PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
- TLEMCEN -

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
- تلمسان -

Mémoire de fin d'études

Pour l'obtention du diplôme de Master

Filière : Automatique

Spécialité : Automatique

Présenté par :

Abdelhadi AOUAICHIA et Othmane BOUHALLOUFA

Thème :

Synthèse sur l'avancement théorique et l'application des méthodes d'intelligence artificielle appliquées sur les régulateurs automatiques

Soutenue à huis clos, le 09/09/2020, devant le jury composé de :

Dr. Ghouti ABDELLAOUI,	MCB	ESSA-Tlemcen	Président
Dr. Sidi Mohammed ABDI,	MCB	ESSA-Tlemcen	Directeur de mémoire
Dr. Fouad MALIKI,	MCB	ESSA-Tlemcen	Examinateur 1
Pr. Ahmed TAHOUR,	Prof	ESSA-Tlemcen	Examinateur 2

Année universitaire : 2019/2020

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
- TLEMCEN -

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
- تلمسان -

Mémoire de fin d'études

Pour l'obtention du diplôme de Master

Filière : Automatique

Spécialité : Automatique

Présenté par :

Abdelhadi AOUAICHIA et Othmane BOUHALLOUFA

Thème :

Synthèse sur l'avancement théorique et l'application des méthodes d'intelligence artificielle appliquées sur les régulateurs automatiques

Soutenue à huis clos, le 09/09/2020, devant le jury composé de :

Dr. Ghouti ABDELLAOUI,	MCB	ESSA-Tlemcen	Président
Dr. Sidi Mohammed ABDI,	MCB	ESSA-Tlemcen	Directeur de mémoire
Dr. Fouad MALIKI,	MCB	ESSA-Tlemcen	Examinateur 1
Pr. Ahmed TAHOUR,	Prof	ESSA-Tlemcen	Examinateur 2

Année universitaire : 2019/2020

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

À nos parents pour leur dévouement,

à nos proches pour leurs ténacités,

à tous nos enseignants,

et à tout lecteur de ce mémoire.

Remerciements

Tout d'abord, nous remercions le bon Dieu tout-puissant, de nous avoir donnés la force et l'audace de dépasser toutes les difficultés, et de mener à bien ce travail. Au nom d'ALLAH le clément et le miséricordieux.

Nous tenons à remercier notre encadrant et enseignant monsieur Sidi Mohammed ABDI qui nous a guidés avec cordialité et bienveillance durant le temps de ce mémoire. Merci pour toute l'aide que tu nous as apportée, pour ton encouragement et pour ton enthousiasme permanent.

De même, nous souhaitons remercier les différents membres constituant le jury de mémoire pour avoir accepté de donner de leur temps pour évaluer notre travail.

Enfin, nous voudrons remercier nos familles et nos proches pour leurs soutiens sans failles qu'ils nous ont apportés depuis toujours. Nous remercions également tous nos amis et tous ceux qui ont été impliqués d'une manière ou d'une autre dans la réussite de ce travail.

Table des matières

Table des figures

Liste des tableaux

Introduction

> La réussite de la création de l'IA serait le plus grand événement de l'histoire de l'humanité. Malheureusement, ce pourrait aussi être le dernier, à moins que nous apprenions à éviter les risques.
>
> Stephen Hawking

1 Motivation

L'étude de l'intelligence artificielle a bénéficié d'une grande attention au cours des dernières décennies, et cette technologie a été appliquée dans de nombreux domaines. Dans l'ingénierie des systèmes de contrôle, les techniques d'intelligence artificielle ont été utilisées pour mettre en œuvre des contrôleurs physiques ou des systèmes de supervision en utilisant des systèmes experts en temps réel et des réseaux de neurones, d'autres applications comprennent l'identification des systèmes et la conception des contrôleurs.

Une définition populaire de l'intelligence artificielle, telle qu'exprimée par Neill Graham, est :

> « L'intelligence artificielle est la branche de l'informatique consacrée à la programmation des ordinateurs pour effectuer des tâches qui, si elles étaient effectuées par des êtres humains, nécessiteraient une intelligence. »

Une autre définition est proposée par Elaine Rich :

> « L'intelligence artificielle est l'étude de la façon dont les ordinateurs font des choses auxquelles, pour le moment, les humains sont meilleurs. »

La conception d'un système de contrôle nécessite certainement une intelligence humaine, c'est une combinaison de créativité et de prise de décision intelligente. Cependant, l'objectif de tout système de contrôle intelligent est de produire un régulateur automatique, et pas nécessairement de remplacer les humains. Ils doivent plutôt être considérés comme une unité ou une équipe, où chaque membre apporte des compétences spécifiques tout en travaillant à un objectif commun. Les humains, par exemple, ont de puissantes capacités d'abstraction et de reconnaissance des formes, tandis que les ordinateurs peuvent effectuer des calculs complexes avec rapidité, précision et fiabilité. La combinaison de l'homme et de la machine pourrait contribuer grandement à éliminer les faiblesses de l'un ou de l'autre. Trouver cette combinaison devrait être l'un des objectifs de la recherche sur l'intelligence artificielle.

2 Problématique

Bien que plusieurs techniques classiques de contrôle aient été introduites, elles ne sont pas assez robustes pour de nombreux problèmes du monde réel où le degré d'incertitude est élevé, et donc ces méthodes de modélisation mathématique et de contrôle ne sont plus efficaces.

Les changements d'environnement, les perturbations non mesurables, les variations des modèles de référence et les défaillances des composants sont quelques-unes des caractéristiques qui nécessitent un contrôle intelligent. Les développements dans le domaine de l'intelligence artificielle ont atteint un stade qui contribuera à réduire ces complexités de contrôle en incorporant l'intelligence dans les systèmes de contrôle.

Le défi pour tout système de contrôle intelligent se situe en deux étapes. Premièrement, il est souvent difficile de combiner les relevés de divers capteurs pour obtenir une image cohérente et précise du système (un processus appelé « fusion de capteurs »). Deuxièmement, l'évaluation des actions nécessite généralement de prévoir leurs conséquences afin de déterminer leur adéquation à la situation actuelle. Les systèmes de contrôle intelligents présentent deux caractéristiques uniques qui les différencient des systèmes de contrôle conventionnels, à savoir la « capacité de prise de décision » et la « capacité

d'apprentissage ».

La problématique de cette mémoire est de présenter certaines des applications de l'intelligence artificielle pour les contrôleurs automatiques, avec une brève explication des algorithmes utilisés.

Un autre point que nous adressons est le développement d'un algorithme d'intelligence artificielle pour l'identification des systèmes afin de concevoir un régulateur approprié.

3 Plan de mémoire

Lors du chapitre 1, nous expliquerons certains éléments de l'intelligence artificielle, et nous verrons quelques techniques les plus souvent utilisées.

Dans le chapitre 2, nous exposerons quelques types des régulateurs les plus connus, à savoir le régulateur Proportionnel-Intégral-Dérivé, le régulateur par synthèse H_∞ et le régulateur adaptatif.

Pendant le chapitre 3, nous présenterons quelques travaux et publications sur l'application des algorithmes d'intelligence artificielle pour la régulation automatique. Nous expliquons les différents éléments consécutifs de chaque méthode et les résultats obtenus.

Finalement, au fil du chapitre 4, nous développerons un algorithme pour l'identification des fonctions de transfert, le système obtenu nous permet de construire un régulateur approprié.

Nous terminerons par une conclusion de ce que nous avons fait, les potentiels de l'intelligence artificielle pour les systèmes de contrôle intelligent et les difficultés de ces méthodes.

Chapitre 1

Intelligence artificielle

Les technologies modernes comme l'intelligence artificielle, l'apprentissage machine, l'apprentissage profond et la science des données sont devenues les mots les plus courants dont tout le monde parle mais que peu de personnes comprennent. Elles semblent très complexes pour un non-spécialiste. Et ils ressemblent à ceux qui n'ont pas de connaissances techniques. Dans ce chapitre, nous expliquons ces technologies avec des mots simples afin que la différence entre elles soit claire.

1.1 Intelligence artificielle

L'intelligence artificielle (IA) est une branche de l'informatique qui s'efforce de reproduire ou de simuler l'intelligence humaine dans une machine, afin que les machines puissent effectuer des tâches qui nécessitent généralement une intelligence humaine. L'existence de l'IA remonte aux années cinquante, depuis que la question « les machines peuvent-elles penser ? » a été posée par Alan Turing.

L'intelligence artificielle désigne la simulation de fonctionnement du cerveau humain par des machines. Cette simulation permet à ces machines de comprendre des données, d'apprendre à partir de ces données et de prendre des décisions basées sur des relations entre ces données, ou elle permet de faire des déductions qui pourraient autrement être très difficiles (voire presque impossibles) à faire manuellement pour les humains.

Les principales fonctions humaines que les machines d'intelligence artificielle exécutent comprennent le raisonnement logique, l'apprentissage, la planification,

l'autocorrection, la résolution de problèmes et la prise de décision. Cela se fait en créant un réseau neuronal artificiel qui peut montrer l'intelligence en ajustant leurs « connaissances » en fonction de nouvelles entrées qui ne faisaient pas partie des données utilisées pour entraîner ces machines.

Cette technologie est devenue très populaire récemment grâce aux progrès réalisés dans les capacités de traitement. Aujourd'hui, nous disposons des ordinateurs parmi les plus puissants que le monde ait jamais connus. Et les implémentations des algorithmes de l'IA se sont tellement améliorées que nous pouvons les faire fonctionner sur du matériel ordinaire.

L'intelligence artificielle est un vaste domaine aux applications multiples, mais c'est aussi l'une des technologies les plus complexes à mettre en œuvre. Les machines ne sont pas intelligentes par nature et pour les rendre ainsi, nous avons besoin de beaucoup de puissance de calcul et de données pour leur permettre de simuler la pensée humaine.

Comme la recherche sur l'intelligence artificielle vise à faire en sorte que les machines émulent un fonctionnement similaire à celui de l'homme, le degré auquel un système d'IA peut reproduire les capacités humaines est utilisé comme critère pour déterminer les types d'IA. Ainsi, en fonction de leur polyvalence et de performance, le matériel qu'ils utilisent et leurs applications dans le monde réel.

Sur la base de ce critère, l'intelligence artificielle peut être classée dans l'un des quatre types suivants : machines réactives, mémoire limitée, théorie de l'esprit et la conscience de soi. Les deux derniers types existent seulement comme concept sans des vraies applications dans la réalité. De coup, nous intéressons aux deux premiers types :

1.1.1 Machines réactives

Ce sont les plus anciennes formes des systèmes d'intelligence artificielle qui ont une capacité très limitée. Ils imitent la manière dont l'esprit humain réagit à différents types de stimuli. Ces machines n'ont pas de fonctionnalités basées sur la mémoire. Cela signifie qu'elles ne peuvent pas utiliser les expériences acquises précédemment pour ajuster leurs actions actuelles. Ces machines n'ont pas la capacité d'apprendre, et ne peuvent être utilisées que pour répondre

automatiquement à un ensemble ou une combinaison limitée d'entrées.

Un exemple populaire de machine IA réactive est le Deep Blue d'IBM, une machine qui a battu le grand maître d'échecs Garry Kasparov en 1997 .

1.1.2 Mémoire limitée

Les machines à mémoire limitée sont des machines qui, en plus d'avoir les capacités des machines purement réactives, sont également capables d'apprendre à partir de données historiques pour prendre des décisions.

Presque toutes les applications d'intelligence artificielle existantes que nous connaissons entrent dans cette catégorie. Les systèmes d'IA actuels, tels que ceux utilisant l'apprentissage profond, sont formés par de grands volumes de données d'apprentissage qu'ils stockent dans leur mémoire pour entraîner un modèle de référence pour la résolution des problèmes futurs.

Par exemple, une IA de reconnaissance d'images est formée en utilisant des milliers d'images et leurs étiquettes pour lui apprendre à nommer les objets qu'elle scanne. Lorsqu'une image est scannée par une telle IA, elle utilise les images d'entraînement comme références pour comprendre le contenu de l'image qui lui est présentée, et sur la base de son « expérience d'apprentissage », elle étiquette les nouvelles images avec une précision croissante.

1.2 Machine Learning

Machine Learning (ML) ou l'apprentissage machine est un sous-ensemble de l'intelligence artificielle qui utilise des algorithmes d'apprentissage statistique. Il signifie la capacité d'un système informatique à apprendre de l'environnement et à s'améliorer automatiquement à partir de l'expérience sans avoir besoin d'une programmation explicite. L'apprentissage machine vise à permettre aux algorithmes d'apprendre à partir des données fournies, de rassembler des informations et de faire des prédictions sur des données non analysées auparavant en utilisant les informations apprises.

Il existe divers algorithmes en ML qui pourraient être utilisés pour des problèmes de prédiction, de classification, de régression, etc. Il existe de nombreux algorithmes tels que la régression linéaire simple, la régression polynomiale,

Support Vector Machine (SVM), Classification And Regression Trees (CART) et k-Nearest Neighbours (k-NN).

