

الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

جامعة سعد دحلب البليدة

Université SAAD DAHLEB de BLIDA

كلية التكنولوجيا

Faculté de Technologies

قسم الإلكترونيك

Département d'électronique



## Mémoire de projet de fin d'études

Pour l'obtention du diplôme de MASTER en électronique  
option automatique

# Etude et implémentation sur DSP de la commande PID neuronal

Présenté par :

**BENRABAH Mohamed**

**DEHEBI Ali**

proposé par :

**M. K.KARA**

**M.O. AIT SAHED**

Session : juin 2016

## ملخص

يتمحور هذا العمل عن الدراسة النظرية، ومحاكاة المتحكم PID العصبي في نظام كيميائي غير خطي، والمحاكاة، وتصميم وتنفيذ المتحكم PID العصبي للمحرك DC. في الجزء الأول من هذا المشروع وصفنا أنظمة التحكم، التحديد والسيطرة العصبية للأنظمة لننتهي بشرح مفصل عن المتحكم PID العصبي. في الجزء الثاني قمنا ببعض المحاكاة ثم باشرنا في تركيب وتنفيذ المشروع، و قمنا بإنشاء الاتصال التسلسلي RS232 بين الكمبيوتر و DSP، وأخيرا قمنا برمجة واجهة المستخدم الرسومية باستخدام C # لإدارة هذا الاتصال التسلسلي.

الكلمات الدلالية: DSP، المتحكم PID العصبي، RS232، واجهة رسومية.

## Résumé

Le travail réalisé consiste à l'étude théorique et pratique de la commande PID neuronale. A ce fait nous avons la commande de deux systèmes de complexité différente. Le premier système est un réacteur chimique parfaitement mélange et le deuxième est un moteur à courant continu. Nous utilisons le logiciel MatLab pour faire plusieurs simulations et nous comparons les résultats obtenus à ceux de la commande PID classique. La partie pratique consiste à implémenter l'algorithme de commande PID neuronal sur DSP, réaliser les différents circuits électrique pour asservir la vitesse d'un moteur à courant continu et développer une interface graphique pour piloter l'application à partir d'un ordinateur communicant à travers une liaison série RS32 avec la carte de développement eZdsp F2812. Une fois l'application est mise en marche nous effectuons plusieurs essais.

**Mots clés : DSP, PID neuronale, rs232, interface graphique.**

## Abstract

This work consists of the theoretical study, the simulation of neural PID control of a nonlinear chemical system, simulation, design and implementation of neural PID control of a DC motor. In the first part of this project we described the conventional control, identification and neural control systems ending with the demonstration of neural PID control. In the second part we did some simulations then we spent in the realization and implementation of the project, we conducted a RS232 serial communication between the control computer and DSP, and finally we have programmed a graphic interface using C# to manage this serial connection.

**Keywords: DSP, neural PID, RS232, graphical interface.**

# Dédicace

*Je dédie ce mémoire à mes parents, ma mère qui m'a donné le soutien, le support dans toute ma vie, mon père qui m'a guidé toujours vers la réussite et m'a donné toute l'aide que je bescin, je vous remercie Du fond de mon cœur.*

*Mes frères et sœurs Qui m'ont beaucoup aidé tout au long de ma vie, Que dieu vous assiste.*

*Mon binôme Ali ce qui souvent fatigué Sous ce travail.*

*Mes amis qui m'ont perturbé, mais ils me prenaient à travailler sans pression.*

*Mes professeurs de l'université BLIDA1, de l'école ENSI, du technique MALAK BENNABJ, du GEM HAFSA BENFELHADJ et de l'école primaire RAHMANNI ABDELKADER.*

## ***DEDICACE***

Je dédie ce travail à :

Mes parents : pour tous les sacrifices qu'ils ont consacrés pour mes éducations, mes études et mon égard.

Tous les membres de ma famille : je vous souhaite plein de succès et beaucoup de bonheur dans votre vie.

Particulièrement, Mes frères et sœurs qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

Mes enseignants : veuillez trouver dans ce travail l'expression de ma profonde reconnaissance et ma grande estime.

Mon binôme Mohamed, Je ne peux trouver les mots justes et sincères pour t'exprimer mon affection et mes pensées, tu es pour moi un frère et un ami sur qui je peux compter.

Finalement, tous mes amis et à tous personnes qui m'a porté d'aide d'une manière directe ou soit elle indirecte, le long de ma vie.

***DEHEBI ALI***

# Remerciements

Tous les remerciements reviennent à Dieu qui nous a guidé et muni de patience et de courage nécessaire pour accomplir ce projet.

Nous tenons ensuite à remercier fortement nos encadreurs Mr. K.KARA et Mr. O.AITSAHED pour leurs précieux conseils, aide et soutien qui nous ont donné courage et motivation à avancer jusqu'au bout du chemin. Nous sommes très reconnaissants pour leur présence à nos côtés aux moments les plus durs.

Nous exprimons notre gratitude au staff du pavillon 15 qui nous ont beaucoup aidé et soutenu pour l'ambiance et le confort qu'ils nous ont apporté pendant toute la durée de ce travail.

Nous tenons aussi à exprimer notre gratitude au staff du département d'électronique qui nous a aidé et soutenu le long de notre cursus. Une aide matérielle qui est loin d'être négligée.

N'oublions surtout pas de remercier nos parents pour leur amour, leur soutien moral et matériel et leur encouragement qui nous ont toujours donné une confiance en soi et une motivation à aller de l'avant, sans oublier de remercier nos amis pour leur présence et leur sincère amitié.

Nous espérons que ce travail constituera un pas vers un long parcours plein d'objectifs et de succès.

Enfin prions Dieu de préserver notre pays en sécurité de par ses nombreux. Puisque Dieu a gardé l'unité du peuple algérien. Soyons vigilants et gardons notre fraternité, comme dit le proverbe algérien : « khouk khouk la yaghorek sahbek »

# Liste des figures & tableaux

## Chapitre 01

### COMMANDE DES SYSTEMES

Figure 1.1	Relation entre les caractéristiques de performances d'un système asservi	03
Figure 1.2	Correction en boucle ouverte	04
Figure 1.3	Correction en boucle fermée	05
Figure 1.4	Méthode de la réponse indicielle	08
Figure 1.5	Méthode du point critique	09
Tableau 1.1	Calcul des paramètres des régulateur P, PI ou PID en utilisant la méthode de la réponse indicielle	09
Tableau 1.2	Calcul des paramètres des régulateur P, PI ou PID en utilisant la méthode du point critique	10
Tableau. 1.3	Expression des différents paramètres du PID numérique pour les différentes méthodes d'approximation de l'intégral	12

## Chapitre 02

### IDENTIFICATION ET COMMANDE NEURONALE DES SYSTEMES

Figure 2.1	Topologie d'un neurone formel	14
Figure 2.2	Architecture d'un réseaux MLP à deux couches cachées	17
Figure 2.3	Modèle du neurone j	18
Figure 2.4	Identification direct d'un processus	22
Figure 2.5	Identification direct d'un processus en utilisant les entrées et sorties retardées	23
Figure 2.6	Identification inverse d'un processus	23
Figure 2.7	Principe de la commande directe par modèle inverse	24
Figure 2.8	Commande neuronale à modèle de référence	25
Figure 2.9	Commande neuronale à modèle de référence avec un émulateur	25
Figure 2.10	L'architecture de l'émulateur du processus proposé	26
Figure 2.11	Architecture d'identification du régulateur	28

Figure 2.12	Architecture d'apprentissage en parallèle avec le régulateur	28
Figure 2.13	Architecture d'auto-ajustement des paramètres du PID (PID neuronale)	29
Figure 2.14	Structure du réseau de neurones utilisé	29
Tableau 2.1	Les fonctions d'activation	15

## Chapitre 03

### SIMULATIONS ET RESULTATS

Figure 3.1	Réacteur continu parfaitement mélangé	34
Figure 3.2	Commande du CSTR en utilisant un PID classique	35
Figure 3.3	Sortie du CSTR, la référence et la commande délivrée par le PID	35
Figure 3.4	Ecart entre la sortie réelle du CSTR et la référence	36
Figure 3.5	Sortie du CSTR avec une perturbation de 10% additive à sa sortie, la référence et la commande délivré par le PID	37
Figure 3.6	Ecart entre la sortie réelle du système perturbé dans sa sortie et la référence	37
Figure 3.7	Sortie du CSTR avec une perturbation de 5% dans son entrée, la référence et la commande délivrée par le PID	38
Figure 3.8	Ecart entre la sortie réelle du système perturbé dans son entrée et la référence	39
Figure 3.9	Neuro-PID raccordé avec le CSTR en boucle fermée	39
Figure 3.10	Organigramme de l'opération d'apprentissage de l'émulateur	40
Figure 3.11	Organigramme de l'opération de commande	41
Figure 3.12	Réponse du système et celle de l'émulateur pour une entrée aléatoire	42
Figure 3.13	Réponse du système et celle de l'émulateur pour une entrée aléatoire avec un régime statique	42
Figure 3.14	Sortie du CSTR la référence et la commande délivré par le neuro-PID	43
Figure 3.15	Ecart entre la sortie réelle du CSTR et la référence	44
Figure 3.16	Sortie du CSTR avec une perturbation de 10%, la référence et le signal de commande	45
Figure 3.17	Ecart entre la sortie réelle du système perturbé et la référence	45
Figure 3.18	Sortie du CSTR avec une perturbation de 5% additive en entrée, la référence et le signal de commande	46
Figure 3.19	Ecart entre la sortie réelle du système perturbé et la référence	47

Figure 3.19	Ecart entre la sortie réelle du système perturbé et la référence	48
Figure 3.21	Zoom du tracé des résultats obtenus avec une perturbation additive en sortie	48
Figure 3.22	Zoom du tracé des résultats obtenu avec une perturbation additive en entrée	49
Figure 3.23	Moteur à courant continu	50
Figure 3.24	Schéma électrique simplifié d'un moteur CC en charge	50
Figure 3.25	Commande PID du MCC	53
Figure 3.26	Réponse indicielle du MCC en boucle ouverte	53
Figure 3.27	Vitesse de rotation du MCC la référence et la commande délivré par le PID	54
Figure 3.28	Ecart entre la sortie réelle du MCC et la référence	54
Figure 3.29	Vitesse de rotation du MCC, la référence et la commande délivrée par le PID (cas avec perturbation)	55
Figure 3.30	Ecart entre la sortie réelle du MCC et la référence (cas avec perturbation)	56
Figure 3.31	Commande du MCC en utilisant un PID neuronal	56
Figure 3.32	Réponse du système et celle de l'émulateur pour une entrée aléatoire	57
Figure 3.33	Réponse du système et celle de l'émulateur pour une entrée aléatoire avec un régime statique	58
Figure 3.34	Vitesse de rotation du MCC, trajectoire de référence et la commande délivrée par le PID neuronal	59
Figure 3.35	Ecart entre la sortie réelle du MCC et la référence	59
Figure 3.36	Vitesse de rotation du MCC, trajectoire de référence et signal de commande (cas avec perturbation)	60
Figure 3.37	Ecart entre la sortie réelle du MCC et la référence (cas avec perturbation)	60
Figure 3.38	Zoom du tracé des résultats des PID et PID neuronal (sans perturbation)	61
Figure 3.39	Zoom du tracé des résultats des PID et PID neuronal (avec perturbation)	62

## **Chapitre 04**

### **REALISATION ET ESSAIS PRATIQUES**

Figure 4.1	Schéma synoptique du dispositif réalisé	64
Figure 4.2	Schéma structurel d'un hacheur à quatre quadrants	65
Figure 4.3	Signal PWM et son complément avec une zone morte	65
Figure 4.4	Modes de fonctionnement dans les quatre quadrants	66
Figure 4.5	Schéma électrique du hacheur réalisé	67
Figure 4.6	Schéma fonctionnel de la liaison série RS232	68



Figure 4.7	Liaison simplex entre un émetteur et un récepteur	68
Figure 4.8	Liaison semi-duplex entre deux systèmes	68
Figure 4.9	Liaison duplex intégral entre deux systèmes	69
Figure 4.10	Schéma électrique du circuit d'adaptation	70
Figure 4.11	Schéma électrique de l'alimentation stabilisé 3.3V	70
Figure 4.12	Schéma électrique du circuit d'interface et de mise en forme (entrées CAP du DSP)	71
Figure 4.13	Schéma électrique reliant toutes les sorties PWM du DSP à travers les AM26LS31	72
Figure 4.14	Schéma électrique de l'alimentation stabilisé +5V	73
Figure 4.15	Schéma électrique du circuit d'isolation	74
Figure 4.16	Schéma électrique de l'alimentation stabilisée réglable	75
Figure 4.17	Schéma électrique du circuit de puissance	76
Figure 4.18	Schéma électrique des circuits d'interface et de communication série	77
Figure 4.19	Réalisation finale du projet	78
Figure 4.20	Interface graphique	79
Figure 4.21	Tracé de la sortie du MCC en utilisant le PID classique	80
Figure 4.22	Tracé de la sortie du MCC en utilisant le PID neuronal	80
Figure 4.23	Tracé de la sortie du MCC en utilisant le PID et le PID neuronal	81
Figure 4.24	Tracé de la sortie du MCC en présence d'une perturbation en utilisant le PID neuronal	82
Figure 4.25	Début de la perturbation de 10% à la sortie	83
Figure 4.26	Fin de la perturbation de 10% à la sortie	83
Figure 4.27	Début de la perturbation de 50% à la sortie	84
Figure 4.28	Fin de la perturbation de 50% à la sortie	85
Figure 4.29	Sortie du nouveau système (rhéostat en série avec le MCC)	86

# Sommaire

Notations & symboles

Introduction générale 01

## Chapitre 01

### COMMANDE DES SYSTEMES

1.1	Introduction	03
1.2	Correction des systèmes asservis	03
1.2.1	Correction en boucle ouverte	04
1.2.2	Correction en boucle fermée	04
1.2.2.1	Correcteur proportionnel (P)	05
1.2.2.2	Correcteur proportionnel intégrateur (PI)	06
1.2.2.3	Correcteur proportionnel dérivateur (PD)	06
1.2.2.4	Correcteur proportionnel intégral dérivé (PID)	07
1.2.3	Méthodes de Ziegler et Nichols (ZN)	08
1.2.3.1	Méthode de la réponse indicielle	08
1.2.3.2	Méthode du point critique	09
1.2.4	Le PID numérique	10
1.3	Conclusion	12

## Chapitre 02

### IDENTIFICATION ET COMMANDE NEURONALE DES SYSTEMES

2.1	Introduction	13
2.2	Historique	13
2.3	Neurone formel	14
2.4	Réseaux de neurones	16
2.4.1	Types des réseaux de neurones	16
2.4.1.1	Réseaux statiques	16
2.4.1.2	Réseaux dynamiques	16

2.4.2	Réseaux de neurones multicouches (MLP)	17
2.4.2.1	Algorithme de la retro-propagation du gradient	17
2.5	Identification neuronale d'un système dynamique	22
2.6	Commande neuronale d'un système dynamique	23
2.6.1	Commande directe par model inverse	24
2.6.2	Commande neuronale à modèle de référence	24
2.6.2.1	Détermination du jacobien	26
2.6.3	Commande basé sur l'apprentissage d'un régulateur conventionnel	27
2.6.3.1	Identification du régulateur	27
2.6.3.2	Apprentissage en parallèle avec le régulateur	28
2.6.3.3	Auto-ajustement des paramètres du régulateur	28
2.6.4	PID neuronale	29
2.7	Conclusion	32

## **Chapitre 03**

### **SIMULATIONS ET RESULTATS**

3.1	Introduction	33
3.2	Presentation du CSTR	33
3.2.1	Commande du CSTR en utilisant un régulateur PID classique	34
3.2.1.1	Simulation sans perturbations	35
3.2.1.2	Simulation avec perturbation de 10 % additive en sortie	36
3.2.1.3	Simulation avec perturbation de 5 % additive à l'entrée	38
3.2.2	Commande du CSTR en utilisant un régulateur PID neuronal	39
3.2.2.1	Simulation sans perturbations	43
3.2.2.2	Simulation avec une perturbation de 10 % additive en sortie	44
3.2.2.3	Simulation avec une perturbation de 5 % additive en entrée	46
3.2.3	Comparaisons entre les résultats du PID classique et ceux du PID neuronal	47
3.3	Moteur à courant continu	49
3.3.1	Modélisation du MCC	50
3.3.2	Commande du moteur en utilisant un PID classique	52
3.3.2.1	Simulation sans perturbations	54
3.3.2.2	Simulation avec perturbation additive en sortie	55
		X

3.3.3	Commande du MCC en utilisant un PID neuronal	56
3.3.3.1	Simulation sans perturbations	58
3.3.3.2	Simulation avec perturbation additive en sortie	59
3.3.3.3	Comparaisons entre les résultats du PID classique et ceux du PID neuronal	61
3.4	Conclusion	62

## **Chapitre 04**

### **REALISATION ET ESSAIS PRATIQUE**

4.1	Introduction	63
4.2	Synoptique de l'application	63
4.3	Hacheurs	64
4.3.1	Hacheur à quatre quadrants	65
4.3.2	Réalisation du hacheur a quatre quadrants	66
4.4	Communication série (RS232)	67
4.4.1	Les niveaux de tension imposés	68
4.4.2	Les modes de transmission RS232	68
4.4.3	Le circuit d'adaptation	69
4.5	Entrées de l'unité de capture du DSP (CAP)	71
4.6	Sorties PWM	72
4.7	Circuit d'isolation	73
4.8	Alimentation stabilisé réglable	74
4.9	Circuit électrique global	75
4.10	Interface graphique	78
4.11	Résultats expérimentaux	80
4.11.1	Premier essai	80
4.11.2	deuxième essai	81
4.11.3	Troisième essai	82
4.11.4	Quatrième essai	84
4.11.5	Cinquième essai	85
4.12	Conclusion	86
	Conclusion générale	87

## Notation & symbole

CSTR	Continuous Stirred Tank Reactor.
P	Correcteur proportionnel.
PI	Correcteur proportionnel intégrateur.
PD	Correcteur proportionnel dérivateur.
PID	Correcteur proportionnel, intégrateur et dérivateur.
$k_p$	Gain proportionnel.
$k_i$	Constante d'intégration.
$T_d$	Constante de dérivation.
$r(t)$	La référence.
$y(t)$	Sortie du processus.
$T_u$	Temps de retard.
$K_u$	gain.
$T_a$	Le temps de montée de la tangente.
$\Delta_y$	La différence entre l'amplitude initiale et l'amplitude finale de la sortie du processus
$\Delta_u$	La différence entre l'amplitude initiale et l'amplitude finale de l'échelon.
$T_0$	Période d'échantillonnage.
FRM	Forward Rectangular Method.
BRM	Backward Rectangular Method.
TRAP	Trapezoid method.
RNAs	Réseaux de neurones artificiels.
MLP	Réseaux de neurones a multicouches.
EQM	Erreur quadratique moyenne.
$y_d(k)$	La sortie désirée.
$y_{est}(k)$	La sortie estimée.
$e_a(n)$	Erreur entre la sortie du système et la sortie du modèle.
$v$	Le volume utilisé du réacteur est constant.
$C_{a0}$	La charge fraiche de concentration exprimée en mol/l.
$T_0$	La température initiale exprimée en Kelvin.
$T_c$	La température du jacket.
$T_{c0}$	La température initiale du jacket.
$C_a(t)$	La concentration du produit A exprimée en mol/l.

T(t)	La température du mélange supposée uniforme, exprimée en Kelvin.
$q_c(t)$	Le débit du liquide de refroidissement utilisé pour contrôler la température de la réaction, il est exprimé en l/min.
q	Le débit du processus exprimé en l/min.
MCC	Moteur a courant continu.
CC	Courant continu (DC : Direct current).
n	La vitesse de rotation du MCC (Tr/min).
U	La tension d'alimentation (V).
Z	Le nombre de conducteurs sur l'induit.
$\emptyset$	Flux par pôle (Wb).
CCS	Code Composer Studio.
DSP	Digital Signal Processor (Processeur de traitement numérique du signal).
PWM	Pulse Width Modulation.
ADC	Analog to Digital Converter (Convertisseur analogique/numerique).
CAP	Unité de capture du <i>DSP</i> .
MLI	Modulation de largeur d'impulsion.
MOS	Metal Oxide Semiconductor.
IGBT	Insulated Gap Bipolar Transistor.
$T_i$	Interrupteur, ( $i = 1,2,3,4$ ).
EIA	Electronic Industries Association.

## *Introduction générale*

L'évolution de la technologie a influencé tous les domaines de la vie (automatique, télécommunication, robotique, aéronautique, biomédicale, architecture, ... etc.), Parmi ces domaine on compte l'automatisme, Ce substantif a été utilisé pour la première fois en 1914 dans un article « Essai sur l'automatisme » publié dans une revue scientifique, ensuite ce domaine a connu plusieurs découvertes intéressantes mais malheureusement, elles ont été classées comme des théories jusqu'à la découverte des semi-conducteurs qui nous a conduit à réaliser des calculateurs numériques avec des propriétés très intéressantes (puissant, frugal, petite taille), Donc cette évolution nous a permis de faire la translation de ces théories vers la pratique mais les translations faites au début sont très simples à cause de la simplicité des calculateurs en ce temps la, ensuite on a constaté l'apparition des «microcontrôleurs» qui ont connu un grand succès dans plusieurs applications mais malgré leurs grand succès ces microcontrôleurs n'ont pas réussi à s'imposer devant certaines applications, qui nécessitent une fréquence de calcul élevée donc on se retrouvait face à face à un problème de rapidité. Ce qui a donné lieu à l'émergence des processeurs spécialisés tel que les DSPs (Digital Signal Processor) et les circuits reprogrammables tels que les FPGAs qu'ont permis de donner un intérêt important à la commande numérique des systèmes.

Les systèmes sont généralement de nature non linéaire, ce qui nécessite des algorithmes de contrôle compliqués et non linéaires. Un de ces systèmes est le processus chimique CSTR qui possède des fortes non-linéarités à cause de ces réactions chimiques, l'objectif de cette étude est de contrôler ce système en utilisant un des algorithmes développés en automatisme (neuro-PID). Pour faire l'implémentation de l'algorithme neuro-PID on va utiliser le DSP TMS320F2812 de Texas Instrument pour contrôler un moteur à courant continu comme à default du CSTR introuvable à l'université. Donc dans la pratique on va faire la commande de vitesse d'un moteur en utilisant un régulateur neuro-PID.

On peut effectuer beaucoup de calculs très complexes en un minimum de temps en utilisant un processeur DSP, et d'autre part on peut aussi accéder facilement à un nombre élevé d'entrées / sorties. Les unités de calcul utilisées avec la plupart des DSP sont en virgule fixe. Le prix de ce composant est abordable vu l'absence d'unité arithmétique en virgule flottante permettant aussi une grande vitesse de traitement des données. Un multiplieur entier est donc plus simple qu'un multiplieur à virgule flottante. Bien que certaines DSP possèdent des unités

de calcul en virgule flottante, une précision suffisante est obtenue pour la plupart des applications avec les nombres entiers.

En utilisant une commande neuro-PID, on peut garder un niveau de performances adéquat indépendamment des différentes perturbations (erreur statique nulle, rapidité, stabilité). On peut adapter un régulateur auto-ajustable en temps réel aux différentes perturbations observées dans le système d'une manière totalement automatique. Cette auto-adaptation est très demandée en industrie. On trouve plusieurs méthodes de synthèse d'un régulateur adaptatif, qui offrent des performances et une complexité très variables dans la littérature actuelle.

Parmi tout cela, le régulateur PID reste l'algorithme de commande le plus simple et le plus utilisé en industrie, il permet des performances qu'on peut offrir aux systèmes en boucle fermée, et satisfait très souvent les cahiers de charge, lorsque les paramètres sont choisis correctement. De notre côté, nous avons opté pour l'implémentation de la commande neuro-PID. Le but d'utiliser un algorithme implémenté (neuro-PID) est de minimiser une fonction de coût en utilisant l'algorithme de rétro-propagation du gradient.

Dans ce mémoire le travail présenté est structuré comme suit :

Le premier chapitre est consacré à la commande classique en exploitant le contrôle en boucle ouverte et en boucle fermée et quelques contrôleurs utilisés (proportionnel, proportionnel intégrale, proportionnel dérivé, proportionnelle intégrale dérivé).

Dans le deuxième chapitre, on a présenté les réseaux de neurones artificiels en commençant par leurs principes de fonctionnement ensuite l'algorithme d'apprentissage utilisé et les différentes méthodes de la commande et d'identification des systèmes en utilisant les RNAs en finalisant par la commande neuro-PID.

Le troisième chapitre est destiné à une présentation brève sur les systèmes utilisés (CSTR et MCC) et les résultats de simulation en considérant plusieurs cas (sans perturbations avec perturbations).

Le quatrième chapitre est consacré pour citer les différents schémas électriques et présenter brièvement quelques pièces utilisées et expliquer l'interface graphique programmée pour contrôler et superviser le MCC et à la fin du chapitre on a effectué quelques expériences pour montrer les résultats obtenus.



# **Chapitre 1**

**Commande des systèmes**

## 1.1 Introduction

Il existe plusieurs méthodes de commande des systèmes, on peut les classer dans deux majeures sections, les méthodes classiques comme la méthode de placement des pôles, le régulateur PID, la méthode des valeurs propres ... etc et les méthodes basées sur les outils de l'intelligence artificielle, comme la commande neuronale, les régulateurs flous, la commande neuro-floue, ... etc. Dans l'industrie, on préfère toujours implémenter des régulateurs simples tels que le régulateur proportionnel intégral dérivé (PID). En effet, le PID donne des résultats acceptables et il est facile à exploiter.

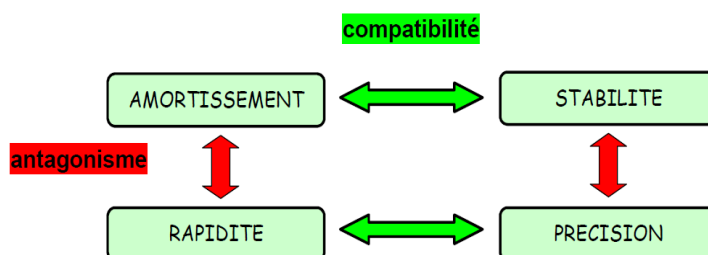
Le but de ce chapitre est de rappeler brièvement la notion de correction de systèmes et les différents correcteurs classiques utilisés dans la pratique.

## 1.2 Correction des systèmes asservis

Le comportement d'un système asservi est défini à partir de 4 caractéristiques principales [10]:

- La rapidité : le temps mis par la sortie pour atteindre la valeur de consigne.
- La stabilité : l'aptitude d'avoir une sortie bornée pour une entrée bornée.
- La précision statique : aptitude d'avoir une erreur statique nulle entre la sortie et la référence.
- L'amortissement : aptitude d'avoir une réponse dans le régime transitoire sans un dépassement excessif (entre 15 et 25 %).

L'objectif est de trouver le correcteur adéquat qui permet de satisfaire aux critères de performances exigés, dans le cahier des charges, pour le système en boucle fermée. En général, il n'est pas possible d'obtenir toutes ces caractéristiques en même temps et un compromis doit être fait. Par exemple (Fig.1.1), l'augmentation du gain en boucle ouvert permet d'augmenter la précision et la rapidité mais il cause une diminution de la marge de stabilité et peut engendrer de grands dépassements dans la sortie du processus.

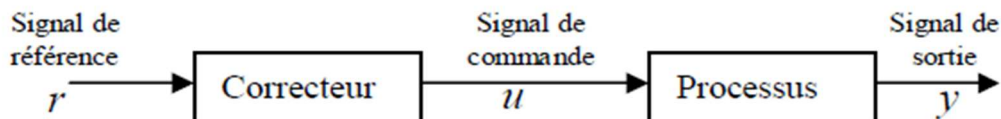


**Figure 1.1** : Relation entre les caractéristiques de performances d'un système asservi.

Il existe deux approches de correction : la première est basée seulement sur la valeur désirée à la sortie du processus (commande en boucle ouverte) tandis que la deuxième est basée sur la valeur désirée et sur la sortie du système réellement obtenue (commande en boucle fermée).

### 1.2.1 Correction en boucle ouverte

Dans ce cas (Fig. 1.2), la grandeur de commande  $u$  ne dépend que du signal de référence  $r$ , et le correcteur ne reçoit aucune information sur la sortie réelle du système. Ce cas serait intéressant, si on sait qu'une commande quelconque appliquée au système va toujours engendrer la même réponse indépendamment du fait que les perturbations sont présentes ou non. Un tel correcteur sera simple à concevoir avec un coût faible, c'est l'avantage majeur de cette approche.



**Figure 1.2 :** Correction en boucle ouverte.

La fonction de transfert de ce type de correction est donnée par :

$$F(p) = \frac{Y(p)}{R(p)} = C(p) \cdot G(p) \quad (1.1)$$

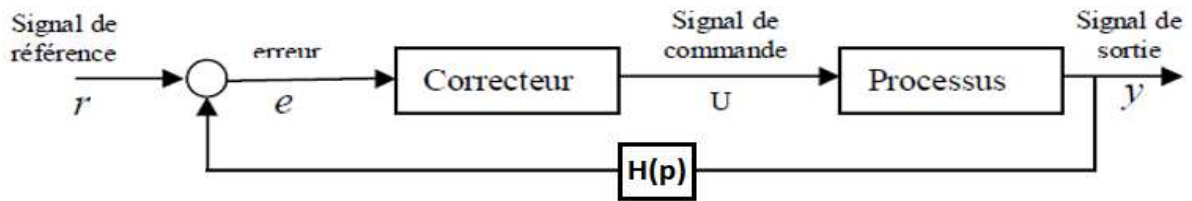
Avec :

$$C(p) = \frac{U(p)}{R(p)} : \text{La fonction de transfert du correcteur}$$

$$G(p) = \frac{Y(p)}{U(p)} : \text{La fonction de transfert du processus}$$

### 1.2.2 Correction en boucle fermée

Dans ce cas (Fig. 1.3), la sortie du processus est comparée à la référence, et selon le résultat obtenu, le correcteur va agir sur le processus. Ce mode de correction est le plus robuste vis-à-vis des perturbations.



**Figure 1.3 :** Correction en boucle fermée.

La fonction de transfert de ce type de correction est donnée par :

$$F(p) = \frac{Y(p)}{R(p)} = \frac{G(p).C(p)}{(1 + G(p).C(p).H(p))} \quad (1.2)$$

Le correcteur peut être placé, dans la chaîne directe, en série avec le système, ou en parallèle, dans la chaîne de retour principale ou une chaîne de retour secondaire. Il existe plusieurs types de correcteurs, les plus populaires sont :

### 1.2.2.1 Correcteur proportionnel (P)

Le correcteur proportionnel est le plus simple des correcteurs. On peut l'utiliser si le cahier des charges n'est pas trop contraignant ou si le système a un comportement assez simple.

Le correcteur à action proportionnelle délivre une commande de la forme :

$$u(t) = k_p e(t) = K_p [r(t) - y(t)] \quad (1.3)$$

Tel que :

- $K_p$  est le gain proportionnel.
- $r(t)$  représente le signal de référence.
- $y(t)$  est la sortie du processus.

La fonction de transfert du correcteur est donnée par :

$$C(p) = K_p \quad (1.4)$$

L'un des avantages de ce correcteur est son coût économique. Son inconvénient est la présence d'une erreur en régime statique.

Expérimentalement, son réglage peut se faire directement en partant d'un gain faible et en augmentant petit à petit jusqu'à atteindre un comportement satisfaisant. Pour un gain trop

élevé, le système deviendra instable ce qui se manifestera d'abord par des oscillations de plus en plus importantes.

### 1.2.2.2 Correcteur proportionnel intégrateur (PI)

Ce correcteur contient une action de type P auquel on a ajouté un terme intégral, il élabore alors une commande qui peut être donnée par la relation suivante :

$$u(t) = K_p e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau \quad (1.5)$$

Où  $T_I$  est la constante d'intégration.

On remarque que cette action dépend de toutes les valeurs précédentes de l'erreur. La fonction de transfert du correcteur est de la forme :

$$C(p) = K_p + \frac{1}{T_I p} \quad (1.6)$$

La présence d'un pôle dans l'origine du plan complexe permet d'éliminer l'erreur en régime statique quand le signal de référence est un échelon [11].

Le problème de l'action intégrale est l'apparition de saturation de la commande. Si la commande de l'actionneur est limitée à une valeur maximale (qui est presque toujours le cas), la sortie du correcteur peut se saturer. Le terme intégral peut devenir de plus en plus grand, et la commande se stabilise autour de la valeur de la saturation. Même si la référence a changé, la sortie du correcteur reste saturée. Cette saturation va dégrader les performances du système. Un dépassement important ainsi qu'un temps de réponse très lent vont se manifester. Donc, lors de l'implémentation de cette action, il faut prendre en considération ce problème. La méthode la plus simple pour y remédier, est l'annulation du terme intégrateur de la commande lors de la saturation, et sa réinsertion en dehors de cette zone.

### 1.2.2.3 Correcteur proportionnel dérivateur (PD)

Le correcteur proportionnel dérivateur est basé sur la prédiction de l'erreur du système dans le futur. L'action dérivation est donnée par :

$$u(t) = T_D \frac{de(t)}{dt} \quad (1.7)$$

Où  $T_D$  est la constante de dérivation.

Sa fonction de transfert est la suivante :

$$C(p) = T_D \cdot p \quad (1.8)$$

Pour bien comprendre le rôle de l'action dérivateur, on applique le développement de *Taylor* de l'équation de la commande précédente, alors on obtient :

$$e(t + T_D) \approx e(t) + T_D \frac{de(t)}{dt} \quad (1.9)$$

En ajoutant l'action proportionnelle à l'équation (1.7), donc la commande d'un régulateur PD est donnée par la relation suivante :

$$u(t) = K_p \left( e(t) + T_D \frac{de(t)}{dt} \right) \quad (1.10)$$

Cette loi représente un correcteur PD, la commande à l'instant  $t$  est calculée à partir de la prédiction de l'erreur à l'instant  $(t+T_D)$ , pour cette raison, l'action dérivateur est aussi appelée *Commande anticipée*.

L'action dérivation a un grand potentiel pour améliorer les performances du système, elle anticipe la tendance de l'erreur et essaie de la corriger. Mais son implémentation pose quelques problèmes, il faut bien filtrer la sortie du processus pour éviter les composantes hautes fréquences (bruit), qui ont tendance à emmener le terme dérivateur de la commande à l'infini.

#### 1.2.2.4 Correcteur proportionnel intégral dérivé (PID)

Ce correcteur contient les trois actions à la fois, il est le plus utilisé dans l'industrie. La loi de commande est donnée par :

$$u(t) = K_p e(t) + \frac{1}{T_I} \int_0^t e(\tau) d(\tau) + T_D \frac{de(t)}{dt} \quad (1.11)$$

Sa fonction de transfert a la forme suivante :

$$C(s) = K \left[ 1 + \frac{1}{T_I \cdot s} + T_D \cdot s \right] \quad (1.12)$$

Le PID classique a des limites malgré ses bonnes performances. Les performances du système seront dégradées lorsque les perturbations varient d'une manière importante.

Il existe plusieurs méthodes pour déterminer les paramètres des régulateurs P, PI, et PID. Nous présentons, dans le paragraphe suivant la méthode de Ziegler et Nichols.

### 1.2.3 Méthodes de Ziegler et Nichols (ZN)

En 1942, Ziegler et Nichols [1] ont proposé deux méthodes basées sur leurs expériences et simulations pour calculer les paramètres des régulateurs P, PI et PID. La première méthode nécessite la réponse indicielle du processus en boucle ouverte et la deuxième pousse le processus en boucle fermée à sa limite de stabilité.

#### 1.2.3.1 Méthode de la réponse indicielle

Dans cette méthode il suffit d'injecter un signal échelon dans l'entrée du processus en boucle ouverte et de récupérer sa réponse indicielle. Ensuite, on trace la tangente au point d'inflexion de la courbe et on mesure le temps de retard  $T_u$ , le temps de montée de la tangente  $T_a$ , la différence entre l'amplitude initiale et l'amplitude finale de la sortie du processus  $\Delta_y$ , et la différence entre l'amplitude initiale et l'amplitude finale de l'échelon  $\Delta_u$  (Fig. 1.4)

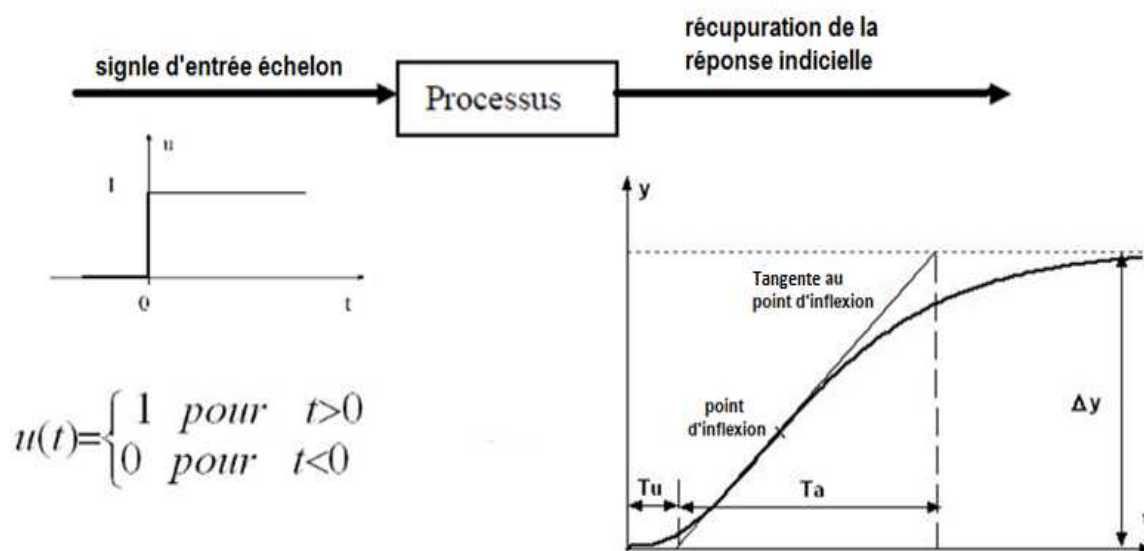


Figure 1.4 : Méthode de la réponse indicielle.

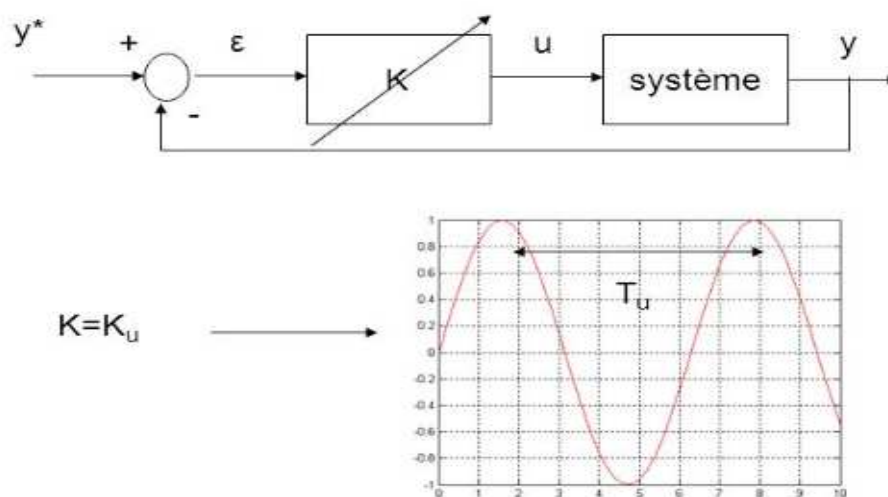
Ziegler et Nichols ont établi le tableau suivant pour calculer les paramètres des régulateur P, PI ou PID à partir de ces mesures ( $T_u$ ,  $T_a$ ,  $\Delta_y$ ,  $\Delta_u$ ).

Régulateur	$K_p$	$T_I$	$T_D$
<b>P</b>	$\frac{T_a}{T_u} \cdot \frac{1}{K}$	/	/
<b>PI</b>	$\frac{T_a}{T_u} \cdot \frac{0.9}{K}$	$3.33T_u$	/
<b>PID</b>	$\frac{T_a}{T_u} \cdot \frac{1.2}{K}$	$2T_u$	$0.5T_u$

**Tableau 1.1 :** Calcul des paramètres des régulateur P, PI ou PID en utilisant la méthode de la réponse indicielle ( $K = \frac{\Delta y}{\Delta u}$ )

### 1.2.3.2 Méthode du point critique

On boucle le processus sur un simple régulateur proportionnel (Fig. 1.5). On augmente le gain du correcteur jusqu'à amener le système à osciller de manière permanente, ensuite on retient cette valeur du gain  $K_u$  et la période d'oscillation  $T_u$  et enfin on calcule les paramètres des régulateurs P, PI, PID à partir du tableau (1.2).



**Figure 1.5 :** Méthode du point critique.



Régulateur	$K_p$	$T_I$	$T_D$
<b>P</b>	$0.5K_u$	/	/
<b>PI</b>	$0.45K_u$	$0.85T_u$	/
<b>PID</b>	$0.6K_u$	$0.5T_u$	$0.12T_u$

**Tableau 1.2 :** Calcul des paramètres des régulateur P, PI ou PID en utilisant la méthode du point critique.

### 1.2.4 Le PID numérique

Pour implanter numériquement un correcteur, il faut discrétiser les résultats obtenus en continu pour avoir la version numérique de la fonction du transfert. Alors, on doit discrétiser le terme intégrateur et le terme dérivateur présents dans l'équation (1.12).

