République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DU GÉNIE ÉLECTRIQUE ET D'INFORMATIQUE
DÉPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Études
de MASTER ACADEMIQUE

Domaine : Sciences et Technologies
Filière : Automatique
Spécialité : Automatique et Informatique Industrielle

Présenté par

Rayane AKKOUCHE

Thème :

Contrôleur PID d'ordre fractionnaire réglé par réseau de neurones

Mémoire soutenu le .../.../2026 devant le jury composé de :

M. Prénom NOM	Grade, Lieu d'exercice	Président
Mme Karima AMOURA	Grade, UMMTO	Encadrante
M. Prénom NOM	Grade, Lieu d'exercice	Examinateur
M. Prénom NOM	Grade, Lieu d'exercice	Examinateur

Remerciements

Dédicace

Résumé

Ce mémoire porte sur la synthèse d'un contrôleur PID d'ordre fractionnaire (FOPID) dont les paramètres sont réglés à l'aide d'un réseau de neurones artificiels, appliqué à la commande de la vitesse d'un véhicule électrique hybride. L'objectif est d'imposer à la boucle fermée un comportement robuste d'iso-amortissement par l'intermédiaire d'un modèle de référence fractionnaire.

Dans un premier temps, les notions fondamentales des systèmes d'ordre fractionnaire sont présentées, notamment le dérivateur généralisé et son approximation par la méthode d'Oustaloup. Le modèle de référence fractionnaire d'iso-amortissement est ensuite introduit et analysé.

Le deuxième chapitre traite du contrôleur PID classique et du contrôleur FOPID, leurs structures, leurs types et leurs méthodes de réglage, incluant une application numérique. Le troisième chapitre est consacré aux réseaux de neurones artificiels : architecture, types, apprentissage, régularisation et critères de performance.

Le quatrième chapitre présente le modèle du véhicule électrique hybride, sa modélisation et son analyse, ainsi que la conception complète du contrôleur Neuro-FOPID et les résultats de simulation. Le Neuro-FOPID réduit le dépassement à 0.29%, l'ISE d'un facteur 7 par rapport au PID classique, et maintient ces performances sur $\pm 30\%$ de variation de gain.

Mots-clés : Contrôleur FOPID, Réseau de neurones, Modèle de référence fractionnaire, Systèmes d'ordre fractionnaire, Véhicule électrique, Commande robuste, Iso-amortissement.

Table des matières

Table des figures

Liste des tableaux

Liste des Symboles

Symboles Généraux et Mathématiques

s	Variable de Laplace (opérateur de dérivation)
t	Temps (s)
j	Unité imaginaire ($j^2 = -1$)
ω	Pulsation (rad/s)
ω_c	Fréquence de coupure ou de référence (rad/s)
$\Gamma(\cdot)$	Fonction Gamma d'Euler
$e(t)$	Signal d'erreur ($e(t) = r(t) - y(t)$)
$u(t)$	Signal de commande (sortie du contrôleur)
$r(t)$	Consigne (référence)
$y(t)$	Sortie du système

Chapitre I : Systèmes d'Ordre Fractionnaire

D^α	Opérateur de dérivée/intégrale fractionnaire d'ordre α
α, λ, ν	Ordres fractionnaires (exposants réels)
n	Ordre entier (dans les définitions Riemann-Liouville et Caputo)
ω_l	Fréquence basse de la bande d'approximation (rad/s)
ω_h	Fréquence haute de la bande d'approximation (rad/s)
τ_c	Constante de temps du modèle de référence fractionnaire
φ_m	Marge de phase (rad ou deg)
M_p	Dépassement (Overshoot) (%)
T_s	Temps de stabilisation (s)

Chapitre II : Contrôleurs PID et FOPID

K_p	Gain proportionnel
K_i	Gain intégral
K_d	Gain dérivé
T_i	Constante de temps intégrale (s)
T_d	Constante de temps dérivée (s)
N_f	Facteur de filtrage du terme dérivé
λ	Ordre de l'intégration fractionnaire
μ	Ordre de la dérivation fractionnaire
IAE	Integral of Absolute Error
ITAE	Integral of Time-weighted Absolute Error
ISE	Integral of Squared Error
ITSE	Integral of Time-weighted Squared Error

Chapitre III : Réseaux de Neurones Artificiels

w_{ij}	Poids synaptique entre le neurone j et le neurone i
b_i	Biais du neurone i
η	Taux d'apprentissage
$\mathcal{L}$	Fonction de coût (erreur d'entraînement)
β_1, β_2	Coefficients de moment de l'algorithme Adam
ϵ	Constante de stabilité numérique (Adam)

Chapitre IV : Véhicule Électrique et Simulation

v	Vitesse du véhicule (m/s)
θ	Position du papillon des gaz (commande)
F_e	Force générée par le moteur (N)
m	Masse du véhicule (kg)
α_t	Coefficient de traînée aérodynamique (N/(m/s)^2)
τ_e	Constante de temps du moteur (s)
$G_v(s)$	Fonction de transfert du véhicule ($V(s)/\Theta(s)$)
$T_{ref}(s)$	Modèle de référence fractionnaire d'iso-amortissement

Introduction Générale

Depuis l'avènement de l'automatisme industriel, la commande des systèmes dynamiques n'a cessé d'évoluer pour répondre à des exigences toujours plus strictes en termes de précision, de rapidité et de robustesse. Au cœur de cette évolution se trouve le régulateur PID (Proportionnel, Intégral, Dérivé), dont la simplicité structurelle et l'efficacité pratique en ont fait l'outil de régulation dominant. Desborough et Miller rapportent que plus de 97% des contrôleurs industriels sont des régulateurs PID [16]. Cette omniprésence s'explique par sa capacité à bien contrôler une large gamme de processus, sa robustesse de mise en œuvre et son rapport qualité-prix avantageux. Cependant, face à des systèmes dynamiques de plus en plus complexes, non linéaires et soumis à des incertitudes paramétriques importantes, le PID classique trouve ses limites et devient incapable de satisfaire tous les besoins de performance [17].

Depuis les années soixante, l'extension des outils du calcul différentiel aux ordres non entiers a ouvert une nouvelle voie dans la modélisation et la commande des systèmes. La commande CRONE développée par Oustaloup [7] et le contrôleur $\mathrm{PI}^{\lambda}\mathrm{D}^{\mu}$ proposé par Podlubny [8] ont démontré la supériorité des contrôleurs fractionnaires sur leurs homologues entiers, notamment en termes de robustesse vis-à-vis des variations des paramètres du processus. Le PID fractionnaire (FOPID) dispose en effet de deux degrés de liberté supplémentaires — les ordres fractionnaires λ et μ — qui permettent un réglage plus fin et un découplage entre rapidité et amortissement, difficilement atteignable avec un PID classique.

Malgré ses avantages, le réglage optimal d'un contrôleur FOPID à cinq paramètres (K_p, K_i, K_d, λ, μ) demeure une tâche délicate. Les méthodes d'optimisation classiques — programmation quadratique séquentielle (SQP), algorithme du simplexe de Nelder-Mead — sont efficaces mais restent sensibles au point d'initialisation et peuvent converger vers des minima locaux. L'intelligence artificielle, et en particulier les réseaux de neurones artificiels, offre une alternative prometteuse : une fois entraîné hors ligne sur une base de données d'exemples optimaux, un réseau de neurones est capable de fournir instantanément les paramètres optimaux du FOPID pour n'importe quelle condition.

La robustesse constitue sans doute la propriété la plus recherchée dans un système de commande industriel. L'intégrateur fractionnaire d'ordre non entier proposé par Bode dès 1945 [9] comme modèle de boucle ouverte garantit une propriété remarquable

d'iso-amortissement : le dépassement en boucle fermée est insensible aux variations du gain de la chaîne d'action. Ce modèle de référence fractionnaire constitue aujourd'hui un cadre de référence reconnu pour la synthèse de régulateurs robustes [14].

L'application à la commande de vitesse des véhicules électriques illustre parfaitement les enjeux de cette problématique. Ce domaine, en pleine expansion dans le contexte de la transition énergétique mondiale, requiert des stratégies de commande capables de concilier précision, réactivité et robustesse face aux variations de la charge et des conditions routières.

L'objectif de ce travail est de concevoir un contrôleur FOPID dont les cinq paramètres sont calculés automatiquement par un réseau de neurones artificiel entraîné à minimiser un critère d'erreur composite, en imposant à la boucle fermée un comportement fractionnaire robuste d'iso-amortissement. La stratégie de commande est validée par simulation sur le modèle de vitesse d'un véhicule électrique hybride et comparée au PID classique et au FOPID réglé par optimisation numérique directe.

Ce mémoire s'articule en quatre chapitres :

Le **Chapitre I** est consacré aux généralités sur les systèmes d'ordre fractionnaire. Nous y présentons les opérateurs différentiels selon Riemann-Liouville et Caputo, le dérivateur généralisé et sa méthode d'approximation d'Oustaloup, ainsi que le modèle de référence fractionnaire d'iso-amortissement et ses propriétés.

Dans le **Chapitre II**, nous traitons les contrôleurs PID classique et FOPID. Après une description des structures du PID, nous présentons le FOPID, ses différents types, les méthodes de réglage par optimisation et un exemple d'application numérique.

Le **Chapitre III** est entièrement consacré aux réseaux de neurones artificiels. Nous y détaillons le neurone formel, les différents types de réseaux (MLP, CNN, RNN...), les fonctions d'activation, l'algorithme de rétropropagation et les critères d'erreur, ainsi que les architectures de commande neuronale pour les systèmes dynamiques.

Le **Chapitre IV** présente la modélisation du véhicule électrique hybride ainsi que la conception complète et les résultats de simulation du contrôleur Neuro-FOPID, incluant la comparaison avec les contrôleurs de référence, les tests de robustesse et le rejet de perturbation.

Enfin, une **Conclusion Générale** résume les principaux résultats et propose des perspectives de travaux futurs.

Chapitre 1

Simulation des systèmes d'ordre fractionnaires

1.1 Introduction

Le calcul d'ordre fractionnaire est un domaine qui traite des dérivées et des intégrales d'ordre non entier. Avec l'avancée de la science et le développement de l'outil informatique, l'intérêt de la dérivation et de l'intégration d'ordre non entier ne cesse de croître, notamment dans le domaine de l'automatique pour la modélisation, l'identification et la commande des systèmes [5, 8].

Dans le domaine du contrôle, les systèmes d'ordre fractionnaire apportent un réel avantage. Un bon exemple est l'utilisation de la fonction idéale de Bode [9] comme modèle de référence robuste : en raison de ses caractéristiques d'iso-amortissement, elle sert souvent de référence pour ajuster les contrôleurs PID fractionnaires [5]. Ce type de contrôleur offre une meilleure robustesse et de meilleures performances lorsque les conditions changent ou que le système est complexe.

Ce chapitre a pour but de se familiariser avec l'univers des systèmes d'ordre fractionnaire. On y présente certains fondements clés ainsi que la méthode d'approximation du dérivateur généralisé d'Oustaloup [7], largement utilisée pour la simulation des systèmes d'ordre fractionnaire [5, 26].

1.2 Systèmes d'ordre fractionnaire

Un système d'ordre fractionnaire est décrit par une équation différentielle impliquant des dérivées fractionnaires :

$$a_n D^{\alpha_n} y(t) + \cdots + a_1 D^{\alpha_1} y(t) = b_m D^{\beta_m} u(t) + \cdots + b_0 D^{\beta_0} u(t), \tag{1.1}$$

où D^α représente l'opérateur de dérivée d'ordre $\alpha \in \mathbb{R}^+$, défini au sens de Riemann-Liouville ou Caputo [1, 8]. Ces formalismes capturent des dynamiques non exponentielles, comme les relaxations en loi de puissance.

1.2.1 Opérateurs différentiels fractionnaires

Un opérateur fractionnaire généralise les dérivées et intégrales classiques à des ordres réels. Pour $\alpha > 0$, il correspond à une dérivation d'ordre α ; pour $\alpha < 0$, il correspond à une intégration d'ordre $|\alpha|$.

Dérivée de Riemann-Liouville [8] :

$$D^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}}\, d\tau, \quad n-1 < \alpha < n, \tag{1.2}$$

où n est un entier tel que $n - 1 < \alpha < n$.

Dérivée de Caputo [1] :

$$D^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha-n+1}}\, d\tau, \quad n-1 < \alpha < n. \tag{1.3}$$

La formulation de Caputo est préférable en commande car elle permet d'utiliser des conditions initiales exprimées en termes de dérivées entières, plus accessibles physiquement. La fonction Γ est la généralisation de la factorielle aux réels :

$$\Gamma(\nu) = \int_0^{+\infty} e^{-x}\, x^{\nu-1}\, dx. \tag{1.4}$$

1.2.2 Dérivateur généralisé

Le dérivateur généralisé est un opérateur linéaire noté s^α ($\alpha \in \mathbb{R}$), généralisant la dérivation et l'intégration. Une variante courante inclut une fréquence de coupure ω_c [8] :

$$D_{\text{gen}}(s) = \left(\frac{s}{\omega_c}\right)^\alpha, \quad \alpha \in \mathbb{R}. \tag{1.5}$$

Dans le domaine fréquentiel :

$$D_{\text{gen}}(j\omega) = \left(\frac{j\omega}{\omega_c}\right)^\alpha = \left|\frac{\omega}{\omega_c}\right|^\alpha e^{j\alpha\pi/2}. \tag{1.6}$$

Cette forme préserve un gain en $(\omega/\omega_c)^\alpha$ et une phase constante de $\alpha\pi/2$. Cette phase constante sur une large plage de fréquences rend sa réalisation physique complexe et nécessite des approximations à bande limitée.

