



REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE
SCIENTIFIQUE
UNIVERSITE FRERES MENTOURI CONSTANTINE
FACULTE DES SCIENCES DE LA TECHNOLOGIE
DEPARTEMENT D'ELECTRONIQUE



NO d'ordre : 06/D3C/2022

Série : 01/Elec/2022

THESE

Présentée pour obtenir le diplôme de

DOCTORAT EN SCIENCES

Spécialité : ELECTRONIQUE

Option : ELECTRONIQUE

Par :

BOUMAZA Hamza

THEME

Approximation de la commande prédictive par un système d'inférence flou

Soutenu publiquement le 23/02/2022 devant le Jury :

Président	Z. Hammoudi	<i>Professeur,</i>	<i>Université Frères Mentouri Constantine</i>
Rapporteur	K. Belarbi	<i>Professeur</i>	<i>Ecole nationale polytechnique de Constantine</i>
Examineurs	M. Belhocine	<i>Directeur de recherche</i>	<i>Centre de développement des technologies avancées Alger</i>
	B. Boukhezar	<i>Professeur</i>	<i>Université Frères Mentouri Constantine</i>
	K. Lamamra	<i>Professeur</i>	<i>Université Larbi ben M'hidi Oum el-Bouaghi</i>
	S. Ziani	<i>Maitre de conférences A</i>	<i>Université Frères Mentouri Constantine</i>

Résumé de la Thèse

D

ans cette thèse nous avons exploité la propriété d'approximation universelle des systèmes flous de Takagi-Sugeno pour construire une approximation floue de la solution optimale de contrôle prédictif de modèle linéaire et non linéaire (MPC). Les systèmes non linéaires considérés sont affines dans la loi de commande. L'approximateur flou introduit présente quelques propriétés généralement non montrées par les approximateurs précédents. En particulier, il est constitué d'un ensemble de lois de commande à retour d'état qui sont fusionnées pour obtenir la loi de commande non linéaire finale. La loi de commande construite est très similaire à la solution explicite du MPC linéaire. La stabilité est analysée a posteriori, pour le MPC linéaire et non linéaire, sur la base d'un nouveau test théorique développé pour les systèmes flous de type Takagi Sugeno. Les exemples traités dans la partie simulation ont montré la promptitude et les bons résultats qui prouvent l'efficacité de la stratégie de contrôle développée.

Mots clés : Commande prédictive à base de modèle, Systèmes flous Takagi-Sugeno, Solution approximante, Condition de stabilité.

Abstract

In this thesis, we exploited the universal approximation property of Takagi-Sugeno fuzzy systems to construct a fuzzy approximation of the optimal linear and nonlinear model (MPC) predictive control solution. The nonlinear systems considered are affine in the control law. The fuzzy approximator presents some properties generally not shown by the preceding ones. In particular, it consists of a set of state feedback control laws which are merged to obtain the final nonlinear control law. The constructed control law is very similar to the explicit solution of the linear MPC. The stability is analyzed a posteriori, for the linear and non-linear MPC, on the basis of a new theoretical test developed for fuzzy systems of the Takagi Sugeno type. The examples treated in the simulation have shown the promptness and the good results which prove the effectiveness of the control strategy developed.

Keywords: Model predictive control, Takagi-Sugeno Fuzzy Systems, Approximate solution, Stability Condition.

ملخص

ف ي هذه الأطروحة ، قمنا باستغلال خاصية التقريب العام لأنظمة طاكاجي سيجينو الضبابية لبناء تقريب للحل الأمثل للتحكم التنبئي باستخدام نموذج و هذا في حالة النموذجين الخطي وغير الخطي على السواء. تعتبر الأنظمة غير الخطية التي تم النظر فيها توافقية مع قانون التحكم. يعرض التقريب الضبابي المقترح بعض الخصائص التي لا تظهر بشكل عام في الاساليب التقريبية الاخرى. على وجه الخصوص ، يتكون من مجموعة من قوانين مراقبة محصل عليها من التغذية المرتدة و التي يتم دمجها للحصول على قانون التحكم غير الخطي النهائي. قانون التحكم المُنشأ مشابه جدًا للحل الصريح للتحكم التنبئي الخطي. يتم تحليل استقرار المراقب المحصل عليه بشكل لاحق، من أجل التحكم التنبئي الخطي وغير الخطي، على أساس اختبار نظري جديد تم تطويره للأنظمة الضبابية من نوع طاكاجي سيجينو . أظهرت الأمثلة التي تم تناولها في المحاكاة السرعة والنتائج الجيدة التي تثبت فعالية استراتيجية التحكم المطورة.

الكلمات الرئيسية: نموذج التحكم التنبئي، أنظمة طاكاجي سيجينو الضبابية، الحل التقريبي، حالة الاستقرار.

Dédicaces

Je dédie cette Thèse

À mon défunt père, ma mère, ma sœur

et à ma future nièce.

Hamza

Remerciements

Mes remerciements vont en premier lieu à ALLAH Tout Puissant qui a illuminé mon chemin de la lueur du savoir et de la science et pour la volonté, la santé et la patience qu'il ma prodiguées durant toutes ces années d'études.

Je tiens aussi à exprimer ma reconnaissance et ma profonde gratitude à Monsieur BELARBI Khaled, professeur à l'Ecole nationale polytechnique de Constantine, pour avoir assuré l'encadrement de ce travail. Son aide, sa grande disponibilité ont joué un rôle essentiel dans l'aboutissement de ce travail.

J'exprime mes vifs remerciements à Monsieur HAMMOUDI Zoheir, professeur à l'université frères Mentouri Constantine, qui m'a fait l'honneur de présider ce jury.

Mes sincères remerciements vont également à Messieurs BOUKHEZZAR Boubekour Professeur à l'université frères Mentouri Constantine, ZIANI Salim, Maitres de Conférences à l'université frères Mentouri Constantine, BELHOCINE Mahmoud Directeur de recherche au Centre de Développement des Technologies Avancées et LAMAMRA Kheireddine Professeur L'université de Oum Elbouaghi qui ont bien voulu examiner cette thèse.

Enfin, mes remerciements vont aussi à mes chers parents pour leur patience, leurs encouragements continus et leur soutien inconditionnel, pour ma sœur. Qu'il trouve ici toute ma gratitude et mon amour.

Que toutes les personnes que j'ai involontairement oubliées, trouvent ici, en cette heureuse et solennelle circonstance, l'expression de notre profonde gratitude et de notre indéfectible dévouement.

Table de matières

Chapitre 1 Introduction générale	1
Introduction.....	2
Résumé	4
Chapitre 2 Commande prédictive à base de modèle.....	5
2.1 Introduction.....	6
2.2 Commande Prédictive à base de modèle	6
2.2.1- Principe général	6
2.2.2- La solution du problème MPC	10
2.2.2.a- Cas linéaire	10
2.2.2.b- Cas non linéaire	13
2.3 Commande prédictive explicite	18
2.4 Commande prédictive approximante	21
2.4.1- Approches basée sur l'apprentissage	23
2.4.2- Formulation basée sur les PWA	25
2.4.3. Autre approche	27
2.5 Commande prédictive rapide	28
2.6 Outils solution MPC	29
2.6.1- Logiciel	29
2.6.2- Solution MPC par PSO	30

2.7 Conclusion	33
Chapitre 3 Commande prédictive approximante par apprentissage flou	34
3.1 Introduction	35
3.2 Apprentissage supervisé	35
3.3 Système flous de type Takagi Sugeno	37
3.3.1- Structure générale d'un système flou Takagi Sugeno discret.....	37
3.3.2- Structure d'un modèle flous de type T-S	38
3.3.3- Secteur de non linéarité (Sector nonlinearity)	38
3.3.4- Propriété d'approximation universelle pour les systèmes flous de type TS	39
3.4 Commande prédictive approximante par apprentissage flou	41
3.4.1- Principe général	41
3.4.2- Stratégie de la commande prédictive approximante par apprentissage flou	42
3.4.2.a. Solution MPC	42
3.4.2.b. Synthèse du contrôleur flou	43
3.5 Conclusion	46
Chapitre 4 Stabilité de la commande prédictive par apprentissage flou.....	47
4.1 Introduction	48
4.2 Stabilité au sens de Lyapunov pour un system flou de type TS	49
4.2.1- Stabilité Quadratique	49
4.2.2- Stabilité Non Quadratique	50
4.3 Conditions de stabilité pour les systèmes flous de type TS	51
4.3.1- matrices d'intervalle	52

Bibliographie

4.3.2-Condition suffisante pour la stabilité globale d'un système à matrices d'intervalle	52
4.3.3- Stabilité locale d'un système flou de type TS	53
4.4 Stabilité du contrôleur FAMPC en boucle fermée	55
4.4.1- Stabilité cas linéaire	55
4.4.2- Stabilité cas non linéaire	56
4.4.3- Algorithme de test de stabilité du Contrôleur en boucle fermée	58
4.5 Conclusion	59
Chapitre 5 Simulations	60
5.1 Introduction	61
5.2 Exemple 5.1.....	62
5.3 Exemple 5.2.....	68
5.4 Exemple 5.3.....	72
5.5 Exemple 5.4.....	75
5.6 Conclusion	78
Conclusion Générale	80
Bibliographie.....	82
Annexe A	88
Annexe B	93

Liste des figures

Chapitre 2

Figure 2.1	Principe de la commande prédictive	8
Figure 2.2	Stratégie de commande prédictive	8
Figure 2.3	Relation entre les nombres d'Etats, Régions et Contraintes.	22
Figure 2.4	Illustration du mouvement des particules d'une itération de PSO	31

Chapitre 3

Figure 3.1	Schéma de principe du processus de l'apprentissage automatique	36
Figure 3.2	Fonctions d'appartenances système flou de type TS	38
Figure 3.3	Secteur de non linéarité : a. secteur globale, b. secteur local	40

Chapitre 4

Figure 4.1	Illustration de la région S_0 : fonctions d'appartenances système flou de type TS	54
------------	---	----

Chapitre 5

Figure 5.1	Exemple 5.1 Surface de commande explicite MPC	64
Figure 5.2	Exemple 5.1 Surface commande prédictive approximante FAMPC	65
Figure 5.3	Exemple 5.1 (a) les états (b) la commande, MPC et FAMPC	66
Figure 5.4	Exemple 5.1 test de robustesse FAMPC	67
Figure 5.5	Exemple 5.2 Surface de commande explicite MPC	69
Figure 5.6	Exemple 5.2 Surface commande prédictive approximante FAMPC	70
Figure 5.7	Exemple 5.2 (a) les états, (b) et (c) les commandes, MPC et FAMPC	71
Figure 5.8	Exemple 5.3 (a) les états, (b) les commandes, MPC et FAMPC	73
Figure 5.9	Exemple 5.3 test de robustesse FAMPC (a) , (b): , Nominale (ligne solide),Perturbé (ligne pointillée)	74
Figure 5.10	Exemple 5.4 (a) position angulaire et (b) la force, MPC et FAMPC	76
Figure 5.11	Exemple 5.4 Position: angulaire, test de robustesse FAMPC (a) , (b): , Nominale (ligne solide),Perturbé (ligne pointillée)	77

Annexe A

Figure A.1	Schéma de principe d'un système flou de Takagi Sugeno	92
------------	---	----

Annexe B

Figure B.1	Ajout du lien de YALMIP à mpt_init	94
Figure B.2	Etats de la MPC par MPT de l'exemple B.1.	97
Figure B.3	Signal de commande de la MPC par MPT de l'exemple B.1	97
Figure B.4	Loi explicite de la commande prédictive de l'exemple B.1.	99
Figure B.5	Régions de la solution MPC explicite pour l'exemple B.1, projection sur l'espace d'état.	100

Liste des tableaux

Chapitre 5

Tableau 5.1	Exemple 5.1: Les gains du contrôleur	63
Tableau 5.2	Exemple 5.2: Les gains du contrôleur	67
Tableau 5.3	Exemple 5.4: Les gains du contrôleur	73
Tableau 5.4	Comparaison de performance	79

Annexe A

Tableau A.1	Formes et Modèles des fonctions d'appartenance les plus utilisées	89
-------------	---	----

LISTE DES ACRONYMES

APC	Approximate Predictive Control
CPE	Commande Prédicte Explicite
DMC	Dynamic Matrix Control
MPC	Model Predictive Control
MBPC	Model Based Predictive Control
FAMPC	Fuzzy Approximate Model Predictive Control
NMPC	Nonlinear Model Based Predictive Control
QP	Quadratic Program
PQM	Programmation Quadratique Multiparamétrique
mp-PQ	multiparametric Quadratic Programming
PSO	Particle Swarm Optimization
PWA	Piecewise Affine
SDRE	State Dependent Riccati Equation
TS	Takagi Sugeno
TSF	Takagi Sugeno Fuzzy
ISS	Input to State Stability

Liste de publication et de communication

Publications

- 1- Boumaza, H., Belarbi, K. Optimal model predictive control solution approximation using Takagi Sugeno for linear and a class of nonlinear systems. *Int. J. Dynam. Control* (2021). <https://doi.org/10.1007/s40435-021-00875-4>

Communications

- 1- H. BOUMAZA and K.BELARBI, " A fuzzy approximator for model based predictive control", In : *14th International Conference on Sciences and Techniques of Automatic Control & Computer Engineering-STA'2013*. IEEE, Tunisia, p. 20-24, December 20-22,2013. doi: 10.1109/STA.2013.6783099.
- 2- K. Belarbi, H. Boumaza and B. Boutamina, "Nonlinear model predictive control based on state dependent Riccati equation," *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering STA' 2014*, IEEE, Tunisia, pp. 65-69, , December 21-23,2014.doi: 10.1109/STA.2014.7086742.
- 3- HAMZA, BOUMAZA; HAMERLAIN Mustapha and KHALED, BELARBI. "An online solution for model predictive control using particle swarm optimization for autonomous vehicle driving", In: *The 3rd International Conference on Electromechanical Engineering, ICEE' 2018*, Skikda, Algeria, November 21-22, 2018.

Chapitre 1

Introduction Générale

1.1- Introduction

La commande prédictive à base de modèle (MPC), avec ses différentes formulations, est une stratégie de contrôle qui a été largement acceptée, tant dans l'industrie que dans la littérature (Mayne, 2014). Ceci est principalement dû à sa simplicité de conception et à sa capacité à gérer des contraintes dures (sévères) qui permettent de résoudre le problème de l'enroulement de manière naturelle. Le signal de contrôle est obtenu en résolvant en ligne, à chaque période d'échantillonnage, un problème d'optimisation statique qui est exprimé sous la forme d'un programme quadratique dans le cas linéaire et d'un programme contraint non linéaire dans le cas non linéaire. Le problème quadratique, QP, peut être résolu efficacement en utilisant la méthode des ensembles actifs, tandis que le MPC non linéaire peut être résolu avec le MPC séquentiel ou le procédé dit de prise de vue multiple (Chen & Allgower, 1998). Cependant, l'application de MPC était limitée par la charge de calcul en ligne associée à ces approches. Des méthodes de solutions rapides ont ensuite été recherchées afin d'étendre l'applicabilité du MPC. Deux chemins ont ensuite été explorés. Une recherche de solutions analytiques a conduit à ce que l'on appelle le MPC explicite (Bemporad, Morari, Dua, & Pistikopoulos, 2002) basé sur l'analyse multiparamétrique de la solution QP, mp-QP. La commande obtenue est une fonction linéaire par morceaux des états. L'espace d'états est décomposé en régions polyédriques dotées chacune d'une loi de contrôle affine. Cependant, il est rapidement apparu que le nombre de régions augmentait de manière exponentielle avec la dimension du problème (Alessio & Bemporad, 2009).

Des approches approximantes de la solution par morceaux ont ensuite été proposées pour alléger la charge de calcul. Fondamentalement, ces méthodes résolvent des problèmes d'ajustement : étant donné la loi de commande optimale calculée hors ligne, elles calculent une loi de commande approximante qui tente de minimiser l'erreur entre les lois approximée et optimale ((Johansen & Grancharova, 2003), (Bemporad & Filippi, 2003), (Johansen, 2004), (Canale, Fagiano, & Milanese, 2009), (Canale, Fagiano, & Milanese, 2010), (Kvasnica, Lofberg, & Fikar, 2011), (Stogiannos, Alexandridis, & Sarimveis, 2018), (Rubagotti, Barcelli, & Bemporad, 2014), (Bakaráč, Holaza, Kalúz, Klaučo, Löfberg, & Kvasnica, 2018), (Bemporad, Oliveri, Poggi, & Storace, 2011), (Hovd, Scibilia, Maciejowski, & Oлару, 2009) et (Scibilia, Hovd, & Oлару, 2012)). Les techniques d'approximation de fonctions hors ligne, telles que les réseaux de neurones artificiels dans (Parisini & Zoppoli, 1995) et l'identification d'ensembles dans (Canale, Fagiano, & Milanese, 2009) et (Canale,

Fagiano, & Milanese, 2010), sont appréciées pour leur concept simple même si le principal inconvénient ici est la difficulté à prouver les propriétés de stabilisation du contrôleur synthétisé. D'autre part, la formulation PWA de l'APC est populaire pour ses propriétés de stabilisation faciles à prouver par rapport à d'autres approches mais toujours limitée par la malédiction de la dimensionnalité de la partition de l'ensemble d'états. Une telle limitation a été rencontrée dans (Bemporad, Oliveri, Poggi, & Storace, 2011) lors de l'utilisation d'un partitionnement simplicial de l'ensemble d'états dans la région d'approximation.

La stabilité de la MPC approximante peut être garantie à priori comme dans (Canale, Fagiano, & Milanese, 2010), (Kvasnica, Lofberg, & Fikar, 2011), (Stogiannos, Alexandridis, & Sarimveis, 2018), (Rubagotti, Barcelli, & Bemporad, 2014) et (Bakaráč, Holaza, Kalúz, Klaučo, Löfberg, & Kvasnica, 2018) ou vérifiée à posteriori, comme par exemple dans (Bemporad, Oliveri, Poggi, & Storace, 2011) et (Hovd, Scibilia, Maciejowski, & Oлару, 2009). Les deux approches sont en quelque sorte équivalente puisqu'il s'agit de procédures itératives. En effet, dans les approches à priori, la stabilité est garantie si l'erreur d'approximation est bornée mais il n'y a aucune garantie que l'approximateur conçu satisfera cette condition. En effet, bien que l'erreur dépende de certains paramètres tels que le nombre de points d'ajustement, il n'y a pas de relation explicite qui rendrait la procédure un coup (one-shot). C'est également le cas pour les approches à posteriori. Notez que la technique d'approximation a été suggérée de manière précoce en utilisant une approximation de réseau neuronal (Parisini & Zoppoli, 1995). Dans la solution explicite et ses différentes approximations, l'élégante procédure d'optimisation de prédiction qui rend MPC si attrayant n'est plus apparente.

D'autre part, des méthodes numériques rapides exploitant la structure du problème MPC ont été proposées. Ce sont principalement des algorithmes classiques d'optimisation sous contrainte, tels que la méthode du point intérieur ou l'algorithme de gradient qui exploite la structure du problème de MPC ((Wang & Boyd, 2010), (Kögel & Findeisen, 2011), (Jerez, Goulart, Richter, Constantinides, Kerrigan, & Morari, 2013) et (Rubagotti, Patrinos, Guiggiani, & Bemporad, 2014)). Le principal inconvénient de l'optimisation en ligne, du moins sur le plan conceptuel, est le risque d'infaisabilité du problème d'optimisation.

Dans ce travail, nous introduisons un système flou de type Takagi Sugeno, TSF, pour approximer la solution optimale de la commande prédictive explicite linéaire et de la solution

numérique du MPC non linéaire. La propriété d'approximation universelle du système flou TS ((Yin, 1998), (Tikk, Koczy, & Gedeon, 2003) et (Mendel, 2018)) est exploitée pour construire l'approximateur flou du MPC, FAMPC, qui peut être appliqué à la fois aux MPC linéaires et non linéaires. Dans ce dernier cas, le modèle du système est non linéaire affine à la commande. Le calcul de la solution optimale est effectué hors ligne, l'approximation obtenue conduit à une simple rétroaction d'état des lois de commande locales. La stabilité de l'approximateur obtenu est analysée à posteriori à l'aide des résultats disponibles sur la stabilité des systèmes de TSF (Belarbi, 2019). Une brève analyse préliminaire, par simulation, de la FAMPC a été présentée dans (Boumaza & Belarbi, 2013).

1.2- Organisation de la thèse

Chapitre 2 Dans ce chapitre, les bases mathématiques nécessaires pour ce travail sont présentées. Nous introduisons les concepts de base de la commande prédictive ainsi un état de l'art sur la commande prédictive approximante est présenté.

Chapitre 3 Ce chapitre nous introduisons, en bref, des généralités de la logique floue pour les systèmes flous de type Takagi Sugeno. Ainsi présente le concept de base du problème de l'approximation de la loi de commande prédictive optimale par un système flou Takagi Sugeno. Étant donné la structure du système flou, sa sortie doit être aussi proche que possible de la loi optimale sans violation des contraintes.

L'approximation est effectuée à l'aide d'un ensemble de solutions optimales du MPC, calculées hors ligne. Ces solutions optimales sont déterminées pour un nombre donné de conditions initiales uniformément réparties sur un sous-ensemble compact inclus dans l'ensemble des solutions possibles et incluant l'origine.

Chapitre 4 L'analyse de la stabilité de la FAMPC se résume à l'étude de stabilité du système flou TS qui en résulte de la résolution du problème d'approximation ci-dessus.

Dans ce travail, l'approche de (Belarbi, 2019), est adoptée pour analyser la stabilité du système FAMPC à *posteriori*. Cette alternative donne une condition nécessaire et suffisante pour la stabilité locale de systèmes TS flous discrets. Ainsi, le chapitre présente les notions de base de la stabilité au sens de Lyapunov en particulier pour les systèmes flous de type TS.

Ensuite, l'approche proposée pour analyser la stabilité de la boucle fermée est exposée, d'abord pour le cas linéaire, puis pour une classe de systèmes non linéaires.

Chapitre 5 Dans ce dernier chapitre, l'application de l'approximation proposée est décrite. Quatre exemples sont présentés, trois problèmes linéaires tirés de la littérature et un problème non linéaire.

Chapitre 2

Commande prédictive à base de modèle

2.1 Introduction

Introduite par Richalet en 1978 dans sa forme actuelle la commande prédictive approximante a gagnée beaucoup d'intérêt surtout dans le domaine des procédés chimique en industrie. Malgré son succès la MPC a été limitée par le temps d'exécution. De ce fait la méthode trouve son application beaucoup plus pour des systèmes lents à grande période d'échantillonnage où l'application des méthodes numériques ne pose pas de problème. Afin d'élargir l'utilisation de la MPC pour les systèmes rapides échantillonnés à haute fréquence, tels que les robots, plusieurs approches ont été proposées. Ainsi, au cours de la dernière décennie la technique dite commande prédictive explicite (*explicit MPC*) a été introduite ((Bemporad, Morari, Dua, & Pistikopoulos, 2002), (Grancharova & Johansen, 2005)). Cette dernière repose sur le calcul de la loi de commande de la MPC hors ligne et la convertit en loi linéaire par morceau (PWA), ce qui réduit le problème en ligne à une évaluation d'une table de consultation (lookup table) de gains d'une commande linéaire. Bien que la MPC explicite eu un certain succès dans quelques applications, elle trouve toujours des limitations concernant le nombre d'ensemble PWA (Grancharova & Johansen, 2005).

D'autre part, la technique dite commande prédictive approximante introduite par (Parisini & Zoppoli, 1995) trouve un grand intérêt récemment avec le développement majeur des solutions en matière de solution numérique surtout sur le plan hardware. De la sorte, étant donné la loi de commande optimale calculée hors ligne, elles calculent une loi de commande approximante qui tente de minimiser l'erreur entre les lois approximée et optimale.

Dans cette section, nous introduisons le concept de base de la commande prédictive. Nous présentons ensuite le concept de la commande prédictive explicite. Finalement, un état de l'art pour le cas de la commande prédictive approximante est établi.

2.2 Commande Prédictive à base de modèle

2.2.1- Principe général

L'algorithme de la commande prédictive est formulé par un problème d'optimisation sujet à un modèle mathématique de prédiction et des contraintes.

La Figure (2.1) illustre le concept de la commande prédictive, à chaque période d'échantillonnage t du contrôleur, la prédiction du comportement du système est calculée grâce à un modèle interne, les mesures acquises à partir des capteurs et la future loi de commande.

La prédiction sur les variables commandées est effectuée sur une fenêtre temporelle de N échantillons (horizon de prédiction). La loi de commande optimale à appliquer est calculée jusqu'à un horizon temporel N_u (horizon de commande), en minimisant un critère de performances.

A la période d'échantillonnage suivante $t+1$, seul le premier élément de la loi de commande calculée est appliqué sur le système. Cette procédure est ensuite répétée : c'est le principe de l'horizon fuyant. Ainsi à chaque période d'échantillonnage, un problème d'optimisation doit être résolu en temps réel. Dans le cas linéaire ce problème est exprimé sous forme d'un programme quadratique qui admet donc un seul minimum global tandis que dans le cas non linéaire, c'est un programme non linéaire avec contraintes non convexe admettant plusieurs minima locaux.

Les éléments de base de la commande prédictive (voir Figure (2.2)) sont :

1. Un modèle du système pour réaliser les prédictions avec parfois des contraintes sur les variables de commandes et/ou les sorties à commander.
2. Une fonction coût à minimiser.
3. Un algorithme d'optimisation (pour calculer la commande future).

Différentes options peuvent être considérées pour chaque élément, ce qui donne une variété d'algorithmes de commande prédictive.

Remarque 2.1 :

- Pour différencier on utilise la lettre t pour désigner l'évolution en temps discret et la lettre k pour les prédictions.
- Dans la figure 2.1 la notation k/t est adoptée qui signifie la prédiction k sachant la mesure à l'instant t .

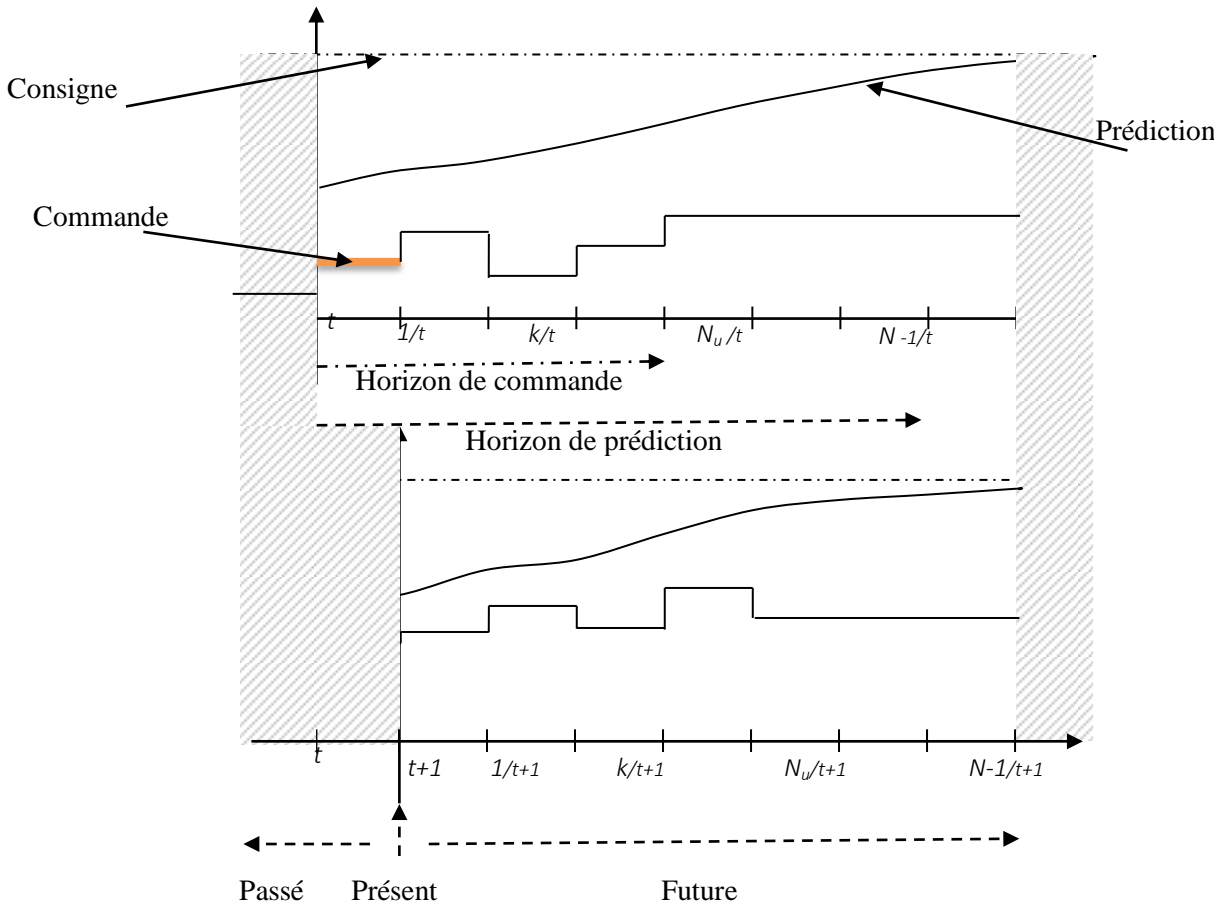


Figure (2.1) Principe de la commande prédictive.

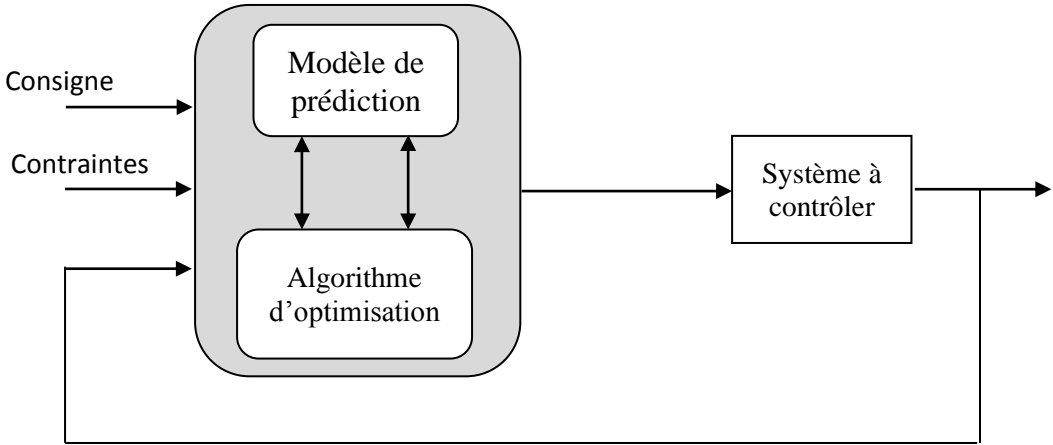


Figure (2.2) Stratégie de commande prédictive.

On considère les deux ensembles $\mathbb{X} \subseteq \mathbb{R}^n$ et $\mathbb{U} \subseteq \mathbb{R}^m$ qui représentent les contraintes sur l'état et l'entrée, respectivement, contenant l'origine.

