الجـــــــــــــــــــــــــــمهـوريـــــــة الجـــــــــــــــــزائـريـة الديمـــــــــقـراطيـة الـشـــــــــــعبيـة

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــــــعـليـم العـــــــــالي والبـــــــــــــحث العـــــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة محمد البشير الإبراهيمي – برج بوعريريج

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté des Sciences et de la Technologie

Département Electromécanique

MÉMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Automatique

Spécialité : Automatique et Informatique Industrielle

Par : - RAGOUB Khaoula

Sujet

La Synthèse d'une Commande intelligente des systèmes Robotiques

Soutenu publiquement, le: 30 / 06 / 2025, devant le jury composé de :

Mme.TAHI wassila	MAA	**Univ-BBA**	Président.
Mme.ZEHAR Djamila	MCB	**Univ-BBA**	Examinateur.
Mr.BENSIDHOUM Tarek	MCA	**Univ-BBA**	Encadrant.

Année Universitaire 2024/2025

Remerciement

Je souhaite exprimer ma profonde gratitude envers le Dieu ToutPuissant qui m'a accordé la f orce et la sagesse nécessaires pour l'accomplissement de ce mémoire.

Avant tout, je tiens à exprimer ma gratitude envers mon directeur de mémoire, « BENSIDHOUM Tarek », pour le temps qu'il a consacré à me fournir les outils méthodologiques essentiels pour mener cette recherche. Sa demande m'a fortement motivé.

Je tiens aussi à remercier ma famille et mes amis pour leur soutien indéfectible et leur compréhension tout au long de nos études.

Je tiens à remercier envers tous les professeurs du Département d'électromécanique de l'Université Mohamed El Bachir El Ibrahimi de BBA pour leurs enseignements et leur disponibilité. Leur dévouement a enrichi mon parcours éducatif et m'a incité à donner le meilleur de moi-même.

(وَآخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ)

Dédicace

A mon père et ma mère.

A mes sœurs et mon frère.

A mes nièces et mes neveux.

Résumé

Dans ce mémoire, nous somme intéresses au domaine du contrôle intelligent des systèmes robotiques non linéaires, et plus particulièrement celui du robot manipulateur PUMA 560. L'objectif principal est de concevoir une commande robuste et performante en combinant trois approches avancées : la commande glissante, la logique floue et les réseaux de neurones. La commande hybride proposée permet de tirer parti des avantages de chaque méthode tout en atténuant leurs inconvénients. Nous avons utilisé l'environnement MATLAB/Simulink pour valider l'efficacité de la technique proposée. Les résultats de simulations démontrent une nette amélioration des performances en termes de suivi de trajectoire, de robustesse et de réduction du chattering, comparée aux approches classiques. D'autre part, Pour faire l'implémentation en temps réel des commandes non linéaire, nous avons présenté une implémentation d'une commande à base des algorithmes génétiques sur un moteur à courant continue en utilisant MATLAB et Arduino.

Abstract

In this thesis, we are interested in the field of intelligent control of nonlinear robotic systems, and more particularly that of the PUMA 560 manipulator robot. The main objective is to design a robust and efficient control by combining three advanced approaches: sliding control, fuzzy logic and neural networks. The proposed hybrid control makes it possible to take advantage of the advantages of each method while mitigating their disadvantages. We used the MATLAB/Simulink environment to validate the effectiveness of the proposed technique. The simulation results demonstrate a significant improvement in performance in terms of trajectory tracking, robustness and chattering reduction, compared to conventional approaches. On the other hand, to implement nonlinear controls in real time, we presented an implementation of a nonlinear scheme of genetic algorithms on a DC motor using MATLAB and Arduino.

ملخص

في هذه المذكرة، نحن مهتمون بمجال التحكم الذكي في الأنظمة الروبوتية غير الخطية، وبشكل خاص في روبوت المعالجة (PUMA 560),الهدف الرئيسي هو تصميم تحكم قوي و فعال من خلال دمج ثلاث نهج متقدمة :التحكم الانزلاقي, المنطق الضبابي، والشبكات العصبية. تسمح التحكم الهجين المقترح بالاستفادة من مزايا كل طريقة مع التخفيف من عيوبها.
لقد استخدمنا بيئة (MATLAB\Simulink) للتحقق من فعالية التقنية المقترحة، تظهر نتائج المحاكاة تحسنا واضحا في الاداء من حيث تتبع المسار، والصلابة وتقليل الاهتزاز، مقارنةً بالنهج التقليدية. من ناحية أخرى، لتنفيذ التحكم غير الخطي في الوقت الحقيقي، للتحكم القائم على الخوارزميات الجينية قدمنا تنفيذًا على محرك تيار مستمر باستخدام الماطلاب والاردوينو

Table de matière

Remerciement I
Dédicace II
Résumé III
Table de matière IV
Table des figures VII
Liste de tableaux IX
Notation X
Introduction Générale 1
Chapitre I : Modélisation de robot puma 560. 2
1. Introduction 2
2. Historique et évaluation 2
3. Les Caractéristiques techniques 3
4. La Modélisation géométrique de puma 560 4
4.1 La Modélisation géométrique Direct (DH) 4
4.2 La modélisation géométrique Inverse (MGI) 5
5. LA Modélisation cinématique 10
5.1 Le Modèle cinématique direct (MCD) 10
5.2 Le Modèle cinématique inverse (MCI) 10
6. Modélisation dynamique 11
6.1 L'équation de Lagrange applique au robot 11
6.2 La forme générale des équations dynamiques 12
6.3 Calculs de l'énergie 12
6.4 Les équations différentielles du PUMA 560 14
7. Conclusion 17
Chapitre2 : les techniques de commande intelligente. 18
1. Introduction 18
2. La Commande par mode glissant 18
2.1 Les surfaces de glissement non linéaires 19
2.2 Mode de glissement pour un système non linéaire 20
3. La logique floue 22
3.1 Ensembles flous 23

3.2 La commande floue et les systèmes experts flous 24
3.3 La logique floue dans la commande à structure variable 28
3.4 Présentation d'une commande Floue-glissante 29
4. Les réseaux de neurones 33
4.1 Historiques de RNA 34
4.2 Le neurone formel 35
4.3 Le réseau de neurones multicouche (MLP) 36
4.4 Les réseaux de neurones dans la commande à structure variable 37
4.5 L'Approximation des fonctions et prédiction à l'aide de réseaux neuronaux 38
5. Conclusion 39
Chapitre III : La commande Neuro-floue-glissante. 39
.1 Introduction 39
2. Commande par mode glissant du robot PUMA 560 39
2.1 La synthèse de la loi de commande 39
2.2 Résultats et simulation 42
3. Synthèse de la commande Neuro-floue-glissante au robot PUMA 560 48
3.1 La commande par logique floue 49
3.2 La commande par réseau de neurone 51
3.3 La commande hybride 55
3.4 Paramètres de simulation 55
3.5 Commentaires sur les résultats 62
4. Conclusion 63
Chapitre IV : implémentation des commandes par des algorithmes génétiques sur Arduino 64
1. Introduction 64
2. Le matériel et logiciels utilises 64
2.1 Modèle de moteur 64
2.2 La carte Arduino UNO 66
2.3 MATLAB-Simulink de Math Works 67
3. Algorithmes utilisés 68
3.1 PSO (Optimisation d'Essaim de Particules) 68
3.2 GWO (l'optimisation par loup gris) 69
4. Conception du régulateur PID 70
5. Implémentation sur la carte Uno 72

6. Résultats et discussion 75
7. Conclusion 76
Conclusion générale 77
Reference et Bibliographie .....
Annex .....

Table des figures

Figure I. 1:presentation de robot puma 560. 3
Figure I. 2:composition des vitesses. 13

Figure II. 1:La commande équivalente et la dynamique inverse du système. 22
Figure II. 2:la logique Classique 24
Figure II. 3 : la logique floue..... 24
figure II. 4..... 24
Figure II. 5:Partition floue de l'espace autour de la surface de glissement. 31
Figure II. 6:Fonctions d'appartenance de l'entrée s et de sortie uf. 31
Figure II. 7:Résultats de l'inférence des règles floues pour différentes valeurs de r. 33
Figure II. 8: Neurone artificiel formel..... 36
Figure II. 9: L'architecture d'un réseau de neurones multicouches. 37
Figure II. 10: Schéma de principe d'identification par réseau de neurones. 39

Figure III. 1 : Schéma bloc de la commande par mode glissant appliquée au robot manipulateur PUMA 560. 42
Figure III. 2: Trajectoire cycloïdale. 43
Figure III. 3: Commandes équivalentes de la commande VSC appliquée au bras manipulateur. 44
Figure III. 4: Positions articulaires de la commande pour un test à vide. 45
Figure III. 5: les erreurs de poursuite de la commande pour un test à vide. 46
Figure III. 6: Surfaces de glissement et commandes effectives pour un test à vide..... 47
Figure III. 7: Schéma bloc de la commande Neuro-floue-glissante proposée pour le bras manipulateur PUMA 560. 49
Figure III. 8: partie prémisse. 50
Figure III. 9: la partie conclusion. 51
Figure III. 10: Réseau de neurones utilisé pour calculer la commande équivalente pour le robot manipulateur. 52
Figure III. 11:la simulation des commandes neurone-floue glissante. 56
Figure III. 12: la commande applique sur chaque articulation. 56
Figure III. 13: Positions articulaires et erreurs de poursuite de la commande Neuro-floue-Glissante. 58
Figure III. 14: Commandes discontinues type floue. 59
Figure III. 15: commandes équivalentes neuronale. 60
Figure III. 16: Surfaces de glissement et commandes effectives de la commande Neuro-floue-Glissante. 61

Figure IV. 1: Moteur à courant continu JGB37-3530. 12V avec encodeur. 66
Figure IV. 2:carte Arduino UNO. 67

Figure IV. 3: L'organigramme de l'algorithme GWO. 70
Figure IV. 4: interface graphique sur Matlab. 71
Figure IV. 5:PID aléatoire vs PSO-PID et GWO-PID. 71
Figure IV. 6:schéma de commande PID sur Simulink. 72
Figure IV. 7: Schéma de câblage du moteur avec la carte Arduino. 73
Figure IV. 8: Expérience mise en place. 73
Figure IV. 9 : Programme Simulink utilisé. 74
Figure IV. 10 : Courbes des réponses des système en implémentation avec le régulateur PID(PSO\GWO). 75
Figure IV. 11 : les vitesse (RPM) et ses erreurs. 76

Liste de tableaux

Tableau I. 1: caractéristiques de puma 560. 3
Tableau I. 2: Paramètre D-H pour puma 560. 4

Tableau III 1:comparaison entre les erreurs des commandes. 62

Tableau IV. 1:Caractéristiques General de moteur JGB37-3530. 64
Tableau IV. 2 : Caractéristiques techniques d'encodeur. 65
Tableau IV. 3: Caractéristiques techniques détalée de moteur. 65
Tableau IV. 4:Caractéristiques techniques de la carte Arduino Uno. 67
Tableau IV. 5 : les valeurs de PID calculée 71
Tableau IV. 6 : comparaisons entre les temps de réponses de PID vs PID(PSO\GWO). 75
Tableau IV. 7 : les valeurs de l'erreurs de vitesse. 76

Notation

PUMA: Programmable Universal Machine Arm.

DH: denavit hartenberg.

SISO: single input single output.

MIMO : Muli input multi output

LF : logique floue.

RNA : réseau de neurone artificielle.

MLP : Perceptron Multicouche.

PID :Proportionnel, Intégral, Dérivé.

PSO (Optimisation d'Essaim de Particules)

GWO (l'optimisation par loup gris)

RPM: revolutions per minute "tours par minute"

PWM: Pulse Width Modulation.

MCC : Moteur à courant continu.

PPR : Impulsions par tour.

TCP : transmission control Protocol.

VSC : commande a structure variable.

Introduction Générale

Dans les dernières années, l'axe de recherche le plus connu pour le domaine de la robotique est le développement des techniques de commande performantes et robustes constitue notamment face à la complexité croissante des systèmes industriels modernes. Les robots manipulateurs, tels que le robot PUMA 560, illustrent parfaitement cette complexité en raison de leur comportement fortement non linéaire, de la présence d'incertitudes, ainsi que des perturbations internes et externes auxquelles ils peuvent être soumis.

Face aux limites des techniques classiques de commande linéaire, plusieurs approches dites « intelligentes » ont vu le jour, permettant d'améliorer la performance, la stabilité et l'adaptabilité des systèmes. Parmi ces approches, on distingue la commande floue, la commande par réseaux de neurones, ainsi que la commande par modes glissants. Chacune de ces techniques possède des avantages distincts : la commande floue permet de modéliser des connaissances imprécises sous forme de règles, les réseaux de neurones sont capables d'apprentissage et d'approximation de fonctions complexes, tandis que la commande par mode glissant est réputée pour sa robustesse face aux incertitudes.

Cependant, l'utilisation isolée de ces méthodes peut ne pas suffire pour garantir un niveau de performance optimal. C'est dans ce contexte qu'apparaît la commande hybride, une stratégie qui combine les forces de différentes techniques afin de tirer parti de leurs complémentarités.

L'objectif de ce travail est ainsi de concevoir et de simuler une commande hybride, intégrant les réseaux de neurones, la logique floue et la commande par mode glissant, pour le pilotage du robot manipulateur PUMA 560.

Pour bien présenté notre travail, nous avons diviser le manuscrit quatre chapitre comme suit : Le première chapitre est consacré à la modélisation cinématique et dynamique du robot PUMA 560. En suit, nous avons introduit dans le deuxième chapitre les concepts de base des commandes non linéaires intelligentes. Tel que la commande du réseau de neurones et la commande floue. Le chapitre trois présente le développement d'une stratégie de commande hybride combinant les trois approches avancées qui sont présenté et détailler dans le chapitre précèdent. Pour faire l'implémentation en temps réel des commandes non linéaire, nous avans présenter dans le dernier chapitre une implémentation des algorithmes génétiques sur un moteur à courant continue en utilisant MATLAB et Arduino.

Chapitre I :

Modélisation de robot puma 560

Chapitre I : Modélisation de robot puma 560.

1. Introduction

L'automatisation industrielle s'appuie de plus en plus sur des robots manipulateurs capables de réaliser des tâches précises et répétitives. Parmi ces robots, le PUMA560 est un bras robotique industriel conçu par Animation dans les années 1980. Son appellation "PUMA", vient de 'acronyme

Pour « Programmable Universal Machine Arm », mettant en avant sa capacité à réaliser mettant en

avant sa capacité à réaliser une variété de tâches d'assemblage et de manutention dans des

Environnements industriels. Avec six degrés de liberté piloté par un ordinateur il a été pionnier dans l'automatisation des chaînes de production.[1]

Ce chapitre est consacré à la modélisation cinématique et dynamique du robot PUMA 560. Nous commencerons par une présentation des caractéristiques techniques de ce robot. Ensuite, nous développerons le modèle cinématique directe et inverse. Enfin, nous aborderons les équations différentielles d'écrivant le modèle dynamique de ce robot qui on va l'utiliser dans le reste de manuscrite

2. Historique et évaluation

Le mot « ROBOT » a fait son apparition pour la première fois dans le langage grâce à l'écrivain tchèque Karel Čapek dans son livre « les Robots Universels de Rossum » en 1920, où « ROBOTA » désigne le travail en tchèque. Depuis ce moment, ce terme a été utilisé pour décrire un large éventail de dispositifs mécaniques, tels que les téléopérateurs, les véhicules sous-marins, les explorateurs autonomes, parmi d'autres. En général, tout appareil fonctionnant avec un certain niveau d'autonomie, piloté par un ordinateur, est considéré comme un robot. Dans ce document, le mot « robot » se réfèrera à un bras manipulateur industriel piloté par un ordinateur. Bien que ce type de dispositif diffère considérablement des robots de science-fiction, il représente un système électromécanique hautement complexe dont la description analytique nécessite des méthodes avancées, posant ainsi un véritable défi de recherche. [2]

Au milieu des années 1970, le PUMA 560, un manipulateur robotique conçue par Victor Scheinmann, d'abord à Stanford Researche Institute apparaît, avant d'étres développé et commercialisé en 1985, pour General Motors par Unimation.

3. Les Caractéristiques techniques

Figure I. 1:presentation de robot puma 560.

Le PUMA 560 est un bras robotisé possédant 6 axes, capable de supporter une charge utile de 2,5 kg et de travailler dans une portée de 864 mm. Ce robot peut être utilisé pour le contrôle à distance du TCP.[3]

Marque	PUMA
Model	560
Type	Bras Robot
Axes	6
Charge utile	2.5 kg
Portée	864 mm
Répétabilité	0.1 mm
Poids	54 kg

Tableau I. 1: caractéristiques de puma 560.

4. La Modélisation géométrique de puma 560

4.1 La Modélisation géométrique Direct (DH)

Les paramètres géométriques du puma 560 sont présentés dans le tableau (I. 1). Dans ce tableau, 'i' représente le numéro de l'articulation, 'α' désigne l'angle entre les axes zi-1 et zi, correspondant à une rotation autour de xi-1 Par ailleurs, 'd'indique la distance entre zi-1 et zi suivant la direction de xi-1. 'θ' désigne quant à lui l'angle entre xi-1 et xi, correspondant à une rotation autour de zi, et 'a' tandis que la distance entre xi-1 et xi est mesurée le long de l'axe zi.

I	α_i	a_i	θ_i	d_i
${}^{0}_{1}T$	0	0	θ_1	0
${}^{1}_{2}T$	-90	0	θ_2	0
${}^{2}_{3}T$	0	a_2	θ_3	d_3
${}^{3}_{4}T$	-90	a_3	θ_4	d_4
${}^{4}_{5}T$	90	0	θ_5	0
${}^{5}_{6}T$	-90	0	θ_6	0

Tableau I. 2: Paramètre D-H pour puma 560.

Les Matrices de Transformations Les homogènes sont un instrument mathématique qui facilite le calcul d'un changement de repère à travers une unique opération matricielle. Leur emploi est fondamental dans la modélisation des robots.

$$ {}^{0}_{1}\mathbf{T} = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{T1} \qquad \text{(I.1)} $$

$$ {}^{1}_{2}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C(-90) & -S(-90) & 0 \\ 0 & S(-90) & C(-90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C2 & -S2 & 0 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C2 & -S2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S2 & -C2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T2 \qquad \text{(I.2)} $$

$$ {}^{2}_{3}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C3 & -S3 & 0 & 0 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C3 & -S3 & 0 & a_2 \\ S3 & C3 & 0 & d_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T3 \qquad \text{(I.3)} $$

$$ {}^{3}_{4}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & \mathrm{c}(-90) & -\mathrm{s}(-90) & 0 \\ 0 & \mathrm{s}(-90) & \mathrm{c}(-90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C4 & -S4 & 0 & 0 \\ S4 & C4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C4 & -S4 & 0 & a_3 \\ -S4 & -C4 & 0 & d_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T4 \qquad \text{(I.4)} $$

$$ {}^{4}_{5}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C(90) & -S(90) & 0 \\ 0 & S(90) & C(90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T5 \qquad \text{(I.5)} $$

$$ {}^{5}_{6}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C(-90) & -S(-90) & 0 \\ 0 & S(-90) & C(-90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C6 & -S6 & 0 & 0 \\ S6 & C6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C6 & -S6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -S6 & C6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T6 \qquad \text{(I.6)} $$

$$ {}^{0}_{6}\mathbf{T} = \mathrm{T1} * \mathrm{T2} * \mathrm{T3} * \mathrm{T4} * \mathrm{T5} * \mathrm{T6} \qquad \text{(I.7)} $$

4.2 La modélisation géométrique Inverse (MGI)

Le modèle inverse géométrique permet d'établir la disposition des articulations (θ1, θ2, θ3, θ4 et θn) en fonction de la position et de l'orientation de l'effecteur d'un bras manipulateur. Il existe plusieurs approches pour établir la configuration de articulations :

1. L'approche analytique qui est employée pour les robots avec 6 degrés de liberté (n=6).

2. L'approche numérique qui est appliquée aux robots ayant moins de 6 degrés de liberté (n>6). La majorité des bras manipulateurs avec 6 degrés de liberté à l'échelle mondiale possèdent un poignet sphérique, comme le puma 560. Un poignet sphérique signifie la présence de trois articulations pivot avec axes qui se croisent. En ce qui concerne le modèle puma 560, nous avons : ā4 perpendiculaire à ā5, et ā6 ayant la même orientation que ā4, tout en restant inchangée malgré les variations de θ4, θ5, et θ6.

La solution au problème est obtenue en appliquant la méthode de Peiper, puisque le puma 560 possède 6 degrés de liberté et un poignet sphérique. La méthode de Peiper repose sur la séparation du problème en deux parties :

1.[θ1,θ2,θ3] définissent la position de poignet.

2. [θ4, θ5, θ6] définissent l'orientation du poignet.[4]

***Cas puma 560 :**

Nous allons maintenant aborder la résolution des équations cinématiques du PUMA 560, que nous avons précédemment développées

$$ {}^{0}_{6}\mathrm{T} = {}^{0}_{1}\mathrm{T}(\theta_1) * {}^{1}_{2}\mathrm{T}(\theta_2) * {}^{2}_{3}\mathrm{T}(\theta_3) * {}^{3}_{4}\mathrm{T}(\theta_4) * \ {}^{4}_{5}\mathrm{T}(\theta_5) * {}^{5}_{6}\mathrm{T}(\theta_6) = {}^{0}_{6}\mathrm{T} = \begin{bmatrix} \mathrm{r}_{11} & \mathrm{r}_{12} & \mathrm{r}_{13} & \mathrm{p}_x \\ \mathrm{r}_{21} & \mathrm{r}_{22} & \mathrm{r}_{23} & \mathrm{p}_y \\ \mathrm{r}_{31} & \mathrm{r}_{32} & \mathrm{r}_{33} & \mathrm{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(I.8)} $$

On a:

$$ {}^{4}_{6}\mathrm{T} = \ {}^{4}_{5}\mathrm{T} * \ {}^{5}_{6}\mathrm{T} = \begin{bmatrix} C5C6 & -C5S6 & -S5 & 0 \\ S6 & C6 & 0 & 0 \\ S5C6 & -S5S6 & C5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(I.9)} $$

Un calcul de (I .8) qui place la dépendance θ_1 sur la partie gauche de l'équation est :

$$ {}^{0}_{6}\mathrm{T}[{}^{0}_{1}\mathrm{T}(\theta_1)]^{-1} = {}^{1}_{2}\mathrm{T}(\theta_2) * {}^{2}_{3}\mathrm{T}(\theta_3) * {}^{3}_{4}\mathrm{T}(\theta_4) * \ {}^{4}_{5}\mathrm{T}(\theta_5) * {}^{5}_{6}\mathrm{T}(\theta_6) \quad \text{(I.10)} $$

Inversant $\mathbf{{}^{0}_{1}T}$ dans (I .8) s'écrit comme suite :

$$ \begin{bmatrix} C1 & S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{r}_{11} & \mathrm{r}_{12} & \mathrm{r}_{13} & \mathrm{p}_x \\ \mathrm{r}_{21} & \mathrm{r}_{22} & \mathrm{r}_{23} & \mathrm{p}_y \\ \mathrm{r}_{31} & \mathrm{r}_{32} & \mathrm{r}_{33} & \mathrm{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{{}^{1}_{6}T} \quad \text{(I.11)} $$

Avec :

$$ -S_1 P_x + C_1 P_y = d_3 \quad \text{(I.12)} $$

Une équation de cette nature est résolue par substitution trigonométrique :

$$ \begin{aligned} P_x &= p\cos\phi \\ P_y &= p\sin\phi \end{aligned} \quad \text{(I.13)} $$

Avec :

$$ \begin{aligned} p &= \sqrt{\mathrm{px}^2 + \mathrm{py}^2} \\ \phi &= \mathrm{Atan2}(P_x, P_y) \end{aligned} \quad \text{(I.14)} $$

En remplaçant (I.13) dans (I.12), on a :

$$ C_1 S\phi - S_1 C\phi = \frac{d_3}{p} \quad \text{(I.15)} $$

De la formule de différence d'angles

$$ \sin(\phi - \theta_1) = \frac{d_3}{p} \quad \text{(I.16)} $$

En conséquence :

$$cos(\text{ø} - \theta_1) = \pm\sqrt{1 - \frac{d_3^2}{p^2}} \quad \text{(I.17)}$$

D'ou :

$$\text{ø} - \theta_1 = Atan2(\frac{d_3}{p} \pm \sqrt{1 - \frac{d_3^2}{p^2}}) \quad \text{(I.18)}$$

Pour finir, l'expression de θ_1 peut s'écrire :

$$\theta_1 = Atan2(p_x p_y) - Atan2\left(d_3 \pm \sqrt{p_x^2 + p_y^2 - d_3^2}\right) \quad \text{(I.19)}$$

L'équation (I.19) offre deux solutions potentielles basées sur le signe + ou –, à condition que la partie gauche de (I.11) soit déjà déterminée.

$$C_1 p + S_1 P_y = a_2 C_{23} - d_4 S_{23} + a_2 C_2 \quad \text{(I.20)}$$

$$-P_x = a_3 S_{23} + d_4 C_{23} + a_2 S_2 \quad \text{(I.21)}$$

En mettant les équations (I.21) et (I.12) sur un même plan et en les additionnant, nous arrivons à :

$$a_3 C_3 - d_4 S_3 = K \quad \text{(I.22)}$$

$$K = \frac{P_X^2 + P_Y^2 + P_Z^2 + a_2^2 + a_3^2 + d_3^2 + d_4^2}{2a_2} \quad \text{(I.23)}$$

L'équation (I.23) présente la même structure que (I.12) et n'est pas dépendante deθ_1En appliquant les mêmes méthodes que pour θ_1, on obtient :

$$\theta_3 = Atan2(a_3 d_4) - Atan2\left(K \pm \sqrt{a_3^2 + d_4^2 + K^2}\right) \quad \text{(I.24)}$$

Le symbole + ou - dans (I.24) mène à deux solutions distinctes pour θ_3.