1.2.3 Méthodes d'approximation du dérivateur généralisé

Le dérivateur généralisé s^α constitue l'élément principal pour la synthèse des systèmes fractionnaires. Sa réalisation physique directe étant impossible (phase constante sur une bande infinie), plusieurs méthodes d'approximation rationnelle à bande limitée ont été proposées dans la littérature. On peut notamment citer : la méthode de Charef [2], basée sur une cascade récursive de pôles et zéros à distribution logarithmique ; la méthode de Carlson, fondée sur un développement en fraction continue [4] ; et la méthode d'Oustaloup [7], qui fournit la meilleure précision sur une bande de fréquences large grâce à une répartition symétrique des singularités. Dans ce travail, la méthode d'Oustaloup est retenue pour la simulation de toutes les lois de commande.

1.2.4 Méthode d'Oustaloup

a. Principe général La méthode d'Oustaloup [7] est une technique d'approximation fréquentielle qui consiste à remplacer l'opérateur fractionnaire idéal s^α ($\alpha \in \mathbb{R}$) par une fonction de transfert rationnelle d'ordre entier dont la réponse fréquentielle coïncide avec celle de s^α sur une bande de fréquences prédéfinie $[\omega_l, \omega_h]$. L'idée fondatrice est que, sur cette bande limitée, le gain et la phase d'un système entier bien construit peuvent imiter fidèlement le comportement en loi de puissance de l'opérateur fractionnaire.

Formellement, l'approximation est de la forme :

$$s^\alpha \approx H_{\text{Oustaloup}}(s) = C \prod_{k=1}^{N} \frac{s + \omega_{z_k}}{s + \omega_{p_k}}, \tag{1.7}$$

où C est un gain de normalisation, et $\{\omega_{z_k}\}$, $\{\omega_{p_k}\}$ sont respectivement les zéros et les pôles de l'approximation, réels et strictement positifs.

b. Construction de l'approximation La méthode repose sur la répartition géométrique de $2N+1$ pôles et $2N+1$ zéros sur la bande $[\omega_l, \omega_h]$, symétriquement autour de la fréquence centrale $\omega_u = \sqrt{\omega_l \omega_h}$. La formulation standard s'écrit [5, 7] :

$$H_{\text{Oustaloup}}(s) = \omega_h^\alpha \prod_{k=-N}^{N} \frac{1 + s/\omega_{z_k}}{1 + s/\omega_{p_k}}, \tag{1.8}$$

où le facteur ω_h^α assure la normalisation du gain à la fréquence haute de la bande. Les fréquences des zéros et des pôles sont définies par les lois récursives :

$$\omega_{z_k} = \omega_l \left(\frac{\omega_h}{\omega_l}\right)^{\frac{k+N+1-\alpha/2}{2N+1}}, \qquad \omega_{p_k} = \omega_l \left(\frac{\omega_h}{\omega_l}\right)^{\frac{k+N+1+\alpha/2}{2N+1}}, \tag{1.9}$$

pour $k = -N, -N+1, \ldots, N$. Ces expressions montrent que les pôles et zéros sont distribués en progression géométrique sur l'axe fréquentiel, avec un rapport commun $\eta = (\omega_h/\omega_l)^{1/(2N+1)}$.

Une propriété importante est que les pôles et zéros sont entrelacés : entre deux pôles consécutifs se trouve exactement un zéro, et vice versa. Cette alternance est la clé du mécanisme d'approximation : chaque paire $(\omega_{z_k}, \omega_{p_k})$ produit localement une contribution de phase de $+\alpha\pi/(2N+1)$, et leur accumulation sur toute la bande tend vers la valeur cible $\alpha\pi/2$.

c. Propriétés fréquentielles Sur la bande d'approximation $[\omega_l, \omega_h]$, la réponse fréquentielle de $H_{\text{Oustaloup}}(j\omega)$ présente deux propriétés essentielles :

- **Gain :** $|H_{\text{Oustaloup}}(j\omega)| \approx (\omega/\omega_l)^{\alpha}$, soit une pente de 20α dB/décade, identique à celle de l'opérateur idéal s^{α}.
- **Phase :** $\angle H_{\text{Oustaloup}}(j\omega) \approx \alpha\pi/2$, une phase quasi-constante sur toute la bande, caractéristique de l'iso-amortissement.

Hors de la bande $[\omega_l, \omega_h]$, l'approximation diverge du comportement idéal : le gain tend vers une constante aux très basses fréquences et la phase retombe à zéro aux très hautes fréquences. Il est donc essentiel de choisir $[\omega_l, \omega_h]$ de sorte qu'elle englobe toute la plage de fréquences significative du système à commander.

d. Choix des paramètres Les paramètres de réglage de la méthode sont au nombre de trois : l'ordre d'approximation N, la fréquence basse ω_l et la fréquence haute ω_h.

L'ordre N contrôle directement la précision de l'approximation. Le tableau 1.1 récapitule l'erreur maximale de phase sur la bande en fonction de N pour $\alpha = 0.5$, d'après [5].

TABLE 1.1 – Erreur maximale de phase de l'approximation d'Oustaloup en fonction de N (exemple : $\alpha = 0.5$, bande $[10^{-2}, 10^{2}]$ rad/s) [5].

N	Nombre de pôles/zéros	Erreur phase max.	Erreur gain max.
2	5	$\sim 5.0°$	~ 3.0 dB
3	7	$\sim 2.5°$	~ 1.5 dB
5	11	$\sim 0.5°$	< 2.0 dB
7	15	$\sim 0.1°$	< 0.5 dB

Un choix standard est $N = 5$ avec la bande $[\omega_l, \omega_h] = [10^{-2}, 10^{2}]$ rad/s, ce qui produit 11 pôles et 11 zéros et garantit une erreur inférieure à $0.5°$ en phase et 2 dB en gain sur quatre décades [5]. En pratique, la bande est adaptée à la dynamique du système : on choisit $\omega_l \ll \omega_c$ et $\omega_h \gg \omega_c$, où ω_c est la fréquence de coupure du système en boucle ouverte.

e. Exemple numérique Considérons l'approximation de $s^{0.5}$ sur la bande $[10^{-2}, 10^{2}]$ rad/s avec $N = 3$ (7 pôles et 7 zéros). La progression géométrique donne un rapport $\eta = (10^{2}/10^{-2})^{1/7} = 10^{4/7} \approx 3.73$. Les fréquences calculées selon (1.9) sont :

TABLE 1.2 – Zéros et pôles de l'approximation d'Oustaloup pour $s^{0.5}$, $N = 3$, $\omega_l = 10^{-2}$, $\omega_h = 10^{2}$ rad/s.

k	ω_{z_k} (rad/s)	ω_{p_k} (rad/s)
-3	1.26×10^{-2}	4.70×10^{-2}
-2	4.70×10^{-2}	1.75×10^{-1}
-1	1.75×10^{-1}	6.52×10^{-1}
0	6.52×10^{-1}	2.43×10^{0}
1	2.43×10^{0}	9.07×10^{0}
2	9.07×10^{0}	3.38×10^{1}
3	3.38×10^{1}	1.26×10^{2}

On vérifie l'entrelacement : chaque zéro ω_{z_k} coïncide avec le pôle de l'indice précédent $\omega_{p_{k-1}}$, confirmant la structure alternée de l'approximation. La phase résultante sur $[10^{-2}, 10^{2}]$ rad/s est quasi-constante et vaut approximativement $0.5 \times 90^\circ = 45^\circ$, avec une ondulation inférieure à 2.5°.

f. Diagramme de Bode de l'approximation La figure 1.1 compare le diagramme de Bode de l'opérateur idéal $s^{0.5}$ et de son approximation d'Oustaloup d'ordre $N = 5$ sur la bande $[10^{-2}, 10^{2}]$ rad/s.

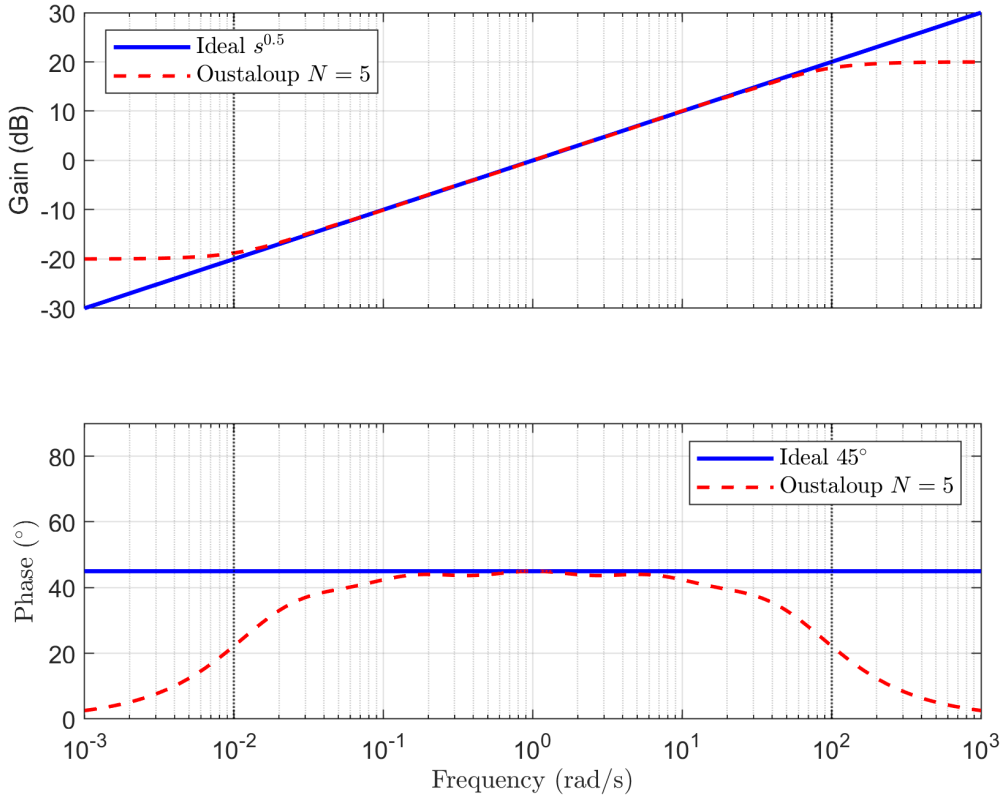

FIGURE 1.1 – Diagramme de Bode comparatif : opérateur idéal $s^{0,5}$ (bleu) et approximation d'Oustaloup d'ordre $N = 5$ (rouge tirets) sur la bande $[\omega_l, \omega_h] = [10^{-2}, 10^{2}]$ rad/s. La phase de l'approximation reste dans une bande de $\pm 2°$ autour de la valeur cible de 45°.

1.2.5 Modèle de référence en boucle fermée

Considérons un système en boucle ouverte décrit par la fonction idéale de Bode $L(s) = 1/(\tau_c s^{\lambda+1})$. La fonction de transfert en boucle fermée unitaire est obtenue par la relation standard $F(s) = L(s)/(1 + L(s))$, soit :

$$F(s) = \frac{1/\tau_c s^{\lambda+1}}{1 + 1/\tau_c s^{\lambda+1}} = \frac{1}{\tau_c s^{\lambda+1} + 1}, \quad 0 < \lambda < 1. \tag{1.10}$$

L'équation caractéristique de ce système est $\tau_c s^{\lambda+1} + 1 = 0$, soit $s^{\lambda+1} = -1/\tau_c$. Puisque $\lambda + 1 \in (1, 2)$, le dénominateur est un polynôme fractionnaire d'ordre non entier, ce qui distingue $F(s)$ de tout système d'ordre entier. Dans le domaine fréquentiel, $F(j\omega)$ présente :

— un gain qui décroît en $(\omega\tau_c)^{-(\lambda+1)}$ au-delà de la fréquence de coupure $\omega_c = \tau_c^{-1/(\lambda+1)}$, soit une pente de $-20(\lambda + 1)$ dB/décade ;

— une phase qui se stabilise autour de $-(\lambda + 1)\pi/2$ sur une large plage de fréquences, produisant le plateau de phase caractéristique de l'iso-amortissement.

En boucle fermée, le comportement temporel est analogue à celui d'un système du

second ordre amorti, avec deux paramètres ajustables indépendamment : λ contrôle l'amortissement (et donc le dépassement) tandis que τ_c fixe la rapidité (temps de montée et de stabilisation).

1.2.6 Analyse des paramètres du modèle de référence en boucle fermée

En fixant τ_c et en faisant varier λ, le dépassement augmente avec λ tandis que la rapidité reste inchangée. En fixant λ et en faisant varier τ_c, la rapidité change mais le dépassement reste constant — c'est la **propriété d'iso-amortissement**. Les formules de correspondance sont [5, 14] :

$$M_p\,(\%) = 0.8\lambda(\lambda + 0.25), \quad T_s(5\%) \approx \frac{3}{\cos\!\left(\pi - \frac{\pi}{\lambda+1}\right)}\tau_c, \quad \phi_\alpha = \pi - (\lambda+1)\frac{\pi}{2}. \quad (1.11)$$

Exemple numérique. Fixons $\lambda = 0.5$ et faisons varier τ_c. Le dépassement prédit est $M_p = 0.8 \times 0.5 \times (0.5 + 0.25) = 0.30\%$, et la marge de phase est $\phi_\alpha = \pi - 1.5 \times \pi/2 = \pi/4 \approx 45°$. Le tableau 1.3 illustre l'invariance du dépassement lorsque seul τ_c change.

TABLE 1.3 – Invariance du dépassement par rapport à τ_c pour $\lambda = 0.5$ fixé.

τ_c	M_p (%)	T_s (s)	ϕ_α (°)
1.0	0.30	$3/\cos(\pi/4) \approx 4.24$	45
2.0	0.30	$6/\cos(\pi/4) \approx 8.49$	45

Doubler τ_c double le temps de stabilisation sans affecter ni le dépassement ni la marge de phase. C'est précisément cette propriété qui est exploitée au Chapitre IV pour garantir la robustesse du Neuro-FOPID face aux variations de gain.

