RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ DES FRERES MENTOURI CONSTANTINE 1
FACULTE DES SCIENCES DE LA TECHNOLOGIE

DÉPARTEMENT D'ÉLECTRONIQUE

Thèse

N° D'ordre :
Série :

Présentée pour obtenir le diplôme de
Doctorat en Sciences

Spécialité : Electronique
Option : Contrôle

Par

Ilyes Boulkaibet

Contribution à la commande prédictive non linéaire

Soutenue le 04 /02 /2018 devant le jury composé de:

Président : Zoheir Hammoudi Professeur, Université de Constantine 1.
Rapporteur : Khaled Belarbi Professeur, Ecole Nationale Polytechnique de Constantine.
Examinateurs : Fella Hachouf Professeur, Université de Constantine 1.
Ahmed Hafaifa Professeur, Université de Djelfa.
Kheireddine Lamamra Maître de conférences A, Université Oum El Bouaghi.

Année : 2018

Résumé

Dans cette thèse, deux stratégies de la commande prédictive généralisée (GPC) floues sont proposées pour les systèmes non linéaires discrets à travers les systèmes flous de type Takagi-Sugeno (TS). Ces stratégies sont basés sur les méthodes à noyaux. Dans la première approche, dans laquelle l'apprentissage des paramètres des conséquences du système TS flou est réalisé par la méthode de Régression Ridge à Noyau (TS-KRR), les systèmes non linéaires inconnus sont modélisés à partir des données d'entrées-sorties. Deux étapes principales sont nécessaires pour construire l'approche hors ligne de TS-KRR. La première consiste à utiliser une méthode de partitionnement basée sur l'optimisation par essaim de particules, Particle Swarm Optimisation (PSO), qui sépare les données d'apprentissage en groupes et obtient les paramètres des antécédents du modèle TS flou. Dans la deuxième étape, les paramètres des conséquences sont obtenus à l'aide d'une méthode de régression à noyaux. En outre, la commande prédictive généralisée floue est construite en intégrant le modèle TS-KRR avec la commande GPC.

Une version adaptative (en ligne) de TS-KRR est proposée et intégrée avec la commande GPC. Dans cette approche, les paramètres des antécédents sont initialisés avec une méthode de partitionnement simple, K-moyennes. La méthode des moindres carrés récursifs à noyau est ensuite utilisée avec une fenêtre coulissante (Sliding-Window KRLS) pour calculer les paramètres des conséquences. Pour chaque méthode proposée (les stratégies basées sur les modélisations en ligne et hors ligne), deux études de simulation ont été présentées pour justifier la validité des approches proposées et les résultats ont été comparés avec d'autres techniques citées dans les références.

Suivant la même méthodologie, nous proposons une autre approche de commande prédictive généralisée flou basée sur un système flou de type Takagi-Sugeno où

l'apprentissage se fait à partir d'une régression à vecteurs de support en moindres carrés multi-noyaux (TS-LSSVR). Dans la première étape, qui est une étape hors ligne, les paramètres des antécédents sont initialisés par un algorithme de partitionnement. La deuxième étape, qui est une étape en ligne, consiste à ajuster les paramètres des antécédents. Enfin, la dernière étape en ligne consiste à utiliser l'algorithme des moindres carrés récursifs à noyau avec un budget fixe (Fixed-Budget KRLS) pour ajuster les paramètres des conséquences. De plus, une commande prédictive floue adaptative est introduite utilisant la combinaison TS-LSSVR et commande GPC. Les résultats de simulation ont montrés que les approches de commande proposées sont efficaces et capables de faire face aux perturbations et variations des systèmes non linéaires.

Mots-clés **:** Commande prédictive généralisé; système flou de type Takagi-Sugeno; régression à noyau; système flou de type TS-KRR ; moindres carrés récursifs à noyau avec une fenêtre coulissante; régression à vecteurs de support en moindres carrés; C-moyennes flous; système flou de type TS-LSSVR ; moindres carrés récursifs du noyau avec un budget fixe.

ملخص

في هذه الأطروحة، قمنا باقتراح اثنين من أساليب التحكم الالي التي تعتمد على فكرة التنبؤ العامة للأنظمة الغير الخطية وذلك بدمجها مع نوع جديد اقترحناه من الانضمة الغامضة التي ترتكز في تركيبتها على أساس النواة. في الاسلوب الأول المقترح من الانضمة الغامضة و الذي بدوره يقوم على استراتيجية الحركة الرجعة و التسلسلية للانوية, يتم تقريب الانضمة الغير خطية و الغير معروفة الشكل عن طريق الانظمة الغامضة {تاكاجي-سيجونو} و التي يتم استخراجها عن طريق بيانات الانضمة و التي تستخرج عن طريق اجراء تجارب. هاته الطريقة تعتمد على خطوتين رئسيتين لبناء هذا الاسلوب الغامضو الحديث: تتمثل الخطوة الأولى في استخدام خوارزمية تجميعية مثل خوارزمية التجميعية التي تعتمد على الخوارزمية سرب المجسمات التحسينية و بهده الطريقة يتم تقسيم البيانات إلى مجموعات, وبهذا يمكننا الحصول على معاملات العنصر الشرطي في القضية المنطقية للانضمة الغامضة. في الخطوة الثانية يتم استعمال استراتيجية الحركة الرجعة و التسلسلية للانوية لحساب معاملات العنصر الناتج في القضية المنطقية للانضمة الغامضة. علاوة على ذلك, نقوم باقتراح استراتيجية التحكم التنبؤي العام والغامضة وذلك بادماج النظام الغامض المبني على استراتيجية الحركة الرجعة و التسلسلية للانوية مع استرايتجية التحكم التنبؤي العامة. وبعد ذلك، اقترحنا طريقة تكيفية، اي تعتمد على القيم الآنية للمتغيرات، من نسخة المقترحة للنضام الغامض الذي يقوم على استراتيجية الحركة الرجعة و التسلسلية للانوية و قمنا ايضا بادماجها مع طريقة التحكم التنبؤي العامة للحصول على وحدة تحكم تكييفية تنبؤية عامة قادرة على التحكم في اي نظام غير خطي. النظام التكييفي الغامض المقترح يعتمد اساسا على خورزمية التجميع المتوسطة لتهيئة قيم المعاملات في العنصر الشرطي في القضية المنطقية للانضمة الغامضة و بعدها يتم تعديل هته المعملات عند كل خطوة زمنية بينما معاملات العنصر الناتج في القضية المنطقية للانضمة الغامضة يتم حسابها عند كل خطوة زمنية عن طريق خوارزمية التكرارية للنافدة المنزلقة للمربعات الصغرى مع الانوية. في هذا العمل, نقدم دراسات للمحاكات و للتحكم لكل هيكل تحكم (استراتيجيات التحكم المبنية على أساس التكييف و المبنية على اسس لا آنية) على ان يتم تبرير صحة النتائج التي حصلنا عليها عن طريق مقارنة النتائج مع غيرها من التقنيات المذكورة في المراجع.

علاوة على ذلك، نقترح ايضا طريقة تحكم تكييفية جديدة للتحكم في الانظمة الغير خطية حيث ان هته الطريقة مبنية على طريقة التحكم التنبؤي العامة و نظام غامض اخر مؤسس على طريقة المربعات الصغري للانوية مع الانحدار المدعم. هته الاستراتيجية الحديثة لبناء الانظمة الغامضة يمكن انشائها من خلال ثلاث خطوات اساسة: الخطوة الاولي, اين يتم تطبيق خورزمية تجميعية تسمي طريقة التجميع الغامضة و التي بدورها تقسم البيانات إلى مجموعات, وبهذا يمكننا تهيئة معاملات العنصر الشرطي في القضية المنطقية للانضمة الغامضة. في الخطوة الثانية, نقوم بتعديل معاملات العنصر

الشرطي في القضية المنطقية للانضمة الغامضة عند كل خطوة زمنية. اخيرا و في الخطوة الثالة يتم حساب معاملات العنصر الناتج في القضية المنطقية للانضمة الغامضة عن طريق خوارزمية التكرارية للميزانية الثابته للمربعات الصغرى مع الانوية. زيادتا على ذلك، نقوم باقتراح التحكم التنبؤي العام المدمجة مع النظام الغامض الحديت لنحصل على طريقة تحكم آلية قادرة على التحكم في اغلب الانظمة الغير خطية. ولكي يتم التحقق من الطريقة الجديدة للتحكم الآلي, نقوم بالتحكم في نظاميين غير خطيين حيث ان النتائج اظهرت كفائت هته الطريقة في السيطره على تلك الانظمة .

الكلمات الرئيسية: التحكم التنبؤي العام؛ النظام الغامض تاكاجي-سوجينو؛ استراتيجية الحركة الرجعة و التسلسلية للانوية؛ خوارزمية التجميع؛ الخوارزمية سرب المجسمات التحسينية ؛ نظام تاكاجي-سوجينو الغامض المبني على استراتيجية الحركة الرجعة و التسلسلية للانوية؛ خوارزمية التكرارية للنافدة المنزلقة للمربعات الصغرى مع الانوية ؛ طريقة التجميع الغامضة؛ خوارزمية التكرارية للميزانية الثابته للمربعات الصغرى مع الانوية.

Abstract

In this thesis, two fuzzy Generalized Predictive Control (GPC) methods are proposed for discrete-time nonlinear systems via Takagi-Sugeno system based Kernel methods. In the first approach, which is based on Kernel ridge Regression strategy (TS-KRR), the unknown nonlinear systems is approximated by learning the Takagi-Sugeno (TS) fuzzy parameters from the input-output data. Two main steps are required to construct the offline TS-KRR approach: the first step is to use a clustering algorithm such as the clustering based Particle Swarm Optimization (PSO) algorithm that separates the data into groups and obtains the antecedent TS fuzzy model parameters. In the second step, the consequent TS fuzzy parameters are obtained using a Kernel ridge regression algorithm. Furthermore, the TS based predictive control is created by integrating the TS-KRR into the Generalized Predictive Controller. Next, an adaptive, online, version of TS-KRR is proposed and integrated with the GPC controller resulting an efficient adaptive fuzzy generalized predictive control methodology. In the adaptive TS-KRR algorithm, the antecedent parameters are initialized with a simple K-means algorithm and updated using a simple backpropagation algorithm. Then, the consequent parameters are obtained using the sliding-window Kernel Recursive Least squares (KRLS) method. For each control structure (the control strategies based on the online and offline strategies), two simulation studies were presented to justify the validity of the proposed approaches and the results were compared with other techniques cited in references.

Furthermore, another adaptive fuzzy Generalized Predictive Control (GPC) is proposed for discrete-time nonlinear systems via Takagi-Sugeno system based Kernel Least Squares Support Vector Regression (TS-LSSVR). The proposed adaptive TS-LSSVR strategy is constructed using a multi-kernel lest squares support vector regression where the learning procedure of the proposed TS-LSSVR is achieved in three steps: In the first step, which is an

offline step, the antecedent parameters of the TS-LSSVR are initialized using a fuzzy c-means clustering algorithm. The second step, which is an online step, deals with the adaptation of the antecedent parameters which can be done using a backpropagation algorithm. Finally, the last online step is to use the Fixed-Budget Kernel Recursive Least Squares method to obtain the consequent parameters. Furthermore, an adaptive generalized predictive control for nonlinear systems is introduced by integrating the proposed adaptive TS-LSSVR into the generalized predictive controller (GPC). The reliability of the proposed adaptive TS GPC controller was investigated by controlling two nonlinear systems: A surge tank and continuous stirred tank reactor (CSTR) systems. The proposed controller has demonstrated good results and efficiently controlled the nonlinear plants. Furthermore, the adaptive TS GPC controllers have the ability to deal with disturbances and variations in the nonlinear systems.

Keywords: Generalized Predictive Control; Takagi-Sugeno fuzzy system; Kernel ridge regression; Clustering algorithm; Particle Swarm Optimization; Takagi-Sugeno system based Kernel ridge regression; Sliding-window Kernel Recursive Least squares; Least Square Support Vector Regression; Fuzzy c-Means Clustering; Fixed-Budget Kernel Recursive Least-Squares.

Remerciements

Élaborer un bon travail est une tâche difficile qu'on ne peut pas réaliser tout seul. En premier, je tiens tout particulièrement à remercier monsieur Khaled BELARBI, Professeur à école nationale Polytechnique de Constantine, pour avoir assuré l'encadrement de ce travail, son savoir et expérience, son orientation et support qu'il m'a offert ont contribué à ma formation scientifique.

Je remercier aussi monsieur Zoheir Hammoudi, Professeur à l'université de Constantine 1, pour avoir présidé le jury de ma soutenance de thèse.

J'exprime mes reconnaissances à Mme Fella Hachouf, Professeur à l'université de Constantine 1, monsieur Ahmed Hafaifa, Professeur à l'université de Djelfa, ainsi qu'à monsieur Kheireddine Lamamra, Maître de conférences A à l'université Oum El Bouaghi, pour l'intérêt qu'ils ont porté à ce travail et qui ont accepté d'en être les rapporteurs. Sans oublier à mon amie et collègue monsieur Sofiane Bououden, pour son amitié sincère, pour ses conseils, et tous les services qu'il m'a pu me rendre durant cette thèse.

A ma famille ma source de bonheur et de support j'exprime mes vives reconnaissance: mes parents, mes chères sœurs et mes frères; mon épouse ainsi ma belle-famille qui ont été toujours là dans les moments difficiles.

Et enfin je voudrais remercier aussi toutes les personnes qui ont participé de près ou de loin à mes recherches et à l'élaboration de ce mémoire.

.

Boulkaibet Ilyes

Table des matières

Liste des abréviations

ANFIS	Adaptive Neuro-Fuzzy Inference System
CARIMA	Controlled Auto Regressive Integrated Moving Average
CSTR	Continuous Stirred Tank Reactor
DMC	Dynamic Matrix Control
FB-KRLS	Moindres carrés récursifs à noyau avec un budget fixe
GA	Algorithme génétique
GPC	Generalized Predictive Control
KRR	Kernel Ridge Regression
KRLS	moindres carrés récursifs à noyau
LS-SVM	Least Squares Support Vector Machine
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Perceptron multicouches
MPC	Model Predictive Control
NARX	Non linéaire autorégressive avec entrée exogène
NN	Réseaux de neurones
PSO	Particle Swarm Optimization
RLS	Moindres carrés récursifs
RMSE	Root Mean Square Error
sMAPE	symmetric Mean Absolute Percentage Error
SVM	Support Vector Machine
SW-KRLS	Sliding-Window Kernel Recursive Least squares
T-S	Takagi-Sugeno
TS-KRR	Takagi-Sugeno basé sur la méthode de Régression Ridge à noyau

TS-LSSVR Takagi-Sugeno basé sur la méthode de LS-SVR

$\mathcal{A}$	Sous-ensemble flou
$\mu_{\mathcal{A}}(x)$	Fonction d'appartenance
$\mathcal{H}$	Espace de Hilbert à Noyau Reproduisant
$\kappa(\boldsymbol{x}, \boldsymbol{y})$	Fonction noyau
$\boldsymbol{\varphi}$	Fonction de re-description
ψ_i	Degré d'activation de $i^{\text{ème}}$ règle
$\widehat{\boldsymbol{\theta}}_i$	Paramètres de conséquences de la $i^{\text{ème}}$ règle
$\boldsymbol{m}_l$	Centre de groupe l
$\boldsymbol{\sigma}_l$	Variance de groupe l
$\boldsymbol{I}$	Matrice identité
$\boldsymbol{K}$	Matrice du noyau
$\acute{\boldsymbol{K}}$	Matrice du noyau régularisé
$Dunn$	L'indice de Dunn
$u(k)$	La commande
$\Delta u(k)$	Incrément du signale du commande
$w(t+\ j)$	Référence

Liste des tableaux

Liste des figures

Introduction Générale

La commande prédictive (*en Anglais Model Predictive Control : MPC*), aussi nommée : la commande à horizon glissant, est devenue très populaire au cours des dernières années dans le milieu industriel [1]. Sa popularité est due à sa tolérance pour différents types de systèmes et le respect des contraintes imposées sur les variables commandées. Généralement, l'avantage principal des algorithmes de contrôle prédictif est son aptitude à prendre en compte les contraintes fonctionnelles et les contraintes d'exploitation du système considéré. Par l'utilisation du modèle dynamique du système à l'intérieur du contrôleur, la méthode prédictive permet d'évaluer le comportement dynamique du système. Ceci crée un effet anticipatif par exploitation de la trajectoire à suivre dans le futur.

La commande prédictive (MPC) a été créé à la fin des années soixante-dix et s'est considérablement développée depuis. *Dynamic Matrix Control* (DMC) de Cutler et Ramaker et *Identification et Commande* (IDCOM) de Richalet et al. [2, 3] sont considérées comme les premiers algorithmes développés dans la classe des algorithmes MPC. La commande DCM a trouvé une grande acceptation dans l'industrie des procédés. Par la suite, la Commande prédictive généralisée (*Generalized Predictive Control : GPC*) développée par Clarke et al. [4] a reçu une grande attention. Dans l'algorithme de GPC, le nombre des paramètres du modèle qui représente le processus est limité [5]. La commande GPC a été implantée dans un grand nombre d'applications industrielles. De plus cette méthode peut être appliquée aux systèmes multivariables et permet de contrôler un grand nombre de processus (instables, à retard pur, à non minimum de phase).

Malheureusement, l'optimisation quadratique associée à l'algorithme GPC ne peut être résolue que pour le cas des systèmes linéaires. Cependant, dans le domaine industriel, la connaissance exacte du modèle du système sur toute sa plage de fonctionnement n'est pas une chose aisée de part la complexité même du système. En réalité, la plupart des systèmes industriels sont complexes, non linéaires, incertains et leurs modèles mathématiques peuvent ne pas être disponibles. Pour faire face à la non linéarité (ou à l'absence des modèles mathématiques) des

systèmes, de nombreux chercheurs ont été engagées dans le développement des méthodes GPC non linéaires (NGPC). Pour utiliser un GPC pour les systèmes non linéaires, certains chercheurs ont considéré la linéarisation des systèmes comme une solution. Cette approche est vue comme l'approche la plus simple. Cependant, les points de fonctionnement des systèmes peuvent changer avec le temps ce qui entraînera une mauvaise performance des contrôleurs.

Au cours des trente dernières années, les méthodes d'approximation telles que la logique floue [6, 7] et les réseaux de neurones (NN) [8] ont été largement appliquées pour représenter la dynamique des procédés non linéaires ou inconnus qui finiront par simplifier les procédées d'obtention des contrôleurs. Ces méthodes présentent des approximateurs universels pour décrire les systèmes non linéaires. La méthodologie des réseaux de neurones a montré de bons résultats dont les approximations des fonctions non linéaires. Le réseau de neurones a l'avantage d'apprentissage à partir des données entrées-sorties et les propriétés de l'intégration de cet méthode dans la commande prédictive généralisée (commande prédictive généralisée neuronale) ont été démontrées par de nombreux chercheurs [9, 10, 11, 12].

D'autre part, le système flou de type Takagi-Sugeno (TS) [13] a été confirmé comme l'une des approches d'approximation efficaces pour les systèmes non linéaires. Généralement, les systèmes d'inférence floue peuvent être considérés comme des systèmes logiques qui utilisent des règles linguistiques pour mettre des relations entre les variables d'entrées et des sorties. Ainsi, le système flou de type Takagi-Sugeno est capable d'approximer des systèmes non linéaires avec une grande précision par des données précédentes et par une connaissance préalable des systèmes non linéaire, et de traiter les non linéarités sans aucune hypothèse sur leur nature. La commande GPC combinée à la théorie de la logique floue s'avère un outil bien adapté à la commande d'une large classe de systèmes (stable, instable en boucle ouverte, à phase non minimale, avec retard) [14, 15, 16].

En outre, de nombreux chercheurs ont été intéressés pour combiner la logique floue avec les réseaux de neurones pour obtenir des approches sophistiquées pour l'identification du système. Jang [17] a introduit l'approche du système d'inférence adaptative neuro-flou (*Adaptive Neuro-Fuzzy Inference System : ANFIS*) pour identifier les systèmes complexes. L'algorithme ANFIS combine la lisibilité et la souplesse de la logique floue (en définissant la base des connaissances) et les capacités d'apprentissage à partir des données (entrées /sorties) des réseaux de neurones. Il

a été montré que les résultats de la commande prédictive peuvent être améliorés en utilisant cette approche [18].

Récemment, les méthodes à noyau tel que les machines à vecteur de support (*Support Vector Machine* : SVM) [19], ont montré des résultats intéressants dans la modélisation et l'identification de systèmes complexes, et ont été intégrées plusieurs fois avec la commande prédictive pour contrôler des systèmes non linéaires [20, 21, 22, 23, 24]. Iplikci [20, 21, 22] a introduit une nouvelle commande prédictive généralisée non linéaire basé sur les machines à vecteurs de support. Dans cette approche, le SVM a été utilisé pour modéliser les systèmes non linéaires alors qu'une extension de Taylor (dérivée de deuxième ordre) était utilisée pour obtenir le signal de contrôle. Malheureusement, le vecteur gradient et la matrice Hessienne (extension de la formule Taylor au deuxième ordre) ont été nécessaires pour obtenir le signal de contrôle à chaque itération (temps d'échantillonnage). De plus, ils ont été obtenus à partir du modèle SVM (un modèle approximatif) ce qui peut diminuer la précision du contrôleur NGPC car la précision des valeurs obtenues dépend de la précision du modèle SVM. En outre, le temps d'exécution nécessaire pour exécuter une itération de cet algorithme est relativement grand (une matrice de noyau avec une taille énorme est nécessaire pour calculer le gradient et la matrice Hessienne). Une autre méthode GPC non linéaire basée sur une machine à vecteurs de support aux moindres carrés en ligne (*On-line Least Squares Support Vector Machine : On-line LSSVM*) a été proposée par Li-Juan Li et al. [24]. Dans cette approche, la matrice Hessienne n'est pas nécessaire; cependant, le modèle approximatif obtenu par la méthode LSSVM est linéarisé à chaque itération (temps d'échantillonnage) ce qui rend cette approche coûteuse en termes de calcul, et aussi le modèle linéarisé peut ne pas être tout à fait exact.

Les méthodes à noyaux peuvent être utilisées de différentes manières pour approximer les systèmes non linéaires. Ces méthodes peuvent être combinées avec le raisonnement flou pour éviter l'utilisation du vecteur gradient ou la matrice Hessienne et donc l'approximation du modèle peut être amélioré [25, 26, 27, 28, 29]. Plusieurs chercheurs ont essayé d'utiliser la méthode de régression par les machines à vecteurs de support (SVR) pour définir un système flou de type Takagi-Sugeno. Chiang et al. [25] a proposé un modèle de réseau flou basé sur l'approche du SVR. Dans cette approche, les paramètres des antécédents du système flou sont produits en fonction des vecteurs de support où les fonctions des noyaux sont considérées comme les fonctions d'appartenance du système flou. Les paramètres des conséquences sont estimés

séparément par l'algorithme des moindres carrées récursif. L'avantage d'utiliser SVR pour exprimer les paramètres des antécédents est que le nombre de règles floues est automatiquement obtenu (égal au nombre de vecteurs de support). Malheureusement, le nombre de vecteurs de support est généralement grand, ce qui rend cette approche coûteuse en termes de calcul. D'autres chercheurs ont essayé d'obtenir les paramètres des conséquences utilisant la méthode de SVR. Juang et al. [27, 29] ont proposé une nouvelle approche flou de type Takagi Sugeno hors ligne (offline) basée sur la méthode SVR. Dans cette approche, les paramètres des conséquences sont identifiés à l'aide d'un SVR alors que les paramètres antécédents sont obtenus par une méthode de partitionnement (*Clustering Algorithm*). Malheureusement, une fonction du noyau complexe (le produit de deux fonctions du noyau) a été utilisée pour obtenir les paramètres des conséquences alors que la forme obtenue du modèle flou de type Takagi-Sugeno a un terme biais. Ce qui est difficile à mettre en œuvre comme une approche adaptative (en ligne).

Dans ce travail, une nouvelle approche flou de type Takagi-Sugeno basée sur les méthodes du noyau sont introduites et utilisées pour l'identification et le contrôle de systèmes. Dans la section suivante, nous allons citer les principales contributions de ce travail.

Contributions de la thèse

Cette thèse a les contributions suivantes:

- Conception d'une nouvelle méthodologie pour l'identification des procédés industriels basée sur un système flou de type Takagi-Sugeno (TS). Les paramètres des antécédents du modèle flou sont obtenus par une méthode de partitionnement alors que les paramètres des conséquences sont obtenus par régression ridge à noyau (*Kernel Ridge Regression: KRR*). Généralement, la structure de la base de règles floues est un facteur important qui affecte la précision ainsi que la performance de modélisation. Lorsque le nombre de règles est grand, les résultats de la modélisation sont bons mais le modèle flou devient complexe. Pour faire face à cela, les méthodes de partitionnement sont utilisées pour réduire les règles floues nécessaires à la description du système réel. Aussi, dans le cas où le système flou est complexe, le modèle TS floue peut-être exécuté hors ligne lorsqu'un ensemble des donnés d'entrée-sortie est suffisante et décrite correctement le processus non linéaire inconnu. En outre, la version adaptative

du système TS flou est présentée lorsque les données d'apprentissage ne suffisent pas pour écrire le processus non linéaire inconnu.

- Après avoir défini le modèle TS basée sur une régression ridge à noyau, l'étape suivante consiste à intégrer le modèle TS proposé avec le GPC pour contrôler les systèmes non linéaires.
- Le modèle flou de type Takagi-Sugeno (TS) est également introduit en fonction de la méthode de régression par les machines à vecteurs de support en moindres carrés multi-noyaux (LSSVR). Dans cette approche la fonction du noyau est une fonction augmentée multi-noyau et les paramètres des conséquences sont définis par la méthode des machines à vecteurs de support en moindres carrés pour la régression. Le système flou adaptatif proposé a été utilisé pour identifier des systèmes non linéaires.
- Enfin, le système flou de type Takagi-Sugeno basée sur la régression à vecteurs de support en moindres carrés multi-noyaux est intégré dans le GPC pour contrôler les systèmes non linéaires

Organisation de la thèse

Cette thèse est divisée en cinq chapitres, qui développent les résultats originaux obtenus durant le travail de recherche:

- **Le premier Chapitre** décrit en détail la théorie et les fondements des méthodes à noyaux. Nous présenterons la régression ridge à noyau et les machines à vecteurs de support dans le cadre de la régression. Le but principal est d'introduire les notions nécessaires pour la suite du manuscrit.
- **Le deuxième chapitre** présente les principes fondamentaux ainsi que les stratégies les plus couramment utilisée pour l'analyse des modèles flous de type Takagi-Sugeno. La structure ainsi que la façon d'obtenir les modèles de Takagi-Sugeno avec différentes approches sont également présentées.
- **Le troisième chapitre** introduit un nouveau modèle floue de type Takagi-Sugeno basé sur la régression ridge à noyau. Le modèle flou de type Takagi-Sugeno sera intégré au GPC pour construire un GPC non linéaire.

- **Le quatrième chapitre** dans ce chapitre, la performance du nouveau modèle TS introduit au chapitre 3 sera étudiée dans deux exemples: deux exemples d'identification et deux exemples de commande des systèmes non linéaires.
- **Le cinquième chapitre:** présente une version modifiée du modèle flou de type Takagi-Sugeno où la méthode du LSSVR sera utilisée à la place de l'algorithme de KRR. Le nouveau modèle flou sera intégré au GPC et la performance du nouveau modèle TS sera étudiée en identifiant et en contrôlant un système non linéaire.

- **Conclusion générale :** Enfin, nous conclurons notre travail par une conclusion générale qui englobera tout ce qui a été développé, et nous esquisserons quelques perspectives de recherche.

Chapitre 1

Méthodes à Noyaux

1.1 Introduction

Les modèles de régression linéaire sont très populaires car conceptuellement simples et faciles à mettre en œuvre. Cependant, ces modèles sont limités à cause de cette hypothèse de linéarité qui est improbable dans la plupart des problèmes réels. Les méthodes à noyaux constituent une classe de modèles qui étend de manière astucieuse les méthodes linéaires au cas non linéaire. La régression non linéaire devient possible grâce à l'utilisation des fonctions à noyaux. Généralement, les méthodes basées sur les fonctions à noyaux sont habituellement appliquées pour les problèmes complexes lorsque les données (entrées\sorties), aussi connu sous le nom d'ensemble d'apprentissage, ne sont pas séparables linéairement. Dans ce cas, les fonctions du noyau sont appliquées pour transformer vers un nouvel espace de dimension supérieure où les données sont en effet séparables linéairement. La régression à noyau a été largement appliquée à divers types de domaines, y compris les finances empiriques, les bio-informatiques et le traitement d'images. Les méthodes à noyaux sont basées sur la manipulation et l'évaluation de fonctions noyaux et que nous définissons dans ce chapitre. Grâce à l'utilisation des fonctions noyau, il devient ainsi possible d'utiliser des techniques simples et rigoureuses et traiter des problèmes non linéaires. C'est pourquoi ces méthodes sont devenues très populaires. Ce chapitre est organisé comme suit :

Tout d'abord, l'espace de Hilbert à noyau est brièvement discuté où plusieurs définitions et théorèmes sont donnés pour comprendre les bases des méthodes à noyaux. Ensuite, certain nombre des fonctions noyaux sont décrites. Après, la régression ridge à noyau, la régression des machines à vecteurs de support et la régression à vecteurs de support en moindres carrés sont présentées en détail.

1.2 Espace de Hilbert à noyau reproduisant :

Généralement, un problème non linéaire peut être transformé en un problème linéaire, par le passage de l'espace d'origine vers un espace dimensionnel très élevé, ce qui peut conduire à un nombre énorme de paramètres inconnus à estimer. Heureusement, le fait de choisir de transférer les problèmes non linéaires donne un Espace de Hilbert à Noyau Reproduisant (RKHS) [30, 31, 32] rend les calculs beaucoup plus faciles.

Définition 1.1 (**théorie des RKHS**) : soit $\mathcal{H}$ une espace de fonctions réelles définies sur un ensemble indexé $\mathcal{X}$: $\mathcal{H} = \{\sum_{i=1}^{m} \alpha_i \kappa(., \boldsymbol{x}_i) : m \in \mathbb{N}, \boldsymbol{x}_i \in \mathcal{X}, \alpha_i \in \mathbb{R}, i = 1, \dots, m\}$. $\mathcal{H}$ est appelé Espace de Hilbert à Noyau Reproduisant s'il existe une fonction $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ possède les propriétés suivant [30, 31, 32] :

1. $\forall \boldsymbol{x} \in \mathcal{X},$ la fonction $\kappa(., \boldsymbol{x})$ est appartient a $\mathcal{H}$.
2. La fonction κ est une fonction noyau reproduisant, c'est-à-dire telle que pour toute fonction $f \in \mathcal{H}$, on a : $\langle f . \kappa(., \boldsymbol{x}) \rangle_{\mathcal{H}} = \sum_{i=1}^{m} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) = f(\boldsymbol{x})$.

Spécifiquement on a la propriété de reproduction $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \kappa(\boldsymbol{y}, .) . \kappa(\boldsymbol{x}, .) \rangle_{\mathcal{H}}$ où $\langle \kappa(\boldsymbol{y}, .) . \kappa(\boldsymbol{x}, .) \rangle_{\mathcal{H}}$ est le produit scalaire dans $\mathcal{H}$. Le fait que la fonction noyau soit reproduisant signifie que toute fonction $f \in \mathcal{H}$ est égale à un produit scalaire qui est aussi une combinaison linéaire finie de fonctions de base. Le produit scalaire sur $\mathcal{H}$ est alors défini comme suit :

Considérons les fonctions $f, g \in \mathcal{H}$ définies par : $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x})$ et $g(\boldsymbol{x}) = \sum_{j=1}^{n} \sigma_j \kappa(\boldsymbol{x}_j, \boldsymbol{x})$, alors :

$$\langle f . g \rangle_{\mathcal{H}} = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \sigma_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{m} \alpha_i g(\boldsymbol{x}_i) = \sum_{j=1}^{m} \sigma_j f(\boldsymbol{x}_j) \tag{1.1}$$

Définition 1.2 (Fonction noyau) : Une fonction noyau est une fonction $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ définie par [32] :

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{z}) \rangle_{\boldsymbol{\mathcal{H}}} \tag{1.2}$$

où $\boldsymbol{\varphi} : \mathcal{X} \to \mathcal{H}$ est une fonction de re-description qui projette les vecteurs de $\mathcal{X}$ dans un espace de Hilbert $\mathcal{H}$.

Ce résultat fournit une interprétation sur les noyaux qui peuvent être vus comme une projection dans un espace des caractéristiques (*feature space*) de dimension potentiellement infinie et dans lequel on utilise des techniques simples pour traiter des problèmes non linéaires. Ceci explique que l'on présente souvent les noyaux comme une mesure de similarité. Il s'agit là d'un changement de paradigme majeur. En effet, chaque objet $\boldsymbol{x} \in \mathcal{X}$ est désormais représenté par un vecteur $\boldsymbol{\varphi}(\boldsymbol{x}) \in \mathcal{H}$ où l'idée principale de l'astuce des noyaux (truc du noyau) [33] consiste à ne pas calculer explicitement $\boldsymbol{\varphi}(\boldsymbol{x})$ mais plutôt à calculer uniquement des produits scalaires dans l'espace des caractéristiques, ce qui revient simplement à évaluer des noyaux. Par la facilité de sa mise en œuvre, l'astuce des noyaux a permis de transformer une série de méthodes linéaires (régression linéaire, analyse en composantes principales, … etc.) en des méthodes non linéaires en remplaçant le produit scalaire dans l'espace de départ par des noyaux non linéaires et ce pour un coût algorithmique nul.

Théorème 1.1 (Théorème de Mercer [34, 35]) : Soit $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ un espace de Hilbert, la fonction $\kappa: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est une fonction noyau si et seulement si elle est positive semi-définie. Une fonction $\kappa: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est une fonction positive semi-définie si pour tout : $m \in \mathbb{N}$, $\{(\alpha_1, \boldsymbol{x}_1), \dots, (\alpha_m, \boldsymbol{x}_m)\} \subseteq \mathcal{X} \times \mathbb{R}$, on a :

$$\sum_{i,j=1}^{m} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0 \tag{1.3}$$

Si la valeur de la somme $\sum_{i,j=1}^{m} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)$ est égale à zéro si et seulement si tous les valeurs $\alpha_i, i = 1, \dots, m$ sont nuls, alors la fonction noyau κ est une fonction strictement positive définie.

Théorème1.2 : la fonction noyau $\kappa: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est une fonction conditionnellement positive semi-définie si pour tout : $m \in \mathbb{N}$, $\{(\alpha_1, \boldsymbol{x}_1), \dots, (\alpha_m, \boldsymbol{x}_m)\} \subseteq \mathcal{X} \times \mathbb{R}$, on a :

$$\sum_{i,j=1}^{m} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$
$$Subject\ to \sum_{i=1}^{m} \alpha_i = 0 \qquad (1.4)$$

Si la valeur de la somme $\sum_{i,j=1}^{m} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)$ est égale à zéro si et seulement si tous les valeurs $\alpha_i, i = 1, \dots, m$ sont nuls, alors la fonction noyau κ est une fonction conditionnellement strictement positive définie. Notez qu'avec la méthode de SVM, l'utilisation d'un noyau conditionnellement positive semi-définie est équivalente à l'utilisation du noyau positive semi-définie associé [36].

Théorème 1.3 (Théorème de Moore-Aronszajn [30, 37]) : Si la fonction $\kappa: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est une fonction positive semi-définie, alors il existe un espace de Hilbert unique à noyau reproduisant $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ tel que la fonction κ est le noyau reproduisant associé à $\mathcal{H}$.

Théorème 1.4 (Théorème du représentant « *Representer Theorem*» [38]) : Soit un noyau reproduisant $\kappa: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, des données entrées-sorties (l'ensemble d'apprentissage) $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ et un risque empirique R_{Emp}. Soit une fonction croissante strictement monotone qui mesure la « régularité » de certains hypothèse $Reg: \mathbb{R} \to [0, \infty[$. Soit $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$espace de Hilbert induit par le noyau reproduisant κ. Alors, toute fonction $f \in \mathcal{H}$ minimisant le risque régularisé :

$$\hat{f}(\boldsymbol{x}) = Arg\underset{f \in \mathcal{H}}{\mathrm{Min}}\{R_{Emp}(f, \mathcal{D}) + \lambda Reg(f)\}$$

admet une représentation de la forme :

$$\hat{f}(x) = \sum_{i=1}^{m} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i)\,, \alpha_i \in \mathbb{R}, i = 1, \dots, m. \qquad (1.5)$$

Le théorème du représentant induit un avantage calculatoire substantiel. En effet, toute solution de la minimisation du risque appartient à un sous-espace de $\mathcal{H}$ de dimension au plus m, le nombre d'observations alors même que le problème d'optimisation porte sur l'espace $\mathcal{H}$ qui est possiblement de dimension infinie. En pratique, résoudre la minimisation du risque régularisé se réduit à un problème dans $\mathbb{R}^m$en estimant les paramètres $\alpha_i, i = 1, \dots, m$.

1.3 Exemples de noyaux

Généralement, la fonction de noyau $\kappa(\boldsymbol{x}, \boldsymbol{z})$ sélectionnée joue un rôle important pour résoudre les problèmes, comme nous le verrons plus tard dans cette thèse. Voici quelques exemples de noyaux populaires et couramment utilisés dans la littérature [32, 33] :

1. le noyau linéaire : défini par $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z} \rangle = \boldsymbol{x}^T \boldsymbol{z}$, il s'agit simplement du produit scalaire usuel dans l'espace d'entrée.
2. le noyau Gaussien: défini par $\kappa(\boldsymbol{x}, \boldsymbol{z}) = exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{z})^T(\boldsymbol{x}-\boldsymbol{z})}{\boldsymbol{\sigma}^2}\right)$ où $\boldsymbol{\sigma}$ est un vecteur appelé vecteur de covariance choisie manuellement.
3. Le noyau polynomial inhomogène : défini par $\kappa(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z} + c)^d$ où $c \in \mathbb{R}$ et $d \in \mathbb{N}$. c et d sont des hyper paramètres. Parfois, ce noyau est appelé noyau polynomial.

De plus, il est possible de construire des fonctions noyaux en combinant et/ou en modifiant des fonctions du noyau populaires et connues.

Propreté 1.1 : Soient κ_1 et κ_2 deux fonctions noyau sur $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$. f(.) est une fonction réelle sur $\mathcal{X}$. Alor, les fonctions suivantes sont des fonctions noyau [32, 33] :

1. $\kappa(\boldsymbol{x}, \boldsymbol{z}) = c\kappa_1(\boldsymbol{x}, \boldsymbol{z})$ où $c \in \mathbb{R}^+$.
2. $\kappa(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})\kappa_1(\boldsymbol{x}, \boldsymbol{z})f(\boldsymbol{z})$.
3. $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \exp(\kappa_1(\boldsymbol{x}, \boldsymbol{z}))$.
4. $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \kappa_1(\boldsymbol{x}, \boldsymbol{z}) + \kappa_2(\boldsymbol{x}, \boldsymbol{z})$.
5. $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \kappa_1(\boldsymbol{x}, \boldsymbol{z})\kappa_2(\boldsymbol{x}, \boldsymbol{z})$.

1.4 Régression Ridge à Noyau

Cette section introduit la méthode de régression ridge à noyau (*Kernel Ridge Regression* : KRR) [28, 39] qui est un cas spécial très simple de la régression de la machine à vecteurs de support (*Support Vector Regression : SVR*) [19]. Généralement, la méthode KRR présente une grande ressemblance avec certaines techniques d'approximation, comme les réseaux de neurones, lorsque la fonction gaussienne est adoptée comme un noyau reproduisant. La forme et la solution de la régression ridge à noyau sont obtenues par la résolution du problème d'optimisation suivante [39] :

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{N_d} \left(y_i - f(\boldsymbol{x}_i)\right)^2 + \lambda \|f\|_{\mathcal{F}}^2 \tag{1.6}$$

où $\lambda > 0$, N_d représente la dimension d'ensemble d'apprentissage (nombre de données entrées/sorties) et $f(\boldsymbol{x})$ est donné par l'équation (1.5). pour simplifier le problème d'optimisation défini par équation (1.6), nous définissent les vecteurs $\boldsymbol{y}$, $\boldsymbol{\alpha}$ et la matrice $\boldsymbol{K}$ tels que : $\boldsymbol{y} = \left(y_1, \dots, y_{N_d}\right)^T$, $\boldsymbol{\alpha} = \left(\alpha_1, \dots, \alpha_{N_d}\right)^T$ et $\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \kappa\left(\boldsymbol{x}_1, \boldsymbol{x}_{N_d}\right) \\ \vdots & \dots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & \kappa\left(\boldsymbol{x}_N, \boldsymbol{x}_{N_d}\right) \end{pmatrix}$, respectivement. Alors, le problème (1.6) peut-être exprimé comme suit :

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^T} (\boldsymbol{y} - \boldsymbol{K\alpha})^{\boldsymbol{T}} (\boldsymbol{y} - \boldsymbol{K\alpha}) + \lambda \boldsymbol{\alpha}^{\boldsymbol{T}} \boldsymbol{K\alpha} \tag{1.7}$$

La minimisation du problème quadratique dans l'équation (1.7) est simple et les coefficients d'expansion sont donnés par:

$$F(\boldsymbol{\alpha}) = \|\boldsymbol{K}^{\boldsymbol{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|^2 + \lambda \boldsymbol{\alpha}^{\boldsymbol{T}} \boldsymbol{K\alpha}$$

$$F(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\boldsymbol{T}} \boldsymbol{K}^{\boldsymbol{2}} \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^{\boldsymbol{T}} \boldsymbol{Ky} + \|\boldsymbol{y}\|^2 + \lambda \boldsymbol{\alpha}^{\boldsymbol{T}} \boldsymbol{K\alpha}$$

$$\nabla F(\boldsymbol{\alpha}) = 2\boldsymbol{K}^{\boldsymbol{2}} \boldsymbol{\alpha} - 2\,\boldsymbol{Ky} + 2\lambda \boldsymbol{K\alpha} = \boldsymbol{0}$$

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-\boldsymbol{1}} \boldsymbol{y} \tag{1.8}$$

où la matrice $\boldsymbol{I}$ est une matrice d'identité de dimension $N_d \times N_d$.

1.5 Machine à Vecteurs de Support

Les bases principales des Machines à Vecteurs de Support (*Support Vector Machines : SVM*), aussi connue sur le nome de Séparateurs à Vastes Marges, a été développées par Vapnik [19]. Ces méthodes sont devenues populaires en raison de plusieurs caractéristiques intéressantes et des performances empiriques prometteuses. Les méthodes SVM ont été d'abord développées pour résoudre le problème de classification, mais récemment elles ont été étendues au domaine de régression [40, 41, 42, 43, 44].