Ainsi, l'équation (I.8) peut être reformulée en termes de θ_2 de la manière suivante :

$$[{}_3^0\mathrm{T}(\theta_2)]_6^0\mathrm{T} = {}_4^3\mathrm{T}(\theta_4) * {}_5^4\mathrm{T}(\theta_5) * {}_6^5\mathrm{T}(\theta_6) \quad \text{(I.25)}$$

Avec :

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1C_{23} & S_1C_{23} & -S_{23} & -a_2C_3 \\ -C_1S_{23} & -S_1S_{23} & -C_{23} & a_2S_3 \\ -S_1 & C_1 & 0 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{3}_{6}\mathbf{T} \tag{I.26}$$

$${}^{4}_{6}\mathrm{T} = {}^{4}_{5}\mathrm{T} * {}^{5}_{6}\mathrm{T} = \begin{bmatrix} C5C6 & -C5S6 & -S5 & 0 \\ S6 & C6 & 0 & 0 \\ S5C6 & -S5S6 & C5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{I.27}$$

$$\begin{aligned} C_1C_{23}P_x + S_1C_{23}P_y - C_{23}P_z - a_2C_3 = a_3 \\ -C_1S_{23}P_x + S_1S_{23}P_y - C_{23}P_z - a_2S_3 = d_4 \end{aligned} \tag{I.28}$$

Basées sur l'équation (I.28), les formules pour C_{23} et S_{23} se présentent ainsi :

$$\begin{aligned} S_{23} &= \frac{(-a_3 - a_3C_3)P_z + (C_1P_x + S_1P_y)(a_2S_3 - d_4)}{P_z^2 + (C_1P_x + S_1P_y)^2} \\ C_{23} &= \frac{(a_2S_3 - d_4)P_z + (a_3 + a_2C_3)(C_1P_x + S_1P_y)}{P_z^2 + (C_1P_x + S_1P_y)^2} \end{aligned} \tag{I.29}$$

Les dénominateurs étant égaux et positifs, nous allons résoudre la somme de θ_2 et θ_3 de la manière suivante :

$$\begin{aligned} \theta_{23} = Atan[(-a_3 - a_2C_3)P_z - (C_1P_x + S_1P_y)(d_4 - a_2S_3), (a_2S_3 - d_4)P_z \\ - (a_3 + a_2C_3)(C_1P_x + S_1P_y)] \end{aligned} \tag{I.30}$$

L'équation (I.30) fournit quatre valeurs deθ_{23}, en fonction des quatre combinaisons possibles des solutions pour $\theta_1$et θ_3. Cela donne ainsi quatre solutions possibles pour $\theta_2$calculées comme suit:

$$\theta_2 = \theta_{23} - \theta_3 \tag{I.31}$$

En équilibrant les éléments situés en position (1,3) et (3,3) de chaque côté de l'équation (I.20), on obtient :

$$r_{31}C_1C_{23} + r_{23}S_1C_{23} - r_{33}S_{23} = -C_4S_5 - r_{11}S_1 + r_{23}C_1 = S_4S_5 \tag{I.32}$$

tant que $S_5 \neq 0$ nous pouvons résoudre θ_4 comme suit :

$$\theta_4 = Atan2(-r_{13}S_1 + r_{23}C_1 - r_{13}C_1C_{23} - r_{23}S_1C_{23} + r_{33}S_{23}) \tag{I.33}$$

Lorsqu'il est établi que θ5 = 0, le manipulateur se trouve dans une configuration singulière, caractérisée par l'alignement des axes communs 4 et 6. Cet alignement induit le même mouvement du dernier maillon du robot. Dans ce contexte, ce qui devient pertinent est la somme ou la différence de θ4 et θ6. Cette situation peut être détectée en vérifiant si les deux arguments de la fonction Atan2 dans

(I.33) sont proches de zéro. Revisitions l'équation (I.8), celle-ci peut désormais être reformulée de manière à ce que tout le côté gauche se constitue uniquement comme une fonction des valeurs connues et de θ4 :

$$({}^{0}_{4}\mathrm{T})^{-1}\,{}^{0}_{6}\mathrm{T} = {}^{4}_{5}\mathrm{T}(\theta_5) * {}^{5}_{6}\mathrm{T}(\theta_6) \tag{I.34}$$

Avec : $\left(\mathbf{^0_4T}\right)^{-1}\mathbf{^0_6T}$ est représenté par :

$$\begin{bmatrix} C_1C_{23}C_4 + S_1S_4 & S_1C_{23}C_4 - C_1S_4 & -S_{23}C_4 & -a_2C_3C_4 + d_3S_4 - a_3C_3 \\ -C_1C_{23}S_4 + S_1C_4 & -S_1C_{23}S_4 - C_1C_4 & -S_{23}S_4 & a_2C_3S_4 + d_3C_4 - a_3S_4 \\ -C_1S_{23} & -S_1S_{23} & -C_{23} & a_2S_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{I.35}$$

L'équation $\mathbf{^4_6T}$, formulée précédemment dans l'équation (I.27), nous permet de procéder à une comparaison des éléments en position (1,3) et (3,3) des deux côtés de l'équation (I.34). Cela nous mène à obtenir ce qui suit :

$$\begin{aligned} -S_5 &= r_{13}(C_1C_{23}C_4 + S_1S_4) + r_{23}(S_1C_{23}C_4 - C_1S_4) - r_{33}(S_{23}C_4) \\ C_5 &= r_{13}(-C_1S_{23}) + r_{23}(-S_1S_{23}) + r_{33} - (C_{23}) \end{aligned} \tag{I.36}$$

En conséquence :

$$\theta_5 = Atan2(S_5, C_5) \tag{I.37}$$

Les valeurs de S_5et C_5 sont définies par l'équation (I.28).

En utilisant la même méthode pour l'équation (I.36), elle peut être réécrite sous la forme suivante :

$$({}^{0}_{4}\mathrm{T})^{-1}\,{}^{0}_{6}\mathrm{T} = {}^{5}_{6}\mathrm{T}(\theta_6) \tag{I.38}$$

En égalisant l'élément (3,1) et l'élément (1,1) de part et d'autre de (I.34), on obtient :

$$\theta_6 = Atan2(S_6, C_6) \tag{I.39}$$

Les signes + ou - présents dans les équations (I.19) et (I.24)conduisent à quatre solutions distinctes. En parallèle, le poignet du manipulateur génère quatre solutions supplémentaires. Pour chaque solution calculée précédemment, une solution inversée est obtenue par :

$$\theta'_4 = \theta_4 + 180^\circ \tag{I.40}$$

$$\theta'_4 = \theta_4 \tag{I.41}$$

$$\theta'_6 = \theta_6 + 180^\circ \tag{I.42}$$

5. LA Modélisation cinématique

Le modèle cinématique établit les liens entre les vitesses des articulations individuelles et les vitesses associées à un point donné dans la chaîne cinématique, souvent situé à l'extrémité de celle-ci. La cinématique directe facilite l'exécution de mouvements d'arcs précis et bien définis, en permettant un contrôle direct des rotations des articulations. En revanche, dans le cadre de la modélisation cinématique inverse, lorsque l'on souhaite atteindre un point spécifique dans l'espace, il est nécessaire de déduire la configuration géométrique, cela repose sur la détermination des angles des articulations.

5.1 Le Modèle cinématique direct (MCD)

Le modèle cinématique directe d'un robot manipulateur représente les vitesses des coordonnées d'opération en fonction des vitesses des articulations.

$$v = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J(q)\dot{q} \qquad \text{(I.43)}$$

Le vecteur v représente la vitesse de l'effecteur terminal, il détermine sa position et son orientation. Nous définissons également le vecteur de vitesse :

$$\dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \qquad \text{(I.44)}$$

D'ou w_x : la rotation autour de $\hat{X}$., w_y: la rotation autour de$\hat{Y}$ et w_z: la rotation autour de $\hat{Z}$.

5.2 Le Modèle cinématique inverse (MCI)

Le modèle d'inversion cinématique fournit les vitesses des articulations associées à une vitesse souhaitée pour l'organe terminal. Ce modèle est caractérisé par l'équation :

$$vJ^{-1}(q) = I\dot{q} \qquad \text{(I.45)}$$

Dans le cas du puma560, un robot à 6DDL, Jacobien se présente sous la forme d'une matrice carrée de dimension 6 par 6 et son rang est de 6.

$$v = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \frac{\delta x'}{\delta_{q'_1}} & \frac{\delta x'}{\delta_{q'_2}} & \frac{\delta x'}{\delta_{q'_3}} & \frac{\delta x'}{\delta_{q'_4}} & \frac{\delta x'}{\delta_{q'_5}} & \frac{\delta x'}{\delta_{q'_6}} \\ \frac{\delta y'}{\delta_{q'_1}} & \frac{\delta y'}{\delta_{q'_2}} & \frac{\delta y'}{\delta_{q'_3}} & \frac{\delta y'}{\delta_{q'_4}} & \frac{\delta y'}{\delta_{q'_5}} & \frac{\delta y'}{\delta_{q'_6}} \\ \frac{\delta z'}{\delta_{q'_1}} & \frac{\delta z'}{\delta_{q'_2}} & \frac{\delta z'}{\delta_{q'_1}} & \frac{\delta z'}{\delta_{q'_1}} & \frac{\delta z'}{\delta_{q'_1}} & \frac{\delta z'}{\delta_{q'_1}} \\ \frac{\delta w_x}{\delta_{q'_1}} & \frac{\delta w_x}{\delta_{q'_2}} & \frac{\delta w_x}{\delta_{q'_3}} & \frac{\delta w_x}{\delta_{q'_4}} & \frac{\delta w_x}{\delta_{q'_5}} & \frac{\delta w_x}{\delta_{q'_6}} \\ \frac{\delta w_y}{\delta_{q'_1}} & \frac{\delta w_y}{\delta_{q'_2}} & \frac{\delta w_y}{\delta_{q'_3}} & \frac{\delta w_y}{\delta_{q'_4}} & \frac{\delta w_y}{\delta_{q'_5}} & \frac{\delta w_y}{\delta_{q'_6}} \\ \frac{\delta w_z}{\delta_{q'_1}} & \frac{\delta w_z}{\delta_{q'_2}} & \frac{\delta w_z}{\delta_{q'_3}} & \frac{\delta w_z}{\delta_{q'_4}} & \frac{\delta w_z}{\delta_{q'_5}} & \frac{\delta w_z}{\delta_{q'_6}} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \quad \text{(I.46)}$$

Le Jacobien établit le lien entre la vitesse de l'effecteur et celles des articulations. Dans le domaine de la robotique, l'accent est mis sur le modèle cinématique inverse, car à partir d'un point cible dans l'espace, il est nécessaire de déterminer la configuration géométrique (angles et vitesses désarticulations).

En présence de matrices singulières, où det(J=0), il existe au moins une direction dans laquelle l'effecteur terminal peut être déplacé, indépendamment des vitesses.

Cela implique que le rang (J(q))<6, par exemple, si θ_5=0 (dans le cas d'un puma560), alors les paramètres θ_4 et θ_6 partagent la même direction, ce qui conduit à une réduction des degrés de liberté à 5 à cet endroit précis (un degré de liberté est perdu). Lorsque l'on modifie θ_4 ou θ_6, le nombre de degrés de liberté demeure à 5, tandis que pour θ_5 différent de 0, les 6 degrés de liberté sont maintenus [4].

6. Modélisation dynamique

6.1 L'équation de Lagrange applique au robot

La dynamique de n'importe quel manipulateur se décrit à travers un ensemble d'équations différentielles du second ordre. Ces équations établissent un lien entre le couple appliqué sur chaque articulation et les positions, vitesses ainsi que les accélérations des liaisons du robot. Afin de simplifier de nombreuses expressions, nous adoptons la convention de sommation introduite par Einstein : chaque fois qu'un indice apparaît deux fois ou plus dans un terme, cela implique que ces termes seront additionnés pour l'ensemble des valeurs possibles de cet indice, sans autre symbole. Tout au long de cette thèse, cette convention sera suivie dans les équations, sauf mention contraire. Parfois, pour plus de clarté ou pour éviter toute équivoque, le signe de sommation E peut être explicitement indiqué.[5]

$$\Gamma_i = \frac{d}{dt}\left(\frac{\partial l}{\partial q}\right) - \frac{\partial l}{\partial q} \quad \text{(I.47)}$$

Avec L = K – P ; L : lagrangien du système, K : énergie cinématique et P : énergie potentielle

6.2 La forme générale des équations dynamiques

L'énergie cinétique d'un système est une fonction quadratique des vitesses des articulations.

$$K = \frac{1}{2}\dot{q}^T A\dot{q} \quad \text{(I.48)}$$

Ou A est la matrice (n×n) de l'énergie cinétique, également appelée matrice d'inertie du robot, qui possède une symétrie et une définition positive. Ces éléments dépendent des variables articulatoires.

L'énergie potentielle dépend des variables articulaires q, et le couple n peut être extrait des équations (I.50) et (I.51) sous la forme suivante.

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q}) + Q(q) \quad \text{(I.49)}$$

Avec $C(q,\dot{q})$: vecteur de dimension (n×1) représentant les forces de Coriolis et centrifuges :

$$C\dot{q} = A\dot{q} - \frac{\partial K}{\partial q}\cdot \quad \text{(I.50)}$$

$Q = [Q1 \dots Qn]^T$:vecteur de couple/force de gravité

6.3 Calculs de l'énergie

La cinétique du système est décrite par :

$$K = \sum_{j=1}^{n} K_j \quad \text{(I.51)}$$

Ou K_j représente l'énergie cinétique du corps Cj, qui s'exprime par :

$$K_j = \frac{1}{2}\left(w_j^T\, I_{Gj}\, w_j + M_j\, V_{Gj}^T\, V_{Gj}\right) \quad \text{(I.52)}$$

Etan donne que (figureI.1)

$$V_{Gj} = Vj + w_j \times S_j \quad \text{(I.53)}$$

Et tanque

$$J_j = I_{Gj} - M_j \widehat{S_j}\widehat{S_j} \quad \text{(I.54)}$$

La relation (I.55) évolue pour devenir :

$$K_j = \frac{1}{2}\left[w_j^T J_j\, w_j + M_j\, V_j^T\, V_j + 2MS_j^T\left(V_j \times w_j\right)\right] \qquad \text{(I.55)}$$

La relation (I.52) n'est pas linéaire en ce qui concerne les paramètres du vecteur S_j, contrairement à la relation (I.55), qui est linéaire par rapport aux éléments de \$M_j$, MS_jet \$J_j$, désignés sous le nom de paramètres inertiels standard. Le calcul de Vj et de w_j s'effectue à l'aide des équations de composition des vitesses (voir figureI.1) :

$$w_j = w_{j-1} \times \bar{\sigma}_j \dot{q}_j\, a_j \qquad \text{(I.56)}$$

$$V_j = V_{j-1} \times L_j + \sigma_j\, \dot{q}\, a_j \qquad \text{(I.57)}$$

Pour un robot dont la base est fixe, nous considérons des conditions initiales spécifiant que V0 = 0 et.w0=0.

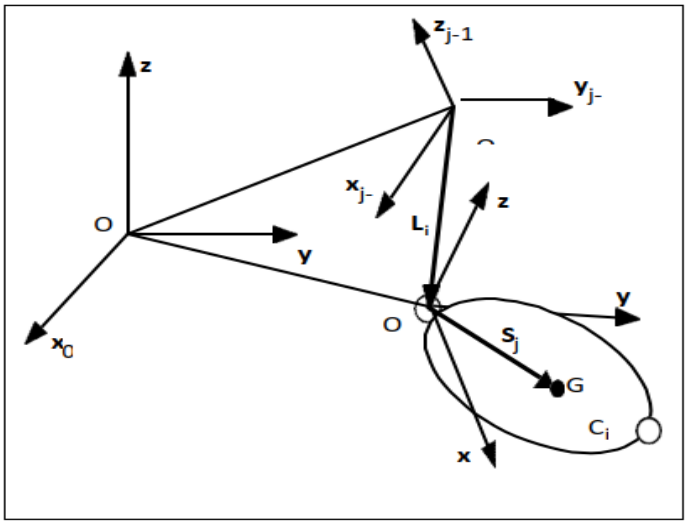


Figure I. 2:composition des vitesses.

Dans l'équation (I.55), il est essentiel que tous les éléments soient exprimés dans le même repère. La méthode la plus simple consiste à les formuler dans le repère Rj. Nous allons donc réécrire les équations (I.55), (I.56)et (I.57)qui fournissent respectivement Ej, jwj et jV_j, de la manière suivante :

$$K_j = \frac{1}{2}\left[\left[{}^j w_j^T J_j\, w_j\, {}^j\dot{w}_j + M_j\, {}^j\dot{V}_j^T\, V_j + 2\,{}^j MS_j^T\left(jV_j \times {}^j\dot{w}_j\right)\right]\right. \qquad \text{(I.58)}$$

Les termes Jj et ${}^{j}_{.}MS_j$ sont constants. Afin de simplifier la notation, nous les désignerons respectivement par Jj et MSj .

L'énergie potentielle peut être formulée comme suit :

$$P = \sum_{j=1}^{n} P_j = \sum_{j=1}^{n} P_j - M_j g^T \left(L_{0,j} + S_j\right) \tag{I.59}$$

$L_{0,j}$ désigne le vecteur dont l'origine est le point O et l'extrémité est le point Oj. En projetant les vecteurs de cette relation dans R0, on obtient alors :

$$P_j = -M_j {}^{0}_{,}g^T \left({}^{0}_{.}pj + {}^{0}_{.}Aj \, {}^{j}_{.}S_j\right) \tag{I.60}$$

Cette expression peut être reformulée sous une forme linéaire en Mj par rapport aux éléments de ${}^{j}_{.}MS_j$:

$$P_j = -{}^{0}_{,}g^T \left(M_j {}^{0}_{.}pj + {}^{0}_{.}Aj \, {}^{j}_{.}MS_j\right) = -[{}^{0}_{,}g^T \quad 0] \, {}^{0}_{,}Tj \begin{pmatrix} {}^{j}_{.}MS_j \\ M_j \end{pmatrix} \tag{I.61}$$

Les énergies cinétiques et potentielles étant directement proportionnelles aux paramètres inertiels, il en découle que le modèle dynamique présente également cette linéarité.

6.4 Les équations différentielles du PUMA 560

Pour commander ou simuler les robots, il est essentiel de développer le modèle dynamique du robot. Cela implique de formuler les équations qui relient les couples et les forces exercées par les actionneurs, ainsi que celles résultant des interactions avec l'environnement, aux mouvements des axes. Ce processus conduit à un système d'équations différentielles non linéaires dépendant de la variable q.

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + u_{mo} = u \tag{I.62}$$

$q \in \Re^n$: vecteur représentant les positions articulaires

$\dot{q} \in \Re^n$: vecteur des vitesses articulaires

$\ddot{q} \in \Re^n$: vecteur des accélérations articulaires

$u(t) \in \Re^n$: vecteur décrivant les forces et/ou couples moteurs

$u_{mo}(t) \in \Re^n$: vecteur des couples provoqués par une charge ponctuelle

$M(q) \in \Re^{mxn}$: matrice symétrique définie positive associée aux accélérations inertielles, où chaque élément M_{ij}est l'inertie de l'articulation i^{eme} par rapport à l'articulation j^{eme}

$V_m(q,\dot{q}) \in \Re^n$: matrice des forces et/ou couples liés aux accélérations de Coriolis et centrifuges $G(q) \in \Re^n$: vecteur des forces et/ou couples résultant de l'effet de la gravité

Le modèle du robot manipulateur se distingue par les caractéristiques suivantes :

- La matrice $M(q)$est symétrique, définie positive et limitée, et son inverse non seulement existe mais est aussi limitée.
- La matrice $M(q) - 2V_m(q,\dot{q})$ satisfait l'égalité suivante pour un vecteur X non nul.

$$X^{\Gamma}[M(q) - 2V_m(q,\dot{q})]X \tag{I.63}$$

Les éléments des matrices et vecteurs $M(q)$, $V_m(q,\dot{q})\dot{q}$ et $G(q)$ sont définis de la manière suivante :

L'inertie de la matrice du robot :

$$M(q) = \begin{bmatrix} I_1 + I_2C_{23}^2 + I_3C_2^2 + I_4C_2C_{23} & I_5S_{23} + I_6S_2 & I_5S_{23} \\ I_5S_{23} + I_6S_2 & I_7 + I_4C_3 & I_8 + 0.5I_4C_3 \\ I_5S_{23} & I_8 + 0.5I_4C_3 & I_9 \end{bmatrix} \tag{I.64}$$

La représentation des forces et/ou des couples causées par les accélérations de Coriolis et centrifuges :

$$V_m(q,\dot{q})\dot{q} = \begin{bmatrix} -(2(I_3S_2C_2 + I_2S_{23}C_{23}) + I_4(C_2S_{23} + S_2C_{23}))\dot{q}_1\dot{q}_2 - (2I_2S_{23}C_{23} + \\ I_4C_2C_{23})\dot{q}_1\dot{q}_3 + (I_6C_2 + I_5C_{23})\dot{q}_2^{\,2} + (2I_5C_{23})\dot{q}_2\dot{q}_3 + (I_5C_{23})\dot{q}_3^{\,2} \\ (I_3S_2C_2 + I_2S_{23}C_{23} + 0.5I_4(C_2S_{23} + S_2C_{23}))\dot{q}_1^{\,2} - (I_4S_3)\dot{q}_2\dot{q}_3 - (0.5I_4S_3)\dot{q}_3^{\,2} \\ (I_2S_{23}C_{23} + 0.5I_4C_2S_{23})\dot{q}_1^{\,2} + (0.5I_4S_3)\dot{q}_2^{\,2} \end{bmatrix} \tag{I.65}$$

Le vecteur des forces et des moments résultant des actions gravitationnelles :

$$G(q) = \begin{bmatrix} 0 \\ -(m_2L_2 + 0.5m_2L_2)gC_2 - 0.5m_3L_3gC_{23} \\ -0.5m_3L_3gC_{23} \end{bmatrix} \tag{I.66}$$

En utilisant les notations ci-après :

$$\begin{aligned} C_i &= cos(q_i), \quad & C_{ij} &= cos\,(q_i + q_j) \\ S_i &= sin\,(q_i), \quad & S_{ij} &= sin\,(q_i + q_j) \end{aligned} \tag{I.67}$$

Et

$$u = \begin{bmatrix} u1 \\ u2 \\ u3 \end{bmatrix} \tag{I.68}$$

Le vecteur des forces et/ou des couples motorisés :

$$\begin{aligned}
I_1 &= I_{yy1} + I_{xx2} + m_2 d_2 (d_2 + e) + m_3 d_2^2 + I_{M1} \\
I_2 &= I_{yy3} - I_{xx3} + I_{yyt} - I_{xxt} + m_t l_3^2 \\
I_3 &= I_{yy3} - I_{xx2} + m_3 l_2^2 + m_t l_2^2 \\
I_4 &= m_3 l_2 \, l_3 + 2 m_t l_2 \, l_3 \, . \\
I_5 &= 0.5 m_3 l_3 \, d_3 + m_t l_3 \, d_3 \, . \\
I_6 &= 0.5 m_2 l_2 \, (d_3 + e) + m_3 l_2 \, d_2 + m_t l_2 \, d_2 \, . \\
I_7 &= I_{zz2} + I_{zz3} + m_3 l_2^2 + I_{zzt} + m_t (l_2^2 + l_3^2) + l_{M2} \, . \\
I_8 &= I_{zz3} + I_{zzt} + m_t l_3^2 \, . \\
I_9 &= I_8 + l_{M3} \, .
\end{aligned} \tag{I.69}$$

Avec l'utilisation de l_{Mi} $i = 1{,}3$ on fait référence aux moments d'inertie des divers moteurs.

I_{xxt} , I_{yyt}, I_{zzt} . Ils correspondent aux instants d'inertie totale par rapport aux axes majeurs de l'effecteur.

Effet de la charge : Le vecteur des couples additionnels, noté mo u, illustre l'effet de la charge. Il est déterminé par la matrice jacobienne, qui représente la dérivée du vecteur position de l'effecteur.

$$J_j(q) = \frac{\partial p}{\partial q_j} \tag{I.70}$$

Spécifiquement pour le robot de type PUMA, nous avons :

$$J\,(q) = [\, J_1(q) \quad J_2(q) \quad J_3(q) \,] \tag{I.71}$$

$$p = [\, C_1(l_2 \, C_2 + l_3 \, C_{23}) - d_2 S_1 \quad S_1 \, (l_2 \, C_2 + l_3 \, C_{23}) + d_2 C_1 \quad -(l_2 \, C_2 + l_3 \, C_{23})] \, T \tag{I.72}$$

La matrice jacobienne est déterminée comme suit :

$$J\,(q) = \begin{bmatrix} -S_1 \, (l_2 \, C_2 + l_3 \, C_{23}) - d_2 C_1 & -C_1 (l_2 \, S_2 + l_3 \, S_{23}) & -C_1 (l_3 \, S_{23}) \\ C_1 (l_2 \, C_2 + l_3 \, C_{23}) - d_2 S_1 & -S_1 \, (l_2 \, S_2 + l_3 \, S_{23}) & -S_1 (l_3 \, S_{23}) \\ 0 & -(l_2 \, C_2 + l_3 \, C_{23}) & -(l_3 \, C_{23}) \end{bmatrix} \tag{I.73}$$

Le couple induit par la charge ponctuelle sera :

$$u_{m_o} = m_o J^T(q)\left[J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} + g\right] \tag{I.74}$$

Avec

$$g = [0 \;\; 0 \;\; 9.81]^T \tag{I.75}$$

7. Conclusion

Dans ce chapitre, nous avons procédé à la modélisation complète du robot PUMA 560, en abordant successivement les caractéristiques techniques principales, la modélisation cinématique tel que la modélisation directe et inverse et la modélisation dynamique. Ce modèle dynamique décrit les relations entre les couples moteurs appliqués aux articulations et les mouvements résultants du robot. Cette représentation est indispensable pour l'analyse des performances et pour la conception des lois de commande visant à assurer un suivi précis des trajectoires souhaitées. Le modèle dynamique présenté dans ce chapitre servira de base pour le reste de ce manuscrit, notamment pour la conception et la synthèse des lois de commande, ainsi que pour les simulations numériques permettant de prédire et d'optimiser les performances du robot dans différents scénarios d'exploitation.

Chapitre II : les techniques de commande intelligente

Chapitre2 : les techniques de commande intelligente.

1. Introduction

Dans le domaine du contrôle automatique, la majorité des systèmes réels présentent un comportement intrinsèquement non linéaire. Ces non-linéarités résultent de la nature physique des processus, des incertitudes, des perturbations externes ou encore des limites opérationnelles des actionneurs.

La meilleure approche pour la commande des systèmes non linéaire est La commande par mode glissant, constitue une approche robuste qui repose sur l'idée de modifier la structure du contrôleur en fonction de l'état du système afin d'atteindre la réponse souhaitée. Le contrôle produit par ce type de commande est de nature binaire. Dans cette méthode, l'état du système détermine le critère de commutation de l'élément de contrôle. Le principe est de segmenter l'espace d'état à l'aide d'une frontière décisionnelle désignée comme surface de glissement. Mais cette commande possède des inconvénients.

Pour surmonter ces défis, l'ingénierie des systèmes de commande a vu émerger des méthodes de commande non linéaires intelligentes, qui intègrent des approches issues de l'intelligence artificielle, telles que les réseaux de neurones, les logiques floues et les algorithmes d'apprentissage automatique. Ces méthodes visent à améliorer les performances des systèmes tout en tenant compte des incertitudes et de la dynamique complexe.

Ce chapitre introduit les concepts de base des commandes non linéaires intelligentes. Tel que la commande du réseau de neurones et la commande floue en commençant par une brève historique puis son architectures et fonctionnement enfin en introduisant l'intégration de ces commandes avec la commande de glissement.

2. La Commande par mode glissant

La commande par mode glissant représente l'une des approches de commande non linéaire qui se distingue par sa solidité et son efficacité. Cette méthode se fonde sur l'utilisation de fonctions discontinues pour modifier la structure du système dynamique de façon à ce quel vecteur d'état suive une trajectoire (x)=0 dans l'espace d'état. Les principaux objectifs de cette approche de commande sont :

* Élaborer une surface de glissement de telle sorte que toutes les trajectoires d'état du système se conforment à un comportement souhaité en termes de suivi, de régulation et de stabilité;
* Établir une loi de commande ou de commutation, capable d'attirer toutes les trajectoires d'état vers la surface de glissement et de les maintenir sur cette dernière ;[6]

2.1 Les surfaces de glissement non linéaires

L'application des surfaces linéaires dans lacération des lois de commande par modes glissants est considérée comme adéquate par les auteurs en matière de stabilité[7]. Cependant, la dynamique en générée par ce choix est assez le nette pour remédier à ce problème, nous pouvons recourir à des surfaces de glissement non linéaires.

On désigne s_i comme une surface de glissement non linéaire de premier ordre, définie par :

$$s_i = \dot{e}_i + \Lambda(e_i) \tag{II.1}$$

avec $\dot{e}_i = x_{di} - x_i$ Représente l'erreur de suivi et $\Lambda(.)$ est une fonction de classe C_1.

On définit une fonction de Lyapunov, V_i, comme suit :

$$V_i = \frac{1}{2}\, e_i^2 \tag{II.2}$$

Donc, $\dot{V}_i$ est la dérivée temporelle

$$\dot{V}_i = -e_i \Lambda(e_i) \tag{II.3}$$

D'après le théorème de Lyapunov, le point zéro est asymptotiquement stable si et seulement si la dérivée $\dot{V}_i$est strictement inférieure à zéro. Il s'agit donc de déterminer la fonction $\Lambda(.)$de manière que

$$e_i \Lambda(e_i) > 0 \; \forall e_i \neq 0 \tag{II.4}$$

Cette condition assure que l'erreur de poursuite e_i tend asymptotiquement vers zéro au fur et à mesure que le temps t tend vers l'infini. Dans un cadre linéaire où $\Lambda(e_i)$est défini comme $k_i e_i$, l'erreur e_i décroît exponentiellement vers zéro, ce qui signifie que $e_i(t)$ devient inférieur à $e_i(0)$ lorsque t augmente vers l'infini. Dans notre étude, nous avons choisi de définir la fonction $\Lambda(.)$comme une fonction sigmoïde.