On définit les fonctions suivantes :

La fonction f décrit le modèle de prédiction :

$$\begin{aligned} f: \mathbb{R}^n \times \mathbb{R}^m &\rightarrow \mathbb{R}^n \\ f(x_t, u_t) &= x_{t+1}, t \geq 0 \\ f(0,0) &= 0 \end{aligned} \quad , \quad (2.1)$$

La fonction F_c le coût sur l'état final :

$$\begin{aligned} F_c: \mathbb{R}^n &\rightarrow \mathbb{R}_+ \\ F_c(0) &= 0 \end{aligned} \quad , \quad (2.2)$$

et la fonction coût ℓ :

$$\begin{aligned} \ell: \mathbb{R}^n \times \mathbb{R}^m &\rightarrow \mathbb{R}_+ \\ \ell(0,0) &= 0 \end{aligned} \quad , \quad (2.3)$$

On définit le problème d'optimisation suivant :

$$\min_{U_t} \left\{ J(x_t, U_t) = F_c(x_{t+N/t}) + \sum_{k=0}^{N-1} \ell(x_{t+k/t}, u_{t+k/t}) \right\} \quad (2.4. a)$$

$$s. \text{ à } x_{t+k+1/t} = f(x_{t+k/t}, u_{t+k/t}), k = 0, \dots, N-1 \quad (2.4. b)$$

$$x_{t+k/t} \in \mathbb{X}, k = 0, \dots, N \quad (2.4. c)$$

$$u_{t+k/t} \in \mathbb{U}, k = 0, \dots, N-1 \quad (2.4. d)$$

Où le vecteur d'état $x_{0/t}$ obtenu à l'instant t par mesure ou observation, le vecteur $U_t = (u_{0/t}, \dots, u_{N-1/t})$ est la séquence d'entrée optimale prédit. L'élément $u_{0/t}$ est la commande à appliquer pour l'instant $t+1$.

2.2.2- La solution du problème MPC

2.2-2.a Cas linéaire

Dans le cas linéaire avec contraintes, on considère le problème d'optimisation (2.4) dont le modèle est donné sous forme d'équations d'état :

$$\begin{cases} x_{t+1} = A x_t + B u_t \\ y_t = C x_t \end{cases}, \quad (2.5)$$

Où $x_t \in \mathbb{X}$ est le vecteur d'états, $u_t \in \mathbb{U}$ le vecteur d'entrées, $y_t \in \mathbb{Y} \subseteq \mathbb{R}$ le vecteur de sorties du système, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, (A,B) est une paire gouvernable et (C,A) est une paire observable.

Le problème d'optimisation (2.4) se transforme alors :

$$\begin{aligned} \min_{U_t} & \left\{ \begin{array}{l} J(x_t, U_t) = x_{t+N/t}^T P x_{t+N/t} + \\ \sum_{k=0}^{N-1} (x_{t+k/t}^T Q x_{t+k/t} + u_{t+k/t}^T R u_{t+k/t}) \end{array} \right\} \\ \text{s. à } & x_{t+k+1/t} = A x_{t+k/t} + B u_{t+k/t}, k = 0, \dots, N-1, \\ & x_{t+k/t} \in \mathbb{X}, k = 0, \dots, N \\ & u_{t+k/t} \in \mathbb{U}, k = 0, \dots, N-1 \end{aligned} \quad (2.6)$$

Avec les matrices symétriques $Q \geq 0$ (semi définie positive) et $R > 0$ (définie positive), sont la matrice de pondération sur l'état et la matrice de pondération d'entrée, respectivement, et P est une matrice de pondération sur l'état final. Pour garantir la stabilité du système commandé la matrice P est choisie comme la solution de l'équation algébrique de Riccati :

$$PA + A^T P - PBR^{-1}B^T P + Q = 0, \quad (2.7)$$

En effet, il a été démontré (Rawlings & Muske, 1993) et (Chmielewski & Manousiouthakis, 1996) qu'il est possible de transformer le problème LQR infini en un problème à horizon fini si le coût est considéré comme la somme de deux composantes :

$$\begin{aligned} \sum_{k=0}^{\infty} (x_{t+k/t}^T Q x_{t+k/t} + u_{t+k/t}^T R u_{t+k/t}) &= \sum_{k=0}^{N-1} (x_{t+k/t}^T Q x_{t+k/t} + u_{t+k/t}^T R u_{t+k/t}) + \\ &\quad \sum_N^{\infty} (x_{t+k/t}^T Q x_{t+k/t} + u_{t+k/t}^T R u_{t+k/t}), \end{aligned} \quad (2.8)$$

Où $N < \infty$ est l'horizon de prédiction choisi. Il est à noter qu'après un certain temps les contraintes sont satisfaites de façon naturelle (dans l'horizon de prédiction), les entrées considérées à l'optimisation sont uniquement celles de la première composante du coût, ce qui nous mène à introduire la notion du coût final.

$$\sum_{k=0}^{\infty} \left(x_{t+k/t}^T Q x_{t+k/t} + u_{t+k/t}^T R u_{t+k/t} \right) = \sum_{k=0}^{N-1} \left(x_{t+k/t}^T Q x_{t+k/t} + u_{t+k/t}^T R u_{t+k/t} \right) + x_{t+N/t}^T P x_{t+N/t}, \quad (2.9)$$

Où $x_{t+N/t}^T P x_{t+N/t}$ est la fonction coût final.

Calcul de la solution MPC (formulation de base (Maciejowski, 2002))

On suppose que le vecteur d'état est mesurable, d'où $x(t+0 | t) = x(t)$, ainsi on considère le cas où il n'y a aucune perturbations. On développe les N prédictions suivant le modèle (2.5) :

$$\begin{aligned} x_{t+1/t} &= A x_t + B u_{t/t} \\ x_{t+2/t} &= A x_{t+1/t} + B u_{t+1/t} \\ &= A^2 x_t + AB u_{t/t} + B u_{t+1/t} \\ &\quad \vdots \\ x_{t+N/t} &= A x_{t+N-1/t} + B u_{t+N-1/t} \\ &= A^N x_t + A^{N-1} B u_{t/t} + \dots + B u_{t+N-1/t} \end{aligned}$$

Ou d'une manière générale :

$$x_{t+k/t} = A^k x_t + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j/t} \quad (2.10)$$

Sachant que $\Delta u_{t+k/t} = u_{t+k/t} - u_{t+k-1/t}$, et à l'instant t on a $u(t-1)$, on aura :

$$\begin{aligned} x_{t+1/t} &= A x_t + B \left(\Delta u_{t/t} + u_{t-1} \right) \\ x_{t+2/t} &= A^2 x_t + (AB + B)u_{t-1} + (AB + B)\Delta u_{t/t} + B\Delta u_{t+1/t} \\ &\quad \vdots \\ x_{t+N_u/t} &= A^{N_u} x_t + (A^{N_u-1}B + \dots + B)u_{t-1} + (A^{N_u-1}B + \dots + B)\Delta u_{t/t} + \dots \\ &\quad + B \Delta u_{t+N_u-1/t} \end{aligned}$$

$$x_{t+N_u+1/t} = A^{N_u+1} x_t + (A^{N_u}B + \dots + B)u_{t-1} + (A^{N_u}B + \dots + B)\Delta u_{t/t} + \dots \\ + (AB + B) \Delta u_{t+N_u-1/t}$$

⋮

$$x_{t+N/t} = A^N x_t + (A^{N-1}B + \dots + B)u_{t-1} + (A^{N-1}B + \dots + B)\Delta u_{t/t} + \dots \\ + (A^{N-N_u}B + \dots + AB + B) \Delta u_{t+N_u-1/t}$$

Avec N_u l'horizon de commande.

Finalement, on aura le prédicteur optimal sous forme matricielle :

$$\hat{\mathbf{x}}_t = \begin{bmatrix} x_{t/t} \\ x_{t+1/t} \\ \vdots \\ x_{t+N/t} \end{bmatrix} = \underbrace{\Psi_0 x_t + \Gamma_0 u_{t-1}}_{\text{passé}} + \underbrace{\Lambda \Delta U_t}_{\text{futur}} \quad (2.11)$$

$$\text{Où } \Delta U_t = \begin{bmatrix} \Delta u_{t/t} \\ \Delta u_{t+1/t} \\ \vdots \\ \Delta u_{t+N_u-1/t} \end{bmatrix}, \quad \Psi_0 = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \Gamma_0 = \begin{bmatrix} B \\ AB + B \\ \vdots \\ A^{N-1}B + \dots + B \end{bmatrix},$$

$$\Lambda = \begin{bmatrix} B & 0 & \dots & 0 \\ AB + B & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B + \dots + B & A^{N-2}B + \dots + B & \dots & A^{N-N_u}B + \dots + B \end{bmatrix}$$

Ou encore :

$$\hat{\mathbf{y}}_t = C \hat{\mathbf{x}}_t = \Psi_y x_t + \Gamma_y u_{t-1} + \Lambda_y \Delta U_t \quad (2.12)$$

avec $\Psi_y = C \Psi_0$, $\Gamma_y = C \Gamma_0$, $\Lambda_y = C \Lambda$, et ΔU_t vecteur de taille $s = mN_u$

On définit la fonction coût suivante :

$$J = \sum_{k=1}^N \left\| y_{t+k/t} - r_{t+k/t} \right\|_Q^2 + \sum_{k=0}^{N_u-1} \left\| \Delta u_{t+k/t} \right\|_R^2 \quad (2.13)$$

Où N est l'horizon de prédiction et N_u l'horizon de commande, on assume que $N_u \leq N$, et que $\Delta u_{t+k/t} = 0$ pour $k \geq N_u$, donc $u_{t+k/t} = u_{t+N_u-1/t}$ pour tout $k \geq N_u$.

La fonction coût J pénalise, à chaque instant situé dans l'horizon de prédiction ($1 \leq k \leq N$), les écarts entre les sorties prédites commandées $y_{t+k/t}$ et la trajectoire de référence $r_{t+k/t}$. La trajectoire de référence peut dépendre de mesures effectuées à l'instant k , en particulier, son point de départ peut être la mesure de la sortie y_t ; ainsi elle peut être un ensemble de point fixe ou autre trajectoire prédéterminée.

Pour le cas de la régulation à l'origine la référence r est nulle. Ainsi, pour obtenir la loi de commande, on remplace le prédicteur (2.12) dans la fonction coût :

$$\begin{aligned} J &= \|C \hat{x}_t\|_Q^2 + \|\Delta U_t\|_R^2 \\ &= \|\Psi_y x_t + \Gamma_y u_{t-1} + \Lambda_y \Delta U_t\|_Q^2 + \|\Delta U_t\|_R^2 \end{aligned} \quad (2.14)$$

En prenant le gradient de J par rapport à la commande ΔU_t :

$$\nabla_{\Delta U_t} J = 0 \quad (2.15)$$

On aura l'incrément de commande :

$$\Delta U_t = \frac{1}{2} H_{\Delta u}^{-1} G_{\Delta u} \quad (2.16)$$

$$\text{Où } H_{\Delta u} = \Lambda_y^T Q \Lambda_y + R, \quad G_{\Delta u} = -2 \Lambda_y^T Q (\Psi_y x_t + \Gamma_y u_{t-1})$$

On applique seulement les premières m composantes de ΔU_t au système :

$$\Delta u_t = \begin{bmatrix} I_m, \underbrace{0_m, \dots, 0_m}_{(N_u-1)} \end{bmatrix} \Delta U_t \quad (2.17)$$

2.2.2.b Cas non linéaire

Les modèles ou les fonctions de nature non linéaire sont souvent utilisés pour représenter avec précision les phénomènes physiques, ce qui est d'une importance capitale dans le cas de la commande prédictive à base de modèle.

Considérons le modèle non linéaire continu suivant :

$$\begin{cases} \dot{x} = f(x) + g(x) u \\ y = h(x) \end{cases} \quad (2.18)$$

Où $x \in \mathbb{R}^n$ est le vecteur d'états appartenant à un espace d'état $M \subset \mathbb{R}^n$, $u \in \mathbb{R}^m$ le vecteur d'entrées et $y \in \mathbb{R}^l$ le vecteur de sorties du système. On suppose qu'il existe un point d'équilibre $x_0 \in M$ avec $f(x_0) = 0$. De plus, on suppose que le système est contrôlable, observable, et que tous les états sont mesurables. Les perturbations et les incertitudes du modèle ne sont pas tenues en compte.

Afin de simplifier, le système (2.18) peut être formulé sous la forme linéaire à dépendance d'état :

$$\begin{cases} \dot{x} = A(x) x + B(x) u \\ y = C(x) x \end{cases} \quad (2.19)$$

Où $f(x) = A(x) x$, $g(x) = B(x)$ et $h(x) = C(x) x$.

Dans ce qui suit on note l'évolution dans le temps du vecteur d'état et de la loi de commande, respectivement, par \mathbf{x}_t et \mathbf{u}_t , en ce qui concerne les prédictions le vecteur \mathbf{x}_k représente le vecteur d'état et \mathbf{u}_k la future commande.

a- Solution numérique

Afin d'obtenir la solution optimale de la commande prédictive sujet au système (2.18) ou (2.19) on procède à la discrétisation par la technique d'Euler avec un pas δt :

$$\begin{cases} x_{t+1} = x_t + \delta t \times (A(x_t) x_t + B(x_t) u_t) \\ y_t = C(x_t) x_t \end{cases} \quad (2.20.a)$$

Ou par la méthode de Range Kutta :

$$\left\{ \begin{array}{l} x_{t,\frac{1}{2}} = x_t + \frac{\delta t}{2} \times F(x_t) \\ \check{x}_{t,\frac{1}{2}} = x_t + \frac{\delta t}{2} \times F\left(x_{t,\frac{1}{2}}\right) \\ x_{t,1} = x_t + \delta t \times F\left(\check{x}_{t,\frac{1}{2}}\right) \\ x_{t+1} = x_t + \frac{\delta t}{6} \times \left(F(x_t) + 2\left(F\left(x_{t,\frac{1}{2}}\right) + F\left(\check{x}_{t,\frac{1}{2}}\right) \right) + F(x_{t,1}) \right) \\ F(x) = A(x)x + B(x)u \\ y_t = C(x_t) x_t \end{array} \right. \quad (2.20.b)$$

Considérant le problème de régulation du système discret (2.20.a) ou (2.20.b) à l'origine. L'objectif est de chercher la loi de commande optimale en minimisant le critère à horizon fini suivant:

$$J = \sum_{k=0}^{N-1} (x_k^T Q(x_k) x_k + u_k^T R(x_k) u_k) + x_N^T P(x_k) x_N \quad (2.21)$$

Où x_k est le vecteur d'état prédit à l'instant $t+k$ en appliquant les k premier élément du vecteur d'entrée u_k ($u_0 \dots u_N$) au système (2.20.a) ou (2.20.b), en prenant $x_0 = x_t$. Avec les matrices symétriques $Q(x_k) \geq 0$ (semi définie positive) et $R(x_k) > 0$ (définie positive), sont la matrice de pondération d'état et la matrice de pondération d'entrée, respectivement.

Pour alléger l'écriture on met :

$$\begin{cases} A(x) = A_x \\ B(x) = B_x \\ Q(x) = Q_x \\ R(x) = R_x \end{cases}$$

Où $x_N^T P(x) x_N$ est la fonction coût finale et $P_x = P(x)$ est la solution de l'équation algébrique de Riccati obtenue, DARE (Discrete Algebraic Riccati Equation), par :

$$P_x - A_x^T P_x A_x - Q_x + A_x^T P_x B_x (B_x^T P_x B_x + R_x)^{-1} B_x^T P_x A_x = 0 \quad (2.22)$$

Le problème d'optimisation MPC a horizon fini s'écrit comme suit :

$$\min_u \{x_N^T P_x x_N + \sum_{k=0}^{N-1} (x_k^T Q_x x_k + u_k^T R_x u_k)\} \quad (2.23.a)$$

$$\text{s. à Système (2.a) ou (2.b) avec } x_0 = x_t \quad (2.23.b)$$

$$u_k = -K(x) x_k, N_u \leq k \leq N - 1 \quad (2.23.c)$$

$$x_k \in E_x \in \mathbb{R}^n, 0 \leq k \leq N - 1 \quad (2.23.d)$$

$$u_k \in E_u \in \mathbb{R}^m, 0 \leq k \leq N - 1 \quad (2.23.e)$$

$$y_k \in E_y \in \mathbb{R}^l, 1 \leq k \leq N \quad (2.23.f)$$

Où N est l'horizon de prédiction, N_u est l'horizon de commande, $P_x = P_x^t \geq 0$ est la solution de l'équation de Riccati, $Q_x = Q_x^t \geq 0$ et $R_x = R_x^t \geq 0$ sont des matrices de pondérations, $K(x)$ matrice gain de la commande, avec les ensembles (E_x, E_u et E_y) qui définissent respectivement les contraintes sur les états, les entrées et les sorties.

Il existe de nombreuses approches pour la solution directe du problème d'optimisation associé à MPC. Puisque nous effectuons une simulation hors ligne pour trouver la loi de commande il n'y a pas de limitation sur le temps de calcul et nous pouvons utiliser toute technique d'optimisation capable de gérer les contraintes.

Dans ce travail, nous avons choisi un méta heuristique, le PSO (particle swarm optimization), qui est très simple à mettre en œuvre et qui s'est avéré très efficace (Kennedy & Eberhart, 1995).

b- Solution NMPC basée sur l'Équation de Riccati Dépendante De L'état

Dans ce qui suit nous proposons une approche pour la solution de la MPC non linéaire basée sur l'équation de Riccati dépendante des états, SDRE, State Dependent Riccati Equation. Cette approche consiste à factoriser un système non linéaire (2.18) en une structure linéaire avec des matrices à coefficients dépendant de l'état (Erdem, 2001) tel que donné par (2.19). Sous l'hypothèse que $f(x)$ soit continûment dérivable et $f(0) = 0$, $A(x)$ peut-être trouvé par une paramétrisation avec des matrices dont les éléments dépendent des états, SDC, State Dependent Coefficients.

Proposition 2.1 : (Cloutier, 1997)

Soit $f(x)$ continûment dérivable dans le domaine d'intérêt avec $f(0) = 0$, alors pour tout x appartenant à ce domaine, une paramétrisation SDC de $f(x)$ existe toujours.

Les conditions suivantes doivent être satisfaites pour appliquer les méthodes SDRE.

- 1 La fonction $f(x)$ doit être continûment dérivable
- 2 L'origine est un point d'équilibre du système
- 3 Les matrices de pondération satisfont $(x) \geq 0$, $R(x) > 0$ et les deux sont symétriques
- 4 $(A(x), B(x))$ doit être stabilisable et $(C(x), A(x))$ doit être détectable.

L'approche SDRE procède de manière similaire au régulateur quadratique linéaire infini et la loi de contrôle par rétroaction d'état est obtenue comme suit :

$$u_{op}(k) = (B_x^t P_x B_x + R_x)^{-1} B_x^t P_x A_x x(k) \quad (2.24)$$

Où $P(x)$ est la solution de l'Equation algebrique de Riccati discrète (2.22).

Clairement, l'approche nécessite que l'équation de Riccati ait une solution en tout point $x(t)$ dans le domaine d'intérêt. En raison de la non unicité de $A(x)$, des choix différents peuvent produire des matrices de contrôlabilité différentes et on peut toujours trouver une paire stabilisable $(A(x), B(x))$. Cependant, bien que cela puisse être assez facile pour les systèmes d'ordre inférieur, cela devient laborieux pour les systèmes d'ordre élevé.

Le principal inconvénient de l'approche est que le contrôle peut ne pas être optimal mais plutôt sous-optimal (Cloutier, 1997) Il a été montré (Erdem, 2001) qu'il existe au plus un $P(x)$ qui donne la solution optimale et cela dépend du choix de $A(x)$ ce qui n'est pas une tâche facile. De nombreux auteurs soutiennent que la sous-optimalité n'a pas de conséquence grave pondérée par la qualité de la solution obtenue (Cimen, 2010).

D'autre part, l'équation algébrique discrète de Riccati, Discrete Algebraic Riccati Equation, DARE, doit être résolue à chaque intervalle d'échantillonnage. C'est un problème majeur (Erdem, 2001). L'idéal serait d'obtenir une solution analytique au DARE, mais cela n'est possible que pour des systèmes d'ordre inférieur avec une structure spéciale.

Afin d'appliquer l'approche SDRE pour résoudre le non-linéaire MPC on considère le problème (2.19), (2.23), étant donné $x(t)$ la valeur de l'état obtenu à partir de mesure à l'instant t .

Si la matrice de coût final $P(x) > 0$ et la matrice gain $K(x)$ sont calculés à chaque période d'échantillonnage à partir de la DARE (2.20) alors le problème (2.21) résout exactement le problème SDRE équivalent. Dans les conditions où $(A(x), B(x))$ est stabilisable et $(C(x), A(x))$ est détectable, il hérite de ses propriétés stabilisantes.

Les calculs impliqués dans le NLMPC stable basé sur SDRE sont résumés dans l'algorithme 2.1.

Le problème d'optimisation est résolu en utilisant l'algorithme Particle Swarm Optimisation, PSO (Eberhart & Kennedy, 1995).

Algorithme 2.1 pseudo code NLMPC basée sur SDRE

- 1: **pour** *chaque instant* t
 - 2: Mesurer $\mathbf{x}(t)$
 - 3: Calculer les matrices $\mathbf{A}(\mathbf{x})$, $\mathbf{B}(\mathbf{x})$, $\mathbf{Q}(\mathbf{x})$ et $\mathbf{R}(\mathbf{x})$
 - 4: Résoudre le DARE (2.22) pour obtenir la matrice $\mathbf{P}(\mathbf{x})$
 - 5: Résoudre le programme quadratique pour obtenir le signal de commande $\mathbf{u}(t)$
 - 6: Appliquer $\mathbf{u}(t)$ au système
 - 7: **fin**
-

2.3 Commande prédictive explicite :

Suivant la définition donnée dans (Olaru, 2005), la solution numérique des problèmes d'optimisation est dite soit explicite soit implicite. Quand les variables dépendantes sont définies par des ensembles couplés d'équations, par des matrices de forme variable ou quand des techniques itératives sont nécessaires pour obtenir la solution, la méthode numérique est implicite. En revanche, quand un calcul direct des variables dépendantes peut être fait mettant en évidence des quantités connues, le calcul est alors explicite.

La construction de la solution de la commande prédictive explicite (CPE) est basée sur l'optimisation multiparamétrique (OM). Cette dernière technique d'optimisation permet de déterminer la solution optimale comme étant une fonction explicite de certains variables d'état. Par conséquent, l'OM ne nécessite pas la résolution d'un nouveau problème d'optimisation quand les paramètres changent, car la solution optimale peut être facilement mise à jour en utilisant la fonction pré-calculée.

Dans la littérature Il existe une multitude de méthodes pour la construction de la solution explicite d'un problème d'optimisation multiparamétriques (Bemporad, Morari, Dua, & Pistikopoulos, 2002) et (Olaru, 2005). En ce qui suit, la méthode développée en (Bemporad, Morari, Dua, & Pistikopoulos, 2002) est présenté succinctement.

Définition 2.1 : (Bemporad & Filippi, 2001)

Soit l'ensemble $\mathbb{X} \subseteq \mathbb{R}^n$, \mathbb{X} est polyédrique s'il existe des matrices M_1, M_2 et vecteurs v_1, v_2 de tel sorte que :

$$\mathbb{X} = \{x \in \mathbb{R}^n : M_1 x \leq v_1, M_2 x < v_2\} \quad (2.25)$$

Définition 2.2 : (Bemporad, Morari, Dua, & Pistikopoulos, 2002)

Soit la fonction $z(x) : \mathbb{X} \rightarrow \mathbb{R}^s$, et l'ensemble polyédrique $\mathbb{X} \subseteq \mathbb{R}^n$, $z(x)$ est une fonction affine linéaire par morceaux si \mathbb{X} peut être partitionné en N_r régions polyédrique convexe, CR_i , avec

$$z(x) = a_i x + b_i, \forall x \in CR_i, (i = 1, \dots, N_r) \quad (2.26)$$

La séquence d'entrée optimale U_t peut être obtenue en résolvant un programme quadratique multiparamétrique (mp-QP).

En substituant (2.10) dans (2.6), ainsi le problème d'optimisation peut s'écrire :

$$\min_{U_t} \frac{1}{2} x_t^T Y x_t + \left\{ \frac{1}{2} U_t^T H U_t + x_t^T F^T U_t \right\}, \quad (2.27)$$

s à $GU_t \leq W + Ex_t$

Où les matrices $Y \in \mathbb{R}^{n \times n}$, $H \in \mathbb{R}^{s \times s}$, $F \in \mathbb{R}^{s \times n}$, $G \in \mathbb{R}^{q \times s}$, $W \in \mathbb{R}^q$ et $E \in \mathbb{R}^{q \times n}$ sont en fonction des matrices de pondérations P, Q, R et les contraintes sur les états et la commande, avec q le nombre des contraintes. Pour $P \geq 0$, $Q \geq 0$ et $R > 0$, alors $H = H^T > 0$ et le problème est convexe. Autant plus, et comme l'optimisation est par rapport au U_t , le terme $\frac{1}{2} x_t^T Y x_t$ est généralement omis (Bemporad, Morari, Dua, & Pistikopoulos, 2002).

Donc le problème d'optimisation multiparamétrique (2.27) s'écrit :

$$\min_{U_t} \left\{ \frac{1}{2} U_t^T H U_t + x_t^T F^T U_t \right\}, \quad (2.28)$$

s à $GU_t \leq W + Ex_t$

Avec les conditions KKT du premier ordre pour le problème (2.28) :

$$HU_t + Fx_t + G^T \lambda = 0, \lambda \in \mathbb{R}^q, \quad (2.29a)$$

$$\lambda^T (GU_t - W - Fx) = 0, \quad (2.29b)$$

$$\lambda \geq 0, \quad (2.29c)$$

$$GU_t \leq W + Ex, \quad (2.29d)$$

Les conditions de Karush-Kuhn-Tucker (2.29) sont alors des conditions suffisantes d'optimalité, et la solution optimale U_t est unique.

Le problème (2.28) peut être reformulé en se basant sur la programmation quadratique multiparamétrique (PQM) où le vecteur d'état x_t est un vecteur de paramètres.

On définit la variable suivante :

$$z = U_t + H^{-1}F^T x_t, \quad (2.30)$$

Alors le problème (2.28) s'écrit :

$$\min_z \left\{ \begin{array}{l} \frac{1}{2} z^T H z \\ s \text{ à } Gz \leq W + Sx_t \end{array} \right\}, \quad (2.31)$$

Le problème (2.31) est multiparamétrique par rapport à z , et x_t est le vecteur paramètre.

Où la matrice S s'écrit :

$$S = E + GH^{-1}F^T, \quad (2.32)$$

Considérant les conditions KKT suivantes (Bazaraa, Sherali, & Shetty, 2006) :

$$Hz + G^T \lambda = 0, \lambda \in \mathbb{R}^q, \quad (2.33a)$$

$$\lambda^T (Gz - W - Sx) = 0, \quad (2.33b)$$

$$\lambda \geq 0, \quad (2.33c)$$

$$Gz \leq W + Sx, \quad (2.33d)$$

la solution z^* du problème (2.31) est optimale dans le voisinage de x_t tant que l'ensemble actif est optimal.

Ainsi, suivant la définition 2, la nouvelle loi de commande s'écrit sous forme affine :

$$u_t(x_t) = \begin{cases} F_0 x_t & \text{Si } M_0 x_t \leq v_0 \\ F_1 x_t + g_1 & \text{Si } M_1 x_t \leq v_1 \\ \vdots & \vdots \\ F_{N_r-1} x_t + g_{N_r-1} & \text{Si } M_{N_r-1} x_t \leq v_{N_r-1} \end{cases} \quad (2.34)$$

Ou encore dans une forme compacte :

$$u_t(x_t) = F_i x_t + g_i, \quad x_t \in \mathcal{X}_i \subseteq \mathbb{X} \quad (i = 0, \dots, N_r - 1) \quad (2.35)$$

Où N_r est le nombre de régions, $F_i \in \mathbb{R}^{m \times n}$, $g_i \in \mathbb{R}^m$, M_i et v_i matrices et vecteur à déterminer pour chaque région polyédrique $\mathcal{X}_i = \{x_t : M_i x_t \leq v_i\}$, ($i = 1, \dots, N_r$).

De cette manière, le calcul en ligne est réduit à une évaluation d'une série de fonction au lieu de l'optimisation en temps réel.

La solution explicite optimale est un ensemble de fonctions des variables d'état du système par morceaux, où le domaine d'intérêt est un sous ensemble de l'espace d'état qui est divisé en un nombre fini de régions critiques. Pour chaque région critique, une loi de commande par retour d'état donne la valeur de l'entrée (commande). L'ensemble de ces lois de commande (fonctions par morceaux) forme la solution optimale explicite.

Cependant, l'implémentation en ligne de la solution explicite nécessite la sauvegarde de l'ensemble des fonctions par morceaux et, pour chaque mesure, de trouver la région critique qui contient le vecteur d'état actuel et d'évaluer la loi de commande par retour d'état correspondante.

2.4 Commande prédictive approximante :

Malgré le succès qu'elle a eu dans certaines applications (Bazaraa, Sherali, & Shetty, 2006) et (Di Cairano, Yanakiev, Bemporad, Kolmanovsky, & Hrovat, 2008), la commande prédictive explicite a toujours des limitations concernant le nombre d'ensemble PWA (Bemporad, Oliveri, Poggi, & Storaice, 2011) et (Di Cairano, Yanakiev, Bemporad, Kolmanovsky, & Hrovat, 2008). En effet, le nombre des ensembles générés dépend directement d'une façon exponentielle du nombre de variables d'entrées et de sorties et des contraintes incluses dans le problème d'optimisation. Tant que leurs valeurs augmentent le nombre de régions augmente (Bemporad A. , 2011) (voir figure 2.3). Afin de simplifier la complexité de la CPE, plusieurs

approches approximantes de la solution par morceaux ont été proposées pour alléger la charge de calcul (Bakaráč, Holaza, Kalúz, Klaučo, Löfberg, & Kvasnica, 2018), (Bemporad, Oliveri, Poggi, & Storace, 2011) et (Hovd, Scibilia, Maciejowski, & Oлару, 2009).

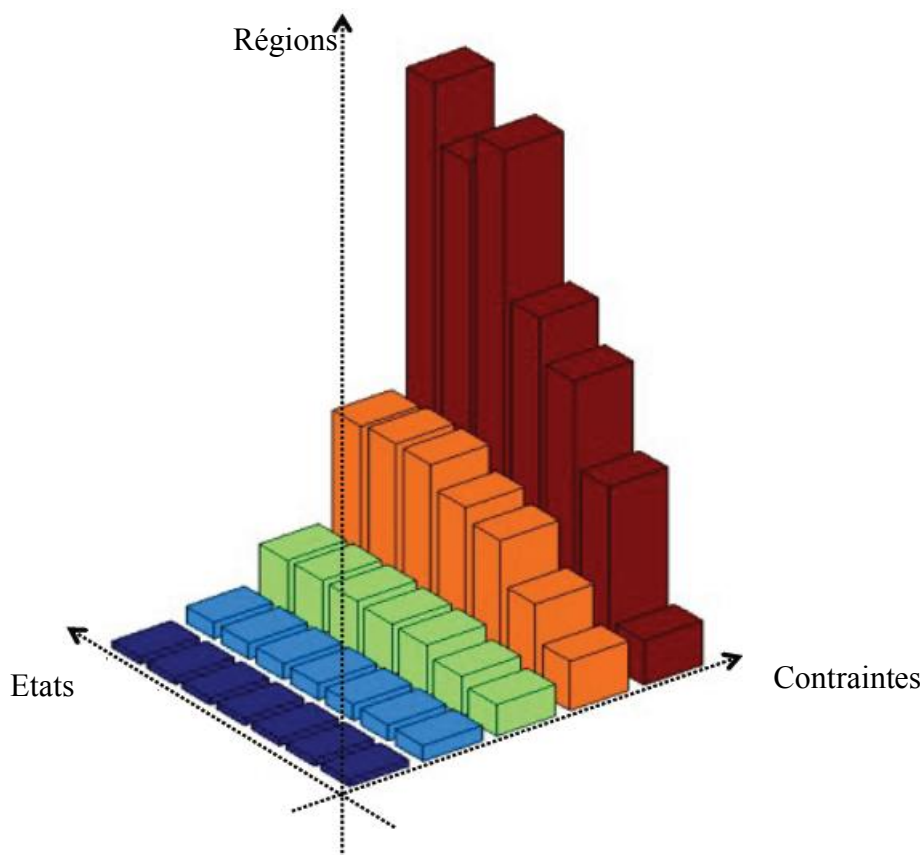


Figure (2.3) Relation entre les nombres d'Etats, Régions et Contraintes (Bemporad A. , 2011).