Les algorithmes de l'apprentissage machine peuvent être classés en trois grandes catégories :

1.2.1 Apprentissage supervisé

Dans l'apprentissage supervisé, nous avons des variables d'entrée et une variable de sortie, et nous utilisons un algorithme pour apprendre la correspondance entre l'entrée et la sortie. En d'autres termes, un algorithme d'apprentissage supervisé prend un ensemble connu de données d'entrée et ses réponses connues pour apprendre le modèle de régression/classification. Un algorithme d'apprentissage entraîne ensuite un modèle pour générer une prédiction de la réponse à de nouvelles données ou aux ensembles de données de test.

1.2.2 Apprentissage non supervisé

L'apprentissage non supervisé est utilisé lorsque nous n'avons pas de données étiquetées. Son objectif principal est d'en apprendre davantage sur les données en déduisant des caractéristiques dans l'ensemble de données sans référence aux résultats connus. Il est appelé non supervisé parce que les algorithmes sont laissés à eux-mêmes pour regrouper les informations non triées en trouvant des similitudes, des différences et des caractéristiques des données. L'apprentissage non supervisé est principalement effectué dans le cadre de l'analyse exploratoire des données. Il est le plus souvent utilisé pour trouver des groupes de données et pour la réduction de la dimensionnalité.

1.2.3 Apprentissage par renforcement

En termes simples, l'apprentissage par renforcement peut être expliqué comme un apprentissage par interaction continue avec l'environnement. Il s'agit d'un type d'algorithmes d'apprentissage automatique dans lequel un agent apprend d'un environnement par des interactions essaies-erreurs en utilisant en permanence le retour d'informations de ses actions et ses expériences précédentes. L'agent reçoit des récompenses pour ses actions dans l'environnement.

1.3 Deep Learning

Deep Learning ou l'apprentissage profond peut être considéré comme une technique d'apprentissage machine. Il s'inspire de la façon dont le cerveau humain filtre l'information, c'est-à-dire qu'il apprend essentiellement à partir d'exemples. Elle permet à un modèle informatique de filtrer les données d'entrée à travers des couches pour prédire et classer l'information. Comme l'apprentissage profond traite l'information de la même manière que le cerveau humain, il est principalement utilisé pour effectuer des tâches qui nécessitent généralement des humains. C'est la technologie clé des voitures sans conducteur.

Il est également important de noter que l'apprentissage profond nécessite un matériel très puissant pour l'entraînement, qu'il prend beaucoup plus de temps pour l'entraînement des modèles, et qu'il est généralement plus difficile à mettre en œuvre par rapport à l'apprentissage machine.

La plupart des méthodes d'apprentissage profond utilisent des architectures de réseaux de neurones multiples contenant un grand nombre de paramètres et de couches, c'est pourquoi ils sont souvent appelés réseaux de neurones profonds. Les trois architectures de réseau fondamentales sont indiquées ci-dessous :

1.3.1 Réseau de neurones convolutif

Le réseau de neurones convolutif est essentiellement un réseau de neurones artificiel qui est le plus utilisé dans le domaine de Computer Vision pour l'analyse et la classification des images. Il s'agit d'un algorithme d'apprentissage profond qui prend l'image d'entrée et attribue des poids et des biais à divers aspects ou objets de l'image, afin de pouvoir les différencier les uns des autres. Les couches cachées de ce réseau sont généralement constituées de couches convolutives, de couches de regroupement, de couches entièrement connectées et de couches de normalisation.

1.3.2 Réseau de neurones récurrent

Les réseaux de neurones récurrents sont un type d'architecture qui est utilisé dans les problèmes de prédiction des séquences et qui est très utilisé dans le domaine du traitement du langage naturel. Ces réseaux sont appelés récurrents parce qu'ils

effectuent la même tâche pour chaque élément de la séquence, et dont la sortie dépend des calculs précédents. Ils ont comme une « mémoire » qui capture les informations sur ce qui a été calculé jusqu'à présent.

1.3.3 Réseau de neurones récursif

Un réseau de neurones récursif ressemble davantage à un réseau hiérarchique dans lequel la séquence d'entrée n'a pas vraiment d'aspect temporel, mais l'entrée doit être traitée de manière hiérarchique, sous forme d'arbre.

1.4 La science des données

La science des données est l'extraction d'informations pertinentes à partir de données. Elle utilise diverses techniques issues de nombreux domaines comme les mathématiques, l'apprentissage machine, la programmation informatique, la modélisation statistique, l'ingénierie, la reconnaissance des formes, la modélisation de l'incertitude et l'entreposage des données.

1.5 Logique floue

La logique floue est un outil d'intelligence artificielle, c'est une extension de la logique booléenne créée par Lotfi Zadeh en 1965 sur la base de sa théorie mathématique des ensembles flous. En introduisant la notion de degré dans la vérification d'une condition, permettant ainsi à une condition d'être dans un autre état que vrai ou faux, la logique floue confère une souplesse très appréciable au raisonnement qui l'utilise, permettant de prendre en compte les imprécisions et les incertitudes.

Un des intérêts de la logique floue pour formaliser le raisonnement humain est que les règles sont énoncées en langage naturel sous la forme $Si\ p\ alors\ q$. Les variables appartiennent à des sous-ensembles flous avec un degré d'appartenance. La prémisse p exprime des relations entre les variables par des opérateurs flous, et elle vraie avec un degré de validé, autrement dit du degré d'appartenance des variables aux ensembles flous. L'idée sous-jacente est que plus les propositions en prémisse sont vérifiées, plus l'action préconisée pour les sorties doit être respectée.

1.6 Différence entre IA, ML et la science des données

L'intelligence artificielle est un terme très large dont les applications vont de la robotique à l'analyse de texte. L'apprentissage machine est un sous-ensemble de l'IA qui se concentre sur une gamme étroite d'activités. Il s'agit en fait de la seule véritable intelligence artificielle ayant quelques applications dans des problèmes du monde réel.

La science des données n'est pas exactement un sous-ensemble de l'apprentissage machine mais elle utilise le Machine Learning pour analyser des données et faire des prédictions sur l'avenir. Elle combine l'apprentissage machine avec d'autres disciplines comme l'analyse de Big Data et le Cloud Computing. La science des données est une application pratique de l'apprentissage machine qui se concentre entièrement sur la résolution de problèmes du monde réel.

1.7 Différence entre l'apprentissage profond et RL

L'apprentissage profond et l'apprentissage par renforcement sont deux systèmes qui apprennent de manière autonome. La différence entre eux est que l'apprentissage profond est l'apprentissage à partir d'un ensemble d'entraînement et ensuite l'application de cet apprentissage à un nouvel ensemble de données, tandis que l'apprentissage par renforcement est un apprentissage dynamique par l'ajustement des actions basé sur un retour d'information continu afin de maximiser une récompense.

L'apprentissage approfondi et l'apprentissage par renforcement ne s'excluent pas mutuellement. En fait, vous pouvez utiliser l'apprentissage profond dans un système d'apprentissage par renforcement, qui est appelé apprentissage profond par renforcement.

La figure suivante montre l'évolution de l'utilisation des différents types de l'intelligence artificielle :

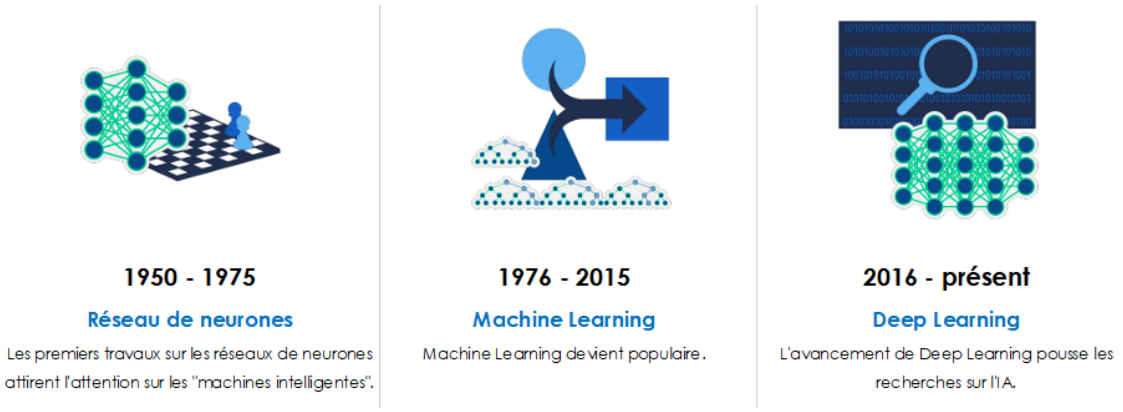

FIGURE 1.1: L'évolution de l'utilisation de l'intelligence artificielle

1.8 Conclusion

L'intelligence artificielle est probablement la création la plus complexe et la plus étonnante de l'humanité à ce jour. Et cela sans tenir compte du fait que le domaine reste largement inexploré, ce qui signifie que chaque application d'IA étonnante que nous voyons aujourd'hui ne représente que la partie émergée de l'iceberg de l'IA. Bien que ce fait ait été affirmé et répété à plusieurs reprises, il est encore difficile d'avoir une vision globale de l'impact potentiel de l'IA à l'avenir. La raison est l'impact révolutionnaire que l'IA a sur la société, même à ce stade relativement récent de son évolution.

Chapitre 2

Les types des régulateurs

Le régulateur en boucle est un élément d'un système asservi qui lui permet d'atteindre un état associé à une valeur de consigne et de s'y maintenir. Ce mécanisme de régulation cherche à réduire l'écart entre cette valeur de consigne et une grandeur physique mesurée par un ou plusieurs capteurs dont le système est équipé. Dans ce chapitre nous allons présenter quelques types et formes des régulateurs les plus connus.

2.1 Régulateur Proportionnel-Intégral-Dérivé

Le régulateur Proportionnel-Intégral-Dérivé, appelé aussi correcteur PID est un système de contrôle permettant d'améliorer les performances d'un asservissement, c'est-à-dire un système ou procédé en boucle fermée. C'est le régulateur le plus utilisé dans l'industrie où ses qualités de correction s'appliquent à de multiples grandeurs physiques.

2.1.1 Actions élémentaires

Le PID délivre un signal de commande à partir de la différence entre la consigne et la mesure, Il utilise les trois actions suivantes :

- Action proportionnelle : l'erreur est multipliée par un gain K_p .
- Action intégrale : l'erreur est intégrée et multipliée par un gain K_i .
- Action dérivée : l'erreur est dérivée et multipliée par un gain K_d .

2.1.2 Types de régulateurs PID

i. Proportionnel (P) :

Dans le cas d'un contrôle proportionnel, l'erreur est virtuellement amplifiée d'un certain gain constant qu'il conviendra de déterminer en fonction du système.

$$u\left(p\right) = K_p.\varepsilon\left(p\right)$$

L'idée étant d'augmenter l'effet de l'erreur sur le système afin que celui-ci réagisse plus rapidement aux changements de consignes. Plus la valeur de K_p est grande, plus la réponse l'est aussi. En revanche, la stabilité du système s'en trouve détériorée et dans le cas d'un K_p démesuré le système peut même diverger.

ii. Proportionnel-Intégral (PI) :

Au contrôle proportionnel, nous pouvons ajouter l'intégration de l'erreur. Dans ce cas nous obtenons une régulation Proportionnel-Intégral.

L'erreur entre la consigne et la mesure est ici intégrée par rapport au temps et multipliée par une constante qu'il faudra aussi déterminer en fonction du système.

$$u\left(p\right) = K_p.\varepsilon\left(p\right) + K_i\frac{\varepsilon\left(p\right)}{p}$$

Lors d'un simple contrôle proportionnel, il subsiste une erreur statique. Le terme intégral permet ainsi de compenser l'erreur statique et fournit, par conséquent, un système plus stable en régime permanent. Plus K_i est élevé, plus l'erreur statique est corrigée.

iii. Proportionnel-Intégral-Dérivé (PID) :

Pour obtenir un contrôle en PID, il nous faut encore rajouter un terme. Celui-ci consiste à dériver l'erreur entre la consigne et la mesure par rapport au temps et à le multiplier lui aussi par une constante.

$$u\left(p\right) = \left(K_p + K_i\frac{1}{p} + K_dp\right)\varepsilon\left(p\right)$$

Cette forme s'appelle la structure parallèle. Il existe une structure série :

$$u(p) = K_p \left(1 + K_i \frac{1}{p}\right)(1 + K_d p)\,\varepsilon(p)$$

Nous pouvons aussi trouver la structure mixte :

$$u(p) = K_p \left(1 + K_i \frac{1}{p} + K_d p\right)\varepsilon(p)$$

Le contrôle PI peut amener à un dépassement de la consigne, ce qui n'est pas toujours très souhaitable. Le terme dérivé permet de limiter cela. Lorsque le système s'approche de la consigne, ce terme freine le système en appliquant une action dans le sens opposé et permet ainsi une stabilisation plus rapide.