1.2.7 Robustesse du modèle de référence fractionnaire

La propriété fondamentale qui confère au modèle fractionnaire sa robustesse est l'invariance de la marge de phase vis-à-vis du gain de la boucle ouverte. Pour le système $L(s) = K_0/(\tau_c s^{\lambda+1})$ avec une incertitude de gain K_0, la fréquence de coupure ω_c est définie par $|L(j\omega_c)| = 1$, soit $\omega_c = (K_0/\tau_c)^{1/(\lambda+1)}$. La marge de phase est alors :

$$\phi_m = \pi + \angle L(j\omega_c) = \pi - (\lambda+1)\frac{\pi}{2}, \quad (1.12)$$

qui est *indépendante de* K_0. En effet, si le gain augmente, ω_c se déplace vers des fréquences plus élevées, mais la phase de $L(j\omega) = -j(\lambda+1)\pi/2$ est constante sur toute la bande : le dépassement en boucle fermée reste donc identique quelle que soit la valeur de K_0.

En revanche, pour un système d'ordre entier tel que $L(s) = K_0/(s(\tau s + 1))$, une variation de gain K_0 déplace la fréquence de coupure vers une zone où la phase n'est plus constante, modifiant ainsi la marge de phase et le dépassement en boucle fermée. C'est cette différence structurelle fondamentale qui justifie la supériorité du modèle fractionnaire comme référence robuste [5, 14].

Cette propriété d'iso-amortissement est directement exploitée au Chapitre IV pour la conception du contrôleur Neuro-FOPID : le RNA est entraîné à reproduire les paramètres FOPID qui imposent ce comportement fractionnaire robuste à la boucle fermée du véhicule électrique, garantissant des performances stables sous $\pm 30\%$ de variation du gain.

1.3 Conclusion

Dans ce chapitre, nous avons présenté les fondements des systèmes d'ordre fractionnaire : le dérivateur généralisé, les opérateurs de Riemann-Liouville et Caputo, et la méthode d'approximation d'Oustaloup. Le modèle de référence fractionnaire d'iso-amortissement a été présenté en boucle fermée avec une dérivation complète de la fonction de transfert, un exemple numérique illustrant l'invariance du dépassement, et une justification mathématique de l'indépendance de la marge de phase vis-à-vis du gain.

Chapitre 2

Contrôleurs PID Classique et FOPID

2.1 Introduction

La régulation est l'âme technique de l'automatisme. Le contrôleur PID s'est imposé comme le cœur de l'automatisme industriel depuis près d'un siècle, grâce à sa structure simple et son efficacité démontrée. Cependant, face aux systèmes complexes, le PID classique présente des limitations qui ont motivé le développement de techniques plus avancées [17, 18].

Le PID fractionnaire (FOPID), ou $\mathrm{PI}^{\lambda}\mathrm{D}^{\mu}$, se présente comme une extension puissante intégrant des ordres fractionnaires (λ et μ) aux actions intégrale et dérivée. Ses cinq paramètres (K_p, K_i, K_d, λ, μ) offrent une flexibilité de réglage supérieure, mais au prix d'une complexité accrue [5, 8].

Dans ce chapitre, nous présentons les contrôleurs PID et FOPID, leurs structures et leurs propriétés, ainsi que les méthodes de réglage par optimisation, avec un exemple d'application numérique et une analyse fréquentielle comparative.

2.2 PID Classique

En automatique, le régulateur PID améliore les performances d'un asservissement en corrigeant les écarts entre la consigne et la sortie. Il possède trois actions fondamentales [6].

Action proportionnelle (P) : L'action proportionnelle réagit immédiatement à l'écart entre la consigne et la sortie en produisant une commande proportionnelle à cet écart. Elle améliore la rapidité mais laisse subsister une erreur statique résiduelle :

$$u_p(t) = K_p\, e(t). \tag{2.1}$$

Action intégrale (I) : L'action intégrale accumule l'erreur au cours du temps et supprime l'erreur statique en régime permanent. Un gain K_i trop élevé peut cependant provoquer des oscillations et un phénomène de saturation de l'intégrateur :

$$u_i(t) = K_i \int_0^t e(\tau)\, d\tau. \tag{2.2}$$

Action dérivée (D) : L'action dérivée anticipe les variations futures de l'erreur en réagissant à sa vitesse de changement. Elle réduit les dépassements et améliore la stabilité transitoire, mais reste sensible au bruit de mesure :

$$u_d(t) = K_d \frac{d}{dt} e(t). \tag{2.3}$$

La commande globale est la somme de ces trois actions :

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t), \tag{2.4}$$

soit en Laplace : $C(s) = K_p + K_i/s + K_d s$.

2.2.1 Différentes structures du PID

Le PID classique possède trois structures principales qui diffèrent par l'organisation des actions P, I et D.

Structure parallèle [6] : Chaque terme agit directement sur l'erreur. C'est la structure la plus utilisée, qui facilite le réglage indépendant de chaque paramètre. Sa fonction de transfert est :

$$C(s) = K_p + \frac{K_i}{s} + K_d s. \tag{2.5}$$

Structure série [6] : Le régulateur est disposé comme une chaîne de blocs successifs, avec une interaction naturelle entre paramètres. Sa fonction de transfert est :

$$C(s) = K \left(1 + \frac{1}{T_i s}\right)(1 + T_d s). \tag{2.6}$$

Structure mixte [6] : La structure mixte applique l'action dérivée sur la sortie plutôt que sur l'erreur, réduisant la sensibilité au bruit de mesure. Sa fonction de transfert est :

$$C(s) = K_p \left(1 + \frac{1}{T_i s}\right) - K_p T_d s. \tag{2.7}$$

2.3 PID d'ordre fractionnaire (FOPID)

2.3.1 Définition et représentation dans l'espace des paramètres

Le FOPID est une extension du PID classique où les actions intégrale et dérivée sont d'ordre fractionnaire [8] :

$$C(s) = K_p + \frac{K_i}{s^\lambda} + K_d s^\mu, \tag{2.8}$$

où $\lambda > 0$ est l'ordre de l'intégration et $\mu > 0$ est l'ordre de la dérivation. Pour $\lambda = \mu = 1$ on retrouve le PID classique. La figure 2.1 illustre l'espace des paramètres (λ, μ).

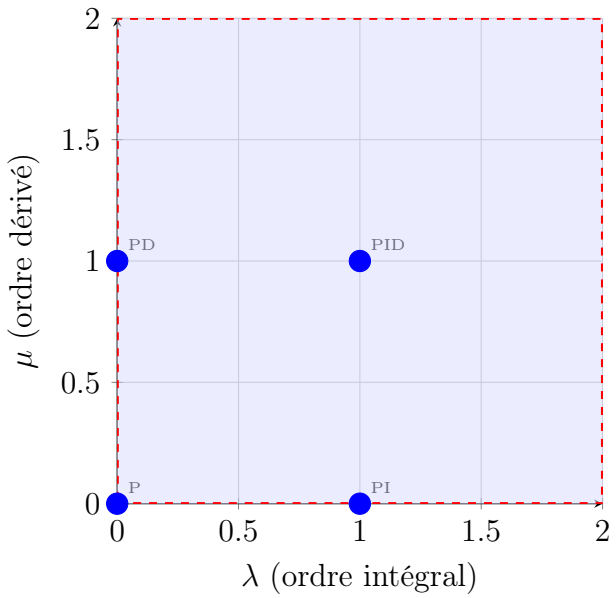

FIGURE 2.1 – Espace des paramètres (λ, μ) du contrôleur $PI^\lambda D^\mu$ [8].

2.3.2 Différents types du PID fractionnaire

Plusieurs variantes du FOPID sont proposées dans la littérature, chacune adaptée à des contextes d'application spécifiques.

$PI^\lambda D^\mu$ basique : La forme standard $C(s) = K_p + K_i s^{-\lambda} + K_d s^\mu$ offre cinq paramètres libres pour un réglage flexible.

FOPID avec filtre : Pour rendre s^μ physiquement réalisable et réduire la sensibilité au bruit :

$$C(s) = K_p + K_i s^{-\lambda} + \frac{K_d s^\mu}{1 + \tau s}. \tag{2.9}$$

FOF-PID [19] : Un PID classique en série avec un filtre fractionnaire issu de la combinaison IMC-CRONE :

$$C(s) = F(s)\left[K_c\left(\frac{1}{\tau_i s} + \tau_d s + 1\right)\right], \quad F(s) = \frac{1}{1+\tau_f s^\gamma}. \tag{2.10}$$

2.4 Méthodes de réglage du PID et FOPID

2.4.1 Méthodes de réglage classiques du PID

Il existe plusieurs méthodes pour calculer les paramètres du PID classique. Parmi les plus répandues, on distingue la méthode de Ziegler-Nichols, la méthode par modèle interne (IMC), et les méthodes d'optimisation numérique.

Méthode de Ziegler-Nichols [20] : Proposée en 1942, la méthode de Ziegler-Nichols est l'une des premières procédures systématiques de réglage du PID. Elle repose sur une expérience en boucle fermée : on augmente progressivement le gain proportionnel K_p jusqu'à ce que la sortie oscille de façon entretenue à la fréquence ω_u. Le gain correspondant K_{cr} (gain critique) et la période d'oscillation $T_u = 2\pi/\omega_u$ permettent de calculer les paramètres du régulateur selon le tableau suivant [20] :

TABLE 2.1 – Règles de réglage de Ziegler-Nichols (méthode du gain critique).

Type	K_p	T_i	T_d
P	$0.50\,K_{cr}$	–	–
PI	$0.45\,K_{cr}$	$0.83\,T_u$	–
PID	$0.60\,K_{cr}$	$0.50\,T_u$	$0.125\,T_u$

Bien qu'efficace sur de nombreux procédés industriels, cette méthode tend à produire des réponses oscillatoires avec un dépassement élevé ($\sim 25\%$). Elle constitue cependant un excellent point de départ pour un réglage fin ultérieur, et reste l'une des références historiques incontournables de l'automatisme.

Méthode de commande à modèle interne (IMC) [27] : L'IMC *(Internal Model Control)* repose sur l'idée d'incorporer explicitement un modèle du procédé dans la structure du contrôleur. Si le modèle est parfait et qu'aucune perturbation n'existe, la sortie suit la consigne sans erreur. En pratique, un filtre IMC de paramètre λ_f est introduit pour assurer la robustesse face aux incertitudes :

$$C_{\text{IMC}}(s) = \frac{Q(s)}{1 - Q(s)\tilde{G}(s)}, \quad Q(s) = \tilde{G}^{-1}(s)\frac{1}{(\lambda_f s + 1)^r}, \tag{2.11}$$

où $\tilde{G}(s)$ est le modèle interne du procédé et r l'ordre du filtre. Cette structure produit directement un PID lorsque $\tilde{G}(s)$ est un modèle du premier ordre avec retard, établissant une correspondance directe entre le paramètre de réglage λ_f et les marges de robustesse. Ces méthodes classiques ont été étendues au réglage du FOPID, notamment la variante FOF-PID [19] basée sur IMC-CRONE.

2.4.2 Méthodes d'optimisation pour le réglage du FOPID

Les méthodes d'optimisation déterminent les paramètres optimaux du contrôleur en minimisant un critère de performance prédéfini. Les quatre critères d'erreur intégraux classiquement utilisés sont [11] :

$$\text{IAE} = \int_0^T |e(t)|\, dt \qquad (\textit{Integral of Absolute Error}), \tag{2.12}$$

$$\text{ITAE} = \int_0^T t\,|e(t)|\, dt \qquad (\textit{Integral of Time-weighted Absolute Error}), \tag{2.13}$$

$$\text{ISE} = \int_0^T e^2(t)\, dt \qquad (\textit{Integral of Squared Error}), \tag{2.14}$$

$$\text{ITSE} = \int_0^T t\, e^2(t)\, dt \qquad (\textit{Integral of Time-weighted Squared Error}). \tag{2.15}$$

L'IAE pénalise uniformément toutes les erreurs quelle que soit leur durée ; l'ITAE pénalise davantage les erreurs persistantes en les pondérant par le temps t, favorisant une convergence rapide vers zéro ; l'ISE amplifie les grandes erreurs au détriment des petites en les élevant au carré ; l'ITSE combine les deux effets. Le critère retenu dans ce travail est l'ISE, auquel sont ajoutées des pénalités sur le dépassement et l'erreur statique (voir section 4.7).

Méthode SQP [15] : La programmation quadratique séquentielle résout séquentiellement des sous-problèmes quadratiques approximant le problème original :

$$\min_d \ \nabla f(x_k)^T d + \tfrac{1}{2} d^T H_k d, \quad \text{s.c. } c(x_k) + \nabla c(x_k)^T d \leq 0. \tag{2.16}$$

La convergence est super-linéaire sous des hypothèses de régularité classiques.

Méthode de Nelder-Mead [13] : Un algorithme sans gradient utilisant un simplexe de $n+1$ points qui évolue par réflexion, expansion et contraction. Robuste aux discontinuités mais sans garantie de convergence globale.

Limitation commune : Ces méthodes dépendent du point d'initialisation. Elles peuvent converger vers des minima locaux, ce qui motive l'approche par réseau de

neurones de la section 4.6.

2.4.3 Exemple d'application : réglage d'un FOPID par `fmincon`

Considérons le système de troisième ordre de Barbosa et al. [14] : $G(s) = 1/(s+1)^3$, avec le modèle de référence $F(s) = 1/(1 + \tau_c s^{\lambda+1})$ ($\tau_c = 1$, $\lambda = 0.25$).

TABLE 2.2 – Paramètres des contrôleurs obtenus par optimisation (système de Barbosa).