Fondamentalement, ces méthodes résolvent des problèmes d'ajustement de courbe ((Johansen & Grancharova, 2003), (Bemporad & Filippi, 2003), (Canale, Fagiano, & Milanese, 2009), (Canale, Fagiano, & Milanese, 2010), (Kvasnica, Lofberg, & Fikar, 2011), (Bemporad, Oliveri, Poggi, & Storace, 2011) et (Scibilia, Hovd, & Oлару, 2012)): étant donné la loi de commande optimale calculée hors ligne, elles calculent une loi de contrôle approximante qui tente de minimiser l'erreur entre la valeur approximante et la valeur optimale. En effet, un ensemble de solutions optimales est déterminé pour entrainer le nouveau contrôleur, en résolvant le problème (2.6). La loi de commande approximante est calculée sur la base de ces données optimales.

2.4-1 Approches basées sur l'apprentissage :

Les méthodes d'apprentissage supervisé sont les procédures les plus simples pour concevoir des APC car elles sont basées sur la collecte d'un ensemble de paires de solutions MPC optimales $[x^{*l}(t), u^{*l}(t)]$, ($l = \dots L$) pour former un nouveau système afin d'imiter le contrôleur MPC optimal d'origine.

Notez que la technique d'approximation de MPC a été suggérée tôt en utilisant une approximation de réseau de neurones dans (Parisini & Zoppoli, 1995).

Principalement dans ces approches, ils recherchent un vecteur de paramètres qui minimise l'erreur d'approximation

$$\min_{\theta} \sum_x \|u^*(x(t)) - \hat{u}(\theta, x(t))\|^2 \quad (2.36)$$

où la précision avec laquelle la nouvelle loi de commande $\hat{u}(\theta, x(t))$ se rapproche de l'optimal $u^*(x(t))$. est évaluée de telle sorte que

$$\|u^*(x(t)) - \hat{u}(\theta, x(t))\| < \varepsilon, \forall x(t) \in \mathcal{X} \quad (2.37)$$

avec $\varepsilon \in R$, $\varepsilon > 0$.

Les réseaux de neurones (RN) :

Dans (Parisini & Zoppoli, 1995) les auteurs choisissent un réseau de neurones à action directe (RN) multicouche pour approcher la loi de contrôle optimale, avec une fonction d'activation sigmoïde décalée $g(x) = \tanh(x)$. Le développement des résultats de stabilité était essentiellement lié à la relaxation de la contrainte d'état terminal $x(t + N)$ et en mettant en œuvre une procédure d'essais et d'erreurs pour tester si la stabilité a bien été atteinte. Les auteurs de (Pin, Filippo, Pellegrino, Fenu, & Parisini, 2013) ont suivi (Parisini & Zoppoli, 1995) et utilisé des RN avec des fonctions d'activation douces pour leurs propriétés favorables, qui permettent de réduire la complexité de l'approximation par rapport à l'approche de l'interpolation au plus proche voisin (*nearest point interpolation*). Où la technique garantit la stabilité pratique de l'entrée à l'état par rapport aux erreurs induites par l'approximation.

Dans (Paulson & Mesbah, 2020), les auteurs ont abordé le problème de la stabilité en boucle fermée et de la faisabilité d'une MPC linéaire robuste approximante pour les systèmes soumis à une incertitude additive. Un réseau de neurones profonds (DNN) a été utilisé pour approximer la loi MPC robuste. La stabilité a été vérifiée a posteriori à l'aide des fonctions de Lyapunov de l'ISS.

Polynôme:

Dans (Kvasnica, Lofberg, & Fikar, 2011) la solution optimale est issue de la résolution du problème d'optimisation linéaire suivant :

$$\begin{aligned}
 \min_{U_t} \left\{ J(x_t, U_t) = \left\| P x_{t+N/t} \right\|_p + \sum_{k=0}^{N-1} \left(\left\| Q x_{t+k/t} \right\|_p + \left\| R u_{t+k/t} \right\|_p \right) \right\} \\
 s. \text{ à } x_{t+k+1/t} = f_{PWA} \left(x_{t+k/t}, u_{t+k/t} \right) \\
 \quad = A_d x_{t+k/t} + B_d u_{t+k/t} + a_d, d = 1, \dots, N_r, \\
 x_{t+k/t} \in \mathbb{X}, k = 0, \dots, N \\
 u_{t+k/t} \in \mathbb{U}, k = 0, \dots, N-1
 \end{aligned} \tag{2.38}$$

Avec $\| \cdot \|_p$, ($p = \{1, \infty\}$) est la norme standard 1 ou ∞ , les systèmes considérés $x_{t+k} = f_{PWA}(x_t, u_t)$ sont discrets linéaires par morceaux.

De plus, afin de garantir la stabilité, l'auteur a utilisé les tubes de stabilité définis comme suit :

$$\mathcal{S}(V, \gamma) = \left\{ \begin{bmatrix} x_t \\ u_t \end{bmatrix} \mid u_t \in \mathbb{U}, x_t \in \mathbb{X}, f(x_t, u_t) \in \mathbb{X}, V(f(x_t, u_t)) \leq \gamma V(x_t) \right\} \tag{2.39}$$

V fonction de Lyapunov, $\gamma \in [0, 1)$.

Ainsi, l'approximation de la loi de commande optimale est réalisée par un polynôme $\hat{\mu}(x_t)$ de degré δ fixé a priori :

$$\hat{\mu}(x_t) = \alpha_1 x_t^1 + \alpha_2 x_t^2 + \dots + \alpha_\delta x_t^\delta, \tag{2.40}$$

Où $\alpha_i \in \mathbb{R}^{m \times n}$, $i = 1, \dots, \delta$, sont les coefficients à déterminer, x_t^i est l'élément à la puissance i du vecteur $x_t \in \mathbb{R}^n$. Davantage, pour garantir la stabilité la loi approximée doit remplir la condition (2.39) de telle sorte $\begin{bmatrix} x_t \\ \hat{\mu}(x_t) \end{bmatrix} \in \mathcal{S}(V, \gamma)$.

Interpolation :

Les auteurs de (Summers, Jones, Lygeros, & Morari, 2009) ont proposé d'approximer la loi de commande optimale par une interpolation adaptative en ondelettes. Semblable à (Johansen & Grancharova, 2003) la loi de contrôle a été définie sur un ensemble hiérarchique d'hyper-cubes qui fournit un temps de calcul en ligne rapide. Le choix d'une interpolation de second ordre aboutit à une loi d'interpolation barycentrique qui se situe dans l'enveloppe convexe des points interpolés. Sur la base de cette dernière propriété, la faisabilité et la stabilité ont été évaluées avec précision à priori par un test hiérarchique.

2.4-2 Formulation basée sur les PWA :

Les approches basées sur PWA sont les plus populaires dans APC car elles découlent directement de la solution MPC explicite. L'idée principale est de trouver des contrôleurs approximatifs à faible complexité pour les contrôleurs PWA à haute complexité. Les nouvelles lois approximantes définies sur des partitions avec moins de régions que les partitions d'origine et/ou des régions de formes plus « régulières » qui améliorent la mise en œuvre en ligne en termes de consommation de temps et d'efficacité mémoire, tout en conservant d'importantes propriétés en boucle fermée telles que stabilité et satisfaction des contraintes.

Dans (Bemporad & Filippi, 2003), les auteurs ont proposé une approximation de la solution explicite basée sur l'introduction des variables de relaxation, notées $s \in \mathbb{R}^q$, afin de relaxer les conditions d'optimalité KKT avec certains vecteurs de relaxation ϵ pour déterminer le degré d'approximation. Cette dernière définit un polyèdre sur l'espace (U, x, λ, s) , ceci donne une région critique approximante CR_ϵ qui est la projection du dernier polyèdre dans l'espace des x . La solution approximante, notée par $\hat{U}_t(x)$, est de la forme $\hat{U}_t(x) = \hat{a}x + \hat{b}, \forall x \in CR_\epsilon$. Alors la solution $\hat{U}_t(x)$ est obtenue par la résolution d'un problème de moindres carrés quadratique avec contraintes suivant :

$$\min_{\hat{a}, \hat{b}} \left\{ \sum_{i=1}^h \left\| \begin{bmatrix} I_m & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} (U_t^*(\vartheta_i) - (\hat{a}\vartheta_i + \hat{b})) \right\|^2 \right\}, \quad (2.41)$$

$s \text{ à } G(\hat{a}\vartheta_i + \hat{b}) \leq W + SV_i, \quad i = 1, \dots, h, \quad \hat{a} \in \mathbb{R}^{s \times n} \hat{b} \in \mathbb{R}^s$

Avec les vecteurs $\vartheta_i \in \mathbb{R}^n, (i = 1, \dots, h)$ sont l'ensemble $\vartheta \in \mathbb{R}^n$ des vertex des régions critique. La stabilité est garantie *à priori* en imposant certaines conditions sur les vecteurs de relaxation ϵ . Par ailleurs, (Johansen & Grancharova, 2003) ont proposé de représenter l'espace d'état par un arbre de recherche orthogonale. De cette façon, la solution explicite est calculée pour 2^n vertex de l'hyper cube $X_0 \subseteq \mathbb{X} \in \mathbb{R}^n$ en solvant 2^n QPs. Ainsi la résolution de (2.39) donne la solution approximante. La stabilité est analysée à travers la tolérance sur l'erreur d'approximation, où une limite sur cette tolérance peut être calculée *à priori*. Johanson dans (Johansen, 2004) a procédé de la même façon que (Johansen & Grancharova, 2003) en utilisant un arbre de recherche binaire et en approximant la solution optimale obtenue par résolution du problème multiparamétrique non linéaire.

Dans ce travail (Bemporad, Oliveri, Poggi, & Storace, 2011), les auteurs ont adopté une classe spéciale de fonctions canonique affine linéaire par morceaux, afin d'approximer la loi de commande optimale de la MPC.

Selon (2.34) $\mathbb{X} = \bigcup_{i=0}^{N_r-1} \mathcal{X}_i$ ainsi la loi de commande PWA est $u_t(x_t) = F_i x_t + g_i, x_t \in \mathcal{X}_i$. Le domaine compact \mathbb{X} est partitionné en un ensemble de simplexes ayant la même forme, assumant que $\mathbb{X} = \{x \in \mathbb{R}^n: x_{minj} \leq x_j \leq x_{maxj}, j = 1, \dots, n\}$. Les éléments de cette class, appelé fonctions affine linéaire par morceaux simpliciaux ou PWAS (PWA Simplicial). Ainsi, \mathbb{X} est partitionné en p_j sous intervalles $(x_{minj} - x_{maxj})/p_j, j = 1, \dots, n$. Donc, est divisé en $\prod_{j=1}^n p_j$ hyper rectangles et admet $N_v = \prod_{j=1}^n (p_j + 1)$ vertex v . De plus, chaque rectangle est partitionné en $n!$ simplex ce qui donne $L = n! \prod_{j=1}^n p_j$ simplex \mathcal{S}_i , tel que $\mathbb{X} = \bigcup_{i=0}^{L-1} \mathcal{S}_i$.

Afin de trouver les gains du contrôleur un problème QP ou LP est résolu pour trouver les valeurs de la loi de commande approximante aux vertex $\hat{u}(v_i), i = 1, \dots, N_v$, ensuite, ces valeurs sont utilisées pour déterminer les gains \hat{F}_i, \hat{g}_i du contrôleur suivant :

$$\hat{u}(x_t) = \hat{F}_i x_t + \hat{g}_i, x_t \in \mathcal{S}_i, i = 0, \dots, L - 1 \quad (2.42)$$

Ainsi, l'algorithme de localisation du simplex \mathcal{S}_i en question, lors de l'implémentation des fonctions PWAS sur circuit, est basée sur les lemmes de Kuhn (Kuhn, 1960), (Kuhn, 1968) et (Chien & Kuh, 1977), où le temps d'évaluation des PWAS est fonction de n et indépendant de p_j .

Les propriétés de la stabilité sont analysées *à posteriori* par le biais des fonctions de Lyapunov linéaires par morceaux autour l'origine et certaines informations sur l'ensemble polyédrique contenant \mathbb{X} .

De la même manière (Scibilia, Hovd, & Olaru, 2012) ont simplifié la répartition de l'ensemble faisable \mathbb{X} en utilisant la Triangulation de Delaunay, et suivant la même procédure que (Bemporad, Oliveri, Poggi, & Storace, 2011) pour déterminer les gains \hat{F}_i, \hat{g}_i du contrôleur suivant:

$$\hat{u}(x_t) = \hat{F}_i x_t + \hat{g}_i, \quad x_t \in \mathcal{S}_i, i = 0, \dots, mN - 1 \quad (2.43)$$

Cependant, la stabilité a été adressée de la même façon que (Johansen & Grancharova, 2003).

2.4-3 Autre approche :

Set membership :

Dans (Canale, Fagiano, & Milanese, 2009) les auteurs ont utilisé une des techniques de la théorie des ensembles, dite *set membership* ou appartenance à un ensemble, pour construire un estimateur afin d'avoir une approximation de la loi de commande optimale. La technique repose sur l'utilisation de certaines informations sur la fonction de la loi de commande optimale à approximer, ainsi que le calcul d'un nombre fini h , hors ligne, de solution exacte au problème d'optimisation :

$$\tilde{u}_k = (\tilde{u}_{k,1}, \dots, \tilde{u}_{k,m}) = (f_1^e(\vartheta_k), \dots, f_m^e(\vartheta_k)) = f^e(\vartheta_k), (k = 1, \dots, h) \quad (2.44)$$

utilisant $\vartheta_k \in \mathcal{X} \subseteq \mathbb{R}^n$ conditions initiales issues d'un maillage uniforme de l'ensemble compact \mathcal{X} où l'approximation de la loi optimale est effectuée. De plus, la fonction approximante $\hat{f}(x)$ doit satisfaire les deux conditions suivantes :

1- Les contraintes sont satisfaites :

$$\hat{f}(x) \in \mathbb{U}, \forall x \in \mathcal{X} \quad (2.45)$$

2- Pour un h donné, l'erreur de l'approximation est bornée par $\xi(h)$ de sorte que :

$$\|f^e(x) - \hat{f}(x)\|_2 \leq \xi(h) \in \mathbb{R}_+, \forall x \in \mathcal{X} \quad (2.46)$$

Sachant que

$$\lim_{h \rightarrow \infty} \xi(h) = 0. \quad (2.47)$$

Le système en boucle fermée $x_{t+1} = F^e(x_t) = A x_t + B f^e(x)$ est supposé asymptotiquement stable à l'origine pour toute condition initiale $x_0 \in \mathcal{X}$, ou encore si la solution $\varphi^e(t, x_0) = \underbrace{F^e(F^e(\dots F^e(x_0) \dots))}_{t \text{ fois}}$ est bornée. Par analogie le système $x_{t+1} = A x_t + B \hat{f}(x)$ est asymptotiquement stable si $\hat{\varphi}(t, x_0)$ est bornée.

Ainsi, Canale et al dans (Canale, Fagiano, & Milanese, 2010) ont proposé l'extension des travaux réalisés dans (Canale, Fagiano, & Milanese, 2009) pour le cas linéaire au cas non linéaire où ils ont utilisé la technique SM ou UBB pour calculer une borne du pire cas de l'erreur d'approximation et de la réduire à une valeur arbitrairement petite.

Data mining :

Dans cette technique de (Goebel & Allgower, 2013) le but principal est de trouver une simplification du problème (2.6) en résolvant un nombre de problèmes paramétrés comme (2.28) au sens de \tilde{U} où $U = M_i \tilde{U} + a_i, i \in \{1, \dots, K\}$ avec $\tilde{U} \in R^q, (q < mN)$ les paramètres, $M_i \in R^{mN \times q}$ les paramétrages et $a_i \in R^{mN}$ partie constante supplémentaire.

Ces dernières paramétrisations sont obtenues hors ligne par exploration de données qui approchent les solutions optimales de MPC, puis le système résultant est résolu en ligne pour calculer l'entrée de contrôle réelle. Par construction, la satisfaction des contraintes et la stabilité de la boucle fermée sont garanties.

Programmation Dynamique :

Dans (Bakaráč, Holaza, Kalúz, Klaučo, Löfberg, & Kvasnica, 2018), une solution explicite de la commande prédictive est synthétisée par le biais de la programmation dynamique approximante, ce qui produit une loi de rétroaction stabilisante par construction.

2.5 Commande prédictive rapide :

La commande prédictive dite rapide ou Fast MPC est une collection de méthodes numériques rapides qui exploitent la structure du problème MPC pour le redéfinir et rendre la résolution du problème plus rapide. Principalement, des algorithmes d'optimisation sont utilisés, tel que le point intérieur en (Wang & Boyd, 2010) où l'auteur a proposé des simplifications sur la

méthode originale qui donne un algorithme approximant qui peut être 100 fois plus rapide que l'optimiseur générique. A titre d'exemple, l'algorithme proposé peut calculer la solution d'un problème de 12 états et 3 entrées de commande en 5 ms avec un horizon de prédiction de 30 échantillons.

La méthode du gradient rapide a été utilisée par (Kögel & Findeisen, 2011) et (Jerez, Goulart, Richter, Constantinides, Kerrigan, & Morari, 2013) pour résoudre le problème MPC. En effet, la technique a été utilisée dans (Kögel & Findeisen, 2011) combinée à la méthode lagrangienne augmentée, où cette dernière a été utilisée pour gérer les contraintes d'égalité. Cependant, dans (Jerez, Goulart, Richter, Constantinides, Kerrigan, & Morari, 2013), le solveur à gradient rapide a été implémenté sur un FPGA pour les problèmes MPC quadratiques linéaires avec des contraintes d'entrée. Le papier rapporte une bonne performance, à un taux d'échantillonnage de 1 MHz, sur un système de microscope à force atomique industriel (AFM). À l'opposé des derniers articles, (Rubagotti, Patrinos, Guiggiani, & Bemporad, 2014) a proposé une formulation qui garantit la stabilité malgré l'utilisation de solveurs QP inexacts.

2.6 Outils Solution MPC :

2.6.1 Logiciel

Avec l'avancée technologique, spécialement l'outil informatique, plusieurs méthodes récentes de résolution des problèmes de contrôle optimal (OCP) ont été développées. La mise en œuvre d'une approche de contrôle optimal intégrée puissante a joué un grand rôle dans l'introduction de MPC dans des applications en temps réel, permettant le développement de logiciels tels que la boîte à outils multiparamétrique MPT, une boîte à outils Matlab populaire pour la MPC explicite (Kvasnica, Grieder, Baotić, & Morari, 2004) et (Herceg, Kvasnica, Jones, & Morari, 2013). Pour la MPC à base de modèle non linéaire (NMPC) l'environnement logiciel appelé ACADO toolkit (Houska, Ferreau, & Diehl, 2011 a) offre la possibilité de générer un environnement logiciel efficace en code C basé sur la méthode d'itération en temps réel de Gauss-Newton (Houska, Ferreau, & Diehl, 2011 b). Le logiciel est implémenté en tant que code C++ autonome et fournit une manière symbolique pour formuler l'OCP. Une extension de ce logiciel nommée ACADOS est présentée dans (Verschuere R. , et al., 2018) et (Verschuere R. , et al., 2019). Acados offre de nombreuses fonctionnalités : des algorithmes

de contrôle optimal rapidement embarqués et efficaces en C, une bibliothèque d'algèbre linéaire haute performance basée sur BLASFEO, des interfaces enable Python et MATLAB et CasADi. Une autre boîte à outils appelée VIATOC a été développée dans (Kalmari, Backman, & Visala, 2015) pour la génération de code NMPC. L'outil VIATOC emploie un algorithme de projection de gradient afin de résoudre le problème d'optimisation NMPC. Les performances du contrôleur résolu par le logiciel VIATOC ont été comparées à celles générées avec ACADOS toolkit. MATMPC (Chen, Bruschetta, Picotti, & Beghi, 2019) et (Lars'en, et al., 2019) est sur une boîte à outils open source interfacée dans MATLAB pour résoudre la conception de contrôleur prédictif de modèle non linéaire. Il permet une large gamme d'applications NMPC et offre de nombreux composants d'algorithme, comptage de différenciation automatique, prise de vue multiple directe, condensation, solveur de programme quadratique linéaire (QP) et mondialisation. La boîte à outils fournit un moyen de programmation de haut niveau depuis son implémentation dans MATLAB. Dans (Jacquet, Corsini, Bicego, & Franchi, 2020) MATMPC a été utilisé pour concevoir et implémenter un contrôleur en temps réel pour véhicule aérien multi-rotor basé sur un cadre NMPC à contraintes de perception sur le contrôle prédictif à base de modèle dans différents domaines.

2.6.2 Solution MPC par PSO :

La solution de MPC peut être calculée par l'algorithme d'optimisation par essaim particulaire PSO (particle swarm optimization) (Eberhart & Kennedy, 1995), qui est initialisé dans l'espace du problème considéré par une population de solutions, appelées particules, avec une position et une vitesse aléatoires pour chaque particule. A chaque nouvelle itération, chaque particule se déplace selon sa vitesse actuelle vers sa meilleure solution *pbest* et la meilleure solution obtenue dans son voisinage *gbest* (voir Figure 2.4), selon les deux formules suivantes:

Formule de mise à jour de la vitesse

$$v_{k+1}^i = wv_k^i + c_1r_1(p_k^i - x_k^i) + c_2r_2(p_k^g - x_k^i) \quad (2.48)$$

Formule de mise à jour de la position

$$x_{k+1}^i = x_k^i + v_{k+1}^i \quad (2.49)$$

Où:

x_k^i :	Position de la particule ;	w :	inertie,
v_k^i :	Vitesse de la particule ;	c_1, c_2 :	Facteurs de corrections ;
p_k^i :	La meilleure position locale ;	r_1, r_2	variables aléatoires entre 0 et 1,
p_k^g :	La meilleure position globale ;	i	indice particule,
k	indice itération.		

Clerc en 2002 (Clerc & Kennedy, 2002) a suggéré les valeurs optimales suivantes pour les paramètres PSO : $\omega = 0.7298$ et $c1 = c2 = 1.49618$.

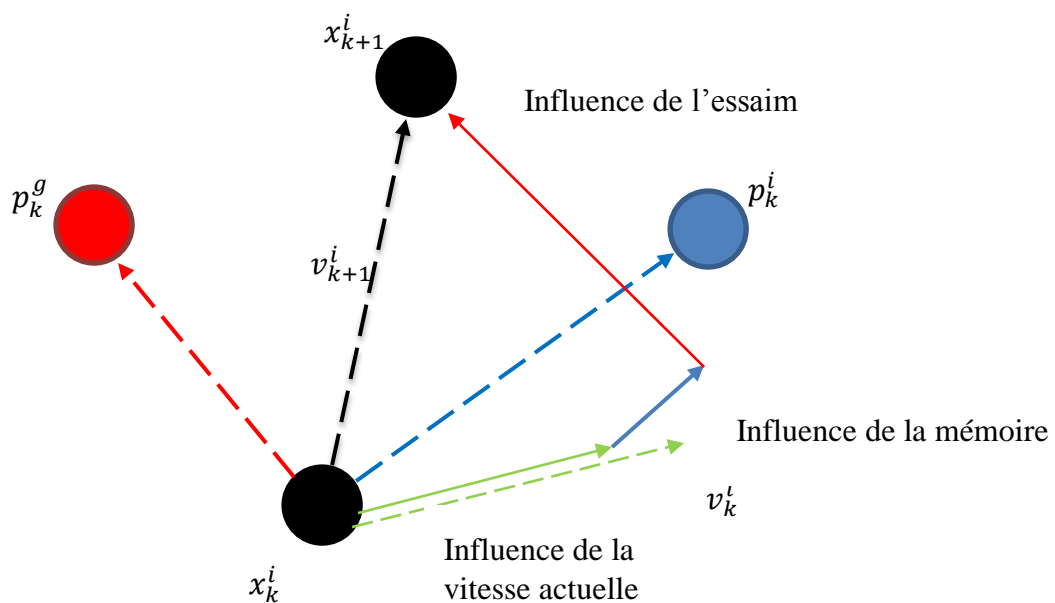


Figure 2.4 Illustration du mouvement des particules d'une itération de PSO (Wang, Tan, & Liu, 2018)

L'Algorithme 2.2 donne le pseudo code de la technique PSO.

Algorithme 2.2 pseudo code PSO

1: **pour** *Tout les particules*

2: Initialisation de la position particule en valeur aléatoire dans l'espace de travail

3: Initialisation de la vitesse particule en valeur nul

4: **fin**

5: **répéter**

6: **pour** *Toute particule*

7: calculer la valeur du coût

8: **Si** le coût est mieux que le meilleur coût enregistré

9: Sauvegarder la valeur actuelle comme meilleur coût *pbest*

10: **fin**

11: Choisir la particule avec la meilleure valeur du coût comme étant *gbest*

12: **pour** *Tout les particules*

13: mise à jour de la vitesse (eq 2.48)

14: mise à jour de la position (eq 2.49)

15: **fin**

16: **jusqu'à** *itération max*

2.7. Conclusion

Ce chapitre introduit les concepts de base de la commande prédictive. Dans la première partie, une présentation générale de la commande prédictive est donnée pour les deux cas linéaire et non linéaire. Les approches de la commande prédictive présentées tournent autour de la problématique de cette thèse. En effet, pour réduire le temps d'exécution la commande prédictive explicite, CPE, a été introduite via la programmation multiparamétrique. Malgré le succès qu'elle a eu, la CPE peut être implémenté que pour des petits problèmes de contrôle (où le nombre d'état n'excède pas cinq (Wang & Boyd, 2010)). Un état de l'art est établi des principales approches utilisées pour la commande prédictive approximante, où cette dernière est une bonne alternative pour accélérer considérablement le calcul de l'action de commande.

Chapitre 3

Commande prédictive approximante par apprentissage flou

3.1 Introduction

Cette section présente le problème de l'approximation de la loi de commande prédictive optimale par un système flou Takagi Sugeno. Étant donné la structure du système flou, sa sortie doit être aussi proche que possible de la loi optimale sans violation des contraintes.

L'approximation est effectuée à l'aide d'un ensemble de solutions optimales du MPC, calculées hors ligne. Ces solutions optimales sont déterminées pour un nombre donné de conditions initiales uniformément réparties sur un sous-ensemble compact inclus dans l'ensemble des solutions possibles et incluant l'origine.

La commande prédictive, dans sa forme classique, est une technique de commande gourmande en temps de calcul, pour cette raison elle était toujours mieux adaptée aux systèmes lents à grande période d'échantillonnage tel que les procédés chimiques. La commande prédictive explicite est une alternative pour réduire le temps de calcul afin de permettre d'élargir l'application de la MPC aux problèmes de contrôle rapides (temps d'échantillonnage petit). En effet, le problème MPC est transformé en un problème multi-paramétrique et résolu hors ligne, où l'espace des états est partitionné en régions régies par une loi de commande linéaire, chacune étant explicitement liée aux états. Le contrôleur résultant est alors un ensemble de contrôleurs linéaires, un seul actif à la fois.

3.2. Apprentissage supervisé

L'apprentissage supervisé fait partie de l'apprentissage automatique. Ce dernier est un ensemble de techniques qui utilisent des algorithmes qui apprennent de manière itérative à partir des données et permettent à la machine d'améliorer son comportement. L'organigramme suivant donne un aperçu général d'un système d'apprentissage automatique

Afin de réaliser un apprentissage supervisé, il faut effectuer les étapes suivantes :

- Rassemblez un ensemble de données pour l'apprentissage. Cet ensemble doit être représentatif de l'utilisation réelle de la fonction. Ainsi, un ensemble d'entrée-sorties est collecté, soit à partir d'experts humains, soit à partir de mesures.
- Déterminez la représentation des caractéristiques descriptives de l'ensemble de données en question. Le nombre de caractéristiques ne doit pas être trop important, à cause de la malédiction de la dimensionnalité ; mais doit contenir suffisamment

d'informations pour prédire avec précision la sortie. (cette étape peut être facultative où on peut utiliser les données brut)

- Déterminer et Exécutez un algorithme d'apprentissage sur l'ensemble d'apprentissage rassemblé. Certains algorithmes d'apprentissage supervisé nécessitent des paramètres de contrôle. Ces paramètres peuvent être ajustés en optimisant les performances sur un sous-ensemble (appelé ensemble de validation) de l'ensemble d'apprentissage.
- Évaluer la précision de la fonction apprise. Après ajustement et apprentissage des paramètres, les performances de la fonction résultante doivent être mesurées sur un ensemble de test distinct de l'ensemble d'apprentissage.

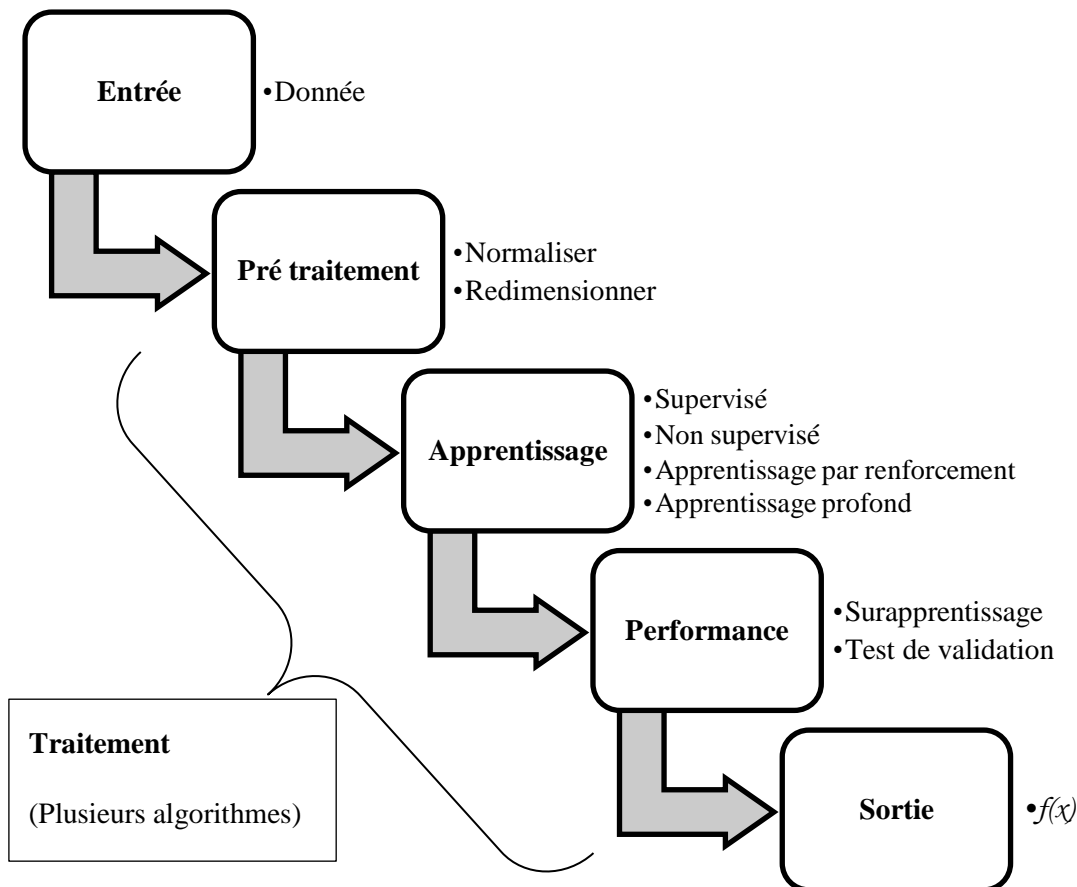


Figure 3.1 Schéma de principe du processus de l'apprentissage automatique

3.3. Système flous de type Takagi-Sugeno:

Proposés par Takagi et Sugeno (Takagi & Sugeno, 1985) ces systèmes flous, décrit par des règles *Si-Alors* floues, représentent une fonction ou un système non linéaire par des relations d'entrées-sorties linéaires locales. Un système flou Takagi-Sugeno (TS) est défini de la manière que combinaison d'entrée donne une implication floue $f_i(x(t))$ exprimée par une fonction de système linéaire. La fusion des dynamiques locales linéaires donne une fonction floue globale du système non linéaire originale.