2.1.3 Les qualités d'une régulation

Le réglage d'un PID consiste à déterminer les coefficients afin d'obtenir une réponse adéquate du procédé et de la régulation. Les objectifs sont :

• La robustesse est sans doute le paramètre le plus important et délicat. Un système est dit robuste si la régulation fonctionne toujours même si le modèle change un peu. Par exemple, les fonctions de transfert de certains procédés peuvent varier en fonction de la température ambiante ou de l'hygrométrie ambiante relativement à la loi de Pascal. Un régulateur doit être capable d'assurer sa tâche même avec des perturbations afin de s'adapter à des usages non prévus/testés (dérive de production, vieillissement mécanique, environnements extrêmes ...)

• La rapidité du régulateur dépend du temps de montée et du temps de réponse du régime stationnaire

• Le critère de précision est basé sur l'erreur statique.

2.1.4 Réglage du PID

L'analyse du système avec un PID est simple mais sa conception peut être délicate, voire difficile, car il n'existe pas de méthode unique pour résoudre ce problème. Il faut trouver des compromis, le régulateur idéal n'existe pas. En général, nous nous fixons un cahier des charges à respecter sur la robustesse, le

dépassement et le temps de réponse du régime stationnaire. Les méthodes de réglage les plus utilisées en théorie sont les méthodes de Ziegler-Nichols (en boucle ouverte et boucle fermée), la méthode de Naslin (polynômes normaux à amortissement réglable) et la méthode du lieu de Nyquist inverse (utilise le diagramme de Nyquist).

De manière plus pratique et rapide les professionnels utilisent soit l'identification par modèle de Broïda pour les systèmes stables ou le modèle intégrateur retardé pour les systèmes instables, soit la méthode par approches successives qui répond à une procédure rigoureuse : nous fixons d'abord l'action P seule pour avoir un dépassement de 10% à 15 % puis l'action dérivée de façon à « raboter » au mieux le dépassement précédent, enfin, nous ajustons si nécessaire l'action intégrale en se fixant un dépassement final compris entre 5% et 10% . Il existe aussi une méthode qui, en supposant connue la fonction de transfert du système, permet de déterminer un régulateur PID robuste dans le sens où la marge de phase et la pulsation au gain unité (donc la marge de phase/retard) sont fixées à l'avance (lorsqu'une solution existe).

2.2 Régulateur H infini

La synthèse H_∞ est une méthode qui sert à la conception des commandes robustes. Il s'agit essentiellement d'une méthode d'optimisation, qui prend en compte une définition mathématique des contraintes en ce qui concerne le comportement attendu en boucle fermée. La commande H_∞ a pour principal avantage la capacité d'inclure dans un même effort de synthétisation les concepts liés à la commande classique et à la commande robuste.

2.2.1 H infini standard

La synthèse H_∞ utilise la notion de problème standard, qui est représenté sur la figure 2.1, la matrice de transfert $P(s)$ modélise les interactions dynamiques entre deux ensembles d'entrées et deux ensembles de sorties : le vecteur w représente des entrées extérieures, telles que les signaux de références, perturbations, bruits. Le vecteur u représente les commandes. Les signaux e sont choisis pour caractériser le bon fonctionnement de l'asservissement. Le vecteur y représente les mesures

disponibles pour élaborer la commande. Enfin $K(s)$ est le correcteur H_∞ .

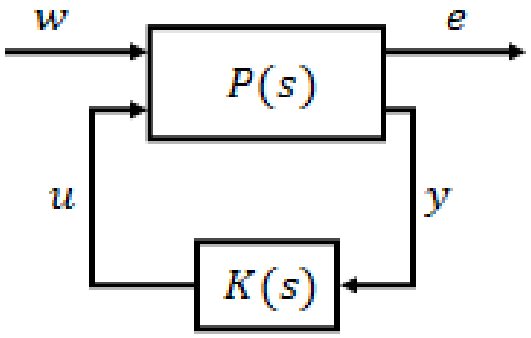

FIGURE 2.1: Problème H_∞ standard

Nous pouvons effectuer une partition de la matrice $P(s)$ de façon cohérente avec les dimensions de $w,\ u,\ e,\ y$:

$$P(s) = \begin{bmatrix} P_{ew}(s) & P_{eu}(s) \\ P_{yw}(s) & P_{yu}(s) \end{bmatrix}$$

Le problème H_∞ standard : $P(s)$ et $\gamma > 0$ étant donnés, nous cherchons le correcteur $K(s)$ qui stabilise la boucle de la figure 2.1, et assure :

$$\left\| P_{ew}(s) + P_{eu}(s) K(s) (I - P_{yu}(s) K(s))^{-1} P_{yw}(s) \right\|_\infty < \gamma$$

Différentes méthodes peuvent être envisagées pour résoudre le problème H_∞ standard. Les méthodes les plus utilisées sont l'approche par équation de Riccati dans laquelle la valeur optimale de γ est recherchée par dichotomie, et l'approche par Inégalités Matricielles Affines (Linear Matrix Inequalities LMI).

2.3 Régulateur adaptatif

La régulation adaptative est une méthode de régulation utiliser pour les systèmes avec des paramètres variables ou incertains, elle signifie l'adaptation de la commande afin qu'elle soit conforme aux changements et circonstances nouvelles.

Le contrôle adaptatif est différent du contrôle robuste en ce sens qu'il ne nécessite pas d'informations a priori sur les limites de ces paramètres variables dans le temps ou incertains. Le contrôle robuste garantit que si les changements se situent dans

des limites données, la loi de contrôle n'a pas besoin d'être modifiée, tandis que le contrôle adaptatif concerne la loi de contrôle qui se modifie elle-même.

La commande adaptative est composée d'un estimateur de paramètres en ligne avec une loi de commande pour commander des systèmes dont la classe et les paramètres sont généralement inconnus. Plusieurs types de commande peuvent être obtenus en combinant différents choix d'estimateur et de loi de contrôle.

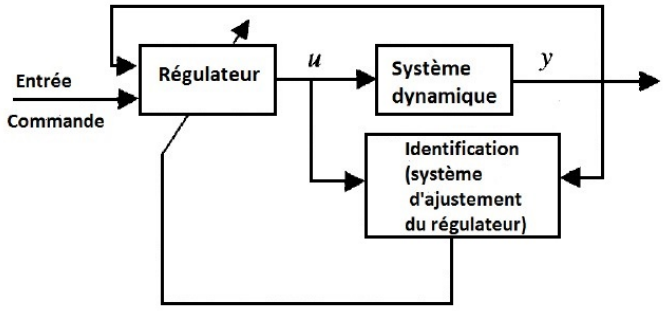

FIGURE 2.2: Structure d'une commande adaptative d'un système dynamique

Les approches de commande adaptative peuvent être classées en deux classes :

2.3.1 Commande adaptative indirecte

La première approche est la commande adaptative indirecte, où les paramètres du système à commander sont estimés en ligne et utilisés pour calculer les paramètres du régulateur. À chaque instant t (période d'échantillonnage), le système estimé est formé et traité comme s'il était le véritable système à commander (à partir duquel nous faisons le calcul des paramètres du régulateur). Cette approche est appelée aussi commande adaptative explicite.

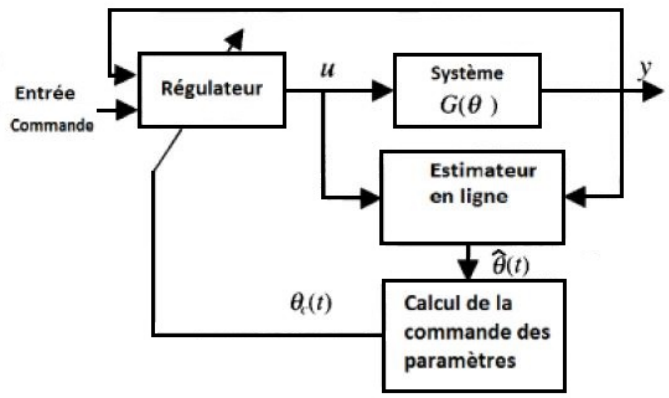

FIGURE 2.3: Structure d'une commande adaptative indirecte

2.3.2 Commande adaptative directe

La deuxième approche est la commande adaptative directe, où le système d'identification est paramétré en fonction des paramètres du régulateur souhaité, qui sont estimés directement (sans calculs intermédiaires impliquant des estimations de paramètres du système dynamique). Cette approche est appelée aussi commande adaptative implicite.

La structure de base du contrôle adaptatif indirect est représentée dans la figure suivante. Le modèle de système $G(\theta^*)$ est paramétré par rapport à un vecteur de paramètre inconnu θ^* .

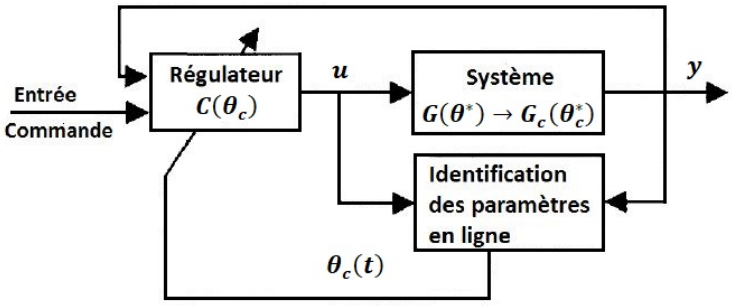

FIGURE 2.4: Structure d'une commande adaptative directe

2.4 Conclusion

Dans ce chapitre, nous avons présenté quelques types de régulateur, notamment le régulateur PID qui est le plus connu et utilisé, le régulateur robuste H_∞ et le régulateur adaptatif. Il existe d'autres formes moins souvent utilisées ou adaptées à un type spécifique des problèmes.

Chapitre 3

Des applications de l'IA pour la régulation

L'intelligence artificielle s'est montrée un outil important pour les développeurs dans de nombreux domaines allant de la médecine aux applications industrielles. L'une des utilisations prometteuses étant l'ingénierie de contrôle pour la conception et le réglage des contrôleurs. Dans ce chapitre, nous présenterons certains des travaux publiés qui utilisent des algorithmes d'intelligence artificielle pour la régulation automatique des systèmes, en expliquant l'implémentation de ces algorithmes et les résultats obtenus.

3.1 Régulateur PID avec apprentissage par renforcement

En 2013, des chercheurs indonésiens ont publié un article intitulé « Application of Reinforcement Learning on Self-Tuning PID Controller for Soccer Robot Multi-Agent System »[9]. Leur travail consiste à trouver des paramètres du régulateur PID par un algorithme d'apprentissage par renforcement pour une application d'un robot de football, en comparant les résultats avec d'autres méthodes classiques.

Le domaine du robot de football est un environnement de recherche interdisciplinaire qui combine la perception, la prise de décision dynamique et la communication entre le système du robot et le système de contrôle du mouvement. Ce dernier est l'une des parties les plus importantes qui vise à réguler le mouvement du robot de manière rapide et précise. Lors de cette publication, les

développeurs ont utilisé une combinaison de deux types différents de robot dans une équipe de cinq robots. Chaque robot peut se déplacer linéairement vers une certaine position avec un certain angle. Le but est de trouver deux régulateurs pour chaque robot, $C_a(p)$ pour contrôler le déplacement linéaire et $C_b(p)$ pour la rotation.

Le régulateur PID a été largement utilisé pour obtenir des réponses optimales, avec trois paramètres qui doivent être réglés, à savoir K_p, K_i et K_d . Comme les robots ont des spécifications déférentes, l'utilisation des méthodes conventionnelles d'ajustement du PID sera difficile. C'est pourquoi l'algorithme d'apprentissage par renforcement est utilisé.

L'algorithme utilisé dans cette publication est le Q-Learning, qui utilise un tableau-Q pour son apprentissage. Où un agent prenait des actions dans l'environnement en fonction son état afin de maximiser une récompense.

L'état de l'environnement est la distance *de* entre la position actuelle du robot et sa position finale, la figure suivante présente les trois parties de l'états. La récompense dépend de cet état et du temps nécessaire pour atteindre l'état final. L'action consiste à sélectionner des paramètres du PID parmi un ensemble des valeurs prédéfinies.

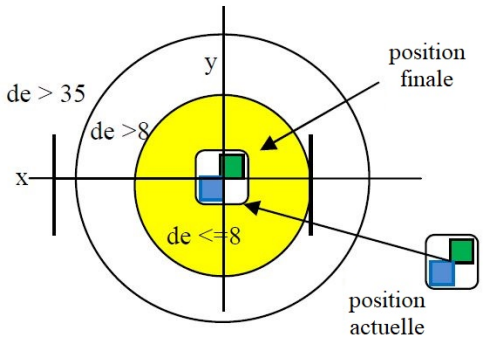

FIGURE 3.1: Les trois parties de l'état

Le tableau-Q qui met en relation l'état actuel s et l'action a, est actualisé par la fonction suivante :

$$Q^{new}(s,a) = Q(s,a) + \alpha\left[R_{t+1} + \gamma \max_a Q(s',a') \; - Q(s,a)\right]$$

Avec s' et a' sont l'état et l'action suivants, α et γ sont des paramètres

d'apprentissage et les valeurs $Q(s, a)$ sont le contenu du tableau-Q.