La fonction $\Lambda(.)$ non linéaires utilisée est du type **sigmoïde :**

$$\begin{cases} \Lambda_i(x) = \dfrac{2\eta_i}{1+e^{-x}} - \eta_i \\ \dfrac{d\Lambda_i(x)}{dx} = \dfrac{1}{2\eta_i}[\eta_i^2 - \Lambda_i(x)] \end{cases} , \eta_i \in \Re \quad \text{(II.5)}$$

2.2 Mode de glissement pour un système non linéaire

Prenons en compte un système non linéaire :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(X) + g(X).u \\ y = x_1 \end{cases} \quad \text{(II.6)}$$

où X = [x1, x2]^T est le vecteur d'état, u ∈ ℝ représente la commande, et f et g sont deux fonctions continues et suffisamment dérivables

Pour ce système, une surface de glissement non linéaire est définie de manière à :

$$\begin{cases} s = \dot{e} + \Lambda_i(x) \\ \Lambda(x) = \dfrac{2}{1+e^{-x}} - 1 \end{cases} \quad \text{(II.7)}$$

Comme précédemment démontré, l'erreur e = x1 - yr converge vers zéro lorsque s = 0, selon la dynamique établie par les paramètres de la fonction Λ(·). Par ailleurs, pour garantir la stabilité du point d'équilibre (s = 0), nous définissons V(s), une fonction candidate de Lyapunov, de manière à ce que :

$$V = \frac{1}{2}s^2 \quad \text{(II.8)}$$

Il est évident que cette fonction est définie comme étant positive, où V(s = 0) = 0 et V(s) > 0 pour tout s différent de zéro. Par conséquent, conformément à la stabilité au sens de Lyapunov, le point d'équilibre (s = 0) est considéré comme asymptotiquement stable si, et seulement si, la loi de commande u(t) satisfait l'inégalité.

$$\frac{dv(s)}{dt} < 0 \quad \text{(II.9)}$$

$$\frac{dv(s)}{dt} = -s^T.\bar{K}sign(s) \quad \text{(II.10)}$$

Où $K > 0$ représente un gain et $sign(.)$ est la fonction signe, définie de la manière suivante :

$$\text{sing}(s) = \begin{cases} +1 & s > 0 \\ 0 & s = 0 \\ -1 & s < 0 \end{cases} \quad \text{(II.11)}$$

En mettant la dérivée de (II.10) en équivalence avec (II.11), on obtient :

$$s.\frac{ds}{dt} = -s.\bar{K}sign(s) \quad \text{(II.12)}$$

Avec

$$\begin{aligned} \frac{ds}{dt} &= \ddot{e} + \dot{e}.\frac{d\Lambda(x)}{dx}\Big|_{x=e} \\ &= (\dot{x_2} - \ddot{y}_r) + \dot{e}.\frac{1}{2}.(1 - \Lambda(e)^2) \\ &= f(X) + g(X).u - \ddot{y}_r + \dot{e}.\frac{1}{2}.(1 - \Lambda(e)^2) \\ \frac{ds}{dt} &= (f(X) - \ddot{y}_r) + \dot{e}.\tfrac{1}{2}.(1 - \Lambda(e)^2) + g(X).u \end{aligned} \quad \text{(II.13)}$$

En substituant l'équation (II.13) dans l'équation (II.12) nous obtenons :

$$u = -g(X)^{-1}.\left((f(x) - \ddot{y}_r) + \dot{e}.\tfrac{1}{2}.(1 - \Lambda(e)^2)\right) - g(X)^{-1}.\bar{K}\ sign(s) \quad \text{(II.14)}$$

Ou

$$u(t) = u_{eq} + \Delta_u(t) \quad \text{(II.15)}$$

Avec

$$\begin{cases} u_{eq} = -g(X)^{-1}.\left((f(x) - \ddot{y}_r) + \dot{e}.\dfrac{1}{2}.(1 - \Lambda(e)^2)\right) \\ et \\ \Delta_u = -g(X)^{-1}.\bar{K}\ sign(s) = -\bar{K}\ sign(s), K > \left|\dfrac{\bar{K}}{g(X)}\right| \end{cases} \quad \text{(II.16)}$$

La commande par mode de glissement permet de transformer un système d'équations différentielles d'ordre n en une équation différentielle d'ordre (n-1) pour les systèmes SISO, et en une équation différentielle d'ordre (n-m) pour les systèmes MIMO. Dès que le régime glissant est atteint, cette nouvelle équation dynamique est représentée par l'équation différentielle suivante :

$$\frac{ds}{dt} = 0 \qquad \text{(II.17)}$$

La résolution de l'équation (II.55) en fonction de la commande souhaitée u_d nous permet de décrire :

$$\frac{ds\,(x_d, u_{d,} t)}{dt} = 0 \qquad \text{(II.18)}$$

En prenant en considération ce qui est énoncé au point (II.51), il est possible d'affirmer que :

$$\frac{ds}{dt} = (f(X) - \ddot{y}_r) + \dot{e}.\frac{1}{2}.(1 - \Lambda(e)^2) + \; g(X_d)., u_{d,} = 0 \qquad \text{(II.19)}$$

Finalement, la commande souhaitée, correspondant à la dynamique inverse du système, s'exprime comme suit :

$$u_d = \; -g(Xd)^{-1}.\left((f(Xd) - \ddot{y}_r) + \dot{e}.\frac{1}{2}.(1 - \Lambda(e)^2)\right) \qquad \text{(II.20)}$$

Lorsque le système est en régime de glissement, les deux termes $f(Xd))$ et $g(Xd)$ peuvent être substitués dans l'équation (II.58**)** par l'état réel $f(X)$et $g(X)$.Ainsi, dans le régime de glissement, une similarité existe entre la commande équivalente et la dynamique inverse du système (voir figure II.1)

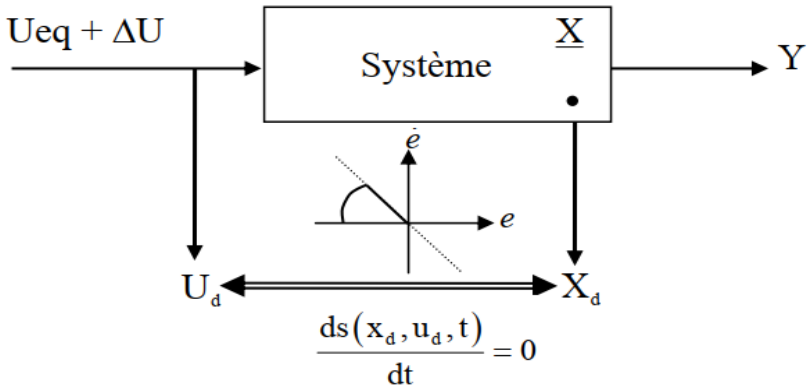


Figure II. 1:La commande équivalente et la dynamique inverse du système.

3. La logique floue

La logique floue (en anglais fuzzy logic) est une variante de la théorie mathématique de la logique booléenne. La théorie classique des ensembles est généralisée à l'utilisation des ensembles flous. En À travers l'introduction de la notion de degré dans la vérification d'une condition, une

condition peut ainsi être vérifiée. La logique floue permet de raisonner de manière flexible, ce qui permet d'être dans un état autre que vrai ou faux. Il est extrêmement bénéfique, car il permet de tenir compte des imprécisions et des incertitudes. Celui-ci La technique est solide et ne nécessite aucune entrée précise. Les systèmes instables peuvent tolérer Différentes catégories d'entrées, telles que des informations vague, déformées ou imprécises.

On a introduit le concept de logique floue avec la théorie des ensembles flous. Le mathématicien azerbaïdjanais iranien Lotfi Zadeh a été créé en 1965. Néanmoins, la logique confuse avait. Depuis les années 1920, elle a été étudiée en tant que logique à valeurs infinies, en particulier par Tarski et Łukasiewicz.

3.1 Ensembles flous

Dans la théorie des ensembles conventionnels, une chose appartient ou n'appartient pas à un certain ensemble. Toutefois, dans la réalité, il est rare de rencontrer des choses dont le statut est précisément défini. Par exemple, où est exactement la différence entre une personne grande et une autre de grandeur moyenne ? C'est à partir de ce genre de constatation que Zadeh a développé sa théorie. Il a défini les ensembles flous comme étant des termes linguistiques du genre : zéro, grand, négatif, petit… ces termes existent aussi dans les ensembles conventionnels. Cependant, ce qui différencie ces deux théories sur les ensembles provient des limites des ensembles. Dans les ensembles flous, il est permis qu'une chose appartienne partiellement à un certain ensemble ; ceci s'appelle le degré d'appartenance. Dans les ensembles conventionnels, le degré d'appartenance est 0 ou 1 alors que dans la théorie des ensembles flous, le degré d'appartenance peut varier entre 0 et 1 (on parle alors de fonction d'appartenance μ). [8]

Exemple : considérons une variable x (la taille) et un univers U (les individus), un ensemble flou A est caractérisé par une fonction d'appartenance $\mu_{(x)}$ qui illustre le niveau selon lequel l'élément x est associé à A. Voici un bref exemple d'explication :

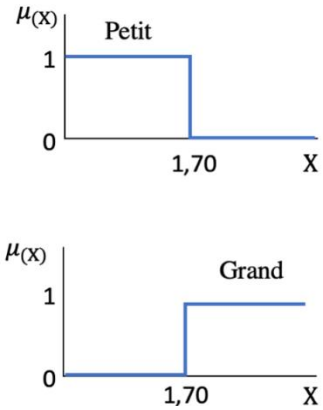


Figure II. 2:la logique Classique

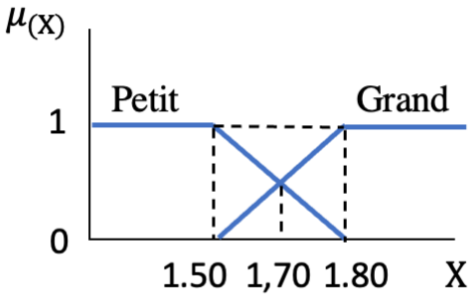


Figure II. 3 : la logique floue.

$$\mu_{(x)} = \begin{Bmatrix} 1, si\ x \in 1 \\ 0, si\ non \end{Bmatrix}$$

$$\mu_{(x)} = \in [0,1]$$

On parle de variable linguistique « La taille » et de valeur linguistique « Petit et Grand » Celle-ci associé avec la fonction d'appartenance $\mu_{(x)}$ constitue des ensembles flous.

3.2 La commande floue et les systèmes experts flous

Le problème de régulation est formulé par un ensemble de règles qui définissent en termes linguistiques (grâce à des adjectifs qualificatifs) la façon dont les paramètres de contrôle d'un système à gouverner doivent être modifiés (selon certaines observations spécifiques du système) pour atteindre un état précis de ce dernier. Plusieurs techniques sont disponibles pour appliquer la théorie de la logique floue à un problème de contrôle-commande. Celle que nous décrivons ici est connue sous le nom de Mamdani.[9]

Elle passe par un processus en quatre phases : la fuzzification, l'établissement des règles de contrôle, la mise en place des mécanismes d'inférence des règles floues et enfin, la défuzzification.

a. La Fuzzification

La fuzzification est le mécanisme qui permet de transformer une valeur d'entrée précise (quantité physique) en une valeur indéfinie (variable linguistique) par l'établissement des fonctions d'appartenance pour chacune des variables d'entrée. Bien qu'il existe différents types de courbes qui pourraient être Selon la littérature, les formes gaussiennes, triangulaires et trapézoïdales sont celles qui sont le plus souvent employées dans le processus de fuzzification. Bien qu'il n'y ait pas de

directives strictes pour cette sélection, nous prenons l'exemple de la fuzzification qui suit. [10], [11], [12]

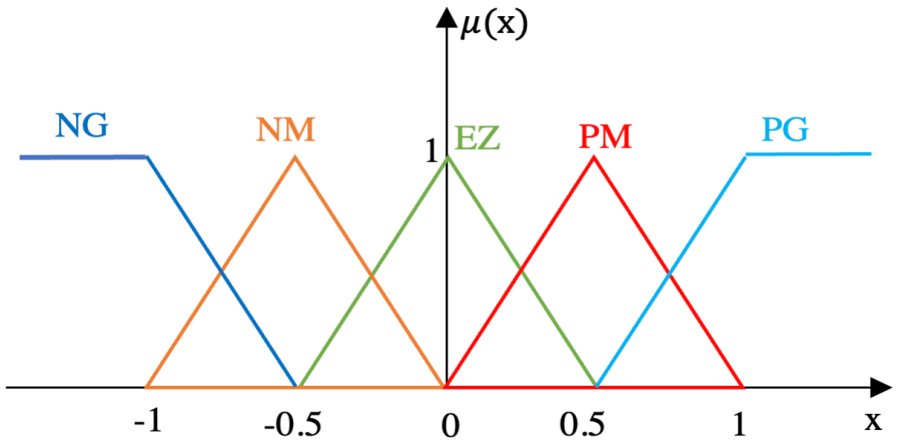


Figure II. 4 : Ensemble pour 5 règles floues**.**

Ici, x représente une variable linguistique d'entrée (normalisée sur un univers de discours [-1,1]) qui correspond en réalité à 5 sous-ensembles flous définis par des fonctions d'appartenance (variables linguistiques) de la manière suivante :

- NG: négatif grand, NM: négatif moyen, EZ: environ zéro, PM: positif moyen, PG: positif grand.

En testant maintenant ces fonctions d'appartenance, nous obtenons :

$$\mu_{\mathrm{NG}(x)} = \begin{cases} 1 & si \quad x \le -1 \\ -2x - 1 & si \quad -1 \le x \le -0.5 \\ 0 & , \quad ailleurs \end{cases} \qquad \text{(II.21)}$$

$$\mu_{\mathrm{NM}(x)} = \begin{cases} 2x + 2 & si \quad -1 < x \le -0.5 \\ -2x & si \quad -0.5 < x \le 0 \\ 0 & , \quad ailleurs \end{cases} \qquad \text{(II.22)}$$

$$\mu_{\mathrm{EZ}(x)} = \begin{cases} 2x + 1 & si \quad -0.5 < x \le 0 \\ -2x + 1 & si \quad 0 < x \le 0.5 \\ 0 & , \quad ailleurs \end{cases} \qquad \text{(II.23)}$$

$$\mu_{\mathrm{PM}(x)} = \begin{cases} 2x & si \quad 0 < x \le 0.5 \\ -2x + 2 & si \quad 0.5 < x \le 1 \\ 0 & , \quad ailleurs \end{cases} \qquad \text{(II.24)}$$

$$\mu_{\mathrm{PG}(x)} = \begin{cases} 2x - 1 & si \quad 0.5 < x \le 1 \\ 1 & si \quad x > 1 \\ 0 & , \quad ailleurs \end{cases} \qquad \text{(II.25)}$$

Les règles de contrôle

il s'agit d'un ensemble de règles linguistiques garantissant la supervision du processus. La compréhension du processus, qu'elle soit physique ou basée sur les témoignages d'experts, permet de réfléchir en temps réel et de compléter des tableaux de règles.

Le processus d'inférence dans les règles floues

Il s'agit d'un processus décisionnel qui, en utilisant le bloc de règles, prend une décision. Plusieurs stratégies d'inférence sont disponibles.

Exemple : Mamdani (Max, Min.). Larsen (Max, Production). Sugeno (Somme, Produit).

b. La défuzzification

Les techniques d'inférence génèrent une fonction d'appartenance pour la variable de sortie, ce qui constitue une information floue nécessitant conversion en une dimension physique.

On trouve diverses techniques de défuzzification.[13]

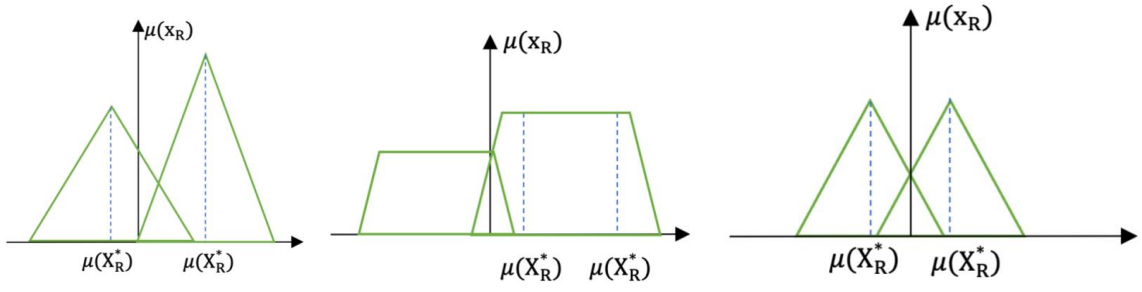
- **Méthode de maximum**

Ici, la sortie représente l'abscisse du pic de la fonction d'appartenance résultante. On peut exprimer

$$S = X_R^* \; tq. \max\left(\mu_{res}(X_R)\right) = \mu_{res}(X_R^*) \qquad \text{(II.26)}$$

Ou S représente la sortie du RLF.

On peut envisager trois scénarios possibles comme suit :



En conclusion, cette approche est simple, rapide et aisée, néanmoins elle présente des ambiguïtés et une discontinuité dans le résultat.

- **Méthode du moyen des maximums**

Si plusieurs sous-ensembles flous présentent une hauteur maximale identique, nous effectuons leur moyenne arithmétique. Cette démarche permet de réduire la confusion liée à la méthode des maximas.

Dans ce contexte, on pourrait formuler :

$$S = X_R^* = \frac{\int U' X_R \mu_{res}(X_R) dX_R}{\int U' \mu_{res}(X_R) dX_R} \quad \text{(II.27)}$$

Où : $U' \subset U : X_R \in U' si\ \mu_{res}(X_R) = \mu_{max}(\mu_{res}(X_R))$

- **Méthode centroïde (centre de gravité)**

Le résultat correspond à la coordonnée x du centre de gravité de la surface de la fonction produite par le bloc d'inférence. On peut évaluer cette abscisse de deux façons :

- Nous formons l'union des sous-ensembles flous issus des sorties afin de déterminer le centroïde global.

On considère chaque sous-ensemble flou individuellement et on détermine son centroïde, puis on procède à la moyennisation de tous les centroïdes.

On peut déterminer la sortie de la manière suivante :

$$X_R^* = \frac{\int_{-1}^{1} X_R f(X_R) dX_R}{\int_{-1}^{1} f(X_R) dX_R} \quad \text{(II.28)}$$

$$X_R^* = \frac{\sum_{i=1}^{n_r} \mu_{c_i} X_{G_i} S_i}{\sum_{i=1}^{n_r} \mu_{c_i} S_i} \quad \text{(II.29)}$$

Avec :

X_{G_i}: l'abscisse du centre de gravité du sous-ensemble flou de la sorite correspondante à la iéme règle.

S_i : est la surface du sous-ensemble flou de la sorite correspondante à la i éme règle.

n_r : est le nombre des règles.

- **Méthode de la somme pondérée**

C'est un compromis entre les 2 méthodes précédentes. Calculant individuellement les sorties relatives à chaque règle selon le principe de la moyenne du maximum, puit on réalise leur moyenne pondérée. La sortie est calculée par :

$$X_R^* = \frac{\sum_{i=1}^{nr} \mu_{c_i} X_{R_i}}{\sum_{i=1}^{nr} \mu_{c_{iMax}}} \quad \text{(II.30)}$$

$\mu_{c_{iMax}}$: le maximum de la fonction de sortie de la règle i et X_{R_i} est son abscisse.

Obtention des règles de commande

Commande ambiguë. Voici les méthodes les plus courantes pour établir les règles de commande :

• Consultation auprès des spécialistes afin qu'ils partagent leur expertise ou celle de l'opérateur.

Qualifiés pour qu'ils puissent expliquer leurs compétences. L'enquête peut être réalisée soit par entrevue verbale, soit par formulaire.

• Surveillance des opérations de contrôle effectuées par l'opérateur certifié, et création de correspondances entre les données d'entrée et de sortie associées au processus supervisé.

• Formation sur les règles de commande basée sur l'ajustement aux objectifs de la commande à partir d'expériences.

• Établissement d'un modèle flou du processus contrôlé, basé sur une description verbale de ses caractéristiques.

3.3 La logique floue dans la commande à structure variable

Permettant de résoudre des problèmes à la manière d'un être humain. Les lois floues reposent sur des fonctions d'appartenance, créées à partir de l'expérience humaine. Cette approche est particulièrement intéressante pour contrôler des systèmes dépourvus de modèles mathématiques acceptables pour la synthèse de contrôleurs classiques, là où d'autres méthodes échouent. Cependant, le contrôle flou présente aussi plusieurs défis, notamment le développement de méthodes garantissant stabilité et robustesse. Pour surmonter les problèmes associés à la commande par modes glissants, comme le phénomène de réticence et l'absence d'une méthode d'analyse de stabilité, et en raison des

similarités entre ces deux systèmes, la commande "Floue-glissante" a vu le jour dans plusieurs recherches. Cette approche combine les meilleures caractéristiques des deux méthodes, et elle a prouvé son efficacité en éliminant le broutement tout en offrant une méthode de conception simple et élégante pour les contrôleurs flous.

3.4 Présentation d'une commande Floue-glissante

Dans cette section, nous présentons un système de contrôle flou aligné avec la théorie des systèmes à structure variable, dans le but d'assurer la stabilité. Cette perspective innovante repose sur une interprétation spécifique des règles de contrôle flou. Habituellement, une règle est représentée comme une relation floue de la forme suivante. [14]

$$R^i \; si \; (A_i, B_i) \; alors \; C_i \tag{II.31}$$

où $A_i \; et \; B_i$ désignent des quantités floues représentant les mesures du processus, et Ci est une quantité floue qui représente le signal de commande. On constate que la règle R^i constitue une "structure" du système de contrôle flou, et elle évolue en fonction des états du processus. Ainsi, la séparation des états basée sur cette "structure" est perçue comme un système de contrôle à structure variable. Avec cette idée en tête, on peut considérer le système de contrôle flou comme un type de système à structure variable, où le contrôleur flou agit comme un contrôleur à structure variable avec une couche limite.

Tel que nous l'avons déjà démontré, le terme de correction dans une commande à structure variable est exprimé par :

$$u = -K \cdot sign(s) \tag{II.32}$$

Théoriquement, on observe que le mode glissant est optimal. Cependant, en raison des retards de commutation, il est peu probable que ce modèle idéal se manifeste dans un système réel, notamment à cause des oscillations autour de la surface de glissement. Ce problème est atténué par un lissage de la commande discontinue au sein d'une bande limite entourant la surface de glissement. En définissant ainsi une bande limite Φ, la loi de contrôle VSC est ajustée de la manière suivante[8]

$$\begin{cases} \boldsymbol{k} & \boldsymbol{s} > \Phi \\ -\boldsymbol{k}.\frac{\boldsymbol{s}}{\Phi} & -\Phi \leq s \leq \Phi \\ -\boldsymbol{k} & \boldsymbol{s} < -\Phi \end{cases} \tag{II.33}$$

Dans cette section, nous proposons un contrôleur flou, aussi connu sous le nom de système d'inférence floue, qui se révèle être équivalent au contrôleur VSC modifié décrit dans la section **(II.34)**. La conception de ce contrôleur repose sur une idée fondamentale : il est envisageable d'étendre la notion de surface de glissement, initialement définie par s = 0, pour inclure une surface floue caractérisée par une expression linguistique spécifique. Cette approche permet d'intégrer une logique multivalente, enrichissant ainsi le cadre traditionnel du contrôle par surface de glissement en ajoutant une dimension floue qui prend en compte les nuances et les variations présentes dans les systèmes réels. Cela offre un potentiel accru pour modéliser des comportements dynamiques plus complexes et pour proposer des solutions adaptatives dans des environnements incertains.

$$\tilde{s} \; est \; zero \tag{II.34}$$

Avec la variable linguistique s, où "zéro" est considéré comme l'un de ses ensembles flous. Dans le but de rendre flou l'espace autour de la surface de glissement s, cinq sous-ensembles flous sont définis (voir figure II.5), tels que :

$$\mathrm{T}(\tilde{s}) = \{\mathrm{NG}, \mathrm{NM}, \mathrm{EZ}, \mathrm{PM}, \mathrm{PG}\} = \{\mathrm{F}\, s^1, \ldots, Fs^5\} \tag{II.35}$$

avec les prédicats suivants :

NG : négatif grand ; NM : négatif moyen ; EZ : environ zéro ;

PM : positif moyen ; PG : positif grand. Pour la commande,

Nous définissons également cinq sous-ensembles flous comme suit :

$$\mathrm{T}(\tilde{u}) = \{\mathrm{NG}, \mathrm{NM}, \mathrm{EZ}, \mathrm{PM}, \mathrm{PG}\} = \{\mathrm{F}\, u^1, \ldots, Fu^5\} \tag{II.36}$$

Dans la figure II.5, sont représentées les fonctions d'appartenance associées aux deux variables, s et u . Pour la surface de glissement s, on adopte des fonctions d'appartenance de type triangulaire, tandis que pour la variable de commande u, on utilise des singletons.

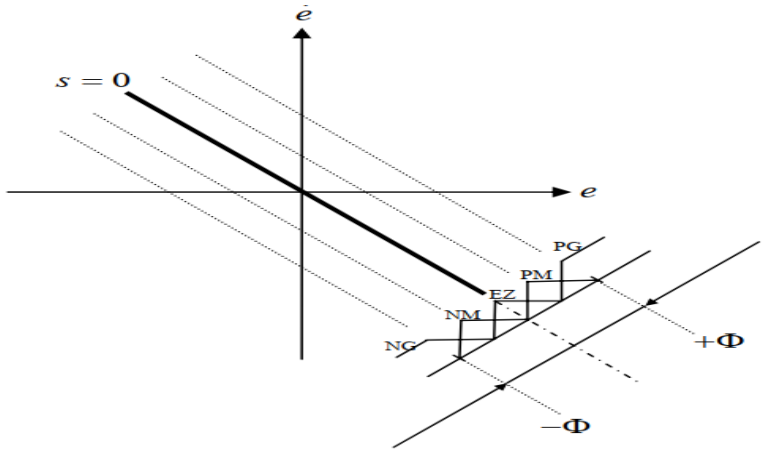
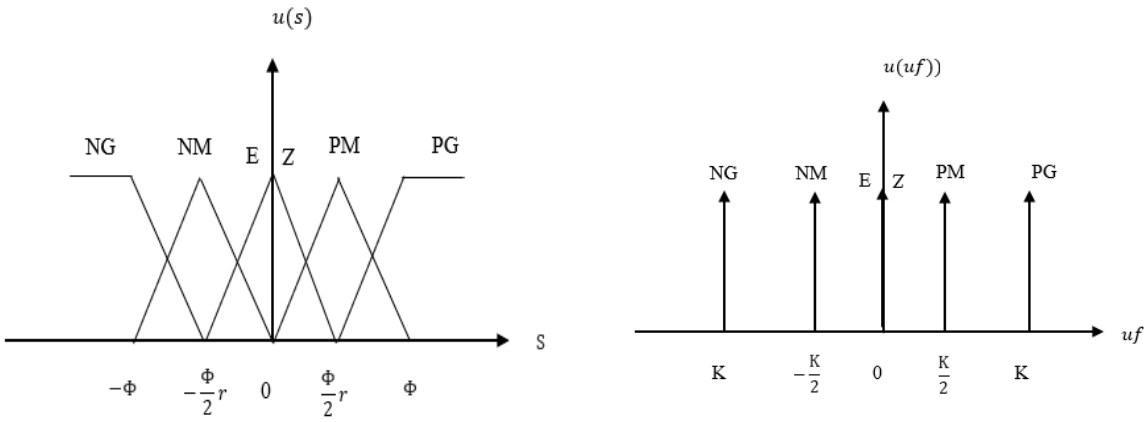


Figure II. 5: Partition floue de l'espace autour de la surface de glissement.



(a) **(b)**

Figure II. 6: Fonctions d'appartenance de l'entrée s et de sortie uf.