En outre, les systèmes flous TS sont souvent utilisés pour la représentation des systèmes non linéaire. En fait, il est démontré que les systèmes flous de Takagi-Sugeno sont des approximateurs universels.

En ce qui suit, on donne un aperçu sur la structure générale des systèmes flou et les modèles flous de type TS.

3.3.1. Structure générale d'un système flou Takagi Sugeno discret

Pour un vecteur $x(t) \in \mathbb{R}^n$, la structure générale de la i -ème règle d'un système flou de type T-S discret est :

$$\text{Si } x_1(t) \text{ est } S_{i1} \text{ et } x_2(t) \text{ est } S_{i2} \dots \text{et } x_n(t) \text{ est } S_{in} \text{ alors } \tilde{x}_i(t+1) = f_i(x(t)) \quad (3.1)$$

Où $f_i: \mathbb{R}^n \rightarrow \mathbb{R}^n$. Avec $i = 1, \dots, r$ (r est le nombre de règles). S_{ij} sont les ensembles flous associés au n variables du vecteur $x(t)$.

La fusion donne la sortie du système flou

$$x(t+1) = \sum_{i=1}^r h_i(z(t)) \tilde{x}_i(t+1) \quad (3.2.a)$$

ou encore

$$x(t+1) = \sum_{i=1}^r h_i(z(t)) f_i(x(t)) \quad (3.2.b)$$

Avec $h_i(z(t)) = \frac{\omega_i(x(t))}{\sum_{i=1}^r \omega_i(x(t))}$ est le poids normalisé de pondération pour chaque règle, et $\omega_i(z(t)) = \prod_{j=1}^p S_{ij}(x_j(t))$. $S_{ij}(x_j(t))$ est le degré d'appartenance de $x_j(t)$ à l'ensemble S_{ij} .

3.3.2. Structure d'un modèle flou de type T-S :

La règle d'un modèle flou de type T-S est de la forme suivante (Takagi & Sugeno, 1985):

$$\text{Si } z_1(t) \text{ is } S_{i1} \text{ et } z_2(t) \text{ is } S_{i2} \dots \text{et } z_p(t) \text{ is } S \text{ alors } \dot{x}(t+1) = A_i x(t) + B_i u(t) \quad (3.3)$$

Où $x(t) \in \mathbb{R}^n$ le vecteur d'état, $u(t) \in \mathbb{R}^m$ le vecteur des entrées $A_i \in \mathbb{R}^{n \times n}$ et $B_i \in \mathbb{R}^{m \times n}$. Avec $i = 1, \dots, r$ (r est le nombre de règles). S_{ij} sont les ensembles flous associés au p variables de prémisses $z(t)$ (Figure 3.2),

La fuzzification suivant la méthode du centre de gravité donne la sortie du système flou

$$x(t+1) = \sum_{i=1}^r h_i(z(t)) (A_i x(t) + B_i u(t)) \quad (3.4)$$

Avec $h_i(z(t)) = \frac{\omega_i(z(t))}{\sum_{i=1}^r \omega_i(z(t))}$ est le poids normalisé de pondération pour chaque règle, et $\omega_i(z(t)) = \prod_{j=1}^p S_{ij}(z_j(t))$. $S_{ij}(z_j(t))$ est le degré d'appartenance de $z_j(t)$ à l'ensemble flou S_{ij} .

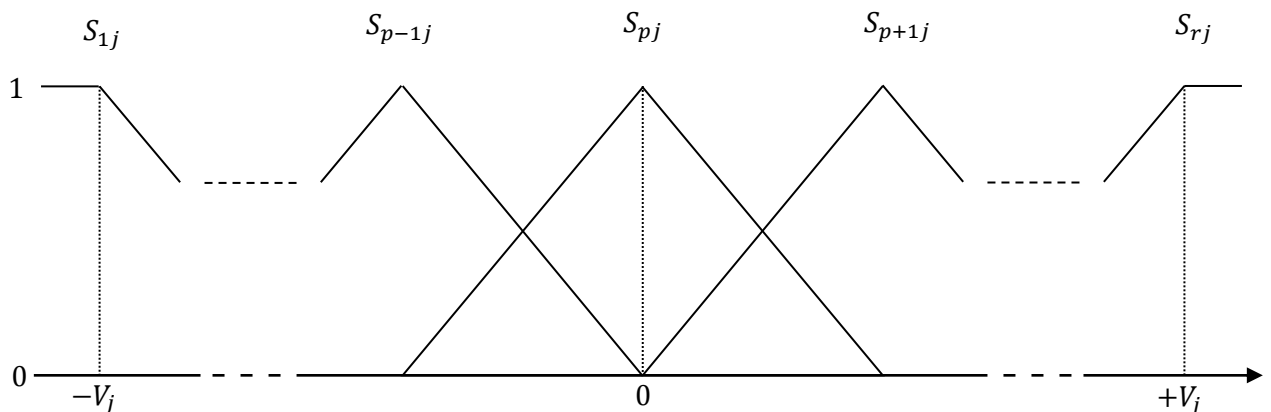


Figure 3.2 Fonctions d'appartenance système flou de type TS

3.3.3. Secteur de non linéarité (Sector nonlinearity) :

L'approche de secteur de non linéarité (Tanaka, Ikeda, & Wang, 1998), (Tanaka, Hori, & Wang, 2001) et (Tanaka & Wang, 2001) garantit la construction d'un modèle flou qui représente exactement le modèle non linéaire original. L'idée générale est de trouver des limites globales du secteur où évolue le système non linéaire.

Soit le système non linéaire suivant :

$$\begin{aligned} f: \mathbb{R} &\rightarrow \mathbb{R} \\ x(t) &\rightarrow f(x(t)) \end{aligned} \tag{3.5}$$

Le système f est dit appartenir au secteur $[a_1, a_2]$ si pour tout x_t on a $f(x_t)$ est entre les droites $a_1x(t)$ et $a_2x(t)$, i.e, f appartient au secteur $[a_1, a_2]$ si $\forall x(t) \in \mathbb{R}$, $f(x(t)) \in [a_1, a_2]x(t)$.

La figure 3.2 illustre le concept de secteur global de non linéarité. Cependant, il n'est pas toujours facile de trouver ces secteurs globaux de non linéarité. Ainsi, et comme les variables des systèmes physiques sont toujours bornées, le cas du secteur local de non-linéarité peut être considéré. Dans ce cas de figure, $f(x(t)) \in [a_1, a_2]x(t)$ est valable que pour $x(t) \in [c_1, c_2]$ (Figure 3.3.b), et le modèle est exacte que pour la région considérée.

3.3.4. Propriété d'approximation universelle pour les systèmes flous de type TS

Une propriété importante des systèmes flous TS est leurs capacités à approximer des fonctions lisses non linéaires. Dans les années 90 un grand nombre de chercheurs ont étudié la propriété au cours en montrant que la TSF ((3.1) avec (3.2)) ou ((3.3) avec (3.4)) était capable d'approximer uniformément toute fonction non linéaire continue réelle sur un ensemble compact avec un degré quelconque de précision (Ying, 1998) et (Tikk, Koczy, & Gedeon, 2003) une analyse critique de ces résultats est donnée dans (Tikk, Koczy, & Gedeon, 2003). Récemment, une analyse approfondie de l'approximation a été développée dans (Mendel, 2018).

Selon (Ying, 1998), le lemme suivant définit l'approximateur flou:

Lemme 2.1: Soit la fonction continue $f(x)$ définie comme suit $f: \mathbb{X} \rightarrow \mathbb{R}$ où $\mathbb{X} \subset \mathbb{R}^n$ et un ensemble compact.

Pour chaque $\varepsilon > 0$ arbitraire il existe un système TS flou linéaire, $\hat{f}(x)$, de telle sorte que :

$$\|f(x) - \hat{f}(x)\|_{\infty} < \varepsilon. \tag{3.6}$$

De plus, il a été montré (Ying, 1998) et (Tikk, Koczy, & Gedeon, 2003) que l'erreur est inversement proportionnelle au nombre de règles du système flou et au nombre de données utilisé pour la conception du TS.

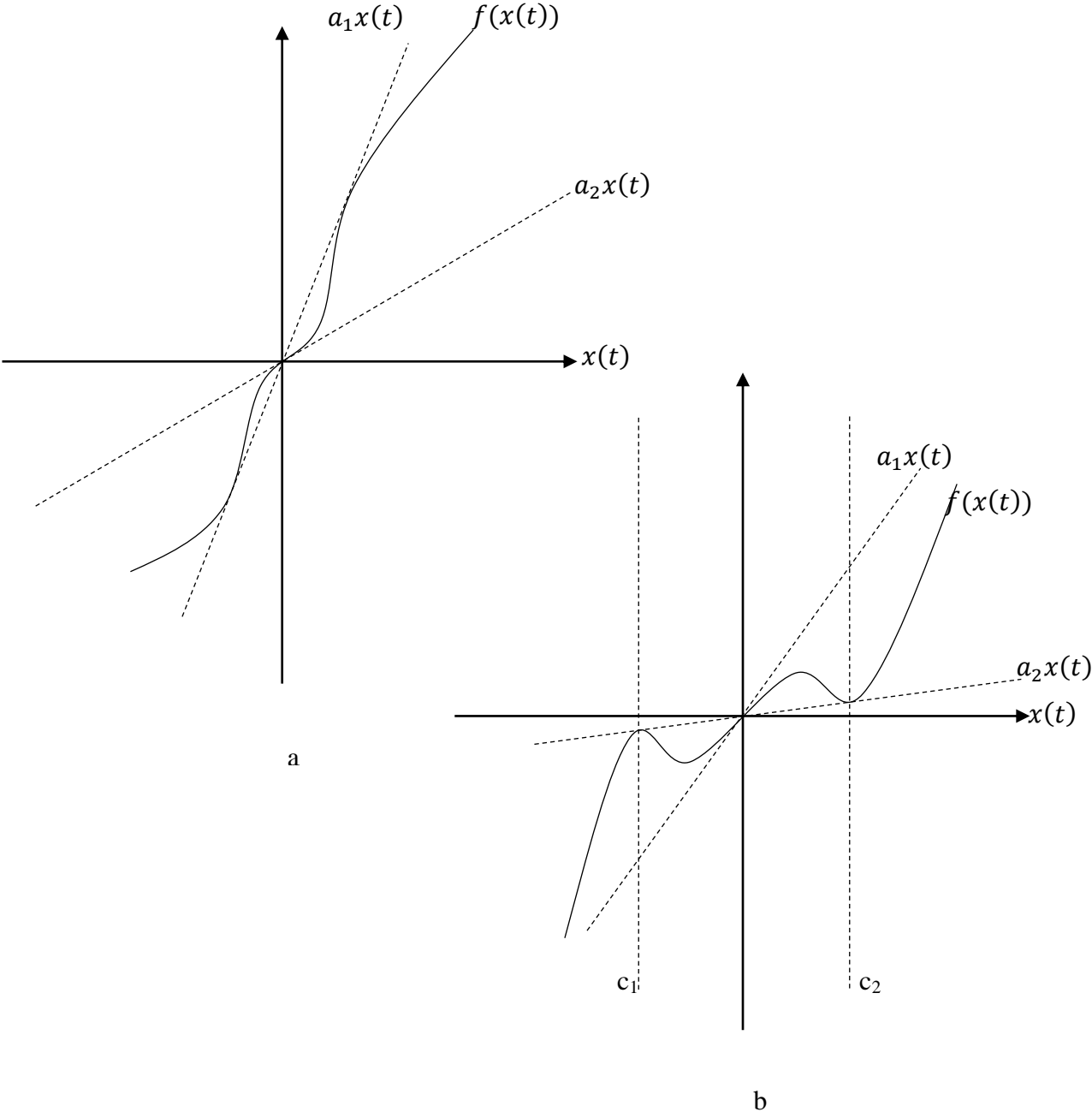


Figure 3.3 Secteur de non linéarité : a. secteur globale, b. secteur local (Tanaka & Wang, 2001)

3.4. Commande prédictive approximante par apprentissage flou

3.4.1. Principe général

La commande prédictive approximante est réalisée généralement par les techniques de courbes d'ajustement (Johansen & Grancharova, 2003), (Bemporad & Filippi, 2003), (Johansen, 2004), (Canale, Fagiano, & Milanese, 2009), (Canale, Fagiano, & Milanese, 2010), (Kvasnica, Lofberg, & Fikar, 2011) et (Bemporad, Oliveri, Poggi, & Storaice, 2011) ou bien par d'autre technique telles que la programmation dynamique dans le cas de (Bakaráč, Holaza, Kalúz, Klaučo, Löfberg, & Kvasnica, 2018).

La commande prédictive approximante par apprentissage se résume à un problème d'approximation de la loi de commande optimale par une fonction approximante où la relation entre la loi de commande et le vecteur d'état se figure explicitement. L'idée générale est de construire une base de paires (x^*, u^*) de solutions optimales calculées hors ligne afin d'estimer la nouvelle loi de commande approximante.

La nouvelle loi approximante est estimée par la minimisation de l'erreur entre la valeur optimale et la valeur issue de la fonction approximante.

$$e(t) = u^*(t) - \hat{u}(t)$$

avec $e(t)$ l'erreur à l'instant t , $u^*(t)$ la loi optimale, $\hat{u}(t)$ la loi de commande approximante fonction de $x^*(t)$.

Ou encore elle est calculée par la résolution du problème d'optimisation suivant :

$$\min \left\{ \sum_{t=1}^{N_e} (u^*(t) - \hat{u}(t))^2 \right\} \quad (3.7.a)$$

$$\text{s. à } \quad \hat{u}(t) = f(x^*(t)) \quad (3.7.b)$$

$$\hat{u}_{j,k} \in E_u \in \mathbb{R}^m \quad (3.7.c)$$

Où $u^*(t)$ est la commande MPC calculée hors ligne, $\hat{u}(t)$ est la commande MPC estimée par la fonction $f(x^*(t))$, N_e nombre total d'échantillons.

Le but est de minimiser l'erreur quadratique entre la loi de commande MPC calculée hors ligne et la loi de commande estimée.

3.4.2. Stratégie de la commande prédictive approximante par apprentissage flou

La synthèse du contrôleur prédictive approximant par apprentissage flou comporte deux étapes, la première est la résolution du problème prédictive hors ligne, tandis que la deuxième est l'estimation de la nouvelle loi approximante utilisant les solutions optimales déjà calculé.

3.4.2. a. Solution MPC

Ces solutions optimales sont déterminées pour un nombre donné L de conditions initiales x^l , $l = 1 \dots L$, uniformément répartis sur une grille G avec $G \subseteq \Omega$, Ω est un sous-ensemble compact inclus dans l'ensemble des solutions possibles et comprenant l'origine. L'ensemble G doit être choisi de telle façon que la distance de Hausdorff $d_H(G, \Omega)$ entre G et Ω satisfasse

$$\lim_{L \rightarrow \infty} d_H(G, \Omega) = 0$$

Notons qu'une grille uniforme sur Ω satisfait cette condition (Canale, Fagiano, & Milanese, 2009).

La région Ω est définie par

$$\underline{x}_i \leq x_i \leq \bar{x}_i \quad i = 1 \dots n \quad (3.8)$$

À partir de chaque point initial de G , le problème MPC est résolu jusqu'à la stabilisation à l'origine.

Les trajectoires optimales obtenues :

$$\bar{x}^{l*} = \{x^{l*}(1) \ x^{l*}(2) \dots x^{l*}(T)\} \quad l = 1 \dots L \quad (3.9.a)$$

et les séquences optimales de contrôles :

$$\bar{u}^{l*} = \{u^{l*}(1) \ u^{l*}(2) \dots u^{l*}(T)\} \quad l = 1 \dots L \quad (3.9.b)$$

sont utilisés dans le problème d'approximation comme cible de l'approximateur flou.

Dans ce qui suit, nous résumons les principales étapes de la procédure de conception de contrôleur flou discutée dans les sections précédentes dans l'Algorithme 3.1.

Algorithme 3.1 Trajectoires optimales $\{\bar{x}^{l*}, \bar{u}^{l*}\}$ calcul hors ligne

Entrées $\mathbb{X}, \mathbb{U}, N, L$

- 1: Sélectionner une grille uniforme $\mathbf{G} \subseteq \Omega$
- 2: **pour** $l = 1 \dots L$
- 3: **répéter** a chaque instant d'échantillonnage $t=1,2,\dots$
- 4: Résoudre $\min_{\mathbf{u}} J_N(\mathbf{x}, \mathbf{k}, \mathbf{u})$ (problème (2.6))
- 5: Sauver $\{\mathbf{x}^{l*}(t), \mathbf{u}^{l*}(t)\}$:
- 7: **jusqu'à** $t=T$
- 8: **fin**

Sortie trajectoires optimales $\{\bar{x}^{l*}, \bar{u}^{l*}\}$

3.4.2. b. Synthèse du contrôleur flou

Introduite au point 3.2.1, la synthèse du contrôleur approximant par apprentissage flou utilise les séquences de solutions optimales (3.9.a) et (3.9.b), par le biais de l'erreur quadratique, afin d'estimer les coefficients du contrôleur flou de type Takagi Sugeno.

Pour cela, la première étape est la fuzzification de la séquence (3.9.a) suivant les fonctions d'appartenances suivantes :

$$\begin{cases} h_j(x(t)) = \frac{\mu_{\tilde{R}_j}(x(t))}{\sum_j \mu_{\tilde{R}_j}(x(t))} \\ \mu_{\tilde{R}_j}(x(t)) = \prod_{j=1}^n \mu_{\tilde{F}_{ij}}(x_j) \end{cases} \quad (3.10)$$

Les poids normalisés satisfont la propriété de somme convexe suivante : $\sum_{j=1}^r h_j(x) = 1$, $0 \leq h_j(x) \leq 1$. avec: x_i $i = 1 \dots n$ sont les prémisses, \tilde{F}_{ij} ensemble flou associé à la variable x_i et la règle \tilde{R}_j , $x(t) \in \mathbb{R}^n$ est le vecteur d'état, $u(t) \in \mathbb{R}^m$ est le vecteur de commande. $K_j \in \mathbb{R}^{n \times m}$ les gains locaux a calculé pour le problème d'approximation.

La loi de commande est donnée par

$$u_f(t) = \sum_j^r h_j(x(t))K_j x(t) \quad (3.11)$$

La fonction objective à minimiser pour obtenir les gains locaux est définie comme suit :

$$\min_{K_i} \sum_l^L \sum_t^T (u^{l*}(t) - u_f^l(t))^2 \quad (3.12)$$

$$\text{où } u_f^l(t) = \sum_j^r K_j h_j(x^{l*}(t))x^{l*}(t)$$

On définit la matrice des vecteurs gains

$$\mathcal{K} = (K_1 \ K_2 \ \dots \ K_r) \quad (3.13)$$

et la matrice de données optimaux pondérées

$$\Phi^{l*}(t) = [h_1(x^{l*}(t)) \ x^{l*}(t) \ h_2(x^{l*}(t)) \ x^{l*}(t) \ \dots \ h_r(x^{l*}(t)) \ x^{l*}(t)]^T \quad (3.14)$$

On aura alors

$$u_f^l(t) = \mathcal{K} \Phi^{l*}(t) \quad (3.15)$$

Définir la séquence associée

$$\bar{u}_f^l = (u_f^l(1) \ u_f^l(2) \ \dots \ u_f^l(T)) \quad (3.16)$$

$$\bar{u}_f^l = (\mathcal{K} \Phi^{l*}(1) \ \mathcal{K} \Phi^{l*}(2) \ \dots \ \mathcal{K} \Phi^{l*}(T)) = \mathcal{K} \bar{\Phi} \quad (3.17)$$

Donc la fonction objective (3.12) devient

$$\min_K \sum_{l=1}^L \|\bar{u}^{l*} - \mathcal{K} \bar{\Phi}\|_2 \quad (3.18)$$

L'approximation floue de la loi MPC optimal doit être réalisable, c'est-à-dire qu'elle doit satisfaire les contraintes (2.4.c) et (2.4.d). Dans le cas où les contraintes (2.4.d) sont des contraintes de saturation, elles sont ajoutées explicitement

$$U_{min} \leq \mathcal{K} \bar{\Phi} \leq U_{max} \quad (3.19)$$

Le problème (3.18) (3.19) est un programme quadratique. La précision de la solution de ce problème dépend du nombre de règles du FAMPC.

La synthèse du contrôleur flou FAMPC peut être résumée dans l'algorithme 3.2.

Algorithme 3.2 Calcul hors ligne des paramètres du contrôleur FAMPC

Entrées $\{\bar{x}^{l*}, \bar{u}^{l*}\}$, paramètres du système d'inférence

1: Construire le système d'inférence FAMPC

2: **Répéter** a chaque itération $i=1,2,\dots$

3: **pour** $l = 1 \dots L$

3: **pour** $t = 1 \dots T$

4: Calculer $\Phi^{l*}(t) = [h_1(x^{l*}(t)) \ x^{l*}(t) \dots h_r(x^{l*}(t)) \ x^{l*}(t)]$

5: **fin**

6: **fin**

7: $\bar{\Phi}^l = (\Phi^{l*}(1) \ \Phi^{l*}(2) \dots \Phi^{l*}(T)), \ l = 1, 2 \dots L$

8: **min** $_{\mathcal{K}} \sum_{l=1}^L (\bar{u}^{l*} - \mathcal{K}\bar{\Phi}^l)^2$ s.à (3.19)

9: **jusqu'à** erreur quadratique moyenne soit satisfaite

Sortie gains optimaux \mathcal{K} de FAMPC

Puisque la procédure de conception est hors ligne, le choix de l'optimiseur pour résoudre le problème (2.4) ($\min_{\mathcal{U}} J_N(x, k, \mathcal{U})$) dans l'algorithme 3.1 est ouvert. Dans ce travail, pour le cas linéaire, ce problème est résolu à l'aide de la boîte à outils MPT (Herceg, Kvasnica, Jones, & Morari, 2013) tandis que pour le cas non linéaire, la boîte à outils d'optimisation de Matlab ou par optimisation par essaim de particule PSO. De plus, dans l'algorithme 3.2, le problème d'optimisation est un programme quadratique. Le système d'inférence de FAMPC est construit avec la structure la plus simple possible recherchant les gains optimaux qui donne une erreur d'approximation raisonnable. Ainsi, la grille et le système d'inférence peuvent être choisis de manière à établir un bon compromis entre précision d'approximation, temps de calcul et occupation mémoire.

Remarque 3.1. Les contraintes (2.4.c) sont satisfaites implicitement. En effet, puisque (3.18) (3.19) est un problème d'apprentissage supervisé, il n'est pas nécessaire d'imposer des contraintes sur les états. En fait, il n'y a pas de génération de nouveaux états dans le processus d'apprentissage les seules valeurs générées sont les valeurs \bar{u}_f^l de l'approximateur flou soumis à la contrainte (3.19).

Remarque 3.2. L'exactitude de la solution du problème susmentionné dépend du nombre de règles du FAMPC. De plus, trouver la solution n'implique pas la stabilité, comme on peut le prouver dans le chapitre suivant.

3.5. Conclusion

Ce chapitre propose la synthèse d'un contrôleur prédictif approximant. Au premier lieu, une bref présentation des outils utilisés pour la synthèse du contrôleur notamment l'apprentissage supervisé et la logique flous en ce qui concerne les systèmes de type Takagi-Sugeno. Effectivement, ces systèmes flous sont populaires pour leurs simplicités, transparences et leurs propriétés d'approximation où ils se classent comme étant des approximateurs universels.

La majeure partie de ce chapitre a été consacré pour présenter l'approche proposée basée sur l'apprentissage flou. Principalement, les approches de la commande prédictive approximante résolvent des problèmes d'ajustement de courbe. Ce type de commande est utilisé pour palier le problème de temps d'exécution de la MPC. A l'image des autres approches la FAMPC résolve un problème d'ajustement de courbe par le biais d'un système flou de type TS. Simple a synthétisé, la FAMPC est aussi valide dans les deux cas que ce soit pour les systèmes linéaires ou non linéaires.

Chapitre 4

Stabilité de la commande prédictive par apprentissage flou

4-1- Introduction :

L'analyse de la stabilité de la FAMPC se résume à l'étude de stabilité du système flou TS qui en résulte de la résolution du problème d'approximation discuté dans le chapitre 3. En effet, l'introduction des modèles flous de Takagi Sugeno (Takagi & Sugeno, 1985), TS, ont permis de traiter le problème de la stabilité de manière formelle. Le fil de la recherche pour la stabilité des systèmes flous TS a été déclenchée par les travaux de Tanaka et Sugeno (Tanaka & Sugeno, 1992) et (Tanaka, Ikeda, & Wang, 1998)). Ces derniers ont proposé des conditions suffisantes de stabilité, fondées sur la stabilité quadratique de Lyapunov, afin d'obtenir la matrice commune P via les inégalités de matrices linéaires, LMI. Toutefois, ces conditions étaient assez conservatrices (Tanaka & Wang, 2001). Depuis lors, les recherches se sont focalisées sur la relaxation de ces conditions pour assouplir le conservatisme.

Fondamentalement, l'analyse de la stabilité et la stabilisation étaient basées sur la théorie de Lyapunov. Dans ce contexte, trois voies principales ont été suivies à la fois pour les systèmes TS continus et discrets : l'approche de la fonction de Lyapunov quadratique (Tanaka, Ikeda, & Wang, 1998), (Tanaka & Wang, 2001), (Kim & Lee, 2000) et (Sala & Arino, 2007)), la fonction de Lyapunov par morceaux (Johansson, Rantzer, & Arzen, 1999), (Feng, 2003) et (Campos, Souza, Torres, & Palhares, 2013)) et la Lyapunov non quadratique et ses extensions (Chadli, Maquin, & Ragot, 2000), (Tanaka, Hori, & Wang, 2001), (Guerra & Vermeiren, 2004) et (Ding, Sun, & Yang, 2006)).

Récemment, l'analyse de la stabilité locale, à la fois pour les cas continus et discrets, a incité une grande attention où l'objectif est d'essayer d'assouplir le conservatisme en fournissant les conditions de la stabilité et de la stabilisation locales et en déterminant le domaine d'attraction ou son estimation (Lee & Joo, 2014) et (Lendek & Lauber, 2016).

Dans ce travail, l'approche de (Belarbi, 2019) est adoptée pour analyser la stabilité du système FAMPC à posteriori. Cette alternative utilisant un résultat pour l'analyse de la stabilité des systèmes linéaires variant dans le temps dû à Bauer (Bauer, Premaratne, & Duran, 1993), où elle donne une condition nécessaire et suffisante pour la stabilité d'un système discret linéaire variant dans le temps, basé sur des normes matricielles plutôt que sur la théorie de Lyapunov. La stabilité est vérifiée en testant les normes des produits de matrices. Effectivement, ce test peut être appliqué pour analyser à la fois la stabilité globale et locale de systèmes TS flous discrets. Un algorithme de stabilisation basé sur la minimisation de la norme et l'LMI sont alors proposés.

Dans ce qui suit, le chapitre présente les notions de base de la stabilité au sens de Lyapunov en particulier pour les systèmes flous de type TS. Ensuite, l'approche proposée pour analyser la stabilité de la boucle fermée est exposée, d'abord pour le cas linéaire, puis pour une classe de systèmes non linéaires.

4-2- Stabilité au sens de Lyapunov pour un system flou de type TS :

L'analyse de la stabilité au sens de Lyapunov des systèmes flous de type Takagi-Sugeno s'effectue principalement en utilisant la deuxième méthode (directe) (Wang, Zeng, & Fu, 2004).

Soit le système discret non linéaire (4.1), réécrit sous forme linéaire, suivant:

$$x_{t+1} = \mathcal{A}(x_t)x_t \quad (4.1)$$

Où :

$$\mathcal{A}(x_t) = \sum_{i=1}^r h_i(x_t) \mathcal{A}_i, \quad (i = 1 \dots r) \quad (4.2)$$

Et

$$\sum_{i=1}^r h_i(x_t) = 1 \quad (4.3)$$

4-2-1 Stabilité quadratique:

On considère la fonction de Lyapunov candidate du système discret (4.1)

$$V(x_t) = x_t^T P(x_t) x_t \quad (4.4)$$

Où $P(x_t)$ matrice définie positive

De (4. 1), (4.2) et (4.4) on peut réécrire $\Delta V(x_t)$, comme suit :

$$\begin{aligned} \Delta V(x_t) &= V(x_{t+1}) - V(x_t) \\ &= x_{t+1}^T P(x_{t+1}) x_{t+1} - x_t^T P(x_t) x_t \\ &= x_t^T \sum_{i=1}^r h_i(x_t) \mathcal{A}_i^T P \sum_{i=1}^r h_i(x_t) \mathcal{A}_i x_t - x_t^T P x_t \end{aligned}$$

Par conséquent,

$$\Delta V(x_t) = x_t^T \left[\sum_{i=1}^r h_i(x_t) (\mathcal{A}_i^T P \mathcal{A}_i - P) \right] x_t$$

Puisque $h_i(x_t) \geq 0$, $\Delta V(x_t) < 0$ si $(\mathcal{A}_i^T P \mathcal{A}_i - P) < 0$ est vraie.

Theorem 4.1 :

Le système discret (4.1) est asymptotiquement stable s'il existe une matrice $P=P^T$ définie positive tel que

$$\mathcal{A}_i^T P \mathcal{A}_i - P < 0 \tag{4.5}$$

Soit vraie pour $i = 1, 2, \dots, r$.

4-2-2 Stabilité non quadratique :

On considère la fonction de Lyapunov candidate du système discret (4.1)

$$V(x_t) = x_t^T P(x_t) x_t, \quad P(x_t) = \sum_{i=1}^r h_i(x_t) P_i \tag{4.6}$$

Où P_i ($i = 1, 2, \dots, r$) des matrices définies positive.

De (4.1), (4.2) et (4.6), on peut réécrire $\Delta V(x_t)$, comme suit :

$$\begin{aligned} \Delta V(x_t) &= V(x_{t+1}) - V(x_t) \\ &= x_{t+1}^T P(x_{t+1}) x_{t+1} - x_t^T P(x_t) x_t \\ &= x_t^T \sum_{i=1}^r h_i(x_t) \mathcal{A}_i^T \left[\sum_{l=1}^r h_l(x_{t+1}) P_l \right] \sum_{j=1}^r h_j(x_t) \mathcal{A}_j x_t - x_t^T \sum_{i=1}^r h_i(x_t) P_i x_t \end{aligned}$$

Où $h_i(x_{k+1}) \geq 0$ et $\sum_{i=1}^r h_i(x_{k+1}) = 1$.

