire : on souhaite donc avoir un histogramme des erreurs absolues commises par le modèle comprenant 80% des points en deçà du seuil de 10°C, et ne comprenant aucun point dont l'erreur est supérieure à 30°C.

4.2 Choix d'une hypothèse et élaboration d'une base d'apprentissage

Les travaux que nous avons menés pour modéliser la T_{apt} par apprentissage font suite à une étude préalable fondée sur une analyse physique des phénomènes intervenant dans l'évolution de cette température. En effet, nous allons tirer profit de cette étude pour constituer un ensemble de données pour l'apprentissage, ainsi qu'une hypothèse.

Il a été établi que la température T_{apt} peut être décrite par un modèle dynamique non linéaire du deuxième ordre, possédant 6 variables exogènes (qui sont à la fois des variables de contrôle du moteur, ou des variables d'état accessibles). La dynamique du processus autorise d'autre part une acquisition des données à une fréquence d'échantillonnage de 10 Hz. Ces caractéristiques sont résumées dans le Tableau 1.

	CARACTERISTIQUES	*COMMENTAIRES*
Variables de commande	• régime moteur (N) • débit d'air (Q_{air}) • débit d'injection (Q_{inj}) • température avant turbine (T_{avt}) • pression compresseur (P_{comp})	instant courant instant courant instant courant instant courant + retard 1T instant courant
Dynamique	ordre 2	*période d'échantillonnage* : 10Hz
Non linéarités	oui	*origine* : convection, capacités calorifiques, échanges thermiques…

Tableau 1 : informations préalables pour la modélisation de la T_{apt}, découlant d'une analyse physique des phénomènes entrant en jeu.

Sur la base de ces informations, on envisage donc d'élaborer un modèle non linéaire dynamique correspondant à l'hypothèse suivante :

$$\begin{cases} \mathbf{x}(k) = \mathbf{\Phi}\left(\mathbf{x}(k-1), \mathbf{u}(k-1)\right) \\ T_{apt}(k) = \Psi\left(\mathbf{x}(k)\right) + w(k) \end{cases} \tag{16}$$

où :

$\mathbf{u}(k) = \left[N(k), Q_{air}(k), Q_{inj}(k), T_{avt}(k), T_{avt}(k-1), P_{comp}(k) \right]^T$ est le vecteur des entrées de commande,

$\mathbf{x}(k)$ est le vecteur d'état de dimension 2,

$w(k)$ est un bruit de sortie additif,

$\mathbf{\Phi}$ et Ψ sont deux fonctions non linéaires.

Pour constituer des ensembles d'apprentissage et de validation, il convient de mesurer les 6 entrées de commande ainsi que la sortie T_{apt} dans des situations permettant de rendre compte de la plupart des comportements de roulage d'un véhicule. Sachant d'autre part que ce véhicule doit être homologué, vis-à-vis des émissions polluantes, sur des cycles normalisés (en Europe, le cycle NMVEG, pour « New Motor Vehicle Exhausted Gas »), de tels cycles ont été incorporés à la base d'apprentissage. De plus, des cycles FTP (pour « Federal Test Procedure ») ont été ajoutés ; ils correspondent à la norme de certification américaine, et présentent la caractéristique de comporter davantage de zones transitoires, où les effets

dynamiques sont plus importants. On dispose ainsi d'une grande quantité de points de mesures, à partir desquels ont été extraites deux bases, l'une d'apprentissage, qui servira à l'ajustement des paramètres du modèle, et l'autre de validation, pour en évaluer les performances.. Une grande attention a été portée à l'établissement de la base d'apprentissage, qui doit couvrir l'ensemble des situations extrêmes de fonctionnement, et bien représenter la diversité des régimes de roulage. Citons notamment la présence de séquences caractérisées par:

- des valeurs maximales et minimales des entrées de commande
- des zones de régénération du filtre à particules (FAP)
- des zones de roulage en altitude
- des zones de roulage à température extérieure variable (entre 0°C et 40°C)
- des zones de régénération vis-à-vis des oxydes de souffre (DeSOx)

On obtient finalement une base d'apprentissage comprenant 4232 mesures (échantillonnées à 10 Hz), correspondant à 6 séquences temporelles disjointes, et une base de validation de 2563 mesures correspondant à 2 séquences temporelles disjointes.

4.3 Résultats et discussion

4.3.1 Elaboration du modèle neuronal récurrent

Nous avons supposé dans la relation (16) que seul un bruit de sortie perturbe la mesure de la T_{apt}, et chercherons donc à élaborer un modèle de simulation fondé sur un prédicteur bouclé. D'autre part, les états du processus ne seront pas accessibles, de sorte que nous allons réaliser un modèle neuronal récurrent, correspondant à une représentation d'état dont les états ne sont pas mesurés. Nous utiliserons donc l'algorithme d'apprentissage dit *semi dirigé* (défini et décrit au paragraphe 3.2.2). La Figure 7 décrit la structure envisagée.

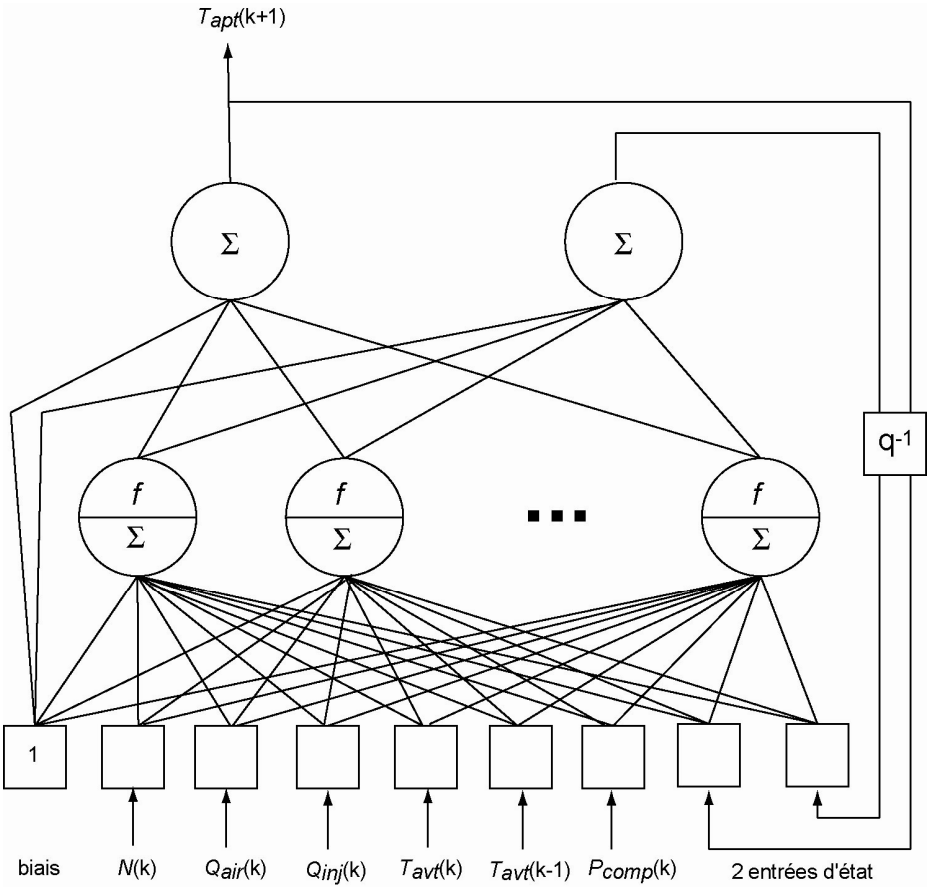


Figure 7 : structure adoptée pour le modèle neuronal récurrent

La seule caractéristique qu'il est possible de faire varier pour l'apprentissage, et qui contrôle la capacité de la famille de fonctions candidates, est donc le nombre de neurones cachés N_{cc}. Des modèles de complexité croissante ont été réalisés[1].

Du fait que la base d'apprentissage est constituée d'un ensemble de 6 séquences temporelles disjointes, le calcul du gradient de la fonction de coût, durant la phase d'apprentissage, doit tenir compte de cette caractéristique : l'erreur est donc propagée au sein de chacune des séquences, et le gradient total est défini comme la somme des gradients rétropropagés sur ces 6 séquences.

La structure optimale obtenue, au sens du minimum de l'erreur quadratique sur la base de validation, correspond à un nombre de neurones cachés N_{cc} égal à 4. Les racines carrées de l'erreur quadratique moyenne (REQM) sur les bases d'apprentissage et de validation ont pour

[1] Optimisation de la fonction de coût par l'algorithme BFGS (cf. Chapitre 2), précédée d'une phase d'optimisation par méthode du gradient simple avec pas asservi ; 100 initialisations des paramètres par tirage aléatoire dans une distribution gaussienne de variance 0.125 (soit $1/n$ où n=8 est le nombre de variables en entrée du réseau); normalisation et centrage préalables des variables et des sorties

cette structure des valeurs comparables, respectivement de 6.9°C et de 8.2°C. Les différentes informations relatives au modèle obtenu sont résumées dans le Tableau 2.

PROPRIETE	CARACTERISTIQUE
Structure	Nombre d'entrées d'état : $N_s = 2$ Nombre de neurones cachés : $N_{cc} = 4$
Nombre de paramètres	$N_p = 46$
Algorithme d'apprentissage	Gradient à pas asservi : 500 itérations BFGS : 1000 itérations
Base d'apprentissage	Taille de la base: 4232 points, 6 séquences REQM = 6.9°C
Base de validation	Taille de la base: 2563 points, 2 séquences REQM = 8.2°C

Tableau 2 : caractéristiques du meilleur réseau de neurones récurrent obtenu pour la modélisation de la température après turbine

La figure ci-dessous présente les mesures et estimations de la T_{apt} sur un cycle normalisé NMVEG appartenant à la base de validation, ainsi que l'histogramme des erreurs correspondantes. On constate que le modèle obtenu remplit les objectifs initialement fixés, tant vis-à-vis de l'erreur quadratique moyenne que vis-à-vis de la répartition des erreurs.

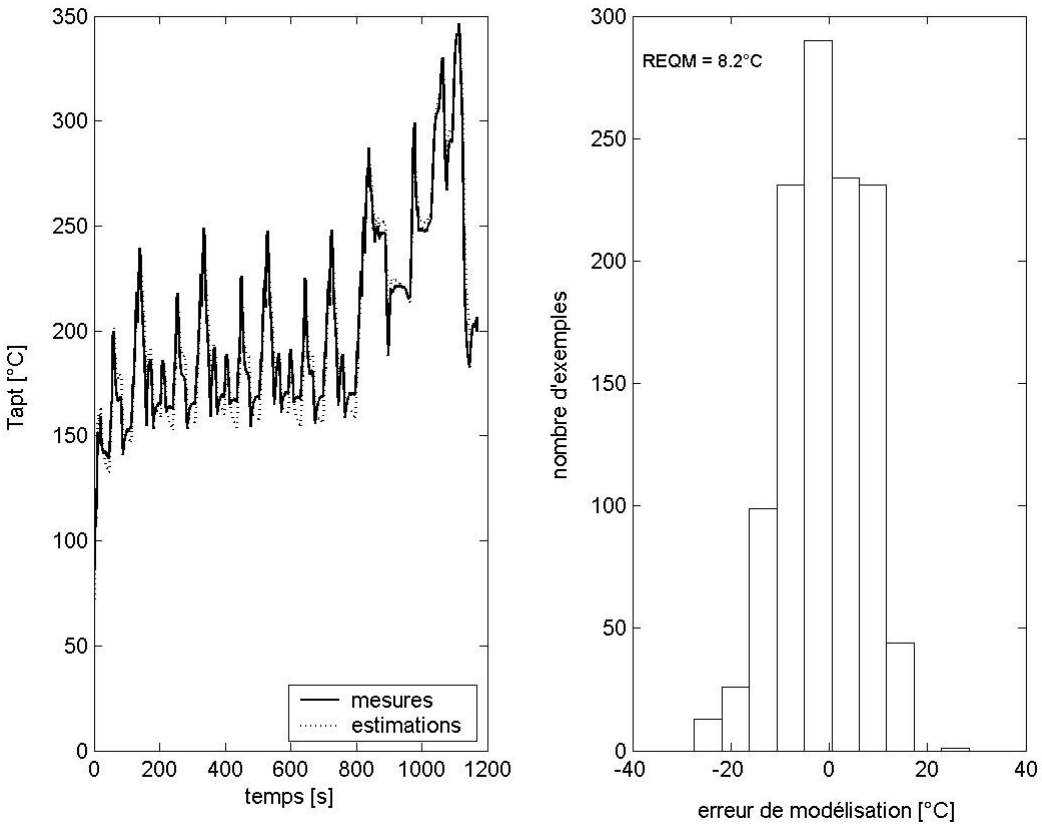


Figure 8 : *gauche* **:** mesures et estimations de la T_{apt} sur un cycle NMVEG de validation. *droite* **:** histogramme des erreurs de modélisation.

4.3.2 Comparaison à un modèle semi-physique existant

Le modèle neuronal élaboré suivant la procédure décrite dans le paragraphe précédent a été comparé à un modèle semi physique existant, développé par Pascal Barrillon et Sebastien Romieux du service Contrôle et Post-traitement (66162) de la Direction de l'Ingénierie Mécanique.

4.3.2.1 Description du modèle : équations physiques

Nous allons dans un premier temps décrire sommairement ce modèle, représenté de manière schématique sur la Figure 9.

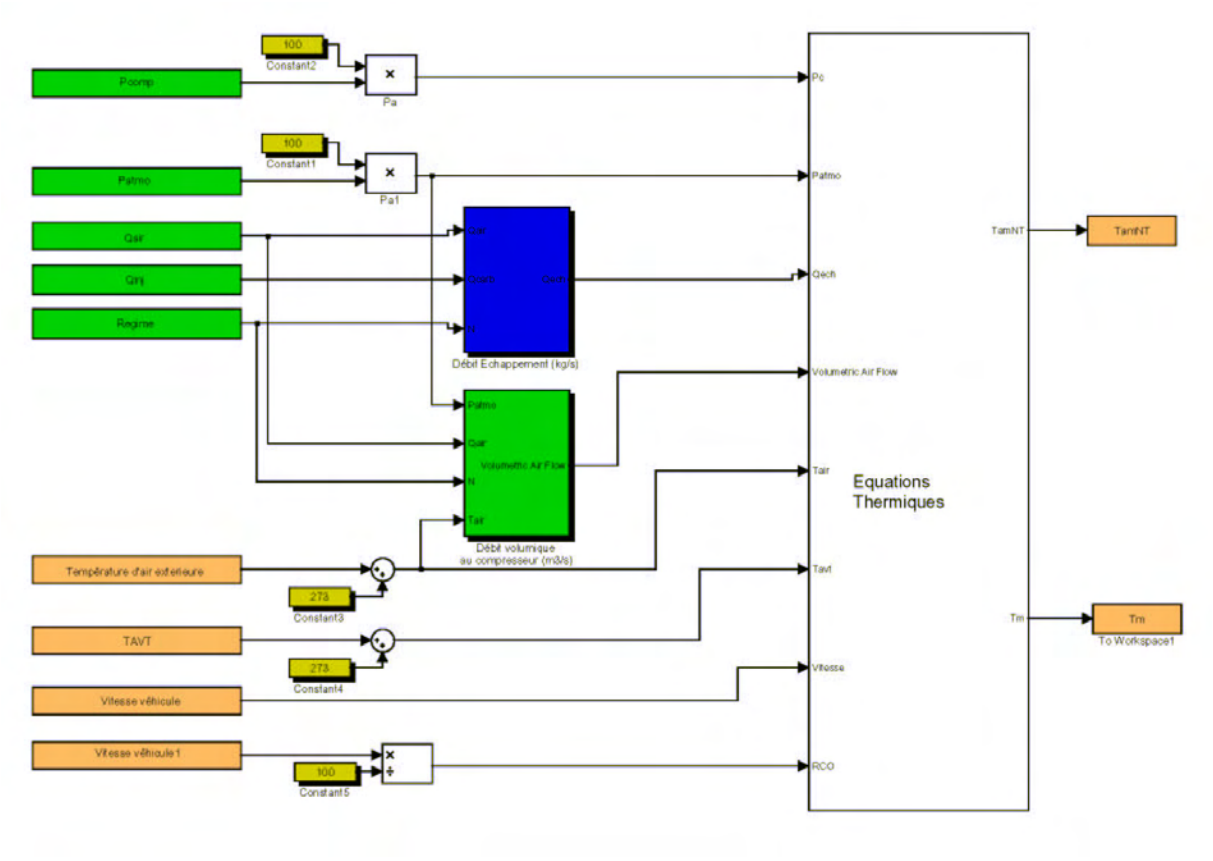


Figure 9 : schéma de principe du modèle semi-physique existant : à gauche figurent les 9 variables d'entrées du modèle, qui sont converties en un ensemble de 8 grandeurs avant traitement par le bloc « Equations Thermiques ».

La première opération réalisée par ce modèle consiste en une conversion pour obtenir, à partir des 9 variables d'entrée, un ensemble de 8 grandeurs qui sont transmises au bloc « Equations Thermiques », où sont pris en considération les différents phénomènes physiques intervenant dans le processus. Notons que, par comparaison avec le modèle neuronal, ce modèle semi-physique fait intervenir trois variables d'entrée supplémentaire : la température d'air extérieur, la pression atmosphérique et la vitesse véhicule. Il s'avère cependant que ces trois variables n'ont que peu d'influence sur le résultat final, ce qui justifie qu'elles ne soient pas présentes en entrée du modèle neuronal.

Le bloc « Equations Thermiques » se décompose en une série de 4 sous blocs (Figure 10), réalisant les opérations suivantes :

- le bloc « Puissance Mécanique » calcule les échanges thermomécaniques entre le gaz et le turbine (en tenant compte des pertes énergétiques au niveau de la turbine)
- le bloc « Convection » calcule les termes de convection interne et externe à la turbine
- le bloc « Température Matériau » calcule la température moyenne du matériau (constitué de la turbine et du tube)

- le bloc « Température Gaz » calcule la température du gaz après la turbine

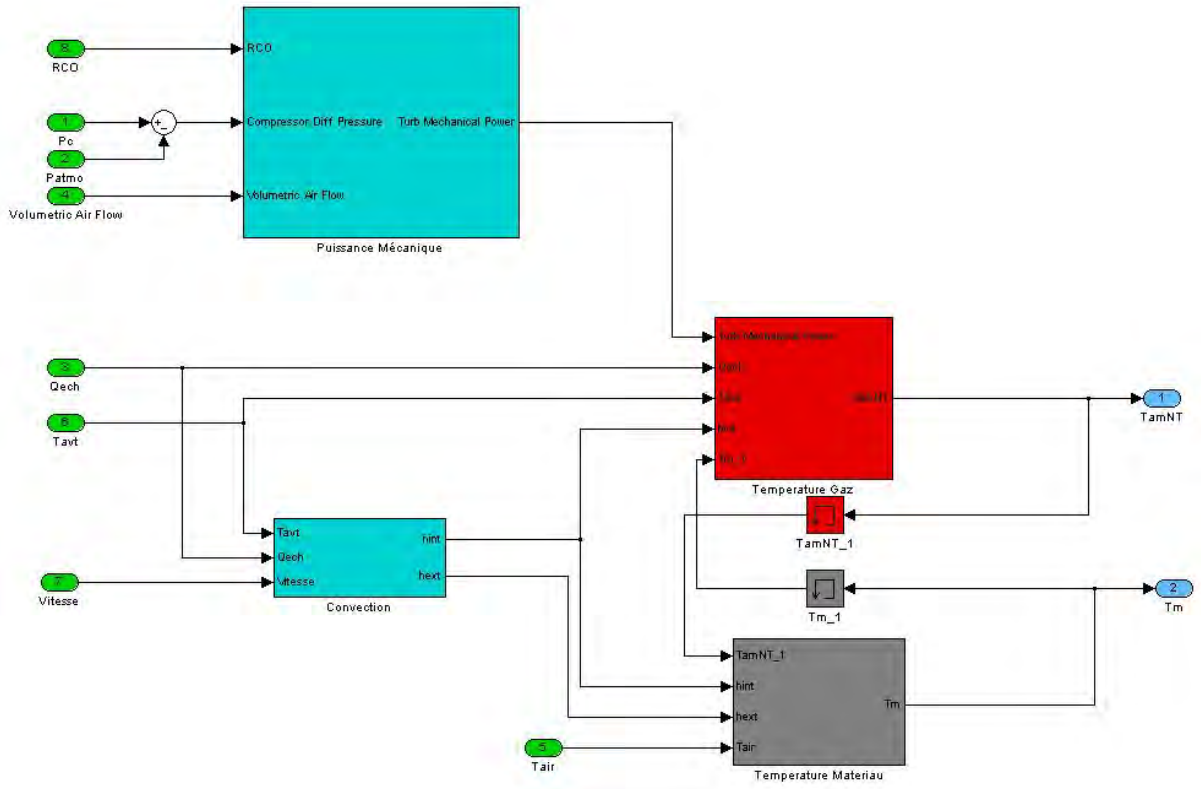


Figure 10 : sous-bloc « Equations Thermiques » de la Figure 9, représenté de manière détaillée.

L'ensemble de ces opérations exprime le principe de conservation de l'énergie, qui s'écrit en l'occurrence :

$$\begin{cases} \dfrac{d}{dt}\left(E_{gaz}\right) = \dfrac{d}{dt}\left(Q_{\text{int}} + W_t + E_c + E_p\right) \\ \dfrac{d}{dt}\left(E_m\right) = -\dfrac{d}{dt}\left(Q_{\text{int}} + Q_{ext}\right) \\ \dfrac{d}{dt}\left(E_{tc}\right) = -\dfrac{d}{dt}\left(W_t + W_c + W_f\right) \end{cases} \tag{17}$$

où :

E_{gaz} est l'énergie thermique du gaz

E_m est l'énergie thermique du matériau

E_{tc} est l'énergie mécanique du turbo compresseur

Q_{ext} est la chaleur fournie à l'air par le matériau

Q_{int} est la chaleur fournie au gaz par le matériau

W_t est l'énergie mécanique fourni par la turbine au gaz à l'échappement

W_c est l'énergie mécanique fourni par le compresseur au gaz à l'admission

W_f est l'énergie mécanique du turbo compresseur perdue par frottements mécaniques

E_p est l'énergie potentielle du gaz à l'échappement

E_c est l'énergie cinétique du gaz à l'échappement

L'énergie mécanique apportée par la turbine est transmise aux gaz (sous forme d'énergie thermique et convective), et au milieu extérieur (sous forme de pertes d'énergie par radiation). En explicitant, par une analyse thermodynamique et mécanique, les différents termes intervenant dans l'équation (17), on aboutit aux relations à temps discret suivantes :

$$\begin{cases} T_{apt}(k+1) = \dfrac{m_{gaz}c_{p,gaz}T_{apt}(k) + \left[Q_{ech}c_{p,gaz}T_{avt}(k+1) + h_{c,int}S_{int}T_m(k) + \dot{W}_t\right]\tau_{samp}}{m_{gaz}c_{p,gaz} + \left[Q_{ech}c_{p,gaz} + h_{c,int}S_{int}\right]\tau_{samp}} \\ T_m(k+1) = \dfrac{m_m c_{p,m}T_m(k) + \left[h_{c,int}S_{int}T_{apt}(k) + h_{c,ext}S_{ext}T_{air}(k+1)\right]\tau_{samp}}{m_m c_{p,m} + \left[h_{c,int}S_{int} + h_{c,ext}S_{ext}\right]\tau_{samp} + F_{2,3}\sigma S_{ext}T_m^3(k)} \\ \varpi(k+1) = \dfrac{J\varpi(k) + \left[\left(P_{avt} - P_{apt}\right)S_{e,t}R_{m,t} + \left(P_{atmo} - P_{col}\right)S_{e,c}R_{m,c} + \left(f_a\varpi^2(k) - f_s\right)\right]\tau_{samp}}{J + \left(2f_a\varpi(k) + f_v\right)\tau_{samp}} \end{cases} \quad (18)$$

où :

$T_{apt}(k+1)$ est la température après turbine à l'instant discret k+1

$T_m(k+1)$ est la température du matériau à l'instant discret k+1

$T_{air}(k+1)$ est la température d'air à l'instant discret k+1

$\varpi(k+1)$ est la vitesse de rotation de la turbine à l'instant discret k+1

$T_{avt}(k+1)$ est la température avant turbine à l'instant discret k+1

τ_{samp} est la période d'échantillonnage

$c_{p,gaz}$, $c_{p,m}$ sont respectivement les capacités calorifiques du gaz et du matériau

m_{gaz} est la masse moyenne du gaz en amont du filtre à NOx

m_m est la masse effective du matériau

Q_{ech} est le débit d'échappement

S_{int}, S_{ext} sont respectivement les surfaces d'échange intérieure et extérieure

$h_{c,int}$, $h_{c,ext}$ sont respectivement les cœfficients de convection interne et externe

$\dot{W}_t$ est la puissance mécanique de la turbine

σ est la constante de Stefan-Boltzman (=5.67 $Wm^{-2}K^{-4}$)

$F_{2,3}$ est un facteur de forme pour les échanges entre le matériau et le gaz extérieur

J est l'inertie de la turbine

$P_{avt}, P_{apt}, P_{atmo}, P_{col}$ sont respectivement les pressions avant turbine, après la turbine, atmosphérique, et du collecteur

$S_{e,t}, S_{e,c}$ sont les surfaces efficaces à la turbine et au compresseur respectivement (sur lesquelles agissent les différentiels de pression)

$R_{m,t}, R_{m,c}$ sont les rayons efficaces à la turbine et au compresseur respectivement (permettant de définir des couples efficaces)

f_a, f_s, f_v sont les coefficients de frottement respectivement aéraulique, sec et visqueux à la turbine

La puissance mécanique de la turbine est modélisée de la façon suivante :

$$\dot{W}_t = \eta_{RCO} Q_{vol} \left(\alpha_9 \Delta P_c^2 + \alpha_{10} \Delta P_c + \alpha_{11} \right)$$

où η_{RCO} est le rendement de la turbine, fonction de la position des ailettes RCO, où Q_{vol} est le débit volumique de l'air à la turbine, et où ΔP_c est le différentiel de pression aux bornes de la turbine. Les coefficients α_i sont des paramètres non physiques à déterminer dans une phase d'identification.

Les termes de convections $h_{c,int} S_{int}$ et $h_{c,ext} S_{ext}$ sont traités de la manière suivante :

$$h_{c,int} S_{int} = k \left[\alpha_5 \left(\frac{Q_{ech}}{\mu} \right)^2 + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right]$$

$$h_{c,ext} S_{ext} = \alpha_6 V_h + \alpha_7$$

où k est la conductivité thermique du gaz, μ est la viscosité du fluide, V_h est la vitesse du véhicule, et où les coefficients α_i sont des paramètres à déterminer par apprentissage.

Le modèle représenté par les équations (18) peut être simplifié en négligeant l'inertie de la turbine et les frottements aérauliques. Il convient d'autre part d'ajouter un terme correctif proportionnel à la dérivée de la température avant turbine dans l'équation dictant l'évolution de $T_{apt}(k+1)$.

L'analyse physique des différents phénomènes intervenant dans l'évolution de la température après turbine permet ainsi d'aboutir au modèle simplifié suivant, où l'ensemble des paramètres (physiques ou non) sont représentés par des coefficients α_i :

$$T_{apt}(k+1) = \frac{\alpha_2 c_{p,gaz} T_{apt}(k) + \left[\alpha_{12} \frac{T_{avt}(k+1) - T_{avt}(k)}{\tau_{samp}} + Q_{ech} c_{p,gaz} T_{avt}(k+1) + k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) T_m(k) + \eta_{RCO} Q_{vol} \left(\alpha_9 \Delta P_c^2 + \alpha_{10} \Delta P_c + \alpha_{11} \right) \right] \tau_{samp}}{\alpha_2 c_{p,gaz} + \left[Q_{ech} c_{p,gaz} + k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) \right] \tau_{samp}}$$

$$T_m(k+1) = \frac{\alpha_1 c_{p,m} T_m(k) + \left[k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) T_{apt}(k) + \left(\alpha_6 V_h + \alpha_7 \right) T_{air}(k+1) \right] \tau_{samp}}{\alpha_1 c_{p,m} + \left[k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) + \alpha_6 V_h + \alpha_7 + \alpha_8 T_m^3(k) \right] \tau_{samp}}$$

$$\eta_{RCO} = 1 + \alpha_0 RCO$$

(19)

4.3.2.2 Corrections apportées : modèle « boîte grise »

Dans la formule (19), l'ensemble des coefficients α_i sont considérés comme constants. Il s'avère cependant que deux d'entre eux représentent des quantités qui sont appelées à varier suivant le point de fonctionnement. Il s'agit des coefficients:

- $\alpha_2 = m_{gaz}$, représentant la masse de gaz en amont du filtre à NOx
- α_{12}, qui est un terme correctif portant sur la dérivée de la température avant turbine

Ces deux coefficients sont donc remplacés par des neurones formels, à fonction d'activation *tanh* et admettant pour entrées les 5 variables de commande suivantes : régime *N,* pression au compresseur P_{comp}, débit d'air Q_{air}, débit de carburant Q_{inj}, température avant turbine T_{avt}

$$\tilde{\alpha}_2(k) = \tanh\left[\alpha_{2_1} N(k) + \alpha_{2_2} P_{comp}(k) + \alpha_{2_3} Q_{air}(k) + \alpha_{2_4} Q_{inj}(k) + \alpha_{2_5} T_{avt}(k) + \alpha_{2_6} \right]$$

$$\tilde{\alpha}_{12}(k) = \tanh\left[\alpha_{12_1} N(k) + \alpha_{12_2} P_{comp}(k) + \alpha_{12_3} Q_{air}(k) + \alpha_{12_4} Q_{inj}(k) + \alpha_{12_5} T_{avt}(k) + \alpha_{12_6} \right]$$

D'autre part, un neurone formel (à fonction d'activation *tanh*) est appliqué en sortie du modèle. L'évolution de la T_{apt} devient alors dictée par la formule suivante (en remplacement de la première des équations (19)):

$$T_{apt}(k+1) = \tanh\left[w \frac{\left[\tilde{\alpha}_{12}(k) \frac{T_{avt}(k+1) - T_{avt}(k)}{\tau_{samp}} + Q_{ech} c_{p,gaz} T_{avt}(k+1) + k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) T_m(k) + \eta_{RCO} Q_{vol} \left(\alpha_9 \Delta P_c^2 + \alpha_{10} \Delta P_c + \alpha_{11} \right) \right] \tau_{samp}}{\alpha_2 c_{p,gaz} + \left[Q_{ech} c_{p,gaz} + k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) \right] \tau_{samp}} + w \frac{\tilde{\alpha}_2(k) c_{p,gaz} T_{apt}(k)}{\alpha_2 c_{p,gaz} + \left[Q_{ech} c_{p,gaz} + k\left(\alpha_5 \frac{Q_{ech}^2}{\mu^2} + \alpha_4 \frac{Q_{ech}}{\mu} + \alpha_3 \right) \right] \tau_{samp}} + b \right]$$

où w et b sont le poids et le biais du neurone formel placé en sortie du modèle.

Le modèle résultant peut ainsi être qualifié de modèle semi physique [14], en ce qu'il associe un modèle de connaissance issu de l'analyse des mécanismes physico-chimiques mis en œuvre par le processus, ainsi que des fonctions non linéaires paramétrées qui sont ajustées par apprentissage. Ce modèle a été développé et identifié sous le logiciel Neuro One©, puis implémenté sous Matlab® Simulink®.

4.3.2.3 Résultats et comparaison

Les erreurs commises par ce modèle semi physique sont du même ordre de grandeur, sur les bases d'apprentissage et de validation, que celles commises par le réseau de neurones récurrent, soit respectivement des REQM de 7.8°C et 8.2°C. À titre d'exemple, sur le cycle NMVEG de validation présenté sur la Figure 8, l'erreur commise par le modèle semi physique est de REQM = 6.2°C, contre 8.2°C pour le modèle neuronal. Pour ce problème, les deux approches modélisation semi physique et modélisation boîte noire permettent d'obtenir des modèles de performances équivalentes.

Du point de vue du temps de développement, il est cependant clair que le modèle neuronal est avantageux : les différences et évolutions techniques (entre générations de véhicules), ou le déploiement sur toute la gamme (différents modèles et/ou motorisations), requièrent l'élaboration d'un nouveau modèle qui sera bien plus aisé à réaliser par apprentissage. Les variables du modèle resteront en effet les mêmes (en première approximation et sauf évolution technique majeure), de sorte que l'apprentissage d'un nouveau modèle neuronal se résume à l'élaboration d'une base de données représentative, puis à la mise en œuvre des algorithmes d'apprentissage usuels (en faisant varier la complexité du modèle par le nombre de neurones cachés). En revanche, la prise en considération des évolutions techniques dans le modèle semi-physique ne peut se faire qu'au travers d'une analyse détaillée de celles-ci (par exemple : réorganisation de l'agencement des éléments de la ligne d'échappement, nouveau dimensionnement de ces éléments, caractéristiques différentes de la turbine,…).

Concernant l'élaboration de la base de données, les deux modèles exigent a priori une quantité comparable de données pour l'identification (ou l'apprentissage) des paramètres du modèle, mais également pour vérifier l'adéquation de celui-ci sur l'ensemble du domaine de fonctionnement.

Enfin, notons qu'une étude préliminaire a montré la possibilité d'implémenter un réseau de neurones de type Perceptron à une couche cachée sur un calculateur embarqué sans dégradations notable de sa performance. La mise en œuvre du modèle semi physique ne

devrait a priori pas poser de problèmes particuliers, bien que la diversité des fonctions qu'il contient complique sensiblement les choses. Cette étude doit être poursuivie.
La méthode consistant à modéliser la température après turbine par apprentissage à l'aide d'un réseau de neurones récurrent a fait l'objet d'un dépôt de brevet en 2004 (numéro de dépôt : 04-10523 ; date de dépôt : 06/10/2004 ; titre: Procédé et système améliorés d'estimation d'une température des gaz d'échappement et moteur à combustion interne équipé d'un tel système).

5 Modélisation des émissions polluantes d'un véhicule de série

5.1 Contexte et motivation

On s'intéresse à présent à l'estimation des débits de cinq familles de polluants réglementés en fonction des réglages du moteur. Sans entrer dans une analyse détaillée des réactions chimiques à l'origine de ces émissions, on considérera les composés suivants :

- les oxydes d'azotes (NO_x) : il s'agit principalement du monoxyde d'azote NO, mais également d'une faible proportion de dioxyde d'azote NO_2, qui est un composé hautement toxique. Ces composés polluants apparaissent lors de la combustion d'hydrocarbures dans des conditions de température et pression élevées.
- les hydrocarbures non brûlés (HC) : dus au fait que la combustion est nécessairement incomplète.
- les particules de suie (PM) : il s'agit d'agrégats complexes de matériaux solides et liquides, issus de la combustion des hydrocarbures. Leur composition et leur taille changent suivant les conditions de régime et de charge du moteur. Autour d'un cœur constitué d'une sphérule de carbone, différents composés organiques et inorganiques sont adsorbés, pour former une particule dont la taille varie de 0.1 à 1 micron. De telles particules apparaissent également lors de la combustion essence, mais leur quantité est bien moindre que dans le cas diesel.
- le monoxyde de carbone (CO) : ce composé hautement toxique provient également d'une combustion incomplète du carburant.
- le dioxyde de carbone (CO_2) : participant activement à l'effet de serre, sa présence est limitée par les réglementations actuelles, bien qu'elle soit inévitable en tant que produit d'une combustion même parfaite. La manière la plus efficace de réduire les taux d'émissions de CO_2 est donc de réduire la consommation des moteurs.

Le renforcement de la réglementation depuis plusieurs années impose de contrôler de plus en plus finement ces rejets polluants, à la fois au moyen de méthodes actives, c'est-à-dire en développant de nouveaux organes de dépollution (du type : pot catalytique, filtre à oxydes d'azote, filtre à particules), et en prêtant une attention accrue aux réglages du moteur. À titre d'exemple, l'industrie automobile européenne se fixe pour objectif de réduire à 140g par km les émissions moyennes de CO_2 à l'horizon 2008, pour une moyenne actuelle de 164g par km. Le Tableau 3 présente l'évolution des normes européennes qui réglementent les émissions polluantes, depuis leur apparition en 1990 et jusqu'en 2009, date prévue pour l'entrée en vigueur de la norme Euro 5.

Norme	**Euro 0 (1990)**	**Euro 1 (1993)**	**Euro 2 (1996)**	**Euro 3 (2001)**	**Euro 4 (2006)**	**Euro 5 (2009)**
NO_x	14.4	8	7	5	3.5	2
CO	11.2	4.5	4	2.1	1.5	1.5
HC	2.4	1.1	1.1	0.66	0.46	0.25
PM	-	0.36	0.15	0.1	0.02	0.02

Tableau 3 : évolution des normes européennes en matière de réglementation des émissions polluantes (émissions moyennes sur cycle exprimées en g/kWh)

Notons que cette réglementation s'applique sur des cycles normalisés (cycle NMVEG en Europe), de sorte que ces valeurs ne représentent pas nécessairement les taux d'émissions en conditions de fonctionnement normal.

Du point de vue du motoriste, il est très délicat de déterminer l'influence exacte de telle ou telle variable de contrôle moteur sur les émissions polluantes, d'autant que celles-ci suivent parfois des variations antagonistes (une diminution des NO_x s'accompagnant généralement d'une augmentation des PM, par exemple). Les processus d'émission font en effet intervenir un grand nombre de phénomènes physiques, fortement dynamiques, souvent corrélés et non linéaires, de sorte qu'une description purement analytique n'est pas envisageable.

Les réglages du moteur sont définis par un ensemble de tableaux de valeurs, appelées cartographies, et qui déterminent, en fonction des deux grandeurs fondamentales que sont le régime et la charge moteur, les valeurs que doivent prendre toutes les autres variables de contrôle. Entre deux valeurs discrètes de régime et de charge, la grandeur à définir est établie par interpolation linéaire à partir des valeurs présentes dans la cartographie.

Pour la mise au point du moteur, on procède donc actuellement par essais et erreurs, en contrôlant, sur un banc d'essai, les émissions qui résultent d'un jeu particulier de réglages cartographiques. Le savoir-faire du motoriste, associé à quelques règles empiriques, permet seul de déterminer les réglages finaux du moteur. Notons que les taux d'émissions polluantes ne sont pas les seuls critères à prendre en considération dans la phase de mise au point du moteur : il convient également de prêter attention aux performances du moteur, à son comportement dans les différents régimes de fonctionnement, à son bruit moyen, ainsi qu'à tout un ensemble de critères de qualité définis par le constructeur ou le législateur. La mise au point constitue donc un compromis délicat à réaliser, d'autant plus que peu de règles précises sont à disposition du motoriste relativement à ces différents critères. Cette démarche de mise au point requiert un grand nombre d'essais sur banc, dont le coût est très important.

On se propose donc d'élaborer par apprentissage un modèle permettant de prédire les débits de ces 5 familles de polluants en fonction des variables de contrôle moteur découlant des réglages cartographiques. L'intérêt d'un tel modèle serait de s'affranchir de la plupart des essais nécessaires à la mise au point du moteur : idéalement, seule une vérification finale, correspondant à un cycle d'essai, serait nécessaire après avoir déterminé les réglages optimaux sur la base du modèle de prédiction. Cette étude s'inscrit dans le projet appelé MAPAO (pour Mise Au Point Assisté par Ordinateur), piloté par Josselin Visconti et Yann Collette du service Optimisation (64240) de la Direction de la Recherche de Renault, et qui fait l'objet d'une collaboration avec Renault Trucks. L'objectif final est d'optimiser un critère de qualité prenant en considération l'ensemble des débits totaux sur cycle NMVEG, à partir des cartographies plutôt que des variables de contrôle du moteur.

5.2 Dispositifs expérimentaux et élaboration de la base de données

5.2.1 Mesure des émissions polluantes

La mesure des émissions polluantes s'effectue de deux manières, suivant que l'on s'intéresse au moteur seul ou à l'ensemble du véhicule (incluant les éléments de dépollution de la ligne d'échappements). On distingue donc :

- le banc haute dynamique (BHD), pour la mesure des émissions du moteur seul (Figure 11). Le pilotage du moteur, en termes de régime, de charge ou de rapport de boîte, s'effectue alors depuis une station de contrôle, où sont également recueillies l'ensemble des mesures d'état du moteur en fonctionnement. Une baie d'analyse

particulière permet la mesure des différents polluants émis. Les réglages cartographiques du moteur sont définis depuis la station de contrôle.

- le banc à rouleaux (BAR), qui permet de mesurer les émissions du véhicule complet (Figure 12). Un opérateur pilote alors le véhicule, tandis que les données (de contrôle ou de mesures des émissions polluantes) sont recueillies dans la station de contrôle. Les réglages cartographiques du moteur sont également définis depuis la station de contrôle, qui se substitue alors aux calculateurs embarqués du véhicule.

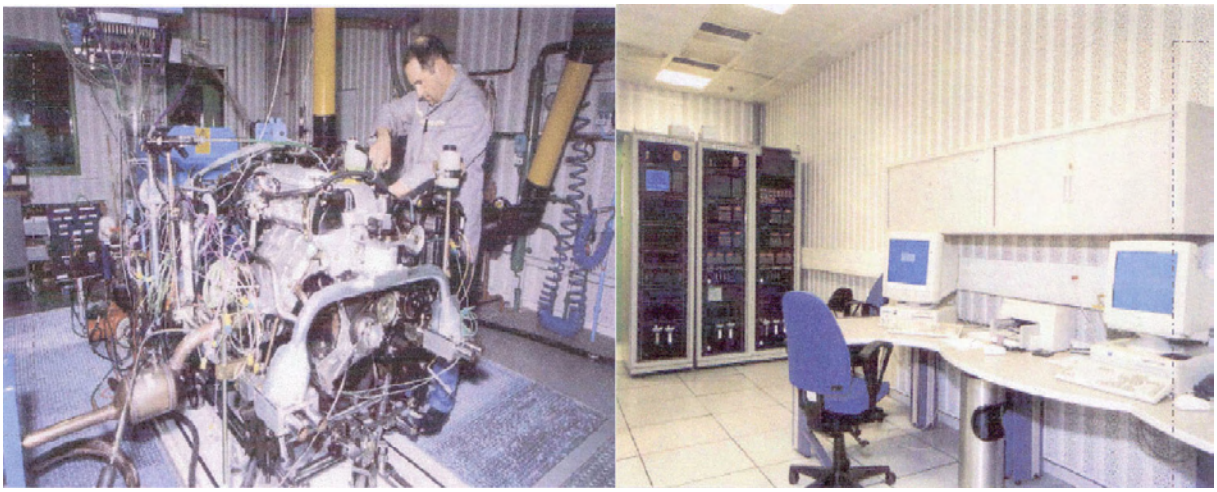


Figure 11 : *gauche* : banc haute dynamique (BHD) pour la mise au point moteur. *droite* : station de contrôle et baie d'analyse des gaz polluants.