Si la période d'échantillonnage  $T_0$  est petite et le bruit dans le signal de sortie est bien filtré, on peut remplacer la dérivée par une différence du premier ordre :

$$\frac{de(t)}{dt} \approx \frac{e(k) - e(k - 1)}{T_0} = \frac{\Delta e(k)}{T_0} \quad (1.13)$$

Pour le terme intégral, il y'a trois approximations. La première consiste à remplacer ce terme par une somme des rectangles inférieurs (FRM : Forward Rectangular Method), cela va donner :

$$\int_0^t e(\tau).d(\tau) \approx T_0 \sum_{i=1}^k e(i - 1) \quad (1.14)$$

L'équation du PID discrétisée est alors donnée par :

$$u(k) = K \left\{ e(k) + \frac{T_0}{T_I} \sum_{i=1}^k e(i - 1) + \frac{T_D}{T_0} [e(k) - e(k - 1)] \right\} \quad (1.15)$$

Si le terme intégral est remplacé par une somme des rectangles supérieurs (BRM: Backward Rectangular Method) :

$$\int_0^t e(\tau).d(\tau) \approx T_0 \sum_{i=1}^k e(i) \quad (1.16)$$

L'équation du PID discrétisée est alors donnée par :

$$u(k) = K \left\{ e(k) + \frac{T_0}{T_I} \sum_{i=1}^k e(i) + \frac{T_D}{T_0} [e(k) - e(k - 1)] \right\} \quad (1.17)$$

La dernière approximation utilise la méthode des trapézoïdes (TRAP : Trapezoid method), elle est la plus précise des trois. Le terme intégral est donné par :

$$\int_0^t e(\tau).d(\tau) \approx T_0 \sum_{i=1}^k \frac{e(i) + e(i - 1)}{2} \quad (1.18)$$

L'équation du PID discrétisée sera :

$$u(k) = K \left\{ e(k) + \frac{T_0}{T_I} \left[ \frac{e(0) + e(k)}{2} + \sum_{i=1}^{k-1} e(i) \right] + \frac{T_D}{T_0} [e(k) - e(k - 1)] \right\} \quad (1.19)$$

Si la période d'échantillonnage est suffisamment petite, on ne remarque pas de différences (apparentes) entre les trois équations (1.15, 1.17 et 1.19). L'équation (1.19) est la plus utilisée en pratique. Les trois équations (1.15, 1.17 et 1.19) sont de nature non-récurrente, il faut connaître toutes les valeurs de l'erreur depuis le début de l'opération de contrôle, ce qui n'est pas pratique. C'est pour cette raison qu'on utilise les versions récursives de ces trois équations.

Pour l'équation (1.17), sa version récursive est donnée par:

$$u(k) = u(k - 1) + K \left\{ e(k) - e(k - 1) + \frac{T_0}{T_I} e(k) + \frac{T_D}{T_0} [e(k) - 2e(k - 1) + e(k - 2)] \right\} \quad (1.20)$$

Ou sous la forme suivante :

$$u(k) = u(k - 1) + K_p e(k) + K_D [e(k) - e(k - 1)] + K_I [e(k) - 2e(k - 1) + e(k - 2)] \quad (1.21)$$

Où :  $K_I = K \cdot \frac{T_D}{T_0}$  ,  $K_p = K \cdot \frac{T_0}{T_I}$  et  $K_D = K$

Ou encore sous la forme générale suivante :

$$u(k) = u(k - 1) + q_0 e(k) + q_1 e(k - 1) + q_2 e(k - 2) \quad (1.22)$$

Les paramètres  $q_0, q_1$  et  $q_2$  sont donnés dans le tableau suivant :

Paramètres du correcteur	FRM	BRM	TRAP
$q_0$	$K_p \left(1 + \frac{T_D}{T_0}\right)$	$K_p \left(1 + \frac{T_0}{T_I} + \frac{T_D}{T_0}\right)$	$K_p \left(1 + \frac{T_0}{2T_I} + \frac{T_D}{T_0}\right)$
$q_1$	$-K_p \left(1 - \frac{T_0}{T_I} + \frac{2T_D}{T_0}\right)$	$-K_p \left(1 + \frac{2T_D}{T_0}\right)$	$-K_p \left(1 - \frac{T_0}{2T_I} + \frac{2T_D}{T_0}\right)$
$q_2$	$K_p \frac{T_D}{T_0}$	$K_p \frac{T_D}{T_0}$	$K_p \frac{T_D}{T_0}$

**Tableau. 1.3** : Expression des différents paramètres du PID numérique pour les différentes méthodes d'approximation de l'intégral.

### 1.3 Conclusion

Dans ce chapitre, nous avons introduit la notion de correction des systèmes asservis ainsi que les différents correcteurs classiques. Il n'existe pas une méthode générale qui permet de déterminer les valeurs optimales des paramètres de trois correcteurs présentés. La méthode de Ziegler et Nichols ne permet pas d'obtenir les meilleures valeurs des paramètres et souvent un ajustement des valeurs obtenues est nécessaire. Nous avons aussi présenté la version numérique du régulateur PID. Cette dernière va servir dans la simulation est l'implémentation de la commande PID.

# Chapitre 2

Identification et commande neuronale des  
systèmes.

## 2.1 Introduction

Les outils de l'intelligence artificielle telle que les réseaux de neurones artificielles (RNAs), la logique floue, les algorithmes génétiques ont été largement utilisés dans l'identification et la commande des systèmes dynamiques. Les RNAs ont connu un développement très vaste depuis plusieurs années et ont été utilisés dans des applications variées. A cause des fonctions d'activation non linéaires utilisées, les RNAs peuvent modéliser n'importe quelle fonction linéaire ou non linéaire.

Nous présentons, dans ce chapitre, quelques notions de base relatives aux réseaux de neurones et à leur utilisation dans l'identification et la commande des systèmes. Plus particulièrement, après avoir rappelé les différentes méthodes de commande en utilisant les réseaux de neurones, nous donnons les différentes étapes de développement d'un PID neuronal.

## 2.2 Historique

Au début, en 1943, McCulloch et Pitts ont démontré que des réseaux de neurones formels ont la capacité de réaliser des fonctions logiques et arithmétiques, Hebb a continué ses recherches en introduisant une loi d'apprentissage « loi de Hebb ».

En 1957, Rosenblatt a développé le modèle du perceptron, il l'a utilisé dans le domaine de la reconnaissance de forme.

En 1960, Widrow a développé le modèle ADALIN (adaptive linear neuron), la seule différence entre cette structure et le perceptron est la loi d'apprentissage, c'est la base de l'algorithme de rétro-propagation du gradient qui est très utilisé aujourd'hui dans les MLPs (multi layer perceptron) et que nous allons utiliser dans notre projet.

En 1969, Minsky et Papert ont montré l'incapacité de traiter des problèmes non linéaires en utilisant le modèle ADALIN.

En 1985, l'algorithme d'apprentissage adapté aux MLPs, la rétro-propagation du gradient, apparaît grâce à Rumelhart et McClelland. En utilisant les MLPs on peut traiter des problèmes non linéaires.

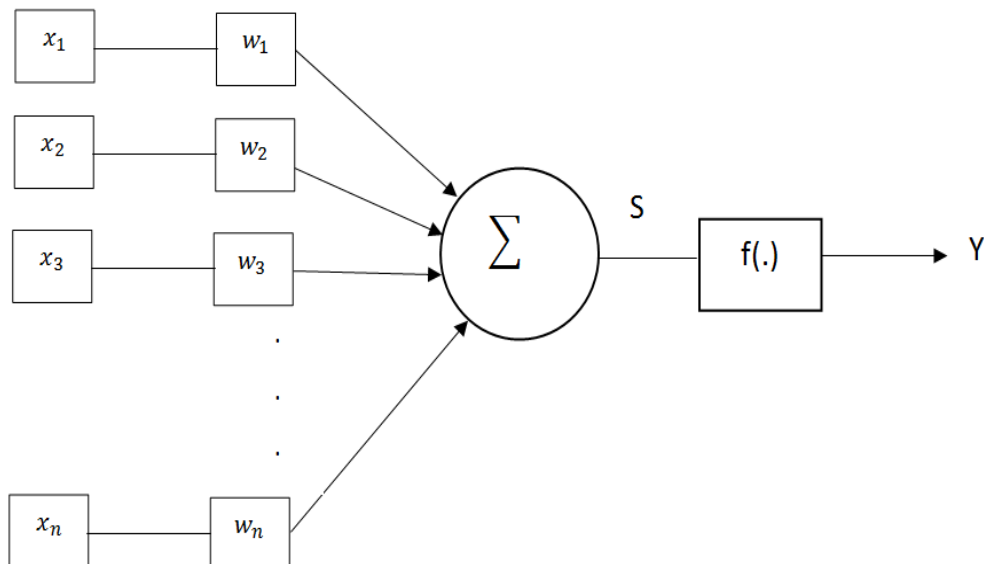
Après le développement de la rétro-propagation du gradient, plusieurs autres nouvelles structures et algorithmes d'apprentissage ont été proposés.

### 2.3 Neurone formel

Un neurone formel ou artificiel (Fig. 2.1) réalise une opération algébrique simple de la forme :

$$y = f(S) = f\left(\sum_{i=1}^n w_i * x_i - b\right) \quad (2.1)$$

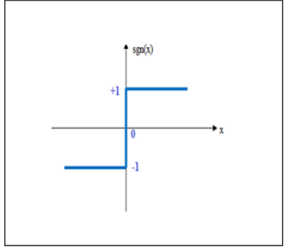
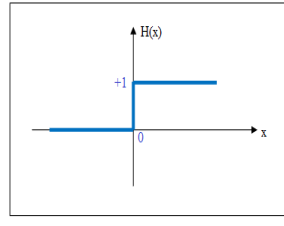
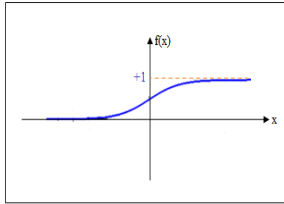
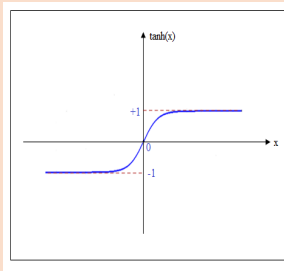
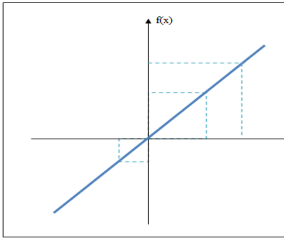
Où :  $y$  est la sortie du neurone,  $x_i$  sont les entrées du neurones,  $n$  est le nombre des entrées,  $w_i$  sont les poids de pondération synaptique,  $b$  est le biais (seuil d'activation) du neurone et  $f(.)$  est la fonction d'activation du neurone. Les entrées  $x_i$  peuvent être des entrées externes où des sorties des autres neurones.



**Figure 2.1.** Topologie d'un neurone formel

On distingue plusieurs types de fonctions d'activation, les plus utilisées sont (tableau 2.1) :

- La fonction signe.
- La fonction Heaviside.
- La fonction sigmoïde unipolaire.
- La fonction tangente hyperbolique (sigmoïde bipolaire).
- La fonction identité (pour la sortie du réseau).

<p><b>La fonction signe</b></p>	$\forall x \in \mathbf{R}, \text{sgn}(x) = \begin{cases} -1 & \text{si } x < 0 \\ 0 & \text{si } x = 0 \\ 1 & \text{si } x > 0 \end{cases}$	
<p><b>La fonction Heaviside</b></p>	$\forall x \in \mathbf{R}, H(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$	
<p><b>La fonction sigmoïde unipolaire</b></p>	$f(x) = \frac{1}{1 + e^{-\gamma x}}$	
<p><b>La fonction tangente hyperbolique (sigmoïde bipolaire)</b></p>	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
<p><b>La fonction identité</b></p>	$\forall x \in \mathbf{R}, \quad f(x) = x$	

**Tableau 2.1.** Les fonctions d'activation

## 2.4 Réseaux de neurones

Un réseau de neurones est un ensemble de neurones formels connectés entre eux, formalisant des structures bien déterminées. Le réseau de neurones a une loi d'apprentissage pour adapter les poids synaptiques selon un critère à minimiser. Les réseaux de neurones sont dotés de deux propriétés importantes : l'aptitude à l'apprentissage et la capacité de généralisation.

L'apprentissage consiste à ajuster les poids synaptiques de telle sorte que le réseau se comporte selon un comportement désiré. Il existe trois types d'apprentissage :

- L'apprentissage supervisé où on a la sortie désirée, et en ajustant les poids synaptiques on minimise l'écart entre la sortie du réseau et la sortie désirée.
- L'apprentissage non supervisé où l'algorithme d'adaptation adapte les poids synaptiques en minimisant un critère donné sans avoir aucune information sur la sortie désirée.
- L'apprentissage par renforcement où le réseau est informé indirectement sur les conséquences de son action choisie, si elle est bonne on va la renforcer.

La généralisation est la capacité du réseau de neurone à donner des performances satisfaisantes pour les entrées qui n'ont pas été utilisées dans la phase d'apprentissage.

### 2.4.1 Types des réseaux de neurones

Selon les connexions entre les neurones, on distingue deux structures principales :

#### 2.4.1.1 Réseaux statiques

Les réseaux statiques sont des réseaux non récurrents où la transmission du signal se fait dans un sens unique de l'entrée vers la sortie ; il n'existe pas de boucles de retour. L'architecture la plus populaire dans ce type de réseaux est l'architecture multicouche dite MLP [17].

#### 2.4.1.2 Réseaux dynamiques

Les réseaux dynamiques sont des réseaux récurrents où il existe des connexions de retour dans ce type de réseau. On peut distinguer trois types de réseaux dynamiques.

- Local où le retour de sortie se fait dans le neurone lui-même.



- Globale où le retour de sortie se fait d'un neurone vers un autre.
- Générale qui est la combinaison des deux types précédents (local et global).

### 2.4.2 Réseaux de neurones multicouches (MLP)

Un réseau de neurones multicouches est un réseau statique constitué d'une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie (Fig. 2.2). La couche est un ensemble de neurones uniformes non connectés entre eux.

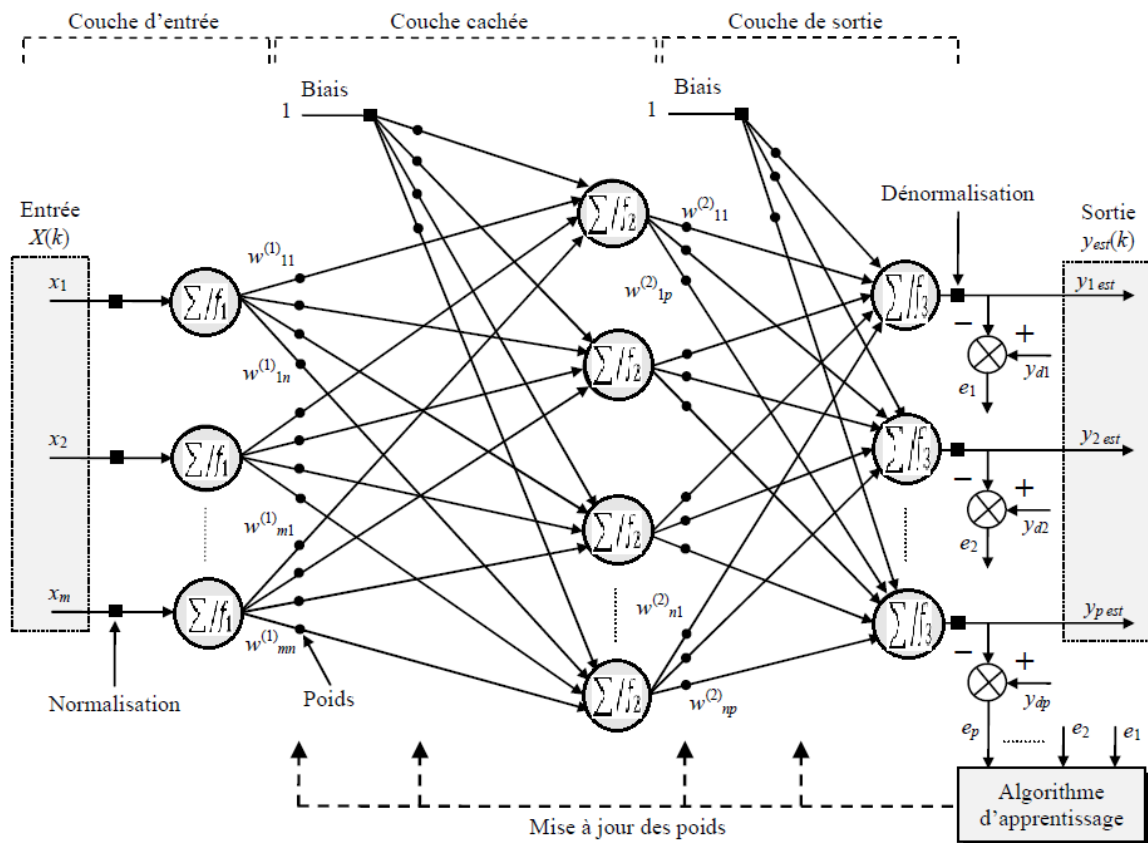


Figure 2.2. Architecture d'un réseaux MLP à deux couches cachées.

#### 2.4.2.1 Algorithme de la retro-propagation du gradient

Avant de commencer l'apprentissage on doit générer la base de données qui contient des entrées aléatoires ( $x$ ) et les sorties désirées correspondantes ( $d$ ), on suppose qu'on a :  $p$  entrées  $\overline{(x(n))} : x_1(n), \dots, x_p(n)$ ,  $q$  sorties désirées  $\overline{(d(n))} : d_1(n), \dots, d_q(n)$  et  $q$  sorties observées du réseau  $\overline{(y(n))} : y_1(n), \dots, y_q(n)$ .

L'algorithme de la rétro-propagation du gradient consiste à minimiser l'erreur entre la sortie observée et la sortie désiré en utilisant la descente du gradient. Dans un réseau MLP les

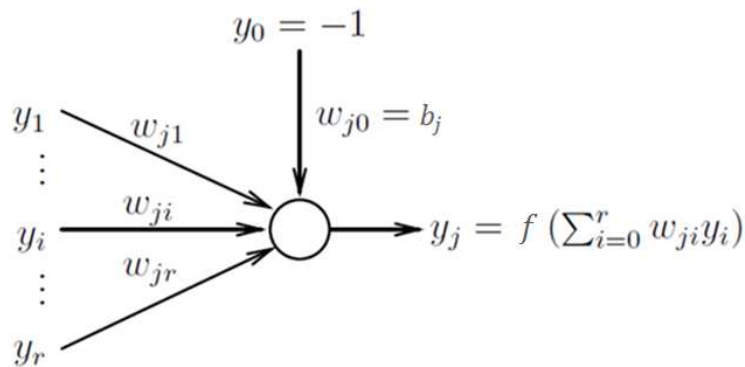
entrées sont propagées vers l'avant de la couche d'entrée vers la couche de sortie tandis que l'erreur est rétro-propagée (vers l'arrière) de la couche de sortie vers celle d'entrée.

### ❖ Couche de sortie

Soit le couple  $(\vec{x}(n), \vec{d}(n))$  désignant la  $n^{\text{ème}}$  donnée d'entraînement, l'erreur observé de la couche de sortie est donnée par :

$$e_j(n) = d_j(n) - y_j(n) \quad (2.2)$$

Où  $j$  correspond au  $j^{\text{ème}}$  neurone formel de la couche de sortie, et  $r$  le nombre des neurones de la dernière couche cachée (Fig. 2.3).



**Figure 2.3.** Modèle du neurone  $j$ .

L'indice  $i$  représente le  $i^{\text{ème}}$  neurone formel de la couche précédente.

Soit  $E$  l'erreur quadratique moyenne observé sur les neurones de la couche de sortie :

$$E(n) = \frac{1}{2} \sum_{j \in 1, q} e_j^2(n) \quad (2.3)$$

La sortie  $y_j(n)$  est définie par :

$$y_j(n) = f[S_j(n)] = f \left[ \sum_{i=0}^r \omega_{ji}(n) y_i(n) \right] \quad (2.4)$$

Pour minimiser l'erreur observée, il faut ajuster les poids synaptiques dans le sens opposé du gradient du critère donné par l'équation (2.3).

La variation des poids est donc donnée par :

$$\Delta\omega_{ji}(n) = -\eta \frac{\partial E(n)}{\partial \omega_{ji}(n)} \quad (2.5)$$

Où :  $\eta \in [0,1]$  représente le taux de l'apprentissage.

En utilisant la règle de chaînage des dérivés partiels le gradient est donné par :

$$\frac{\partial E(n)}{\partial \omega_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial S_j(n)} \frac{\partial S_j(n)}{\partial \omega_{ji}(n)} \quad (2.6)$$

Où :

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\partial \left[ \frac{1}{2} \sum_k e_k^2(n) \right]}{\partial e_j(n)} = \frac{1}{2} \frac{\partial e_j^2(n)}{\partial e_j(n)} = e_j(n) \quad (2.7)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial [d_j(n) - y_j(n)]}{\partial y_j(n)} = -1 \quad (2.8)$$

Dans le cas où la fonction d'activation dans le neurone j est sigmoïde unipolaire, on obtient :

$$\begin{aligned} \frac{\partial y_j(n)}{\partial S_j(n)} &= \frac{\partial \left[ \frac{1}{1 + e^{-S_j(n)}} \right]}{\partial S_j(n)} = \frac{e^{-S_j(n)}}{[1 + e^{-S_j(n)}]^2} = y_j(n) \left[ \frac{e^{-S_j(n)}}{1 + e^{-S_j(n)}} \right] \\ &= y_j(n) \left[ \frac{e^{-S_j(n)} + 1}{1 + e^{-S_j(n)}} - \frac{1}{1 + e^{-S_j(n)}} \right] = y_j(n) [1 - y_j(n)] \end{aligned} \quad (2.9)$$

Dans le cas où la fonction d'activation dans le neurone j est linéaire, on obtient :

$$\frac{\partial y_j(n)}{\partial S_j(n)} = \frac{\partial S_j(n)}{\partial S_j(n)} = 1 \quad (2.10)$$

Dans le cas où la fonction d'activation dans le neurone j est la tangente hyperbolique, on obtient :

$$\frac{\partial y_j(n)}{\partial S_j(n)} = \frac{\partial \left[ \frac{e^{S_j(n)} - e^{-S_j(n)}}{e^{S_j(n)} + e^{-S_j(n)}} \right]}{\partial S_j(n)} = \frac{(e^{S_j(n)} + e^{-S_j(n)})^2 - (e^{S_j(n)} - e^{-S_j(n)})^2}{(e^{S_j(n)} + e^{-S_j(n)})^2} = 1 - y_j^2(n) \quad (2.11)$$

$$\frac{\partial S_j(n)}{\partial \omega_{ji}(n)} = \frac{\partial \left[ \sum_{l=0}^r \omega_{jl}(n) y_l(n) \right]}{\partial \omega_{ji}(n)} = \frac{\partial \left[ \omega_{ji}(n) y_i(n) \right]}{\partial \omega_{ji}(n)} = y_i(n) \quad (2.12)$$

Alors :

A partir de (2.5) et (2.6) le résultat est :

- Dans le cas de la sigmoïde unipolaire :

$$\Delta \omega_{ji}(n) = \eta \cdot e_j(n) \cdot y_j(n) [1 - y_j(n)] \cdot y_i(n) \quad (2.13)$$

- Dans le cas de la fonction linéaire :

$$\Delta \omega_{ji}(n) = \eta \cdot e_j(n) \cdot y_i(n) \quad (2.14)$$

- Dans le cas de la tangente hyperbolique :

$$\Delta \omega_{ji}(n) = \eta * e_j(n) \cdot [1 - y_j^2(n)] \cdot y_i(n) \quad (2.15)$$

#### ❖ Couche cachée

Le gradient du critère (2.3) se calcule de la manière suivante :

Le problème ici : on n'a pas l'erreur observée, donc on ne dérive pas l'EQM par rapport à l'erreur :

$$\frac{\partial E(n)}{\partial \omega_{ji}(n)} = \frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial S_j(n)} \frac{\partial S_j(n)}{\partial \omega_{ji}(n)} \quad (2.16)$$

Où :  $i$  désigne un neurone sur la couche précédente,  $j$  désigne un neurone sur la couche actuelle et  $k$  désigne un neurone sur la couche suivante.

Les deux derniers termes de l'équation (2.16) restent inchangés, nous calculons donc le premier terme seulement.

$$\frac{\partial E(n)}{\partial y_j(n)} = \frac{\partial \left[ \frac{1}{2} \sum_k e_k^2(n) \right]}{\partial y_j(n)} = \sum_k \frac{1}{2} \frac{\partial e_k^2(n)}{\partial y_j(n)} = \sum_k \frac{1}{2} \frac{\partial e_k^2(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_j(n)} \quad (2.17)$$

On sait que tous les  $e_k(n)$  dépendent de  $y_j(n)$  alors on ne peut pas se débarrasser de la somme ci-dessus, comme pour la couche de sortie, mais on peut écrire :

$$\begin{aligned}
 \frac{\partial E(n)}{\partial y_j(n)} &= \sum_K \left[ e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \right] \\
 &= \sum_K \left[ e_k(n) \frac{\partial e_k(n)}{\partial S_k(n)} \frac{\partial S_k(n)}{\partial y_j(n)} \right] \\
 &= \sum_K \left[ e_k(n) \frac{\partial [d_k(n) - f(S_k(n))]}{\partial S_k(n)} \frac{\partial [\sum_l \omega_{kl}(n) y_l(n)]}{\partial y_j(n)} \right] \quad 2.18
 \end{aligned}$$

- Dans le cas où  $f$  est la sigmoïde unipolaire on obtient :

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_K [e_k(n)(-y_k(n))[1 - y_k(n)]\omega_{kj}] \quad (2.19)$$

A partir de (2.16) et en utilisant (2.9), (2.12) et (2.19) :

$$\frac{\partial E(n)}{\partial \omega_{ji}(n)} = -y_j(n)[1 - y_j(n)] \left[ \sum_{K \in C} e_k(n) \cdot y_k(n) \cdot (1 - y_k(n)) \cdot \omega_{kj}(n) \right] \cdot y_i(n) \quad (2.20)$$

et à partir de l'équation (2.5) et (2.20) :

$$\Delta \omega_{kj}(n) = \eta \cdot y_j(n)[1 - y_j(n)] \left[ \sum_{K \in C} e_k(n) \cdot y_k(n) \cdot (1 - y_k(n)) \cdot \omega_{kj}(n) \right] y_i(n) \quad (2.21)$$

- Dans le cas où  $f$  est tangente hyperbolique on obtient :

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_K [e_k(n)(-1)[1 - y_k^2]\omega_{kj}] \quad (2.22)$$

A partir de (2.16) et en utilisant (2.11), (2.12), (2.22) :

$$\frac{\partial E(n)}{\partial \omega_{ji}(n)} = -[1 - y_j^2] \left[ \sum_{K \in C} e_k(n) \cdot (1 - y_k^2) \cdot \omega_{kj}(n) \right] y_i(n) \quad (2.23)$$

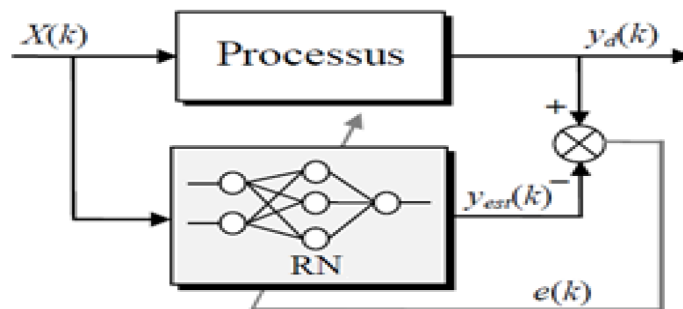
Et a partir de (2.5) et (2.23) :

$$\Delta\omega_{kj}(n) = \eta \cdot [1 - y_j^2] \left[ \sum_{K \in C} e_k(n) \cdot (1 - y_k^2) \cdot \omega_{kj}(n) \right] \cdot y_i(n) \quad (2.24)$$

**Remarque :** ces résultats sont valables pour toutes les couches cachées, seulement dans la première couche cachée on remplace  $y_i(n)$  par  $x_i(n)$ .

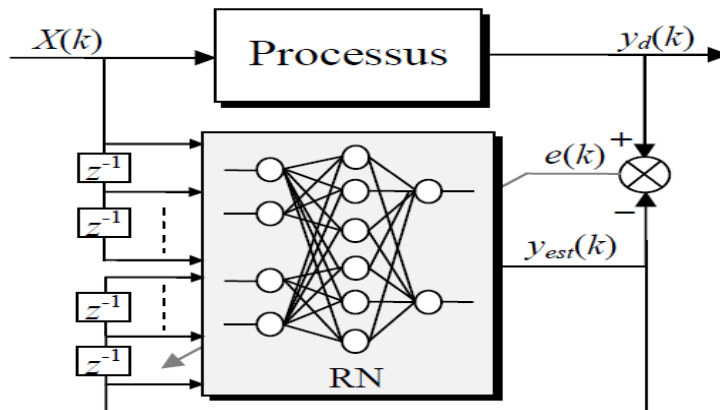
## 2.5 Identification neuronale d'un système dynamique

Avant de passer à l'identification on doit d'abord générer une base de données contenant des entrées aléatoires et les sorties désirées correspondantes. La figure 2.4 montre le principe de l'identification direct d'un processus, on applique un vecteur aléatoire d'entrée  $X(k)$  au processus et au réseau de neurones, les quels génèrent la sortie désirée  $y_d(k)$  et la sortie estimée  $y_{est}(k)$  respectivement. Ensuite, l'algorithme d'apprentissage adapte les poids du réseau en minimisant l'erreur observé  $e(k)$ , ce cycle est répété plusieurs fois jusqu'à ce que l'erreur devient acceptable.



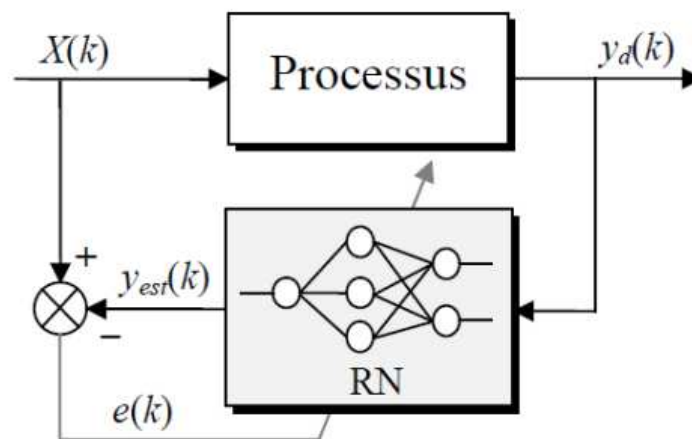
**Figure 2.4.** Identification direct d'un processus.

Il existe deux solutions pour prendre en compte l'effet dynamique du système à identifier, la première consiste à utiliser des réseaux de neurones récurrents et la deuxième consiste à ajouter des entrées retardées au réseau de neurones  $[x(k), \dots, x(k - m), y(k), \dots, y(k - n)]$ . La deuxième solution (Fig. 2.5) est préférée à cause de la simplicité des algorithmes d'apprentissage des réseaux statiques.



**Figure 2.5.** Identification direct d'un processus en utilisant les entrées et sorties retardées.

Il existe une autre approche d'identification dite l'identification inverse, le principe de fonctionnement reste le même seulement on inverse les entrées avec les sorties du réseau de neurones c'est-à-dire on identifie le modèle inverse du système (Fig. 2.6).



**Figure 2.6.** Identification inverse d'un processus.

## 2.6 Commande neuronale d'un système dynamique

A cause de leur propriété d'approximation des fonctions non linéaires, les réseaux de neurones sont très utiles dans la commande des systèmes non linéaires. Le but du réseau de neurones est d'approcher la fonction de commande, qui est dans la majorité des cas non linéaire, en ajustant les poids de ces connexions dans l'étape de l'apprentissage. L'apprentissage peut se faire en ligne (commande adaptative) ou hors ligne (on doit avoir une base de données définissant la fonction de commande).

Il existe plusieurs stratégies de commande neuronale. On peut les classer dans trois catégories principales :

- la commande directe par modèle inverse.
- La commande basée sur l'erreur de sortie du processus.
- La commande adaptative.

### 2.6.1 Commande directe par modèle inverse

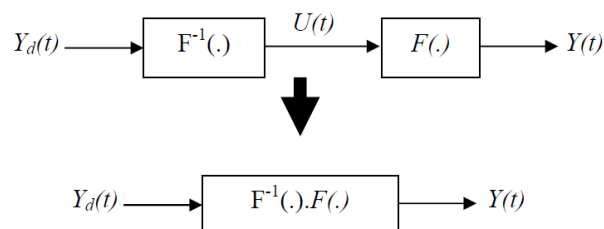
La commande directe basée sur le modèle inverse du système est l'architecteur la plus simple de la commande neuronale ; elle peut être réalisée en deux étapes successives :

- Construction hors ligne du modèle inverse du processus comme le montre la figure 2.6
- Mise en marche du réseau devant le processus en boucle ouverte pour le commander.

Considérons le processus défini par la fonction de récurrence  $F(\cdot)$  suivante :

$$y(k) = F(u(k), \dots, u(k-m), y(k-1), \dots, y(k-n)) \quad (2.25)$$

Tant que le régulateur (le réseau de neurone) simulé par la fonction inverse  $F^{-1}(\cdot)$  du processus est placé en boucle ouvert devant ce dernier alors la fonction globale devient l'identité donc  $y(k) = y_d(k)$ . La figure 2.7 montre le principe de fonctionnement de cette commande :



**Figure 2.7.** Principe de la commande directe par modèle inverse.

Cette méthode de commande présente plusieurs inconvénients. Le processus peut ne pas être inversible (le cas où le système donne la même réponse à des entrées différents), un autre inconvénient si une ou plusieurs paramètres du processus subissent une variation ou une perturbation apparaît le produit  $F(\cdot)F^{-1}(\cdot)$  devient différent de un et la sortie  $y(k)$  diverge.

### 2.6.2 Commande neuronale à modèle de référence

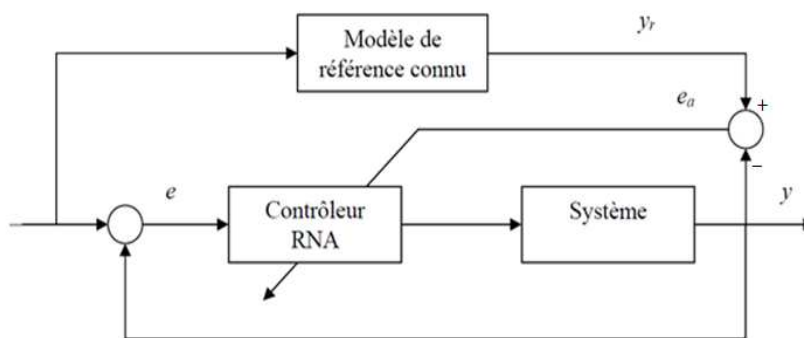
Dans cette architecture (Fig. 2.8) l'apprentissage du régulateur se fait en deux étapes : la première se fait hors ligne et sert à trouver les poids initiaux pour commencer la commande, et la deuxième se fait en ligne et sert à modifier ces poids initiaux en cas d'un changement



dans les paramètres du processus ou une apparition d'une perturbation. En utilisant cette architecture les inconvénients de l'architecture directe par modèle inverse disparaissent.

Ce type d'architecture vise à minimiser l'erreur entre la sortie du système et la sortie du modèle.

$$e_a(n) = y_r(n) - y(n) \tag{2.26}$$

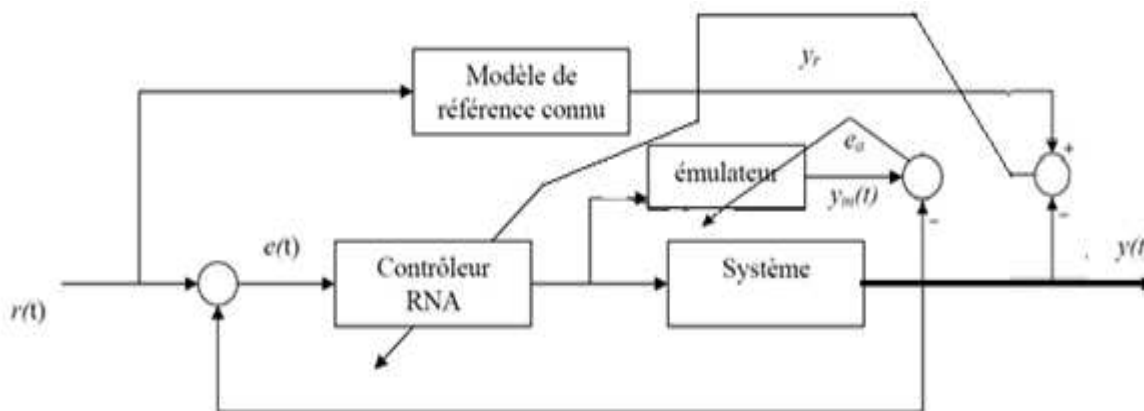


**Figure 2.8.** Commande neuronale à modèle de référence.

Pour faire l'apprentissage du régulateur on utilise les équations (2.3), (2.5) et

$$\frac{\partial E(n)}{\partial \omega_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial S_j(n)} \frac{\partial S_j(n)}{\partial \omega_{ji}(n)} \tag{2.27}$$

A part le terme  $\frac{\partial y_j(n)}{\partial u_j(n)}$ , représentant le Jacobien du processus, le calcul des autres termes a été donné dans le paragraphe 2.4.2.1. Une des solutions proposées pour résoudre le problème du Jacobien est l'implantation d'un émulateur neuronale qui modélise le processus (Fig. 2.9).

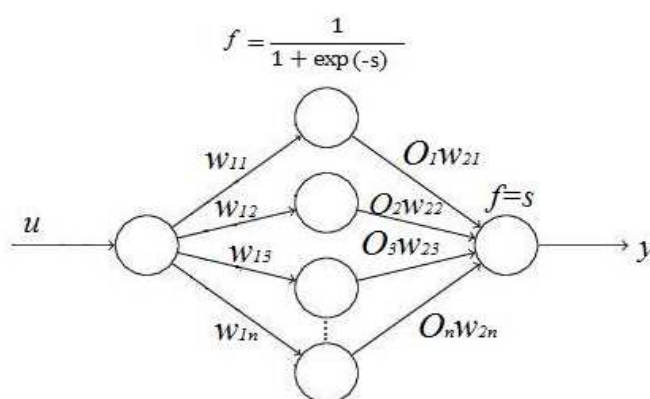


**Figure 2.9.** Commande neuronale à modèle de référence avec un émulateur.

Avant de commencer la commande et l'adaptation de l'émulateur en ligne on doit construire, hors ligne, l'émulateur et faire, hors ligne, l'apprentissage du régulateur pour déterminer les poids initiaux.

### 2.6.2.1 Détermination du jacobien

Le jacobien du processus peut s'obtenir à partir de son émulateur neuronal. Considérons l'émulateur MLP à deux couches : une couche cachée contient n neurones avec une fonction d'activation sigmoïde unipolaire et une couche de sortie contient un seul neurone avec une fonction d'activation identité (Fig. 2.10).



**Figure 2.10.** L'architecture de l'émulateur du processus proposé.

A partir de la figure 2.10 on a :

$$y = \sum_{i=1}^n O_i w_{2i} \tag{2.28}$$

$$O_i = \frac{1}{1 + \exp(-w_{1i} \cdot u)} \tag{2.29}$$

Donc, à partir de (2.28) et (2.29) :

$$y = \sum_{i=1}^n \frac{1}{1 + \exp(-w_{1i} \cdot u)} \cdot w_{2i} \tag{2.30}$$

Maintenant on dérive la sortie y (l'équation 2.30) par rapport à l'entrée u :

$$\frac{\partial y}{\partial u} = \frac{\partial}{\partial u} \left[ \sum_{i=1}^n \frac{1}{1 + \exp(-w_{1i} \cdot u)} \cdot w_{2i} \right] = \sum_{i=1}^n w_{2i} \cdot \frac{\partial}{\partial u} \left[ \frac{1}{1 + \exp(-w_{1i} \cdot u)} \right]$$

$$\begin{aligned} \frac{\partial y}{\partial u} &= \sum_{i=1}^n w_{2i} \cdot \frac{w_{1i} \cdot \exp(-w_{1i} \cdot u)}{[1 + \exp(-w_{1i} \cdot u)]^2} = \sum_{i=1}^n w_{2i} \cdot w_{1i} O_i \cdot \frac{\exp(-w_{1i} \cdot u)}{1 + \exp(-w_{1i} \cdot u)} \\ \frac{\partial y}{\partial u} &= \sum_{i=1}^n w_{2i} \cdot w_{1i} O_i \cdot \left[ \frac{\exp(-w_{1i} \cdot u) + 1}{1 + \exp(-w_{1i} \cdot u)} - \frac{1}{1 + \exp(-w_{1i} \cdot u)} \right] \\ \frac{\partial y}{\partial u} &= \sum_{i=1}^n w_{2i} \cdot w_{1i} O_i [1 - O_i] \end{aligned} \quad (2.31)$$

A partir de (2.31), le jacobien peut s'exprimer en utilisant la relation suivante :

$$jac \cong \frac{\partial y}{\partial u} = \sum_{i=1}^n w_{2i} \cdot w_{1i} O_i [1 - O_i] \quad (2.32)$$

### 2.6.3 Commande basé sur l'apprentissage d'un régulateur conventionnel

Dans ce paragraphe on va présenter brièvement deux différentes architectures de cette commande et ensuite on va détailler la troisième architecture que nous avons utilisé dans notre travaille. Ces architectures sont :

- Identification du régulateur.
- Apprentissage en parallèle avec le régulateur.
- Auto-ajustement des paramètres régulateur.