L'équation (4.1) peut s'écrire:

$$P(x_t) = \sum_{i=1}^r h_i(x_t) P_i = \sum_{i=1}^r \sum_{l=1}^r \sum_{j=1}^r h_i(x_t) h_l(x_{t+1}) h_j(x_t) P_i$$

Par conséquent,

$$\Delta V(x_t) = x_t^T \left[\sum_{i=1}^r \sum_{l=1}^r \sum_{j=1}^r h_i(x_t) h_l(x_{t+1}) h_j(x_t) (\mathcal{A}_i^T P_l \mathcal{A}_j - P_i) \right] x_t$$

Notons que

$$\sum_{i=1}^r \sum_{j=1}^r h_i(x_t) h_j(x_t) (\mathcal{A}_i^T P_l \mathcal{A}_j - P_i) \leq \sum_{i=1}^r h_i(x_t) (\mathcal{A}_i^T P_l \mathcal{A}_i - P_i)$$

Donc

$$\Delta V(x_t) \leq x_t^T \left[\sum_{i=1}^r \sum_{l=1}^r h_i(x_t) h_l(x_{t+1}) (\mathcal{A}_i^T P_l \mathcal{A}_i - P_i) \right] x_t$$

Puisque $h_i(x_t) \geq 0$ et $h_l(x_{t+1}) \geq 0$, $\Delta V(x_t) < 0$ si $(\mathcal{A}_i^T P_l \mathcal{A}_i - P_i) < 0$ est vraie.

Theorem 4.2 :

Le système discret (4.1) est asymptotiquement stable s'il existe des matrices $P_i (P_l)$ définies positive tel que

$$\mathcal{A}_i^T P_l \mathcal{A}_i - P_i < 0 \tag{4.7}$$

Soit vraie pour $i, l = 1, 2, \dots, r$.

4-3- Conditions de stabilité pour les systèmes flous de type TS :

La solution du problème d'approximation ci-dessus n'implique pas la stabilité, elle doit être ensuite vérifiée a posteriori une fois que les gains de la FAMPC sont obtenus. Les conditions de stabilité à priori peuvent être restrictives en raison du conservatisme des résultats théoriques de la capacité d'approximation des systèmes flous TS. Effectivement, le théorème des systèmes TS ci-dessus étant une condition suffisante, il est connu pour être assez conservatif. Plusieurs approches basées sur la théorie de stabilité de Lyapunov ont été proposées pour réduire ce conservatisme (Guerra, Kruszewski, & Bernal, 2009). Récemment,

une nouvelle approche pour l'étude de la stabilité des systèmes de TS discrets a été introduite dans (Belarbi, 2019). Elle est basée sur un théorème de stabilité robuste des systèmes discrets variants (Bauer, Premaratne, & Duran, 1993). Il a été montré que la condition de stabilité de (Belarbi, 2019) est moins conservative parce qu'elle est basée sur la décroissance des états sur plusieurs périodes de temps et non plus comme dans le théorème 4.2 ci-dessus sur une seule période

4-3-1 matrices d'intervalle :

Soit le système discret variant dans le temps :

$$x_{t+1} = \mathcal{A} x_t \quad (4.8)$$

la représentation suivante de la matrice intervalle peut être considérée (Bauer, Premaratne, & Duran, 1993), où on définit l'ensemble de matrices intervalles \mathbb{A} :

$$\mathbb{A} = \{ \mathcal{A} \in \mathbb{R}^{n \times n} \mid \mathcal{A} = \sum_{i=1}^r \lambda_i \mathcal{A}_i, \lambda_i \in [0,1], i = 1 \dots r; \mathcal{A}_i \in \mathbb{R}^{n \times n} \} \quad (4.9)$$

4-3-2-Condition suffisante pour la stabilité globale d'un système à matrices d'intervalle :

Le théorème suivant a été établi dans (Belarbi, 2019) concernant la stabilité globale d'un système à matrices d'intervalle.

Théorème 4.3 :

Le système variant dans le temps (4.1) est globalement asymptotiquement stable, ssi il existe un k finie de tel sorte que :

$$\| \mathcal{A}_{i_1} \mathcal{A}_{i_2} \dots \mathcal{A}_{i_l} \| < 1, i_1 = 1 \dots r, i_2 = 1 \dots r, \dots, i_l = 1 \dots r, l = 2 \dots k \quad (4.10)$$

Selon (4.9) les valeurs extrêmes de $\lambda_i, i = 1 \dots r$ sont $\underline{\lambda}_i = 0$ et $\overline{\lambda}_i = 1$, alors l'ensemble des matrices extrêmes est :

$$\mathbb{E} = \{ \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_r \} \quad (4.11)$$

Donc l'ensemble \mathbb{P} de tous les produits des matrices extrêmes est :

$$\mathbb{P} = \{ \mathbb{P}_k(i_1, \dots, i_k) = \mathcal{A}_{i_1} \mathcal{A}_{i_2} \dots \mathcal{A}_{i_k} \mid (i_1, \dots, i_k) \in \{1, \dots, r\}^k \} \quad (4.12)$$

De plus, pour différentes valeurs de k :

$$k = 2, \mathcal{A}_1\mathcal{A}_1, \mathcal{A}_1\mathcal{A}_2, \dots, \mathcal{A}_1\mathcal{A}_r, \mathcal{A}_2\mathcal{A}_1, \mathcal{A}_2\mathcal{A}_2, \dots, \mathcal{A}_2\mathcal{A}_r, \dots, \mathcal{A}_r\mathcal{A}_1, \mathcal{A}_r\mathcal{A}_2, \dots, \mathcal{A}_r\mathcal{A}_r.$$

Pour k=3 on multiplie produit pour k=2 par $\mathcal{A}_i, i = 1 \dots r$:

$$k = 3, \mathcal{A}_1\mathcal{A}_1\mathcal{A}_1, \dots, \mathcal{A}_1\mathcal{A}_r\mathcal{A}_r, \dots, \mathcal{A}_r\mathcal{A}_1\mathcal{A}_1, \dots, \mathcal{A}_r\mathcal{A}_r\mathcal{A}_r.$$

De même pour k=4 on multiplie produit pour k=3 par $\mathcal{A}_i, i = 1 \dots r$:

$$k = 4, \mathcal{A}_1\mathcal{A}_1\mathcal{A}_1\mathcal{A}_1, \dots, \mathcal{A}_1\mathcal{A}_r\mathcal{A}_r\mathcal{A}_r, \dots, \mathcal{A}_r\mathcal{A}_1\mathcal{A}_1\mathcal{A}_1, \dots, \mathcal{A}_r\mathcal{A}_r\mathcal{A}_r\mathcal{A}_r. \text{ Et ainsi de suite.}$$

A partir de là, la condition (4.10) du théorème 4.3 est suffisante mais pas nécessaire. Ceci est évident du fait que s'il y a au moins une matrice, \mathcal{A}_{i_s} , qui n'est pas Schur (n'est pas borné quand $k \rightarrow \infty$) et donc instable. Par conséquent, la condition (4.10) ne peut être satisfaite mais le système peut être stable.

4-3-3- Stabilité locale d'un système flou de type TS :

Dans la section précédente une condition pour la stabilité globale pour un système à matrices d'intervalle a été présentée. Notez cependant que la stabilité globale est difficile à atteindre dans le cas de systèmes flous, même lorsque toutes les matrices sont Schur. En effet, le système flou étant non linéaire, la stabilité globale des systèmes non linéaires est plutôt l'exception. En conséquence, l'application du théorème 4.3 à la stabilité locale est étudiée dans ce qui suit.

Étant donné que la condition (4.10) peut ne pas être concluante dans le cas de la stabilité globale, il serait intéressant de vérifier la stabilité locale d'une région autour d'un point d'équilibre.

On considère le système flou suivant :

$$x_{t+1} = \sum_{i=1}^r h_i(x_t) A_i x_t, \quad i = 1 \dots r \quad (4.13)$$

Où

$$h_i(z(t)) = \mu_{R_i}(z(t)) = \prod_{j=1}^n \mu_{F_{ij}}(z_j(t)) \quad (4.14)$$

Le théorème 4.3 peut être appliqué sur le système flou quand $z \in S_0$, où la région S_0 est définie par $\underline{z}_j < 0, \bar{z}_j > 0$ voir Figure 4.1, pour la détermination de l'ensemble des matrices externes dans cette région.

Selon (4.12), Prenant l'équivalence $\lambda_i \equiv h_i(z(t))$, $h_i(z(t))$ est borné :

$$\min_{z \in S_0} h_i(z(t)) \leq h_i(z(t)) \leq \max_{z \in S_0} h_i(z(t))$$

Ou encore

$$\underline{\alpha}_i \leq h_i(z(t)) \leq \bar{\alpha}_i, i = 1 \dots r, z \in S_0 \quad (4.15)$$

$$\text{Avec } \bar{\alpha}_i = \max_{z \in S_0} h_i(z(t)), \underline{\alpha}_i = \min_{z \in S_0} h_i(z(t))$$

De (4.15) et (4.11), l'ensemble des matrices extrêmes associé à la région S_0 est :

$$\mathbb{E}' = \{(\alpha_1 A_1, \alpha_2 A_2, \dots, \alpha_r A_r), \quad \alpha_i = \bar{\alpha}_i \text{ or } \alpha_i = \underline{\alpha}_i, i = 1 \dots r \},$$

En appliquant le théorème 4.3 sur cet ensemble de matrices extrêmes vérifie la relation (4.9).

Les bornes $\bar{\alpha}_i$ et $\underline{\alpha}_i$ sont calculées à partir de l'équation 4.

$$\bar{\alpha}_i = \max_{z \in S_0} h_i(z(t)) = \max_{z \in S_0} \prod_{j=1}^n \mu_{F_{ij}}(z_j(t))$$

$$\underline{\alpha}_i = \min_{z \in S_0} h_i(z(t)) = \min_{z \in S_0} \prod_{j=1}^n \mu_{F_{ij}}(z_j(t))$$

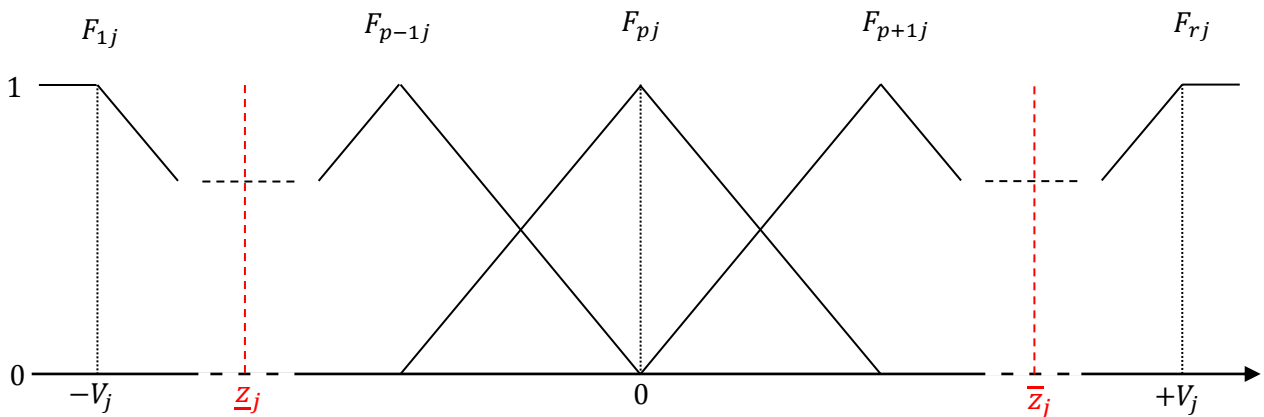


Figure 4.1 Illustration de la région S_0 : fonctions d'appartenances système flou de type TS

Soit la condition nécessaire et suffisante pour la stabilité locale au voisinage de l'origine, donné par le théorème 4.4 (Belarbi, 2019).

Theorem 4.4 :

Le système (4.13) est localement asymptotiquement stable dans une région S_0 au voisinage du point d'équilibre, définie par :

$$-V_j < \underline{z}_j < z < \bar{z}_j < +V_j, \quad \underline{z}_j < 0, \bar{z}_j > 0, j = 1 \dots n \quad (4.16)$$

ssi il existe un k finie de tel sorte que :

$$\|P'_k(i_1, \dots, i_k)\| < 1, \text{ pour tous } P'_k(i_1, \dots, i_k) \in \mathbb{P}'. \quad (4.17)$$

Où l'ensemble des matrices extrêmes :

$$\mathbb{E}' = \{(\alpha_1 A_1, \alpha_2 A_2, \dots, \alpha_r A_r), \quad \alpha_i = \bar{\alpha}_i \text{ or } \alpha_i = \underline{\alpha}_i, i = 1 \dots r \},$$

Avec $0 < \alpha_i < 1, 0 < \alpha_i < 1, i = 1 \dots r$

$$\begin{aligned} \bar{\alpha}_i &= \max_{z \in S_0} h_i(z(t)) = \max_{z \in S_0} \prod_{j=1}^n \mu_{F_{ij}}(z_j(t)) \\ \underline{\alpha}_i &= \min_{z \in S_0} h_i(z(t)) = \min_{z \in S_0} \prod_{j=1}^n \mu_{F_{ij}}(z_j(t)) \end{aligned} \quad (4.18)$$

4-4- Stabilité du contrôleur FAMPC en boucle fermée:

Dans ce qui suit, la stabilité de la boucle fermée est analysée en utilisant l'approche présentée dans la section 4.3, d'abord pour le cas linéaire, puis pour une classe de systèmes non linéaires.

4.4.1. Stabilité cas linéaire

Afin d'analyser la stabilité de la boucle fermée, la loi de commande du FAMPC (3.16) est insérée dans (2.5) pour obtenir l'équation en boucle fermée :

$$x_{t+1} = Ax_t + B \sum_{i=1}^r h_i(x) K_i x_t \quad (4.19)$$

comme $\sum_{i=1}^r h_i(x) = 1$, (4.19) peut s'écrire

$$x_{t+1} = \sum_{i=1}^r h_i(x)(A + BK_i)x_t \quad (4.20)$$

ou encore

$$x_{t+1} = \sum_{i=1}^r h_i(x)(\mathcal{A}_i x_t) \quad (4.21)$$

Qui a la même forme que (4.13), alors le théorème suivant donne la condition de stabilité de (4.21).

Théorème 4.5:

La stabilité locale du système discret AFMPC en boucle fermée (4.21) est localement asymptotiquement stable dans une région d'approximation Ω autour de l'origine, définie par :

$$\underline{x}_i \leq x_i \leq \bar{x}_i \quad i = 1 \dots$$

si et seulement s'il existe un k fini tel que:

$$\|P'_k(i_1, \dots, i_k)\| < 1 \quad \text{pour tout } P'_k(i_1, \dots, i_k) \in \mathbb{P}'_k, \quad (4.22)$$

où \mathbb{P}'_k est défini comme dans (4.12) avec l'ensemble des matrices extrêmes:

$$\mathbb{A}' = \left\{ (\alpha_1 \mathcal{A}_1, \dots, \alpha_i \mathcal{A}_i, \dots, \alpha_r \mathcal{A}_r), \right. \\ \left. \alpha_i = \bar{\alpha}_i \text{ or } \alpha_i = \underline{\alpha}_i, i = 1 \dots r \right\} \quad (4.23)$$

avec $0 < \bar{\alpha}_i < 1, \quad 0 < \underline{\alpha}_i < 1, i = 1 \dots r$

$$\bar{\alpha}_i = \max_{x \in \Omega} h_i(x_t) = \max_{x \in \Omega} \prod_{i=1}^n \mu_{F_{ij}}(x_i(t)) \quad (4.24a)$$

$$\underline{\alpha}_i = \min_{x \in \Omega} h_i(x_t) = \min_{x \in \Omega} \prod_{i=1}^n \mu_{F_{ij}}(x_i(t)) \quad (4.24b)$$

Preuve. La preuve est simple, il suffit d'appliquer le théorème 4.4 à (4.21).

4.4.2. Stabilité cas non linéaire

Nous considérons le système non linéaire affine en la commande de la forme:

$$x(k+1) = F(x(k))x(k) + G(x(k))u(k) \quad (4.25)$$

Ces systèmes peuvent être représentés exactement par un système flou TS utilisant l'approche dite de secteur de non-linéarité décrite dans (Tanaka, Hori, & Wang, 2001).

Le modèle de système flou TS (4.25) est

$$x(t + 1) = \sum_{i=1}^{r_1} h_{1_i}(x(t))(A_i x(t) + B_i u(t)) \quad (4.26)$$

En insérant, (3.16), l'équation en boucle fermée est la suivante :

$$x(t + 1) = \sum_{i=1}^{r_1} h_{1_i}(x(t)) \left(A_i x(t) - B_i \sum_{j=1}^{r_2} h_{2_j}(x(t)) K_j x(k) \right)$$

qui peut être écrit comme :

$$x(t + 1) = \sum_{i=1}^{r_1} h_{1_i}(x(t)) \left(\sum_{j=1}^{r_2} h_{2_j}(x(t)) A_i x(t) - \sum_{j=1}^{r_2} h_{2_j}(x(t)) B_i K_j x(k) \right) \quad (4.27)$$

Ou encore

$$x(t + 1) = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} h_{1_i}(x(t)) h_{2_j}(x(t)) (A_i - B_i K_j) x(t) \quad (4.28)$$

$$x(t + 1) = h_{11} h_{21} \mathcal{A}_{11} + h_{11} h_{22} \mathcal{A}_{12} + \dots + h_{11} h_{2r_2} \mathcal{A}_{1r_2} + h_{12} h_{21} \mathcal{A}_{21} + \dots + h_{12} h_{2r_2} \mathcal{A}_{2r_2} + h_{1r_1} h_{2r_2} \mathcal{A}_{r_1 r_2} + \dots + h_{1r_1} h_{2r_2} \mathcal{A}_{r_1 r_2} \quad (4.29)$$

avec $\mathcal{A}_{ij} = A_i - B_i K_j, i = 1 \dots r_1, j = 1, \dots, r_2$

On a : $\bar{h}_l(x) = h_{1_i}(x(t)) h_{2_j}(x(t)),$

Et $\bar{h}_l(x) \in [0,1], l = 1 \dots r_1 \times r_2$

donc (4.34) devient

$$x(t + 1) = \sum_{l=1}^{r_1 \times r_2} \bar{h}_l(x) \mathcal{A}_l x(t) \quad (4.30)$$

Où elle a la même forme que dans (4.13). La condition de stabilité de (4.30) est donnée par le théorème page suivante.

Théoreme 4.6:

Stabilité locale de la boucle fermée sous contrôle FAMPC : Le système (4.30) est localement asymptotiquement stable dans la région d'approximation Ω autour de l'origine, définie par :

$$\underline{x}_i \leq x_i \leq \bar{x}_i \quad i = 1 \dots r$$

si et seulement s'il existe un k fini tel que :

$$\|P'_k(i_1, \dots, i_k)\| < 1 \text{ pour tout } P'_k(i_1, \dots, i_k) \in \mathbb{P}'_k \quad (4.31)$$

où \mathbb{P}'_k est défini comme dans (4.19) avec l'ensemble des matrices extrêmes:

$$\mathcal{A}'^E = \{(\alpha_1 \mathcal{A}_{11}, \alpha_2 \mathcal{A}_{12} \dots, \alpha_l \mathcal{A}_{r_1 r_2}), \alpha_l = \bar{\alpha}_l \text{ or } \alpha_l = \underline{\alpha}_l, l = 1 \dots r_1 \times r_2\} \quad (4.32)$$

$$\text{et avec :} \quad \bar{\alpha}_l = \max_{x \in \Omega} \bar{h}_l(x) \quad (4.33a)$$

$$\underline{\alpha}_l = \min_{x \in \Omega} \bar{h}_l(x) \quad (4.33b)$$

Preuve: la preuve est obtenue en appliquant le théorème 4.4 à (4.30)

4.4.3 Algorithme de test de stabilité du Contrôleur en boucle fermée :

Soit la condition suffisante de stabilité pour concevoir un contrôleur FAMPC:

Corollaire 4.1 :

La boucle fermée est localement asymptotiquement stable dans la région d'approximation Ω autour de l'origine, définie par :

$$\underline{x}_j \leq x_j \leq \bar{x}_j \quad j = 1 \dots r$$

si les gains K_i satisfont la condition suivante:

$$\|\mathcal{A}_i\| = \|A(x_t) + K_i B(x_t)\| < 1, i = 1 \dots r \quad (4.34)$$

Algorithme de test de stabilité :

La procédure de conception est résumée comme suit:

Algorithme 4.1 Test de stabilité

Entrées $\mathbb{X}, \mathbb{U}, \mathbb{N}, L$

1: Construire l'ensemble des matrices extrêmes: $\mathbb{A} = \left\{ \begin{array}{l} \mathcal{A}'_i = \alpha_i \mathcal{A}_i \\ \alpha_i = \underline{\alpha}_i \text{ or } \alpha_i = \bar{\alpha}_i, i = 1 \dots r \end{array} \right\}$

2: **Répéter**

2: **si** $\| \mathcal{A}'_{i_1} \times \mathcal{A}'_{i_1} \times \dots \times \mathcal{A}'_{i_k} \| < 1$ (i_1, \dots, i_k) $\in \{1, \dots, r\}^k$

3: **Sortie** système stable

4: fin test

5: **sinon**

6: **Sortie** pas de conclusion encore

7: $k=k+1$

8: **jusqu'à** la valeur max de k

4-5- Conclusion

Ce chapitre traite la stabilité de l'approche FAMPC proposée. Cela commence par une présentation générale sur la stabilité, en particulier dans le sens de Lyapunov, puis son application aux systèmes flous TS. La conservation de ces systèmes rend la méthode non concluante. Pour résoudre le problème, un test de stabilité, simple à mettre en œuvre est proposé, dans lequel la structure du système TS est exploitée pour tester sa stabilité à posteriori.

Chapitre 5

Simulations et résultats

5.1. Introduction

Le contrôleur FAMPC décrit précédemment au chapitre 3 est testé ici sur plusieurs exemples linéaires et non linéaire. La MPT MATLAB Multi Parametric Toolbox (Herceg, Kvasnica, Jones, & Morari, 2013) a été utilisé pour obtenir la solution explicite aux problèmes de MPC linéaire. Tous les calculs ont été effectués à l'aide de MATLAB fonctionnant sur un processeur Pentium Dual-Core 2.3 GHz, avec 2 Go de RAM.

Quatre exemples sont présentés, trois problèmes linéaires tirés de la littérature et un problème non linéaire. Le premier exemple est le double intégrateur qui possède deux variables d'états et une entrée de commande (Canale, Fagiano, & Milanese, 2009), le deuxième exemple, tiré de (Bemporad, Oliveri, Poggi, & Storace, 2011) est un système LTI multi entrées multi sorties et le troisième (Holaza, Takacs, & Kvasnica, 2013) est un multi entrées multi sorties avec quatre variable d'état et deux entrée de commande. L'exemple 4 est un système de pendule inversé avec deux états et une entrée de commande.

La résolution du problème d'optimisation non linéaire est faite utilisant l'algorithme PSO ((Kennedy & Eberhart, 1995), (Boumaza & Belarbi, 2013)) (voir annexe B).

Les étapes suivantes résument la procédure de synthèse du contrôleur FAMPC :

Etape 1

Paramètre FAMPC : nombre de règles, nombre des ensembles flous pour les variables d'états et l'univers de discours.

Sélectionner la grille uniforme G

Etape 2

Résoudre le problème (2.4) pour toutes les conditions initiales de la grille et construire l'ensemble des trajectoires optimales $\{\bar{x}^{l*}, \bar{u}^{l*}\} l = 1 \dots L$. (Algorithme 3.1)

Etape 3

Résoudre le problème (3.18) s.à (3.19) pour obtenir les gains \mathcal{K} du contrôleur (Algorithme 3.2)

Etape 4

Exécuter le test de stabilité (Algorithme 4.1)

Si le test de stabilité est satisfait arrêter le test

Sinon aller à l'étape 1

Remarque 5.1. Dans tous les exemples, nous avons recherché la FAMPC qui donne une erreur d'approximation raisonnable avec la structure la plus simple possible.

5.2. Exemple 5.1 : le double intégrateur

On considère le système double intégrateur (Canale, Fagiano, & Milanese, 2009) suivant :

$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(t) \\ y(t) = [1 \quad 0]x(t) \end{cases}$$

$$\text{Sujet à } \begin{bmatrix} -1 \\ -1 \end{bmatrix} < x_k < \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ and } -1 < u_k < 1$$

$$\text{avec } Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = 1, N = 5$$

La solution explicite de la commande prédictive possède 5 régions.

La région d'approximation Ω est :

$$-1 \leq x_1 \leq +0.5 \quad -0.5 \leq x_2 \leq +1$$

Les points de la grille sont équidistants de 0.12 , ce qui donne $L = 384$. Une fois la solution explicite obtenue, des simulations ont été effectuées pour tous les points de la grille. Chaque simulation a durée 10 échantillons.

Le nombre total de points de montage est de 3840. Notez que, selon le principe d'optimalité, tous les points appartiennent aux trajectoires optimales.

Le FAMPC a les deux variables d'état comme entrées avec trois ensembles flous, comme le montre la figure 3.2, ce qui donne 9 règles. De même, l'univers du discours est celui de la figure 1. Avec $V = 1.2$. L'erreur quadratique moyenne de l'approximation est $6 \cdot 10^{-5}$. Les gains locaux $K_i, i = 1 \dots 9$ sont donnés dans le tableau 1. La stabilité locale du FAMPC TS a été vérifiée à l'aide du théorème 1 dans la région Ω avec 9 matrices extrêmes (4.24). Le test était concluant pour $k = 6$ avec la norme maximale est 0.9390953. La validation du contrôleur flou a ensuite été effectuée pour certains points initiaux n'appartenant pas aux trajectoires de la grille G . Les figures 5.3 (a) et 5.3 (b) montrent le signal de commande et les états pour le MPC et le FAMPC. . Comme on peut le constater, il existe un ajustement quasi parfait entre

le MPC d'origine et son approximation. Le temps nécessaire pour exécuter le FAMPC dans MATLAB était de 3 μ s. Notez que l'implémentation FPGA des systèmes flous est bien documentée et peut être exécutée en quelques nanosecondes ((Kim D. , 2000) , (Holaza, Takacs, & Kvasnica, 2013), (Sun, Tang, Meng, Zhao, & Yang, 2015)).

La figure 5.2 montre le résultat de la simulation, où l'on peut observer que les performances ne sont pas dégradées. Pour des perturbations plus importantes, les performances sont toujours maintenues mais les contraintes sur le signal de commande sont violées.

Table 5.1 Exemple 5.1: Les gains du contrôleur

Matrices Gain	
K_1	[-0.6994 -0.2820]
K_2	[-0.4309 -1.0816]
K_3	[-0.4884 -1.0752]
K_4	[-0.7635 -0.9866]
K_5	[-0.4907 -1.0783]
K_6	[-0.3374 -0.9565]
K_7	[-0.8412 -1.4287]
K_8	[-0.4328 -1.4739]
K_9	[-0.3037 -0.6576]