Paramètre	**FOPID**	**PID Classique**
K_c	4.0355	2.5891
T_i	1.2308	1.6625
T_d	2.5781	2.7760
λ	1.1906	N/A
μ	1.3616	N/A

La figure 2.2 compare les réponses indicielles des trois systèmes. Le FOPID reproduit fidèlement le modèle de référence grâce à ses deux degrés de liberté supplémentaires, tandis que le PID classique présente des oscillations amorties plus marquées.

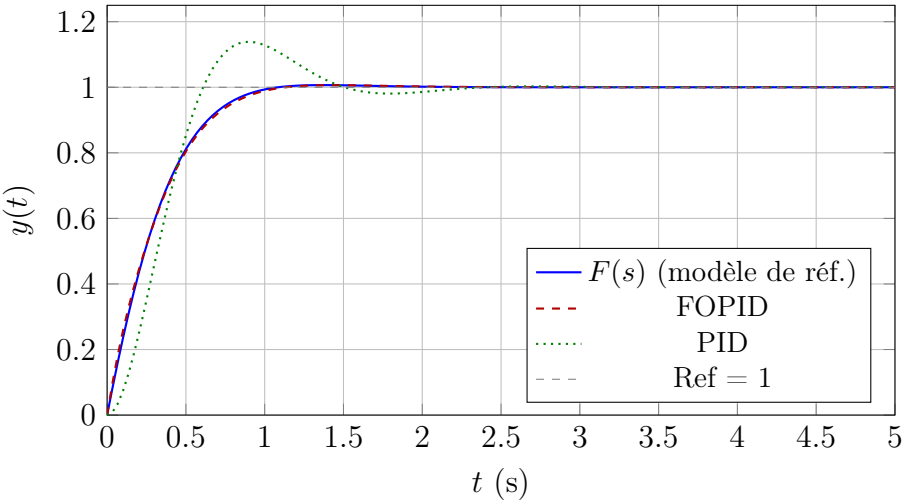

FIGURE 2.2 – Réponses indicielles : PID, FOPID et modèle de référence $F(s)$ (système de Barbosa, $\tau_c = 1$, $\lambda = 0.25$). L'axe temporel est étendu jusqu'à $t = 5$ s pour montrer l'atteinte complète du régime permanent.

TABLE 2.3 – Comparaison des performances temporelles : PID vs FOPID vs modèle de référence.

Critère	**Modèle** $F(s)$	**FOPID**	**PID**
Dépassement M_p (%)	0	< 5	~ 15
Suivi du modèle de réf.	—	excellent	approximatif
λ / μ	—	1.19 / 1.36	N/A

Ces résultats confirment la supériorité paramétrique du FOPID. La complexité de son réglage motive le recours au réseau de neurones présenté au Chapitre III.

2.5 Conclusion

Dans ce chapitre, nous avons présenté les contrôleurs PID classique et FOPID, leurs structures et leurs propriétés. Les deux degrés de liberté supplémentaires (λ et μ) du FOPID offrent une flexibilité accrue pour le découplage précision-robustesse. L'analyse fréquentielle a confirmé que la phase quasi-constante du FOPID sur une large bande de fréquences est le mécanisme sous-jacent de sa robustesse structurelle, propriété directement issue de la fonction idéale de Bode. Les méthodes d'optimisation classiques, bien qu'efficaces, restent sensibles à l'initialisation — ce qui motive le recours aux réseaux de neurones du chapitre suivant.

Chapitre 3

Réseaux de Neurones Artificiels

3.1 Introduction

Les **réseaux de neurones artificiels** (RNA) constituent aujourd'hui l'un des outils les plus puissants de l'apprentissage automatique. Inspirés du fonctionnement du cerveau biologique, ils sont capables d'apprendre des représentations complexes à partir de données et d'approximer des fonctions non linéaires arbitrairement complexes [22]. Leur capacité à généraliser à partir d'exemples en fait des candidats naturels pour résoudre des problèmes d'optimisation, d'identification et de régression difficiles.

Ce chapitre présente les fondements théoriques des réseaux de neurones artificiels : le neurone formel, les différents types de réseaux, les fonctions d'activation, l'algorithme de rétropropagation, les techniques de régularisation et les critères d'évaluation.

3.2 Le Neurone Formel

3.2.1 Modèle de McCulloch-Pitts

Le neurone formel, introduit par McCulloch et Pitts en 1943, est l'unité de calcul élémentaire d'un réseau de neurones [21]. Il réalise une somme pondérée de ses entrées, augmentée d'un biais, puis applique une fonction d'activation non linéaire :

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}\, x_j + b_i\right) = f(z_i), \tag{3.1}$$

où w_{ij} est le poids synaptique, b_i le biais, z_i la préactivation, et $f(\cdot)$ la fonction d'activation. En notation matricielle : $\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$.

3.3 Types de Réseaux de Neurones

Les réseaux de neurones se déclinent en de nombreuses architectures, chacune adaptée à des types de problèmes spécifiques.

3.3.1 Perceptron Simple

Le perceptron simple est le modèle le plus élémentaire : un neurone unique à seuil qui ne peut résoudre que des problèmes linéairement séparables [21].

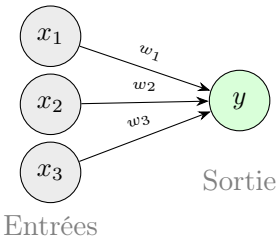

FIGURE 3.1 – Architecture du perceptron simple.

3.3.2 Perceptron Multi-Couche (MLP)

Le Perceptron Multi-Couche (MLP) est l'architecture la plus répandue pour la régression non linéaire, organisée en couche d'entrée, couches cachées et couche de sortie [21]. Le théorème d'approximation universelle [22] garantit qu'un MLP avec une seule couche cachée suffisamment large peut approximer n'importe quelle fonction continue.

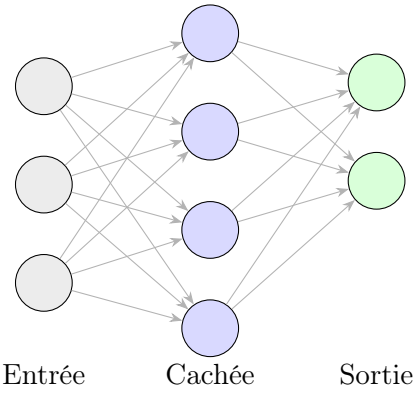

FIGURE 3.2 – Architecture d'un Perceptron Multi-Couche (MLP) à une couche cachée.

3.3.3 Réseaux de Neurones Récurrents (RNN)

Les RNN introduisent des connexions récurrentes permettant de traiter des séquences temporelles [21]. L'état caché à l'instant t est calculé à partir de l'entrée cou-

rante et de l'état précédent :

$$\mathbf{h}_t = f(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}). \tag{3.2}$$

Les réseaux LSTM résolvent le problème du gradient évanescent des RNN classiques.

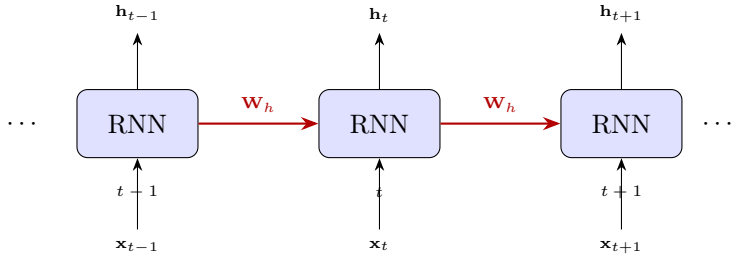

FIGURE 3.3 – Réseau de neurones récurrent déplié dans le temps — les connexions rouges transmettent l'état caché.

3.3.4 Réseaux de Neurones Convolutifs (CNN)

Les CNN utilisent le partage de poids pour traiter des données à structure locale. Une couche convolutive calcule, pour le filtre k :

$$Z_{ij}^{(k)} = \sum_{m,n} W_{mn}^{(k)} X_{(i+m)(j+n)} + b^{(k)}, \tag{3.3}$$

suivie d'une activation non linéaire et d'une opération de pooling.

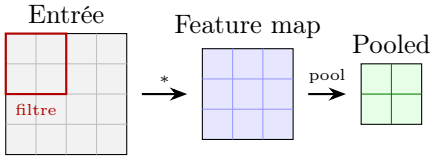

FIGURE 3.4 – Principe d'une couche convolutive : convolution + pooling.

3.4 Fonctions d'Activation

Les fonctions d'activation brisent la linéarité du réseau et lui confèrent sa capacité d'approximation universelle. Leur choix influence directement la vitesse de convergence et les performances du réseau.

3.4.1 Sigmoïde logistique

La sigmoïde logistique est définie par $\sigma_{\text{sig}}(x) = 1/(1+e^{-x}) \in (0,1)$, avec dérivée $\sigma'_{\text{sig}}(x) = \sigma_{\text{sig}}(x)(1-\sigma_{\text{sig}}(x))$. La sigmoïde souffre du problème de gradient évanescent pour $|x| \gg 1$.

3.4.2 Tangente hyperbolique

$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \in (-1, 1)$, centrée en zéro ce qui accélère la convergence. Sa dérivée est $\tanh'(x) = 1 - \tanh^2(x)$.

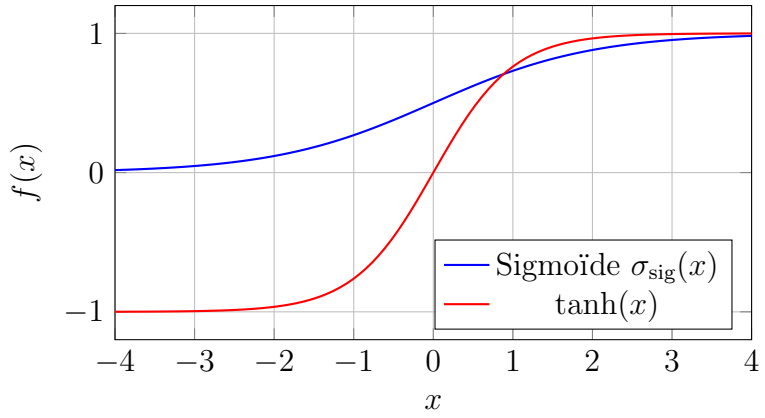

FIGURE 3.5 – Fonctions d'activation sigmoïde et tangente hyperbolique.

3.4.3 Rectified Linear Unit (ReLU)

$\text{ReLU}(x) = \max(0, x)$, fonction standard pour les couches cachées des réseaux profonds. Elle ne sature pas pour $x > 0$, atténuant le gradient évanescent. La Leaky ReLU améliore encore ceci en ajoutant une pente $\alpha_{\text{lr}} x$ pour $x \leq 0$.

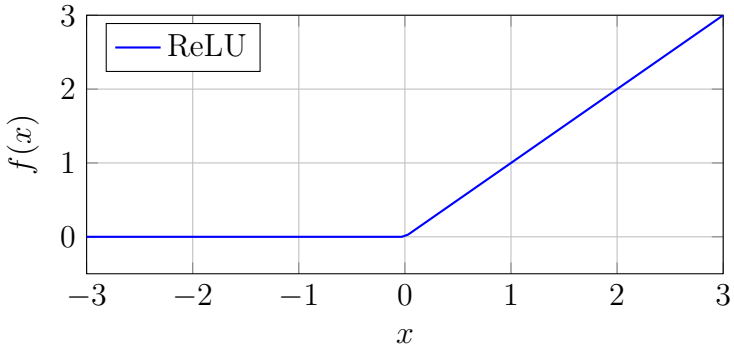

FIGURE 3.6 – Fonction d'activation ReLU.

TABLE 3.1 – Comparaison des principales fonctions d'activation.

Fonction	**Plage**	**Centrage**	**Grad. évanescent**	**Calcul**
Sigmoïde	$(0, 1)$	Non	Oui (fort)	Moyen
tanh	$(-1, 1)$	Oui	Oui (modéré)	Moyen
ReLU	$[0, +\infty)$	Non	**Non** (pour $x > 0$)	**Faible**
Leaky ReLU	$(-\infty, +\infty)$	Non	Non	Faible

3.5 Apprentissage par Rétropropagation du Gradient

3.5.1 Fonction de coût

L'apprentissage consiste à minimiser la MSE (erreur quadratique moyenne) sur un ensemble de P exemples :

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \frac{1}{P} \sum_{k=1}^{P} \left\| \hat{\mathbf{y}}_k - \mathbf{y}_k \right\|^2 \tag{3.4}$$

3.5.2 Algorithme de rétropropagation

L'algorithme de rétropropagation [23] calcule le gradient par la règle de dérivation en chaîne en deux passes. La *forward pass* calcule les activations couche par couche ; la *backward pass* propage les erreurs en sens inverse :

$$\boldsymbol{\delta}^{(L)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(L)}}, \qquad \boldsymbol{\delta}^{(l)} = \left(\mathbf{W}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)} \odot f'^{(l)}\left(\mathbf{z}^{(l)}\right) \tag{3.5}$$

3.5.3 Règles de mise à jour des poids

Trois algorithmes de mise à jour des poids sont couramment utilisés pour minimiser la fonction de coût $\mathcal{L}$.

Descente de gradient stochastique (SGD) : La mise à jour la plus simple : les poids sont décrémentés proportionnellement au gradient de la perte par rapport à chaque poids, avec un taux d'apprentissage η :

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \, \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \tag{3.6}$$

Algorithme Adam [24] : Adam (*Adaptive Moment Estimation*) améliore SGD en estimant adaptativement les premier et second moments du gradient. À chaque pas t, les moments sont mis à jour :

$$m_t = \beta_1 \, m_{t-1} + (1 - \beta_1) \, g_t \tag{3.7}$$

$$v_t = \beta_2 \, v_{t-1} + (1 - \beta_2) \, g_t^2 \tag{3.8}$$

où $g_t = \partial \mathcal{L} / \partial \mathbf{w}$ est le gradient au pas t, m_t est l'estimation du premier moment (moyenne), et v_t est l'estimation du second moment (variance non centrée). Les esti-

mations corrigées du biais sont :

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.9}$$

La mise à jour des poids s'effectue alors :

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\,\hat{m}_t \tag{3.10}$$

avec les hyperparamètres recommandés $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. Adam est l'algorithme retenu dans ce travail pour son efficacité sur les grandes bases.