#### 2.6.3.1 Identification du régulateur

Cette architecture consiste à identifier hors ligne le régulateur en utilisant un réseau de neurones (Fig. 2.11). Quand la phase d'identification est terminée, on remplace le régulateur conventionnel par ce réseau de neurones dans la boucle de commande [14].

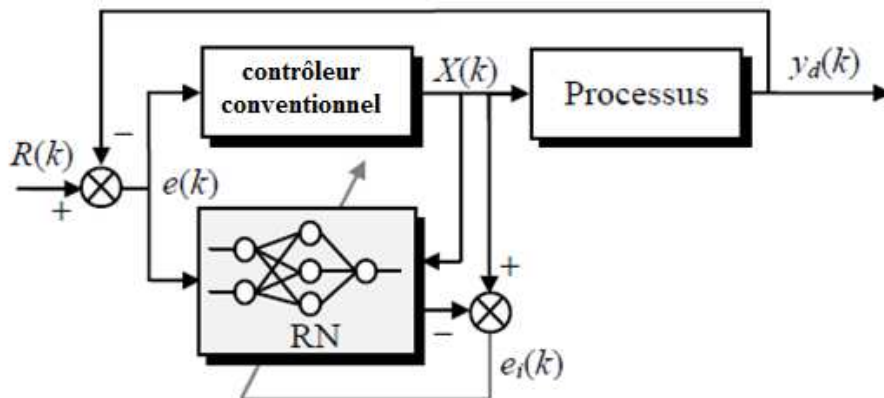


Figure 2.11. Architecture d'identification du régulateur.

### 2.6.3.2 Apprentissage en parallèle avec le régulateur

Dans cette architecture on a deux régulateurs dans la boucle de commande (Fig. 2.12), le premier est un régulateur conventionnel placé en parallèle avec le deuxième régulateur qui est un réseau de neurones. L'apprentissage de ce dernier se fait en ligne pour corriger et améliorer les performances du régulateur conventionnel.

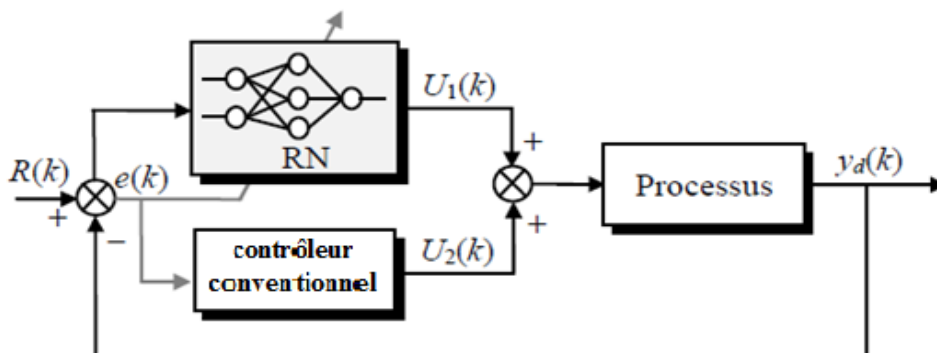


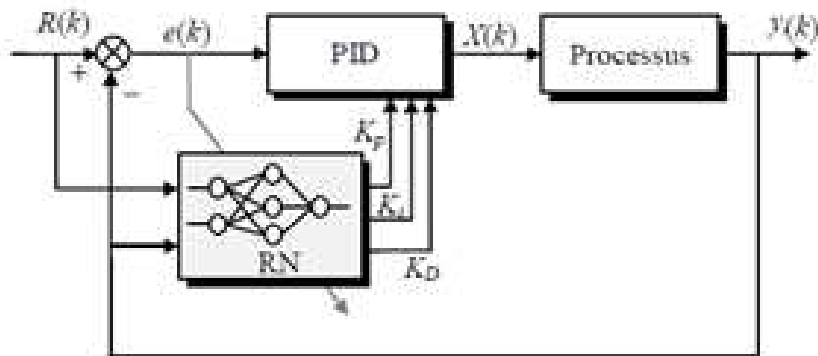
Figure 2.12. Architecture d'apprentissage en parallèle avec le régulateur.

### 2.6.3.3 Auto-ajustement des paramètres du régulateur

Dans cette architecture, le réseau de neurones ajuste les paramètres du régulateur conventionnel en ligne, cette architecture rend le régulateur conventionnel adaptatif. Pour avoir de bonnes performances au début de la régulation on doit trouver les paramètres du régulateur en utilisant les méthodes classiques ou un apprentissage du réseau hors ligne et on utilise ces paramètres comme des valeurs initiales à l'apprentissage en ligne [15].

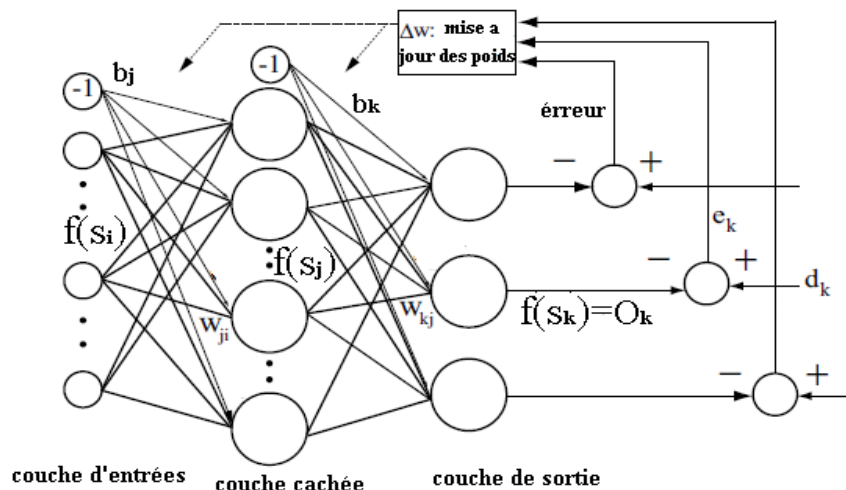
### 2.6.4 PID neuronale

La figure 2.13 montre l'architecture de la commande PID auto-ajustement en utilisant un réseau de neurones.



**Figure 2.13.** Architecture d'auto-ajustement des paramètres du PID (PID neuronale).

La structure du réseau de neurones utilisé (un réseau MLP) est donnée par la figure 2.14.



**Figure 2.14.** Structure du réseau de neurones utilisé.

Notre objectif est de minimiser l'erreur quadratique moyenne définie par :

$$E = \frac{1}{2} \sum_{k=0}^K e^2(k) \tag{2.33}$$

Où  $e(k) = R(k) - y(k)$  est l'écart entre la référence et la sortie du système.

On suit la même démarche présentée dans le paragraphe 2.4.2.1 pour déterminer le gradient du critère à minimiser.

❖ Pour la couche de sortie :

$$\frac{\partial E(n)}{\partial w_{kj}} = \frac{\partial E(n)}{\partial e(n)} \frac{\partial e(n)}{\partial y(n)} \frac{\partial y(n)}{\partial u(n)} \frac{\partial u(n)}{\partial O_k} \frac{\partial O_k}{\partial S_k} \frac{\partial S_k}{\partial w_{kj}} \quad (2.34)$$

$$\frac{\partial E(n)}{\partial e(n)} = \frac{\partial \left[ \frac{1}{2} \sum_{k=0}^K e^2 \right]}{\partial e(n)} = \frac{1}{2} \frac{\partial e^2(n)}{\partial e(n)} = e(n) \quad (2.35)$$

$$\frac{\partial e(n)}{\partial y(n)} = \frac{\partial [R(k) - y(k)]}{\partial y(n)} = -1 \quad (2.36)$$

$$\frac{\partial y(n)}{\partial u(n)} = jac(n) \cong \sum_{i=1}^n w_{2i} \cdot w_{1i} O_i [1 - O_i]$$

$$\frac{\partial u(n)}{\partial O_k} = \begin{cases} e(n) - e(n-1) & (k=1) \\ e(n) & (k=2) \\ e(n) - 2e(n-1) + e(n-2) & (k=3) \end{cases} \quad (2.37)$$

Car la commande délivrée par le PID est donnée par :

$$u(t) = u(t-1) + k_p[(e(t) - e(t-1))] + k_i e(t) + k_d[e(t) - 2e(t-1) + e(t-2)] \quad (2.38)$$

Avec  $K_p = O_1$  ;  $K_i = O_2$  ;  $K_d = O_3$

Etant donné que la fonction d'activation utilisée dans la couche de sortie est la fonction identité, on obtient :

$$\frac{\partial O_k}{\partial S_k} = 1 \quad (2.39)$$

$$\frac{\partial S_k}{\partial w_{kj}} = f(S_j) = \frac{1}{1 + \exp(-\sum_{i=1}^I w_{ji} O_i - b_j)} = O_j \quad (2.40)$$

On pose :

$$\delta_k = e(n) \cdot jac(n) \frac{\partial u(n)}{\partial O_k} ; \quad k = 1,2,3 \quad (2.41)$$

On peut écrire donc :

$$\Delta w_{kj}(t + 1) = \eta \delta_k O_j \quad (2.42)$$

❖ Pour la couche cachée :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial O_j(n)} \cdot \frac{\partial O_j(n)}{\partial S_j(n)} \cdot \frac{\partial S_j(n)}{\partial w_{ji}(n)} \quad (2.43)$$

$$\begin{aligned} \frac{\partial E(n)}{\partial w_{ji}(n)} &= \left[ \frac{\partial \left[ \frac{1}{2} \sum_k e^2(n) \right]}{\partial O_j(n)} \right] \cdot \frac{\partial O_j(n)}{\partial S_j(n)} \cdot \frac{\partial S_j(n)}{\partial w_{ji}(n)} \\ &= \left[ \sum \frac{1}{2} \frac{\partial e^2(n)}{\partial e(n)} \cdot \frac{\partial e(n)}{\partial O_j(n)} \right] \cdot \frac{\partial O_j(n)}{\partial S_j(n)} \cdot \frac{\partial S_j(n)}{\partial w_{ji}(n)} \\ &= \left[ \sum e(n) \frac{\partial e(n)}{\partial y(n)} \cdot \frac{\partial y(n)}{\partial O_k(n)} \cdot \frac{\partial O_k(n)}{\partial O_j(n)} \right] \cdot \frac{\partial O_j(n)}{\partial S_j(n)} \cdot \frac{\partial S_j(n)}{\partial w_{ji}(n)} \\ &= \left[ \sum e(n) \frac{\partial e(n)}{\partial y(n)} \cdot \frac{\partial y(n)}{\partial u(n)} \cdot \frac{\partial u(n)}{\partial O_k(n)} \cdot \frac{\partial O_k(n)}{\partial O_j(n)} \right] \cdot \frac{\partial O_j(n)}{\partial S_j(n)} \cdot \frac{\partial S_j(n)}{\partial w_{ji}(n)} \\ &= \left[ \sum e(n) \cdot (-1) \cdot jac(n) \cdot \frac{\partial u(n)}{\partial O_k(n)} \cdot w_{kj} \right] O_j(n) [1 - O_j(n)] O_i(n) \quad (2.44) \end{aligned}$$

On a :

$$S_k = e(n) \cdot jac(n) \cdot \frac{\partial u(n)}{\partial O_k(n)} \quad (2.45)$$

A partir de (2.43) et (2.45) :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = - \sum S_k \cdot w_{kj} \cdot O_j(n) [1 - O_j(n)] O_i(n) \quad (2.46)$$

A partir de (2.5) et (2.46) :

$$\Delta w_{ji}(n) = \eta \cdot \sum S_k \cdot w_{kj} \cdot O_j(n) [1 - O_j(n)] O_i(n) \quad (2.47)$$

On pose :

$$S_j = \sum S_k \cdot w_{kj} \cdot O_j(n) [1 - O_j(n)] \quad (2.48)$$

On peut écrire alors :

$$\Delta w_{ji}(n) = \eta \cdot S_j \cdot O_i(n) \quad (2.49)$$

## 2.7 Conclusion

Dans ce chapitre, nous avons présenté brièvement les notions élémentaires relatives aux réseaux de neurones, rappelé les différentes méthodes d'identification et de commande en utilisant les réseaux de neurones et donné en détaille les étapes à suivre pour mettre en œuvre la commande PID neuronale. Cette méthode de commande, auto-ajustable, est simple à implémenter et permet d'améliorer les performances de la commande PID classique.



# **Chapitre 3**

**Simulations et résultats**

### 3.1 Introduction

Afin de mettre en évidence les performances de la commande PID neuronal, nous considérons, dans ce chapitre, la commande de deux systèmes de complexité différente : la commande d'un réacteur chimique parfaitement mélangé (CSTR : Continuous Stirred Tank Reactor) et la commande d'un moteur à courant continu. Nous présentons une brève description des deux systèmes, ensuite nous donnons les différents résultats, obtenus par simulation, de la commande PID neuronal ainsi que ceux de la comparaison avec la commande PID conventionnelle.

### 3.2 Présentation du CSTR

Le CSTR (Fig. 3.1) est un réacteur continu parfaitement mélangé, dans lequel un produit  $A$  sera transformé en un autre produit  $B$  selon une réaction chimique exothermique. Le volume  $v$  utilisé du réacteur est constant et le mélange est considéré comme parfait avec une température  $T$  supposée uniforme. Le réacteur travaille en continu, il est alimenté par la charge fraîche de concentration et température initiales  $C_{a0}$  et  $T_0$  respectivement, et les produits de la réaction sont soutirés de manière constante. Ce processus de transformation ( $A$  vers  $B$ ) est décrit par les équations d'état suivantes [4] [5]:

$$\dot{C}_a(t) = \frac{q}{v}(C_{a0} - C_a(t)) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \quad (3.1)$$

$$\dot{T}(t) = \frac{q}{v}(T_0 - T(t)) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} + k_2 q_c(t) \left( 1 - e^{-\frac{k_3}{q_c(t)}} (T_{c0} - T(t)) \right) \quad (3.2)$$

Tel que :

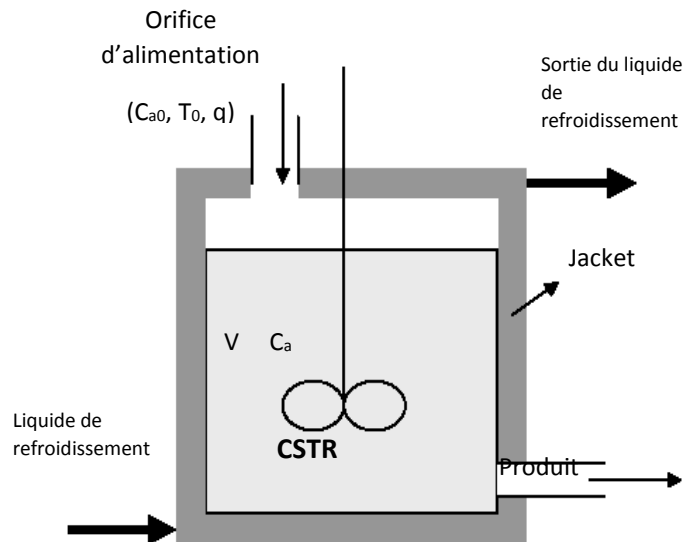
$$k_1 = -\frac{\Delta H k_0}{\rho C_p} , \quad k_2 = \frac{\rho_c C_{pc}}{\rho C_p v} , \quad k_3 = \frac{h_a}{\rho_c C_{pc}}$$

Avec :

- $v$  : le volume utilisé du réacteur est constant.
- $C_{a0}$  : la charge fraîche de concentration exprimée en  $mol/l$ .
- $T_0$  : la température initiale exprimée en *Kelvin*.
- $T_c$  : la température du jacket.
- $T_{c0}$  : la température initiale du jacket.
- $C_a(t)$  : la concentration du produit  $A$  exprimée en  $mol/l$ .
- $T(t)$  : la température du mélange supposée uniforme, exprimée en *Kelvin*.

- $q_c(t)$  : le débit du liquide de refroidissement utilisé pour contrôler la température de la réaction, il est exprimé en  $l/min$ .
- $q$  : le débit du processus exprimé en  $l/min$ .

Les valeurs de toutes les constantes sont données dans [5].



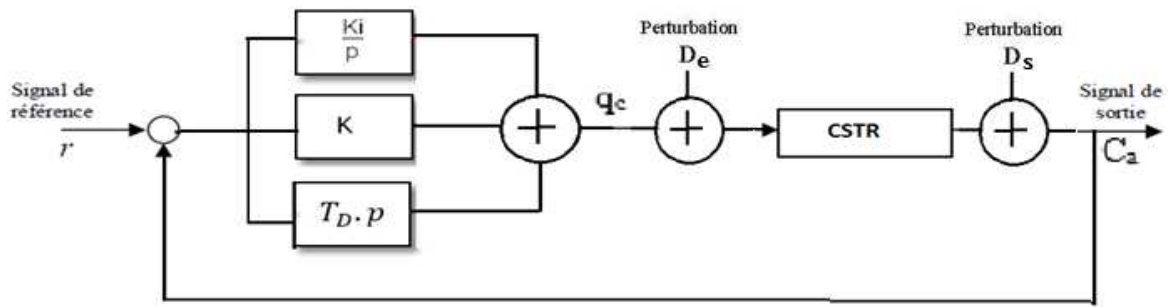
**Figure 3.1** : Réacteur continu parfaitement mélangé

La transformation chimique du produit A en B est irréversible et exothermique, donc le dégagement de la chaleur durant la transformation implique une augmentation de la température  $T(t)$  du réacteur. Cette augmentation de la température peut nous poser deux problèmes : l'un est que le produit B désirable devient un produit C indésirable quand la température dépasse la température d'activation relative à la réaction et l'autre est l'emballement thermique du réacteur qui peut faire une explosion.

A partir de la représentation d'état du réacteur 3.1 et 3.2 on remarque qu'on peut contrôler la concentration  $C_a(t)$  en utilisant la température  $T(t)$  et on peut agir sur la température  $T(t)$  en utilisant le débit du liquide de refroidissement  $q_c(t)$ , on peut donc contrôler la concentration  $C_a(t)$  (la grandeur de sortie) en utilisant  $q_c(t)$  ( la grandeur de commande ).

### 3.2.1 Commande du CSTR en utilisant un régulateur PID classique

Le schéma de commande est donné par la figure 3.2, où la grandeur de commande  $q_c(t)$  est délivrée par le régulateur PID. Nous considérons les cas : sans perturbation ( $D_e = D_s = 0$ ), perturbation additive en entrée ( $D_s = 0$ ) et perturbation additive en sortie ( $D_e = 0$ ).

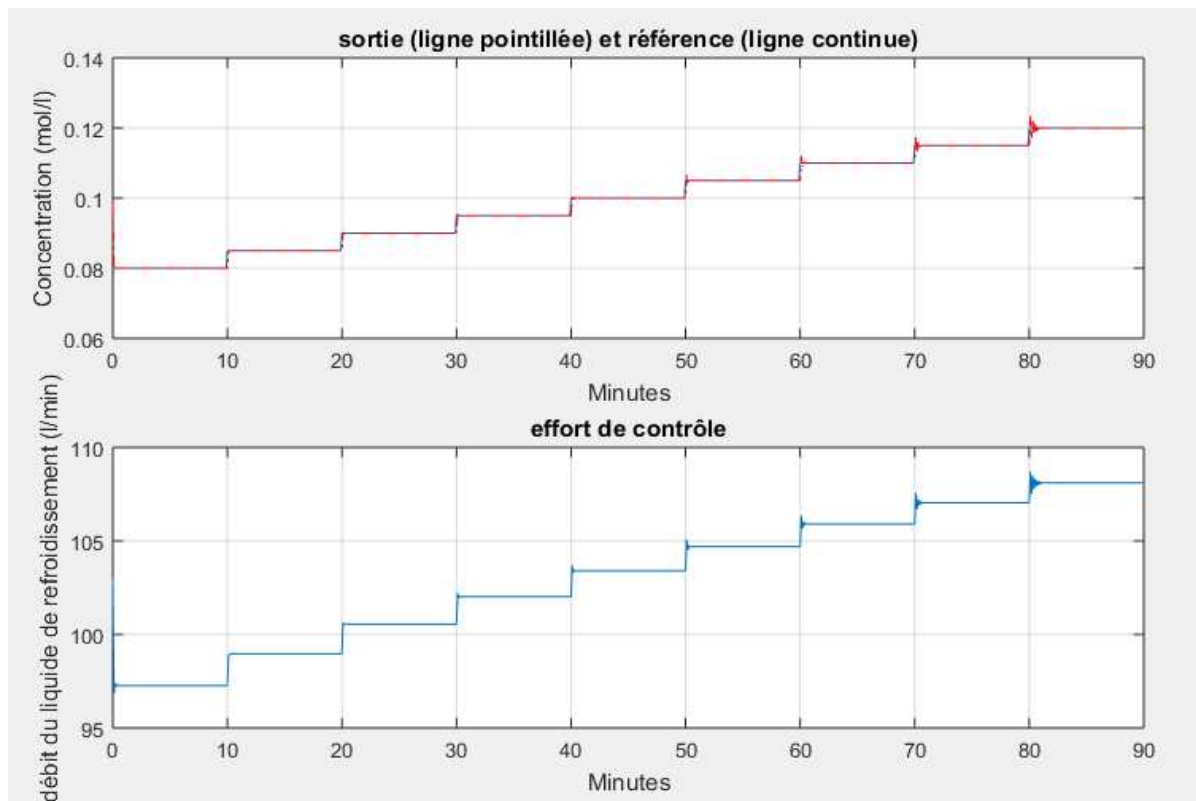


**Figure 3.2 :** Commande du CSTR en utilisant un PID classique.

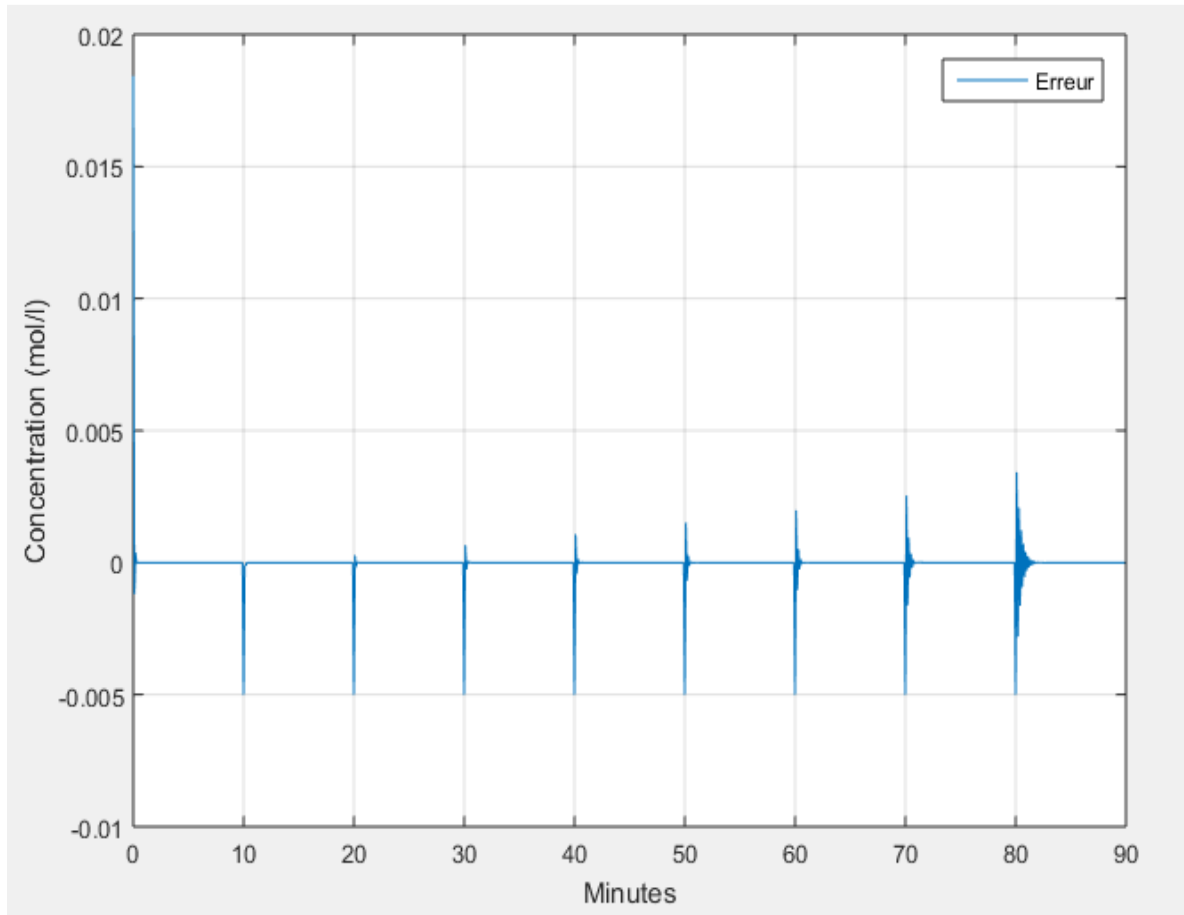
Pour la mise en œuvre du régulateur PID nous utilisons les valeurs des paramètres calculées dans [4]. Elles valent :  $K_p = 91.2$ ,  $T_i = 0.178$  s,  $T_d = 0.208$  s. Ces valeurs ont été optimisées pour la trajectoire de référence utilisée.

### 3.2.1.1 Simulation sans perturbations

La trajectoire de référence utilisée, la sortie du CSTR obtenue et le signal de commande sont donnés par la figure 3.3. L'écart entre la référence et la sortie du CSTR est représenté sur la figure 3.4.



**Figure 3.3 :** Sortie du CSTR, la référence et la commande délivrée par le PID.

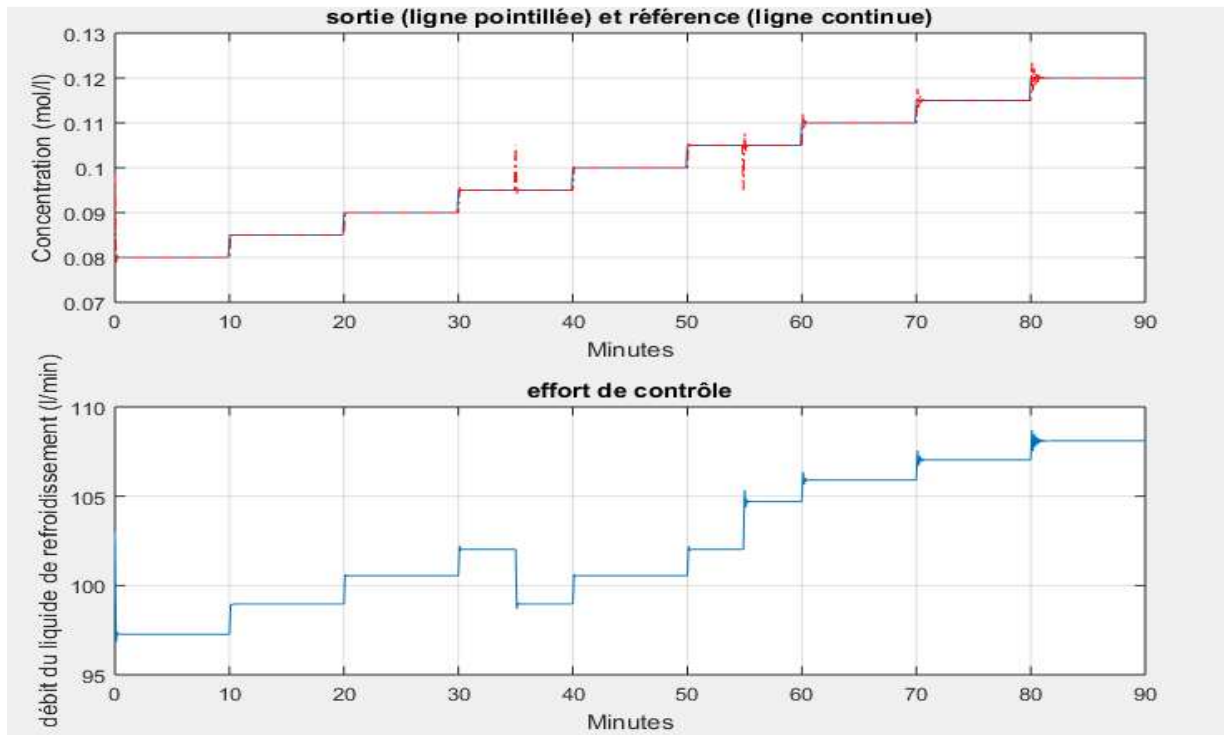


**Figure 3.4 :** Ecart entre la sortie réelle du CSTR et la référence.

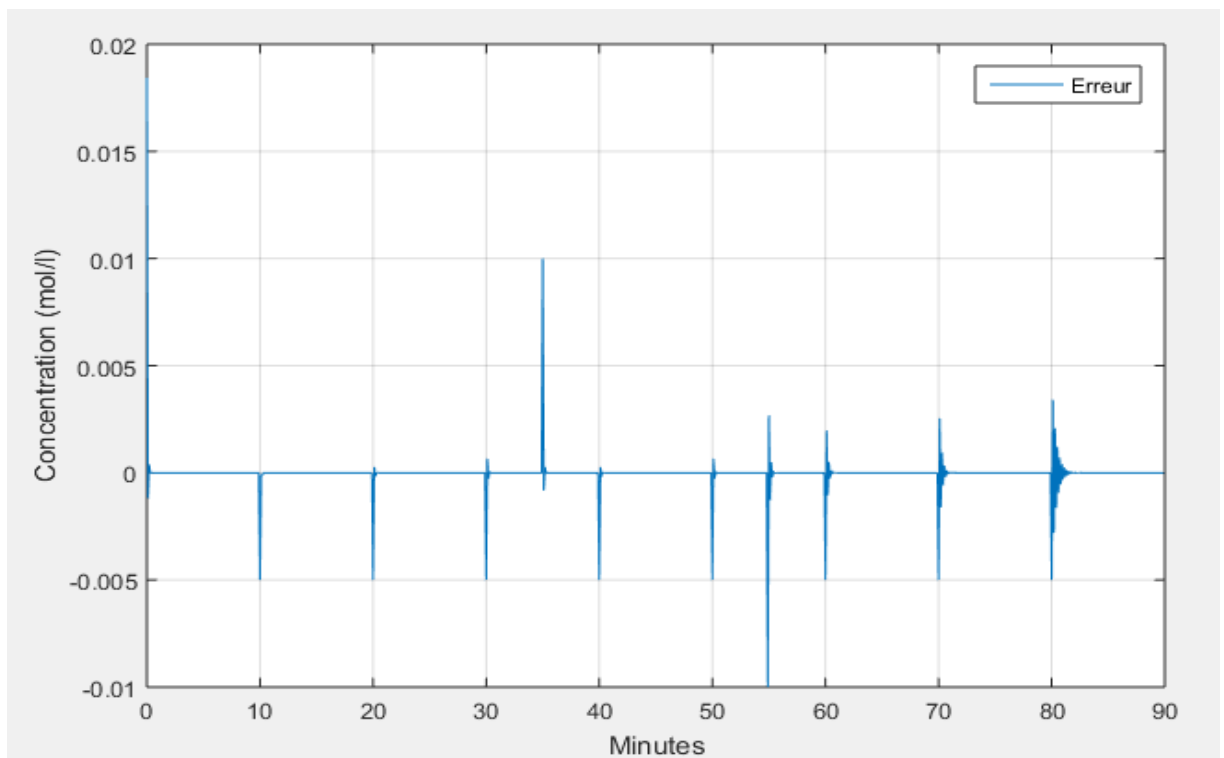
A partir des figures 3.3 et 3.4 nous remarquons qu'il y a une bonne poursuite de la trajectoire de référence, cependant une dégradation des performances est observée lorsque l'amplitude de la référence augmente et s'éloigne du point de linéarisation.

### 3.2.1.2 Simulation avec perturbation de 10 % additive en sortie

Pour tester la robustesse du régulateur vis-à-vis des perturbations externes. Une perturbation additive à la sortie du système de 10 % est introduite durant l'intervalle [35 min 55 min]. Les résultats obtenus sont donnés par les figures 3.5 et 3.6.



**Figure 3.5:** Sortie du CSTR avec une perturbation de 10% additive à sa sortie, la référence et la commande délivré par le PID.

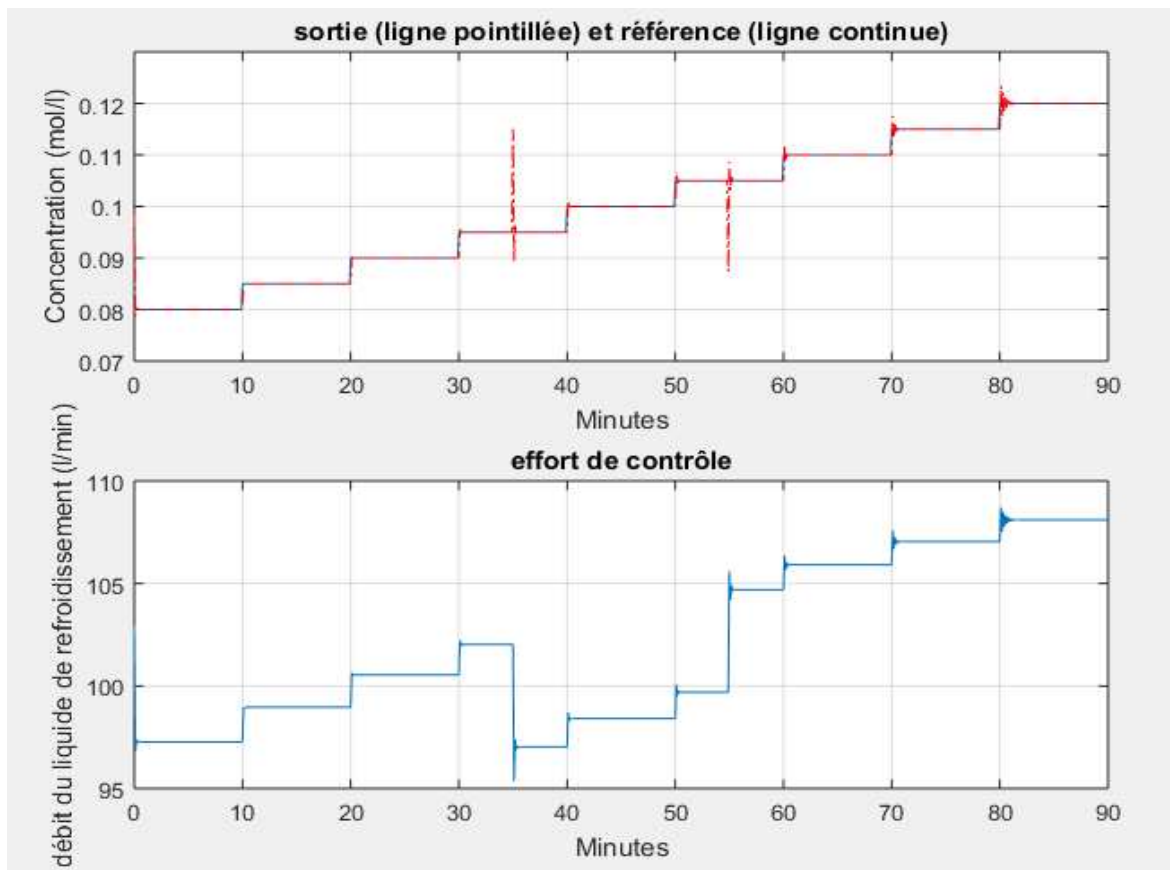


**Figure 3.6:** Ecart entre la sortie réelle du système perturbé dans sa sortie et la référence.

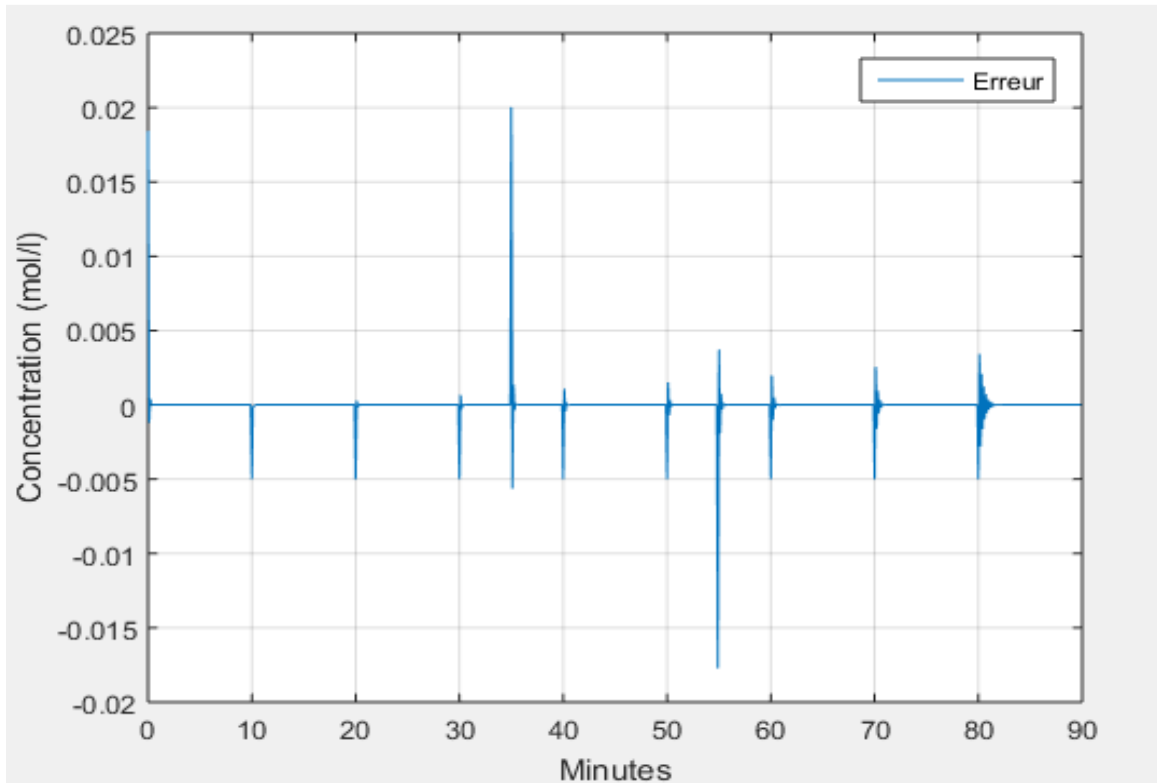
A partir de les figures 3.5 et 3.6 nous remarquons qu'avec l'apparition (minute 35) et la disparition (minute 65) de la perturbation l'erreur augmente, ensuite le PID agit sur la commande pour compenser cette erreur.

### 3.2.1.3 Simulation avec perturbation de 5 % additive à l'entrée

Une perturbation additive à l'entrée du système de 5 % est introduite durant l'intervalle [35min - 55 min]. Les résultats obtenus sont donnés par les figures 3.7 et 3.8.



**Figure 3.7 :** Sortie du CSTR avec une perturbation de 5% dans son entrée, la référence et la commande délivrée par le PID.

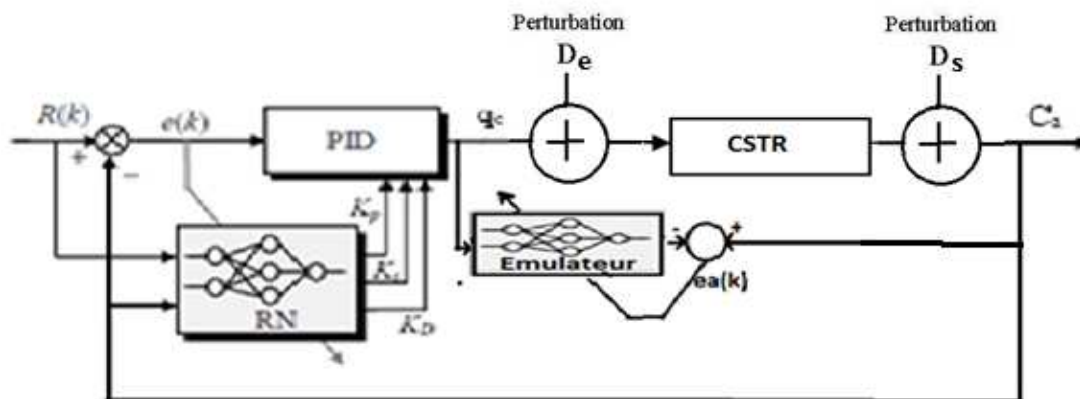


**Figure 3.8 :** Ecart entre la sortie réelle du système perturbé dans son entrée et la référence.

A partir de les figures 3.7 et 3.8 nous remarquons que le régulateur arrive à compenser l'erreur due à l'introduction de la perturbation.

### 3.2.2 Commande du CSTR en utilisant un régulateur PID neuronal

Le schéma de commande est donné par la figure 3.9, où les valeurs des paramètres du PID sont obtenues à partir d'un réseau de neurones. Nous considérons les cas sans et avec perturbation.



**Figure 3.9 :** Neuro-PID raccordé avec le CSTR en boucle fermée.



Tout d'abord nous utilisons la procédure donnée dans le chapitre précédent pour faire l'apprentissage, hors-ligne, à l'émulateur. L'apprentissage du réseau de neurones délivrant les valeurs des paramètres du PID se fait en ligne. L'opération de l'apprentissage de l'émulateur est expliquée par l'organigramme de la figure 3.10. La procédure de commande est illustrée par l'organigramme de la figure 3.11.