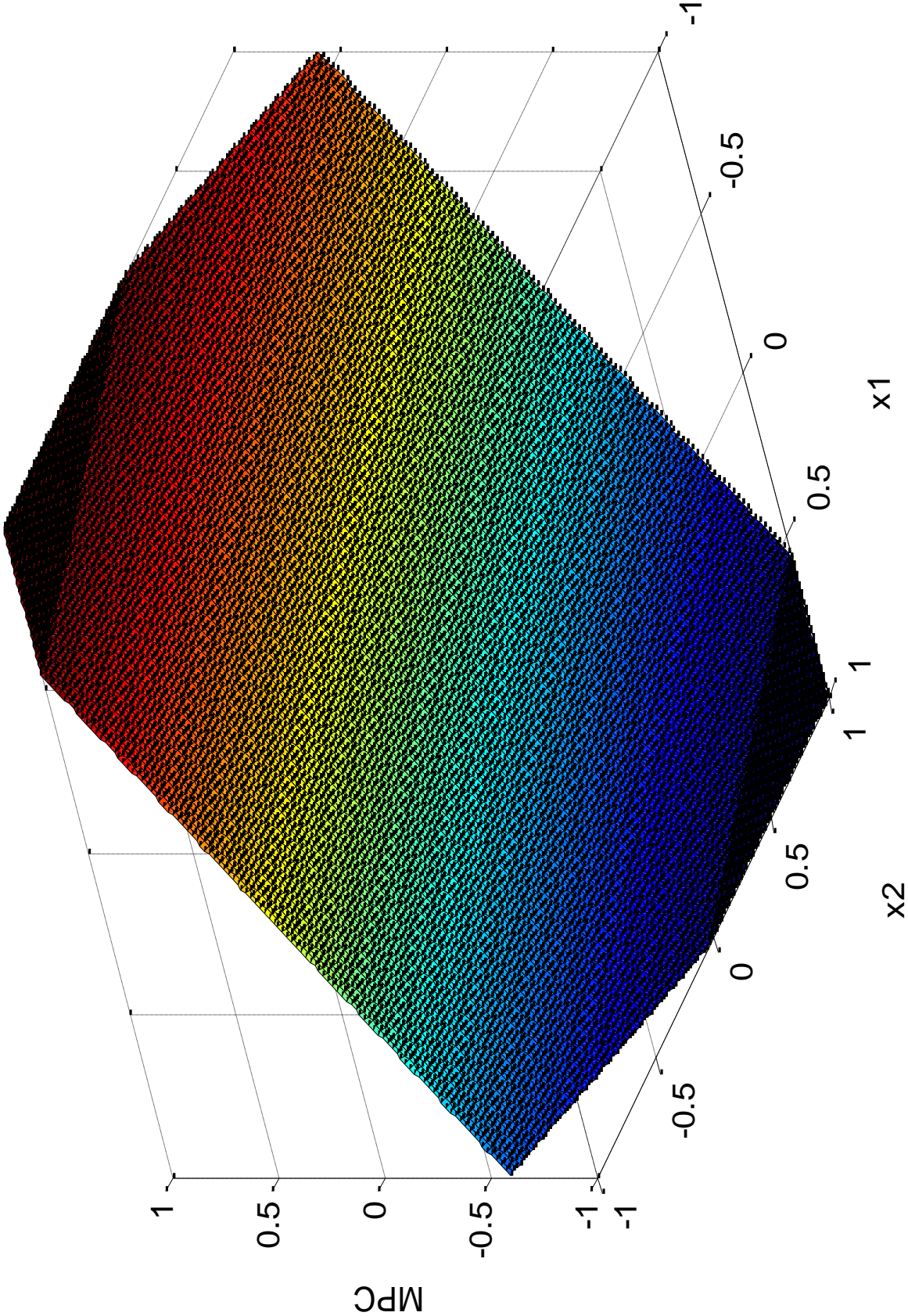


Figure 5.1 Exemple 5.1 Surface de commande explicite MPC

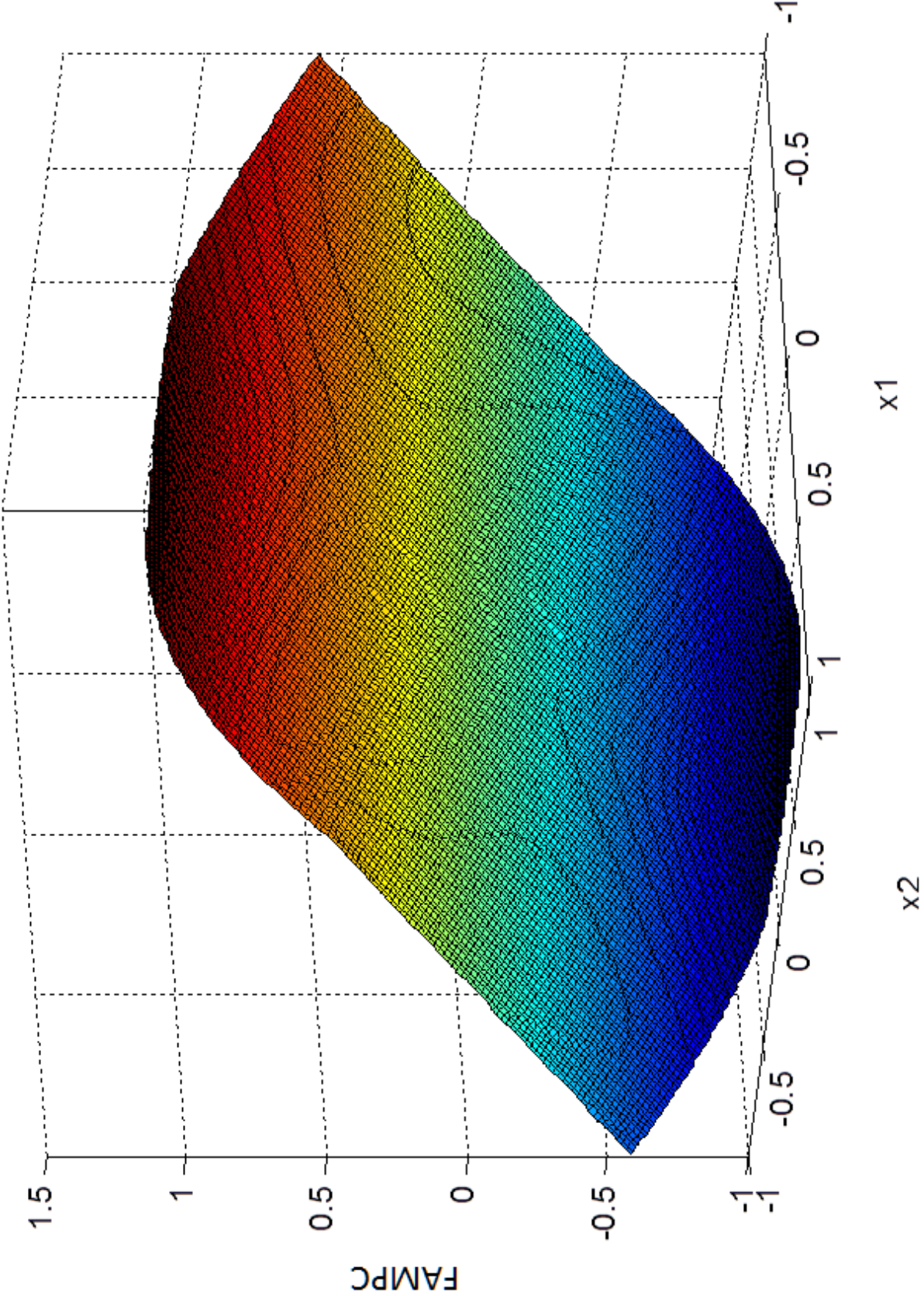
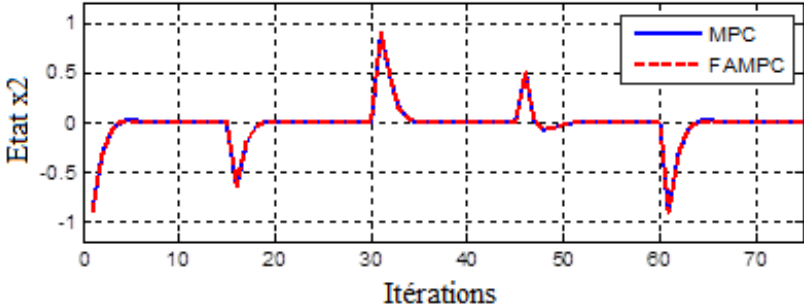
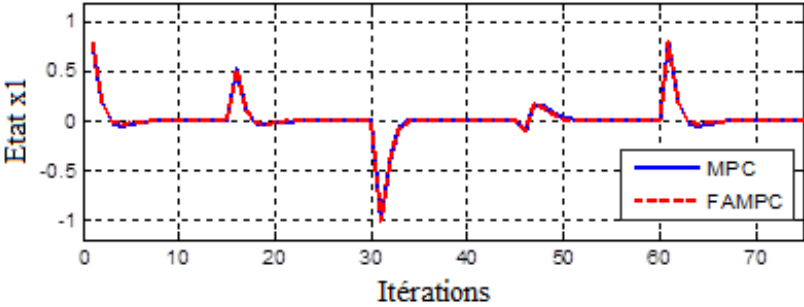
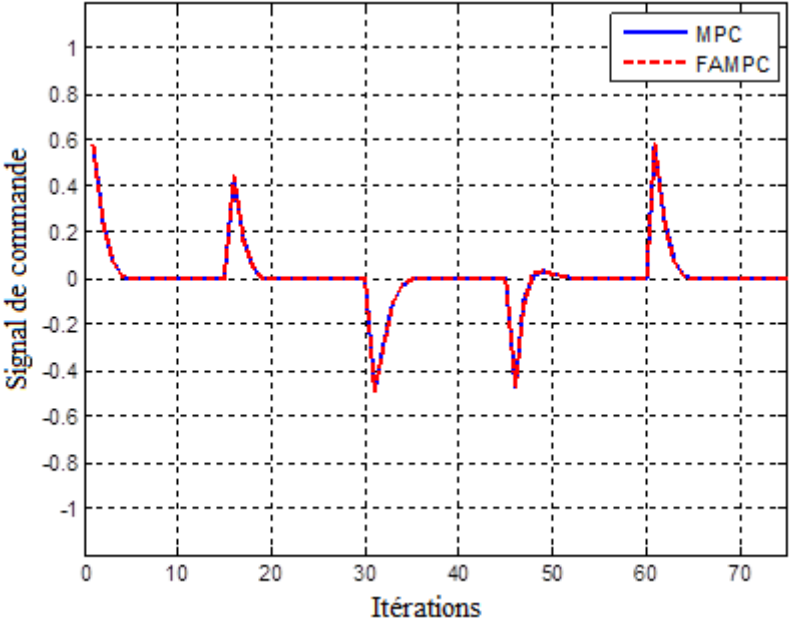


Figure 5.2 Exemple 5.1 Surface commande prédictive approximante FAMPC



(a)



(b)

Figure 5.3 Exemple 5.1 (a) les états (b) la commande, MPC et FAMPC

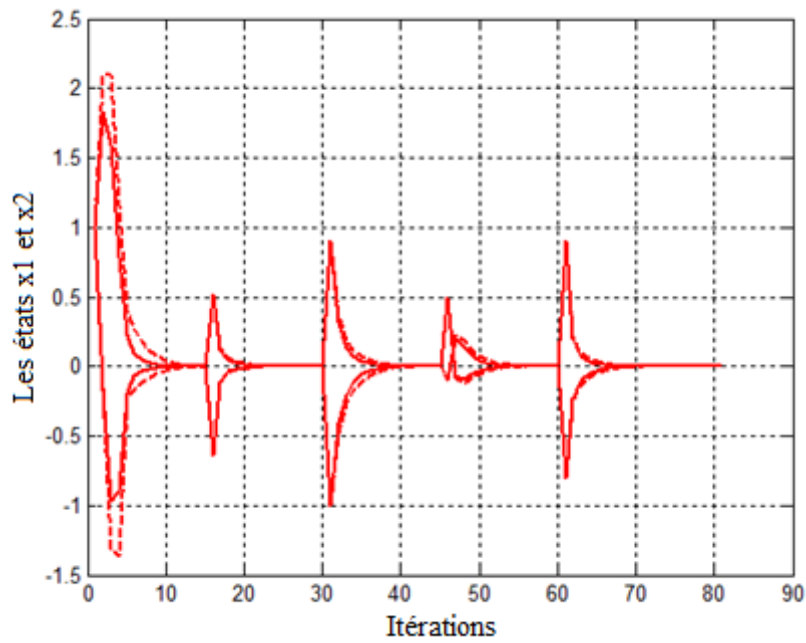


Figure 5.4 Exemple 5.1 test de robustesse FAMPC

Tableau 5.2 Exemple 5.2 : Les gains du contrôleur.

Matrices Gain	
K_1	$\begin{bmatrix} -0.422 & -0.208 \\ -0.193 & -0.308 \end{bmatrix}$
K_2	$\begin{bmatrix} 0.091 & -0.665 \\ -0.211 & 0.204 \end{bmatrix}$
K_3	$\begin{bmatrix} 0.021 & -0.170 \\ -0.533 & -0.800 \end{bmatrix}$
K_4	$\begin{bmatrix} -0.085 & -0.176 \\ 0.255 & -0.237 \end{bmatrix}$
K_5	$\begin{bmatrix} 0.860 & -0.435 \\ -0.962 & -0.895 \end{bmatrix}$
K_6	$\begin{bmatrix} -0.640 & -0.445 \\ 0.274 & -0.284 \end{bmatrix}$
K_7	$\begin{bmatrix} -0.052 & -0.001 \\ -0.735 & -1.446 \end{bmatrix}$
K_8	$\begin{bmatrix} -0.036 & -0.603 \\ -0.208 & 0.357 \end{bmatrix}$
K_9	$\begin{bmatrix} -0.130 & -0.250 \\ -0.355 & -0.219 \end{bmatrix}$

5.3. Exemple 5.2:

Le deuxième exemple à étudier est tiré de (Bemporad, Oliveri, Poggi, & Storaice, 2011). C'est un modèle linéaire avec les matrices suivantes:

$$\begin{cases} x(t+1) = \begin{bmatrix} 1.2 & 1 \\ 0 & 1.1 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} x(t) \end{cases}$$

avec les contraintes dures sur la commande :

$$-0.5 \leq u_1(t) \leq 0.5 \quad -0.6 \leq u_2(t) \leq 0.6$$

et les contraintes souples sur les sorties :

$$-2 - \varepsilon \leq y_{1,2}(t) \leq 2 + \varepsilon$$

avec les paramètres suivants pour résoudre le problème MPC :

$$Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ et } N=5$$

La solution explicite obtenue à l'aide de la toolbox MPT est donnée par 52 régions. La région d'approximation Ω est la suivante:

$$-1.3 \leq x_1 \leq +1.3 \quad -1.3 \leq x_2 \leq +1.3$$

Les points de la grille sont équidistants de 0,15, ce qui donne $L = 289$. Le FAMPC flou a la même structure que dans l'exemple précédent avec $V = 2.2$. L'approximation MSE est 0.0011. Les gains obtenus sont donnés dans le tableau 2. Le test de stabilité local était concluant avec les 9 matrices extrêmes (4.28) pour $k = 5$ et la norme maximale est 0,992488. La figure 5.5 illustre quelques simulations avec diverses conditions initiales.

Certaines réponses très proches coïncidentes presque sans aucun autre terme additif dans la loi de contrôle, comme dans (Bemporad, Oliveri, Poggi, & Storaice, 2011). Le temps pour exécuter le FAMPC est 14 μ s. Comme le montre la figure 5.4, la surface FAMPC présente une bonne interpolation de la surface MPC d'origine. Notez que la mise en œuvre FPGA des systèmes flous est bien documentée et peut s'exercer en quelques nanoseconde ((Kim D. , An implementation of fuzzy logic controller on the reconfigurable FPGA system, 2000), (Holaza, Takacs, & Kvasnica, 2013), (Sun, Tang, Meng, Zhao, Yang, & Yunhu, A scalable accuracy fuzzy logic controller on FPGA , 2015)).

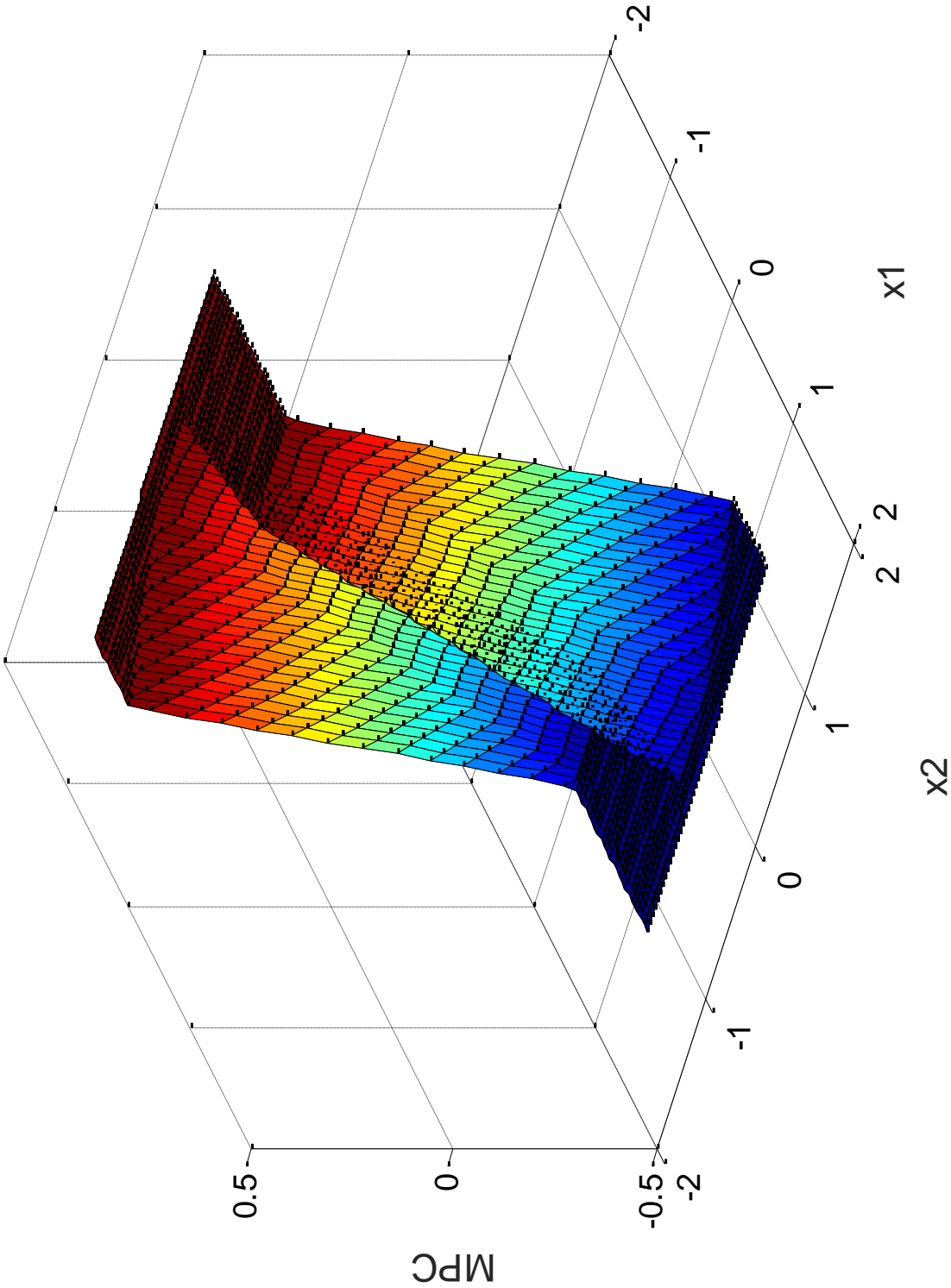


Figure 5.5 Exemple 5.2 Surface de commande explicite MPC

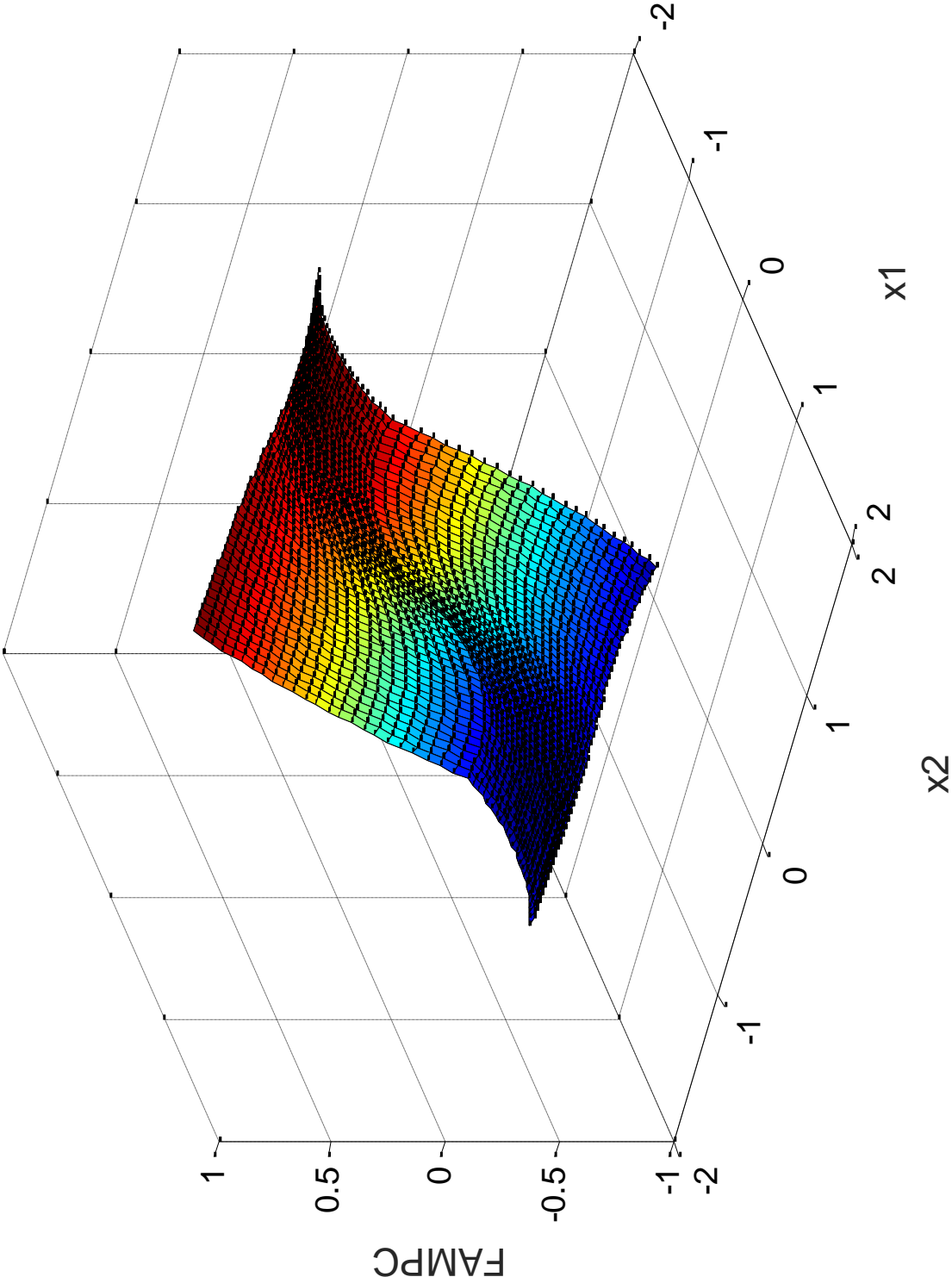
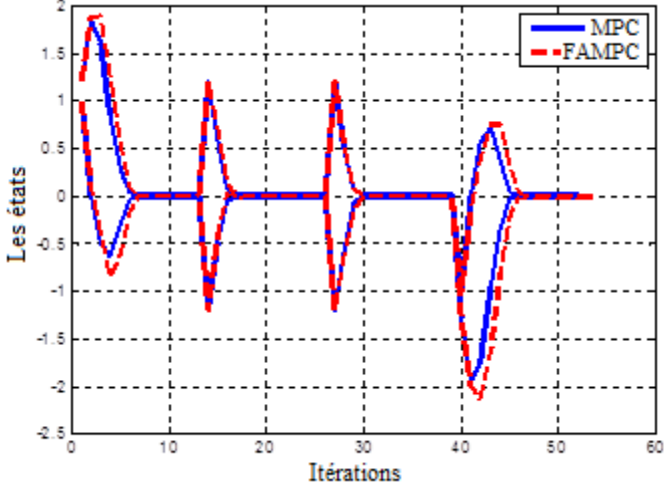
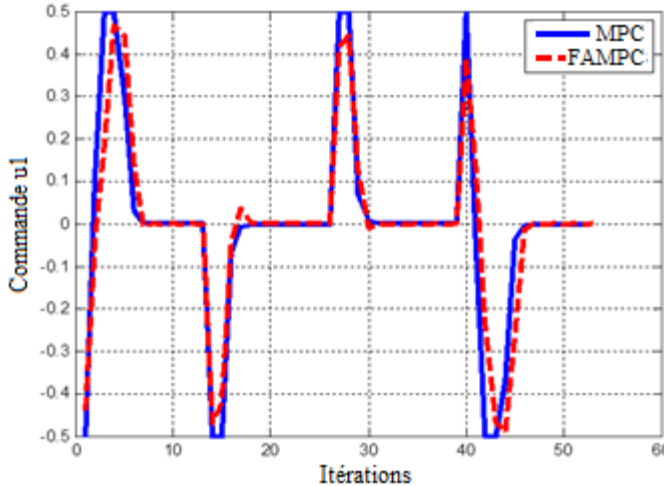


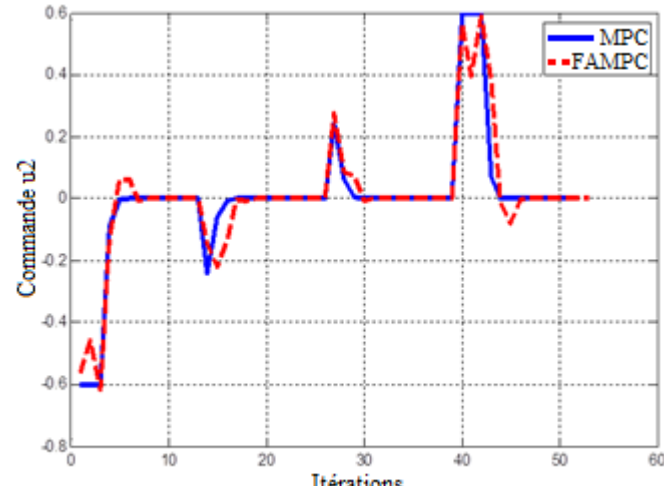
Figure 5.6 Exemple 5.2 Surface de commande prédictive approximante FAMPC



(a)



(b)



(c)

Figure 5.7 Exemple 5.2 (a) les états, (b) et (c) les commandes, MPC et FAMPC

5.4. Exemple 5.3 :

Soit le système a 4 états et 2 entrées, (Holaza, Takacs, & Kvasnica, 2013):

$$\begin{cases} x(t+1) = \begin{bmatrix} 0.7 & -0.1 & 0 & 0 \\ 0.2 & -0.5 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0 \\ -0.5 & 0 & 0.5 & 0.5 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0.1 \\ 0.1 & 1 \\ 0.1 & 0 \\ 0 & 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(t) \end{cases}$$

Sujet.à

$$\begin{bmatrix} -5 \\ -5 \\ -5 \\ -5 \end{bmatrix} < x(t) < \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}, \begin{bmatrix} -2 \\ -2 \end{bmatrix} < y(t) < \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix} < u(t) < \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Le problème prédictif est résolu avec les paramètres suivants:

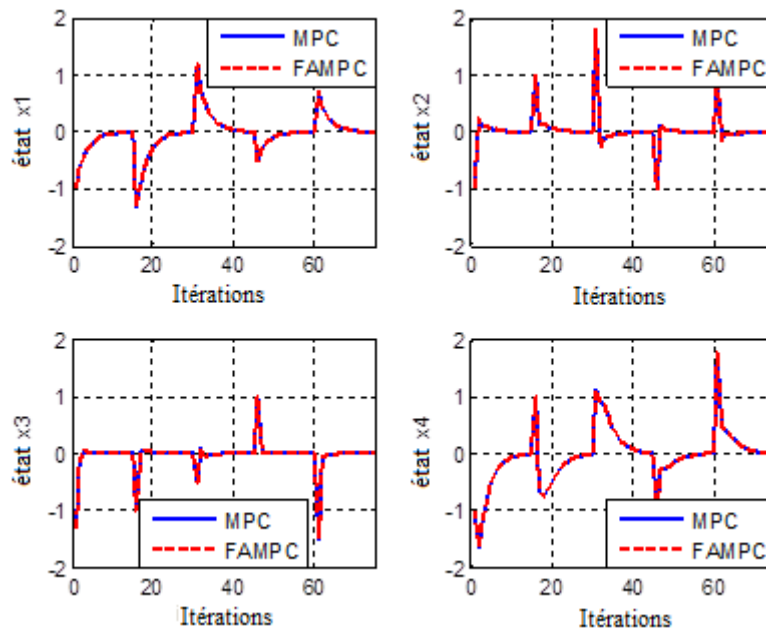
$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R = 0.1 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, N = 15$$

La solution explicite, obtenue en utilisant la toolbox MPT, possède 235 régions. La région d'approximation est définie sur:

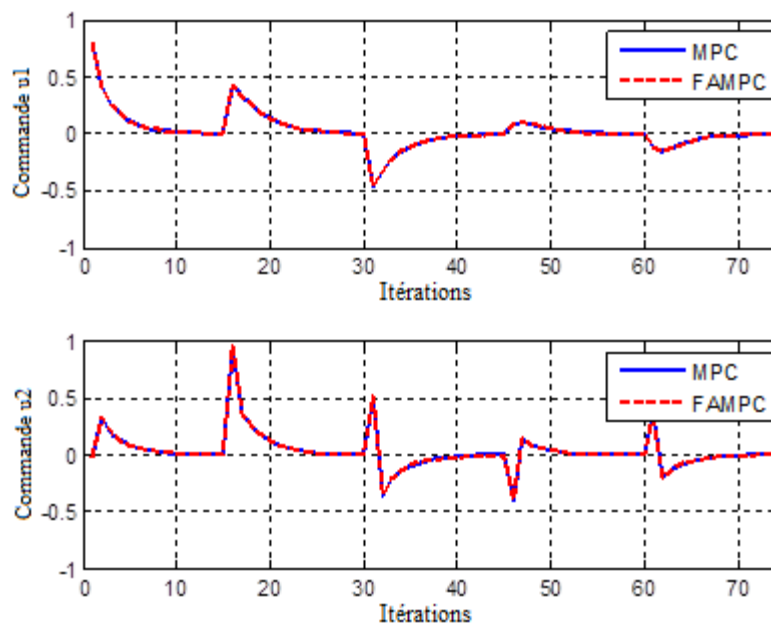
$$-2 \leq x_1 \leq +2, -2 \leq x_2 \leq +2, -1.5 \leq x_3 \leq +1.5 \text{ and } -1 \leq x_4 \leq +1$$

où les deux ensembles flous donnés par la fig.1(b) avec $V=2.2$, ce qui donne 16 règles. L'erreur d'approximation est de 6×10^{-4} .

Les gains obtenus sont testés pour la stabilité en boucle fermée avec les 16 matrices extrêmes (4.24). Bien que la grille soit très lâche, le test de stabilité a été concluant pour $k = 4$ et la norme maximale est 0,9623439. La figure 3 présente les résultats des simulations pour les conditions initiales autres que celles utilisées pour l'optimisation. Certaines réponses sont très proches. Le temps de calcul nécessaire pour obtenir la solution en ligne est de 27 μ s. La figure 5.6 montre les résultats du test de robustesse, où seuls x_1 et x_4 sont représentés, les autres états sont pratiquement inchangés. Comme dans l'exemple précédent, les contraintes sont violées pour des perturbations plus importantes.

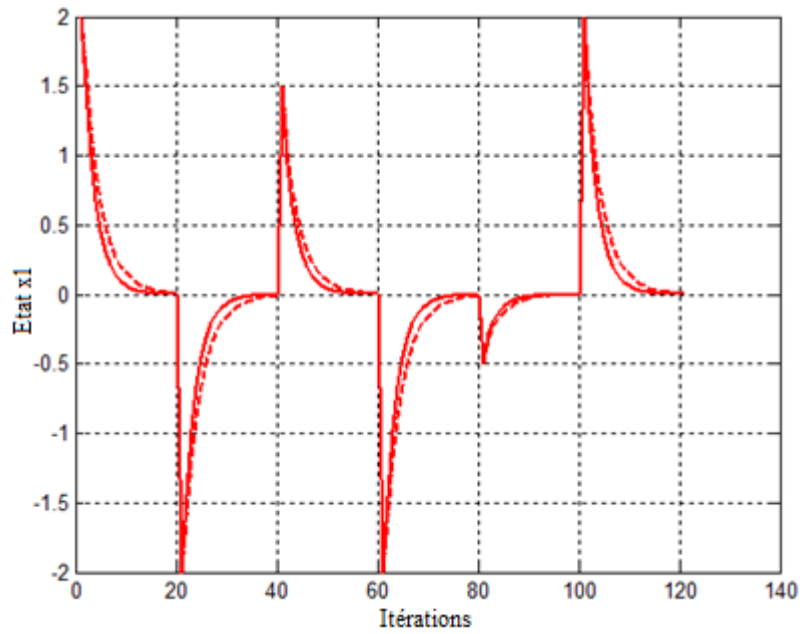


(a)

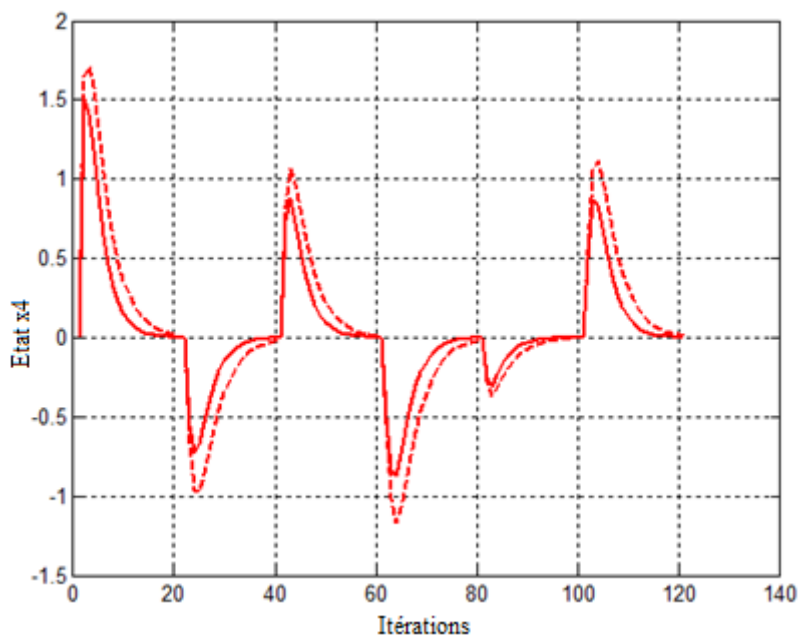


(b)

Figure 5.8 Exemple 5.3 (a) les états, (b) les commandes, MPC et FAMPC



(a)



(b)

Figure 5.9: Exemple 5.3 test de robustesse FAMPC (a) x_1 , (b): x_4 , Nominale (ligne solide), Perturbé (ligne pointillée)

5.5. Exemple 5.4:

Considérant le modèle du pendule inversé suivant (Tanaka & Wang, 2001) :

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = \\ \frac{g \sin(x_1(t)) - \frac{amlx_2^2(t) \sin(2x_1(t))}{2} - a \cos(x_1(t)) u(t)}{\frac{4l}{3} - aml \cos^2(x_1(t))} \end{cases}$$

$$-200 \leq u(t) \leq 200$$

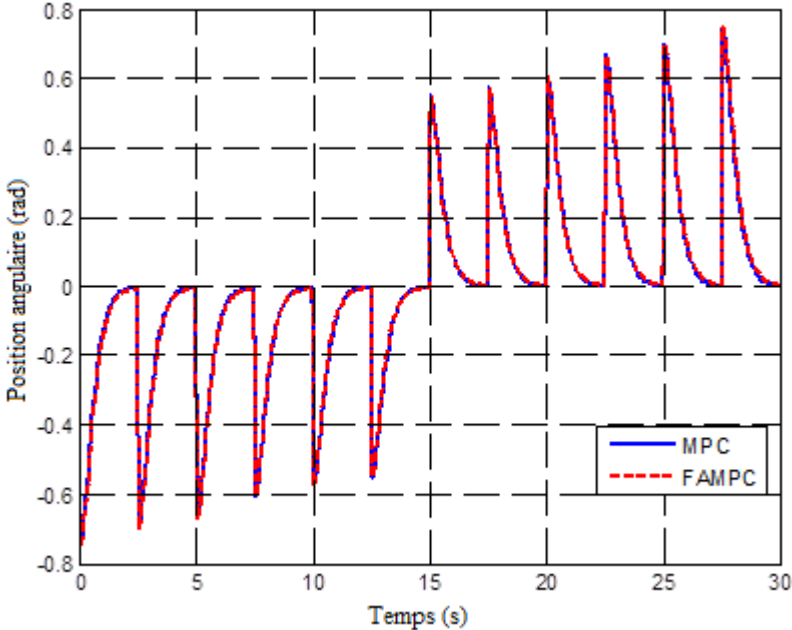
Où $x_1(t)$ est l'angle du pendule (en radians) par rapport à la verticale et $x_2(t)$ est la vitesse angulaire, $g=9.8\text{m/s}^2$ la constante gravité, m est la masse du pendule, M est la masse du chariot, $2l$ est la longueur du pendule, u est la force appliquée au chariot (en newton), $a = 1/(m + M)$. Le modèle non linéaire peut être exactement représenté par un modèle TS flou avec 16 règles données dans (Tanaka & Wang, 2004). Dans les simulations, le modèle discret a été implémenté en utilisant la méthode d'Euler avec les paramètres suivants $m = 2 \text{ kg}$, $M = 8 \text{ kg}$ and $2l = 1 \text{ m}$ (Tanaka & Wang, 2004).