Dans la figure II.8.a, le paramètre r appartenant à l'intervalle (0,1] sert à ajuster les points modaux du sous-ensemble EZ, où la précision de la commande est représentée par l'expression indiquant que (s est 0). Pour ce système d'inférence flou, nous établissons les règles suivantes :

R^1: SI s est NG Alors u_f est PG

R^2 : SI s est NM Alors u_f est PM

R^3 : SI s est EZ Alors u_f est EZ

R^4 : SI s est PM Alors u_f est NM

R^5 : SI s est PG Alors u_f est NG

On peut également ajouter :

$$R^1\text{: SI s est } F_{S_i}^i \text{ Alors } u_f \text{ est } F_u^i \tag{II.37}$$

Avec i=1,…...,5

Nous prenons en compte X et Y, qui représentent respectivement les espaces d'entrée et de sortie des règles floues. Pour un ensemble flou quelconque $\tilde{F}_x$ dans l'espace X, un ensemble flou $\tilde{F}_x \circ R^i$ est déterminé dans l'espace Y par la règle R^i.

Par l'application de la méthode d'inférence max-min :

$$u_{\tilde{F}_x \circ R^i}(u_f) = \overset{MAx}{s\in X}[\min[\, u_{\tilde{F}_x}(s), \min\,[u_{\tilde{F}_x}(s), u_{\tilde{F}_u}(u_f)]]] \tag{II.38}$$

Dans ce contexte, les sous-ensembles flous ainsi que les fonctions d'appartenance ont été définis. À présent, il ne reste plus que la phase de défuzzification afin d'extraire la valeur de la commande u. Dans le cas où le sous-ensemble flou Fx prend la forme d'un singleton, il est possible d'exprimer cela de la manière suivante :

$$u_{\tilde{F}_x}(s) = \begin{cases} 1 & si \quad x \le -1 \\ 0 & , \quad ailleurs \end{cases} \tag{II.39}$$

Pour la phase de défuzzification, la méthode du centre de gravité est utilisée, ce qui nous donne :

$$u = \frac{\sum_{i=1}^{5} u_i(s).uf^i}{\sum_{i=1}^{5} u_i(s)} \tag{II.40}$$

Dans ce contexte, $u_i(s)$ représente le degré d'appartenance de l'élément s au sous-ensemble désigné par Fsii. À terme, le résultat de l'inférence, appliqué pour chaque élément s, se formalise ainsi:

$$u = -K \cdot sign(\frac{s}{\Phi}) \tag{II.41}$$

$$sig(z) = \begin{cases} -1 & z < -1 \\ \dfrac{z + r - 1}{2 - r} & -1 \le z < -r/2 \\ \dfrac{z}{r} & -r/2 \le z < 0 \\ \dfrac{z}{r} & 0 \le z < r/2 \\ \dfrac{z + 1 - r}{2 - r} & r/2 \le z < 0 \\ 1 & z \ge 1 \end{cases} \quad \text{(II.42)}$$

La figure II.7 présente les résultats de l'inférence des règles floues en fonction des différentes valeurs de la variable r. Il est évident que la valeur de cette variable influence considérablement la forme de la fonction. Lorsque ri est égal à 1, la fonction peut être assimilée à une fonction de saturation. Ainsi, en ajustant cette variable, il est possible d'optimiser les performances de la commande.

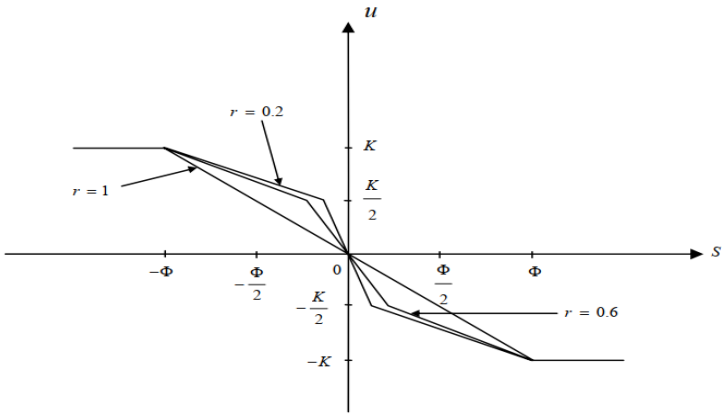


Figure II. 7:Résultats de l'inférence des règles floues pour différentes valeurs de r.

4. Les réseaux de neurones

Chez les organismes vivants, la discrimination de formes, le traitement des signaux, l'accumulation de connaissances, la rétention d'informations et la capacité à généraliser sont des activités menées quotidiennement de manière instinctive. C'est sur la base de l'idée que des comportements intelligents se développent à partir de la structure et du fonctionnement des éléments fondamentaux du cerveau que les réseaux de neurones artificiels (RNA) ont vu le jour. L'intégration des RNA dans le champ scientifique de l'ingénierie a été une avancée notable. Tout a commencé en 1943 avec la création du premier neurone artificiel, qui n'était rien d'autre qu'un

produit scalaire entre un vecteur d'entrée et un vecteur de poids, suivi d'un seuil. Grâce à des caractéristiques telles que le parallélisme, l'adaptabilité, la généralisation et l'approximation, les RNA représentent aujourd'hui un outil précieux pour la résolution d'un grand nombre de problèmes où les approches traditionnelles ont démontré leurs limites. L'optimisation des systèmes non linéaires, par exemple, fait partie des applications des RNA, surtout après l'avènement de la technique d'apprentissage par back propagation.

Les réseaux de neurones artificiels se composent de processeurs élémentaires hautement connectés, qui opèrent en parallèle. Chaque processeur génère une sortie distincte en se basant sur les informations qu'il reçoit. Ainsi, toute structure hiérarchique de ces réseaux constitue indéniablement un réseau.

Cette section de cours a pour objectif de présenter les RNA en réalisant une revue historique de leur développement, ainsi que de leurs architectures et applications variées.

Nous expliquerons ensuite comment un RNA génère des valeurs prédictives en fonction de ses entrées pour modéliser ou contrôler un processus. Enfin, nous aborderons les techniques d'apprentissage ainsi que la méthodologie associée aux décisions relatives à la complexité des RNA.[13]

4.1 Historiques de RNA

L'histoire des réseaux de neurones artificiels (RNA) remonte aux années 1940, période où Warren McCulloch et Walter Pitts publient un article innovant sur des modèles informatiques de neurones, introduisant ainsi le concept du premier neurone artificiel. Dans les décennies suivantes, c'est-à-dire les années 1950 et 1960, des recherches approfondies ont été menées pour explorer les potentialités des RNA dans la résolution de problèmes de reconnaissance de formes et de traitement du signal. En 1957, F. Rosenblatt présente le modèle du perceptron, défini comme un neurone formel, représentant ainsi le réseau neuronal le plus basique. Toutefois, en 1969, M. Minsky et S. Papert soulignent, dans un ouvrage marquant, les limitations théoriques de ce perceptron. Il est important de noter que ces premiers modèles étaient profondément contraints par leur complexité et leurs capacités de traitement.

En 1985, la présentation d'un perceptron multicouche a marqué une avancée significative avec l'apparition de l'algorithme d'apprentissage, notamment la rétropropagation du gradient, adapté aux

réseaux de neurones multicouches. Au cours des décennies suivantes, les années 1980 et 1990 ont vu des progrès notables dans le traitement de l'information et le stockage de données, ce qui a permis le développement de réseaux de neurones artificiels de plus en plus sophistiqués, capables de relever des défis de plus en plus complexes. Ces avancées ont ouvert la voie à l'utilisation des réseaux de neurones dans divers domaines, parmi lesquels se distinguent la vision par ordinateur, la reconnaissance vocale et la robotique.

Aujourd'hui, les réseaux de neurones sont omniprésents dans de nombreuses applications, telles que la reconnaissance de la parole et des images, le traitement du langage naturel, ainsi que la prévision de séries chronologiques. L'émergence du deep learning, ou réseau de neurones profonds, a également contribué à leur succès. Ces technologies continuent d'évoluer et de s'enrichir grâce aux progrès continus de la recherche et des capacités informatiques.[15]

4.2 Le neurone formel

Le neurone formel, qui constitue la brique fondamentale des réseaux de neurones artificiels, est un automate inspiré du fonctionnement d'un neurone biologique. On peut le décrire à travers plusieurs éléments, en prenant comme référence un neurone désigné par l'indice :

• Son état, également connu sous le nom d'activation Y, qui peut revêtir une valeur réelle ou booléenne. Cet état est généralement considéré comme la valeur de sortie du neurone.

• Ses connexions d'entrée, auxquelles sont associés des poids ωij, (j représente l'indice du neurone connecté.)

• Sa fonction d'entrée, qui effectue un prétraitement des données, souvent sous la forme d'une somme

Pondérée des entrées.

• Sa fonction d'activation (ou fonction de transfert) gi(x), qui calcule le résultat en se basant sur la

Sortie de la fonction d'entrée et l'activation du neurine

La figure II.8 illustre un neurone formel qui applique une fonction de seuil sur la somme pondérée de ses différentes entrées.

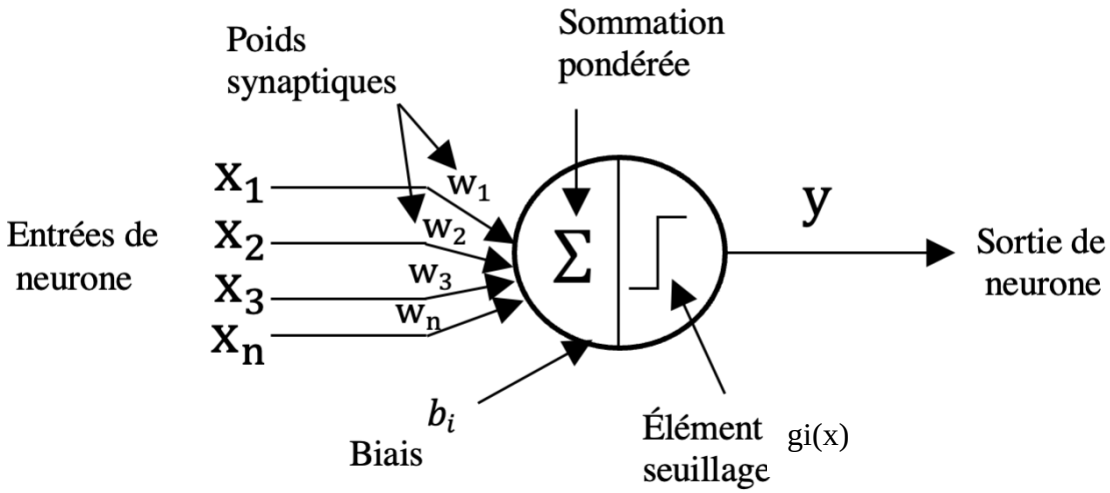


Figure II. 8: Neurone artificiel formel.

4.3 Le réseau de neurones multicouche (MLP)

Un ensemble de neurones interconnectés forme un réseau de neurones. Chaque neurone est capable de recevoir des signaux de plusieurs autres neurones et de transmettre une réponse à d'autres neurones. Les connexions entre les neurones sont pondérées, ce qui signifie qu'elles possèdent des poids ajustables pour influencer la sortie du neurone.

Le réseau de neurones sont structurés en couches : une couche d'entrée pour recevoir les données initiales, une ou plusieurs couches cachées pour le traitement des informations, et une couche de sortie pour produire le résultat final. Ce type d'organisation est connu sous le nom de réseau de neurones multicouche. Les connexions entre les neurones de différentes couches sont généralement pondérées de manière à accorder plus d'importance à certaines connexions par rapport à d'autres.[16]

La figure II.9 présente un exemple de MLP (Perceptron Multicouche).

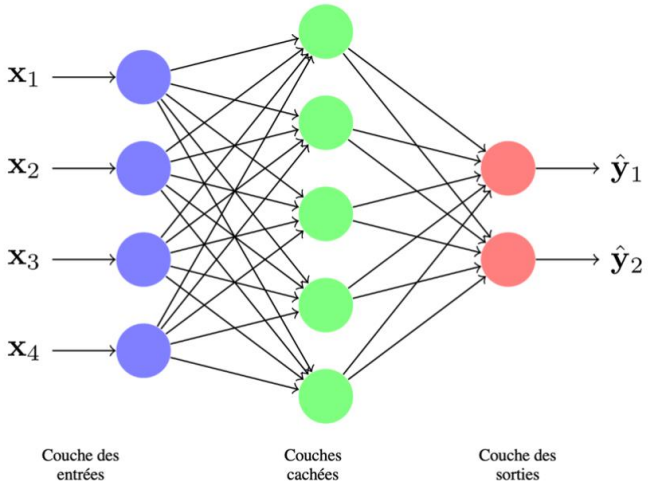


Figure II. 9: L'architecture d'un réseau de neurones multicouches.

Les réseaux neuronaux artificiels présentent un attrait majeur en raison de leurs capacités d'apprentissage. L'apprentissage implique généralement la modification automatique des poids des connexions, voire plus rarement du nombre et de la disposition des neurones, afin d'ajuster le fonctionnement du réseau à une tâche spécifique. On distingue trois types d'apprentissage en fonction des informations disponibles et des objectifs visés :

- L'apprentissage supervisé nécessite un ensemble de paires de données {entrées du réseau ; sorties souhaitées correspondantes}, appelées exemples ou motifs. La différence entre la sortie du réseau et la sortie souhaitée fournit une mesure quantitative de l'erreur du calcul effectué par le réseau, utilisée pour l'ajustement.

- L'apprentissage semi-supervisé ou par renforcement

4.4 Les réseaux de neurones dans la commande à structure variable

Comme mentionné dans les parties précédents, une commande par mode de glissement se compose d'un composant continu, qualifié de commande équivalente, qui sert à anticiper l'état futur du système, représentant en fait la valeur souhaitée pour le régime stationnaire, ainsi qu' un composant

De correction, désigne par la commande discontinue.

Pour déterminer la commande équivalente, il est nécessaire de comprendre la dynamique du système (ou sa dynamique inverse). Néanmoins, de nos jours, la modélisation des systèmes est

considérée comme un souci majeur dans le domaine de la commande. Une solution à ce défi consiste à recourir à un estimateur linéaire pour établir cette commande.

Dans cette méthode, la commande équivalente est élaborée à partir d'un filtrage de la commande efficace (quiInclut à la fois la commande équivalente et la commande discontinue). Malheureusement, une telle structure présente des inconvénients, notamment en matière de chattering.

Dans cette section, nous proposons une approche innovante pour le calcul de la commande équivalente en recourant aux réseaux de neurones. En effet, de nombreux travaux de recherche explorent la commande de systèmes à l'aide de réseaux de neurones. Une des voies privilégiées consiste à utiliser ces réseaux pour estimer la dynamique inverse du système à contrôle[17], [18], [19]

4.5 L'Approximation des fonctions et prédiction à l'aide de réseaux neuronaux

Les réseaux neuronaux sont couramment utilisés en tant qu'outils d'approximation des fonctions non linéaires liant les variables de sortie aux variables d'entrée dans le contexte des systèmes industriels. Leur principal objectif réside dans la reconstruction d'une variable réelle pertinente pour des prises de décision ultérieures. Cette méthodologie se distingue par son indépendance vis-à-vis des connaissances préalables concernant le système, qu'elles soient de nature mathématique, physique ou autre.

En ce sens, les réseaux neuronaux peuvent être considérés comme une boîte noire non linéaire, dont la caractérisation peut être soit statique, soit dynamique, en fonction de la nature du réseau (intégration temporelle ou non). Le processus d'identification par réseaux neuronaux repose sur l'élaboration d'un modèle à partir de l'analyse des signaux couplés entrée-sortie.

Ce processus vise à établir une relation fonctionnelle entre l'entrée et la sortie du modèle. Deux enjeux majeurs motivent cette démarche :

- Prédire le comportement du système sous différentes conditions de fonctionnement.

- Concevoir une loi de commande permettant au système d'atteindre l'objectif défini.

La figure II.10 illustre le schéma général de l'identification neuronale. Dans cette configuration, le réseau neuronal opère en parallèle avec une boîte noire représentant le processus à identifier. La sortie réelle du processus y est comparée à la sortie estimée par le réseau neuronal $\hat{y}$,L'erreur résultante,

définie comme $e = y - \hat{y}$ alimente un algorithme d'apprentissage conçu pour ajuster les paramètres du réseau neuronal et optimiser ses performances. [20]

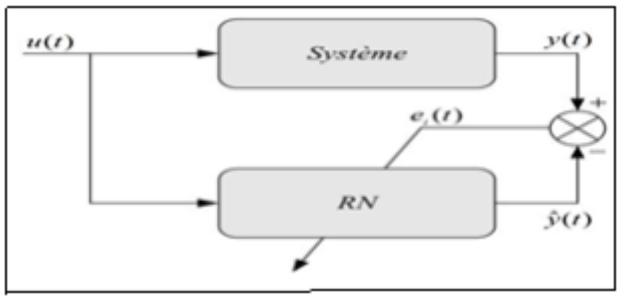


Figure II. 10: Schéma de principe d'identification par réseau de neurones.

Afin de prendre en compte les dynamiques propres au processus, le réseau neuronal est enrichi d'éléments supplémentaires comme les valeurs passées des signaux d'entrée et de sortie. Cela permet d'améliorer la modélisation et de mieux appréhender les relations temporelles existantes dans le système étudié.

5. Conclusion

Ce chapitre a présenté les concepts fondamentaux des principales techniques de commande intelligente, à savoir la commande glissante, la commande floue, et la commande par réseau de neurones. Ces approches représentent une évolution par rapport aux méthodes classiques de commande, notamment pour les systèmes non linéaires, incertains ou difficiles à modéliser précisément. Ces commandes intelligentes offrent des solutions puissantes et flexibles pour le contrôle de systèmes non linéaires et incertains. Chacune de ces méthodes présente des avantages spécifiques, et leur intégration dans des approches hybrides constitue une orientation pertinente pour répondre aux exigences croissantes en termes de performance, d'adaptabilité et de robustesse dans les systèmes de commande modernes. C'est ce que nous verrons dans le reste de Manus.

Chapitre III :

La commande Neuro-floue-glissante

Chapitre III : La commande Neuro-floue-glissante.

1. Introduction

La commande glissante apporte une bonne robustesse, mais engendre des phénomènes de chattering indésirables. La logique floue est introduite pour adoucir la transition entre les modes de glissement et améliorer la stabilité. Quant aux réseaux de neurones, ils sont utilisés pour compenser les incertitudes liées à la modélisation du robot et aux perturbations non modélisées.

Ce chapitre présente le développement et l'implémentation d'une stratégie de commande hybride combinant les trois approches avancées qui sont présenté et détailler dans le chapitre précèdent : la commande par mode glissant, la logique floue et les réseaux de neurones artificiels, appliquée sur le modèle dynamique de robot manipulateur PUMA 560 présenté dans le premier chapitre.

L'objectif principal est d'assurer un suivi de trajectoire précis tout en maintenant une robustesse face aux incertitudes et aux perturbations du système.

2. Commande par mode glissant du robot PUMA 560

Dans cette partie, nous avons conçu une loi de commande stabilisante, en nous appuyant sur la théorie du contrôle à structure variable. Cette commande est destinée à répondre à des exigences spécifiques telles que la stabilité, la vitesse, l'exactitude et la solidité. Nous présentons les résultats de la simulation, qui concerne le suivi de trajectoires, et avons opté pour le test de chute de charge comme critère d'évaluation de la robustesse.

2.1 La synthèse de la loi de commande

Le modèle du robot PUMA 560, lorsqu'on ne prend pas en compte l'effet de la charge, se présente ainsi :

$$M\ddot{q} + V_m\dot{q} + G = u \qquad \text{(III.1)}$$

Le modèle en question détaille le fonctionnement dynamique d'un robot manipulateur doté de trois degrés de liberté. Pour réussir à contrôler ce robot, il est impératif de concevoir trois systèmes de commande distincts. Chaque système de commande requiert également la création d'une surface, désignée par s_i, où $i = [1:3]$. Il est important de noter que chaque articulation du robot est perçue

comme un sous-système autonome ayant un degré relatif, noté. $r_i = 2$ Cela signifie que chaque surface associée, si, est de premier ordre, correspondant à un ordre calculé par la formule (r_i - 1). Ce principe permet d'assurer un contrôle fluide et coordonné des mouvements du manipulateur.

Le vecteur d'erreur est déterminé comme suit :

$e = q - q_d$ avec $e_i = q_i - q_{id}$, et la surface de glissement, $s = [s_1, \quad s_2, \quad s_3]^T$,
telle que :

$$s_i = \dot{e}_i + \Lambda(e_i) \tag{III.2}$$

Et

$$\Lambda(x) = [\Lambda_1 \quad \Lambda_2 \quad \Lambda_3] \tag{III.3}$$

Avec

$$\begin{cases} \Lambda_i(x) = \frac{2\eta_i}{1+e^{-x}} - \eta_i \\ \frac{d\Lambda_i(x)}{dx} = \frac{1}{2\eta_i}[\eta_i^2 - \Lambda_i(x)] \end{cases} , \eta_i \in \Re \tag{III.4}$$

On est capable d'écrire:

$$\mathrm{s} = \dot{\mathrm{q}} - (\dot{\mathrm{q}}_\mathrm{d} - \Lambda\,(\mathrm{e})) \tag{III.5}$$

Où

$$\mathrm{s} = \dot{\mathrm{q}} - \dot{\mathrm{q}}_\mathrm{r} \tag{III.6}$$

$\dot{q}_r = \dot{q}_d - \Lambda\,(e)$ est utilisé comme référence pour la vitesse articulaire.
nous pouvons mettre :

$$\mathrm{M}.\dot{\mathrm{s}} = \mathrm{u} - \mathrm{V_m}.\dot{\mathrm{q}} - \mathrm{G} - \mathrm{M}.\ddot{\mathrm{q}}_\mathrm{r} \tag{III.7}$$

$$\mathrm{M}.\dot{\mathrm{s}} = \mathrm{u} - \mathrm{V_m}.(\mathrm{s} + \dot{\mathrm{q}}_\mathrm{r}) - \mathrm{G} - \mathrm{M}.\ddot{\mathrm{q}}_\mathrm{r} \tag{III.8}$$

$$\mathrm{M}.\dot{\mathrm{s}} = \mathrm{u} - \mathrm{M}\ddot{\mathrm{q}}_\mathrm{r} - \mathrm{V_m}.\mathrm{s} - \mathrm{V_m}.\dot{\mathrm{q}}_\mathrm{r} - \mathrm{G} \tag{III.9}$$

La fonction de Lyapunov est définie de la manière suivante :

$$V = \frac{1}{2}s^T M s \tag{III.10}$$

Selon la propriété (II.6), la matrice (M) est définie positive, ce qui implique également $V > 0$ lorsque $s \neq 0$. Ainsi, en considérant l'analyse de la stabilité selon le critère de Lyapunov, le système est réputé stable si la dérivée temporelle de(V), notée $\dot{V} < 0$, satisfait la condition $\dot{V} < 0$.

Avec

$$\begin{cases} \dot{V} = s^T M\dot{s} + \frac{1}{2}s^T \dot{M}s \\ = s^T[u - M\ddot{q}_r - V_m.s - V_m.\dot{q}_r - G] + \frac{1}{2}s^T \dot{M}s \end{cases} \quad \text{(III.11)}$$

Et ensuite, on a $s^T[M \ -2.V_m]s = 0$

Cela signifie que

$$\dot{V} = s^T[u - M\ddot{q}_r - V_m.s - V_m.\dot{q}_r - G] \quad \text{(III.12)}$$

En sélectionnant la commande u, de manière à:

$$u = u_{eq} - K^T sign(s) \quad \text{(III.13)}$$

Avec :

$$u_{eq} = M\ddot{q}_r + V_m\dot{q}_r + G \quad \text{(III.14)}$$

$sign$: La fonction signe, et les gains de commutations

$k = [\,K_1\,,\ K_2\,,\ K_3]$, avec $K_i > 0$.

En prenant en compte le choix du commande u, nous arrivons à obtenir :

$$\dot{V} = K^T sign(s) < 0 \quad \text{(III.15)}$$

En conséquence, la commande (III.13) garantit la stabilité du système en boucle fermée. (Le point d'équilibre $e_i = 0$, avec $i = 1{:}\,3$, est asymptotiquement stable).

En utilisant la surface de glissement, la loi de commande à structure variable pour le robot manipulateur PUMA 560 est finalement établie.

$$\begin{cases} u = u_{eq} - K^T sign(s) \\ \quad et \\ u_{eq} = M\ddot{q}_r + V_m\dot{q}_r + G \end{cases} \quad \text{(III.16)}$$

Avec :

$$\begin{cases} s = \dot{e} + \Lambda(e) \\ \quad et \\ \dot{q}_r = (\dot{q}_d - \Lambda(e)\,) \\ \quad et \\ \Lambda_i(x) = \dfrac{2\eta_i}{1 + e^{-x}} - \eta_i \end{cases} \quad \text{(III.17)}$$

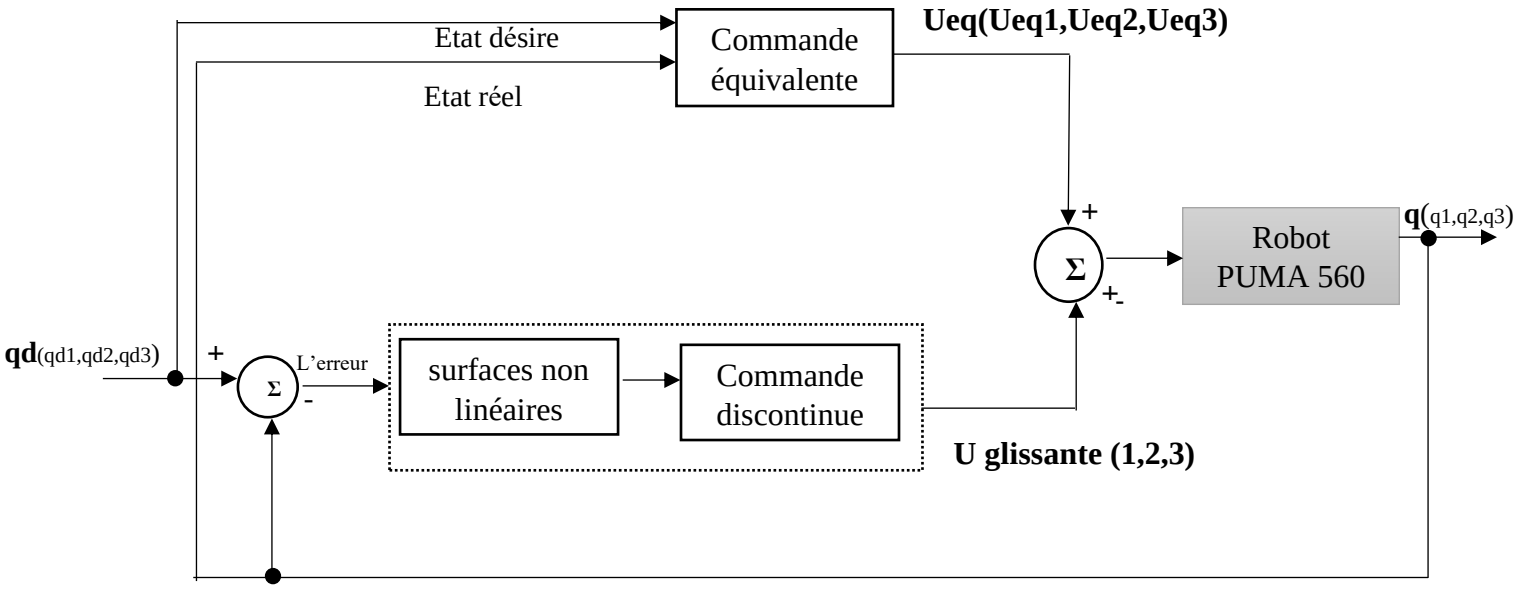


Figure III. 1 : Schéma bloc de la commande par mode glissant appliquée au robot manipulateur PUMA 560.

2.2 Résultats et simulation

La commande (III.16) a été simulée en utilisant le modèle du robot PUMA 560, et les résultats sont présentés dans les figures III.3, III.4, III.5 et III.6. Dans ce contexte, pour simuler la dynamique du robot, nous prenons en compte le couple généré par la charge mo, et nous utilisons le modèle intégral.