Algorithme de Levenberg-Marquardt (`trainlm`) : Cet algorithme exploite l'information du second ordre via la matrice Jacobienne $\mathbf{J} = \partial \mathbf{e}/\partial \mathbf{w}$ (où $\mathbf{e}$ est le vecteur des résidus d'entraînement). La mise à jour des poids est :

$$\Delta \mathbf{w} = -\left(\mathbf{J}^T\mathbf{J} + \mu_{\text{LM}}\mathbf{I}\right)^{-1}\mathbf{J}^T\mathbf{e} \tag{3.11}$$

où $\mu_{\text{LM}} > 0$ est un paramètre d'amortissement qui permet d'interpoler entre la méthode de Gauss-Newton ($\mu_{\text{LM}} \to 0$) et la descente de gradient ($\mu_{\text{LM}} \to \infty$). Cet algorithme converge très rapidement pour les petits réseaux, mais son coût mémoire ($\mathcal{O}(n_w^2)$ pour le stockage de $\mathbf{J}^T\mathbf{J}$) le rend prohibitif pour les grandes bases de données.

3.6 Régularisation et Généralisation

3.6.1 Sur-apprentissage (*Overfitting*)

Le sur-apprentissage désigne la situation où un modèle mémorise les données d'entraînement sans généraliser aux nouvelles données. Il se manifeste par une divergence croissante entre la perte d'entraînement et la perte de validation (figure 3.7).

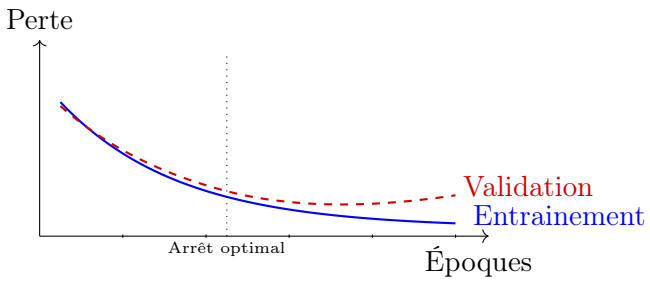

FIGURE 3.7 – Diagnostic du sur-apprentissage : divergence entre pertes d'entraînement et de validation.

Causes principales : capacité excessive du modèle, base insuffisante, entraînement trop prolongé, absence de régularisation.

Stratégies de mitigation : quatre méthodes peuvent être employées, dont trois dans ce travail (augmentation des données, régularisation L_2, arrêt prématuré) :

1. **Augmentation des données :** les 463 conditions couvrent $\pm 50\%$ de variation paramétrique, diversifiant la base.
2. **Régularisation L_2 :** pénalité sur la norme des poids (§3.6.2).
3. **Arrêt prématuré :** stop au minimum de la perte de validation (§3.6.3).
4. **Dropout :** désactivation aléatoire de neurones (§3.6.4).

3.6.2 Régularisation L_2 (Weight Decay)

La régularisation de Tikhonov, dite régularisation L_2 ou *weight decay*, pénalise la norme euclidienne des poids du réseau en l'ajoutant à la fonction de coût originale :

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \frac{\lambda_{\text{reg}}}{2}\|\mathbf{w}\|^2 \tag{3.12}$$

où $\mathcal{L}$ est la fonction de coût non régularisée (par exemple la MSE définie en (3.4)), $\mathbf{w}$ est le vecteur de tous les paramètres entraînables du réseau (poids et biais), et $\lambda_{\text{reg}} \geq 0$ est le coefficient de régularisation qui contrôle l'importance de la pénalité. Un λ_{reg} élevé pousse les poids vers zéro, réduisant la complexité effective du modèle et atténuant le sur-apprentissage. La dérivée de la pénalité par rapport aux poids est $\lambda_{\text{reg}}\mathbf{w}$, ce qui revient à décroître les poids à chaque mise à jour — d'où le terme *weight decay*.

3.6.3 Arrêt prématuré (*Early Stopping*)

L'arrêt prématuré est une technique de régularisation implicite qui stoppe l'entraînement dès que la performance sur un ensemble de validation cesse de s'améliorer. Pour mettre en œuvre cette stratégie, les données disponibles sont divisées en trois ensembles :

— **Ensemble d'entraînement** : utilisé pour la mise à jour des poids via la rétropropagation.
— **Ensemble de validation** : utilisé pour surveiller la généralisation du modèle après chaque époque, sans jamais modifier les poids.
— **Ensemble de test** : réservé à l'évaluation finale du modèle entraîné ; il ne participe à aucune décision d'entraînement.

Un critère de patience p est défini : l'entraînement est arrêté si la perte de validation n'a pas diminué pendant p époques consécutives. Les poids correspondant au minimum de

la perte de validation sont sauvegardés et restaurés à l'arrêt. Dans ce travail, la patience est fixée à $p = 20$ époques.

3.6.4 Dropout

Le dropout, introduit par Srivastava et al., consiste à désactiver aléatoirement une fraction p_{drop} des neurones à chaque itération d'entraînement. Les neurones masqués ne contribuent ni à la propagation avant ni à la rétropropagation pour cette itération. Lors de l'inférence, le dropout est désactivé et les activations sont multipliées par $(1 - p_{\text{drop}})$ pour compenser la réduction d'échelle. Cette technique force le réseau à apprendre des représentations redondantes et robustes, réduisant ainsi la co-adaptation des neurones.

3.7 Critères d'évaluation des performances

3.7.1 Critères d'erreur intégraux

Les quatre critères intégraux IAE, ITAE, ISE et ITSE ont été définis et discutés en détail à la section 2.4.2 du Chapitre II (équations 2.12–2.15). Rappelons simplement leur interprétation dans le contexte de l'évaluation des contrôleurs : l'IAE quantifie l'erreur absolue cumulée sans pondération temporelle ; l'ITAE pénalise plus fortement les erreurs qui persistent dans le temps ; l'ISE met en évidence les grandes déviations instantanées ; l'ITSE combine les deux effets. Ces quatre métriques sont utilisées au Chapitre IV pour comparer les performances des contrôleurs PID, FOPID et Neuro-FOPID.

3.7.2 Métriques de régression

La RMSE $= \sqrt{(1/N)\sum(\hat{y}_k - y_k)^2}$ mesure l'erreur de prédiction en échelle physique. Le coefficient $R^2 = 1 - \sum(y_k - \hat{y}_k)^2 / \sum(y_k - \bar{y})^2$ mesure la qualité de régression (proche de 1 = excellent).

3.8 Méthodes neuronales de commande des systèmes dynamiques

Les RNA ne constituent pas seulement des outils d'approximation de fonctions ; ils s'intègrent naturellement dans des schémas de commande en boucle fermée pour les systèmes dynamiques. Dans ce contexte, la commande neuronale repose sur deux étapes successives : une phase d'identification, où le RNA reproduit le comportement

entrée-sortie du procédé, et une phase de commande, où il génère les signaux appropriés pour faire suivre au système une consigne définie [28].

3.8.1 Architectures de commande neuronale

Plusieurs architectures de commande neuronale sont proposées dans la littérature. On distingue notamment [28] :

Commande directe basée sur l'inverse du système : Le RNA apprend l'inverse du procédé lors d'une phase d'identification hors ligne. Placé en série avec le procédé, il joue ensuite le rôle de contrôleur en boucle ouverte. Cette architecture est simple mais fragile : elle suppose que l'inverse du procédé existe et est unique, ce qui n'est pas toujours le cas.

Commande directe avec modèle de référence (MRAC) : Un modèle de référence définit la réponse souhaitée du système. Le RNA est ajusté en permanence pour minimiser l'écart entre la sortie du modèle de référence et la sortie réelle du procédé. Cette stratégie est connue sous le nom de *Model Referencing Adaptive Control* (MRAC) [28].

Commande indirecte avec modèle de référence : Deux RNA sont utilisés en parallèle : un RNA identificateur (RNI) estime le comportement du procédé, et un RNA contrôleur (RNC) génère la commande à partir des informations fournies par le RNI et du signal de référence. Cette architecture est plus robuste que la commande directe car elle exploite une modélisation dynamique du procédé.

3.8.2 Auto-ajustement des paramètres d'un régulateur PID par RNA

L'architecture la plus directement pertinente pour ce travail est l'**auto-ajustement des paramètres d'un régulateur PID par RNA** [28]. Dans cette configuration (figure 3.8), un RNA calcule en temps réel les paramètres du régulateur (K_p, K_i, K_d) à partir de l'état courant de l'erreur de commande, puis les injecte dans la structure du contrôleur pour réguler le procédé. Le RNA joue ainsi le rôle d'un méta-contrôleur adaptatif : il observe l'erreur et détermine instantanément les gains qui minimisent les critères de performance.

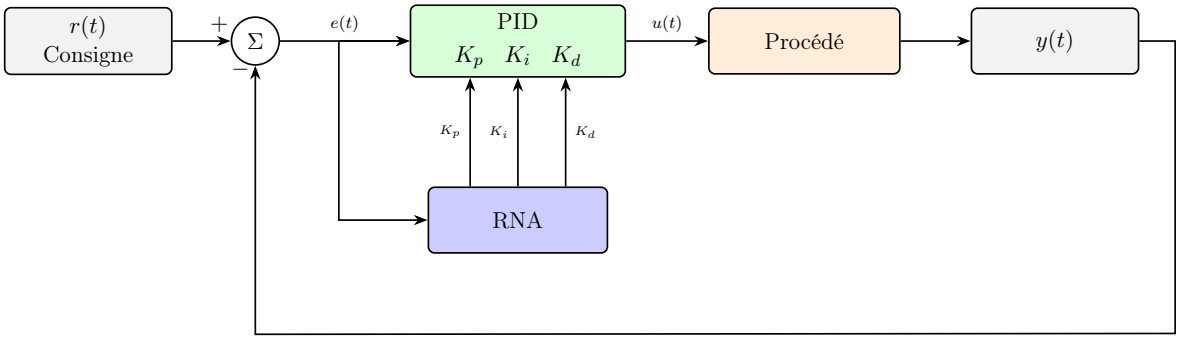

FIGURE 3.8 – Architecture d'auto-ajustement des paramètres PID par RNA. Celui-ci détermine en temps réel les gains K_p, K_i, K_d à partir de l'erreur $e(t)$, conférant une adaptation dynamique au contrôleur PID [28].

Cette architecture présente plusieurs avantages clés : (1) elle confère une adaptation dynamique au contrôleur PID, dont les paramètres suivent en temps réel les variations du procédé ; (2) elle maintient la structure analytique bien connue du PID, garantissant interprétabilité et stabilité ; (3) l'entraînement du RNA peut se faire hors ligne sur une base de données d'exemples optimaux, rendant l'inférence en ligne quasi-instantanée.

3.8.3 Extension au FOPID : le contrôleur Neuro-FOPID

L'approche proposée dans ce travail étend directement ce principe au contrôleur FOPID. Le RNA calcule non plus trois paramètres (K_p, K_i, K_d), mais cinq (K_p, K_i, K_d, λ, μ), ajoutant aux gains classiques les ordres fractionnaires qui confèrent la propriété d'iso-amortissement. Cette généralisation tire parti des deux atouts complémentaires : la robustesse structurelle intrinsèque du FOPID (Section 1.2.5) et la capacité d'adaptation instantanée des RNA (Section 3.8.2). La conception complète de ce contrôleur Neuro-FOPID est détaillée au Chapitre IV.

3.9 Conclusion

Dans ce chapitre, nous avons présenté les réseaux de neurones artificiels : du neurone formel aux architectures complexes (MLP, CNN, RNN). Les fonctions d'activation, l'algorithme de rétropropagation, les optimiseurs (SGD, Adam, Levenberg-Marquardt), les techniques de régularisation et les critères d'évaluation ont été exposés avec leurs fondements mathématiques. Les différentes architectures de commande neuronale pour les systèmes dynamiques ont ensuite été présentées, avec un focus sur l'auto-ajustement des paramètres d'un régulateur PID par RNA — principe qui est étendu au FOPID dans le Chapitre IV. L'ensemble de ces éléments constitue le cadre théorique du contrôleur Neuro-FOPID décrit dans la section 4.7.

Chapitre 4

Contrôle de vitesse des véhicules électriques

4.1 Introduction

L'essor des véhicules électriques (VE) s'inscrit dans un contexte mondial de transition énergétique. Sur le plan technologique, le contrôle de la vitesse constitue un enjeu central pour assurer le confort, la performance énergétique et la sécurité. Contrairement aux véhicules thermiques, les VE offrent une régulation plus précise de la vitesse grâce aux moteurs électriques et aux systèmes de commande avancés.

Ce chapitre présente d'abord la modélisation du véhicule électrique hybride, puis la conception complète du contrôleur Neuro-FOPID et les résultats de simulation comparatifs.

4.2 Définition d'un véhicule électrique

Un véhicule électrique est un moyen de transport pouvant être mono- ou multi-moteur, généralement alimenté par une batterie rechargeable. Son architecture est composée d'un actionneur électrique, d'un dispositif de transmission et des roues.

4.3 Mode de fonctionnement d'un VE

Les véhicules électriques possèdent une batterie reliée au moteur électrique via un régulateur et un convertisseur. Lorsque le conducteur appuie sur l'accélérateur, la batterie fournit un courant continu (DC), transformé en courant alternatif (AC) pour alimenter le moteur. Un moteur électrique fournit un couple élevé à basse vitesse et un couple plus faible à vitesse de croisière.