Dans la littérature, la terminologie pour SVM peut être un peu confuse. Le terme SVM est généralement utilisé pour décrire la classification avec des méthodes de machine à vecteurs de support et le terme « la régression de la machine à vecteurs de support » (SVR) est utilisé pour décrire les problèmes de régression. Les SVR peuvent également être appliqués à des problèmes de régression par l'introduction d'une fonction de perte qui inclura une mesure de distance. Dans ce manuscrit, l'approche ϵ-insensible (ϵ-*insensitive*) est utilisée comme une fonction de perte [40]. Cette fonction a été introduite comme une approximation de la fonction de perte de Huber [40] qui permet d'obtenir un ensemble épars de vecteurs de support.

1.5.1 La régression linéaire

Considérons le problème de l'approximation de l'ensemble d'apprentissage: $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_{N_d}, y_{N_d})\}, \boldsymbol{x} \in \mathbb{R}^n, y \in \mathbb{R}$, où les approximations sont faites par la fonction linéaire suivante:

$$\boldsymbol{f}(\boldsymbol{x}) = \langle \boldsymbol{\omega}, \boldsymbol{x} \rangle + \mathrm{b} \tag{1.9}$$

La fonction de régression optimale est obtenue en minimisant la quantité:

$$\psi(\boldsymbol{\omega}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{N_d}(\xi_i^- + \xi_i^+) \tag{1.10}$$

où la constante C est une valeur pré-spécifiée, et ξ_i^-, ξ_i^+ sont des variables flexibles représentant les contraintes supérieures et inférieures sur les sorties du système [40]. Dans ce manuscrit, la fonction de perte ϵ-insensible sera utilisée pour l'optimisation. La fonction de perte ϵ-insensible peut être décrite par l'équation (1.11) :

$$L_\varepsilon(y) = \begin{cases} 0 & pour \quad |f(\boldsymbol{x}) - y| < \varepsilon \\ |f(\boldsymbol{x}) - y| - \varepsilon & autrement \end{cases} \tag{1.11}$$

alors que la solution du problème d'optimisation ci-dessus est donnée par :

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \mathcal{W}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = \max_{\alpha,\alpha^*}\left(-\frac{1}{2}\sum_{i=1}^{N_d}\sum_{j=1}^{N_d}(\alpha_i-\alpha_i^*)\left(\alpha_j-\alpha_j^*\right)\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle\right) + \sum_{i=1}^{N_d}\left(\alpha_i(y_i-\varepsilon)-\alpha_i^*(y_i+\varepsilon)\right) \tag{1.12}$$

Ou bien,

$$\bar{\boldsymbol{\alpha}},\bar{\boldsymbol{\alpha}}^* = \operatorname*{Arg\,min}_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}\left(\frac{1}{2}\sum_{i=1}^{N_d}\sum_{j=1}^{N_d}(\alpha_i-\alpha_i^*)\left(\alpha_j-\alpha_j^*\right)\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle\right) - \sum_{i=1}^{N_d}(\alpha_i-\alpha_i^*)\,y_i + \sum_{i=1}^{N_d}(\alpha_i+\alpha_i^*)\,\varepsilon \tag{1.13}$$

En respectant les contraintes :

$$\begin{cases} 0 \le \alpha_i,\alpha_i^* \le C \qquad i=1,\dots,N_d \\ \sum_{j=1}^{N_d}(\alpha_i-\alpha_i^*)=0 \end{cases} \tag{1.14}$$

Résoudre le problème d'optimisation dans l'équation (1.12) avec des contraintes donné par (1.14), se fait à l'aide de l'optimisation quadratique (QP), détermine les multiplicateurs de Lagrange $\boldsymbol{\alpha},\boldsymbol{\alpha}^*$, et la fonction de régression est donnée par l'équation (1.9).

Les résultats des $\boldsymbol{\omega}$ et $\boldsymbol{b}$ sont :

$$\begin{cases} \bar{\boldsymbol{\omega}} = \sum_{j=1}^{N_d}(\alpha_i-\alpha_i^*)\boldsymbol{x}_i \\ \bar{\boldsymbol{b}} = -\frac{1}{2}\langle\bar{\omega},(\boldsymbol{x}_r+\boldsymbol{x}_s)\rangle \end{cases} \tag{1.15}$$

et les conditions de Karush-Kuhn-Tucker (KKT) qui sont satisfaites par la solution sont les suivantes :

$$\alpha_i\alpha_i^* = 0, i=1,\dots,N_d \tag{1.16}$$

où $\boldsymbol{x_r}$ et $\boldsymbol{x_s}$ sont les points des vecteurs de support où exactement l'un des multiplicateurs de Lagrange est supérieur à zéro.

1.5.2 La régression no linéaire

Dans le cas non linéaire, un fonction non linéaire peut être utilisé pour transformer l'ensemble d'apprentissage dans un espace de grande dimension où la régression linéaire est effectuée [40]. L'approche du noyau est à nouveau utilisée pour résoudre le problème de la dimensionnalité où la solution de SVR non linéaire utilisant une fonction de perte ϵ-insensible est donnée parle problème d'optimisation suivant:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \mathcal{W}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \left(-\frac{1}{2}\sum_{i=1}^{N_d}\sum_{j=1}^{N_d}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)\kappa\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right) \right) + \sum_{i=1}^{N_d}\left(\alpha_i(y_i-\varepsilon) - \alpha_i^*(y_i+\varepsilon)\right) \tag{1.17}$$

En respectant les contraintes :

$$\begin{cases} 0 \le \alpha_i, \alpha_i^* \le C \qquad i = 1, \dots, N_d \\ \sum_{i=1}^{N_d}(\alpha_i - \alpha_i^*) = 0 \end{cases} \tag{1.18}$$

Similaire à la solution de la régression linéaire précédente, la fonction de régression est donnée par:

$$f(\boldsymbol{x}) = \sum_{SVs}(\alpha_i - \alpha_i^*)\,\kappa(\boldsymbol{x}_i,\boldsymbol{x}) + b \tag{1.19}$$

où $\boldsymbol{\alpha_i}, \boldsymbol{\alpha_i^*}$ sont la solution au problème QP et $\bar{\boldsymbol{\omega}}$, $\bar{\boldsymbol{b}}$ sont donnés par :

$$\begin{cases} \langle \bar{\omega}, x\rangle = \sum_{i=1}^{N_d}(\alpha_i - \alpha_i^*)\kappa\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right) \\ \bar{b} = -\frac{1}{2}\sum_{i=1}^{N_d}(\alpha_i - \alpha_i^*)\left(\kappa(\boldsymbol{x}_i,\boldsymbol{x}_r) + \kappa(\boldsymbol{x}_i,\boldsymbol{x}_s)\right). \end{cases} \tag{1.20}$$

1.6 Régression par la machines à vecteurs de support en moindres carrés

Suykens et Vandewalle [45] ont proposé une simplification de la méthode SVM basée sur l'approche des moindres carrés dit « machines à vecteurs de support en moindres carrés » (*Least Squares Support Vector Machine : LSSVM*). Dans ce manuscrit, nous sommes intéressés à la version de régression (LSSVR). L'approche LSSVM a été appliquée avec succès dans divers domaines [46]. Le LSSVM a des avantages similaires à ceux de la méthode SVM classique, mais un avantage supplémentaire est qu'il ne nécessite que la résolution d'un ensemble d'équations linéaires, qui est beaucoup plus facile et plus simple en termes de calcul. Généralement, la méthode de régression par les machines à vecteurs de support en moindres carrés (LSSVR) utilise des contraintes d'égalité au lieu des contraintes d'inégalité et adopte le système linéaire des moindres carrés comme sa fonction de perte. Ceci qui rend cette méthode attrayante pour le calcul. Le LSSVR a également une bonne convergence et une grande précision. Par conséquent, cette méthode est plus facile à utiliser par rapport à la programmation quadratique utilisée par la méthode SVR. En outre, les derniers études [46] montrent que le LSSVR est comparable à la SVR classique en termes de performance et de robustesse. L'avantage majeur de la méthode LSSVR est qu'elle est efficace en termes de calcul alors que le LSSVR préserve les propriétés importantes associées à une SVR classique.

En général, dans le cas où un ensemble des données d'apprentissage (N_d données entrées-sorties) est disponible, le théorème de LSSVR suggère que l'espace d'entrée $\mathbb{R}^n$ peut-être transformer vers un nouvel espace de dimension supérieure $\mathcal{H}$ (espace de Hilbert ou l'espace des caractéristiques) avec une fonction de re-description non linéaire présélectionnée $\boldsymbol{\varphi}$ pour obtenir le modèle d'approximation non linéaire. La sortie prédite du système non linéaire est définie comme suit :

$$\boldsymbol{y} = \langle \boldsymbol{\omega}, \boldsymbol{\varphi}(\boldsymbol{x}) \rangle + \boldsymbol{b} = \boldsymbol{\omega}^{\boldsymbol{T}} \boldsymbol{\varphi}(\boldsymbol{x}) + \boldsymbol{b} \tag{1.21}$$

où $\boldsymbol{\omega}, \boldsymbol{\varphi} \in \mathcal{H}$, le biais $b \in \mathbb{R}$ et l'opération $\langle .,. \rangle$ représentes le produit scalaire dans $\mathcal{H}$. Les paramètres inconnus $\boldsymbol{\omega}$ et b peuvent être identifiés à l'aide des données d'apprentissage.

Dans le cas de l'approche LSSVR, le problème d'optimisation est défini par :

$$\min_{\boldsymbol{\omega},\boldsymbol{e}} J(\boldsymbol{\omega},\boldsymbol{e}) = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + \frac{\gamma}{2}\sum_{i=1}^{N_d} e_i^2, \qquad \gamma > 0$$
$$\text{Respectant: } y_i = \omega^T\varphi(\boldsymbol{x}_i) + b + \ e_i, \ \ i = 1, \dots, N_d \tag{1.22}$$

où e_i représente l'erreur entre la sortie réelle et la sortie prédictive. Le problème définit dans (1.22) peut être résolu en construisant le Lagrangien et en dérivant le problème dual. Le Lagrangien pour ce problème est défini par [46] :

$$\mathcal{L}(\boldsymbol{\omega},\boldsymbol{e},\boldsymbol{\alpha},b) = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + \frac{\gamma}{2}\sum_{i=1}^{N_d} e_i^2 - \sum_{i=1}^{N_d}\alpha_i(\boldsymbol{\omega}^T\varphi(\boldsymbol{x}_i) + b + \ e_i - y_i) \tag{1.23}$$

où $\boldsymbol{\alpha}_i, i = 1,..,N_d$ sont les multiplicateurs de Lagrange. Les conditions d'optimalité sont données par:

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial\boldsymbol{\omega}} = 0 \rightarrow \ \omega = \displaystyle\sum_{i=1}^{N_d}\alpha_i\varphi(\boldsymbol{x}_i) \\ \dfrac{\partial\mathcal{L}}{\partial b} = 0 \ \rightarrow \ \displaystyle\sum_{j=1}^{N_d}\alpha_i = 0 \\ \dfrac{\partial\mathcal{L}}{\partial e_i} = 0 \ \rightarrow \ \alpha_i = \gamma e_i, \qquad\qquad i = 1,..,N_d \\ \dfrac{\partial\mathcal{L}}{\partial\alpha_i} = 0 \ \rightarrow \ \omega^T\varphi(\boldsymbol{x}_i) + b + \ e_i - y_i = 0, \qquad i = 1,..,N_d. \end{cases} \tag{1.24}$$

Après l'élimination des variables $\boldsymbol{\omega}$ et $e_i, i = 1,..,N_d$ on obtient la solution suivante :

$$\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{1}^T\\ \mathbf{1} & \boldsymbol{K} + \gamma^{-1}\boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{0}\\ \boldsymbol{Y}\end{bmatrix} \tag{1.25}$$

où $\mathbf{1} = (1,\dots,1)^T$, $\boldsymbol{Y} = (y_1,\dots,y_{N_d})^T$, $\boldsymbol{I}$ est une matrice d'identité de dimension $N_d \times N_d$ et $\boldsymbol{K}$ est une matrice de dimension $N_d \times N_d$ tel que $K_{ij} = \boldsymbol{\varphi}(\boldsymbol{x}_i)^T\boldsymbol{\varphi}(\boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j), i.j = 1,\dots,N_d$ ou $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$ représente une fonction noyau qui satisfait le théorème de Mercer. Alors, la forme du modèle LSSVR est donnée par :

$$\hat{y}(k) = \sum_{i=1}^{N_d}\alpha_i\kappa(\boldsymbol{x},\boldsymbol{x}_i) + b \tag{1.26}$$

où $\alpha_i, i = 1, \dots, N_d$ et b sont la solution au système linéaire (1.25).

1.7 Conclusion

Dans ce chapitre, plusieurs concepts clés dans les méthodes à noyaux ont été brièvement définis. En particulier, nous avons présenté trois méthodes à noyaux différentes qui peuvent être utilisées dans l'apprentissage supervisé: la régression ridge à noyau (KRR), machine à vecteurs de support (SVM) et la régression par les machines à vecteurs de support en moindres carrés (LSSVR). Dans le chapitre suivant, nous discutons les systèmes d'inférences floues et plusieurs stratégies qui peuvent être adoptées pour utiliser les systèmes flous dans l'identification du système.

Chapitre 2

Modélisation Floue

2.1 Introduction

La modélisation analytique des systèmes est un outil très important dans plusieurs disciplines des sciences et de l'ingénierie. Généralement, la modélisation analytique est basée sur la compréhension des éléments physiques du système où les équations de ces éléments qui régissent le comportement dynamique du système sont utilisées pour construire un modèle mathématique.

D'une manière générale, la modélisation analytique offre une description précise facilitant la compréhension du comportement dynamique de procédé. Malheureusement, elle est souvent complexe et peu utilisée pour la commande et dans certaines situations la modélisation du système ne peut être obtenue puisque la réponse dynamique du système est inconnue ou partiellement connue. Parfois, les paramètres des systèmes varient avec le temps, ce qui constitue une grande restriction au niveau pratique. Généralement, le développement des modèles est extrêmement coûteux et la complexité de modélisation des interactions entre les différents mécanismes du système est très élevée. En outre, les difficultés de calcul associées à la résolution d'un système d'équations différentielles peuvent prévenir une modélisation analytique simple (l'explosion des temps de calcul pour une simulation de Monte-Carlo).

De l'autre côté, les données d'entrées-sorties des systèmes constituent souvent la plus forte et la plus sûre source d'information pour comprendre les comportements dynamiques de procédé où les méthodes de modélisation guidée par les données ont la capacité d'apprendre le comportement dynamique du système, et capturer les relations subtiles entre données, même si ces relations sont inconnues ou difficiles à décrire.

Dans cette thèse, les modèles flous de type Takagi-Sugeno [13], aussi connus sur le nom des modèles dynamiques flous, sont adoptés pour construire les modèles des systèmes à commander.

Les systèmes flous de type Takagi-Sugeno largement utilisés pour représenter les procédés lorsque les comportements de ces procédés ne sont pas faciles à établir. L'avantage principal d'utiliser les systèmes flous de type Takagi-Sugeno est que seules les données appropriées du comportement du procédé sont suffisantes pour la représentation du système.

Ce chapitre est consacré à la description des éléments de base de la théorie des systèmes flous. Les concepts présentés ici constituent la plateforme pour les différents travaux exposés dans cette thèse.

2.2 Ensembles flous

La notion des sous-ensembles flous [6, 7] introduit un caractère partiel de l'appartenance d'un élément à un ensemble donné. Cela permet une meilleure représentation des termes et des connaissances vagues que nous, les humains, manipulons au quotidien.

Un sous-ensemble flou $\mathcal{A}$ d'un univers de discours U, est généralement caractérisé par une fonction d'appartenance $\mu_{\mathcal{A}}$ à valeur dans l'intervalle $[0, 1]$ et qui associe à chaque élément $x \in U$ un degré d'appartenance $\mu_{\mathcal{A}}(x)$ indiquant le degré d'appartenance de l'élément x au sous-ensemble $\mathcal{A}$. Le sous-ensemble flou $\mathcal{A}$ de U est défini par [47, 48] :

$$\mathcal{A} = \{(x, \mu_{\mathcal{A}}(x)), x \in U, 0 \leq \mu_{\mathcal{A}}(x) \leq 1\} \tag{2.1}$$

Le concept d'ensembles flous est souvent critique pour modéliser les expressions linguistiques couramment utilisés comme par exemple: « la voiture est rapide ». Une variable linguistique peut être représentée par un triplet $(x, \mathcal{T}(x), U)$ dans lequel x est le nom de la variable linguistique, $\mathcal{T}(x)$ l'ensemble des valeurs linguistiques de x et U l'univers de discours.

L'exemple illustratif dans figure (2.1) [49] montre un variable linguistique associée au concept de température [49], représentée par les sous-ensembles flous où les termes linguistiques sont définis par : {*froide*, *moyenne*, *chaude*} sur l'univers de discours représenté par les températures comprises dans l'intervalle [0°,80°].

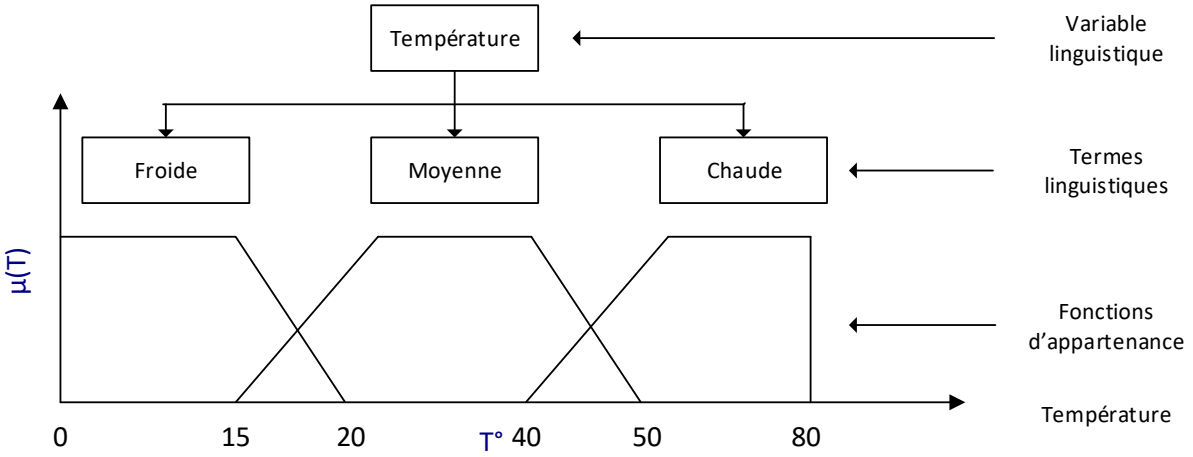

Figure 2.1 : Exemple d'ensembles flous pour la variable température [49]

2.3 Les opérations sur les sous-ensembles flous

Soient $\mathcal{A}$ et $\mathcal{B}$ deux sous-ensembles flous dans univers de discours U ayant respectivement $\mu_{\mathcal{A}}$ et $\mu_{\mathcal{B}}$ comme fonctions d'appartenance. Les opérations : Egalité, réunion, intersection, complémentation et l'inclusion des sous-ensembles flous sont définis par leur fonction d'appartenance [47, 49].

a) ***Union*** : L'union de sous-ensembles $\mathcal{A}$ et $\mathcal{B}$, que l'on note $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$, est le sous-ensemble $\mathcal{C}$ constitué des éléments de univers de discours U affectés du plus grand des deux degrés d'appartenance $\mu_{\mathcal{A}}$ et $\mu_{\mathcal{B}}$:

$$\forall\, x \in U, \mu_{\mathcal{C}}(x) = \mu_{\mathcal{A}\cup\mathcal{B}}(x) = \max(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)) \tag{2.2}$$

b) ***Intersection*** : L'intersection de sous-ensembles $\mathcal{A}$ et $\mathcal{B}$, que l'on note $\mathcal{C} = \mathcal{A} \cap \mathcal{B}$, est le sous-ensemble flou $\mathcal{C}$ constitué des éléments de univers de discours U affectés du plus petit des deux degrés d'appartenance $\mu_{\mathcal{A}}$ et $\mu_{\mathcal{B}}$:

$$\forall\, x \in U, \mu_{\mathcal{A}\cap\mathcal{B}}(x) = \min(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)) \tag{2.3}$$

c) ***Egalite*** : $\boldsymbol{\mathcal{A}}$ et $\boldsymbol{\mathcal{B}}$ sont dits égaux, propriété que l'on note $\boldsymbol{\mathcal{A} = \mathcal{B}}$, si leurs fonctions d'appartenance prennent la même valeur en tout point de univers de discours $\boldsymbol{U}$:

$$\forall\, x \in U, \mu_{\mathcal{A}}(x) = \mu_{\mathcal{B}}(x) \tag{2.4}$$

d) ***Inclusion*** : le sous-ensemble $\mathcal{A}$ est dit inclus dans le sous-ensemble $\mathcal{B}$, notée par $\mathcal{A} \subseteq \mathcal{B}$, si tout élément $x \in U$ qui appartient à $\mathcal{A}$ appartient aussi à $\mathcal{B}$:

$$\forall\, x \in U, \mu_{\mathcal{A}}(x) \leq \mu_{\mathcal{B}}(x) \tag{2.5}$$

e) ***Complément*** : Le complément de sous-ensemble $\mathcal{A}$, notée par $\bar{\mathcal{A}}$, est l'ensemble flou de univers de discours U constitué des éléments x lui appartenant d'autant plus qu'ils appartiennent peu à $\mathcal{A}$:

$$\forall\, x \in U, \mu_{\bar{\mathcal{A}}}(x) = 1 - \mu_{\mathcal{A}}(x) \tag{2.6}$$

2.4 Règles floues

Une règle floue de type : $(R : Si\ x\ est\ \mathcal{A}\ Alors\ y\ est\ \mathcal{B})$ est une relation entre deux propositions floues ayant chacune un rôle particulier [49]. La première partie de la règle $(x\ est\ \mathcal{A})$ est appelée prémisse de la règle alors que la dernière $(y\ est\ \mathcal{B})$ est la conclusion. Dans le cas de propositions floues élémentaires (proposition définie par une seule variable linguistique), la prémisse et la conclusion sont définies à partir de deux variables linguistiques $\mathcal{A}$ et $\mathcal{B}$, qui sont a priori indépendantes, décrivant les connaissances relatives aux univers de discours $U_{\mathcal{A}}$ et $U_{\mathcal{B}}$ de manière à prendre en compte l'imprécision relative aux modalités de $\mathcal{A}$ et $\mathcal{B}$. Une proposition floue élémentaire est généralement insuffisante pour représenter l'ensemble des informations à manipuler. Dans ce cas, plusieurs propositions floues peuvent alors être combinées pour obtenir une proposition floue générale afin d'améliorer et détailler la représentation. La relation floue $\mathcal{R}$ entre la prémisse et la conclusion de la règle floue est déterminée par une implication floue dont le degré de vérité est défini par une fonction d'appartenance $\mu_{\mathcal{R}}(x, y)$ qui dépend du degré de vérité $\mu_{\mathcal{A}}(x)$ et $\mu_{\mathcal{B}}(y)$ de chacune des deux propositions élémentaires. L'implication floue est également notée par :$\mathcal{A} \Longrightarrow \mathcal{B}$ où les

implications les plus courantes permettant la détermination de la fonction d'appartenance résultante décrivant la proposition floue $\mathcal{R}$ sont donnés par :

$$\begin{cases} \text{l'implication de Mamdani:} & \mu_{\mathcal{R}}(x,y) = \min(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)) \\ \text{l'implication de Larsen:} & \mu_{\mathcal{R}}(x,y) = \mu_{\mathcal{A}}(x) \times \mu_{\mathcal{B}}(y) \\ x \in \mathcal{A}, y \in \mathcal{B} & \end{cases} \tag{2.7}$$

2.5 Fonction d'appartenance

Soit un ensemble $\mathcal{F}$ et un sous-ensemble $\mathcal{A} \subset \mathcal{F}$, et x un élément de $\mathcal{F}$ appartenant à $\mathcal{A}$ ($x \in \mathcal{A}$). Pour illustrer cette caractéristique, on utilise la fonction d'appartenance $\mu_{\mathcal{A}}(x) \in [0,1]$ qui représente le degré d'appartenance de x à l'ensemble flou $\mathcal{A}$. Le plus souvent, la fonction d'appartenance est déterminée par l'une des fonctions suivantes (figure 2.2) [47, 48, 49] :

1) ***La fonction Triangulaire*** : elle est caractérise par trois paramètres (a, b, c), les sommets du triangle :

$$\mu_{\mathcal{A}}(x) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{2.8}$$

2) ***La fonction Trapézoïdale*** : définie par quatre paramètres (a, b, c, d) :

$$\mu_{\mathcal{A}}(x) = max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{2.9}$$

3) ***La fonction Gaussienne*** : définie par deux éléments (m, σ) où m et σ sont respectivement le centre et l'épaisseur (variance) de la fonction Gaussienne :

$$\mu_{\mathcal{A}}(x) = exp\left(-\frac{(x-m)^2}{\sigma^2}\right) \tag{2.10}$$

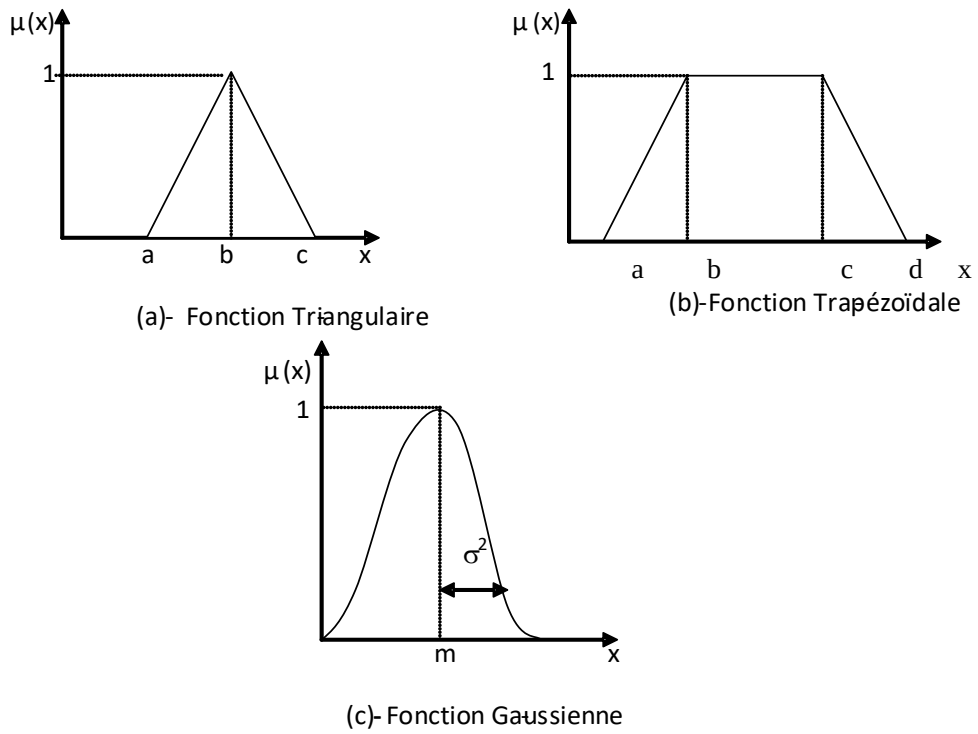

Figure 2.2 : les formes populaires des fonctions d'appartenance

2.6 Structure d'un système flou

La structure générale d'un système flou se compose de quatre éléments de base : La fuzzification, la base des connaissances, le moteur d'inférence flou et la defuzzification comme c'est montre dans la figure (2.3) [50, 51, 54].

2.6.1 La fuzzification

La fuzzification est une interface numérique-flou où la fuzzification d'une valeur précise d'une variable d'entrée x consiste à caractériser le degré auquel cette mesure appartient à un sous-ensemble flou donné, c'est-à-dire transformer les données numériques en données linguistiques.

2.6.2 La base des connaissances

Elle composée par la base de données et la base de règles :

- Base de données : qui contient le type et les caractéristiques des fonctions d'appartenances, univers de discours, nombre des variables linguistiques, et le nombre d'ensemble flous associes à chaque variable linguistique.

- Base de règles : c'est une collection de règles floues de type : *SI* (l'ensemble des conditions sont satisfaites) *ALORS* (un ensemble de conséquences peuvent être inférées) où ces collections de règles floues permettent de tirer des conclusions.

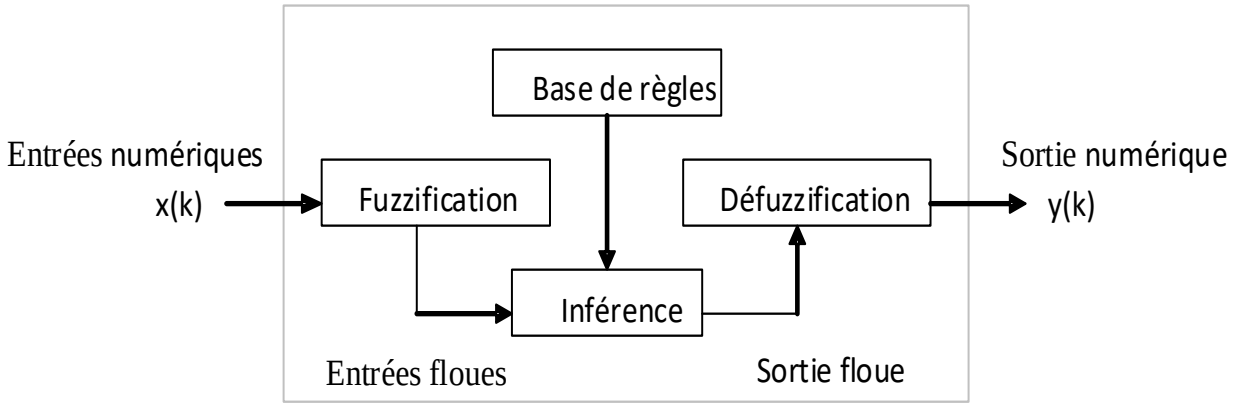

Figure 2.3 : Structure d'un système flou [54]

2.6.3 Moteur d'inférence (la décision)

C'est le mécanisme qui interprète une règle floue *SI-ALORS*. On distingue deux types d'opérateurs, l'opérateur de conjonction *ET* qui lie les différentes variables de prémisse, l'opérateur de disjonction *OU* qui lie des règles formant la base de connaissance. Parmi les mécanismes d'inférences les plus utilises on distingue :

- Mécanisme d'inférence ***Max-Min*** : Où l'opérateur *ET* est réalisé par la formation du minimum, l'opérateur *OU* est réalisé par la formation du maximum, et l'implication (*ALORS*) est réalisée par la formation du minimum.
- Mécanisme d'inférence ***Max-Pro*** : Dans ce mécanisme, l'opérateur *ET* est réalisé par la formation du produit, l'opérateur *OU* est réalisé par la formation du maximum, et finalement l'implication (*ALORS*) est réalisée par la formation du produit.
- Mécanisme d'inférence ***Som-Pro*** : L'opérateur *OU* est réalisé par la formation de la somme (valeur moyenne), et l'opérateur *ET* par la formation du produit. Pour la conclusion, l'opérateur *ALORS* est réalisé par un produit.

2.6.4 Defuzzification

C'est l'interface flou-numérique, transforme les données linguistiques en données numériques. Il existe plusieurs stratégies de defuzzification :

- Technique du maximum : c'est la technique la plus simple. Dans cette technique, seule la règle qui présentant le maximum de validité, pour chaque sortie, est considérée, alors que les règles secondaires qui peuvent néanmoins être importantes pour le fonctionnement et la stabilité du système sont ignorées. Elle est par conséquent peu employée.
- Technique de la moyenne des maximums : elle considère, comme valeur de sortie, la moyenne de toutes les valeurs pour lesquelles la fonction d'appartenance issues de l'inférence est maximale.
- Technique du centre de gravité : plus performante, elle consiste à tracer, sur un même diagramme représentant les ensembles flous de sortie, les différentes zones correspondantes à chacune des règles et à calculer le centre de gravité de la zone consolidée. La méthode de defuzzification la plus mentionnée dans la littérature est la defuzzification par la méthode de centre de gravité.

2.7 Les modèles flous

Le modèle de raisonnement flou est régulièrement formé d'un ensemble de règles qui décrit le comportement du système où chaque règle décrit une partie du comportement du système alors que les règles auront la structure suivante :

Si (antécédent, prémisse, condition) $\longrightarrow$ (Conséquences)

Par conséquent, la modélisation floue est déduite d'un raisonnement élaboré des variables du système et d'une liste de règles floues décrivant la manière selon laquelle le modèle doit fonctionner pour réaliser les performances désirées. Plusieurs modèles de raisonnement ont été développées, les plus courants sont le modèle de Mamdani [53] et le modèle de Takagi-Sugeno [13].

2.7.1 Le modèle de Mamdani

Un modèle floue de Mamdani ne traite pas une relation mathématique bien défini mais utilise des inférences floues avec plusieurs règles en se basant sur des variables linguistiques. Les inférences utilisées dans le modèle de Mamdani sont de la forme :

Si (condition, antécédent ou prémisses) Alors (actions, conclusion ou conséquences)

L'antécédent et la conséquence (conclusion) sont des opérations floue dans lesquelles on trouve des variables linguistiques liées par des opérateurs de la logique floue, ainsi l'ensemble des opérations affectées par l'opérateur de la logique floue, porte non pas sur les valeurs réelles mais sur les valeurs qualificatives exprimées par des termes linguistiques ou par un code approprié. Dans le cas général :

Si $(x_1$est $\mathcal{A}_1)$ et $(x_2$ est $\mathcal{A}_2)$ … et $(x_n$ est $\mathcal{A}_n)$ Alors y est B

où $\mathcal{A}_j, j = 1, \dots, n$ sont les sous-ensembles flous, $x_j, j = 1, \dots, n$ sont les variables de antécédents et B représentés la sortie.

2.7.2 Le modèle de Takagi-Sugeno

Le modèle de raisonnement de Takagi-Sugeno (TS) est défini par un ensemble de règles qui sont constituées d'antécédent symbolique (prémisses) et de conséquences ou conclusion sous forme d'équations linéaire, les deux premières étapes la fuzzification des entrées et l'inférence sont présentées comme dans le modèle de Mamdani mais la defuzzification est remplacée par une procédure qui calcule directement la sortie de chaque règle floue, en fonction des états du système. L'action ou bien la sortie globale revient à une somme pondéré de toutes les sorties de chaque règle floue. Les règles sont du type :

$$R^i\text{: Si } (x_1\text{est } \mathcal{A}_1, x_2\text{est } \mathcal{A}_2, \dots, x_n \text{ est } \mathcal{A}_n) \text{ Alors } y^i = a_0^i + a_1^i x_1 + \cdots + a_n^i x_n \tag{2.11}$$

avec $i = 1 \dots N$, N le nombre des règles, y^i est la variable du conséquent, qui représente la sortie du système flou pour chaque règle, $x_j, j = 1, \dots, n$ est la variable de l'antécédent, qui représente l'entrée du système, $\mathcal{A}_j, j = 1, \dots, n$ est les sous-ensembles flous et $a_j^i, i = 1 \dots N, j = 1 \dots n$ est

les paramètres constants de la conséquence. La valeur finale de la sortie du modèle de Takagi-Sugeno est calculée par :

$$y = \frac{\sum_{i=1}^{N} F^i(x)\, y^i}{\sum_{i=1}^{N} F^i(x)} = \frac{\sum_{i=1}^{N} F^i(x)\left(a_0^i + a_1^i x_1 + \cdots + a_n^i x_n\right)}{\sum_{i=1}^{N} F^i(x)} \tag{2.12}$$

où $F^i(x)$ représente le degré d'accomplissement de l'antécédent qui est donnée par :

$$F^i(x) = \mu_{\mathcal{A}_1}(x) \wedge \mu_{\mathcal{A}_2}(x) \ldots \wedge \mu_{\mathcal{A}_n}(x) \tag{2.13}$$

et $\mu_{\mathcal{A}_i}(x), i = 1, \ldots, n$ sont les fonctions d'appartenance et $\wedge$ représente l'opérateur minimum (ou le l'opérateur produit).

2.8 Identification du modèle TS

Généralement, la construction de modèles flous peut se faire à partir de la modèle non linéaire mathématique des systèmes ou les données (mesures) des systèmes. Dans le cas où un modèle de système non linéaire est disponible, on peut obtenir son modèle approximatif flou de type TS en linéarisant le modèle non linéaire donné autour d'un certain nombre de points des fonctionnements d'intérêt.

Lorsque le modèle de système non linéaire n'est pas disponible où les seules informations disponibles sur le système sont les variables mesurables, l'approche usuelle consiste à apprendre le comportement du système à l'aide de l'historique des données (données d'apprentissage) [54]. Les données sont disponibles sous forme d'enregistrements de l'opération du procédé ou bien il est possible de réaliser des expériences d'identification afin d'obtenir les données appropriées du comportement du système. Dans ce cas, on peut obtenir un modèle approximatif flou de type TS à partir de ce données.

La construction de modèles TS flous à partir de données implique des méthodes basées sur une base des connaissances mais aussi des idées issues du domaine des réseaux de neurones, l'identification des systèmes conventionnels et l'analyse des données. L'utilisation des techniques et d'algorithmes pour la construction de modèles flous guidés par les données entrées-sorties est habituellement appelée «identification floue».

2.8.1 Identification du modèle TS à partir des systèmes non linéaires

Supposons que nous avons un modèle mathématique de système non linéaire qui est donné par :

$$x(t+1) = f\big(x(t), u(t)\big) \tag{2.14}$$

L'objectif est de trouver un modèle flou approximative basé sur ce modèle non linéaire. On suppose que le nombre de points de fonctionnements d'intérêt est donné a priori. Alors les points de fonctionnements sont donnés par : $\{(x^1, u_1),\ (x^2, u_2),\ \dots, (x^N, u_N)\}$ où le premier point de fonctionnement (x^1, u_1) est le point d'équilibre du système, qui est obtenu en résolvant $f\big(x(t), u(t)\big) = 0$. Ensuite, pour obtenir les modèles linéaires, on utilise le l'expansion de série du Taylor au premier ordre de fonction non linéaire $f\big(x(t), u(t)\big)$ autour des points de fonctionnement. Donc, l'approximation de system non linéaire $f\big(x(t), u(t)\big)$ autour le point de fonctionnement $\big(x^j, u_j\big)$ sont données par :

$$f\big(x(t), u(t)\big) \simeq \big(x - x^j\big)\frac{\partial f}{\partial x}\Big|_{(x=x^j, u=u^j)} + \big(u - u^j\big)\frac{\partial f}{\partial u}\Big|_{(x=x^j, u=u^j)} + \varepsilon_j \tag{2.15}$$

où ε_j représentés les erreurs d'approximation d'ordre supérieur. Le système linéaire autour un point de fonctionnement $\big(x^j, u_j\big)$ est alors donné par :

$$\begin{cases} x(t+1) \simeq A_j x(t) + B_j u(t) + a_j, & A_j = \dfrac{\partial f}{\partial x}\Big|_{(x=x^j, u=u^j)} \\ B_j = \dfrac{\partial f}{\partial u}\Big|_{(x=x^j, u=u^j)} & a_j = -x^j \dfrac{\partial f}{\partial x}\Big|_{(x=x^j, u=u^j)} - u^j \dfrac{\partial f}{\partial u}\Big|_{(x=x^j, u=u^j)} \end{cases} \tag{2.16}$$

Généralement, les fonctions d'appartenance telles que les fonctions triangulaires, trapézoïdales et gaussiennes, sont utilisés pour compléter la construction du modèle flou ou les centres de ces fonctions d'appartenance peuvent être déterminés par les points des fonctionnements $\{(x^1, u_1),\ (x^2, u_2),\ \dots, (x^N, u_N)\}$, tandis que le reste des paramètres peut être choisi par le concepteur pour réduire l'espace des paramètres à optimiser.

2.8.2 Identification du modèle TS à partir des données

Dans les applications d'ingénierie, les modèles mathématiques des systèmes complexes sont très difficiles, parfois impossibles, à obtenir. Dans ce cas, nous devrons choisir des approches basées sur les données expérimentales pour développer un modèle flou de type TS ce qui peut remplacer le système d'origine afin de calculer le signal de commande. Les modélisations floues à partir de données sont des outils efficaces pour approximer les systèmes non linéaires [54]. Deux approches principales pour l'intégration de la connaissance et des données dans un modèle flou peuvent être distinguées:

- La connaissance experte qui est généralement exprimée sous une forme verbale est traduite dans une collection de règles du type "Si-Alors". De cette façon, une structure floue du modèle est génère où les paramètres de cette structure (paramètres des conséquences, fonctions d'appartenance, …) peuvent être définis précisément à l'aide des données entrée-sortie. Les algorithmes particuliers d'ajustement exploitent le fait qu'au niveau du calcul, le modèle flou peut être vu comme une structure par couches, similaire aux réseaux artificiels de neurones, pour laquelle des algorithmes standard d'apprentissage peuvent être appliqués.
- L'approche où aucune connaissance antérieure n'est initialement utilisée pour définir les règles du modèle flou, et le modèle flou est générée à partir des données. Ensuite, la performance des règles extraites sera vérifiée si elles fournissent un comportement dynamique précis du système. Un expert peut toujours confronter cette information avec ses propres connaissances, peut modifier des règles ou fournir des règles différentes, et peut concevoir des expériences supplémentaires afin d'obtenir plus de données informatives.

Ces techniques peuvent, évidemment, être combinées selon l'application particulière. Aussi, il faut que ce modèle doit maintenir un minimum de transparence afin qu'un chercheur puisse intervenir pour modifier les paramètres.

Dans ce qui suit, nous présentons quelques méthodes populaires utilisées pour définir les modèles flous, et pour chaque méthode nous décrivons les étapes pour sélectionner la structure d'un modèle flou et pour déterminer les paramètres à partir des données d'entrée-sortie. Parmi les modèles approximatifs largement utilisés pour construire des modèles flous

est les méthodes de partitionnement combiné aux techniques des moindres carrés. Avant de considérer ces méthodes, on va d'abord discuter des méthodes basées sur la méthode des moindres carrées seul. Dans les méthodes basées sur l'algorithme de moindres carrées, on fixe généralement par avance le type et le nombre des fonctions d'appartenance et on calcule leurs positions et la valeur des conséquences.

Dans les sous-sections suivantes, nous présenterons les méthodes dit mosaïque, les méthodes basées sur le moindres carrées et les méthodes de partitionnement combiné aux techniques des moindres carrés.