- Les résultats de la publication sont :

- Le système de contrôle PID développé dans cette étude peut effectuer des réglages automatiques pour obtenir les valeurs optimales des paramètres.

- L'apprentissage sur une période relativement longue permet d'obtenir des paramètres optimaux pour les régulateurs $C_a(p)$ et $C_b(p)$, donc si les robots continuent à fonctionner pendant le processus d'apprentissage, les paramètres sont mieux optimisés.

- Le contrôle par apprentissage par renforcement donne des paramètres de contrôle plus stables et plus rapides que le réglage Ziegler-Nichols.

3.2 Un contrôleur par apprentissage par renforcement

Des chercheurs de l'université des sciences appliquées en Allemagne ont publié en 2010 un article intitulé « Controller Design for Quadrotor UAVs using Reinforcement Learning »[2]. Leur travail consiste à utiliser un algorithme d'apprentissage par renforcement pour contrôler un drone.

Le quad-rotor est un système à quatre hélices en configuration croisée. Alors que les moteurs avant et arrière tournent dans le sens des aiguilles d'une montre, les moteurs gauche et droite tournent dans le sens inverse, ce qui élimine pratiquement les effets gyroscopiques et les couples aérodynamiques en vol compensé. En variant la vitesse des différents moteurs, un mouvement vertical et/ou latéral peut être généré. Cependant, malgré les quatre actionneurs, le quad-rotor est un système non linéaire dynamiquement instable qui doit être stabilisé par un système de contrôle approprié. Les variables d'état de ce système $x^T = \left(\dot{x},\ \dot{y},\ \dot{z},\ \phi,\ \theta,\ \psi,\ \dot{\phi},\ \dot{\theta},\ \dot{\psi}\right)$.

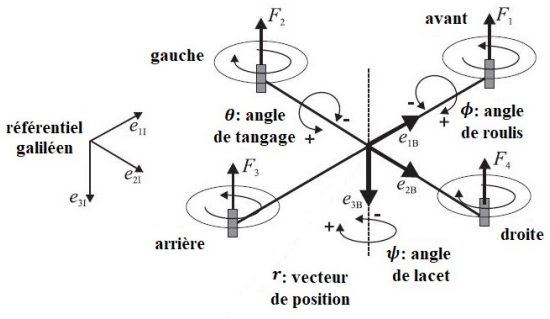

FIGURE 3.2: Représentation du quad-rotor

L'algorithme d'apprentissage par renforcement utilisé lors de cette publication est nommé Fitted Value Iteration (FVI). L'État est défini par $S^T = [\phi,\ \theta,\ \psi]$ et l'état de référence définissant la position de vol stationnaire $S^T_{ref} = [\phi_{ref},\ \theta_{ref},\ \psi_{ref}]$.

La fonction de récompense est choisie comme :

$$R\left(S, S_{ref}\right) = -c_1(\phi - \phi_{ref})^2 - c_2(\theta - \theta_{ref})^2 - c_3(\psi - \psi_{ref})^2$$

Avec $c_1 > 0$, $c_2 > 0$ et $c_3 > 0$ sont des constantes qui donnent la possibilité de se concentrer particulièrement sur le contrôle d'un des angles d'Euler plutôt que sur les autres.

Une approximation paramétrique de la fonction de valeur est choisie en incorporant des termes quadratiques et de couplage :

$$V\left(s\right) = \left(a_1,\ a_2,\ a_3,\ a_4,\ a_5\right) \cdot \begin{pmatrix} \phi^2 \\ \theta^2 \\ \psi^2 \\ \phi\psi \\ \theta\psi \end{pmatrix}$$

Ces variables ont été choisies en raison de leur importance telle qu'obtenue à partir des considérations théoriques. Après une sélection aléatoire de 1000 états du système en utilisant le modèle non linéaire du quad-rotor, l'objectif principal de l'algorithme est d'adapter les constantes $(a_1,\ \ldots,\ a_5)$ afin d'obtenir la meilleure approximation de la fonction de valeur. Pour obtenir ces dernières constantes, les équations normales ont été résolues en utilisant la technique de factorisation (QR) pour calculer le pseudo inverse de la matrice de conception A qui avait été choisie pour satisfaire les contraintes.

$$A = \begin{bmatrix} \phi\left(i\right)^2 & \theta\left(i\right)^2 & \psi\left(i\right)^2 & \phi\left(i\right)\psi\left(i\right) & \theta\left(i\right)\psi\left(i\right) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi\left(m\right)^2 & \theta\left(m\right)^2 & \psi\left(m\right)^2 & \phi\left(m\right)\psi\left(m\right) & \theta\left(m\right)\psi\left(m\right) \end{bmatrix}$$

Où i et m représentent les nombres d'exemplaires d'entraînement qui varient de 1 au nombre d'échantillons. Ensuite, les valeurs des constantes peuvent être obtenues

en utilisant la factorisation QR.

Les constantes $(a_1,\ \ldots,\ a_5)$ sont actualisées en utilisant l'erreur de moindre carré entre $V(s)$ et $\max_a [R(S_i) + \gamma V(S')]$.

Le résultat de contrôle obtenu est indiqué dans la figure suivante sous la forme d'un tracé temporel de tous les angles du quad-rotor, avec un état initial $S_0^T = [\phi_0 = 30°,\ \theta_0 = -20°,\ \psi_0 = 10°]$, où le but du contrôle est de stabiliser une position de vol stationnaire $S_{ref}^T = 0$.

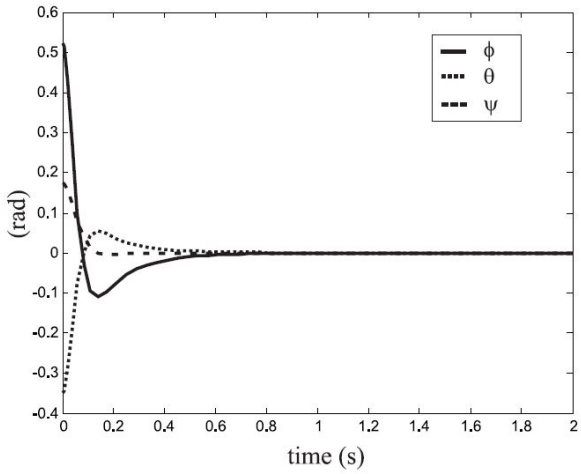

FIGURE 3.3: Résultat de simulation du quad-rotor

Il y a une phase de transition très courte avec un petit dépassement, et l'état de vol stationnaire est atteint après $t \approx 0.6\ sec$.

- Les résultats de la publication sont :

L'avantage d'un algorithme d'apprentissage est le fait qu'aucune connaissance mathématique préalable du modèle n'est nécessaire pour concevoir un contrôleur. Cela montre qu'un modèle du quad-rotor peut être approximé à l'aide de différentes techniques en reliant simplement les données d'entrée et de sortie par une approche non paramétrique.

3.3 Différentes techniques pour le contrôle d'un moteur

En 2009, des professeurs de l'université de Newcastle en Royaume-Uni ont publié un article intitulé « Artificial intelligence-based speed control of DTC induction motor drives - A comparative study »[11]. Leur travail consiste à faire une comparaison entre quatre types de régulateur par intelligence artificielle, pour

le contrôle d'un moteur à induction par une commande directe du couple (Direct Torque Control DTC).

La représentation d'état du moteur à induction avec les courants du stator et du rotor comme variables d'état peut être écrite en coordonnées d-q :

$$\begin{bmatrix} p \cdot i_{sd} \\ p \cdot i_{sq} \\ p \cdot i_{rd} \\ p \cdot i_{rq} \end{bmatrix} = A \cdot \begin{bmatrix} i_{sd} \\ i_{sq} \\ i_{rd} \\ i_{rq} \end{bmatrix} + B \cdot \begin{bmatrix} v_{sd} \\ v_{sq} \end{bmatrix}$$

Le régulateur proportionnel-Intégral est largement utilisé dans les applications de contrôle des moteurs. Il a une structure simple et peut assurer des performances satisfaisantes. En raison de la variation continue des paramètres du système et des conditions de fonctionnement non linéaires, les contrôleurs PI à gain fixe peuvent ne plus fournir les performances de contrôle requises. Le réglage en ligne du contrôleur devient donc souhaitable lorsque des performances élevées sont exigées du système.

- Réglage du contrôleur PI par l'algorithme génétique (AG) :

L'algorithme génétique est une technique d'optimisation de recherche globale stochastique basée sur les mécanismes de la sélection naturelle. C'est une technique efficace pour résoudre les problèmes d'optimisation des systèmes non linéaires. Généralement l'algorithme génétique consiste de trois étapes :

- Étape de sélection : à ce stade, les individus de la population initiale sont sélectionnés pour la reproduction avec une probabilité proportionnelle à leur valeur d'évaluation (fitness value).

- Étape de croisement : Après la phase de sélection, l'opération de croisement génétique est ensuite appliquée entre les paires de parents pour générer de nouveaux individus ou des descendants qui obtiennent de bonnes caractéristiques de leurs parents.

- Étape de mutation : L'application d'une mutation génétique introduit un changement dans le chromosome selon un facteur de mutation.

L'application de ces trois étapes basiques permet la création de nouveaux individus, qui peuvent être meilleurs que leurs parents. Cet algorithme est répété

d'une génération à l'autre et s'arrête finalement lorsqu'il atteint les individus qui apportent une solution optimale au problème. Les paramètres de l'algorithme génétique choisi pour le réglage sont indiqués dans le tableau suivant :

Propriété	Valeur/méthode
Nombre de générations	10
Nombre de chromosomes de chaque génération	8
Nombre de gènes de chaque chromosome	2
Longueur des chromosomes	40 bit
Méthode de sélection	Stochastic Universal Selection (SUS)
Méthode de croisement	Double-point
Probabilité de croisement	0.7
Facteur de mutation	0.05

Tableau 3.1: Les paramètres de l'algorithme génétique

- Réglage du contrôleur PI par la logique floue :

Les méthodes de réglage basées sur la logique floue (PI Fuzzy Logic PI-FL) se sont avérées avantageuses pour traiter des systèmes imprécis, non linéaires ou variant dans le temps avec des paramètres et des variations de structure incertaines ou inconnues. Cela permet d'utiliser le PI-FL pour régler un contrôleur de vitesse dans un système du moteur DTC. En outre, un PI-FL est relativement facile à mettre en œuvre et ne nécessite pas de modèle mathématique du système contrôlé. Lorsque la logique floue est utilisée pour le réglage en ligne du régulateur de vitesse PI, elle reçoit les valeurs actualisées de l'erreur de vitesse et sa variation. La sortie de l'algorithme est la mise à jour des gains du contrôleur PI.

La mise à jour est basée sur des ensembles de règles floues de ΔK_p, ΔK_i et du régulateur PI pour maintenir d'excellentes performances de contrôle même en présence des variations des paramètres. Le schéma fonctionnel du système de contrôle est illustré dans la figure suivante. Chaque entrée du PI-FL possède cinq fonctions d'appartenance triangulaires de même largeur. La première sortie (ΔK_p) a trois fonctions d'appartenance triangulaires, tandis que la deuxième sortie (ΔK_i) a cinq fonctions d'appartenance. La table des règles d'inférence comporte 25 règles.

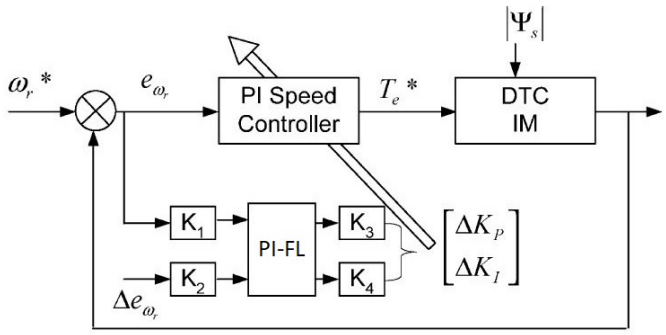

FIGURE 3.4: Régulateur de vitesse PI par logique floue

- Contrôleur de vitesse par logique floue :

Comme le PI-FL peut faire face aux non-linéarités et aux perturbations du moteur DTC, il a été utilisé pour remplacer entièrement le contrôleur PI traditionnel sous forme d'un contrôleur par logique floue (Fuzzy Logic Controller FLC).

Les entrées sont les valeurs normalisées de l'erreur de vitesse et le taux de variation à maintenir entre les limites de ± 1 de l'erreur de vitesse. Deux facteurs (k_e et k_d) sont utilisés pour normaliser les entrées. La sortie du contrôleur est la variation normalisée de la commande de couple moteur qui, lorsqu'elle est multipliée par un troisième facteur (k_u), elle génère la valeur réelle du taux de variation du couple moteur. Enfin, une intégration discrète est effectuée pour obtenir la commande du couple électromagnétique.