La solution online du problème MPC non linéaire est déterminée avec les paramètres suivants :

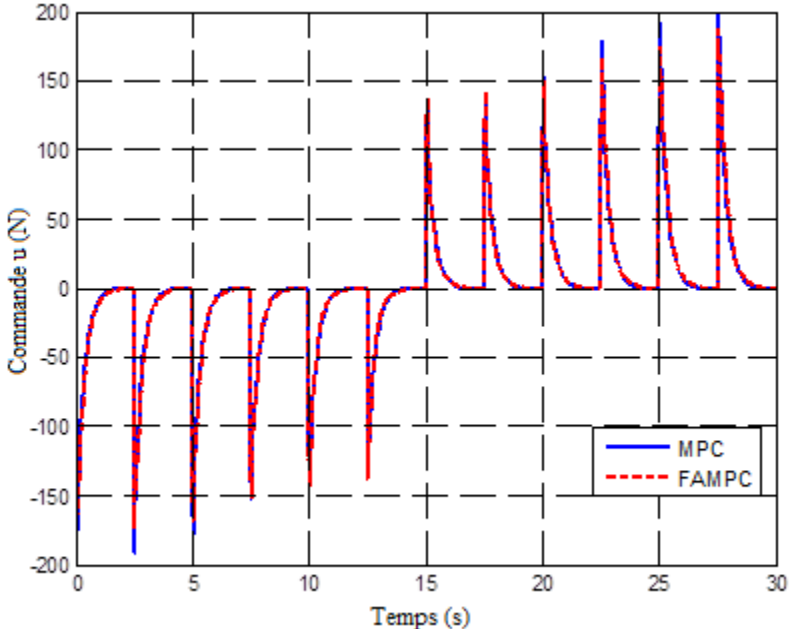
$$N = 5, Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, R = 0.1 \text{ et un temps d'échantillonnage } \Delta T = 0.05\text{s}$$

Tableau 5.3 Exemple 5.4: Gains contrôleur.

Matrices Gain	
K_1	[102.83 -73.34]
K_2	[383.34 161.13]
K_3	[449.10 226.76]
K_4	[37.51 -8.47]



(a)



(b)

Figure.5.10 Exemple 5.4 (a) position angulaire et (b) la force, MPC et FAMPC

La région d'approximation Ω est définie sur:

$$-0.65\text{rad} \leq x_1 \leq 0.65\text{rad}.$$

Les points initiaux ont été choisis avec un incrément de 0.0524 afin d'obtenir 25 points. Le temps moyen nécessaire pour obtenir la solution en ligne était de 0,5 seconde. La FAMPC a les deux variables d'état en tant qu'entrées avec deux ensembles flous pour chacune (Fig. 1 (b)) avec $V = 1$. Les gains obtenus sont donnés dans le tableau 5.3. La stabilité fermée du FAMPC a été vérifiée à l'aide du théorème 4.3 avec des matrices extrêmes données par (4.31) et (4.32). Le test était concluant pour $k = 4$ et une norme maximale de 0,991.

Les figures 4 (a) et (b) montrent la comparaison entre le MPC en ligne et le FAMPC pour un certain nombre de points initiaux autres que ceux utilisés pour l'approximation. Les lignes rouges continues indiquent les réponses avec le contrôleur approximant flou. Les lignes en pointillé bleu indiquent celles du contrôleur MPC optimal. Le FAMPC montre une bonne approximation des propriétés avec un temps de calcul moyen de $2\mu\text{s}$. La figure 7 montre le test de robustesse avec la masse du pôle fixée au double de la valeur nominale. Le temps de décantation est légèrement augmenté en raison de la masse plus lourde.

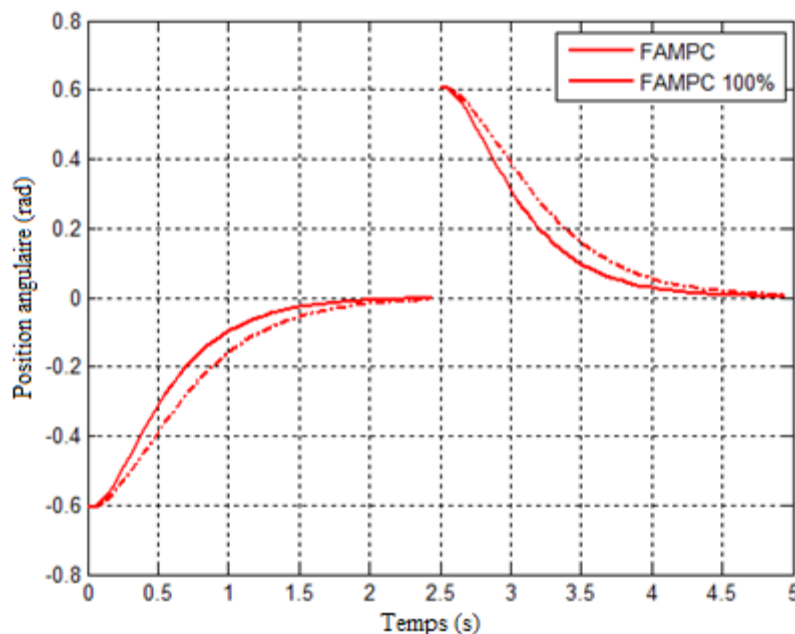


Figure 5.11: Exemple 5.4 Position angulaire: test de robustesse de FAMPC , Nominale (ligne solide), Perturbé (ligne pointillée)

5.6. Conclusion :

Pour mettre en évidence notre stratégie de commande proposée et donner plus de visibilité parmi les techniques existantes dans la littérature, une comparaison des performances est donnée dans le tableau 5.4. Ainsi, nous avons choisi d'effectuer une comparaison avec les principaux travaux dans le domaine de la commande prédictive approchée qui ont étudié la stabilité.

Les résultats obtenus montrent les bonnes performances de l'approche proposée. En effet, toutes les figures montrent la correspondance quasi parfaite entre les signaux de la MPC optimale et de la FAMPC (MPC approximé), et qui se traduit aussi par de faible erreur comprise entre $[5.18 \cdot 10^{-5}, 6 \cdot 10^{-4}]$ dans les exemples étudiés. On voit aussi que cette approche nous permet d'accélérer le processus de commande plus de 1000 fois en temps CPU.

Les résultats du test de robustesse montrent que les performances ne sont pas dégradées. Pour des perturbations plus importantes, les performances sont toujours maintenues mais les contraintes sur le signal de commande sont violées.

Tableau 5.4 Comparaison de performance

		Stratégies de commandes					Notre Approche	Observations	
Technique utilisée		(Canale, Fagiano, & Milanese, 2009) (2010)	(Kvasnica, Lofberg, & Fikar, 2011)	(Bemporad, Oliveri, Poggi, & Storage, 2011)	(Pin, Filippo, Pellegrino, Fenu, & Parisini, 2013)	(Bakaráč, Holaza, Kalúz, Klaučo, Löfberg, & Kvasnica, 2018)	(Paulson & Mesbah, 2020)		
		Set membership (UBB Unknown But Bounded)	Utilise un polynôme pour approximer la loi optimale	un PWAS approximation de l'espace d'état par des petit simplexes	Des fonctions d'approximation neuronales générales	Programmation dynamique approximante	Apprentissage approfondie par réseau de neurone	Approximation de la loi optimale par apprentissage flou	Notre approche est valable pour les deux cas linéaire et non linéaire
Simulation		Double intégrateur (Linéaire), Oscillateur deux dimensions (Nonlinéaire)	Exemples Numériques (Linéaire)	Exemples Numériques (Linéaire)	Oscillateur non amorti (Nonlinéaire)	Pendule inversé (Expérimental) (Linéaire)	Double intégrateur (Linéaire)	Exemples Numériques (Linéaire), Pendule inversé (Nonlinéaire)	
Temps de calcul	Approximation Offline	De l'ordre de Millisecondes du temps CPU pour obtenir un pas de commande	De l'ordre de Millisecondes du temps CPU pour obtenir un pas de commande	Toutes l'approximation a durée [14s,485s]	Non mentionné	Toutes l'approximation a durée 82s	De l'ordre de Millisecondes du temps CPU	De l'ordre de Millisecondes à quelques secondes du temps CPU pour obtenir un pas de commande	Notre approche accélère le processus de commande plus de 1000 fois
	Online	Dizaines de Microsecondes du temps CPU	Real time	In order of Nanoseconds	$5 \cdot 10^{-5}$	De l'ordre de Millisecondes	De l'ordre de Millisecondes	Microsecondes du temps CPU	
Erreur d'approximation		$< 10^{-2}$	Not mentioned	< 0.2	$2.75 \cdot 10^{-4}$	Not mentioned	$[4.4 \cdot 10^{-4}, 386 \cdot 10^{-4}]$	$[5.18 \cdot 10^{-5}, 6 \cdot 10^{-4}]$	
Robustesse		Non considérée	Non considérée	Non considérée	Considérée	Vérifiée pour rejet de perturbation	Considérée	Vérifiée par simulation	
Stabilité		BIBO	À priori , théorie des tubes de stabilité	A posteriori par des fonctions de lyapunov linéaires	Input to state stability (ISS)	Commande par une loi de rétroaction stabilisante par construction	Input to state stability (ISS)	A posteriori condition suffisante basée sur les matrices intervalles	

Conclusion générale

Cette thèse s'est intéressée à la commande prédictive approximante sous contraintes. Celle-ci, permet de traiter le problème du temps d'exécution de la commande prédictive à base de modèle dans sa forme classique. Différents travaux de recherche ont traité cette problématique, essentiellement celles basées sur la commande prédictive approximante utilisant les techniques d'ajustement de courbe. Un état de l'art de ces travaux est présenté au deuxième chapitre.

L'investigation du contrôleur approximant ce fait en deux étapes une en ligne et l'autre hors ligne. La première consiste à résoudre le problème MPC pour trouver la solution optimale, pour cela différentes méthodes peuvent être utilisées telle que la toolbox MPT, le méta heuristique PSO et la SDRE utilisées dans notre travail. Ensuite, dans la deuxième étape ces solutions optimales sont exploitées afin de synthétiser un contrôleur approximant qui a le même comportement du contrôleur original.

Dans ce travail nous avons profité de la propriété d'approximation universelle des systèmes flous de Takagi-Sugeno pour construire une approximation floue de la solution optimale de contrôle prédictif à base de modèle linéaire et non linéaire MPC. Les systèmes non linéaires considérés ici sont affines dans la loi de commande.

L'approche proposée basée sur l'apprentissage supervisé par la résolution d'un problème d'ajustement de courbe par le biais d'un système flou de type TS. L'approximateur flou du MPC introduit dans ce travail présente quelques caractéristiques intéressantes qui ne sont généralement pas montrées par les approximateurs précédents. Le plus important de ceux-ci est sa transparence, dans le sens où il est constitué d'un ensemble de lois de contrôle à rétroaction d'états linéaires locaux (en fonction des états du système) qui sont ensuite fusionnées pour obtenir la loi de contrôle non linéaire finale. Le problème d'approximation est réduit à la recherche des gains en retour locaux. La structure de l'approximateur obtenu est très similaire à la solution explicite du MPC linéaire. La stabilité locale des approximateurs obtenus dans la région d'approximation est analysée a posteriori à l'aide d'un simple test existant basé sur le produit de normes matricielles. Cette procédure peut être appliquée à

l'approximation de MPC linéaires et non linéaires. Les approximations de linéaire et de non linéaire sont ensuite étudiées dans le même cadre.

Les exemples traités montrent qu'une très bonne précision peut être obtenue en utilisant quelques règles; cela confirme le caractère conservateur des résultats théoriques sur la capacité d'approximation du système flou TS.

Enfin, il convient de noter que la mise en œuvre du système flou sur FPGA est bien documentée et est devenue une routine avec un temps de calcul de l'ordre de quelques dizaines de nanosecondes.

Perspective :

A partir de ce travail de recherche, nous avons plusieurs axes à parcourir dans l'avenir afin d'améliorer cette approche et bien cerner la problématique traitée ici, parmi lesquelles nous citons :

- L'étude approfondie de la technique proposée pour de grands systèmes (nombre des états et contraintes grands).
- L'utilisation d'autres techniques d'apprentissage automatique pour synthétiser le contrôleur approximant.
- L'extension de l'approche proposée au contrôle dit prédictif hybride est à l'étude.

Bibliographie

- Alessio, A., & Bemporad, A. (2009). A Survey on Explicit Model Predictive Control. Dans L. Magni, D. Raimondo, & F. Allgöwer, *Nonlinear Model Predictive Control* (pp. 345-369). Berlin, Heidelberg: Springer.
- Bakarác, P., Holaza, J., Kalúz, M., Klaučo, M., Löfberg, J., & Kvasnica, M. (2018). Explicit MPC based on Approximate Dynamic Programming. *European Control Conference (ECC)* (pp. 1172-1177). Limassol: IEEE.
- Bauer, P., Premaratne, K., & Duran, J. A. (1993). Necessary and Sufficient Condition for Robust Systems Asymptotic Stability of Time-Variant Discrete . *IEEE Transactions on automatic control* , 1427-1430.
- Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). *Nonlinear Programming: Theory and Algorithms, 3rd Edition*. Hoboken, New Jersey, USA: John Wiley & Sons, Inc.
- Belarbi, K. (2019). On Matrix Norms, Stability and Stabilization of a Class of Discrete Takagi Sugeno Fuzzy Systems. *IEEE Transactions on Fuzzy Systems* , 1999-2008.
- Bemporad, A. (2011). *Explicit Model Predictive Control*. Bertinoro, Forlì-Cesena, Italy: Scuola Nazionale di Dottorato SIDRA.
- Bemporad, A., & Filippi, C. (2001). Suboptimal explicit MPC via approximate multiparametric quadratic programming . *Proceedings of the 40th IEEE Conference on Decision and Control* (pp. 4851-4856). Orlando, FL, USA: IEEE.
- Bemporad, A., & Filippi, C. (2003). Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications*, 9-38.
- Bemporad, A., Morari, M., Dua, V., & Pistikopoulos, E. (2002). The explicit linear quadratic regulator for constrained systems . *Automatica* , 3-20.
- Bemporad, A., Oliveri, A., Poggi, T., & Storace, M. (2011). Ultra-Fast Stabilizing Model Predictive Control via Canonical Piecewise Affine Approximations . *IEEE transactions on automatic control*, 2883-2897.
- Boumaza, H., & Belarbi, K. (2013). A Fuzzy Approximator for Model based Predictive Control. *14th international conference on Sciences and Techniques of Automatic control & computer engineering* (pp. 20-24). Sousse, Tunisia: STA.
- Campos, V. C., Souza, F. O., Torres, L. A., & Palhares, R. M. (2013). “New stability conditions based on piecewise fuzzy Lyapunov functions and tensor product transformations . *IEEE Transactions on Fuzzy Systems*, 748-760.
- Canale, M., Fagiano, L., & Milanese, M. (2009). Set membership approximation theory for fast implementation of model predictive control. *Automatica* , 45-54.
-

- Canale, M., Fagiano, L., & Milanese, M. (2010). Efficient Model Predictive Control for Nonlinear Systems via Function Approximation Techniques . *IEEE Transactions on Automatic Control* , 1911-1916.
- Chadli, M., Maquin, D., & Ragot, J. (2000). Relaxed stability conditions for Takagi-Sugeno fuzzy systems . *IEEE International conference on Systems Man and Cybern, SMC* (pp. 3514-3519). Nashville, USA: IEEE.
- Chen, H., & Allgower, F. A. (1998). Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability . *Automatica* , 1205-1217.
- Chen, Y., Bruschetta, M., Picotti, E., & Beghi, A. (2019). Matmpc-a matlab based toolbox for real-time nonlinear model predictive control. in: *18th European Control Conference (ECC)* (pp. 3365-3370). Naples, Italy: IEEE.
- Chien, M., & Kuh, E. (1977). Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision . *IEEE Trans. Circuits Syst*, 305-317.
- Chmielewski, D., & Manousiouthakis, V. (1996). On constrained infinite-horizon linear quadratic optimal control. *Systems Control Letters* , 121–129.
- Cimen, T. (2010). Systematic and effective design of nonlinear feedback controllers via the state-dependent Riccati equation (SDRE) method . *Annual Reviews in Control* , 32-51.
- Clerc, M., & Kennedy, J. (2002). The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Trans. on evolutionary computation* , 58 - 73.
- Cloutier, J. (1997). State dependent Riccati Equation techniques An overview . *Proceedings of the American Control Conference* (pp. 932-936). Albuquerque, NM, USA : IEEE.
- Di Cairano, S. i., Yanakiev, D., Bemporad, A., Kolmanovsky, I., & Hrovat, D. (2008). An MPC design flow for automotive control and applications to idle speed regulation . *Proc. 47th IEEE Conf. Decision Control* (pp. 5686-5691). Cancun, Mexico: IEEE.
- Ding, B. C., Sun, H. X., & Yang, P. (2006). Further studies on LMI based relaxed stabilization conditions for nonlinear systems in Takagi–Sugeno’s form . *Automatica*, 503-508.
- Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43). Nagoya, Japan: IEEE.
- Erdem, E. (2001). *Analysis and real time implementation of State Dependent Riccati Equation controlled systems, Ph.D, thesis* . Urbana-Champaign, IL, United States: University of Illinois.
- Feng, G. (2003). Controller synthesis of fuzzy dynamic systems based on piecewise Lyapunov functions. *IEEE Transactions on Fuzzy Systems*, 605-612.

- Goebel, G., & Allgower, F. (2013). “Obtaining and employing state dependent parametrizations of prespecified complexity in constrained mpc . *52nd IEEE Conference on Decision and Control* (pp. 7077-7082). Florence, Italy: IEEE.
- Guerra, T. M., Kruszewski, A., & Bernal, M. (2009). Control law proposition for the stabilization of discrete Takagi–Sugeno models. *IEEE Transactions on fuzzy systems*, 724-731.
- Guerra, T., & Vermeiren, L. (2004). LMI-based relaxed nonquadratic stabilization conditions for nonlinear systems in Takagi–Sugeno’s form . *Automatica*, 823-829.
- H, Y. (1998). Sufficient conditions on uniform approximation of multivariate functions by general TS fuzzy systems with linear rule consequents . *IEEE Transactions on SMC*, 28-44.
- Herceg, M., Kvasnica, M., Jones, C. N., & Morari, M. (2013). Multi-parametric toolbox 3.0 . *European control conference(ECC)* (pp. 502-510). Zurich, Switzerland: IEEE.
- Holaza, J., Takacs, B., & Kvasnica, M. (2013). Synthesis of simple explicit MPC optimizers by function approximation . *International conference on Process Control (PC)* (pp. 377-382). Strbske Pleso, Slovakia: IEEE.
- Houska, B., Ferreau, H. J., & Diehl, M. (2011 a). Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 298-312.
- Houska, B., Ferreau, H. J., & Diehl, M. (2011 b). An auto-generated real-time iteration algorithm for nonlinear mpc in the mi-crosecond range. *Automatica*, 2279-2285.
- Hovd, M., Scibilia, F., Maciejowski, J., & Olaru, S. (2009). Verifying stability of approximate explicit MPC. *Proceedings of the 48th Conference on Decision and Control (CDC)28th Chinese Control Conference* (pp. 6345-6350). Shanghai : IEEE.
- Jacquet, M., Corsini, G., Bicego, D., & Franchi, A. (2020). Perception-constrained and motor-level nonlinear mpc for both underactuated and tilted-propeller uavs. *in: International Conference on Robotics and Automation (ICRA)* (pp. 4301-4306). Paris, France: IEEE.
- Jerez, J., Goulart, P., Richter, S., Constantinides, G., Kerrigan, E., & Morari, M. (2013). Embedded Predictive Control on an FPGA using the Fast Gradient Method . *European Control Conference (ECC)* (pp. 3614-3620). Zurich : IEEE.
- Johansen, T. (2004). Approximate explicit receding horizon control of constrained nonlinear systems . *Automatica*, 293–300.
- Johansen, T., & Grancharova, A. (2003). Approximate explicit constrained linear model predictive control via orthogonal search tree . *Transactions on Automatic Control*, IEEE .
- Johansson, M., Rantzer, A., & Arzen, M. (1999). Piecewise quadratic stability of fuzzy systems . *IEEE Transactions Fuzzy Systems*, 713-722.

- Kalmari, J., Backman, J., & Visala, A. (2015). A toolkit for nonlinear model predictive control using gradient projection and codegeneration, *Control Engineering Practice* 39 (2015) 56–66. *Control Engineering Practice*, 56-66.
- Kennedy, J., & Eberhart, R. (1995). "Particle swarm optimization . *Proceedings of ICNN'95 - International Conference on Neural Networks* (pp. 1942-1948). Perth, WA, Australia: IEEE.
- Kim, D. (2000). An implementation of fuzzy logic controller on the reconfigurable FPGA system. *IEEE Transactions on Industrial Electronics*, 703-715.
- Kim, D. (2000). An implementation of fuzzy logic controller on the reconfigurable FPGA system. *IEEE Transactions on Industrial Electronics*, 703-715.
- Kim, E., & Lee, H. (2000). New Approaches to Relaxed Quadratic Stability Condition of Fuzzy Control Systems . *IEEE Transactions on Fuzzy Systems*, 523-534.
- Kögel, M., & Findeisen, R. (2011). Fast predictive control of linear systems combining Nesterov's gradient method and the method of multipliers . *50th IEEE Conference on Decision and Control and European Control Conference* (pp. 501-506). Orlando, FL: IEEE.
- Kuhn, H. (1960). Some combinatorial lemmas in topology . *IBM Journal of Research and Development*, 518-524.
- Kuhn, H. (1968). Simplicial approximation of fixed points . *Proceedings of the National Academy of Sciences of the USA*, 1238-1242.
- Kvasnica, M., Grieder, P., Baoti'c, M., & Morari, M. (2004). Multi-parametric toolbox (mpt). *in: International workshop on hybrid systems: Computation and control* (pp. 448–462). Philadelphia, PA, USA: Springer.
- Kvasnica, M., Lofberg, J., & Fikar, M. (2011). Stabilizing Polynomial approximation of explicit MPC . *Automatica*, 2292-2297.
- Lars'en, A. K., Chen, Y., Bruschetta, M., Carli, R., Cenedese, A., Varagnolo, D., et al. (2019). A computationally efficient model predictive control scheme for space debris rendez vous . *IFAC-PapersOnLine* , 103-110.
- Lee, D. H., & Joo, Y. H. (2014). On the Generalized Local Stability and Local Stabilization Conditions for Discrete-Time Takagi–Sugeno Fuzzy Systems . *IEEE Transactions on Fuzzy Systems*, 1654-1668.
- Lendek, Z., & Lauber, J. (2016). Local quadratic and nonquadratic stabilization of discrete-time TS fuzzy systems . *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 24-29). Vancouver, BC, Canada: IEEE.
- Maciejowski, J. M. (2002). *Predictive Control with Constraints* . London: Prentice Hall UK.
- Mayne, D. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 2967–2986.

- Mendel, J. (2018). Explaining the Performance Potential of Rule-Based Fuzzy Systems as a Greater Sculpting of the State Space. *IEEE Transactions on Fuzzy Systems* , 2362-2373.
- Olaru, S. (2005). Contribution à l'étude De La Commande Prédictive Sous Contraintes Par Approche Géométrique . Paris: Supélec Université PARIS XI ORSAY.
- Parisini, T., & Zoppoli, R. (1995). A receding-horizon regulator for nonlinear systems and a neural approximation . *Automatica*, 1443-1451.
- Paulson, J. A., & Mesbah, A. (2020). Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction . *IEEE Control Systems Letters*, 719-724.
- Pin, G., Filippo, M., Pellegrino, F. A., Fenu, G., & Parisini, T. (2013). Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and online design . *International Journal of Control*, 804-820.
- Rawlings, J. B., & Muske, K. R. (1993). Stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 723-757.
- Rubagotti, M., Barcelli, D., & Bemporad, A. (2014). Robust explicit model predictive control via regular piecewise-affine approximation. *International Journal of Control* , 2583-2593.
- Rubagotti, M., Patrinos, P., Guiggiani, A., & Bemporad, A. (2014). Real-Time Model Predictive Control Based on Dual Gradient Projection: Theory and Fixed-Point FPGA Implementation . *International Journal of Robust and Nonlinear Control* , 3292-3310.
- Sala, A., & Arino, C. (2007). Asymptotically necessary and sufficient conditions for stability and performance in fuzzy control: Applications of Polya's theorem . *Fuzzy Sets Systems*, 2671-2686.
- Scibilia, F., Hovd, M., & Olaru, S. (2012). Explicit model predictive control via Delaunay tessellations . *Journal Européen des Systèmes Automatisés (JESA)*, 267-290.
- Stogiannos, M., Alexandridis, A., & Sarimveis, H. (2018). Model predictive control for systems with fast dynamics using inverse neural models . *ISA Transactions*, 161-177.
- Summers, S., Jones, C. N., Lygeros, J., & Morari, M. (2009). A multiscale approximation scheme for explicit model predictive control with stability, feasibility, and performance guarantees. *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) Chinese Control Conference* (pp. 6327-6332). Shanghai, China: Chinese Control Conference, IEEE.
- Sun, Y., Tang, S., Meng, Z., Zhao, Y., & Yang, Y. (2015). A scalable accuracy fuzzy logic controller on FPGA . *Expert Systems with Applications*, 6658-6673.
- Sun, Y., Tang, S., Meng, Z., Zhao, Y., Yang, & Yunhu. (2015). A scalable accuracy fuzzy logic controller on FPGA . *Expert Systems with Applications*, 6658-6673.

- Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 116-132.
- Tanaka, K., & Sugeno, M. (1992). Stability analysis and design of fuzzy control systems . *Fuzzy Sets and Systems*, 135-156.
- Tanaka, K., & Wang, H. O. (2001). *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*. New York, NY: John Wiley & Sons, Inc.
- Tanaka, K., & Wang, H. O. (2004). *Fuzzy control systems design and analysis: a linear matrix inequality approach*. . John Wiley & Sons.
- Tanaka, K., Hori, T., & Wang, H. O. (2001). A fuzzy Lyapunov approach to fuzzy control system design. *in: Proceedings of the 2001 American Control Conference* (pp. 4790–4795). Arlington, VA, USA: IEEE.
- Tanaka, K., Ikeda, T., & Wang, H. O. (1998). Fuzzy regulators and fuzzy observers: Relaxed stability conditions and LMI-based designs . *IEEE Transactions on Fuzzy Systems*, 250-265.
- Tikk, D., Koczy, L., & Gedeon, T. (2003). A survey on universal approximation and its limits in soft computing techniques. *International Journal of Approximate reasoning*, 185-202.
- Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Novoselnik, B., et al. (2019). Acados: A modular open-source framework for fast embedded optimal control. *arXiv preprint*, arXiv:1910.13753 .
- Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., et al. (2018). Towards a modular software package for embedded optimization . *IFAC-PapersOnLine* , 374-380.
- Wang Y, B. S. (2010). Fast Model Predictive Control Using Online Optimization . *IEEE Transactions on Control Systems Technology* , 267-278.
- Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview . *Soft Computing*, 387-408.
- Wang, Y., Zeng, Q. S., & Fu, C. S. (2004). Stability Analysis and Control of Discrete-time Fuzzy Systems: A Fuzzy Lyapunov Function Approach. *5th Asian Control Conference* (pp. 1855-1860). Melbourne, Victoria, Australia: IEEE.
- Ying, H. (1998). Sufficient conditions on uniform approximation of multivariate functions by general TS fuzzy systems with linear rule consequents . *IEEE Transactions on SMC*, 28-44.

ANNEXE A

Système d'inférence flou

Nous introduisons dans cette partie les concepts de base des systèmes flous de type TS. Ainsi, la modélisation par les systèmes flous TS est introduite. Finalement, on fait le point sur la propriété d'approximation des systèmes flous TS.

Généralités sur les systèmes flous :

Proposé en 1965 par Lotfi Zadah les systèmes d'inférence floue ont de très nombreuses applications aujourd'hui, outre la commande, ils sont largement utilisés pour la modélisation, le diagnostic et la reconnaissance de formes. Ces systèmes logiques utilisent des règles linguistiques pour établir des relations entre leurs variables d'entrée et de sortie.

La structure de base d'un système d'inférence flou comprend :

- Interface de fuzzification,
- Base de règles,
- Mécanisme d'inférence floue,
- Interface de défuzzification.

Définition A.1 Fonction d'appartenance

Soit $X \subset \mathbb{R}$ un espace vectoriel (appeler univers de discours), soit l'ensemble $S \subset X$, on appelle μ_S fonction d'appartenance de $x \in X$ associée à l'ensemble S , qui mappe X à l'espace d'appartenance $[0,1] \subset \mathbb{R}^+$, i.e. ,

$$\begin{aligned} \mu_S: X &\rightarrow [0,1] \subset \mathbb{R}^+ \\ x &\rightarrow \mu_S(x) \end{aligned} \tag{A.1}$$

Où $\mu_S(x)$ est appelé degré d'appartenance de x .

Le Tableau A.1 suivant donne quelques exemples des fonctions utilisées.