2.8.2.1 Mosaïque (*table lookup*)

Cette méthode nécessite la présence d'un expert (ou l'utilisateur) qui doit définir les fonctions d'appartenance, c'est à dire choisir le nombre et les formes des fonctions d'appartenance dans lequel une bonne distribution est réalisée pour couvrir l'espace des entrées. Notons, que dans cette méthode, les règles sont générées de façon à assurer toutes les combinaisons possibles [52]. Les paramètres non-linéaires sont fixés par le choix de la répartition des fonctions d'appartenances. Par contre, les paramètres linéaires ne sont pas ajustés manuellement. Pour les ajuster, on peut utiliser l'algorithme des Moindres Carrés (MC).

2.8.2.2 L'utilisation de l'algorithme des moindres carrées

Dans cette méthode, il est nécessaire de choisir la forme et nombre de fonctions d'appartenance. L'opérateur $\wedge$ dans l'équation (2.13) est fixé pour être le "produit" car une expression analytique est nécessaire pour optimiser une fonction de coût. Aussi, il est toujours important de choisir les variables qui doivent être utilisées comme entrées du modèle. Pour le modèle sous la forme non linéaire autorégressive avec entrée exogène (NARX) il faut simplement définir le nombre de retards, d'entrée et de sortie, nu et ny. Généralement, la connaissance sur le comportement du système permet de faciliter ces choix. En outre, les valeurs initiales des paramètres des fonctions d'appartenance (les paramètres des antécédentes) doivent être sélectionnées par un expert (ou l'utilisateur). D'autre part, les paramètres des conséquences peuvent être obtenus par la technique des moindres carrés. Cette approche conduit à une formulation de N_d problèmes indépendants de type moindres carrés pondérés dans laquelle le i-

ème sous modèle linéaire local est multiplié par le degré d'appartenance $\omega^i(x) = \frac{\mu^i(x)}{\sum_{i=1}^{N} \mu^i(x)}$, où $y_i = \omega^i(x^i)\left(a_0^i + a_1^i x_1^i + \cdots + a_n^i x_n^i\right), 1 \leq i \leq N_d$.

Les N_d valeurs de sortie peuvent être représentées comme un vecteur $\boldsymbol{Y}$ où :

$$\boldsymbol{Y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\boldsymbol{Y}} = \underbrace{\begin{bmatrix} \omega^1 x_1^1 & \cdots & \omega^1 x_n^1 & \omega^1 & \cdots & \omega^1 x_1^{N_d} & \cdots & \omega^1 x_n^{N_d} & \omega^1 \\ \omega^2 x_1^1 & \cdots & \omega^1 x_n^1 & \omega^2 & \cdots & \omega^2 x_1^{N_d} & \cdots & \omega^1 x_n^{N_d} & \omega^1 \\ \vdots & \cdots & \vdots & \cdots & \cdots & \vdots & \cdots & \cdots & \vdots \\ \omega^N x_1^1 & \cdots & \omega^1 x_n^1 & \omega^{N_d} & \cdots & \omega^N x_1^{N_d} & \cdots & \omega^1 x_n^{N_d} & \omega^1 \end{bmatrix}}_{\boldsymbol{W}} \underbrace{\begin{bmatrix} a_1^1 \\ a_2^1 \\ \vdots \\ a_n^1 \\ a_0^1 \\ \vdots \\ a_1^N \\ a_2^N \\ \vdots \\ a_n^N \\ a_0^N \end{bmatrix}}_{\boldsymbol{\theta}} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}}_{\boldsymbol{E}} \tag{2.17}$$

Le vecteur $\boldsymbol{E}$ est l'erreur d'approximation et l'objectif est de réduire la norme de ce vecteur autant que possible. En utilisant la norme quadratique pour l'optimisation, nous obtenons fonction de coût :

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{E}\|_2 = \min_{\boldsymbol{\theta}} \|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{\theta}\|_2 \tag{2.18}$$

Alors la solution de problèmes présentée par l'équation (2.18) est donnée finalement par l'expression :

$$\boldsymbol{\theta} = (\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{Y}^T\boldsymbol{W} \tag{2.19}$$

Notez que cette solution est applicable uniquement lorsque le $rang\,(\boldsymbol{W}^T\boldsymbol{W}) = dim\,(\boldsymbol{\theta})$; sinon il faut choisir d'autres méthodes pour garantir un ensemble de conséquences fiables pour les règles flous.

2.8.2.3 L'utilisation des méthodes de partitionnement et moindres carrés récursifs

Généralement, une méthode d'identification basée sur les méthodes de partitionnement et moindres carrés récursifs est différente des méthodes présentées ci-dessus. Cette approche permet

de spécifier la structure et de générer automatiquement l'espace de partition des ensembles flous afin d'ajuster les paramètres de base (les paramètres des antécédents). Par suite, les paramètres des conséquences, qui sont considérés comme des paramètres linéaires, peuvent être ajustés à l'aide de moindres carrés récursifs. L'avantage principale de ces méthodes est que l'intervention humaine est réduite à choisir uniquement le nombre de règles floues (parfois, le nombre de règles flous peut également être identifié automatiquement si un ensemble de données est disponible).

L'idée principale des méthodes de partitionnement est de séparer la base de données en un nombre déterminé de classes (*clusters)* ou groupes, et d'associer à chaque règle floue du modèle TS un groupe. Chaque groupe a un centre $\boldsymbol{m}^i = \left(m_1^i, m_2^i, \dots, m_n^i\right), i = 1, \dots, N$, et un variance $\boldsymbol{\sigma}^i = \left(\sigma_1^i, \sigma_2^i, \dots, \sigma_n^i\right), i = 1, \dots, N$ (ce dernier représente la distribution des éléments dans leur groupe). Les centres et les variances (rayons) des groupes sont ceux des fonctions d'appartenance du modèle TS, et le nombre de règles flou du modèle TS est égale au nombre de groupes.

Plusieurs méthodes de partitionnement ont été utilisées pour construire des modèles TS flous tels que: K-moyennes (voir l'annexe B), C-Moyennes Flous (voir chapitre 5), Gustafson-Kessel (GK) [54], etc. Dans ce travail, une méthode de partitionnement basée sur les méthodes de *Particle Swarm Optimization* (PSO) et K-moyennes est utilisée dans le chapitre suivant, tandis que la méthode de C-Moyennes Floues sera utilisée au chapitre 5 pour construire des modèles flous de type TS. La méthode de partitionnement et moindres carrés récursifs est résumée en l'algorithme (2.1), et l'algorithme des moindres carrés récursifs (MCR) qui est utilisé à l'étape 5 de l'algorithme est présenté par des équations :

$$\begin{cases} \widehat{\boldsymbol{\theta}}(t) = \widehat{\boldsymbol{\theta}}(t-1) + \boldsymbol{K}_t\left(y(t) - \boldsymbol{x}(t)^T\widehat{\boldsymbol{\theta}}(t-1)\right) \\ \boldsymbol{K}_t = \dfrac{\boldsymbol{P}_{t-1}\boldsymbol{x}(t)^T}{(\lambda\boldsymbol{I} + \boldsymbol{x}(t)\boldsymbol{P}_{t-1}\boldsymbol{x}(t)^T)} \\ \boldsymbol{P}_t = (\boldsymbol{P}_{t-1} - \boldsymbol{K}_t\boldsymbol{x}(t)\boldsymbol{P}_{t-1})\dfrac{1}{\lambda} \end{cases} \tag{2.20}$$

avec $\widehat{\boldsymbol{\theta}}(t)$ sont les nouveaux paramètres estimés à l'instant t et λ est un facteur d'oubli où $0 < \lambda \leq 1$. L'avantage principal de l'approche de moindres carrés récursifs par rapport aux autres méthodes d'apprentissage réside dans la formulation des termes d'ajustement. Ces paramètres sont ajustés en fonction de l'erreur mesurée à chaque instant d'échantillonnage où $\boldsymbol{K}_t$ et $\boldsymbol{P}_t$ ont été évalués à l'instant précédent. Maintenant, $\boldsymbol{K}_t$ et $\boldsymbol{P}_t$ sont pratiquement ajustée à chaque itération

et ne sont pas recalculés directement, de façon à ce que l'information des instants précédents est préservée.

Étapes hors ligne

1) Compte tenu de l'ensemble de données $\boldsymbol{Z}^t = \{(\boldsymbol{x}^t)^T, \boldsymbol{y}^t\}, t = 1, \dots, N_d, \boldsymbol{x}^t \in \mathbb{R}^n, \boldsymbol{y}^t \in \mathbb{R}$ où $\boldsymbol{x}^t$ et $\boldsymbol{y}^t$ sont, respectivement, le vecteur d'entrée et la sortie du système.

2) Séparer les données en un nombre déterminé de groupes à l'aide de méthode de partitionnement.

3) Vérifiez les ressemblances entre les groupes car parfois, deux groupes peuvent décrire le même hyperplan.

4) Utilisez les résultats du partitionnement pour construire les fonctions d'appartenance (triangulaire, gaussienne, polynomiale, trapézoïdale, etc.).

Étapes en ligne

5) Calculer les paramètres des conséquences en utilisant les moindres carrés récursifs.

6) Ajustez les paramètres des antécédents (si nécessaire) en utilisant la méthode du gradient.

Algorithme 2.1 : Identification basée sur les méthodes de partitionnement et moindres carrés récursifs

2.9 Conclusion

Ce chapitre présente brièvement quelques principes théoriques pour comprendre la modélisation floue et l'identification floue de systèmes basés sur les données entrées-sorties. Après avoir introduit les concepts de base et la structure générale des modèles flous, nous avons étudié plus précisément le modèle Takagi-Sugeno (TS). En effet, au cours des vingt dernières années, la discipline de modélisation floue a évolué de façon graduelle mais décidée, vers une utilisation pratiquement exclusif des systèmes flous dans lesquels la conséquence des règles utilise des variables numériques sous la forme des fonctions (modèle de type TS) plutôt que des variables linguistiques telles que le modèle de type Mamdani. Les modèles flous de type TS

gardent les caractéristiques de transparence et d'interprétabilité linguistique qui séparent les modèles flous d'autres approches similaires de type boîtes noires.

Dans le cas de l'identification floue des systèmes, le formalisme Takagi-Sugeno est mieux adapté à une approche plus systématique pour la construction de modèles non linéaires multi variables, grâce à leur bonne interpolation numérique et leur capacité d'apprentissage à partir des données.

Chapitre 3

Commande Prédictive Floue Basée Sur la Méthode de Régression Ridge à Noyau

3.1 Introduction

Les modèles flous de type TS guidés par des données d'entrées-sorties sont généralement utilisée pour décrire les systèmes non linéaires inconnus à partir de règles Si-Alors, comme mentionnée dans chapitre 2. Les procédures d'apprentissage (l'identification de paramètres des conséquences) sont pratiquement les mêmes que celles utilisés par les méthodes basées sur les réseaux des neurones. Les techniques basées sur les réseaux des neurones ont été largement utilisées pour l'identification du système, telles que le perceptron multicouches (MLP) et l'approche du système d'inférence adaptative neuro-flou (ANFIS). Ces techniques se sont avérées robustes et efficaces dans les domaines de l'identification et de contrôle, mais elles ont montré aussi quelques inconvénients tels que les problèmes de plusieurs minima locaux et sur-ajustement (Overfitting). Ces inconvénients peuvent être évités en utilisant des méthodes basées sur les noyaux car ces techniques offrent une plus grande performance de généralisation.

Dans ce chapitre, nous définirons une nouvelle modélisation floue de type TS basé sur la méthode de Régression Ridge à noyau, TS-KRR où les fonctions d'appartenance et leurs paramètres, les paramètres des antécédents, sont définis par une méthode de partitionnement (aussi connu sur le nom : algorithme de coalescence ou regroupement) alors que les paramètres des conséquences (les paramètres linéaires) sont obtenus à l'aide de la méthode de régression ridge à noyau (KRR). En outre, la commande prédictive généralisée floue est construite en intégrant le modèle TS-KRR dans la commande GPC.

Nous examinerons la performance de la méthode floue proposée sous ses deux formes, le modèle TS-KRR hors ligne et le modèle TS-KRR adaptatif pour l'identification et le contrôle des systèmes dans le chapitre 4. Une attention particulière sera accordée au modèle TS-KRR adaptatif dans le cas de la commande prédictive généralisée floue.

3.2 Structure d'un modèle TS flous basé sur la régression ridge à noyau

D'une manière générale, un modèle flou de type TS du premier ordre constitue un outil très efficace pour approximer les systèmes non linéaires. Dans ce travail, on ne s'intéresse qu'à une seule sortie (Multi-Entrées Mono-Sortie) car nous pouvons toujours faire une décomposition de modèle MIMO en sous-systèmes MISO.

Le modèle flou de type TS-KRR proposé est basé sur une décomposition floue de l'espace des entrées où pour chaque région de cet espace, une règle floue peut être construite afin de faire une approximation linéaire de la sortie. La sortie globale de modèle flou de type TS-KRR est donc obtenue par une combinaison spéciale de l'ensemble des règles construites où la defuzzification de ce modèle est la seule couche différente de celle du modèle flou classique de type TS (ou des modèles neuro-flous basé sur Takagi-Sugeno). Le modèle floue de type TS-KRR peut aussi être vu comme une structure multi modèles composés de différents modèles linéaires qui ne sont pas nécessairement indépendants. Considérons la figure (3.1) décrivant l'architecture du modèle TS-KRR pour expliquer le mécanisme d'inférence.

Généralement, un modèle flou de type TS-KRR est basé sur une collection de règles du type :

$$\begin{aligned} R_i: \quad & \text{Si } x_1(k) \; est \; A_1^i, et \; \dots, et \; x_n(k) \; est \; A_n^i \\ & \text{Alors } y_i(k) = a_1^i x_1(k) + \cdots + a_n^i x_n(k) \\ & \qquad i = 1, \dots, N_r \end{aligned} \tag{3.1}$$

avec $R_i, i = 1, 2, \dots, N_r$ est la $i^{\text{ème}}$ règle floue et N_r est la nombre de règles flous. Les variables d'entrées sont : $x_1(k), \dots, x_n(k)$, où k dénote l'échantillon du temps discret, et $y_i(k)$ est la sortie de la $i^{\text{ème}}$ règle floue. Les $A_j^i, i = 1, 2, \dots, N_r, j = 1, 2, \dots, n$ son les sous-ensembles flous antécédents où ces sous-ensembles flous sont caractérisés par les fonctions d'appartenance floue $\mu_{A_j^i}(x_j),\ i = 1, 2, \dots, N_r, j = 1, 2, \dots, n$. Comme montré sur la figure (3.1), l'architecture du

modèle TS-KRR est composée par cinq couches. Dans cette section, la structure de chaque couche est présentée en détail.

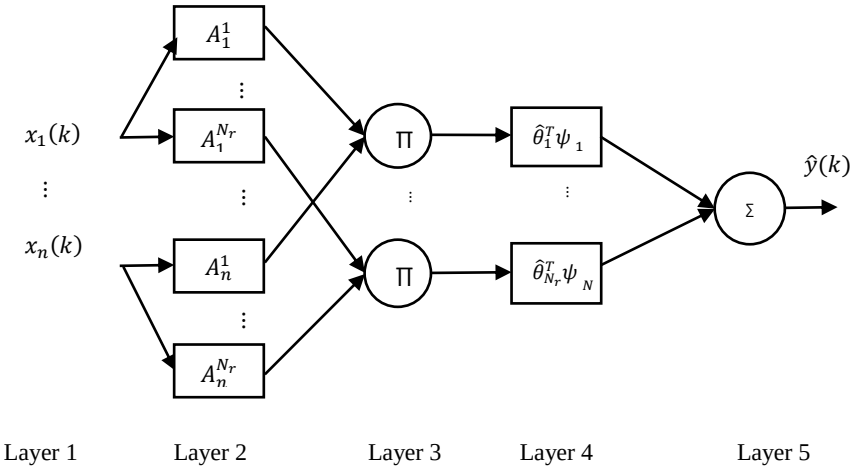

Figure 3.1 : La configuration de la structure de modèle TS-KRR

1. Couche 1 : qui représente les variables d'entrée du modèle où ces variables sont transmises directement à la couche 2.
2. Couche 2 (fuzzification) : dans cette couche les termes linguistiques qui sont définis par les sous-ensembles A_j^i de l'antécédent de la $i^{\text{ème}}$ règle. Ils sont caractérisés par leurs fonctions d'appartenance $\mu_{A_j^i}$. Cette couche peut être considérée comme la couche qui calcule le degré d'appartenance de la $j^{\text{ème}}$ variable d'entrée $x_j(k)$ dans la $i^{\text{ème}}$ règle où la valeur d'appartenance de la $j^{\text{ème}}$ variable d'entrée est définie par la fonction d'appartenance $\mu_{A_j^i}(x_j)$.
3. Couche 3 : dans cette couche, le degré d'activation (aussi connu comme le degré d'accomplissement) de chaque règle peut être obtenu par le produit cartésien des ensembles flous relatifs à la règle. Dans notre cas, on peut dire que le degré d'accomplissement de $i^{\text{ème}}$ règle est simplement égal au degré d'appartenance d'entrée multidimensionnelle.
4. Couche 4 : cette couche calcule la contribution de toutes les règles floues ou la contribution de $i^{\text{ème}}$ règle est donné par $\widehat{\boldsymbol{\theta}}_i\psi_i$ avec : ψ_i est le degré d'activation de $i^{\text{ème}}$ règle et $\widehat{\boldsymbol{\theta}}_i$ sont les paramètres de conséquences de la $i^{\text{ème}}$ règle.

5. Couche 5 (aussi vu comme la defuzzification du modèle) : dans cette couche, la sortie du modèle floue TS est calculée par l'addition de toutes les contributions des règles du système.

Comme montré à la figure (3.1), l'élément de defuzzification dans le modèle TS-KRR n'effectue aucune normalisation et on prouvera, dans la section suivante, que le modèle flou de type TS-KRR proposé n'a pas besoin de normalisation lorsque la régression ridge à noyau est implémentée pour identifier les paramètres des conséquences ($\widehat{\boldsymbol{\theta}}_i,\ i = 1, \dots, N_r$).

Les fonctions d'appartenances utilisées par le modèle TS-KRR sont représentées par des fonctions gaussiennes pour assurer la meilleure généralisation possible (permet de couvrir l'ensemble du domaine des variables). Les fonctions gaussiennes sont données par la forme suivant :

$$\mu_{A_j^i}(x_j) = exp\left\{-\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2}\right\}, i = 1, \dots, N_r \ and \ j = 1, \dots, n \tag{3.2}$$

avec m_{ij} et σ_{ij} sont respectivement le centre et la variance de la $i^{\text{ème}}$ fonction d'appartenance. Le degré d'activation de la $i^{\text{ème}}$ règle est donné par le produit cartésien de toutes les fonctions d'appartenances qui appartient de cette règle :

$$\begin{aligned}\mu_i(\boldsymbol{x}) = \prod_{j=1}^{n} \mu_{A_j^i}(x_j) &= exp\left\{-\sum_{j=1}^{n}\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2}\right\} \\ &= exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \Sigma (\boldsymbol{x} - \boldsymbol{m}_i)\right\}, i = 1, \dots, N_r\end{aligned} \tag{3.3}$$

où $\Sigma = \begin{bmatrix} {\sigma_{i1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {\sigma_{in}}^2 \end{bmatrix}^{-1}$. Dans la quatrième couche, la fonction ψ_i qui définit le degré d'activation de $i^{\text{ème}}$ règle est donné par : $\psi_i = \boldsymbol{x}.\mu_i(\boldsymbol{x})$. Enfin, la sortie du modèle floue de type TS-KRR est calculée par :

$$\hat{y}(k) = \sum_{i=1}^{N_r}\left(\widehat{\boldsymbol{\theta}}_i^T.\boldsymbol{x}\right).\mu_i(\boldsymbol{x}) = \sum_{i=1}^{N_r}\left(\widehat{\boldsymbol{\theta}}_i^T.\boldsymbol{x}\right).exp\left\{-\sum_{j=1}^{n}\left(\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2}\right)\right\} \tag{3.4}$$

Dans les sections suivantes, les procédures utilisées pour l'identification des vecteurs des conséquences $\widehat{\boldsymbol{\theta}}_i, i = 1, \dots, N_r$ du modèle TS-KRR ainsi que les valeurs des centres et des variances des fonctions d'appartenance sont présentés en détail.

3.3 Identification des paramètres des conséquences du modèle TS-KRR

Dans cette section, on va utiliser la méthode de régression ridge à noyau, qui a été présenté dans chapitre 1, afin d'identifier hors ligne les paramètres des conséquences du modèle TS.

Selon le théorème du représentant définit au chapitre 1 (Théorème 1.4), l'objectif des méthodes d'apprentissage à noyaux est de trouver une relation non linéaire $f: \chi \rightarrow \mathbb{R}$ tels que la sortie est exprimée par une extension des noyaux :

$$y(k) = f(\boldsymbol{x}) = \sum_{i=1}^{N_d} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) \tag{3.5}$$

où N_d représente la dimension d'ensemble d'apprentissage (nombre de données entrées-sorties), $\alpha_i \in \mathbb{R}, i = 1, \dots, N_d$ sont les coefficients d'expansion, $\kappa(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \boldsymbol{\varphi}(\boldsymbol{x},), \boldsymbol{\varphi}(\boldsymbol{x}_i) \rangle$ est une fonction noyau avec $\boldsymbol{\varphi}: \mathcal{X} \rightarrow \mathcal{H}$ une fonction de re-description qui projette les vecteurs d'entrée $\mathcal{X}$ dans un espace de Hilbert $\mathcal{H}$ (aussi connu sur le nom d'espace des caractéristiques).

Généralement, si les données d'apprentissage représentent différentes propriétés et (ou) peuvent être séparées en différents groupes (dans lequel, chaque groupe décrit une règle floue), une fonction multi-noyaux peut être utilisée pour obtenir des meilleures prédictions [55, 56, 57, 58, 60, 61, 62]. La fonction multi-noyaux est alors définie comme suit :

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{N_r} \kappa_l(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{N_r} \langle \boldsymbol{\varphi}_l(\boldsymbol{x}), \boldsymbol{\varphi}_l(\boldsymbol{z}) \rangle = \sum_{l=1}^{N_r} \boldsymbol{\varphi}_l(\boldsymbol{x})^T . \boldsymbol{\varphi}_l(\boldsymbol{z}) \tag{3.6}$$

Une fonction multi-noyaux est une combinaison de N_r fonctions noyaux où chaque fonction est associée à un groupe (une règle). D'une manière général, selon la propriété 1.1 (chapitre 1), la fonction $\kappa(\boldsymbol{x}, \boldsymbol{z})$ dans l'équation (3.6) est une fonction de noyau valide. Cela peut être facilement vérifié en utilisant le théorème de Mercer (théorème 1.1, chapitre 1) :

$$\sum_{i,j=1}^{N_d} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$= \sum_{i,j=1}^{N_d} \alpha_i \alpha_j \sum_{l=1}^{N_r} \langle \boldsymbol{\varphi}_l(\boldsymbol{x}_i), \boldsymbol{\varphi}_l(\boldsymbol{x}_j) \rangle$$

$$= \sum_{l=1}^{N_r} \sum_{i,j=1}^{N_d} \alpha_i \alpha_j \langle \boldsymbol{\varphi}_l(\boldsymbol{x}_i), \boldsymbol{\varphi}_l(\boldsymbol{x}_j) \rangle$$

$$= \sum_{l=1}^{N_r} \langle \sum_{i=1}^{N_d} \alpha_i \, \boldsymbol{\varphi}_l(\boldsymbol{x}_i), \sum_{j=1}^{N_d} \alpha_j \boldsymbol{\varphi}_l(\boldsymbol{x}_j) \rangle = \sum_{l=1}^{N_r} \left\| \sum_{i=1}^{N_d} \alpha_i \, \boldsymbol{\varphi}_l(\boldsymbol{x}_i) \right\|^2 \geq 0$$

ce qui est évident car $\kappa_l(\boldsymbol{x}, \boldsymbol{z})$ est une fonction noyau valide et les normes ne sont pas négatives ($\left\| \sum_{i=1}^{N_d} \alpha_i \, \boldsymbol{\varphi}_l(\boldsymbol{x}_i) \right\| \geq 0, \; l = 1, 2, \dots, N_r$).

Donc, pour obtenir la formulation de modèle TS-KRR présenté dans l'équation (3.4) deux choix doivent être sélectionnés par un expert ou par l'utilisation les données d'entrées-sorties :

- Le nombre de fonctions des noyaux N_r nécessaires pour définie le modèle TS-KRR.
- La forme de la fonction de re-description $\boldsymbol{\varphi}_l(\boldsymbol{x})$ qui projette les vecteurs d'entrées $\mathcal{X}$ dans un espace des caractéristiques pour définir chaque fonction noyau $\kappa_l(\boldsymbol{x}, \boldsymbol{z})$.

L'idée principale du modèle flou basée sur la méthode de KRR est de proposer une forme spéciale de la fonction de re-description qui conduira à la structure TS-KRR définie dans l'équation (3.4) où les paramètres des conséquences sont obtenus à partir des coefficients d'expansion. Dans ce travail, les fonctions des re-descriptions $\boldsymbol{\varphi}_l(\boldsymbol{x}), l = 1, 2, \dots, N_r$ sont définies par :

$$\begin{aligned} \boldsymbol{\varphi}_l(\boldsymbol{x}) = (\boldsymbol{x}).\mu_l(\boldsymbol{x}) &= (\boldsymbol{x}).exp\left\{ -\sum_{j=1}^{n} \frac{\left(x_j - m_{lj}\right)^2}{2{\sigma_{lj}}^2} \right\} \\ &= (\boldsymbol{x}).exp\left\{ -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_l)^T.\Sigma.(\boldsymbol{x} - \boldsymbol{m}_l) \right\}, l = 1, \dots, N_r \end{aligned} \tag{3.7}$$

où $\boldsymbol{x} = (x_1, \dots, x_n)^T$ est le vecteur d'entrée, $\boldsymbol{m}_l = (m_{1l}, \dots, m_{nl})^T$ et $\boldsymbol{\sigma}_l = (\sigma_{1l}, \dots, \sigma_{nl})^T$ sont respectivement les vecteurs de centre et de variance, avec la matrice $\Sigma = \begin{bmatrix} {\sigma_{l1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {\sigma_{ln}}^2 \end{bmatrix}^{-1}$.

La dimension de fonction de re-description $\boldsymbol{\varphi}_l(\boldsymbol{x}), l = 1, 2, \dots, N_r$ est identique à celles du vecteur d'entrée $\boldsymbol{x}$.

Maintenant, après avoir défini les fonctions des re-descriptions, nous remplaçons ces fonctions dans le modèle de la méthode de régression ridge à noyau (équation 3.5). Le résultat est donné par :

$$
\begin{aligned}
y(k) &= \sum_{i=1}^{N_d} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x_i}) = \sum_{i=1}^{N_d} \alpha_i . \left(\sum_{l=1}^{N_r} \kappa_l(\boldsymbol{x}, \boldsymbol{x}_i) \right) \\
&= \sum_{i=1}^{N_d} \alpha_i \left(\sum_{l=1}^{N_r} \langle \boldsymbol{\varphi}_l(\boldsymbol{x}), \boldsymbol{\varphi}_l(\boldsymbol{x}_i) \rangle \right) \\
&= \sum_{i=1}^{N_d} \alpha_i . \left(\sum_{l=1}^{N_r} \boldsymbol{\varphi}_l(\boldsymbol{x}_i)^{\boldsymbol{T}} \boldsymbol{\varphi}_l(\boldsymbol{x}) \right) \\
&= \sum_{l=1}^{N_r} \left(\sum_{i=1}^{N_d} \alpha_i . \boldsymbol{\varphi}_l(\boldsymbol{x}_i)^{\boldsymbol{T}} \right) \boldsymbol{\varphi}_l(\boldsymbol{x}) \\
&= \sum_{l=1}^{N_r} \left(\sum_{i=1}^{N_d} \alpha_i . \boldsymbol{x}_i{}^{\boldsymbol{T}} \mu_l(\boldsymbol{x}_i) \right) \boldsymbol{x} . \mu_l(\boldsymbol{x})
\end{aligned}
$$

Simplement, si en considère $\widehat{\boldsymbol{\theta}}_l = \left(\sum_{i=1}^{N_d} \alpha_i \boldsymbol{\varphi}_l(\boldsymbol{x}_i) \right) = \sum_{i=1}^{N_d} \alpha_i \boldsymbol{x}_i \mu_l(\boldsymbol{x}_i)$, alors le modèle floue de type TS-KRR présentée par l'équation (3.4) est obtenu. Donc, les paramètres des conséquences sont :

$$
\widehat{\boldsymbol{\theta}}_l = \left(\sum_{i=1}^{N_d} \alpha_i \boldsymbol{\varphi}_l(\boldsymbol{x}_i) \right) = \sum_{i=1}^{N_d} \alpha_i \boldsymbol{x}_i \mu_l(\boldsymbol{x}_i) , l = 1, 2, \dots, N_r \tag{3.8}
$$

et le modèle floue de type TS-KRR est donnée par :

$$y(k) = \sum_{l=1}^{N_r} \widehat{\boldsymbol{\theta}}_l^T \, \boldsymbol{\varphi}_l(\boldsymbol{x}) = \sum_{l=1}^{N_r} \widehat{\boldsymbol{\theta}}_l^T \, \boldsymbol{x}.\mu_l(\boldsymbol{x}) = \sum_{l=1}^{N_r} \left(\widehat{\boldsymbol{\theta}}_l^T . \boldsymbol{x}\right) . exp\left\{ - \sum_{j=1}^{n} \frac{\left(x_j - m_{lj}\right)^2}{2{\sigma_{lj}}^2} \right\} \tag{3.9}$$

Il est clair que pour un certain ensemble de données d'entrée-sortie, les paramètres des conséquences du modèle TS-KRR peuvent être identifiés à partir des coefficients d'expansion $\boldsymbol{\alpha} = \left(\alpha_1, \dots, \alpha_{N_D}\right)^T$ qui peuvent être facilement obtenus à partir du problème quadratique dans l'équation (1.8) (chapitre 1). Les coefficients d'expansion $\boldsymbol{\alpha}$ sont donnés par (voir section 1.4) :

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \tag{3.10}$$

où $\lambda > 0$, $\boldsymbol{y} = \left(y_1, \dots, y_{N_d}\right)^T$, $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d)^T$ et $\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \kappa\left(\boldsymbol{x}_1, \boldsymbol{x}_{N_d}\right) \\ \vdots & \dots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & \kappa\left(\boldsymbol{x}_N, \boldsymbol{x}_{N_d}\right) \end{pmatrix}$.

La seule partie qui manque pour finaliser la construction du modèle flou de type TS-KRR est l'identification des paramètres des antécédents (nombre de règles, $\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$). Dans ce travail, une méthode de partitionnement sera utilisée pour identifier les paramètres des antécédents. Dans la prochaine section, on va discuter une méthode de partitionnement basée sur l'optimisation par essaim particulaire (*Particle Swarm Optimization* : PSO) où le nombre de règles sera défini comme le nombre de groupes, et $\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$ sont le centre et la variance de chaque groupe l, respectivement.

3.4 Identification des paramètres des antécédents du modèle floue TS-KRR

Dans cette section, les paramètres des antécédents du modèle flou de type TS-KRR sont identifiés à partir d'une méthode de regroupement (partitionnement) basée sur l'optimisation par essaim particulaire, où les centres et les variances des groupes représenteront les vecteurs $\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$ des fonctions noyaux.

Généralement, il y a plusieurs façons de choisir une méthode de partitionnement. Dans le cas d'un modèle flou adaptatif (en ligne) de type TS-KRR (qui sera présenté plus tard), une méthode

simple comme la méthode K-moyennes [66] pourrait être suffisante pour initialiser l'algorithme alors que les vecteurs $\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$ sont ajustés à chaque itération.

Cependant, pour le modèle floue hors ligne de type TS-KRR, un algorithme de partitionnement précis aide à améliorer les prédictions. Dans ce chapitre, une méthode de partitionnement basée sur un algorithme PSO est utilisé pour la modélisation floue hors ligne de type TS-KRR. Cette méthode de partitionnement est proposée par Van der Merwe et al. [67]. Dans cette méthode, l'algorithme des K-moyennes est utilisée pour initialiser les valeurs des vecteurs $\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$. Ensuite, l'algorithme PSO est utilisé pour obtenir un regroupement précis des données d'entrées-sorties car il existe toujours la possibilité que l'algorithme k-moyennes peut classer de manière incorrecte certaines paires des données.

3.4.1 La méthode de K-moyennes

D'une manière générale, les méthodes de regroupement sont des tâches qui consiste à regrouper, d'une façon non supervisée, un ensemble des données d'entrées-sorties de telle manière que les vecteurs des données (ou certaines objectes) d'un même groupe sont plus proches (au sens d'un critère de similarité choisi comme par exemple la distance euclidienne) les unes aux autres que celles des autres groupes. L'idée principale de ces méthodes est donc de découvrir des groupes au sein des données, de façon automatique [68]. Dans ce cadre plusieurs méthodes ont été développées, la plus populaire est celle de k-moyennes (K-means), elle doit sa popularité à sa simplicité et sa capacité de traiter de larges ensembles de données [69]. Voir l'annexe B pour plus d'informations sur cette méthode.

La méthode de k-moyennes donne, la plupart du temps, un partitionnement localement optimal, il est donc conseiller d'effectuer plusieurs exécutions et comparer les différents résultats obtenus. Dans ce travail, en va appliquer la méthode des k-moyennes dans une première étape à un ensemble de données, et les résultats obtenus de partitionnement à utiliser pour initialiser une approche de partitionnement basée sur l'optimisation PSO pour obtenir un bon partitionnement [28].

3.4.2 Optimisation par Essaim Particulaire

L'optimisation par essaim particulaire (*Particle Swarm Optimization* : PSO) est une méthode évolutionnaire basée sur une population de solutions candidates afin de trouver une solution

optimale au problème. Cette méthode a été proposée par Eberhart et Kennedy [70] où l'idée principale de cette approche est inspirée par le comportement social des animaux évoluant en essaim, tels que les vols groupés d'oiseaux et les bancs de poissons.

L'essaim de particules se compose d'une population de particules où chaque particule est considérée comme une solution potentielle au problème d'optimisation. Généralement, chaque particule possède une vecteur position (notée par : la vecteur solution) et une vecteur vitesse. De plus, chaque particule à une mémoire lui permettant de se souvenir de sa meilleure performance (la position et sa valeur de fonction d'objectif) et aussi de la meilleure performance atteinte par les voisines (particules) [71]. Le déplacement d'une particule est influencé par trois composantes qui sont résumées comme suit :

- Une composante d'inertie dans laquelle la particule tend à suivre sa direction courante de déplacement.
- La particule tend à se diriger vers le meilleur site par lequel elle est déjà passée en utilisant une composante cognitive.
- Une composante sociale dans laquelle la particule tend à se diriger vers le meilleur site déjà atteint par ses voisins.

Dans un espace de recherche de dimension n, la particule i est modélisée par son vecteur position $\boldsymbol{\check{x}_i} = (\check{x}_i^1, \dots, \check{x}_i^n)$ et par son vecteur vitesse $\boldsymbol{v_i} = (v_i^1, \dots, v_i^n)$. La qualité de sa position est déterminée par la valeur de la fonction objective à minimiser. Aussi, la particule i garde en mémoire la meilleure position par laquelle elle est déjà passée, que l'on note par $\boldsymbol{p_i} = (p_i^1, \dots, p_i^n)$ et sa valeur de la fonction objectif. La meilleure position atteinte par les particules, notée par : global best, de l'essaim est notée par : $\boldsymbol{p_g} = (p_g^1, \dots, p_g^n)$.

Au départ de l'algorithme, toutes les particules sont initialisées de manière aléatoire dans l'espace de recherche du problème. Ensuite, à chaque itération, chaque particule se déplace, en combinant linéairement les trois composantes citées ci-dessus où ces trois composantes sont réalisées par le vecteur vitesse et le vecteur position qui sont donnée respectivement par les équations (3.11) et (3.12) :

$$\boldsymbol{v_i} = w\boldsymbol{v_i} + c_1 r_1(\boldsymbol{p_i} - \boldsymbol{\vec{x}_i}) + c_2 r_2(\boldsymbol{p_g} - \boldsymbol{\vec{x}_i}) \tag{3.11}$$

$$\boldsymbol{\vec{x}_i} = \boldsymbol{\vec{x}_i} + \boldsymbol{v_i} \tag{3.12}$$

où le constante w est un coefficient d'inertie, les constantes c_1 et c_2 sont appelées les coefficients d'accélération et enfin r_1 et r_1 sont deux variable aléatoire uniformément distribuées dans l'intervalle $[0, 1]$. Dans ce travail, le coefficient d'inertie est réduit pendant l'évaluation de l'algorithme PSO pour permettre la recherche locale où :

$$w = \frac{iter\_\max - iter}{iter\_max} \times w_0,$$

avec w_0 la valeur initiale du coefficient d'inertie, $iter\_max$ est le nombre final d'itération du algorithme et $iter$ est la valeur courant d'itération.

3.4.3 Méthode de partitionnement basée sur l'algorithme de PSO

L'algorithme PSO est ajusté pour faire un partitionnement des données. Dons ce cas, une particule $\boldsymbol{\vec{x}_i}$ contient tous les vecteurs des centres des groupes tels que $\boldsymbol{\vec{x}_i} = (\boldsymbol{m_i^1}, \dots, \boldsymbol{m_i^{N_r}})$ où chaque particule représentent une solution possible de partitionnement. La fonction d'objectif est alors définie par :

$$J = \frac{\sum_{i=1}^{N_r} \left(\sum_{\forall\, \boldsymbol{z_p} \in C_i} \frac{d(\boldsymbol{z_p}, \boldsymbol{m_i})}{|C_i|} \right)}{N} \tag{3.13}$$

avec les éléments de groupe i notés par C_i, et le nombre des vecteurs (éléments) des données qui appartient à groupe i est donné par $|C_i|$. La distance euclidienne entre deux vecteurs est calculée comme suit :

$$d(\boldsymbol{z}_p, \boldsymbol{m_i}) = \sqrt{\sum_{j=1}^{n} \left(z_j - m_{ij}\right)^2} \tag{3.14}$$

avec le nouveau centre et variance de groupe C_i sont calculés par :

$$\boldsymbol{m_i} = \frac{\mathbf{1}}{|C_i|} \sum_{\forall\, \boldsymbol{z_p} \in C_i} \boldsymbol{z}_p$$

$$\sigma_{ij} = \frac{\mathbf{1}}{|C_i|} \left(\sum_{\forall\, \boldsymbol{z_p} \in C_i} \left(z_{pj} - m_{ij}\right)^{\mathbf{2}} \right)^{\frac{1}{2}} \tag{3.15}$$

Finalement, l'algorithme de partitionnement est résumé comme suit :

Algorithme

1) Initialisation:
   Initialiser les particules dans lesquelles chaque particule contient N_r centres.
   Les particules sont initialisées d'une manière aléatoire, sauf pour une particule qui présente les résultats de l'algorithme K-moyennes.
2) pour i=1: $iter\_max$
   a) Pour chaque particule i faire
      i) Pour chaque vecteur des données $\boldsymbol{z}_p$ faire
         (1) La distance par rapport à tous les centres est calculée.
         (2) Assigner le vecteur $\boldsymbol{z}_p$ au groupe qui a distance minimale.
         (3) Calculez la valeur de fonction d'objectif en utilisant équation 3.13
      ii) Ajuster la meilleure position globale et locale en utilisant équations 3.11 et 3.12.
      iii) Calculer les nouveaux centres et variance en utilisant les équations dans 3.15

Algorithme 3.1 : Méthode de partitionnement basée sur l'algorithme de PSO

Après avoir introduit les procédures pour calculer les paramètres des antécédents et les paramètres des conséquences, la modélisation floue à partir de modèle floue hors ligne de type TS-KRR est résumé comme suit :

Algorithme

1) Initialisation: la dimension de vecteur d'entrée, le nombre de groupes N, les paramètres d'algorithme PSO (w, c_1, c_2, $iter\_max$ et le nombre de particules).
2) Exécuter Algorithme 3.1 pour calculer les paramètres des antécédents.
3) Obtenir les coefficients d'expansion en utilisant l'équation 3.10.
4) Calculer les paramètres des conséquences $\widehat{\boldsymbol{\theta}}_l, l = 1, 2, \dots, N_r$ en utilisant l'équation 3.8.
5) Enfin, la sortie du modèle flou de type TS-KRR est calculée par l'équation 3.4.

Algorithme 3.2 : Le modèle floue de type TS-KRR hors ligne

3.5 Adaptation en ligne du modèle flou

Dans la section précédente, l'identification hors ligne du modèle flou de type TS-KRR à partir des données d'entrée-sortie a été introduite. Dans le cas où les données d'entrées-sorties ne suffisent pas à décrire toutes les zones des fonctionnements des systèmes non linéaires, l'adaptation en ligne est nécessaire pour obtenir un modèle capable de poursuivre le système dans son évolution.

L'adaptation en ligne du modèle floue de type TS-KRR peut être résumée en deux étapes: la première étape consiste à adapter les paramètres des antécédents ($\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$, $l = 1, \dots, N_r$). Cela peut être réalisé par un simple algorithme de rétro-propagation. La deuxième étape consiste à utiliser une méthode de régression en ligne, moindres carrés récursifs à noyau (KRLS) [72], pour ajuster les paramètres des conséquences. Dans ce travail, en va utiliser une méthode modifiée du KRLS dite la méthode de moindres carrés récursifs à noyau avec une fenêtre coulissante (*sliding-window KRLS*) [73]. L'idée principale est alors de modifier récursivement les coefficients d'expansion, puis d'utiliser ces coefficients pour obtenir les nouvelles valeurs de paramètres des conséquences $\widehat{\boldsymbol{\theta}}_l, l = 1, \dots, N_r$.

Généralement, la dimension de l'ensemble d'apprentissage, est souvent très grande, voire infinie dans le cadre d'une acquisition en temps réel (identification en ligne). Dans le cas d'identification en ligne d'un système dynamique à l'aide des méthodes à noyaux adaptative, le développement de la solution conduit à une série dont la taille croit infiniment avec le temps.

Dans ce cas, il est impossible d'utiliser les méthodes à noyaux dans le cas d'applications en ligne et même dans le cas où l'ensemble d'apprentissage est de dimension très grande [74].

Pour surmonter cet obstacle, plusieurs méthodes ont été développées pour rendre ce type de méthode applicable à l'identification en ligne. Dans le cas où il y a un ensemble d'apprentissage avec une très grande dimension, en peut décomposer le problème en plusieurs sous-problèmes, comme par exemple la technique de décomposition [76, 77], la technique de segmentation (*chunking*) [75], la sélection de sous-ensembles optimaux ou l'approche de contraction (*shrinking*) [78]. Dans le cas de l'identification en ligne, l'adaptation d'un coefficient d'expansion du modèle à noyau à chaque itération est proposée par la méthode d'Adatron qui est basée sur l'algorithme de descente de gradient stochastique [79]. D'autre part, l'approche d'optimisation par minimisation séquentielle [80] peut actualiser deux coefficients du modèle à chaque itération.