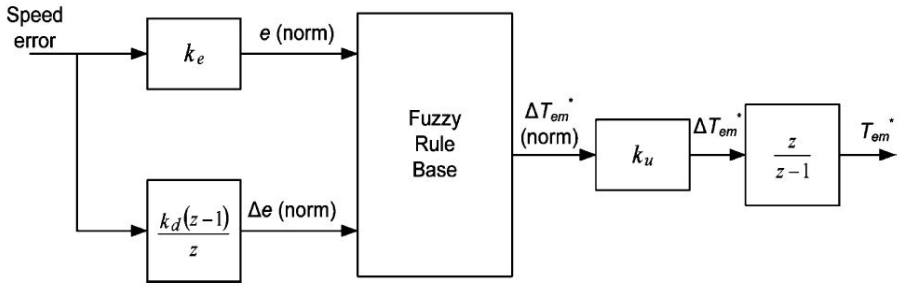

FIGURE 3.5: Le schéma fonctionnel du FLC

- Contrôleur par mode glissant flou :

Une autre méthode de remplacement du contrôleur PI est l'utilisation d'une commande en mode glissant (Sliding Mode Control SMC). Il s'agit d'une stratégie avec un contrôle à rétroaction commutée à haute fréquence qui force les états du système à glisser sur une hypersurface prédéfinie. Le SMC est connu pour sa capacité

à faire face aux perturbations limitées ainsi que pour les imprécisions des modèles, ce qui le rend idéal pour la commande non linéaire robuste des moteurs à induction. La fonction de commutation est définie par :

$$s(t) = e(t) - \int_0^t ke(\tau)\, d\tau$$

Avec $e(t) = \omega_r - \omega_r^*$.

La structure du contrôleur en mode glissant peut s'écrire comme :

$$u^* = u_{eq} + u_s$$

Où u_{eq} est le contrôle sur le mode glissant. Le choix de u_s comme une fonction signe provoque un effet de chattering à haute fréquence en raison de l'action de contrôle discontinue, ce qui représente un grave problème lorsque l'état du système est proche de la surface de glissement. Pour obtenir un contrôle par mode glaisant flou (Fuzzy Sliding Mode FSM), la commande u_s est remplacé par un système d'inférence floue.

Le schéma fonctionnel du système de contrôle est illustré dans la figure suivante :

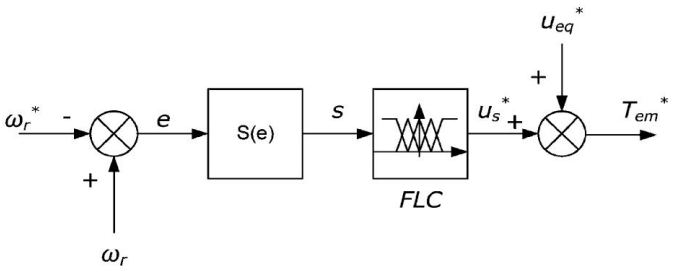

FIGURE 3.6: Contrôleur de vitesse par mode glissant flou

- Comparaison :

Pour comparer les différentes stratégies de conception des régulateurs de vitesse, un moteur à induction à cage d'écureuil de 7.5 kW est simulé à l'aide du logiciel Matlab-Simulink en utilisant le modèle de machine à deux axes bien établi.

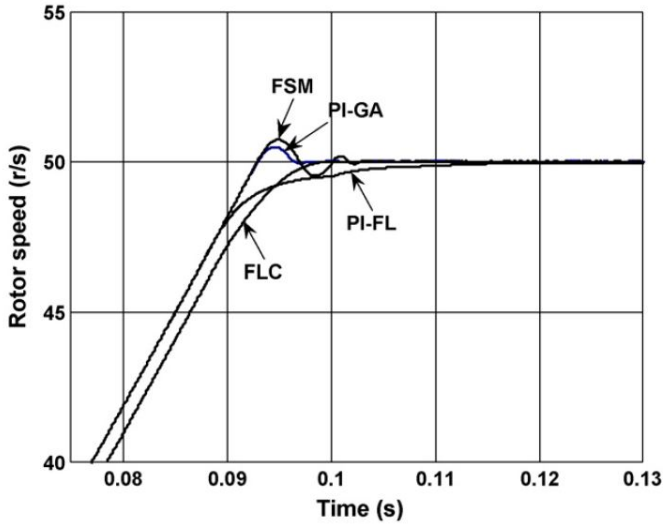

FIGURE 3.7: Comparaison des réponses des régulateurs

PI-GA : régulateur PI par l'algorithme génétique.

PI-FL : régulateur PI par logique floue.

FLC : régulateur par logique floue.

FSM : régulateur par mode glaisant flou.

Le FLC a la meilleure réponse transitoire lorsque la vitesse du moteur augmente approximativement en moins de 0.1 *sec* sans dépassement. Le PI-FL a une réponse sur-amortie où la vitesse du moteur s'accumule en 0.115 *sec* sans dépassement. Le PI-GA et le FSM ont un dépassement de vitesse de 1% et 1.4% , respectivement, ce qui est encore très faible.

Le PI-GA fonctionne bien dans des conditions de fonctionnement normales, produisant un léger dépassement mais ayant une faible capacité de rejet des perturbations du couple en raison du contrôleur à gain fixe, il peut prendre beaucoup de temps pour converger vers la solution optimale. Le PI-FL est plus performant que le PI-GA à gain fixe lors d'une perturbation du couple de charge. Par rapport au FLC, le PI-FL est plus robuste contre les variations des paramètres du moteur et offre de meilleures performances en régime permanent puisque la mise à jour du gain s'arrête après une limite donnée de précision de la vitesse. Il offre également de meilleures performances en régime permanent par rapport au FSM qui est affecté par le chattering en régime permanent. Le FLC a une meilleure capacité de rejet des perturbations par rapport au PI-FL et une meilleure réponse transitoire au démarrage.

• Les résultats de la publication sont :

- Dans cet article, quatre stratégies de conception du régulateur de vitesse du moteur à induction DTC sont présentées : le régulateur PI par un algorithme génétique et par logique floue, le régulateur par mode glissant flou et le régulateur par logique floue. Ces méthodes de conception sont basées sur des techniques d'intelligence artificielle qui ne nécessitent aucune modélisation mathématique. Toutes ces techniques fonctionnent bien dans des conditions de fonctionnement normales.

- Les contrôleurs à structure adaptative présentent une plus grande robustesse contre les variations des paramètres des moteurs ainsi qu'une capacité de rejet des perturbations élevée par rapport aux techniques à structure fixe.

- Le contrôleur de vitesse à logique floue nécessite quelques modifications pour améliorer ses performances en régime permanent.

3.4 Un contrôleur adaptatif à l'aide d'un réseau de neurones

En 2009, des professeurs de l'institut indien des sciences ont publié un article intitulé « A direct adaptive neural command controller design for an unstable helicopter »[14]. Leur travail consiste à développer un contrôleur adaptatif à l'aide d'un réseau de neurones pour la commande d'un hélicoptère instable.

La conception traditionnelle des régulateurs de vol pour les hélicoptères implique la linéarisation de la dynamique du véhicule dans plusieurs conditions de fonctionnement, et la planification des gains des régulateurs linéaires pour chaque condition avec un schéma d'interpolation. La limite des techniques classiques de planification des gains est qu'elles n'exploitent le comportement qu'au voisinage des points de fonctionnement à l'équilibre et qu'elles imposent généralement au système des variations lentes, pour faire en sorte que l'état reste proche de l'équilibre.

Une approche par réseau de neurones tente de détendre les restrictions pour fonctionner près du point d'équilibre. Le réseau est entraîné hors ligne à l'aide de l'algorithme de rétro-propagation par apprentissage temporel pour se rapprocher de la loi de contrôle inconnue.

Le contrôleur par réseau de neurones utilisé dans cet article est un contrôleur neuronal adaptatif direct. Dans cette stratégie, un modèle de référence est mis en œuvre dans le schéma. L'erreur est la différence entre la sortie du modèle de l'hélicoptère et le bloc de référence est renvoyé au contrôleur obtenu à partir d'un modèle de référence.

Ce réseau de neurones a une couche d'entrée composée des entrées de contrôle et des sorties retardées et actuelles, une couche cachée, et la sortie du réseau est la commande u . Les paramètres du contrôleur sont mis à jour à l'aide d'une synthèse de Lyapunov.

Le schéma fonctionnel du contrôleur neuronal adaptatif direct (Direct Adaptive Neural Controller DANC) est présenté dans la figure suivante. Le contrôleur par retour d'état classique dans la boucle interne est utilisé pour stabiliser l'hélicoptère et le contrôleur neuronal dans la boucle externe se rapproche de la non-linéarité inconnue et fournit les performances de suivi nécessaire. Le contrôleur neural est formé pour minimiser l'écart entre le signal de référence et la sortie réelle de l'hélicoptère. L'effort de contrôle appliqué à l'hélicoptère est la somme des signaux du contrôleur conventionnel et du contrôleur neuronal.

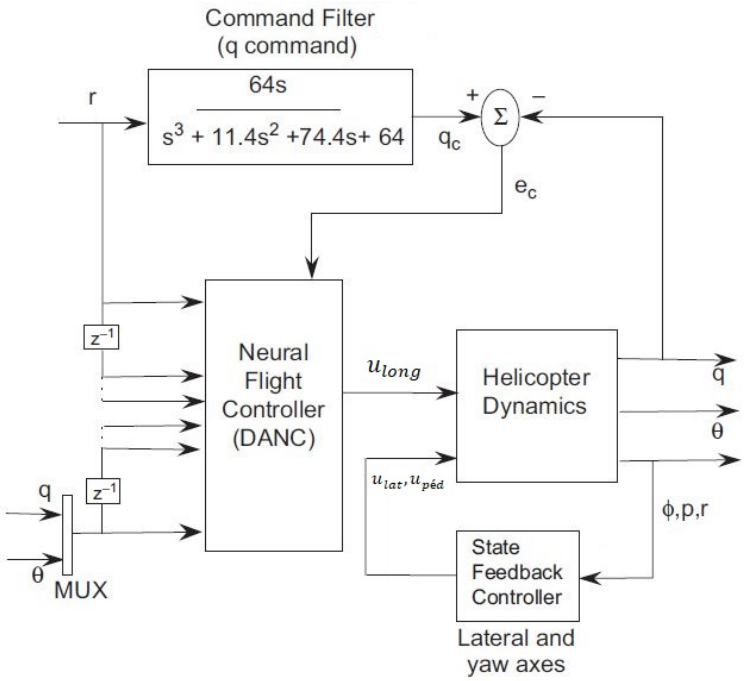

FIGURE 3.8: Le schéma fonctionnel du DANC

Où u_{lat} est l'entrée cyclique latérale, u_{long} est l'entrée cyclique longitudinale et

$u_{péd}$ est l'entrée de la pédale.

- Les résultats de la publication sont :

- Le contrôleur DANC est capable de s'adapter aux défauts aérodynamiques et aux défauts de la surface de contrôle.

- Ces résultats démontrent la capacité d'apprentissage en ligne du réseau neuronal pour s'adapter aux incertitudes des paramètres.

- La stratégie de contrôle proposée utilisant le neuro-contrôleur présente de bonnes performances de suivi pour un système d'hélicoptère instable.

3.5 Conclusion

Lors de ce chapitre, nous avons présentés quatre publications concernant les applications de l'intelligence artificielle sur les régulateurs automatiques, la première s'intéresse par l'application de Q-Learning pour ajuster les paramètres d'un PID pour un robot de football, la deuxième applique une régulation par l'algorithme Fitted Value Iteration pour un drone, la troisième est une comparaison entre quatre algorithmes : un régulateur PI par un algorithme génétique, un régulateur PI par logique floue, un contrôleur par logique floue et un contrôleur par mode glissant flou, la dernière utilise un régulateur adaptatif à l'aide d'un réseau de neurones.

Chapitre 4

Identification par réseau de neurones pour la régulation

La fonction de transfert est la représentation du système, et la connaissance de cette fonction permet d'appliquer plusieurs méthodes de régulation. Le problème avec les processus et les systèmes réels est que la fonction de transfert est inconnue. Dans ce chapitre nous avons développé un algorithme d'intelligence artificielle pour l'identification de la fonction de transfert d'un système.

4.1 L'identification pour la régulation

L'une des méthodes de régulation d'un système est d'identifier le procédé à réguler. Cette identification consiste à trouver les paramètres permettant de décrire ce procédé sous forme d'un modèle mathématique. Nous avons choisi d'identifier des systèmes de deuxième ordre de la forme suivante :

$$G(p) = \frac{a_0}{p^2 + b_1 p + b_0}$$

Avec $a_0 \neq 0$.

La fonction de transfert nous permettre de trouver un régulateur pour le système. Si elle est connue, nous pouvons utiliser la méthode de lieu des racines ou de placement des pôles pour la régulation. Il existe d'autres méthodes et algorithmes qui peuvent être utilisés.

Nous avons appliqué l'algorithme développé dans notre mémoire de fin d'études

intitulé « Apprentissage par renforcement pour l'ajustement automatique des régulateurs en boucle », où nous utilisons l'algorithme Deep Q-Network pour trouver les paramètres d'un régulateur PID.