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + u_{mo} = u \qquad \text{(III.18)}$$

On utilise des trajectoires de référence de type cycloïdale pour garantir la continuité en termes de position, vitesse et accélération. Cette opportunité permet au robot de se mouvoir d'un point de départ à un point d'arrivée grâce à des instructions acceptables.

Pour stimuler toutes les dynamiques du robot, une trajectoire cycloïdale de test est mise en place où les diverses articulations passent respectivement de la position {-55, -135, 130} à la position {40, -80, 30}, en une durée de déplacement de 2 secondes.

$$\begin{cases} q_{di}(0) + \frac{D_i}{2\pi}\left[2\pi\frac{t}{t_f} - sin\left(2\pi\frac{t}{t_f}\right)\right] & pour\ \ 0 \le t \le t_f \\ q_{di}(t_f) & pour\ \ t_f < t \end{cases} \qquad \text{(III.19)}$$

$D_i = q_{di}(t_f) - q_{di}(0)$: Le déplacement, et t_f est l'instant final du mouvement.

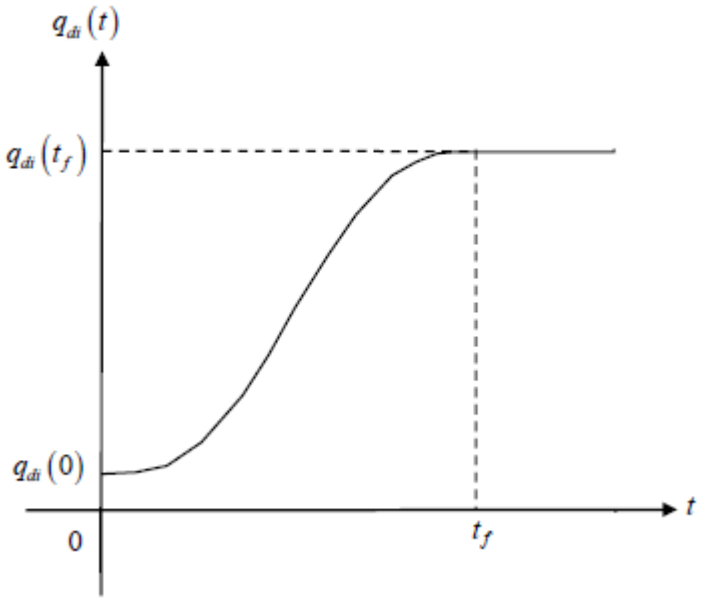


Figure III. 2: Trajectoire cycloïdale.

Paramètres de simulation

Les simulations réalisées sur le modèle dynamique du robot PUMA 560 ont été effectuées avec ηt = [30, 30, 30]pour les surfaces glissantes et K^t= [150, 150, 150]pour les gains de commutation du contrôle discontinu.

Avec une période d'échantillonnage de T = 0,001 sec et une durée de simulation de t = 2 sec.

Essai de solidité

Pour évaluer la solidité des lois de commande suggérées,Dans ce scénario, le robot effectue un traçage de trajectoires et dès que la vitesse maximale est atteinte

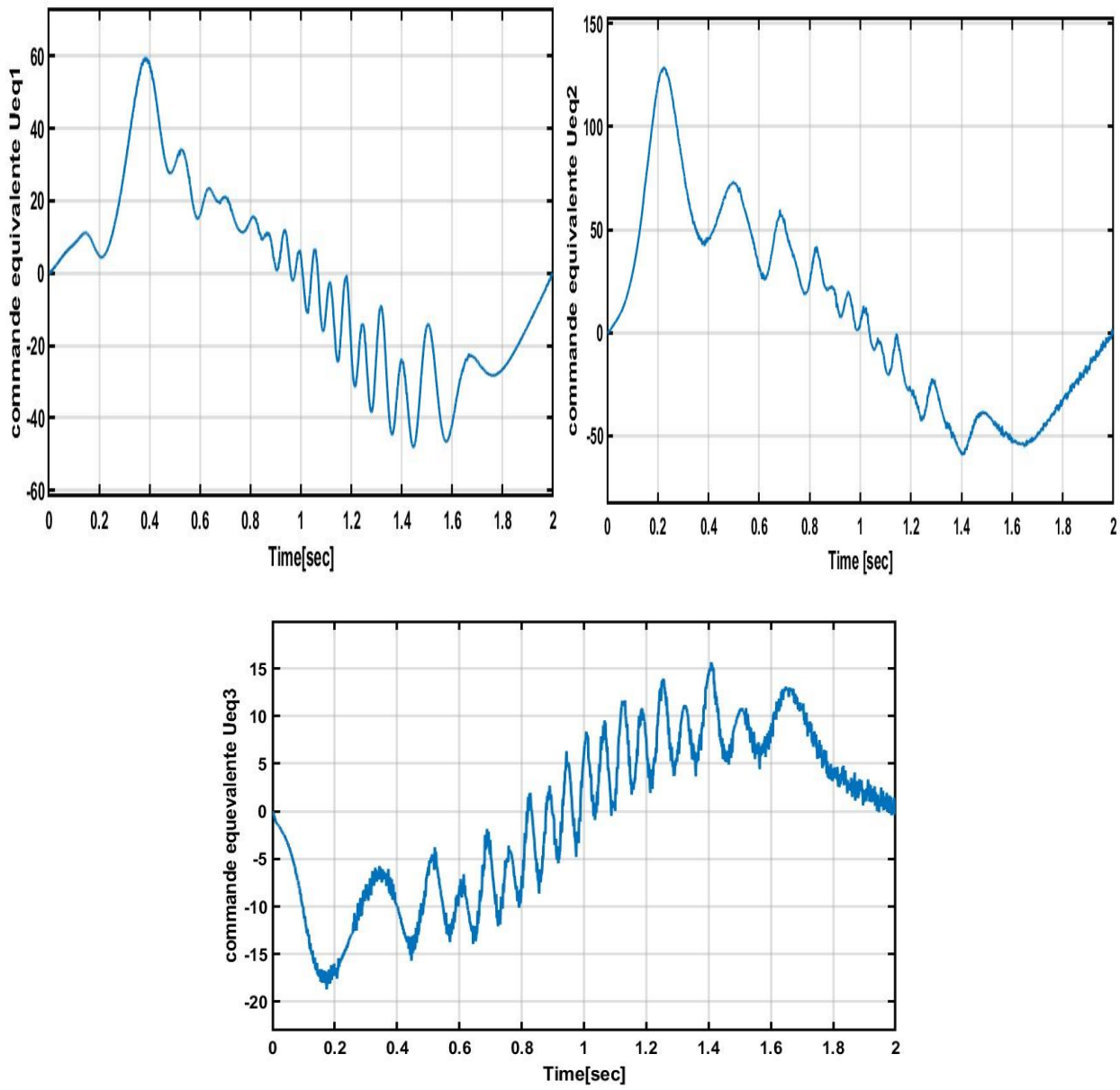


Figure III. 3: Commandes équivalentes de la commande VSC appliquée au bras manipulateur.

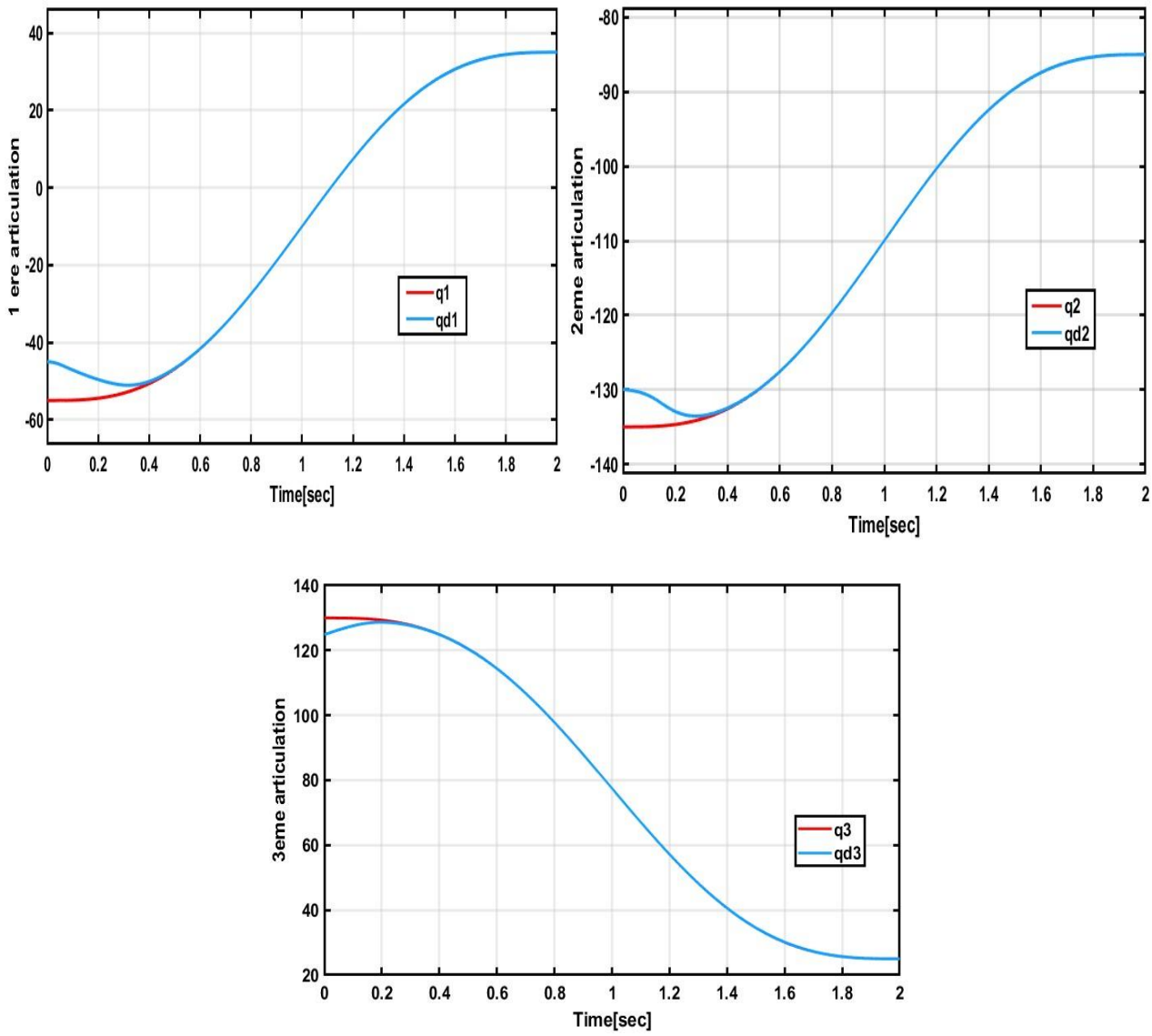


Figure III. 4: Positions articulaires de la commande pour un test à vide.

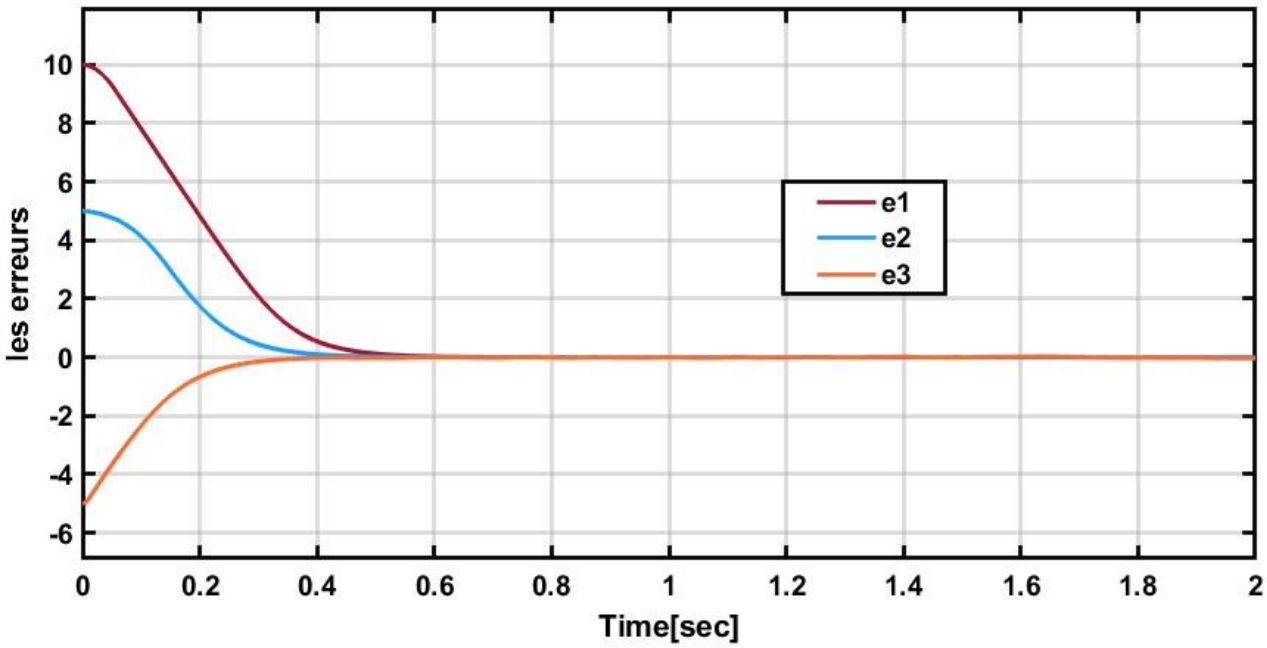
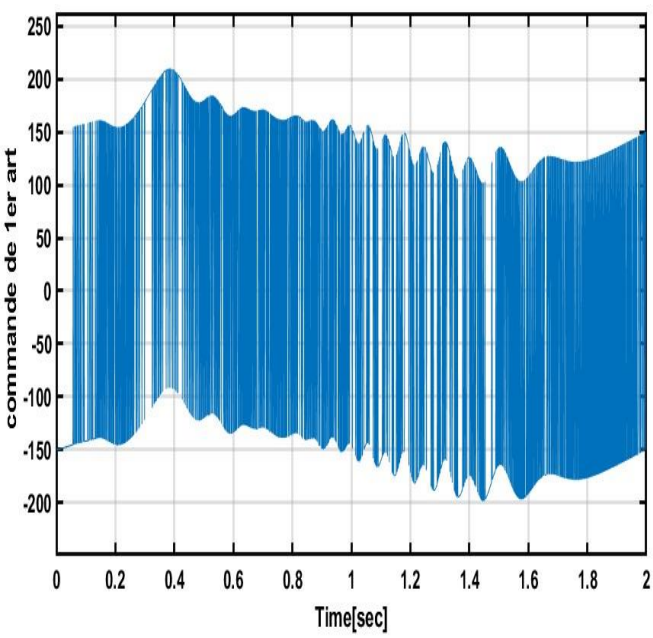
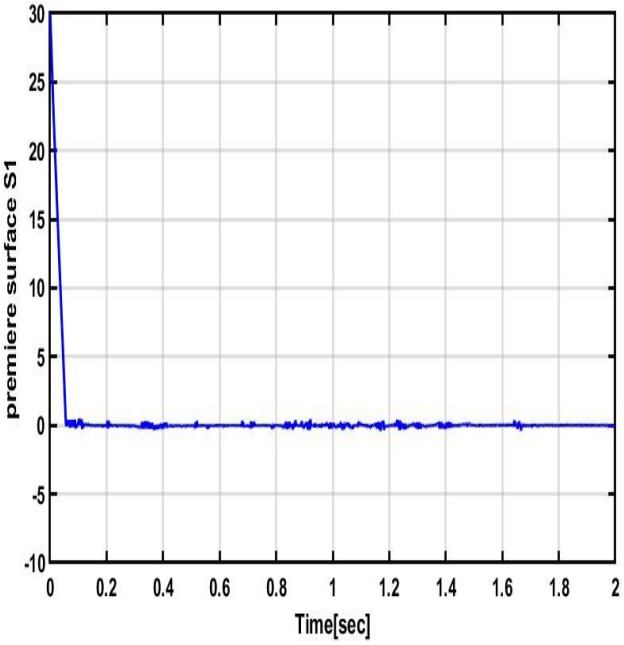


Figure III. 5: les erreurs de poursuite de la commande pour un test à vide.





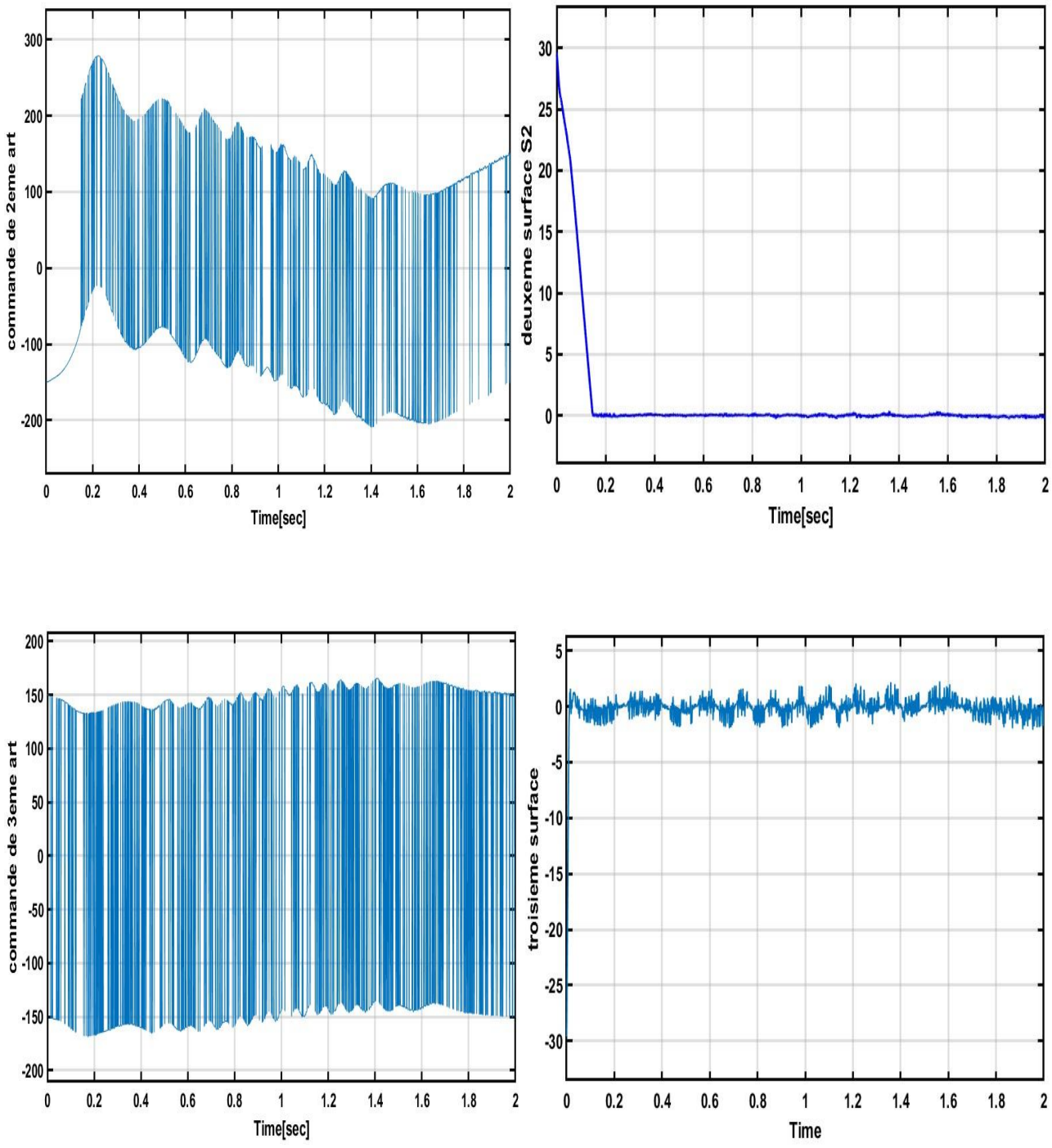


Figure III. 6: Surfaces de glissement et commandes effectives pour un test à vide.

Commentaires sur les résultats

Dans notre revue de littérature concernant la commande des bras manipulateurs, nous avons découvert que les contrôleurs à régime glissant sont largement employés pour les robots manipulateurs dans les laboratoires de recherche. En effet, de nombreux auteurs soulignent que cette méthode est l'une des plus adaptées en robotique.

Dans la section précédente, nous avons développé une loi de commande par modes glissants spécifiquement conçue pour le contrôle de position du robot PUMA 560, en s'appuyant sur des surfaces non linéaires définies au préalable. Les résultats des simulations, présentés dans les figures III.3, III.4, III.5 et III.6, permettent de confirmer la validité de l'approche adoptée.

La figure III.3 nous donne l'opportunité d'observer les commandes équivalentes appliquées au bras manipulateur par la commande VSC.

Ainsi, la figure III.4 met en évidence une convergence asymptotique des positions articulaires vers les valeurs désirées. À titre d'exemple, les temps de réponse observés sont respectivement de 0,35 seconde pour les premières et deuxième articulation, et d'environ 0,25 seconde pour la troisième articulation.

Par ailleurs, la figure III.5 représentés erreurs de suivi acceptables.

Et la figure II.6 illustre également une convergence asymptotique des surfaces de glissement vers l'origine. Cependant, il est important de noter que les commandes utilisées pour piloter le robot manipulateur présentent des fréquences élevées, ce qui constitue une limitation significative en raison des effets potentiellement néfastes associés.

3. Synthèse de la commande Neuro-floue-glissante au robot PUMA 560

La commande (III.16), synthétisée dans ce chapitre par la théorie des systèmes à structure variable avec des surfaces de glissement non linéaires proposées, constitue une loi de commande stabilisante. De surcroît, elle offre des performances acceptables en termes de rapidité durant le régime transitoire et présente également une robustesse face aux variations paramétriques. Toutefois, le phénomène de "chattering" (tremblement) représente un inconvénient majeur associé à cette loi. En effet, une commande opérant à de hautes fréquences peut exciter la dynamique non modélisée du robot, entraînant ainsi une détérioration des actionneurs au niveau des

articulations. Par ailleurs, pour le calcul de la commande équivalente, il est nécessaire de disposer du modèle du robot à commander ; cependant, ce modèle s'avère difficile à obtenir avec précision. Afin de pallier ces deux problèmes, nous proposons une commande neuro-floue-glissante. En effet, un réseau de neurones est utilisé pour le calcul de la commande équivalente, tandis qu'une commande discontinue est déterminée par un système d'inférence floue. La dynamique en boucle fermée est préalablement établie à partir des dynamiques des surfaces de glissement (voir figure III.7).

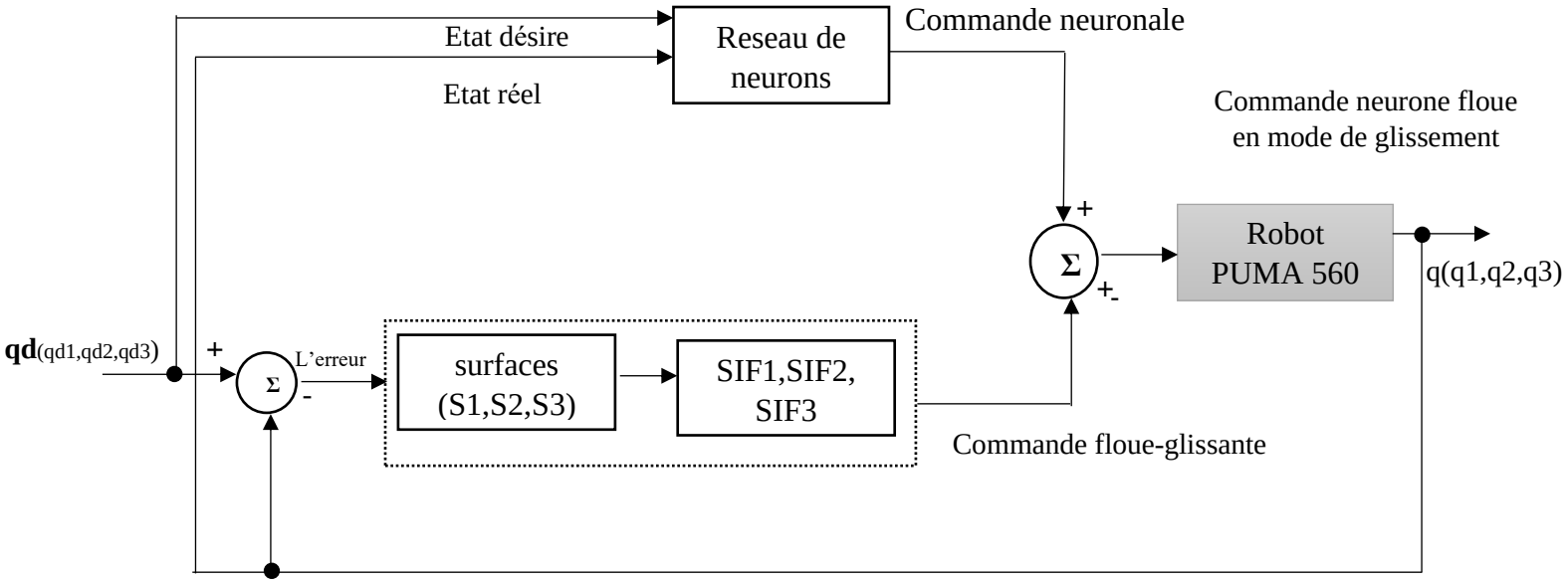


Figure III. 7: Schéma bloc de la commande Neuro-floue-glissante proposée pour le bras manipulateur PUMA 560.

Dans le cadre de l'approche proposée, la commande effective appliquée au bras manipulateur se compose de deux éléments. Un système d'inférence floue joue le rôle de la commande discontinue, agissant en tant que mécanisme de correction, tandis qu'un réseau de neurones est utilisé pour calculer la commande équivalente. La dynamique souhaitée en boucle fermée est déterminée par la surface de glissement.

3.1 La commande par logique floue

L'emploi de la logique floue pour contrôler les systèmes offre une loi de commande remarquablement efficace pour des systèmes complexes et hautement non linéaires, sans nécessité d'un modèle mathématique précisément défini. Cela s'effectue par le biais d'inférences utilisant des règles floues basées sur des variables langagières.[21]

$$\begin{cases} u = u_{eq_neuro} + u_{glis_floue} \\ et \\ s = \dot{e} + V(e) \\ avec \\ \Lambda_i(x) = \frac{2\eta_i}{1+e^{-x}} - \eta_i \end{cases} \quad \text{(III.20)}$$

$$\begin{cases} u_{glis_floue} = [u_1, u_2, u_3]^T \\ \boldsymbol{et} \\ u_j = \frac{\sum_{i=1}^{5} u_i(s_j^i).uf_j^i}{\sum_{i=1}^{5} u_i(s_j^i)},\ j = 1{:}3 \end{cases} \quad \text{(III.21)}$$

calculé par un système d'inférence flou, sa description est détaillée par les schémas suivants.