L'homologation finale d'un véhicule, avant commercialisation, a nécessairement lieu sur un BAR, mais l'utilisation d'un BHD permet la mise au point du moteur, en phase de développement, indépendamment des autres organes du véhicule.

Figure 12 : *gauche* : banc à rouleaux (BAR) pour la mise au point véhicule. *droite* : baie d'analyse des gaz polluants.

Des conditions climatiques particulières peuvent être reproduites au sein de la zone d'essai : température ambiante, taux d'humidité de l'air, pression, vitesse de l'air ambiant, …

5.2.2 Choix des variables du modèle et élaboration de la base de données

Les connaissances des motoristes ont permis de mettre en évidence 9 variables de contrôle du moteur, dont l'influence sur les émissions polluantes est avérée :

1. Régime du moteur : exprimée en tours/min, cette variable mesure la vitesse de rotation du moteur.
2. EGR (« Exhaust Gaz Recirculation » ou recirculation des gaz d'échappements) : cette variable, exprimée en pourcentage d'ouverture de la vanne EGR, quantifie la proportion de gaz d'échappements réinjectés dans les cylindres.
3. Débit d'air : caractérise la quantité totale d'air (en mg/coup) injectée dans chaque cylindre avant explosion.
4. Débit de préinjection : cette variable décrit la quantité de carburant injectée dans la phase dite de préinjection, en mm^3/coup (l'injection du carburant étant constituée d'une préinjection et d'une injection principale).
5. Débit carburant total : cette variable décrit la quantité totale de carburant (exprimée en mm^3/coup) injectée dans les cylindres avant chaque explosion.
6. Avance principale : cette variable, exprimée en degrés vilebrequin (360° vilebrequin correspondant à un cycle complet du moteur), détermine l'instant auquel l'injection principale a lieu dans les cylindres.
7. Phasage : cette variable, exprimée en degrés vilebrequin, détermine l'intervalle entre l'injection principale et la préinjection.
8. Pression de suralimentation : exprimée en mbar, cette quantité caractérise la surpression à laquelle l'air sera injecté dans les cylindres.
9. Pression rail : cette variable (exprimée en bar) est propre au moteur diesel à rampe commune d'injection, et décrit la pression à laquelle le carburant est injecté directement dans les cylindres.

A chacune de ces variables est associée une cartographie (excepté pour le régime moteur), qui déterminera la valeur qu'elle doit prendre en fonction du régime et de la charge du moteur à l'instant considéré. La Figure 13 donne une représentation des différents organes du moteur influencés par les variables de contrôle énoncées ci-dessus. Le moteur sur lequel les mesures seront effectuées est de type diesel à rampe commune d'injection (moteur Renault F9Q/750, cat. 2 : 1.9L DCi, 120ch).

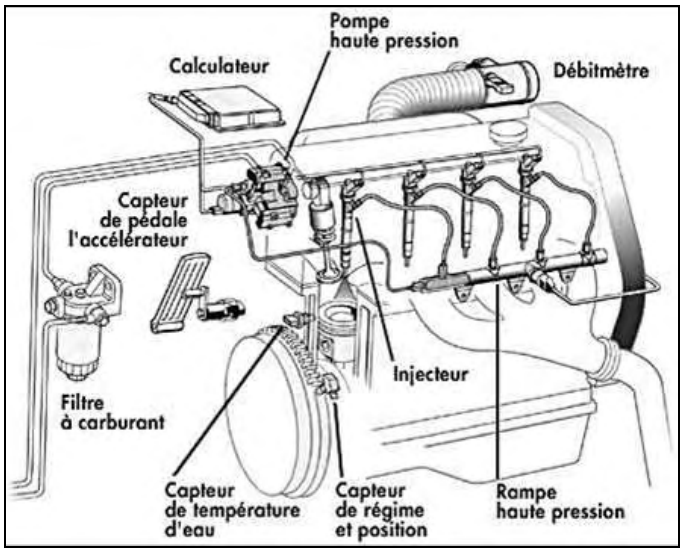


Figure 13 : quelques organes du moteur influencés par les variables de contrôle utilisées. Le calculateur, où sont stockées les cartographies, gère l'ensemble de ces variables, en fonction du régime du moteur (mesuré par un capteur) et de la charge du moteur (mesurée par la position de la pédale).

Pour réaliser un modèle d'aide à la mise au point vis-à-vis des émissions polluantes, il est nécessaire de couvrir tout le domaine de fonctionnement des variables de contrôle, en faisant varier les cartographies sous-jacentes. Rappelons que notre objectif n'est pas de prédire les émissions polluantes pour une situation de roulage quelconque avec un jeu de cartographies fixé, mais de les prédire pour une situation de roulage particulière (le cycle NMVEG) en fonction de réglages cartographiques variables. La base de données a donc été élaborée à partir de cycles NMVEG, en perturbant les réglages cartographiques du moteur autour de valeurs de référence, par ajout d'un décalage global sur l'ensemble de la cartographie relative à l'une des variables de contrôle. On a ainsi obtenu un total de 6 cycles NMVEG (cf. Tableau 4). Chaque cycle, d'une durée d'environ 20min, comprend approximativement 1200 points (échantillonnés à 1Hz).

Cycle NMVEG	Variable perturbée	Valeur du décalage	Commentaire
1 (référence)	(aucune)	(aucun)	Validation
2	Pression rail	+100 bar	Apprentissage
3	Avance	+5°	Apprentissage
4	Phasage	-5°	Apprentissage
5	Débit de préinjection	+0.5 mg/coup	Apprentissage
6	EGR	-50 %	Apprentissage

Tableau 4 : bases de données constituées de cycles NMVEG réalisés avec offset global d'une cartographie moteur particulière.

La manière dont a été élaborée la base de données n'est pas optimale pour la réalisation de modèles par apprentissage (utilisation à laquelle cette base n'était pas destinée initialement) : l'ajout d'un décalage global sur la cartographie d'une variable particulière renseigne en effet sur son influence, mais ne fournit que peu d'informations. Une étude sur la façon de créer des réglages cartographiques dédiés à l'apprentissage est actuellement en cours : l'idée serait de partir d'une cartographie de référence, dont chacun des points serait ensuite perturbé de manière aléatoire, tout en s'assurant que la perturbation reste admissible pour le moteur (en termes de valeur maximale et de variation d'un point à ses voisins). En procédant de cette façon, il serait a priori possible de réduire la quantité de mesures nécessaires pour couvrir convenablement le domaine de fonctionnement du moteur (sur le cycle NMVEG) et les domaines de définition des variables de contrôle. La mise en œuvre des développements récents sur la planification d'expériences *D*-optimale pour modèles non linéaires (neuronaux ou de connaissances) serait certainement envisageable [15].

5.3 Elaboration du modèle neuronal

Des études préliminaires, consacrées à la modélisation des émissions polluantes au moyen de réseaux de neurones statiques de type Perceptron, ont été menées à la Direction de la Recherche de Renault. L'hypothèse sur laquelle était fondée cette modélisation s'écrit :

$$y(k) = \Phi\left(u_1(k-1),\ldots,u_1(k-m_1),u_2(k-1),\ldots,u_2(k-m_2),\ldots,u_9(k-1),\ldots,u_9(k-m_6)\right) + w(k) \quad (20)$$

où :

$u_i(k)$ représente la *i*-ème entrée de commande, caractérisée par un horizon temporel entier m_i,

Φ est une fonction non linéaire,

$y(k)$ est la sortie du processus (le débit de l'un des polluants étudiés),

$w(k)$ est un bruit de sortie.

Une telle approche suppose que les seules dynamiques présentes dans le processus interviennent sur les variables d'entrées, et qu'aucune équation aux différences ne caractérise l'évolution temporelle de la sortie. Compte tenu de la nature des phénomènes physiques à l'origine des émissions polluantes, et notamment l'aspect cinétique des réactions chimiques lors de la combustion, il semble peu probable qu'une telle hypothèse soit vérifiée. Les résultats obtenus par cette approche restent en effet d'une qualité moyenne, en particulier vis-à-vis de l'erreur instantanée maximale qui s'avère très importante.

Nous nous sommes donc intéressés dans un deuxième temps à des modèles neuronaux récurrents, permettant la prise en considération de la dynamique de la sortie du processus. L'hypothèse proposée s'écrit :

$$\begin{cases} \mathbf{x}(k) = \mathbf{\Phi}\left(\mathbf{x}(k-1), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-m)\right) + \mathbf{w}_1(k) \\ y(k) = \Psi\left(\mathbf{x}(k)\right) + w_2(k) \end{cases} \tag{21}$$

où :

$\mathbf{\Phi}$ et Ψ sont deux fonctions non linéaires paramétrées,

$\mathbf{x}(k)$ est le vecteur d'état (dont la dimension caractérise l'ordre du modèle) à l'instant discret k,

$\mathbf{u}(k)$ est le vecteur des 9 entrées de commande du modèle,

$\mathbf{w}_1(k)$ et $w_2(k)$ sont respectivement des bruits d'état et de sortie,

m est l'horizon temporel maximal à prendre en compte pour ces 9 variables.

En faisant l'hypothèse d'un bruit de sortie et d'un bruit d'état (modèle NARMAX), il convient d'élaborer, pour chaque polluant étudié, un prédicteur bouclé, en utilisant l'algorithme d'apprentissage semi dirigé et la méthode de BFGS.

Nous avons procédé par validation croisée [10], en découpant la base de données initiale en une série de 6 bases distinctes, chacune étant constituée d'un cycle NMVEG complet. Pour une structure donnée du réseau de neurones, on réalise alors 6 apprentissages successifs, en excluant à tour de rôle de la base d'apprentissage l'un de ces cycles, qui servira de base de validation. Du fait que chaque base d'apprentissage comprend alors plusieurs séquences temporelles disjointes (les 5 cycles NMVEG sélectionnés), le gradient total vis-à-vis de l'un des paramètres du réseau a été calculé comme la somme des gradients rétropropagés au sein de chacune de ces 5 séquences.

Il est délicat de déterminer a priori le meilleur ordre pour le modèle, afin de bien modéliser toutes les dynamiques présentes dans le processus. Nous avons donc procédé par essais et erreurs, en faisant varier l'ordre du modèle (entre 1 et 4) et la mémoire sur chacune des variables de commande (entre 1 et 3).

La structure choisie pour le réseau de neurones est du même type que celle employée pour la modélisation de la température après turbine (cf. Figure 7) : il s'agit d'un réseau récurrent, utilisant une représentation d'état, et dont la partie acyclique est un Perceptron à une couche cachée avec des neurones de sortie linéaires. Les grandeurs qui déterminent la complexité du modèle sont donc le nombre d'entrées d'état (N_s) et le nombre de neurones cachés (N_{cc}).

5.4 Résultats et discussion

En appliquant la méthodologie décrite dans le paragraphe précédent, nous avons élaboré des modèles dynamiques des émissions de deux des polluants les plus délicats à modéliser : les oxydes d'azote NO_x et les particules de suies PM. Il s'agissait principalement d'étudier la faisabilité d'une telle modélisation, à partir de la base de données dont nous disposions, qui, rappelons-le, n'avait pas été conçue pour servir de base d'apprentissage. Les critères utilisés pour caractériser la qualité du modèle sont la racine carrée de l'erreur quadratique moyenne (REQM), l'erreur relative sur le cumul (ERC), et le coefficient de corrélation R^2, définis pour la régression de la façon suivante :

$$R^2 = 1 - \frac{\sigma^2_{erreur}}{\sigma^2_{mesures}} = 1 - \frac{\sum_{k=1}^{N}\left(\left(y^p(k) - y(k)\right) - \left(\overline{y}^p - \overline{y}\right)\right)^2}{\sum_{k=1}^{N}\left(y^p(k) - \overline{y}^p\right)^2}$$

$$ERC = \frac{\left|\sum_{k=1}^{N}\left(y^p(k) - y(k)\right)\right|}{\sum_{k=1}^{N} y^p(k)}$$

où :

$y^p(k)$ représente la mesure de la sortie à l'instant discret k,

$y(k)$ est son estimation,

$\overline{y}^p$ représente la moyenne de la sortie mesurée sur tous les instants,

$\overline{y}$ représente la moyenne de la sortie estimée sur tous les instants.

Le coefficient adimensionnel R^2 caractérise la partie expliquée de la variation de la sortie : le modèle sera d'autant meilleur que R^2 sera proche de 1. L'erreur relative sur le cumul est prise en considération pour caractériser la qualité du modèle, dans la mesure où le critère d'homologation d'un véhicule vis-à-vis des émissions polluantes porte sur la quantité moyenne d'émissions sur un cycle NMVEG : l'ERC fournit alors une indication importante en vue de l'homologation. Signalons d'autre part que l'ERC est moins sensible au bruit instantané de mesure, du fait que cette quantité moyenne sur l'ensemble du cycle les réalisations particulières du bruit, qui par hypothèse est de moyenne nulle.

Le Tableau 5 présente les résultats obtenus pour la modélisation des débits de PM et de NO_x. Du fait des caractéristiques de la base de données (décalages successifs sur l'une des variables de commande, taille réduite), nous avons choisi de ne pas conserver de base de test dans la

procédure d'apprentissage par validation croisée. Les résultats présentés font apparaître les trois critères de performance retenus (REQM, R^2 et ERC) en terme d'apprentissage (moyenne sur les 6 bases d'apprentissage successives) et de validation (moyenne sur les 6 bases de validation successives).

	Structure	**Apprentissage**	**Validation**
PM	$N_s = 3$ $N_{cc} = 7$ aucun retard	REQM = $8.5 \cdot 10^{-2}$ mg/s $R^2 = 0.96$ ERC = 4.4%	REQM = $9.8 \cdot 10^{-2}$ mg/s $R^2 = 0.95$ ERC = 7.2%
NO_x	$N_s = 3$ $N_{cc} = 6$ aucun retard	REQM = 1.7 mg/s $R^2 = 0.91$ ERC = 2.9%	REQM = 2.0 mg/s $R^2 = 0.93$ ERC = 2.5%

Tableau 5 : modélisation par réseau de neurones récurrents des débits de particules de suies (PM) et d'oxydes d'azote (NO_x). Apprentissage par validation croisée (découpage en 6 bases). Paramètres de structure des modèles, et racine carrée de l'erreur quadratique moyenne (REQM), coefficient de corrélation (R^2), et erreur relative sur le cumul (ERC) moyennés sur les 6 apprentissages et les 6 validations.

On constate que les coefficients de corrélation R^2 présentent, pour les PM ainsi que pour les NO_x, des valeurs proches de 1, ce qui traduit une bonne qualité des estimations instantanées. Le critère d'erreur relative sur le cumul assure d'autre part une estimation satisfaisante des quantités totales de polluants émises sur un cycle NMVEG. Les valeurs des REQM de validation sont à comparer aux écarts types du bruit de mesure pour ces polluants. Sachant que ces mesures sont entachés d'un bruit dont l'écart type est de 5% pour les NO_x et de 18% pour les PM, on calcule, sur l'ensemble des 6 bases de validation employées pour la validation croisée, des écarts types égaux à : $9.9 \cdot 10^{-2}$ mg/s pour les PM (à comparer à la REQM de validation de $9.8 \cdot 10^{-2}$ mg/s) et à 1.7 mg/s pour les NO_x (à comparer à la REQM de validation de 2.0 mg/s).

La Figure 14 présente, à titre d'exemple, les mesures et les estimations du débit de PM, ainsi que l'erreur de modélisation correspondante, sur un cycle NMVEG de validation. Le même type de résultats est présenté sur la Figure 15 pour les NO_x.

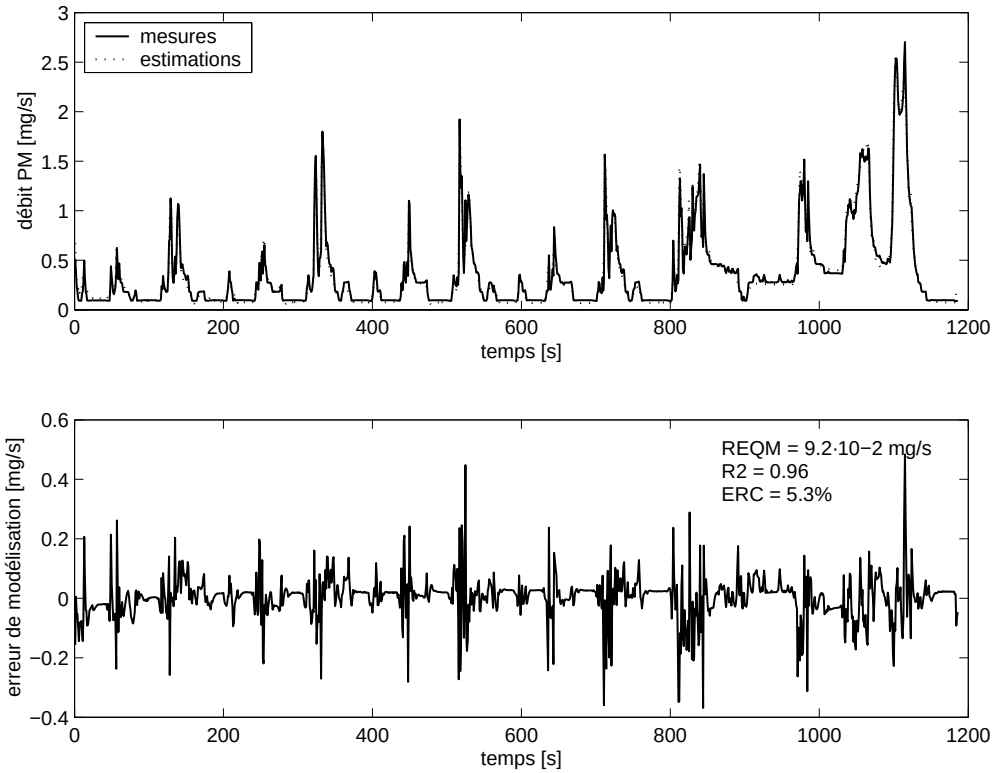


Figure 14 : mesures et estimations du débit de particules de suies (PM), et erreur de modélisation correspondante, sur un cycle NMVEG de validation.

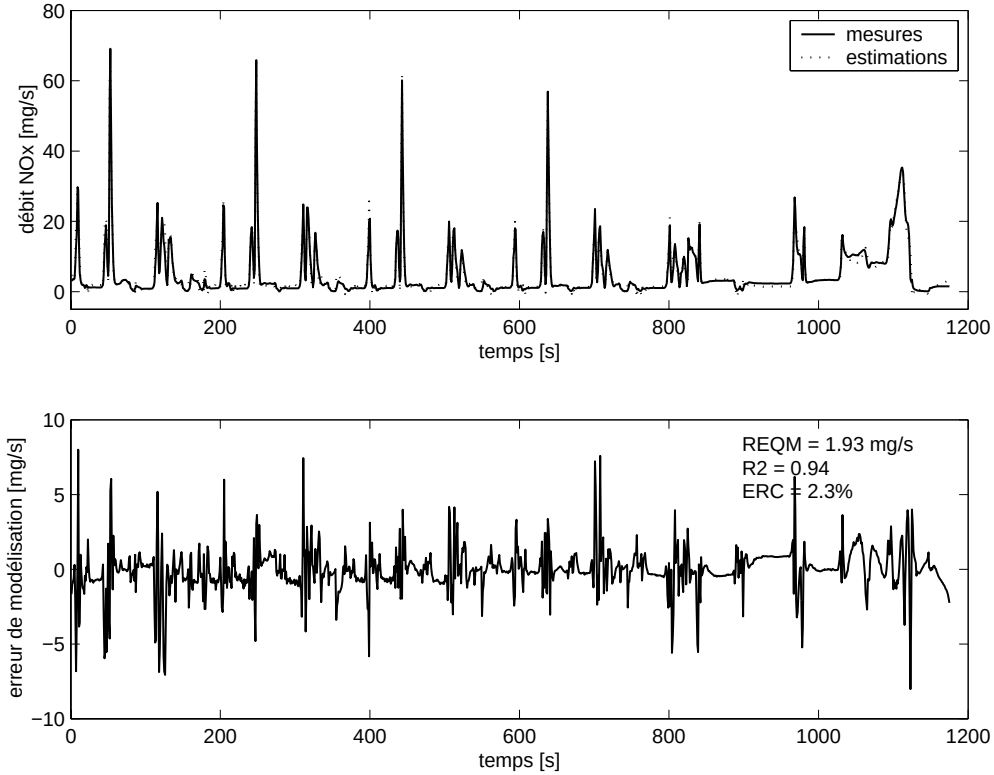


Figure 15 : mesures et estimations du débit d'oxydes d'azote (NO_x), et erreur de modélisation correspondante, sur un cycle NMVEG de validation.

5.5 Conclusion

Dans le cadre du projet de Mise au Point Assistée par Ordinateur (MAPAO), nous avons mis en évidence la possibilité d'estimer sur cycle NMVEG les débits des principaux polluants réglementés en fonction de 9 variables de contrôle moteur. Les résultats obtenus sur deux familles de polluants (les NO_x et les PM) par des modèles neuronaux récurrents présentent une qualité satisfaisante tant vis-à-vis de l'erreur instantanée que vis-à-vis de l'erreur relative sur le cumul.

Une étude comparative de quelques outils de modélisation (réseaux de neurones statiques, réseaux de neurones récurrents, et Locally Linear Model Trees (LOLIMOT) dont le principe est donné dans [16]) avait d'autre part permis de conclure que les réseaux de neurones récurrents et les LOLIMOT semblent les mieux adaptés à ce type de modélisation, en terme de performance et de facilité de mise en oeuvre. De nouvelles campagnes d'essais s'intéressent par ailleurs à la réalisation de bases de données spécifiques pour l'apprentissage de modèles boîte noire.

La prochaine étape concernera l'optimisation, sur la base des modèles prédictifs élaborés, des réglages cartographiques du moteur vis-à-vis des quantités totales de polluants émises sur cycle NMVEG. Notons que des travaux préliminaires, menés en collaboration avec Renault Trucks, ont montré la faisabilité d'une telle optimisation (à partir de modèles par réseaux de neurones récurrents) pour des moteurs de camions, dont les stratégies de contrôle sont sensiblement similaires à celles des véhicules particuliers.

6 Système de commande neuronale prédictive en boucle ouverte

6.1 Introduction

La commande prédictive est largement utilisée dans le domaine industriel pour contrôler le comportement de processus complexes [17][18][19][20]. Son principe général, représenté sur la Figure 16, consiste à s'appuyer sur le modèle du processus pour déterminer par anticipation la valeur optimale des entrées de commande à appliquer, au sens d'un critère de performance préalablement défini qui caractérise le but que l'on souhaite atteindre (régulation, poursuite, ou tout autre critère de qualité).

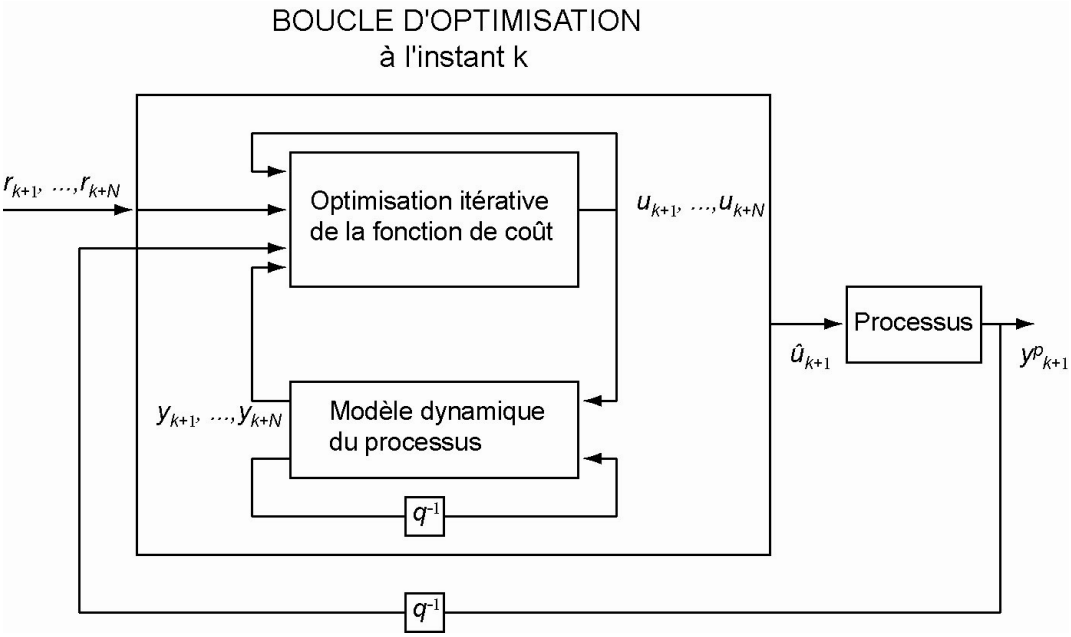


Figure 16 : schéma de principe de la commande optimale prédictive.

Dans cette partie, nous proposons une manière d'appliquer une stratégie de commande analogue à des processus dynamiques modélisés par un réseau de neurones récurrent. L'utilisation d'algorithmes d'optimisation du deuxième ordre, qui nécessitent le calcul de la matrice hessienne de la fonction de coût, nous a amené à introduire une méthodologie pour le calcul exact de cette matrice, dans le cas général de modèles récurrents mis sous forme canonique.

Nous illustrerons l'intérêt de cette méthode sur un processus dynamique simulé, dont on cherchera à minimiser les variations rapides de la sortie, en définissant une fonction de coût adéquate. Ce processus oscillant vise à reproduire qualitativement les à-coups de couple d'un véhicule, consécutifs à une variation rapide de l'enfoncement de la pédale d'accélérateur : pour réduire ces oscillations, qui détériorent de façon importante le confort de conduite, un système de commande prédictive du même type pourrait être mis en œuvre.

6.2 Description du système de commande

Notre objectif est d'élaborer un système de commande qui minimise les variations rapides de la sortie y^p d'un processus en réponse à une modification de son entrée de commande u, sans modifier la valeur de cette commande de manière trop importante. Pour poursuivre l'analogie automobile, il s'agit, connaissant la demande de couple voulue par le conducteur (via la

position de la pédale d'accélérateur), de déterminer une nouvelle valeur qui permette une atténuation suffisante des à-coups, sans trop différer de la valeur définie par le conducteur.

Nous supposons, dans un premier temps, que nous disposons d'un modèle dynamique à temps discret du processus à commander. La méthode proposée consiste à remplacer, à chaque instant *k*, la valeur de référence r_{k+1} de cette entrée de commande à venir, telle que définie par l'utilisateur, par une nouvelle valeur $\hat{u}_{k+1}$ qui minimise une fonction de coût adéquate *J*. Du fait du caractère dynamique du processus, la valeur de la commande à l'instant *k* aura une influence sur l'ensemble des valeurs futures de la sortie. Ceci nous amène à considérer non seulement la commande de référence à l'instant suivant r_{k+1}, mais la séquence des commandes de référence à venir $\{r_{k+1},\ldots,r_{k+L}\}$ sur un horizon temporel *L* plus grand que 1 (le calcul étant effectué à chaque instant *k*, on suppose que l'on a accès par anticipation aux valeurs des commandes de référence sur un horizon temporel *L*, ce qui dans la pratique se traduit par un délai entre l'instant où sont définies par l'opérateur les commandes de référence, et le moment d'application de ces commandes au processus). La fonction de coût $J(u_{k+1},\ldots,u_{k+L})$ à optimiser à chaque instant est donc définie comme la somme de deux termes, le premier pénalisant les variations importantes de la sortie, tandis que le second tend à conserver des signaux de commande proches de leur valeur de référence :

$$J(u_{k+1},\ldots,u_{k+L}) = \frac{1}{2}\sum_{n=1}^{L}(y_{k+n} - y_{k+n-1})^2 + \frac{1}{2}\sum_{n=1}^{L}\rho(n)(u_{k+n} - r_{k+n})^2 \qquad (22)$$

où :

$\{u_{k+1},\ldots,u_{k+L}\}$ est la séquence des entrées de commande par rapport à laquelle la fonction de coût *J* sera optimisée à l'instant discret *k*,

y_{k+n} est la sortie estimée par le modèle dynamique à l'instant $k+n$,

r_{k+n} est la commande de référence au même instant,

$\rho(n)$ est un réel qui détermine le compromis entre la diminution des variations de la sortie et l'écart entre valeurs de référence et valeurs optimisées de la commande.

Il faut noter que l'élaboration de la séquence de commande ne prend pas en considération la sortie y^p du processus : il s'agit donc ici d'une commande en boucle ouverte.

Le schéma de principe de ce système de commande est représenté sur la Figure 17, où l'on observe que, à la différence de la Figure 16, la sortie du processus n'intervient pas dans l'élaboration de la commande.

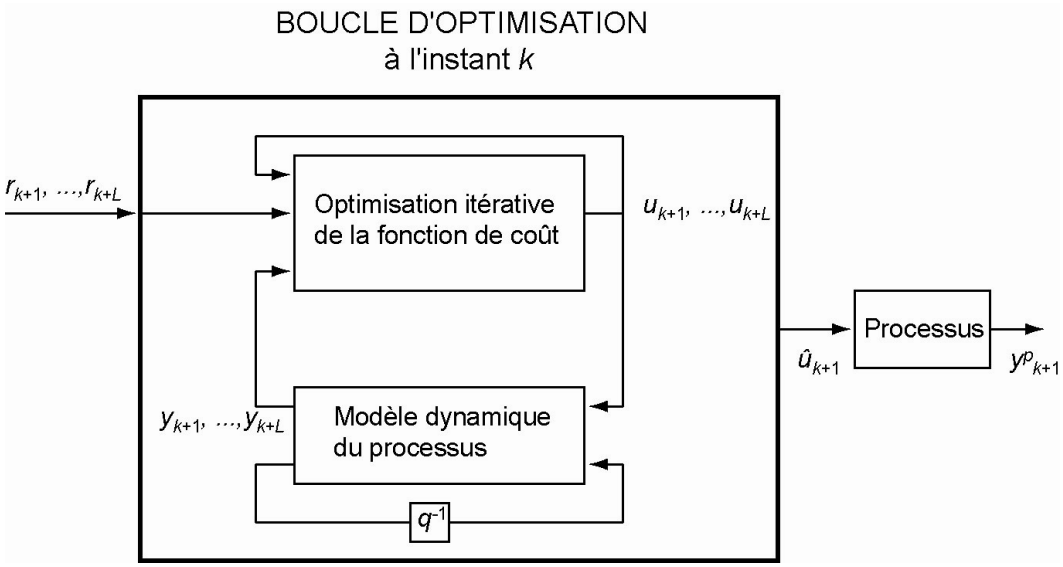


Figure 17 : schéma de principe du système de commande étudié : à l'instant *k*, on réalise une optimisation itérative de la fonction de coût par rapport à la séquence des entrées de commande $\{u_{k+1},\ldots,u_{k+L}\}$, pour obtenir une séquence optimale $\{\hat{u}_{k+1},\ldots,\hat{u}_{k+L}\}$ dont seul le premier élément $\hat{u}_{k+1}$ est appliqué au processus.

6.3 Optimisation de la fonction de coût

Dans le cas d'un modèle non linéaire vis-à-vis des entrées de commande, et en particulier pour des modèles neuronaux, la fonction de coût *J* n'est pas quadratique. On est alors amené à mettre en œuvre des algorithmes d'optimisation itératifs. Nous utiliserons l'algorithme du second ordre de Levenberg-Marquardt, dont la formule de mise à jour à l'itération *n* s'écrit :

$$\mathbf{u}_{(n)} = \mathbf{u}_{(n-1)} - \left[\frac{\partial^2 J}{\partial \mathbf{u}_{(n-1)}^T \partial \mathbf{u}_{(n-1)}} + \mu \mathbf{I} \right]^{-1} \frac{\partial J}{\partial \mathbf{u}_{(n-1)}} \tag{23}$$

où $\mathbf{u} = \left[u_{k+1}, u_{k+2}, ..., u_{k+L}\right]^T$, **I** est la matrice identité, et où μ est le paramètre de Levenberg-Marquardt. L'utilisation d'un algorithme du second ordre assure une convergence rapide vers le minimum, mais requiert le calcul de la matrice hessienne de *J* vis-à-vis des valeurs de commande $u_{k+1}, u_{k+2}, ..., u_{k+L}$. Une méthode générale de calcul de cette matrice, dans le cas de modèles dynamiques, est proposée en Annexe [21]. Cette méthode s'applique à la classe des modèles décrits par des équations récurrentes du type :

$$z_i(k+1) = g_i\left(\left\{z_j\left(k-\tau_{ij,m}+1\right)\right\}, \left\{u_l\left(k-\tau_{il,m'}+1\right)\right\}\right),\ i,j = 1 \text{ to } M,\ l = 1 \text{ to } M',\ m,m' > 0 \tag{24}$$

où g_i est une fonction quelconque, $\tau_{ij,m}$ est un entier caractérisant le retard à prendre en considération sur la variable z_j intervenant dans le calcul de $z_i(k+1)$, et $\tau_{il,m'}$ est le retard entier de la *m'*-ième valeur retardée de la variable u_l intervenant dans le calcul de $z_i(k+1)$. On peut

montrer [3] qu'un tel modèle admet une représentation d'état minimale, appelée forme canonique, décrite par les équations suivantes :

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{\Phi}\left(\mathbf{x}(k), \mathbf{u}(k+1)\right) \\ y(k+1) = \Psi\left(\mathbf{x}(k+1), \mathbf{u}(k+1)\right) \end{cases} \tag{25}$$

où $\mathbf{x}(k)$ est le vecteur d'état à l'instant k, $\mathbf{u}(k)$ est le vecteur des entrées de commande au même instant, $y(k)$ est la sortie du modèle, et où $\mathbf{\Phi}$ et Ψ sont deux fonctions déduites des fonctions g_i de la relation (24). Nous avons adopté cette représentation canonique pour le calcul exact de la matrice Hessienne de la fonction de coût J par rapport aux valeurs des entrées de commande.

Il devient ainsi possible de mettre en œuvre l'algorithme de Levenberg-Marquardt. En début d'optimisation, les entrées de commande à optimiser $u_{k+1}, u_{k+2}, ..., u_{k+L}$ sont initialisées aux valeurs des entrées de référence $r_{k+1}, r_{k+2}, ..., r_{k+L}$. Seule la première des commandes optimisées $\hat{u}_{k+1}$ sera appliquée au processus à l'instant k+1, avant de réaliser une nouvelle optimisation à cet instant sur la nouvelle séquence de référence $r_{k+2}, r_{k+3}, ..., r_{k+L+1}$. Cette démarche suppose évidemment que l'on ait accès par anticipation aux valeurs de référence sur un horizon temporel L.

6.4 Illustration sur un processus simulé

6.4.1 Description du processus et élaboration d'un modèle neuronal récurrent

Nous allons à présent appliquer l'ensemble de la procédure décrite précédemment à un processus dynamique simulé, décrit par les équations aux différences suivantes :

$$\begin{cases} x_k^p = f\left(x_{k-1}^p, x_{k-2}^p, u_k\right) = \dfrac{24 + x_{k-1}^p}{30} x_{k-1}^p - 0.8 \dfrac{u_k^2}{1+u_k^2} x_{k-2}^p + 0.5u_k \\ y_k^p = x_k^p + e_k \end{cases} \tag{26}$$

où u_k est l'entrée de commande à l'instant discret k, y_k^p est la sortie au même instant, et e_k est un bruit de sortie additif gaussien d'écart type σ_{bruit}=0,1.

La réalisation d'un système de commande optimale requiert, dans un premier temps, l'élaboration d'un modèle de ce processus, que nous réaliserons au moyen d'un réseau de neurones récurrent basé sur une représentation d'état (Figure 7). Deux séquences de données ont été créées (Figure 18), par application des relations (26).

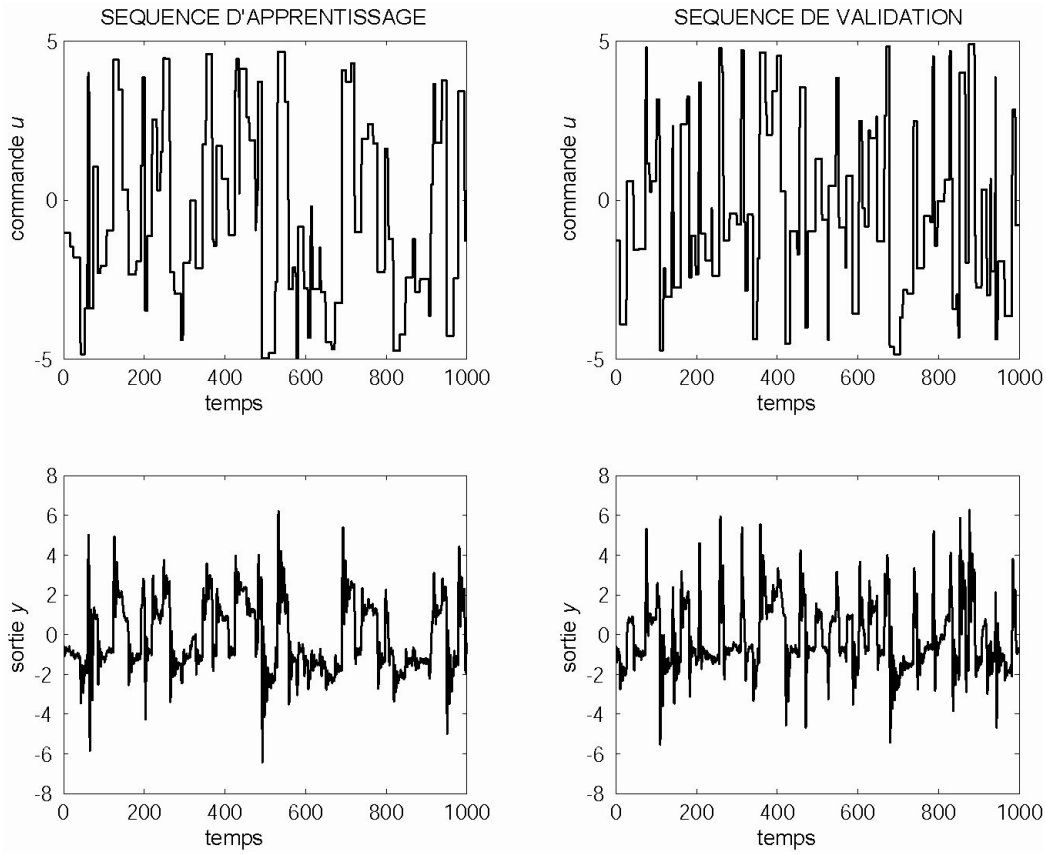


Figure 18 : processus simulé : séquence d'apprentissage (gauche) et de validation (droite)

La meilleure performance a été obtenue avec un réseau de neurones récurrent à 2 entrées d'état (modèle d'ordre 2) et 6 neurones cachés, issu d'un apprentissage semi-dirigé puisque le processus est soumis à un bruit de sortie. Les erreurs commises par le modèle neuronal sur les séquences d'apprentissage et de validation sont indiquées dans le Tableau 6.

	SEQUENCE D'APPRENTISSAGE	**SEQUENCE DE VALIDATION**
REQM	$9.5\ 10^{-2}$	$1.1\ 10^{-1}$

Tableau 6 : racine carrée de l'erreur quadratique moyenne (REQM) commise par le modèle neuronal sur les séquences d'apprentissage et de validation. Ces erreurs sont de l'ordre de grandeur de l'écart type du bruit de sortie σ_{bruit}=0.1, ce qui prouve que le modèle est proche du prédicteur idéal.

On constate que les erreurs commises par le modèle sont du même ordre de grandeur que le bruit additif de sortie : l'apprentissage est donc convenable, en ce qu'il a permis d'expliquer la partie déterministe contenues dans les séquences de mesures. D'autre part, le fait que les erreurs sur les bases d'apprentissage et de validation soient également du même ordre de grandeur suggère une bonne capacité de généralisation du modèle.

6.4.2 Commande prédictive en boucle ouverte du processus simulé

Nous allons à présent mettre en œuvre la procédure de commande prédictive introduite précédemment, sur la base du modèle neuronal ainsi élaboré. Dans un premier temps, nous avons appliqué la procédure sur la séquence de données de validation, avec un paramètre constant ρ = 2, et sur un horizon L = 10. Les entrées de commande résultantes sont représentées sur la Figure 19. Nous avons également porté l'erreur commise par le modèle neuronal sur cette séquence optimisée : en valeur quadratique moyenne, celle-ci reste du même ordre de grandeur que lors de l'apprentissage, ce qui confirme la qualité du modèle et l'adéquation de la méthode sur cet exemple.

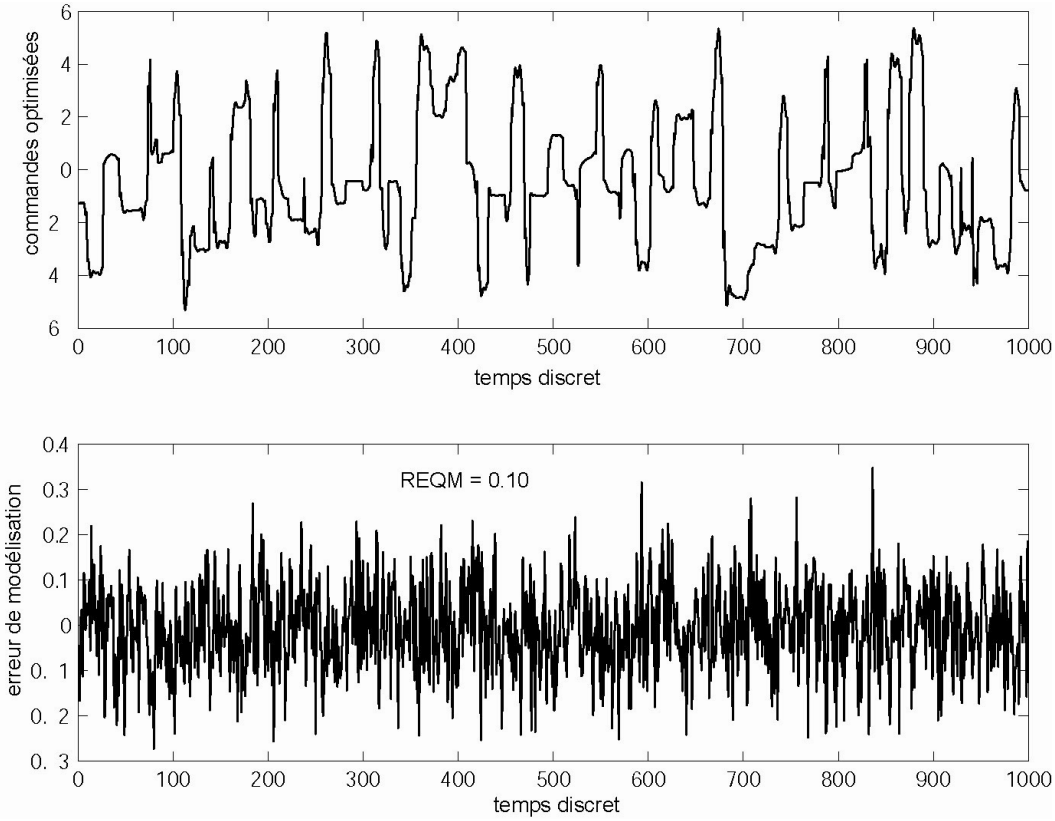


Figure 19 : *en haut :* séquence de commandes optimisées (L=10, ρ=2) sur la base de validation. *en bas* : erreur commise par le modèle neuronal sur cette séquence optimisée.