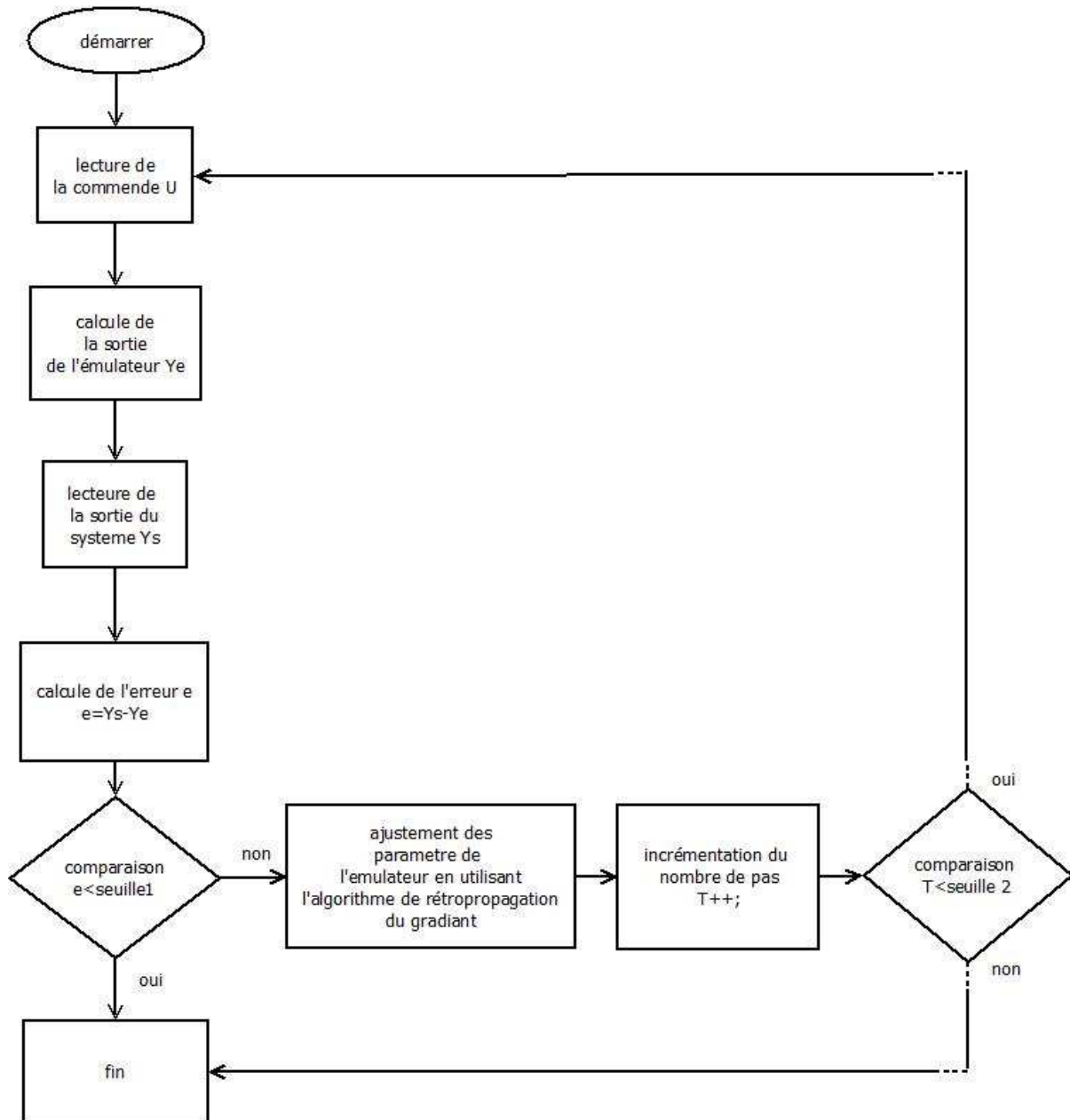
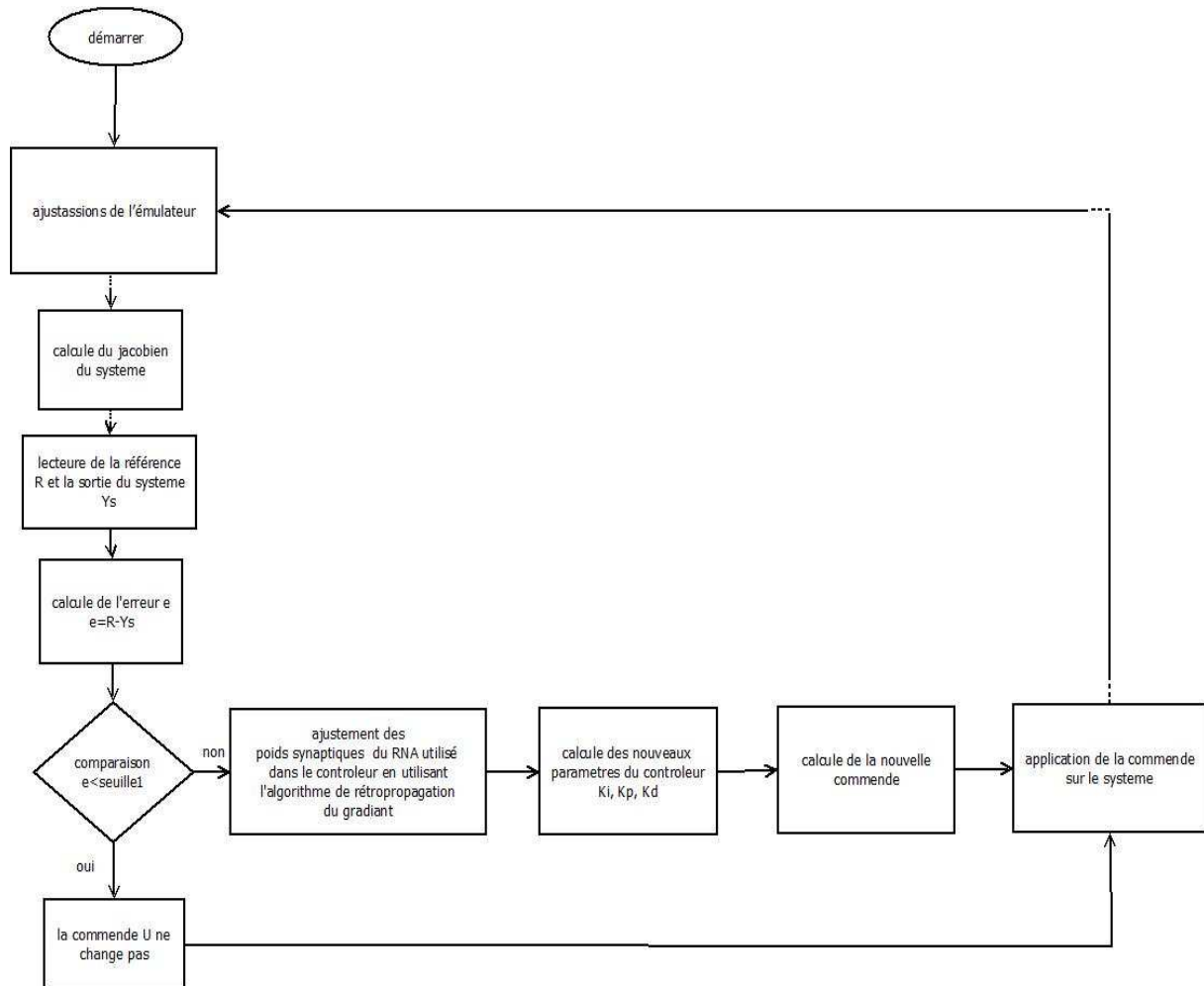


Figure 3.10 : Organigramme de l'opération d'apprentissage de l'émulateur.

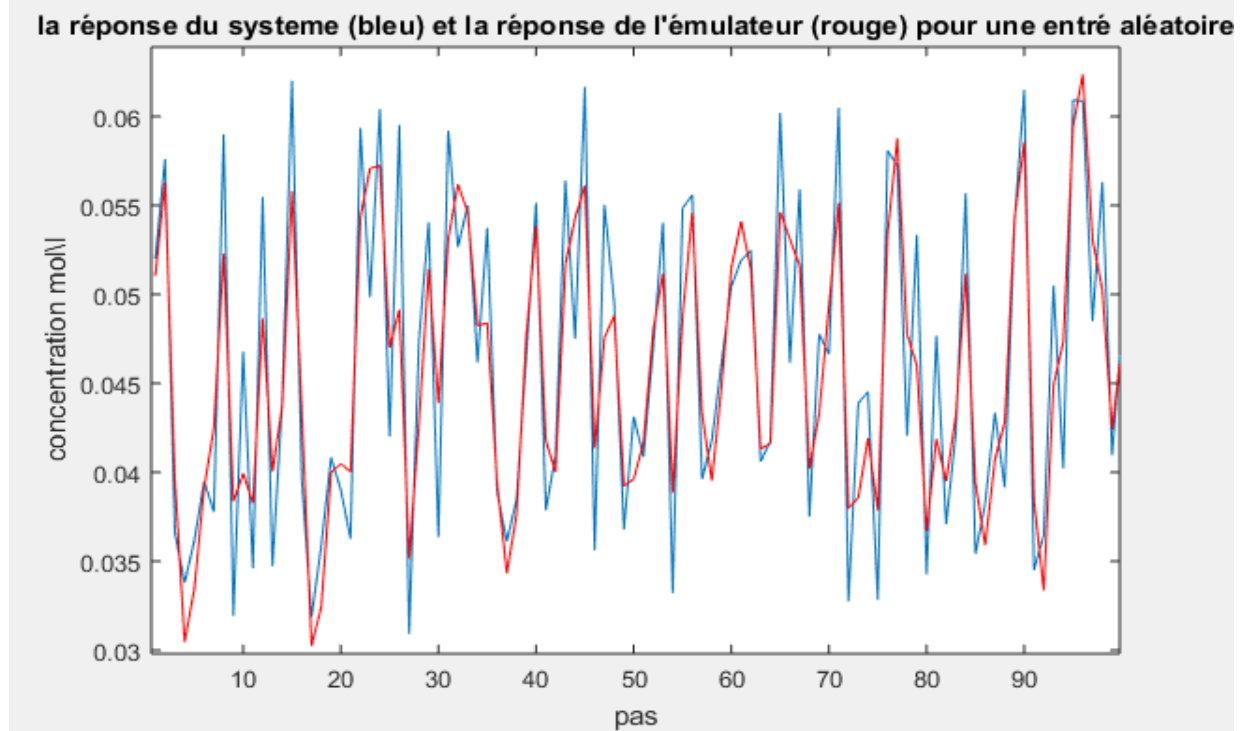


**Figure 3.11 :** Organigramme de l'opération de commande.

L'étape de l'apprentissage de l'émulateur hors-ligne est la même pour tous les cas (sans perturbation et avec perturbation), elle est donc réalisée seule fois.

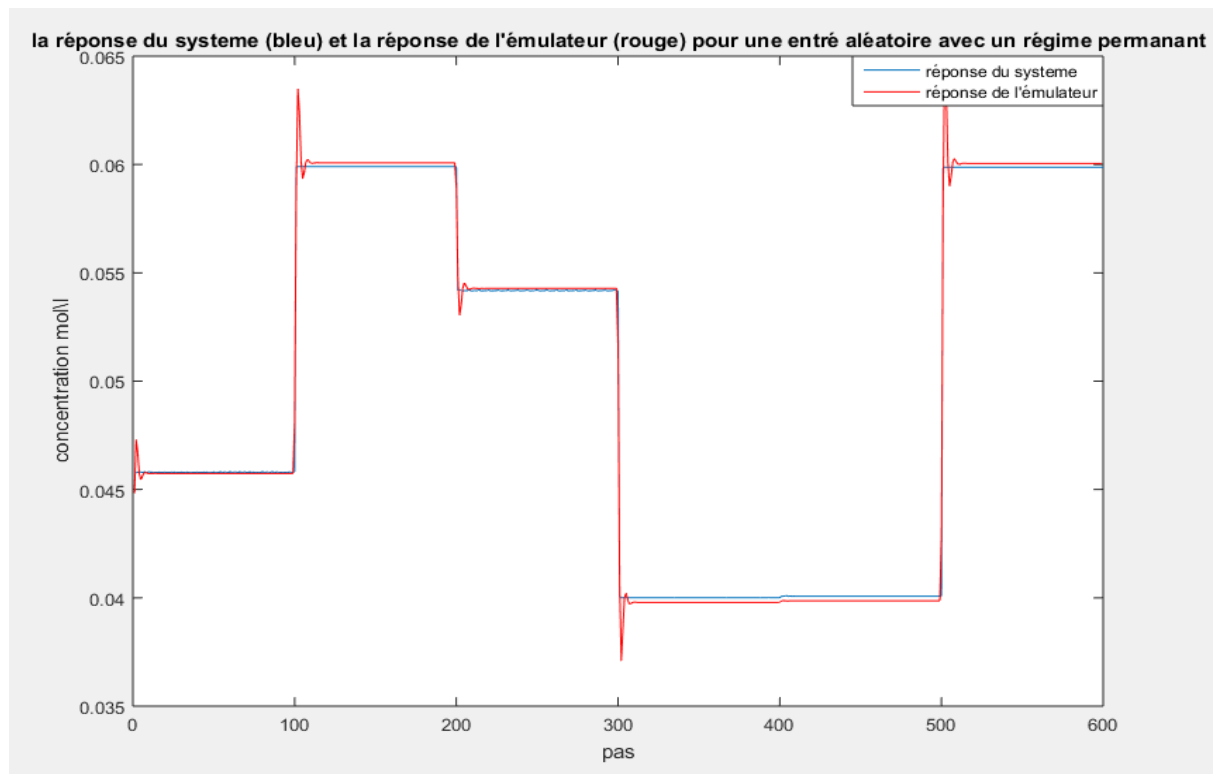
Pour l'émulateur nous avons fait plusieurs essais et nous avons conclu qu'un réseau de (4,10,1) à quatre entrées  $q_c(n)$ ,  $q_c(n-1)$ ,  $C_a(n-1)$ ,  $C_a(n-2)$  est suffisant pour émuler notre système (CSTR).

La figure 3.12 représente la réponse du système et la réponse de l'émulateur pour une entrée aléatoire.



**Figure 3.12 :** Réponse du système et celle de l'émulateur pour une entrée aléatoire.

Pour voir les choses plus claires on injecte une commande aléatoire avec un régime permanent (voir figure 3.13):



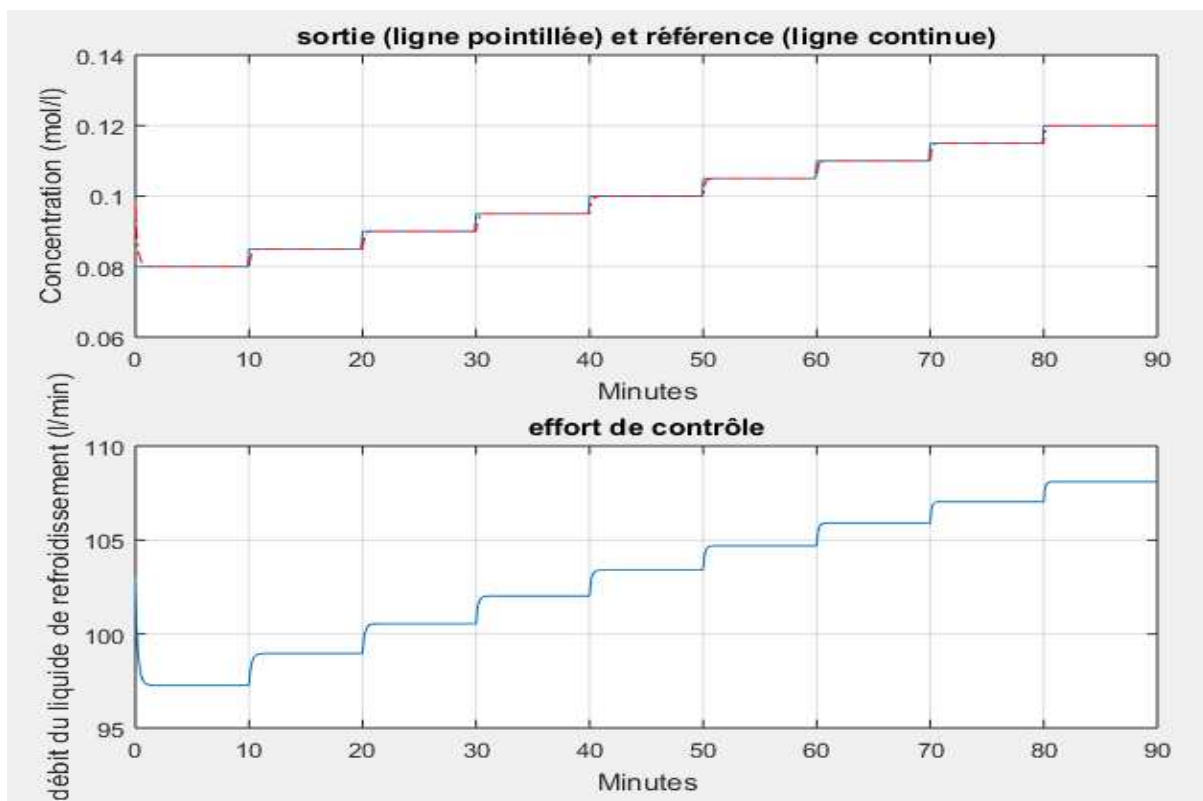
**Figure 3.13:** Réponse du système et celle de l'émulateur pour une entrée aléatoire avec un régime statique.

A partir des deux figures (3.12) et (3.13) nous remarquons que la réponse de l'émulateur suit, avec une précision acceptable, la réponse du système pour des entrées aléatoires.

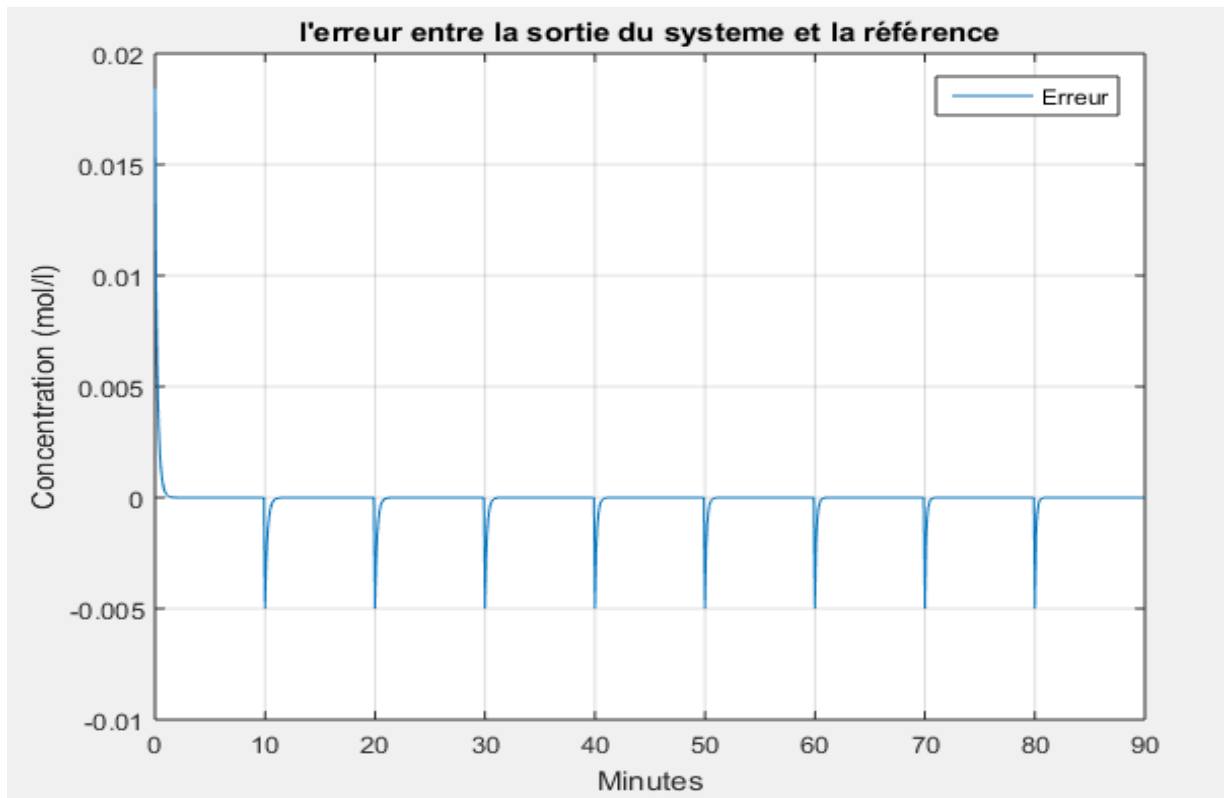
### 3.2.2.1 Simulation sans perturbations

Dans cette partie on considère  $D_e=0$ , et  $D_s=0$ .

La figure 3.14 représente la sortie du CSTR, la référence utilisée et la commande délivrée par le régulateur PID neuronale dans la deuxième partie. L'écart de poursuite est donné par la figure 3.15.



**Figure 3.14 :** Sortie du CSTR la référence et la commande délivré par le neuro-PID.

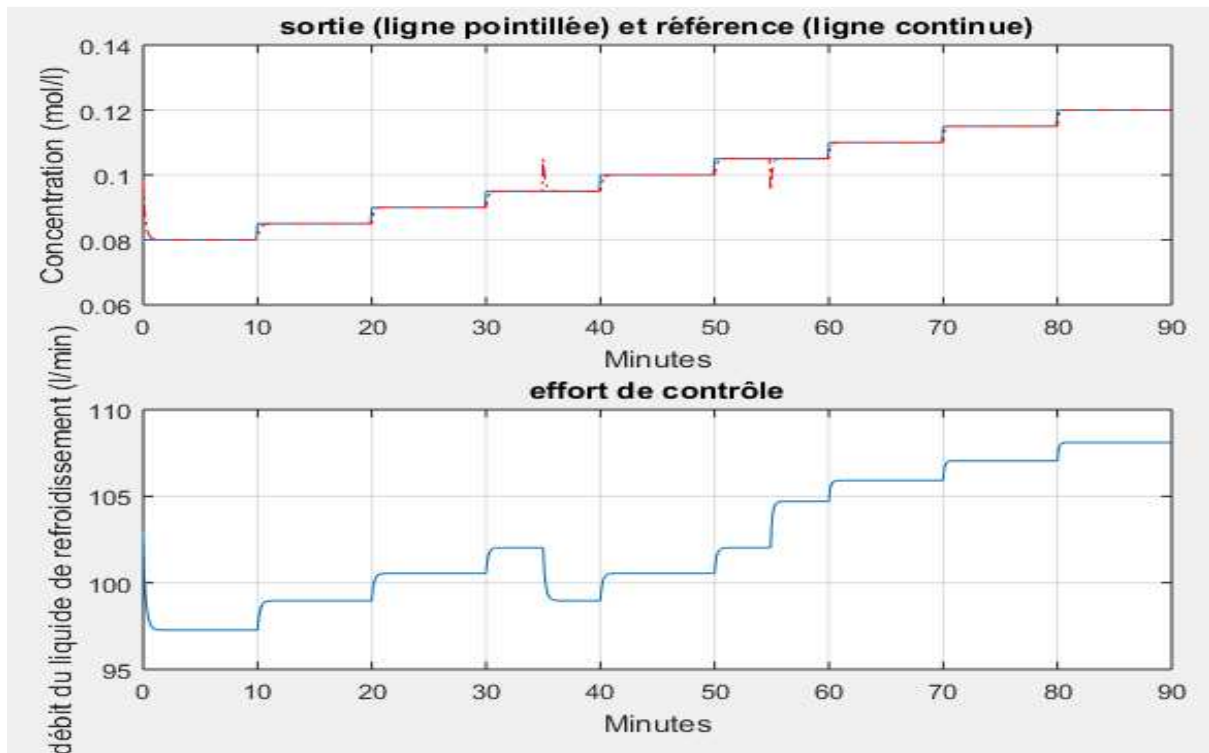


**Figure 3.15 :** Ecart entre la sortie réelle du CSTR et la référence.

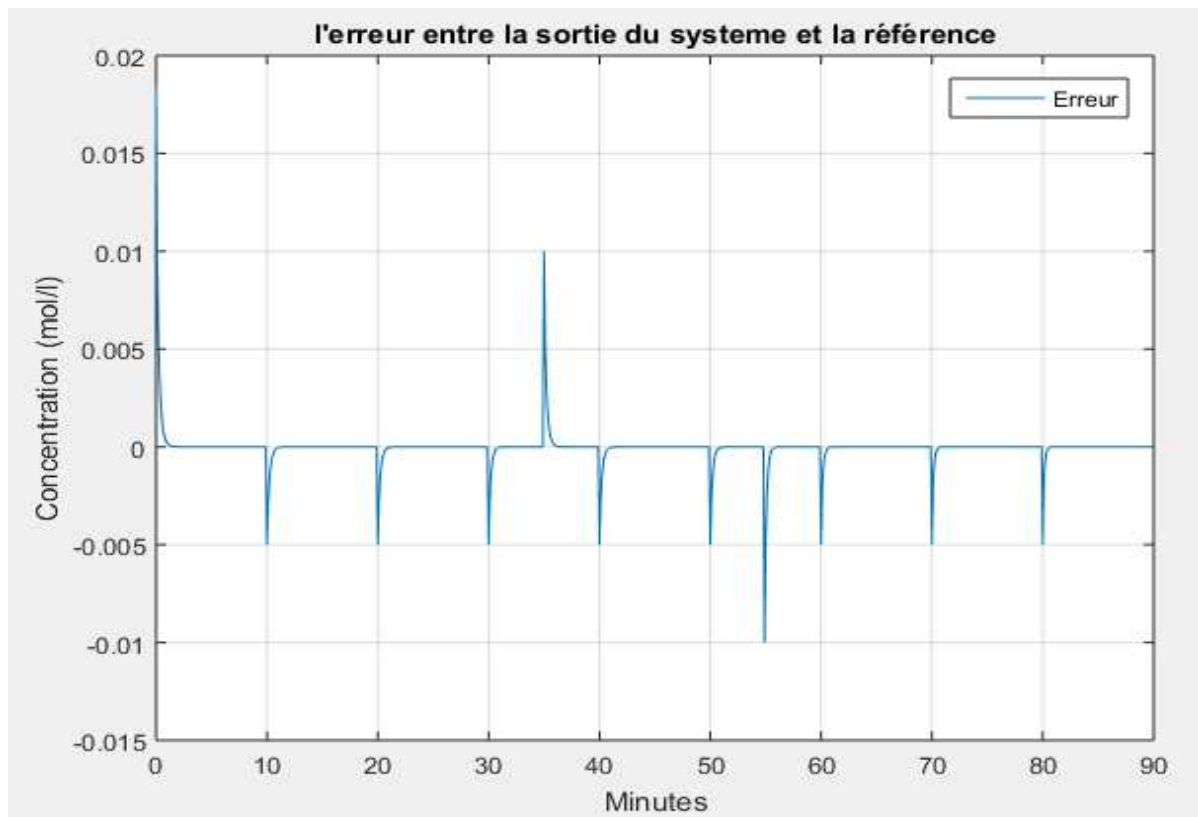
A partir des figures 3.14 et 3.15 nous pouvons conclure qu'il y a une bonne poursuite de la trajectoire de référence et que les oscillations observées dans le cas du PID classique, lorsque l'amplitude de la référence augmente, ont disparu. Nous remarquons aussi qu'il n'y a pas des oscillations rapides sur le signal de commande.

### 3.2.2.2 Simulation avec une perturbation de 10 % additive en sortie

Une perturbation de 10% additive en sortie du système est introduite durant l'intervalle [35min - 55 min]. Les résultats obtenus sont donnés par les figures 3.16 et 3.17, nous remarquons que le régulateur agit rapidement pour compenser l'effet de la perturbation.



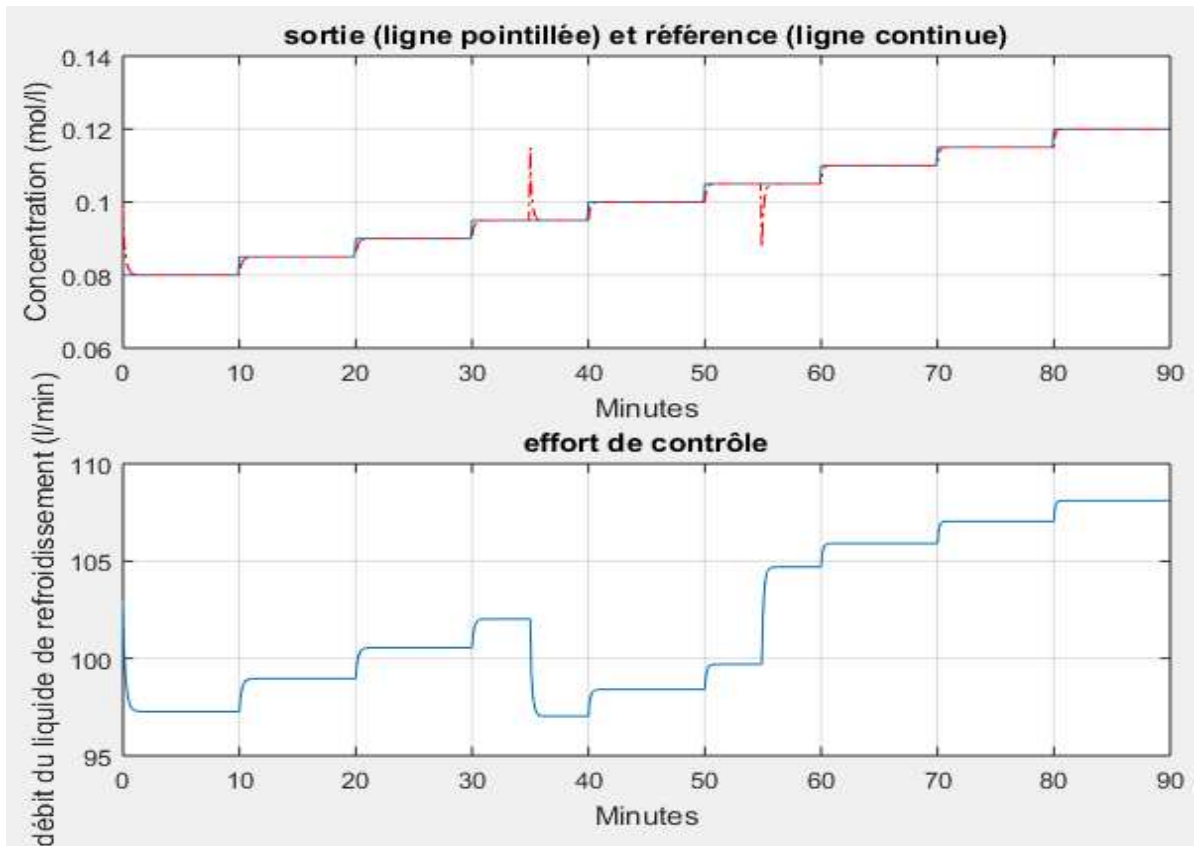
**Figure 3.16:** Sortie du CSTR avec une perturbation de 10%, la référence et le signal de commande.



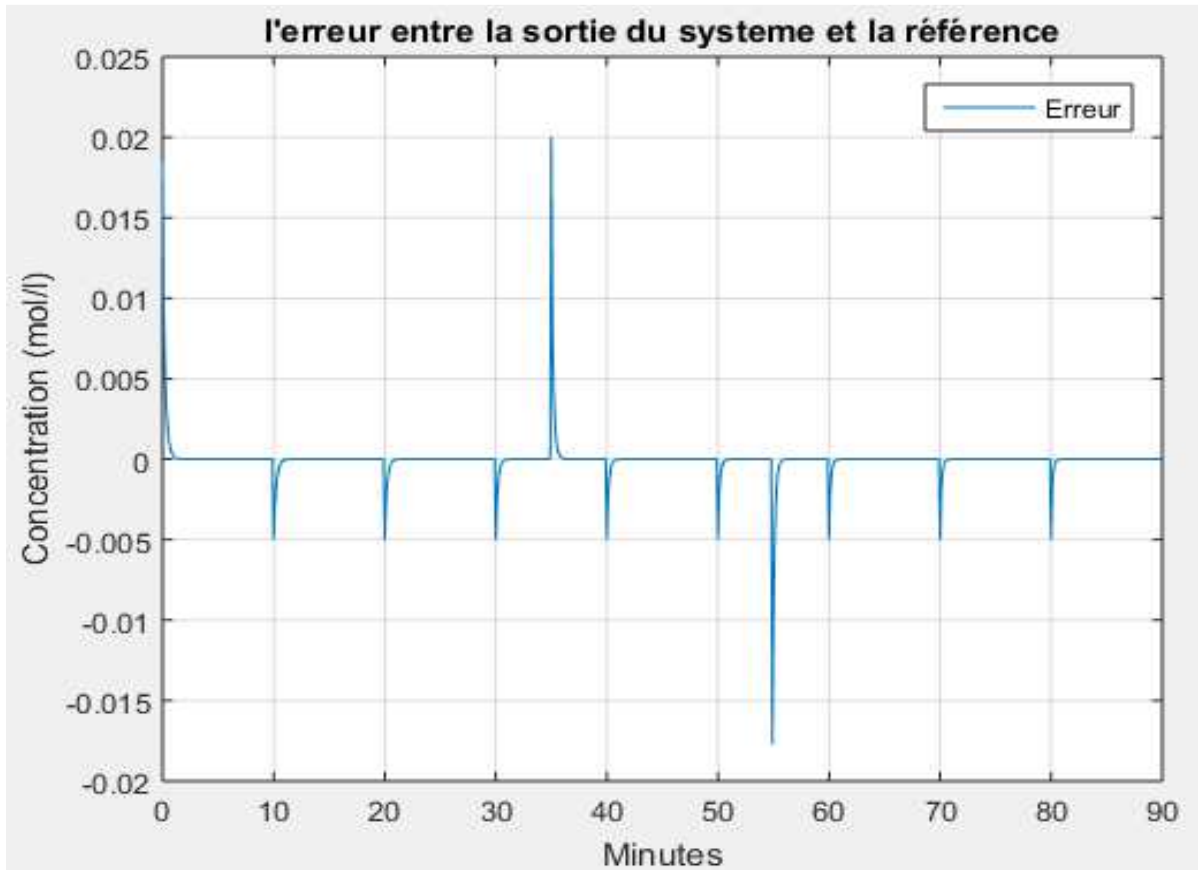
**Figure 3.17 :** Ecart entre la sortie réelle du système perturbé et la référence.

### 3.2.2.3 Simulation avec une perturbation de 5 % additive en entrée

Une perturbation de 5% additive en entrée du système est introduite durant l'intervalle [35min 55 min]. Les résultats obtenus sont donnés par les figures 3.18 et 3.19, nous remarquons que l'effet de la perturbation a été compensé par le régulateur.



**Figure 3.18 :** Sortie du CSTR avec une perturbation de 5% additive en entrée, la référence et le signal de commande.



**Figure 3.19** : Ecart entre la sortie réelle du système perturbé et la référence.

### 3.2.3 Comparaisons entre les résultats du PID classique et ceux du PID neuronal

Pour bien voir la différence entre les résultats des deux régulateurs nous avons choisi de superposer les réponses dans une seule figure et de présenter que le zoom d'une partie du tracé (figures 3.20, 3.21 et 3.22).



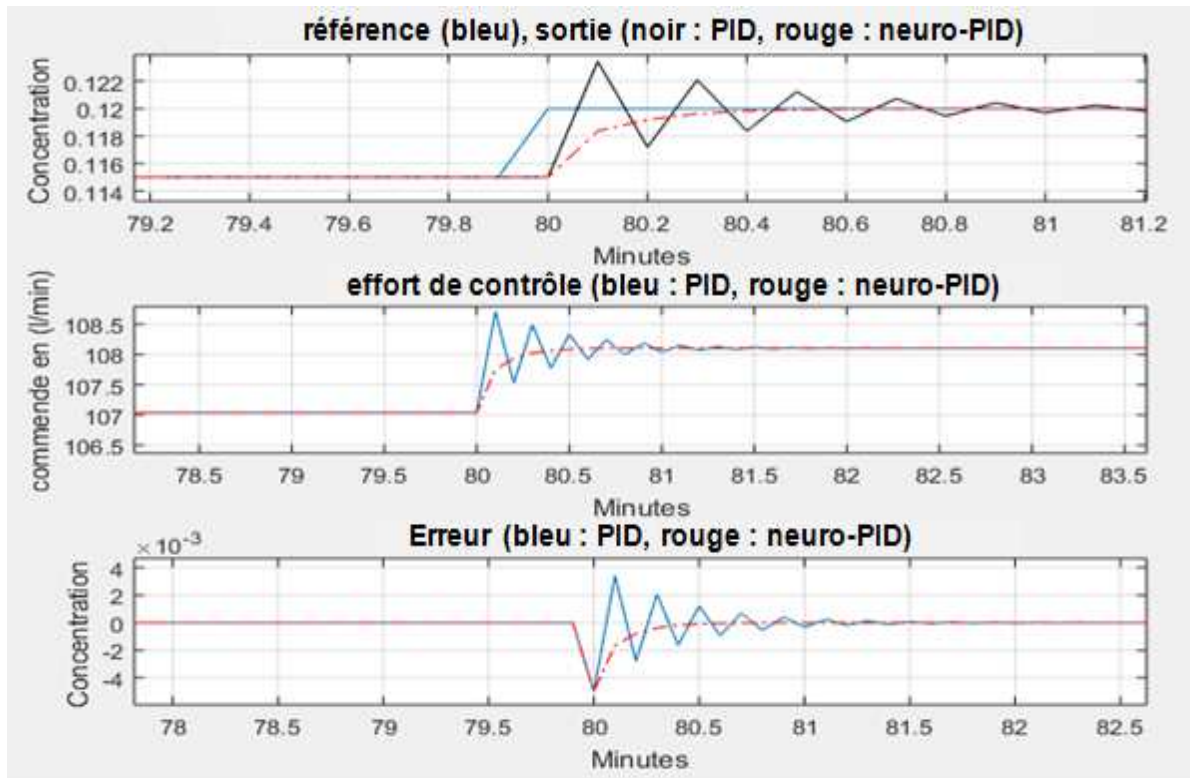


Figure 3.20 : Zoom du tracé des résultats obtenus sans perturbation

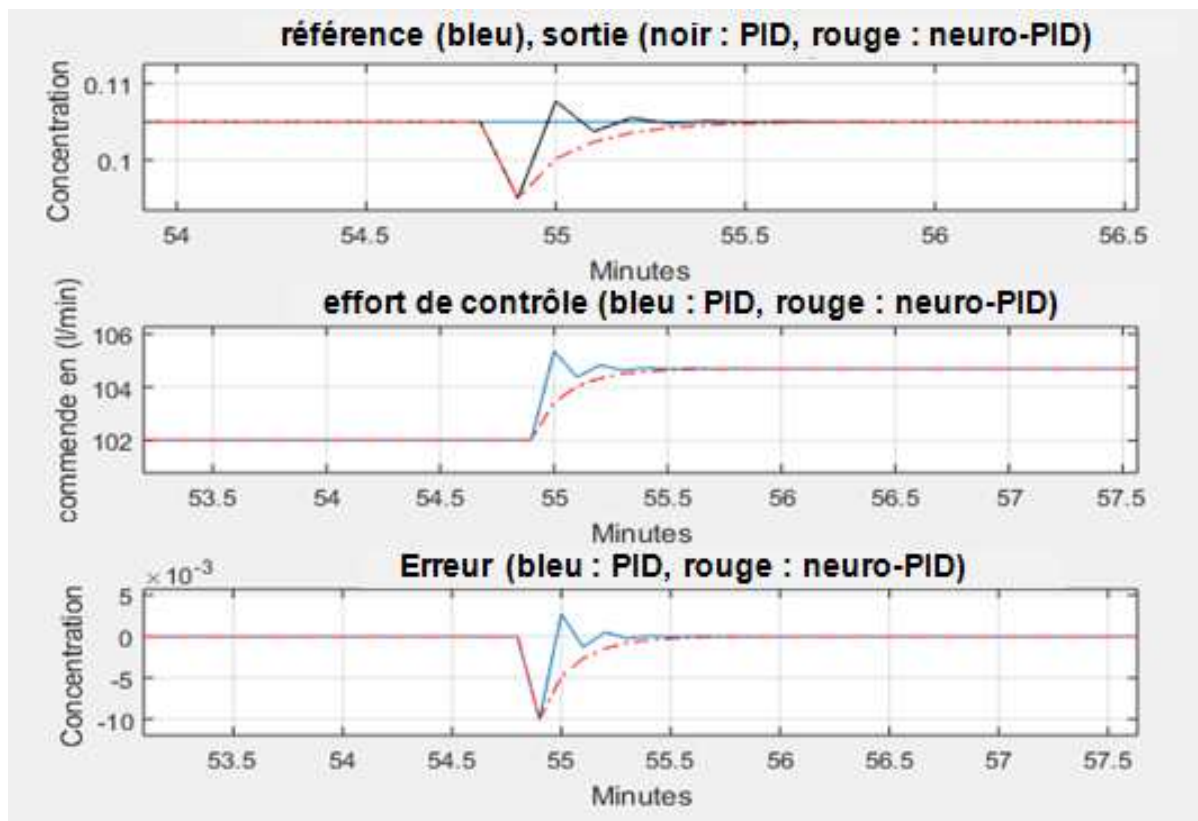
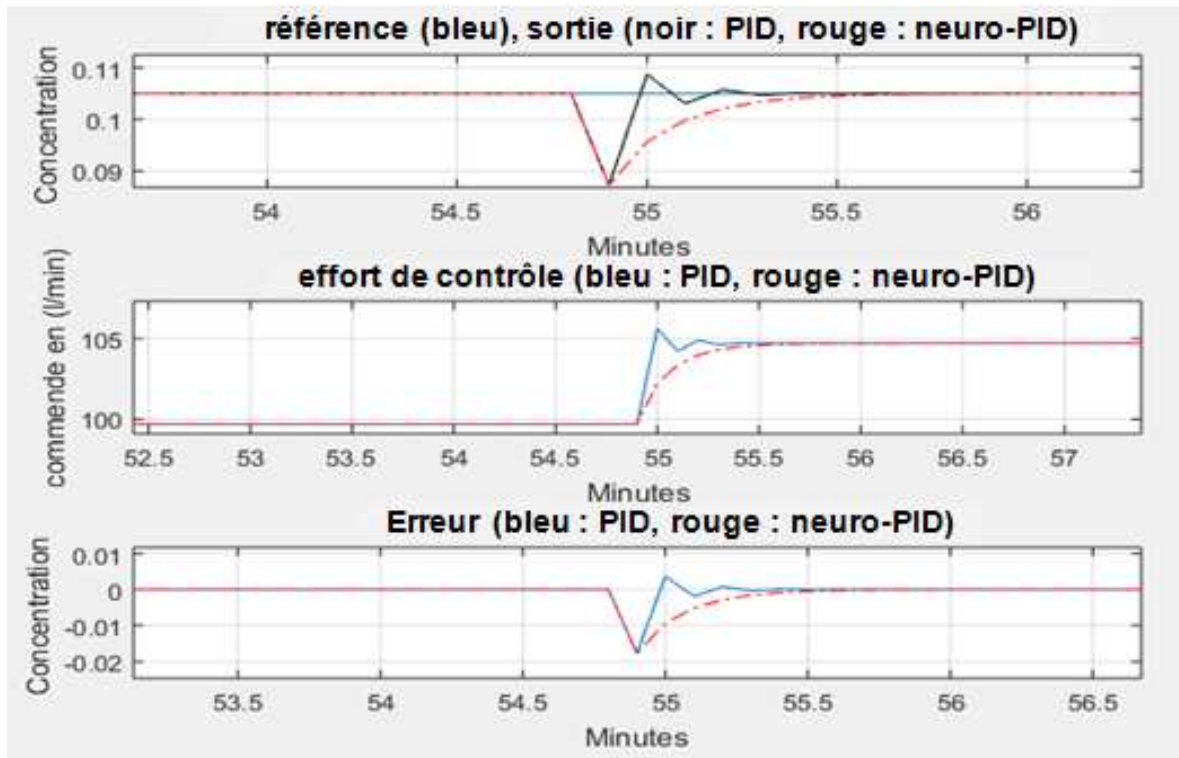


Figure 3.21 : Zoom du tracé des résultats obtenus avec une perturbation additive en sortie.