La figure suivante présente la partie prémisse de système d'inférence :

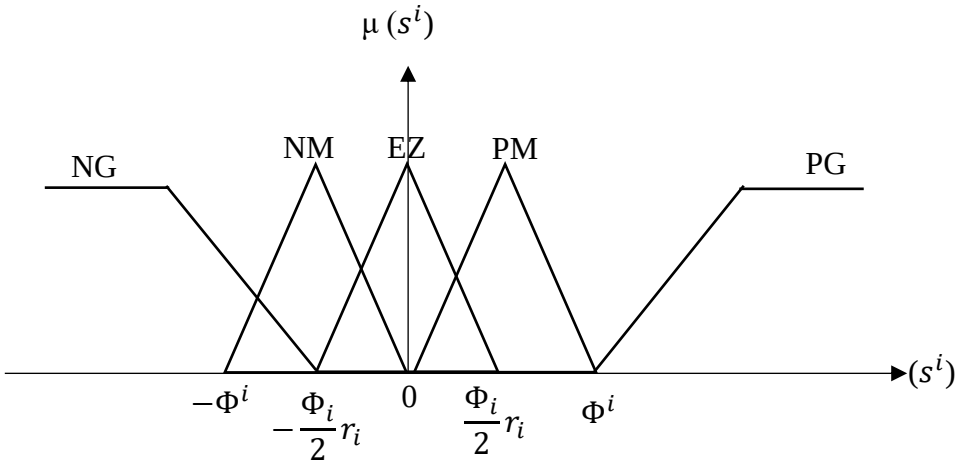


Figure III. 8: partie prémisse**.**

Dans ce système d'inférence flou, nous définissons les règles ci-après :

R^1: SI s^i est NG Alors $u_f{}^i$est PG

R^2 : SI s^iest NM Alors $u_f{}^i$est PM

R^3 : SI s^i est EZ Alors $u_f{}^i$ est EZ

R^4 : SI s^i est PM Alors $u_f{}^i$est NM

R^5 : SI s^i est PG Alors $u_f{}^i$ est NG

En fin de compte, nous avons à présent la figure suivante pour la partie conclusion :

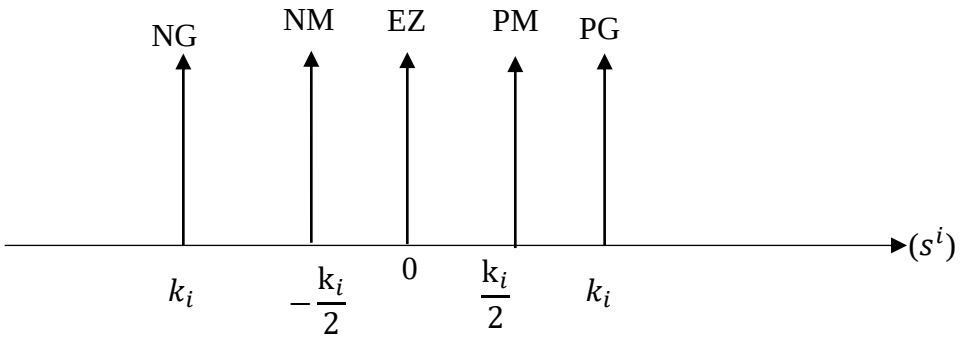


Figure III. 9: la partie conclusion.

ueq−Neuro : c'est une commande neuronale calculée par un réseau de neurones.sa structure fait l'objet de la partie suivant :

3.2 La commande par réseau de neurone

Pour le calcul de la commande équivalente se compose d'une couche cachée et d'une couche de sortie. La sélection des entrées et des sorties est déterminée à partir de l'équation de la commande équivalente, comme l'illustre la figure (III.4)

Pour mieux illustrer cette approche ou architecture, nous utilisons la notation suivante :

zi: Oui La deuxième entrée de la couche d'entrée du réseau.

Ynetj : Oui Le jème nœud dans la couche cachée.

Yout J : Y en a-t-il ? Sortie de la couche cachée.

Unetk : Oui Le k-ième nœud de la couche de sortie.

Uoutk : Oui La k-ième sortie de la couche de sortie.

Wzi,j : indique Le ème neurone dans la couche d'entrée et Couche cachée.

Wyj,k : C'est Le jème neurone dans la couche cachée et le k ème neurone dans la couche cachée Couche de sortie.

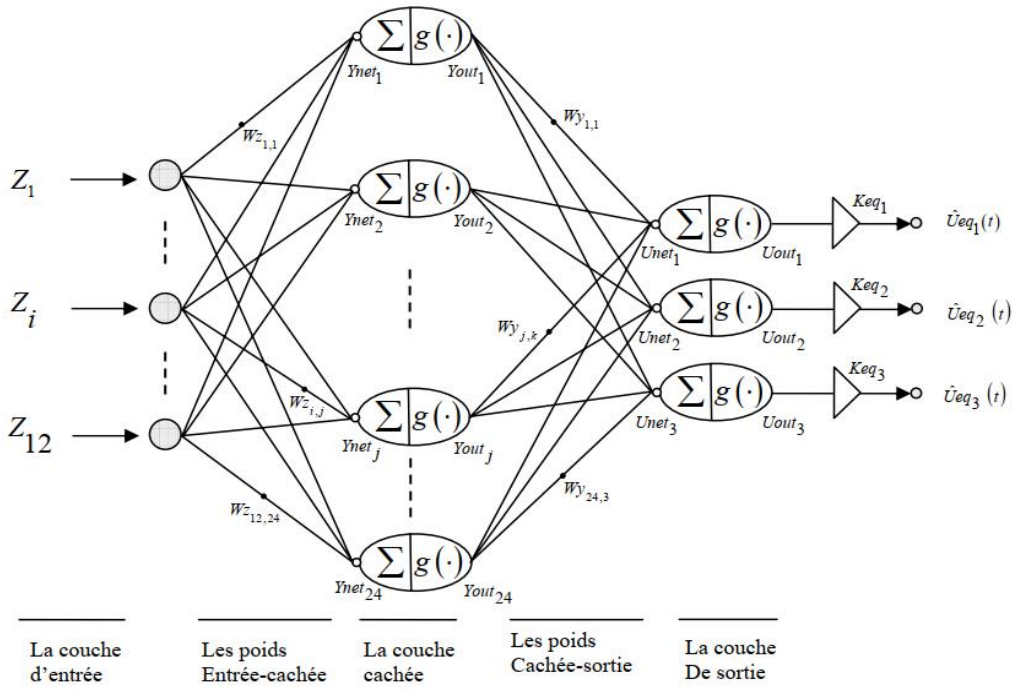


Figure III. 10: Réseau de neurones utilisé pour calculer la commande équivalente pour le robot manipulateur**.**

Les diverses formules qui exposent cette configuration neuronale sont les suivantes :

$$\begin{cases} Y_{net_j} = \sum_{i=1}^{24} W_{Z_{i,j}} . z_i, \quad j = 1{:}\,24 \\ Y_{out_j} = g(Y_{net_j}) \end{cases} \tag{III.22}$$

$$\begin{cases} U_{net_j} = \sum_{j=1}^{24} W_{y_{j,k}} . Y_{out_j} \\ U_{out_k} = g(U_{net_k}) \qquad k = 1{:}\,3 \\ et \\ U^k_{eq_neuro} = \widehat{U}_{eq_k}(t) = K_{eq_k} . U_{out_k} \end{cases} \tag{III.23}$$

g(·) représente la fonction d'activation, définie comme suit :

$$g(x)\frac{2}{1+e^{-x}} - 1 \tag{III.24}$$

Les entrées zi du réseau correspondent aux états et aux états souhaités, tandis que les sorties sont :

$U_{eq}(t)$ Représente les ordres estimés par le réseau, tandis que K_{eq_k} est la valeur maximale que peut obtenir l'ordre équivalent.

Le nombre de neurones dans la couche de sortie correspond au total des commandes du système, tandis que le nombre de neurones dans la couche cachée du réseau est déterminé pour pouvoir appréhender l'intégralité de la dynamique inverse du système.

L'adaptation des poids

Habituellement, dans le domaine des réseaux de neurones, la rétropropagation avec l'algorithme du gradient descend est la technique la plus couramment employée pour construire un réseau de neurones à plusieurs couches.

Dans notre contexte et dans le cadre de l'apprentissage, nous avons choisi d'utiliser l'algorithme de rétropropagation du gradient pour réduire la valeur quadratique moyenne de l'erreur entre la commande équivalente estimée et sa valeur réelle.

Voici le critère E que nous devons optimiser :

$$E = \frac{1}{2} \cdot \sum_{k=1}^{3} (U_{eq_k} - \widehat{U}_{eq_k})^2 \tag{III.25}$$

Les lois d'adaptation des poids sont élaborées de manière à réduire au minimum le critère E. Pour ce faire, on utilise l'algorithme de descente de gradient où les poids sont modifiés dans le sens de la diminution de la valeur de E.

$$\begin{aligned} W_{y_{jk}}(t) &= W_{y_{jk}}(t-1) - \alpha \cdot \frac{\partial E}{\partial W_{y_{jk}}} \\ W_{z_{ij}}(t) &= W_{z_{ij}}(t-1) - \alpha \cdot \frac{\partial E}{\partial W_{z_{ij}}} \end{aligned} \tag{III.26}$$

le taux d'apprentissage est noté $\alpha > 0$.

L'adaptation de $W_{y_{jk}}$

$$W_{y_{jk}}(t) = W_{y_{jk}}(t-1) - \alpha \cdot \frac{\partial E}{\partial W_{y_{jk}}} \tag{III.27}$$

$$\frac{\partial E}{\partial W_{y_{jk}}} = \frac{\partial E}{\partial U_{net_k}} \cdot \frac{\partial U_{net_k}}{\partial W_{y_{jk}}} \tag{III.28}$$

En se basant sur l'équation (III.23**)** on peut formuler :

$$\frac{\partial E}{\partial W_{y_{jk}}} = \frac{\partial E}{\partial U_{net_k}} \cdot Y_{out_j} \tag{III.29}$$

$$= \left[\frac{\partial E}{\partial \widehat{U}_{eq_k}} \cdot \frac{\partial \widehat{U}_{eq_k}}{\partial U_{net_k}}\right] \cdot Y_{out_j}$$

$$= -\left[\left(U_{eq_k} - \widehat{U}_{eq_k}\right) \cdot K_{eq_k} \cdot g'\left(U_{net_k}\right)\right].Y_{out_j} \quad \text{(III.30)}$$

En fin de compte, on découvre :

$$\frac{\partial E}{\partial W_{y_{jk}}} = -\left[\left(U_{eq_k} - \widehat{U}_{eq_k}\right) \cdot K_{eq_k} \cdot \frac{1}{2} \cdot \left(1 - g\left(U_{net_k}\right)^2\right]\right].Y_{out_j} \quad \text{(III.31)}$$

Donc

$$W_{y_{jk}}(t) = W_{y_{jk}}(t-1) + \frac{\alpha}{2} \cdot \left[\left(U_{eq_k} - \widehat{U}_{eq_k}\right) \cdot K_{eq_k} \cdot \left(1 - g\left(U_{net_k}\right)^2\right]\right].Y_{out_j} \quad \text{(III.32)}$$

L'adaptation de $W_{z_{ij}}$

$$W_{z_{ij}}(t) = W_{z_{ij}}(t-1) - \alpha \cdot \frac{\partial E}{\partial W_{z_{ij}}} \quad \text{(III.33)}$$

On a

$$\frac{\partial E}{\partial W_{z_{ij}}} = \frac{\partial E}{\partial Y_{net_j}} \cdot \frac{\partial Y_{net_j}}{\partial W_{z_{ij}}} \quad \text{(III.34)}$$

On peut commencer à mettre en place à partir de (III.22) :

$$\frac{\partial E}{\partial W_{z_{ij}}} = \frac{\partial E}{\partial Y_{net_j}} \cdot z_i \quad \text{(III.35)}$$

$$= \left[\frac{\partial E}{\partial Y_{out_j}} \cdot \frac{\partial Y_{out_j}}{\partial Y_{net_j}}\right] \cdot z_i \quad \text{(III.36)}$$

Avec

$$\frac{\partial E}{\partial Y_{out_j}} = \frac{\partial E}{\partial \widehat{U}_{eq_k}} \cdot \frac{\partial \widehat{U}_{eq_k}}{\partial U_{net_k}} \cdot \frac{\partial U_{net_k}}{\partial Y_{out_j}} \quad \text{(III.37)}$$

$$= -\sum_{k=1}^{3}\left(U_{eq_k} - \widehat{U}_{eq_k}\right) \cdot k_{eq_k} \cdot g'\left(U_{net_k}\right) \cdot W_{y_{jk}} \quad \text{(III.38)}$$

$$= -\frac{1}{2} \cdot \sum_{k=1}^{3}\left(U_{eq_k} - \widehat{U}_{eq_k}\right) \cdot k_{eq_k} \cdot \left(1 - g\left(U_{net_k}\right)^2\right) \cdot W_{y_{jk}} \quad \text{(III.39)}$$

En fin

$$\frac{\partial E}{\partial W_{z_{ij}}} = -\frac{1}{4}\left[\sum_{k=1}^{3}\left(U_{eq_k} - \widehat{U}_{eq_k}\right)\cdot k_{eq_k}\cdot\left(1 - g\left(U_{net_k}\right)^2\right)\cdot W_{y_{jk}}\right]\cdot\left(1 - g\left(y_{net_k}\right)^2\right)\cdot z_i \quad \text{(III.40)}$$

3.3 La commande hybride

Les lois d'adaptation de $W_{z_{ij}}$ et $W_{y_{jk}}$ dépendent respectivement de la commande équivalente U_{eq_k}. Malheureusement, cette valeur reste inconnue, rendant ainsi ce calcul théoriquement irréalisable. Afin de contourner la problématique du calcul de l'écart $\left(U_{eq} - \widehat{U}_{eq}\right)$, nous proposons une méthode qui remplace cette valeur d'écart par la commande de correction uglis−Floue. De plus, nous normalisons cette commande avant son utilisation dans les lois d'adaptation, afin de prévenir la saturation des poids. En ce qui concerne le passé, le critère E coïncide avec un critère J, tel que :

$$J = \frac{1}{2}\cdot\sum_{k=1}^{3}\left(U^i_{glis-floue}\right)^2 \quad \text{(III.41)}$$

L'avantage de cette substitution est double, d'une part nous réglons la question du calcul de l'écart $\left(U_{eq_k} - \widehat{U}_{eq_k}\right)$

et d'autre part, cette substitution a un impact significatif sur la réduction dubroutement.

Ainsi, les règles d'adaptation des poids dans la couche de sortie peuvent s'exprimer :

$$\begin{cases} W_{y_{jk}}(t) = W_{y_{jk}}(t-1) - \alpha\cdot\frac{\partial E}{\partial W_{y_{jk}}} \\ et \\ \frac{\partial E}{\partial W_{y_{jk}}} = -\left[u^k_{glis-floue}\cdot k_{eq_k}\cdot\frac{1}{2}\left(1 - g\left(U_{net_k}\right)^2\right)\right]\cdot Y_{out_j} \end{cases} \quad \text{(III.42)}$$

Par ailleurs, les règles d'ajustement des poids au sein de la couche cachée sont les suivantes :

$$\begin{cases} W_{z_{ij}}(t) = W_{z_{ij}}(t-1) - \alpha\cdot\frac{\partial E}{W_{z_{ij}}} \\ et \\ \frac{\partial E}{\partial W_{z_{ij}}} = -\frac{1}{4}\left[\sum_{k=1}^{3} u^k_{glis-floue}\cdot k_{eq_k}\cdot\left(1 - g\left(U_{net_k}\right)^2\right)\cdot W_{y_{jk}}\right]\cdot\left(1 - g\left(y_{net_k}\right)^2\right)\cdot z_i \end{cases} \quad \text{(III.43)}$$

3.4 Paramètres de simulation

Pour mettre en œuvre la commande en a Employé l'environnement de simulation de Matlab.

Les figures suivantes représentent la simulation des commandes neurone-floue en mode de glissements applique sur le robot puma 560 :

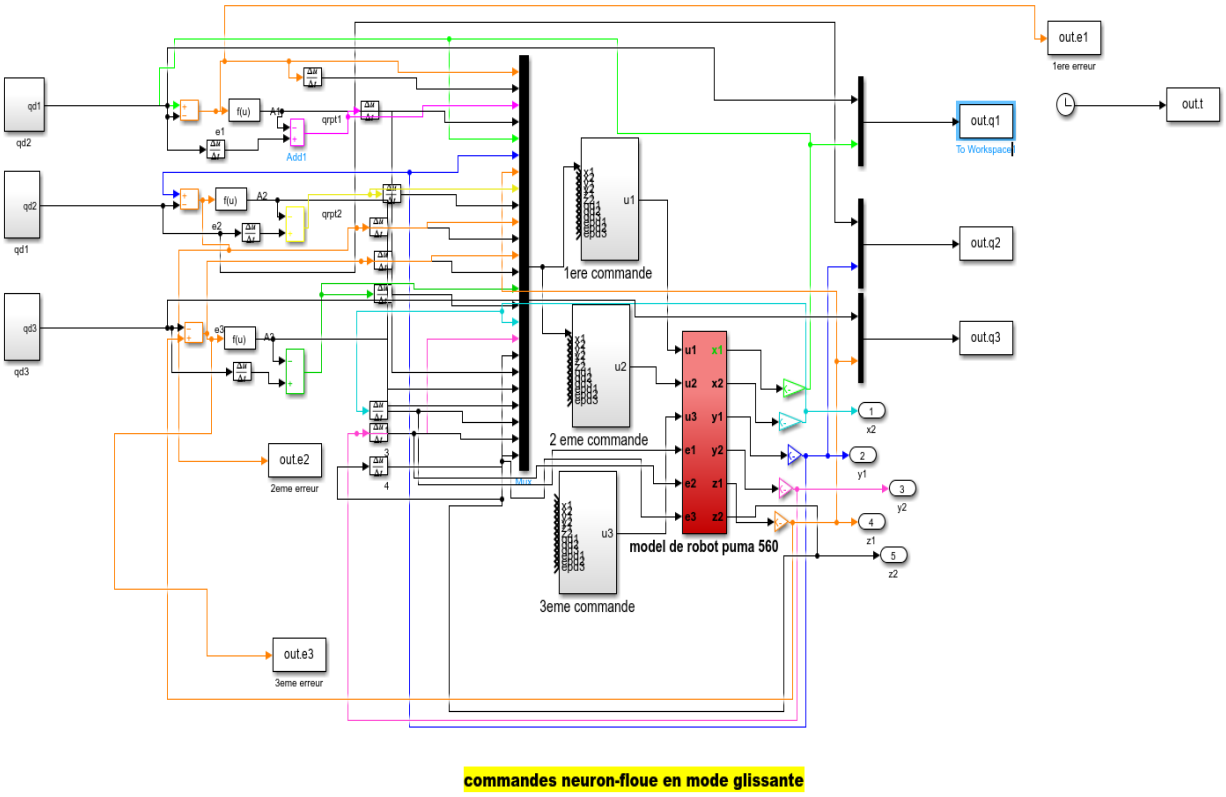


Figure III. 11:la simulation des commandes neurone-floue glissante.

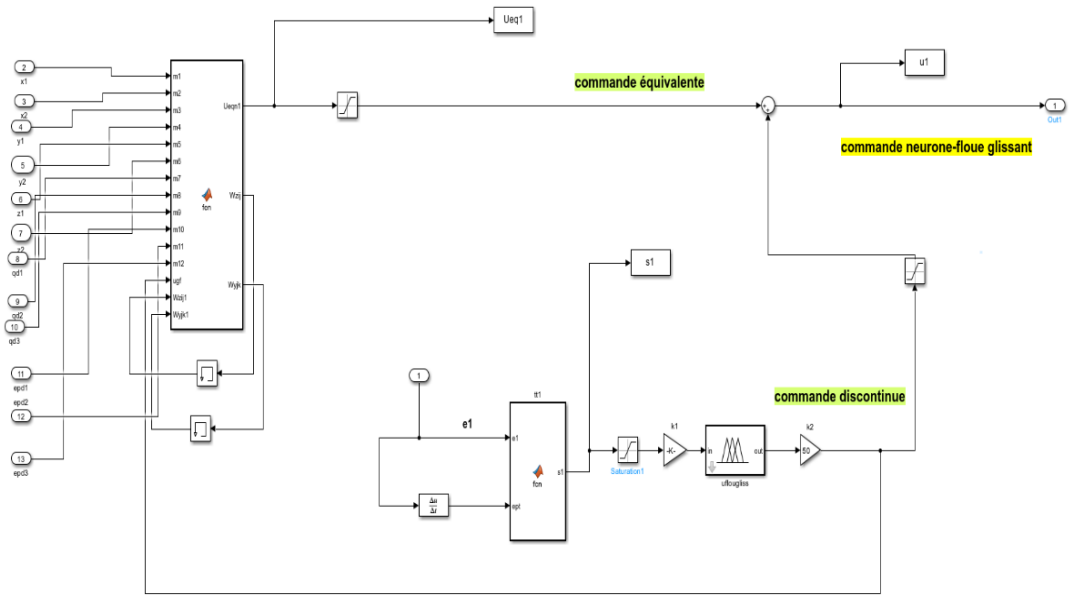


Figure III. 12: la commande applique sur chaque articulation.

La commande Neuro-floue-glissante, élaborée dans cette section, a été validée à travers une simulation sur le modèle dynamique du robot manipulateur PUMA 560. Les résultats de cette commande ont été obtenus avec les paramètres suivants :

Les poids initiaux de la couche cachée ont été sélectionnés de manière aléatoire entre ± 0.01, tandis que ceux de la couche de sortie ont été choisis entre ± 0. 05, avec un taux d'apprentissage $\alpha = 0.\ 01$. Les valeurs maximales pouvant être atteintes par les commandes équivalentes sont :

Keq = [50, 50, 50]. Le vecteur des paramètres d'ajustement est r = [0. 7, 0. 7, 0. 7], la largeur de la bande est $\Phi = [0.\ 3, 0.\ 3, 0.\ 3]$ et les gains sont K = [50, 50, 50]. Les paramètres des surfaces de glissement sont définis par $\eta = [30, 30, 30]$.

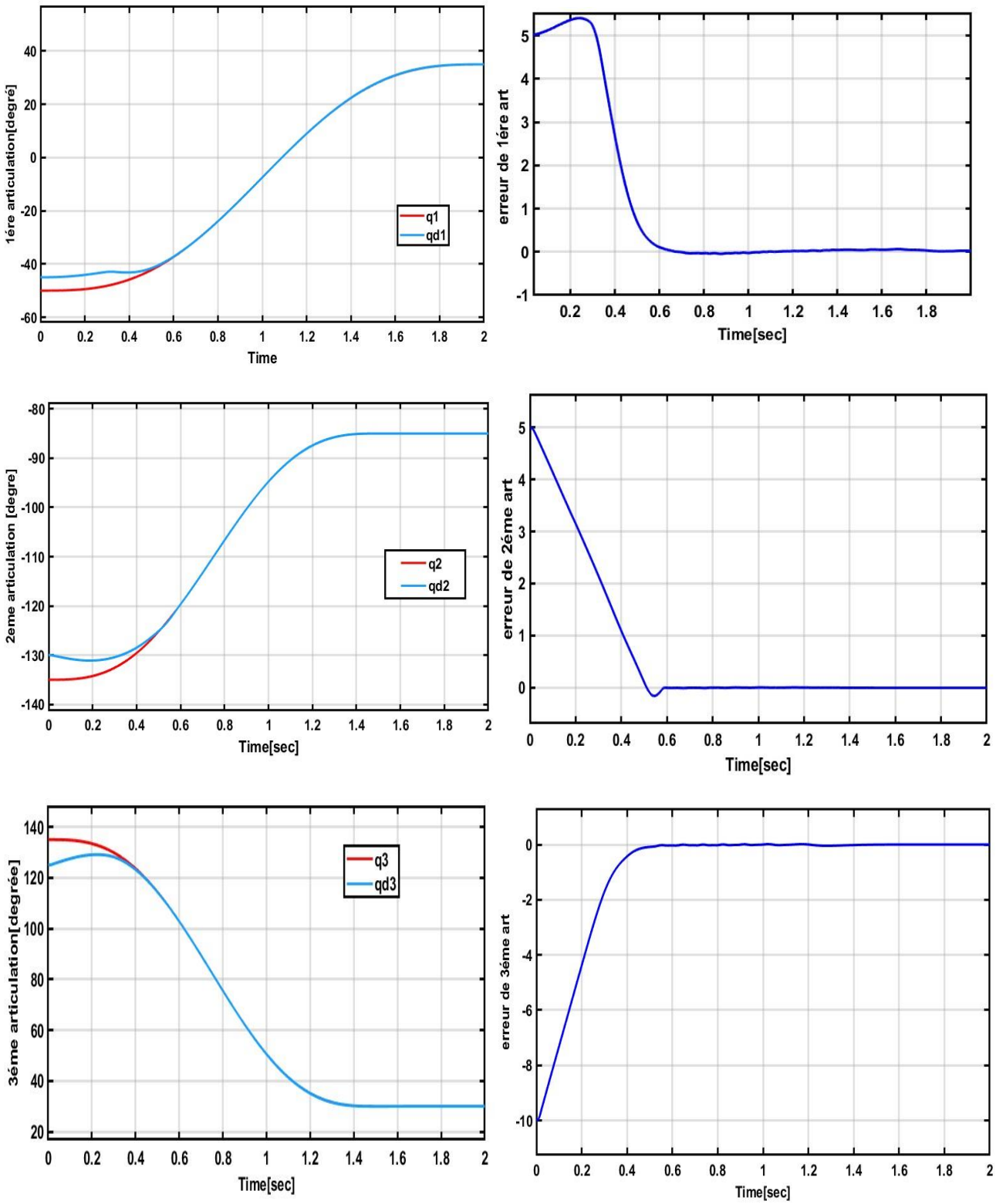


Figure III. 13: Positions articulaires et erreurs de poursuite de la commande Neuro-floue-Glissante.

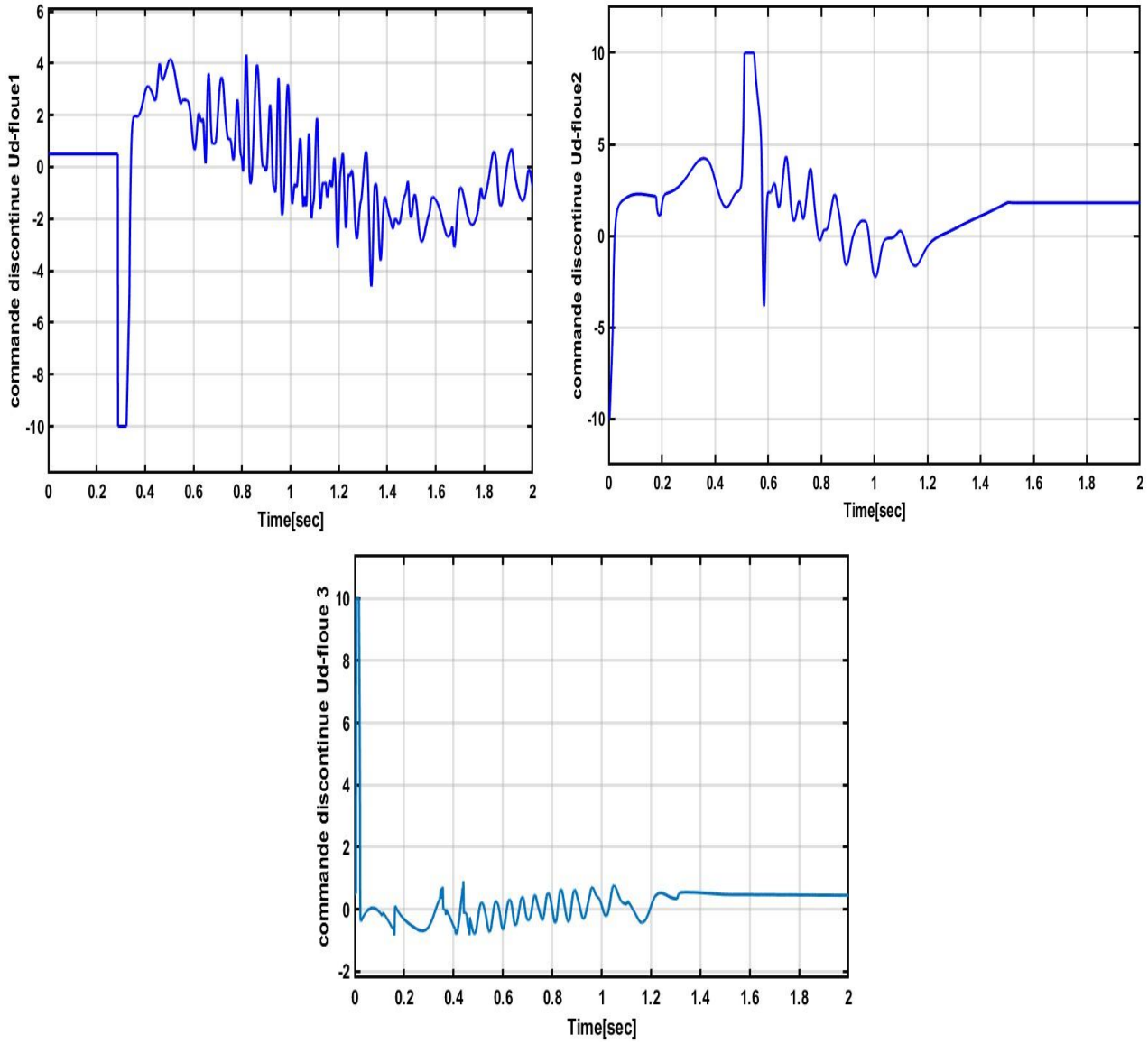
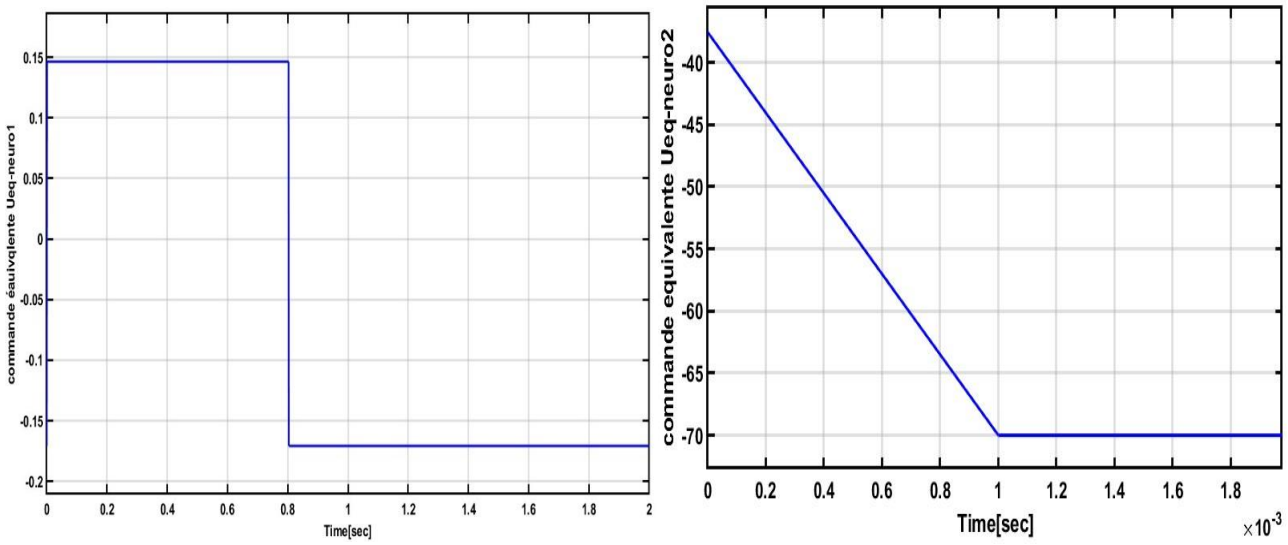


Figure III. 14: Commandes discontinues type floue.