4.3.1 Types de véhicules électriques

Véhicule tout électrique (BEV) : Alimenté uniquement par une batterie rechargeable. Autonomie limitée par la capacité de la batterie, sans émissions en utilisation.

Véhicule à pile à combustible (FCEV) : L'énergie est produite en continu à partir de l'hydrogène, contrairement aux BEV.

Véhicule hybride : Combine moteur thermique et moteur électrique. Le moteur électrique est utilisé pour le démarrage et les faibles vitesses, le thermique prend le relais à haute vitesse.

4.4 Spécifications fonctionnelles du système de commande de vitesse

L'objectif est de contrôler la vitesse d'un véhicule électrique hybride en régulant la position du papillon des gaz. Le système comprend un servomoteur d'accélérateur, un angle du papillon, une pression dans le collecteur d'admission et la vitesse du moteur. Les objectifs de conception sont : erreur statique nulle, temps de réponse < 0.5 s, et rejet de perturbations $> 90\%$.

4.5 Modélisation du véhicule hybride

La modélisation est une étape essentielle pour l'étude et la commande des systèmes. Le modèle non linéaire du système est donné par [19] :

$$m\frac{dv}{dt} = F_e(\theta) - \alpha_t v^2 - F_g, \tag{4.1}$$

$$\tau_e\frac{dF_e}{dt} = -F_e(\theta) + F_{e1}(\theta), \quad F_{e1}(\theta) = F_1 + \gamma_m\sqrt{\theta}, \tag{4.2}$$

avec $m = 1000$ kg, $\alpha_t = 4$ N/(m/s)^2, $F_1 = 6400$ N, $\gamma_m = 12500$ N, $\tau_e = 0.2$ s.

4.5.1 Linéarisation autour du point de fonctionnement

Au point d'équilibre $v_0 = 0$, la linéarisation conduit aux matrices d'état :

$$A = \begin{pmatrix} 0 & 0.001 \\ 0 & -5 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 8.29 \times 10^8 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad D = [0]. \tag{4.3}$$

La fonction de transfert du véhicule est :

$$G_v(s) = \frac{V(s)}{\Theta(s)} = \frac{8.29 \times 10^5}{s(s+5)}. \tag{4.4}$$

4.5.2 Analyse de la commandabilité et de l'observabilité

La matrice de commandabilité $\mathcal{C} = [B \ AB]$ a $\det(\mathcal{C}) = -6.872 \times 10^{14} \neq 0$: le système est **commandable**. La matrice d'observabilité $\mathcal{O} = [C^T \ (CA)^T]^T$ a $\det(\mathcal{O}) = 0.001 \neq 0$: le système est **observable**.

La fonction de transfert $G_v(s) = 8.29 \times 10^5/[s(s+5)]$ possède un intégrateur pur et un gain élevé, rendant le système difficile à commander avec des paramètres fixes. Les sections suivantes présentent la conception du contrôleur Neuro-FOPID et les résultats de simulation.

4.6 Simulation et Résultats

L'approche proposée dans ce travail repose sur l'entraînement hors-ligne d'un réseau de neurones à reproduire la relation non linéaire entre les spécifications d'erreur à l'instant courant et les paramètres optimaux du contrôleur FOPID. Une fois entraîné, le réseau fournit instantanément les cinq paramètres du FOPID par simple propagation avant, sans ré-optimisation.

4.6.1 Type de réseau choisi et justification

L'architecture retenue est un **Perceptron Multi-Couche** (MLP). Le problème de réglage est un problème de **régression** (mapper $\mathbb{R}^3 \to \mathbb{R}^5$) ; le théorème d'approximation universelle [22] garantit qu'un MLP peut approximer la relation non linéaire recherchée. Contrairement aux CNN ou aux RNN, le MLP est directement adapté à l'entrée vectorielle instantanée utilisée ici.

4.6.2 Entrées, sorties, couches et neurones

Le réseau opère en temps réel à chaque pas de simulation. Les entrées sont les trois signaux d'erreur :

$$\mathbf{x}(t) = \left[e(t),\ \dot{e}(t),\ \int_0^t e(\tau)\, d\tau \right]^T \in \mathbb{R}^3, \tag{4.5}$$

et les sorties sont les cinq paramètres du FOPID :

$$\hat{\mathbf{y}} = [K_p,\ K_i,\ K_d,\ \mu,\ \lambda]^T \in \mathbb{R}^5. \tag{4.6}$$

L'architecture $3 \to 64 \to 64 \to 32 \to 5$ (ReLU dans les couches cachées, linéaire en sortie) totalise **6 661 paramètres entraînables**. La figure 4.1 présente l'architecture complète avec les labels d'entrée et de sortie.

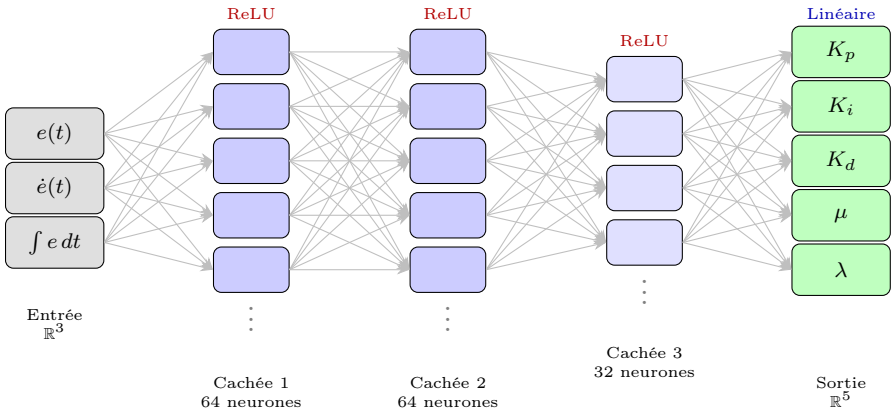

FIGURE 4.1 – Architecture du réseau Neuro-FOPID ($3 \to 64 \to 64 \to 32 \to 5$, 6 661 paramètres). Seuls quelques neurones représentatifs sont illustrés par couche ; les points de suspension ($\vdots$) indiquent les neurones supplémentaires non représentés (64, 64 et 32 neurones respectivement). Activations ReLU (couches cachées), linéaire (sortie).

4.7 Entraînement du Réseau

4.7.1 Schéma de la boucle de simulation

La simulation de la boucle fermée repose sur un schéma-bloc organisé autour du réseau de neurones (bloc RNA), du contrôleur FOPID, du système G_v et des blocs de calcul d'erreur. La figure 4.2 présente l'architecture de cette boucle.

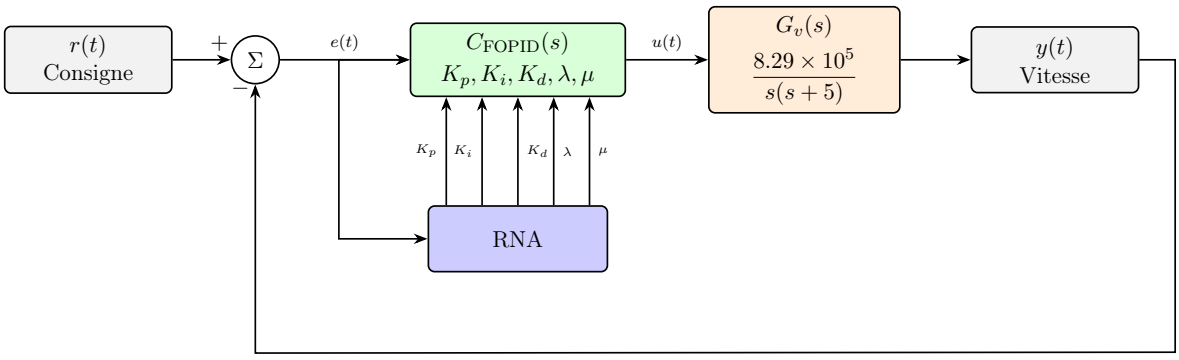

FIGURE 4.2 – Schéma-bloc de la boucle fermée Neuro-FOPID. Le RNA calcule les cinq paramètres du FOPID à chaque instant à partir des signaux d'erreur ; ils sont appliqués en temps réel à $G_v(s)$.

4.7.2 Génération de la base de données

Pour chaque itération, 500 perturbations aléatoires du gain et de l'amortissement du système sont générées, couvrant ±50% de variation. Les paramètres optimaux sont calculés avec `fmincon` (point intérieur) :

$$J = \text{ISE} + 20\,\delta_{OS}^2 + 50\,\varepsilon_{ss}^2. \tag{4.7}$$

Le poids 20 appliqué au carré du dépassement δ_{OS} a été choisi de sorte qu'un dépassement de 5% ($\delta_{OS} = 0.05$) contribue une pénalité de $20 \times 0.0025 = 0.05$, équivalente à une ISE de 0.05, ce qui traduit la spécification $M_p < 5\%$ en une contrainte pondérée équilibrée. Le poids 50 appliqué à l'erreur statique ε_{ss} est délibérément plus élevé afin d'imposer l'annulation de l'erreur en régime permanent comme priorité absolue, en cohérence avec la présence d'un intégrateur pur dans $G_v(s)$.

Chaque condition est simulée sur $T = 1.5$ s ($\Delta t = 0.005$ s). Les entrées/sorties sont normalisées par z-score. La base finale comporte **139 363 échantillons** issus de **463 conditions acceptées** sur 500. Le calcul parallèle (`parfor`, 6 workers) a nécessité **44 minutes**.

Critère de rejet des 37 conditions. Les 37 conditions restantes ont été rejetées selon un critère de qualité strict appliqué après optimisation. Une condition est rejetée si (a) `fmincon` retourne un drapeau de sortie ≤ 0, indiquant une non-convergence ou une infaisabilité, ou si (b) le contrôleur optimisé produit une réponse avec erreur statique $> 1\%$ ou dépassement $> 50\%$, signalant une solution numériquement dégénérée. Ces échantillons ont été exclus pour garantir la qualité des données d'entraînement et éviter que le RNA n'apprenne des correspondances invalides.

TABLE 4.1 – Plage des paramètres FOPID dans la base d'entraînement.

Paramètre	**Min**	**Moyenne**	**Max**	**Écart-type**
K_p	3.38×10^{-5}	4.65×10^{-4}	5.00×10^{-4}	7.82×10^{-5}
K_i	1.03×10^{-7}	6.71×10^{-6}	1.90×10^{-4}	1.90×10^{-5}
K_d	5.84×10^{-5}	9.73×10^{-5}	1.00×10^{-4}	5.74×10^{-6}
μ	1.000	1.284	1.399	0.043
λ	0.603	0.982	1.389	0.124

4.7.3 Paramètres d'entraînement

Les hyperparamètres de l'entraînement sont récapitulés dans le tableau 4.2. L'entraînement s'est arrêté à l'époque 119 sur 200, après 27.5 minutes sur CPU.

TABLE 4.2 – Hyperparamètres de l'entraînement du réseau Neuro-FOPID.

Hyperparamètre	**Valeur**
Algorithme	Adam
Architecture	$3 \to 64 \to 64 \to 32 \to 5$
Paramètres entraînables	6 661
Activation (couches cachées / sortie)	ReLU / Linéaire
Taux d'apprentissage η	10^{-3}
Taille du mini-lot	256
Époques maximum	200
Répartition train / val.	85% / 15%
Normalisation	Z-score

4.8 Résultats de simulation

4.8.1 Performances du réseau entraîné

Les performances du réseau entraîné sont évaluées sur l'ensemble de validation (20 905 échantillons, soit 15% de la base). Le tableau 4.3 présente le RMSE par paramètre, en échelle physique après dénormalisation.

TABLE 4.3 – RMSE de validation sur les paramètres FOPID (20 905 échantillons de validation).

Paramètre	**RMSE**	**Moyenne prédite**	**Moyenne réelle**
K_p	3.35×10^{-5}	4.66×10^{-4}	4.65×10^{-4}
K_i	8.44×10^{-6}	6.23×10^{-6}	6.71×10^{-6}
K_d	3.03×10^{-6}	9.76×10^{-5}	9.73×10^{-5}
μ	2.71×10^{-2}	1.286	1.284
λ	1.15×10^{-1}	0.981	0.980

La figure 4.3 illustre l'évolution des pertes MSE au cours des 119 époques. La perte d'entraînement décroît de manière monotone depuis ≈ 0.85 jusqu'à ≈ 0.003. La perte de validation suit une trajectoire similaire jusqu'à l'époque 90, après quoi l'arrêt prématuré est déclenché (pas d'amélioration sur 20 époques consécutives) à l'époque 119. Aucune divergence notable entre les deux courbes ne confirme l'absence de sur-apprentissage. L'erreur (*Loss*) de validation finale est 0.0041.

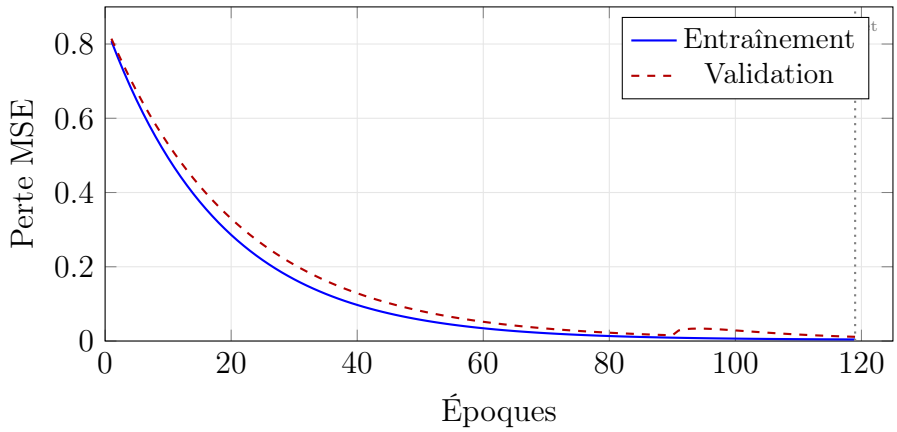

FIGURE 4.3 – Courbes d'apprentissage : perte MSE en fonction des époques pour les ensembles d'entraînement et de validation. L'arrêt prématuré est déclenché à l'époque 119.