Afin d'étudier l'influence de l'horizon de prédiction L sur le résultat de l'optimisation, nous avons réalisé une série d'optimisations en faisant varier la valeur de cet horizon entre 1 et 10 (avec un paramètre constant ρ = 2) ; pour chacune des séquences résultantes, un facteur de qualité Q a été calculé. Sa définition est voisine de celle de la fonction de coût optimisée J :

$$Q = \frac{1}{2}\sum_{k=1}^{N-1}\left(y_{k+1}^{p} - y_{k}^{p}\right)^2 + \frac{1}{2}\rho\sum_{k=1}^{N}\left(\hat{u}_k - r_k\right)^2 \qquad (27)$$

où N = 1000 est la taille de la séquence étudiée.

Ce facteur de qualité caractérise donc le résultat de l'optimisation à l'échelle de la séquence complète : le compromis atténuation des variations rapides / pénalisation des écarts à la référence est d'autant meilleur sur l'ensemble de la séquence que Q est petit. La Figure 20 présente le résultat de cette étude (la valeur $L = 0$ correspondant aux signaux de référence sans optimisation).

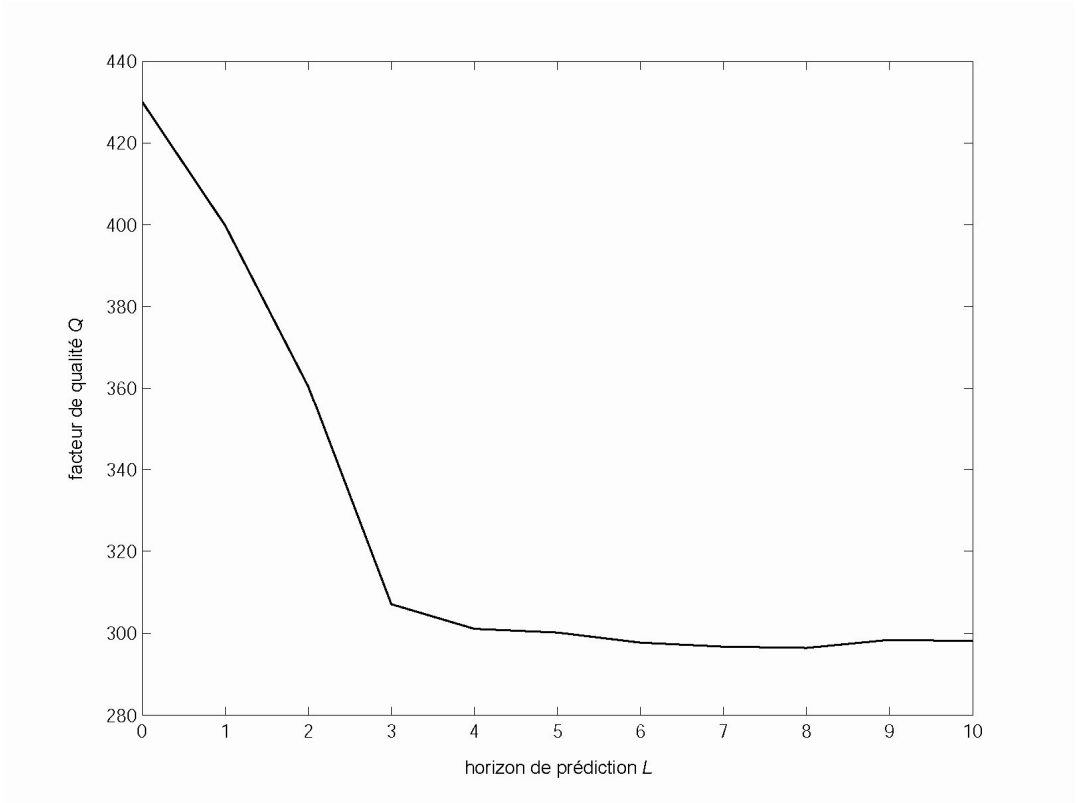


Figure 20 : facteur de qualité sur l'ensemble de la séquence en fonction de l'horizon de prédiction L (optimisation réalisées à partir de la base de validation avec une valeur constante ρ=2).

Il apparaît donc qu'en augmentant la valeur de l'horizon de prédiction L, l'optimisation résultante devient plus « efficace », bien que l'amélioration devienne marginale au-delà de L=3. D'un point de vue qualitatif, le fait de travailler avec un horizon L>1 permet d'anticiper les variations importantes de la sortie du processus, et la compensation peut commencer d'autant plus tôt que l'horizon est grand. Cet effet d'anticipation des variations de la sortie est illustré sur la Figure 21, où sont représentés les signaux de commande optimisés et les sorties du processus correspondantes, pour des valeurs de l'horizon de prédiction comprises entre 1 et 3, autour d'une zone de variation importante.

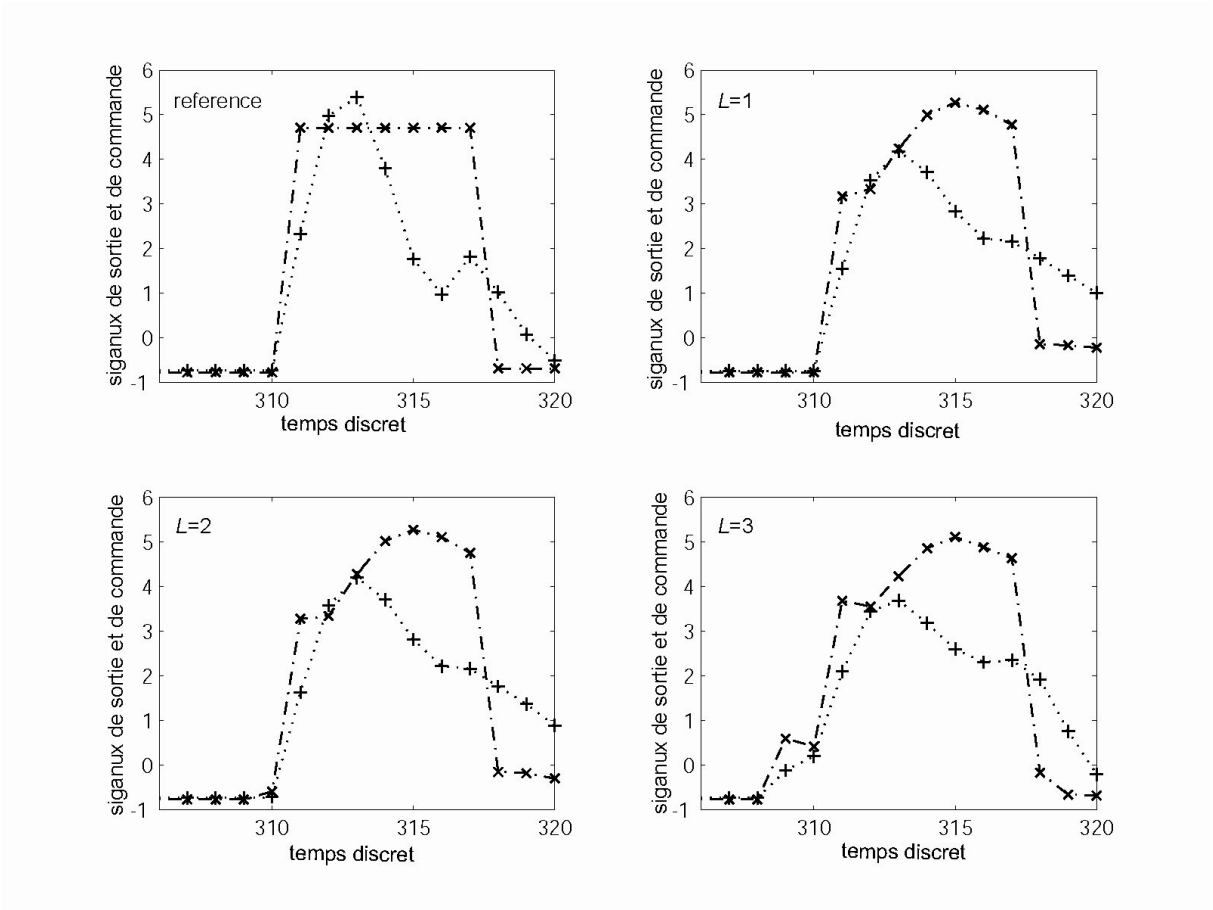


Figure 21 : signaux de commande optimisés (en trait mixte) et sorties du processus correspondantes (en pointillés), pour des valeurs de *L* comprise entre 0 et 3.

On observe clairement sur cette figure l'effet d'anticipation des signaux de commande : sans optimisation, l'augmentation importante de la commande de référence à l'instant 310 induit des variations rapides de la sortie du processus. Pour L=3, on note une légère augmentation de la commande optimisée dès l'instant 308, qui a pour effet de diminuer les variations de la sortie. D'autre part, l'horizon de prédiction a une influence sur la valeur maximale prise par la sortie, la faisant passer de 5,4 sans optimisation à 3,7 pour L=3.

6.5 Conclusion

Nous avons présenté dans ce paragraphe un système de commande prédictive en boucle ouverte, destiné à atténuer les variations rapides de la sortie d'un processus dynamique. L'ensemble de la démarche de conception (modélisation par réseau de neurones récurrent du processus, optimisation du second ordre sur une horizon L>1) a été illustré sur un processus simulé, mettant en évidence l'intérêt d'une prédiction sur un horizon temporel supérieur à 1. Cette méthode pourrait être appliquée, dans le domaine automobile, afin de diminuer les à-coups de couple d'un véhicule.

Nous avons parallèlement été amenés à concevoir une méthode de calcul exact de la matrice Hessienne de la fonction de coût, dans le cas de modèles décrits par des équations récurrentes, méthode qui est susceptible de s'appliquer à une large classe de modèles dynamiques, ainsi que pour la commande en boucle fermée. Il semble possible, par extension, d'adopter une approche similaire pour calculer de manière exacte la matrice hessienne de la fonction de coût des moindres carrés vis-à-vis des paramètres du réseau, pour une utilisation lors de la phase d'apprentissage de modèles neuronaux récurrents.

Bibliographie

[1] G. Cybenko. *Approximation by superposition of a sigmoidal function.* Mathematics of Control, Signals and Systems, Vol. 2, pp. 961-1005, 1990.

[2] K. Hornik, M. Stinchcombe, H. White. *Multilayer feedforward networks are universal approximators.* Neural Networks 2, pp. 359-366, 1989.

[3] G. Dreyfus, Y. Idan. *The canonical form of discrete-time non-linear models.* Neural Computation, Vol. 10, No. 1, pp. 133-164, 1998.

[4] G. Dreyfus, J.M. Martinez, M. Samuelides, M.B. Gordon, F. Badran, S. Thiria, L. Hérault. *Réseaux de neurones: méthodologie et applications, 2ème edition*. Eyrolles, 2004.

[5] M.I. Jordan. *The learning of representations for sequential performance.* Thèse de doctorat, University of California, San Diego, 1985.

[6] D.E. Rumelhart, J.L. McCelland. *Parallel distributed processing.* MIT Press, Cambridge, MA, 1986.

[7] G. Thimm, E. Fiesler. *High-order and multilayer perceptron initialization.* IEEE Transactions on Neural Networks 8, pp. 349-359, 1997.

[8] S. Geman, E. Bienenstock, R. Doursat. *Neural networks and the bias / variance dilemma.* Neural Computation 4, pp. 1-58, 1992.

[9] D. Urbani. *Méthodes statistiques de sélection d'architectures neuronales : application à la conception de modèles de processus dynamiques.* Thèse de doctorat, Université Pierre et Marie Curie, 1995.

[10] M. Stone. *Cross-validatory choice and assessment of statistical predictions.* Journal of the Royal Statistical Society, B36, pp. 111-147, 1974.

[11] G. Monari, G. Dreyfus. *Withdrawing an example from the training set: an analytic estimation of its effect on a non-linear parameterised model.* Neurocomputing, 35, pp. 195-201, 2000.

[12] T.Poggio, V. Torre, C. Koch. *Computational vision and regularization theory.* Nature, 317, pp. 314-319.

[13] D.J.C. MacKay. *A practical bayesian framework for backpropagation networks.* Neural Computation 4, pp. 448-472, 1992.

[14] Y. Oussar, G. Dreyfus. *How to be gray box : dynamic semi-physical modelling.* Neural Networks, vol. 14, pp. 1161-1172, 2001.

[15] R. Quach, A. Masson, G. Dreyfus, J.P. Gauchi, S. Issanchou. *Plans d'expériences pour modèles neuronaux*. Colloque de Maîtrise des Risques et Sûreté de Fonctionnement « Lambda-Mu 2004 », 2004.

[16] O. Nelles, M. Fischer. *Local linear model trees (LOLIMOT) for nonlinear system identification of a cooling blast.* In European Congress on Intelligent Techniques and Soft Computing (EUFIT), Aachen, Germany, 1996.

[17] A.A. Bogdanov, E.A. Wan, M. Carlsson, Y. Zhang, R. Kieburtz and A. Baptista. *Model predictive neural control of high-fidelity helicopter model*. AIAA Guidance Navigation and Control Conference, Montreal, Canada, August 2001 (2001).

[18] C.E. Garcia, D. Prett and D.M. Morari. *Model predictive control: Theory and practice - A survey*. Automatica, Vol. 25, No. 3, pp. 335-348, 1989.

[19] D. Gu and H. Hu. *Neural predictive control for a car-like mobile robot*. International Journal of Robotics and Autonomous Systems, Vol. 39, No. 2-3, 2002.

[20] N.L. Ricker. *Model predictive control: state of the art*. Fourth International Conference on Chemical Process Control, Elsevier, Amsterdam, pp. 271-296, 1991.

[21] M. Lucea, Y. Oussar, G. Dreyfus. *Exact computation of the Hessian matrix of discrete-time dynamical models: application to optimal control.* En cours de correction pour publication dans Neurocomputing.

Les machines à vecteurs supports et autres méthodes de noyaux pour la régression

1 Introduction

Les travaux de V. Vapnik [1], publiés en 1979, sur la théorie statistique de l'apprentissage (cf. Chapitre 1 : Modélisation) peuvent être considérés comme le point de départ de ce qui devint par la suite les machines à vecteurs supports (ou SVM pour Support Vector Machines). L'intérêt pour ce sujet ne prit de l'ampleur qu'au début des années 1990, en réponse à différents problèmes d'apprentissage supervisé : les concepts relatifs à la théorie statistique de l'apprentissage furent alors exploités pour aborder d'une façon nouvelle la question du dilemme biais / variance [2], c'est-à-dire le compromis à trouver entre la capacité d'apprentissage et la capacité de généralisation d'un modèle par apprentissage. Vapnik et Chervonenkis avaient introduit en 1971 [3], dans le domaine de la classification, une manière de quantifier la capacité d'apprentissage d'une famille de fonctions en définissant une quantité (appelée depuis dimension de Vapnik-Chervonenkis ou VC-dimension) définie comme le nombre maximum d'exemples séparables par un classifieur à deux classes issu de cette famille. Le principe de minimisation du risque structurel, issu de la théorie statistique de l'apprentissage, a permis d'autre part de définir des bornes à la capacité de généralisation d'un classifieur [4]. Il devint ainsi possible, au sein d'une famille de fonctions particulières, de formuler et de traiter d'un point de vue pratique des problèmes de classification, afin de trouver le meilleur compromis biais / variance, plutôt que de traiter séparément ces deux aspects du problème d'apprentissage.

Les machines à vecteurs supports ont été inventées en 1992 [5], mais le terme « machines à vecteurs supports » n'est apparu qu'en 1995 [6]. Depuis lors, de nombreux développements ont été réalisés pour adapter cette méthode d'apprentissage à la régression [7], pour établir des liens avec les méthodes de régularisation [8], ou pour proposer des variantes, qui, comme les SVM, entrent dans la catégorie des méthodes de noyaux, et qui en reprennent les principes essentiels [9]. Dans de nombreuses applications, ces outils de modélisation se sont avérés très performants, surpassant parfois d'autres méthodes, notamment les réseaux de neurones, pour des problèmes de classification [10] ou de régression [11].

Dans ce chapitre, nous introduirons dans une première partie les espaces de Hilbert à noyaux reproduisants (RKHS : Reproducing Kernel Hilbert Space), espaces dans lesquels s'effectue

de manière implicite la régression non linéaire par les méthodes de noyaux, avant d'illustrer la démarche d'apprentissage correspondante pour une de ces méthodes, appelée Least Squares SVM (LSSVM). Nous définirons ensuite les SVM proprement dites et décrirons en détail leur apprentissage dans le cas de la régression, avant de présenter quelques autres méthodes de noyaux, adaptables à des problèmes de régression non linéaire.

Dans la seconde partie du chapitre, nous proposons une méthodologie d'adaptation des algorithmes d'apprentissage existants au cas des modèles dynamiques à temps discret, décrits par des équations aux différences récurrentes. Il s'agira tout d'abord d'introduire une approche analytique, inspirée des travaux de Suykens & al. [31], pour formaliser un problème d'apprentissage adapté aux modèles récurrents, dans le cas d'une méthode de noyaux particulière (les LSSVM). Nous adopterons ensuite une approche algorithmique visant à calquer la méthode d'apprentissage semi dirigé utilisée pour les réseaux de neurones récurrents.

2 Espaces de Hilbert à noyaux reproduisants

2.1 Définition et propriétés

2.1.1 Définition

On appelle « Espace de Hilbert à noyau reproduisant » ou RKHS (Reproducing Kernel Hilbert Space) un espace de Hilbert H qui possède les propriétés suivantes :

- étant donné un espace X, H contient toutes les fonctions telles que

$$\forall \mathbf{x} \in X \quad K_{\mathbf{x}} : \mathbf{t} \to K(\mathbf{x}, \mathbf{t})$$

- il existe une fonction $K_{\mathbf{x}}$ telle que, pour tout $\mathbf{x}$ de X et pour tout $f(.)$ de H, on ait :

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_H$$

où $\langle f, g \rangle_H$ désigne le produit scalaire défini dans H. $K_{\mathbf{x}}$ est le noyau reproduisant de H ; s'il existe (donc si H est un RKHS), il est unique.

2.1.2 Propriétés

- Un noyau reproduisant vérifie la *propriété de reproduction* :

$$\langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_H = K(\mathbf{x}, \mathbf{y})$$

- Un noyau reproduisant existe si et seulement si $f(\mathbf{x})$ est continue.

- Tout noyau reproduisant est un *noyau défini positif* : étant donnés N vecteurs $\{\mathbf{x}_i, i = 1, \ldots N\}$, la matrice d'élément général $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ est semi-définie positive.
- Toute combinaison linéaire de fonctions noyaux est une fonction noyau ;
- Tout produit de fonctions noyaux est une fonction noyau.

2.1.3 Exemple : noyaux linéaires

Considérons l'espace de Hilbert H des formes linéaires :

$$\mathbf{x} \in X \rightarrow f(\mathbf{x}) = \sum_i a_i \mathbf{x}_i.\mathbf{x} = \mathbf{v}.\mathbf{x} \text{ avec } \mathbf{v} = \sum_i a_i \mathbf{x}_i$$

(où l'opération . désigne le produit scalaire dans X). Cet espace H est muni du produit scalaire

$$\langle f, g \rangle_H = \mathbf{v}.\mathbf{w} \text{ où } f(\mathbf{x}) = \mathbf{v}.\mathbf{x} \text{ et } g(\mathbf{x}) = \mathbf{w}.\mathbf{x}$$

Cet espace de Hilbert est un RKHS, qui admet comme noyau le *noyau linéaire* $\forall \mathbf{x} \in X \quad K_{\mathbf{x}} : \mathbf{t} \rightarrow K(\mathbf{x}, \mathbf{t}) = \mathbf{x}.\mathbf{t}$. Les deux propriétés des RKHS mentionnées plus haut sont bien vérifiées :

- $\langle f, K_{\mathbf{x}} \rangle_H = \langle \mathbf{v}.\mathbf{t}, \mathbf{x}.\mathbf{t} \rangle_H = \mathbf{v}.\mathbf{x} = f(\mathbf{x})$
- $\langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_H = \langle \mathbf{x}.\mathbf{t}, \mathbf{y}.\mathbf{t} \rangle_H = \mathbf{x}.\mathbf{y} = K(\mathbf{x}, \mathbf{y})$

Pour les problèmes d'apprentissage qui nous intéressent, il est utile de noter que le noyau $K_{\mathbf{x}}$ est la fonction qui exprime la distance entre le vecteur $\mathbf{x}$ et tout vecteur $\mathbf{t}$ du même espace.
Le noyau linéaire est un noyau défini positif, puisque celui-ci est le produit scalaire $\mathbf{x}_i.\mathbf{x}_j$.

2.1.4 Cas des noyaux non linéaires

Le résultat qui justifie l'intérêt des noyaux dans le cadre de l'apprentissage est le suivant : étant donné un noyau reproduisant semi-défini positif $K_{\mathbf{x}} = K(\mathbf{x}, \mathbf{t})$ et son RKHS associé H, il existe une fonction $\phi(\mathbf{x})$ à valeurs dans un espace F muni d'un produit scalaire telle que

$$K(\mathbf{x}, \mathbf{t}) = \langle \phi(\mathbf{x}), \phi(\mathbf{t}) \rangle_F$$

Ainsi, la fonction noyau représente une distance, non plus entre les vecteurs $\mathbf{x}$ et $\mathbf{t}$ eux-mêmes comme c'est le cas pour le noyau linéaire, mais entre des caractéristiques (« features ») qui représentent $\mathbf{x}$ et $\mathbf{t}$ dans un autre espace F, de dimension (éventuellement infinie) différente de celle de l'espace des variables d'origine. Tout algorithme qui fait intervenir des distances entre éléments de l'ensemble d'apprentissage pourra donc être utilisé indifféremment dans l'espace d'origine ou dans l'espace des caractéristiques, sous réserve que l'on trouve un

noyau convenable, mais sans qu'il soit nécessaire de définir explicitement la fonction ϕ. On appellera noyau SV-admissible toute fonction noyau représentant un produit scalaire dans un espace image F donné : $K(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_F$.

Une fonction noyau invariante par translation (telle que $K(\mathbf{x},\mathbf{x}')=K(\mathbf{x}-\mathbf{x}')$) est SV-admissible si et seulement si sa transformée de Fourier est non-négative.

Si $s(\mathbf{x},\mathbf{x}')$ est une fonction symétrique telle que $K(\mathbf{x},\mathbf{x}') = \int_{\mathbf{X}} s(\mathbf{x},\mathbf{z}) s(\mathbf{x}',\mathbf{z}) d\mathbf{z}$ existe, alors K est une fonction noyau SV-admissible.

2.2 Construction d'un RKHS

Le théorème de Mercer [12] donne une condition nécessaire et suffisante d'existence d'un noyau, et fournit une méthode de construction du RKHS correspondant.

Pour la situation qui nous intéressera par la suite, on peut construire un espace de Hilbert de fonctions f de la forme : $f(\mathbf{x}) = \sum_{n=1}^{N} a_n \phi_n(\mathbf{x})$ où $\{\phi_n\}_{n=1}^{N}$ est un ensemble de fonctions de base linéairement indépendantes (et où N est potentiellement infini), muni du produit scalaire

$$\left\langle \sum_n a_n \phi_n(\mathbf{x}), \sum_n b_n \phi_n(\mathbf{x}) \right\rangle_H = \sum_n \frac{a_n b_n}{\lambda_n}$$

où $\{\lambda_n\}_{n=1}^{N}$ est une séquence positive et décroissante de valeurs réelles dont la somme est finie.

La norme d'une fonction f est alors $\|f\|_H^2 = \sum_{n=1}^{N} \frac{a_n^2}{\lambda_n}$. Les ensembles $\{\phi_n\}_{n=1}^{N}$ et $\{\lambda_n\}_{n=1}^{N}$ définissent alors une fonction noyau, symétrique, définie positive : $K(\mathbf{x},\mathbf{y}) = \sum_{n=1}^{N} \lambda_n \phi_n(\mathbf{x}) \phi_n(\mathbf{y})$.

À titre d'exemple, citons quelques fonctions noyaux usuelles que nous emploierons dans la suite de ce mémoire :

- noyau gaussien : $K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2} \right)$;
- noyau polynomial : $K(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle^p$ et $K(\mathbf{x},\mathbf{y}) = \left(\langle \mathbf{x},\mathbf{y} \rangle + c \right)^p$, où p est un entier et $c \geq 0$, et où $\langle .,. \rangle$ est un produit scalaire dans l'espace considéré ;
- noyau tangente hyperbolique : $K(\mathbf{x},\mathbf{y}) = \tanh\left(\alpha + \beta \langle \mathbf{x},\mathbf{y} \rangle \right)$, où α et β sont deux réels.

2.3 Lien avec la minimisation du risque structurel

La notion de minimisation du risque structurel (SRM en anglais, pour Structural Risk Minimization) a été introduite au paragraphe 3.3 du Chapitre 1. Rappelons que le principe général de cette méthode est de rechercher une fonction dont le risque empirique s'approche du risque moyen, en définissant une séquence d'espaces hypothèses imbriqués, au sein desquels est minimisée une quantité tenant compte à la fois du risque empirique et de la capacité de la classe de fonctions envisagée. On peut montrer que les espaces de la forme $H_n = \left\{ f \in RKHS : \|f\|_H \leq A_n \right\}$ ont des capacités h_n qui sont fonctions croissantes de A_n. Il est ainsi aisé de construire une séquence d'espaces imbriqués $H_1 \subset H_2 \subset \ldots \subset H_m$ en définissant une série de valeurs $A_1 < A_2 < \ldots < A_m$. Le problème d'apprentissage par SRM devient alors, pour chaque A_n :

$$\begin{cases} \min\limits_f \sum\limits_{i=1}^{N} V\left(y_i, f\left(\mathbf{x}_i\right)\right) \\ \text{sous } \|f\|_H \leq A_n \end{cases}$$

Le problème précédent peut également s'écrire sous la forme fonctionnelle :

$$\min_f \left[\sum_{i=1}^{N} V\left(y_i, f\left(\mathbf{x}_i\right)\right) + \lambda \|f\|_H \right] = \min_f F[f] \qquad (1)$$

où λ est un paramètre à valeur réelle positive. Notons que ce paramètre λ joue le rôle d'un paramètre de régularisation, en ce qu'il pondère l'importance relative donnée à la limitation de la capacité des fonctions dans la recherche de la solution (pour de plus amples informations sur les liens entre la régularisation et la théorie statistique de l'apprentissage, on pourra se référer à [13]). En comparant la relation (1) ci-dessus à la relation (21) du paragraphe 3.4 du Chapitre 1, on observe que la norme sur H de la fonction f ($\|f\|_H$ dans la relation (1)), joue le rôle du terme pénalisant la capacité de la classe de fonctions candidates (la fonction croissante $\phi\left(\sqrt{h/N}, \eta\right)$ dans la relation (21)). On retrouve le principe de minimisation du risque structurel, exprimé sous forme d'un problème d'optimisation dont le premier terme pénalise le risque empirique, tandis que le second pénalise la capacité des fonctions candidates, assurant ainsi de bonnes propriétés de généralisation à la solution. La relation (1) tire donc profit du fait que la norme sur H d'une classe de fonctions croît avec la capacité de celle-ci : cette caractéristique permet de s'affranchir du problème de la détermination sous forme explicite de bornes supérieures au risque moyen (cf. relation (19) du Chapitre 1).

Le *théorème de représentation* (« representer theorem » [14]) fournit un résultat remarquable pour la résolution de ce problème. Étant donnés

- un ensemble X muni d'un noyau reproduisant K, et le RKHS H_K associé,
- un ensemble fini d'objets $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N\}$,
- une fonction $\psi\left(\|f\|_{H_K}\right)$ strictement croissante par rapport à $\|f\|_{H_K}$,
- une fonction de coût $C\left(\left\{f\left(\mathbf{x}_i\right), y_i\right\}_{1\le i\le N}\right)$ ne dépendant que des valeurs prises par la fonction f en les points $\mathbf{x}_i$

la solution du problème de minimisation $\min_{f\in H_K}\left[C\left(\left\{f\left(\mathbf{x}_i\right), y_i\right\}_{1\le i\le N}\right)+\psi\left(\|f\|_{H_K}\right)\right]$ admet une représentation de la forme

$$f\left(\mathbf{x}\right)=\sum_{i=1}^{N} c_i K\left(\mathbf{x}_i, \mathbf{x}\right) \tag{2}$$

Clairement, le problème d'apprentissage (1) obéit aux conditions du théorème de représentation. On peut donc lui trouver une solution sous la forme d'une combinaison linéaire de noyaux, c'est-à-dire d'une combinaison linéaire des distances entre l'objet inconnu $\mathbf{x}$ et les éléments de l'ensemble d'apprentissage $\{\mathbf{x}_i\}$, dans l'espace de représentation défini *implicitement* par le noyau choisi.

2.4 Résumé : apprentissage et noyaux

Résumons la démarche, dont nous avons décrit les grandes lignes, qui vise à réaliser un apprentissage par minimisation du risque structurel :

- L'apprentissage par minimisation du risque structurel, décrit au chapitre 1, exige la résolution successive de problèmes d'optimisation dans des espaces de fonctions de complexités croissantes.
- Ces problèmes d'optimisation peuvent être mis sous une forme telle que l'on puisse appliquer le théorème de représentation, c'est-à-dire exprimer leur solution sous la forme d'une combinaison linéaire de fonctions noyaux, sous réserve de connaître celles-ci. La fonctionnelle à minimiser contient un terme de régularisation
- Le théorème de Mercer fournit des conditions d'existence et une méthode de construction de fonctions noyaux.
- Ces fonctions noyaux peuvent être interprétées géométriquement comme des produits scalaires entre des caractéristiques des exemples d'apprentissage, dans un espace

différent de l'espace des variables originales : c'est « l'astuce des noyaux » (« kernel trick »). Cet espace des caractéristiques peut être de dimension infinie.

Ainsi, l'apprentissage peut être considéré soit d'un point de vue fonctionnel (par le théorème de Mercer et le théorème de représentation), soit d'un point de vue géométrique (par le « kernel trick »). Ces différents points de vue sont évidemment équivalents.

Nous allons à présent illustrer cette démarche pour une classe particulière de méthodes de noyaux, appelée Least Squares Support Vector Machines (LSSVM), pour laquelle les principes que nous venons de présenter apparaîtront de manière claire. Bien qu'historiquement les LSSVM ne soient apparus qu'après les SVM, la nature de la fonction de coût utilisée permet en effet une compréhension plus intuitive des différentes approches possibles pour parvenir à la solution du problème d'apprentissage.

3 Least Squares SVM (LSSVM)

3.1 Description de la méthode et résolution

La régression de crête (« ridge regression »), méthode à laquelle appartiennent les LSSVM, est une technique de régression qui constitue une forme régularisée des moindres carrés [15] ; il est possible de l'inscrire dans le cadre du principe de minimisation du risque structurel.

On dispose d'un ensemble de données $D = \left\{ \left(\mathbf{x}_i, y_i \right) \in X \times Y \right\}_{i=1}^{N}$, où $X \subset \mathbb{R}^d$ et $Y \subset \mathbb{R}$.

L'objectif est de trouver une fonction *f* dans un espace de Hilbert *H*, par minimisation du risque structurel (SRM), établissant une relation entre les variables **x** et la grandeur à modéliser *y*, à partir de l'ensemble de mesures *D*.

Plaçons-nous tout d'abord dans le cas linéaire : on cherche une fonction solution qui s'écrive sous la forme $f\left(\mathbf{x}\right) = \mathbf{w} \cdot \mathbf{x} + b$, où **w** et *b* sont les paramètres du modèle, et où $\mathbf{w} \cdot \mathbf{x}$ est le produit scalaire entre **w** et **x** dans l'espace *X*. En se référant à la relation (1), la fonctionnelle que l'on cherche à optimiser s'écrit :

$$F\left[f\right] = \gamma \sum_{i=1}^{N} \left(y_i - f\left(\mathbf{x}_i\right) \right)^2 + \frac{1}{2} \left\| f \right\|_H^2 \tag{3}$$

où $\left\| \cdot \right\|_H \equiv \left\| \cdot \right\|$ est une norme sur l'espace *H*.

Cette fonctionnelle (3) se traduit en le problème d'optimisation suivant :

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{N} \xi_i^2$$
$$\text{sous } \xi_i = y_i - f(\mathbf{x}_i) = y_i - \mathbf{w}\cdot\mathbf{x}_i - b, \forall i \in [\![1, N]\!] \tag{4}$$

où le premier terme de la fonction objectif représente la norme de la fonction *f* dans l'espace *H* (terme de régularisation), et où le second terme représente la fonction de coût des moindres carrés au travers des variables d'erreurs ξ_i : on retrouve donc l'expression classique du problème d'optimisation des moindres carrés régularisés [16].

Nous allons montrer que ce problème d'optimisation (4) peut être résolu de différentes manières, comme décrit au paragraphe 2.4.

3.1.1 Recherche de la solution par utilisation du théorème de représentation

En omettant le terme de biais *b* (nous reviendrons ultérieurement sur ce point), le problème d'optimisation (4) peut s'écrire :

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{N} (y_i - \mathbf{w}\cdot\mathbf{x}_i)^2, \tag{5}$$

soit, de manière équivalente et sous forme matricielle :

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{4\gamma}\|\mathbf{w}\|^2 + \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 \tag{6}$$

où $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$, $\mathbf{Y} = [y_1 \dots y_N]^T$.

Le problème (6) vérifie bien les hypothèses du théorème de représentation. On peut donc écrire que la solution *f* admet une représentation du type :

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \mathbf{x}_i \cdot \mathbf{x}$$

Ceci impose au vecteur **w** de s'écrire sous la forme :

$$\mathbf{w} = \sum_{i=1}^{N} c_i \mathbf{x}_i \equiv \mathbf{X}^T\mathbf{c}$$

où $\mathbf{c} = [c_1 \dots c_N]^T$. La fonction objectif du problème d'optimisation (6) devient :

$$J(\mathbf{c}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{X}\mathbf{X}^T\mathbf{c}\|^2 + \frac{1}{4\gamma}\|\mathbf{X}^T\mathbf{c}\|^2$$

En imposant la stationnarité de *J* vis-à-vis de **c**, on obtient :

$$\frac{\partial J}{\partial \mathbf{c}} = \mathbf{0} \Leftrightarrow \mathbf{X}\mathbf{X}^T \left(\frac{1}{2\gamma}\mathbf{I} + \mathbf{X}\mathbf{X}^T \right) \mathbf{c} = \mathbf{X}\mathbf{X}^T \mathbf{Y}$$

où $\mathbf{I}$ est la matrice identité (*N*, *N*).

Comme la matrice définie positive $\mathbf{X}\mathbf{X}^T$ est inversible, le minimum de *J*, qui correspond à la solution du problème d'optimisation (6), vérifie la relation suivante :

$$\left(\frac{1}{2\gamma}\mathbf{I} + \mathbf{X}\mathbf{X}^T \right) \mathbf{c} = \mathbf{Y} \tag{7}$$

La recherche de la solution se ramène donc au calcul de l'inverse (ou de la pseudo inverse) d'une matrice qui ne fait intervenir que les produits scalaires entre éléments de l'espace de régression.

Cette caractéristique de la solution va nous permettre de faire usage de l' « astuce des noyaux» pour étendre la méthode de régression au domaine non linéaire [17]. En réalisant une transformation d'espace préliminaire, définie par une fonction $\phi : X \to F$, où *F* est un espace image muni du produit scalaire $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_F = K(\mathbf{x}, \mathbf{y})$, le problème d'apprentissage devient :

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{N} \left(y_i - \mathbf{w} \cdot \phi(\mathbf{x}_i) \right)^2 \tag{8}$$

où **w** appartient désormais à l'espace image *F*, de dimension potentiellement infinie.

Le résultat obtenu dans le cas linéaire nous permet d'affirmer que la solution du problème (8) est définie par la relation suivante :

$$\left(\frac{1}{2\gamma}\mathbf{I} + \mathbf{K} \right) \mathbf{c} = \mathbf{Y} \tag{9}$$

où $\mathbf{K} = \left[k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \right]_{i,j=1}^{N}$ joue le rôle de la matrice $\mathbf{X}\mathbf{X}^T$ du cas linéaire.

La fonction *f* solution s'écrit alors de la façon suivante :

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^{N} c_i K(\mathbf{x}_i, \mathbf{x}) \tag{10}$$

En définitive, l'utilisation du théorème de représentation nous a permis, dans le cas linéaire, d'exprimer la solution du problème d'apprentissage comme une somme pondérée de produits scalaires sur les exemples de la base d'apprentissage. Ce résultat permet d'étendre la méthode de régression au cas non linéaire par une simple transformation d'espace, qui n'intervient qu'implicitement au travers de la définition d'une fonction noyau *K* (« astuce des noyaux »).

Concernant le terme de biais *b*, que nous avons omis lors de la recherche de la solution,

notons que celui-ci n'a que peu d'influence dans le cas non linéaire, puisqu'il revient à ajouter une dimension supplémentaire à un espace de dimension potentiellement infinie.

3.1.2 Recherche de la solution par exploitation des conditions de points selle

Revenons à l'expression du problème initial (4), en présence du terme de biais, et après une transformation d'espace $\phi : X \to F$ (nous nous plaçons donc directement dans le cas non linéaire et noterons $\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle \equiv \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle_F$ le produit scalaire dans *F*) :

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{N} \xi_i^2 \\ &\text{sous } \xi_i = y_i - f(\mathbf{x}_i) = y_i - \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle - b, \forall i \in [\![1, N]\!] \end{aligned}$$

Nous allons à présent montrer qu'en exploitant les caractéristiques de ce problème d'optimisation, il est également possible d'aboutir à l'expression de la solution.

Le Lagrangien du problème d'optimisation ci-dessus s'écrit :

$$L(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{N} \xi_i^2 - \sum_{i=1}^{N} \left[\alpha_i \left(\xi_i - y_i + \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b \right) \right] \tag{11}$$

où les coefficients $\{\alpha_i\}_{i=1}^{N}$, appelés multiplicateurs de Lagrange, sont des réels quelconques du fait des contraintes d'égalité du problème d'optimisation.

La convexité de la fonction objectif et la linéarité des contraintes d'égalité permettent d'affirmer l'existence d'un point selle en l'unique solution du problème [18]. Il est donc possible de rechercher le minimum vis-à-vis des variables primales du Lagrangien (11), avant de le rechercher vis-à-vis des variables duales (multiplicateurs de Lagrange).

La stationnarité du Lagrangien vis-à-vis des variables primales s'écrit :

$$\begin{cases} \dfrac{\partial L}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} - \displaystyle\sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) = 0 \Leftrightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \\ \dfrac{\partial L}{\partial b} = 0 \Leftrightarrow \displaystyle\sum_{i=1}^{N} \alpha_i = 0 \\ \dfrac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow 2\gamma\xi_i - \alpha_i = 0 \Leftrightarrow \xi_i = \dfrac{\alpha_i}{2\gamma}, \forall i \in [\![1, N]\!] \end{cases}$$

En substituant ces trois relations dans l'expression du Lagrangien primal on obtient le problème d'optimisation dual suivant :

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle - \frac{1}{4\gamma} \sum_{i=1}^{N} \alpha_i^2 + \sum_{i=1}^{N} y_i \alpha_i \\ &\textit{sous} \quad \sum_{i=1}^{N} \alpha_i = 0 \end{aligned} \tag{12}$$

Sous forme matricielle, celui-ci s'écrit :

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T \mathbf{K}\boldsymbol{\alpha} - \frac{1}{4\gamma}\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{y} \\ \textit{sous} \quad & \mathbf{1}^T\boldsymbol{\alpha} = 0 \end{aligned} \tag{13}$$

où $\mathbf{K} = \left[k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle \right]_{i,j=1}^{N}$, $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$,

$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T$, $\mathbf{1} = [1, \ldots, 1]^T$.

Une fois déterminés les coefficients α_i, le terme de biais b peut être calculé sur chaque exemple (ou en moyennant sur l'ensemble des N exemples) par :

$$b = y_i - \left\langle \mathbf{w}, \phi(\mathbf{x}_i) \right\rangle - \xi_i = y_i - \sum_{j=1}^{N} \alpha_j \left\langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \right\rangle - \frac{\alpha_i}{2\gamma}, \forall i \in [\![1, N]\!]$$

Le fait d'adopter un formalisme Lagrangien et d'exploiter les caractéristiques du problème d'optimisation permet donc d'aboutir à l'expression d'un nouveau problème d'optimisation dont la résolution est plus aisée (les contraintes étant plus simples à traiter).

Remarque : en l'absence du terme de biais b, la contrainte $\sum_{i=1}^{N} \alpha_i = 0$ disparaît, et le problème d'optimisation est encore simplifié. La solution du problème d'optimisation s'obtient alors simplement en imposant la stationnarité au Lagrangien dual $W(\boldsymbol{\alpha})$: $\frac{\partial W}{\partial \boldsymbol{\alpha}} = 0 \Leftrightarrow \left(\mathbf{K} + \frac{1}{2\gamma}\mathbf{I} \right)\boldsymbol{\alpha} = \mathbf{y}$, où $\mathbf{I}$ est la matrice identité (N, N). On retrouve alors la relation (9), obtenue par utilisation du théorème de représentation en l'absence de biais, et l'on est ramené au calcul de la pseudo-inverse d'une matrice (N, N).

3.1.3 Recherche de la solution par résolution des équations de Karush, Kuhn et Tucker

Une troisième approche est également envisageable : elle s'appuie à nouveau sur les caractéristiques du problème d'optimisation, mais les exploite à travers des conditions particulières, que nous allons décrire.

Revenons au problème initial :

$$\begin{aligned} & \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{N} \xi_i^2 \\ & \text{sous } \xi_i = y_i - f(\mathbf{x}_i) = y_i - \left\langle \mathbf{w}, \phi(\mathbf{x}_i) \right\rangle - b, \forall i \in [\![1, N]\!] \end{aligned}$$

Outre leur caractère différentiable, la convexité de la fonction objectif et des fonctions exprimant les contraintes d'égalité, nous permet d'affirmer que les conditions de Karush, Kuhn et Tucker (KKT) sont nécessaires et suffisantes [18].

En d'autres termes, $(\mathbf{w}, b)^T$ est minimum global du problème d'optimisation ci-dessus si et seulement si il existe des réels **α** et *b* tels que :

$$\begin{cases} \dfrac{\partial L}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} - \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) = 0 \Leftrightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \\ \dfrac{\partial L}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{N} \alpha_i = 0 \\ \dfrac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow 2\gamma\xi_i - \alpha_i = 0 \Leftrightarrow \xi_i = \dfrac{\alpha_i}{2\gamma}, \forall i \in [\![1, N]\!] \\ \xi_i = y_i - \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle - b, \forall i \in [\![1, N]\!] \end{cases}$$

En éliminant **w** et **ξ** dans les *N* dernières de ces équations, on se ramène ainsi à la résolution du système de *N*+1 équations linéaires suivant :

$$\begin{bmatrix} \mathbf{K} + \dfrac{1}{2\gamma}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \tag{14}$$

où **I** est la matrice identité (*N, N*).