Dans ce mémoire, nous allons développer un algorithme qui permet d'identifier les paramètres d'une fonction de transfert à partir de la réponse indicielle de ce système. Nous utilisons un réseau de neurones artificiel dont le concept et le fonctionnement sont présentés dans la section suivante.

4.2 Le réseau de neurones artificiel

Les réseaux de neurones artificiels sont parmi les principaux outils utilisés en machine learning, ils sont efficaces pour résoudre les problèmes de reconnaissance des formes, de classification, de regroupement et d'approximation des fonctions.

Un réseau de neurones est constitué d'un ensemble de nœuds, appelés neurones formels, modelés sur le fonctionnement et la constitution d'un vrai neurone biologique, le réseau reçoit des entrées et sur la base de son traitement interne, il génère une sortie. La meilleure sortie est celle qui produit la plus petite erreur entre les valeurs réelles Y et les valeurs prédites par le réseau $\hat{Y}$ sur différents échantillons de l'ensemble de données.

Maintenant, nous allons essayer de comprendre comment un neurone formel est modélisé mathématiquement (figure 4.1).

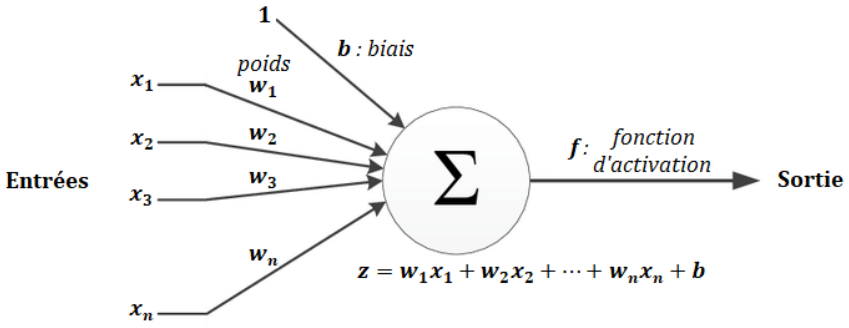

FIGURE 4.1: La structure d'un neurone formel

Les valeurs d'entrée $X = \{x_1, x_2, \ldots, x_n\}$ sont fournies au neurone, ces entrées sont pondérées avec les poids $W = \{w_1, w_2, \ldots, w_n\}$. En plus, le neurone pourrait avoir un biais b .

C'est ce vecteur de poids W que nous devons entraîner ou, en d'autres termes, trouver les valeurs optimales des éléments de poids afin d'obtenir le meilleur résultat.

Le traitement à l'intérieur du neurone est effectué à l'aide d'une fonction appelée « fonction d'activation ». L'entrée de la fonction d'activation f est égale à :

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

La sortie du neurone pourrait être représentée par $f(z)$ .Le biais peut-être considérer comme une entrée égale à 1 pondérée par le poids b .

Une fonction d'activation appropriée doit être choisie pour l'objectif visé par le réseau de neurones.

4.2.1 Fonction d'activation

Parmi les fonctions les plus utilisées comme fonctions d'activation :

Nom	Équation
Identité	$f(z) = z$
Rampe (Rectified Linear Units ReLU)	$f(z) = \begin{cases} 0 & si\ z < 0 \\ z & si\ z \geq 0 \end{cases}$
Sigmoïde	$f(x) = \dfrac{1}{1+e^{-z}}$
Tangente hyperbolique	$f(z) = \tanh(z) = \dfrac{2}{1+e^{-2z}} - 1$

Tableau 4.1: Exemple de fonction d'activation

4.2.2 Perceptrons multicouches

Il existe de nombreuses structures de réseaux de neurones. Nous avons choisi de travailler avec des perceptrons multicouches pour des raisons de simplicité et pour leurs bonnes performances, en théorie ils peuvent approximer n'importe quelle fonction avec un nombre de neurones suffisant. Le réseau est organisé en couches où chaque neurone est connecté à l'ensemble des neurones de la prochaine couche.

Un perceptron multicouche est obtenu en combinant plusieurs neurones sur plusieurs couches. Il s'agit d'un modèle non linéaire à la fois en sortie et en paramètre.

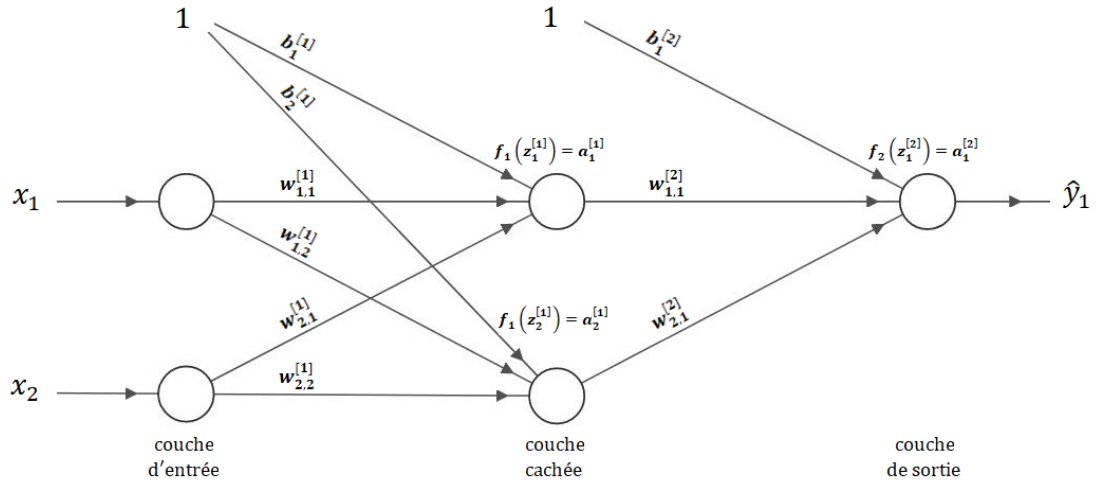

FIGURE 4.2: Un perceptron multicouches

- Les sorties de la couche d'entrée :

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Les sorties de la couche cachée :

$$\begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} = \begin{bmatrix} f_1\left(z_1^{[1]}\right) \\ f_1\left(z_2^{[1]}\right) \end{bmatrix} = \begin{bmatrix} f_1\left(w_{1,1}^{[1]}x_1 + w_{2,1}^{[1]}x_2 + b_1^{[1]}\right) \\ f_1\left(w_{1,2}^{[1]}x_1 + w_{2,2}^{[1]}x_2 + b_2^{[1]}\right) \end{bmatrix}$$

- La sortie de la couche de sortie :

$$\left[a_1^{[2]}\right] = \left[f_2\left(z_1^{[2]}\right)\right] = \left[f_2\left(w_{1,1}^{[2]}a_1^{[1]} + w_{2,1}^{[2]}a_2^{[1]} + b_1^{[2]}\right)\right]$$

4.2.3 Processus d'approximation

a) Initialisation :

La première étape de l'apprentissage c'est de commencer quelque part. Les réseaux de neurones peuvent commencer de n'importe où, c'est pourquoi une initialisation aléatoire des poids est une pratique courante.

Le raisonnement est que nous pouvons atteindre le modèle pseudo-idéal par un modèle approprie et un nombre d'itérations suffisant.

b) Propagation en avant (forward-propagation) :

Nous partons des données dont nous disposons, nous les faisons passer par les couches du réseau et nous calculons directement les sorties.

Généralement, l'apprentissage dans un réseau de neurones ne se fait pas par une seule donnée mais plutôt par un nombre d'exemplaire m appelé lot (batch), l'entrée serait donc la matrice X de taille $n \times m$ (n est le nombre de neurones de la couche d'entrées et m est le nombre d'exemplaire pour chaque entrée) :

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{bmatrix}$$

On note aussi $W^{[1]}$ la matrice des poids de la couche cachée de taille $n_1 \times n$, et le vecteur des biais $B^{[1]}$ de taille $n_1 \times 1$:

$$W^{[1]} = \begin{bmatrix} w_{1,1}^{[1]} & w_{2,1}^{[1]} & \dots & w_{n,1}^{[1]} \\ w_{1,2}^{[1]} & w_{2,2}^{[1]} & \dots & w_{n,2}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,n_1}^{[1]} & w_{2,n_1}^{[1]} & \dots & w_{n,n_1}^{[1]} \end{bmatrix} \quad , \quad B^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_{n_1}^{[1]} \end{bmatrix}$$

La matrice des poids de la couche de sortie $W^{[2]}$ est de taille $n_2 \times n_1$ avec un vecteur des biais $B^{[2]}$ de taille $n_2 \times 1$.

L'entrée des neurones de la couche cachée est la matrice $Z^{[1]}$ de taille $n_1 \times m$, et la matrice de sortie est $A^{[1]}$ de taille $n_1 \times m$ (n_1 nombre de neurones de la première couche cachée) :

$$Z^{[1]} = \begin{bmatrix} z_1^{[1](1)} & z_1^{[1](2)} & \dots & z_1^{[1](m)} \\ z_2^{[1](1)} & z_2^{[1](2)} & \dots & z_2^{[1](m)} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n_1}^{[1](1)} & z_{n_1}^{[1](2)} & \dots & z_{n_1}^{[1](m)} \end{bmatrix} = W^{[1]}X + B^{[1]} \quad , \quad A^{[1]} = f_1\left(Z^{[1]}\right)$$

L'entrée de la couche de sortie est la matrice $Z^{[2]}$ de taille $n_2 \times m$, et la matrice de sortie est $\hat{Y} = A^{[2]}$ de taille $n_2 \times m$ (n_2 nombre de neurones de la couche de sortie) :

$$Z^{[2]} = W^{[2]}A^{[1]} + B^{[2]} \quad , \quad \hat{Y} = A^{[2]} = f_2\left(Z^{[2]}\right)$$

c) Fonction de perte (loss function) :

Pendant l'apprentissage du réseau, nous devons calculer l'erreur (loss) pour optimiser les poids dans chaque couche. Cette erreur est calculée à l'aide d'une fonction de perte.

La fonction de perte est une mesure de performance du réseau pour atteindre son objectif de produire des résultats proches des valeurs souhaitées.

Nous avons $\hat{Y} = \begin{bmatrix} \hat{Y}^{(1)} & \hat{Y}^{(2)} & \dots & \hat{Y}^{(m)} \end{bmatrix}$ la sortie prédite de notre réseau neuronal, et $Y = \begin{bmatrix} Y^{(1)} & Y^{(2)} & \dots & Y^{(m)} \end{bmatrix}$ la sortie réelle souhaitée.

Parmi les fonctions de perte utilisées pour chaque $\hat{Y}^{(i)}$ (i entre 1 et m) de n_2 neurones en sortie :

i. L'erreur absolue est la plus simple et intuitive :

$$loss\,(i) = \left|Y^{(i)} - \hat{Y}^{(i)}\right| = \sum_{j=1}^{n_2} \left|y_j^{(i)} - \hat{y}_j^{(i)}\right|$$

ii. La fonction erreur quadratique pour les problèmes de régression :

$$loss\,(i) = \frac{1}{2}\left(Y^{(i)} - \hat{Y}^{(i)}\right)^2 = \frac{1}{2}\sum_{j=1}^{n_2} \left(y_j^{(i)} - \hat{y}_j^{(i)}\right)^2$$

iii. La fonction entropie croisée (CrossEntropy) pour les problèmes de

classification :

$$loss\,(i) = -\sum_{j=1}^{n_2} \left[\left(y_j^{(i)} \right) \log \left(\hat{y}_j^{(i)} \right) \; + \left(1 - y_j^{(i)} \right) \log \left(1 - \hat{y}_j^{(i)} \right) \right]$$

iv. La fonction de perte Huber (Huber loss) :

$$loss\,(i) = \sum_{j=1}^{n_2} e_j$$

Avec :

$$e_j = \begin{cases} \frac{1}{2}\left(y_j^{(i)} - \hat{y}_j^{(i)} \right)^2 & si \;\; \left| y_j^{(i)} - \hat{y}_j^{(i)} \right| < 1 \\ \left| y_j^{(i)} - \hat{y}_j^{(i)} \right| - \dfrac{1}{2} & ailleurs \end{cases}$$

On peut aussi utiliser d'autres formes suivant l'objectif du réseau de neurones.

d) Fonction de coût (cost function) :

Nous utilisons une fonction de coût (cost function) pour évaluer les performances du réseau sur des lots de taille m , et pour actualiser les poids et les biais :

$$cost = \frac{1}{m} \sum_{i=1}^{m} loss\,(i)$$

e) Descente de gradient :

Le gradient des poids :

$$\frac{\partial cost}{\partial W^{[k]}} = \frac{\partial cost}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial W^{[k]}}$$

Et le gradient des biais :

$$\frac{\partial cost}{\partial B^{[k]}} = \frac{\partial cost}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial B^{[k]}}$$