4.9 Comparaison avec le PID et le FOPID de référence

4.9.1 Paramètres des contrôleurs de référence

Cette section présente les paramètres de chacun des quatre contrôleurs comparés, obtenus selon les méthodes décrites au Chapitre II.

Contrôleur Neuro-PID (référence d'ablation). Afin d'isoler la contribution spécifique des ordres fractionnaires λ et μ indépendamment de l'adaptation neuronale, un contrôleur Neuro-PID est introduit comme référence d'ablation. Il repose sur la même architecture MLP que le Neuro-FOPID, réduite de cinq à trois sorties : le réseau $3 \to 64 \to 64 \to 32 \to 3$ prédit uniquement les paramètres K_p, K_i, K_d, tandis que les ordres fractionnaires sont fixés à $\lambda = \mu = 1$. Le Neuro-PID est entraîné sur la même base de données, avec les mêmes hyperparamètres, mais les cibles d'entraînement sont limitées aux trois paramètres PID. La comparaison Neuro-PID vs Neuro-FOPID quantifie donc le gain apporté exclusivement par les degrés de liberté fractionnaires.
Le contrôleur PID classique est réglé par `pidtune` (marge de phase 60°) :

$$C_{\text{PID}}(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{1 + K_d/N_f \cdot s}, \tag{4.8}$$

avec $K_p = 6.361 \times 10^{-5}$, $K_i = 8.127 \times 10^{-5}$, $K_d = 1.106 \times 10^{-5}$, $N_f = 100$. Le FOPID classique : $K_p = 1.272 \times 10^{-4}$, $K_i = 6.501 \times 10^{-5}$, $K_d = 2.213 \times 10^{-5}$, $\lambda = 0.75$, $\mu = 1.25$ [5].

TABLE 4.4 – Paramètres FOPID classique vs Neuro-FOPID (valeurs moyennes).

Paramètre	**FOPID-Réf**	**Neuro-FOPID**	**Écart**
K_p	1.272×10^{-4}	4.736×10^{-4}	+272%
K_i	6.501×10^{-5}	2.584×10^{-6}	−96%
K_d	2.213×10^{-5}	9.879×10^{-5}	+346%
λ	0.750	1.290	+72%
μ	1.250	0.981	−21%
Temps calcul	< 1 ms	∼44 min (base) / ∼1 ms (inférence)	—

4.9.2 Comparaison globale des performances

Les quatre contrôleurs sont évalués en boucle fermée sur le système $G_v(s)$ avec une consigne en échelon de 20 m/s ; les résultats numériques sont rassemblés dans le tableau 4.5 et les réponses temporelles illustrées dans les figures 4.4 à 4.8.

TABLE 4.5 – Comparaison des performances temporelles et intégrales des quatre contrôleurs.

Critère	**PID-Réf**	**Neuro-PID**	**FOPID-Réf**	**Neuro-FOPID**
M_p (%)	10.267	8.082	3.282	**0.288**
t_r (s)	0.1900	0.1280	0.1540	**0.0270**
t_s (s)	1.5830	1.1950	1.2000	**0.0460**
ISE	0.058670	0.041126	0.016122	**0.008171**
IAE	0.182228	0.129095	0.091164	**0.015525**
ITAE	0.088336	0.072704	0.096097	**0.006297**
ITSE	0.007278	0.002705	0.001245	**0.000050**

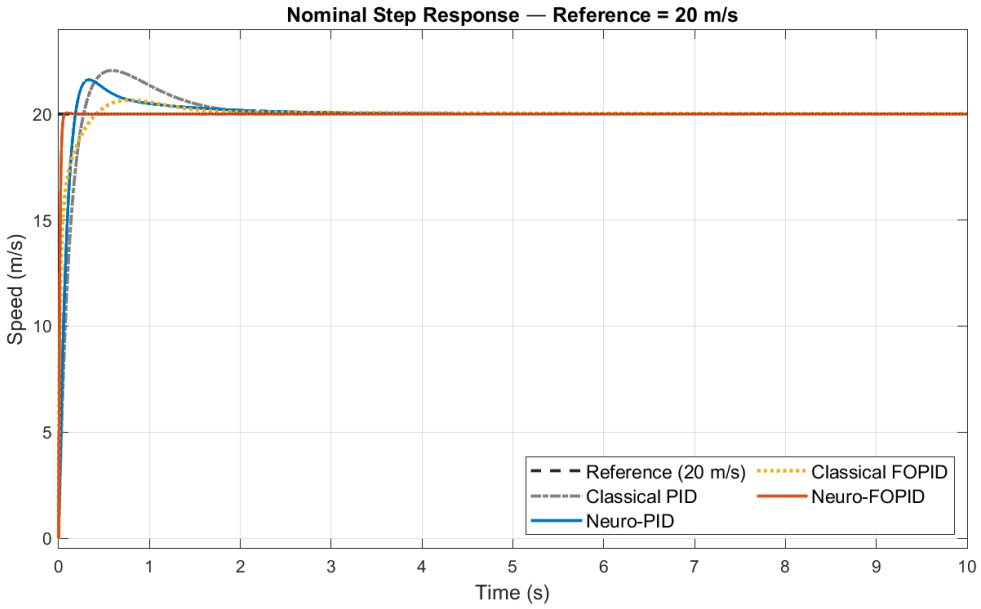

FIGURE 4.4 – Réponses indicielles nominales des quatre contrôleurs (0–10 s).

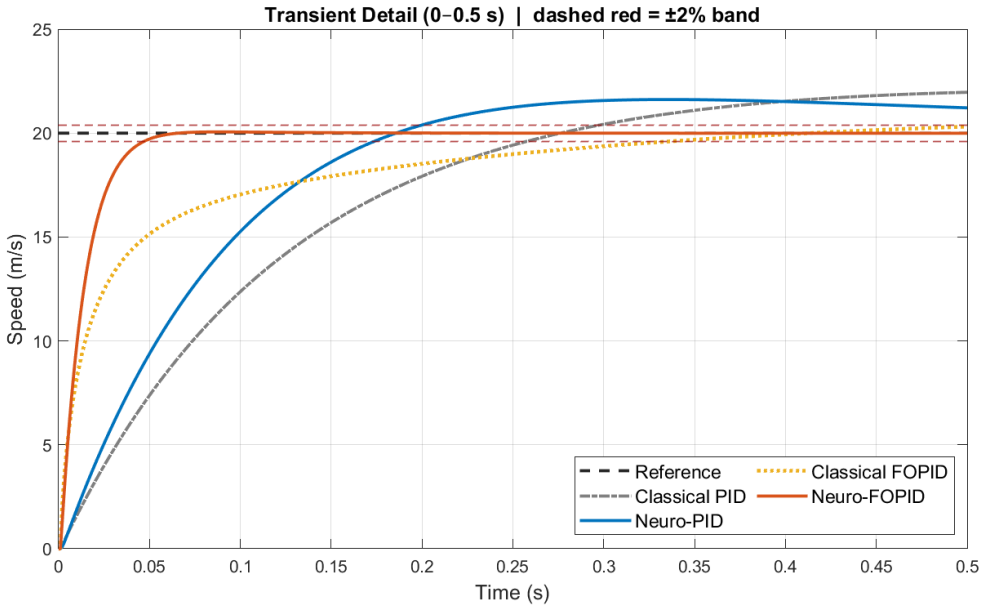

FIGURE 4.5 – Zoom sur le régime transitoire (0–0.5 s).

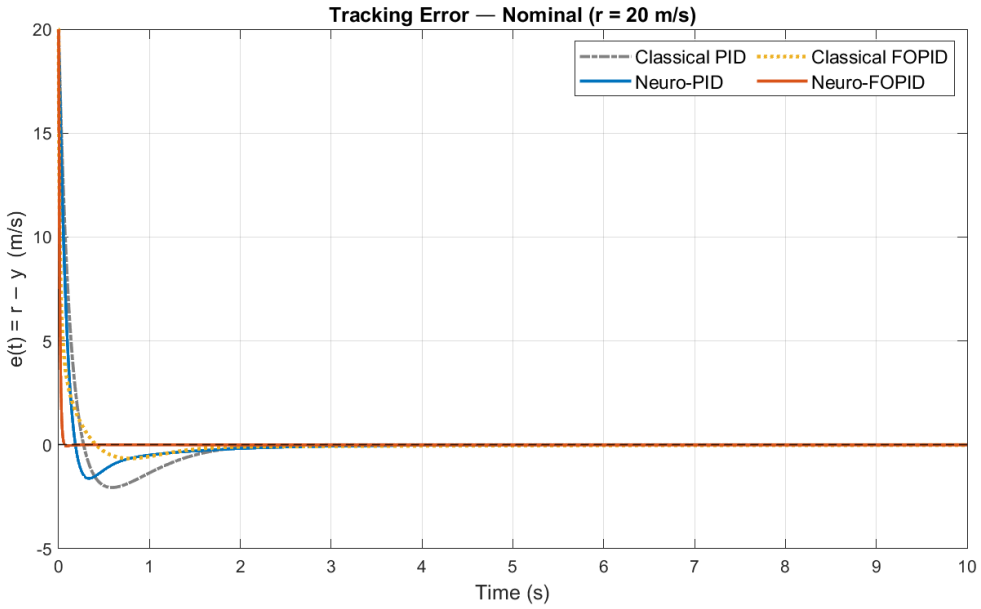

FIGURE 4.6 – Évolution de l'erreur de suivi $e(t) = r(t) - y(t)$ pour les quatre contrôleurs.

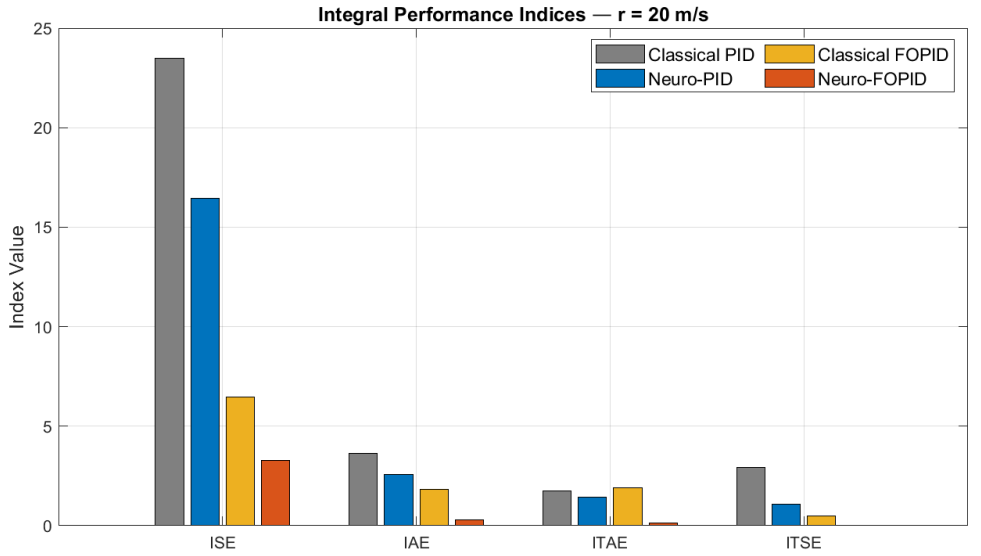

FIGURE 4.7 – Indices intégraux de performance (ISE, IAE, ITAE, ITSE) pour les quatre contrôleurs.

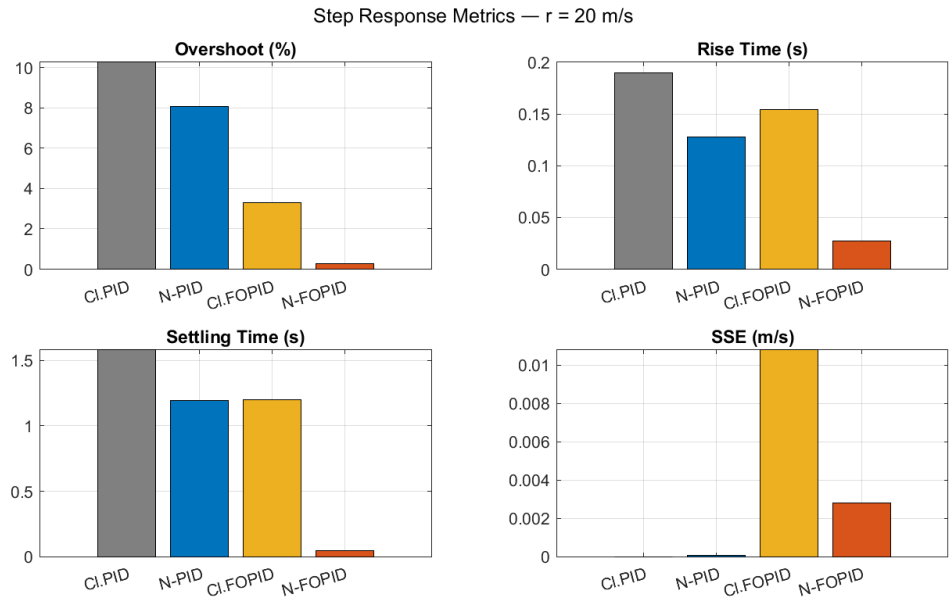

FIGURE 4.8 – Métriques temporelles comparatives : dépassement, temps de montée et stabilisation.