La fonction *f*, solution du problème initial, s'écrira finalement :

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Les LSSVM aboutissent donc à un développement de la solution sur l'ensemble des exemples d'apprentissage : du fait du choix de la fonction de coût, qui s'est traduit par des contraintes d'égalité dans la formulation du problème d'optimisation, tous les multiplicateurs de Lagrange sont non nuls. Dans ce cas, on dit que tous les exemples de l'ensemble d'apprentissage sont des vecteurs supports. Cette caractéristique peut devenir problématique, du point de vue du temps de calcul ou de la capacité mémoire, pour des bases de données de taille importante. Une variante, exposée ci-dessous, permet de ne prendre en considération, dans le développement de **w**, qu'un nombre limité d'exemples.

3.2 Variante parcimonieuse : Sparse least squares SVM (sLSSVM)

La présente méthode [19] consiste à écrire le vecteur **w** sous la forme d'une somme pondérée d'un nombre limité d'exemples : $\mathbf{w} = \sum_{i=1}^{n} \beta_i \phi(\mathbf{z}_i)$, où $n < N$.

La fonction de coût à minimiser devient alors :

$$\begin{aligned} J(\boldsymbol{\beta}, b) &= \frac{1}{2}\sum_{i,j=1}^{n} \beta_i \beta_j K(\mathbf{z}_i, \mathbf{z}_j) + \gamma \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{n} \beta_j K(\mathbf{x}_i, \mathbf{z}_j) - b \right)^2 \\ &= \frac{1}{2}\sum_{i,j=1}^{n} \beta_i \beta_j k_{ij} + \gamma \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{n} \beta_j \widehat{k_{ij}} - b \right)^2, \text{ où } k_{ij} = K(\mathbf{z}_i, \mathbf{z}_j) \text{ et } \widehat{k_{ij}} = K(\mathbf{x}_i, \mathbf{z}_j) \end{aligned} \tag{15}$$

En imposant aux dérivées partielles de *J* par rapport à **β** et *b* de s'annuler, on obtient :

$$\begin{cases} \sum_{i=1}^{n} \beta_i \left(\frac{1}{2\gamma} k_{ir} + \sum_{j=1}^{N} \widehat{k_{jr}} \widehat{k_{ji}} \right) + b \sum_{j=1}^{N} \widehat{k_{jr}} = \sum_{j=1}^{N} y_j \widehat{k_{jr}} \text{ , } \forall r \in [\![1, n]\!] \\ \sum_{i=1}^{n} \beta_i \sum_{j=1}^{N} \widehat{k_{ji}} + b = \sum_{j=1}^{N} y_j \end{cases}$$

Ce système de (*n*+1) équations à (*n*+1) inconnues peut s'écrire sous la forme matricielle suivante :

$$\begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \sum_{i=1}^{N} y_i \end{bmatrix}$$

où $\boldsymbol{\Omega} = \left[\omega_{ij} \right]_{i,j=1}^{n} = \left[\frac{1}{2}\gamma^{-1} k_{ij} + \sum_{r=1}^{N} \widehat{k_{rj}} \widehat{k_{ri}} \right]_{i,j=1}^{n}$, $\boldsymbol{\Phi} = \left[\phi_i \right]_{i=1}^{n} = \left[\sum_{j=1}^{N} \widehat{k_{ji}} \right]_{i=1}^{n}$, $\mathbf{c} = \left[c_i \right]_{i=1}^{n} = \left[\sum_{j=1}^{N} y_j \widehat{k_{ji}} \right]_{i=1}^{n}$

On se ramène ainsi à la forme suivante :

$$\left(\mathbf{R} + \mathbf{Z}^T \mathbf{Z} \right) \mathbf{p} = \mathbf{Z}^T \mathbf{y} \tag{16}$$

où $\mathbf{R} = \begin{bmatrix} \frac{1}{2\gamma}\mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$, $\mathbf{Z} = \left[\widehat{\mathbf{K}} \quad \mathbf{1} \right]$, $\mathbf{K} = \left[k_{ij} = K(\mathbf{z}_i, \mathbf{z}_j) \right]_{i,j=1}^{n}$, $\widehat{\mathbf{K}} = \left[\widehat{k_{ij}} = K(\mathbf{x}_i, \mathbf{z}_j) \right]_{i=1,\ldots,N}^{j=1,\ldots n}$,

et $\mathbf{p} = (\boldsymbol{\beta}, b)^T$, $\mathbf{1} = [1]_{i=1}^{N}$, $\mathbf{y} = [y_i]_{i=1}^{N}$

Comme l'apprentissage des sLSSVM se résume à la résolution d'un système d'équations linéaires sans contraintes, elles se prêtent bien à différents types de calculs analytiques comme décrit dans [20] (calculs des leviers relatifs aux exemples d'apprentissage, calcul rapide du score de leave-one-out, etc...).

En revanche, la question du choix des *n* exemples participant au développement de **w** est délicate. En effet, il n'existe pas de méthode systématique pour répondre à cette question. Diverses heuristiques ont été proposées : Baudat & Anouar [21] ainsi que Cawley & al. [20] proposent de choisir ces vecteurs de référence comme constituant le jeu minimum de vecteurs formant une base du sous-espace engendré dans *F* par l'ensemble des exemples. Une autre méthode, proposée par Suykens & al. [19][22], consiste à supprimer itérativement du développement les exemples sur lesquels est commise une faible erreur, après avoir débuté par un développement sur l'ensemble des exemples.

4 Les Machines à Vecteurs Supports (SVM) pour la régression

4.1 Formulation du problème dans le cas linéaire

Après avoir illustré les principes de base, introduits au paragraphe 2.4, dans le cas des LSSVM, nous allons à présent décrire les machines à vecteurs supports sous leur forme conventionnelle.

On dispose d'un ensemble de données $D = \left\{ \left(\mathbf{x}_i, y_i \right) \in X \times Y \right\}_{i=1}^{N}$, où $X \subset \mathbb{R}^d$ et $Y \subset \mathbb{R}$.

L'objectif est de trouver une fonction *f* dans un espace de Hilbert *H*, par minimisation du risque structurel (SRM), établissant une relation entre les variables **x** et la grandeur à modéliser *y*, à partir de l'ensemble de mesures *D*. La fonctionnelle que l'on cherche à optimiser s'écrit :

$$F[f] = C\sum_{i=1}^{N} \left| y_i - f\left(\mathbf{x}_i\right) \right|_{\varepsilon} + \frac{1}{2}\left\| f \right\|_H^2 \tag{17}$$

où $\left\| \cdot \right\|_H$ est une norme sur *H*, et où $\left| \cdot \right|_{\varepsilon}$ représente la fonction de coût ε-insensible suivante :

$$V\left(y, f\left(\mathbf{x}\right)\right) = \left| y - f\left(\mathbf{x}\right) \right|_{\varepsilon} = \begin{cases} 0 & \text{si } \left|\mathbf{x}\right| < \varepsilon \\ \left|\mathbf{x}\right| - \varepsilon & \text{sinon} \end{cases}$$

Cette fonction de coût ne pénalise que les déviations supérieures à ε entre les sorties mesurées et estimées.

Nous allons, dans un premier temps, réaliser un modèle linéaire, en cherchant la fonction de régression *f* sous la forme suivante :

$f\left(\mathbf{x}\right) = \mathbf{w} \cdot \mathbf{x} + b$, où $\mathbf{w} \cdot \mathbf{x}$ est un produit scalaire dans l'espace *X*.

En introduisant des variables d'erreurs ξ_i et ξ_i^* (pour les erreurs respectivement positives et négatives), on parvient à intégrer la fonction de coût ε-insensible au sein du problème d'optimisation suivant :

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) \\ &\text{sous} \begin{cases} y_i - \mathbf{w}\cdot\mathbf{x}_i - b \le \varepsilon + \xi_i \\ \mathbf{w}\cdot\mathbf{x}_i + b - y_i \le \varepsilon + \xi_i^*, \ \forall i \in [\![1, N]\!] \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{18}$$

Le premier terme de la fonction objectif limite la capacité de la fonction de régression (terme de régularisation), tandis que le second pénalise les erreurs supérieures à ε, qu'elles soient par valeur positive ou négative : ce compromis biais/variance est contrôlé par la constante C.

Le lagrangien associé à la fonction objectif sous contrainte ci-dessus est :

$$\begin{aligned} &L\left(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\eta}, \boldsymbol{\eta}^*\right) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right) \\ &-\sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - y_i + \mathbf{w}\cdot\mathbf{x}_i + b\right) - \sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \mathbf{w}\cdot\mathbf{x}_i - b\right) \end{aligned} \tag{19}$$

où $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ sont les multiplicateurs de Lagrange positifs (variables duales) associés aux contraintes inégalités du problème initial (18).

On peut montrer que, pour le problème d'optimisation ci-dessus (fonction de coût convexe, contraintes linéaires), les conditions complémentaires de KKT sont nécessaires et suffisantes. De plus, l'existence d'un point selle impose que le minimum par rapport aux variables primales ($\mathbf{w}$, b, $\boldsymbol{\xi}$, $\boldsymbol{\xi}^*$) coïncide avec le maximum par rapport aux variables duales ($\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^*$, $\boldsymbol{\eta}$, $\boldsymbol{\eta}^*$). En imposant la stationnarité du lagrangien vis-à-vis des variables primales, on obtient :

$$\begin{cases} \dfrac{\partial L}{\partial b} = 0 \Leftrightarrow \displaystyle\sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0 \\ \dfrac{\partial L}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} = \displaystyle\sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\mathbf{x}_i \\ \dfrac{\partial L}{\partial \xi_i^{(*)}} = 0 \Leftrightarrow C = \alpha_i^{(*)} + \eta_i^{(*)}, \ \forall i \in [\![1, N]\!] \end{cases}$$

où $\boldsymbol{\alpha}^{(*)}$ représente $\boldsymbol{\alpha}$ ou $\boldsymbol{\alpha}^*$.

En substituant les relations ci-dessus dans l'expression du Lagrangien (19), on aboutit au problème d'optimisation suivant, qui n'est fonction que des variables duales :

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -\frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\mathbf{x}_i\cdot\mathbf{x}_j-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{N}y_i\left(\alpha_i-\alpha_i^*\right)$$
$$sous\begin{cases}\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0\\ \alpha_i,\alpha_i^*\in[0,C]\end{cases} \tag{20}$$

On est ainsi ramené à la résolution d'un problème d'optimisation quadratique, dont les contraintes d'égalité sont plus aisées à manipuler que celles du problème primal.

Les conditions complémentaires de KKT s'écrivent :

$$\begin{cases}\alpha_i\left(\varepsilon+\xi_i-y_i+\mathbf{w}\cdot\mathbf{x}_i+b\right)=0\\ \alpha_i^*\left(\varepsilon+\xi_i^*+y_i-\mathbf{w}\cdot\mathbf{x}_i-b\right)=0\\ \left(C-\alpha_i\right)\xi_i=0\\ \left(C-\alpha_i^*\right)\xi_i^*=0\end{cases},\forall i\in[\![1,N]\!]$$

Seuls les coefficients α_i relatifs aux exemples pour lesquels les contraintes sont saturées peuvent être non nuls : on appelle ces points les *vecteurs supports*. Les conditions de KKT permettent également de déterminer le biais *b*.

Après détermination des coefficients α_i,α_i^* par résolution du problème d'optimisation (20), la fonction *f* recherchée s'écrit :

$$f\left(\mathbf{x}\right)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\mathbf{x}_i\cdot\mathbf{x}+b \tag{21}$$

Cette fonction s'écrit donc comme une somme, pondérée par les coefficients $\alpha_i-\alpha_i^*$, de produits scalaires sur l'ensemble des exemples de la base d'apprentissage. Cette caractéristique, qui est conforme au résultat énoncé dans la relation (2), va à présent être exploitée pour étendre ces résultats au cas non linéaire (« astuce des noyaux »).

Remarque : la différence fondamentale entre les SVM conventionnelles et les LSSVM introduites au paragraphe précédent réside dans la fonction de coût utilisée : plutôt que d'employer la fonction de coût quadratique, on se sert ici d'une fonction de coût ε-insensible. Ce choix a une incidence sur la nature du problème d'optimisation obtenu (présence de contraintes d'inégalité) ainsi que sur l'expression de la solution (développement sur un nombre limité de vecteurs supports, correspondant à des contraintes saturées).

4.2 Extension au cas non linéaire

Comme nous l'avons indiqué dans le paragraphe 2, il est possible de traiter le problème non linéaire de manière essentiellement identique à ce qui vient d'être fait dans le cas linéaire, en

cherchant la solution dans un RKHS muni d'un noyau convenablement choisi. Ceci revient à effectuer un « prétraitement » des variables, qui consiste en une transformation non linéaire de l'espace des variables X vers un espace image F, grâce à une fonction $\phi : X \to F$ associée au noyau K. Les résultats obtenus au paragraphe précédent dans l'espace initial X sont alors transposables à l'espace image F, le noyau choisi définissant la « distance » entre les objets dans l'espace des caractéristiques.

Le problème d'optimisation (20) devient donc :

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -\frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle\phi\left(\mathbf{x}_i\right),\phi\left(\mathbf{x}_j\right)\right\rangle-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{N}y_i\left(\alpha_i-\alpha_i^*\right)$$
$$sous\begin{cases}\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0\\ \alpha_i,\alpha_i^*\in[0,C]\end{cases} \qquad (22)$$

Notons que la fonction de régression, qui par construction sera non linéaire vis-à-vis des entrées, reste toutefois linéaire en ses paramètres. La propriété de convexité du problème d'optimisation (22) garantit donc l'unicité de la solution.

D'autre part, la transformation ϕ, qui n'intervient dans le problème (22) qu'à travers des produits scalaires dans l'espace image F, peut être définie de manière implicite au moyen d'une fonction noyau SV-admissible K, telle que : $\left\langle\phi\left(\mathbf{x}_i\right),\phi\left(\mathbf{x}_j\right)\right\rangle=K\left(\mathbf{x}_i,\mathbf{x}_j\right)$. Comme indiqué dans le paragraphe 2, il n'est donc pas nécessaire de connaître explicitement la transformation ϕ : celle-ci est entièrement définie par le choix du noyau K qui lui est associé.

La solution du problème d'optimisation (22) s'écrira finalement :

$$f\left(\mathbf{x}\right)=\left\langle\mathbf{w},\phi\left(\mathbf{x}\right)\right\rangle+b=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)K\left(\mathbf{x}_i,\mathbf{x}\right)+b \qquad (23)$$

4.3 Choix des hyperparamètres

La résolution des problèmes d'optimisation quadratique (20) ou (22) s'effectue à l'aide d'algorithmes classiques[1] (on pourra se référer à [23] pour un inventaire des principaux algorithmes utilisés). Elle nécessite la définition d'un ensemble d'hyperparamètres, regroupés dans le vecteur $\boldsymbol{\theta}_{\text{hyper}}$, dont les composantes sont :

- ε : la valeur d'insensibilité
- C : le compromis entre régularisation et pénalisation des erreurs

[1] Nous avons utilisé la fonction *quadprog* (ou *qp* suivant les versions) de la boîte à outils optimisation de Matlab.

- les différents paramètres relatifs au noyau utilisé (écart type des gaussiennes dans le cas de noyaux gaussiens)

Afin de choisir au mieux ces hyperparamètres, des méthodes classiques peuvent être mises en œuvre (notamment la validation croisée). Il est également possible, dans ce cas particulier des SVM, d'établir des bornes à certains indices de performance (erreur de validation, erreur de leave-one-out, …), et d'optimiser ceux-ci par une descente de gradient vis-à-vis des hyper paramètres [24].

La solution obtenue par un algorithme de SVM, c'est-à-dire l'ensemble des coefficients $\left\{\alpha_i, \alpha_i^*\right\}_{i=1}^{N}$, dépend évidemment des hyperparamètres utilisés. Toutefois, le résultat suivant simplifie grandement le problème [24].

Théorème : soit $\mathbf{v}_\theta$ un vecteur de dimension (n x 1) et $\mathbf{P}_\theta$ une matrice (n x n), dépendant continûment du paramètre θ. Considérons la fonction :

$$L(\theta) = \max_{\mathbf{x} \in F} \left(\mathbf{x}^T \mathbf{v}_\theta - \frac{1}{2} \mathbf{x}^T \mathbf{P}_\theta \mathbf{x} \right)$$

$$où\ F = \left\{ \mathbf{x} : \mathbf{b}^T \mathbf{x} = c, \mathbf{x} \geq 0 \right\}$$

Soit $\mathbf{x}_o$ la valeur en laquelle le maximum est atteint. Si ce maximum est unique, alors on a :

$$\frac{\partial L}{\partial \theta} = \mathbf{x}_o^T \frac{\partial \mathbf{v}_\theta}{\partial \theta} - \frac{1}{2} \mathbf{x}_o^T \frac{\partial \mathbf{P}_\theta}{\partial \theta} \mathbf{x}_o \qquad \blacksquare$$

En d'autres termes, on peut différentier L par rapport à θ comme si $\mathbf{x}_o$ ne dépendait pas de θ. Ce résultat reste valable si certaines contraintes dans la définition de F sont supprimées.

Il est donc possible de rechercher, en premier lieu, la solution SVM avec un jeu particulier d'hyper paramètres $\boldsymbol{\theta}_{\text{hyper}}$, puis de chercher, par descente de gradient, le minimum d'un indice de performance convenable par rapport à $\boldsymbol{\theta}_{\text{hyper}}$, par différentiation de celui-ci comme si les coefficients $\alpha_i^{(*)}$ ne dépendaient pas de $\boldsymbol{\theta}_{\text{hyper}}$.

4.4 Choix de la fonction de coût

4.4.1 Fonctions de coût et modèles de bruit associés

Nous avons souligné, au paragraphe 4.1, que la différence fondamentale entre les SVM et les LSSVM réside dans le choix de la fonction de coût. Indépendamment du fait que ce choix aboutit à des problèmes d'optimisation (et donc à des solutions) différents, nous allons montrer que, sous certaines conditions, le choix de la fonction de coût revient à faire une hypothèse sur la nature du bruit qui affecte les données.

Supposons que les données $D = \{(\mathbf{x}_i, y_i) \in X \times Y\}_{i=1}^{N}$ dont nous disposons sont engendrées par une fonction *f* purement déterministe, à laquelle s'ajoute un bruit additif, représenté par une variable aléatoire ξ suivant une distribution de probabilité *p* :

$$y_i = f(\mathbf{x}_i) + \xi_i$$

Nous supposons de plus la fonction *f* connue : l'objectif est alors de déterminer une fonction de coût optimale, au sens où sa minimisation correspond au maximum de vraisemblance de l'estimation $D_f = \{(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_N, f(\mathbf{x}_N))\}$ connaissant *D*.

Sous les hypothèses de bruit additif et de réalisations indépendantes et identiquement distribuées, la vraisemblance de l'estimation D_f connaissant *D* est définie par :

$$P(D_f \mid D) = \prod_{i=1}^{N} P(f(\mathbf{x}_i) \mid (\mathbf{x}_i, y_i)) = \prod_{i=1}^{N} P(f(\mathbf{x}_i) \mid y_i)$$

L'estimation D_f étant réalisée par la fonction génératrice des données *f*, la relation ci-dessus devient :

$$P(D_f \mid D) = \prod_{i=1}^{N} p(y_i - f(\mathbf{x}_i))$$

où *p* est la distribution de probabilité associée au bruit qui entache les données *D*.

L'estimateur du maximum de vraisemblance est celui pour lequel $P(D_f \mid D)$ est maximum, soit, de manière équivalente, celui pour lequel $-\log(P(D_f \mid D)) = -\sum_{i=1}^{N} \log(p(y_i - f(\mathbf{x}_i)))$ est minimum.

La fonction de coût optimale, au sens du maximum de vraisemblance, est alors définie comme :

$$V(y, f(\mathbf{x})) = -\log(p(y - f(\mathbf{x})))$$

L'utilisation de cette fonction de coût dans un problème d'apprentissage assure donc, en principe, la possibilité de retrouver la fonction génératrice des données *f*, à partir de l'ensemble de données *D*.

Le tableau ci-dessous indique les fonctions de coût usuelles ainsi que les distributions associées du bruit.

	Fonction de coût	**Distribution du bruit associée**

ε-insensible	$V(\xi)=\lvert\xi\rvert_\varepsilon$	$p(\xi)=\frac{1}{2(1+\varepsilon)}\exp\left(-\lvert\xi\rvert_\varepsilon\right)$
Laplacienne	$V(\xi)=\lvert\xi\rvert$	$p(\xi)=\frac{1}{2}\exp\left(-\lvert\xi\rvert\right)$
Gaussienne	$V(\xi)=\frac{1}{2}\xi^2$	$p(\xi)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\xi^2}{2}\right)$
Fonction de coût robuste de Huber	$V(\xi)=\begin{cases}\frac{1}{2\sigma}\xi^2 & \text{si } \lvert\xi\rvert\le\sigma \\ \lvert\xi\rvert-\frac{\sigma}{2} & \text{sinon}\end{cases}$	$p(\xi)\propto\begin{cases}\exp\left(-\frac{\xi^2}{2\sigma}\right) & \text{si } \lvert\xi\rvert\le\sigma \\ \exp\left(\frac{\sigma}{2}-\lvert\xi\rvert\right) & \text{sinon}\end{cases}$
Polynomiale	$V(\xi)=\frac{1}{p}\lvert\xi\rvert^p$	$p(\xi)=\frac{p}{2\Gamma(1/p)}\exp\left(-\lvert\xi\rvert^p\right)$
Polynomiale par morceaux	$V(\xi)=\begin{cases}\frac{1}{p\sigma^{p-1}}\xi^p & \text{si } \lvert\xi\rvert\le\sigma \\ \lvert\xi\rvert-\sigma\frac{p-1}{p} & \text{sinon}\end{cases}$	$p(\xi)\propto\begin{cases}\exp\left(-\frac{\xi^p}{p\sigma^{p-1}}\right) & \text{si } \lvert\xi\rvert\le\sigma \\ \exp\left(\sigma\frac{p-1}{p}-\lvert\xi\rvert\right) & \text{sinon}\end{cases}$

Il est à noter [25] que le choix de la fonction ε-insensible de Vapnik, fréquemment utilisée dans la régression par SVM, est équivalent à une hypothèse de bruit additif et gaussien, dont la variance et la moyenne des gaussiennes sont des variables aléatoires.

4.4.2 Problème d'optimisation pour une fonction de coût générale

Supposons que la fonction de coût V choisie vérifie les propriétés suivantes :

- V est symétrique
- la dérivée première de V présente au plus deux discontinuités à $\pm\varepsilon$
- V est nulle dans l'intervalle $[-\varepsilon,+\varepsilon]$

Nous pouvons alors écrire : $V\left(y,f(\mathbf{x})\right)=\widetilde{V}\left(\left|y-f(\mathbf{x})\right|_\varepsilon\right)$

En raisonnant de la même manière que précédemment, on aboutit au problème d'optimisation suivant (l'extension au cas non linéaire se faisant par introduction d'une fonction noyau en lieu et place des produits scalaires) :

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\left(\widetilde{V}\left(\xi_i\right)+\widetilde{V}\left(\xi_i^*\right)\right)$$

$$\text{sous}\begin{cases} y_i - \mathbf{w}\cdot\mathbf{x}_i - b \le \varepsilon + \xi_i \\ \mathbf{w}\cdot\mathbf{x}_i + b - y_i \le \varepsilon + \xi_i^*, \forall i \in [\![1, N]\!] \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

Ce problème d'optimisation peut se réécrire :

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\mathbf{x}_i\cdot\mathbf{x}_j + \sum_{i=1}^{N}\left[y_i\left(\alpha_i-\alpha_i^*\right)-\varepsilon\left(\alpha_i+\alpha_i^*\right)\right]+C\sum_{i=1}^{N}\left[T\left(\xi_i\right)+T\left(\xi_i^*\right)\right]$$

$$sous\begin{cases} \sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0 \\ \boldsymbol{\alpha} \le C\dfrac{\partial\widetilde{V}}{\partial\boldsymbol{\xi}} \\ \boldsymbol{\xi} = \inf\left\{\xi \,\middle|\, C\dfrac{\partial\widetilde{V}}{\partial\boldsymbol{\xi}} \ge \boldsymbol{\alpha}\right\} \\ \boldsymbol{\alpha},\boldsymbol{\xi} \ge 0 \end{cases} \qquad (24)$$

où $\mathbf{w} = \sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\mathbf{x}_i$, $\quad T\left(\xi\right) = \widetilde{V}\left(\xi\right) - \xi\dfrac{\partial\widetilde{V}}{\partial\xi}\left(\xi\right)$

Remarque : la condition de nullité dans l'intervalle [-ε,+ ε] peut être contournée par ajout de variables « élastiques» supplémentaires (« slack variables »). Il est également possible de définir une fonction de coût et une valeur de ε différentes pour chaque exemple. Plus de deux discontinuités sont possibles, au prix de multiplicateurs de Lagrange supplémentaires.

5 Quelques autres méthodes de noyaux

5.1 Analyse en Composantes Principales par noyaux (K-PCA)

5.1.1 Principe général

Le principe général de l'Analyse en Composantes Principales (ACP) [26] est de réduire l'espace des représentations, défini par les variables d'origine, en déterminant de nouveaux vecteurs orthogonaux, appelés composantes principales, dont le nombre est inférieur à la dimension de l'espace initial. Ces vecteurs sont construits par diagonalisation de la matrice centrée de covariance des variables $\{\mathbf{x}_1,\ldots,\mathbf{x}_N\} \in \mathbb{R}^d$, définie par :

$$\mathbf{C} = \left\langle \left(\mathbf{x}_i - \left\langle \mathbf{x}_k \right\rangle_k \right) \left(\mathbf{x}_i - \left\langle \mathbf{x}_k \right\rangle_k \right)^T \right\rangle_i \tag{25}$$

où $\left\langle \mathbf{x}_k \right\rangle_k$ représente la moyenne de $\mathbf{x}_k$, pour k variant de 1 à N. La valeur propre associée à un vecteur propre $\mathbf{V}$ de $\mathbf{C}$ est d'autant plus grande que la variance des données dans la direction de $\mathbf{V}$ est grande. Il est ainsi possible de classer ces composantes principales par ordre décroissant de variance, et de n'en retenir que les $d' < d$ premières.

Si l'on effectue une analyse en composantes principales non plus dans l'espace $\mathbb{R}^d$ initial mais après une transformation préliminaire $\phi : \mathbb{R}^d \to F$, où F est un espace image défini par la transformation ϕ associée au noyau K, on est alors amené à diagonaliser la matrice suivante :

$$\widetilde{\mathbf{C}} = \left\langle \phi(\mathbf{x}_k) \phi(\mathbf{x}_k)^T \right\rangle_k = \frac{1}{N} \sum_{k=1}^{N} \phi(\mathbf{x}_k) \phi(\mathbf{x}_k)^T \tag{26}$$

Afin que les données soient centrées dans l'espace image, il est nécessaire de les centrer à nouveau, en remplaçant $\phi(\mathbf{x}_k)$ par $\tilde{\phi}(\mathbf{x}_k) = \phi(\mathbf{x}_k) - \left\langle \phi(\mathbf{x}_k) \right\rangle_k$.

Soit $\mathbf{V}$ un vecteur propre de $\widetilde{\mathbf{C}}$, et λ la valeur propre correspondante ; on a alors :

$$\lambda \mathbf{V} = \tilde{\mathbf{C}} \mathbf{V}, \mathbf{V} \in F$$

Tout vecteur propre $\mathbf{V}$ appartenant nécessairement à l'espace engendré par les données transformées par ϕ, il existe des réels $\alpha_1, \ldots, \alpha_N$ tels que $\mathbf{V} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$, et l'équation précédente peut s'écrire, en projetant sur chacun des $\phi(\mathbf{x}_k)$:

$$\lambda \left\langle \phi(\mathbf{x}_k), \mathbf{V} \right\rangle = \left\langle \phi(\mathbf{x}_k), \tilde{\mathbf{C}} \mathbf{V} \right\rangle, \ \forall k \in [\![1, N]\!]$$

$$\Leftrightarrow \lambda \sum_{i=1}^{N} \alpha_i \left\langle \phi(\mathbf{x}_k), \phi(\mathbf{x}_i) \right\rangle = \left\langle \phi(\mathbf{x}_k), \left(\frac{1}{N} \sum_{j=1}^{N} \phi(\mathbf{x}_j) \phi(\mathbf{x}_j)^T \right) \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \right\rangle, \ \forall k \in [\![1, N]\!]$$

Sachant que le produit scalaire dans l'espace F est défini par $\left\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \right\rangle \equiv \phi(\mathbf{x})^T \phi(\mathbf{y}) \equiv K(\mathbf{x}, \mathbf{y})$, la relation précédente peut s'écrire :

$$N\lambda \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_k, \mathbf{x}_i) = \sum_{i=1}^{N} \alpha_i \sum_{j=1}^{N} K(\mathbf{x}_k, \mathbf{x}_j) K(\mathbf{x}_j, \mathbf{x}_i), \ \forall k \in [\![1, N]\!]$$

$$\Leftrightarrow N\lambda \mathbf{K}\boldsymbol{\alpha} = \mathbf{K}\mathbf{K}\boldsymbol{\alpha}$$

où $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$, et où $\mathbf{K}$ est la matrice définie positive (donc inversible) de terme général $K_{i,j} = \left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle$.

Le problème de diagonalisation de la matrice $\tilde{\mathbf{C}}$ se ramène donc à la résolution du système suivant [28] :

$$N\lambda\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha} \tag{27}$$

À chaque vecteur propre $\mathbf{V}^k$, déterminé par résolution de l'équation (27), est associée une valeur propre λ^k. Une méthode itérative, procédant par projections successives sur les sous-espaces orthogonaux au vecteur propre précédemment défini, permet de classer ces composantes par ordre de variance décroissante. On obtient alors une représentation des données d'entrées dans une nouvelle base orthogonale, dont la dimension est plus petite que celle de l'espace des variables initiales.

5.1.2 Détermination du nombre de composantes principales

Du fait de la transformation d'espace ϕ, les variables sur lesquelles est effectuée l'ACP appartiennent à un espace de dimension potentiellement infinie. Si le nombre n de composantes principales orthogonales est suffisamment grand pour prendre en considération l'ensemble des directions correspondant à des valeurs propres non nulles, la projection d'un point $\mathbf{x} \in \mathbb{R}^d$ dans cette nouvelles base coïncide avec l'image de $\mathbf{x}$ par la transformation ϕ :

$$P_n\phi(\mathbf{x}) = \sum_{k=1}^{n} \left\langle \mathbf{V}^k, \phi(\mathbf{x}) \right\rangle \mathbf{V}^k \equiv \phi(\mathbf{x})$$

L'analyse en composantes principales ayant généralement pour objectif de réduire la taille de la représentation, on fixe habituellement une borne inférieure $\lambda_{\min}$ aux valeurs propres : on considère alors que les n composantes principales associées aux valeurs propres $\lambda > \lambda_{\min}$ suffisent à rendre compte de la variance des données d'entrée [27].

Quel que soit le choix du nombre n de composantes principales, on peut montrer [28] que les directions des n premiers vecteurs propres (correspondant aux n plus grandes valeurs propres) couvrent autant de variance qu'il est possible de le faire avec n directions orthogonales, et que l'erreur quadratique de reconstruction définie par $\sum_{i=1}^{N} \left\| P_n\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i) \right\|^2$ est alors minimale.

Notons que, dans le domaine de la régression, la réduction de la dimension de l'espace des représentations constitue une forme de régularisation, dans la mesure où cela limite le nombre de fonctions candidates (diminution de la capacité de la famille de fonctions hypothèses).

5.1.3 Coordonnées d'un point dans la nouvelle base et reconstruction

Les composantes d'un vecteur $\mathbf{x} \in \mathbb{R}^d$, relativement à l'une des composantes principales $\mathbf{V}^k$ ainsi obtenue, sont déterminées en calculant la projection de son image par ϕ sur ce vecteur propre $\mathbf{V}^k$, conformément à la relation suivante :

$$\left\langle \mathbf{V}^k, \phi(\mathbf{x}) \right\rangle = \sum_{i=1}^{N} \alpha_i^k K(\mathbf{x}_i, \mathbf{x})$$

En pratique, on cherche souvent à travailler dans l'espace initial plutôt que dans l'espace image *F*. La question qui se pose est alors de déterminer un vecteur $\mathbf{z} \in \mathbb{R}^d$ vérifiant :

$$\phi(\mathbf{z}) = P_n \phi(\mathbf{x})$$

L'existence et l'unicité d'un tel **z** n'étant pas garanties, l'approche proposée par Milka & al. [28] consiste à minimiser par rapport à **z** la quantité suivante :

$$\rho(\mathbf{z}) = \left\| \phi(\mathbf{z}) - P_n \phi(\mathbf{x}) \right\|^2 = K(\mathbf{z}, \mathbf{z}) - 2 \sum_{k=1}^{n} \left\langle \mathbf{V}^k, \phi(\mathbf{x}) \right\rangle \sum_{i=1}^{N} \alpha_i^k K(\mathbf{z}, \mathbf{x}_i) + \Omega$$

où Ω est un terme qui ne dépend pas de **z**.

Il est possible [28] de calculer le gradient de $\rho(\mathbf{z})$, et donc de minimiser cette quantité à l'aide d'algorithmes classiques d'optimisation.

Cette technique de reconstruction permet finalement de mettre en œuvre des outils classiques d'analyse linéaire (régression linéaire par l'algorithme des moindres carrés, extraction de caractéristiques, débruitage, …) dans l'espace image *F*, puis de projeter dans l'espace initial les résultats de ces analyses.

La K-PCA est donc une méthode de noyau qui exploite la caractéristique importante des RKHS : la non-linéarité est prise en considération par une transformation qu'il n'est pas nécessaire de connaître analytiquement, mais qui est définie de manière implicite par le choix de la fonction noyau.

5.2 Moindres carrés partiels par noyaux (K-PLS)

La méthode des moindres carrés partiels (« PLS » pour Partial Least Squares [29], [30]) peut être considérée comme une extension de l'analyse en composantes principales, dans laquelle l'information relative aux sorties du processus à modéliser est prise en considération. Initialement adaptée au cas de la régression linéaire, la mise en oeuvre d'une transformation préliminaire d'espace, par une fonction ϕ associée au noyau *K*, permet d'étendre cette méthode au cas de la régression non linéaire.

Le principe de la PLS est, comme pour l'ACP, de réduire la dimension de l'espace de représentation, en ne retenant qu'un petit nombre de directions principales. En revanche, la manière de sélectionner ces directions diffère sensiblement entre ces deux méthodes.

Pour l'ACP, les composantes principales sont recherchées de manière à maximiser la variance des données d'entrée suivant leurs directions, ce qui peut se traduire par le problème d'optimisation suivant :

$$\begin{aligned} &\min_{\mathbf{V}} \sum_{i=1}^{N} \left\| \mathbf{x}_i - \left(\mathbf{x}_i \cdot \mathbf{V} \right) \mathbf{V} \right\|^2 \\ &\text{sous } \mathbf{V}^T \mathbf{V} = 1 \end{aligned} \tag{28}$$

L'équation (28) correspond en effet à la recherche d'une direction, définie par un vecteur **V** unitaire, suivant laquelle les projections de l'ensemble des données d'entrée présentent des résidus minimaux. La direction **V** obtenue par résolution de ce problème correspond alors à la valeur propre la plus importante de la matrice de covariance (25) introduite au paragraphe précédent. Une fois les composantes principales déterminées, l'algorithme des moindres carrés permet d'obtenir une fonction de régression linéaire dans cet espace transformé à dimension réduite.

Pour la PLS, l'objectif est de déterminer une fonction de régression, sous la forme suivante :

$$f\left(\mathbf{x}\right) = \sum_{k=1}^{n} \mathbf{V}^k \cdot \mathbf{x} \tag{29}$$

où $\mathbf{V}^k$ est la *k*-ième des *n* directions choisies.

Le choix des vecteurs $\mathbf{V}^k$ s'inspire alors de l'ACP : si la quantité $\mathbf{x}_i \cdot \mathbf{V}^1$ approche convenablement la sortie y_i pour tout *i*, le problème de régression (29) admet une solution développée sur un unique vecteur de base $\mathbf{V}^1$. En raisonnant par analogie avec le problème (28), on cherche donc à déterminer la première direction choisie comme solution du problème suivant :

$$\begin{aligned} &\min_{\mathbf{V}} \sum_{i=1}^{N} \left\| \mathbf{x}_i - y_i \mathbf{V} \right\|^2 \\ &\text{sous } \mathbf{V}^T \mathbf{V} = 1 \end{aligned} \tag{30}$$

En supposant les données d'entrée et de sortie centrées et réduites, on montre que le problème ci-dessus revient à maximiser la covariance entre le vecteur des entrées projetées sur **V**, soit **XV** (où **X** est la matrice des observations, constituée des *N* vecteurs ligne $\mathbf{x}_i^T$), et le vecteur des sorties **y**. La détermination de la première direction se ramène donc à la résolution du problème suivant :

$$\max_{\mathbf{V}} \operatorname{cov}(\mathbf{XV}, \mathbf{y}) \quad \text{sous } \mathbf{V}^T\mathbf{V} = 1 \tag{31}$$

dont la solution est donnée par :

$$\mathbf{V}^1 = \left(\mathbf{y}^T\mathbf{X}\mathbf{X}^T\mathbf{y}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{32}$$

À titre de comparaison, la recherche de la première composante principale pour l'ACP se traduit par un problème d'optimisation qui s'écrit :

$$\max_{\mathbf{V}} \operatorname{var}(\mathbf{XV}) \quad \text{sous } \mathbf{V}^T\mathbf{V} = 1$$

Une fois déterminée la direction $\mathbf{V}^1$, une méthode procédant par orthogonalisations successives permet la recherche de directions supplémentaires $\mathbf{V}^k$, en répétant la même procédure.

Une extension de la méthode PLS au cas non linéaire s'effectue par une transformation préliminaire d'espace ϕ, associée à une fonction noyau K. On modifie alors le problème d'optimisation (30) correspondant à la recherche de la première direction de la façon suivante :

$$\min_{\mathbf{V}} \sum_{i=1}^{N} \left\| \phi(\mathbf{x}_i) - y_i\mathbf{V} \right\|^2 \quad \text{sous } \mathbf{V}^T\mathbf{V} = 1 \tag{33}$$

Une fois de plus, il n'est pas nécessaire de connaître explicitement la transformation d'espace ϕ : son choix se résume à celui d'un noyau K, qui définira implicitement la transformation sous-jacente (« astuce des noyaux »).

6 Apprentissage de modèles récurrents : approche analytique

Tout ce qui précède est relatif à des modèles statiques. Nous allons à présent examiner le cas des modèles dynamiques, exprimés par des équations récurrentes, et proposer une approche analytique pour tenter de tenir compte du caractère récurrent du modèle dès la phase d'apprentissage. Cette étude portera sur les LSSVM, dont les caractéristiques permettent un traitement simple de la solution. Le traitement que nous allons détailler s'inspire des Recurrent Least Squares Support Vector Machines (R-LSSVM), introduit par Suykens & al. [31], mais diffère de cette technique dans la manière d'exprimer les contraintes associés au modèle récurrent. Notons d'ailleurs que le problème d'optimisation obtenu par Suykens & al.

n'a été utilisé dans la pratique que sous une version simplifiée, où la plupart des contraintes sont négligées afin d'obtenir un problème d'optimisation plus facile à résoudre.

Étant donnée une séquence de données $\left\{\mathbf{u}_k, y_k^p\right\}_{k=p+1}^{p+N}$, on s'intéresse à l'élaboration d'un modèle correspondant à l'hypothèse suivante :

$$\begin{cases} x_k^p = f\left(x_{k-1}^p, \ldots, x_{k-p}^p, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-p}\right) \\ y_k^p = x_k^p + w_k \end{cases} \tag{34}$$

où y_k^p est la sortie mesurée à l'instant discret *k*, où $\mathbf{u}_k$ est le vecteur des entrées de commande au même instant, où w_k est un bruit de sortie, et où *f* est une fonction non linéaire de ses arguments, qui sera réalisée par une méthode de noyau. De manière à simplifier les notations, nous omettrons l'entrée de commande (qui pourra être traitée de manière classique) et chercherons à élaborer des modèles prédicteurs de la forme :

$$y_k = f\left(y_{k-1}, \ldots, y_{k-p}\right) \tag{35}$$

Le modèle défini par la formule (35) se caractérise par le fait qu'il fait intervenir les sorties estimées aux instants précédents et non les mesures correspondantes de la sortie : c'est donc un modèle récurrent. Pour conserver une approche de type LSSVM, nous chercherons des solutions s'exprimant comme un produit scalaire dans un espace image, défini par une transformation ϕ :

$$y_k = \left\langle \mathbf{w}, \phi\left(\left[y_{k-1}, \ldots, y_{k-p}\right]^T\right)\right\rangle + b \tag{36}$$

où **w** est un vecteur de l'espace image, et où *b* est un terme de biais.

Le problème d'apprentissage associé à ce modèle s'exprime alors comme suit :

$$\begin{aligned} &\min_{\mathbf{w},b,\mathbf{e}} J\left(\mathbf{w},b,\mathbf{e}\right) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \gamma\frac{1}{2}\sum_{k=p+1}^{p+N} e_k^2 \\ &\text{sous } y_k^p - e_k = \left\langle \mathbf{w}, \phi\left(\mathbf{x}_{k-1|k-p} - \boldsymbol{\xi}_{k-1|k-p}\right)\right\rangle + b, \text{ pour } k = p+1, \ldots, p+N \end{aligned} \tag{37}$$

où $e_k = y_k^p - y_k$ est la variable d'erreur à l'instant *k*, où $\mathbf{x}_{k-1|k-p} = \left[y_{k-1}^p, \ldots, y_{k-p}^p\right]^T$, et où $\xi_{k-1|k-p} = \left[e_{k-1}, \ldots, e_{k-p}\right]^T$.

Le Lagrangien associé au problème d'optimisation (37) s'écrit :

$$\begin{aligned} L\left(\mathbf{w},b,\mathbf{e};\boldsymbol{\alpha}\right) &= J\left(\mathbf{w},b,\mathbf{e}\right) + \sum_{k=p+1}^{p+N} \alpha_{k-p}\left[y_k^p - e_k - \mathbf{w}^T\phi\left(\mathbf{x}_{k-1|k-p} - \boldsymbol{\xi}_{k-1|k-p}\right) - b\right] \\ &= \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \frac{\gamma}{2}\sum_{k=p+1}^{p+N} e_k^2 + \sum_{k=p+1}^{p+N} \alpha_{k-p}\left[y_k^p - e_k - \mathbf{w}^T\phi\left(\mathbf{x}_{k-1|k-p} - \boldsymbol{\xi}_{k-1|k-p}\right) - b\right] \end{aligned} \tag{38}$$

où $\boldsymbol{\alpha} = \left[\alpha_1, \ldots, \alpha_N\right]^T$ est le vecteur des N multiplicateurs de Lagrange associés aux N contraintes d'égalité du problème (37).