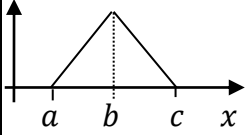
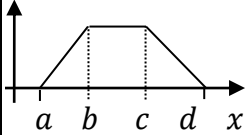
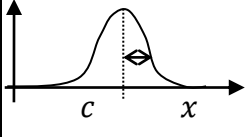
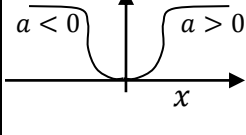
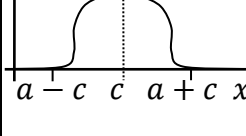
Fonction	Forme	Modèle mathématique
Triangulaire		$F(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$
Trapézoïdale		$F(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-b}\right), 0\right)$
Gaussienne		$F(x; \sigma, c) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right)$
Sigmoïde		$F(x; a, c) = \frac{1}{1 + \exp(-a(x-c))}$
En cloche		$F(x; a, b, c) = \frac{1}{1 + \left \frac{x-a}{a}\right ^{2b}}$

Tableau A.1 Formes et Modèles des fonctions d'appartenance les plus utilisées

Définition A.2 Ensemble flou

Soit l'univers de discours $X \subset \mathbb{R}$, un ensemble flou S en X est un ensemble de couples :

$$S = \{(x, \mu_S(x)) \mid x \in X\} \quad (\text{A.2})$$

μ_S Fonction d'appartenance de x en S .

Définition A.3 Variable flou

On appelle une variable flou v le couple $(S, \mu_S(x))$.

Pour un ensemble flou le degré d'appartenance d'un élément arbitraire $x \in X$ en S est $\mu_S(x)$.

De plus si l'espace signal X est couvert par plusieurs ensembles flous :

$$S_i = \{(x, \mu_{S_i}(x)) \mid x \in X\}, i = 1, 2, \dots, N_S \quad (\text{A.3})$$

le degré d'appartenance d'un élément arbitraire x à ces ensembles flous S_i est :

$$\mu_{S_i}(x), i = 1, 2, \dots, N_S \quad (\text{A.4})$$

Définition A.4 Vecteur flou

L'opérateur de fuzzification \mathcal{F} mappe un élément $x \in X$ vers l'ensemble de variables flous

$$\{v_1 = (S_1, \mu_{S_1}(x)), v_2 = (S_2, \mu_{S_2}(x)), \dots, v_{N_S} = (S_{N_S}, \mu_{S_{N_S}}(x))\} \quad (\text{A.5})$$

Ou encore sous forme vectorielle

$$\mathcal{F}(x) = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N_S} \end{bmatrix} \cong \begin{bmatrix} \mu_{S_1}(x) \\ \mu_{S_2}(x) \\ \vdots \\ \mu_{S_{N_S}}(x) \end{bmatrix} \quad (\text{A.6})$$

Définition A.5 Règle floue

Une règle floue est constituée d'un ensemble de conditions et de conséquences, où elle est représentée sous forme de phrase linguistique du type :

$$\text{Si ensemble de conditions est rempli, ALORS ensemble de conséquences est déduit.} \quad (\text{A.7})$$

Définition A.6 *La Base de règles*

La base de règles est constituée d'un ensemble de règles du type (A.7), où elle peut être construite à partir de la connaissance d'un expert ou par optimisation.

Définition A.7 *Fuzzification*

La fuzzification consiste à traduire les valeurs numériques en valeurs floues c.à.d. en degrés d'appartenance à travers un opérateur de fuzzification.

Définition A.8 *Le mécanisme d'inférence*

Permet d'interpréter dans un calculateur la base de règles.

L'interprétation mathématique de la règle (A.7) est réalisée par :

$T - norm: [0,1] \rightarrow [0,1]$

$$\mu_{r\grave{e}gle} = T - norm \left(\mu_{condition_1}, \mu_{condition_2}, \dots, \mu_{condition_{N_c}} \right) \quad (A.8)$$

Où N_c est le nombre total des conditions, $\mu_{r\grave{e}gle}$ est le degré d'appartenance de la règle, $\mu_{condition_i}$ est le degré d'appartenance de la condition $_i$ ($i=1,2, \dots, N_c$).

Définition A.9 *La fusion*

La fusion est l'interprétation de l'ensemble des règles. Dans le cas des systèmes flous de type Takagi Sugeno la fusion consiste à déterminer la valeur de sortie par :

$$sortie = \sum_{i=1}^r \frac{\mu_{r\grave{e}gle_i}}{\sum_{i=1}^r \mu_{r\grave{e}gle_i}} * cons\acute{e}quence_i \quad (A.9)$$

Où $\mu_{r\grave{e}gle_i}$ est le degré d'appartenance de la règle $_i$ ($i=1 \dots r$), $cons\acute{e}quence_i$ sont des fonctions dans les systèmes de type de Takagi Sugeno.

Définition A.10 *Structure générale d'un système flou Takagi Sugeno discret (Takagi & Sugeno, 1985)*

Pour un vecteur $x(t) \in \mathbb{R}^n$, la structure générale de la i -ème règle d'un système flou de type T-S discret est :

$$Si \ x_1(t) \ est \ S_{i1} \ et \ x_2(t) \ est \ S_{i2} \ \dots \ et \ x_n(t) \ est \ S_{in} \ alors \ \tilde{x}_i(t+1) = f_i(x(t)) \quad (A.10)$$

Où $f_i: \mathbb{R}^n \rightarrow \mathbb{R}^n$. Avec $i = 1, \dots, r$ (r est le nombre de règles). S_{ij} sont les ensembles flous associés au n variables du vecteur $x(t)$.

La fusion donne la sortie du système flou

$$x(t + 1) = \sum_{i=1}^r h_i(z(t)) \tilde{x}_i(t + 1) \quad (\text{A.11.a})$$

ou encore

$$x(t + 1) = \sum_{i=1}^r h_i(z(t)) f_i(x(t)) \quad (\text{A.11.b})$$

Avec $h_i(z(t)) = \frac{\omega_i(x(t))}{\sum_{i=1}^r \omega_i(x(t))}$ est le poids normalisé de pondération pour chaque règle, et $\omega_i(z(t)) = \prod_{j=1}^p S_{ij}(x_j(t))$. $S_{ij}(x_j(t))$ est le degré d'appartenance de $x_j(t)$ à l'ensemble S_{ij} .

Le schéma ci-après (voir Figure A.1) donne un aperçu général du principe de fonctionnement du système flou de type Takagi Sugeno.

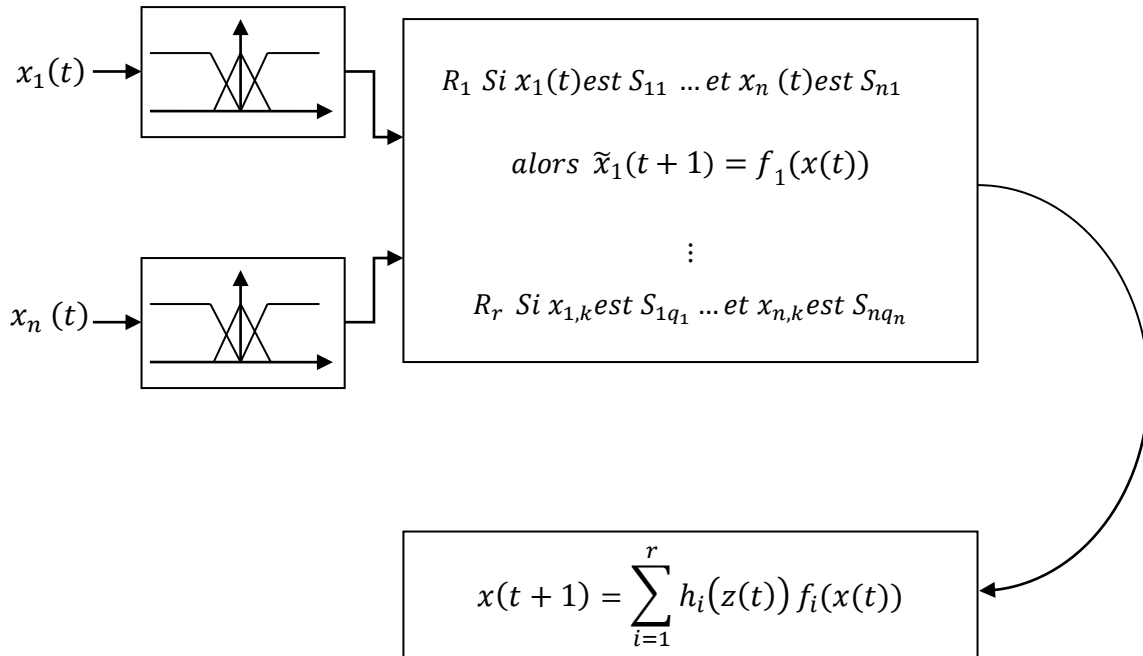


Figure A.1 Schéma de principe d'un système flou de Takagi Sugeno

Annexe B

Solution MPC

Cette annexe a pour but d'initier le lecteur et lui donner un aperçu général sur l'utilisation de la toolbox MPT et l'algorithme PSO afin de résoudre le problème MPC

1. Toolbox MPT :

La Multi-Parametric Toolbox (MPT) est une collection d'algorithmes pour Matlab qui vise à résoudre les problèmes d'optimisation paramétrique dans la commande optimale soumise à des contraintes. En particulier, pour la modélisation, le contrôle, l'analyse et le déploiement de contrôleurs optimaux contraints. Ainsi, son objectif principal est de fournir des moyens de calcul efficaces pour la conception et l'application d'un contrôleur prédictif explicite à base de modèle (MPC) [HKJM13].

Installation

La nouvelle version de MPT est disponible sur

<https://www.mpt3.org/Main/Installation>

Lors de l'installation sur MATLAB en système d'exploitation Windows la YALMIP toolbox n'est pas installée automatiquement. Cette dernière est disponible sur

<https://yalmip.github.io/download/>

Après l'installation manuelle de YALMIP il faut ajouté le lien à la fonction *mpt_init* de la toolbox MPT (voir figure B.1) :

```
addpath(genpath('C:\Users\BMZ\Documents\MATLAB\mpt\YALMIP-master'));
```

La même procédure est à faire avec les autres packages tel que *fourier*.

```
function mpt_init
% MPT3 installation script

% global iteration counter (to detect how often mpt_solve is called)
global MPTOPTIONS

.
.
.

addpath(genpath('C:\Users\BMZ\Documents\MATLAB\mpt\YALMIP-master'));
% add path to these subfolders if not set
subfolders = {'demos';
              'utils';
              'modules';
              'tests'};
addpath(genpath('C:\Users\BMZ\Documents\MATLAB\mpt\fourier'));
% add current folder (in case we're somewhere else)
addpath(mpt3_main_path);

.
.
.
end
```

Figure B.1 Ajout du lien de YALMIP à mpt_init

Application

Afin d'illustrer les étapes à suivre pour résoudre un problème MPC utilisant MPT on prend l'exemple d'application suivant d'un double intégrateur :

Exemple B.1

Considérons le problème de commande d'un système double intégrateur [] :

$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(t) \\ y(t) = [1 \quad 0]x(t) \end{cases}$$

tel que les contraintes sur les états

$$-1 < x_{k,1} < 1 \text{ et } -1 < x_{k,2} < 1$$

ainsi la commande est soumise à la contrainte

$$-1 < u_k < 1,$$

avec les pondérations $Q = I_2, R = 1$, l'horizons de prédiction $N = 5$ et les conditions initiales $x_0 = [0.5 \ 1]$.

Code Matlab pour trouver la solution explicite de la MPC utilisant la toolbox MPT :

```
clear all;

clc;

% Modèle du double intégrateur

sysStruct.A=[1 1; 0 1];

sysStruct.B=[0.5; 1];

sysStruct.C=[ 1 0];

sysStruct.D=0;

% Contraintes sur les états

sysStruct.xmax = [ 1; 1];

sysStruct.xmin = [-1;-1];

% Contraintes sur les entrées

sysStruct.umax = 1;

sysStruct.umin = -1;

probStruct.N=5;           % Horizon de prédiction

probStruct.Q=[1 0; 0 1]; % Poids de pondération sur les états
```

```
probStruct.R=1;           % Poids de pondération sur les états

probStruct.norm=2;       % soit 1 ou Inf pour une fonction coût
                          linéaire, ou 2 pour un coût
                          quadratique.

probStruct.subopt_lev=0;% lev niveau d'optimalité

x0=[0.5,1] ;             % Condition initial

% Construire le contrôleur

model = mpt_import(sysStruct, probStruct);

mpc = MPCController(model,probStruct.N);%

loop = ClosedLoop(mpc,model);

data = loop.simulate( x0',15);

figure; % Figure B.2

plot(data.X(1,:), 'r');hold on;

plot(data.X(2,:), 'b');hold off; grid on;

figure; % Figure B.3

plot(data.U);hold on; grid on;
```

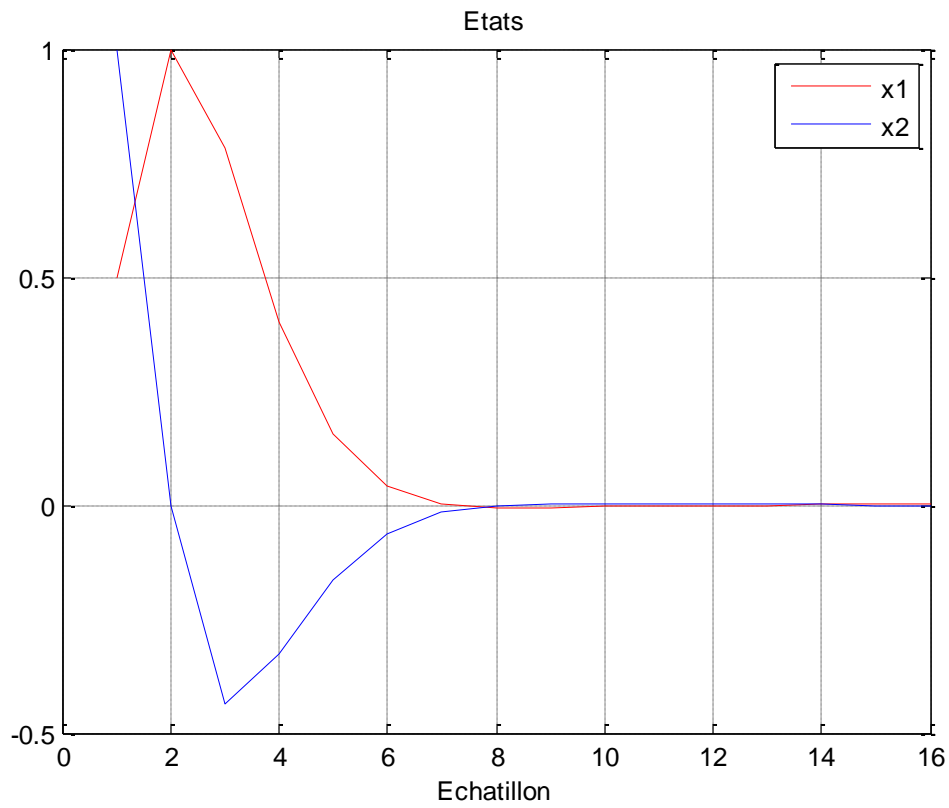



Figure (B.2) Etats de la MPC par MPT de l'exemple B.1.

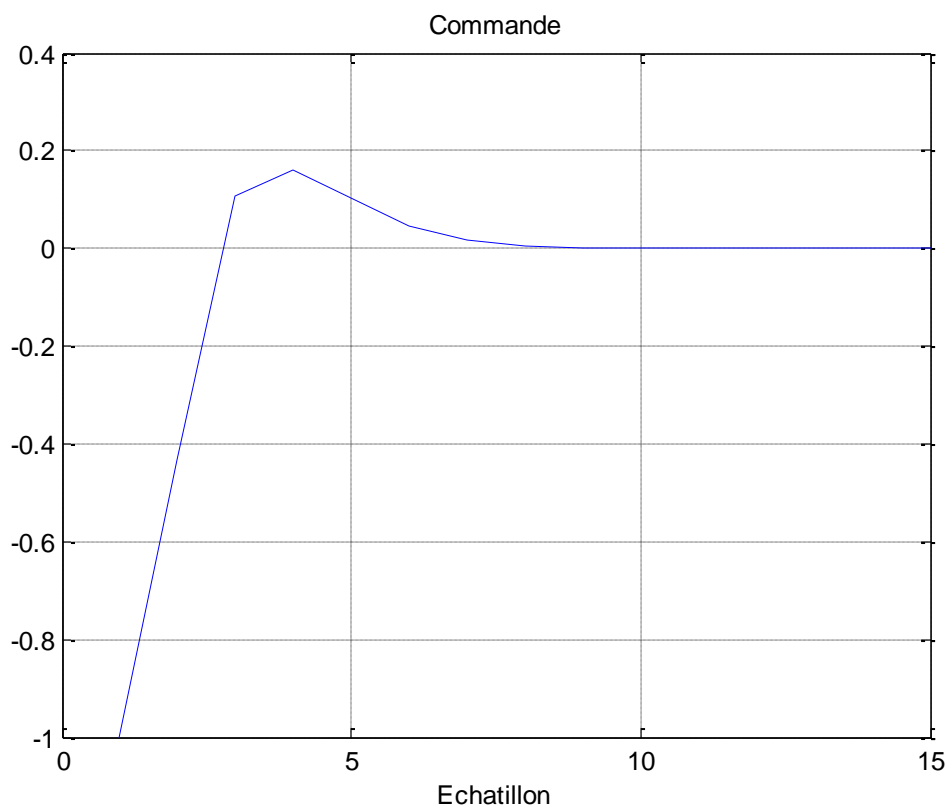


Figure (B.3) Signal de commande de la MPC par MPT de l'exemple B.1.

Code Matlab pour pour trouver la solution explicite de la MPC utilisant la toolbox MPT :

```
clear all;

clc;

% Modèle du double intégrateur

sysStruct.A=[1 1; 0 1];

sysStruct.B=[0.5; 1];

sysStruct.C=[ 1 0];

sysStruct.D=0;

% Contraintes sur les états

sysStruct.xmax = [ 1; 1];

sysStruct.xmin = [-1;-1];

% Contraintes sur les entrées

sysStruct.umax = 1;

sysStruct.umin =-1;

probStruct.N=5;           % Horizon de prédiction

probStruct.Q=[1 0; 0 1];% Poids de pondération sur les états

probStruct.R=1;          % Poids de pondération sur les états

probStruct.norm=2;       % soit 1 ou Inf pour une fonction coût
                           linéaire, ou 2 pour un coût
                           quadratique.
```

```
probStruct.subopt_lev=0;% lev niveau d'optimalité

% Construire le contrôleur

model = mpt_import(sysStruct, probStruct);

mpc = MPCController(model,probStruct.N);%

empc = mpc.toExplicit();

figure; empc.feedback.fplot(); % Figure B.4

figure; empc.partition.plot(); % Figure B.5
```

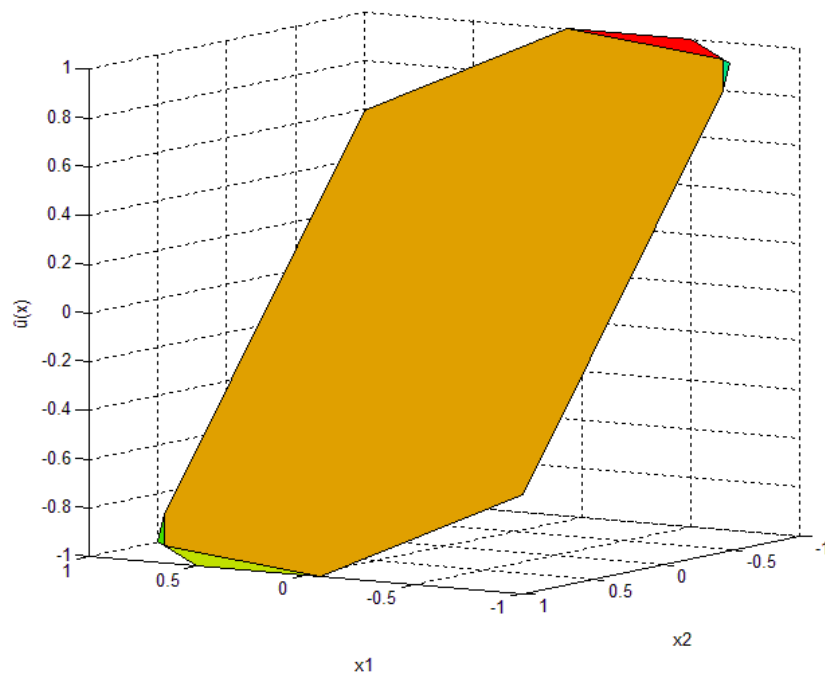


Figure (B.4) Loi explicite de la commande prédictive de l'exemple 2.1.

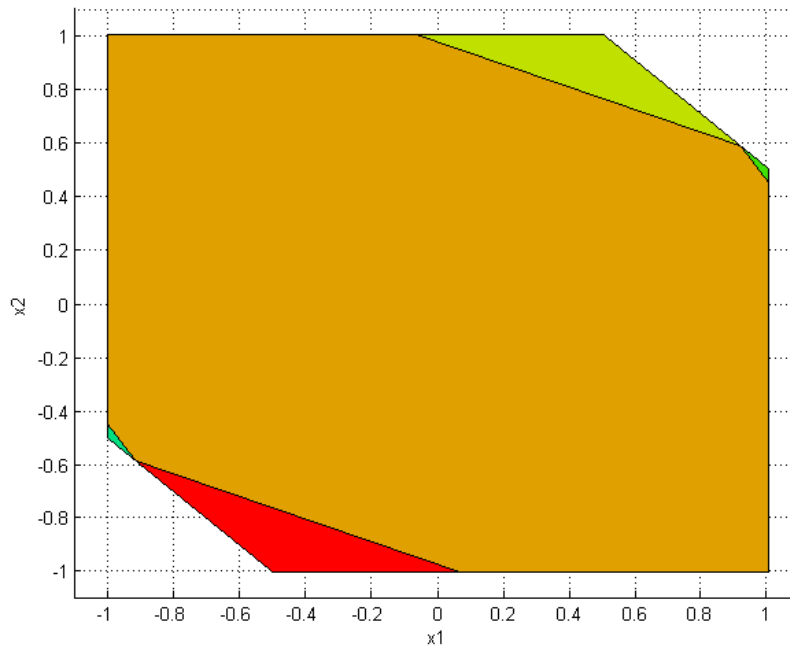


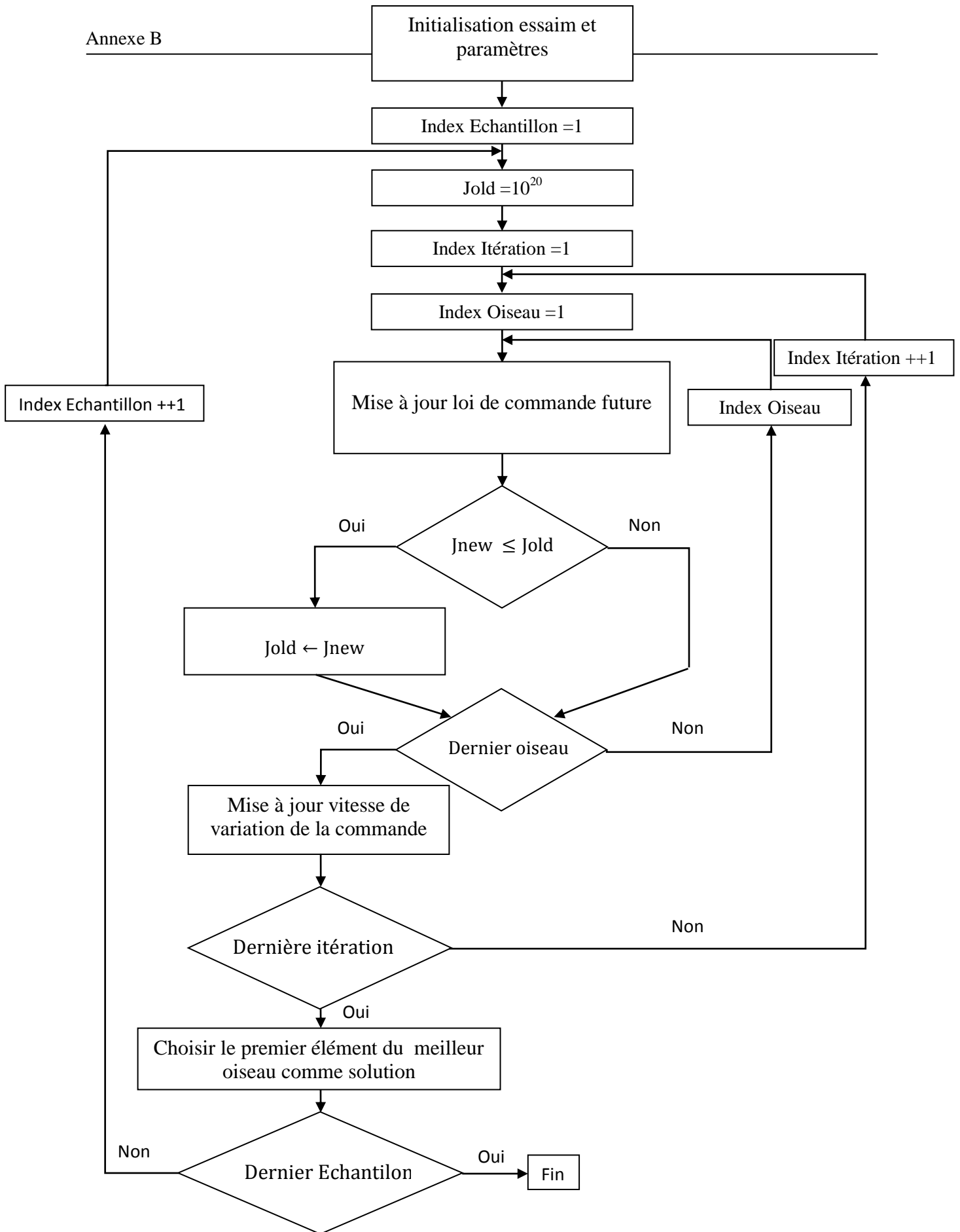
Figure (B.5) Régions de la solution MPC explicite pour l'exemple 2.1, projection sur l'espace d'état.

La figure B.2 montre la loi explicite de la commande prédictive et sa projection sur l'espace d'état. Où la solution explicite possède 5 régions. Ainsi la figure B.3 donne la projection sur l'espace d'état de la solution MPC explicite.

2. Algorithme Optimisation par particule d'essaim PSO :

L'organigramme (3.1) illustre le principe général des différentes étapes de l'implémentation de la commande prédictive en utilisant l'optimisation par essaim particulaire.

L'essaim est constitué par un ensemble de particules où chaque particule représente le signal de commande futur. Il est initialisé par une combinaison de valeurs aléatoires, à chaque période d'échantillonnage la fonction coût est initialisée à une grande valeur et le calcul de la loi de commande futur est effectué à travers la minimisation de cette fonction. Au prochain coup d'horloge, seul le premier élément de la loi de commande est appliqué sur le système.



Organigramme (B.1) Calcul de la solution MPC en utilisant l'optimisation par Essaim particulaire

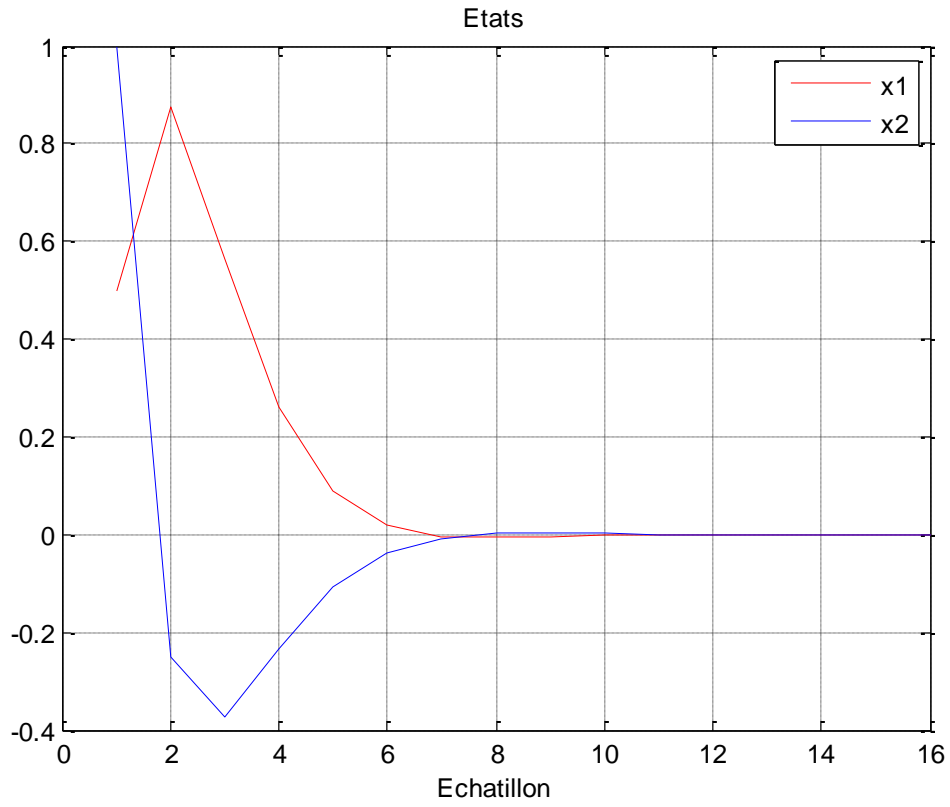


Figure (B.6) Etats de la MPC par PSO de l'exemple B.1.

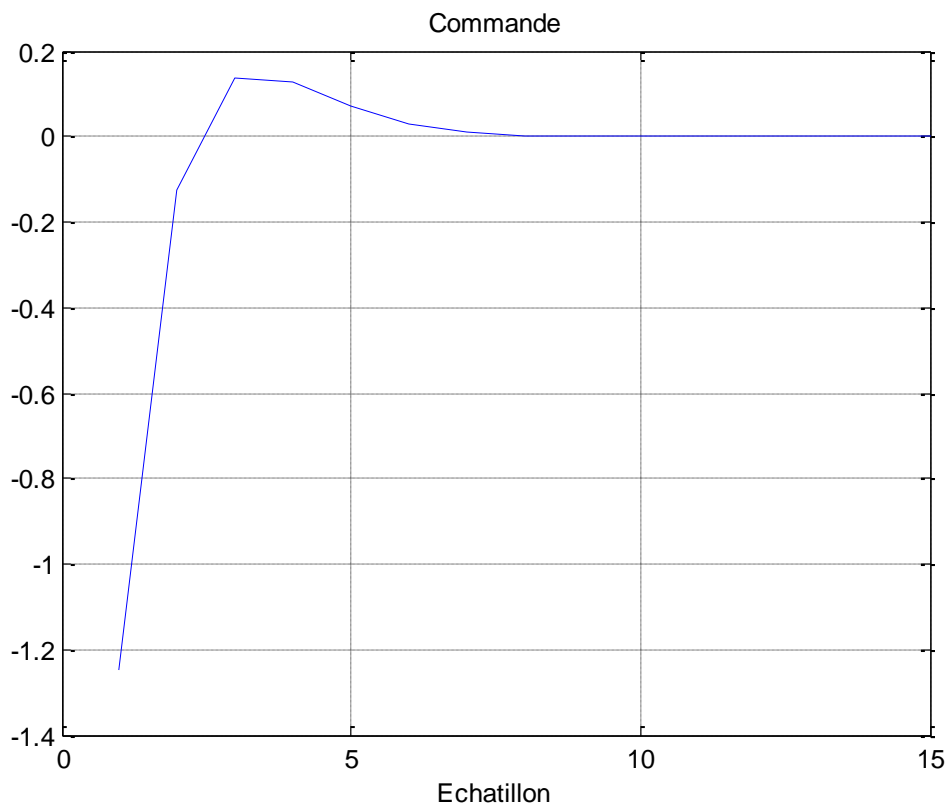


Figure (B.7) Signal de commande de la MPC par PSO de l'exemple B.1.