Dans cette thèse, les méthodes basées sur l'élagage sont adoptées pour l'adaptation en ligne du modèle floue de type TS-KRR. Le principe de l'élagage (*pruning*) a été proposé dans plusieurs travaux relatifs aux méthodes à noyau. Le contrôle de complexité varie alors entre l'élimination des échantillons d'apprentissage (paires des données d'entrées-sorties) qui sont les plus anciennes (simplement les fonctions noyau les plus anciennes sont éliminées) [81], l'élagage aléatoire [82], l'élagage de la fonction avec le plus petit coefficient (l'élagage de donnée d'entrée-sortie qui donne cette fonction) [83] et l'élimination des fonctions noyau à faible contribution dans le modèle [84].

Dans ce chapitre, on utilise une méthode basée sur un élagage des fonctions noyau les plus anciennes [81] dite la méthode de moindres carrés récursifs à noyau avec une fenêtre coulissante (*Sliding-Window Kernel Recursive Least squares* : SW-KRLS). Le concept principal de cette méthode est de préserver un dictionnaire d'une dimension fixe égale à M alors que l'échantillon d'apprentissagele plus ancienne est rejetée de ce dictionnaire et remplacées par le nouvel échantillon d'apprentissage arrivé à l'instant k. Mais, avant de discuter la méthode de SW-KRLS, nous discuterons d'abord la méthode de rétro-propagation utilisée pour ajuster les paramètres des antécédents.

Le principe de la rétro-propagation est de traiter un échantillon de formation itérative, et de comparer la prédiction obtenue par le modèle flou de type TS-KRR pour chaque échantillon avec la valeur de sortie réelle. Afin de minimiser l'erreur quadratique entre la prédiction du modèle

floue et la sortie réelle, les paramètres des antécédents sont donc modifiés pour chaque échantillon d'apprentissage. Généralement, l'erreur quadratique entre la prédiction du modèle floue et la sortie réelle du système est donnée par :

$$e(k) = \frac{1}{2}(y(k) - \hat{y}(k))^2 \tag{3.16}$$

avec $y(k)$ et $\hat{y}(k)$ sont respectivement les valeurs de la sortie prédite par le modèle floue et la sortie réel du système. Alors, les valeurs ajuster de m_{ij} et σ_{ij} à l'instant $k+1$ sont données par :

$$m_{ij}(k+1) = m_{ij}(k) - \eta \frac{\partial e(k)}{\partial m_{ij}} = m_{ij}(k) + \eta(y(k) - \hat{y}(k)) \frac{\partial \hat{y}(k)}{\partial m_{ij}} \tag{3.17}$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \eta \frac{\partial e(k)}{\partial \sigma_{ij}} = \sigma_{ij}(k) + \eta(y(k) - \hat{y}(k)) \frac{\partial \hat{y}(k)}{\partial \sigma_{ij}} \tag{3.18}$$

où η est une constante positive appelée taux, ou pas, d'apprentissage donnée par : $\eta = \frac{\beta}{\sum_{j=1}^{N}\sum_{i=1}^{n}\left(\left(\frac{\partial \hat{y}(k)}{\partial m_{ij}}\right)^2 + \left(\frac{\partial \hat{y}(k)}{\partial \sigma_{ij}}\right)^2\right)}$, et :

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial m_{ij}} &= \frac{\partial \hat{y}(k)}{\partial \boldsymbol{\varphi}_l(x)} . \frac{\partial \boldsymbol{\varphi}_l(x)}{\partial \mu_{A_j^i}(x_j)} . \frac{\partial \mu_{A_j^i}(x_j)}{\partial m_{ij}} \\ &= \boldsymbol{\varphi}_l{}^T(x) . \left(\boldsymbol{\varphi}_l(x) . \left(\sum_{l=1}^{M} \alpha_i \right) + \widehat{\boldsymbol{\theta}}_i \right) . \frac{(x_j - m_{ij})}{\sigma_{ij}{}^2} \end{aligned} \tag{3.19}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial \sigma_{ij}} &= \frac{\partial \hat{y}(k)}{\partial \boldsymbol{\varphi}_l(x)} . \frac{\partial \boldsymbol{\varphi}_{l_l}(x)}{\partial \mu_{A_j^i}(x_j)} . \frac{\partial \mu_{A_j^i}(x_j)}{\partial \sigma_{ij}} \\ &= \boldsymbol{\varphi}_l{}^T(x) . \left(\boldsymbol{\varphi}_l(x) . \left(\sum_{l=1}^{M} \alpha_i \right) + \widehat{\boldsymbol{\theta}}_i \right) . \frac{(x_j - m_{ij})^2}{\sigma_{ij}{}^3} \end{aligned} \tag{3.20}$$

avec M est la dimension du dictionnaire utilisé par la méthode des moindres carrés récursifs à noyau avec une fenêtre coulissante (SW-KRLS).

L'importance de l'utilisation des méthodes à noyau pour l'adaptation en ligne de modèle flou réside dans le fait que le modèle flou sera écrit sous forme d'une combinaison linéaire de

fonctions noyau. Le modèle est mis à jour à chaque itération basé sur l'erreur entre la réponse réel du système et la sortie du modèle flou.

Supposons qu'un ensemble de données de taille $k-1$ ait été reçue et traitée à la $(k-1)$ -ème itération. La solution de régression de l'équation (3.10) peut être exprimée par :

$$\boldsymbol{\alpha}_{k-1} = \acute{\boldsymbol{K}}_{k-1}^{-1}\boldsymbol{y}_{k-1} \tag{3.21}$$

avec la matrice du noyau régularisé est donné par $\acute{\boldsymbol{K}} = \boldsymbol{K} + \lambda \boldsymbol{I}$, $\boldsymbol{I}$ est une matrice identité. Lorsqu'un nouvelle échantillon d'apprentissage $\big(\boldsymbol{x}(k), y(k)\big)$ arrive à l'instant k, pour ajuster les coefficient d'expansion dans l'équation (3.21), la sortie du modèle floue est d'abord calculée comme : $\hat{y}(k) = \boldsymbol{\kappa}_k^T \boldsymbol{\alpha}_{k-1}$, avec $\boldsymbol{\kappa}_k = (\kappa(\boldsymbol{x}(k), \boldsymbol{x}(1)), \dots, \kappa(\boldsymbol{x}(k), \boldsymbol{x}(k-1)))^T$.

Ensuite, l'erreur entre la sortie réelle et la sortie prédite est obtenue comme : $e_k(k) = y(k) - \hat{y}(k)$. Enfin, la nouvelle matrice du noyau régularisé $\acute{\boldsymbol{K}}_k$ est obtenue à partir de la matrice $\acute{\boldsymbol{K}}_{k-1}$. Ceci peut être facilement fait en ajoutant une ligne et une colonne comme suit :

$$\acute{\boldsymbol{K}}_k = \begin{pmatrix} \acute{\boldsymbol{K}}_{k-1} & \boldsymbol{\kappa}_k \\ \boldsymbol{\kappa}_k^T & \kappa\big(\boldsymbol{x}(k), \boldsymbol{x}(k)\big) + \lambda \end{pmatrix} \tag{3.22}$$

Cette étape est désignée par : "*upsizing*" de la matrice. La matrice inverse de $\acute{\boldsymbol{K}}_k$ peut être facilement obtenue lorsque de nouvelles variables tels que : $\boldsymbol{h}_k = \acute{\boldsymbol{K}}_{k-1}^{-1}\boldsymbol{\kappa}_k$ et $\Upsilon_k = \kappa\big(\boldsymbol{x}(k), \boldsymbol{x}(k)\big) + \lambda - \boldsymbol{\kappa}_k^T \boldsymbol{a}_k$ sont introduites pour simplifier la formule de la matrice inverse à l'instance k, et la matrice inverse est donc donnée par :

$$\acute{\boldsymbol{K}}_k^{-1} = \frac{1}{\Upsilon_k}\begin{pmatrix} \Upsilon_k \acute{\boldsymbol{K}}_{k-1}^{-1} + \boldsymbol{h}_k \boldsymbol{h}_k^T & -\boldsymbol{h}_k \\ -\boldsymbol{h}_k & 1 \end{pmatrix} \tag{3.23}$$

Enfin, les mises à jour des coefficients d'expansion, lorsque la taille du dictionnaire est plus petite que la valeur maximal M, est donné par :

$$\boldsymbol{\alpha}_k = \begin{pmatrix} \boldsymbol{\alpha}_{k-1} - \dfrac{\boldsymbol{h}_k e_k(k)}{\Upsilon_k} \\ \dfrac{e_k(k)}{\Upsilon_k} \end{pmatrix} \tag{3.24}$$

L'objectif de l'approche de fenêtre coulissante SW-KRLS est de supprimer les données les plus anciennes qui correspondent à la première ligne et à la première colonne de la matrice du

noyau régularisé $\acute{\boldsymbol{K}}_k$. L'opération de suppression de la première ligne et la première colonne dans chaque itération est connu par *downsizing*, réduction de taille, de la matrice $\acute{\boldsymbol{K}}_k$. La réduction de taille de la matrice du noyau régularisé est réalisée uniquement lorsque la taille maximale du dictionnaire M est atteinte. Donc, l'inverse de la matrice réduite est obtenue facilement en fonction de la matrice $\acute{\boldsymbol{K}}_k^{-1}$ dans l'équation (3.23). Pour obtenir l'inverse de la matrice réduite $\overline{\boldsymbol{K}}_k$, la matrice $\acute{\boldsymbol{K}}_k$ est reformulée comme suit :

$$\acute{\boldsymbol{K}}_k = \begin{pmatrix} \kappa\big(\boldsymbol{x}(k-M), \boldsymbol{x}(k-M)\big) + \gamma^{-1} & \widehat{\boldsymbol{\kappa}}_k \\ \widehat{\boldsymbol{\kappa}}_k^T & \overline{\boldsymbol{K}}_k \end{pmatrix} \tag{3.25}$$

où la valeur M représente la taille maximale du dictionnaire avec :

$\widehat{\boldsymbol{\kappa}}_k = (\kappa(\boldsymbol{x}(k), \boldsymbol{x}(k-M)), \dots, \kappa(\boldsymbol{x}(k), \boldsymbol{x}(k-1)))^T$.

Ensuite, l'inverse $\acute{\boldsymbol{K}}_k^{-1}$ dans l'équation (3.23) est formalisé par la division suivante :

$$\acute{\boldsymbol{K}}_k^{-1} = \begin{pmatrix} q & \boldsymbol{E^T} \\ \boldsymbol{E} & \boldsymbol{V} \end{pmatrix} \tag{3.26}$$

Enfin, la matrice inverse $\overline{\boldsymbol{K}}_k^{-1}$ est obtenue en multipliant les matrices dans (3.25) et (3.26), et après quelques simplifications la matrice $\overline{\boldsymbol{K}}_k^{-1}$ est calculée par :

$$\overline{\boldsymbol{K}}_k^{-1} = \boldsymbol{V} - \frac{\boldsymbol{E}\boldsymbol{E^T}}{q} \tag{3.27}$$

et la solution de régression de l'équation (3.10) peut être exprimée par :

$$\boldsymbol{\alpha}_k = \overline{\boldsymbol{K}}_k^{-1} \boldsymbol{y}_k \tag{3.28}$$

Après avoir adapté les coefficients d'expansion, les paramètres des conséquences $\widehat{\boldsymbol{\theta}}_l, l = 1, \dots, N_r$ sont mis à jour en utilisant uniquement M paires des données entrées-sorties telles que :

$$\widehat{\boldsymbol{\theta}}_l = \sum_{i=1}^{M} \alpha_i . (\boldsymbol{x}_i) . exp\left\{ - \sum_{j=1}^{n} \frac{\left(x_j - m_{lj}\right)^2}{2{\sigma_{lj}}^2} \right\}, l = 1, \dots, N_r \tag{3.29}$$

La méthode des moindres carrés récursifs à noyau avec une fenêtre coulissante est résumée comme suit :

Algorithme

1. Calculer l'erreur $e_k(k)$
2. Calculer la matrice inverse donnée par l'équation (3.23)
3. Si la taille de dictionnaire est plus que M Alors
   (1) Calculer la matrice inverse donnée par l'équation (3.26)
   (2) Calculer the matrice donnée par l'équation (3.25)
   (3) Calculer la matrice inverse $\overline{\boldsymbol{K}}_k^{-1}$ donnée par l'équation (3.27)
   (4) Calculer les coefficients d'expansion donnés par l'équation (3.28)
4. Sinon :
   (1) ajuster les coefficients d'expansion donnés par l'équation (3.24)

Algorithme 3.3 : L'algorithme des moindres carrés récursifs à noyau avec une fenêtre coulissante

Enfin, l'adaptation en ligne du modèle floue de type TS-KRR est résumée par l'algorithme (3.4) comme suit :

Algorithme

1. Initialisation : la taille de vecteur d'entrée, nombre des groupes N_r, .. etc.
2. Réaliser l'algorithme de K-moyennes pour initialiser les paramètres d'antécédents (hors ligne).
3. choisir la taille du dictionnaire M.
4. Mesurer le signal de sortie.
5. Ajuster les paramètres des antécédents m_{ij}, σ_{ij} dans les équations (3.17) et (3.18), respectivement.
6. Réaliser l'algorithme (3.3) pour obtenir les coefficients d'expansion.
7. Obtenir les paramètres des conséquences $\widehat{\boldsymbol{\theta}}_l$ donné par l'équation (3.29), et le modèle flou représenté par l'équation (3.9).
8. Retourner à l'étape 4

Algorithme 3.4 : Identification en ligne du modèle floue de type TS-KRR

Après avoir présenté le modèle floue de type TS-KRR (les deux versions du modèle TS-KRR : en ligne et hors ligne), la section suivante explique comment le modèle TS-KRR est intégré dans la commande GPC afin de contrôler les systèmes non linéaires.

3.6 Commande Prédictive Floue Basée Sur la Méthode de Régression Ridge à Noyau

La commande GPC appartient à la classe des méthodes de commande discrète à base de modèle. Cette technique de commande est capable de contrôler des procédés à phase non minimale, instables en boucle ouverte ou même les systèmes avec un temps mort variable (ou inconnu). La commande GPC, a été développée pour des systèmes ayant des modèles linéaires [4], mais la nature de la majorité des systèmes qui sont en général non linéaire conduit à proposer une commande GPC non linéaire. Généralement, le développement de modèles mathématiques pour décrire les comportements dynamiques des procédés non linéaire est très difficile, et parfois il n'existe pas un modèle correct pour représenter de façon général les non-linéarités des systèmes. C'est là que les modèles flous de type TS peuvent intervenir en tant qu'approximateur universel pour résoudre le problème de modélisation non linéaire, ce qui conduit à la commande prédictive généralisée floue (FGPC). Dans ce travail, on va présenter une commande prédictive généralisée floue basée sur un modèle floue de type TS-KRR où le système non linéaire sera décrit par le modèle flou TS-KRR alors que la loi de commande du GPC sera utilisée pour calculer le signal de commande.

Un système non linéaire est considéré comme ayant la structure non linéaire générale suivante:

$$y(k) = f\Big(y(k-1), y(k-2), \dots, y\big(k-n_y\big), u(k-d), u(k-d-1), \dots, u(k-n_u)\Big) \tag{3.30}$$

avec y et u sont respectivement le sortie et le signal des commandes du système, f est une fonction non linéaire qui décrit la relation entre les entrées et les sorties du système, cette fonction est supposée inconnue, n_y et n_u sont respectivement les ordres du signal de commande et de la sortie de système.

L'idée principale de la commande GPC floue est que la fonction non linéaire $f(*)$ est approximée en utilisant plusieurs modèles linéaires locaux où le système dans équation (3.30) peut être décrit par les règles floues suivantes de type TS :

$$\begin{aligned} R_i: \quad & \text{Si } x_1(k) \; est \; A_1^i, et \; \dots, et \; x_n(k) \; est \; A_n^i \\ & \text{Alors } y_i(k) = a_i(z^{-1})y(k-1) + b_i(z^{-1})u(k-d-1) \\ & i = 1, \dots, N_r \end{aligned} \tag{3.31}$$

où le vecteur d'entrée est donnée par: $x(k) = \big(y(k-1), \dots, y(k-n_a), u(k-d-1), \dots, u(k-n_b)\big)$, $a_i(z^{-1})$ et $b_i(z^{-1})$ sont des polynômes linéaires avec $n_a \leq n_y$ et $n_b \leq n_u$, tel que :

$$\begin{aligned} a_i(z^{-1}) &= a_i^1 z^{-1} + \cdots + a_i^{n_a} z^{-n_a} \\ b_i(z^{-1}) &= b_i^0 + b_i^1 z^{-1} + \cdots + b_i^{n_b} z^{-n_b} \end{aligned} \tag{3.32}$$

Donc, le modèle flou de type TS-KRR est donnée par :

$$\begin{aligned} \hat{y}(k) = \sum_{i=1}^{N_r} \big(& a_i(z^{-1})y(k-1) \\ & + b_i(z^{-1})u(k-d-1)\big). exp\left\{ - \sum_{j=1}^{n} \left(\frac{\big(x_j - m_{ij}\big)^2}{2{\sigma_{ij}}^2} \right) \right\} \end{aligned} \tag{3.33}$$

avec $\widehat{\boldsymbol{\theta}}_i = (a_i, \; b_i) \; i = 1, \dots, N_r$ qui peut être obtenu par l'équation (3.8) (ou équation 3.29 dans le cas d'identification en ligne de modèle flou). Le modèle flou dans l'équation (3.33) peut être réécrit dans une représentation de la forme CARIMA :

$$\bar{A}(z^{-1})y(k) = \bar{B}(z^{-1})u(k-d-1) + \frac{\xi(k)}{\Delta} \tag{3.34}$$

où $\bar{A}(z^{-1}) = 1 - \bar{a}^1 z^{-1} - \cdots - \bar{a}^{n_a} z^{-n_a}$ et $\bar{B}(z^{-1}) = \bar{b}^0 + \bar{b}^1 z^{-1} + \cdots + \bar{b}^{n_b} z^{-n_b}$, avec:

$$\begin{aligned} \bar{a}^j &= \sum_{i=1}^{N_r} a_i^j . exp\left\{ - \sum_{j=1}^{n} \left(\frac{\big(x_j - m_{ij}\big)^2}{2{\sigma_{ij}}^2} \right) \right\} \\ \bar{b}^j &= \sum_{i=1}^{N_r} b_i^j . exp\left\{ - \sum_{j=1}^{n} \left(\frac{\big(x_j - m_{ij}\big)^2}{2{\sigma_{ij}}^2} \right) \right\} \end{aligned} \tag{3.35}$$

Après avoir défini le modèle flou de type TS-KRR dans une représentation de modèle CARIMA, le calcul de la prédiction et la loi de commande de méthode GPC sont présentés dans l'annexe A. selon la méthode d'indentification utilisée pour faire l'apprentissage du modèle floue de type TS-KRR, nous avons distingué deux méthodes de la commande GPC floue (hors ligne et adaptive). Tout d'abord, nous présentons la commande GPC floue lorsque les données d'entrées-sorties du système existent et suffisent pour réaliser une identification hors ligne. Ensuite, nous discuterons de la commande GPC floue lorsque les données ne sont pas suffisantes pour avoir

une bonne représentation du système, ce qui conduit à l'utilisation d'une identification en ligne du modèle floue.

3.6.1 La commande prédictive généralisée floue avec une identification hors ligne

Généralement, l'identification hors ligne de modèle floue de type TS-KRR peut être suffisante pour modéliser le système non linéaire lorsque des données d'entrées-sorties sont disponibles. Dans ce cas, l'objectif est de proposer une stratégie de commande prédictive non linéaire à base de modèle TS-KRR. En premier lieu, le modèle floue de type TS-KRR est entraîné hors ligne, c'est-à-dire que les paramètres du modèle sont pris à la fin de l'application de toutes les données des couples d'apprentissage, pour avoir un modèle flou fixe. Ensuite, celui-ci utilisé en ligne pour donner les prédictions de la sortie à chaque instant d'échantillonnage. La fonction de coût de la commande prédictive est aussi optimisée en ligne à chaque itération comme indiqué dans l'annexe A.

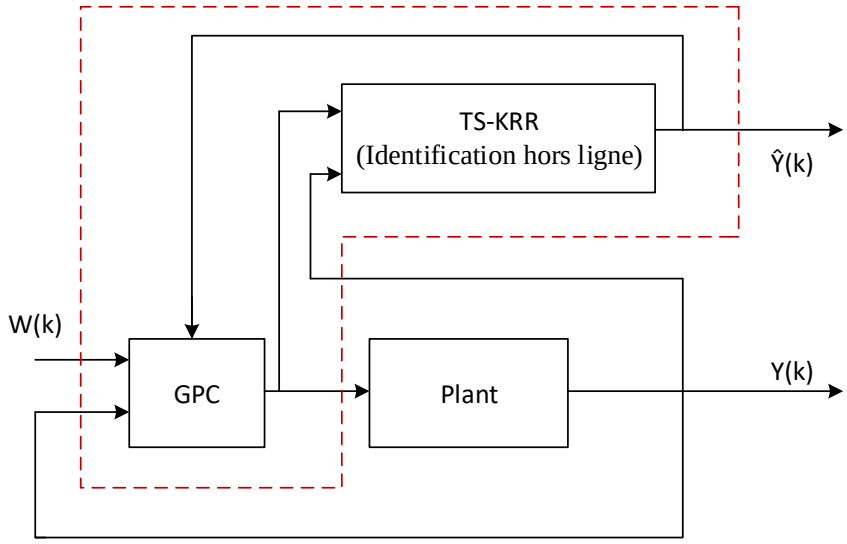

Figure 3.2 : Schéma de la commande prédictive généralisée à base de modèle TS-KRR avec une identification hors ligne (TS-KRR GPC)

Le schéma de la commande GPC non linéaire à base de modèle floue de type TS-KRR est donné sur la figure (3.2). Il comporte quatre composantes: le procédé à commander (plante), un modèle de référence $W(k)$ spécifiant les performances désirées du système, un modèle flou de

type TS-KRR, qui est entraîné hors ligne et modélise le système, et enfin la commande GPC linéaire qui fait la minimisation de la fonction de coût pour déterminer la commande nécessaire pour produire les performances désirées du système. L'algorithme TS-KRR GPC comprend le bloc GPC et le bloc de modèle TS-KRR. La procédure de la commande TS-KRR GPC (avec une identification hors ligne) est résumée comme suit :

Algorithme

1. Spécifier le signal de référence $W(k)$.
2. sélectionner les paramètres d'identification tels que: nombre de règles floues, les paramètres de PSO et les paramètres de GPC (N_p, N_u et ϑ).
3. Réaliser l'Algorithme (3.2) pour obtenir les paramètres des antécédents et des conséquences, et puis obtenez $\bar{A}(z^{-1})$ et $\bar{B}(z^{-1})$.
4. L'incrément du signal de commande $\Delta u(k)$ est calculé en utilisant l'équation (A.15).
5. Le signal de commande $\Delta u(k) + u(k-1)$ est appliqué au système.
6. Mesurer le signal de sortie.
7. Répéter les étapes 4, 5 et 6.

Algorithme 3.5 : La commande TS-KRR GPC

3.6.2 La commande prédictive généralisée floue avec une identification en ligne

Dans cette section, l'objectif est d'utiliser une commande prédictive généralisée linéaire pour contrôler un système ayant un comportement dynamique non linéaire, où les données d'apprentissage ne suffisent pas à décrire toutes les zones des fonctionnements des systèmes non linéaires. L'adaptation en ligne est alors nécessaire pour obtenir un modèle flou capable de poursuivre le système dans son évolution. Le modèle floue de type TS-KRR est entraîné en ligne, c'est-à-dire que les paramètres du modèle sont ajustés à chaque instant d'échantillonnage. Le modèle est donc utilisé en ligne pour donner les prédictions de la sortie à chaque instant. La fonction de coût de la commande prédictive est aussi optimisée en ligne à chaque itération. Le schéma de la commande TS-KRR GPC adaptative est donné par la figure (3.3).

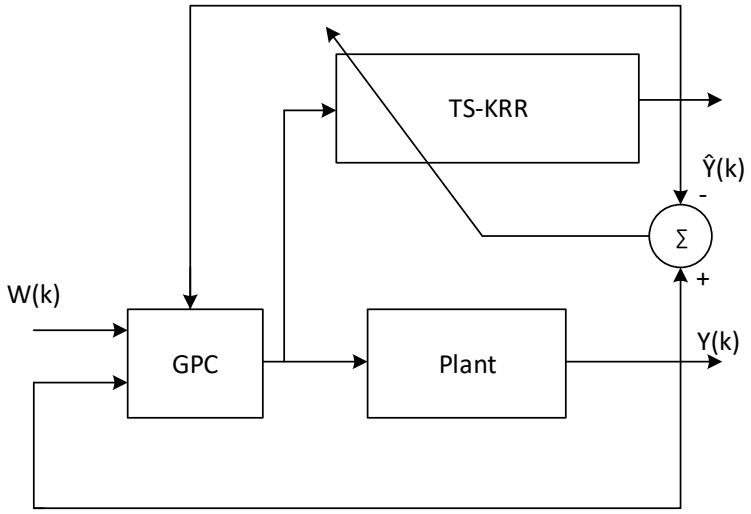

Figure 3.3 : Le schéma du la commande TS-KRR GPC adaptative

La commande TS-KRR GPC adaptatif est résumée comme suit :

Algorithme

1. Spécifier le signal de référence $W(k)$.
2. Sélectionner les paramètres d'identification tels que: nombre de règles floues et les paramètres de GPC (N_p, N_u et ϑ).
3. Réaliser la méthode de K-moyennes pour initialisée les paramètres des antécédentes.
4. Mesurer le signal de sortie.
5. Calculer les paramètres : m_{ij}, σ_{ij} et η.
6. Appliquer Algorithme 3.3 pour obtenu les coefficients d'expansion et ensuite calculer les paramètres conséquences $\widehat{\boldsymbol{\theta}}$ dans l'équation (31), et puis obtenez $\bar{A}(z^{-1})$ et $\bar{B}(z^{-1})$.
7. L'incrément du signal de commande $\Delta u(k)$ est calculé en utilisant l'équation (A.15).
8. Le signal de commande $\Delta u(k) + u(k-1)$ est appliqué au système.
9. Répéter les étapes 4 - 8.

Algorithme 3.6 : La commande TS-KRR GPC adaptatif

Chapitre 4

Résultats de Simulation

4.1 Introduction

Dans ce chapitre nous présentons les résultats de simulation obtenus en utilisant les approches proposées dans le chapitre 3. Deux systèmes sont étudiés, la commande de niveau d'un réservoir et la commande de concentration d'un réacteur exothermique. Notez que toutes les simulations d'identification et des commandes ont été effectuées en utilisant Matlab, et sur un ordinateur portable d'un processeur Intel Core Duo à 2.6 GHz.

4.2 Exemple 1 : identification et contrôle d'un système de réservoir

Considérons le réservoir décrit par le schéma de la figure (4.1) [64, 85].

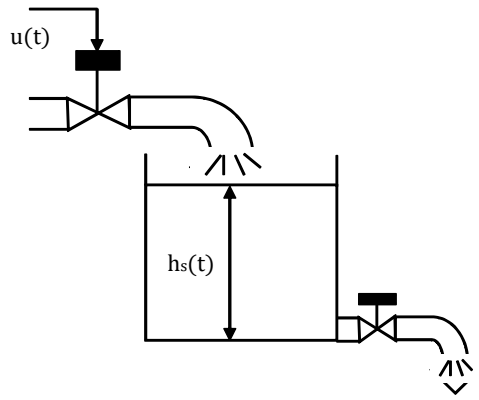

Figure 4.1 : Le réservoir

Le modèle du système est donné par l'équation non linéaire suivante :

$$\frac{dh_s(t)}{dt} = \frac{-c_s\sqrt{2g_s h_s(t)}}{A(h_s(t))} + \frac{1}{A(h_s(t))}u(t) \tag{4.1}$$

où $h(t)$ est le niveau de liquide dans le réservoir, $u(t)$ est le débit à l'entrée qui est utilisé pour contrôler le niveau de liquide dans le réservoir, les constants : $c_s = 1$ et $g_s = 9.8$, $A(h_s(t))$ est la surface de la base du réservoir qui est donnée par : $A\big(h_s(t)\big) = a_s\big(h_s(t)\big)^2 + b_s$, avec les constants: $a_s = 0.01$ et $b_s = 0.2$.

Dans la première partie de cette étude, le modèle flou de type TS-KRR sera utilisé pour faire une identification hors ligne du système.

L'objectif de cette étude est donc d'analyser les performances de cette méthode lorsqu'un échantillon d'apprentissage suffisant est disponible pour décrire le comportement dynamique du système. En outre, les effets des méthodes de partitionnement sur la performance de modèle TS-KRR seront analysés lorsque plusieurs méthodes des partitionnements sont utilisées pour calculer les paramètres des antécédents. Enfin, le modèle sera intégré dans la commande GPC pour contrôler le niveau de liquide dans le réservoir.

4.2.1 Identification hors ligne du système

La procédure d'identification hors ligne du système est réalisée par un ensemble de données d'apprentissage généré à partir du modèle mathématique. La période d'échantillonnage du système est fixée à $t_s = 0.1$ seconde et le modèle mathématique présenté dans l'équation (4.1) est utilisé pour générer un ensemble de 900 ($N_d = 900$) couples des données d'entrées-sorties. Les premiers 400 échantillons de ces couples sont utilisés pour construire le modèle TS-KRR (donnée d'apprentissage), alors que le reste de ces échantillons (500 échantillons) ont été utilisés pour tester la validation du modèle flou. Le signal de commande représenté dans la figure (4.2) a été utilisé pour générer ces 900 échantillons, et la taille du vecteur d'entrée est fixée à 4 ($n_a = 3$ et $n_b = 1$).

Avant de commencer la procédure d'identification, nous expliquerons quelques idées concernant l'identification du nombre de groupes à partir des données d'apprentissage qui est une étape importante, car les centres et les variances des groupes sont ceux des fonctions d'appartenances du modèle TS-KRR, et le nombre de règles du modèle TS-KRR est le nombre de groupes. En outre, il est aussi important de mentionner qu'avec un grand nombre de règles floues, la performance du modèle flou sera meilleure et plus précise; cependant, cela pourrait causer une énorme charge de calcul, qui peut être évitée par la détermination d'un nombre approprié de groupes dans l'ensemble de données.

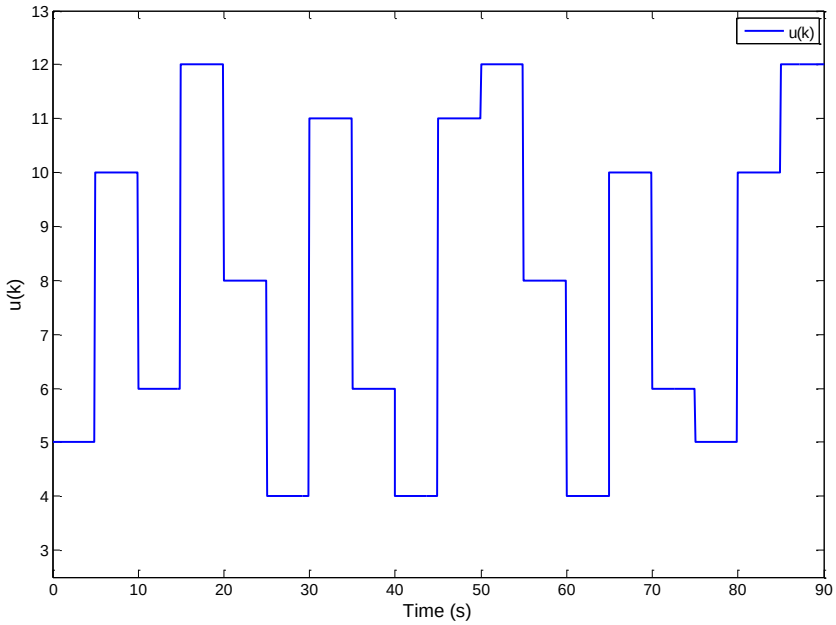

Figure 4.2 : Le signal de commande utilisée pour générer l'ensemble de 900 échantillons

Généralement, il y a plusieurs méthodes pour valider le nombre de groupes à partir des données. Mendes et al. [86] a défini le nombre de groupes soit égal au nombre des hyperplans (points de fonctionnement) du système décrit par les données d'entrées-sorties. Cette méthode n'est pas très précise car un hyperplan du système peut être répété plusieurs fois dans le même ensemble de données. Dans cette thèse, le nombre des groupes de partitionnement peut être identifié comme suit : le nombre des groupes est initialisé au nombre des hyperplans. Ensuite, la méthode de K-moyennes est appliquée pour obtenir les centres des groupes. Enfin, les ressemblances entre ces groupes sont éliminées car deux groupes (ou plus) peuvent être décrits le même hyperplan du système, et le nombre des groupes sera réduit si des similarités sont détectées. Cette procédure sera répétée jusqu'à ce que les similarités entre les groupes n'existent pas. Étant donné que la forme des données d'apprentissage est très simple (voire figures 4.2), l'approche précédente peut être facilement appliquée. Ainsi, le nombre obtenu de groupes (nombre de règles floues) est $N_r = 7$.

Dans le cas où les données d'entrées-sorties sont dans une forme complexe dans laquelle les hyperplans sont impossibles à identifier, l'analyse de validité du nombre des groupes est réalisée en exécutant la méthode de partitionnement pour différentes valeurs de N_r, à plusieurs reprises

pour chaque N_r avec une initialisation différente (si cela est possible). Ensuite, l'indice de Dunn [87, 88] est calculé pour chaque exécution de la méthode de partitionnement et le nombre de groupes qui maximise l'indice de Dunn (voir l'annexe C pour plus de détails sur l'indice de Dunn) est alors choisi comme étant le nombre "adéquat" de groupes pour l'ensemble des données. Pour tester cette stratégie, le nombre de groupes devrait changer entre 2 et 16. Ensuite, l'algorithme K-moyennes est exécuté pour tous les nombres possibles de groupes et l'indice Dunn est calculé. Donc, le nombre convenable de groupes est sélectionné en fonction de la valeur la plus élevée de l'indice de Dunn (voir la valeur maximale de cet indice dans la figure 4.3). Il est clair que la méthode basée sur l'indice de Dunn a obtenu le même nombre de groupes.

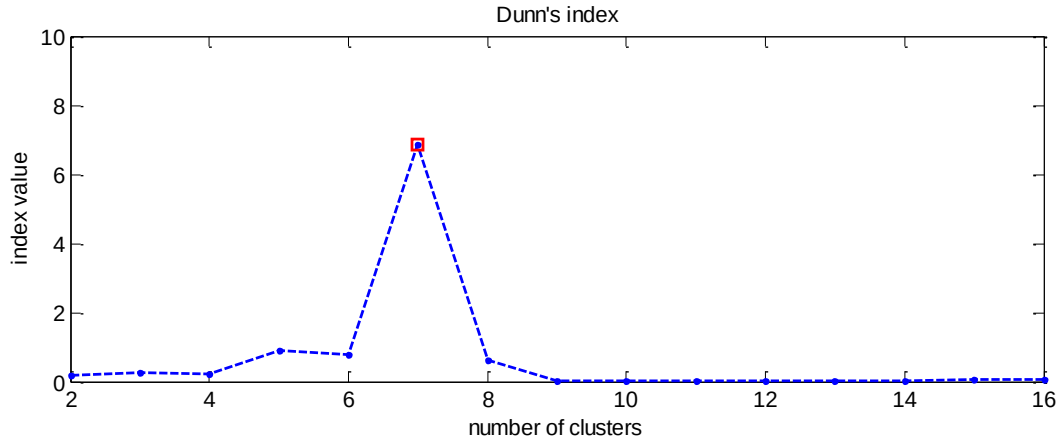

Figure 4.3 : La variation de l'indice de Dunn

Enfin, la méthode de partitionnement basée sur l'algorithme de PSO est utilisée pour calculer les centres et les variances des fonctions d'appartenances. Le nombre de particules utilisées dans l'algorithme PSO est 16 alors que les paramètres de PSO : w_0, c_1, c_2 et$iter_max$ sont mis respectivement à $0.65, 1.7, 1.7$ et 100.

Pour initialiser les 16 particules de PSO, on exécute l'algorithme K-moyennes et les résultats obtenus seront considérés comme les premières particules (stocker dans $\breve{\boldsymbol{x}}_1$). Le reste des particules sont choisies d'une façon aléatoire où les limites maximale et minimale de ces particules sont définies respectivement comme : $\breve{\boldsymbol{x}}_{max} = \breve{\boldsymbol{x}}_1 + 0.25 \times \breve{\boldsymbol{x}}_1$ et $\breve{\boldsymbol{x}}_{min} = \breve{\boldsymbol{x}}_1 - 0.25 \times \breve{\boldsymbol{x}}_1$. Pour résoudre le problème quadratique dans l'équation (1.6), la constante λ est fixée à $0{,}001$, et après avoir exécuté les étapes d'identification hors ligne du modèle TS-KRR dans

l'algorithme (3.2), les résultats d'identification du niveau de liquide dans le réservoir $h_s(t)$ sont présentés dans la figure (4.4).

La performance de partitionnement basé sur l'algorithme de PSO est évaluée en comparant ses résultats d'identification avec les résultats d'identification obtenus par deux autres algorithmes de partitionnements : la méthode standard de K-moyennes et une méthode de partitionnement basé sur l'algorithme génétique (GA). Les paramètres de GA sont : nombre de population égale à 16, le taux de sélection est 0.5, le taux de croisement est 0.7, le taux de mutation est 0.2 et les mêmes limites maximales et minimales de PSO sont utilisées pour initialiser la population de GA. Enfin, Pour réaliser la méthode de partitionnement basée sur GA, les mêmes étapes dans l'Algorithme (3.1) sont exécutées sauf l'étape 3, où les solutions globales et locales sont ajustées à l'aide des opérateurs de GA suivants : la sélection, la mutation et le croisement.

La figure (4.4) montrer les résultats d'identification lorsque trois algorithmes de partitionnements sont utilisés pour calculer les paramètres des antécédents de modèle TS-KRR. Les résultats d'identification sont désignés par : TS-KRR (PSO), TS-KRR (GA) et TS-KRR (K-means). De plus, les résultats des deux autres méthodes d'identification hors ligne : le modèle flou de type Takagi–Sugeno basé sur la SVR (*Takagi–Sugeno Fuzzy System-based Support Vector Regression* : TSFS-SVR) [27] et le système d'inférence floue à base de réseaux neuronaux généralisés (*Generalized neural networks based fuzzy inference system* : GNN-FIS) [89] sont aussi présentés dans la figure (4.4). Les résultats montrent que le modèle flou de type TS-KRR (dans tous les cas) à une bonne performance et la précision de la modélisation est meilleure que les méthodes de GNN-FIS et TSFS-SVR. En outre, TS-KRR (PSO), TS-KRR (GA) et TS-KRR (K-means) montrent des résultats similaires ce qui est attendu en raison de la simplicité des données d'apprentissage.

Pour avoir une comparaison efficace entre la performance de modèle de type TS-KRR et les performances des méthodes : GNN-FIS et TSFS-SVR, nous proposons l'utilisation d'erreur quadratique moyenne (*Root Mean Square Error* : RMSE), l'erreur absolue moyenne (*Mean Absolute Error* : MAE), l'erreur absolue moyenne en pourcentage (*Mean Absolute Percentage Error* : MAPE) et l'erreur de pourcentage moyenne absolue symétrique (*symmetric Mean*

Absolute Percentage Error : sMAPE) comme des mesures pour valider par la simulation l'utilisation du modèle floue de type TS-KRR comme un outil de prédiction.

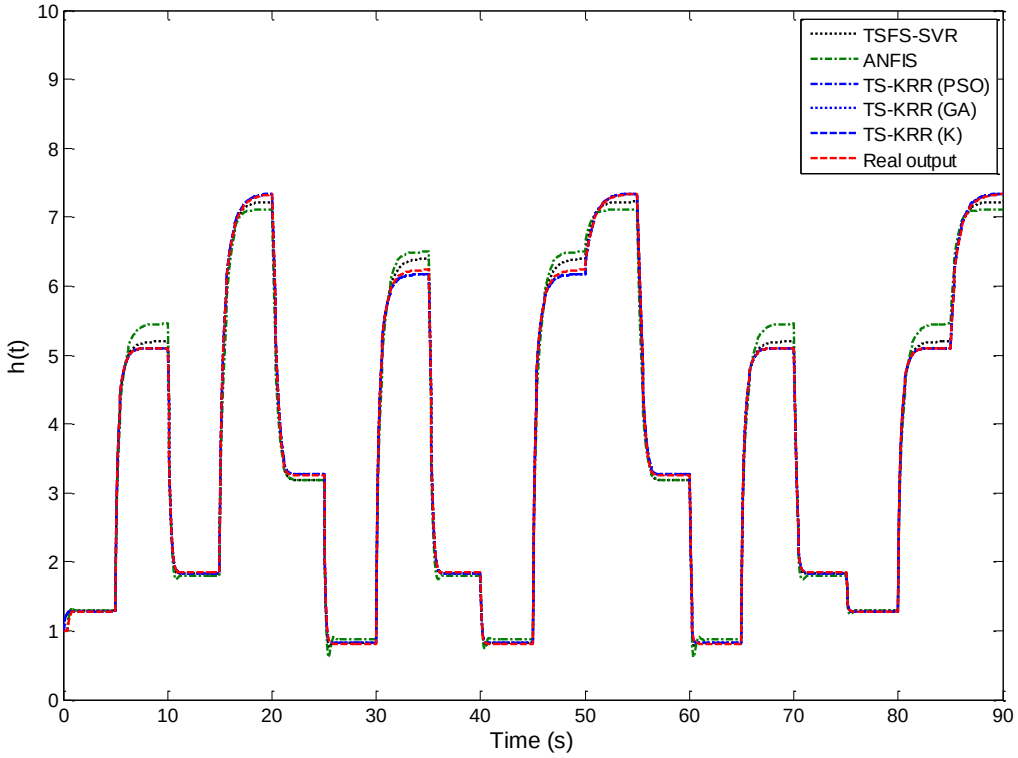

Figure 4.4 : La performance de modélisation à partir de modèle floue de type TS-KRR

Le RMSE, bien que dépendant de l'échelle de la série de données d'apprentissage et validation, reste la mesure d'erreur la plus utilisée. L'identification hors ligne est d'autant meilleure que le RMSE calculé est faible :

$$RMSE = \sqrt{\frac{1}{N_d}\sum_{k=1}^{N_d}(y(k) - \hat{y}(k))^2} \tag{4.2}$$

Le MAE qui représente la moyenne arithmétique des valeurs absolues des erreurs entre les valeurs réelles de la sortie du système et les valeurs obtenues par le modèle flou :

$$MAE = \frac{1}{N_d}\sum_{k=1}^{N_d}|y(k) - \hat{y}(k)| \tag{4.3}$$

Le MAPE permet de s'affranchir de l'ordre de grandeur des données mais cette mesure est préférable lorsque les valeurs sont proches de 0 :

$$MAPE = \frac{100}{N_d}\sum_{k=1}^{N_d}\left|\frac{y(k)-\hat{y}(k)}{y(k)}\right| \tag{4.4}$$

Enfin, Le sMAPE, qui est aussi une mesure de la précision basée sur les erreurs de pourcentage, est donnée par :

$$sMAPE = \frac{200}{N_d}\sum_{k=1}^{N_d}\frac{|y(k)-\hat{y}(k)|}{|y(k)|+|\hat{y}(k)|} \tag{4.5}$$

Les résultats de comparaison sont présentés dans le tableau (4.1) :

Tableau 4.1 : Résultats de comparaison pour identifier le niveau dans un réservoir

Méthodes	Nombre des règles	Nombre des entrées	RMSE	MAE	MAPE (%)	sMAPE (%)	La durée du simulation (s)
TS-KRR (PSO)	07	$n_a = 3$, $n_b = 1$	0.0039194	0.0018413	0.04609	0.04621	12.9901
TS-KRR (GA)	07	$n_a = 3$, $n_b = 1$	0.0039194	0.0018413	0.04609	0.04621	13.4612
TS-KRR (k-means)	07	$n_a = 3$, $n_b = 1$	0.0039219	0.0018427	0.04611	0.04628	07.1240
GNN-FIS	07	$n_a = 2$, $n_b = 2$	0.4597423	0.2167553	5.9112	5.8145	00.7088
GNN-FIS	20	$n_a = 2$, $n_b = 2$	0.1812871	0.0947221	3.0451	3.1132	00.8104
TSFS-SVR	07	$n_a = 3$, $n_b = 1$	0.1031247	0.0561423	1.0244	0.9846	48.1437
TSFS-SVR	14	$n_a = 3$, $n_b = 1$	0.0787312	0.0354782	0.8247	0.8098	52.0758

Les modèles TS-KRR (PSO) et TS-KRR (GA) produiront les mêmes valeurs d'erreurs (RMSE, MAE, MAPE et sMAPE), alors que les valeurs d’erreurs obtenues par TS-KRR (K-means) sont un peu différentes (voir tableau 4.1). Ceci indique que l'algorithme K-moyennes

réalise efficacement un partitionnement des données d'apprentissage et peut être utilisé pour calculer les centres et les variances des fonctions d'appartenances sans aucune modification.