Les conditions d'optimalité de ce Lagrangien s'expriment alors de la façon suivante :

$$\begin{cases} \dfrac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \displaystyle\sum_{k=p+1}^{p+N} \alpha_{k-p} \phi\left(\mathbf{x}_{k-1|k-p} - \boldsymbol{\xi}_{k-1|k-p}\right) = \mathbf{0} \\ \dfrac{\partial L}{\partial b} = \displaystyle\sum_{k=p+1}^{p+N} \alpha_{k-p} = 0 \\ \dfrac{\partial L}{\partial \alpha_{k-p}} = y_k^p - e_k - \mathbf{w}^T \phi\left(\mathbf{x}_{k-1|k-p} - \boldsymbol{\xi}_{k-1|k-p}\right) - b = 0\,, \quad \forall k \in [\![p+1, p+N]\!] \\ \dfrac{\partial L}{\partial e_k} = \gamma e_k - \alpha_{k-p} - \dfrac{\partial}{\partial e_k}\left[\displaystyle\sum_{i=p+1}^{p+N} \alpha_{i-p} \mathbf{w}^T \phi\left(\mathbf{x}_{i-1|i-p} - \boldsymbol{\xi}_{i-1|i-p}\right)\right] = 0\,, \quad \forall k \in [\![p+1, p+N]\!] \end{cases} \tag{39}$$

Le caractère récurrent du modèle intervient pour le calcul de la dernière de ces relations d'optimalité : seules les sorties estimées aux instants postérieurs à k seront influencées par e_k, sur un horizon p correspondant à l'ordre du modèle. La dernière des relations (39) peut alors s'écrire :

$$\begin{aligned} \frac{\partial L}{\partial e_k} &= \gamma e_k - \alpha_{k-p} - \frac{\partial}{\partial e_k}\left[\sum_{i=k+1}^{p+k} \alpha_{i-p} \mathbf{w}^T \phi\left(\mathbf{x}_{i-1|i-p} - \boldsymbol{\xi}_{i-1|i-p}\right)\right] \\ &= \gamma e_k - \alpha_{k-p} - \sum_{i=1}^{p} \alpha_{k-p+i} \frac{\partial}{\partial e_k}\left[\mathbf{w}^T \phi\left(\mathbf{x}_{k-1+i|k-p+i} - \boldsymbol{\xi}_{k-1+i|k-p+i}\right)\right] = 0\,, \quad \forall k \in [\![p+1, p+N]\!] \end{aligned} \tag{40}$$

En substituant l'expression de $\mathbf{w}$ déterminée par la première des relations (39) dans les autres relations d'optimalité, on aboutit à :

$$\begin{cases} \mathbf{w} = \displaystyle\sum_{i=p+1}^{p+N} \alpha_{i-p} \phi\left(\mathbf{x}_{i-1|i-p} - \boldsymbol{\xi}_{i-1|i-p}\right) \\ \displaystyle\sum_{k=p+1}^{p+N} \alpha_{k-p} = 0 \\ y_k^p - e_k - b - \displaystyle\sum_{i=p+1}^{p+N} \alpha_{i-p} K\left(\mathbf{x}_{i-1|i-p} - \boldsymbol{\xi}_{i-1|i-p}, \mathbf{x}_{k-1|k-p} - \boldsymbol{\xi}_{k-1|k-p}\right) = 0\,, \quad \forall k \in [\![p+1, p+N]\!] \\ \gamma e_k - \alpha_{k-p} - \displaystyle\sum_{i=1}^{p} \sum_{j=p+1}^{p+N} \alpha_{k-p+i} \alpha_{j-p} \frac{\partial}{\partial e_k}\left[K\left(\mathbf{x}_{j-1|j-p} - \boldsymbol{\xi}_{j-1|j-p}, \mathbf{x}_{k-1+i|k-p+i} - \boldsymbol{\xi}_{k-1+i|k-p+i}\right)\right] = 0, \\ \qquad\qquad \forall k \in [\![p+1, p+N]\!] \end{cases} \tag{41}$$

où K est la fonction noyau associée à la transformation d'espace Φ.

Le Lagrangien dual s'obtient de même en substituant l'expression de $\mathbf{w}$ dans la relation (38) :

$$L^D\left(b,\mathbf{e};\boldsymbol{\alpha}\right)=\frac{\gamma}{2}\sum_{i=p+1}^{p+N}e_i^2+\sum_{i=p+1}^{p+N}\alpha_{i-p}\left(y_i^p-e_i-b\right)$$
$$-\frac{1}{2}\sum_{i=p+1}^{p+N}\sum_{j=p+1}^{p+N}\alpha_{i-p}\alpha_{j-p}K\left(\mathbf{x}_{i-1|i-p}-\xi_{i-1|i-p},\mathbf{x}_{j-1|j-p}-\xi_{j-1|j-p}\right) \tag{42}$$

On aboutit ainsi au nouveau problème d'optimisation suivant :

$$\max_{b,\mathbf{e};\boldsymbol{\alpha}} \quad L^D\left(b,\mathbf{e};\boldsymbol{\alpha}\right)$$
$$sous \quad \begin{cases} \displaystyle\sum_{k=p+1}^{p+N}\alpha_{k-p}=0 \\ \displaystyle y_k^p-e_k-b-\sum_{i=p+1}^{p+N}\alpha_{i-p}K\left(\mathbf{x}_{i-1|i-p}-\xi_{i-1|i-p},\mathbf{x}_{k-1|k-p}-\xi_{k-1|k-p}\right)=0\,, \quad \forall k\in[\![p+1,p+N]\!] \\ \displaystyle \gamma e_k-\alpha_{k-p}-\sum_{i=1}^{p}\sum_{j=p+1}^{p+N}\alpha_{k-p+i}\alpha_{j-p}\frac{\partial}{\partial e_k}\left[K\left(\mathbf{x}_{j-1|j-p}-\xi_{j-1|j-p},\mathbf{x}_{k-1+i|k-p+i}-\xi_{k-1+i|k-p+i}\right)\right]=0\,, \\ \qquad\qquad \forall k\in[\![p+1,p+N]\!] \end{cases} \tag{43}$$

Les caractéristiques de ce nouveau problème d'optimisation, notamment la non-convexité de la fonction objectif et des contraintes qui lui sont associées, ne permettent plus d'affirmer l'existence d'une solution unique. La fonction objectif L^D reste relativement simple, bien qu'elle ne soit plus quadratique vis-à-vis des variables d'erreurs e_k. La complexité du problème d'optimisation (43) réside essentiellement dans le calcul des $2N+1$ contraintes d'égalité. Étant donnée une fonction noyau K, il est néanmoins possible de calculer l'ensemble de ces contraintes (et en particulier les N dernières qui font intervenir les dérivées de K par rapport à e_k) en fonction des variables du problème et des hyperparamètres associés au noyau.

La différence principale entre le problème d'optimisation (43) et celui proposé par Suykens & al. réside dans la manière de calculer les dérivées du Lagrangien L vis-à-vis des variables d'erreurs e_k.

Il reste à tester empiriquement l'implémentation de cette méthode d'apprentissage, en particulier en termes de temps de calcul : bien que ce problème semble formellement soluble [32], il n'est pas exclu qu'il requière des temps de calcul extrêmement longs, difficilement conciliables avec les contraintes industrielles. Sous Matlab®, la fonction *fmincon* permet a priori de résoudre ce type de problème d'optimisation où la fonction objectif est continue et non linéaire, en présence de contraintes d'égalité également non linéaires.

7 Apprentissage de modèles récurrents : approche algorithmique

On s'intéresse dans cette partie à l'élaboration, par des méthodes de noyaux, de prédicteurs bouclés, capables de modéliser des processus dynamiques décrits par des équations récurrentes. Nous tenterons, dans un premier temps, d'adapter l'algorithme d'apprentissage semi dirigé, utilisé pour l'apprentissage de réseaux de neurones récurrents, au cas des modèles de noyaux. Nous décrirons en particulier la manière d'adopter une démarche maître-élève pour les modèles de noyaux, ainsi que les différentes variantes algorithmiques qui apparaîtront lors de la mise en œuvre de cette approche.

Au cours de cette étude, nous serons amenés à considérer différentes méthodes de noyaux, notamment les SVM, les LSSVM, et les sLSSVM.

7.1 Approche maître-élève pour les modèles de noyaux

Toute technique d'apprentissage nécessite des choix méthodologiques tels que celui d'une méthode d'optimisation, ou d'hyperparamètres, qui sont essentiellement indépendants de la nature, du nombre ou de la complexité des données à traiter. Ainsi, dans le cas de l'apprentissage de réseaux de neurones, il convient de choisir une méthode d'apprentissage (gradient simple, méthodes du second ordre), ainsi que des hyperparamètres tels que le pas du gradient, qui ne dépendent pas des détails du modèle. Afin de valider ces choix, voire de déceler d'éventuelles erreurs de programmation, il est utile d'appliquer la technique d'apprentissage à tester à un problème dit « maître-élève ». Dans ce but, on considère une base d'apprentissage constituée de données engendrées par un modèle, appelé *maître*, dont les paramètres sont connus, et dont la nature et la complexité sont les mêmes que celles du modèle *élève*, dont l'apprentissage doit être effectué par la procédure à tester.

Dans le cas des réseaux de neurones, cette approche se traduit par un modèle élève ayant la même structure que le maître, en termes de nombres de neurones cachés, de variables d'état et de variables exogènes, pour une structure de type canonique. L'apprentissage du modèle élève est alors considéré comme satisfaisant si, en fin d'apprentissage, le modèle élève possède des paramètres dont les valeurs sont les mêmes (à une permutation près) que celles des paramètres de son modèle maître.

Nous allons montrer que, dans le cas des modèles de noyaux, ce critère d'évaluation de la qualité de la procédure d'apprentissage ne peut pas s'appliquer, du fait que la solution s'appuie sur des vecteurs d'entrée particuliers, entachés de bruit, ou du fait d'une estimation à ε près des sorties (dans le cas des SVM).

Pour définir un réseau de neurones maître, il suffit de fixer le nombre de variables, de neurones cachés, et de sorties, ainsi que les valeurs des paramètres. Dans le cas des méthodes à noyaux, quelle que soit la variante envisagée (SVM, LSSVM, sLSSVM), le modèle, après apprentissage, s'écrit toujours de la façon suivante (en omettant le terme de biais) :

$$f(\mathbf{x}) = \sum_{i=1}^{N} \beta_i K(\mathbf{x}_i, \mathbf{x}) \tag{44}$$

où K est la fonction noyau utilisée (dépendant généralement d'un certain nombre d'hyper paramètres, tels que l'écart type dans le cas d'un noyau gaussien), et où les paramètres β_i sont les solutions du problème (dont l'expression dépend de la technique utilisée). Rappelons que l'on a en effet, comme indiqué aux paragraphes 3 et 4 :

- $f(\mathbf{x}) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x})$ pour des SVM
- $f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \mathbf{x})$ pour des LSSVM
- $f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i K(\mathbf{z}_i, \mathbf{x})$ pour des sLSSVM

Un modèle maître est donc défini par un ensemble $\{\beta_i, \mathbf{x}_i\}_{i=1}^{N}$, comprenant, outre les valeurs des paramètres β_i, les vecteurs de variables $\mathbf{x}_i$ sur lesquels s'appuie la solution.

Pour réaliser un modèle maître, nous partirons donc d'une base de données engendrée par un processus réel, avec laquelle nous effectuerons un premier apprentissage. Cette solution, qui caractérisera le modèle maître (élaborée avec des hyper paramètres choisis arbitrairement) servira alors à engendrer une nouvelle base de données, à laquelle pourra être ajouté un bruit de sortie additif, pour l'apprentissage de l'élève.

7.1.1 Cas des modèles statiques

Dans le cas de modèles statiques, les vecteurs de variables $\mathbf{x}_i$ sont constitués uniquement de variables exogènes. Dans le cas où ces variables ne sont pas entachées de bruit, on pourrait s'attendre à ce que la solution soit la même (en termes de valeurs des paramètres) entre le maître et l'élève. Toutefois, les méthodes d'optimisation utilisées pour effectuer l'apprentissage font systématiquement intervenir les valeurs de sorties de la base de données (algorithmes supervisés), et du fait que la solution est unique, la moindre différence entre les données du processus réel et celles du processus maître conduit à une solution différente. Les équations (22) pour les SVM, (14) pour les LSSVM, et (16) pour les sLSSVM illustrent cette dépendance de la solution vis-à-vis des données de sortie de la base d'apprentissage. L'ajout

d'un bruit de sortie à la base de données engendrée par le modèle maître se traduit alors par une solution unique qui est nécessairement différente.

D'autre part, la nature même de la technique de modélisation envisagée peut conditionner la solution. En effet, notamment dans le cas des SVM, les sorties correspondant aux vecteurs supports diffèrent des « mesures » de la base de données d'une quantité ε, ce qui conduit à des valeurs différentes des paramètres β_i. Cet effet est illustré sur la Figure 1, où la fonction génératrice des données est une fonction sinus cardinal.

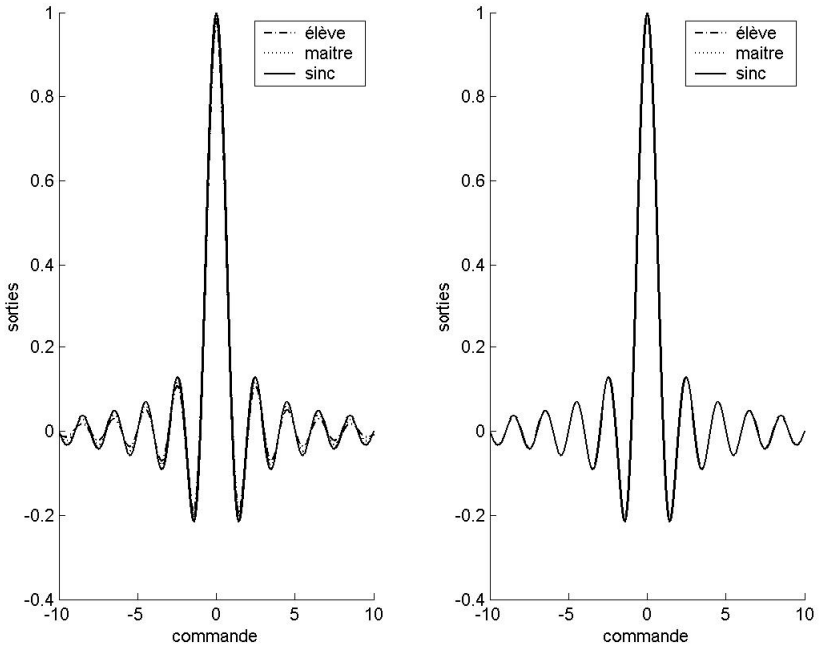


Figure 1 : *Gauche* : fonction génératrice *sinc* (en trait plein), modèles maître (en pointillés) et élève (en tirets pointillés) pour ε=0.1. *Droite* : idem pour ε=0 (les trois courbes apparaissent alors superposées). Modèles SVM (C=10, noyau gaussien σ_{noyau}=0.5, apprentissage sur 200 points).

En imposant la valeur ε=0, le nombre de vecteurs supports devient égal à la taille de la séquence d'apprentissage (200 points pour notre exemple). Bien que l'erreur quadratique moyenne entre maître et élève soit alors très faible, leurs paramètres β_i sont très nettement différents, comme illustré sur la Figure 2.

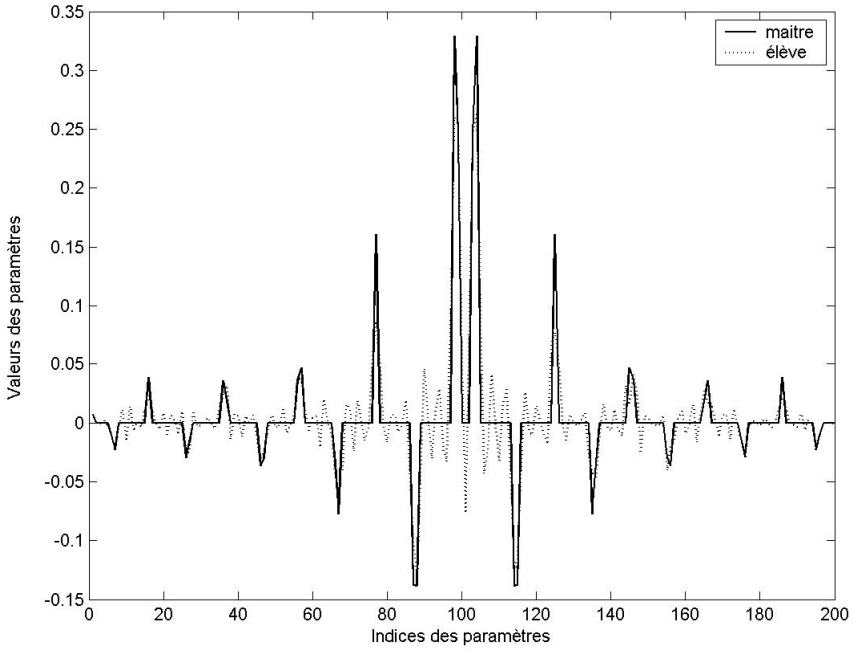


Figure 2 : Comparaison des paramètres des modèles maître et élève d'un sinus cardinal pour ε=0. Modèles SVM (C=10, noyau gaussien σ_{noyau}=0.5, apprentissage sur 200 points).

En conclusion, et dans le cas de modèles statiques, il ne faut pas s'attendre à ce que les paramètres solution du problème d'apprentissage soient les mêmes pour le modèle maître et le modèle élève.

7.1.2 Cas des modèles dynamiques

Si l'on considère un processus réel dynamique, décrit par des équations récurrentes, la différence entre les paramètres maître et élève a une origine supplémentaire, par rapport au cas de la modélisation statique. Soit un processus réel (sur lequel nous reviendrons au paragraphe 7.3) décrit par la représentation entrée-sortie suivante :

$$\begin{cases} x_n = f\left(x_{n-1}, x_{n-2}, u_{n-1}\right) = \dfrac{24 + y_{n-1}}{30} x_{n-1} - 0.8 \dfrac{u_{n-1}^2}{1 + u_{n-1}^2} x_{n-2} + 0.5 u_{n-1} \\ y_n = x_n + e_n \end{cases} \tag{45}$$

où y_n et u_n sont respectivement la sortie et l'entrée de commande à l'instant n, et où e_n est un bruit de sortie gaussien additif, d'écart type σ_{bruit}. Ce processus est stable si u reste dans l'intervalle [-10,10]. Si l'on réalise un apprentissage dirigé (cf. paragraphe 3.2 du Chapitre 3), le vecteur $\mathbf{x}_n$ des entrées utilisées pour l'apprentissage de ce processus s'écrit alors, à l'instant n :

$$\mathbf{x}_n = \left[y_{n-1}, y_{n-2}, u_{n-1} \right]^T$$

Ce vecteur fait intervenir les sorties « mesurées » du processus. En se référant aux équations qui définissent le problème d'apprentissage pour les différentes techniques envisagées (SVM, LSSVM, et sLSSVM), on constate que les problèmes maître et élève différent alors pour deux raisons :

- les valeurs des sorties désirées ne sont pas les mêmes entre le maître et l'élève (du fait de l'ajout d'un bruit de sortie, ou pour une raison inhérente à la technique de modélisation utilisée)
- les valeurs des vecteurs d'entrée, qui serviront potentiellement de supports, ne sont également pas les mêmes

Le caractère unique de la solution impose à nouveau des solutions différentes pour des problèmes d'apprentissage différents. On aura donc finalement des paramètres β_i dont les valeurs différent, mais également des vecteurs supports distincts pour le modèle maître et le modèle élève.

7.1.3 Conclusion sur l'approche maître-élève. Critère retenu

Nous venons de montrer que, dans le cas statique comme dans le cas dynamique, les solutions obtenues pour les modèles maître et élève ne peuvent être identiques. Cette approche présente néanmoins l'avantage de s'affranchir de l'influence des hyperparamètres propres au modèle. Pour poursuivre l'étude de la modélisation de processus dynamiques sans s'attarder à rechercher des hyper paramètres optimaux, qui seraient susceptibles de différer d'un modèle à l'autre et pourraient également dépendre de la base de données utilisée, nous conserverons cette démarche maître-élève en prenant pour critère non pas la similitude des solutions $\left\{ \beta_i, \mathbf{x}_i \right\}_{i=1}^N$, mais en étudiant l'évolution de l'erreur quadratique moyenne du modèle élève. Cette quantité ne sera à l'évidence pas la mieux adaptée au cas des SVM, qui nécessiteraient la prise en compte de la fonction de coût ε-insensible, mais l'essentiel de l'étude à venir portera sur les LSSVM et sLSSVM.

7.2 Description de l'algorithme d'apprentissage semi dirigé

7.2.1 Principe

Dans cette partie, nous proposons une méthodologie pour l'apprentissage de modèles dynamiques bouclés élaborés par des méthodes de noyaux. La méthode d'apprentissage

proposée est fondée sur une approche itérative. Pour tenir compte du caractère bouclé du prédicteur que l'on souhaite élaborer, l'idée est de partir d'un modèle réalisé par un apprentissage dirigé (c'est-à-dire dont les entrées d'état sont les mesures retardées de la sortie mesurée, pour un modèle utilisant une représentation entrée-sortie), et de remplacer les mesures de la sortie par les estimations du prédicteur, avant d'effectuer un nouvel apprentissage. Dans le cas des SVM, on peut justifier cette démarche en observant que les sorties correspondant aux vecteurs supports différent des mesures d'une quantité ε : en remplaçant la mesure de la sortie par cette estimation « à ε près » dans l'expression des vecteurs supports, on favorise donc la prise en compte du caractère récurrent du modèle.

Rappelons que, dans le cas des réseaux de neurones récurrents, l'algorithme d'apprentissage semi-dirigé se caractérise par une estimation des sorties du modèle récurrent menée simultanément avec la recherche itérative de la solution (par descente du gradient de la fonction de coût) : après chaque modification des paramètres du réseau, les sorties de tous les neurones du réseau, notamment les variables d'état, sont mises à jour (phase de propagation). C'est à partir de ces nouvelles valeurs qu'est calculé le gradient de la fonction de coût pour l'itération suivante de l'algorithme d'optimisation.

Pour les méthodes de noyaux qui nous intéressent, la solution du problème d'apprentissage est unique, et se ramène (pour les LSSVM ou les sLSSVM) au calcul de la pseudo-inverse d'une matrice. Il n'est alors pas envisageable de coupler recherche de la solution et propagation à travers le modèle bouclé.

Nous adopterons pour hypothèse une représentation entrée-sortie :

$$\begin{cases} x_k^p = f\left(\mathbf{u}_k, x_{k-1}^p, \ldots, x_{k-m}^p\right) \equiv f\left(\mathbf{x}_k^p\right) \\ y_k^p = x_k^p + w_k \end{cases}$$

où $\mathbf{u}_k$ représente le vecteur des entrées exogènes du processus à l'instant discret k, où $\mathbf{x}_k^p$ est le vecteur d'entrée au même instant, qui regroupe l'ensemble des entrées exogènes et d'état, et où w_k est un bruit de sortie supposé de moyenne nulle. En notant $\left\{\beta_k^{(n)}, \mathbf{x}_k^{(n)}\right\}_{k=1}^N$ la solution du problème d'apprentissage à la n-ième itération et $y = F_{\text{noyau}}\left(\left\{\beta_i^{(n)}, \mathbf{x}_i^{(n)}\right\}_{i=1}^N, \mathbf{x}\right)$ la valeur de la sortie y du modèle en un point $\mathbf{x}$, l'algorithme d'apprentissage proposé s'écrit alors :

- Itération 1 :

Apprentissage dirigé : base d'apprentissage $\left\{\mathbf{x}_k^{(1)}, y_k^p\right\}_{k=1}^N$ avec $\mathbf{x}_k^{(1)} = \left[\mathbf{u}_k, y_{k-1}^p, \ldots, y_{k-m}^p\right]^T$.

Solution : $\left\{\beta_k^{(1)}, \mathbf{x}_k^{(1)}\right\}_{k=1}^{N}$

- Itération 2 :

On calcule pour k variant de 1 à N : $y_k^{(1)} = F_{\text{noyau}}\left(\left\{\beta_i^{(1)}, \mathbf{x}_i^{(1)}\right\}_{i=1}^{N}, \mathbf{x}_k^{(1)}\right)$

Génération d'une nouvelle base d'apprentissage : $\left\{\mathbf{x}_k^{(2)}, y_k^p\right\}_{k=1}^{N}$ avec $\mathbf{x}_k^{(2)} = \left[\mathbf{u}_k, y_{k-1}^{(1)}, \ldots, y_{k-m}^{(1)}\right]^T$

Solution : $\left\{\beta_k^{(2)}, \mathbf{x}_k^{(2)}\right\}_{k=1}^{N}$

- Itération n :

On calcule pour k variant de 1 à N : $y_k^{(n-1)} = F_{\text{noyau}}\left(\left\{\beta_i^{(n-1)}, \mathbf{x}_i^{(n-1)}\right\}_{i=1}^{N}, \mathbf{x}_k^{(n-1)}\right)$

Génération d'une nouvelle base d'apprentissage : $\left\{\mathbf{x}_k^{(n)}, y_k^p\right\}_{k=1}^{N}$ avec $\mathbf{x}_k^{(n)} = \left[\mathbf{u}_k, y_{k-1}^{(n-1)}, \ldots, y_{k-m}^{(n-1)}\right]^T$

Solution : $\left\{\beta_k^{(n)}, \mathbf{x}_k^{(n)}\right\}_{k=1}^{N}$

Les propriétés de convergence de cet algorithme ne sont pas garanties. On se propose, comme dans le cas des réseaux de neurones récurrents, de vérifier empiriquement cette convergence.

7.2.2 Différentes variantes

On distingue plusieurs variantes de l'algorithme d'apprentissage semi dirigé décrit ci-dessus, suivant la manière dont sont prédites les sorties du modèle, qui seront utilisées pour mettre à jour les entrées d'état à l'itération suivante.

Prédicteur courant :

Dans la description précédente, les prédictions des sorties, pour construire la nouvelle base d'apprentissage de l'itération n, sont données par $y_k^{(n-1)} = F_{\text{noyau}}\left(\left\{\beta_i^{(n-1)}, \mathbf{x}_i^{(n-1)}\right\}_{i=1}^{N}, \mathbf{x}_k^{(n-1)}\right)$. Elles font donc intervenir le vecteur d'entrée $\mathbf{x}_k^{(n-1)}$, ce qui revient à utiliser les sorties prédites à l'itération précédente comme entrées d'état. On appellera cette variante « prédiction courante ».

Prédicteur bouclé :

Une alternative consiste à calculer les sorties du modèle par la relation $y_k^{(n-1)} = F_{\text{noyau}}\left(\left\{\beta_i^{(n-1)}, \mathbf{x}_i^{(n-1)}\right\}_{i=1}^{N}, \mathbf{x}_k^{(n)}\right)$, ce qui revient à prendre les estimations des sorties à l'itération courante dans le vecteur d'entrée $\mathbf{x}_k^{(n)}$. On appellera cette variante « prédiction bouclée ».

Prédicteur à 1 pas :

On peut également calculer les sorties du modèle par $y_k^{(n-1)} = F_{\text{noyau}}\left(\left\{\beta_i^{(n-1)}, \mathbf{x}_i^{(n-1)}\right\}_{i=1}^{N}, \mathbf{x}_k^{(1)}\right)$, ce qui revient à utiliser les sorties mesurées dans le vecteur d'entrée. On appellera cette variante « prédiction à 1 pas ».

Prédicteur sur un horizon variable :

Une solution intermédiaire entre prédicteur à 1 pas et prédicteur bouclé consiste à estimer les sorties du modèle, à l'itération *n*, sur un horizon intermédiaire *d*. Le calcul de la valeur prise à l'instant *k*+*d* par le modèle nécessite alors le « recalage », à l'instant *k*, de ses entrées d'état sur les sorties mesurées, c'est-à-dire le remplacement de son vecteur d'entrée $\mathbf{x}_k^{(n)}$ par le vecteur d'entrée initial $\mathbf{x}_k^{(1)}$. Les sorties sont alors calculées à la façon du prédicteur bouclé entre les instants *k* et *k*+*d*, et seule sera retenue cette dernière valeur, qui correspond à une prédiction sur un horizon *d*.

Notons que jusqu'à l'itération 2, qui nous intéressera par la suite, les variantes prédictions à 1 pas et prédictions courantes sont identiques. Le vecteur d'entrée $\mathbf{x}_k^{(1)}$ est constitué des entrées exogènes **u** et des *m* sorties mesurées aux instants précédents.

La Figure 3 présente, sous forme d'un dépliement temporel, le principe général de cet algorithme ainsi que les différentes variantes identifiées.

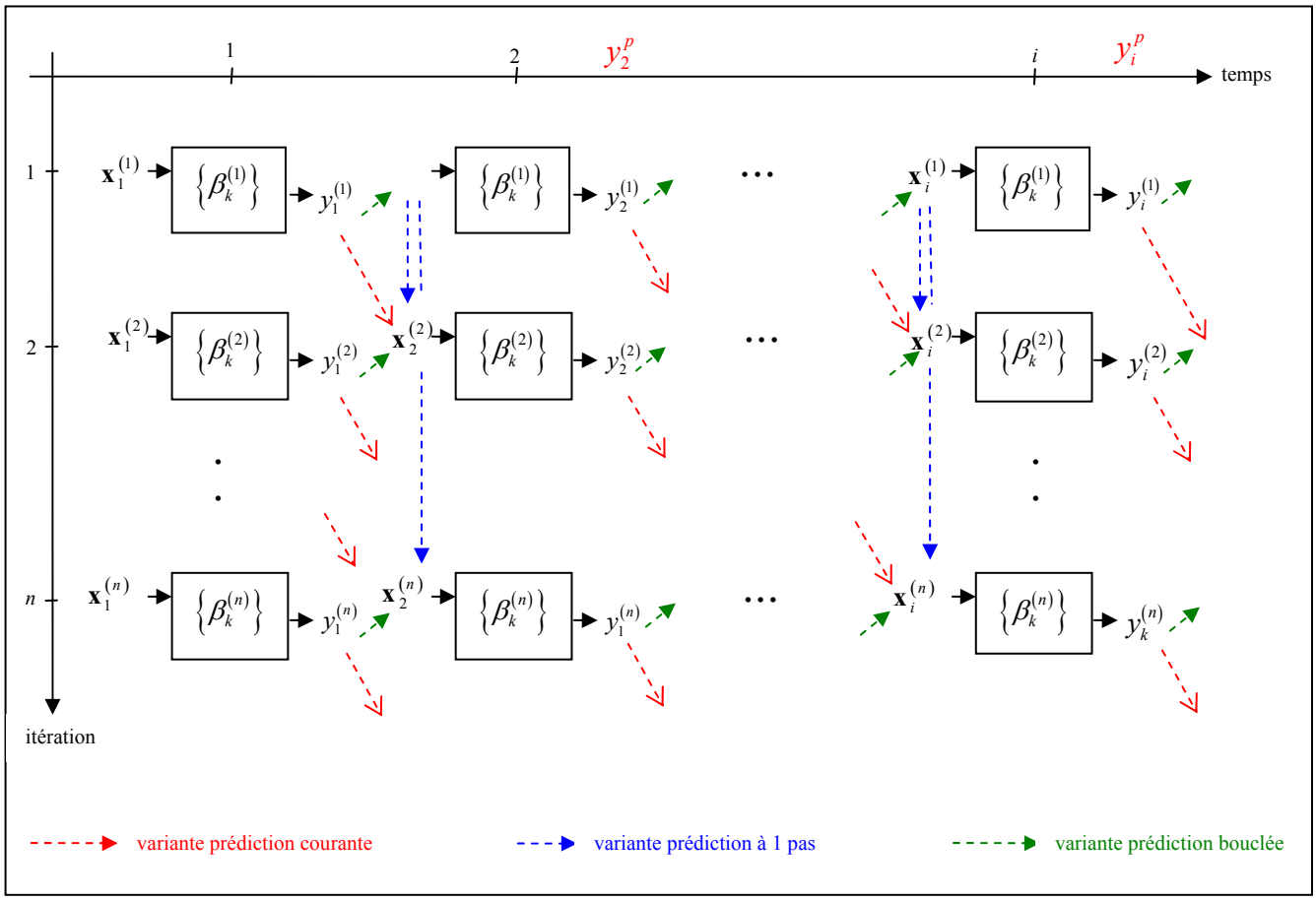


Figure 3 : schéma de principe de l'algorithme d'apprentissage semi dirigé applicable aux méthodes de noyaux, et différentes variantes envisagées suivant la façon de calculer les sorties du modèle.

7.3 Résultats et conclusion

En adoptant une démarche maître-élève, nous avons appliqué l'algorithme décrit au paragraphe précédent pour la modélisation d'un processus maître, élaboré à partir du processus dynamique défini par l'équation (45). L'étude a été effectuée en utilisant les méthodes de modélisation SVM et sLSSVM.

Dans un deuxième temps, nous mettrons en œuvre l'algorithme proposé pour modéliser le processus (45) lui-même en utilisant les méthodes SVM et sLSSVM.

7.3.1 Algorithme semi dirigé : approche maître-élève

Modélisation par SVM

À partir d'une séquence d'apprentissage $\{u_k, y_k^p\}_{k=1}^N$ de taille N=1000 points, issue du processus dynamique défini par l'équation (45) (avec un bruit gaussien d'écart-type σ_{bruit}=10$^-$

[2]), un modèle maître a été élaboré par apprentissage dirigé (algorithme SVM, C=10, ε=0.1, noyau gaussien d'écart-type σ_{noyau}=0.5). Une nouvelle séquence de données $\left\{u_k, y_k^{(maitre)}\right\}_{k=1}^{N}$ a ainsi pu être engendrée, en prédiction bouclée, sur la base de ce modèle maître, afin d'effectuer l'apprentissage du modèle élève à l'aide de l'algorithme semi dirigé à prédiction courante, en utilisant les mêmes hyper paramètres (algorithme SVM, C=10, ε=0.1, noyau gaussien d'écart-type σ_{noyau}=0.5). Un bruit de sortie gaussien additif, d'écart-type $\sigma_{bruit}=10^{-2}$, a été ajouté aux prédictions du modèle maître. La Figure 4 représente, pour l'élève, la Racine carrée de l'Erreur Quadratique Moyenne d'Apprentissage (REQMA) du prédicteur à 1 pas et celle du prédicteur bouclé en fonction du numéro de l'itération de l'apprentissage.

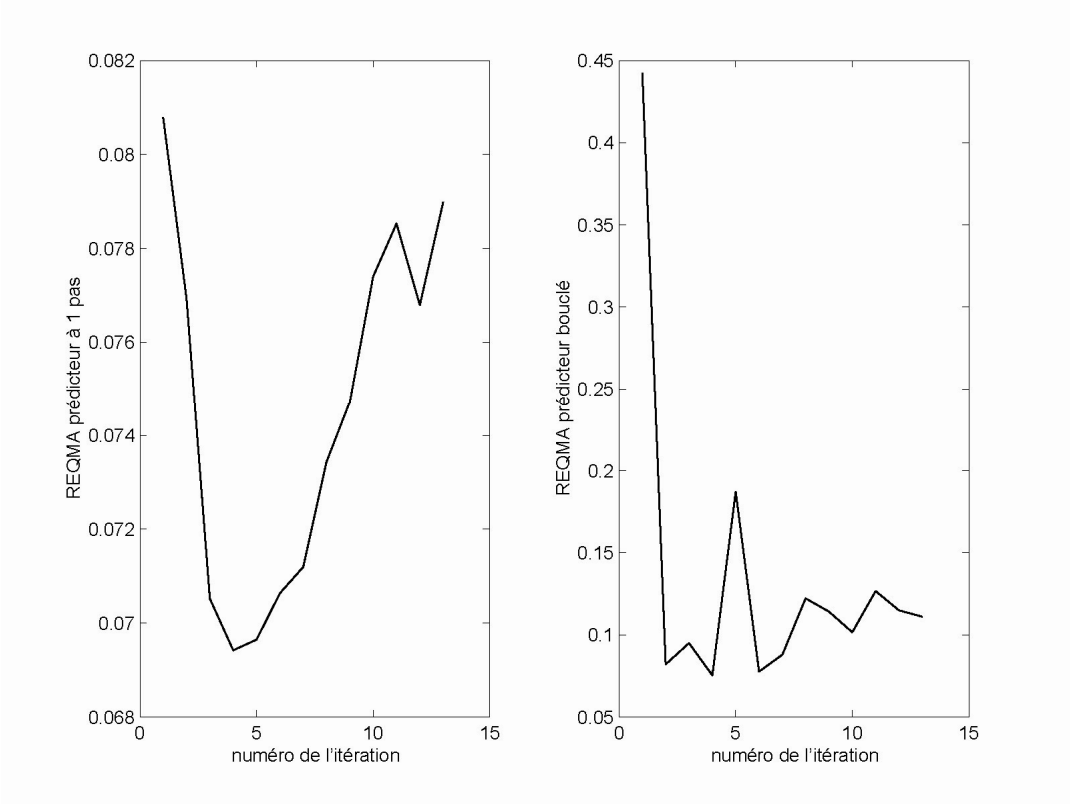


Figure 4 : REQMA du prédicteur à 1 pas et du prédicteur bouclé en fonction de l'itération de l'algorithme semi dirigé. Modèles SVM apprentissage semi dirigé à prédiction courante (C=10, ε=0.1, σ_{noyau}=0.5, base de données de taille N=1000 points).

On observe une décroissance de la REQMA du prédicteur bouclé et du prédicteur à 1 pas entre la première et la seconde itération. La REQMA du prédicteur bouclé à l'itération 2 est d'ailleurs très voisine de celle du prédicteur à 1 pas à l'itération 1 (c'est-à-dire pour le modèle issu d'un apprentissage dirigé). L'amélioration obtenue, entre les itérations 1 et 2, pour le

prédicteur bouclé est significative sur la base d'apprentissage. Une validation de ces modèles a été effectuée sur une seconde base de données. Le Tableau 1 résume les résultats obtenus.

	APPRENTISSAGE	**VALIDATION**
ITERATION 1 (dirigé)		
prédicteur à 1 pas	$8.1*10^{-2}$	$9.6*10^{-2}$
prédicteur bouclé	$44.3*10^{-2}$	$27.9*10^{-2}$
ITERATION 2 (semi dirigé)		
prédicteur bouclé	$8.2*10^{-2}$	$15.5*10^{-2}$

Tableau 1 : racines carrées des erreurs quadratiques moyennes d'apprentissage et de validation pour les prédicteurs bouclés et à 1 pas. Comparaison entre apprentissage dirigé et semi dirigé (à l'itération 2, variante prédiction courante). Modèles SVM (C=10, ε=0.1, σ_{noyau}=0.5).

Sur la séquence de validation, l'erreur commise par le prédicteur bouclé issu d'un apprentissage dirigé reste bien supérieure à celle du prédicteur bouclé à l'itération 2 (apprentissage semi dirigé). Il demeure néanmoins un facteur 2 entre les erreurs commises par ce dernier sur les séquences d'apprentissage et de validation.

Au-delà de l'itération 2, la REQMA du prédicteur bouclé ne semble pas converger et « oscille » autour de la valeur obtenue à l'itération 2, tandis que la REQMA du prédicteur à 1 pas augmente. Notons que pour réaliser des prédictions à 1 pas dans le cas sans bruit ou avec un bruit d'état, la mise en œuvre de l'algorithme proposé n'est pas nécessaire : un apprentissage dirigé fournira alors de meilleurs résultats. D'autre part, les valeurs prises par les paramètres β_k de l'élève ne semblent pas non plus converger. Le nombre de vecteurs supports varie autour de celui obtenu à l'itération 2, soit 266 (275 à l'itération 1) pour une séquence de longueur 1000, et a tendance à augmenter (345 à l'itération 13).

La Figure 5 présente les estimations fournies par le prédicteur bouclé à l'itération 2, et celles du prédicteur à 1 pas à l'itération 1 sur la séquence de validation.

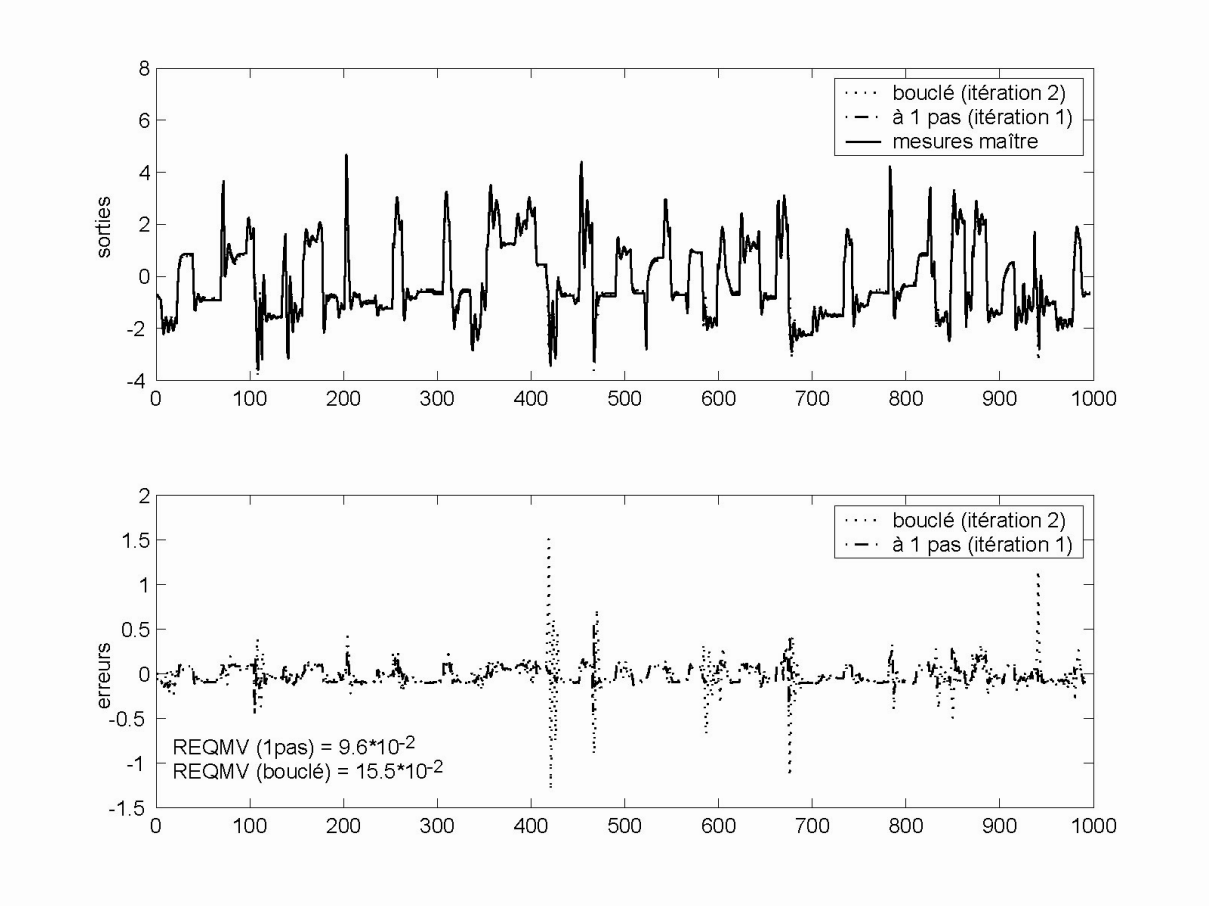


Figure 5 : *en haut :* estimations fournies par le prédicteur à 1 pas issu d'un apprentissage dirigé (itération 1) et par le prédicteur bouclé issu d'un apprentissage semi dirigé (itération 2), comparées aux mesures du modèle maître. *en bas :* erreurs correspondantes.