L'exemple suivant nous montre comment calculer les gradients des poids et des biais pour un réseau d'une seule couche cachée. Pour $m = 1$, $n_2 = 1$, des fonctions d'activation identités et une fonction de perte erreur quadratique, la fonction de coût est :

$$cost = \frac{1}{1} \sum_{i=1}^{1} loss\,(i) = loss\,(1) = \frac{1}{2} \sum_{j=1}^{1} \left(y_j^{(1)} - \hat{y}_j^{(1)} \right)^2 = \frac{1}{2} \left(y_1^{(1)} - \hat{y}_1^{(1)} \right)^2$$

- Gradient de $W^{[2]}$:

$$\frac{\partial cost}{\partial W^{[2]}} = \frac{\partial loss\,(1)}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial Z^{[2]}} \frac{\partial Z^{[2]}}{\partial W^{[2]}}$$

Nous avons d'une part :

$$\frac{\partial loss\,(1)}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial Z^{[2]}} = \frac{\partial \frac{1}{2}\left(y_1^{(1)} - \hat{y}_1^{(1)}\right)^2}{\partial \hat{y}_1^{(1)}} \frac{\partial Z^{[2]}}{\partial Z^{[2]}} = -\left(y_1^{(1)} - \hat{y}_1^{(1)}\right)$$

Et d'autre part :

$$\frac{\partial Z^{[2]}}{\partial W^{[2]}} = \frac{\partial \left(W^{[2]} A^{[1]} + B^{[2]}\right)}{\partial W^{[2]}} = \left(A^{[1]}\right)^T$$

Donc :

$$\frac{\partial cost}{\partial W^{[2]}} = -\left(y_1^{(1)} - \hat{y}_1^{(1)}\right)\left(A^{[1]}\right)^T$$

- Gradient de $B^{[2]}$:

$$\frac{\partial cost}{\partial B^{[2]}} = \frac{\partial cost}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial Z^{[2]}} \frac{\partial Z^{[2]}}{\partial B^{[2]}}$$

Nous avons :

$$\frac{\partial Z^{[2]}}{\partial B^{[2]}} = \frac{\partial \left(W^{[2]} A^{[1]} + B^{[2]}\right)}{\partial B^{[2]}} = 1$$

Donc :

$$\frac{\partial cost}{\partial B^{[2]}} = -\left(y_1^{(1)} - \hat{y}_1^{(1)}\right)$$

- Gradient de $W^{[1]}$:

$$\frac{\partial cost}{\partial W^{[1]}} = \left(W^{[2]}\right)^T \left(-y_1^{(1)} + \hat{y}_1^{(1)}\right)\left(X^{[1]}\right)^T$$

- Gradient de $B^{[1]}$:

$$\frac{\partial cost}{\partial B^{[1]}} = \left(W^{[2]}\right)^T \left(-y_1^{(1)} + \hat{y}_1^{(1)}\right)$$

f) Rétro-propagation (back-propagation) :

Nous avons le point de départ des erreurs, qui est la fonction de perte, et nous savons comment calculer le gradient de chaque poids, nous pouvons propager l'erreur de la fin au début.

Un taux d'apprentissage α est introduit comme une constante entre 0 et 1 (généralement très faible), afin de forcer l'actualisation du poids à être très douce et lente (pour éviter les grands pas et les comportements chaotiques).

Nous utilisons la règle suivante pour l'actualisation des poids à chaque itération t , où $t-1$ est l'itération précédente :

$$W_t^{[k]} = W_{t-1}^{[k]} - \alpha \frac{\partial cost}{\partial W^{[k]}}$$

Et pour l'actualisation des biais :

$$B_t^{[k]} = B_{t-1}^{[k]} - \alpha \frac{\partial cost}{\partial B^{[k]}}$$

g) Convergence :

Comme nous mettons à jour les poids avec un petit pas, il faudra plusieurs itérations pour apprendre le réseau et minimiser l'erreur.

Le gradient de la fonction de perte nous assure que la sortie de notre réseau converge vers la valeur réelle $\hat{Y} = Y$.

4.3 Développement

Dans le cas de notre algorithme, nous avons utilisé un réseau de neurones à plusieurs couches cachées, où les poids et les biais sont initialisés aléatoirement.

Les entrées de ce réseau sont des paramètres obtenus à partir de la réponse indicielle du système. Ces paramètres sont passés par lot (batch) à travers les différentes couches pour obtenir les sorties prédites (les paramètres du système). Ensuite, le coût est calculé à partir des pertes entre les prédictions et les valeurs réelles.

Cette erreur est propagée de la sortie vers l'entrée en actualisant les poids et biais de notre réseau. Nous évaluons les performances de notre réseau par la valeur de la perte (*loss*).

4.3.1 Les données utilisées

Nous avons utilisé un algorithme sur Matlab pour créer une base de données de taille entre 300000 et 500000 exemplaires. Les valeurs des entrées sont les paramètres de la réponse indicielle d'un système $G(p)$, obtenues par la fonction « stepinfo ». Les sorties sont les trois paramètres a_0, b_0 et b_1 de la fonction de transfert $G(p)$.

Cette base de données est divisée en deux ensembles : un ensemble d'entraînement et un ensemble de test. Le premier ensemble est utilisé pour entraîner un modèle, ce dernier est testé par le deuxième ensemble. Nous utilisons le même ensemble de test pour la validation, cette étape est très importante pour savoir si nous avons un problème de sous-apprentissage ou de sur-apprentissage en comparant les valeurs de l'erreur et la variance.

4.4 Environnement de travail

4.4.1 Environnement matériel

Dans la plupart des expériences, nous avons utilisé la configuration matérielle suivante :

- Système d'exploitation : Windows 10 professionnel
- CPU : Intel $i7$, 3.5 GHz
- RAM : 12 GO, 2400 MHz

4.4.2 Environnement logiciel

a) Langage de programmation et librairie utilisées :

Le développement de notre algorithme était fait par le langage de programmation Python, approprie pour les applications d'intelligence artificielle. Nous avons utilisé la version 3.6 .

Parmi les librairies que nous avons utilisées : TensorFlow pour les applications d'apprentissage automatique, Keras pour la création des réseaux de neurones, NumPy pour les calculs matriciels et Scipy.io pour importer des fichiers Matlab sur Python.

b) Environnement de développement :

Nous avons utilisé l'environnement Spyder dans le logiciel Anaconda pour sa flexibilité et son support des outils d'inspection des données.

c) Matlab 2019b :

Matlab est un environnement très puissant que nous avons utilisé pour générer des données à partir des fonctions de transfert.

4.5 Implémentation de l'algorithme

4.5.1 Les expériences

Afin de maximiser les performances de l'algorithme, nous avons effectué plusieurs expériences sur ses différents éléments. Nous avons généré des données par des fonctions de transfert aléatoires, et après l'entraînement du modèle nous comparons les valeurs de la précision.

Le tableau suivant présente quelques expériences et les valeurs de la précision obtenues :

L'expérience	La précision
Des fonctions de transfert aléatoires	34.2%
Taille du réseau	57.8%
Des fonctions de transfert avec des valeurs < 1	84.4%
Normalisation des entrées	92.8%
Changement des fonctions d'activation	98.1%

Tableau 4.2: Comparaison entre les expériences

4.5.2 Algorithme

```
import scipy.io as scio
import numpy as np
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Dense, Dropout
```

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

var1 = scio.loadmat('Matlab_input.mat')
var2 = scio.loadmat('Matlab_output.mat')
py_input=np.array(var1['Matlab_input'])
py_output=np.array(var2['Matlab_output'])

X_train, X_test, Y_train, Y_test = train_test_split(py_input,
    py_output, test_size = 0.01, random_state = 0)

sc = StandardScaler()
X_train_n = sc.fit_transform(X_train)
X_test_n = sc.transform(X_test)

model =Sequential()
model.add(Dense(16, input_shape=(7,)))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(8))
model.add(Activation('relu'))
model.add(Dense(3))

model.compile(loss='mean_squared_error', optimizer='adam', metrics
    =['accuracy'])
model.fit(X_train_n, Y_train_n, epochs=150, batch_size=32)

model.evaluate(X_test_n, Y_test, batch_size=32)