Dans cette simulation, les progrès apportés par les algorithmes de PSO et GA ne sont pas significatifs en raison de la simplicité des données, la méthode de K-moyennes est suffisante pour calculer les paramètres des antécédents. D'autre part, l'approche TS-KRR a donné de meilleures résultats que les méthodes de GNN-FIS et TSFS-SVR, ce qui peut être vu dans le tableau (4.1) où le modèle de type TS-KRR requiers moins de règles floues (seulement 7 règles pour le TS-KRR) que les deux autres méthodes. Comme on peut le voir dans le tableau (4.1), les méthodes de GNN-FIS et TSFS-SVR requièrent plus des règles floues pour générer des résultats relativement acceptables. En outre, les valeurs des erreurs (les valeurs de RMSE, MAE, MAPE et sMAPE) obtenues par la prédiction du modèle TS-KRR sont plus petites que celles obtenues par les méthodes de GNN-FIS et TSFS-SVR. Bien que le modèle de GNN-FIS ait relativement une mauvaise performance (RMSE = 0.1813, MAE = 0.0947, MAPE = 3.0451% et sMAPE = 3.1132%), cet algorithme est plus rapide que les méthodes de TS-KRR et TSFS-SVR où le temps de simulation du modèle GNN-FIS était inférieur à 1 seconde.

4.2.2 La commande prédictive floue du système

Généralement, le modèle flou hors ligne de type TS-KRR pourrait être suffisant pour identifier le système non linéaire lorsque des données suffisantes sur ce système sont disponibles. Dans ce cas, la commande GPC peut être implémentée en remplaçant le système par le modèle TS-KRR déterminé hors ligne lorsque le signal de commande est calculé.

Pour implémenter le commande TS-KRR GPC avec une identification hors ligne, plusieurs paramètres sont sélectionnés : la taille du vecteur d'entrée qui est définie par ($n_a = 3$ et $n_b = 1$), et les paramètres de commande GPC qui sont : $N_p = 7$, $N_u = 2$ et $\vartheta = 30$. Le nombre de règles floues est égal à $N_r = 7$ et le reste des paramètres du modèle TS-KRR sont similaires à ceux discutés précédemment (les paramètres d'algorithme PSO et λ). Après l'exécution de l'algorithme (3.5), la sortie du système et le signal de commande appliqué au système sont illustrés respectivement dans les figures (4.5) et (4.6). La performance du commande TS-KRR GPC est analysée lorsque deux autres algorithmes de partitionnement sont utilisés pour calculer les centres et les variances des groupes. Les résultats de la simulation sont donnés sur les figures (4.5).

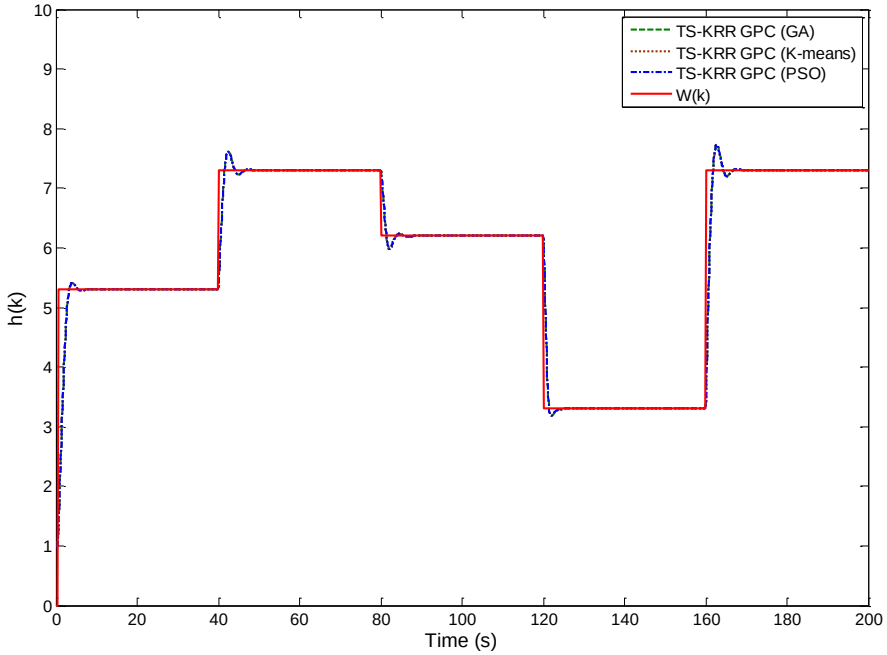

Figure 4.5 : Les résultats du commande TS-KRR GPC

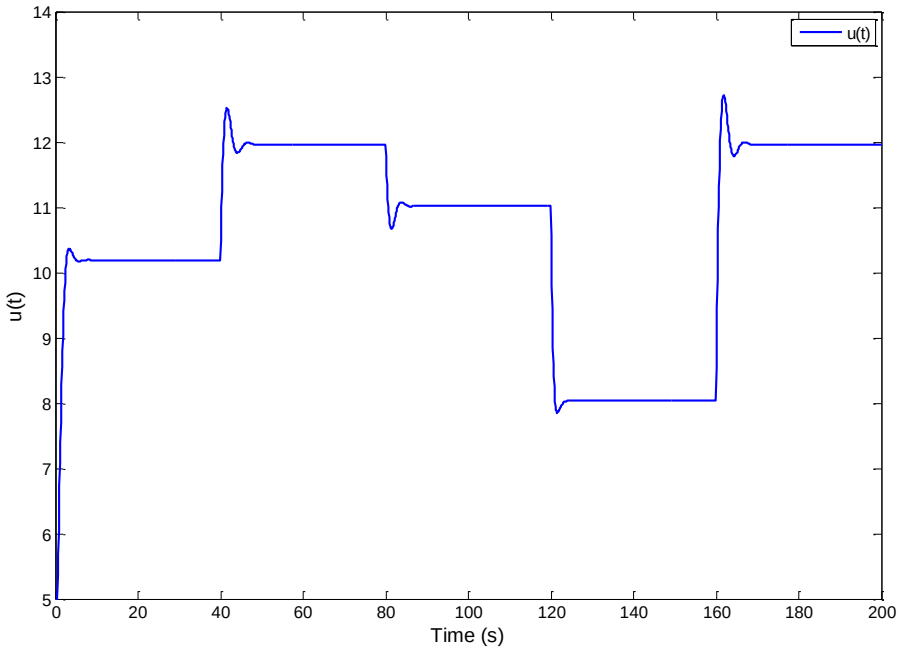

Figure 4.6 : Le signal de commande appliqué au système

Les résultats de la simulation montrent une bonne poursuite de la sortie du système à la référence $W(k)$. La méthode de commande proposée garantit de bonnes performances pour le système contrôlé. La commande GPC TS-KRR avec les trois méthodes de partitionnement a donné des résultats similaires en raison de la simplicité des données. Ceci conclut que la méthode de K-moyennes est suffisante pour calculer les paramètres des antécédents.

En général, la précision des modèles flous basés sur une identification hors ligne est limitée en raison des incertitudes associées aux les systèmes à commander. En outre, les données d'entrées-sorties utilisées pour faire l'apprentissage du modèle sont incomplètes et ne peuvent pas décrire tous les points de fonctionnements du procède. Aussi, le comportement dynamique du système peut changer de façon inattendue avec le temps. Dans ce cas, les méthodes d'adaptation sont considérée plus pratiques puisque les paramètres du modèle TS-KRR peuvent être ajustés pour améliorer les approximations. Dans la prochaine section, l'identification adaptative (en ligne) du modèle floue de type TS-KRR est analysée. La méthode K-moyennes sera utilisée pour initialiser les paramètres des antécédents du modèle floue puisque les améliorations apportés par l'algorithme PSO sont insignifiantes.

4.2.3 Identification en ligne du système

La procédure d'identification en ligne du système est réalisée par le même ensemble de données d'apprentissage généré à partir du modèle mathématique. Nous garderons les mêmes paramètres que ceux utilisés dans l'identification hors ligne du modèle. Le β du taux d'apprentissage est égal à 0.09 et la taille de dictionnaire $M = 50$. La méthode de K-moyennes est utilisée pour initialiser les paramètres des antécédents. Après avoir exécuté l'algorithme (3.4), la prédiction du niveau de liquide dans le réservoir est présentée dans la figure (4.7). La figure (4.7) contient aussi le signal original du niveau de liquide dans le réservoir $h(t)$, la sortie prédite par la méthode de ANFIS et le signal de sortie obtenu par un méthode adaptative de type Takagi-Sugeno-Kang (TSK) [16].

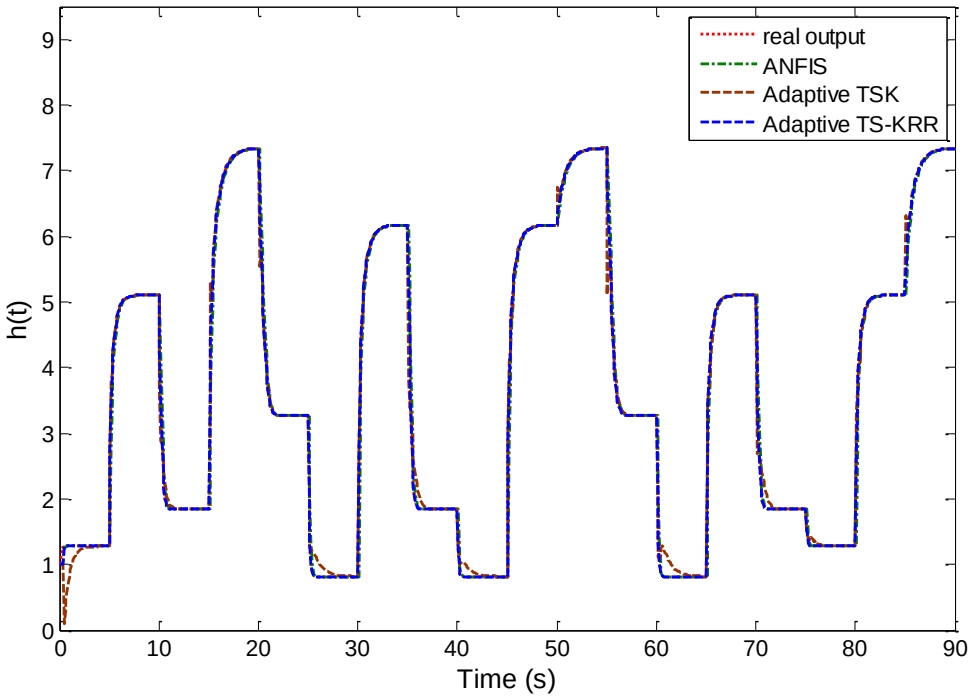

Figure 4.7 : La performance de modélisation à partir de modèle floue adaptatif de type TS-KRR

Comme l'illustre la figure (4.7), les trois modèles de prédiction ont bien fonctionné et la précision de leur modélisation est relativement satisfaisante. Le tableau (4.2) illustre les résultats de précision pour les trois méthodes dans lesquels les mesures d'erreur de : RMSE, MAE, MAPE et sMAPE sont calculées pour les trois méthodes. Évidemment, la performance du modèle adaptatif flou de type TS-KRR a surpassé les performances de celle de ANFIS et de TSK adaptatif où le modèle adaptatif de type TS-KRR requiert moins de règles floues (seulement 7 règles) alors que les deux autres méthodes requièrent 20 règles pour générer des bonnes prédictions. De plus, les valeurs d'erreur obtenues par TS-KRR adaptatif sont inférieures à celles obtenues par ANFIS et ATSK (voir tableau 4.2).

Tableau 4.2 : Résultats de comparaison pour identifier le niveau dans un réservoir

Méthodes	Nombre des règles	Nombre d'entrées	RMSE	MAE	MAPE (%)	sMAPE (%)
TS-KRR Adaptive	07	$n_a = 3$, $n_b = 1$	0.0347	0.0130	0.9030	0.8848
ANFIS	20	$n_a = 2$, $n_b = 2$	0.09864	0.0396	2.7151	2.7062
ATSK	20	$n_a = 2$, $n_b = 2$	0.18332	0.0752	5.1174	5.2018

4.2.4 La commande prédictive floue adaptative du système

Dans cette section, la commande TS-KRR GPC adaptative est utilisée pour contrôler le système non linéaire. Les paramètres du modèle TS-KRR adaptatif sont similaires à ceux présenté précédemment dans la partie d'identification en ligne, et les paramètres de commande GPC sont : $N_p = 7$, $N_u = 2$ et $\vartheta = 30$. Apres l'exécution de l'algorithme (3.6), la sortie du système et le signal de commande appliqué au système sont illustrés respectivement dans les figures (4.8) et (4.9). En outre, les solutions obtenu par la commande ANFIS GPC (les paramètres du contrôle de ANFIS GPC sont : $N_p = 7$, $N_u = 2$, $\vartheta = 30$ et le nombre de règles floues est $N_r = 20$) et la commande adaptative TSK GPC (les paramètres du contrôle de ATSK GPC sont : $N_r = 20$, $N_p = 7$, $N_u = 2$, $\vartheta = 30$ et $N = 20$) sont aussi illustrés dans la figure (4.8).

Les résultats de la figure (4.8) montrent que la sortie du système est totalement confondue avec la référence $W(k)$, ce qui montre que la méthode TS-KRR GPC adaptatif utilisée est bien adaptée au modèle du système. Les mêmes commentaires peuvent être faits pour les résultats obtenus par ANFIS GPC et ATSK GPC.

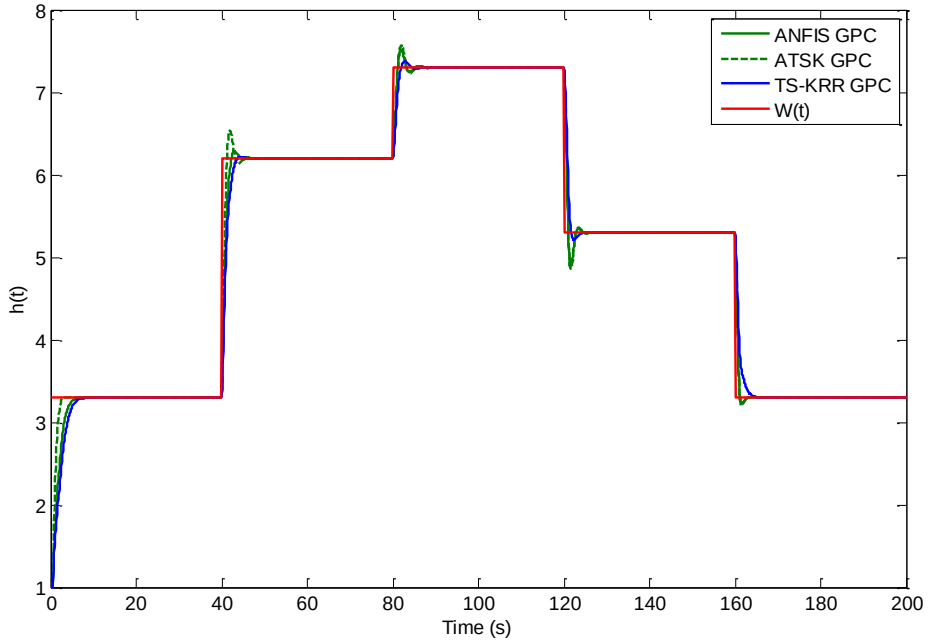

Figure 4.8 : Les résultats du commande TS-KRR GPC adaptatif

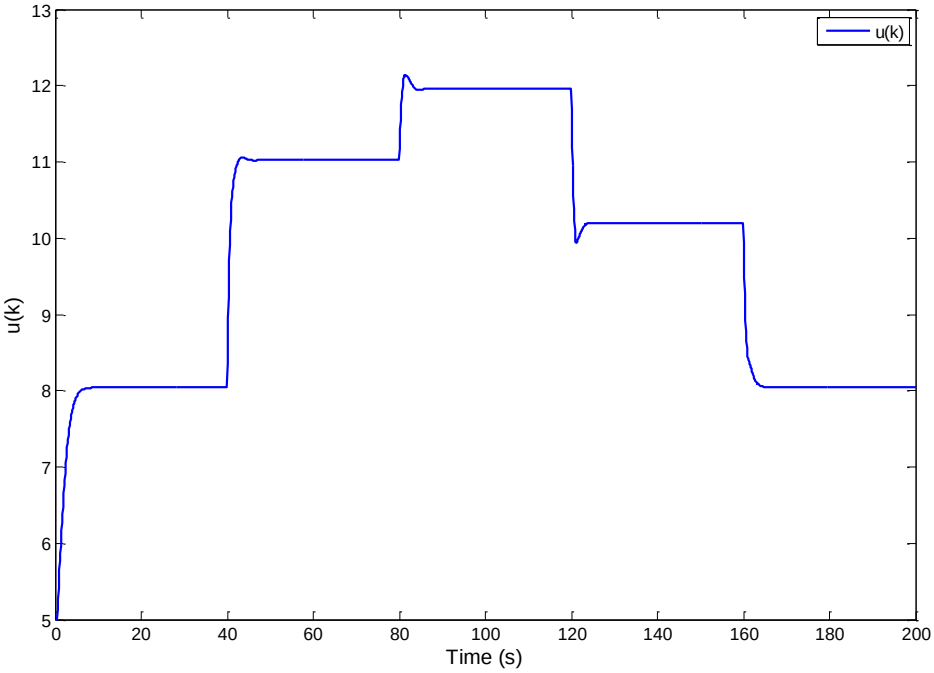

Figure 4.9 : Le signal de commande appliqué au système

Cependant, les commandes ANFIS GPC et ATSK GPC présentent des dépassements relativement élevés en régime dynamique du système lorsqu'en passe d'un niveau de référence à un autre, alors que la commande TS-KRR GPC adaptative présente des dépassements négligeables. La valeur moyenne pour exécuter une itération (l'exécution des étapes : 4, 5, 6, 7 et 8 dans l'algorithme 3.6) pour une taille de dictionnaire de $M = 50$ est 0.003552 seconde, ce qui est considéré comme une période de temps courte et s'adapte à la plupart des systèmes industriels (dans cet exemple $0.003552s \ll t_s = 0.1$ seconde). D'autre part, la commande ANFIS GPC a un temps d'exécution plus petit (0.000713 seconde) car son algorithme ne demande pas un grand nombre de données précédentes. Le même commentaire peut être fait pour la commande ATSK GPC où le temps d'exécution pour un échantillon est relativement petit (0.000689 seconde).

Afin de tester la robustesse de stabilité de cette méthode, nous avons injecté une perturbation au système contrôlé. Les perturbations appliquées au système sont définies comme des constantes où une perturbation avec l'amplitude de 1 a été appliquée au système dans l'intervalle de temps $20 \leq k \leq 60$ seconde. Une autre perturbation de l'amplitude -1 a été appliquée au système dans l'intervalle $60 < k \leq 110$ seconde. Enfin, une perturbation avec l'amplitude de 1 a été injectée sur le système dans l'intervalle $140 < k \leq 200$ seconde. L'objectif de la poursuite de trajectoire de référence est atteint en présence des défauts (perturbations) où les figures (4.10) et (4.11) montre les résultats de cette simulation.

Nous remarquons de plus que les performances obtenues avec la commande TS-KRR GPC adaptative sont meilleures que les autres commandes. Les perturbations ont été rapidement éliminées. En outre, la commande proposée présente de plus petits dépassements que les commandes prédictives basées sur les modèles ANFIS et ATSK. Généralement, les résultats de la simulation démontrent la capacité de la commande TS-KRR GPC adaptive à commander un tel processus malgré l'addition de bruit au système.

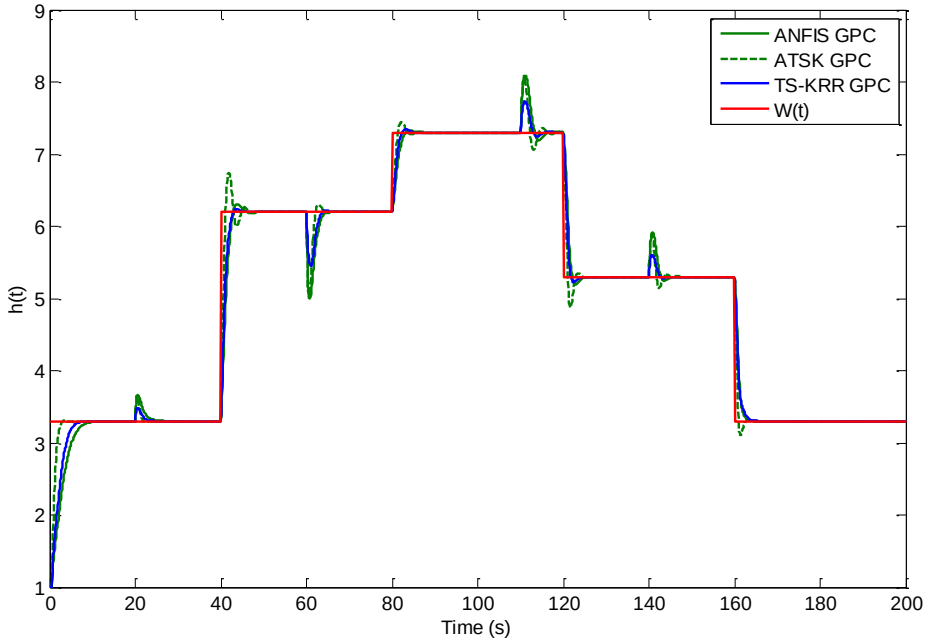

Figure 4.10 : Les résultats du commande TS-KRR GPC adaptatif

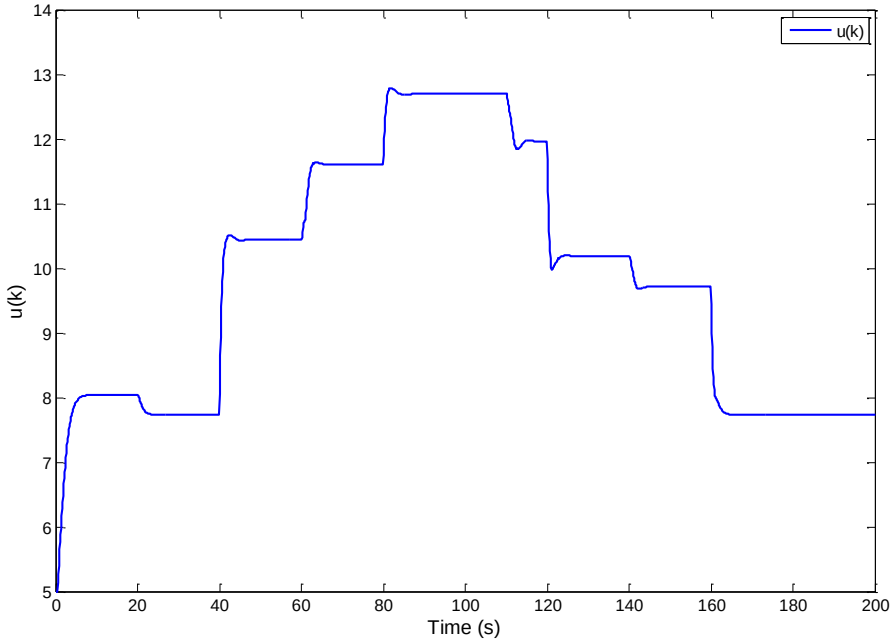

Figure 4.11 : Le signal de commande appliqué au système

4.3 Exemple 2 : identification et contrôle d'un réacteur exothermique

Dans cette section, nous considérons un exemple de contrôle de la concentration dans un réacteur chimique CSTR (Continuous Stirred Tank Reactor) [63, 64]. Ce système décrit le processus de décomposition d'un produit A_a en un autre produit B_b où $C_a(t)$ représente la concentration du produit A_a (la sortie du système), alors que $T(t)$ est la température dans le réacteur. La réaction, qui est décrite par un système non linéaire, est exothermique et le taux d'écoulement du liquide de refroidissement $q_c(t)$ dans le réacteur est utilisé pour contrôler cette réaction.

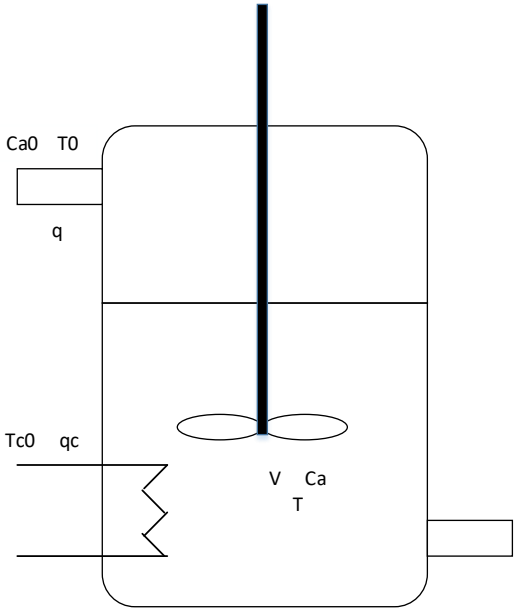

Figure 4.12 : Schéma du réacteur CSTR

La réaction peut être écrite comme suit [22] :

$$\begin{aligned}\dot{C}_a(t) &= \frac{q}{v}\big(C_{a0} - C_a(t)\big) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \\ \dot{T}(t) &= \frac{q}{v}\big(T_0 - T(t)\big) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} \\ &\quad + k_2 q_c(t)\left(1 - e^{-\frac{k_3}{q_c(t)}}\right)\big(T_{c0} - T(t)\big)\end{aligned} \tag{4.6}$$

Les constantes thermodynamiques et chimiques sont données dans le tableau (4.3), et les paramètres k_1, k_2 et k_3 sont donné par : $k_1 = \frac{\Delta H k_0}{\rho C_p}$, $k_2 = \frac{\rho_c C_{pc}}{\rho C_p v}$ et $k_3 = \frac{h_a}{\rho_c C_{pc}}$.

Tableau 4.3 : Paramètres nominaux du CSTR

Les paramètres	La signification	La valeur nominale
q	Taux du débit	$100\ l/min$
k_0	Constante du taux de réaction	$7.2 \times 10^{10} min^{-1}$
v	volume du réacteur	$100\ l$
E/R	Terme d'activation d'énergie	$1 \times 10^4\ K$
T_{c0}, T_0	Température Nominale	$350\ K$
ΔH	La chaleur de la réaction	$2 \times 10^5\ cal/mol$
ρ, ρ_c	Densité du liquide	$1 \times 10^3\ g/l$
C_p, C_{pc}	Capacité de chaleur	$1\ cal\ /g/K$
C_{a0}	Concentration nominale	$1\ mol/\ l$
h_a	Terme de transfert	$7 \times 10^5\ cal\ /min/K$

Pour avoir une concentration de $C_a = 0.1\ mol/l$, les conditions nominales de la température et le taux d'écoulement du liquide de refroidissement sont respectivement $438.54\ K$ et $0.1\ l/min$.

4.3.1 Identification hors ligne du système

La procédure d'identification hors ligne du système CSTR est réalisée par un ensemble de données d'apprentissage généré à partir du modèle mathématique du système. La période d'échantillonnage du système est fixée à $t_s = 6$ seconde et le modèle mathématique présenté dans l'équation (4.6) est utilisé pour générer cet ensemble ($N_d = 900$). Les premiers 400 échantillons sont utilisés pour construire le modèle flou de type TS-KRR, alors que le reste de ces échantillons ont été utilisés pour faire la validation du modèle. Le signal de commande représenté dans la Figure (4.13) a été utilisé pour générer les 900 échantillons, et la taille du vecteur d'entrée est fixée à 8 ($n_a = 5$ et $n_b = 3$).

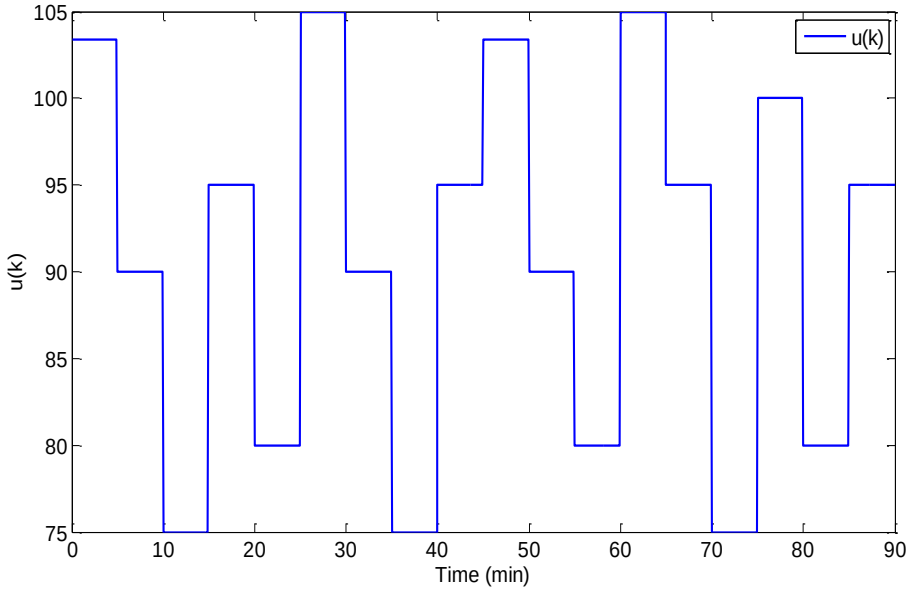

Figure 4.13 : Le signal de commande utilisée pour générer l'ensemble de 900 échantillons du système CSTR

Maintenant, nous utilisons la même technique pour identifier le nombre de groupes N_r. Le nombre de groupes devrait changer entre 2 et 16. Ensuite, l'algorithme K-moyennes est exécuté pour tous les nombres possibles de groupes et l'indice de Dunn est calculé. Enfin, le nombre convenable de groupes est sélectionné en fonction de la valeur la plus élevée de l'indice de Dunn (voir figure 4.14). Le nombre convenable de groupes trouvé est $N_r = 7$.

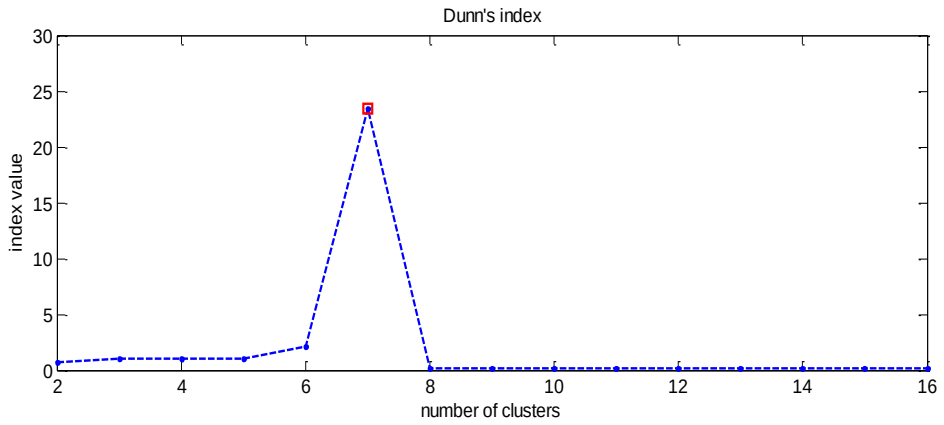

Figure 4.14 : La variation de l'indice de Dunn

De plus, la méthode de partitionnement basé sur l'algorithme de PSO est utilisée pour calculer les paramètres des antécédents. Le nombre de particules utilisées dans l'algorithme PSO est 12 alors que les paramètres de PSO : w_0, c_1, c_2 et $iter\_max$ sont mis respectivement à $0.7, 1.6, 1.6$ et 100. Les particules d'algorithme de PSO sont initialisées comme suit : la méthode de K-moyennes est exécuté et les résultats seront considérés comme les premières particules (déposer dans $\breve{\boldsymbol{x}}_1$). Le reste des particules sont choisies d'une façon aléatoire mais limités par des valeurs maximale et minimale, qui sont définies respectivement comme : $\breve{\boldsymbol{x}}_{max} = \breve{\boldsymbol{x}}_1 + 0.25 \times \breve{\boldsymbol{x}}_1$ et $\breve{\boldsymbol{x}}_{min} = \breve{\boldsymbol{x}}_1 - 0.25 \times \breve{\boldsymbol{x}}_1$.

La constante λ est fixée à $0{,}0005$. Après avoir exécuté les étapes d'identification hors ligne du modèle TS-KRR dans l'algorithme (3.2), le résultat d'identification de la concentration du produit A_a est présenté dans la figure (4.15).

Encore une fois, une étude comparative est réalisée pour évaluer la performance de partitionnement basé sur l'algorithme de PSO, la méthode standard de K-moyennes et une méthode de partitionnement basé sur l'algorithme génétique (GA). Les paramètres de GA sont : nombre de population égale à 12, le taux de sélection est 0.5, le taux de croisement est 0.7, le taux de mutation est 0.2 et les mêmes limites maximales et minimales de PSO sont utilisées pour initialiser la population de GA. Enfin, la même stratégie utilisée dans l'exemple précédent est adoptée pour effectuer la méthode de partitionnement basé sur GA.

La figure (4.15) contient les résultats des prédictions du modèle TS-KRR lorsque les paramètres des antécédents sont calculés par trois méthodes de partitionnement différentes : TS-KRR (PSO), TS-KRR (GA) et TS-KRR (K-means). En outre, le résultat du modèle TS-KRR (PSO), lorsqu'une perturbation de 6% est ajoutée aux données d'apprentissage, est présentée dans la figure (4.15). Enfin, la figure (4.15) inclure aussi les résultats de prédiction des modèles GNN-FIS et TSFS-SVR.

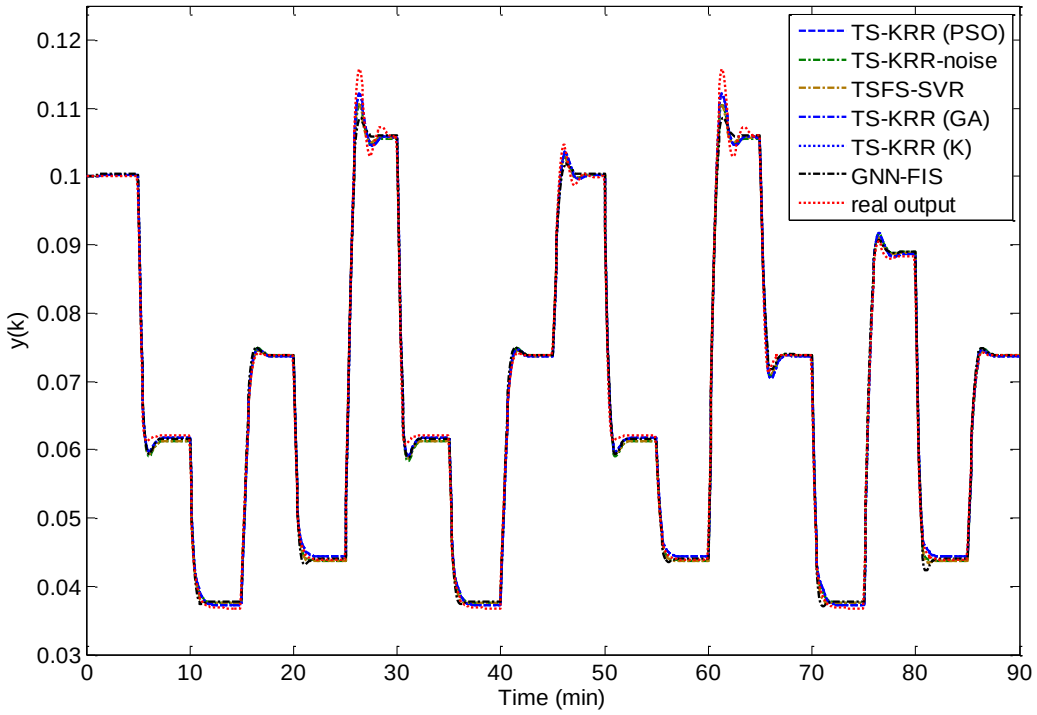

Figure 4.15 : La performance de modélisation à partir de modèle floue de type TS-KRR

Comme prévu, le modèle flou de type TS-KRR fonctionne bien (avec perturbation et aussi sans perturbation) et fournit de bons résultats (voir la Figure 4.15). En outre, les améliorations apportées par la méthode de partitionnement basé sur l'algorithme de PSO sont mineures (voir le tableau 4.4) où les mesures de précision montrent que les résultats obtenus par le modèle TS-KRR (PSO) sont légèrement meilleures que les résultats obtenus par le modèle TS-KRR basée sur les deux autres méthodes de partitionnement. En outre, l'approche TS-KRR a donné de meilleurs résultats que les méthodes de GNN-FIS et TSFS-SVR, qui peut être vu dans le tableau (4.4) où le modèle de type TS-KRR requiers moins de règles flous (seulement 7 règles pour le TS-KRR) que les deux autres méthodes. On peut le voir aussi, dans le tableau (4.4), que les méthodes de GNN-FIS et TSFS-SVR requièrent plus des règles floues pour générer des résultats relativement acceptables.

Les valeurs des erreurs obtenues par la prédiction du modèle TS-KRR sont inférieures à celles obtenues par les méthodes de GNN-FIS et TSFS-SVR.

Les modèles TS-KRR (GA) et TS-KRR (K-means) donnent des résultats un peu différents du modèle TS-KRR (PSO) ce qui signifie que la méthode de K-moyennes regroupe efficacement les données d'apprentissage.

Tableau 4.4 : Résultats de comparaison pour identifier la concentration du produit

Méthodes	Nombre de règles	Nombre d'entrées	RMSE	MAE	MAPE (%)	sMAPE (%)	Temps de simulation(s)
TS-KRR (PSO)	07	$n_a = 5$, $n_b = 3$	3.132132 $\times 10^{-5}$	1.754412 $\times 10^{-5}$	2.662045 $\times 10^{-4}$	2.664721 $\times 10^{-4}$	14.1892
TS-KRR (GA)	07	$n_a = 5$, $n_b = 3$	3.139146 $\times 10^{-5}$	1.754966 $\times 10^{-5}$	2.671203 $\times 10^{-4}$	2.681160 $\times 10^{-4}$	14.6214
TS-KRR (K-means)	07	$n_a = 5$, $n_b = 3$	3.159611 $\times 10^{-5}$	1.778136 $\times 10^{-5}$	2.696238 $\times 10^{-4}$	2.714248 $\times 10^{-4}$	05.1732
TS-KRR (PSO) (6%)	07	$n_a = 5$, $n_b = 3$	4.415412 $\times 10^{-5}$	2.184041 $\times 10^{-5}$	3.167274 $\times 10^{-4}$	3.186142 $\times 10^{-4}$	14.1892
GNN-FIS	07	$n_a = 5$, $n_b = 3$	6.338147 $\times 10^{-3}$	5.758142 $\times 10^{-3}$	0.030611	0.0030044	0.8114
GNN-FIS	20	$n_a = 5$, $n_b = 3$	8.782313 $\times 10^{-4}$	6.024566 $\times 10^{-4}$	0.009402	0.009378	0.9874
TSFS-SVR	07	$n_a = 5$, $n_b = 3$	6.933145 $\times 10^{-4}$	5.883695 $\times 10^{-4}$	0.008538	0.008601	51.1698
TSFS-SVR	14	$n_a = 5$, $n_b = 3$	6.527324 $\times 10^{-4}$	4.212638 $\times 10^{-4}$	0.006614	0.006616	55.4168

4.3.2 La commande prédictive floue du système CSTR

Dans cette section, nous étudierons la commande TS-KRR GPC avec une identification hors ligne en contrôlant le système CSTR. Avant de commencer la procédure de commande, les paramètres qui doivent être sélectionnés sont : la taille du vecteur d'entrée qui est définie par ($n_a = 5$ et $n_b = 3$), et les paramètres de commande GPC qui sont : $N_p = 10$, $N_u = 2$ et $\vartheta = 0.0008$. Le nombre de règles floues, qui est obtenu dans la section d'identification, est égal à $N_r = 7$ et le reste des paramètres du modèle TS-KRR sont similaires à ceux discutés

précédemment. Apres la réalisation de l'algorithme (3.5), la sortie du système et le signal de commande appliqué au système sont illustrés respectivement dans les figures (4.16) et (4.17). Encore, la performance du commande TS-KRR GPC est analysée lorsque deux autres algorithmes de partitionnement sont utilisés pour calculer les paramètres des antécédents. Les résultats de la simulation sont inclus dans les figures (4.16).