**Figure 3.22** : Zoom du tracé des résultats obtenu avec une perturbation additive en entrée.

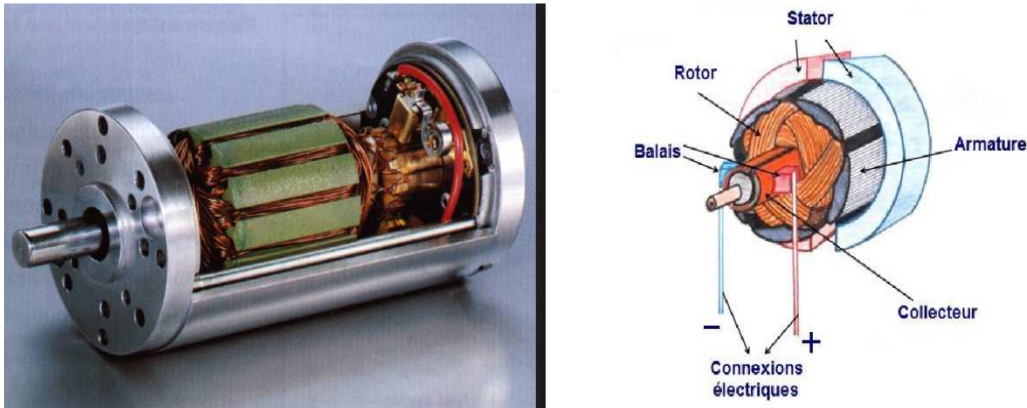
A partir des résultats obtenus, nous concluons que le PID neuronal ne présente, en aucun cas, de dépassement au contraire du PID classique où des dépassements ont été observés. L'amplitude de ces dépassements augmente avec l'augmentation de l'amplitude de référence. Le temps de réponse à 2% dans le cas du PID neuronal est inférieur à celui du PID classique. Le PID neuronal est donc plus rapide que le PID classique.

### 3.3 Moteur à courant continu

Le moteur à courant continu (MCC) est l'une des premières machines utilisées pour convertir l'énergie électrique à une énergie mécanique. Il est constitué de trois parties principales (Fig. 3.23) :

- l'inducteur (stator) : Le stator est situé dans la partie fixe du moteur, son rôle sert à créer le champ magnétique dans le moteur donc le stator est un aimant ou électroaimant (bobinage parcouru par un courant continu).
- l'induit (rotor) : Le rotor est situé dans la partie tournante du moteur, il est constitué de plusieurs bobines identiques réparties uniformément autour d'un noyau cylindrique et disposées de telle façon que leurs deux cotés coupent respectivement le flux provenant d'un pôle nord et d'un pôle sud de l'inducteur.

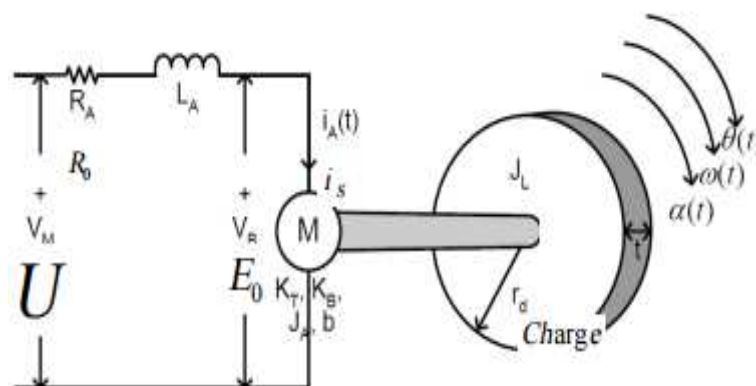
- le dispositif collecteur / balais : Le collecteur est situé dans la partie tournante du MCC, c'est un ensemble de lames de cuivre son rôle est de relier les extrémités du bobinage du rotor, les balais (charbons) sont situés dans la partie fixe du MCC leur rôle est d'alimenter les bobine du rotor en utilisant le frottement entre les lames du collecteur et les charbons.



**Figure 3.23** : Moteur à courant continu

### 3.3.1 Modélisation du MCC

La fonction de transfert d'un moteur à courant continu englobe ses caractéristiques électriques ainsi que ses caractéristiques mécaniques. Cette fonction permet de simuler la réaction du moteur pour les différents stimulants existants dans son environnement (charge, tension,...etc.). Le schéma électrique simplifié d'un MCC est donné par la figure 3.24.



**Figure 3.24** : Schéma électrique simplifié d'un moteur CC en charge.

D'après la figure 3.24 en utilisant la loi des mails on aura :

$$U(t) = i_A(t) \cdot R_0 + L_A \frac{di_A(t)}{dt} + E_0(t) \quad (3.3)$$

D'autre part, on a :

$$E_0(t) = K_B \cdot w(t) = K_B \frac{d\theta(t)}{dt} \quad (3.4)$$

Où :

$U(t)$  : la tension d'entrée.

$K_B$  : la constante de la force contre-électromotrice.

$\omega(t)$  : la vitesse de rotation angulaire du moteur en Rad/s.

$\theta(t)$  : la position angulaire du moteur en Rad.

Pour la partie mécanique, on a les équations suivantes :

$$T(t) = K_T \cdot i_A(T) \quad (3.5)$$

$$T(t) = J_T \cdot \alpha(t) + b \cdot w(t) \quad (3.6)$$

$$T(t) = J_T \cdot \frac{dw(t)}{dt} + b \cdot \frac{d\theta(t)}{dt} \quad (3.7)$$

$$T(t) = J_T \cdot \frac{d^2\theta(t)}{dt^2} + b \cdot \frac{d\theta(t)}{dt} \quad (3.8)$$

Avec :

$T(t)$  : est le couple moteur

$K_T$  : est la constante du couple moteur

$b$  : est le coefficient de frottement visqueux

$J_T$  : est l'inertie de la charge et du moteur

En utilisant la transformée de Laplace, on peut déduire la fonction de transfert reliant la position du MCC avec la tension appliqué :

$$\frac{\theta(S)}{U(S)} = \frac{K_T}{S^3 J_T L_A + S^2 (J_T R_0 + b L_A) + S (b R_0 + K_B K_T)} \quad (3.9)$$

Pour avoir la relation entre la vitesse angulaire et la tension d'entrée il suffit de multiplier la fonction de transfert (3.17\_3.9) par la variable de Laplace  $s$ :

$$\frac{\Omega(S)}{U(S)} = \frac{K_T}{S^2 J_T L_A + S (J_T R_0 + b L_A) + (b R_0 + K_B K_T)} \quad (3.10)$$

En utilisant la transformation bilinéaire, on obtient une version discrète de la fonction de transfert (3.3.10) de la forme :

$$G_{moteur}(z) = \frac{\alpha Z + \beta}{Z^2 + \gamma Z + \delta}$$

Les valeurs des coefficients  $(\alpha, \beta, \gamma, \delta)$  peuvent être obtenus directement à partir des paramètres du moteur ou par une procédure d'identification.

Dans notre travail nous avons utilisé la fonction de transfert (3.11), obtenue dans [ ] par une procédure d'identification.

$$G_{moteur}(z) = \frac{13.6 Z + 26.3}{Z^2 - 0.86 Z + 0.1} \quad (3.11)$$

A partir de (3.11) nous pouvons obtenir l'équation aux différences décrivant le moteur, elle est donnée par :

$$\omega(i) = 0.86 * \omega(i - 1) - 0.1 * \omega(i - 2) + 13.6 * u(i - 1) + 26.3 * u(i - 2) \quad (3.12)$$

### 3.3.2 Commande du moteur en utilisant un PID classique

Le schéma de commande est illustré par la figure 3.25. Nous utilisons pour le calcul des paramètres du PID la méthode de Ziegler-Nichols. La réponse indicielle du MCC, simulée par MatLab, est donnée par la figure 3.26.

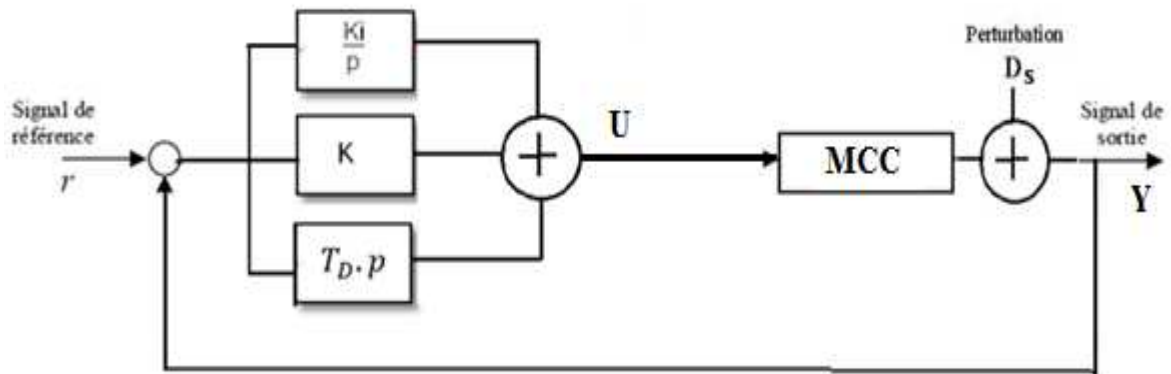


Figure 3.25 : Commande PID du MCC.

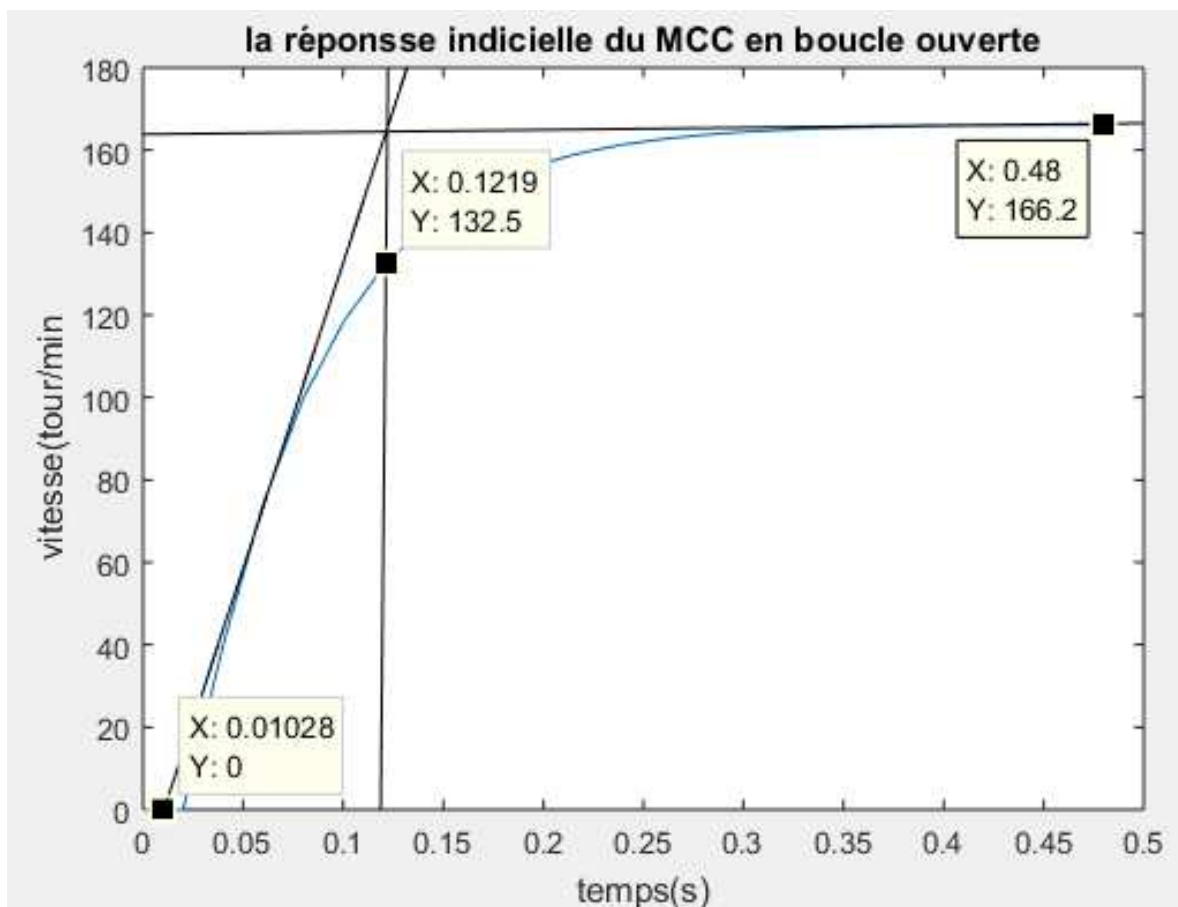


Figure 3.26 : Réponse indicielle du MCC en boucle ouverte.

A partir de la figure (3.26) nous obtenons ( $T_u = 0.01\text{ s}$ ,  $T_a = 0.1219\text{ s}$ ,  $\Delta_y = 166.2$ ,  $\Delta_u = 1$ ). En se référant au tableau 1.1 nous trouvons :

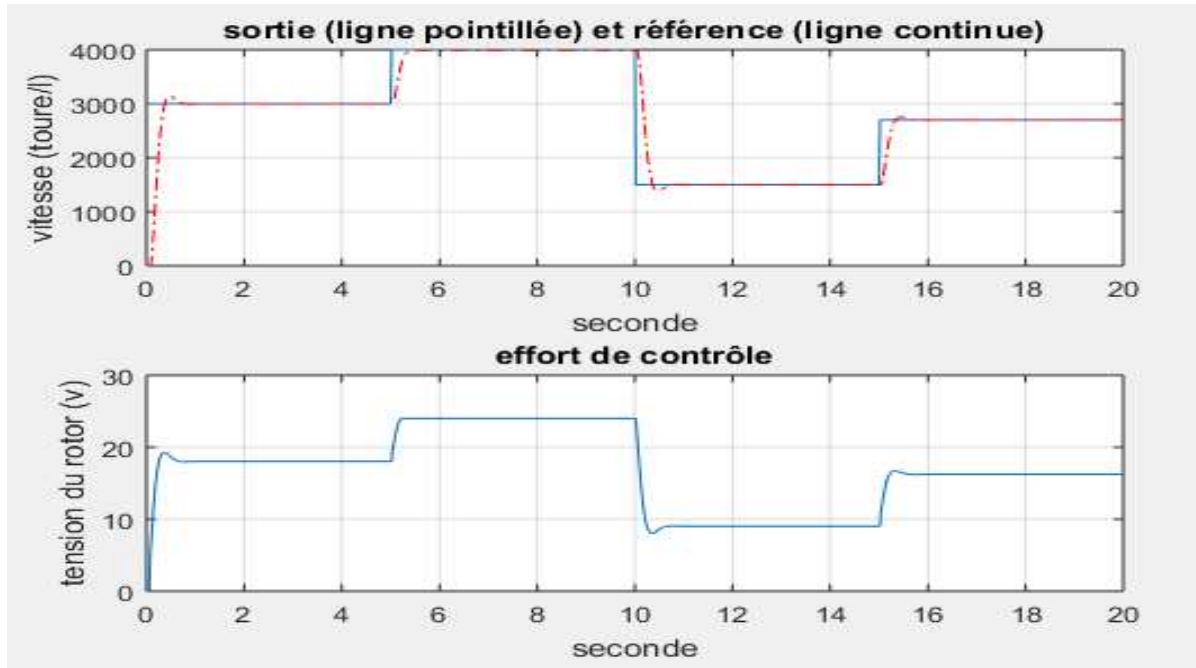
$$K_p = 0.08$$

$$T_I = 0.02$$

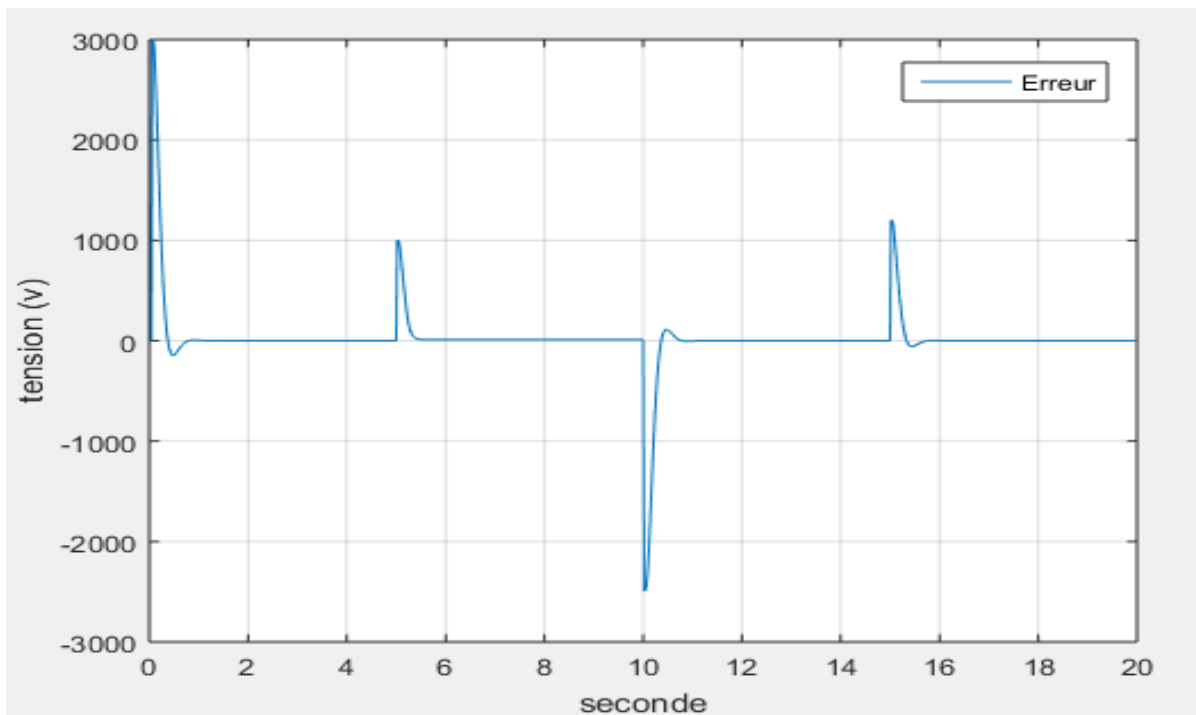
$$T_D = 0.005$$

### 3.3.2.1 Simulation sans perturbations

Dans cette partie nous prenons  $D_s=0$ . La figure 3.27 présente la vitesse de rotation du MCC, la trajectoire de référence utilisée et la commande délivrée par le PID classique. L'écart de poursuite est donné par la figure 3.28.



**Figure 3.27** : Vitesse de rotation du MCC la référence et la commande délivré par le PID.



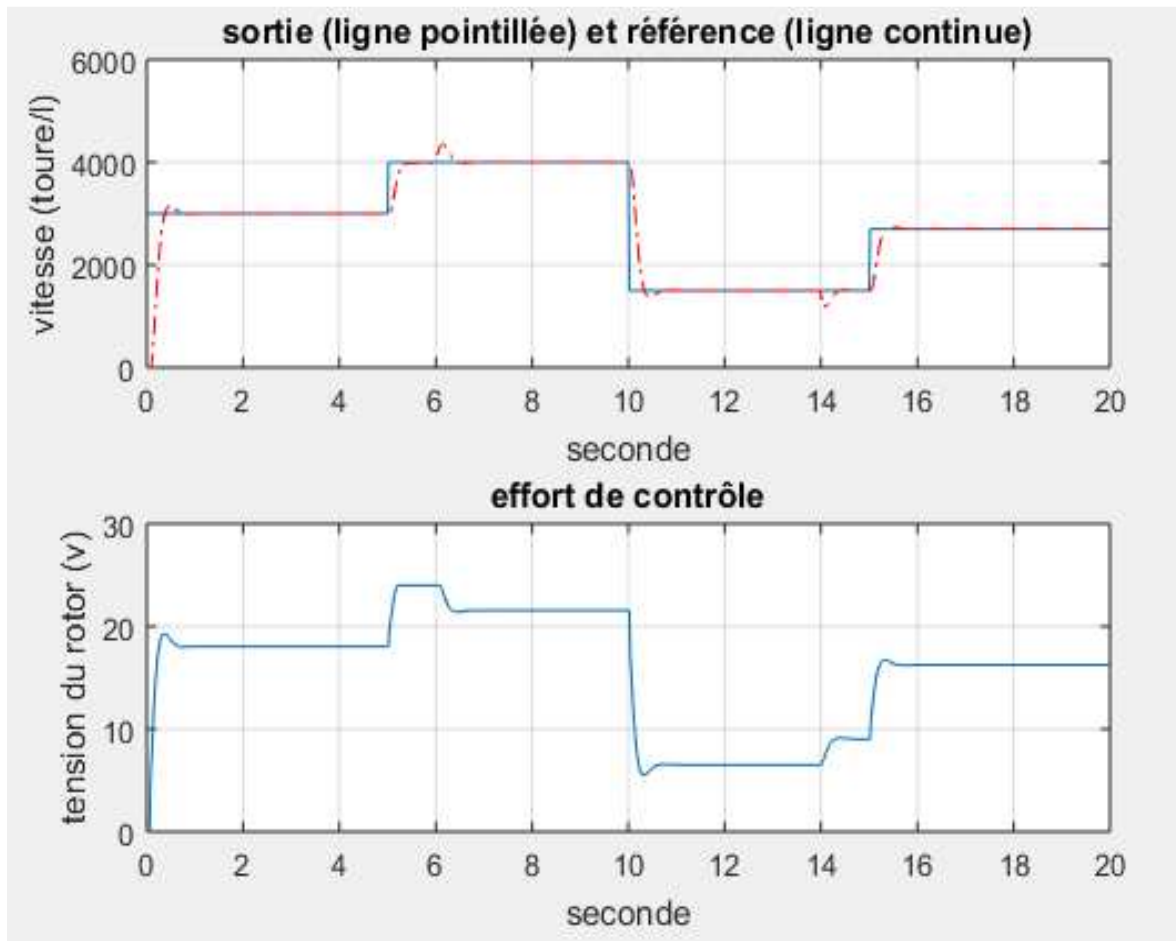
**Figure 3.28** : Ecart entre la sortie réelle du MCC et la référence.



A partir de les figures 3.27 et 3.28 nous remarquons que la vitesse du moteur suit la référence et qu'il y a des dépassements de faible d'amplitude. Le temps de réponse est de l'ordre de 620ms.

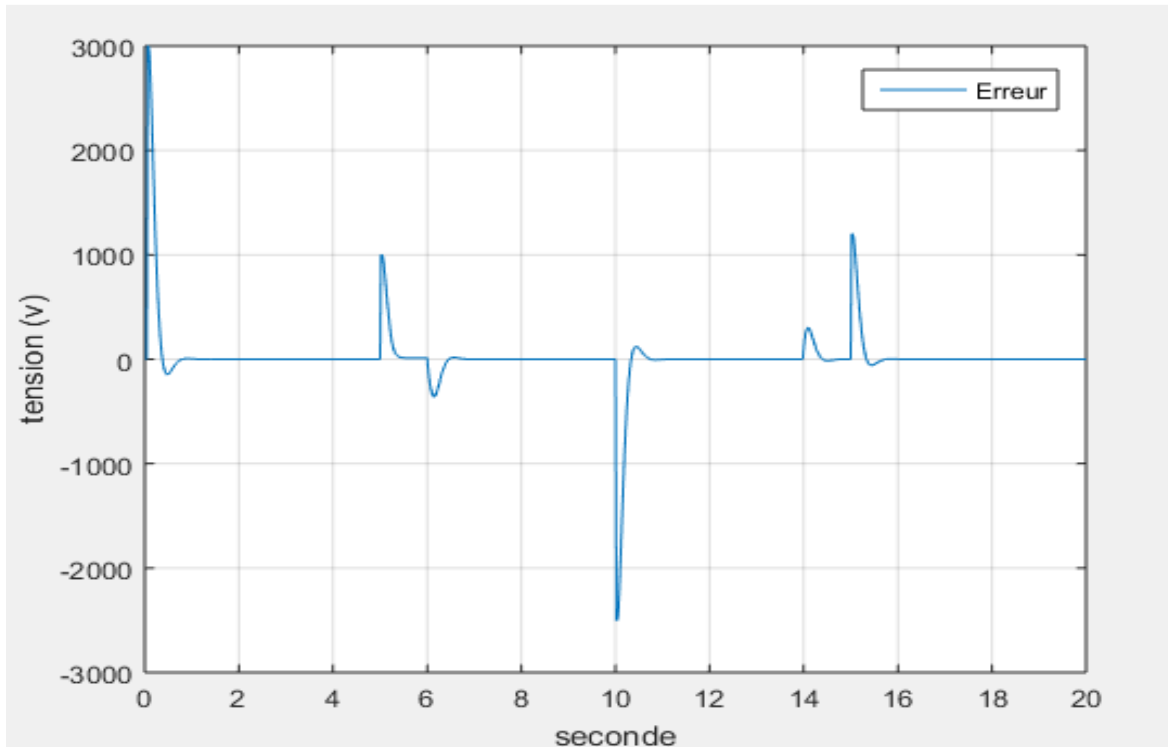
### 3.3.2.2 Simulation avec perturbation additive en sortie

Nous introduisons une perturbation de 10 % durant l'intervalle [6 s 14s]. Les résultats obtenus sont donnés par les figures 3.29 et 3.30.



**Figure 3.29** Vitesse de rotation du MCC, la référence et la commande délivrée par le PID (cas avec perturbation).



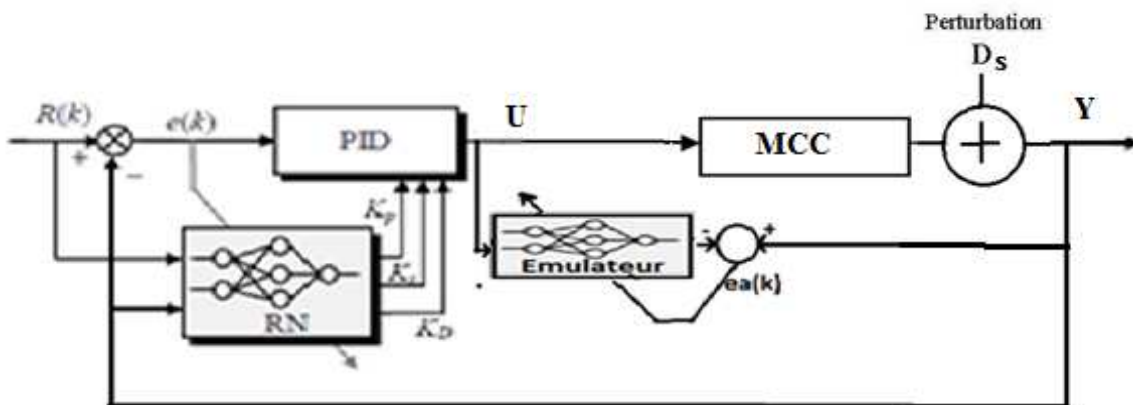


**Figure 3.30** : Ecart entre la sortie réelle du MCC et la référence (cas avec perturbation).

A partir des figures 3.29 et 3.30 nous remarquons que le PID met un temps de l'ordre de 620 ms pour compenser l'effet de la perturbation.

### 3.3.3 Commande du MCC en utilisant un PID neuronal

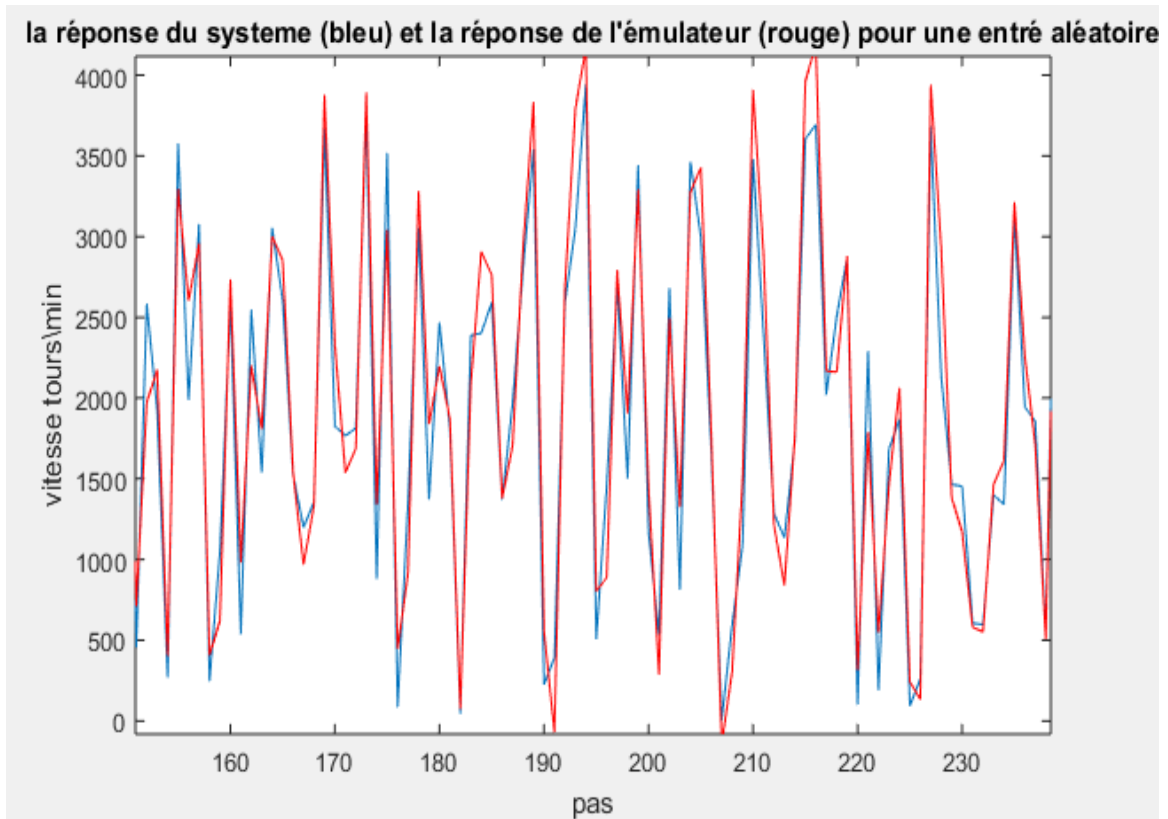
Dans cette partie nous allons présenter les résultats de la simulation de la commande en vitesse du MCC en utilisant un PID neuronal, nous considérons les deux cas sans et avec perturbation. La figure 3.31 présente le schéma fonctionnel de cette commande.



**Figure 3.31** : Commande du MCC en utilisant un PID neuronal.

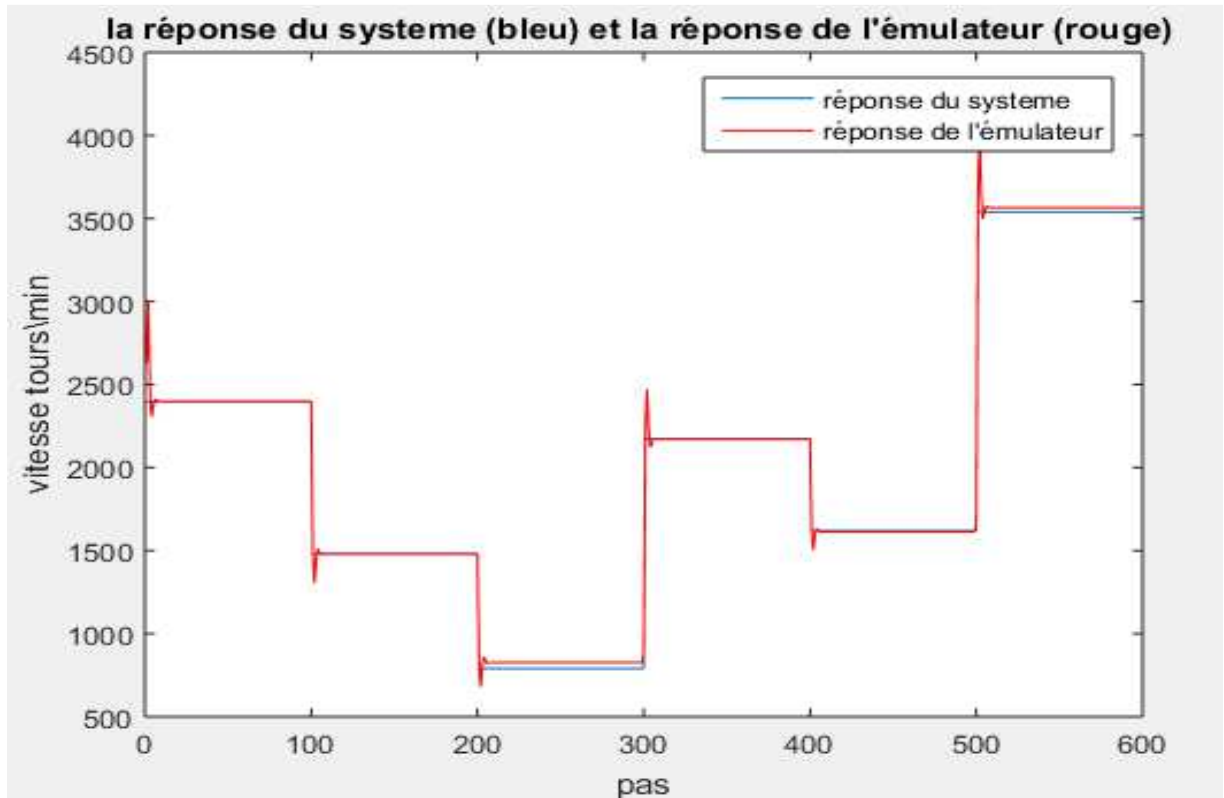
Nous procédons de la même manière utilisée dans le cas de la commande du CSTR pour faire l'apprentissage à l'émulateur et le réseau de neurones délivrant les valeurs des paramètres du PID.

La figure 3.32 représente la réponse du système et celle de l'émulateur pour une entrée aléatoire.



**Figure 3.32 :** Réponse du système et celle de l'émulateur pour une entrée aléatoire.

Pour voir monter clairement que la réponse de l'émulateur suit celle du moteur, nous avons utilisé une commande aléatoire avec un régime permanent (Fig. 3.33).

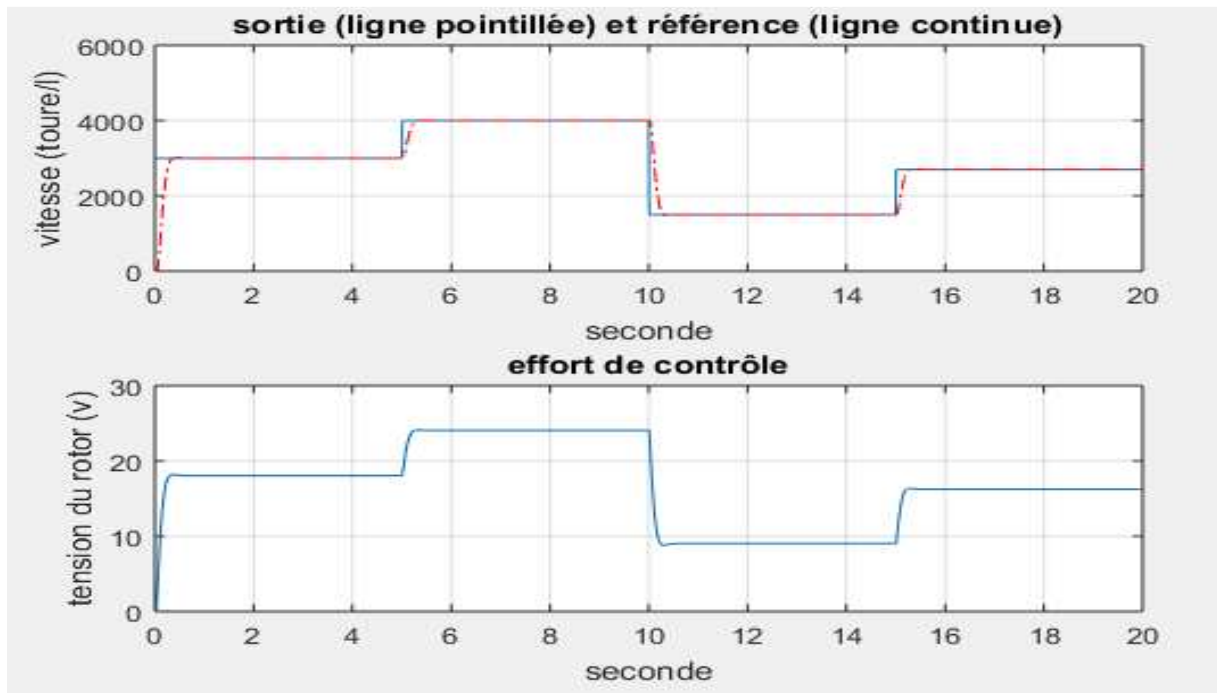


**Figure 3.33:** Réponse du système et celle de l'émulateur pour une entrée aléatoire avec un régime statique.

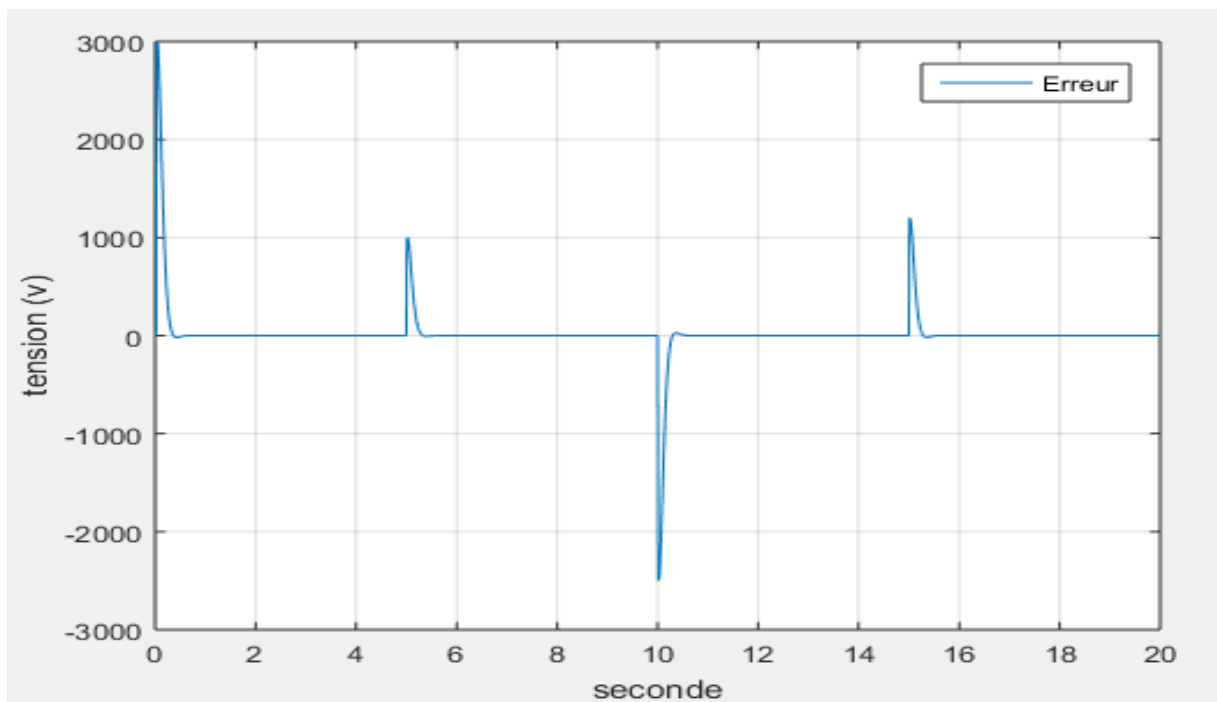
A partir des deux figures (3.32) et (3.33) nous remarquons que la réponse de l'émulateur suit celle du moteur avec une bonne précision.