result=model.predict(X_test_n)
```

Au début de l'algorithme nous importons toutes les librairies nécessaires, et la base de données de taille 324110 (l'entrée est *Matlab_input.mat* et la sortie est *Matlab_output.mat*). Pour pouvoir utiliser ces données il faut les convertir à des tableaux *py_input* et *py_output* .

Ces tableaux sont divisés à deux ensembles pour l'entraînement et le test par la commande *train_test_split* . Les entrées sont ensuite normalisées par l'outil *StandardScaler* qui permet de normaliser l'entrée d'entraînement et l'entrée de test par la même normalisation.

Par la suite, le réseau de neurones est créé avec une couche d'entrée de 7 neurones (les paramètres de « stepinfo »), trois couches cachées de 16, 32 et 8 neurones respectivement et une couche de sortie de 3 neurones (les paramètres de la fonction de transfert). Les fonctions d'activation sont du type ReLU.

Nous spécifions la fonction de perte (*loss*), l'actualisation des poids et biais (*optimizer*) et le critère d'évaluation (*metrics*) par la commande (*compile*). L'entraînement se fait sur tout l'ensemble de l'entraînement par la commande fit sur 150 époques et avec un lot de taille 32 .

Nous évaluons notre modèle par la commande *evaluate* pour éviter les problèmes de sous-apprentissage ou sur-apprentissage. Et les résultats de test sont obtenus par la commande *predict* .

4.5.3 Résultat

	loss	*accuracy*
Résultat d'entraînement	4.3515×10^{-4}	98.12%
Résultat d'évaluation	4.5297×10^{-4}	98.56%

Tableau 4.3: Comparaison entre l'entrainement et l'évaluation

Les valeurs de perte et de précision sont très proche ce qui signifie que l'entraînement est bon.

4.6 Applications et résultats

4.6.1 Applications 1

Nous allons utiliser le modèle obtenu par l'entraînement pour l'identification de la fonction de transfert suivante :

$$G_1(p) = \frac{2}{p^2 + 1.5p + 0.025}$$

Notre modèle a prédit la fonction suivante :

$$G_{1,p}(p) = \frac{1.98701}{p^2 + 1.50320p + 0.02557}$$

La figure suivante est une comparaison entre les réponses indicielles de ces deux fonctions de transfert :

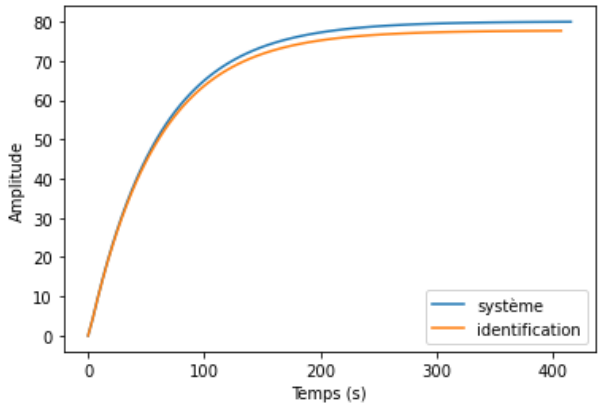

FIGURE 4.3: Le système G_1 et son identification

Nous avons appliqué l'algorithme de régulation pour le système identifié $G_{1,p}$, puis nous utilisons les paramètres obtenus du PID pour la régulation du système réel G_1 :

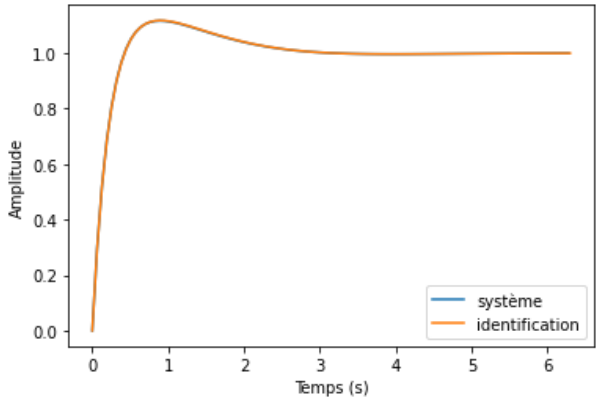

FIGURE 4.4: La régulation du système G_1 et son identification

Le système identifié est très proche du système réel et la régulation des deux fonctions de transfert est presque la même.

4.6.2 Applications 2

Nous identifions de la fonction de transfert suivante :

$$G_2(p) = \frac{7.1}{p^2 + 0.73p + 0.75}$$

Notre modèle à prédit le système suivant :

$$G_{2,p}(p) = \frac{6.99293}{p^2 + 0.\ 73611p + 0.74819}$$

La figure suivante est une comparaison entre les réponses indicielles de ces deux fonctions de transfert :

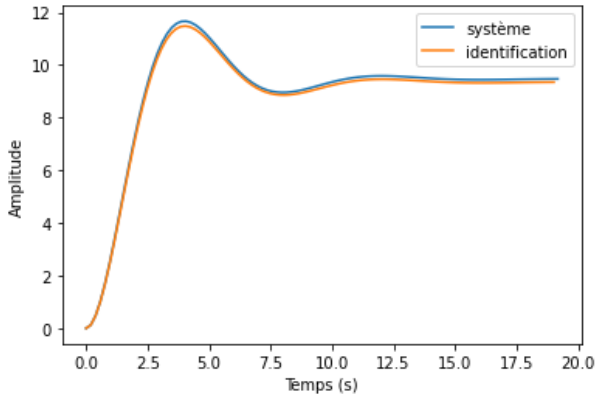

FIGURE 4.5: Le système G_2 et son identification

Nous avons appliqué l'algorithme de régulation pour le système identifié $G_{2,p}$, puis nous utilisons les paramètres obtenus du PID pour la régulation du système réel G_2 :

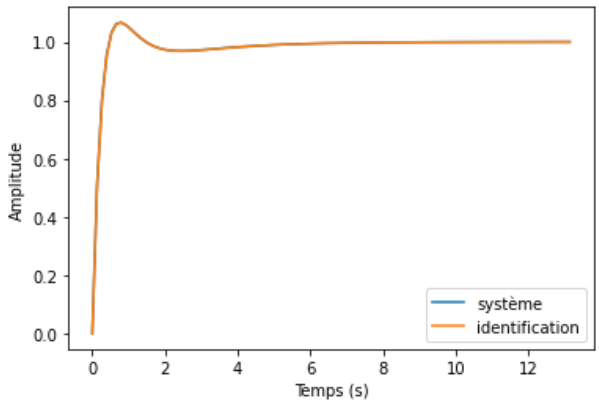

FIGURE 4.6: La régulation du système G_2 et son identification

Le système identifié est très proche du système réel et la régulation des deux fonctions de transfert est presque la même.

4.6.3 Applications 3

Nous identifions de la fonction de transfert suivante :

$$G_3(p) = \frac{0.8}{p^2 + 0.137p + 0.064}$$

Notre modèle à prédit la fonction suivante :

$$G_{3,p}(p) = \frac{0.77166}{p^2 + 0.13308p + 0.05452}$$

La figure suivante est une comparaison entre les réponses indicielles de ces deux fonctions de transfert :

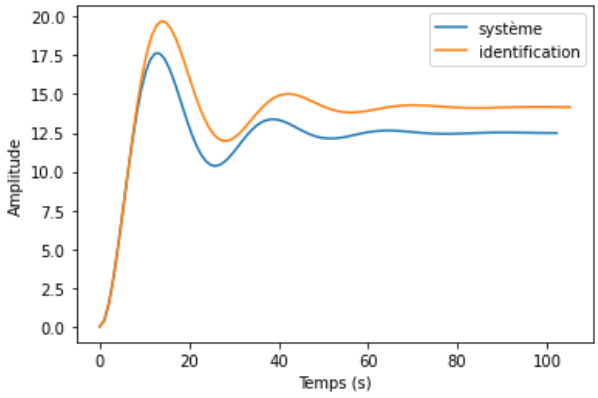

FIGURE 4.7: Le système G_3 et son identification

Nous avons appliqué l'algorithme de régulation pour le système identifié $G_{3,p}$, puis nous utilisons les paramètres obtenus du PID pour la régulation du système réel G_3 :

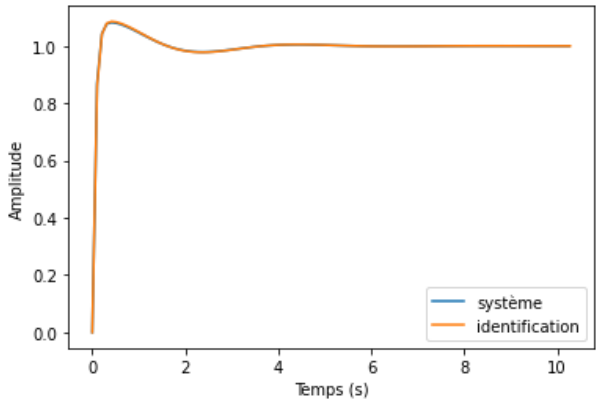

FIGURE 4.8: La régulation du système G_3 et son identification

Le système identifié est similaire au système réel mais il n'est pas proche, par contre, la régulation des deux fonctions de transfert est presque la même.

4.6.4 Résultats

L'algorithme développé dans ce chapitre permet d'identifier les fonctions de transfert des systèmes de deuxième ordre avec une bonne précision pour la majorité des cas. L'identification de certains systèmes n'était pas possible ce qui indique que l'algorithme nécessite plus de développement.

Les fonctions de transfert obtenues par l'identification permettent de trouver des bons paramètres de régulation pour les systèmes réels.

4.7 Conclusion

Durant ce chapitre, nous avons développé un algorithme qui permet d'identifier les paramètres d'une fonction de transfert à partir de sa réponse indicielle. Notre algorithme utilise un réseau de neurones pour l'estimation du système. Le modèle obtenu par l'entraînement permet d'identifier la plupart des systèmes de deuxième ordre.

La fonction de transfert obtenue par l'identification peut être utilisée pour obtenir un régulateur pour le système réel.

Conclusion générale

L'objectif de ce mémoire est de donner un aperçu des techniques émergentes en matière d'intelligence artificielle qui peuvent être utilisées pour contrôler les systèmes physiques. La conception des systèmes de contrôle intelligents utilisant ces techniques d'intelligence artificielle apporte ainsi de nouvelles dimensions au-delà de celles des systèmes de contrôle conventionnels avec la capacité de gérer des systèmes incertains, qui peuvent impliquer une représentation floue et/ou des opérations décrites symboliquement avec apprentissage et prise de décision. Bien qu'il soit évident que des progrès majeurs sont réalisés dans le domaine des systèmes de contrôle, nous n'en sommes encore qu'au début de la compréhension complète de la puissance des techniques d'intelligence artificielle et de leur utilisation dans nos opérations de contrôle. Des idées nouvelles et importantes font actuellement l'objet de recherches qui vont changer la nature même de l'ingénierie de contrôle dans les prochaines années.

Nous pouvons nous attendre à voir des contrôleurs capables de gérer des processus difficiles qui se sont révélés impossibles à modéliser avec les techniques de contrôle classiques. Il reste beaucoup à faire pour affiner ces techniques à l'aide des théories solides et cohérentes, pour évaluer et classer leur applicabilité afin que le processus de conception de contrôleurs intelligents puisse être placé sur une base technique ferme.

Les deux problèmes les plus difficiles pour l'application des systèmes de contrôle intelligents dans un usage industriel réel sont les suivants : il n'existe pas de méthodologie mature pour concevoir et vérifier un système de contrôle intelligent pour une variété de processus, et il n'existe pas d'environnement de mise en œuvre solide. Si ces problèmes sont résolus, nous assisterons à l'émergence d'un certain nombre de systèmes de contrôle intelligents pour des systèmes

complexes dans un avenir proche.

L'algorithme développé durant ce mémoire permet d'identifier un système, nous avons utilisé un réseau de neurones pour trouver les paramètres de sa fonction de transfert afin de concevoir un contrôleur pour le système à identifier.

Bibliographie

[1] Anurag. Artificial intelligence vs machine learning vs data science. (Consulté le 23/06/2020)
URL : https://www.newgenapps.com/blog/artificial-intelligence-vs-machine-learning-vs-data-science/.

[2] Bou-Ammar, H., Voos, H., et Ertel, W. Controller design for quadrotor uavs using reinforcement learning. In *2010 IEEE International Conference on Control Applications* (2010), IEEE.

[3] Boullart, L., Krijgsman, A., et Vingerhoeds, R. *Application of artificial intelligence in process control : lecture notes Erasmus intensive course*. Elsevier, 2013.

[4] Brahim, C. *Introduction à la commande adaptative*. 2019.

[5] CHEMORI, A. *Introduction à la commande adaptative*. Montpellier, 2015.

[6] Crowe, J., Chen, G., Ferdous, R., Greenwood, D., Grimble, M., Huang, H., Jeng, J., Johnson, M. A., Katebi, M., Kwong, S., et al. *PID control : new identification and design methods*. Springer, 2005.

[7] Dernoncourt, F. Introduction à la logique floue. (Consulté le 23/06/2020)
URL : https://franck-dernoncourt.developpez.com/tutoriels/algo/introduction-logique-floue.

[8] Duc, G. *Commande H_∞ et μ-Analysis, des outils pour la robustesse*. HERMES Science Publishing, Paris, 1999.

[9] El Hakim, A., Hindersah, H., et Rijanto, E. Application of reinforcement learning on self-tuning pid controller for soccer robot multi-

agent system. In *2013 joint international conference on rural information & communication technology and electric-vehicle technology (rICT & ICeV-T)* (2013), IEEE.

[10] Frank, S. A. *Adaptive Control*. Springer, 2018.

[11] Gadoue, S. M., Giaouris, D., et Finch, J. Artificial intelligence-based speed control of dtc induction motor drives—a comparative study. *Electric Power Systems Research* (2009).

[12] Johnson, J. 4 types of artificial intelligence. (Consulté le 23/06/2020)
URL : `https://www.bmc.com/blogs/artificial-intelligence-types`.

[13] Joshi, N. 7 types of artificial intelligence. (Consulté le 23/06/2020)
URL : `https://www.forbes.com/sites/cognitiveworld/2019/06/19/7-types-of-artificial-intelligence/#1809898a233e`.

[14] Kumar, M. V., Suresh, S., Omkar, S., Ganguli, R., et Sampath, P. A direct adaptive neural command controller design for an unstable helicopter. *Engineering Applications of Artificial Intelligence* (2009).

[15] Kumar, V., et Mani, N. The application of artificial intelligence techniques for intelligent control of dynamical physical systems. *International Journal of adaptive control and signal processing* (1994).

[16] L'encyclopédie libre Wikipédia. Adaptive control. (Consulté le 25/06/2020)
URL : `https://en.wikipedia.org/wiki/Adaptive_control`.

[17] L'encyclopédie libre Wikipédia. Choix des boucles de régulation. (Consulté le 25/06/2020)
URL : `https://fr.wikipedia.org/wiki/Choix_des_boucles_de_régulation`.

[18] L'encyclopédie libre Wikipédia. Logique floue. (Consulté le 23/06/2020)
URL : `https://fr.wikipedia.org/wiki/Logique_floue`.

[19] L'encyclopédie libre Wikipédia. Régulateur pid. (Consulté le 25/06/2020)
URL : https://fr.wikipedia.org/wiki/Régulateur_PID.

[20] Marr, B. Artificial intelligence : What's the difference between deep learning and reinforcement learning ? (Consulté le 23/06/2020)
URL : https://www.forbes.com/sites/bernardmarr/2018/10/22/artificial-intelligence-whats-the-difference-between-deep-learning-and-reinforcement-learning/#4d779067271e.

[21] O'Carroll, B. What are the 3 types of ai ? (Consulté le 23/06/2020)
URL : https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible.

[22] Ross, D., Deguine, E., et Camus, M. Asservissement par pid.

[23] Roy, R. Ai, ml, and dl : How not to get them mixed ! (Consulté le 23/06/2020)
URL : https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c?gi=702c9e0a886b.

[24] Srinidhi, S. Data science vs. artificial intelligence vs. machine learning vs. deep learning. (Consulté le 23/06/2020)
URL : https://towardsdatascience.com/data-science-vs-artificial-intelligence-vs-machine-learning-vs-deep-learning-9fadd8bda583?gi=60842b23f499.

[25] Tebbutt, C. D. *Expert aided control system design.* Springer Science & Business Media, 2012.

ملخص

يقدم الذكاء الاصطناعي أدوات قوية للمهندسين في مختلف المجالات، ولا سيما لتصميم أنظمة التحكم والمنظمين.

تهتم هذه المذكرة بتطورات وتطبيقات الذكاء الاصطناعي في مجال automatic controllers .
سوف نقدم مقالات حول هذا الموضوع والنتائج التي تم الحصول عليها.

لقد طورنا خوارزمية تستخدم شبكة عصبية لتحديد معاملات transfer function ، مما يسمح بتصميم controller للنظام الحقيقي.

الكلمات المفاحية: الذكاء الاصطناعي، automatic controller ، identification ، **الشبكة العصبية**.

Abstract

Artificial intelligence offers high-performance tools for engineers in various fields, particularly for the design of control systems and regulators.

This thesis focuses on the progress and applications of artificial intelligence for automatic controllers. We will present publications on this subject and the obtained results.

We have developed an algorithm that uses a neural network to identify the parameters of a transfer function, which allows to design a controller for the real system.

Keywords: artificial intelligence, automatic controller, identification, neural network.

Résumé

L'intelligence artificielle offre des outils performants pour les ingénieurs dans différents domaines, notamment pour la conception des systèmes de contrôles et des régulateurs.

Ce mémoire s'intéresse par l'avancement et les applications de l'intelligence artificielle pour les régulateurs automatiques, nous présenterons des publications sur ce sujet et les résultats obtenus.

Nous avons développé un algorithme qui utilise un réseau de neurones pour l'identification des paramètres d'une fonction de transfert, ce qui permet de concevoir un régulateur pour le système réel.

Mots-clés: intelligence artificielle, régulateur automatique, identification, réseau de neurones.