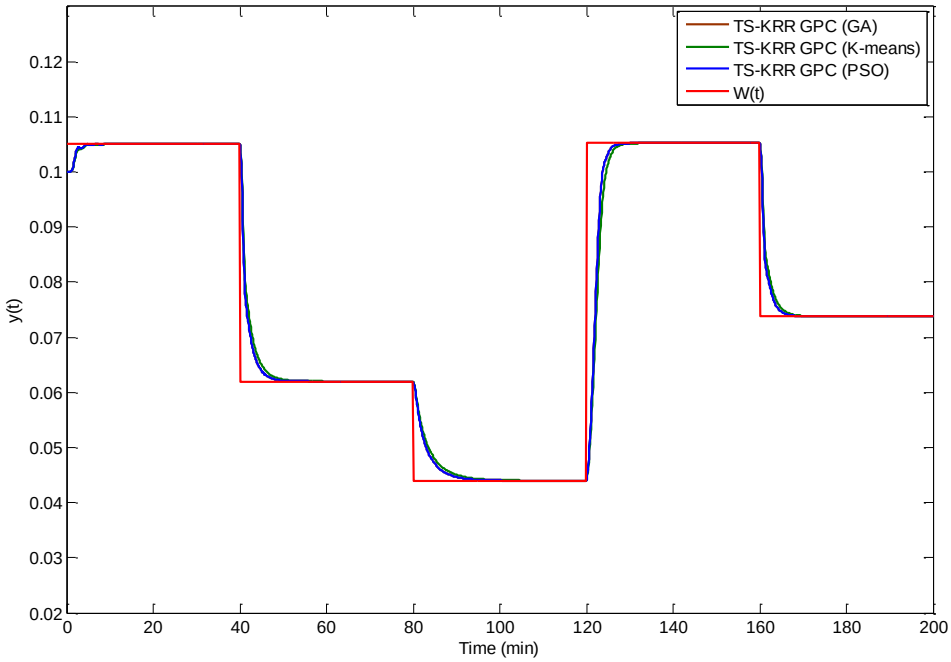

Figure 4.16 : Les résultats du commande TS-KRR GPC

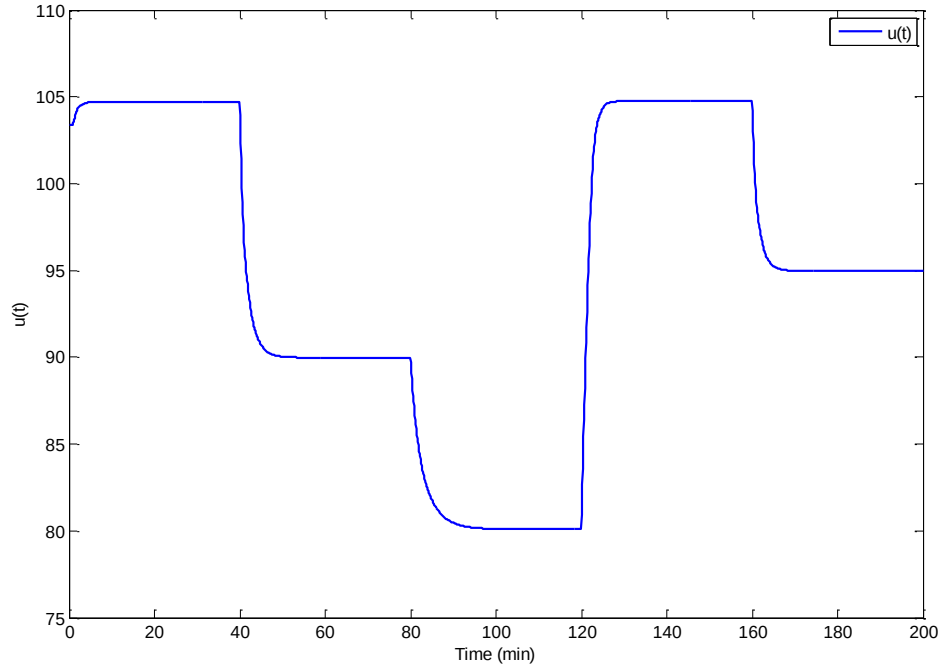

Figure 4.17 : Le signal de commande appliqué au système

Les résultats de la simulation montrent une bonne poursuite de la sortie du système à la référence $W(k)$, et que la méthode de commande TS-KRR GPC garantit une bonne performance (une réponse rapide avec des dépassements nuls) pour le système commandé. Aussi, la commande GPC TS-KRR avec les trois différentes méthodes de partitionnement a donné des résultats presque similaires. La méthode de K-moyennes est alors suffisante pour calculer les paramètres des antécédents. Dans la prochaine section, l'identification adaptative du modèle floue de type TS-KRR est encore investigué lorsque le même système est commandé. La méthode K-moyennes sera utilisée pour initialiser les paramètres des antécédents du modèle floue puisque les améliorations apportées par l'algorithme PSO sont insignifiantes.

4.3.3 Identification en ligne du système CSTR

La procédure d'identification en ligne du système est réalisée par le même ensemble de données d'apprentissage généré à partir du modèle mathématique, et nous garderons les mêmes paramètres que ceux utilisés dans l'identification hors ligne du modèle. Le β de taux d'apprentissage est égal à 0.3 et la taille de dictionnaire $M = 50$. La méthode de K-moyennes est utilisée pour initialiser les paramètres des antécédents. Après avoir exécuté l'algorithme (3.4), la

prédiction de la concentration du produit A_a est présentée dans la figure (4.18). En outre, figure (4.18) donne le signal original de la concentration, la concentration prédite par la méthode de ANFIS et la prédiction de concentration obtenue par une méthode adaptative de type Takagi-Sugeno-Kang (ATSK).

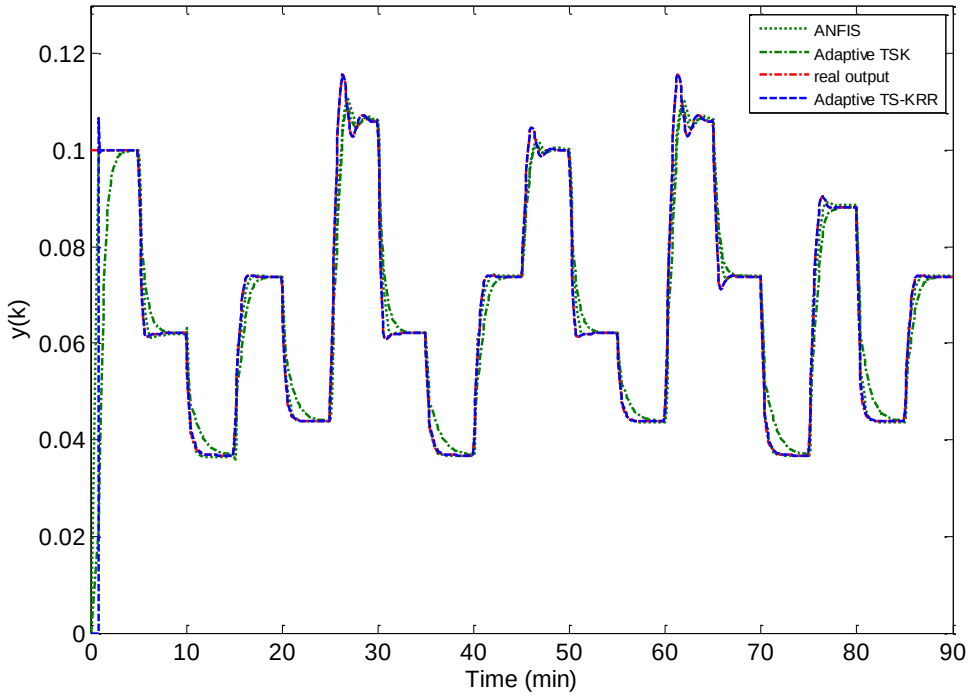

Figure 4.18 : La performance de modélisation à partir de modèle floue adaptative de type TS-KRR

L'objectif, ici, est de comparer entre les performances de la prédiction obtenue par le modèle TS-KRR et les prédictions données par les modèles ANFIS et ATSK. On peut remarquer que la stratégie de TS-KRR est plus performante (plus précis) que les deux autres approches (voir figure 4.18). Cela peut être aussi vérifié à partir du tableau (4.5) où les mesures d'erreurs obtenues indiquent que le modèle TS-KRR adaptatif a effectivement prédit la concentration C_a et donne une meilleure précision que les modèles ANFIS et ATSK. En outre, le nombre de règles utilisées par le modèle TS-KRR adaptatif est inférieur à celui utilisé par les modèles ANFIS et ATSK.

Tableau 4.5 : Résultats de comparaison pour identifier la concentration du produit

Méthodes	Nombre de règles	Nombre d'entrées	RMSE	MAE	MAPE (%)	sMAPE (%)
TS-KRR	07	$n_a = 5$, $n_b = 3$	1.023 $\times 10^{-3}$	4.030 $\times 10^{-4}$	6.097 $\times 10^{-3}$	6.100 $\times 10^{-3}$
ANFIS	20	$n_a = 5$, $n_b = 3$	2.381 $\times 10^{-2}$	7.620 $\times 10^{-3}$	0.1156	0.1167
ATSK	20	$n_a = 5$, $n_b = 3$	5.173 $\times 10^{-2}$	2.061 $\times 10^{-2}$	0.3116	0.3147

Nous pouvons aussi vérifier les performances du modèle TS-KRR adaptatif dans le cas où des perturbations sont présentes dans le signal de commande. Le signal d'entrée utilisé pour générer les données 'd'apprentissage est donné dans la figure (4.19).

Dans ce cas, le signal de commande a les mêmes niveaux que celui-ci présenté précédemment à la figure (4.13); cependant, une perturbation avec une forme gaussienne et une amplitude de 10 est ajoutée à ce signal de commande. Nous garderons les mêmes paramètres que ceux utilisés dans l'identification précédente. Le β est égal à 0.3 et la taille de dictionnaire $M = 50$. La méthode de K-moyennes est utilisée pour initialiser les paramètres des antécédents. Après avoir exécuté l'algorithme (3.4), la prédiction de la concentration est présentée dans la figure (4.20).

On peut voir que le modèle TS-KRR adaptatif s'est bien comporté et a produit des résultats de prédiction très précis. De plus, les valeurs d'erreur obtenues par le modèle TS-KRR adaptatif sont très faibles malgré le fait que les bruits qui ont été injectés au signal d'entrée sont très élevés ($RMSE = 7.312 \times 10^{-3}$).

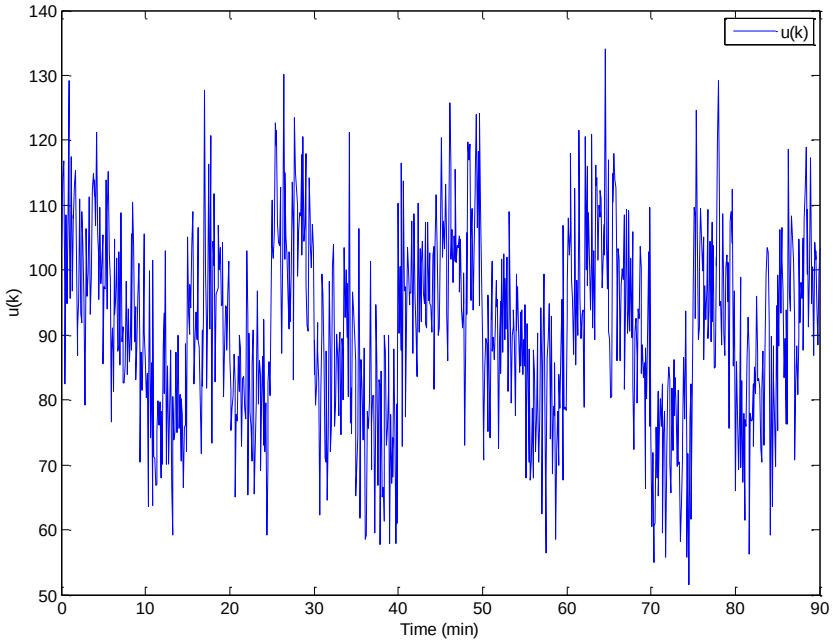

Figure 4.19 : Le signal de commande utilisée pour générer l'ensemble de 900 échantillons

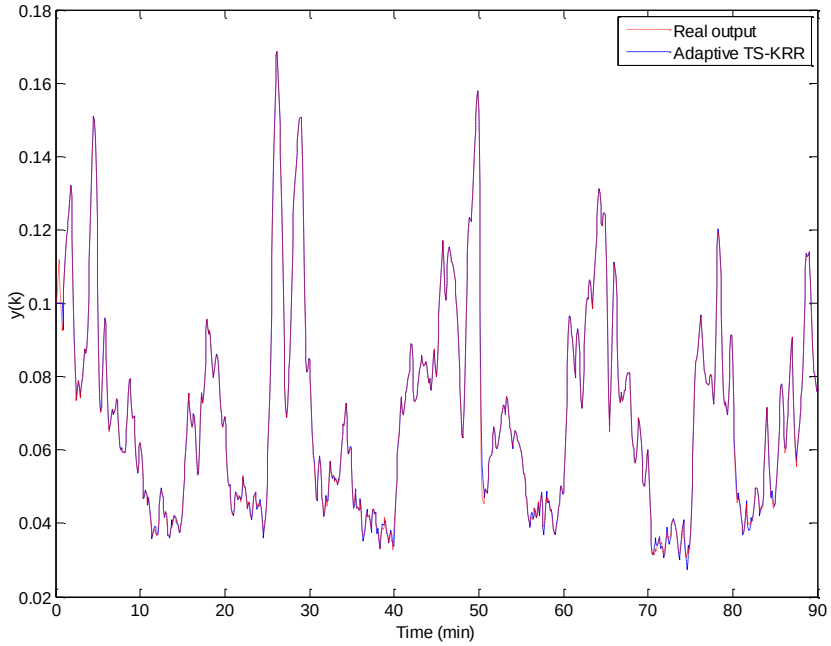

Figure 4.20 : La performance de modélisation à partir de modèle TS-KRR adaptatif

Dans la sous-section suivante, le modèle adaptatif TS-KRR est intégré à la commande GPC et utilisé pour contrôler le système non linéaire CSTR.

4.3.4 La commande prédictive floue adaptatif du système

La commande TS-KRR GPC adaptative est utilisée pour contrôler le système CSTR. Les paramètres du modèle TS-KRR utilisant dans la commande sont les mêmes paramètres que dans la partie d'identification en ligne, et les paramètres de commande GPC sont : $N_p = 10$, $N_u = 2$ et $\vartheta = 0.0008$. Après avoir exécuté l'algorithme (3.6), la concentration du système et le signal de commande appliqué au système sont illustrés respectivement dans les figures (4.21) et (4.22).

En outre, les solutions obtenu par la commande ANFIS GPC (les paramètres du contrôle de ANFIS GPC sont : $N_p = 10$, $N_u = 2$, $\vartheta = 0.00076$ et le nombre de règles floues est $N_r = 20$) et la commande ATSK GPC (les paramètres du contrôle de ATSK GPC sont : $N_p = 10$, $N_u = 2$, $\vartheta = 0.00076$ et $N_r = 20$) sont aussi illustrés dans la figure (4.21).

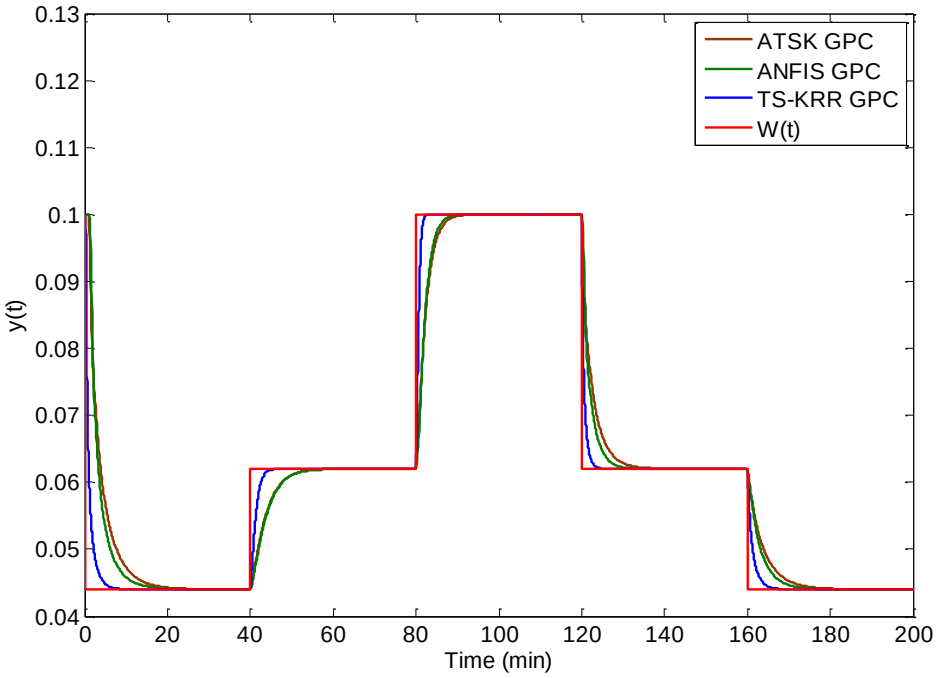

Figure 4.21 : Les résultats du commande TS-KRR GPC adaptatif

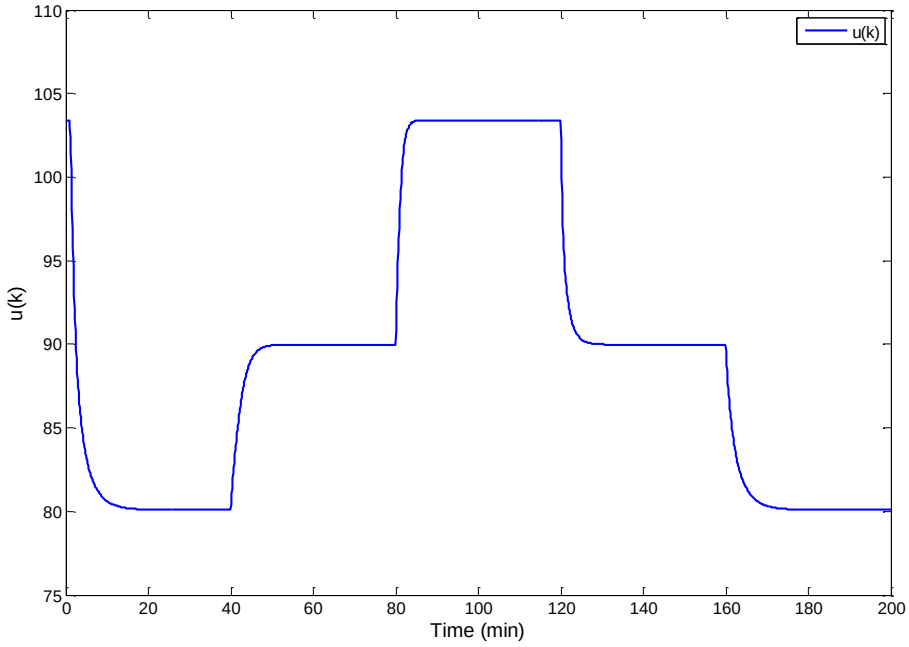

Figure 4.22 : Le signal de commande appliqué au système

Les résultats de la figure (4.21) montrent que la sortie du système est totalement confondue avec la référence $W(k)$, ce qui montre que la méthode TS-KRR GPC adaptative utilisée est efficace. Les mêmes commentaires peuvent être faits pour les résultats obtenus par les commandes ANFIS GPC et ATSK GPC. On peut remarquer aussi que la stratégie de commande TS-KRR GPC adaptative est plus rapide que la régulation par les méthodes ANFIS GPC et ATSK GPC. La valeur moyenne pour exécuter une itération pour une taille de dictionnaire de $M = 50$ est 0.00781 seconde, ce qui est considéré comme une période de temps compatibles avec la plupart des systèmes réels. D'autre part, la commande ANFIS GPC a un temps d'exécution plus petit (0.000912 seconde) car son algorithme ne demande pas un grand nombre de données précédentes. Le même commentaire peut être fait pour la commande ATSK GPC où le temps d'exécution pour un échantillon est relativement petit (0.000796 seconde). La figure (4.23) représente l'erreur absolue produite par les trois méthodes de commandes où cette erreur est définie par la différence entre le signal de sortie et le signal de référence $e(t) = |y(t) - w(t)|$.

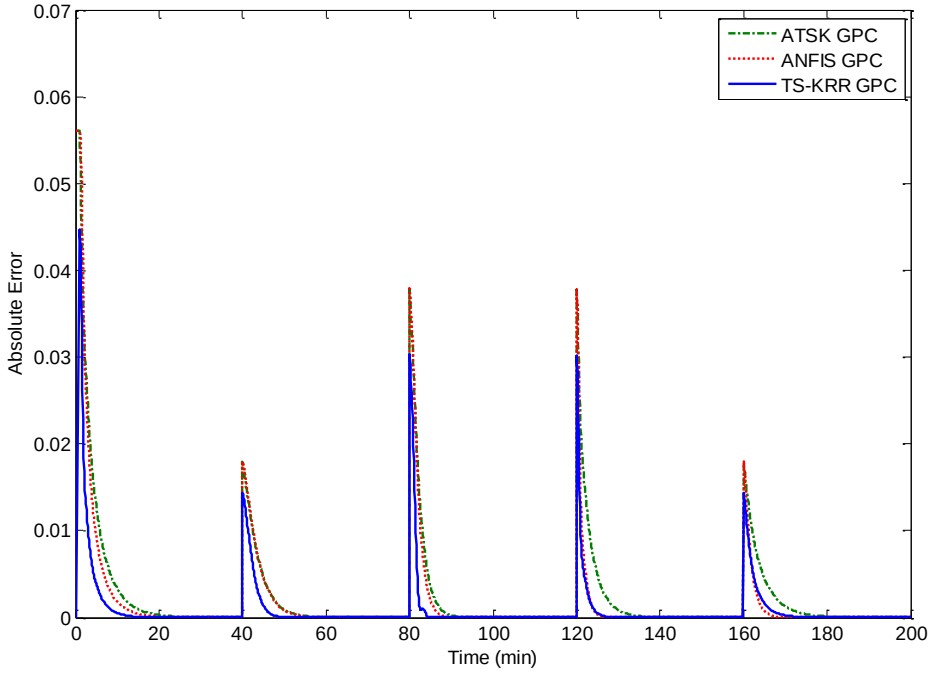

Figure 4.23 : L'erreur absolue produit par les trois méthodes de commandes

Comme prévu, le commande TS-KRR GPC adaptatif est plus rapide et montrer moins d'erreurs lorsqu'il se déplace d'un niveau de référence à un autre. En outre, les résultats indiquent que la commande ATSK GPC produit relativement les plus grandes erreurs lorsqu'il se déplace entre les niveaux du signal de références qui pourraient être liés aux fonctions d'appartenance utilisées dans ce modèle.

Dans cette section, nous avons injecté une perturbation au système contrôlé pour tester la robustesse de stabilité de la méthode de commande proposée. Les perturbations appliquées au système sont définies comme des constantes où une perturbation avec l'amplitude de 0.012 a été appliquée au système dans l'intervalle de temps $600 \leq k \leq 100$ minutes. Une autre perturbation avec une amplitude de 0.02 a été appliquée au système dans l'intervalle $100 < k \leq 140$ minutes. L'objectif de la poursuite de trajectoire de référence est atteint en présence des perturbations où les figures (4.24) et (4.25) montre les résultats de cette simulation.

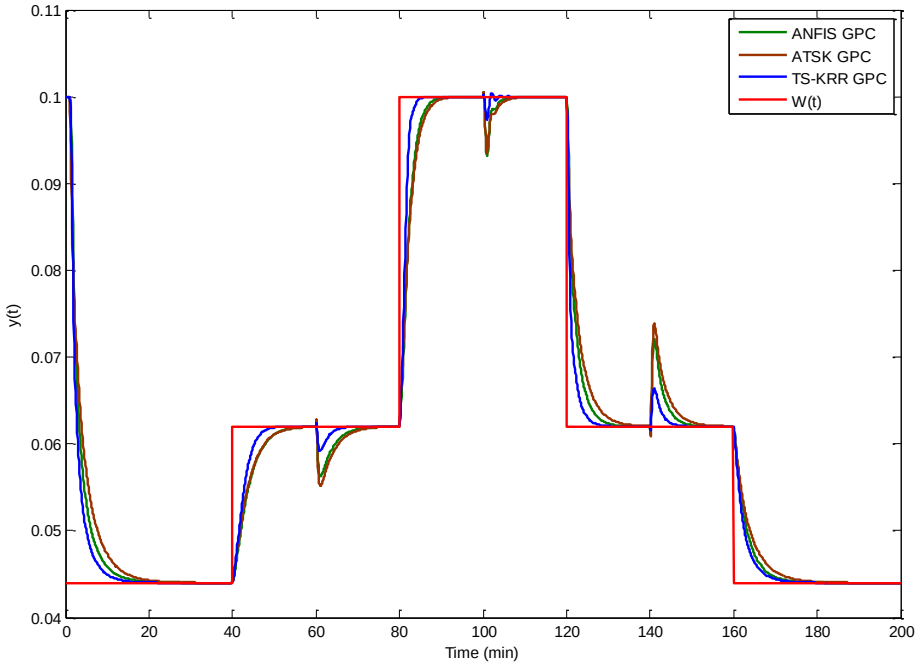

Figure 4.24 : Les résultats du commande TS-KRR GPC adaptatif

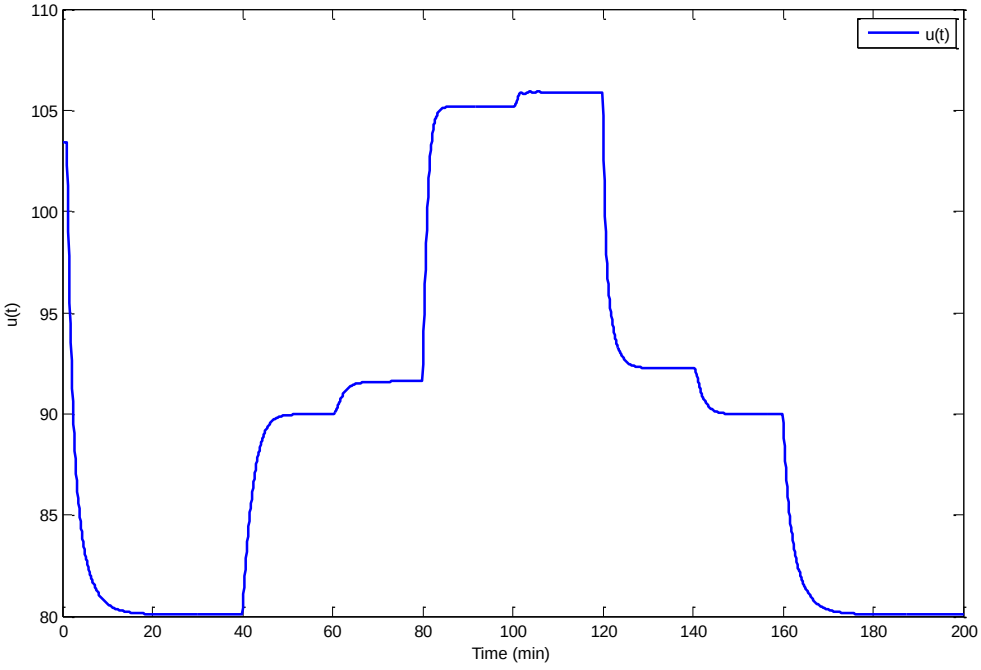

Figure 4.25 : Le signal de commande appliqué au système

Nous remarquons de plus que les performances obtenues avec la commande TS-KRR GPC adaptative sont meilleures que les autres méthodes. Dans le cas du rejet de perturbation, la commande proposée présente de plus petits dépassements en régime dynamique du système que les commandes prédictives basées sur les modèles ANFIS et ATSK. Généralement, les résultats de la simulation démontrent la capacité de la commande TS-KRR GPC adaptive à commander un tel processus malgré l'adjonction de bruit au système. Cela peut être vérifié à partir de la figure (4.26) où l'erreur absolue de la sortie obtenue par la commande TS-KRR GPC adaptative (lorsque les perturbations sont appliquées) est inférieure aux erreurs absolues produites par les deux autres contrôleurs. Encore une fois, les résultats de la figure (4.26) montrent que la commande ATSK GPC produit relativement les plus grandes erreurs lorsque les perturbations sont appliquées au système.

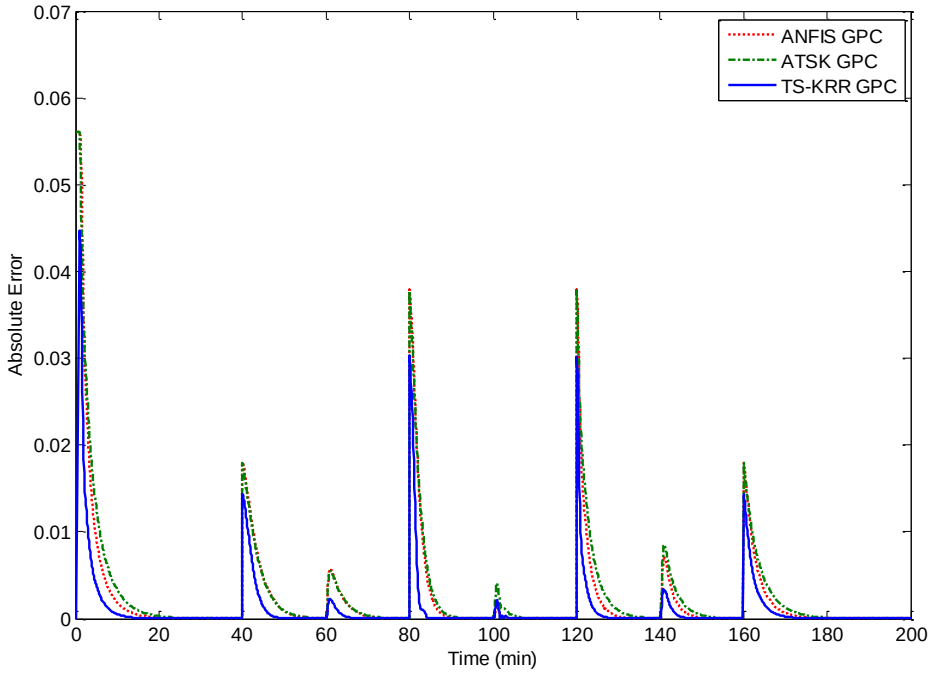

Figure 4.26 : L'erreur absolue produit par les trois méthodes de commandes

Il est clair que la stratégie de calcul utilisée pour générer les fonctions d'appartenance par le modèle de type ATSK a un impact négatif sur la précision d'identification et de la commande (le modèle de ATSK n'ajoute pas les paramètres des antécédents pendant le processus de la commande).

Chapitre 5

Commande Prédictive Floue Basée Sur la Méthode de Régression Par les Machines à Vecteurs de Support en Moindres Carrés

5.1 Introduction

Dans les deux derniers chapitres, nous avons proposé une commande prédictive floue adaptative pour les systèmes non linéaires où le système flou de type TS utilisée par la commande a été construit en utilisant la méthode de KRR, et la procédure d'apprentissage en ligne a été réalisé par la méthode de SW-KRLS. Cependant, la méthode proposée dans ces deux chapitres ne traite que les cas où le modèle n'a pas le terme biais (ce qui est le cas du modèle KRR).

Dans ce chapitre, nous généralisons l'approche basée sur les méthodes à noyau pour inclure des modèles avec un terme de biais. Dans ce cas, on propose une méthode qui utilise la régression par les machines à vecteurs de support en moindres carrés (LSSVR) pour identifier les paramètres des conséquents d'un modèle flou de type TS. Grâce au théorème présenté dans [90], le modèle flou adaptatif de type TS a une forme simple et peut-être facilement utilisé pour l'identification et la commande des systèmes. Ce théorème montre que le terme de biais peut être négligé ($b = 0$) si la fonction noyau employée est augmentée pour fournir un biais implicite, ce qui est exactement le cas des fonctions noyaux strictement positives définies [90, 91, 92, 93].

Deux objectifs principaux sont considérés dans ce chapitre: le premier objectif est de proposer un modèle flou de type TS adaptatif basé sur la méthode de LSSVR pour l'identification des

systèmes non linéaires, le second objectif est d'introduire une commande non linéaire, la commande TS-LSSVR GPC adaptatif, par l'intégration du modèle TS-LSSVR adaptatif dans la commande GPC [97].

Dans ce chapitre, les paramètres des antécédents de la nouvelle modélisation floue de type TS basé sur la méthode de LSSVR sont initialisés à partir d'une méthode de partitionnement appelé «C-moyennes flous » alors que la mise à jour de ces paramètres est réalisée en utilisant un algorithme simple de rétro-propagation. Les paramètres linéaires sont obtenus à l'aide de la méthode de LSSVR. De plus, la mise à jour des paramètres des conséquences du TS-LSSVR adaptatif est réalisée par la méthode de moindres carrés récursifs à noyau avec un budget fixe (FB-KRLS) [84].

Nous examinerons la performance de la méthode floue proposée pour l'identification et la commande des systèmes non linéaires.

5.2 Structure d'un modèle TS flous basé sur la méthode de LSSVR

Comme défini au chapitre 3, le modèle flou de type TS-LSSVR est aussi basé sur une décomposition floue de l'espace des entrées où pour chaque région de cet espace une règle floue peut être construite afin de faire une approximation linéaire de la sortie. La sortie globale du modèle flou de type TS-LSSVR est aussi obtenue par la même combinaison spéciale, qui a été définie au chapitre 3, de l'ensemble des règles construites. L'architecture du modèle flou de type TS-LSSVR peut être décrite avec la même figure présentée au chapitre 3 (figure 3.1).

Généralement, les seules différences entre le modèle flou défini au chapitre 3, modèle flou de type TS-KRR, et le modèle basé sur la méthode de LSSVR est que le vecteur d'entrée du modèle flou de type TS-LSSVR est augmenté comme suit: $\dot{\boldsymbol{x}}(k) = (1, \boldsymbol{x}(k)) = (1, x_1(k), \dots, x_n(k))^T$, et les paramètres des conséquents de la $i^{\text{ème}}$ règle floue sont définis comme : $\widehat{\boldsymbol{\theta}}_i = (a_0^i, \dots, a_n^i)$. Le modèle flou de type TS-LSSVR est alors basé sur une collection de règles du type :

$$\begin{aligned} R_i: \quad & \text{Si} x_1(k) \; est \; A_1^i, et \dots, et \; x_n(k) \; est \; A_n^i \\ & \text{Alors } y_i(k) = a_0^i + a_1^i x_1(k) + \cdots + a_n^i x_n(k) \\ & i = 1, \dots, N_r \end{aligned} \tag{5.1}$$

avec $R_i, i = 1, 2, \dots, N_r$ la $i^{\text{ème}}$ règle floue et N_r la nombre de règles flous. Les variables d'entrées sont : $x_1(k), \dots, x_n(k)$, avec k l'échantillon du temps discret, et $y_i(k)$ est la sortie de la $i^{\text{ème}}$ règle floue. Les $A_j^i, i = 1, 2, \dots, N_r, j = 1, 2, \dots, n$ son les sous-ensembles flous antécédents qui sont caractérisés par les fonctions d'appartenance floue $\mu_{A_j^i}(x_j)$, $i = 1, 2, \dots, N_r, j = 1, 2, \dots, n$. Comme présenté dans la figure (3.1), l'architecture du modèle flou de type TS-LSSVR est composée de cinq couches. Il est important de mentionner que toutes les couches du modèle flou de type TS-LSSVR sont similaires aux couches du modèle flou de type TS-KRR sauf la 4$^{\text{ème}}$ couche. Comme défini précédemment au chapitre 3, la contribution des règles floues est calculée dans la 4$^{\text{ème}}$ couche, où la contribution de $i^{\text{ème}}$ règle est donné par $\widehat{\boldsymbol{\theta}}_i \psi_i$ avec : $\psi_i = \dot{\boldsymbol{x}}.\mu_i(\dot{\boldsymbol{x}})$ est le degré d'activation de $i^{\text{ème}}$ règle, $\dot{\boldsymbol{x}}$ représente le vecteur d'entrée augmenté et $\widehat{\boldsymbol{\theta}}_i$ sont les paramètres de conséquences de la $i^{\text{ème}}$ règle.

Enfin, pour calculer la sortie $\hat{y}(k)$ du modèle TS-LSSVR, les fonctions d'appartenance utilisées sont représentées par des fonctions gaussiennes qui sont données par la forme suivant :

$$\mu_{A_j^i}(x_j) = exp\left\{-\frac{(x_j - m_{ij})^2}{2{\sigma_{ij}}^2}\right\}, i = 1, \dots, N_r \; et \; j = 1, \dots, n \tag{5.2}$$

avec m_{ij} et σ_{ij} sont respectivement le centre et la variance de la $i^{\text{ème}}$ fonction d'appartenance. Le degré d'activation de la $i^{\text{ème}}$ règle est donné par le produit cartésien de toutes les fonctions d'appartenances qui appartient de cette règle :

$$\begin{aligned} \mu_i(\dot{\boldsymbol{x}}) = \prod_{j=1}^{n+1} \mu_{A_j^i}(x_j) &= exp\left\{-\sum_{j=1}^{n} \frac{(x_j - m_{ij})^2}{2{\sigma_{ij}}^2}\right\} = \mu_i(\boldsymbol{x}) \\ &= exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \Sigma (\boldsymbol{x} - \boldsymbol{m}_i)\right\}, i = 1, \dots, N_r \end{aligned} \tag{5.3}$$

où $\Sigma = \begin{bmatrix} {\sigma_{i1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {\sigma_{in}}^2 \end{bmatrix}^{-1}$.

Le produit cartésien de toutes les fonctions d'appartenances dans le cas du vecteur d'entrée augmenté $\dot{\boldsymbol{x}}(k) = (1, \boldsymbol{x}(k))$ est égal au produit cartésien de toutes les fonctions d'appartenances dans le cas du vecteur d'origine $\boldsymbol{x}(k)$ ($\mu_i(\dot{\boldsymbol{x}}) = \mu_i(\boldsymbol{x})$). Finalement, la fonction ψ_i qui définit le degré d'activation de $i^{\text{ème}}$ règle est donnée par : $\psi_i = \dot{\boldsymbol{x}}.\mu_i(\dot{\boldsymbol{x}}) = \dot{\boldsymbol{x}}.\mu_i(\boldsymbol{x})$, et la sortie du modèle floue de type TS-LSSVR est calculée par :

$$\hat{y}(k) = \sum_{i=1}^{N_r} \left(\widehat{\boldsymbol{\theta}}_i^T . \hat{\boldsymbol{x}}\right). \mu_i(\hat{\boldsymbol{x}}) = \sum_{i=1}^{N_r} \left(\widehat{\boldsymbol{\theta}}_i^T . \hat{\boldsymbol{x}}\right). exp\left\{-\sum_{j=1}^{n}\left(\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2}\right)\right\} \tag{5.4}$$

Dans les sections suivantes, les procédures utilisées pour l'identification en ligne des paramètres des conséquences $\widehat{\boldsymbol{\theta}}_i, i = 1, \dots, N_r$ du modèle TS-LSSVR ainsi que les valeurs des centres et des variances des fonctions d'appartenance sont présentées en détail.

5.3 Identification des paramètres des conséquences du modèle TS-LSSVR

Dans cette section, on va utiliser la méthode de régression par les machines à vecteurs de support en moindres carrés, qui a été présenté dans chapitre 1, afin d'identifier les paramètres des conséquences du modèle flou de type TS. La forme du modèle LSSVR est donnée par (voire chapitre 3) :

$$\hat{y}(k) = \sum_{i=1}^{N_d} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{5.5}$$

où N_d représente la dimension d'ensemble d'apprentissage (nombre de données entrées-sorties), $\alpha_i \in \mathbb{R}, i = 1, \dots, N_d$ sont les multiplicateurs de Lagrange, $b \in \mathbb{R}$ représente le biais, $\kappa(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \boldsymbol{\varphi}(\boldsymbol{x},), \boldsymbol{\varphi}(\boldsymbol{x}_i) \rangle$ est une fonction noyau avec $\boldsymbol{\varphi}: \mathcal{X} \to \mathcal{H}$ une fonction de re-description qui projette les vecteurs d'entrée $\mathcal{X}$ dans un espace de Hilbert $\mathcal{H}$. Comme démontré au chapitre 1, les valeurs des multiplicateurs de Lagrange et le biais sont donné par :

$$\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{1}^T \\ \mathbf{1} & \boldsymbol{K} + \gamma^{-1}\boldsymbol{I} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{Y} \end{bmatrix} \tag{5.6}$$

avec :$\mathbf{1} = (1, \dots, 1)^T$, $\boldsymbol{Y} = (y_1, \dots, y_{N_d})^T$, $\boldsymbol{I}$ est une matrice d'identité et $\boldsymbol{K}$ est une matrice de dimension $N_d \times N_d$ tel que $K_{ij} = \boldsymbol{\varphi}(\boldsymbol{x}_i)^T \boldsymbol{\varphi}(\boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j), i.j = 1, \dots, N_d$.

La solution analytique donnée par l'équation (5.6) peut être réduite (ou simplifié) en choisissant une fonction noyau strictement positive définie. Grâce au théorème présenté par Poggio *et* al. [90], le terme de biais peut être négligé pour certaines conditions, ce qui contribuera à simplifier le modèle de LSSVR présenté dans l'équation (5.5) et sa solution analytique donnée par l'équation (5.6).

Théorème 5.1: Si une fonction noyau fournit un terme de biais (*implicit bias*), alors le terme de biais b dans l'équation (5.5) peut être omis [90, 91, 92, 93].