On constate que, en dehors de certaines zones où l'erreur du prédicteur bouclé est importante (ce qui se traduit par une REQMV du prédicteur bouclé issu d'un apprentissage semi dirigé double de celle du prédicteur à 1 pas issu d'un apprentissage dirigé), les performances de ces prédicteurs sont comparables.

La mise en oeuvre de l'algorithme d'apprentissage semi dirigé dans sa variante à prédiction courante, pour des modèles SVM et sur un exemple maître-élève, nous a permis de faire les observations suivantes :

- en termes d'erreur quadratique moyenne comme en termes de valeurs des coefficients solutions, l'algorithme ne semble pas présenter de propriétés de convergence.
- pour les modèles SVM, le nombre de vecteurs supports augmente au cours de l'algorithme.
- à l'itération 2, qui correspond simplement à un réapprentissage du modèle, la performance du prédicteur bouclé approche celle du prédicteur à 1 pas issu d'un apprentissage dirigé.

Le temps de calcul nécessaire à l'élaboration d'un tel modèle est cependant très important. Pour réduire ce temps de calcul, tout en contrôlant le nombre de vecteurs supports, et afin de tester les différentes variantes de l'algorithme, nous allons à présent appliquer ce même algorithme en utilisant des sLSSVM, qui présentent l'avantage, contrairement aux LSSVM, de conserver la propriété de parcimonie (nombre réduit de vecteurs supports).

Modélisation par sLSSVM

Un modèle sLSSVM maître du processus dynamique décrit par l'équation (45) a été élaboré par apprentissage dirigé (C=1, noyau gaussien d'écart-type σ_{noyau}=1, nombre de vecteurs supports NSV=165, apprentissage sur 998 points). Ce modèle, utilisé en prédicteur bouclé sur des séquences d'apprentissage et de validation, fournit de nouvelles bases pour construire un modèle élève, en utilisant les mêmes hyper paramètres (C=1, noyau gaussien σ_{noyau}=1, NSV=165). Un bruit de sortie additif a été ajouté aux sorties du modèle maître (distribution gaussienne d'écart type $\sigma_{bruit}=10^{-2}$).

Nous avons testé les 3 variantes d'apprentissage semi dirigé pour l'apprentissage de l'élève : mise à jour des entrées d'état par prédictions à 1 pas, courantes, ou bouclées. Pour chacune de ces trois variantes, les racines carrées des erreurs quadratiques moyennes d'apprentissage et de validation (REQMA et REQMV) des prédicteurs bouclés et à 1 pas sont représentées sur la Figure 6 en fonction du numéro de l'itération.

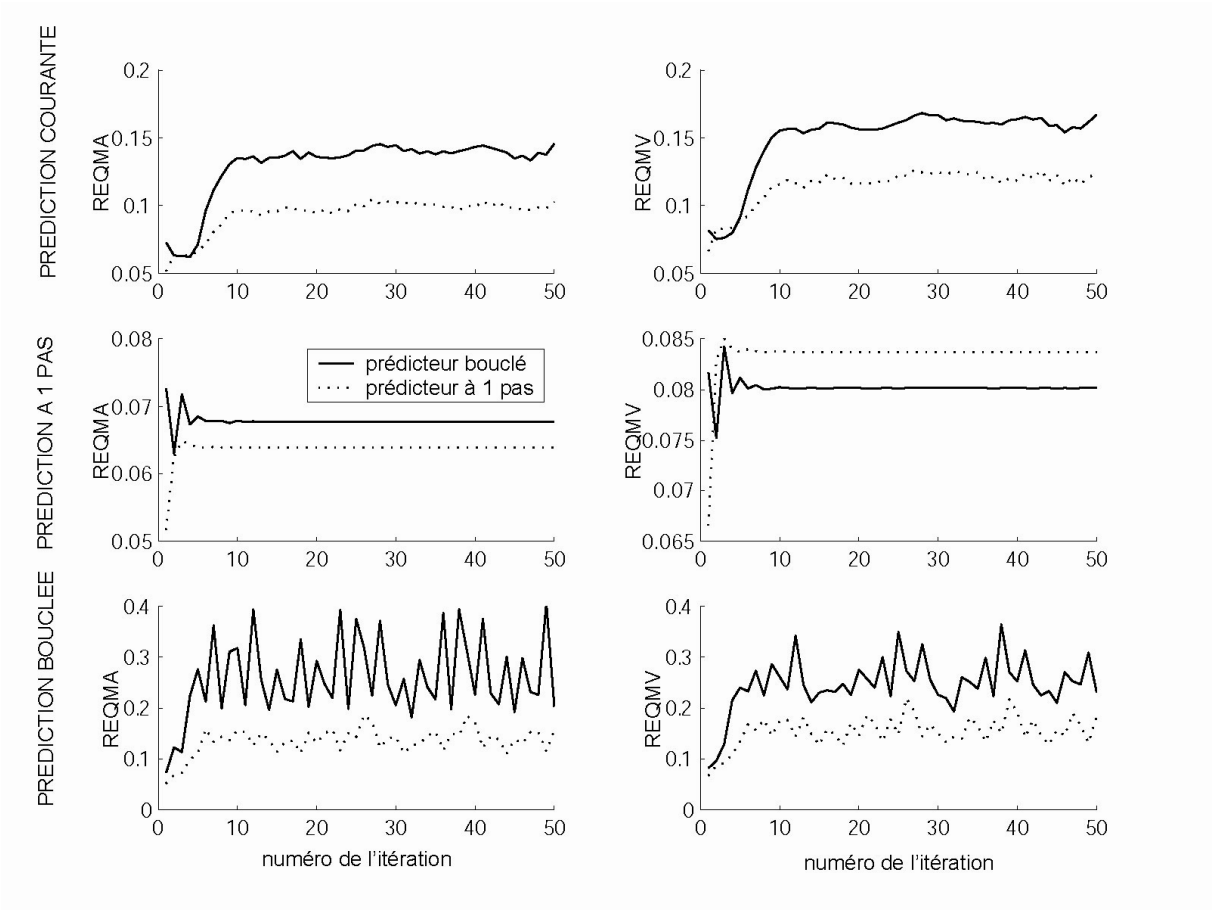


Figure 6 : Apprentissage de l'élève. Racines des erreurs quadratiques du prédicteur bouclé et du prédicteur à 1 pas en fonction de l'itération de l'algorithme semi dirigé. Modèles sLSSVM par apprentissage semi-dirigé (C=1, σ_{noyau} =1, NSV =165, bases de données d'apprentissage et de validation de taille N=996 points).

On observe, pour les modèles élève à prédictions courantes et à prédictions à 1 pas, une diminution de la REQMA et de la REQMV du prédicteur bouclé entre la première et la seconde itération (ces deux variantes sont d'ailleurs équivalentes sur les deux premières itérations). Cette tendance est la même que celle observée dans le cas des modèles SVM.

On note par ailleurs que, pour la variante à 1 pas, les REQM convergent après une dizaine d'itérations.

Pour la variante à prédictions bouclées, on constate par contre une augmentation importante des REQMA et REQMV sur les quelques premières itérations, et celles-ci ne semblent pas se stabiliser par la suite.

Le tableau ci-dessous compare les REQMA et REQMV des prédicteurs bouclés et à 1 pas entre un apprentissage dirigé (première itération) et semi dirigé (itération 2 de la variante à prédictions à 1 pas).

	APPRENTISSAGE	VALIDATION
ITERATION 1 (dirigé)		
prédicteur à 1 pas	$5.2*10^{-2}$	$6.7*10^{-2}$
prédicteur bouclé	$7.3*10^{-2}$	$8.2*10^{-2}$
ITERATION 2 (semi dirigé)		
prédicteur bouclé	$6.3*10^{-2}$	$7.5*10^{-2}$

Tableau 2 : racines carrées des erreurs quadratiques moyennes d'apprentissage et de validation pour les prédicteurs bouclés et à 1 pas. Comparaison entre apprentissage dirigé et semi dirigé (à l'itération 2, variante prédiction à 1 pas ou courante). Modèles sLSSVM (C=1, noyau gaussien σ_{noyau} =1, NSV=165, bases de données d'apprentissage et de validation de taille N=996 points).

Pour conclure sur la mise en œuvre de l'algorithme semi dirigé pour des modèles fondés sur des sLSSVM suivant une approche maître-élève, nous retiendrons que :

- l'algorithme ne présente pas de propriétés de convergence, dans ses variantes à prédictions courantes et à prédictions bouclées. La variante à prédictions à 1 pas se stabilise après une dizaine d'itérations, mais le résultat correspondant n'est pas satisfaisant en termes d'erreurs commises sur les bases d'apprentissage et de validation.
- à l'itération 2 de l'algorithme (dans sa variante à prédictions à 1 pas) la performance du prédicteur bouclé est améliorée par comparaison avec celle d'un modèle issu d'un apprentissage dirigé. La REQMV du prédicteur bouclé issu de cet apprentissage semi dirigé (soit $7.5*10^{-2}$) approche à nouveau celle du prédicteur à 1 pas issu d'un apprentissage dirigé (soit $6.7*10^{-2}$).
- le temps de calcul nécessaire à l'élaboration de ces modèles sLSSVM est bien moindre que celui des modèles SVM.

D'un point de vue qualitatif, la tendance est la même entre les cas SVM et sLSSVM : le résultat le plus performant correspond dans les deux cas à la deuxième itération de l'algorithme semi dirigé (qui consiste donc en un réapprentissage du modèle), et se traduit par une REQMV du prédicteur bouclé qui approche celle du prédicteur à 1 pas issu d'un apprentissage dirigé.

7.3.2 Algorithme semi dirigé : processus dynamique simulé

L'approche maître élève, décrite dans le paragraphe précédent, a permis de mettre en évidence l'amélioration apportée par l'algorithme semi dirigé à l'itération 2, en termes de performance

du prédicteur bouclé. Nous allons à présent appliquer cet algorithme à un processus simulé, en nous limitant au résultat de l'itération 2. Nous utiliserons à cette fin des séquences de données engendré par le processus dynamique non linéaire défini par l'équation (45), en présence d'un bruit additif de sortie (distribution gaussienne d'écart-type $\sigma_{bruit}=10^{-1}$). Nous serons alors amené à déterminer les valeurs optimales des hyper paramètres associés aux modèles étudiés.

Modélisation par SVM

À partir d'une séquence d'apprentissage de longueur 300, nous avons comparé les REQM des prédicteurs à 1 pas et bouclé d'un modèle SVM élaboré par apprentissage dirigé, à la REQM du prédicteur bouclé d'un modèle SVM correspondant à l'itération 2 de l'algorithme d'apprentissage semi dirigé. Les données d'entrée et de sortie de la séquence d'apprentissage ont préalablement été centrées et réduites. Le noyau employé est de type gaussien, avec un écart type variant entre $\sigma_{noyau}=0.1$ et $\sigma_{noyau}=10$. La fonction de coût employée est la fonction ε-insensible, avec $\varepsilon=0.1$.La valeur de l'hyperparamètre établissant un compromis entre régularisation et pénalisation des erreurs a été prise égale à $C=10$. La Figure 7 présente les résultats de cette étude, qui porte essentiellement sur les données d'apprentissage : l'objectif n'était pas ici d'étudier la capacité de généralisation des modèles obtenus, mais de comparer la qualité des apprentissages, en terme de REQM du prédicteur bouclé.

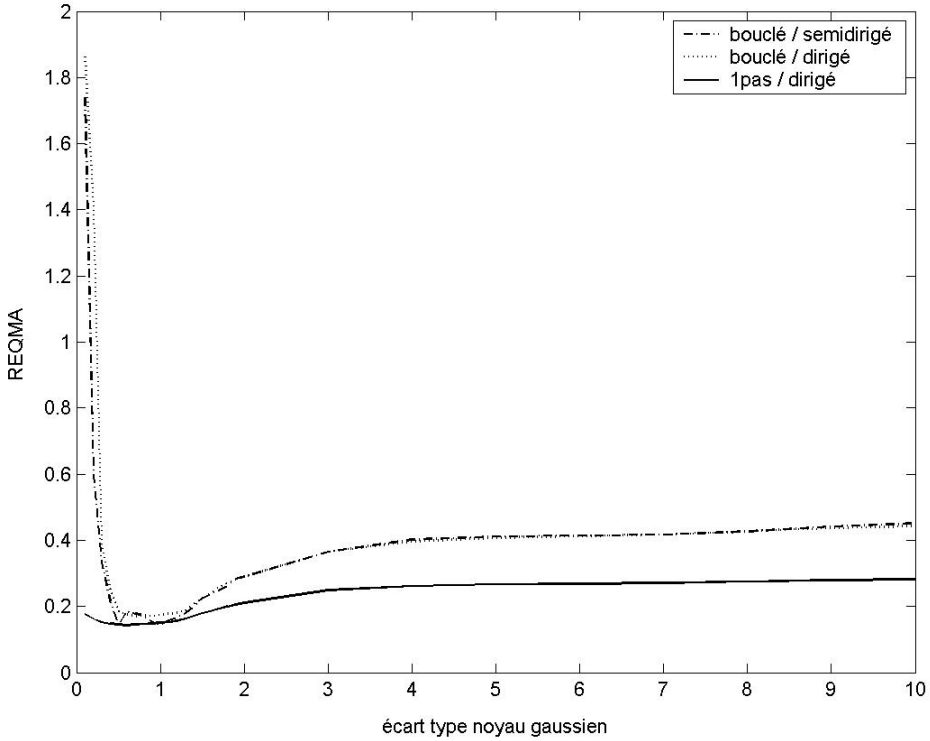


Figure 7 : REQMA des prédicteurs bouclé et à 1 pas. Comparaison entre apprentissage dirigé et semi dirigé (itération 2). Modèles SVM (noyau gaussien, C=10, ε=0.1, N=300).

Le Tableau 3 indique les valeurs optimales (au sens du minimum de la REQMA) de l'écart-type du noyau gaussien pour les apprentissages dirigé et semi dirigé, ainsi que les valeurs correspondantes des REQMA.

APPRENTISSAGE	Valeur optimale de σ_{noyau}	REQMA
Dirigé	0.6 0.7	0.14 (prédicteur à 1 pas) 0.17 (prédicteur bouclé)
Semi dirigé (itération 2)	0.5	0.14 (prédicteur bouclé)

Tableau 3 : valeurs optimales de l'écart type du noyau gaussien et REQMA correspondantes. Comparaison entre apprentissages dirigé et semi dirigé (itération 2), modèles SVM.

On constate à nouveau que la REQMA du prédicteur bouclé issu d'un apprentissage semi dirigé prend une valeur très proche de celle du prédicteur à 1 pas issu d'un apprentissage dirigé (dans ce cas, elles sont égales). Cette observation, qui porte sur la modélisation d'un processus simulé, confirme les conclusions obtenues dans le cas du problème maître-élève.

Modélisation par sLSSVM

En reproduisant la procédure décrite dans le cas de la modélisation SVM, nous avons élaboré des modèles sLSSVM (noyau gaussien, C=10, NSV=82) en faisant varier l'écart-type du noyau gaussien entre σ_{noyau}=0.1 et σ_{noyau}=2. Le Tableau 4 compare les performances des prédicteurs issus d'apprentissages dirigé et semi-dirigé.

APPRENTISSAGE	Valeur optimale de σ_{noyau}	REQMA
Dirigé	0.9 0.8	0.19 (prédicteur à 1 pas) 0.23 (prédicteur bouclé)
Semi dirigé (itération 2)	0.7	0.21 (prédicteur bouclé)

Tableau 4 : valeurs optimales de l'écart type du noyau gaussien et REQMA correspondantes. Comparaison entre apprentissages dirigé et semi dirigé (itération 2), modèles sLSSVM.

La REQMA du prédicteur bouclé issu d'un apprentissage semi dirigé est également inférieure à celle du prédicteur bouclé issu d'un apprentissage dirigé.

À titre de comparaison, un réseau de neurones récurrents (avec N_s=2 entrées d'état, et N_{cc}=6 neurones cachés) élaboré par apprentissage semi-dirigé sur la même séquence de données conduit à une REQMA du prédicteur bouclé égale à : REQMA=9.5*10^{-2}, soit une valeur très proche de la variance du bruit de sortie (σ_{bruit}=10^{-1}), et 2 fois moindre que celle obtenue avec des sLSSVM. L'adaptation de l'algorithme d'apprentissage semi-dirigé au cas des méthodes de noyaux ne fournit donc pas une performance équivalente à celle obtenue avec des réseaux de neurones récurrents.

8 Conclusion

Nous avons introduit dans ce chapitre plusieurs méthodes de régression non linéaire, appelées méthodes de noyaux, qui font intervenir une transformation d'espace implicite, à partir de laquelle il est possible de mettre en œuvre des outils de modélisation linéaires. Cette présentation ne prétend pas être exhaustive, mais offre une introduction aux méthodes les plus employées dans ce domaine.

Nous nous sommes intéressés ensuite à la manière d'adapter certains de ces algorithmes d'apprentissage au cas de modèles dynamiques, décrits par des équations aux différences récurrentes. L'objectif était de parvenir à élaborer des prédicteurs bouclés plus performants que ceux obtenus par apprentissage dirigé. La première approche a consisté en un traitement analytique du problème d'apprentissage relatif aux modèles récurrents, dans le cas LSSVM. Nous avons abouti à l'expression d'un problème d'optimisation non linéaire sous contraintes,

qui semble a priori utilisable numériquement, mais pourrait nécessiter des temps de calculs prohibitifs. Il reste à vérifier expérimentalement l'intérêt et la performance de cet algorithme.
La seconde approche a consisté en une tentative d'adaptation de l'algorithme d'apprentissage semi dirigé utilisé pour l'élaboration de modèles neuronaux récurrents. Malgré de profondes différences entre ce dernier et l'algorithme ainsi proposé, il s'avère qu'il est possible d'obtenir, avec un prédicteur ainsi bouclé, de meilleurs résultats qu'avec un prédicteur non bouclé ; ceux-ci restent néanmoins de moins bonne qualité que ceux d'un réseau de neurones récurrent.

Bibliographie

[1] V.N. Vapnik. *Estimation of dependences based on empirical data.* Nauka, Moscow, 1979 (traduction anglaise: Springer Verlag, Berlin, 1982).

[2] S. Geman, E. Bienenstock, R. Doursat. *Neural networks and the bias / variance dilemma.* Neural Computation 4, pp. 1-58, 1992.

[3] V.N. Vapnik, A.Y. Chervonenkis. *On the uniform convergence of relatives frequences of events to their probabilities.* Th. Prob. And its Applications, 17(2), pp. 264-280, 1971.

[4] I. Guyon, V. Vapnik, B. Boser, L. Bottou, S.A. Solla. *Structural risk minimization for character recognition.* Advances in Neural Information Processing Systems, 4, pp. 471-479, 1992.

[5] B. E. Boser, I. M. Guyon, V. N. Vapnik. *A training algorithm for optimal margin classifiers*. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, Pittsburgh, PA, pp. 144-152, ACM Press, 1992.

[6] C. Cortes, V. Vapnik. *Support vector networks.* Machine Learning, 20, pp. 273-297, 1995.

[7] K. R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, V. Vapnik. *Predicting time series with support vector machines.* In Proceedings of the International Conference on Artificial Neural Networks, pp. 999-1004, Springer Lecture Notes in Computer Science, vol. 1327, 1997.

[8] A. Smola, B. Schölkopf, K.R. Müller. *General cost functions for support vector regression*. In Proceedings of the Ninth Australian Conference on Neural Networks, Brisbane, Australia, pp. 79-83, 1998.

[9] N. Cristianini, J. Shawe-Taylor. *Support Vector Machines and other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[10] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, V. Vapnik. *Comparing support vector machines with Gaussian kernels to radial basis function classifiers.* IEEE Transaction on Signal Processing, Vol. 45, pp. 2758-2765, 1997.

[11] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, V. Vapnik. *Support Vector Regression Machines*. In Advances in Neural Information Processing Systems, Vol. 9, pp. 155-161, 1997.

[12] J. Mercer. *Functions of positive and negative type and their connection with the theory of integral equations.* Philosophical Transactions of the Royal Society, London, vol. 209, pp. 415-446, 1909.

[13] T. Evgeniou, T. Poggio, M. Pontil, A. Verri. *Regularization and statistical learning theory for data analysis*. Computational Statistics & Data Analysis, Vol. 38, Issue 4, pp. 421-432, Elsevier, 2002.

[14] G.S. Kimeldorf, G. Wahba. *Some results on Tchebycheffian spline functions.* Journal of Mathematical and Analytical Applications, 33, pp. 82-95, 1971.

[15] A. Smola, B. Schölkopf, K.R. Müller. *The connection between regularization operators and support vector kernels.* Neural Networks, vol. 11, pp. 637-649, 1998.

[16] A. N. Tikhonov, V. Y. Arsenin. *Solutions of Ill-posed problems.* W. H. Winston, Washington D.C. (1977).

[17] C. Saunders, A. Gammerman, V. Vovk. *Ridge regression learning algorithm in dual variables.* In Proceedings of the 15th International Conference on Machine Learning, Madison, WI, July 24-27, pp.24-27, 1998.

[18] M. Minoux. *Programmation mathématique: théorie et algorithmes.* Dunod, 1983.

[19] J.A.K. Suykens, L. Lukas, J. Vandewalle. *Sparse approximation using least-squares support vector machines.* In Proceedings of the IEEE International Symposium on Circuits and Systems, Geneva, Switzerland, May 2000, pp. 11757-11760, 2000.

[20] G.C. Cawley, N.L.C. Talbot. *Fast exact leave-one-out cross-validation of sparse least-squares support vector machines*. Neural Networks 17, pp.1467-1475, 2004.

[21] G. Baudat, F. Anouar. *Kernel-based methods and function approximation.* Proceedings of the IJCNN, Washington, DC, July 2001, pp. 1244-1249, 2001.

[22] J.A.K. Suykens, L. Lukas, J. Vandewalle. *Weighted least squares support vector machines: robustness and sparse approximation.* Neurocomputing, 48(1-4), pp. 85-105, 2002.

[23] A.J. Smola, B. Scholkopf. *A tutorial on support vector regression.* NEUROCOLT Technical Report NC-TR-98-030, Royal Holloway College, London, 1998.

[24] O. Chapelle, V.N. Vapnik, O. Bousquet, S. Mukherjee. *Choosing Multiple Parameters for Support Vector Machines.* Springer, 2002.

[25] M. Pontil, S. Mukherjee, F. Girosi. *On the noise model of support vector machine regression.* In Proceedings of the 11th International Conference on Algorithmic Learning Theory (ALT 2000), Sydney, Australia, December 2000,

[26] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New-York, 1986.

[27] G. Saporta. *Probabilités, analyse des données et statistique.* Editions Technip, 1990.

[28] S. Milka, B. Schölkopf, A. Smola, K.R. Müller, M. Scholz, G. Rätsch. *Kernel PCA and De-Noising in Feature Spaces.* NIPS, MIT Press, 1999.

[29] R. Rosipal, L.J. Trejo. *Kernel partial least squares regression in reproducing kernel Hilbert space.* Journal of Machine Learning Research 2, pp. 97-123, 2001.

[30] K.P. Bennett, M.J. Embrechts. *An optimization perspective on kernel partial least squares regression.* In Proceedings of the NATO Advanced Study Institute on Learning Theory and Practice (LTP 2002), 2003.

[31] J.A.K Suykens, J. Vandewalle. *Recurrent least squares support vector machines.* In IEEE Transaction on Circuits and Systems-I, Vol. 47, No. 7, pp. 1109-1114, July 2000.

[32] T.F. Coleman, Y. Li. *On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds*. Mathematical Programming, Vol. 67, No. 2, pp. 189-224, 1994.

Conclusion

Les travaux menés durant cette thèse ont cherché à concilier deux impératifs :

- d'une part, la prise en compte des intérêts et contraintes industrielles liées à l'utilisation et à la mise en œuvre des modèles ou système de contrôle par apprentissage, dans un contexte automobile.
- et d'autre part l'objectif de développer des méthodes originales de modélisation et de commande, pour contribuer à des problématiques actuelles dans le domaine de l'apprentissage.

Nous avons vu que les considérations de stabilité, de coût, de sûreté de fonctionnement, et de capacités limitées des calculateurs embarqués, déterminent en grande partie l'utilisation industrielle qui peut être faite d'un modèle ou d'une procédure donnés. Pour identifier des domaines d'applications possibles pour les méthodes de modélisation par apprentissage dans l'industrie automobile, il s'agit donc d'intégrer ces contraintes, et de démontrer un gain clair en terme de prestation réalisée, ou de coût de développement et de mise au point.

Nous nous sommes intéressés dans un premier temps aux réseaux de neurones formels, qui ont aujourd'hui atteint un niveau de maturité qui en fait un outil de choix pour aborder des problèmes de modélisation non linéaire délicats : les fondements théoriques sont solides (propriété d'approximation universelle et de parcimonie), et leur mise en œuvre ne soulève pas de difficultés particulières (existence d'algorithmes pour l'apprentissage de modèles statiques et dynamiques). Nous avons appliqué cette méthode à la modélisation de deux processus industriels :

- l'estimation de la température en un point particulier de la ligne d'échappement. La performance du modèle neuronal obtenu respecte le cahier des charges. De plus, l'utilisation en embarqué d'un tel modèle permettrait de faire l'économie d'un thermocouple supplémentaire, ce qui constitue un intérêt industriel évident.
- l'estimation des débits de différents polluants en sortie d'échappement. Les modèles neuronaux correspondants n'ont ici pas vocation à être utilisé en embarqué, mais constituent un outil permettant de réduire le nombre d'essais nécessaires à la mise au

point d'un véhicule. La performance des modèles obtenus est tout à fait acceptable, compte tenu de la précision des mesures correspondantes.

Ces deux applications participent à la maîtrise des émissions polluantes par les véhicules automobiles, à des niveaux différents : soit de manière directe, en intervenant dans les stratégies de contrôle des éléments de dépollution implantés sur la ligne d'échappement ; soit de manière indirecte, en permettant une mise au point plus rapide et moins coûteuse des réglages du moteur.

Nous avons par ailleurs développé un système de commande optimale en boucle ouverte, qui, sur la base d'un modèle neuronal, permet de limiter les variations rapides de la sortie d'un processus dynamique. La mise en œuvre de cette stratégie de contrôle nous a amené à proposer une méthode originale pour le calcul exact de la matrice Hessienne, dans le cas général de modèles décrits par des équations récurrentes. Les performances de ce système de commande ont été testées sur un processus simulé ; il n'a cependant pas été possible de réaliser des tests sur un véhicule réel, pour son application initialement prévue (système anti à-coups).

Dans un deuxième temps, nous nous sommes intéressés aux machines à vecteurs supports, et à la manière d'adapter les algorithmes d'apprentissage à la modélisation de processus dynamiques. Les caractéristiques des SVM (notamment l'unicité de la solution, la régularisation inhérente à la manière de les élaborer, ou la prise en compte du caractère non linéaire de manière implicite) et les performances qu'ont pu montrer ces outils sur différents benchmarks, justifient l'intérêt qui leur est porté. La question de l'application de ces techniques au cas de processus dynamiques décrits par des équations récurrentes reste cependant ouverte. Nous avons proposé dans ce mémoire deux approches permettant de prendre en compte ce caractère récurrent dès la phase d'apprentissage :

- une approche algorithmique, en s'inspirant de la démarche d'apprentissage semi dirigé utilisée dans le cas des réseaux de neurones récurrents. Bien que cette variante apporte une amélioration de la performance des prédicteurs correspondants, elle n'a pas permis d'atteindre celle des réseaux de neurones récurrents.
- une approche analytique, en définissant un nouveau problème d'apprentissage qui intègre le caractère récurrent du modèle, dans le cas particuliers des LSSVM. On aboutit alors à un problème d'optimisation non convexe, dont l'unicité de la solution n'est plus garantie, et qui pourrait requérir des temps de calculs prohibitifs. Il reste à tester cette variante pour conclure sur la pertinence de l'approche.

Nous avons donc pu démontrer dans cette thèse l'intérêt apporté par l'utilisation de techniques de modélisation et de commande par apprentissage dans le contexte de l'industrie automobile : outre le fait qu'elles permettent d'aborder la modélisation de processus complexes, ces techniques permettent également de réduire le temps et les coûts de développement d'un véhicule, ce qui est aujourd'hui un enjeu stratégique majeur.

Annexe 1 :

calcul exact de la matrice Hessienne d'une fonction réalisée par un réseau récurrent

Exact computation of the Hessian matrix of discrete-time dynamical models: application to optimal control

Marc LUCEA[a,b], Yacine OUSSAR[a], Gérard DREYFUS[a]

[a] ESPCI-Paristech, Laboratoire d'Électronique, 10 rue Vauquelin, 75005 Paris, France

[b] RENAULT, Direction de la Recherche, 1 avenue du Golf, 78280 Guyancourt, France

Abstract

The Hessian matrix of a model is widely used in optimization algorithms, in order to increase their speed of convergence. In this paper, we introduce a general framework allowing exact computation of this Hessian matrix, for a large class of discrete-time dynamical models, described by recurrent equations. As an illustration, the design of an open-loop optimal control system based on a recurrent neural network model, intended to optimize the variations of the output of a plant, is described.

1 INTRODUCTION

The Hessian matrix (the matrix of the second derivatives of a function with respect to its parameters), is a basic ingredient of second-order optimization algorithms, which are very attractive due to their speed of convergence ([2], [22]). Its computation is required for performing a second order approximation of the function, in the vicinity of a local minimum ([15], [18]). Approximate computations of the Hessian matrix have also been advocated for fast retraining of neural networks [3], for model pruning [13], or for Bayesian estimations of regularization parameters [14]. For feedforward neural networks (also termed multi-layer Perceptrons), the exact computation of the Hessian matrix of the least squares cost function with respect to its parameters was described by Bishop [4]. For dynamical models, however, the computation of this Hessian matrix is not straightforward, since it requires taking into account the past states of the model.

In part 2 of the present paper, we describe a general framework for computing the Hessian matrix, with respect to the values of an exogenous variable that is considered as a parameter, for a large class of discrete-time dynamical models. Two illustrations are shown in Part 3: first, we compute formally the Hessian matrix of a knowledge-based model; the second illustration is the optimization of a cost function intended to minimize the variations of the output of a plant, in a neural optimal control approach.

2 EXACT COMPUTATION OF THE HESSIAN MATRIX

Since the computation of the Hessian matrix can be useful in many different areas of engineering, we will first describe cursorily our specific motivation, and how the exact computation of the Hessian matrix fits into that framework. The second subsection will be devoted to the computation of the Hessian matrix of discrete-time dynamical models in the general case, which will be applied to recurrent neural networks in the third subsection.

2.1 Motivation

We are interested in minimizing the rate of variation of the output y^p of a plant in response to a variation of a control variable u, while preserving the accuracy of the response. In the automotive field, for instance, the fast variations of the torque response of an engine to a sharp variation of the control variable (namely the position of the accelerator pedal), must be minimized in order to preserve the driving comfort.

We assume that a discrete-time dynamical model of the plant is available. The proposed method, based on an open-loop optimal control scheme, consists in replacing, at each instant k, the user-defined future reference signal r_{k+1} with some new control signal $\hat{u}_{k+1}$, whose value minimizes an appropriate cost function $J(u_{k+1}, \ldots, u_{k+N})$. The approach is summarized on Figure 1.

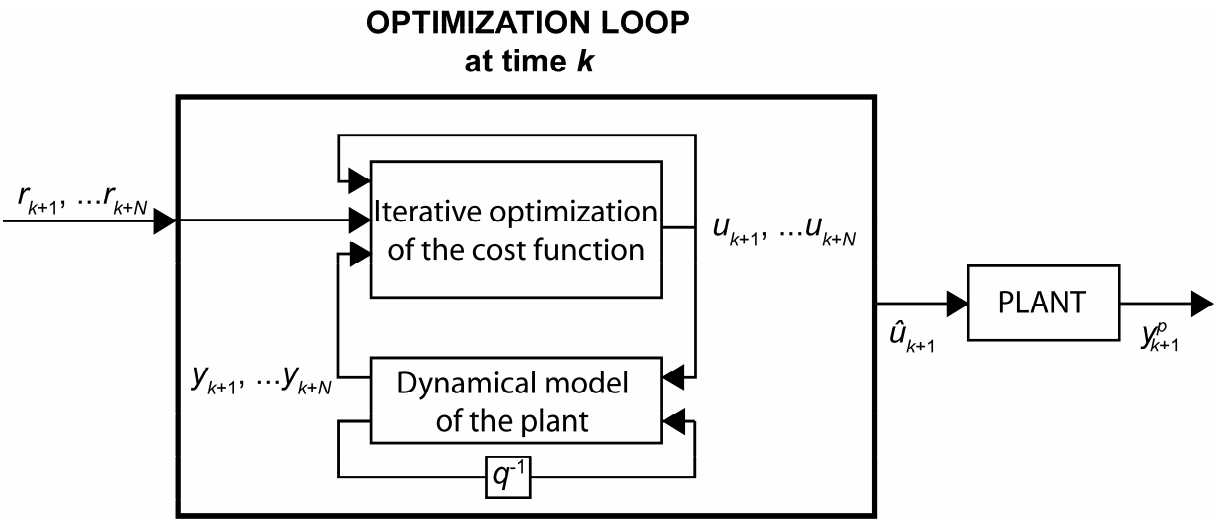


Figure 1

Computation of the optimal control value from a reference sequence: at time k, an iterative optimization of the cost function (1) with respect to $\{u_{k+1}, \ldots u_{k+N}\}$ is performed, providing an optimal sequence $\{\hat{u}_{k+1}, \ldots \hat{u}_{k+N}\}$, whose first element $\hat{u}_{k+1}$ is fed to the plant.

The predictive control of a plant has been widely investigated ([5], [8], [9], [21]). In most cases, it is intended to compute the future value of the control signal that provides an output as close as possible to a user-defined reference output. In that framework, the method consists in minimizing, with respect to the control signal, a quadratic cost function that penalizes the discrepancy between the given reference output and the estimated one, and subsequently applying that optimised control signal to the plant. In some situations, however, the user-defined quantity is a reference control signal (rather than a reference output); in such a case, the purpose of the control strategy is to compute a (slightly) different control signal, which is intended to improve the "quality" of the response: for instance, the torque demand of a car engine is defined by the user through the position of the gas pedal; however, in order to minimize the resulting torque misses (that occur after some sharp variations of the position of the throttle), one has to alter that torque demand to some extent. Therefore, a trade-off must

be found between the minimization of the cost function (associated to the quality criterion), and the discrepancy from the reference control signal.

In the present paper, we extend the latter approach to the case of nonlinear dynamical models. In that case, the value of the control signal at time k has an influence not only on the model output at time k, but also on the future values of that output. Therefore, one has to take into account a temporal horizon N larger than 1, by defining a cost function J, whose first term describes the variations of the estimated outputs, and whose second term describes the discrepancy between the control and reference values:

$$J\left(u_{k+1},\ldots,u_{k+N}\right)=\frac{1}{2}\sum_{n=1}^{N}\left(y_{k+n}-y_{k+n-1}\right)^2+\frac{1}{2}\sum_{n=1}^{N}\rho\left(n\right)\left(u_{k+n}-r_{k+n}\right)^2 \tag{1}$$

where $\{u_{k+i},\ i = 1 \text{ to } N\}$ is the control sequence with respect to which the cost function is optimized at time k, y_{k+n} is the model estimation of the output at the same time (depending on u_{k+n}), r_{k+n} is the reference control signal, and $\rho(n)$ controls the trade-off between the desired decrease of the variations of the output and the difference between the reference and control values.

For a model that is nonlinear with respect to the control variable, the cost function J will not be quadratic. An iterative optimization algorithm is then required to reach the minimum of J. We will use the Levenberg-Marquardt iterative second-order algorithm, which does not require any preliminary gradient descent; at iteration n, one has

$$\mathbf{u}_{(n)}=\mathbf{u}_{(n-1)}-\left[\frac{\partial^2 J}{\partial\mathbf{u}_{(n-1)}^{T}\partial\mathbf{u}_{(n-1)}}+\mu I\right]^{-1}\frac{\partial J}{\partial\mathbf{u}_{(n-1)}} \tag{2}$$

where $\mathbf{u}=\left[u_{k+1},u_{k+2},\ldots,u_{k+N}\right]^{T}$, I is the identity matrix, and where μ is the Levenberg-Marquardt parameter [15]. The use of a second-order algorithm guarantees faster convergence, towards a minimum of J, than simple gradient descent. However, it requires the computation of the Hessian matrix of J, with respect to the control values $u_{k+1},\ldots,u_{k+N}$.

The control values $u_{k+1},\ldots,u_{k+N}$ are initially chosen to be equal to the reference values $r_{k+1},\ldots,r_{k+N}$. Our goal is indeed to reach the local minimum of J that is the closest from the reference control sequence: with such an initialization, one increases the chances to reach it. At time k, only the first optimized control signals, namely $\hat{u}_{k+1}$, will be applied to the plant, before performing a new optimization on a horizon of size N at time k+1 (Figure 1).

To allow the implementation of that scheme, the crucial point is the computation of the Hessian matrix of J, which obviously depends on the nature of the model. In the following, we assume that the plant can be described by a set of M equations of the form:

$$z_i(k+1) = g_i\left(\{z_j(k-\tau_{ij,m}+1)\}, \{u_l(k-\tau_{il,m'}+1)\}\right) \quad i,j = 1 \text{ to } M,\ l = 1 \text{ to } M',\ m,m' > 0 \quad (3)$$

where g_i is an arbitrary function, $\tau_{ij,m}$ is the integer delay of the m-th delayed value of variable z_j involved in the computation of $z_i(k+1)$, and $\tau_{il,m'}$ is the integer delay of the m'-th delayed value of variable u_l involved in the computation of $z_i(k+1)$. It has been shown in [6] that such a model can be cast into a canonical form (or minimal state-space form):

$$\begin{cases} \mathbf{x}(k+1) = \Phi\left(\mathbf{x}(k), \mathbf{u}(k+1)\right) \\ \mathbf{y}(k+1) = \Psi\left(\mathbf{x}(k+1), \mathbf{u}(k+1)\right) \end{cases} \quad (4)$$

where $\mathbf{x}(k)$ is the vector of state variables at time k, $\mathbf{u}(k)$ is the vector of control inputs at time k, and $\mathbf{y}(k)$ is the vector of outputs; Φ and Ψ are derived from the functions z_i of relation (3) that do not involve any delay. It is often convenient to describe graphically such a model as a directed graph, whose nodes are functions and whose edges denote the dependencies between those functions (Figure 2). An illustrative example is provided in Appendix 1. For simplicity, we consider here single-input-single-output (SISO) models, so that $y(k)$ is actually scalar. An extension to the multiple-input-multiple-output (MIMO) case could easily be performed by adopting the same formalism as the one described in the following.

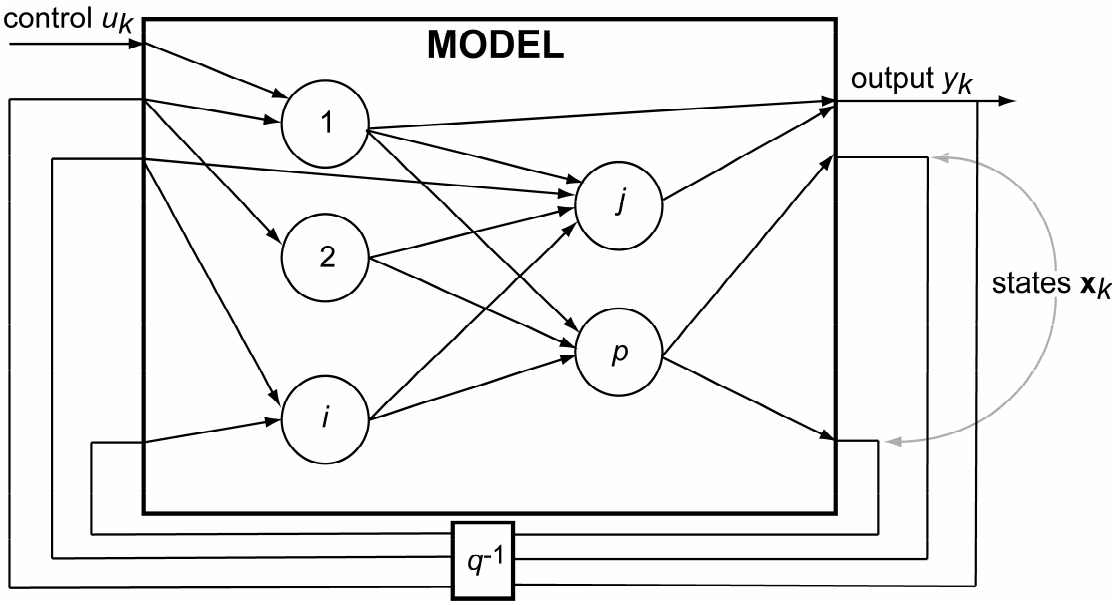


Figure 2

Graph representation of the canonical form of a dynamical nonlinear model

Any dynamical model of the form (3) can be cast in a canonical (minimum state space) form (4); node p of the graph denotes a nonlinear function f_p, whose variables can either be computed by the parent nodes of node p (e.g. node i), or be provided by the external or state inputs (e.g. node 1).

In order to define the various quantities required to compute the Hessian matrix, we first unfold the model in time, as shown in Figure 3.