### 3.3.3.1 Simulation sans perturbations

Nous prenons  $D_s=0$ . Les résultats obtenus sont indiqués par les figure 3.34 et 3.35. Nous remarquons qu'il y a une bonne poursuite de la trajectoire de référence sans dépassement et que le signal de commande est suffisamment lisse.



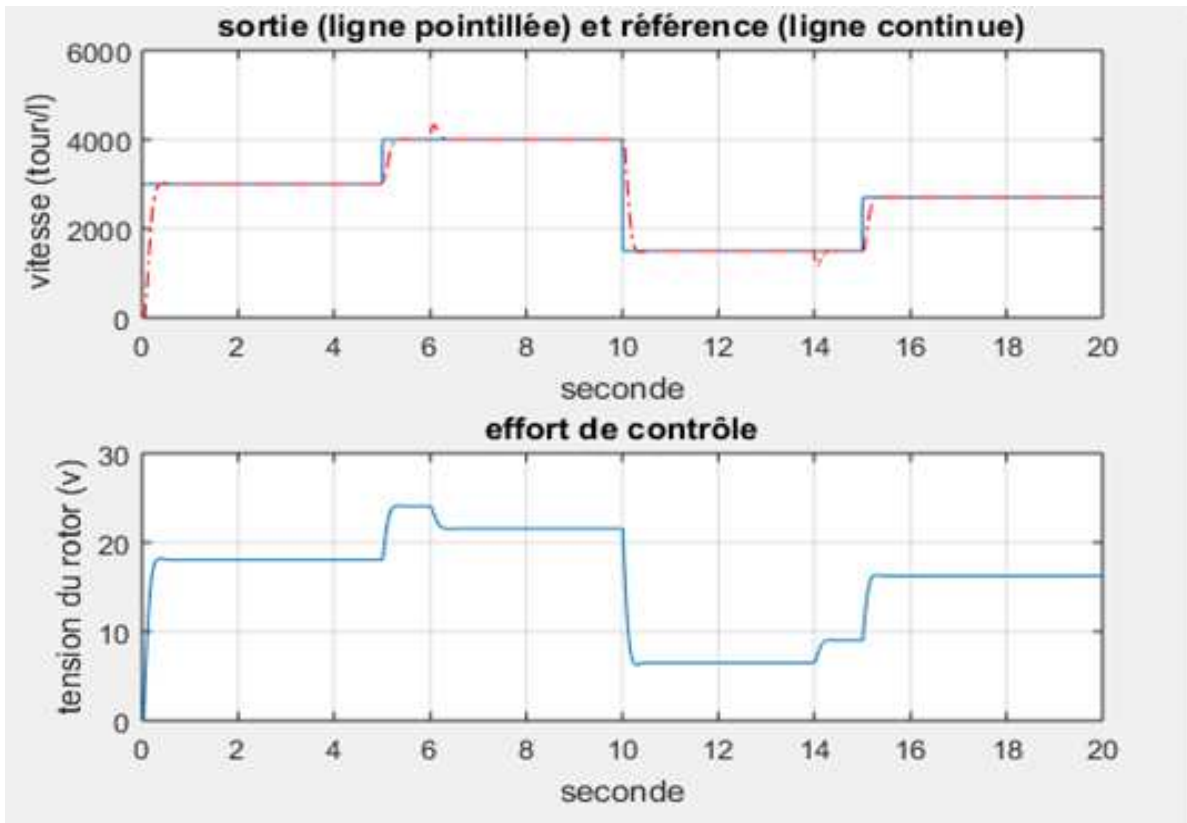
**Figure 3.34** : Vitesse de rotation du MCC, trajectoire de référence et la commande délivrée par le PID neuronal.



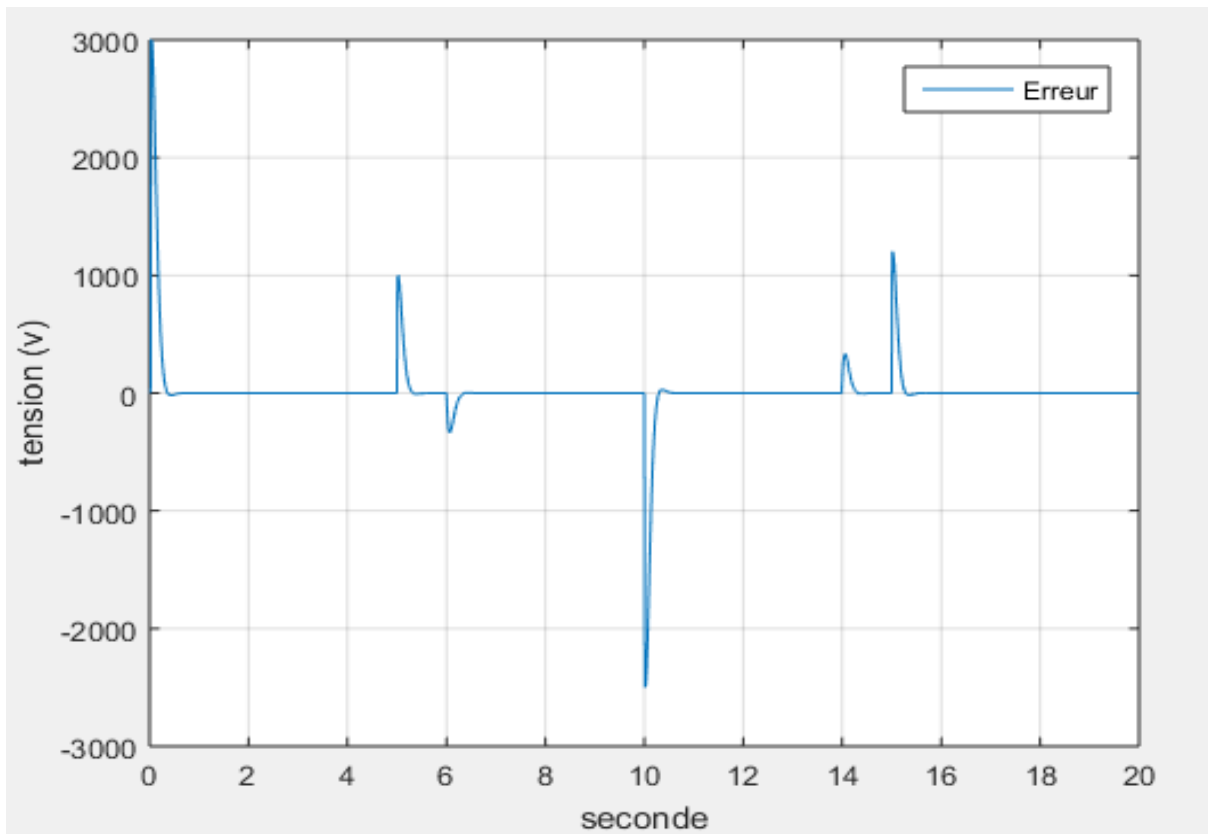
**Figure 3.35** : Ecart entre la sortie réelle du MCC et la référence.

### 3.3.3.2 Simulation avec perturbation additive en sortie

Dans cette partie nous considérons qu'il y a une perturbation de 10% additive en sortie, elle est introduite durant l'intervalle [6 s 14 s]. Les performances de commande obtenues sont illustrées par les figures 3.36 et 3.37.



**Figure 3.36 :** Vitesse de rotation du MCC, trajectoire de référence et signal de commande (cas avec perturbation)

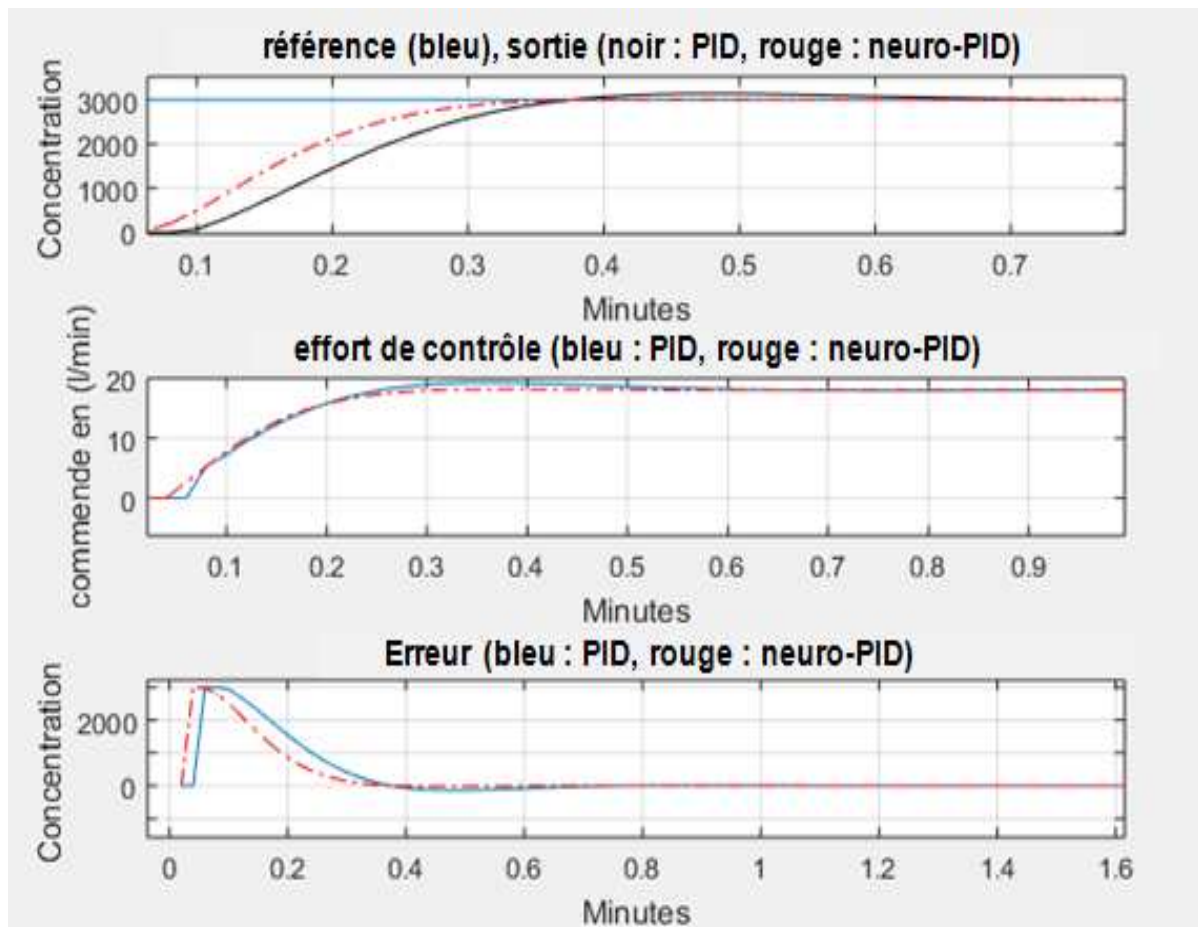


**Figure 3.37 :** Ecart entre la sortie réelle du MCC et la référence (cas avec perturbation).

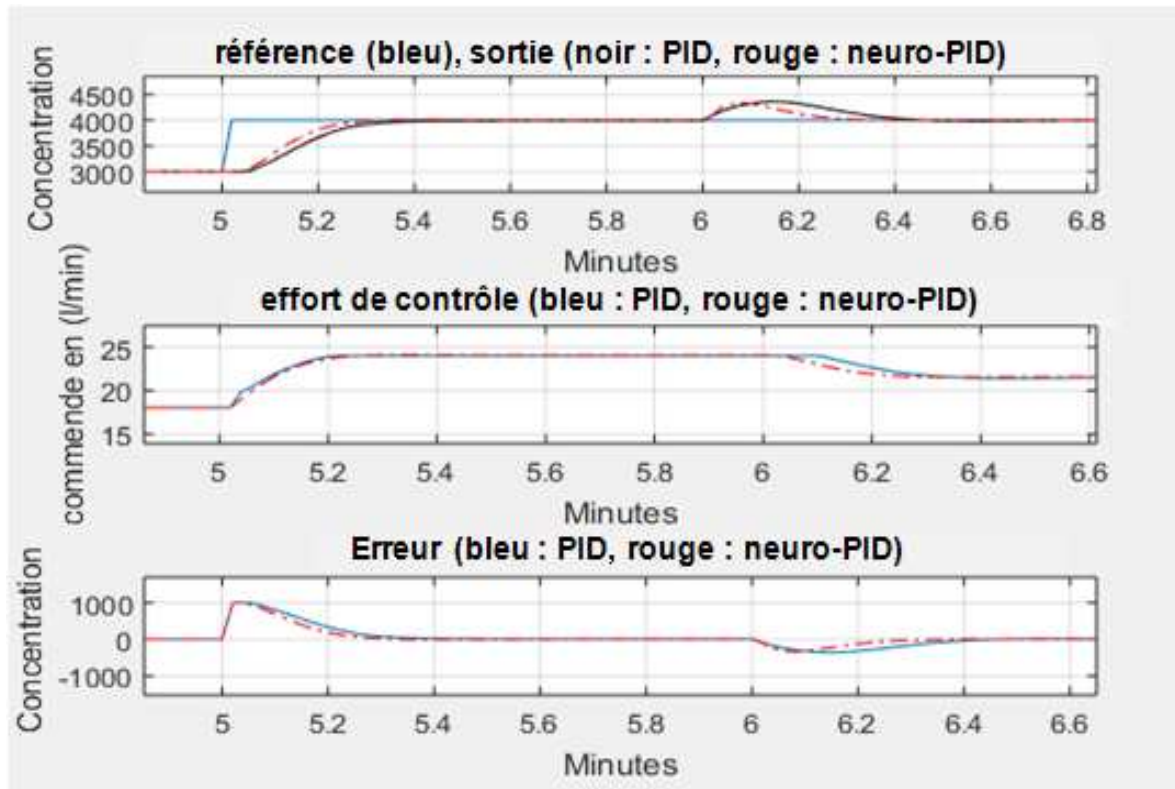
A partir de les figures 3.36 et 3.37 nous constatons que l'effet de la perturbation est rapidement (400 ms) compensé par le régulateur.

### 3.3.3.3 Comparaisons entre les résultats du PID classique et ceux du PID neuronal

Pour pouvoir comparer les performances des deux régulateurs, nous superposons les réponses dans une seule figure et nous présentons que le zoom d'une partie du tracé (figures 3.38, et 3.39).



**Figure 3.38 :** Zoom du tracé des résultats des PID et PID neuronal (sans perturbation).



**Figure 3.39** : Zoom du tracé des résultats des PID et PID neuronal (avec perturbation).

### 3.4 Conclusion

A partir des résultats obtenus nous concluons que le PID neuronal présente de meilleures performances par rapport au PID classique. Contrairement au PID classique, le PID neuronal ne présente pas de dépassements. En plus de la faculté d'auto-adaptation, qui lui permet de s'adapter aux changements dans le système et l'apparition des perturbations, le PID neuronal présente un temps de réponse inférieur à celui du PID classique. Le PID neuronal nécessite une charge de calcul plus importante qu'un PID simple.

# **Chapitre 4**

**Réalisation et essais pratique**



## 4.1 Introduction

Pour évaluer pratiquement les performances de la commande PID neuronal nous considérons la réalisation de la commande d'un moteur à courant continu. L'algorithme de commande est implémenté numériquement en utilisant la carte de développement eZdsp F2812. Nous présentons, dans ce chapitre, les différents circuits nécessaires à la mise en marche de l'application et les résultats des essais pratiques effectués.

## 4.2 Synoptique de l'application

Le schéma synoptique de la figure 4.1 illustre les différents le fonctionnement général de l'application envisagée ainsi que la disposition des différents circuits nécessaires à son mise en marche. Nous trouvons principalement les éléments suivants :

- Carte de développement eZdsp F2812
- Un circuit de conversion DC-DC (Hacheur)
- Circuit de communication série
- Circuits d'adaptation
- Circuits d'alimentation stabilisée
- Circuit d'isolation galvanique

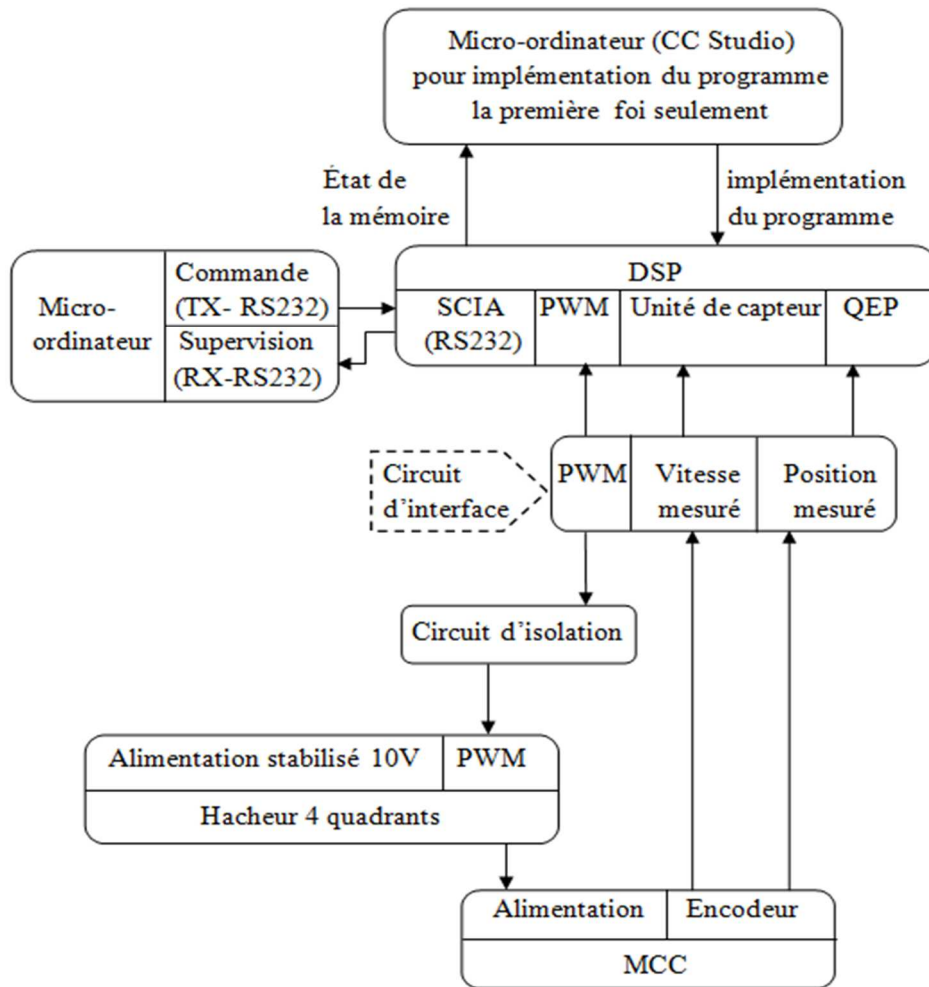


Figure 4.1 : Schéma synoptique du dispositif réalisé.

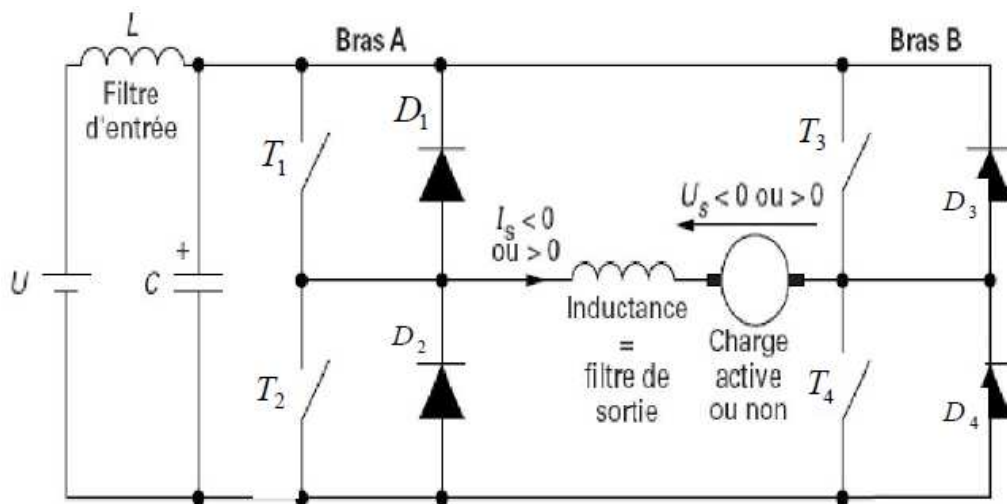
### 4.3 Hacheurs

Les hacheurs sont des convertisseurs continu-continu (DC-DC), leur rôle est de fournir une tension continue à une valeur moyenne réglable à partir d'une tension continue fixe, leurs applications sont nombreuses par exemple la commande en vitesse des MCC, les alimentations à découpage ... etc.

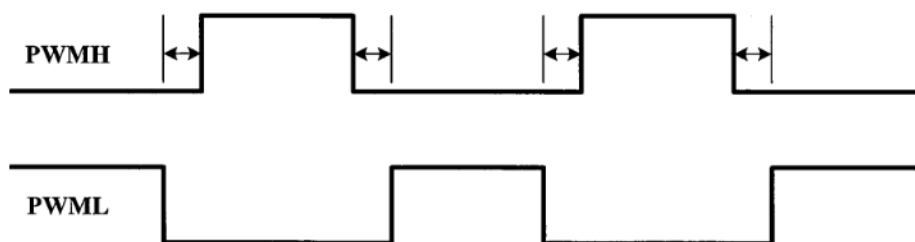
Le principe de fonctionnement d'un hacheur est simple, il consiste à établir une connexion source-charge pendant un laps de temps ( $\alpha T$ ), ensuite il coupe cette connexion pendant un laps de temps ( $(1 - \alpha)T$ ) et il refait ça périodiquement (chaque période  $T$ ). Il existe plusieurs types de hacheurs : hacheur série (ou abaisseur), hacheur parallèle (ou élévateur) et hacheur à quatre quadrants. Dans notre application, pour pouvoir faire tourner le moteur dans les deux sens, nous utilisons le hacheur à quatre quadrants.

### 4.3.1 Hacheur à quatre quadrants

Le schéma de principe d'un hacheur à quatre quadrants est illustré par la figure 4.2, où les interrupteurs T1, T2, T3, et T4 sont commandés par deux signaux MLI (modulation de largeur d'impulsion) ou PWM (Pulse Width Modulation) complémentaires avec une zone morte (Fig.4.3). Lorsque les interrupteurs T1 et T4, commandés par le premier signal PWM, sont fermés, les interrupteurs T2 et T3, commandés par le deuxième signal PWM, doivent être ouverts et inversement.



**Figure 4.2 :** Schéma structurel d'un hacheur à quatre quadrants.

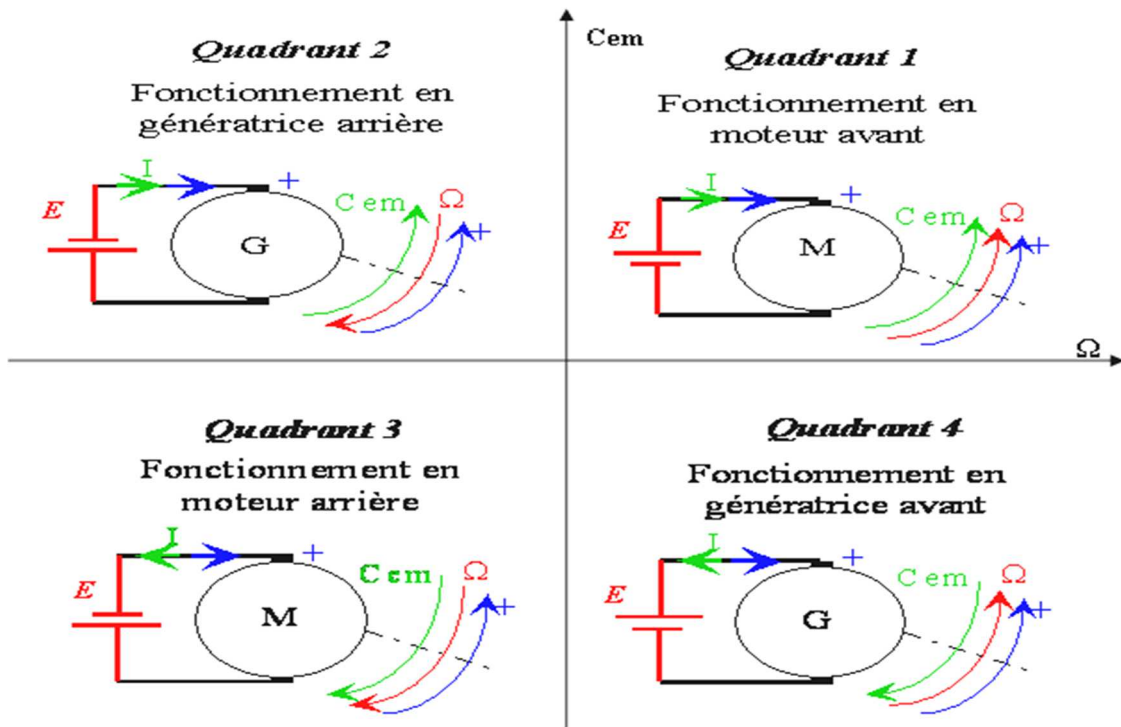


**Figure 4.3 :** Signal PWM et son complément avec une zone morte.

Un hacheur de ce type offre la possibilité de :

- ✓ Varier la valeur moyenne de la tension aux bornes du moteur (variation de la vitesse).
- ✓ Varier le sens de passage du courant dans le moteur (variation du sens du couple).

La figure 4.4 montre les différents modes de fonctionnement du moteur à courant continu dans les quatre quadrants du plan Couple-Vitesse.



**Figure 4.4 :** Modes de fonctionnement dans les quatre quadrants.

La valeur moyenne à la sortie du hacheur est donnée par :

$$V_{s0} = (1 - 2 \alpha)U \quad (4.1)$$

A partir de l'équation (4.1) on remarque que si le rapport cyclique  $\alpha$  est inférieur à 0.5 la tension moyenne observée par le moteur est positive et s'il est supérieur à 0.5 la tension moyenne observée par le moteur est négative. Pour faire varier le sens de rotation du moteur on doit donc, jouer seulement sur la valeur du rapport cyclique.

### 4.3.2 Réalisation du hacheur a quatre quadrants

La réalisation de notre hacheur est basée sur le circuit intégré L6203 (annexe C1). Ce dernier est un hacheur à quatre quadrants, il peut éprouver jusqu'à 48V et 5A (240VA). Le moteur que nous avons utilisé absorbe un courant maximal de 1.5A et une tension DC maximale de 10V.

La fréquence de commutation utilisée dans notre projet est de l'ordre de 10 KHz. Le schéma électrique du hacheur réalisé est illustré par la figure 4.5.

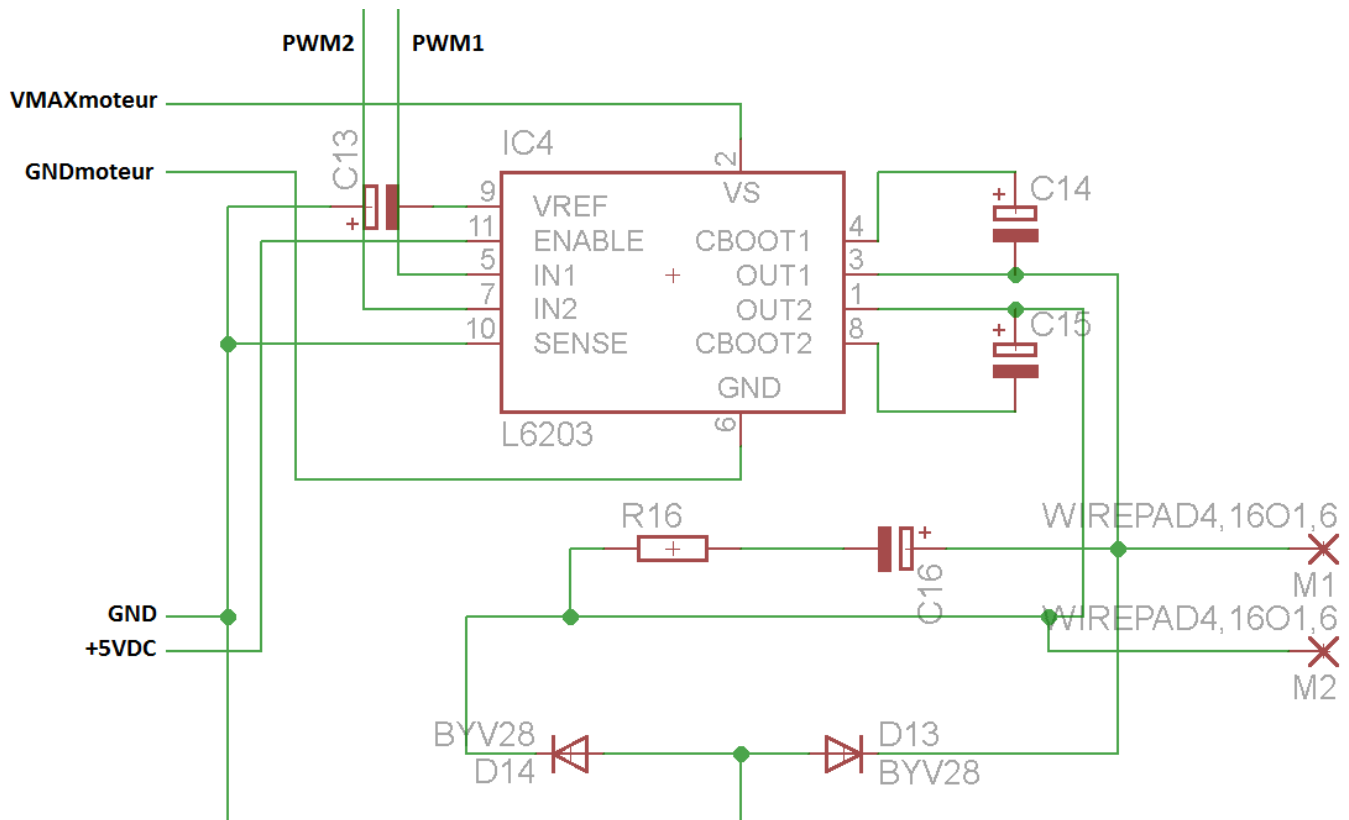


Figure 4.5 : Schéma électrique du hacheur réalisé.

#### 4.4 Communication série (RS232)

Le standard de transmission de données séries entre équipements a été développé dans les années 60 par l'EIA (Electronic Industries Association). Il était destiné pour la transmission de données de type texte ASCII entre les systèmes numériques et les modems [12].

Tous les systèmes de transmission numérique par la liaison série RS232 sont organisés de la manière illustrée dans la figure 4.6.



**Figure 4.6 :** Schéma fonctionnel de la liaison série RS232.

Le système numérique contient un processeur qui stocke et envoie les données sous forme brute. Il existe un circuit de reconstitution des trames RS232 dont le rôle est de sérialiser les données à envoyer en respectant le protocole RS232.

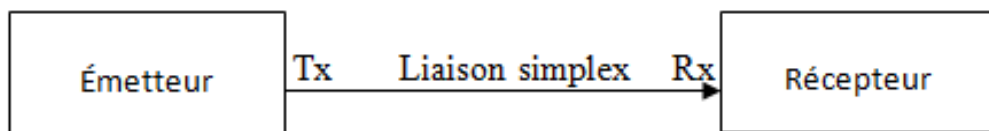
#### 4.4.1 Les niveaux de tension imposés

La liaison série RS232 exploite des signaux à deux niveaux +/- 12v

- + 12 V : niveau logique "0".
- - 12 V : niveau logique "1".

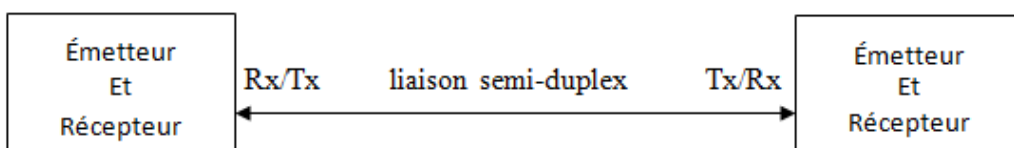
#### 4.4.2 Les modes de transmission RS232

- Simplex : Mode de transmission unidirectionnel



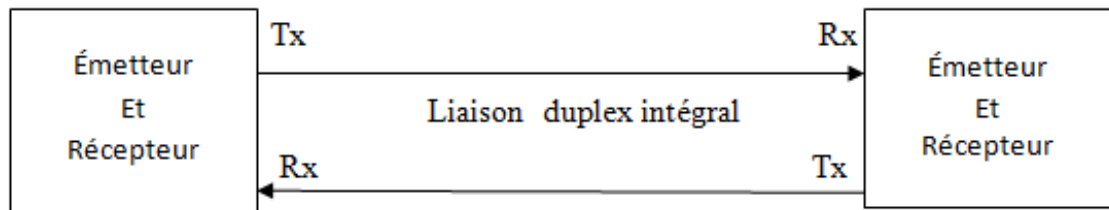
**Figure 4.7:** Liaison simplexe entre un émetteur et un récepteur.

- Semi-duplex (ou half-duplex) : Est un mode de transmission bidirectionnel, où un seul dispositif peut émettre à la fois.



**Figure 4.8 :** Liaison semi-duplex entre deux systèmes.

- Duplex intégral (ou full-duplex) : Est un mode de transmission bidirectionnel. Les deux dispositifs peuvent émettre en même temps.



**Figure 4.9:** Liaison duplex intégral entre deux systèmes.

Pour plus d'information sur les modes de communication voir [13].

Le problème avec la liaison RS232 est l'adaptation de l'alimentation, puisque le DSP exploite des tensions de 0/+3.3v et l'ordinateur exploite des tensions de -12/+12 donc on doit utiliser un circuit d'adaptation entre l'ordinateur et le DSP.

#### 4.4.3 Le circuit d'adaptation

Pour faire l'adaptation nous utilisons le circuit intégré MAX232 (annexe C2). C'est un composant créé par MAXIM qu'on peut trouver sous d'autres références chez d'autres fabricants. Il sert d'interface entre une liaison série TTL (0-5V) et une liaison série RS232 (+12 -12V) en utilisant une simple alimentation stabilisé de 3.3V. (Voir Figure 4.10)

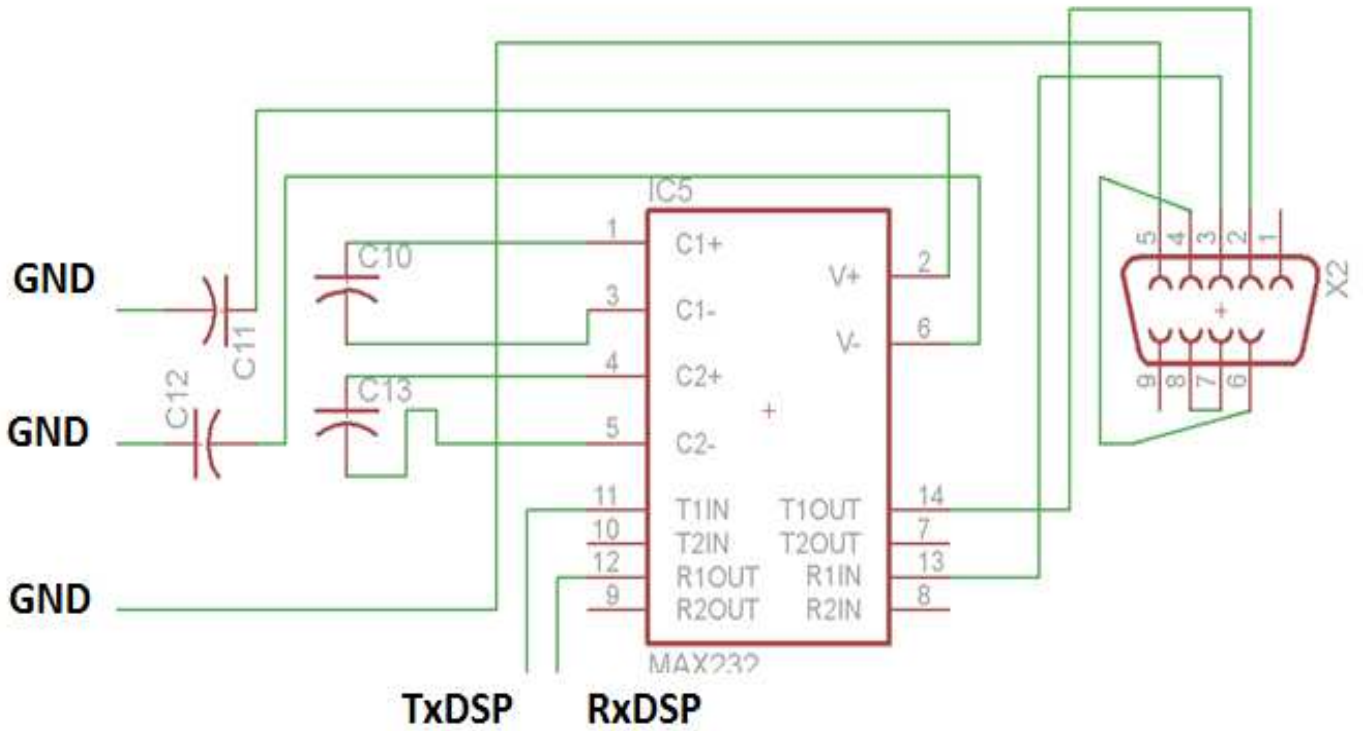


Figure 4.10 : Schéma électrique du circuit d'adaptation.

Pour alimenter ce circuit on doit avoir une alimentation stabilisé de 3.3V. Nous avons réalisé cette alimentation (Fig. 4.11) en utilisant le régulateur de tension LM317.

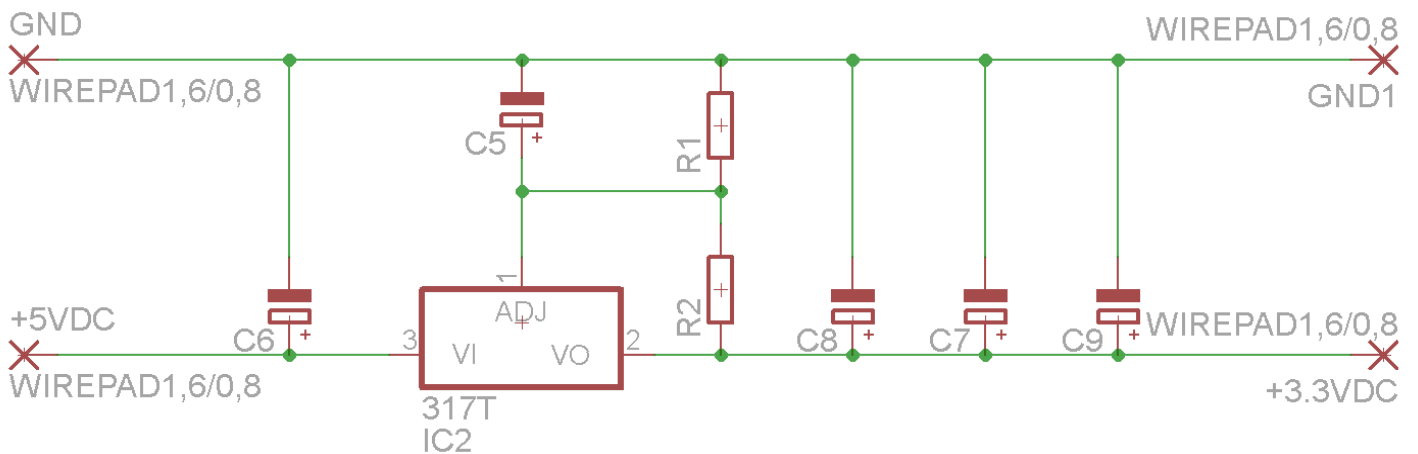


Figure 4.11 : Schéma électrique de l'alimentation stabilisé 3.3V.