Selon ce théorème, le terme de biais peut être négligé ($b = 0$) si la fonction noyau utilisée pour obtenir le model TS-LSSVR inclut un terme de biais, ce qui est exactement le cas des fonctions noyau strictement positive définie [90, 91]. Par exemple, les deux fonctions noyaux gaussien et polynomial fournissent un terme de biais. Poggio *et al*. [90] montrent qu'un expert peut choisir entre le modèle donné par équation (5.5) avec et sans le terme de biais pour décrire les prédictions du système non linéaire. Comme prouvé dans [90], la prédiction $\hat{y}(k)$ obtenue par un modèle sans terme de biais ($\hat{y}(k) = \sum_{i=1}^{N_d} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i)$) avec $\kappa(\boldsymbol{x}, \boldsymbol{x}_i)$ est une fonction noyau strictement positive définie, est égale à la prédiction $\hat{y}(k)$ obtenue par un modèle avec le terme de biais ($\hat{y}(k) = \sum_{i=1}^{N_d} \alpha_i \kappa^*(\boldsymbol{x}, \boldsymbol{x}_i) + b$) où $\kappa^*(\boldsymbol{x}, \boldsymbol{x}_i)$ est une nouvelle fonction du noyau positive semi-définie. De plus, il a été aussi prouvé que la nouvelle fonction noyau positive semi-définie est égale à : $\kappa^*(\boldsymbol{x}, \boldsymbol{x}_i) = \kappa(\boldsymbol{x}, \boldsymbol{x}_i) - \beta$ [90], où $\kappa(\boldsymbol{x}, \boldsymbol{x}_i)$ est la fonction noyau strictement positive définie originale et β est une constante appropriée [90, 91, 92, 93]. Il est également important d'expliquer le sens de ce théorème où le terme de biais n'était pas vraiment négligé, mais plutôt ce terme a été déplacé pour faire partie de la fonction noyau. A la suite de ce théorème, le modèle de LSSVR sans terme de biais démontre une bonne performance dans la régression et les classifications. De plus, l'avantage de négliger le terme de biais est qu'un algorithme simple sera utilisé pour identifier les multiplicateurs de Lagrange car il n'y a aucune contrainte d'égalité supplémentaire nécessaire pendant l'optimisation (surtout dans le cas de SVM classique). La seule différence entre les prédictions faites lorsque le terme de biais b est négligé et le modèle original (voir l'équation 5.5) est que des vecteurs de support supplémentaires sont impliqués dans le cas du modèle avec un terme de biais.

Dans ce chapitre, une fonction noyau strictement positive définie est sélectionnée pour faire la prédiction avec la méthode LSSVR, et selon le théorème 5.1 le terme de biais peut être évité et le modèle de LSSVR simplifié et la solution analytique des multiplicateurs de Lagrange $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{N_d})$ sont donnés par l'équation (5.7) :

$$\hat{y}(k) = \sum_{i=1}^{N_d} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i)$$
$$\boldsymbol{\alpha} = (\boldsymbol{K} + \gamma^{-1}\boldsymbol{I})^{-1}\boldsymbol{Y} \tag{5.7}$$

Maintenant, pour obtenir des fonctions strictement positives définies appropriées $\kappa_l(\boldsymbol{x}, \boldsymbol{z}), l = 1, \dots, N_r$, les fonctions de re-description $\boldsymbol{\varphi}_l, l = 1, \dots, N_r$ doivent être précisément sélectionnées. L'utilisation des fonctions noyaux strictement positives définies est plus intéressante en matières de simplification du modèle flou de type TS-LSSVR, et aussi la simplification de la procédure d'obtention du modèle flou adaptatif de type TS-LSSVR, puisque le terme de biais peut être négligé (voir les équations 5.7).

Généralement, les fonctions du noyau gaussien et polynomial sont les fonctions strictement positives définies les plus reconnues qui ont été utilisées pour résoudre les problèmes de régression et de classification. Plusieurs chercheurs ont combiné ces deux fonctions pour obtenir de meilleures prédictions. De plus, une stratégie différente a été utilisée pour construire des fonctions noyau strictement positives définies à partir de fonctions noyau positives semi-définies. Dans cette stratégie, la fonction noyau positive semi-définie originale $\kappa_{old}(\boldsymbol{x}, \boldsymbol{z})$ est augmentée en ajoutant une constante, et la fonction noyau résultante $\kappa_{new}(\boldsymbol{x}, \boldsymbol{z}) = \kappa_{old}(\boldsymbol{x}, \boldsymbol{z}) + 1$ est une fonction noyau strictement positive définie. Cela peut être facilement vérifié en utilisant le théorème 1.1. Dans ce chapitre, une approche différente est adoptée pour obtenir une fonction noyau strictement positive définie où le vecteur d'entrée du système est augmenté en ajoutant une constante, et par conséquent le vecteur d'entrée est : $\dot{\boldsymbol{x}}(k) = \big(1, \boldsymbol{x}(k)\big) = \big(1, x_1(k), \dots, x_n(k)\big)^T$, $\dot{\boldsymbol{x}}(k) \in \mathbb{R}^{n+1}$, et les fonctions de re-description $\boldsymbol{\varphi}_l, l = 1, \dots, N_r$ avec le nouveau vecteur d'entrée $\dot{\boldsymbol{x}}(k)$ sont définies par:

$$\boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}) = \dot{\boldsymbol{x}}.\mu_l(\dot{\boldsymbol{x}}) = \dot{\boldsymbol{x}}.\mu_l(\boldsymbol{x}) = \dot{\boldsymbol{x}}.exp\left\{-\sum_{j=1}^{n}\frac{\big(x_j - m_{lj}\big)^2}{2{\sigma_{lj}}^2}\right\}, \qquad (5.8)$$
$$l = 1, \dots, N_r$$

où $\boldsymbol{m}_l = (m_{1l}, \dots, m_{nl})^T$ et $\boldsymbol{\sigma}_l = (\sigma_{1l}, \dots, \sigma_{nl})^T$ sont respectivement les vecteurs de centre et de variance. Il est aussi clair que le terme $\mu_l(\dot{\boldsymbol{x}})$ de l'équation (5.8) est similaire au terme $\mu_l(\boldsymbol{x})$ avec le vecteur d'entrée d'origine $\boldsymbol{x}(k)$ ($\mu_l(\boldsymbol{x}) = \mu_l(\dot{\boldsymbol{x}})$) car les deux vecteurs $\boldsymbol{x}(k)$ et $\boldsymbol{m}_l$ ont été augmentés en ajoutant 1, et les valeurs augmentées s'annulent mutuellement. Par conséquent, les fonctions du noyau $\kappa_l(\dot{\boldsymbol{x}}, \dot{\boldsymbol{x}}_i),\ l = 1, \dots, N_r$ obtenues à partir des fonctions de re-description $\boldsymbol{\varphi}_l, l = 1, \dots, N_r$ sont des fonctions noyaux strictement positives définies qui indiquent que la

fonction multi-noyau $\kappa(\grave{\boldsymbol{x}}, \grave{\boldsymbol{x}}_i) = \sum_{l=1}^{N_r} \kappa_l(\grave{\boldsymbol{x}}, \grave{\boldsymbol{x}}_i)$ est une fonction noyau strictement positive définie. Cela peut être vérifié en utilisant le théorème 1.

Preuve :

$$\sum_{i,j=1}^{N_d} \delta_i \delta_j \kappa(\grave{\boldsymbol{x}}_i, \grave{\boldsymbol{x}}_j) = \sum_{i,j=1}^{N_d} \delta_i \delta_j \sum_{l=1}^{N_r} \langle \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_i), \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_j) \rangle = \sum_{l=1}^{N_r} \sum_{i,j=1}^{N_d} \delta_i \delta_j \langle \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_i), \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_j) \rangle$$

$$= \sum_{l=1}^{N_r} \langle \sum_{i=1}^{N_d} \delta_i \, \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_i), \sum_{j=1}^{N_d} \delta_j \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_j) \rangle = \sum_{l=1}^{N_r} \left\| \sum_{i=1}^{N_d} \delta_i \, \boldsymbol{\varphi}_l(\grave{\boldsymbol{x}}_i) \right\|^2$$

$$= \sum_{l=1}^{N_r} \left\| \left(\sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \, , \sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \, x_1^i \;(k), \ldots , \sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \, x_n^i(k) \right)^T \right\|^2$$

$$= \sum_{l=1}^{N_r} \left(\left(\sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \right)^2 + \sum_{j=1}^{n} \left(\sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \, x_j^i(k) \right)^2 \right)$$

$$\geq \sum_{l=1}^{N_r} \left(\left(\sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \right)^2 \right) = \sum_{l=1}^{N_r} \left(\left(\sum_{i=1}^{N_d} \delta_i \mu_l(\boldsymbol{x}_i) \right)^2 \right)$$

Il est clair que la quantité : $\sum_{l=1}^{N_r} \left(\left(\sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \right)^2 \right) > 0, \; \forall \, \boldsymbol{x}_{i,j} \in \mathbb{R}^n, \exists \, \delta_i \neq 0 \; (i = 1, \ldots, N_d)$. Alors, la seule façon que la quantité $\sum_{l=1}^{N_r} \left(\left(\sum_{i=1}^{N_d} \delta_i \mu_l(\grave{\boldsymbol{x}}_i) \right)^2 \right)$ soit égale à zéro si pour tout $\delta_i = 0, i = 1, \ldots, N_d$. Notez que les fonctions d'appartenance: $\mu_l(\grave{\boldsymbol{x}}) = exp\left\{ - \sum_{j=1}^{n} \frac{(x_j - m_{lj})^2}{2\sigma_{lj}{}^2} \right\} > 0$. Par conséquent la fonction multi-noyau avec une vecteur d'entrée augmenté est une fonction noyau strictement positive définie et le modèle TS-LSSVR présenté dans l'équation (5.7) peuvent être utilisés pour définir le modèle flou de type TS-LSSVR.

Finalement, après avoir défini les fonctions des re-descriptions avec une vecteur d'entrée augmenté $\grave{\boldsymbol{x}}(k)$, nous utiliserons le même approche présenté précédemment au chapitre 3 pour obtenir les paramètres des conséquents, où la fonction multi-noyau avec une vecteur d'entrée augmentée est remplacée dans le modèle simplifié de la méthode de LSSVR (équation 5.7). Le résultat est donné par :

$$y(k) = \sum_{i=1}^{N_d} \alpha_i \kappa(\dot{\boldsymbol{x}}, \dot{\boldsymbol{x}}_{\boldsymbol{i}}) = \sum_{i=1}^{N_d} \alpha_i . \left(\sum_{l=1}^{N_r} \kappa_l(\dot{\boldsymbol{x}}, \dot{\boldsymbol{x}}_i) \right)$$

$$= \sum_{i=1}^{N_d} \alpha_i \left(\sum_{l=1}^{N_r} \langle \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}), \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}_i) \rangle \right)$$

$$= \sum_{i=1}^{N_d} \alpha_i . \left(\sum_{l=1}^{N_r} \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}_i)^{\boldsymbol{T}} \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}) \right)$$

$$= \sum_{l=1}^{N_r} \left(\sum_{i=1}^{N_d} \alpha_i . \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}_i)^{\boldsymbol{T}} \right) \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}})$$

$$= \sum_{l=1}^{N_r} \left(\sum_{i=1}^{N_d} \alpha_i . \dot{\boldsymbol{x}}_i{}^{\boldsymbol{T}} \mu_l(\boldsymbol{x}_i) \right) \dot{\boldsymbol{x}} . \mu_l(\boldsymbol{x})$$

Donc, si nous considérons : $\widehat{\boldsymbol{\theta}}_l = \left(\sum_{i=1}^{N_d} \alpha_i \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}_i) \right) = \sum_{i=1}^{N_d} \alpha_i \dot{\boldsymbol{x}}_i \mu_l(\boldsymbol{x}_i)$, alors le modèle flou de type TS-LSSVR présenté par l'équation (5.4) est obtenu. Donc, les paramètres des conséquences sont :

$$\widehat{\boldsymbol{\theta}}_l = \left(\sum_{i=1}^{N_d} \alpha_i \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}_i) \right) = \sum_{i=1}^{N_d} \alpha_i \dot{\boldsymbol{x}}_i \mu_l(\boldsymbol{x}_i) \, , l = 1, 2, \dots, N_r \tag{5.9}$$

et le modèle floue de type TS-LSSVR est donnée par :

$$y(k) = \sum_{l=1}^{N_r} \widehat{\boldsymbol{\theta}}_l^T \, \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}) = \sum_{l=1}^{N_r} \widehat{\boldsymbol{\theta}}_l^T \, \dot{\boldsymbol{x}} . \mu_l(\boldsymbol{x})$$

$$= \sum_{l=1}^{N_r} \left(\widehat{\boldsymbol{\theta}}_l^T . \dot{\boldsymbol{x}} \right) . exp \left\{ - \sum_{j=1}^{n} \frac{\left(x_j - m_{lj} \right)^2}{2 {\sigma_{lj}}^2} \right\} \tag{5.10}$$

Enfin, les paramètres des conséquences du modèle TS-LSSVR peuvent être identifiés à partir des multiplicateurs de Lagrange $\boldsymbol{\alpha} = \left(\alpha_1, \dots, \alpha_{N_d} \right)^T$ qui peuvent être facilement obtenus à partir d'équation (5.7). Les multiplicateurs de Lagrange $\boldsymbol{\alpha}$ sont donnés par :

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \gamma^{-1} \boldsymbol{I})^{-1} \boldsymbol{y} \tag{5.11}$$

où $\gamma > 0$, $\boldsymbol{y} = \left(y_1, \dots, y_{N_d}\right)^T$, $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d)^T$ et $\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \kappa\left(\boldsymbol{x}_1, \boldsymbol{x}_{N_d}\right) \\ \vdots & \dots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & \kappa\left(\boldsymbol{x}_N, \boldsymbol{x}_{N_d}\right) \end{pmatrix}$.

Dans la prochaine section, la méthode de C-moyennes flous pour l'initialisation des paramètres des antécédents ($\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$) est présentée. Le nombre de règles sera défini comme le nombre de groupes, et $\boldsymbol{m}_l$ et $\boldsymbol{\sigma}_l$ sont respectivement le centre et la variance de chaque groupe l. L'identification en ligne des multiplicateurs de Lagrange et l'adaptation des paramètres des antécédents seront ensuite définies.

5.4 L'initialisation des paramètres des antécédents

Dans cette section, les paramètres des antécédents du modèle flou de type TS-LSSVR seront initialisées à partir d'une méthode de regroupement connu sous le nom : C-moyennes flous (*Fuzzy C-means* : FCM) [94, 95].

5.4.1 La méthode de C-moyennes flous

La méthode de FCM a été d'abord introduit par Dunn [94] en 1974 et amélioré plus tard par Bezdek [95]. Dans cette méthode, le nombre de groupes est prédéfini alors que cette méthode a la capacité de déterminer itérativement les values des fonctions d'appartenances de chaque point de donnée. L'idée principale de cette méthode est que chaque point de donnée appartient à tous les groupes avec des valeurs d'appartenance correspondantes. Cette méthode est basée sur la minimisation de la fonction objective de la forme :

$$J_c(\boldsymbol{X}, \boldsymbol{U}_c, \boldsymbol{M}_c) = \sum_{i=1}^{N_r} \sum_{j=1}^{N_d} (\hat{\mu}_{\mathcal{A}_i}(j))^{\eta} d_{ij}\left(\boldsymbol{x}_j(k), \boldsymbol{m}_i\right)^2 \tag{5.12}$$

avec $\boldsymbol{U}_c = \left[\mu_{\mathcal{A}_i}(j)\right] \in \mathbb{R}^{N_d \times N_r}$ est la matrice de partition floue, $\boldsymbol{X}$ est la matrice des données, $\boldsymbol{M}_c = [\boldsymbol{m}_1, \boldsymbol{m}_2, \dots, \boldsymbol{m}_{N_r}]^T$ est un vecteur qui contient tous les centres des groupes, $d_{ij}\left(\boldsymbol{x}_j(k), \boldsymbol{m}_i\right) = \left(\boldsymbol{x}_j(k) - \boldsymbol{m}_i\right)^T \left(\boldsymbol{x}_j(k) - \boldsymbol{m}_i\right)$ est une norme de distance qui définit la mesure de distance entre l'observation $\boldsymbol{x}_j(k)$ et le centre $\boldsymbol{m}_i$, $\hat{\mu}_{\mathcal{A}_i}(j) \in [0, 1], j = 1, \dots, N_d$ est le degré d'activation correspondant au vecteur de données $\boldsymbol{x}_j(k)$ et $\eta \in [1, \infty]$ est un facteur qui indique le degré de la partition flou.

Les fonctions d'appartenance obtenues par la minimisation de la fonction objectif définie par l'équation (5.12) sont présentées par :

$$\hat{\mu}_{\mathcal{A}_i}(j) = \left(d_{ij}\left(\boldsymbol{x}_j(k), \boldsymbol{m}_i\right)^2 \sum_{l=1}^{N_r} \left(d_{lj}\left(\boldsymbol{x}_j(k), \boldsymbol{m}_l\right)^2 \right)^{\frac{1}{\eta-1}} \right)^{-1}, i = 1,2, \dots, N_r \tag{5.13}$$

où les vecteurs des centres et les vecteurs de variances des groupes sont respectivement définis par les équations (5.14) et (5.15) :

$$\boldsymbol{m}_i = \frac{\sum_{j=1}^{N_d} \boldsymbol{x}_j(k) \hat{\mu}_{\mathcal{A}_i}(j)^\eta}{\sum_{j=1}^{N_d} \hat{\mu}_{\mathcal{A}_i}(j)^\eta}, i = 1,2, \dots, N_r \tag{5.14}$$

$$\sigma_{iq} = \sqrt{\frac{2\sum_{j=1}^{N_d}\left(x_{jq}(k) - m_{iq}\right)^2 \hat{\mu}_{\mathcal{A}_i}(j)^\eta}{\sum_{j=1}^{N_d} \hat{\mu}_{\mathcal{A}_i}(j)^\eta}}, i = 1, \dots, N_r, q = 1, \dots, n \tag{5.15}$$

Dans ce travail, la valeur de η est égale à 1.6.

5.5 L'adaptation des paramètres des antécédents

Dans ce chapitre, l'algorithme de rétro-propagation est encore utilisé pour faire la mise à jour des paramètres des antécédents où l'erreur quadratique entre la prédiction du modèle floue de type TS-LSSVR et la sortie réelle du système est donnée par :

$$e(k) = \frac{1}{2}(y(k) - \hat{y}(k))^2 \tag{5.16}$$

avec $\hat{y}(k)$ et $y(k)$ sont respectivement les valeurs de la sortie prédite et la sortie réel du système. Enfin, les valeurs ajuster de m_{ij} et σ_{ij} à l'instant $k+1$ sont données par :

$$m_{ij}(k+1) = m_{ij}(k) - \eta \frac{\partial e(k)}{\partial m_{ij}} = m_{ij}(k) + \eta(y(k) - \hat{y}(k)) \frac{\partial \hat{y}(k)}{\partial m_{ij}} \tag{5.17}$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \eta \frac{\partial e(k)}{\partial \sigma_{ij}} = \sigma_{ij}(k) + \eta(y(k) - \hat{y}(k)) \frac{\partial \hat{y}(k)}{\partial \sigma_{ij}} \tag{5.18}$$

Avec η est une constante positive appelée taux, ou pas, d'apprentissage donnée par : $\eta = \frac{\beta}{\sum_{j=1}^{N}\sum_{i=1}^{n}\left(\left(\frac{\partial \hat{y}(k)}{\partial m_{ij}}\right)^2+\left(\frac{\partial \hat{y}(k)}{\partial \sigma_{ij}}\right)^2\right)}$, $\beta = 0.09$ et :

$$\frac{\partial \hat{y}(k)}{\partial m_{ij}} = \frac{\partial \hat{y}(k)}{\partial \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}})}.\frac{\partial \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}})}{\partial \mu_{A_j^i}(x_j)}.\frac{\partial \mu_{A_j^i}(x_j)}{\partial m_{ij}} = \boldsymbol{\varphi}_l{}^{\boldsymbol{T}}(\dot{\boldsymbol{x}}).\left(\boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}).\left(\sum_{l=1}^{M}\alpha_i\right)+\widehat{\boldsymbol{\theta}}_i\right).\frac{(x_j-m_{ij})}{{\sigma_{ij}}^2} \tag{5.19}$$

$$\frac{\partial \hat{y}(k)}{\partial \sigma_{ij}} = \frac{\partial \hat{y}(k)}{\partial \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}})}.\frac{\partial \boldsymbol{\varphi}_l(\dot{\boldsymbol{x}})}{\partial \mu_{A_j^i}(x_j)}.\frac{\partial \mu_{A_j^i}(x_j)}{\partial \sigma_{ij}} = \boldsymbol{\varphi}_l{}^{\boldsymbol{T}}(\dot{\boldsymbol{x}}).\left(\boldsymbol{\varphi}_l(\dot{\boldsymbol{x}}).\left(\sum_{l=1}^{M}\alpha_i\right)+\widehat{\boldsymbol{\theta}}_i\right).\frac{(x_j-m_{ij})^{\boldsymbol{2}}}{{\sigma_{ij}}^3} \tag{5.20}$$

avec M est la dimension du dictionnaire utilisé par la méthode de moindres carrés récursifs à noyau avec un budget fixe (FB-KRLS).

5.6 L'identification en ligne des multiplicateurs de Lagrange

Dans ce chapitre, l'identification en ligne des multiplicateurs de Lagrange est faite par la méthode de moindres carrés récursifs à noyau avec un budget fixe (FB-KRLS) [84]. Similaire à la méthode de SW-KRLS [73], la méthode de FB-KRLS stocke uniquement M données d'entrées-sorties dans le dictionnaire, ce qui réduira les coûts de calcul. Cependant, la méthode de SW-KRLS ajoute la paire de donnée arrivée au dictionnaire tandis que la plus ancienne est rejetée, ce qui peut réduire la précision de cette méthode où parfois les données les plus pertinentes sont les plus anciennes. La méthode de FB-KRLS a corrigé cette faiblesse en rejetant les données les moins pertinentes du dictionnaire et en la remplaçant par la nouvelle paire de donnée.

La méthode FB-KRLS peut être facilement réalisée basée sur les étapes de « *upsizing* » et « *downsizing* » de la méthode de SW-KRLS. Il suffit simplement d'ajouter deux autres étapes. Dans la première étape, un critère d'élagage est introduit pour rejeter les données les moins

pertinentes (significatives) où les erreurs de toutes les paires de données stockées dans le dictionnaire sont vérifiées. Ensuite, une stratégie de permutation sera utilisée comme une deuxième étape pour déplacer les données les moins pertinentes vers la première rangée et colonne de la matrice $\acute{\boldsymbol{K}}_k$ à supprimer (voire équation 3.22). Les données rejetées sont sélectionnées en fonction de l'erreur (critère) suivante [84] :

$$\boldsymbol{E_r}(\boldsymbol{x}_l(k), y_l(k)) = \frac{|\boldsymbol{\alpha}_l|}{\left[\acute{\boldsymbol{K}}_k^{-1}\right]_{l,l}} \tag{5.21}$$

Selon le critère dans l'équation (5.21), les données les moins significatives sont identifiées par leur valeur d'erreur (la paire de donnée qui a la plus petite erreur). Une fois la donnée le moins pertinente identifiée elle est supprimée, la nouvelle matrice $\acute{\boldsymbol{K}}_k$ ainsi que la matrice inverse sont ensuite calculées. Dans ce cas, les étapes de la méthode de SW-KRLS sont utilisées afin d'obtenir la matrice de noyau inverse réduite où l'indice l de donnée à supprimer peut être n'importe quel nombre (pas nécessairement la paire de donnée le plus ancienne). Dans ce chapitre, une stratégie de permutation simple est utilisée pour réduire les coûts de calcul de la méthode de FB-KRLS. Premièrement, la l-ème rangée et la colonne de la matrice $\acute{\boldsymbol{K}}_k^{-1}$ sont rejetées, et la matrice résultante $\acute{\boldsymbol{K}}_k^{-1}$est sélectionnée comme $\boldsymbol{V}$ dans l'équation (3.26). Ensuite, les données rejetées sont utilisées telles que : $q = \overline{\boldsymbol{K}}_k^{-1}(l,l)$ et $\boldsymbol{E} = (\kappa(\boldsymbol{x}(l), \boldsymbol{x}(k-M)), \dots, \kappa(\boldsymbol{x}(l), \boldsymbol{x}(k-1)))^T$. Ensuite, équation (3.27) est utilisée pour calculer la matrice inverse $\overline{\boldsymbol{K}}_k^{-1}$. Après avoir obtenu les multiplicateurs de Lagrange, la mise à jour des paramètres des conséquents $\widehat{\boldsymbol{\theta}}_l, l = 1, \dots, N_r$ sont fait en utilisant seulement M paires de données d'entrée-sortie telles que :

$$\widehat{\boldsymbol{\theta}}_l = \sum_{i=1}^{M} \alpha_i \grave{\boldsymbol{x}}_i exp\left\{ -\sum_{j=1}^{n} \frac{\left(x_j - m_{lj}\right)^2}{2{\sigma_{lj}}^2} \right\} \tag{5.22}$$

Enfin, la méthode de FB-KRLS est résumée comme suit :

Algorithme

1. Calculer l'erreur $e_k(k)$
2. Calculer la matrice inverse donnée par l'équation (3.23)
3. Si la taille de dictionnaire est supérieure à M Alors
   (1) Déterminer la paire le moins significatif en utilisant l'équation (5.21).
   (2) Faire la permutation de l-ième rangée et la colonne de la matrice $\acute{\boldsymbol{K}}_k^{-1}$.
   (3) Supprimer la l-ième rangée et la colonne de la matrice $\acute{\boldsymbol{K}}_k^{-1}$.
   (4) Redéfinir q et $\boldsymbol{E}$ et obtenir la matrice inverse dans l'équation. (3.27).
   (5) Calculer les multiplicateurs de Lagrange donnés par l'équation (3.28).
4. Sinon :
   (1) Calculer les multiplicateurs de Lagrange donnés par l'équation (3.24)

Algorithme 5.1 : L'algorithme de FB-KRLS

L'identification en ligne du modèle floue de type TS-LSSVR est résumée par l'algorithme (5.2) comme suit :

Algorithme

1. Initialisation : la taille de vecteur d'entrée, nombre des groupes N_r, la taille du dictionnaire M, ... etc.
2. Réaliser l'algorithme de C-moyennes flou pour initialiser les paramètres d'antécédents (hors ligne).
3. Mesurer le signal de sortie.
4. Ajuster les paramètres des antécédents m_{ij} et σ_{ij} dans les équations (5.17) et (5.18), respectivement.
5. Réaliser l'algorithme (5.1) pour obtenir les multiplicateurs de Lagrange.
6. Obtenir les paramètres des conséquences $\widehat{\boldsymbol{\theta}}_l$ (l'équation 5.22).
7. Retourner à l'étape 3

Algorithme 5.2 : Identification en ligne du modèle floue de type TS-LSSVR

Après avoir présenté le modèle flou de type TS-LSSVR adaptatif, la section suivante explique comment le modèle TS-LSSVR est intégré dans la commande GPC afin de contrôler les systèmes non linéaires.

5.7 La commande Prédictive généralisée floue avec une identification en ligne

Comme présenté précédemment au chapitre 3, l'idée principale de la commande GPC floue est que la fonction non linéaire $f(*)$ (voire équation 3.30) est approximée par plusieurs modèles linéaires locaux en utilisant les règles floues suivantes de type TS :

$$\begin{aligned} R_i: \quad & \text{Si } x_1(k) \; est \; A_1^i, et \; \dots, et \; x_n(k) \; est \; A_n^i \\ & \text{Alors } y_i(k) = a_i(z^{-1})y(k-1) + b_i(z^{-1})u(k-d-1) + C_i \\ & i = 1, \dots, N_r \end{aligned} \tag{5.23}$$

où le vecteur d'entrée est donnée par: $x(k) = \big(y(k-1), \dots, y(k-n_a), u(k-d-1), \dots, u(k-n_b)\big)$,$a_i(z^{-1})$, $b_i(z^{-1})$ et C_i sont des polynômes linéaires avec $n_a \leq n_y$, $n_b \leq n_u$ et $n_c = 0$, tel que:

$$\begin{aligned} a_i(z^{-1}) &= a_i^1 z^{-1} + \cdots + a_i^{n_a} z^{-n_a} \\ b_i(z^{-1}) &= b_i^0 + b_i^1 z^{-1} + \cdots + b_i^{n_b} z^{-n_b} \end{aligned} \tag{5.24}$$

Donc, le modèle flou de type TS-LSSVR est donnée par :

$$\begin{aligned} \hat{y}(k) = \sum_{i=1}^{N_r} & (a_i(z^{-1})y(k-1) + b_i(z^{-1})u(k-d-1) \\ & + C_i).exp\left\{-\sum_{j=1}^{n}\left(\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2}\right)\right\} \end{aligned} \tag{5.25}$$

avec $\widehat{\boldsymbol{\theta}}_i = (a_i,\ b_i, C_i)\ i = 1, \dots, N_r$ (obtenu par l'équation 5.22). Le modèle flou dans l'équation (5.25) peut être réécrit dans une représentation de la forme CARIMA :

$$\bar{A}(z^{-1})y(k) = \bar{B}(z^{-1})u(k-d-1) + \frac{\xi(k)}{\Delta} \tag{5.26}$$

où $\bar{A}(z^{-1}) = 1 - \bar{a}^1 z^{-1} - \cdots - \bar{a}^{n_a} z^{-n_a}$, $\bar{B}(z^{-1}) = \bar{b}^0 + \bar{b}^1 z^{-1} + \cdots + \bar{b}^{n_b} z^{-n_b}$, $\Delta = 1 - z^{-1}$ est représenté l'opérateur différentiel d'ordre 1, $\xi(k)$ est un bruit blanc gaussien de moyenne égal a $\bar{C}$, avec :

$$\bar{a}^j = \sum_{i=1}^{M} a_i^j . exp\left\{ -\sum_{j=1}^{n} \left(\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2} \right) \right\}$$
$$\bar{b}^j = \sum_{i=1}^{M} b_i^j . exp\left\{ -\sum_{j=1}^{n} \left(\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2} \right) \right\} \quad (5.27)$$
$$\bar{C} = \sum_{i=1}^{M} C_i . exp\left\{ -\sum_{j=1}^{n} \left(\frac{\left(x_j - m_{ij}\right)^2}{2{\sigma_{ij}}^2} \right) \right\}$$

La constante $\bar{C}$ est intégrée au bruit blanc, et la fonction $\xi(k)$ devient le bruit blanc gaussien avec une moyenne égale à $\bar{C}$. Enfin, le calcul de la prédiction et la loi de commande de méthode GPC sont présentés dans l'annexe A. Le modèle flou de type TS-LSSVR est donc utilisé en ligne pour donner les prédictions de la sortie à chaque instant, et la fonction de coût de la commande prédictive est aussi optimisée en ligne à chaque itération. Le schéma de la commande TS-LSSVR GPC adaptative est donné par la figure (5.1).

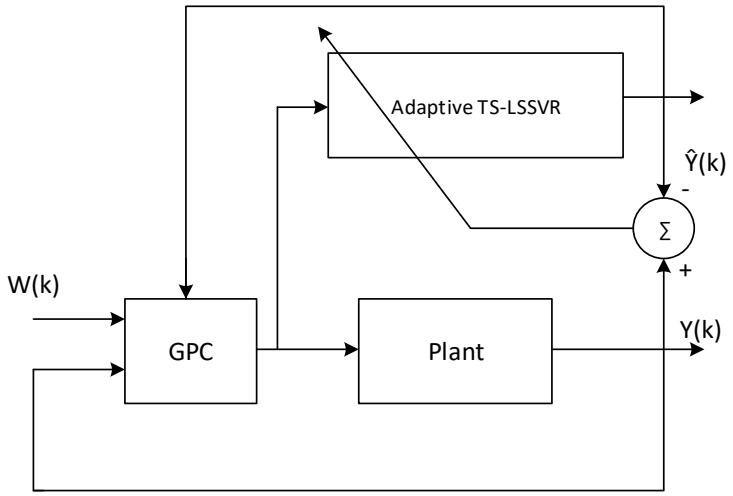

Figure 5.1 : Le schéma du la commande TS-LSSVR GPC adaptative

La commande TS-LSSVR GPC adaptatif est résumée comme suit :

Algorithme

1. Spécifier le signal de référence $W(k)$.
2. Sélectionner les paramètres d'identification tels que: nombre de règles floues et les paramètres de GPC (N_p, N_u et ϑ).
3. Réaliser la méthode de C-moyennes flous pour initialisée les paramètres des antécédentes.
4. Mesurer le signal de sortie.
5. Calculer les paramètres : m_{ij}, σ_{ij} et η.
6. Appliquer Algorithme 5.1 pour obtenu les multiplicateurs de Lagrange et ensuite calculer les paramètres conséquences $\widehat{\boldsymbol{\theta}}$ dans l'équation (5.22), et puis obtenez $\bar{A}(z^{-1})$, $\bar{B}(z^{-1})$ et $\bar{C}$.
7. L'incrément du signal de commande $\Delta u(k)$ est calculé en utilisant l'équation (A.15).
8. Le signal de commande $\Delta u(k) + u(k-1)$ est appliqué au système.
9. Répéter les étapes 4 - 8.

Algorithme 5.3 : La commande TS-LSSVR GPC adaptatif

Dans la section suivante, nous présentons les résultats de simulation obtenus par l'approche proposée de TS-LSSVR adaptatif ou l'identification en ligne et la commande de concentration d'un réacteur exothermique sont discutés en détails.

5.8 Exemple: identification et contrôle d'un réacteur exothermique

Dans ce chapitre, nous considérons le même système CSTR pour étudier la performance du modèle flou de type TS-LSSVR adaptatif dans l'identification et la commande du système non linéaire.

5.8.1 Identification en ligne du système CSTR

La procédure d'identification en ligne du système CSTR est réalisée par un ensemble de données d'apprentissage généré à partir du modèle mathématique du système. La période d'échantillonnage du système est fixée à $t_s = 6$ seconde et le modèle mathématique présenté dans l'équation (4.6) est utilisé pour générer cet ensemble ($N_d = 900$). Les premiers 400 échantillons sont utilisés pour construire le modèle flou de type TS-LSSVR, alors que le reste de ces échantillons est utilisé pour la validation du modèle. Le signal de commande représenté dans

la figure (5.2) a été utilisé pour générer les 900 échantillons. La taille du vecteur d'entrée est fixée à 9 ($n_a = 5, n_b = 3$ et l'augmentation de vecteur d'entrée).

Le signal de commande utilisé pour générer les échantillons d'apprentissage a les mêmes niveaux que celui présenté précédemment à la figure (4.13). Cependant, une perturbation avec une forme gaussienne et une amplitude de 10 est ajoutée à ce signal de commande.

De plus, l'approche basée sur l'indice de Dunn est utilisée pour identifier le nombre convenable de groupes où le nombre convenable de groupes trouvé est $N_r = 7$. Le constant γ est égal à 1000, le β de taux d'apprentissage est égal à 0.09 et la taille de dictionnaire $M = 50$. La méthode de C-moyennes flous est utilisée pour initialiser les paramètres des antécédents. Après avoir exécuté l'algorithme (5.2), la prédiction de la concentration du produit A_a est présentée dans la figure (5.3).

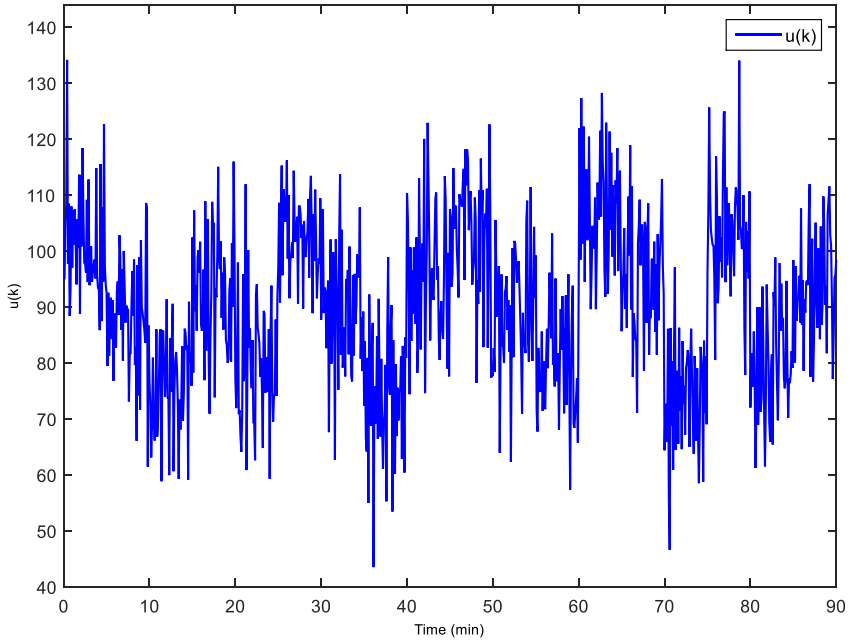

Figure 5.2 : Le signal de commande utilisée pour générer l'ensemble de 900 échantillons du système CSTR

On peut voir que le modèle flou de type TS-LSSVR adaptatif s'est bien comporté et a produit des résultats de prédiction très précis. De plus, la valeur d'erreur obtenue par le modèle TS-LSSVR adaptatif est très faible bien que les bruits qui ont été injectés au signal d'entrée soient très élevés ($RMSE = 7.031 \times 10^{-3}$).

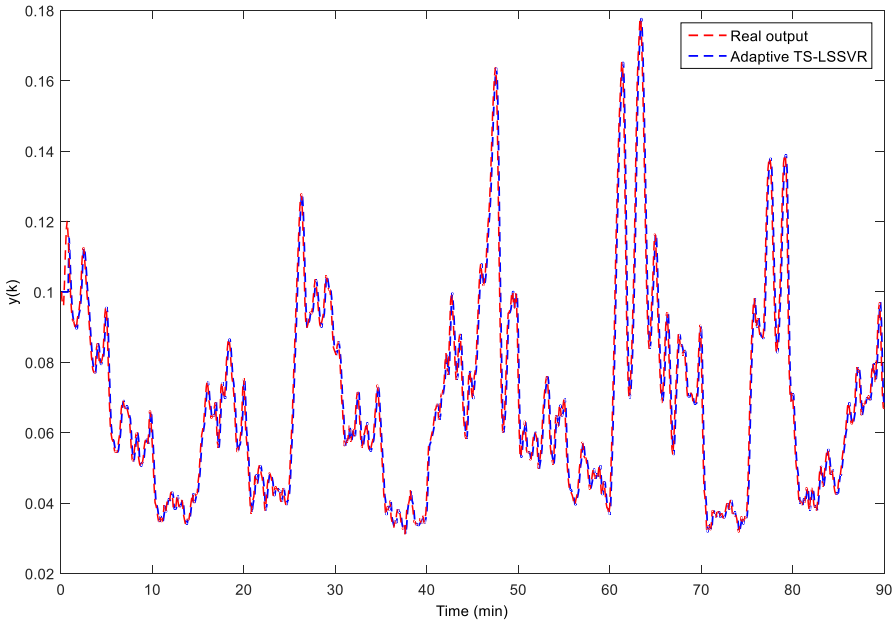

Figure 5.3 : La performance de modélisation à partir de modèle TS-LSSVR adaptatif

Dans la section suivante, le modèle flou de type TS-LSSVR adaptatif est intégré à la commande GPC et utilisé pour contrôler le système non linéaire CSTR.

5.8.2 La commande prédictive floue adaptatif du système

Les paramètres du modèle flou de type TS-LSSVR utilisés pour commander le système sont les mêmes paramètres que dans la partie d'identification en ligne. Les paramètres de commande GPC sont : $N_p = 10$, $N_u = 2$ et $\vartheta = 0.0008$. Après avoir exécuté l'algorithme (5.3), la concentration du système et le signal de commande appliqué au système sont illustrés respectivement dans les figures (5.4) et (5.5).

En outre, les résultats obtenus par la commande ANFIS GPC (les paramètres du contrôle de ANFIS GPC sont : $N_p = 10$, $N_u = 2$, $\vartheta = 0.00076$ et le nombre de règles floues est $N_r = 20$) et la commande ATSK GPC (les paramètres du contrôle de ATSK GPC sont : $N_r = 20$, $N_p = 10$, $N_u = 2$, $\vartheta = 0.00076$ et $N_r = 20$) sont aussi illustrés dans la figure (5.4).

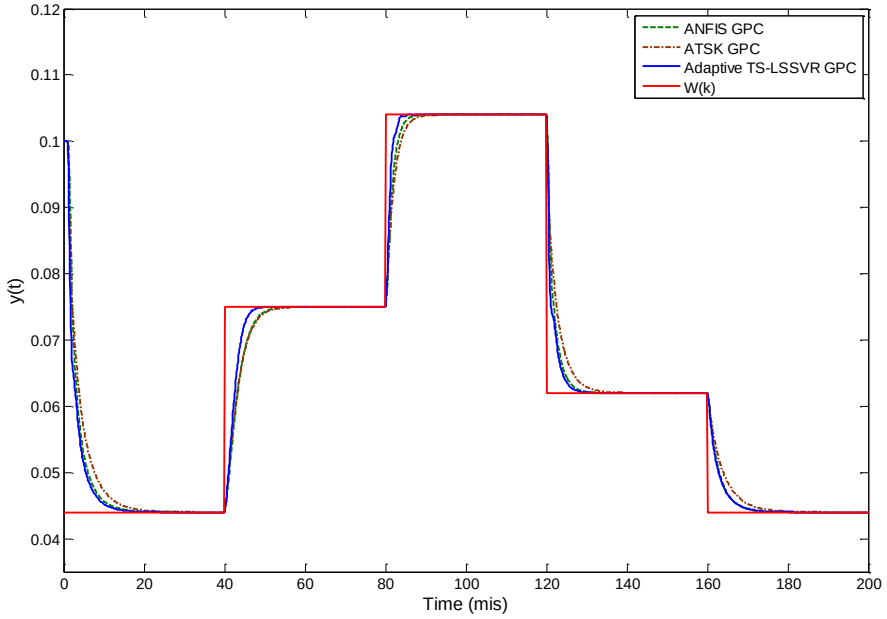

Figure 5.4 : Les résultats du commande TS-LSSVR GPC adaptatif

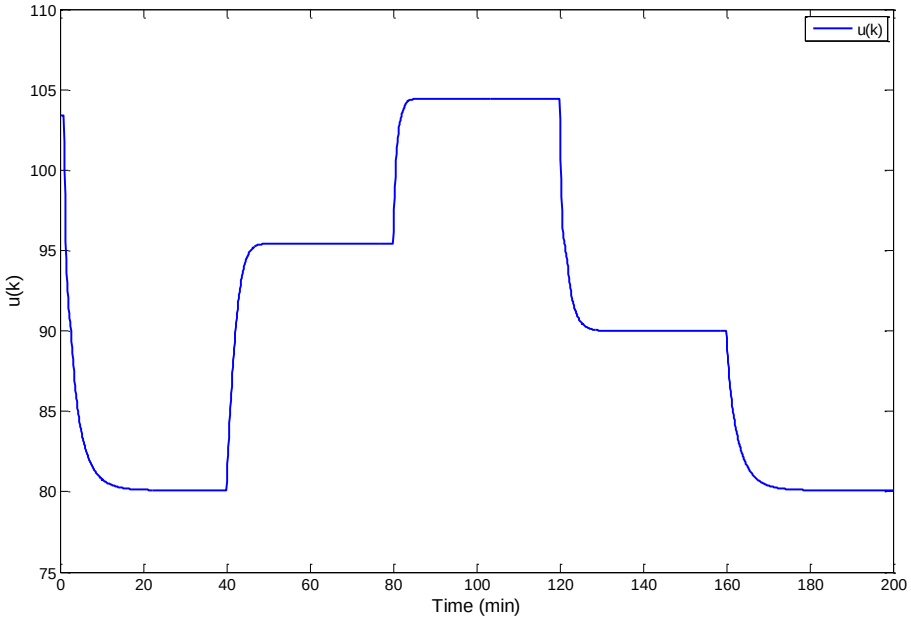

Figure 5.5 : Le signal de commande appliqué au système

Les résultats dans la figure (5.4) montrent que la sortie du système est totalement confondue avec la référence $W(k)$, ce qui montre que la méthode TS-LSSVR GPC adaptative utilisée est

efficace. Les mêmes commentaires peuvent être faits pour les résultats obtenus par les commandes ANFIS GPC et ATSK GPC. On peut remarquer aussi que la stratégie de commande TS-LSSVR GPC adaptative est plus rapide que la régulation par les méthodes ANFIS GPC et ATSK GPC. La valeur moyenne pour exécuter une itération pour une taille de dictionnaire de $M = 50$ est 0.00797 seconde, ce qui est considéré comme une période de temps compatibles avec la plupart des systèmes réels. D'autre part, les commandes ANFIS GPC et ATSK GPC ont des temps d'exécution plus petit car ces algorithmes ne demandent pas un grand nombre de données précédentes.