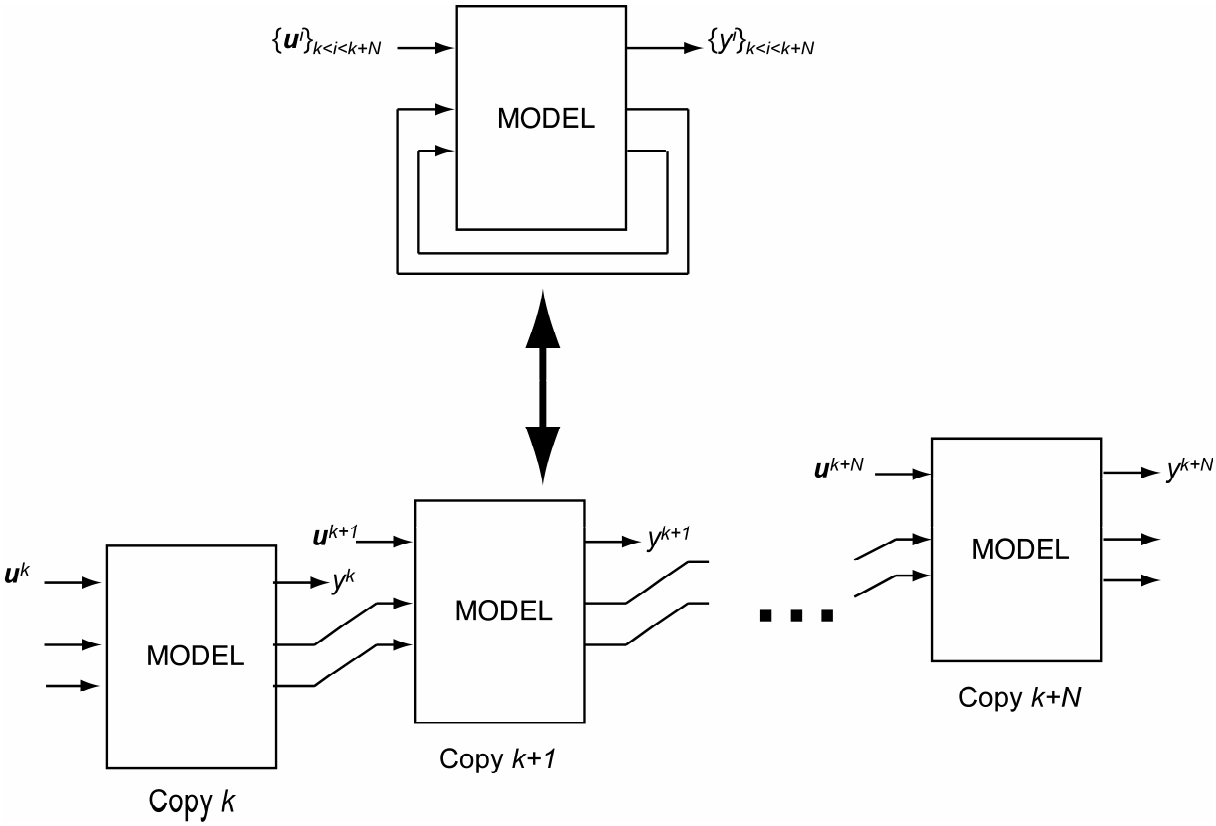


Figure 3

A dynamical model, unfolded in time. For simplicity, $u(k)$ is denoted as u^k. The unfolded model is a directed acyclic graph, since each box contains a directed acyclic graph.

If an integer index i is assigned to each node of the graph of the model, then any node in that unfolded representation can be indexed by two integers (i, n), where i is the index of the node, within copy number n. Denoting by a_i^n the output of node (i, n), one has:

$$a_i^n = f_i\left(\left\{a_j^m\right\}_{(j,m)\in P(i,n)}\right)$$

where $P(i, n)$ denotes the set of the parent nodes of node (i, n), i.e. the set of nodes that have an outgoing edge to node i of copy n.

2.2 Computation in the general case

The exact computation of the Hessian matrix of *J*, with respect to the control signal, requires the computation of four different quantities through forward and backward propagations. We describe below the mechanism whereby those quantities can be derived, in the general case.
We define:

$a_i^n = f_i\left(\left\{a_j^m\right\}_{(j,m)\in P(i,n)}\right)$: the output of node *i* in copy *n*

$$\sigma_i^n = \frac{\partial J}{\partial a_i^n}$$

$$g_{i,n}^{j,m} = \frac{\partial a_j^m}{\partial a_i^n}$$

$$b_{i,n}^{j,m} = \frac{\partial \sigma_j^m}{\partial a_i^n}$$

Computation of a_i^n : *forward path*

From the above definitions, the propagation relation is simply:

$$a_i^n = f_i\left(\left\{a_j^m\right\}_{(j,m)\in P(i,n)}\right) \tag{5}$$

From relation (5), all a_i^n can be computed**,** starting from the first layer of copy *k*, and propagating forward up to the output of copy *k*+*N*.

Computation of $g_{i,n}^{j,m}$: *forward path*

To compute the $g_{i,n}^{j,m}$, it is useful to notice that this quantity is equal to zero if there is no path from node (*j*, *m*) to node (*i*, *n*) in the directed graph of the model unfolded in time.
The propagation relation is:

$$g_{i,n}^{j,m} = \frac{\partial a_j^m}{\partial a_i^n} = \sum_{(r,p)\in P(j,m)} \frac{\partial a_j^m}{\partial a_r^p}\frac{\partial a_r^p}{\partial a_i^n} = \sum_{(r,p)\in P(j,m)} g_{r,p}^{j,m} g_{i,n}^{r,p} \tag{6}$$

The quantity $g_{r,p}^{j,m} = \dfrac{\partial f_j\left(\left\{a_k^l\right\}_{(k,l)\in P(j,m)}\right)}{\partial a_r^p}$ can be readily computed, since (*r*, *p*) is connected to (*j*, *m*) by an edge, and the function f_j is assumed to be known.

The full path can then be spanned by choosing a value of (i, n) and propagating the computation along the paths of the oriented graph. The obvious initialization is: $g_{i,n}^{i,n} = 1$ for any (i, n).

Computation of σ_i^n : *backward path*

For the computation of the σ_i^n, we must take into account the form of the cost function J.

The propagation relation can be written as follows:

$$\sigma_i^n = T_J + \sum_{(r,p)\in C(i,n)} \frac{\partial J}{\partial a_r^p}\frac{\partial a_r^p}{\partial a_i^n} = T_J + \sum_{(r,p)\in C(i,n)} \sigma_r^p g_{i,n}^{r,p}$$

where C(i, n) is the set of all children nodes of node (i, n), i.e. the set of nodes that have an incoming edge from node i of copy n.

The quantity T_J expresses the explicit dependence of the cost function on the outputs of the node functions. For instance, given the form of the cost function J as defined by relation (1), which depends explicitly on the outputs y of the model, and on its controls u, the propagation relation becomes:

$$\sigma_i^n = \sum_{(r,p)\in C(i,n)} \sigma_r^p g_{i,n}^{r,p} + \left[2y_{k+n} - y_{k+n-1} - y_{k+n+1}\right]\delta_{i,output} + \left[\rho(n)\left(u_{k+n} - r_{k+n}\right)\right]\delta_{i,control} \qquad (7)$$

where $\delta_{i,j}$ is the Kronecker symbol ($\delta_{i,j} = 1$ if $i = j$, $\delta_{i,j} = 0$ otherwise).

In order to span the full path and compute all the values of σ_i^n, the computation must start from the last node (*output*, N), and be performed backward using the above relation. The initialization is:

$$\sigma_{output}^N = \frac{\partial J}{\partial a_{output}^N} = y_{k+N} - y_{k+N-1}$$

Computation of $b_{i,n}^{j,m}$: *backward path*

The computation of $b_{i,n}^{j,m}$ is greatly simplified by noticing that:

$$b_{i,n}^{j,m} = \frac{\partial}{\partial a_i^n}\left(\frac{\partial J}{\partial a_j^m}\right) = \frac{\partial}{\partial a_j^m}\left(\frac{\partial J}{\partial a_i^n}\right) = b_{j,m}^{i,n}$$

As a consequence, one only needs to compute $b_{i,n}^{j,m}$ for node pairs $\{(j, m), (i, n)\}$ such that there exists a directed path from (i, n) to (j, m).

The general propagation relation can be written as follows:

$$b_{i,n}^{j,m} = \frac{\partial}{\partial a_i^n}\left(\sigma_j^m\right) = \frac{\partial}{\partial a_i^n}\left(T_J + \sum_{(r,p)\in C(j,m)} \sigma_r^p g_{j,m}^{r,p}\right)$$
$$= \frac{\partial T_J}{\partial a_i^n} + \sum_{(r,p)\in C(j,m)} b_{i,n}^{r,p} g_{j,m}^{r,p} + \sum_{(r,p)\in C(j,m)} \sigma_r^p \frac{\partial\left(g_{j,m}^{r,p}\right)}{\partial a_i^n}$$

For the last term of the above relation, since $(r, p) \in C(j, m)$, one can write:

$$\frac{\partial\left(g_{j,m}^{r,p}\right)}{\partial a_i^n} = \frac{\partial}{\partial a_i^n}\left(\frac{\partial f_r\left(\left\{a_k^l\right\}_{(k,l)\in P(r,p)}\right)}{\partial a_j^m}\right) = \sum_{(q,s)\in P(r,p)} \frac{\partial^2 f_r\left(\left\{a_k^l\right\}_{(k,l)\in P(r,p)}\right)}{\partial a_j^m \partial a_q^s} \frac{\partial a_q^s}{\partial a_i^n}$$
$$= \sum_{(q,s)\in P(r,p)} \frac{\partial^2 f_r\left(\left\{a_k^l\right\}_{(k,l)\in P(r,p)}\right)}{\partial a_j^m \partial a_q^s} g_{i,n}^{q,s}$$

Since the second derivative of f_r with respect to its variables is assumed to be available, the term $\frac{\partial\left(g_{j,m}^{r,p}\right)}{\partial a_i^n}$ can be computed. Moreover, if we take into account the form of T_J, the general backward computation relation becomes:

$$b_{i,n}^{j,m} = \left[2g_{i,n}^{output,m} - g_{i,n}^{output,m-1} - g_{i,n}^{output,m+1}\right]\delta_{j,output}$$
$$+ \sum_{(r,p)\in C(j,m)} b_{i,n}^{r,p} g_{j,m}^{r,p} + \sum_{(r,p)\in C(j,m)} \sum_{(q,s)\in P(r,p)} \sigma_r^p \frac{\partial^2 f_r\left(\left\{a_k^l\right\}_{(k,l)\in P(r,p)}\right)}{\partial a_j^m \partial a_q^s} g_{i,n}^{q,s} \quad (8)$$

The computation of the last term in the above relation can be costly. However, if $g_{j,m}^{r,p}$ depends on a_j^m only, as is the case for recurrent neural networks in their canonical form, the resulting propagation relation becomes much more manageable, as will be shown in the next section.

To span the full path, a node (i, n) should be chosen, and backward computation should be performed from node $(output, N)$ to node (i, n), by using the above relation. The choice of (i, n) may be performed as follows:

(*output*, *N*), …, (1, *N*), (*output*, *N*-1), …, (1, *N*-1),…, (output, 1), …, (1, 1)

The initialization is:

$$b_{i,n}^{output,N} = \frac{\partial}{\partial a_i^n}\left(\frac{\partial J}{\partial a_{output}^N}\right) = \frac{\partial}{\partial a_i^n}\left(a_{output}^N - a_{output}^{N-1}\right) = g_{i,n}^{output,N} - g_{i,n}^{output,N-1} \text{ for any } (i, n)$$

Summary:

After computing all the quantities defined previously, the general term of the Hessian matrix of the cost function J with respect to the control signals $[u_{k+1}, u_{k+2}, \ldots, u_{k+N}]$ can be written as:

$$\frac{\partial^2 J}{\partial u_{k+n}\partial u_{k+m}} = b_{control,n}^{control,m}$$

2.3 Computation for a recurrent neural network

In the present section, it is assumed that the forward part of the model shown on Figure 2 is a standard multilayer Perceptron, as shown on Figure 4, with one hidden layer of *tanh* neurons, and one output layer of identity function neurons. We then consider a recurrent neural network in its canonical form (see [16]), whose parameters have been estimated by running a standard learning algorithm (see for example [16]).

For recurrent neural networks, it is more convenient to define the quantity a_i^n as the potential of neuron *i* of copy *n*, rather than as its output, which will now be denoted as z_i^n :

$$a_i^n = \sum_{(r,p)\in P(i,n)} c_{i,r} z_r^p$$
$$z_i^n = f_i\left(a_i^n\right)$$

where $c_{i,j}$ is the network parameter associated to the connection from neuron *j* to neuron *i*.

Computation of z_i^n : *forward path*

The propagation relation (5) becomes:

$$z_i^n = f_i\left(\sum_{(r,p)\in P(i,n)} c_{i,r} z_r^p\right) \tag{9}$$

Computation of $g_{i,n}^{j,m}$: *forward path*

The propagation relation (6) becomes:

$$g_{i,n}^{j,m} = \frac{\partial a_j^m}{\partial a_i^n} = \sum_{(r,p)\in P(j,m)} \frac{\partial a_j^m}{\partial a_r^p} \frac{\partial a_r^p}{\partial a_i^n} = \sum_{(r,p)\in P(j,m)} c_{j,r} f_r^{'}\left(a_r^p\right) g_{i,n}^{r,p} \tag{10}$$

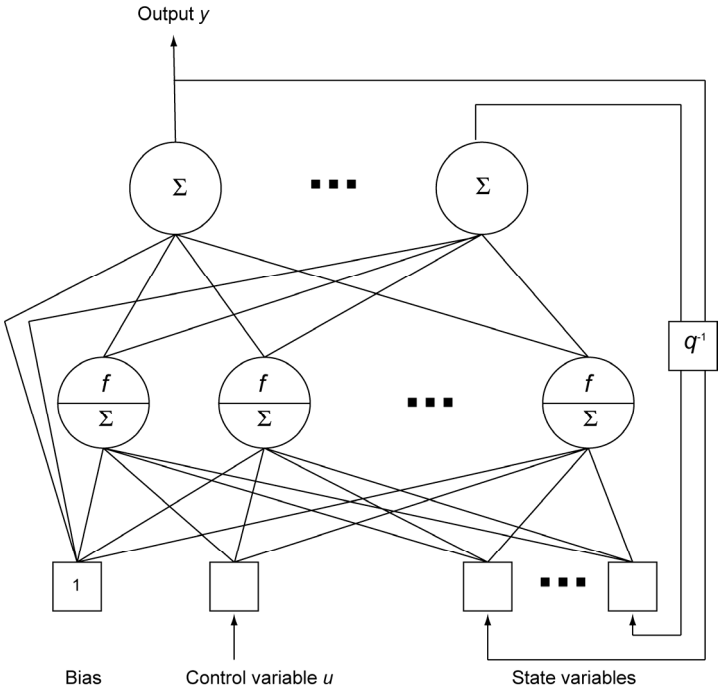


Figure 4

Canonical form of a recurrent network, whose feedforward part is a standard multilayer Perceptron; the output is one of the state variables.

Computation of σ_i^n : *backward path*

The propagation relation (7) can be rewritten as follows:

$$\sigma_i^n = f_i^{'}\left(a_i^n\right) \sum_{(r,p)\in C(i,n)} c_{r,i}\sigma_r^p + \left[2y_{k+n} - y_{k+n-1} - y_{k+n+1}\right]\delta_{i,output} + \left[\rho(n)\left(u_{k+n} - r_{k+n}\right)\right]\delta_{i,control} \tag{11}$$

Computation of $b_{i,n}^{j,m}$: *backward path*

The general propagation relation can be written as:

$$\begin{aligned} b_{i,n}^{j,m} = &\left[2g_{i,n}^{output,m} - g_{i,n}^{output,m-1} - g_{i,n}^{output,m+1}\right]\delta_{j,output} \\ &+ g_{i,n}^{j,m} f_j^{''}\left(a_j^m\right) \sum_{(r,p)\in C(j,m)} c_{r,j}\sigma_r^p + f_j^{'}\left(a_j^m\right) \sum_{(r,p)\in C(j,m)} c_{r,j}\, b_{i,n}^{r,p} \end{aligned} \tag{12}$$

where *f'* is the derivative of *f* with respect to the potential.

The computation of the quantity $\frac{\partial\left(g_{j,m}^{r,p}\right)}{\partial a_i^n}$ can be written as:

$$\frac{\partial\left(g_{j,m}^{r,p}\right)}{\partial a_i^n}=\frac{\partial}{\partial a_i^n}\left(\frac{\partial a_r^p}{\partial a_j^m}\right)=\frac{\partial}{\partial a_i^n}\left(c_{r,j}f_j'\left(a_j^m\right)\right)=c_{r,j}f_j''\left(a_j^m\right)g_{i,n}^{j,m}$$

where *f"* denotes the second derivative of *f* with respect to the potential.

The fact that $g_{j,m}^{r,p}$ only depends on a_j^m greatly simplifies relation (8).

3 ILLUSTRATION AND DISCUSSION

3.1 Description of the simulated plant, and of its neural model

We consider the control of a simulated plant with one output *y* and one control variable *u*, with Gaussian additive output noise *e* (whose standard deviation is σ_{noise}= 0.1), which is defined by the following discrete-time equations [24]:

$$\begin{cases} x_k = f\left(x_{k-1},x_{k-2},u_k\right)=\dfrac{24+x_{k-1}}{30}x_{k-1}-0.8\dfrac{u_k^2}{1+u_k^2}x_{k-2}+0.5u_k \\ y_k = x_k+e_k \end{cases} \qquad (13)$$

This plant is stable provided the control signal remains in the interval [-10,10].

The output sequence has been computed from the above equations, with two different control sequences, in order to generate a training sequence and a validation sequence, as shown in Figure 5.

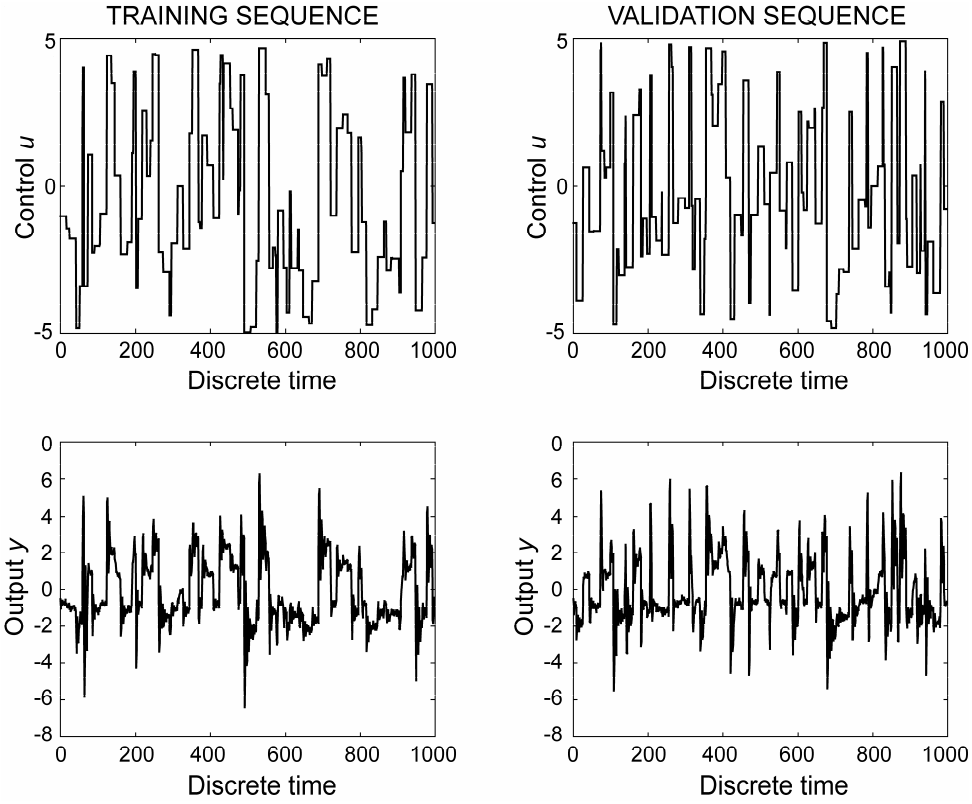


Figure 5

Left: training sequence; right: validation sequence

Using the training sequence, a recurrent neural network model was designed. The validation sequence was used for model selection (number of state variables, number of hidden neurons, minimum of the training cost function). Thus, a recurrent neural network (see Figure 4), with 2 states and 6 hidden neurons having a *tanh* activation function, has been trained with a semi-directed algorithm[1] [16] with 500 iterations of simple gradient followed by 1000 iterations of the BFGS algorithm (Broyden-Fletcher-Goldfarb-Shanno, see for instance [18]). Table 1 displays the root mean square model error (RMSE) on the two sequences. Since the values of the RMSE on these two sequences are of the same order of magnitude as the variance of the noise (i.e. $\sigma_{\text{noise}} = 10^{-1}$), the quality of the model can be considered satisfactory.

	TRAINING SEQUENCE	**VALIDATION SEQUENCE**
RMSE	$9.5\ 10^{-2}$	$1.1\ 10^{-1}$

[1] During training by a semi-directed algorithm, the state inputs of the model are the delayed state variables of the model; by contrast, in a directed (also termed "teacher forcing") algorithm, the state inputs are the measured values of the state variables (assumed to be measurable).

Table 1

Root mean square model error on the two sequences. To be compared with the value of the Gaussian additive noise $\sigma_{noise} = 0.1$

3.2 Exact computation of the Hessian matrix of the simulated plant

To illustrate the general computation procedure of the Hessian matrix, it will be adapted to the simulated plant described in the previous part. First, equation (13) can be described by the graph shown on Figure 6.

The functions shown in Figure 6 are defined as follows:

$$f_1\left(u_k\right) = u_k \equiv a_1^k$$

$$f_2\left(a_1^k\right) = 0.5 \times a_1^k \equiv a_2^k$$

$$f_3\left(a_5^{k-1}\right) = \frac{24 + a_5^{k-1}}{30} a_5^{k-1} \equiv a_3^k$$

$$f_4\left(a_1^k, a_6^{k-1}\right) = -0.8 \times \frac{\left(a_1^k\right)^2}{1 + \left(a_1^k\right)^2} a_6^{k-1} \equiv a_4^k$$

$$f_5\left(a_2^k, a_3^k, a_4^k\right) = a_2^k + a_3^k + a_4^k \equiv a_5^k$$

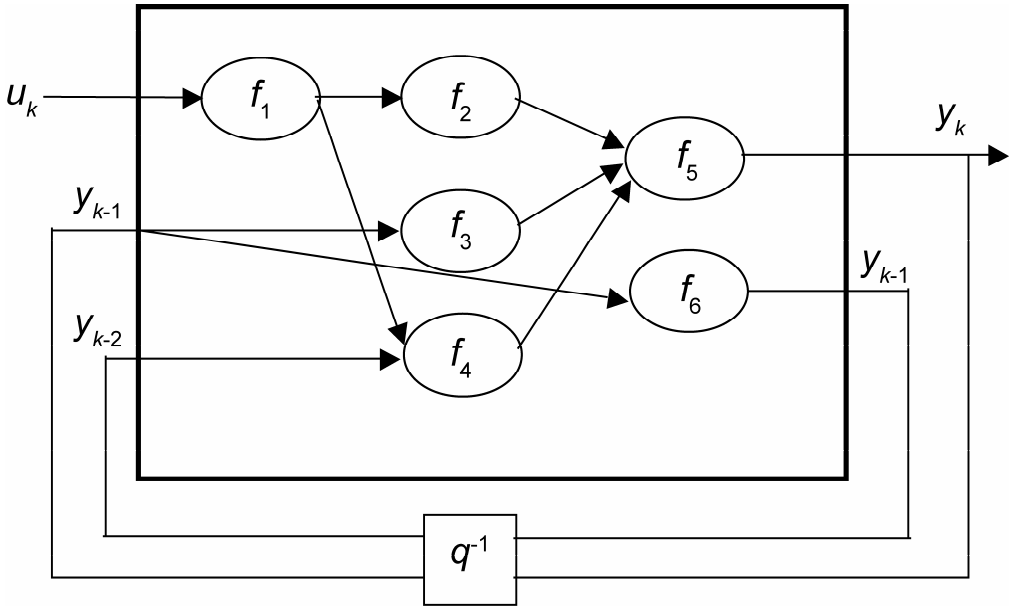


Figure 6

Graph representation of the model associated to equation (13)

Based on that representation, the general propagation relations, derived in Part 143, can be easily rewritten.

3.3 Neural optimal control of the simulated plant

The method described in the previous section has been tested on the validation sequence of the simulated plant.

In order to check whether this method was appropriate, we first compared the outputs of the plant to the estimations of the neural model, on a sequence of optimized control signals (computed on a horizon N=10, with a constant value ρ=2). Typically, each optimization (performed at each time k) requires about 10 iterations of the Levenberg-Marquardt algorithm for reaching the minimum of the cost function J. Figure 7 displays the optimized control signals, and the residuals between the actual outputs on this sequence and the corresponding neural model estimations.

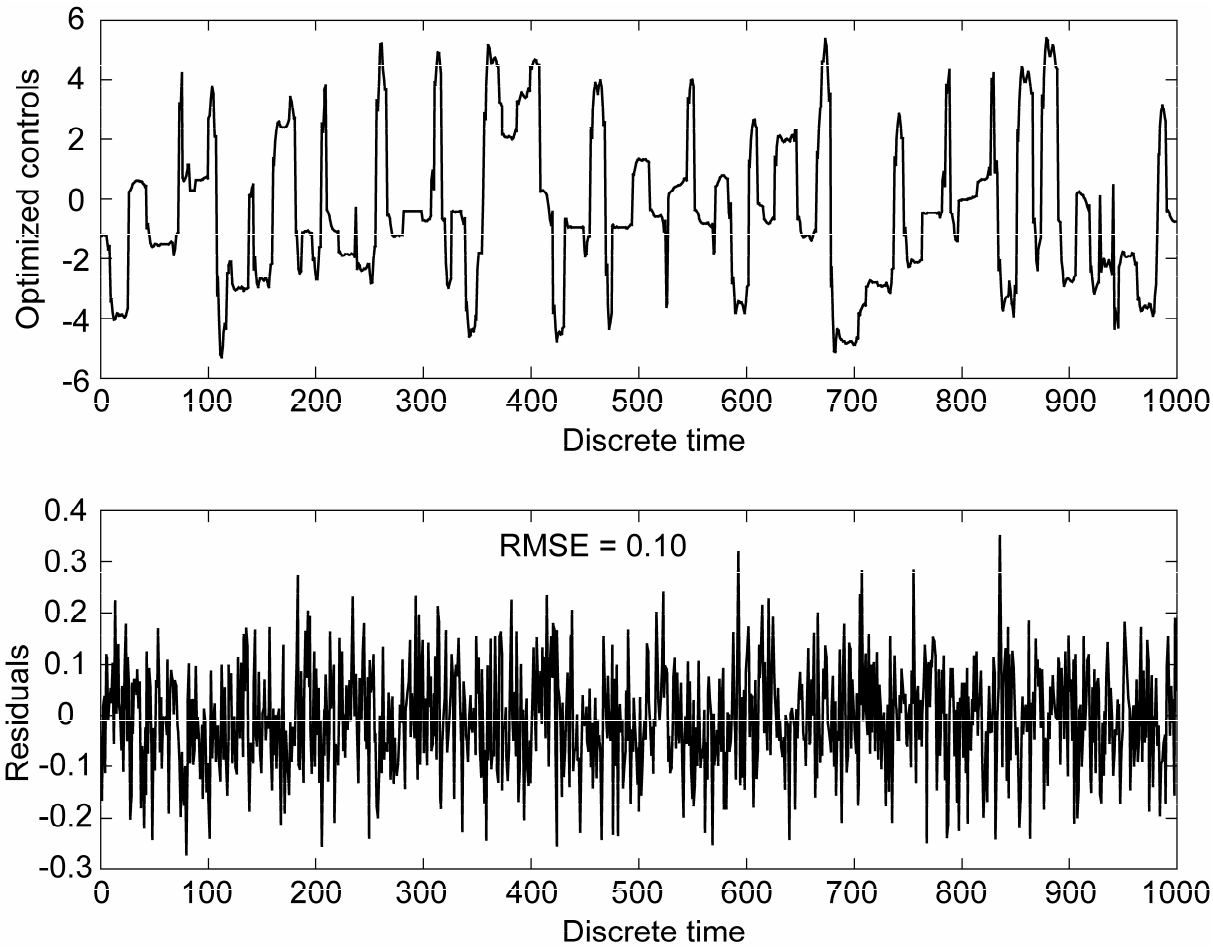


Figure 7

Optimized control signals on the validation sequence (top), and residuals between actual outputs and neural model estimations on this optimized sequence (bottom).

Since the control sequence is different from those present in the training set and in the validation set, the result can be considered as a test set for the neural model. The root mean

square error on this test sequence (RMSE = 0.10) remains on the same order of magnitude as that of the original validation sequence (RMSE = 0.11). This shows that the model can legitimately be used for optimal control.

It is interesting to investigate the influence of the horizon N. Several optimizations were performed with values of N ranging from 1 to 10 (and with a constant value ρ=2). In order to compare the results of these optimizations, in terms of minimization of the variations of the output, a quality index Q can be defined, computed with the actual outputs of the plant (rather than with their neural estimations, as during the optimization process) on the whole sequence. The quality index Q, computed for each value of N, is defined by:

$$Q = \frac{1}{2}\sum_{k=1}^{L-1}\left(y_{k+1}^{p} - y_{k}^{p}\right)^2 + \frac{1}{2}\rho\sum_{k=1}^{L}\left(\hat{u}_k - r_k\right)^2 ,$$

where L=1000 is the size of the sequence, and where y_k^p is the response of the plant to the optimized control value $\hat{u}_k$. Results are displayed on Figure 8 (where the value N=0 corresponds to no optimization).

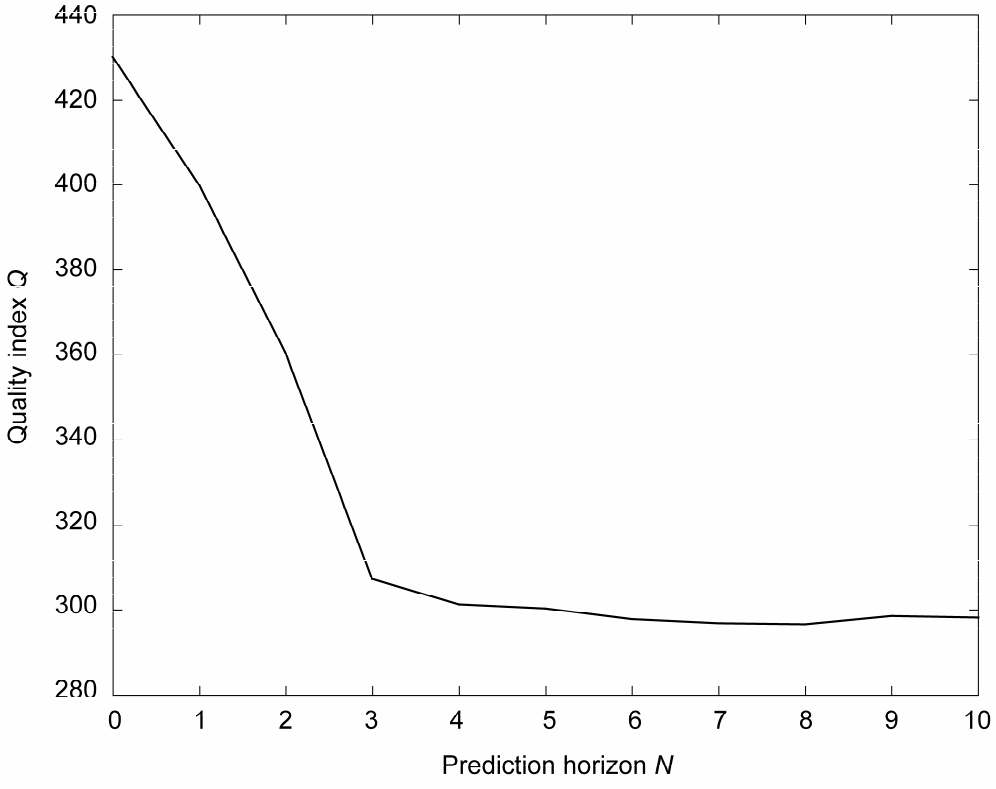


Figure 8

Quality index Q as a function of the prediction horizon N.

This makes it clear that by increasing the length of the prediction horizon N, the quality index decreases, hence the efficiency of the optimization increases, even though values of N larger than 3 do not provide any further substantial improvement.

More qualitatively, the horizon of prediction makes it possible for the control signal to compensate for future sharp variations of the output, and the larger the value of the horizon, the sooner this compensation will start. In other words, a large value of N provides a good anticipation of the variations of the output. This is illustrated on Figure 9, which shows both the optimized control signals, for values of N ranging from 1 to 3, just before a sharp increase of the reference control, and the corresponding outputs of the plant: without optimization, a sharp increase of the reference control signal occurring at time 310 generates fast variations of the output; for N=3, the optimized control signal anticipates this variation by increasing slowly from time 308. Moreover, the maximum of the output signal, which is approximately 5.4 without optimization, is reduced to 4.2 for N=1, and 3.7 for N=3.

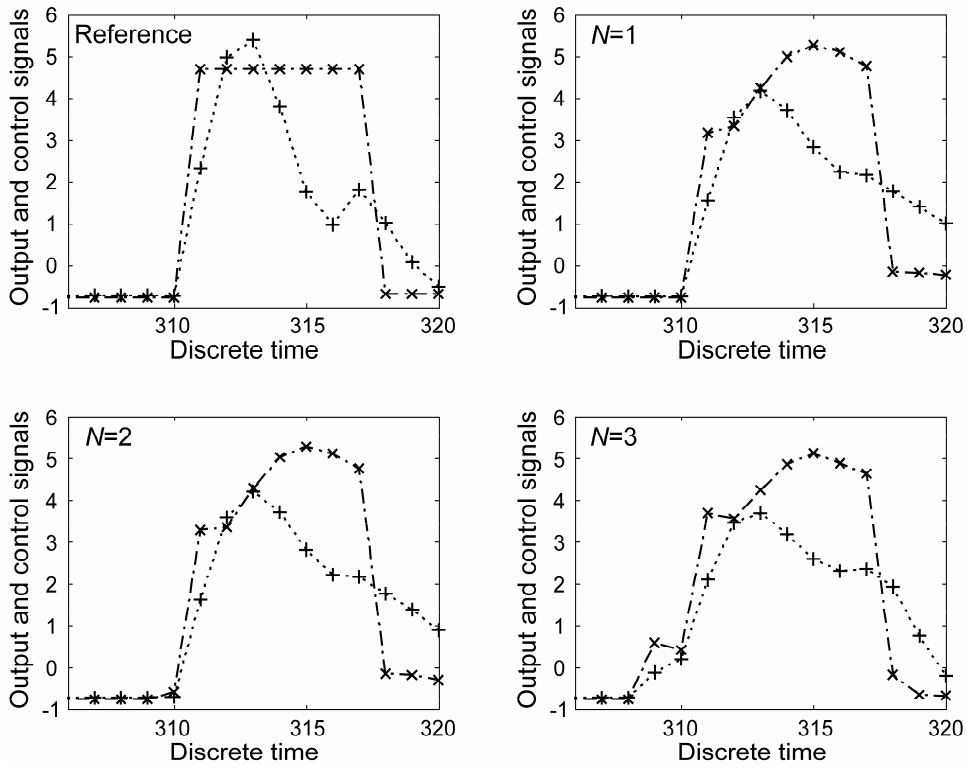


Figure 9

Optimized control signals (dash-dot lines) and output signals (dotted lines) for 3 different values of the predictive horizon N.

4 Conclusion

This paper describes a general framework for computing the exact Hessian matrix of a cost function, depending on the estimations of discrete-time dynamical models with arbitrary structure. The method involves the computation of four different quantities, through forward and backward computations in the model.

The formalism makes it possible to minimize, through second order optimization algorithms, a given cost function, depending either on the predictions of a black-box model such as a recurrent neural network, or on the predictions of a dynamical knowledge-based model described by a set of recurrent equations. An open-loop optimal control approach can then be achieved on a horizon larger than 1, making use of second-order optimization algorithms. We have illustrated its use on a particular plant, showing that the larger the predictive horizon, the more efficient the minimization of the variations of the output.

APPENDIX 1

Consider a model described by the following equations:

$$z_3(k+1) = g_3\left[z_4(k), u_1(k+1), u_2(k)\right]$$
$$z_4(k+1) = g_4\left[z_3(k+1), u_2(k+1)\right]$$
$$z_5(k+1) = g_5\left[z_3(k+1), z_4(k), u_1(k+1)\right]$$
$$y(k+1) = z_5(k+1)$$

It can be shown, by the methods described in [6], that the canonical form of the model is

$$x(k+1) = g_4\left[g_3\left[x(k), u_1(k+1), u_2(k)\right], u_2(k+1)\right]$$
$$y(k+1) = g_5\left[g_3\left[x(k), u_1(k+1), u_2(k)\right], x(k), u_1(k+1)\right]$$

The above equations are in the form of relation (4), where the state vector is a scalar (output of node 4) and the control vector is $\mathbf{u}(k+1)=[u_1(k+1)\ \ u_2(k)\ \ u_2(k+1)]$. The graph of the model is shown on Figure 10.

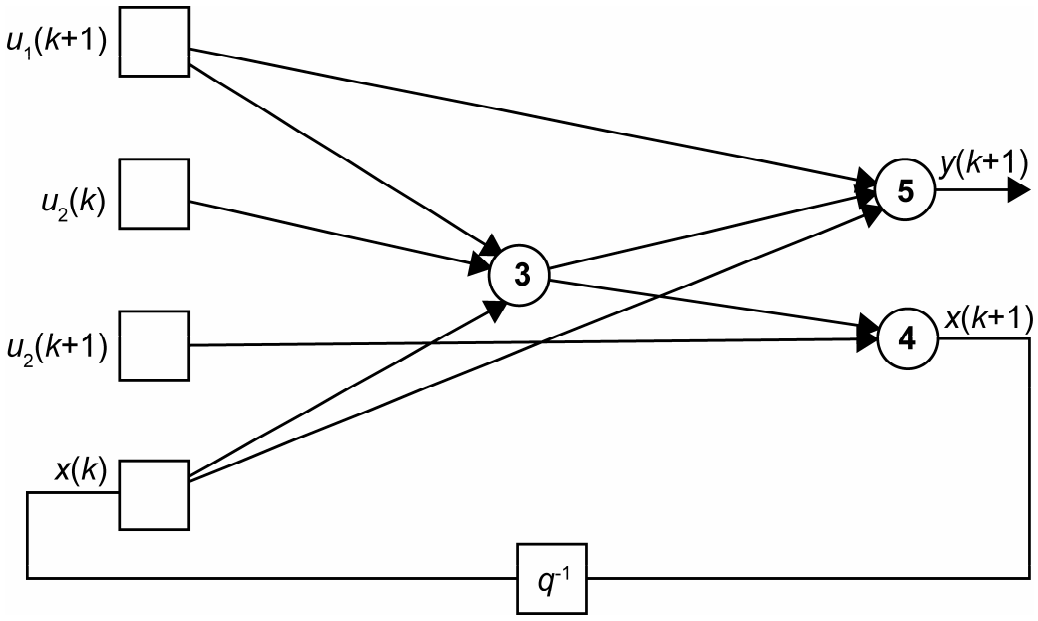


Figure 10

LITERATURE REFERENCES

[1] K.J. Aström and B. Wittenmark, Computer-controlled Systems: Theory and Design (Prentice Hall, 1997).

[2] S. Becker and Y. Le Cun, Improving the convergence of back-propagation learning with second order methods, in: D.S. Touretzky, G.E. Hinton and T.J. Sejnowski, eds., Proceedings of the Connectionist Models Summer School (Morgan Kaufman, Amsterdam, 1988) 29.

[3] C.M. Bishop, A fast procedure for re-training the multi-layer perceptron, International Journal of Neural Systems, Vol. 2, No. 3 (1991) 229-236.

[4] C.M. Bishop, Exact calculation of the Hessian matrix for the multi-layer perceptron, Neural Computation 4 (1992) 494-501.

[5] A.A. Bogdanov, E.A. Wan, M. Carlsson, Y. Zhang, R. Kieburtz and A. Baptista, Model predictive neural control of high-fidelity helicopter model, AIAA Guidance Navigation and Control Conference, Montreal, Canada, August 2001 (2001).

[6] G. Dreyfus and Y. Idan, The canonical form of nonlinear discrete-time models, Neural Computation 10 (1998) 133-164.

[7] G.F. Franklin, J.D. Powell and M.L. Workman, Digital Control of Dynamic Systems (Addison-Wesley, 1998).

[8] C.E. Garcia, D. Prett and D.M. Morari, Model predictive control: Theory and practice - A survey, Automatica, Vol. 25, No. 3 (1989) 335-348.

[9] D. Gu and H. Hu, Neural predictive control for a car-like mobile robot, International Journal of Robotics and Autonomous Systems, Vol. 39, No. 2-3 (2002).

[10] S. Haykin, Neural Network - A comprehensive Foundation (Prentice Hall, 1999).

[11] J.O. Henriques and A. Dourado, A recurrent neural approach for the nonlinear discrete time output regulation, IEEE International Conference on Industrial Electronics Control and Instrumentation, Nagoya, Japan, October 2000 (2000).

[12] R. Johansson, System Modeling & Identification (Prentice Hall, 1993).

[13] Y. Le Cun, J.S. Denker and S.A. Solla, Optimal brain damage, in: D.S. Touretzky, ed., Advances in Neural Information Processing Systems, Vol. 2 (Morgan Kaufmann, 1990) 598.

[14] D.J.C. MacKay, A practical Bayesian framework for backprop networks, Advances in Neural Information Processing Systems, Vol. 4 (Morgan Kaufmann, 1992).

[15] M. Minoux, Programmation mathématique: théorie et algorithmes (Dunod, Paris, 1983).

[16] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus and S. Marcos, Neural networks and non-linear adaptive filtering: unifying concepts and new algorithms, Neural Computation 5 (1993) 165-197.
[17] S.W. Piche, B. Sayyar-Rodsari, D. Johnson and M. Gerules, Nonlinear model predictive control using neural networks, IEEE Control Systems Magazine, Vol. 20, No. 3 (2000).
[18] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, Numerical Recipes in C (Cambridge University Press, Cambridge, 1992).
[19] S.J. Qin and T.A. Badgwell, An overview of industrial model predictive control technology, Chemical Process Control - AIChE Symposium Series (1997) 232-256.
[20] J.B. Rawlings, Tutorial overview of model predictive control, IEEE Control Systems Magazine, Vol. 20, No. 3 (2000) 38-52.
[21] N.L. Ricker, Model predictive control: state of the art, Fourth International Conference on Chemical Process Control (Elsevier, Amsterdam, 1991) 271-296.
[22] L.P. Ricotta, S. Ragazzini and G. Martinelli, Learning of word stress in a sub-optimal second order back-propagation neural network, Proceedings of the IEEE International Conference on Neural Networks, San Diego, Vol. 1 (1988) 355.
[23] R.M. Sanner and J.E. Slotine, Structurally dynamic wavelet networks for the adaptive control of uncertain robotic systems, Proceedings of the 34th Conference on Decision & Control, New Orland, LA (December 1995) 2460-2467.
[24] D. Urbani, Méthodes statistiques de sélection d'architectures neuronales - Application à la conception de modèles de processus dynamiques, Ph.D. Thesis, Université Pierre et Marie Curie, Paris, 1995.