La tension à la sortie du LM317 est régie par l'équation :

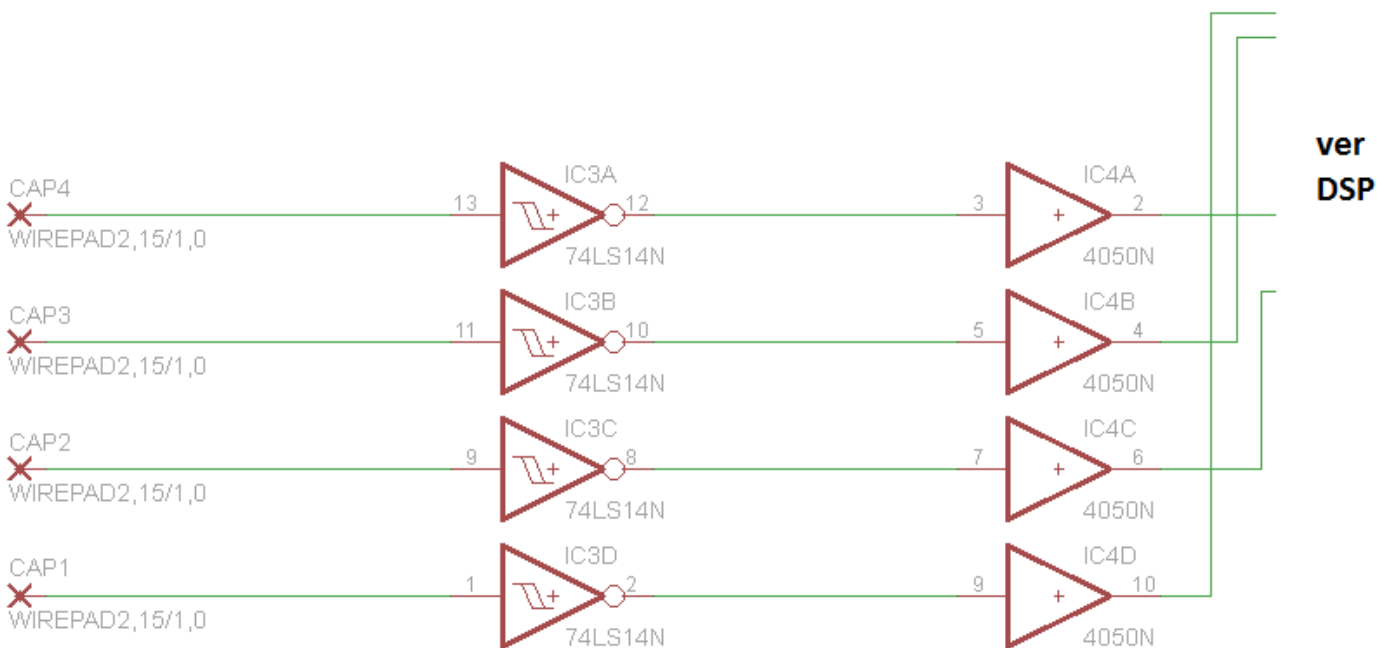
$$V_{out} = V_{ref} \left( 1 + \frac{R_2}{R_1} \right) + I_{ADJ} * R_2$$

Avec:

$$I_{ADJ} = \frac{V_{ref}}{R_1} \quad \text{et} \quad V_{ref} = 1.25V$$

#### 4.5 Entrées de l'unité de capture du DSP (CAP)

Les signaux obtenus à la sortie de l'encodeur sont envoyés à l'unité capture du DSP, pour calculer la vitesse de rotation du moteur, à travers un circuit d'interface et de mise en forme. Ce circuit est constitué par les circuits intègres HEF4050B (Non-Inverting buffers) (annexe C3) et les Trigger de Schmidt SN74LS14 (annexe C4). Le circuit HEF4050B permet de dégager à la sortie une tension égale à sa tension d'alimentation (3.3V) dès qu'il reçoit des tensions de 5V ou plus à son entrée. La figure 4.12 montre le schéma électrique de ce circuit d'interface et de mise en forme.

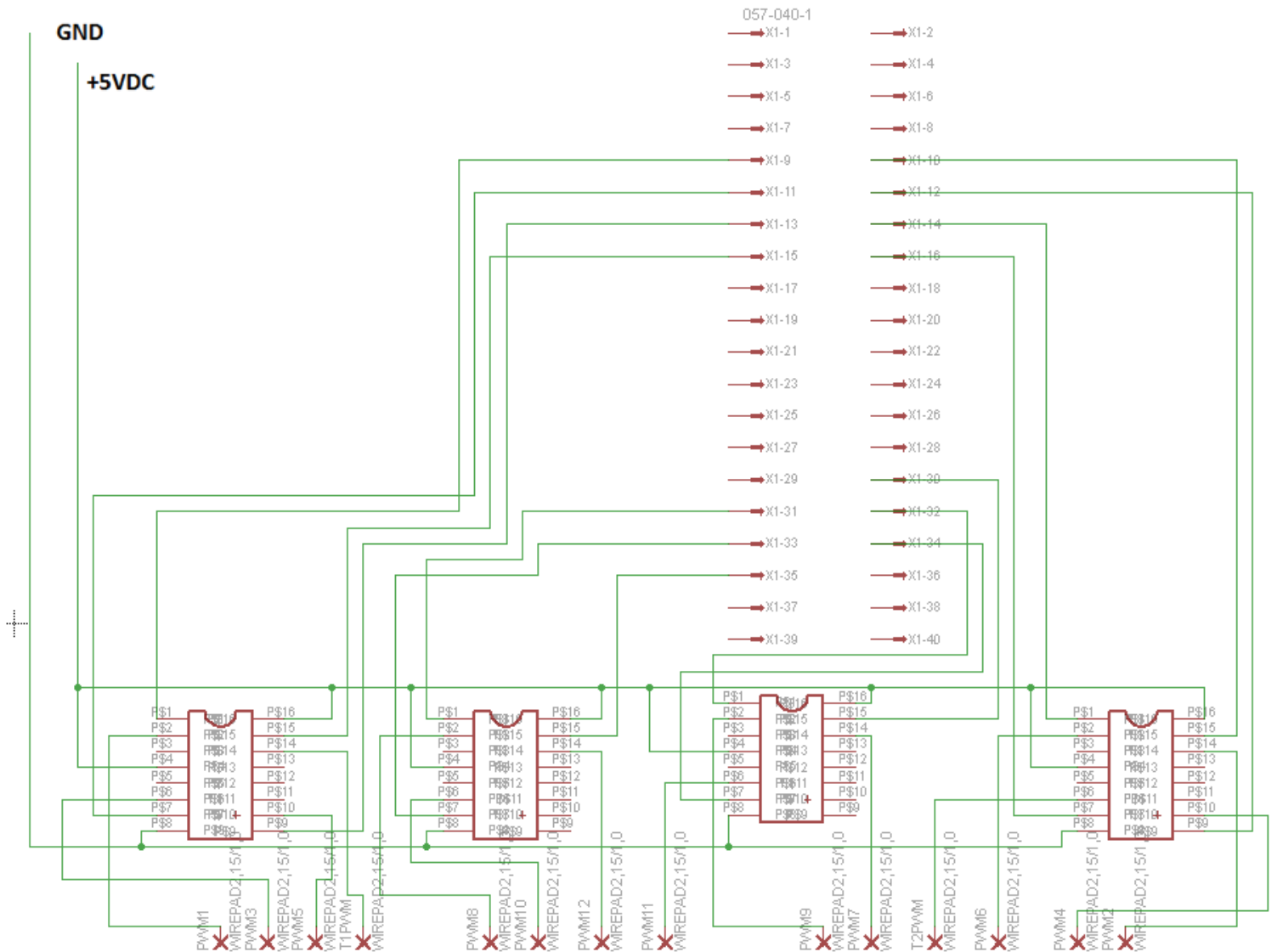


**Figure 4.12:** Schéma électrique du circuit d'interface et de mise en forme (entrées CAP du DSP).

### 4.6 Sorties PWM

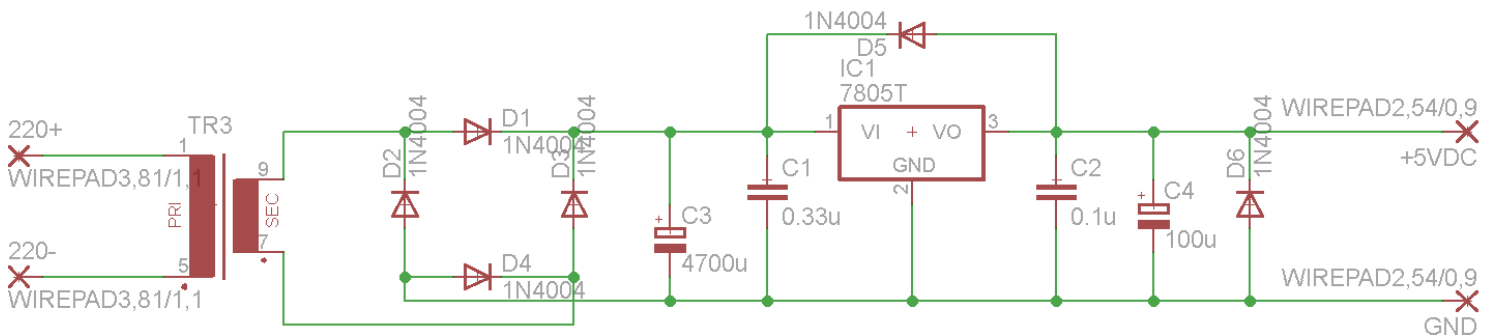
Les sorties PWM du DSP TMS320F2812 sont très sensibles aux erreurs dues au bruit à cause de leur amplitude qui est de 3.3V référencée au GND. Afin de remédier à ce problème, on doit utiliser des signaux différentiels.

En faisant cela, toutes les sorties PWM du DSP seront transformées en signaux différentiels, par le biais du circuit intégré AM26LS31 (Differential Line Driver) (annexe C5). La figure 4.13 schématise la liaison de toutes les liaisons PWM du DSP à travers les AM26LS35.



**Figure 4.13 :** Schéma électrique reliant toutes les sorties PWM du DSP à travers les AM26LS31.

Le circuit intégré AM26LS31 nécessite une alimentation stabilisée de 5V, la figure 4.14 montre le schéma électrique de cette alimentation en utilisant le régulateur de tension LM7805.



**Figure 4.14** : Schéma électrique de l'alimentation stabilisé +5V.

#### 4.7 Circuit d'isolation

En raison de la sensibilité élevée du DSP aux perturbations extérieures (exemple : emplacement par erreur d'un pin à la masse), on doit prévoir un circuit d'isolation entre la carte de développement et la carte de puissance constituée de l'hacheur. Nous avons opté pour des opto-coupleurs de la série 6N137, le schéma de leur branchement est illustré par la figure 4.15.

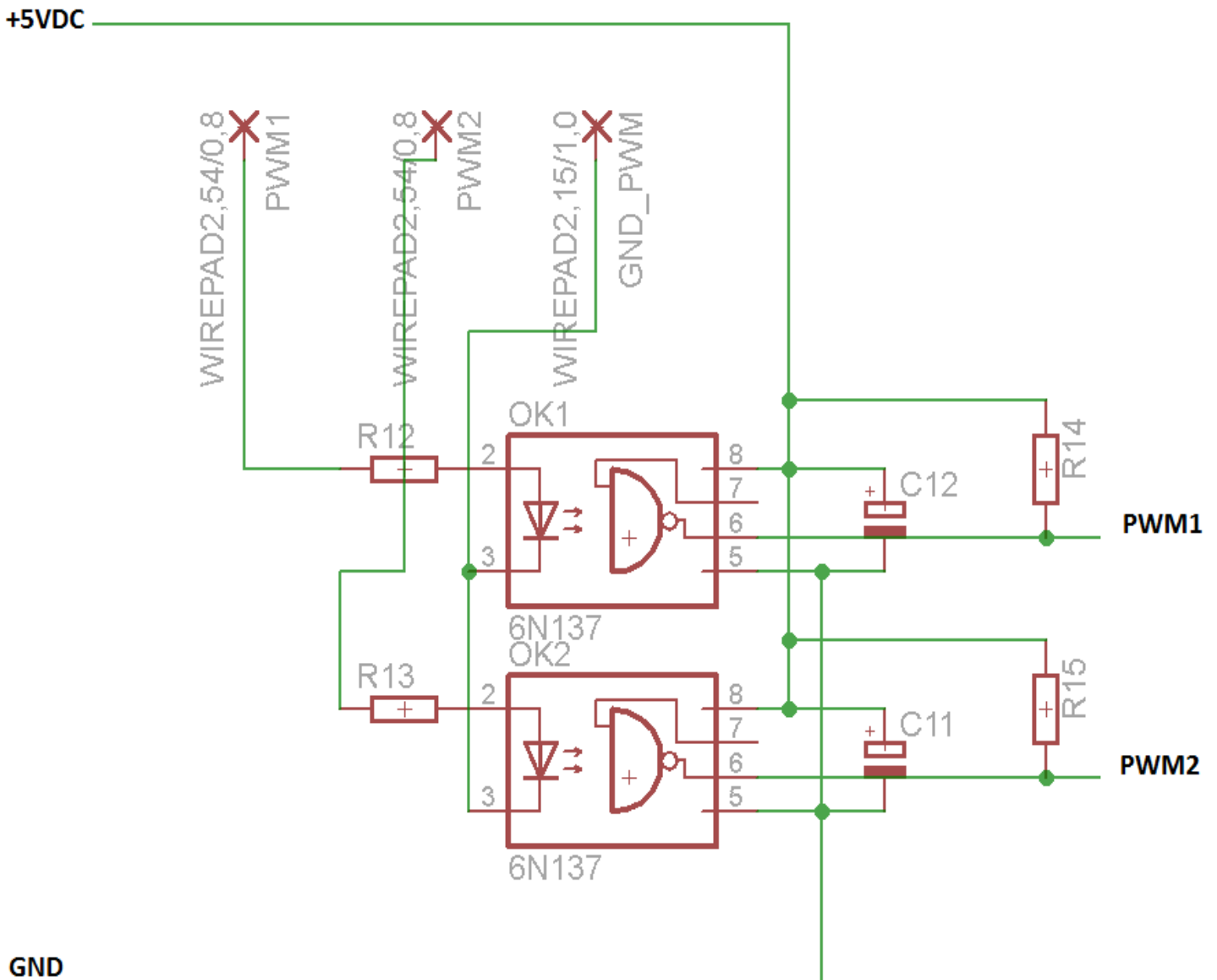


Figure 4.15 : Schéma électrique du circuit d'isolation.

#### 4.8 Alimentation stabilisé réglable

Cette alimentation, réglable, permet de générer la tension d'entrée du hacheur. Elle est dotée d'un circuit, constitué du microcontrôleur 16F88, pour la lecture et l'affichage, en utilisant deux afficheurs sept segments, de la valeur de la tension sélectionnée à l'aide d'un potentiomètre. La figure 4.16 montre le montage électrique de cette alimentation.

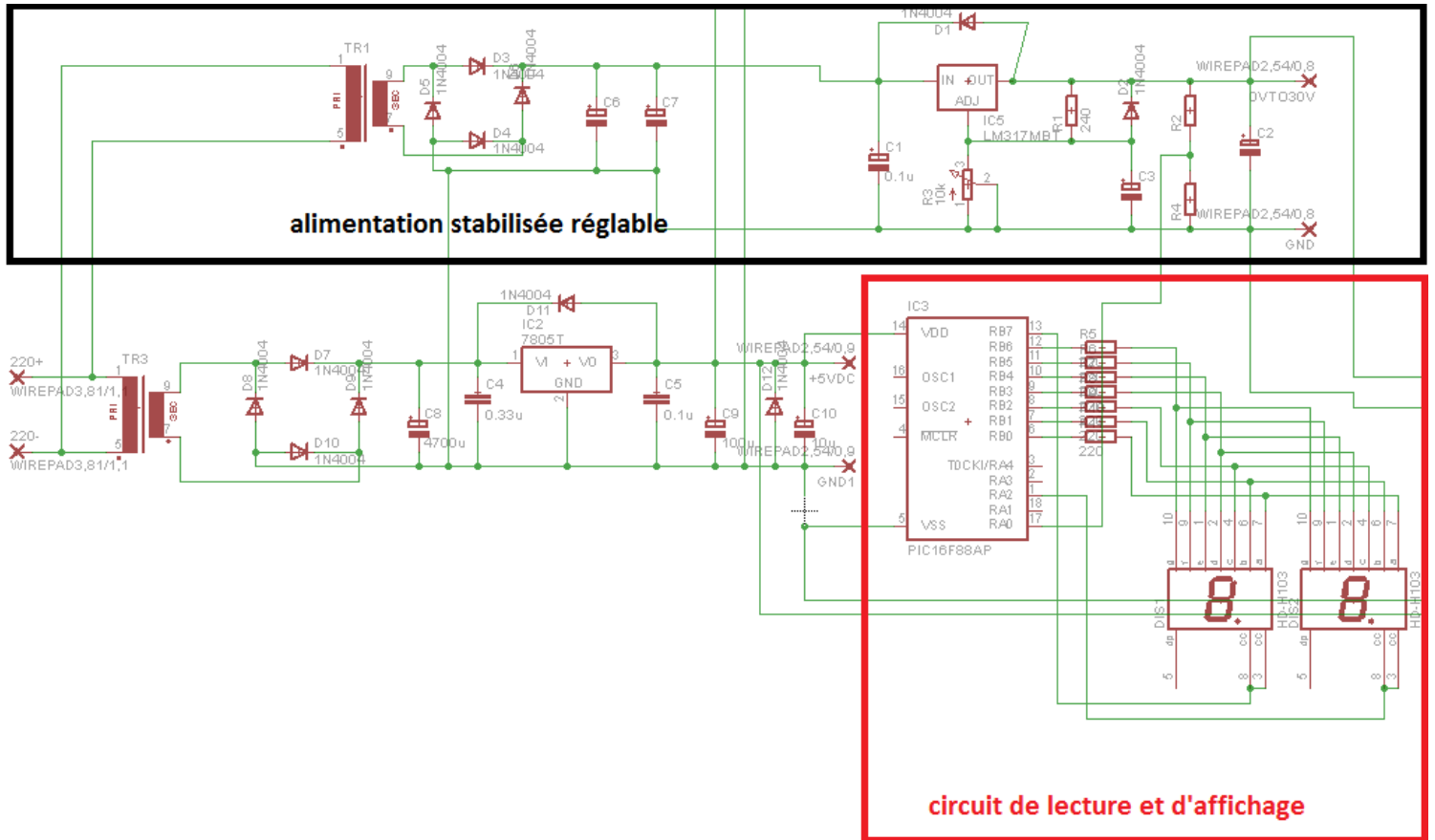


Figure 4.16 : Schéma électrique de l'alimentation stabilisée réglable.

#### 4.9 Circuit électrique global

Le circuit global est constitué de deux parties :

- Circuit de puissance (Fig. 4.17).

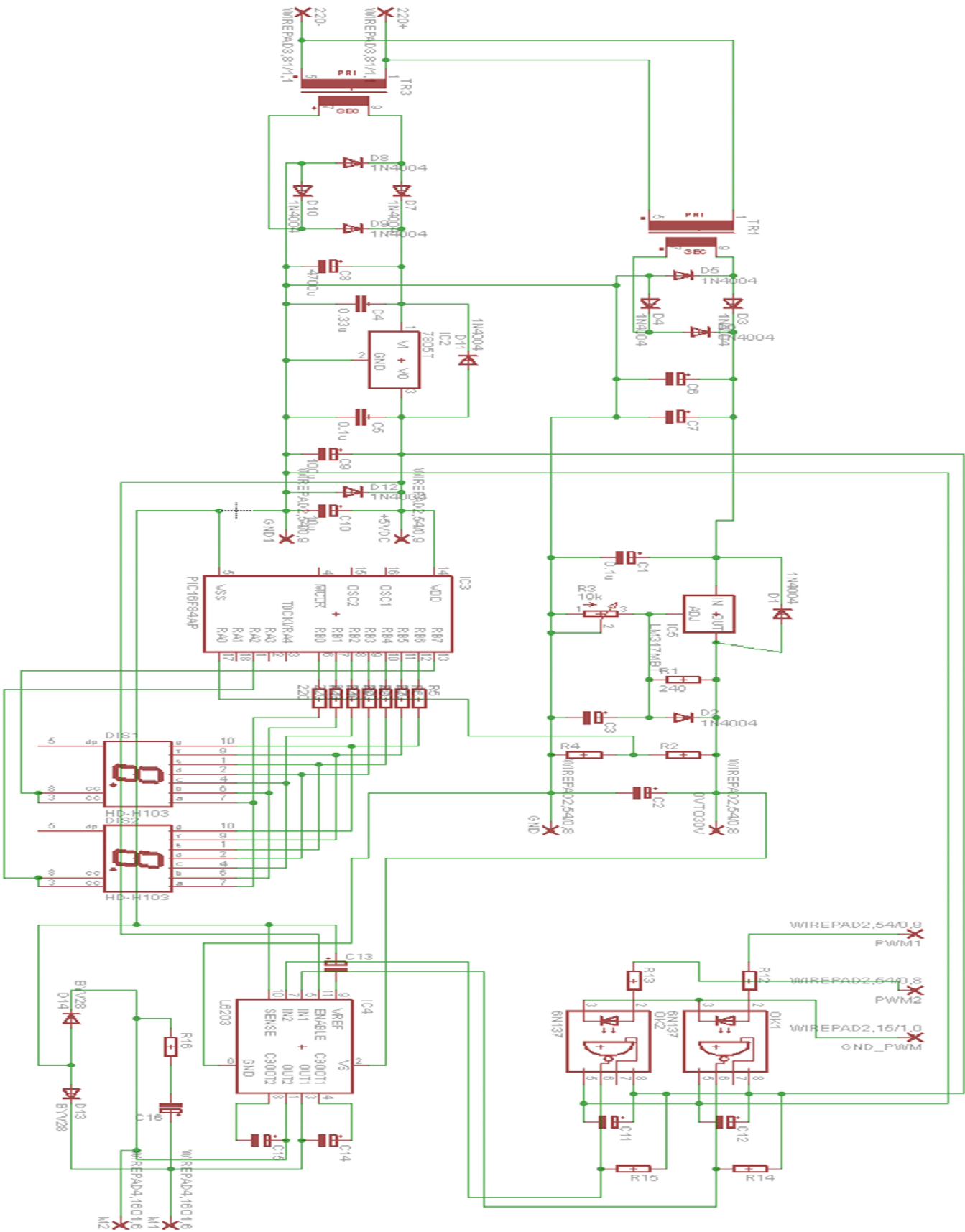


Figure 4.17: Schéma électrique du circuit de puissance.

- Circuits d'interface et de communication série (Fig. 4.18).

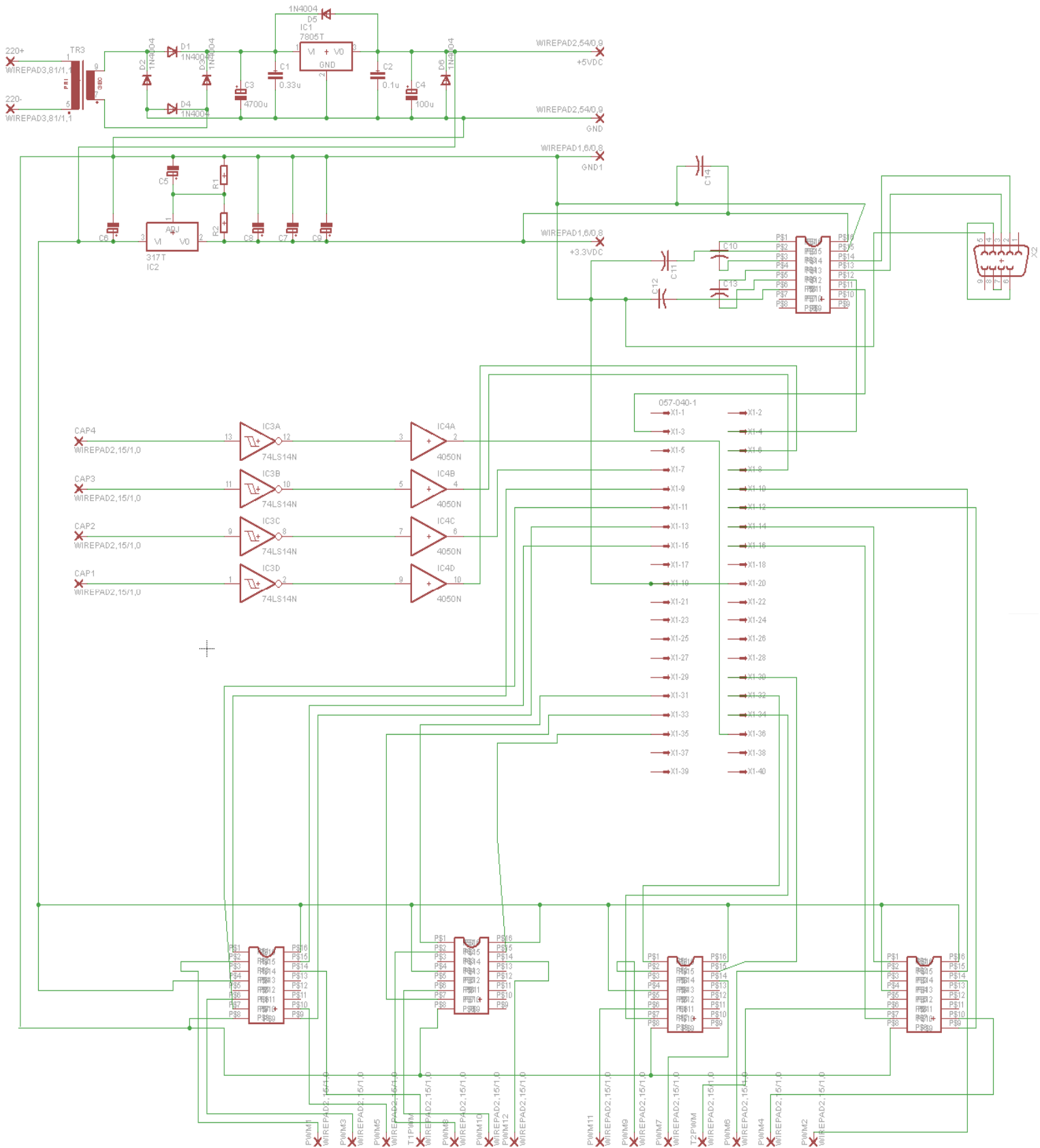
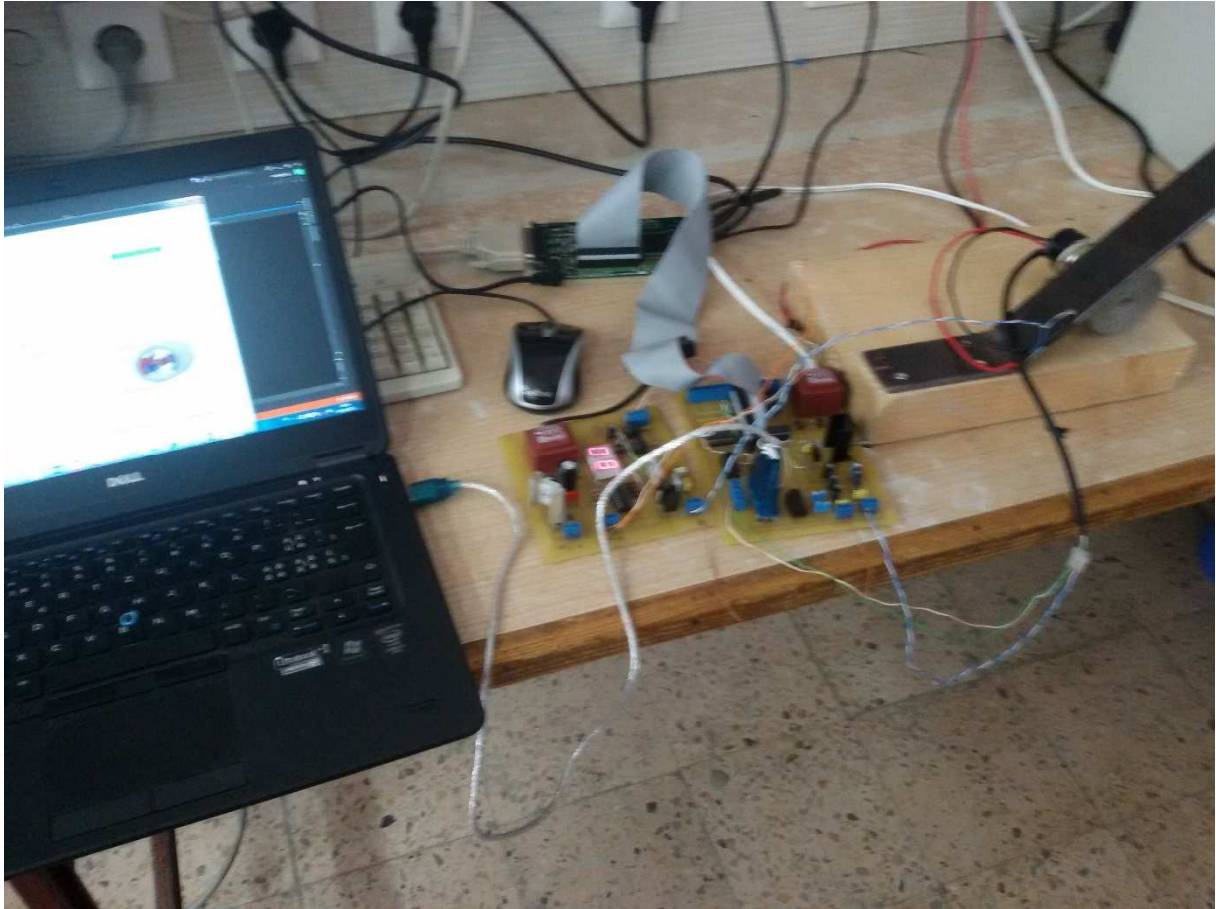


Figure 4.18 : Schéma électrique des circuits d'interface et de communication série.

Voici la réalisation finale du projet (figure 4.19)

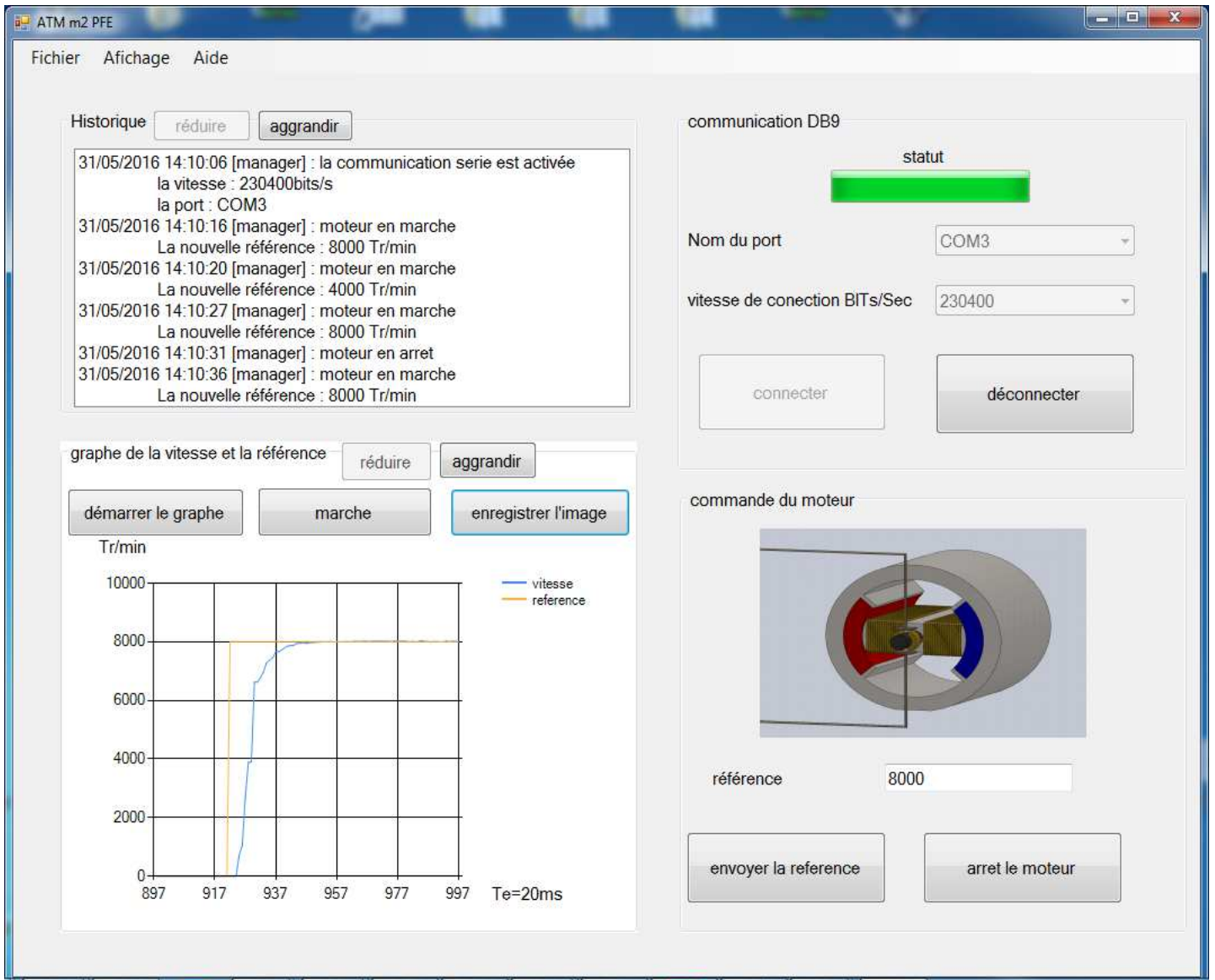


**Figure 4.19** : Réalisation finale du projet.

#### 4.10 Interface graphique

Pour rendre le contrôle et la supervision de l'application facile, en utilisant un ordinateur (commande et supervision à distance par ordinateur), nous avons conçu et mis en œuvre une interface graphique. Nous avons utilisé le logiciel VISUAL STUDIO (annexe A) et C++ pour développer l'interface illustrée par la figure 4.20.





**Figure 4.20:** Interface graphique.

Pour établir la connexion entre l'ordinateur et le DSP, il faut choisir, dans la partie communication DB9 le port et la vitesse (bits/s) de connexion. L'interface développée dispose d'une fenêtre permettant de visualiser l'historique de toutes les opérations effectuées par l'opérateur. Il est possible de visualiser, sur cette interface, l'évolution en temps réel de la vitesse de rotation du moteur, la courbe obtenue peut être enregistrée sous forme d'une image JPG. Il est aussi possible de changer la valeur de la consigne ou d'arrêter le moteur directement à partir de l'interface développée.

### 4.11 Résultats expérimentaux

Nous considérons les deux régulateurs PID et PID neuronal et nous effectuons les essais suivants :

#### 4.11.1 Premier essai

Dans le premier essai nous comparons les performances de la commande PID classique et celle de la commande PID neuronal en l'absence des perturbations. Les résultats de poursuite de la trajectoire de référence, obtenus par l'outil de développement CCS, sont illustrés par les courbes des figures 4.21 et 4.22. Dans la figure 4.23, nous avons refait, en utilisant MATLAB, le tracé de la sortie du moteur obtenue par le PID et celle obtenue par le PID neuronal.

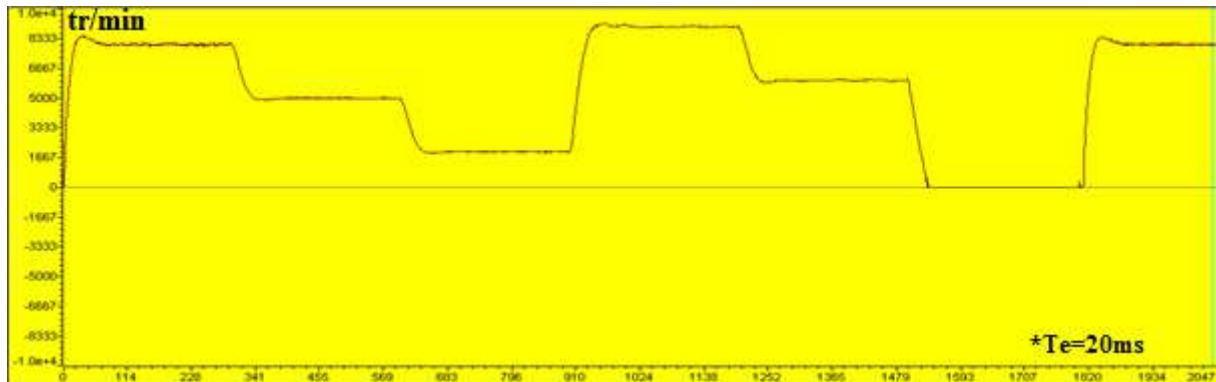


Figure 4.21 : Tracé de la sortie du MCC en utilisant le PID classique.

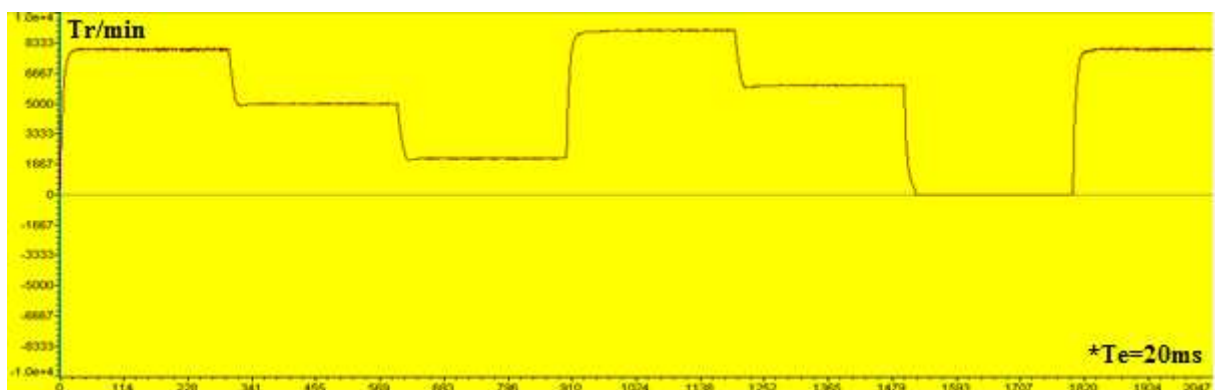
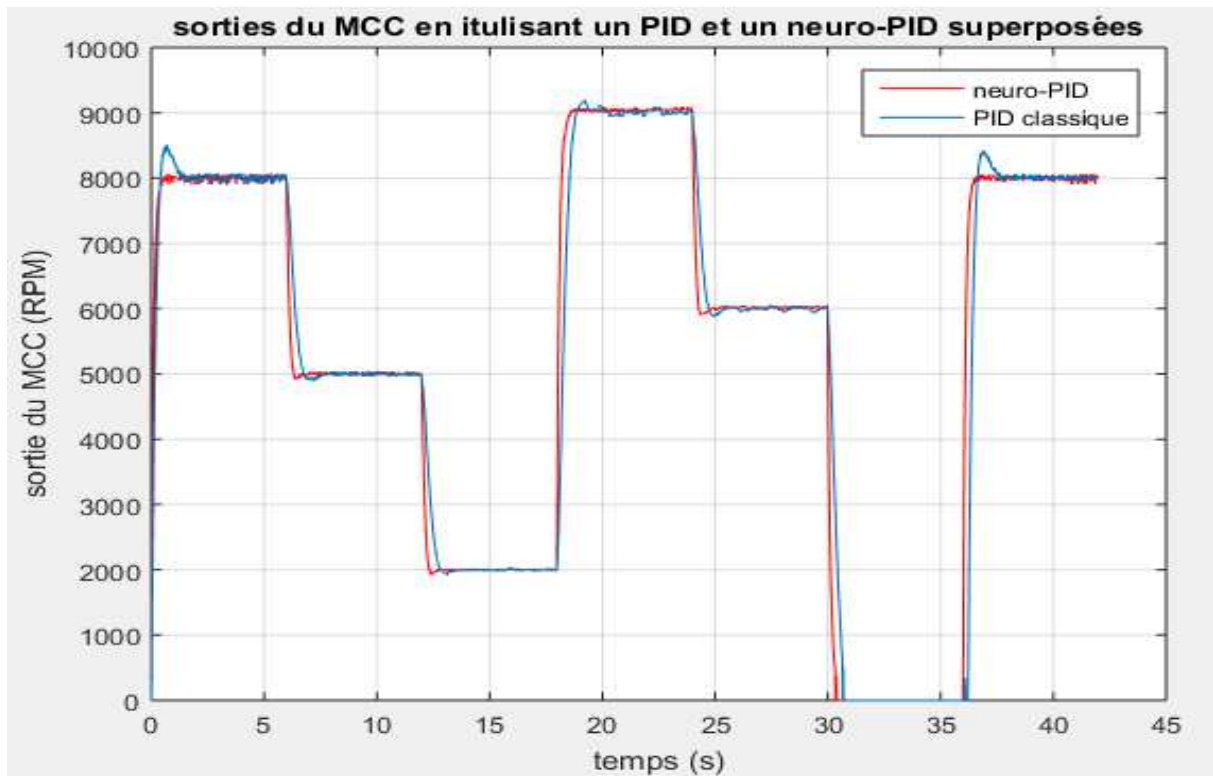


Figure 4.22 : Tracé de la sortie du MCC en utilisant le PID neuronal.



**Figure 4.23 :** Tracé de la sortie du MCC en utilisant le PID et le PID neuronal.

A partir de cet essai, nous constatons que le PID neuronal présente des performances meilleures, en termes de rapidité, précision et dépassement, que le PID classique.