Nous avons injecté une perturbation au système contrôlé pour tester la robustesse de stabilité de la méthode de commande proposée. Les perturbations appliquées au système sont définies comme des constantes où une perturbation avec l'amplitude de 0.012 a été appliquée au système dans l'intervalle de temps $60 \leq k \leq 100$ minutes. Une autre perturbation d'amplitude 0.022 a été appliquée au système dans l'intervalle $100 < k \leq 140$ minutes. L'objectif de la poursuite de trajectoire de référence est atteint en présence des perturbations, les figures (5.6) et (5.7) montre les résultats de cette simulation.

Nous remarquons que les performances obtenues avec la commande TS-LSSVR GPC adaptative sont meilleures que les autres méthodes. Dans le cas du rejet de perturbation, la commande proposée présente de plus petits dépassements en régime dynamique du système que les commandes prédictives basées sur les modèles ANFIS et ATSK.

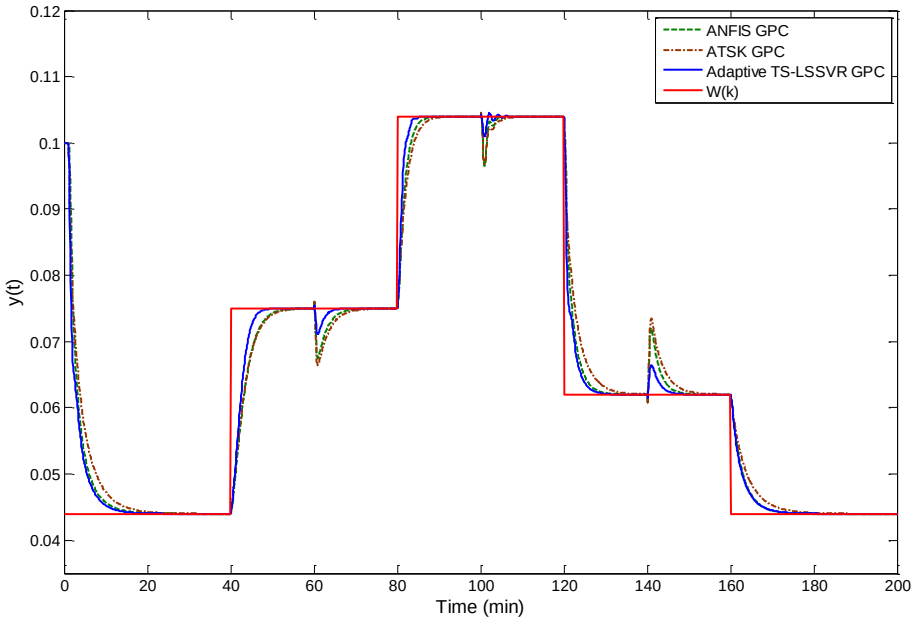

Figure 5.6 : Les résultats du commande TS-LSSVR GPC adaptatif

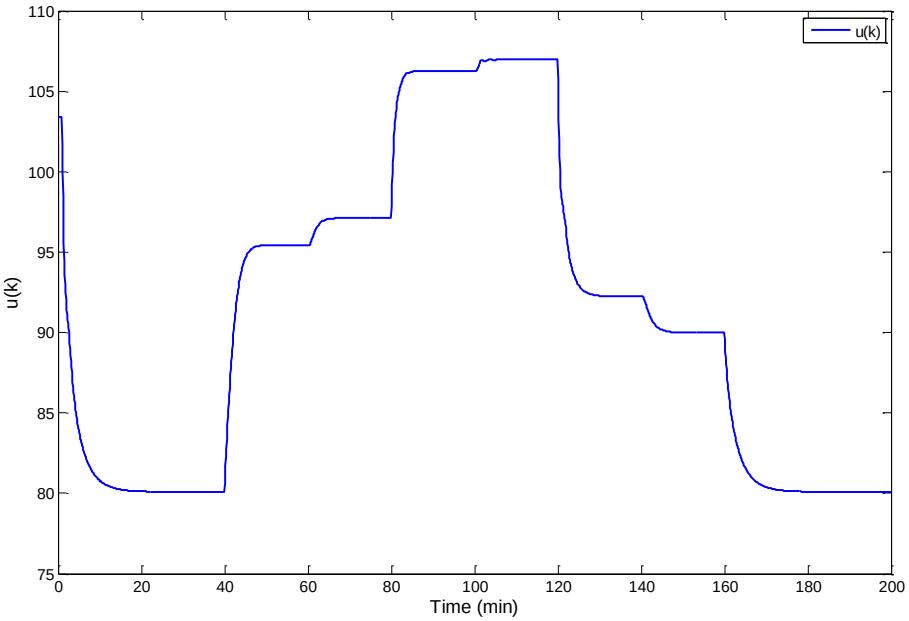

Figure 5.7 : Le signal de commande appliqué au système

Conclusion Générale

La discipline de modélisation floue a évolué d'une façon progressive, vers une utilisation des systèmes flous dans lesquels la conséquence des règles floues utilise des variables numériques sous la forme de fonctions (Takagi-Sugeno) plutôt que des variables linguistiques (Mamdani). Suivant cette évolution, nous avons proposé dans cette thèse, de nouvelles approches de modélisation floue basées sur les méthodes à noyaux et les approches des partitionnements. Ces méthodes d'approximation sont basées sur le théorème d'approximation universelle des systèmes à logique floue de type Takagi-Sugeno. La nouvelle stratégie proposée de modélisation floue de type Takagi-Sugeno est basée sur deux étapes fondamentales :

- La première étape concerne le partitionnement des données d'apprentissage pour définir la structure de modèle flou. Nous avons utilisé deux méthodes de partitionnement : une méthode de partitionnement basée sur l'algorithme de PSO et la méthode de C-moyennes flous. Les méthodes de partitionnement sont donc utilisées pour partitionner les données d'entrées-sorties en groupes (clusters) où chaque groupe décrit une règle floue (tout simplement les paramètres du groupe i sont les paramètres des antécédents de la règle i). Les méthodes de partitionnement réduisent automatiquement le nombre des modèles locaux et par conséquent le nombre des paramètres à estimer. Cette réduction des paramètres est très bénéfique dans le sens qu'elle réduit la charge des calculs et facilite la mise en œuvre de l'algorithme. Nous avons aussi présenté une simple méthode pour optimiser le nombre des règles floues (nombre des groupes).
- Dans la deuxième étape, les paramètres des sous-systèmes linéaires (les paramètres des conséquences) sont identifiés à partir des méthodes à noyaux. Deux méthodes à noyaux ont été utilisées pour identifier les paramètres des conséquences: la méthode de régression ridge à noyau (KRR) et la méthode de régression de la machine à vecteurs de support en moindres carrés (LSSVR). Les méthodes de régression à noyaux ont la capacité de généralisation et de réaliser une régression non linéaire de manière efficace et n'ont pas des inconvénients tels que les problèmes de plusieurs minima locaux et sur-ajustement.

Enfin, deux exemples illustratifs sont utilisés pour analyser l'identification hors ligne et en ligne de ces nouvelles méthodes floue. Ces méthodes ont montré l'applicabilité des méthodes à noyaux avec une meilleure qualité d'approximation. De plus, nous avons réalisé une étude comparative entre nos modèles floue de type TS basée sur des méthodes à noyaux et d'autres approches de modélisation utilisées dans le domaine d'identification floue. Nous pensons que l'identification des paramètres des conséquences avec des méthodes à noyaux peut donner une meilleure précision de la modélisation.

Après la modélisation des systèmes, nous avons considéré la synthèse de la commande prédictive généralisée à base de modèle flou. Nous avons choisi la modélisation floue de type TS comme le meilleur candidat pour le développement de la version non linéaire de la commande GPC. Une version non linéaire de la commande GPC basée sur le modèle flou de type TS-KRR est ensuite proposée. Deux méthodologies de mise en œuvre de la commande GPC non linéaire ont été introduites : l'une fondée sur l'apprentissage hors ligne d'un modèle floue où les paramètres de modèle sont identifiés hors ligne, et une fois le modèle flou de type TS-KRR est validé (devient un modèle fixe), il est utilisé pour prédire à chaque instant la sortie du système pour déterminer le signal de commande. La seconde fondée sur l'identification en ligne d'un modèle flou de type TS-KRR pour l'extraction à chaque pas d'échantillonnage la sortie approximative du système pour déterminer le signal de commande. Les performances de ces deux techniques de commande ont été montrées à travers deux exemples illustratifs, et les avantages et inconvénients de chaque technique ont été énoncés.

De la même façon, nous avons proposé une version non linéaire de la commande GPC basée sur des modèles de type TS-LSSVR. La méthodologie de mise en œuvre en ligne de la commande GPC non linéaire basé sur le modèle flou de type TS-LSSVR a été introduite et les performances de cette technique de commande ont été montrées à travers un exemple illustratif. Dans la commande GPC non linéaire basé sur le modèle de TS-LSSVR, les résultats obtenus montrent l'efficacité de cette loi de commande dans ce genre d'application. En effet, nous avons injecté des perturbations dans le système à commander et ces perturbations ont rapidement été éliminées.

Dans les travaux à venir, nous essayent d'avoir une étude approfondie des équivalences, au niveau du comportement dynamique, entre les modèles flous basés sur les méthodes à noyaux et

le système non linéaire approximé. A notre connaissance, très peu des publications ont été consacrées à ce sujet. Cette étude s'avère intéressante afin d'établir des conditions d'équivalence qui permettraient d'assurer le même type de comportements dynamiques entre la représentation floue et le système. Aussi, le développement d'un observateur flou basé sur le modèle Takagi-Sugeno, afin d'estimer les variables non mesurables du système. Il est aussi important de tester nos modèles flous basés sur les méthodes à noyaux proposés dans le cas où aucune information sur le système n'est disponible, et la structure du modèle flou ne démarre qu'avec une seule règle. Dans ce cas, on n'a pas d'information a priori permettant d'initialiser la structure du modèle floue, main un flux d'échantillons de données est disponible. Comme une solution, une méthode de partitionnement en ligne est capable de construire la structure de modèle floue où la détermination du nombre de fonctions d'appartenance et la distribution de chacune d'elle est faite par la création, d'une manière adaptative, de nouvelles règles et l'ajustement des paramètres prémisses.

Bibliographie

[1] Prett, D.M., & Garcia, C.E.: Fundamental Process Control. Butterworths, Boston, 1988.

[2] Brujin, P.M.,& Verbruggen, H.B.: Model algorithmic control using impulse response models. Journal A,25(2), pp. 69-74, 1984.

[3] Marchetti, J.L., Mellichamp, D.A., & Seborg, D.E.: Predictive control based on discrete convolution models. Industrial Engineering Chemical Process Design and Development, 22, pp. 488-495, 1983.

[4] Clarke, D.W., & Mohtadi., C.: Properties of generalized predictive control. Automatica, 25(6), pp. 859-875, 1989.

[5] Camacho, F.E.: Constrained generalized predictive control. IEEE Transactions on Automatic Control, 38(2), pp. 327-332, 1993.

[6] Zadeh, L.A.: Fuzzy sets. Information and Control. 8 (3), pp. 338-353. ISSN 0019-9958, doi:10.1016/S0019-9958(65)90241-X, 1965.

[7] Zadeh, L.A.: Fuzzy algorithms. Information and Control. 12 (2), pp. 94-102,ISSN 0019-9958. doi:10.1016/S0019-9958(68)90211-8, 1968.

[8] Hebb, D.: The Organization of Behavior. New York: Wiley, ISBN 978-1-135-63190-1, 1949.

[9] Tsai, P.F., Chu, J.Z., Jang, S.S. & Shieh, S.S.: Developing a robust model predictive control architecture through regional knowledge analysis of artificial neural networks, Journal of Process Control, 13, pp. 423-435, 2002.

[10] Kittisupakom, P., Thitiyasook, P., Hussein, M. A. & Daosud, W.: Neural network based model predictive control for a steel pickling process, Journal of Process Control, 19(4), pp. 579-590, 2009.

[11] Alexandris, A.& Sarimvers, H.: Nonlinear adaptive model predictive control based on self-correcting neural network models, AI ChE Journal, 51(9), pp. 2495-2506, 2005.

[12] Atuonwu, J. C., Cao, Y., Rangaiah G. P., & Tad, M.O.: Identification and predictive control of a multistage evaporator, Control Engineering Practice, 18, pp.1418-1428, 2010.

[13] Takagi, T., & Sugeno, M.: Fuzzy identification of systems and its application to modeling and control, IEEE Trans. Syst., Man, Cybern, 15, pp. 116-132, 1985.

[14] Mollov, S., Babusˇka,R., Abonyi, J., Verbruggen, H.B.: Effective optimization for fuzzy model predictive control, IEEE Transactions on Fuzzy Systems 12(5), pp. 661-675, 2004.

[15] Sousa, J.M.D.C., & Kaymak, U.: Modèle prediction control using fuzzy decision functions, IEEE Transactions on System, Man, and Cybernetics- part B: Cybernetics, 31(1), pp. 54-65, 2001.

[16] Chang Y.L. & Tsai C.C.: Adaptive stable generalized predictive control using TSK fuzzy model for nonlinear discrete-time systems with time-delays. International Journal of Fuzzy Systems, 15(2), pp.133-141, 2013.

[17] Jang, J.S.R.: ANFIS: Adaptive-Network-based Fuzzy Inference Systems, IEEE Transactions on Systems, Man, and Cybernetics, 23(3), pp. 665-685, 1993.

[18] Zhang, Y., Chai, T., Wang, H., & Fu, J.: Generalized predictive control method for a class of nonlinear systems using ANFIS and multiple models. In: 49th IEEE conference on decision and control, Atlanta, GA, pp. 15-17, 2010.

[19] Cortes, C. &Vapnik, V.: Support-Vector Networks, Machine Learning, 20(3), pp. 273-297. 1995.

[20] Iplikci, S.: Support vector machines-based generalized predictive control. International Journal of Robust and Nonlinear Control, 16(17), pp. 843-862, 2006.

[21] Iplikci, S.: Support vector machines based generalized predictive control of chaotic systems. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 89(10), pp. 2787-2794, 2006.

[22] Iplikci, S.: Online trained support vector machines-based generalized predictive control of non-linear systems. International Journal of Adaptive Control and Signal Processing, 20(10), pp. 599-621, 2006.

[23] Guo, Z. & Guan, X.: Nonlinear generalized predictive control based on online least squares support vector machines. Nonlinear Dynamics, 79(2), pp.1163-1168, 2015.

[24] Li, L.J., Su, H.Y., & Chu, J.: Generalized predictive control with online least squares support vector machines, Acta Automatica Sinica 33(11), pp. 1182-1188, 2007.

[25] Chiang, J.H., & Hao, P.Y.: Support vector learning mechanism for fuzzy rule-based modeling: a new approach, IEEE Transactions on Fuzzy Systems 12 (1), pp. 1-11, 2004.

[26] Lin, C.T., Liang, S.F., Yeh, C.M., & Fan, K.W.: Fuzzy neural network design using support vector regression for function approximation with outliers, in: Proc. IEEE Internat. Conf. on System, Man and Cybernetics, 3, pp. 2763-2768, 2005.

[27] Juang, C.F.,& Hsieh, C.D.: TS-fuzzy system-based support vector regression, Fuzzy Sets and Systems 160, pp. 2486-2504, 2009.

[28] Boulkaibet, I., Belarbi, K., Bououden, S., Marwala, T., & Chadli, M.: A new T-S fuzzy model predictive control for nonlinear processes, Expert Systems with Applications, 2017.

[29] Juang, C. F., & Hsieh, C. D.: A fuzzy system constructed by rule generation and iterative linear SVR for antecedent and consequent parameter optimization. IEEE Transactions on Fuzzy Systems, 20(2), pp. 372-384, 2012.

[30] Aronszajn, N.: The theory of reproducing kernels and their applications, Cambridge Phil. Soc. Proc. 39, pp. 133-153, 1943.

[31] Aronszajn, N.: Theory of reproducing kernels, Trans. Am. Math. Soc. 68 (3), pp. 337-404, 1950.

[32] Cornuéjols, A., & Miclet, L.: Apprentissage artificiel: concepts et algorithmes. Editions Eyrolles, 2011.

[33] Chellappa, R., & Theodoridis, S.: Signal Processing Theory and Machine Learning. Academic Press, 2014.

[34] Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equation, Philos. Trans. Roy. Soc. Lond., 209, pp. 415-446, 1909.

[35] Mercer, J.: Sturm-Liuville series of normal functions in the theory of integral equations, Philos. Trans. Roy. Soc. Lond. Ser. A, 211, pp. 111-198, 1911.

[36] Boughorbel, S., Tarel, J. P., & Boujemaa, N.: Conditionally positive definite kernels for SVM based image recognition. In Multimedia and Expo, ICME 2005. IEEE International Conference on (pp. 113-116). IEEE, 2005.

[37] Moore, E.: On properly positive Hermitian matrices, Bull. Amer. Math. Soc. 23 (59), pp. 66-67, 1916.

[38] Kimeldorf, G.S., & Wahba, G.: Some results on Tchebycheffian spline functions, J. Math. Anal. Appl. 33, pp. 82-95, 1971.

[39] Kung, S. Y.: Kernel methods and machine learning. Cambridge University Press, 2014.

[40] Gunn, S. R.: Support vector machines for classification and regression. ISIS technical report, 14, pp. 85-86, 1998.

[41] Burges, C.J.C.: A tutorial on support vector machines for pattern recognition, Knowledge Discovery and Data Mining, 2(2), pp. 121-167, 1998.

[42] Cristianini, N., & Shawe-Taylor, J.: An Introduction to Support Vector Machines, Cambridge University Press, 2000.

[43] Schölkopf, B., Burges, & C. Smola, A.: Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, 1998.

[44] Schölkopf, B., & Smola, A.: Learning with Kernels, MIT Press, Cambridge, MA, 2002.

[45] Suykens, J. A., & Vandewalle, J.: Least squares support vector machine classifiers. Neural processing letters, 9(3), pp. 293-300, 1999.

[46] Suykens, J. A., Van Gestel, T., & De Brabanter, J.: Least squares support vector machines. World Scientific, 2002.

[47] Wu, W. Synthèse d'un contrôleur flou par Algorithme Génétique Application au réglage dynamique des paramètres d'un système, PhD dissertation, Université de Lille 1, 1998.

[48] Talbi, N. : Conception des Systèmes d'Inférence Floue par des Approches Hybr ides, PhD dissertation, Université de Constantine 1, 2014.

[49] Ouakka, H. : Contribution à l'identification et la commande floue d'une classe de systèmes non linéaires, Université Sidi Mohamed Ben Abdellah, PhD dissertation, 2009.

[50] Sahraoui, M. : Contrôle robuste des systèmes non linéaires par les approches de l'intelligence artificielle, PhD dissertation, Université d'Oran, 2016.

[51] Bezzini, A. : Commande prédictive non linéaire en utilisant les systèmes Neuro-Flous et les algorithmes Génétiques, Magister dissertation, Université Mohamed Khider-Biskra, 2013.

[52] El Koujok, M. : Contribution au pronostic industriel: intégration de la confiance à un modèle prédictif neuro-flou, PhD dissertation, Université de Franche-Comté, 2010.

[53] Mamdani, E.H. & Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies, 7(1), pp. 1-13, 1975.

[54] Palacio, V. H. G. : Modélisation et commande floues de type Takagi-Sugeno appliquées à un bioprocédé de traitement des eaux usées, Doctoral dissertation, Université Paul Sabatier-Toulouse III, 2007.

[55] Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M.: Learning the kernel matrix with semi-definite programming, J. Mach. Learn. Res. 5, pp. 27-72, 2004.

[56] Argyriou, A., Hauser, R., Micchelli, C., & Pontil, M.: A DC-programming algorithm for kernel selection, in: Proceedings of the 23rd International Conference on, Machine Learning, pp. 41-48, 2006.

[57] Girolami, M., &Rogers, S.: Hierarchic Bayesian models for kernel learning, in: Proceedings of the 22nd International Conference on, Machine Learning, pp. 241-248, 2005.

[58] Li, J. & Sun,S.: Nonlinear combination of multiple kernels for support vector machines, in: Proceedings of the 20th International Conference on, Pattern Recognition, pp. 1-4, 2010.

[59] Micchelli, A., & Pontil, M.: Learning the kernel function via regularization, J. Mach. Learn. Res., 6, pp. 1099-1125, 2005.

[60] Ong, C., Smola, A., &Williamson, R.: Learning the kernel with hyper kernels, J. Mach. Learn. Res., 6, pp. 1043-1071, 2005.

[61] Ying, Y., & Zhou, D.: Learnability of Gaussians with flexible variances, J. Mach. Learn. Res., 8, pp. 249-276, 2007.

[62] Zien, A., & Ong, C.: Multiclass multiple kernel learning, in: Proceedings of the 24th International Conference on, Machine Learning, pp. 1191-1198, 2007.

[63] Oviedo, J. J. E., Vandewalle, J. P., & Wertz, V.: Fuzzy logic, identification and predictive control, Springer Science & Business Media, 2006.

[64] Bououden, S., &Boulkaibet, I.: Commande prédictive floue : Simulation sous Matlab, Éditions universitaires européennes, 2017.

[65] Boucher, P., & Dumur, D.: La commande prédictive (Vol. 8). Editions Technip, 1996.

[66] Forgy, E.W.: cluster analysis of multivariate data: efficiency versus interpretability of classification (abstract), Biometrics 21, 768-769, 1965.

[67] Van der Merwe, D. W., & Engelbrecht, A. P.: Data clustering using particle swarm optimization. In Evolutionary Computation CEC'03, The 2003 Congress, 1, pp. 215-220, 2003.

[68] Chamroukhi, F. : Classification supervisée : Analyse discriminante, Licence 2 Sciences Pour l'Ingénieur, Université du Sud Toulon -Var, 2013.

[69] Kogan, J.: Introduction to Clustering Large and High-Dimensional Data, Cambridge University Press, Cambridge, 2007.

[70] Kennedy, J. & Eberhart, R. C.: Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks IV, pp. 1942-1948, Perth, Australia, November 1995.

[71] El Dor, A. : Perfectionnement des algorithmes d'optimisation par essaim particulaire: applications en segmentation d'images et en électronique, Doctoral dissertation, Université Paris-Est, 2012.

[72] Engel, Y., Mannor, S., & Meir, R.: The kernel recursive least squares algorithm. IEEE Transactions on Signal Processing, 52(8), pp. 2275- 2285, 2004.

[73] Van Vaerenbergh, S., Vía, J., & Santamaria, I.: A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In 2006 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), pp. 789-792, Toulouse, France, 2006.

[74] Saide, C. : Filtrage adaptatif à l'aide de méthodes à noyau: application au contrôle d'un palier magnétique actif, Doctoral dissertation, Troyes, 2013.

[75] Vapnik, V.: Estimation of dependencies based on empirical data. Springer Series in Statistics. Springer-Verlag, New York, 1982.

[76] Osuna, E., Freund, F., & Girosi, F.: An improved training algorithm for support vector machines. In Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Workshop, pages, pp. 276-285, 1997.

[77] Osuna, E., & Girosi, F.: Reducing the run-time complexity in support vector machines. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, Advances in Kernel Methods: Support Vector Learning, pp. 271-283, Cambridge, MA, USA, MIT Press, 1999.

[78] Joachims, T.: Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, Advances in Kernel Methods: Support Vector Learning, pp. 169-184, Cambridge, MA, USA, MIT Press, 1999.

[79] Frieb, T.T. Cristianini, N., &Campbell, C.: The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In Machine Learning: Proceedings of the Fifteenth International Conference. Morgan Kaufmann Publishers, 1998.

[80] Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, Advances

in Kernel Methods: Support Vector Learning, pp. 185-208, Cambridge, MA, USA, MIT Press, 1999.

[81] Vaerenbergh, S., Via, J., & Santamaria, I.: A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006), Toulouse, France, May 2006.

[82] Cavallanti, G., Cesa-Bianchi, N., & Gentile, C.: Tracking the best hyperplane with a simple budget perceptron. Mach. Learn., 69(2-3), pp. 143-167, December 2007.

[83] Dekel, O., Shalev-shwartz, S., & Singer, Y.: The forgetron: A kernel-based perceptron on a fixed budget. In In Advances in Neural Information Processing Systems 18, pp. 259-266. MIT Press, 2005.

[84] Van Vaerenbergh, S., Santamaría, I., Liu, W., & Príncipe, J.C.: Fixed-budget kernel recursive least-squares. In 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1882-1885, 2010.

[85] Eski, I., &Temürlenk, A.: Design of neural network-based control systems for active steering system, Nonlinear Dynamic, 73, pp. 1443-1454, 2013.

[86] Mendes, J., Araújo, R., & Souza, F.: Adaptive fuzzy identification and predictive control for industrial processes. Expert Systems with Applications, 40 (17), pp. 6964-6975, 2013.

[87] Dunn, J. C.: A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Groups, Journal of Cybernetics, 3(3), pp. 32-57, 1973.

[88] Dunn, J. C.: Well-Separated Groups and Optimal Fuzzy Partitions, Journal of Cybernetics, 4 (1), pp. 95-104, 1974.

[89] Jang, J. S. R.: Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm, Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91), pp. 762-767, 1991.

[90] Poggio, T., Mukherjee, S., Rifkin, R., Raklin, A. & Verri, A.: b, Uncertainty in Geometric Computations, Kluwer Academic Publishers, Boston, pp. 131-141.

[91] Vogt, M.,& Kecman, V.: Active-set methods for support vector machines. In Support vector machines: theory and applications: 133-158. Springer Berlin Heidelberg, 2005.

[92] Huang, T.M., & Kecman, V.: Bias Term b in SVMs again. In: Proceedings of the 12th European Symposium on Artificial Neural Networks (ESANN 2004), Bruges, Belgium, pp. 441-448, 2004.

[93] Kecman, V., Huang, T.M.,& Vogt, M.: Iterative single data algorithm for training kernel machines from huge data sets: Theory and performance. Support vector machines: Theory and Applications, 605-605, 2005.

[94] Dunn, J.C.: A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. J. Cybernet, (3), pp. 32-57, 1973.

[95] Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[96] McQueen, J. B.: Some methods of classification and analysis of multivariate observations. In L. L. Cam and J. Neyman, editors, Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, (1), pp. 281-297, Berkeley, University of California Press, 1967.

[97] Boulkaibet, I., Belarbi, K., Bououden, S., chadli, M., & Marwala T.: An Adaptive Fuzzy Predictive Control of Nonlinear Processes Based on Multi-Kernel Least Squares Support Vector Regression, Applied Soft Computing, Under Review, 2017.

Annexe A

A.1 La commande prédictive généralisée

L'idée de base de la commande prédictive généralisée (GPC) [4, 5, 63, 64, 65] la prédiction de la réponse d'un système à contrôler étalée sur un horizon de prédiction $[d+1,N_p]$. Le but de la méthode est de générer à un instant courant 'k' quelle que soit la valeur du retard du système d, un signal de commande $u(k)$, et ceci en calculant les prédictions de la sortie du processus $y(k)$, sur un intervalle $[d+1,\ N_p]$. Généralement, la commande GPC est basée sur un modèle linéaire entrée-sortie, à partir du quel seront dérivés les calculs. La loi de commande est donc obtenue analytiquement et seul le premier élément du vecteur commande futures est appliqué.

La loi de commande est obtenue par la minimisation d'un critère quadratique portant sur les erreurs futures avec un terme de pondération sur la commande [4, 28, 64, 65] :

$$J(k) = \sum_{j=d+1}^{N_p} \left(\hat{y}(k+j|k) - w(k+j)\right)^2 + \sum_{j=d+1}^{N_u+d-1} \vartheta(\mathrm{z}^{-1})\Delta u(k+j-d-1|k)^2 \tag{A.1}$$

où $\Delta u(k+j-d-1|k) = u(k+j-d-1) - u(k+j-d-2)$, et $y(k+j|k)$ sont des valeurs futures de la sortie à commander. Le critère quadratique déterminé par l'équation A.1 nécessite la définition de trois paramètres de réglage :

- N_p: horizon de prédiction maximal ;
- N_u: horizon de prédiction sur la commande ;
- $\vartheta(\mathrm{z}^{-1})$: coefficient de pondération sur la commande avec $\vartheta(\mathrm{z}^{-1}) = \vartheta_0 + \vartheta_1\mathrm{z}^{-1} + \cdots + \vartheta_{N_p+n_b-1}\mathrm{z}^{-(N_p+n_b-1)}$.

A.1.1 Calcul des prédictions

Quand on considère une régulation autour d'un point de fonctionnement particulier, généralement même un processus non linéaire, admet un modèle localement linéaire. Dans la commande GPC, le modèle classiquement utilisé est le modèle CARIMA (*Controlled AutoRegressive Integrated Moving Average*), de la forme (voire figure A.1) [64, 65] :

$$A(z^{-1})y(k) = B(z^{-1})u(k-d-1) + \frac{C(z^{-1})\xi(k)}{\Delta} \qquad \text{(A.1)}$$

où $A(z^{-1})$ et $B(z^{-1})$ sont des polynômes linéaires en de type z^{-1} avec les de degrés n_a et n_b respectivement, $y(k)$ est la sortie du système, $u(k)$ est la commande appliquée au système, k l'instant d'échantillonnage, le temps de retard du système est décrit par le paramètre d, $\xi(k)$ est un bruit blanc gaussien de moyenne nulle, $\Delta = 1 - z^{-1}$ est représenté l'opérateur différentiel d'ordre 1, et enfin $C(z^{-1})$ est un polynôme en l'opérateur retard, lié aux perturbations et par la suite, sans une connaissance supplémentaire sur la nature des perturbations, il sera choisi égal à 1 (sa valeur n'influe pas par ailleurs sur le comportement en suivi de trajectoire, il peut jouer un rôle en rejet de perturbation).

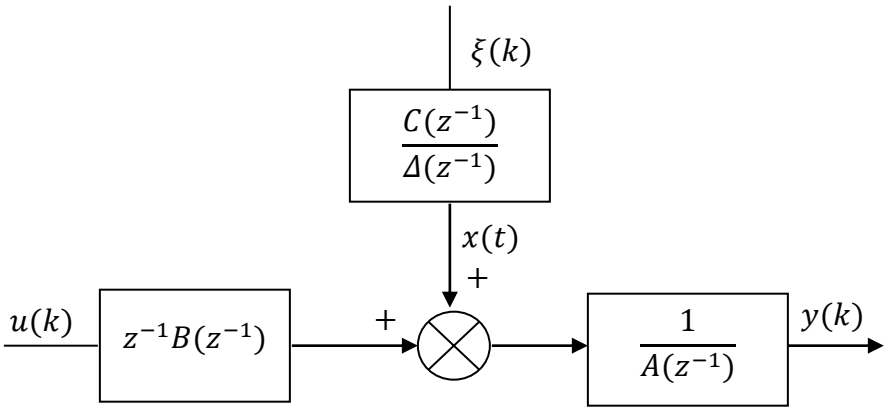

Figure A-5.8 : Modèle CARIMA

Les polynômes $A(z^{-1})$ et $B(z^{-1})$ sont donné par :

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{na} z^{-na} \qquad \text{(A.2)}$$
$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + a_{nb} z^{-nb}$$

Dans ce qui suit on considère $C(z^{-1}) = 1$, et on va utiliser l'équation de Diophantien pour simplifier l'obtention d'expression du prédicteur à j pas. Donc l'équation de Diophantien est donné par :

$$I = E_j(z^{-1})\Delta A(z^{-1}) + z^{-1}F_j(z^{-1}) \tag{A.3}$$

où les polynômes E_j et F_j sont donnés par :

$$\begin{aligned} E_j(\mathrm{z}^{-1}) &= 1 + e_{j,1}\mathrm{z}^{-1} + \cdots + e_{j,j-1}\mathrm{z}^{-(j-1)} \\ F_j(\mathrm{z}^{-1}) &= f_{j,0} + f_{j,1}\mathrm{z}^{-1} + \cdots + f_{j,n_a}\mathrm{z}^{-n_a} \end{aligned} \tag{A.4}$$

Le prédicteur optimal est enfin défini en considérant que la meilleure prédiction du bruit dans le futur est sa moyenne (supposée nulle ici), soit :

$$y(k+j|k) = F_j y(k) + G_j.\Delta u(k+j-d-1) \tag{A.5}$$

avec $G_j = BE_j$.

Une manière simple et effective pour déterminer les polynômes E_j et F_j est d'utiliser l'itération de l'équation diophantienne (une méthode récursive) de telle sorte que les polynômes E_{j+1}et F_{j+1} seront obtenus à partir de E_j et F_j. Donc, la méthode récursive est basée sur la connaissance des valeurs précédentes de $E_j(\mathrm{z}^{-1})$ et $F_j(\mathrm{z}^{-1})$ où les valeurs des coefficients de polynôme $E_{j+1}(\mathrm{z}^{-1})$ est obtenue par :

$$E_{j+1}(\mathrm{z}^{-1}) = E_j(\mathrm{z}^{-1}) + \mathrm{z}^{-j}e_{j+1,j} \tag{A.6}$$

avec $e_{j+1,j} = f_{j,0}$. Les coefficients de polynôme $F_{j+1}(\mathrm{z}^{-1})$ est atteinte par :

$$f_{j+1,i} = f_{j,i+1} - f_{j,0}\tilde{a}_{i+1}, i = 0, \dots, n_a - 1 \tag{A.7}$$

où $f_{j,n_a} = 0$. De la même façon, les coefficients de $G_j(\mathrm{z}^{-1})$ est déterminer par une manière récursive où les premier j confessions de $G_{j+1}(\mathrm{z}^{-1})$ sont exactement le même que les premier j confessions de $G_j(\mathrm{z}^{-1})$, main le reste des confessions est obtenu par :

$$g_{j+1,j+i} = g_{j,j+i} - f_{j,0}b_i, i = 0, \dots, n_b \tag{A.8}$$

A.1.2 La loi de commande GPC

Supposons qu'un ensemble de consigne $w(k+j)$ soit donné forment ainsi vecteur de référence. Pour simplification, on prendra la valeur de vecteur de référence comme une constant : $w(k+j) = w,\ \forall j$. Donc, à un instant d'échantillonnage quelconque, seul le signal $y(k)$ est connu. Les signaux de commande $u(k)$ ne sont pas connus et les sorties prédites du système sont données par la formule :

$$\boldsymbol{y}(k) = \boldsymbol{G}\Delta\boldsymbol{u}(k) + \boldsymbol{F}(\mathrm{z}^{-1})y(k) + \boldsymbol{L}(\mathrm{z}^{-1}) \tag{A.9}$$

où $\boldsymbol{y}$, $\boldsymbol{u}$ and $\boldsymbol{F}$ sont respectivement des vecteurs de dimension : $1 \times N_p$, $1 \times (N_u - 1)$ et $1 \times N_p$ tels que :

$$(k) == \begin{bmatrix} y(k+d+1) \\ y(k+d+2) \\ \vdots \\ y(k+N_p) \end{bmatrix} \quad \Delta\boldsymbol{u}(k) = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta \mathrm{u}(k+N_u-1) \end{bmatrix} \tag{A.10}$$

$$\boldsymbol{F}(\mathrm{z}^{-1}) = \begin{bmatrix} F_{d+1}(\mathrm{z}^{-1}) \\ F_{d+2}(\mathrm{z}^{-1}) \\ \vdots \\ F_{N_p}(\mathrm{z}^{-1}) \end{bmatrix}$$

La matrice $\boldsymbol{G}$, est une matrice triangulaire inférieure de dimension $N_p \times \left(N_p - N_u\right)$:

$$\boldsymbol{G} = \begin{bmatrix} g_{1,0} & 0 & \dots & 0 \\ g_{2,1} & g_{2,0} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ g_{N_p,N_p-1} & g_{N_p,N_p-2} & \cdots & g_{N_p,N_p-N_u} \end{bmatrix} \tag{A.11}$$

et $\boldsymbol{L}(\mathrm{z}^{-1})$ est donné par :

$$\boldsymbol{L}(\mathrm{z}^{-1}) = \begin{bmatrix} \left(G_{d+1}(\mathrm{z}^{-1}) - \tilde{G}_{d+1}(\mathrm{z}^{-1})\right)\mathrm{z}\Delta u(k-1) \\ \left(G_{d+1}(\mathrm{z}^{-1}) - \tilde{G}_{d+1}(\mathrm{z}^{-1})\right)\mathrm{z}^2\Delta u(k-1) \\ \vdots \\ \left(G_{N_p}(\mathrm{z}^{-1}) - \tilde{G}_{N_p}(\mathrm{z}^{-1})\right)\mathrm{z}^{N_p}\Delta u(k-1) \end{bmatrix} \tag{A.12}$$

Si on définit le vecteur consigne : $\boldsymbol{W}$, par $\boldsymbol{W} = [w(t), w(t+1), \dots, w(t,N)]$, et la coefficient de pondération sur la commande est fixé comme une constante $\vartheta(\mathrm{z}^{-1}) = \vartheta$, l'expression du critère de performance est donc peut être écrite sous forme vectorielle comme suit :

$$J(k) = (\boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{F}(\mathrm{z}^{-1})y(k) + \boldsymbol{L}(\mathrm{z}^{-1}) - \boldsymbol{W})^T(\boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{F}(\mathrm{z}^{-1})y(k) + \boldsymbol{L}(\mathrm{z}^{-1}) - \boldsymbol{W}) + \vartheta.\Delta\boldsymbol{u}(k)^T\Delta\boldsymbol{u}(k) \quad \text{(A.13)}$$

La minimisation de $J(k)$ ($\frac{\partial J(k)}{\partial u(k)} = 0$) donne :

$$\Delta\boldsymbol{u}(k) = (\boldsymbol{G}^T\boldsymbol{G} + \vartheta\boldsymbol{I})^{-1}\boldsymbol{G}^T(\boldsymbol{W} - \boldsymbol{F}(\mathrm{z}^{-1})y(k) - \boldsymbol{L}(\mathrm{z}^{-1})) \quad \text{(A.14)}$$

où $\boldsymbol{I}$ représente la matrice identité. Cet incrément contrôle est utilisée dans un contexte à horizon fuyant (*Receding Control Horizon*) c'est-à-dire que seulement le premier élément de $\Delta\boldsymbol{u}(k)$, noté par $\Delta u(k)$, est calculé pour donner la commande courante.
Le signal de commande envoyé au système est seulement le premier élément du vecteur $\boldsymbol{u}(k)$. Dans ce cas, l'incrément du signal de contrôle est donné par :

$$\Delta u(k) = \boldsymbol{\Gamma}(\boldsymbol{W} - \boldsymbol{F}(\mathrm{z}^{-1})y(k) - \boldsymbol{L}(\mathrm{z}^{-1})) \quad \text{(A.15)}$$

où $\boldsymbol{\Gamma}$ représente la premier ligne de la matrice $(\boldsymbol{G}^T\boldsymbol{G} + \vartheta\boldsymbol{I})^{-1}\boldsymbol{G}^T$, et la commande à appliquer au système à l'instant k est alors :

$$u(k) = u(k-1) + \Delta u(k) \quad \text{(A.16)}$$

Annexe B

B.1 La méthode de k-moyennes

La méthode de k-moyennes (*k-means*) [96] est probablement l'un des algorithmes de partitionnement les plus connus. Cette méthode est relativement simple et permet d'obtenir de bonnes partitionnement grâce à la minimisation d'une fonction de coût ce qui indique que cette méthode représente un problème d'optimisation combinatoire. Dans cette méthode, le nombre de groupes est fixé à k alors que le problème est de partager les points (vecteurs) des données en k groupes. La procédure de séparation est obtenue en minimisant une certaine fonction de coût.

On considère la distance d'un point à la moyenne des points de son groupe ; la fonction à minimiser est la somme des carrés de ces distances.

L'algorithme de k-moyennes peut être résumé comme suit :

1) Choisir le nombre de groupes k.
2) Initialiser les centres $\boldsymbol{c}_i, 1 \leq i \leq k$ aléatoirement.
3) Attribuer chaque point de données à un groupe en fonction de la distance la plus proche du centre.
4) Ajuster les centres des groupes de façon à minimiser la fonction de coût suivant :

$$f(\boldsymbol{c}) = \sum_{i=1}^{k} \sum_{x \in C_i} \|\boldsymbol{x} - \boldsymbol{c_i}\|^2$$

5) Retourner à 3 jusqu'à ce que le partitionnement soit converge.

Annexe C

C.1 L’indice de Dunn

Cette indice est proposée par Dunn [87, 88] est utilisée pour analyser la qualité de partitionnement. Ce critère est basé sur l’identification de groupes compacts et bien séparés. L’indice de Dunn est défini par le rapport entre la plus petite dis-similarité interclasse et la plus grande dis-similarité intra-classe. Notez que le terme interclasse signifie une relation entre deux individus de deux groupes différents, et le terme intra-classe signifie une relation entre deux individus de la même groupe. Cet indice est décrit par la formule suivante :

$$Dunn = \min_{1\le i\le n}\left\{\min_{\substack{1\le j\le n\\ i\neq j}}\left\{\frac{d(\boldsymbol{c}_i,\boldsymbol{c}_j)}{\max_{1\le k\le n}\tilde{d}(\boldsymbol{c}_k)}\right\}\right\} \tag{C.1}$$

L'objectif principal de cet critère est de maximiser la dis-similarité interclasse et de minimiser la dis-similarité intra-classe. Ainsi, l'objectif est donc de maximiser l'indice de Dunn.