Annexe 2 :

brevet relatif à l'estimation de la température après turbine

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

PCT

(43) Date de la publication internationale
13 avril 2006 (13.04.2006)

(10) Numéro de publication internationale
WO 2006/037926 A1

(51) Classification internationale des brevets :
F02B 77/08 (2006.01) *F02B 37/00* (2006.01)
G05B 13/02 (2006.01)

(21) Numéro de la demande internationale :
PCT/FR2005/050811

(22) Date de dépôt international :
5 octobre 2005 (05.10.2005)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
0410523 6 octobre 2004 (06.10.2004) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **RENAULT s.a.s** [FR/FR]; 13, 15 quai Alphonse Le Gallo, F-92100 BOULOGNE BILLANCOURT (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **BARILLON, Pascal** [FR/FR]; 41, rue Jean-Pierre Timbaud, F-92130 ISSY-LES-MOULINEAUX (FR). **GAUVIN, Fabrice** [FR/FR]; 14 résidence les Acacias, F-91540 MENNECY (FR). **LUCEA, Marc** [FR/FR]; 17, rue Civiale, F-75010 PARIS (FR). **POILANE, Emmanuel** [FR/FR]; 18, rue Dauvilliers, F-91290 ARPAJON (FR).

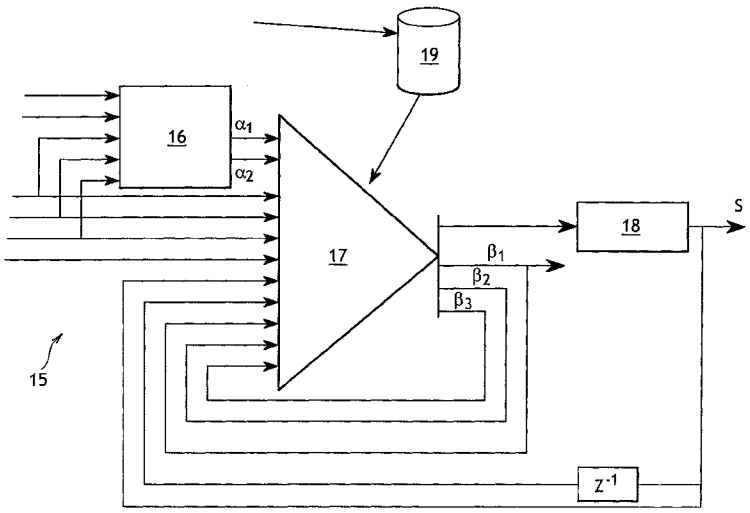
(74) Mandataire : **ROUGEMONT, Bernard**; RENAULT TECHNOCENTRE, Sce 00267 - TCR GRA 2 36, 01, avenue du golf, F-78288 GUYANCOURT (FR).

(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

[Suite sur la page suivante]

(54) **Title:** IMPROVED METHOD AND SYSTEM FOR ESTIMATING EXHAUST GAS TEMPERATURE AND INTERNAL COMBUSTION ENGINE EQUIPPED WITH SUCH A SYSTEM

(54) **Titre :** PROCEDE ET SYSTEME AMELIORES D'ESTIMATION D'UNE TEMPERATURE DES GAZ D'ECHAPPEMENT ET MOTEUR A COMBUSTION INTERNE EQUIPE D'UN TEL SYSTEME



(57) **Abstract:** The invention concerns a method for estimating the temperature of an internal combustion engine exhaust gases, characterized in that it uses an estimator (15) with neural network (17) provided with a feedback loop returning directly or indirectly in the network input one or more quantities of the network output. The invention also concerns a system for estimating exhaust gas temperature as well as an engine equipped with such a system.

[Suite sur la page suivante]

WO 2006/037926 A1

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— *avec rapport de recherche internationale*

— *avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : L'invention concerne un procédé d'estimation d'une température des gaz d'échappement d'un moteur à combustion interne, caractérisé en ce qu'il met en œuvre un estimateur (15) à réseau de neurones (17) disposant d'une boucle de rétroaction retournant directement ou indirectement en entrée du réseau une ou plusieurs des grandeurs de sortie du réseau. L'invention concerne également un système d'estimation d'une température des gaz d'échappement ainsi qu'un moteur équipé d'un tel système.

PROCÉDÉ ET SYSTEME AMÉLIORÉS D'ESTIMATION D'UNE TEMPÉRATURE DES GAZ D'ÉCHAPPEMENT ET MOTEUR A COMBUSTION INTERNE ÉQUIPÉ D'UN TEL SYSTEME

Le domaine de l'invention est celui du pilotage de moteur à combustion interne, et plus particulièrement des dispositifs d'échappement montés en aval du collecteur d'échappement du moteur.

L'invention concerne plus précisément un procédé et un système d'estimation d'une température des gaz d'échappement, ainsi qu'un moteur à combustion interne et qu'un véhicule équipés d'un tel système.

Pour répondre aux émissions de gaz polluants des véhicules automobiles, des systèmes de post-traitement des gaz sont généralement disposés dans la ligne d'échappement des moteurs.

Ces systèmes sont prévus pour réduire aussi bien les émissions du monoxyde de carbone, des hydrocarbures imbrûlés que celles des particules et des oxydes d'azote.

Certains de ces systèmes fonctionnent de manière discontinue ou alternative en ce sens qu'ils alternent des phases de stockage des polluants et des phases de régénération des pièges (c'est-à-dire de conversion des polluants stockés en substances non polluantes).

Afin d'optimiser le traitement de l'ensemble des polluants, il est nécessaire de contrôler au mieux ces phases de stockage et de régénération. Il est notamment nécessaire d'estimer au cours du temps les masses piégées (c'est-à-dire les particules dans le cas du filtre à particule, les oxydes d'azote dans le cas du piège à oxydes d'azote).

De même, il est nécessaire de connaître l'évolution au cours du temps des masses converties lors des phases de régénération.

Or l'évolution de ces masses lors des phases de stockage et/ou de régénération dépend directement de la température du support de ces pièges et des gaz qui les traversent. On cherche donc à connaître, sinon à contrôler, la température des gaz qui pénètrent dans ces pièges.

Par ailleurs, l'augmentation de la complexité des moteurs et de leurs modes de fonctionnement nécessite des moyens de gestion électronique de plus en plus sophistiqués et par là même, des moyens de mesure ou d'estimation de plus en plus nombreux. Mais il s'avère que de

nombreuses grandeurs physiques ne sont pas directement mesurables, ou que les capteurs nécessaires sont trop chers voire inadaptés.

Il apparaît donc nécessaire, pour la commande de la régénération des pièges et pour la commande du moteur proprement dit, de connaître la température des gaz d'échappement du moteur. L'obtention d'une estimation précise de la température des gaz d'échappement en entrée d'un système de post-traitement disposé en aval de la turbine d'un moteur suralimenté peut ainsi s'avérer particulièrement utile.

Pour connaître la température des gaz en amont d'un dispositif de traitement physique et/ou chimique des gaz d'échappement, on a recours aujourd'hui soit à un capteur de température, soit à un modèle d'estimation de cette température mis en œuvre par un calculateur électronique embarqué.

Plusieurs inconvénients sont toutefois liés à l'utilisation d'un capteur placé dans la ligne d'échappement.

La précision d'un capteur est en effet inversement proportionnelle au champ d'utilisation. C'est à dire que plus on souhaite mesurer la température sur une large plage d'utilisation, plus la précision de mesure est médiocre. Cette précision peut par ailleurs dériver avec le vieillissement thermique ou l'encrassement du capteur.

De plus, le coût d'utilisation d'un capteur peut s'avérer important. Il faut effectivement associer au coût intrinsèque au capteur celui de la connectique, du port d'entrée dans le calculateur et du pilote logiciel.

Par ailleurs, il s'avère également nécessaire de disposer de moyens adaptés pour diagnostiquer l'état de fonctionnement du capteur.

L'utilisation de modèles présente également des inconvénients. Si ces modèles sont généralement fidèles en régime de gaz permanent, ils s'avèrent plutôt médiocres en régime transitoire. De plus, de nombreux paramètres (paramètres physiques, variables d'état) nécessaires à ces modèles sont difficilement identifiables ou mesurables sur moteur.

En outre, ces modèles sont la plupart du temps trop « gourmands » en terme de charge de calcul ou de ressource mémoire pour pouvoir être implémentés dans un calculateur de gestion électronique moteur.

Une autre solution a été proposée qui consiste à estimer la température des gaz d'échappement en mettant en œuvre un réseau de neurones statique. On pourra ainsi se référer à la demande de brevet français de la Demanderesse déposée le 19 décembre 2003 sous le n° FR 03 15112.

La solution décrite dans ce document ne permet toutefois qu'une prise en compte partielle de l'historique du système. L'estimation ainsi réalisée de la température, à la fois en régime permanent et en régime dynamique, est alors peu précise.

La présente invention a pour objectif d'améliorer la connaissance d'une température des gaz d'échappement, et en particulier de la température en amont d'un dispositif de traitement physique et/ou chimique des gaz d'échappement.

A cet effet, l'invention propose un procédé d'estimation d'une température des gaz d'échappement d'un moteur, caractérisé en ce qu'il met en œuvre un estimateur à réseau de neurones disposant d'une boucle de rétroaction retournant directement ou indirectement en entrée du réseau une ou plusieurs des grandeurs disponibles en sortie du réseau.

Certains aspects préférés, mais non limitatifs du procédé selon l'invention sont les suivants :

- on fournit en entrée de l'estimateur une information relative à la température des gaz en amont de la turbine disposée sur la ligne d'échappement du moteur, l'estimateur fournissant en sortie l'estimation de la température des gaz d'échappement en aval de ladite turbine ;
- le procédé peut mettre en œuvre un post-traitement de l'estimation de température réalisée par le réseau de neurones ;
- le procédé peut réaliser une rétroaction de la température estimée, disponible en sortie du module de post traitement ;

- le procédé peut mettre en œuvre un prétraitement d'une ou plusieurs des variables en entrée de l'estimateur en réalisant des calculs basés sur des relations physiques connues ;
- le procédé peut mettre en oeuvre un retraitement de certaines des grandeurs en sortie du réseau avant que celles-ci ne soient retournées, selon un rebouclage ainsi dit indirect, vers l'entrée du réseau ;
- le procédé peut comporter une étape préalable d'apprentissage de l'estimateur à l'aide d'une base de données représentatives de zones de fonctionnement intéressantes.

Selon un autre aspect, l'invention concerne un système d'estimation d'une température des gaz d'échappement d'un moteur, caractérisé en ce qu'il comporte un estimateur à réseau de neurones disposant d'une boucle de rétroaction retournant directement ou indirectement en entrée du réseau une ou plusieurs des variables de sortie du réseau.

Selon encore d'autres aspects, l'invention concerne un moteur à combustion interne ainsi qu'un véhicule automobile équipés d'un système d'estimation d'une température des gaz d'échappement selon l'invention.

D'autres aspect, buts et avantages de l'invention apparaîtront à la lecture de la description détaillée suivante de formes de réalisation préférées de celle-ci, donnée à titre d'exemple non limitatif et faite en référence aux dessins annexés sur lesquels :

- la figure 1 est une vue schématique d'un moteur à combustion interne selon un aspect de l'invention ;
- la figure 2 est une vue schématique d'un système d'estimation d'une température de gaz d'échappement selon un aspect de l'invention ;
- la figure 3 est un diagramme montrant les étapes du procédé d'estimation d'une température de gaz d'échappement selon un aspect de l'invention.

Sur la figure 1, seuls les éléments nécessaires à la compréhension de l'invention ont été représentés. Un moteur à combustion interne 1 est destiné à équiper un véhicule tel qu'une automobile. Le moteur 1 est

par exemple un moteur Diesel suralimenté par turbocompresseur à quatre cylindres en ligne et injection directe de carburant. Le moteur 1 comprend un circuit d'admission 2 assurant son alimentation en air, un calculateur 3 de contrôle moteur, un circuit de carburant sous pression 4, et une ligne d'échappement 5 des gaz. L'injection du carburant dans les cylindres est assurée par des injecteurs, non représentés, débouchant dans les chambres de combustion et pilotés par le calculateur 3 à partir du circuit 4.

En sortie du moteur 1, les gaz d'échappement évacués dans la ligne d'échappement 5 traversent un ou plusieurs dispositifs de post-traitement 6 (par exemple filtre à particules, filtre à oxydes d'azote). Un turbocompresseur 7 comprend un compresseur disposé sur le circuit d'admission 2 et une turbine disposée sur la ligne d'échappement 5. Entre le compresseur et le moteur 1, le circuit d'admission 2 comprend un échangeur thermique 8 permettant de refroidir l'air comprimé à la sortie du compresseur et d'accroître ainsi sa masse volumique, un volet d'admission d'air 9 commandé par le calculateur 3, et un capteur de pression 10 relié au calculateur 3.

En sortie du moteur 1, en amont de la turbine, la ligne d'échappement 5 comporte en outre des moyens 30 adaptés pour fournir une information relative à la température des gaz d'échappement en amont de la turbine. Lesdits moyens 30 sont par exemple constitués par une sonde de mesure de température ou encore par un estimateur prévu pour fournir une estimation de ladite température des gaz en amont de la turbine.

Le moteur 1 comprend également un circuit de recyclage de gaz d'échappement 11 équipé d'une vanne 12 dont l'ouverture est pilotée par le calculateur 3 ; on peut ainsi réintroduire des gaz d'échappement dans le circuit d'admission 2. Un debitmètre d'air 13 est monté dans le circuit d'admission 2 en amont du compresseur pour fournir au calculateur 3 des informations relatives au débit de l'air d'admission

alimentant le moteur. Des capteurs 14, par exemple de pression ou de température, peuvent également être prévus.

Le calculateur 3 comprend, de façon classique, un microprocesseur ou unité centrale, des zones mémoires, des convertisseurs analogiques/numériques et différentes interfaces d'entrée et de sortie. Le microprocesseur du calculateur comprend des circuits électroniques et les logiciels appropriés pour traiter les signaux en provenances des différents capteurs, en déduire les états du moteur et générer les signaux de commande appropriés à destination des différents actionneurs pilotés tels que les injecteurs.

Le calculateur 3 commande ainsi la pression du carburant dans le circuit 4 et l'ouverture des injecteurs, et ce, à partir des informations délivrées par les différents capteurs et en particulier de la masse d'air admise, du régime moteur, ainsi que d'étalonnages mémorisés permettant d'atteindre les niveaux de consommation et de performance souhaités.

Le calculateur 3 comprend en outre un estimateur 15 à réseaux de neurones prévu pour réaliser une estimation de la température des gaz d'échappement en amont du ou des dispositifs de post-traitement.

Pour des raisons de simplicité et de partage d'un certain nombre de ressources, notamment de calcul et de mémoire, il est particulièrement avantageux de disposer l'estimateur 15 au sein du calculateur 3.

Comme cela est illustré sur la figure 2, l'estimateur 15 comprend un module de pré-traitement 16, un réseau de neurones 17 et un module de post-traitement 18. L'estimateur 15 comporte en outre une boucle de rétroaction prévue pour retourner en entrée du réseau de neurones une ou plusieurs des variables en sortie dudit réseau.

L'estimateur 15 comprend par ailleurs une entrée relative à l'information relative à la température des gaz d'échappement en amont de la turbine, telle que fournie par exemple par les moyens 30.

L'estimateur peut également comporter une ou plusieurs autres entrées relatives à des grandeurs physiques représentatives d'état du moteur, telles que par exemple :

- le débit des gaz,
- la pression des gaz en amont de la turbine,
- la contre-pression des gaz en aval de la turbine,
- la position des ailettes du turbocompresseur,
- la vitesse du véhicule,
- la température de l'air ambiant,
- la position de la vanne de recyclage des gaz d'échappement,
- la pression de suralimentation mesurée par le capteur de pression disposé en aval du compresseur, de l'échangeur thermique et du volet d'admission d'air,
- la température de l'air dans le circuit d'admission.

L'estimateur peut également recevoir en entrée le régime du moteur, le débit de carburant, le débit d'air.

Le réseau de neurones 17 est constitué d'un certain nombre de neurones définis par leurs paramètres (poids, biais), et par leurs fonctions d'activation. La sortie s d'un neurone est liée aux entrées (e_1, e_2, ..., e_n) du neurone par $s = F (e_1*w_1+ e_2*w_2+...+e_n*w_n + b)$, où F est la fonction d'activation du neurone, w_1, w_2, ...w_n les poids et b le biais.

Le nombre de neurones du réseau, les valeurs des poids et des biais des différents neurones sont des paramètres susceptibles de calibration qui peuvent en particulier être déterminés lors d'une phase d'apprentissage qui sera décrite plus en avant par la suite.

Le module de pré-traitement 16 permet de traiter une ou plusieurs des variables en entrée de l'estimateur en réalisant des calculs basés sur des relations physiques connues. Le module de pré-traitement 16 permet en particulier de réduire le nombre d'entrées du réseau de neurones 17 en calculant une ou plusieurs variables α_1, α_2 chacune représentative d'une ou plusieurs grandeur(s) physique(s), mesurable(s) ou non, absente(s) en

entrée du calculateur 3, à partir de plusieurs variables d'entrée. Le module 16 permet aussi de filtrer certaines entrées susceptibles de prendre des valeurs aberrantes.

Le réseau de neurones 17 dispose d'une voie de sortie relative à l'estimation de température désirée, ainsi qu'éventuellement une ou plusieurs autres voies de sorties relatives à des grandeurs d'état non mesurables destinées à être rebouclées, directement ou indirectement, vers l'entrée du réseau 17.

Le module de post-traitement 18 permet de traiter l'estimation de température disponible en sortie du réseau de neurones 17 pour émettre un signal S de température estimée en sortie de l'estimateur, à disposition du calculateur 3. Ce traitement peut être par exemple un filtrage permettant de rendre l'estimateur robuste en limitant le retour vers l'entrée d'une donnée aberrante.

Comme mentionné précédemment, une ou plusieurs des variables en sortie dudit réseau suivent une boucle de rétroaction et sont retournées en entrée du réseau de neurones.

Certaines des variables en sortie du réseau peuvent être traitées (par exemple filtrées, retardées ou associées à d'autres entrées dans le module de pré-traitement) avant d'être retournées en entrée du réseau.

En référence à la figure 2, on a ainsi illustré le cas où des grandeurs d'état β_1, β_2, β_3 (non mesurables et connues uniquement lors de l'apprentissage) disponibles en sortie du réseau 17 sont rebouclées vers l'entrée dudit réseau 17.

On a aussi illustré le cas où la sortie S de l'estimateur (c'est-à-dire l'estimation de la température en aval de la turbine disposée sur la ligne d'échappement des gaz) disponible en sortie du module de post-traitement 18 (et dont la valeur est ici mesurable lors de l'apprentissage) est d'une part rebouclée directement vers l'entrée du réseau 17, et d'autre part rebouclée après avoir été retardée (cf. retardateur représenté par le symbole z^{-1}) vers l'entrée du réseau 1.

En mettant en oeuvre une telle rétroaction d'une ou plusieurs des variables en sortie du réseau de neurones, ledit réseau utilise en entrée de sa fonction d'estimation une ou plusieurs des valeurs prédites aux pas de calculs précédents par cette même fonction d'estimation, et cela que ces valeurs aient été ou non mesurées lors de la phase d'apprentissage.

Un tel rebouclage de rétroaction présente l'avantage de permettre la prise en compte de phénomènes fortement dynamiques.

L'utilisation de fonctions d'activation non linéaires dans le réseau de neurones permet en outre la prise en compte de phénomènes non linéaires.

Le réseau de neurones 17 peut ainsi être qualifié de récurrent ou dynamique, et diffère à ce propos de la solution à réseau statique exposée dans la demande de brevet français n° FR 03 15112 dont il a été fait état précédemment.

L'estimateur 15 permet finalement d'obtenir une estimation de la température en aval de la turbine et en amont d'un dispositif de post-traitement. L'estimateur peut ainsi être vu comme mettant en œuvre une fonction de transfert prenant en entrée une information relative à la température en amont de la turbine, ainsi qu'éventuellement une ou plusieurs autres informations relatives notamment à des grandeurs physiques, pour fournir en sortie l'estimation de température désirée.

La description ci-après concerne la conception et l'apprentissage de l'estimateur 15.

Une fois l'estimateur élaboré et implémenté par exemple dans le logiciel de gestion électronique du moteur, il s'agit de choisir le réseau de neurones, notamment en déterminant le nombre de neurones et en calibrant les paramètres (poids, biais).

L'apprentissage du réseau de neurones peut être réalisé sur ordinateur à partir de données collectées sur véhicule.

Le réseau de neurones peut faire l'objet d'un apprentissage par une méthode d'algorithme d'apprentissage, notamment à l'aide d'une base de données 19.

La base de données 19 peut être alimentée à partir de résultats d'essais réels sur piste d'un véhicule avec des essais à différents rapports de boîte de vitesses, avec différentes accélérations et décélérations, le tout étant choisi pour être représentatif des conditions normales de fonctionnement du véhicule. En outre, on pourra prévoir de scinder la base de données alimentée à partir des essais sur piste en une base d'apprentissage et en une base de tests, de façon à réduire la taille de la base de données 19 qui forme la base d'apprentissage. Les paramètres (ou calibrations) déterminés lors de la phase d'apprentissage peuvent être stockés dans la mémoire du calculateur 3. Alternativement, la base de données 19 est une base distincte du véhicule, utilisée lors d'opérations d'initialisation du véhicule ou encore lors d'opérations de maintenance.

Sur la figure 3, sont illustrées les étapes de conception et d'apprentissage de l'estimateur à réseau de neurones.

A l'étape 20, on effectue des essais permettant de générer les données pour renseigner la base de données 19 (cf. flèche à destination de la base 19 sur la figure 2), les données étant représentatives de zones de fonctionnement intéressantes. Il s'agit effectivement de parcourir de manière représentative l'espace de variation (ou tout du moins une partie) des entrées et sorties de l'estimateur.

Il peut être utile d'effectuer un nettoyage des données, par traitement du signal, par exemple pour supprimer les points aberrants, pour supprimer les points redondants, pour re-synchroniser ou pour filtrer les données.

À l'étape 21, on découpe les données en deux parties pour former une base de tests et une base d'apprentissage. À l'étape 22, on détermine les pré-traitements devant être exécutés par le module de pré-traitement 16 et le réseau de neurones 17 effectue les apprentissages à partir de la base de données d'apprentissage. À l'étape 23, on teste la performance avec un ou plusieurs critères de validation, sur la base de test et sur la base d'apprentissage. À l'étape 24, on effectue le choix du réseau de neurones

17 et à l'étape 25, on effectue les caractérisations des performances et de tests sur le véhicule.

La flèche issue de la base 19 sur la figure 2 illustre de quelle manière les données de la base 19 sont utilisées pour l'apprentissage et la validation de l'estimateur 15.

Entre les étapes 23 et 24, il peut être prévu un rebouclage 26 permettant de remonter à l'amont de l'étape 22 pour effectuer un certain nombre d'itérations avec des modifications éventuelles sur le nombre de neurones, les entrées, les sorties, le découpage en nombre de réseaux, l'architecture du ou des réseaux de neurones 17, le type d'apprentissage, les critères d'optimisation, etc.

Ce processus itératif (rebouclage 26) permet d'élaborer l'estimateur souhaité (précision, robustesse), à moindre coût (nombre d'entrées, nombre de calculs, nombre et difficulté des essais, etc.).

Entre les étapes 24 et 25, il peut être prévu un rebouclage 27 lorsqu'il s'avère que les données générées à l'étape 20 sont insuffisantes. On repasse alors à l'étape 20 pour générer de nouvelles données appelées à remplacer les données précédemment acquises ou à les compléter afin d'établir un nombre de points de mesure suffisant.

À titre d'exemple, on peut prévoir 5000 points de mesure dans la base d'apprentissage.

Lors de l'étape 23 de test, on peut prévoir des critères de choix pour sélectionner le jeu de paramètres permettant l'estimation la plus précise (par exemple en supprimant des points dont l'erreur est supérieure à 50°C, en recherchant une moyenne d'erreur proche de 5°C, ou encore en recherchant une erreur moyenne glissante faible).

Finalement, la mise en œuvre de l'invention pour l'estimation de la température des gaz d'échappement est peu consommatrice des ressources du calculateur (charge de calcul, mémoire nécessaire), utilise un nombre restreint de paramètres, et peut ainsi être facilement implémentée.

De plus, dès lors que la base de données utile à l'apprentissage est suffisamment représentative de l'ensemble des conditions d'utilisation ultérieures, la température des gaz en sortie de la turbine ou en amont du système de post-traitement peut être estimée avec précision (de l'ordre de quelques degrés), et cela que ce soit en régime permanent ou transitoire.

REVENDICATIONS

1. Procédé d'estimation d'une température des gaz d'échappement d'un moteur (1), caractérisé en ce qu'il met en œuvre un estimateur (15) à réseau (17) de neurones disposant d'une boucle de rétroaction retournant directement ou indirectement en entrée du réseau une ou plusieurs des grandeurs disponibles en sortie du réseau.

2. Procédé selon la revendication précédente, caractérisé en ce qu'on fournit en entrée de l'estimateur (15) une information relative à la température des gaz en amont de la turbine (7) disposée sur la ligne d'échappement (5) du moteur, l'estimateur fournissant en sortie l'estimation de la température des gaz d'échappement en aval de ladite turbine.

3. Procédé selon l'une des revendications précédentes, caractérisé en ce qu'il met en œuvre un post-traitement de l'estimation de température réalisée par le réseau de neurones.

4. Procédé selon la revendication précédente, caractérisé en ce qu'on réalise une rétroaction de la température (S) estimée, disponible en sortie du module de post traitement.

5. Procédé selon l'une des revendications précédentes, caractérisé en ce qu'il met en œuvre un prétraitement d'une ou plusieurs des variables en entrée de l'estimateur en réalisant des calculs basés sur des relations physiques connues.

6. Procédé selon l'une des revendications précédentes, caractérisé en ce qu'il met en oeuvre un retraitement de certaines des grandeurs en sortie du réseau avant que celles-ci ne soient retournées, selon un rebouclage ainsi dit indirect, vers l'entrée du réseau.

7. Procédé selon l'une des revendications précédentes, caractérisé en ce qu'il comporte une étape préalable d'apprentissage de l'estimateur à l'aide d'une base de données (19) représentatives de zones de fonctionnement intéressantes.

8. Système d'estimation d'une température des gaz d'échappement d'un moteur (1), caractérisé en ce qu'il comporte un estimateur (15) à réseau (17) de neurones disposant d'une boucle de rétroaction retournant directement ou indirectement en entrée du réseau une ou plusieurs des variables de sortie du réseau.

9. Système selon la revendication précédente, caractérisé en ce que l'estimateur dispose d'une entrée prévue pour recevoir une information relative à la température des gaz d'échappement en amont de la turbine (7) disposée sur la ligne d'échappement (5) du moteur, l'estimateur fournissant en sortie l'estimation de la température des gaz d'échappement en aval de ladite turbine.

10. Système selon l'une des revendications 8 ou 9, caractérisé en ce que l'estimateur comporte un module de post-traitement (18) disposé en aval du réseau de neurones (17) et adapté pour traiter l'estimation de température réalisée par le réseau de neurones.

11. Système selon la revendication précédente, caractérisé en ce qu'il comporte une rétroaction de la température (S) estimée, disponible en sortie du module de post traitement.

12. Système selon l'une des revendications 8 à 11, caractérisé en ce que l'estimateur comporte un module de prétraitement (16) disposé en amont du réseau de neurones (17) et adapté pour traiter une ou plusieurs des

variables en entrée de l'estimateur (15) en réalisant des calculs basés sur des relations physiques connues.

13. Moteur (1) à combustion interne comportant un système d'estimation d'une température des gaz d'échappement selon l'une quelconque des revendications 8 à 12.

14. Véhicule automobile comportant un système d'estimation d'une température des gaz d'échappement selon l'une quelconque des revendications 8 à 12.

FIG.1

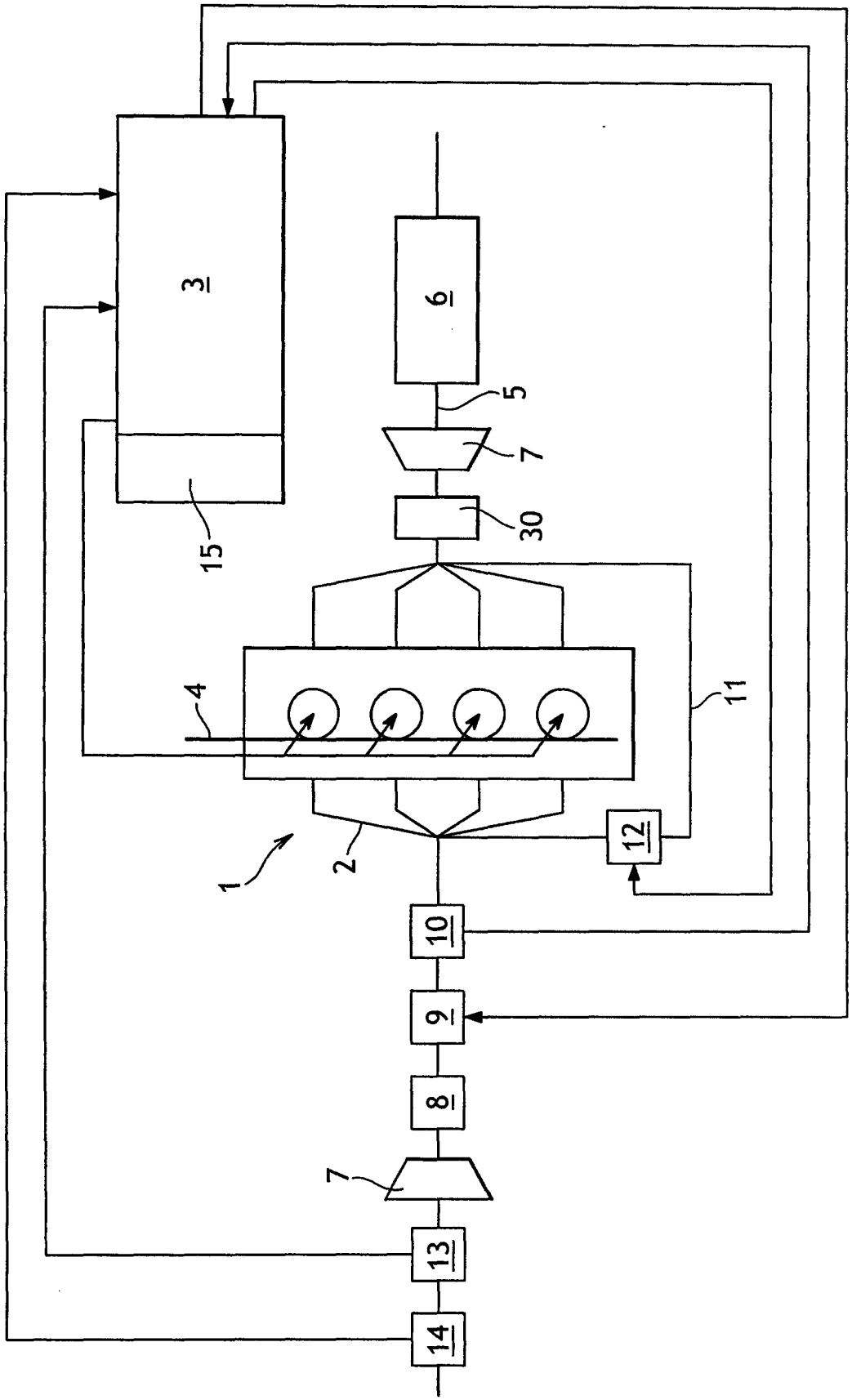


FIG.2

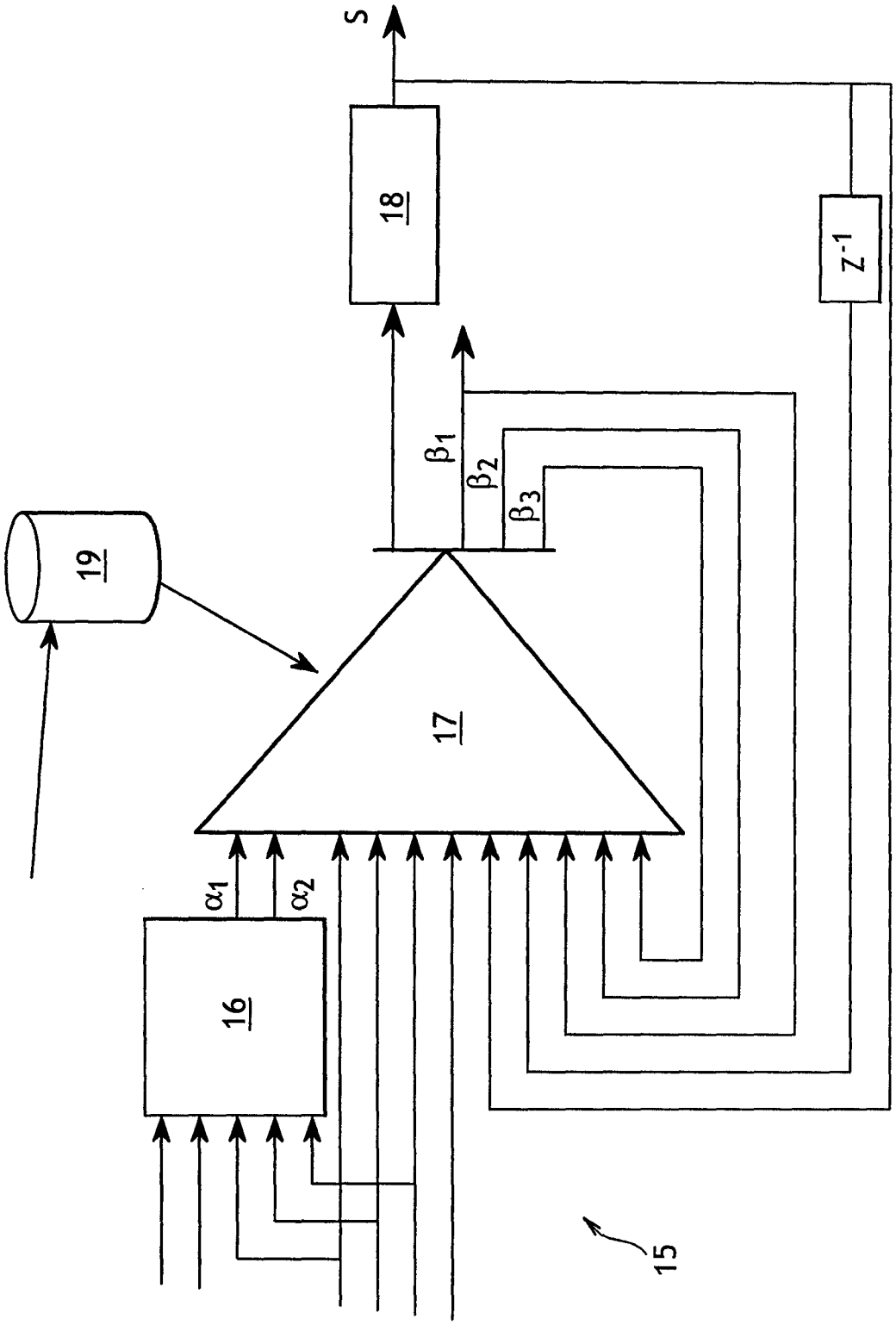
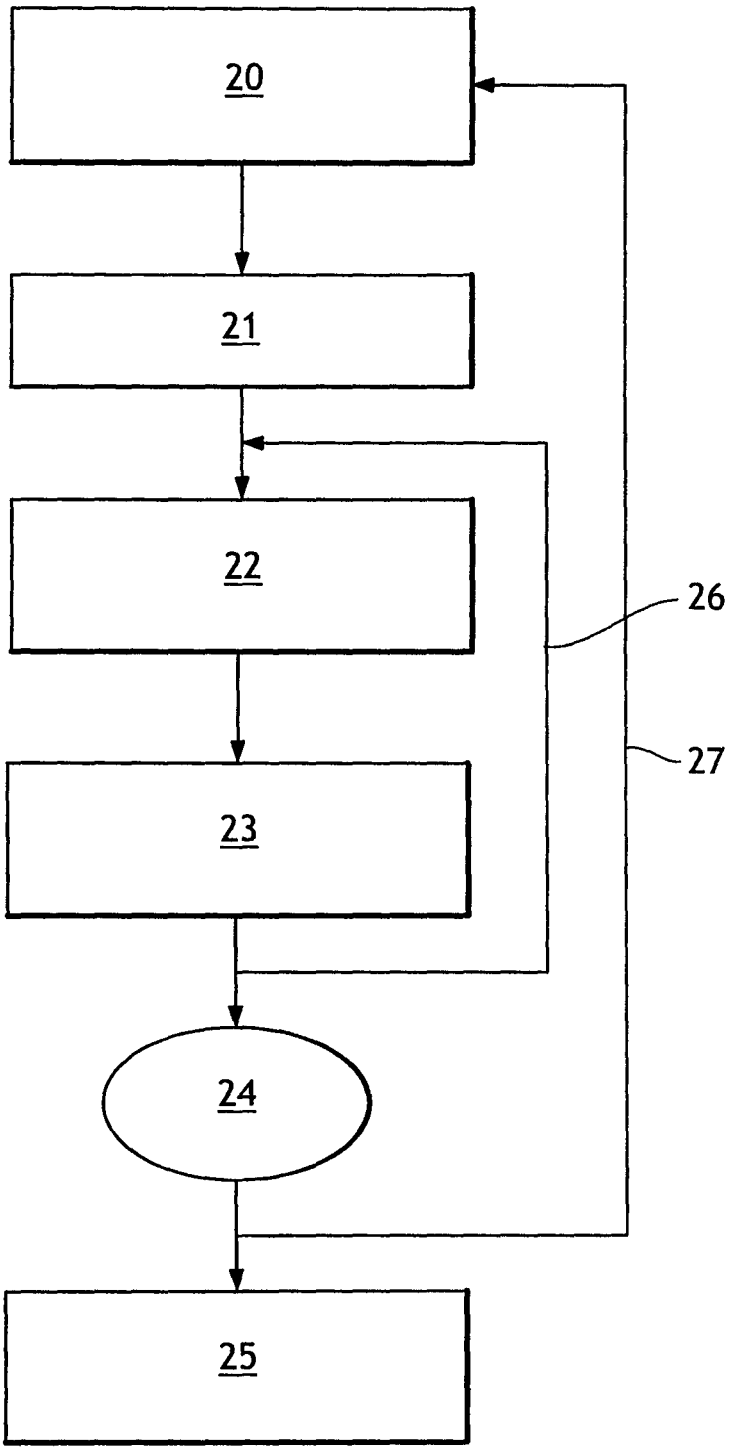


FIG.3



INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2005/050811

A. CLASSIFICATION OF SUBJECT MATTER
F02B77/08 G05B13/02 F02B37/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
F02B G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 401 457 B1 (WANG YUE YUN ET AL) 11 June 2002 (2002-06-11) column 11, line 66 - column 12, line 11; figure 5	1,8
A	EP 1 041 264 A (BAYERISCHE MOTOREN WERKE AKTIENGESELLSCHAFT) 4 October 2000 (2000-10-04) paragraphs '0009! - '0022!; figure 1	1,8
A	EP 1 024 260 A (FORD GLOBAL TECHNOLOGIES, INC) 2 August 2000 (2000-08-02) paragraph '0029!	1,8
A	WO 96/05421 A (MECEL AB; NYTOMT, JAN) 22 February 1996 (1996-02-22) abstract	1,8

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 January 2006	08/02/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Tietje, K

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2005/050811

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 832 421 A (SANTOSO ET AL) 3 November 1998 (1998-11-03) abstract ----	1,8
A	EP 1 357 487 A (UNITED TECHNOLOGIES CORPORATION) 29 October 2003 (2003-10-29) abstract ----	1,8

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/FR2005/050811

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6401457 B1	11-06-2002	GB 2390688 A	14-01-2004
		WO 02061255 A2	08-08-2002
EP 1041264 A	04-10-2000	DE 19914910 A1	26-10-2000
EP 1024260 A	02-08-2000	DE 60000081 D1	11-04-2002
		DE 60000081 T2	17-10-2002
		US 6067800 A	30-05-2000
WO 9605421 A	22-02-1996	SE 509805 C2	08-03-1999
		SE 9402687 A	12-02-1996
US 5832421 A	03-11-1998	AT 205310 T	15-09-2001
		CN 1240521 A	05-01-2000
		DE 69706563 D1	11-10-2001
		DE 69706563 T2	11-07-2002
		EP 0944866 A1	29-09-1999
		ES 2167023 T3	01-05-2002
		JP 2002501584 T	15-01-2002
		KR 2000057472 A	15-09-2000
		PL 333952 A1	31-01-2000
		RU 2213997 C2	10-10-2003
		WO 9826336 A1	18-06-1998
EP 1357487 A	29-10-2003	JP 2004028087 A	29-01-2004
		US 2003200069 A1	23-10-2003

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2005/050811

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
F02B77/08 G05B13/02 F02B37/00

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
F02B G05B

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela estréalisable, termes de recherche utilisés)
EPO-Internal

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	US 6 401 457 B1 (WANG YUE YUN ET AL) 11 juin 2002 (2002-06-11) colonne 11, ligne 66 - colonne 12, ligne 11; figure 5	1,8
A	EP 1 041 264 A (BAYERISCHE MOTOREN WERKE AKTIENGESELLSCHAFT) 4 octobre 2000 (2000-10-04) alinéas '0009! - '0022!; figure 1	1,8
A	EP 1 024 260 A (FORD GLOBAL TECHNOLOGIES, INC) 2 août 2000 (2000-08-02) alinéa '0029!	1,8
A	WO 96/05421 A (MECEL AB; NYTOMT, JAN) 22 février 1996 (1996-02-22) abrégé	1,8

-/--

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent
"E" document antérieur, mais publié à la date de dépôt international ou après cette date
"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)
"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens
"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée
"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention
"X" document particulièrement pertinent; l'inven tion revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément
"Y" document particulièrement pertinent; l'inven tion revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier
"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
31 janvier 2006	08/02/2006

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Tietje, K

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2005/050811

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	US 5 832 421 A (SANTOSO ET AL) 3 novembre 1998 (1998-11-03) abrégé	1,8
A	EP 1 357 487 A (UNITED TECHNOLOGIES CORPORATION) 29 octobre 2003 (2003-10-29) abrégé	1,8

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°
PCT/FR2005/050811

Document brevet cité au rapport de recherche		Date de publication	Membre(s) de la famille de brevet(s)		Date de publication
US 6401457	B1	11-06-2002	GB	2390688 A	14-01-2004
			WO	02061255 A2	08-08-2002
EP 1041264	A	04-10-2000	DE	19914910 A1	26-10-2000
EP 1024260	A	02-08-2000	DE	60000081 D1	11-04-2002
			DE	60000081 T2	17-10-2002
			US	6067800 A	30-05-2000
WO 9605421	A	22-02-1996	SE	509805 C2	08-03-1999
			SE	9402687 A	12-02-1996
US 5832421	A	03-11-1998	AT	205310 T	15-09-2001
			CN	1240521 A	05-01-2000
			DE	69706563 D1	11-10-2001
			DE	69706563 T2	11-07-2002
			EP	0944866 A1	29-09-1999
			ES	2167023 T3	01-05-2002
			JP	2002501584 T	15-01-2002
			KR	2000057472 A	15-09-2000
			PL	333952 A1	31-01-2000
			RU	2213997 C2	10-10-2003
			WO	9826336 A1	18-06-1998
EP 1357487	A	29-10-2003	JP	2004028087 A	29-01-2004
			US	2003200069 A1	23-10-2003