#### 4.11.2 deuxième essai

Dans le deuxième essai nous mettons en évidence les performances de la commande PID neuronal en présence des perturbations. Les résultats de poursuite de la trajectoire de référence sont récupérés à l'aide de l'interface graphique développée. Une perturbation aléatoire additive à la sortie apparaît à l'instant  $87 \cdot T_e = 1,74s$  et disparaît à l'instant  $113 \cdot T_e = 2,26s$  (Fig. 4.24). Nous remarquons que le régulateur a rapidement compensé l'effet de cette perturbation.



**Figure 4.24 :** Tracé de la sortie du MCC en présence d'une perturbation en utilisant le PID neuronal.

#### 4.11.3 Troisième essai

Dans le troisième essai nous appliquons une perturbation additive de 10 % à la sortie, les résultats sont récupérés à l'aide de l'interface graphique développée (figure 4.25 et 4.26). Nous remarquons que le régulateur a compensé l'effet de cette perturbation dans 160ms.

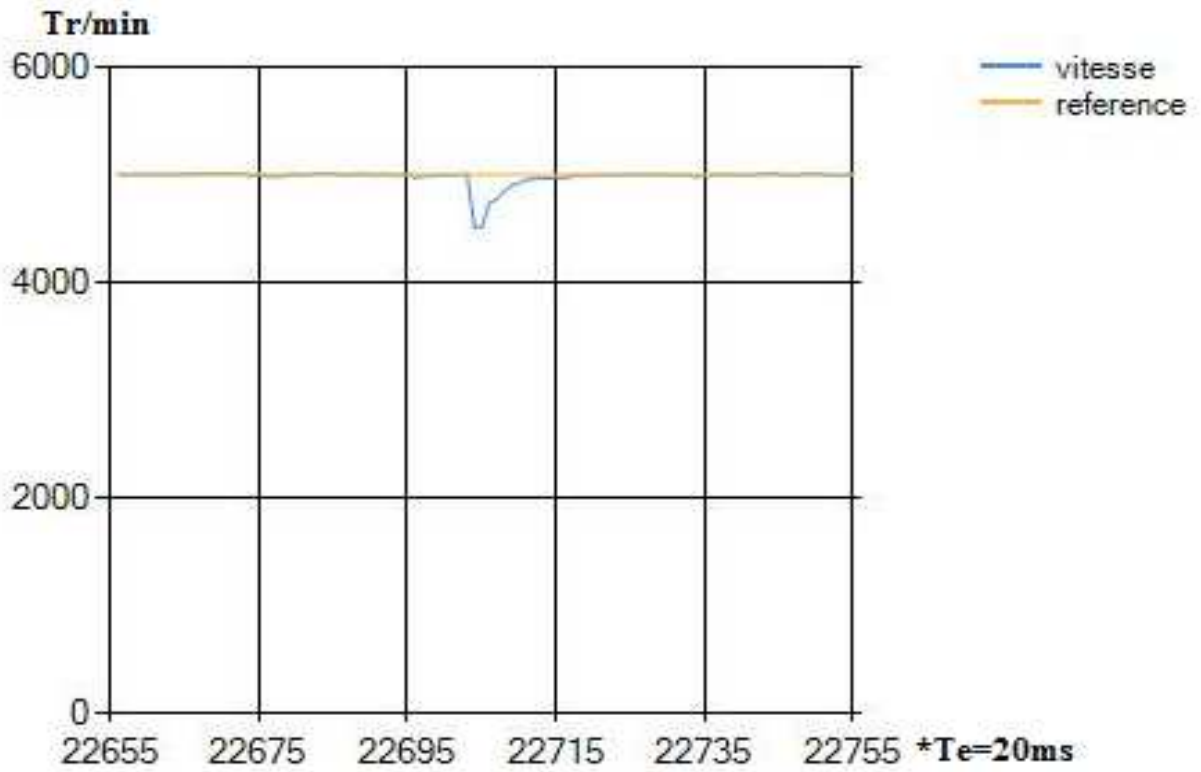


Figure 4.25 : Début de la perturbation de 10% à la sortie.

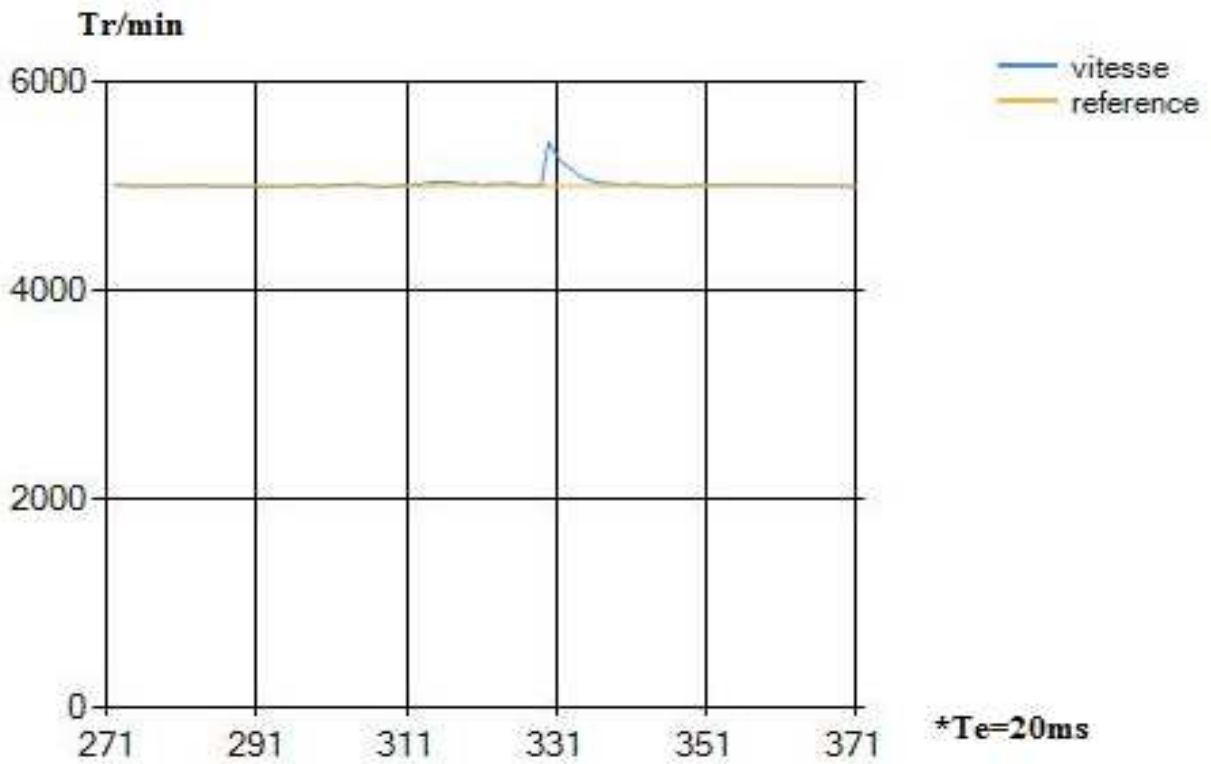


Figure 4.26 : Fin de la perturbation de 10% à la sortie.

#### 4.11.4 Quatrième essai

Dans le quatrième essai nous appliquons une perturbation additive de 50 % à la sortie, les résultats sont récupérés à l'aide de l'interface graphique développée (figure 4.27 et 4.28). Nous remarquons que le régulateur a compensé l'effet de cette perturbation dans 300ms.

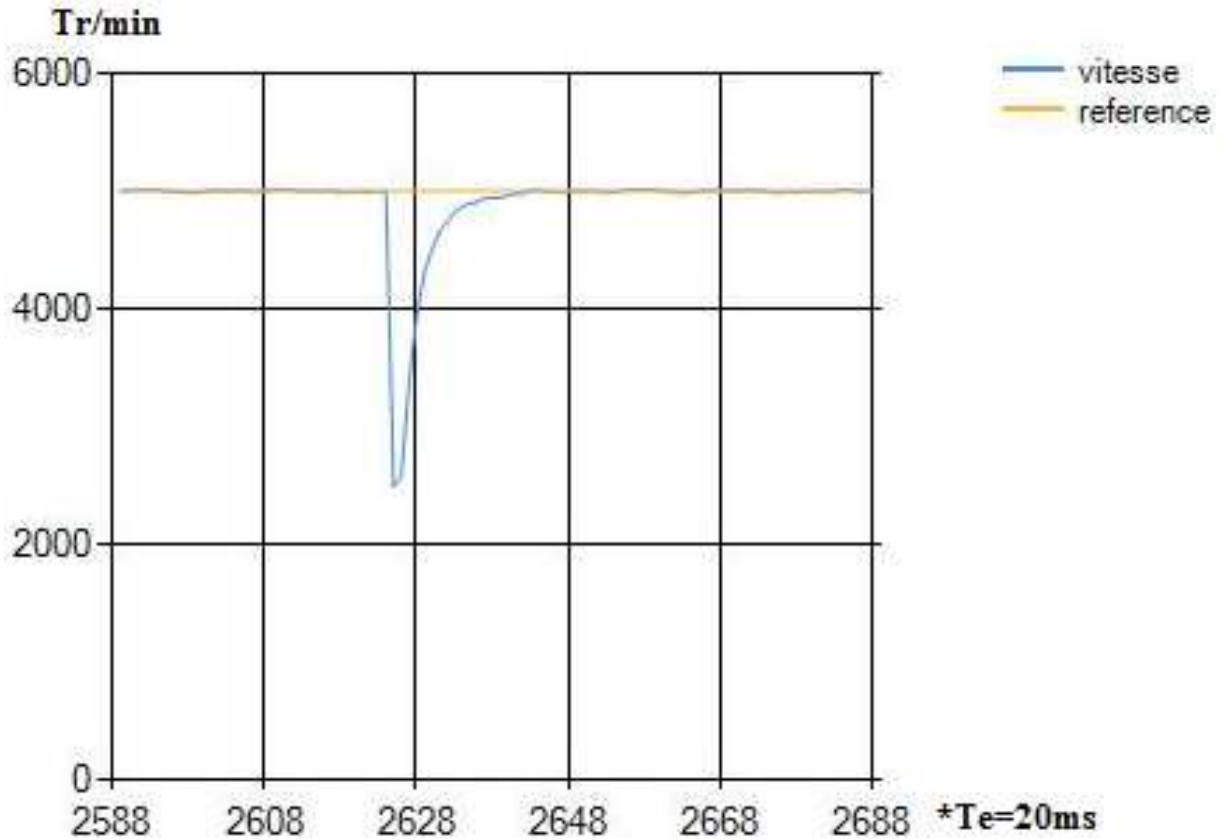
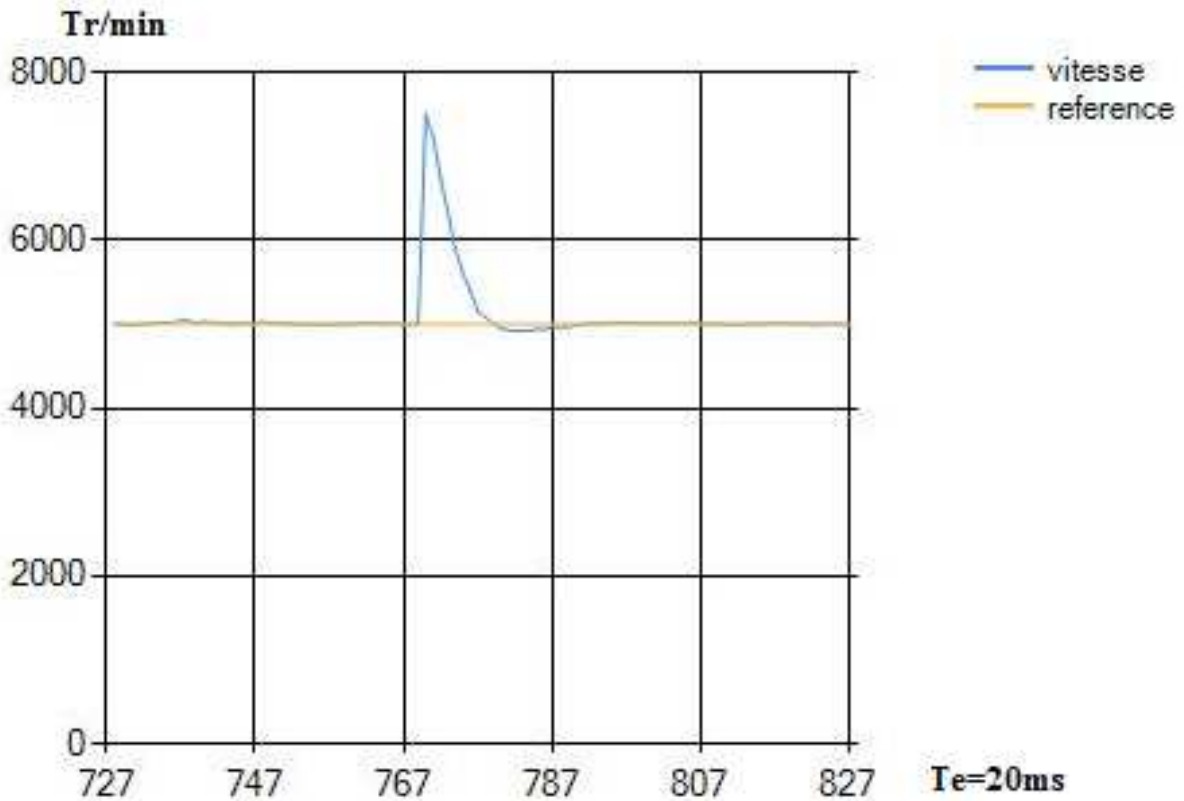


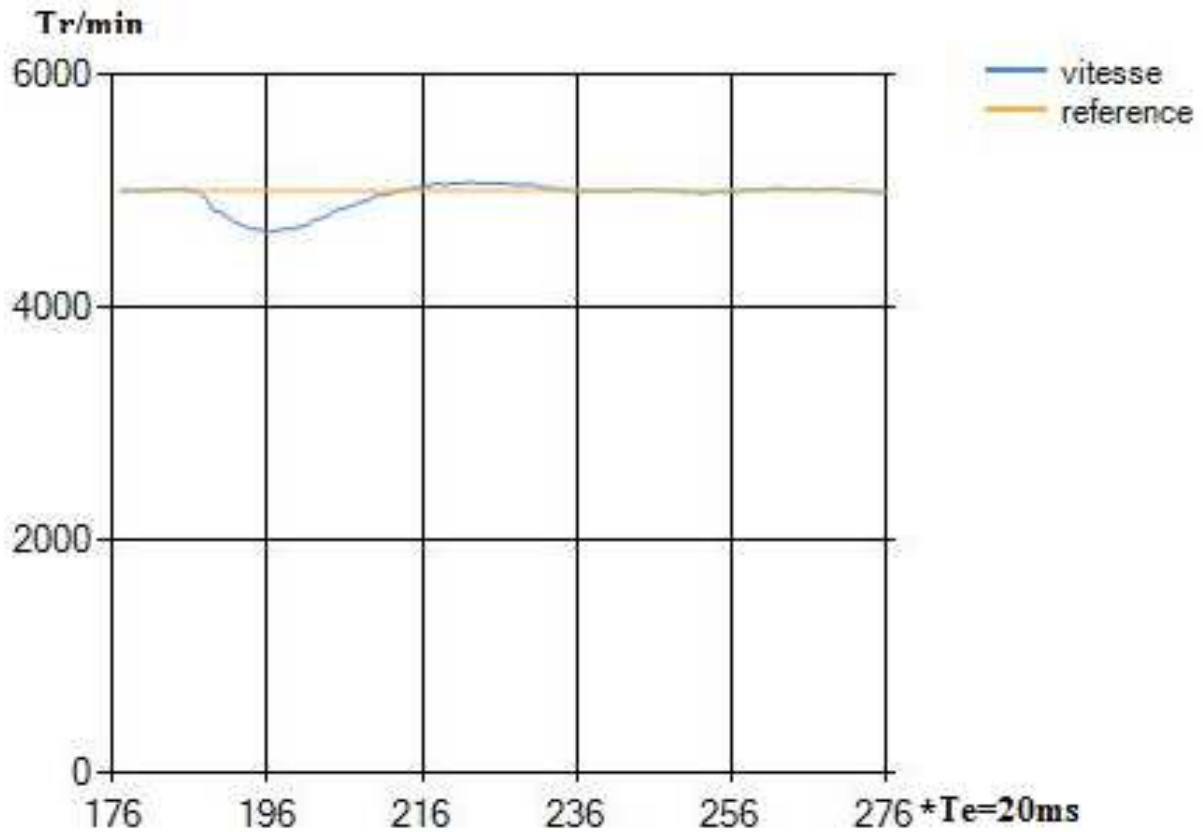
Figure 4.27 : Début de la perturbation de 50% à la sortie.



**Figure 4.28** : Fin de la perturbation de 50% à la sortie.

#### 4.11.5 Cinquième essai

Dans le cinquième essai nous avons ajouté un rhéostat de  $25\Omega$  en série avec le MCC. Cette méthode nous a permis de changer les paramètres du moteur en observant l'effet sur le régulateur PID neuronale, le résultat est récupéré à l'aide de l'interface graphique développée (figure 4.29). Nous remarquons que le régulateur s'adapte avec le nouveau système dans un temps minimal (400ms).



**Figure 4.29** : Sortie du nouveau système (rhéostat en série avec le MCC).

#### 4.12 Conclusion

A partir des résultats obtenus nous constatons que le PID neuronal, implémenté dans la carte de développement TMS320F2812, présente des performances très satisfaisantes en termes de rapidité, stabilité, précision et robustesse vis-à-vis des perturbations.



## Conclusion Générale

Dans ce mémoire, nous avons étudié l'une des techniques de commande par réseaux de neurones. Il s'agit d'une commande PID ajustable dont les paramètres sont calculés en temps réel en utilisant deux réseaux de neurones ; le premier permet d'obtenir le jacobien du système et le deuxième permet de calculer en ligne les paramètres du régulateur PID. Cette commande est dénommée commande PID neuronal. En effet, les réseaux de neurones sont dotés de deux propriétés importantes : l'apprentissage et l'approximation de n'importe quelle fonction non linéaire. Leur association avec un correcteur PID permet d'obtenir un algorithme de commande qui peut donner de bonnes performances dans le cas d'un système non linéaire. Le calcul en ligne des paramètres permet à l'algorithme de commande de s'adapter à la variation des paramètres du système et à l'apparition des perturbations.

Pour étudier les performances de la commande PID neuronal, nous avons considéré la commande d'un réacteur chimique parfaitement mélangé, il s'agit d'un système non linéaire benchmark. Tout d'abord une procédure d'identification neuronale hors ligne est nécessaire pour construire l'émulateur du système. Une fois l'émulateur est obtenu, un deuxième réseau de neurones avec un apprentissage en ligne est mis au point pour calculer les paramètres du régulateur. Nous avons utilisé le logiciel MatLab pour coder les algorithmes d'identification et de commande. Afin de conclure sur l'apport de la commande PID neuronal, nous avons aussi codé l'algorithme de la commande PID conventionnel. Nous avons aussi considéré la commande d'un système linéaire simple à savoir le moteur à courant continu. Les résultats de simulation, que nous avons obtenus, montrent bien l'efficacité de la commande PID neuronal, plus particulièrement dans le cas d'un système non linéaire. En effet, nous avons obtenu une bonne poursuite de la trajectoire de référence et robustesse vis-à-vis des perturbations.

Afin de valider expérimentalement les résultats obtenus par simulation, nous avons considéré la réalisation pratique de l'application qui consiste à asservir la vitesse de rotation d'un moteur à courant continu. La mise au point de cette application nécessite la réalisation du régulateur et d'un certain nombre de circuits électronique. Une manière simple et efficace à la fois pour réaliser le régulateur est celle qui consiste à implémenter numériquement, en utilisant un DSP, l'algorithme de commande. En effet, nous avons utilisé la carte de développement eZdsp F2812, disponible à notre Département, pour implémenter les algorithmes de commande. Cette carte de développement contient des outils bien adaptés à la commande des machines électrique. Pour pouvoir faire varier la vitesse du moteur nous

avons réalisé une carte de puissance contenant principalement un hacheur à quatre quadrants. L'association de cette carte, la carte de développement et le moteur nécessite la réalisation d'autres cartes d'interface, d'isolation et d'alimentations stabilisées. Nous avons conçu et développé une interface graphique pour contrôler et superviser l'application réalisée. Cette interface est exécutée sur un ordinateur qui communique directement, à travers une liaison série RS232, avec la carte de DSP. Une fois l'application a été mise en marche, nous avons effectué plusieurs essais (sans et avec perturbation). Les résultats récupérés confirment bien l'apport et l'efficacité de l'algorithme de commande étudié. L'application que nous avons réalisée pourrait servir comme un outil de base pour tester d'autres algorithmes de commande ou exploitée pédagogiquement dans les travaux pratiques.

A l'issue de cette étude, nous avons découvert des nouvelles techniques de commande efficace, maîtrisé l'implémentation des algorithmes de commande sur DSP et appris à mettre au point des réalisations électronique.

## Références bibliographiques

- [1] Y.X. Su, Dong Sun, B.Y. Duan. *Design of an enhanced nonlinear PID controller* [en ligne]. Hong Kong, 2005, 20p. Format PDF. Disponible sur :< <http://www.sciencedirect.com/science/article/pii/S0957415805000590> > (Consulté le 10/06/2016)
- [2] HASSAN K. Khalil. *Nonlinear Systems*. Third Edition. New Jersey, 2002, 767 p.
- [3] ANTONIO Visioli. *Practical PID control*. Italy, 2006, 322 p.
- [4] J. Espinosa, J. Vandewalle, and V. Wertz. . *Fuzzy Logic, Identification and Predictive Control*.
- [5] S. Chen and S. A. Billings, "Neural Networks for nonlinear system modeling and Identification", *Int. J. Control*, vol. 56, no. 2, pp 319-346, Aug. 1992.
- [6] ELECTRO-CRAFT CORPORATION. *DC motor speed controls servo systems*. USA, 1972, 508 p.
- [7] FABRICE, Sincère. *Electrotechnique*. Version 3.0.5. [En ligne]. Cours, 31 p. Disponible sur :< <http://fabrice.sincere.pagesperso-orange.fr/electrotechnique.htm> > (Consulté le 05/06/2016).
- [8] AIT SAHED Oussama, BENACHOUR Ahmed. *Implémentation sur carte DSP de la commande PID auto-ajustable d'un moteur à courant continu*. Mémoire de projet de fin d'études master : électronique option automatisme. Algérie Blida : université de Blida 1, 2010, 91 p.
- [9] ALIANE, Ahcene. *Electronique de puissance*. Algérie : université de Blida 1, Cours, 2014, 19 p.

- [10] BRIZEUX, Marc. *Introduction à la correction des systèmes asservis*. [En ligne]. Cours, 2010, 8 p. Disponible sur : < [http://www.cpge-brizeux.fr/casiers/marc/PSI0910/C33\\_01\\_Correcteursv2.pdf](http://www.cpge-brizeux.fr/casiers/marc/PSI0910/C33_01_Correcteursv2.pdf)? > (Consulté le 05/06/2016).
- [11] *PID Control New Identification and Design Methods*. united kingdom : Mohammad H.Moradi & Michael A. Johnson, 2005, 558 p.
- [12] CLAUDE Bergmann, *Electronique et communication BTS industriels*. Paris : Dunod, 2005, 238 p.
- [13] JAN Axelson, *Serial port complete programming and circuits for rs-232 and rs-485 links and networks*. USA, 2000, 322 p.
- [14] Athi thilagalakshmi. Simulation of Neuro-PID Controller for Pressure Process. **In :** *International Journal of Computer Applications (0975 – 8887) : International Conference on Innovations In Intelligent Instrumentation, Optimization And Signal Processing*. Inde, 2013, p.18-21.
- [15] Sigeru Omatu, Michifumi Yoshioka, Toshihisa Kosaka, Hidekazu Yanagimoto. Neuro-PID Control of Speed and Torque of Electric Vehicle. **In :** *International Journal on Advances in Systems and Measurements*, vol 3 no 1 & 2. Japon, 2010, p.82-91.
- [16] M.BECHOUCHE, Ali. *Utilisation des techniques avancées pour l'observation et la commande d'une machine asynchrone : Application a une éolienne*. Thèse de doctorat : électrotechnique. Algérie, Tizi-Ouzou : université mouloud Mammeri, 2013, 168p.
- [17] MOHAMED YESSIN, Ammar. *Mise en œuvre de réseaux de neurones pour la modélisation de cinétiques réactionnelles en vue de la transposition batch/continu*. Thèse de doctorat : Génie des procédés et de l'Environnement. Toulouse : INP, 2007, 194 p.

# ANNEXE

## A (Visual Basic)

**Visual studio** est un outil développé par Microsoft pour développer facilement des applications fonctionnant sous Microsoft Windows. C'est une suite de logiciels de développement pour Windows conçue par Microsoft. La dernière version s'appelle **Visual Studio 2015**.

Visual studio est, comme son nom l'indique, un outil visuel permettant de créer sans notion de programmation l'interface graphique (*GUI* - Graphical User Interface) en disposant à l'aide de la souris des éléments graphiques (boutons, images, champs de texte, menus déroulants,...).

Visual Studio Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Developer. L'intérêt de ce langage est de pouvoir associer aux éléments de l'interface des portions de code associées à des événements (clic de souris, appui sur une touche, ...). Pour cela, Visual studio utilise plusieurs langages de programmation (C#, visual basic, java, ... etc.).

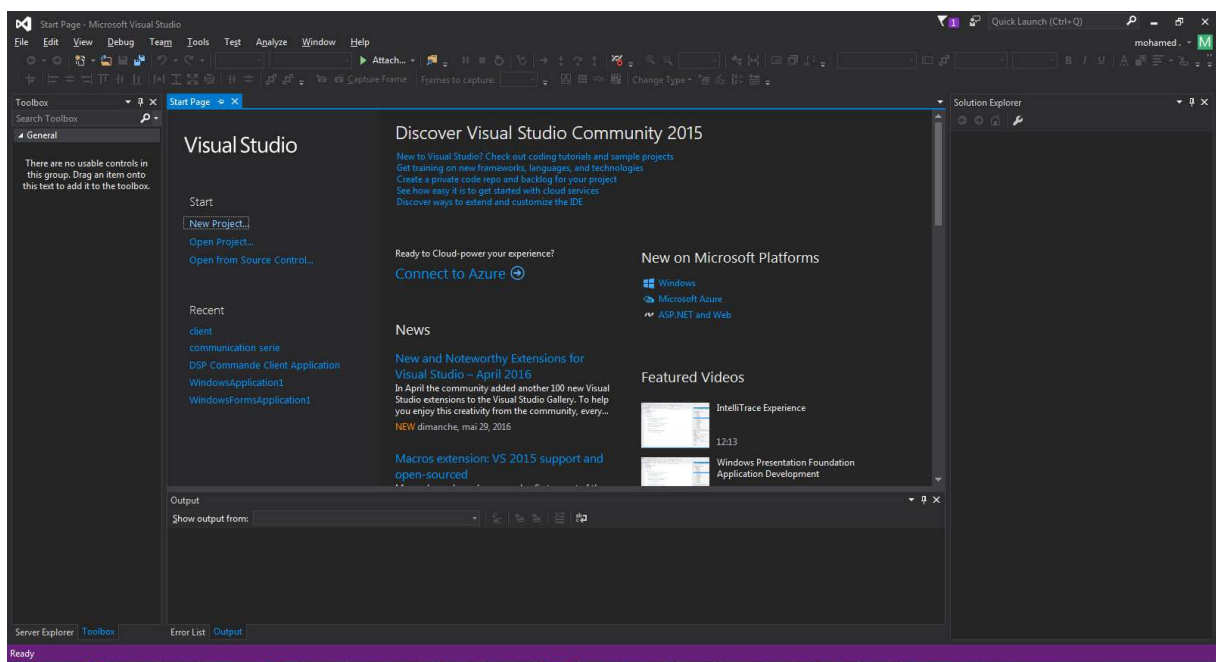


Figure : Accueil du Visual studio

## **B1 (liste de Composants)**

### **Circuits d'interface et de communication série**

Transformateur 220VAC-9VDC.

D1, D2, D3, D4, D5, D6 :1N4007.

IC1 : lm7805.

IC2 : lm317.

IC3 : SN74LS14.

IC4 : HEF4050BP.

IC5 : MAX232.

IC6, IC7, IC8, IC9 : AM26ls14.

C1 : 0,33uF.

C2 : 0,1uF.

C3 : 4700uF.

C4, C7 : 100uF.

C5, C6, C8, C9 : 10nF.

C10, C11, C12, C13, C14 : 1uF.

R1 : 2,2K $\Omega$ .

R2 : 1,5K $\Omega$ .

Connecteur RS232.

Connecteur PC 40 pin.

### **Circuit de puissance**

Transformateur 220VAC-9VDC.

IC2 : 7805.

IC3 : pic16F88.

IC4 : L6203.

DIS1, DIS2 : 7 segments cathode commune.

OK1, OK2 : 6N137.

D7, D8, D9, D10, D11, D12 :1N4007.

D13, D14 : BYV28.

C4 : 0,33uF.

C5 : 0,1uF.

C8 : 4700uF.

C9 : 100uF.

C10, C14, C15 : 10uF.

C11, C12, C13, C16: 10nF.

R5, R6, R7, R8, R9, R10, R11, R12, R13 : 220Ω.

R14, R15 : 1,5k Ω.

## B2 (programme de PIC 16f88)

```
float volt;
unsigned char C1,C2,RR1=100,RR2=10;      // R1 et R2 en K ohm
void septsegments(char C1,char C2) {
    portb.f7=0;   porta.f2=1;
    if (C1==0) {portb=(portb&0x80)+63;}
    if (C1==1) {portb=(portb&0x80)+6;}
    if (C1==2) {portb=(portb&0x80)+91;}
    if (C1==3) {portb=(portb&0x80)+79;}
    if (C1==4) {portb=(portb&0x80)+102;}
    if (C1==5) {portb=(portb&0x80)+109;}
    if (C1==6) {portb=(portb&0x80)+125;}
    if (C1==7) {portb=(portb&0x80)+7;}
    if (C1==8) {portb=(portb&0x80)+127;}
    if (C1==9) {portb=(portb&0x80)+111;}
    delay_ms(20);
    portb.f7=1;   porta.f2=0;
    if (C2==0) {portb=(portb&0x80)+63;}
    if (C2==1) {portb=(portb&0x80)+6;}
    if (C2==2) {portb=(portb&0x80)+91;}
    if (C2==3) {portb=(portb&0x80)+79;}
    if (C2==4) {portb=(portb&0x80)+102;}
    if (C2==5) {portb=(portb&0x80)+109;}
    if (C2==6) {portb=(portb&0x80)+125;}
    if (C2==7) {portb=(portb&0x80)+7;}
    if (C2==8) {portb=(portb&0x80)+127;}
    if (C2==9) {portb=(portb&0x80)+111;}
    delay_ms(20);
}
void main(){
    OSCCON=0b01100110;
    ANSEL =0b00000001;
    trisb=0x00; trisa=0x21;    // configuration les entrées et les sorties
    porta=0x00; portb=0x00;    // initialisation des portes
    ADC_Init(); // Initialiser l'ADC
    while(1)
    {
        volt = ADC_Get_Sample(0);    // lire la valeur analogique
        volt=(volt*5)/1024;
        volt = volt*(RR1+RR2)/RR2;
        C1=volt/10;
        C2=volt-(C1*10);
        septsegments(C1,C2) ;
    }
}
```

# C1 (L6203 datasheet)


## DMOS FULL BRIDGE DRIVER

- SUPPLY VOLTAGE UP TO 48V
- 5A MAX PEAK CURRENT (2A max. for L6201)
- TOTAL RMS CURRENT UP TO  
L6201: 1A; L6202: 1.5A; L6203/L6201PS: 4A
- $R_{DS(ON)}$  0.3  $\Omega$  (typical value at 25 °C)
- CROSS CONDUCTION PROTECTION
- TTL COMPATIBLE DRIVE
- OPERATING FREQUENCY UP TO 100 KHz
- THERMAL SHUTDOWN
- INTERNAL LOGIC SUPPLY
- HIGH EFFICIENCY


### DESCRIPTION

The I.C. is a full bridge driver for motor control applications realized in Multipower-BCD technology which combines isolated DMOS power transistors with CMOS and Bipolar circuits on the same chip. By using mixed technology it has been possible to optimize the logic circuitry and the power stage to achieve the best possible performance. The DMOS output transistors can operate at supply voltages up to 42V and efficiently at high switch-


**MULTIPOWER BCD TECHNOLOGY**




Powerdip 12+3+3



SO20 (12+4+4)



Multiwatt11

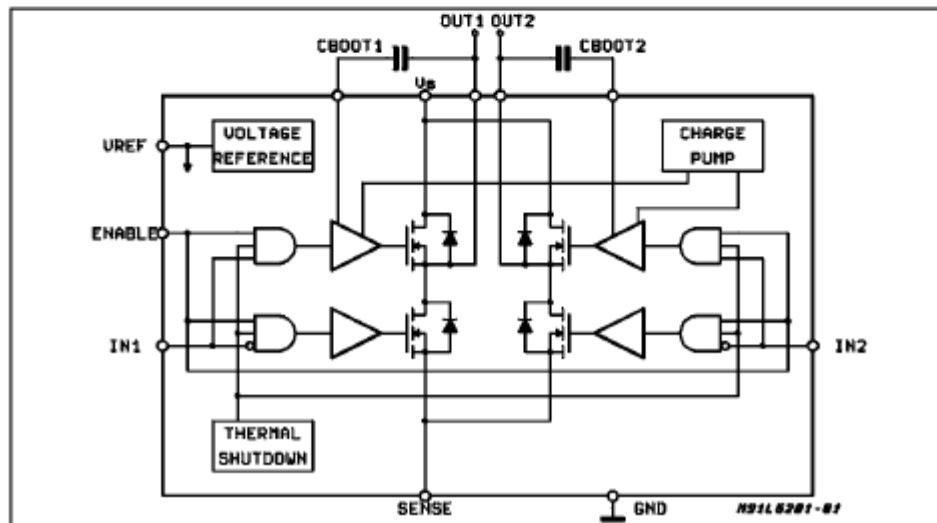


PowerSO20

**ORDERING NUMBERS:**  
L6201 (SO20)  
L6201PS (PowerSO20)  
L6202 (Powerdip18)  
L6203 (Multiwatt)

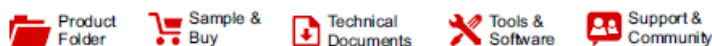
ing speeds. All the logic inputs are TTL, CMOS and  $\mu C$  compatible. Each channel (half-bridge) of the device is controlled by a separate logic input, while a common enable controls both channels. The I.C. is mounted in three different packages.

### BLOCK DIAGRAM





# C2 (Max232 datasheet)



MAX232, MAX232I

SLLS047M – FEBRUARY 1989 – REVISED NOVEMBER 2014

## MAX232x Dual EIA-232 Drivers/Receivers

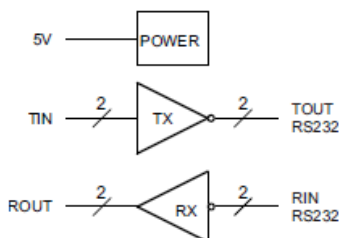
### 1 Features

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0- $\mu$ F Charge-Pump Capacitors
- Operates up to 120 kbit/s
- Two Drivers and Two Receivers
- $\pm$ 30-V Input Levels
- Low Supply Current: 8 mA Typical
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1- $\mu$ F Charge-Pump Capacitors is Available With the MAX202 Device

### 2 Applications

- TIA/EIA-232-F
- Battery-Powered Systems
- Terminals
- Modems
- Computers

### 4 Simplified Schematic



### 3 Description

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept  $\pm$ 30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels.

#### Device Information<sup>(1)</sup>

ORDER NUMBER	PACKAGE (PIN)	BODY SIZE
MAX232x	SOIC (16)	9.90 mm $\times$ 3.91 mm
	SOIC (16)	10.30 mm $\times$ 7.50 mm
	PDIP (16)	19.30 mm $\times$ 6.35 mm
	SOP (16)	10.3 mm $\times$ 5.30 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

## C3 (HEF4050B datasheet)

# HEF4050B

Hex non-inverting buffers

Rev. 9 — 24 March 2016

Product data sheet

### 1. General description

---

The HEF4050B provides six non-inverting buffers with high current output capability suitable for driving TTL or high capacitive loads. Since input voltages in excess of the buffers' supply voltage are permitted, the buffers may also be used to convert logic levels of up to 15 V to standard TTL levels. Their guaranteed fan-out into common bipolar logic elements is shown in [Table 3](#).

It operates over a recommended  $V_{DD}$  power supply range of 3 V to 15 V referenced to  $V_{SS}$  (usually ground). Unused inputs must be connected to  $V_{DD}$ ,  $V_{SS}$ , or another input.

### 2. Features and benefits

---

- Accepts input voltages in excess of the supply voltage
- Fully static operation
- 5 V, 10 V, and 15 V parametric ratings
- Standardized symmetrical output characteristics
- Specified from  $-40\text{ }^{\circ}\text{C}$  to  $+85\text{ }^{\circ}\text{C}$
- Complies with JEDEC standard JESD 13-B

### 3. Applications

---

- LOC MOS (Local Oxidation CMOS) to DTL/TTL converter
- HIGH sink current for driving two TTL loads
- HIGH-to-LOW level logic conversion

# C4 (SN74ls14 datasheet)

## SN5414, SN54LS14, SN7414, SN74LS14 HEX SCHMITT-TRIGGER INVERTERS

SLS049B – DECEMBER 1983 – REVISED FEBRUARY 2002

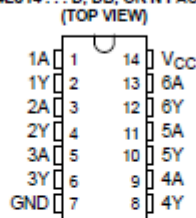
- Operation From Very Slow Edges
- Improved Line-Receiving Characteristics
- High Noise Immunity

### description

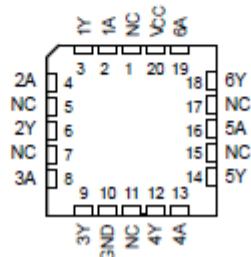
Each circuit functions as an inverter, but because of the Schmitt action, it has different input threshold levels for positive-going ( $V_{T+}$ ) and negative-going ( $V_{T-}$ ) signals.

These circuits are temperature compensated and can be triggered from the slowest of input ramps and still give clean, jitter-free output signals.

SN5414, SN54LS14 . . . J OR W PACKAGE  
SN7414 . . . D, N, OR NS PACKAGE  
SN74LS14 . . . D, DB, OR N PACKAGE



SN54LS14 . . . FK PACKAGE  
(TOP VIEW)



NC – No Internal connection

### ORDERING INFORMATION

$T_A$	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP – N	Tube	SN7414N	SN7414N
		Tube	SN74LS14N	SN74LS14N
	SOIC – D	Tube	SN7414D	7414
		Tape and reel	SN7414DR	
		Tube	SN74LS14D	LS14
	Tape and reel	SN74LS14DR		
SOP – NS	Tape and reel	SN7414NSR	SN7414	
	SSOP – DB	Tape and reel	SN74LS14DBR	LS14
–55°C to 125°C	CDIP – J	Tube	SN5414J	SN5414J
		Tube	SNJ5414J	SNJ5414J
		Tube	SN54LS14J	SN54LS14J
		Tube	SNJ54LS14J	SNJ54LS14J
	COP – W	Tube	SNJ5414W	SNJ5414W
		Tube	SNJ54LS14W	SNJ54LS14W
LCCC – FK	Tube	SNJ54LS14FK	SNJ54LS14FK	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at [www.ti.com/sc/package](http://www.ti.com/sc/package).



Please be aware that an Important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA Information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

 **TEXAS  
INSTRUMENTS**

Copyright © 2002, Texas Instruments Incorporated  
On products compliant to MIL-PRF-38535, all parameters are tested unless otherwise noted. On all other products, production processing does not necessarily include testing of all parameters.

# C5 (Am26ls31 datasheet)



AM26LS31, AM26LS31C, AM26LS31M

www.ti.com

SLLS114J – JANUARY 1979 – REVISED JANUARY 2014

## AM26LS31x Quadruple Differential Line Driver

Check for Samples: [AM26LS31](#), [AM26LS31C](#), [AM26LS31M](#)

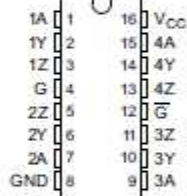
### FEATURES

- Meets or Exceeds the Requirements of ANSI TIA/EIA-422-B and ITU
- Operates From a Single 5-V Supply
- TTL Compatible
- Complementary Outputs
- High Output Impedance in Power-Off Conditions
- Complementary Output-Enable Inputs
- On Products Compliant to MIL-PRF-38535, All Parameters Are Tested Unless Otherwise Noted. On All Other Products, Production Processing Does Not Necessarily Include Testing of All Parameters.

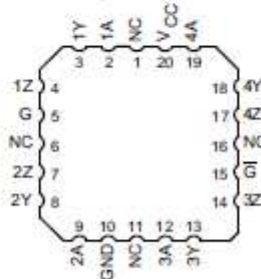
### DESCRIPTION

The AM26LS31 device is a quadruple complementary-output line driver designed to meet the requirements of ANSI TIA/EIA-422-B and ITU (formerly CCITT) Recommendation V.11. The 3-state outputs have high-current capability for driving balanced lines such as twisted-pair or parallel-wire transmission lines, and they are in the high-impedance state in the power-off condition. The enable function is common to all four drivers and offers the choice of an active-high or active-low enable ( $G$ ,  $\overline{G}$ ) input. Low-power Schottky circuitry reduces power consumption without sacrificing speed.

D, DB, N, NS, J, OR W PACKAGE  
(TOP VIEW)



FK PACKAGE  
(TOP VIEW)



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 1979–2014, Texas Instruments Incorporated