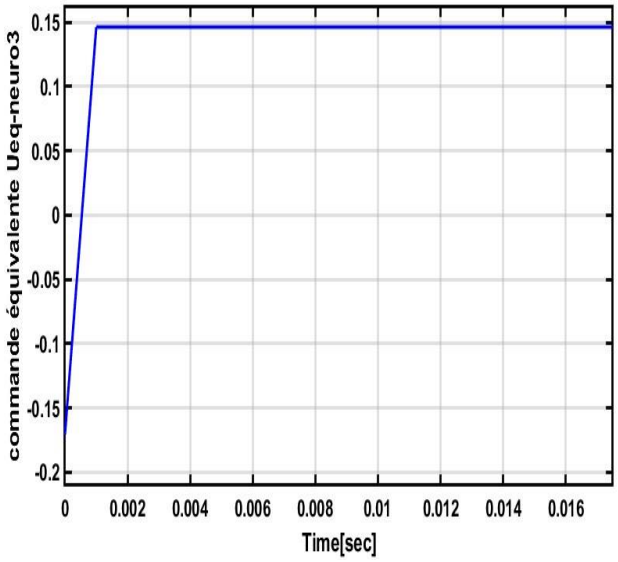


Figure III. 15: commandes équivalentes neuronale.

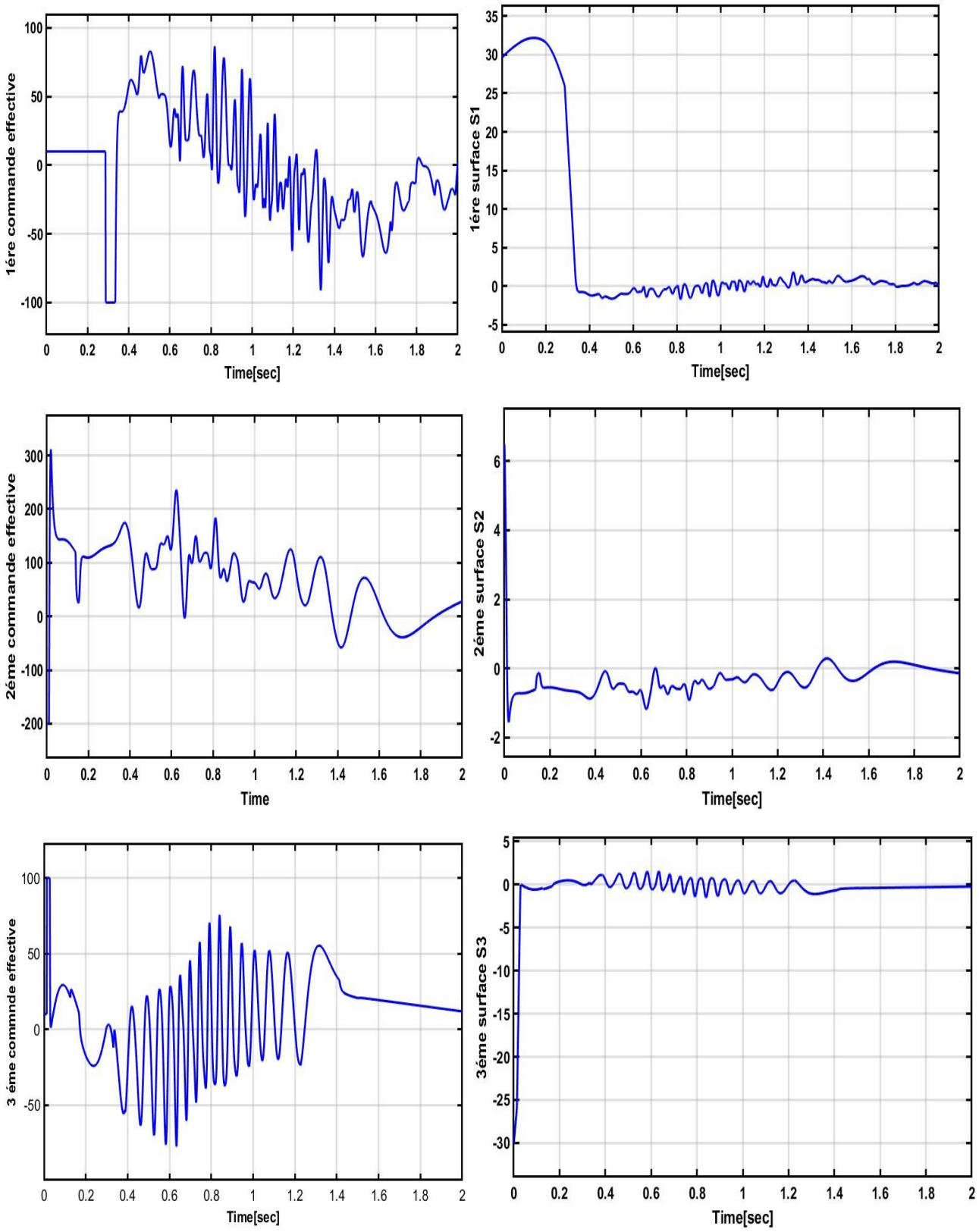


Figure III. 16: Surfaces de glissement et commandes effectives de la commande Neuro-floue-Glissante.

3.5 Commentaires sur les résultats

Les résultats présentés précédemment concernent la commande neuro-floue-glissante proposée. L'observation de la simulation illustrée par la figure (III.11) met en évidence des erreurs de poursuite modestes, accompagnées de dépassements qui demeurent dans des limites acceptables. Par ailleurs, les temps de réponse, indicateurs essentiels du comportement en régime transitoire, se révèlent également réduits.

La régulation discontinue de type flou, ainsi que la commande équivalente déterminée à l'aide d'un réseau de neurones, sont représentées par les figures (III.12) et (III.13). Ces illustrations mettent en évidence que les valeurs adoptées par les commandes floues demeurent faibles, tandis que les commandes équivalentes obtenues via l'estimation du réseau de neurones présentent des résultats jugés satisfaisants.

En remarque que :

	Commande glissante			**Commande hybride**		
	1 ere art	2ème art	3ème art	1 ere art	2ème art	3ème art
Max(erreur)	Vmax=10	Vmax=5	Vmin=-5	Vmax=5	Vmax=5	Vmin=-10
A t=0.5s (l'erreur)	t=0.056s	t=0.090s	t=0.020s	T=0.002 s	t=0.005s	t=0.005s

Tableau III 1:comparaison entre les erreurs des commandes.

Nous déduisons que la commande neurone-floue glissante diminue la probabilité de valeur erronée. De plus, la commande neuro-floue glissante élimine le problème de chatterning, et minimise les temps de réponses.

4. Conclusion

Dans ce chapitre, une stratégie de commande neuro-floue-glissante a été élaborée et testée, tant au moyen de simulations qu'à travers des expérimentations pratiques. Cette approche a permis, dans un premier temps, de piloter le robot manipulateur dans un contexte de suivi de trajectoires. Les avantages et les performances de cette méthode de commande ont été clairement mis en évidence. Parmi ces points forts, il convient de souligner la capacité à calculer la commande équivalente sans nécessiter une connaissance explicite du modèle dynamique du système à contrôler, la réduction significative du phénomène de broutement, ainsi qu'une robustesse accrue face aux perturbations. Concernant cette méthode, la dynamique souhaitée pour le système en boucle fermée est spécifiquement définie par des surfaces de glissement non linéaires, établies en amont. Dans le chapitre suivant, l'accent sera mis sur la commande récursive par back stepping, afin d'introduire une méthode systématique pour la conception et la synthèse des surfaces de glissement.

Chapitre IV :

implémentation des commandes par des algorithmes génétiques sur Arduino

Chapitre IV : implémentation des commandes par des algorithmes génétiques sur Arduino

1. Introduction

Les moteurs à courant continu (DC) sont largement utilisés dans les systèmes automatisés et robotiques en raison de leur simplicité de commande, leur réponse rapide et leur large gamme d'applications. Cependant, la commande optimale de ces moteurs, notamment en présence de perturbations, d'incertitudes ou de non-linéarités, demeure un défi dans certaines situations réelles.

Dans ce chapitre, nous abordons l'implémentation avec deux genres de contrôleurs : le PID (Proportionnel, Intégral, Dérivé) et les algorithmes génétiques PSO : Particle Swarm Optimization et GWO : Grey Wolf Optimization, dans le cadre de leur mise en œuvre pour le contrôle de la vitesse d'un moteur à courant continu (JGB37-3530). Après une présentation des caractéristiques du moteur DC et des principes de base des algorithmes génétiques, nous détaillons l'approche utilisée pour encoder les paramètres de commande, définir la fonction objective, et exécuter les étapes d'évolution en utilisant l'environnement MATLAB combiné avec la carte Arduino.

2. Le matériel et logiciels utilises

2.1 Modèle de moteur

Le moteur que nous avons choisi est un moteur à courant continu fait pour des applications requérant une lecture exacte de la position, comme le retour d'information sur les mouvements d'un robot. Il est doté d'un encodeur à effet Hall fixé sur l'arbre, offrant une sortie en quadrature pour garantir une précision maximale.

Catégorie	Valeur
Model	JGB37-3530.
Catégorie	Motoréducteur CC avec un encodeur Hall
Diamètre	37 mm.
Construction	Engrenage en métal intégral

Tableau IV. 1: Caractéristiques General de moteur JGB37-3530.

Détails sur l'encodeur

Catégorie	Valeur
Type	Encodeur à effet Hall
Impulsions par tour (PPR)	16
Fonction	Offre unretour précis de la rotation, assurant lecontrôle et la surveillance
Construction	Engrenage en métal intégral

Tableau IV. 2 : Caractéristiques techniques d'encodeur.

Détail sur le moteur

Catégorie	Valeur
Tension	12 V ou 24 V
Vitesse	Habituellement de 12 à 1 600 tours par minute, avec des options de vitesse spécifiques disponibles
Rapport de réduction	Le JGB37-3530 se décline en plusieurs rapports de réduction, influençant la vitesse et le couple à la sortie.
Contrôle de la vitesse PWM	Compatible avec la modulation de largeur d'impulsion pour ajuster la vitesse.
Niveau sonore	Fonctionnement discret grâce à la conception métallique des engrenages.
Installation	Les supports sont souvent disponibles en vente séparée pour faciliter le montage.
Usages	Idéal pour des applications demandant un contrôle précis, une modulation de la vitesse et un retour d'information
Notamment	Bras robotiques, Systèmes automatisés, Démonstrations motorisées, Impression 3D et machines CNC.

Tableau IV. 3: Caractéristiques techniques détalée de moteur.

Figure IV. 1: Moteur à courant continu JGB37-3530. 12V avec encodeur.

L'équation (IV.1) représente l'équation d'état de moteur [22]

$$\dot{x} = \begin{bmatrix} -5.5 & 37.5 \\ -0.01852 & -0.1481 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.3704 \end{bmatrix} u$$
$$y = [1 \quad 0]x \tag{IV.1}$$

A partir de (IV.1) on a calcul la fonction de transfert de système :

$$h(s) = \frac{13.89}{s^2 + 5.648s + 1.509} V(s) \tag{IV.2}$$

2.2 La carte Arduino UNO

L'Arduino Uno figure parmi les cartes Arduino les plus prisées. Il s'agit tout simplement d'une évolution du tout premier Arduino. La carte Uno est extrêmement facile à utiliser, elle est la plus couramment employée et la mieux documentée. C'est un choix idéal pour les novices, car elle est abordable, suffisamment puissante pour faire ses premiers pas et offre un bon nombre d'entrées et de sorties pour la plupart des réalisations. En outre, l'Uno est compatible avec quasiment toutes les cartes d'extension, ce qui lui confère de vastes opportunités d'expansion.[23]

Figure IV. 2:carte Arduino UNO.

Les principales caractéristiques de la carte Arduino Uno sont présentées dans le tableau suivant[24]:

Catégorie	Valeur
Microcontrôleur	Atmega328
Fréquence	d'horloge 16MHz
Tension d'entrée (recommandée)	7 − 12V
Tension d'entrée(limites)	6 − 20V
Ports-numériques	14 E/S
Ports-analogiques	6 entrées analogiques
Courant maxi. par broche d'E/S	40mA
Courant maxi.par broche	50mA
Tension de service	5V
Mémoire	32Ko Flash
Interface	USB
Dimensions	68 × 53mm

Tableau IV. 4: Caractéristiques techniques de la carte Arduino Uno.

2.3 MATLAB-Simulink de Math Works

MATLAB-Simulink constitue un cadre avancé pour la modélisation et la simulation, employé dans la conception et l'analyse de systèmes dynamiques. Il propose une interface graphique attrayante, conçue pour les ingénieurs et les chercheurs, leur donnant la possibilité de modéliser des systèmes complexes de manière intuitive.

Le Paquet de support Simulink pour Arduino

Pour faciliter la communication entre Simulink et la carte, il est essentiel d'ajouter le paquet "Paquet de support Simulink pour Arduino". Ce paquet peut être téléchargé à partir du site de MATLAB ; il suffit de récupérer l'installateur du paquet et de l'exécuter dans MATLAB. La création d'un compte Math works est requise. Ce paquet permet de créer des algorithmes dans Simulink en utilisant des blocs pour modéliser des systèmes et leurs commandes. Les blocs configurent et permettent d'accéder aux périphériques reliés à la carte Arduino, et il est même possible d'intégrer un bloc représentant un programme MATLAB. Une fois le modèle Simulink développé, il peut être simule pour ajuster les paramètres de l'algorithme afin d'obtenir les résultats escomptés. Par la suite, il est envisageable de transférer l'algorithme final dans la mémoire de la carte Arduino pour un fonctionnement autonome, ce qui signifie qu'il ne sera plus nécessaire de le gérer via un ordinateur. [25]

3. Algorithmes utilisés

3.1 PSO (Optimisation d'Essaim de Particules)

L'optimisation par essaim de particules (PSO) est une méthode stochastique d'optimisation par population conçue par Dr. Eberhart et Dr. Kennedy en 1995, qui s'inspire des comportements sociaux des oiseaux se regroupant ou des bancs de poissons.

Principe de l'Algorithme PSO

Dans l'algorithme PSO, chaque membre de la population est défini comme une particule, tandis que l'ensemble de la population est nommé Swarm. Il est essentiel de mentionner que la particule peut tirer avantage des déplacements des autres particules de la même population pour ajuster sa position et sa vitesse au cours du processus d'optimisation. Chaque membre s'appuie sur les informations locales accessibles concernant le mouvement de ses voisins les plus proches pour décider de son propre déplacement. Des directives très simples telles que "rester à proximité des autres membres", "avancer dans la même direction", "se déplacer à la même vitesse" suffisent à assurer la cohésion de l'ensemble du groupe.

Au début de l'algorithme, un essaim est réparti aléatoirement dans l'espace de recherche, chaque particule ayant également une vitesse aléatoire. Par la suite, à chaque itération : Chaque particule est capable d'évaluer la qualité de sa position et de mémoriser sa meilleure performance,

c'est-à-dire la position optimale qu'elle a atteinte jusqu'à présent (qui peut parfois correspondre à sa position actuelle) ainsi que la qualité (la valeur de la fonction à optimiser à cette position). Chaque particule peut interroger un certain nombre de ses congénères voisins pour obtenir leur meilleure performance respective.

A chaque itération, Chaque Particule sélectionne la meilleure parmi les meilleures performances dont elle a connaissance, ajuste sa vitesse en fonction de cette information et de ses propres données, et se déplace en conséquence. Sur la base des informations à sa disposition, une particule doit déterminer son prochain mouvement, c'est-à-dire définir sa nouvelle vitesse. Pour cela, elle combine trois éléments : sa vitesse actuelle, sa meilleure position actuelle et la meilleure performance (vitesse et position) de ses voisines. L'élément aléatoire joue un rôle en introduisant une modification aléatoire limitée des coefficients de confiance, ce qui encourage l'exploration de l'espace de recherche.

Naturellement, pour être implémenté, tout ceci est formalisé à travers des équations de mouvement. Un aspect intéressant est que, contrairement à de nombreuses autres heuristiques qui demeurent strictement expérimentales, une analyse mathématique existe pour définir les conditions de convergence ainsi que le choix des paramètres.[26]

3.2 GWO (l'optimisation par loup gris)

L'algorithme (GWO) est une technique métaheuristique, présentée par Seyd Ali Mirjalili, Seyed Mohammed Mirjalili et Andrew Lewis en 2014. Cette méthode s'inspire de la structure hiérarchique et des stratégies de chasse des loups gris dans leur environnement naturel.[27]

Principe de l'Algorithme GWO

Les meutes de loups gris vivent en collectivité et possèdent une structure hiérarchique bien définie. L'organisation sociale des loups gris se décompose en quatre classes, incluant les loups alpha, bêta, delta et oméga :

les α loups: les dirigeants du groupe.

les β loups: les meilleurs candidats à devenir les α-loups.

les δ loups: protéger du danger et fournir de la nourriture au groupe, controler les ω loups .

les ω loups: les pions, les loups âgés, les chasseurs et les gardiens.

L'algorithme GWO modélise le comportement social des loups gris par le biais d'un modèle mathématique. Dans chaque itération, les nouvelles solutions produites par l'algorithme sont élaborées à proximité des positions du loup alpha X1 (la solution la plus performante générée lors de l'itération précédente), du loup bêta X2 (la seconde meilleure solution produite lors de la précédente itération) et du loup delta X3 (la troisième meilleure solution issue de la précédente itération).[27]

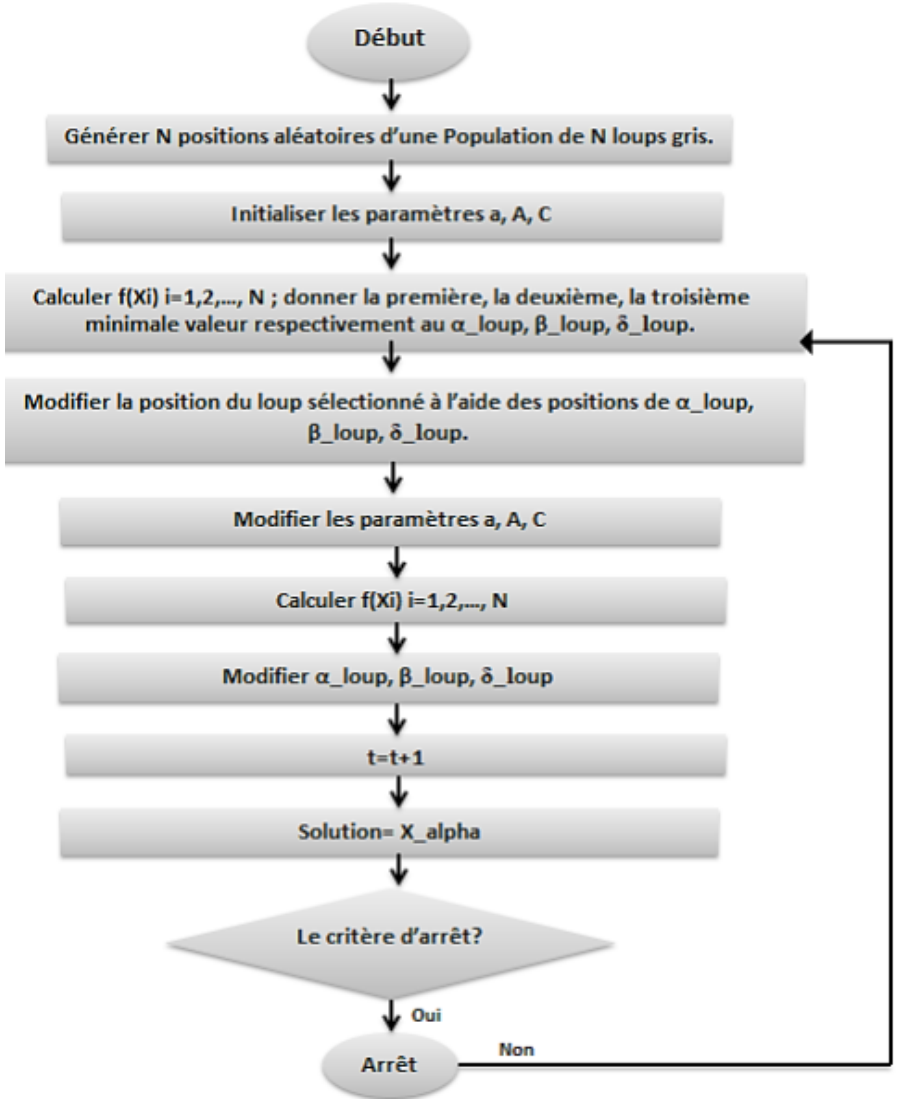


Figure IV. 3: L'organigramme de l'algorithme GWO.

4. Conception du régulateur PID

Après avoir examiné les performances des deux méthodes de régulation par le biais des algorithmes génétiques. Cela en substituant le calcul aléatoire du contrôleur PID par le calcul utilisant des algorithmes génétiques pour le contrôleur PID Concernant le moteur à courant continu.

La figure suivante illustre une interface graphique sur Matlab présentant les résultats des calculs de PID réalisés par les méthodes PSO et GWO.

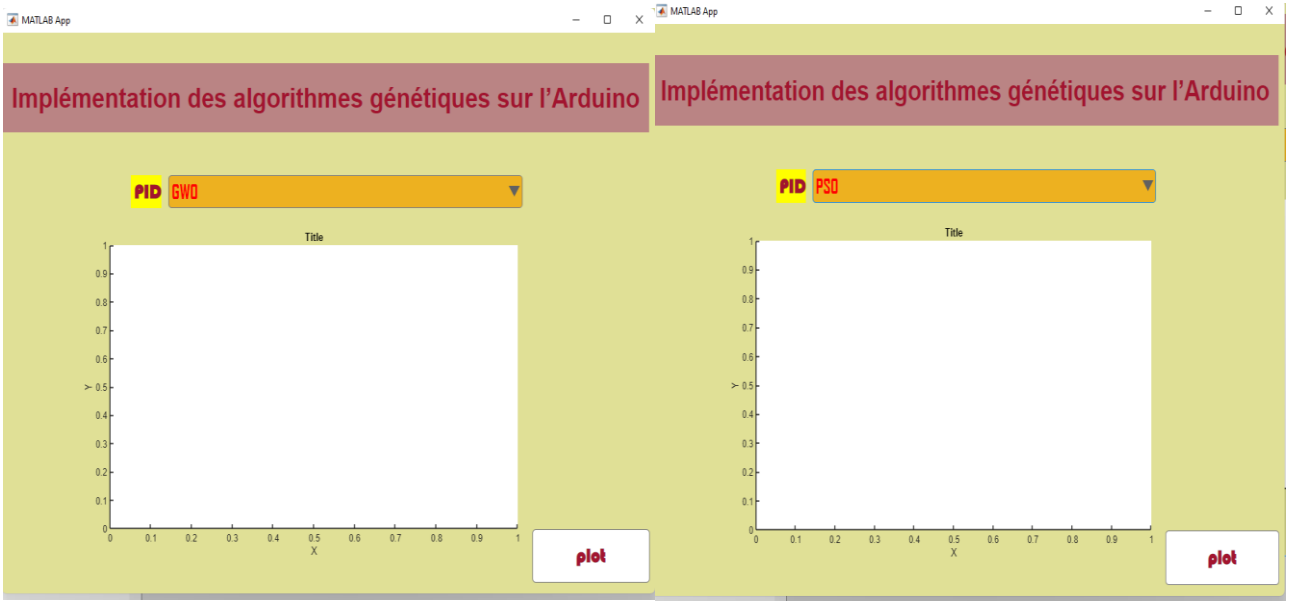


Figure IV. 4: interface graphique sur Matlab.

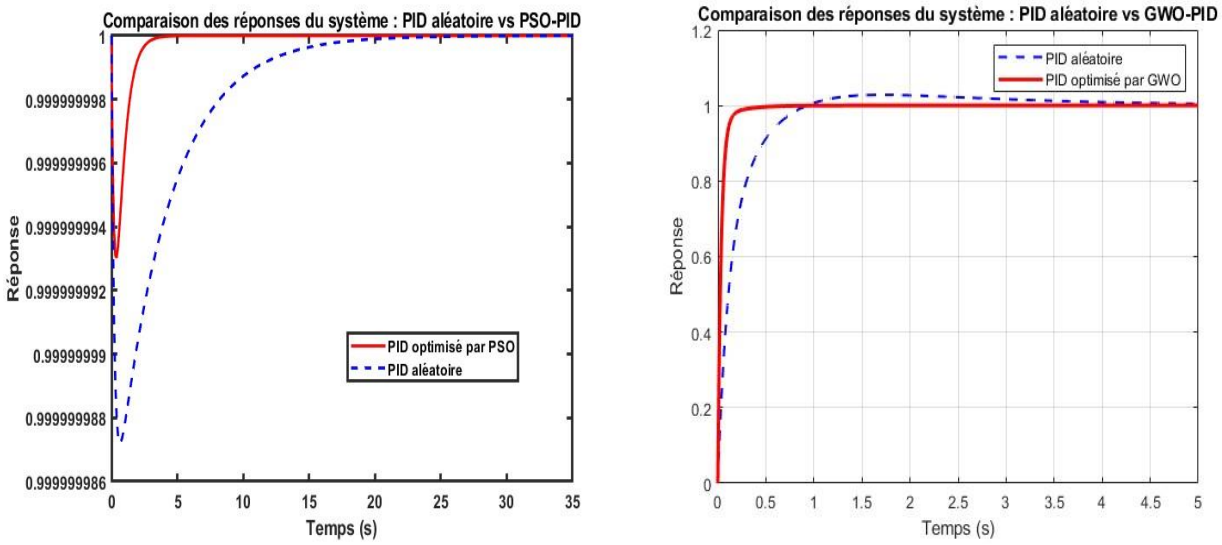


Figure IV. 5:PID aléatoire vs PSO-PID et GWO-PID.

PID\Algorithmes	PSO	GWO
KP	50	10
KI	12	10
KD	10	3.7497

Tableau IV. 5 : les valeurs de PID calculée

Les graphiques (FigureIV.5), et le tableau (TableauIV.5) illustrent les résultats des programmes de comparaison PID sur le moteur à courant continu, aléatoires et calculés avec les algorithmes génétiques (PSO et GWO). On constate que ces algorithmes produisent des résultats PID parfaits, avec un temps de réponse minimal pour les calculs aléatoires.

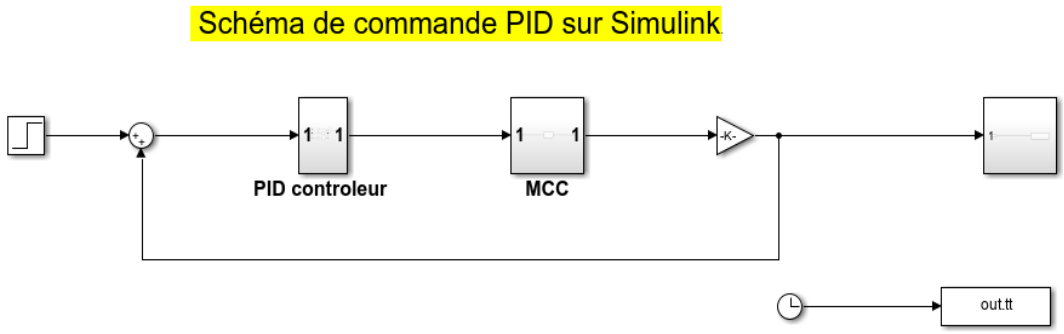


Figure IV. 6:schéma de commande PID sur Simulink.

5. Implémentation sur la carte Uno

Nous avons employé une carte Arduino (Arduino Uno) afin de recueillir les données, Veuillez communiquer les suivantes :

1. Alimentation de 12 V pour alimenter le pilote L298N.
2. La GND pour garantir une référence partagée pour les signaux.
3. Connecter le L298N :

   -Les pins OUT1 sont reliées à la source d'alimentation du moteur.

   -Les pins IN1 et IN2 reçoivent l'ordre concernant la direction du moteur, tandis que la broche ENA reçoit le signal PWM émis par l'Arduino.
4. L'appareil qui encode les données de localisation pour leur transmission.

L’image ci-dessous présente le diagramme de connexion entre les différents composants.

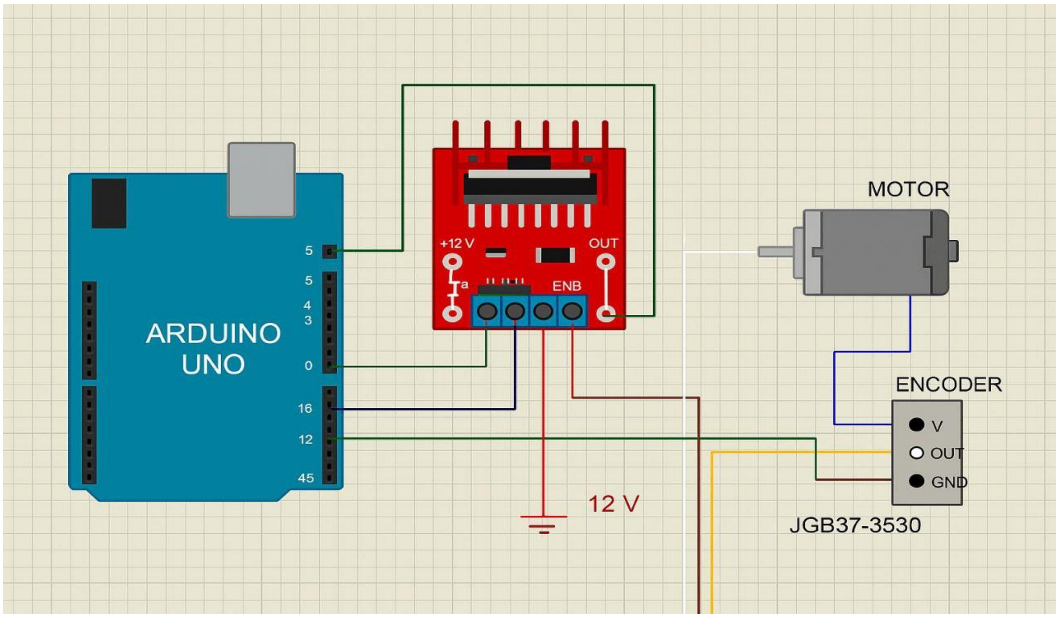


Figure IV. 7: Schéma de câblage du moteur avec la carte Arduino.

L'image ci-après illustre le montage effectué :

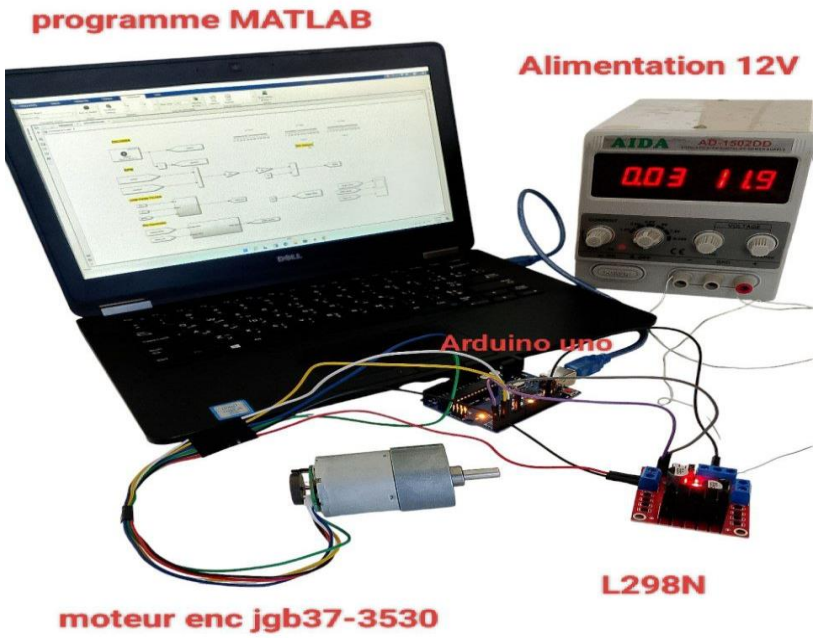


Figure IV. 8: Expérience mise en place.

Nous employons un logiciel Simulink pour l'Arduino Uno qui effectuera une lecture constant e des valeurs. Tout d'abord, nous avons déjà déterminé la valeur de vitesse et du PID précédemment calculé

À chaque cycle, l'Arduino Uno reçoit une nouvelle donnée fournie par l'encodeur. Dès que la valeur est récupérée, elle est directement envoyée à l'aire de travail de MATLAB.Pour af ficher les résultats de la rotation par minute (RPM), le filtre et le signal PWM.

Le programme Simulink utilisé pour lire les valeurs transmises est représenté dans l'image su ivante :

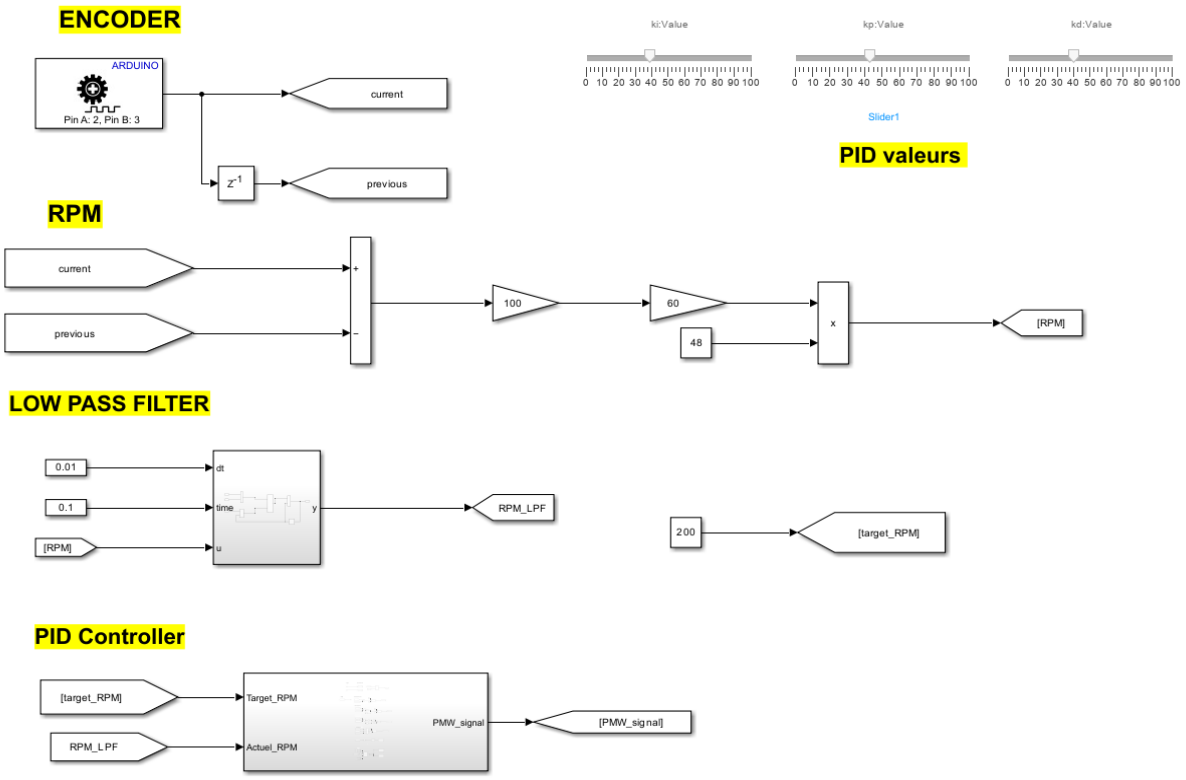


Figure IV. 9 : Programme Simulink utilisé.

6. Résultats et discussion

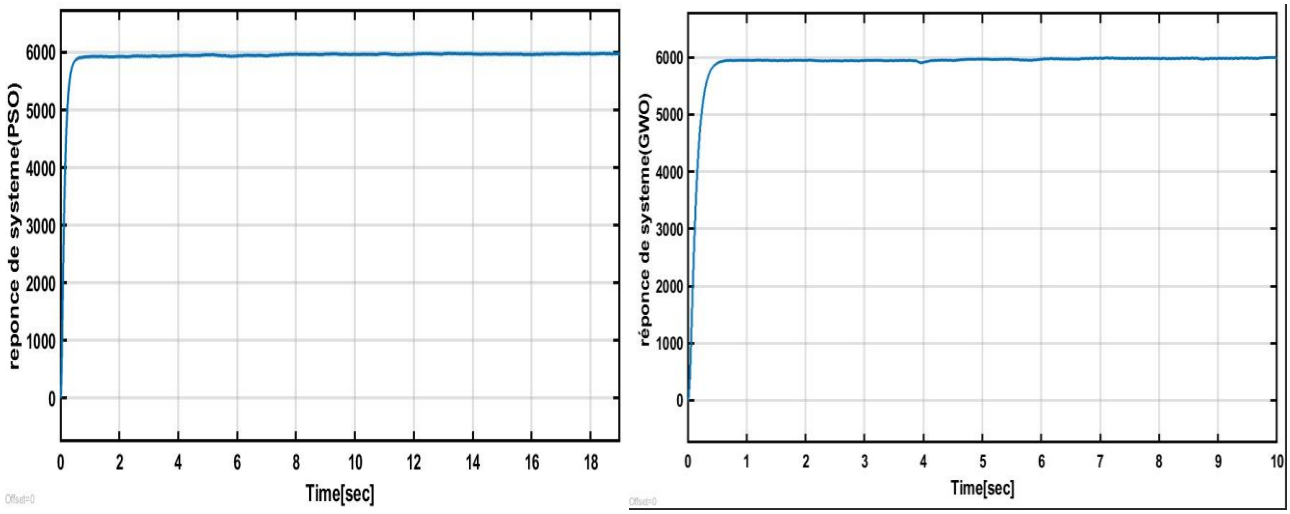
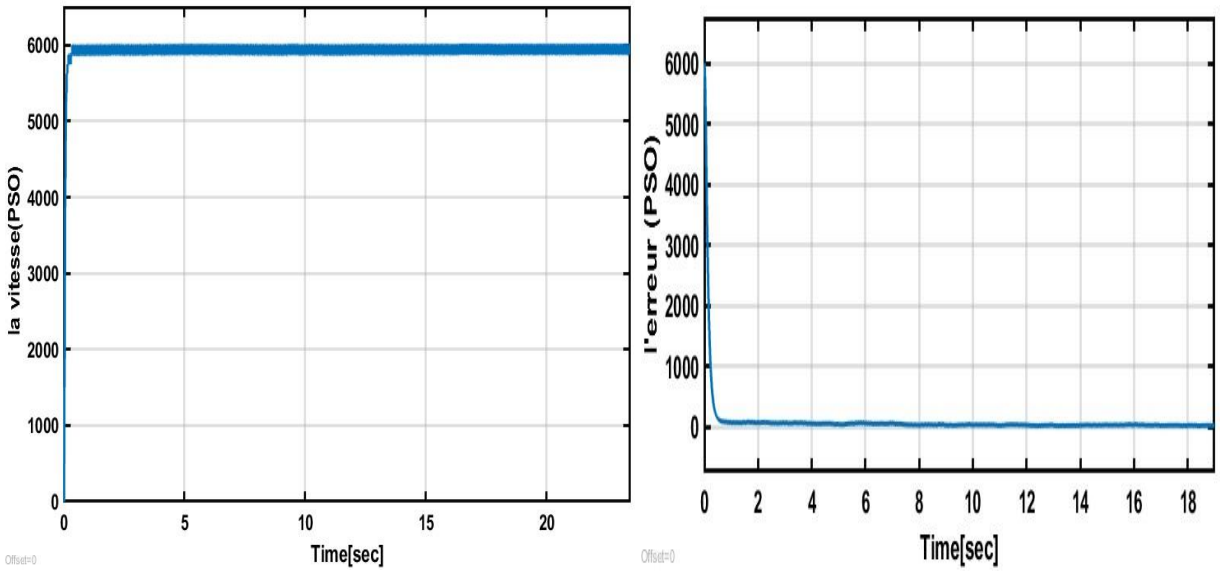


Figure IV. 10 : Courbes des réponses des système en implémentation avec le régulateur PID(PSO\GWO).

La réponse de PSO est également pratiquement identique à celle obtenue avec GWO, les co mmandes produisant presque les mêmes résultats.

	PID(PSO\GWO)		**PID aléatoire**
	PSO	GWO	
Temps de réponse	t=0.001 s	t=0.002 s	t=0.1s

Tableau IV. 6 : comparaisons entre les temps de réponses de PID vs PID(PSO\GWO).



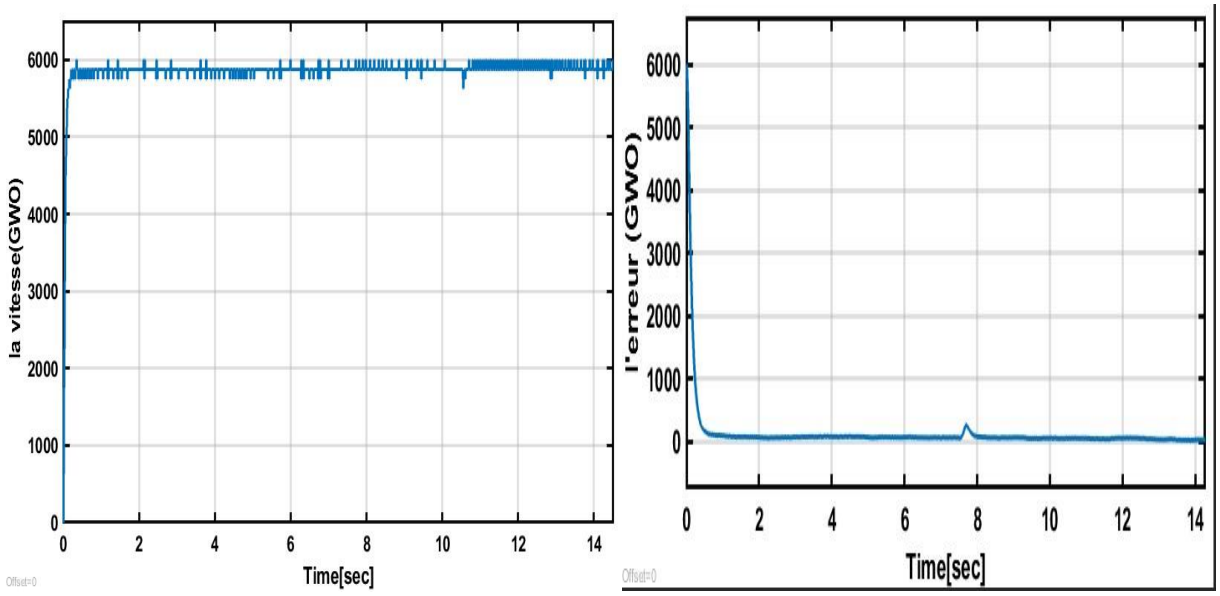


Figure IV. 11 : les vitesse (RPM) et ses erreurs.

	PSO	GWO
Rpm =6000tr\min	≈5994.8	≈5996.7
Valeur de l'erreur	±5.2	±3.24

Tableau IV. 7 : les valeurs de l'erreurs de vitesse.

7. Conclusion

Ce chapitre présente une approche de commande d'un moteur à courant continu (DC) basée sur les algorithmes génétiques (AG), implémentée sur une plateforme Arduino a l'aide de l'environnement MATLAB. L'objectif principal est de faire la commande en temps réelle de la vitesse de moteur et d'intégration de l'algorithme dans l'environnement embarqué. Des résultats expérimentaux sont finalement présentés pour valider l'efficacité de la méthode proposée. Ils montrent que l'utilisation des AG permet une amélioration significative de la qualité de commande du moteur DC. Une comparaison a été présenter entre PID classique et PID-PSO et PID-GW pour valider et assurer les performances de système.

Conclusion générale

Les travaux présentés dans ce travail, portent sur la problématique de commande des systèmes robotiques complexes, en particulier celle du robot manipulateur PUMA 560.

Pour surmonter ce problème nous avons faire la synthèse d'une approche hybride alliant trois techniques puissantes : la commande par mode glissant, les réseaux de neurones artificiels et la logique floue. Dans un premier temps, nous avons présenté la modélisation dynamique du robot PUMA 560, qui constitue une étape essentielle pour toute stratégie de commande avancée. Nous avons ensuite étudié en détail les fondements théoriques et les propriétés respectives de chaque technique de commande intelligente utilisée dans ce mémoire

La synthèse de ces méthodes dans une stratégie de commande hybride a permis de concevoir un contrôleur capable de combiner robustesse, adaptabilité et intelligence décisionnelle. Les simulations réalisées ont montré que cette approche améliore sensiblement la performance du système en matière de précision de suivi de trajectoire, de résistance aux perturbations et de stabilité globale.

L'approche hybride développée dans ce mémoire confirme l'intérêt croissant des méthodes de commande intelligente pour les systèmes non linéaires complexes, et en particulier pour la robotique industrielle, domaine en constante évolution.

Par ailleurs, Pour faire la commande en temps réelle sur les systèmes non linéaire, une seconde partie du projet s'est concentrée sur l'utilisation d'algorithmes génétiques pour le contrôle d'un moteur à courant continu. L'implantation a été réalisée à l'aide de l'environnement de MATLAB et d'une carte Arduino Uno, choisie pour sa fréquence optimale et ses performances adaptées.

Ce travail ouvre la voie à plusieurs perspectives de recherche. Il serait notamment pertinent d'envisager l'implémentation en temps réel de cette commande sur un robot réel afin de valider les résultats obtenus en simulation. De plus, l'optimisation des paramètres du système hybride par des techniques évolutives, telles que les algorithmes génétiques ou les algorithmes de colonies de fourmis, constitue une piste prometteuse pour améliorer davantage les performances du contrôleur.

Références et bibliographies

Reference et Bibliographie

[1] Lucas CARPENTIER et Thibault COURTOIS, « PUMA », 2023.

[2] ALIA KAMEL, « "« PLANIFICATEUR DE TRAJECTOIRE NEURONAL »" », UNIVERSITE EL-HADJ LAKHDAR-BATNA, 2011.

[3] « https://robodk.com/robot/fr/PUMA/560 ».

[4] M. De L', « Etude d'un Bras Manipulateur PUMA 560 sous Matlab », 2019.

[5] George P. Anderson BE, « M o d e llin g a n d S im u la t io n o f a P u m a 5 6 0 M a n ip u la t o r », sept. 1988.

[6] Melle ABDELLI Houaria, « Commande par Mode de Glissement d'une Machine Asynchrone à Double Alimentation ».

[7] V.I. Utkin et K. D. Young, *"Methods for constructing discontinuity planes in multidimensional variable structure systems", Auto. & Remote control, pp.* . 1978.

[8] M. Boubakir Ahcene, « Méthodes de commande par mode de glissement appliquées au bras manipulateur et au système hydraulique CE105 à réservoirs couplés », École Militaire Polytechnique, 2007.

[9] S. Denis et D. Hercouet, « Conception d'un système expert flou appliqué aux contraintes de la conduite de séchoir à bois" », avr. 1995.

[10] G. J. Klir et B. Yuan, , *Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice* , 1st edition,. 1995.

[11] D. Dubois et H. Prade, , *Logique floue et systèmes flous, Hermès,* . 1994.

[12] L. Zadeh, *« Fuzzy sets, » Information and Control,* , vol. 8,. 1965.

[13] B. Khalil, « Commande Intelligente », 2022.

[14] T. Takagi et M. Sugeno, *"Fuzzy identification of systems and its applications to modelling and control", I,* IEEE Tras. Syst., vol. SMC-15,. 1985.

[15] I. Goodfellow, Y. Bengio, et A. Courville, *« Deep Learning, »* , MIT Press,. 2016.

[16] P. Borne, M. Benrejeb, et J. Haggège, *Les réseaux de neurons:* 2007.

[17] , A. Guez et A. Z, *"Solution to the inverse problem in robotics by neural networks", Proc. Int. Conference on Neural Networks,* . 1988.

[18] R. Elsley, *"A learning architecture for control based on back-propagation neural-networks", IEEE Conf. on Neural Networks,* , vol. Vol.2,. 1988.

[19] , D. Psaltis, A. Yamamura, et A. Sideris, *"Neural controllers",* , Proc. IEEE int. 1987.

[20] kamar F, « : Contribution à l'Etude de la Commande d'une Machine Asynchrone Approches Intelligentes », Université Badji Mokhtar-Annaba, 2017.

[21] Hocine Khati., « Commande d'une Architecture de Téléopération par la Carte FPGA. PhD thesis, », Université Mouloud Mammeri de Tizi Ouzou, , 2020.

[22] A. Ma'arif et A. Çakan, « Simulation and arduino hardware implementation of dc motor control using sliding mode controller », *Journal of Robotics and Control (JRC)*, vol. 2, n° 6, p. 582-587, nov. 2021, doi: 10.18196/jrc.26140.

[23] Jean-Christophe Quetin, . « Arduino : Apprivoisez l'électronique et le codage pour don- ner vie à vos projets. », *ENI*, , 2021.

[24] Erik Bartmann, *Le grand livre d'Arduino.* , EYROLLES,. 2013.

[25] « MathWorks. Arduino programming with matlab and simulink. Mathworks.com. ».

[26] M. Bacheri, M. Bekkouche, U. Abdelhamid, I. B. Mostaganem, D. De, et G. Electrique, « MEMOIRE Soutenu le 11 / 07 / 2021 devant les jurys composé de : MINISTERE DE L'ENSEIGNEMENT SUPEREUR ET DE LA RECHERCHE SCIENTIFIQUE Identification des paramètre d'une machine asynchrone par méthode PSO (particle swarm optimization) ».

[27] H. Reddad, M. Zemzami, N. El Hami, et N. Hmina, « Optimisation métaheuristique et en application mécatronique Metaheuristic optimization and mechatronic application », 2022.

Annex

Circuit L298N

L'élément L298N est un pré actionneur doté de deux ponts H, offrant la possibilité de réguler à la fois la vitesse et l'orientation de deux moteurs à courant continu de manière simultanée. Le module en charge des moteurs à courant continu fonctionne avec des tensions se situant entre 5 et 35V, et un courant maximal de 2A.

Il est important de noter que ce composant provoque une diminution de tension de 2V. De ce fait, si vous commandez notre moteur de 12V, il ne pourra délivrer que 10V, ce qui empêche d'atteindre sa vitesse maximale. Les broches OUT1 et OUT2 sont connectées à l'alimentation du moteur. Le module est alimenté par les connecteurs 12V et GND. Les instructions pour contrôler le moteur et l'ENA sont transmises aux broches IN1 et IN2.

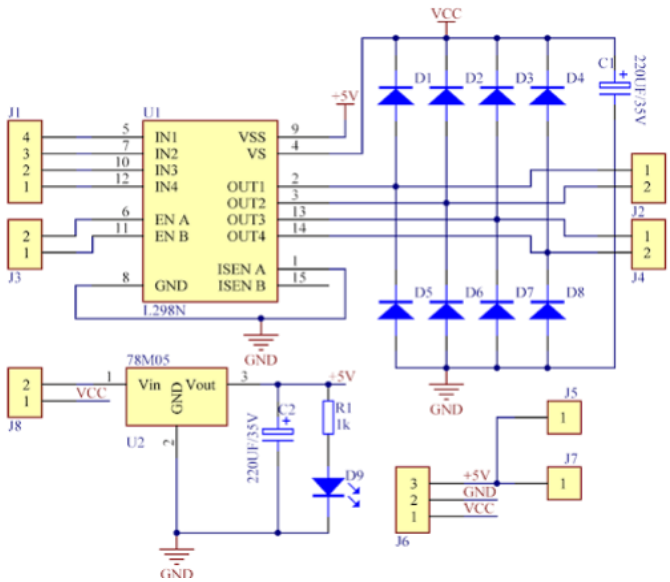
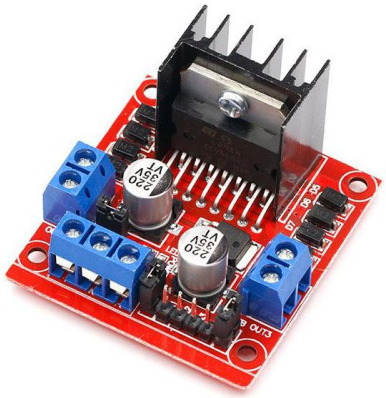


Schéma du circuit électronique du L298N.

Catégorie	Valeur
Tension de fonctionnement	4V à 46V.
Courant total (DC)	jusqu'à 4 A.
Basse	tension de saturation
Protection	contre la surchauffe, Diodes de protection.
Régulateur embarqué	5V à faible chute de tension, accessible par l'utilisateur.
Alimentation logique	5V
Puissance maximale	25W.
Direction du moteur.	Deux voyants
Température de stockage	-25 à +135.

Caractéristiques techniques de L298N.

La Figure suivante présente le module L298N.



le module L298N

Le tableau ci-dessous illustre les fonctions des pins du circuit L298N.

Nom de la pin	Description
S_A	Moteur A
S_B	Moteur B
+12V	Entrée d'alimentation 12V
GND	Terre.
+5V	Sortie 5V
E_A 2	Entrées logiques pour le moteur A
E_B 2	Entrées logiques pour le moteur B
EN_A	Contrôle du moteur A
EN_B	Contrôle du moteur B
EN_5V	Activation 5V

Fonctions des pins du L298N.