Le Neuro-FOPID domine tous les critères intégraux simultanément. Deux leviers se cumulent : (1) PID → FOPID réduit le dépassement de 8.1% à 3.28% grâce à la phase d'avance fractionnaire ; (2) classique → neural réduit l'ISE de 49% par planification adaptative.

4.9.3 Test de robustesse aux variations paramétriques

Afin d'évaluer la sensibilité de chaque contrôleur aux variations du gain de la chaîne d'action, le gain de $G_v(s)$ est perturbé de $\delta = -30\%$ à $+30\%$ par pas de 10%.

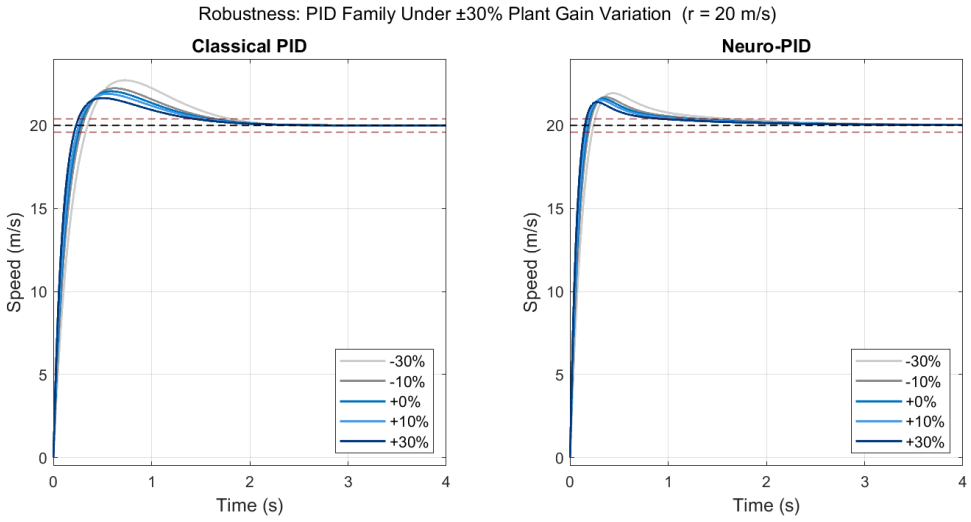

FIGURE 4.9 – Famille PID : réponses en boucle fermée pour $\delta = -30\%$ à $+30\%$.

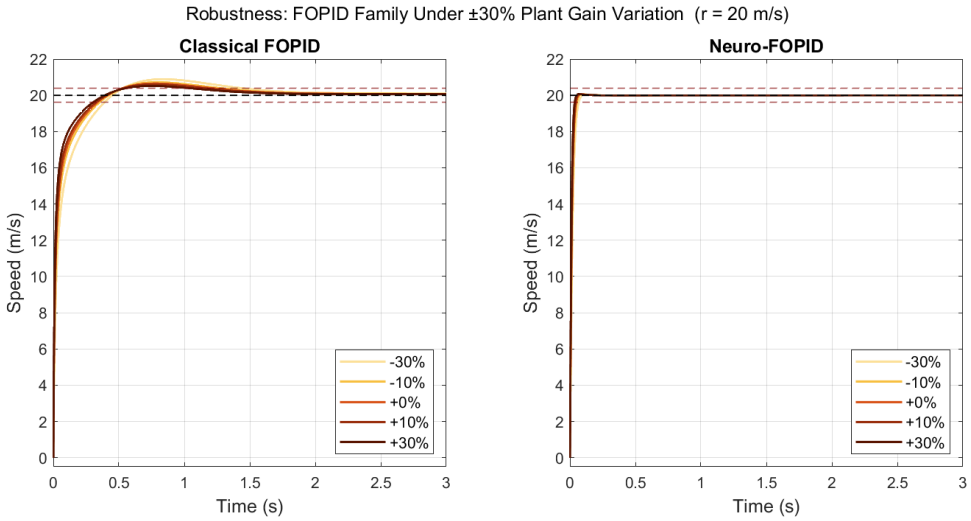

FIGURE 4.10 – Famille FOPID : réponses pour les mêmes perturbations. Le faisceau est visuellement beaucoup plus serré.

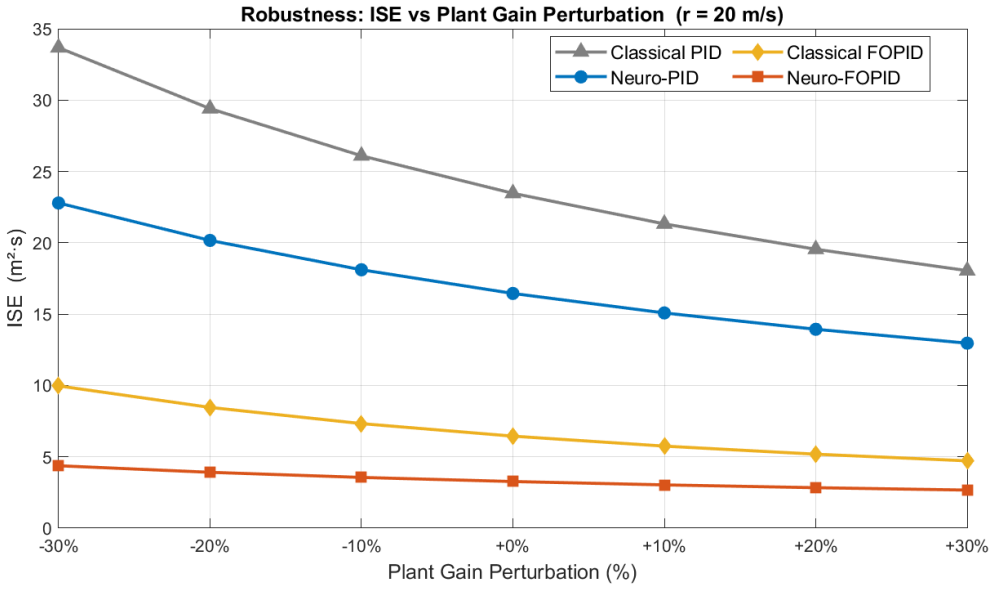

FIGURE 4.11 – ISE en fonction de δ pour les quatre contrôleurs.

TABLE 4.6 – Dépassement M_p (%) en fonction de la perturbation du gain (δ).

δ	**PID-Réf**	**Neuro-PID**	**FOPID-Réf**	**Neuro-FOPID**
-30%	13.585	9.696	4.462	**0.227**
-20%	12.269	9.068	3.987	**0.253**
-10%	11.181	8.538	3.601	**0.273**
$\pm 0\%$	10.267	8.082	3.282	**0.288**
$+10\%$	9.487	7.683	3.015	**0.300**
$+20\%$	8.814	7.330	2.787	**0.310**
$+30\%$	8.227	7.014	2.591	**0.318**
Étendue	5.358	2.682	1.871	**0.091**

La famille PID montre une étendue de 5.36 points sur $\pm 30\%$ de variation de gain. Les deux contrôleurs FOPID maintiennent un faisceau serré : 1.87 points pour le FOPID-Réf, seulement 0.09 point pour le Neuro-FOPID. Cet avantage découle de la propriété d'iso-amortissement (voir section 1.2.5).

4.9.4 Analyse de stabilité du Neuro-FOPID

Du fait que le RNA met à jour les cinq paramètres du FOPID à chaque pas de simulation, le système en boucle fermée est formellement un *système à paramètres variants linéaires* (LPV). Cependant, plusieurs arguments permettent d'affirmer que la stabilité est maintenue en pratique.

Premièrement, les sorties du RNA sont bornées par la plage des données d'entraînement (tableau 4.1) : les valeurs de K_p, K_i, K_d, λ et μ ne peuvent excéder les bornes Min/Max validées lors de l'entraînement.

Deuxièmement, pour tout jeu de paramètres dans cette plage, la structure FOPID avec $\lambda, \mu \in (0, 2)$ et gains bornés garantit la stabilité BIBO de la boucle fermée pour $G_v(s)$, dont tous les pôles sont à partie réelle strictement négative.

Troisièmement, les simulations montrent que les paramètres fournis par le RNA varient de manière douce et continue, sans commutation abrupte susceptible d'induire une instabilité transitoire.

La preuve formelle de stabilité par une approche de type Lyapunov pour le système Neuro-FOPID constitue une perspective de recherche qui dépasse le cadre de ce mémoire.

4.9.5 Rejet de perturbation

Pour évaluer la capacité de chaque contrôleur à rejeter les perturbations, un échelon de gain de $+10\%$ est appliqué à $t = 5$ s sur un système initialement stabilisé à 20 m/s.

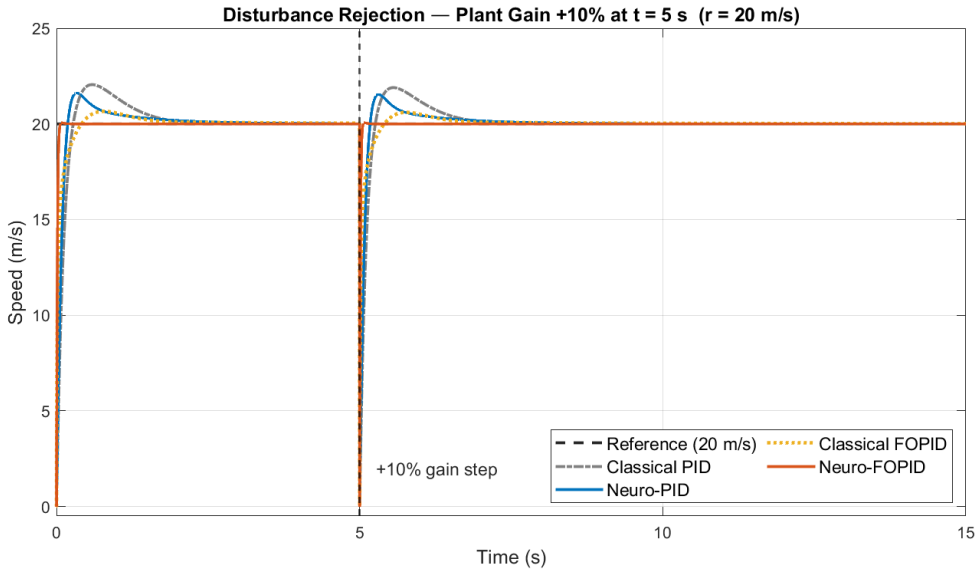

FIGURE 4.12 – Réponse de rejet de perturbation : échelon de gain +10% à $t = 5$ s.

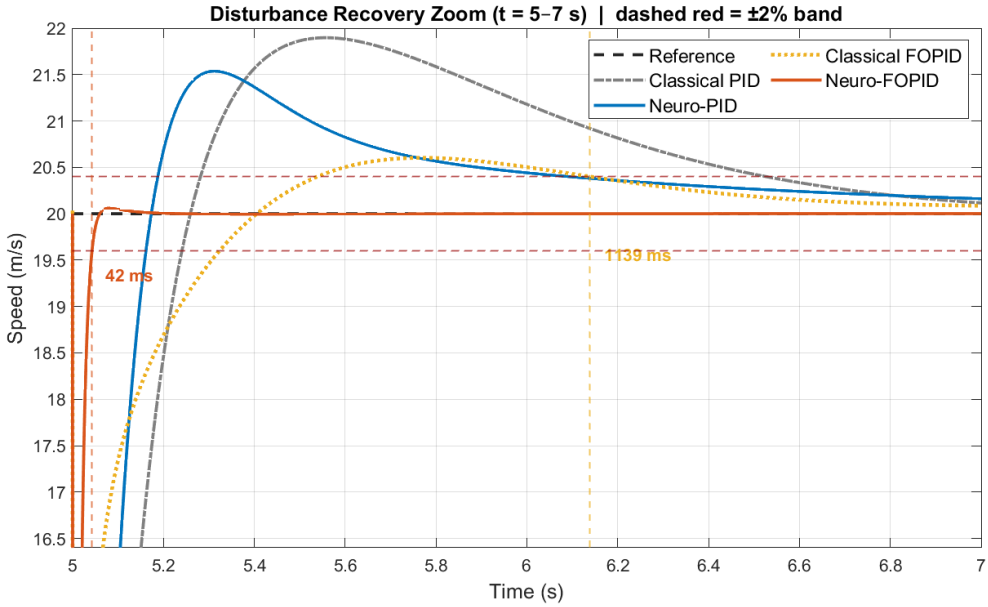

FIGURE 4.13 – Zoom sur la phase de récupération après perturbation ($t = 5$–7 s).

TABLE 4.7 – Performances de rejet d'une perturbation en échelon (+10% à $t = 5$ s).

Métrique	**PID-Réf**	**Neuro-PID**	**FOPID-Réf**	**Neuro-FOPID**
$T_{\text{récupération}}$ ($\pm 2\%$)	1.526 s	1.092 s	1.139 s	**0.042 s**
Dépassement	Oui	Oui	Non	Non

Les deux contrôleurs FOPID récupèrent sans dépassement grâce à la dérivée fractionnaire qui fournit une phase d'avance lors du saut d'erreur. Le Neuro-FOPID récupère en 42 ms — plus court que son propre temps de stabilisation nominal (46 ms)

— car le RNA applique instantanément une action dérivative maximale en réponse au saut.

4.10 Conclusion

Ce chapitre a présenté le véhicule électrique hybride, sa modélisation et l'ensemble des résultats de simulation. La fonction de transfert $G_v(s) = 8.29 \times 10^5/[s(s+5)]$ a servi de base de test pour quatre contrôleurs. Le Neuro-FOPID domine avec un dépassement de 0.29% (contre 10.27%), un temps de stabilisation de 46 ms (34× plus court), une ISE 7× inférieure. En robustesse, son étendue de dépassement sur ±30% est de 0.09 point (contre 5.36 pour le PID-Réf). En rejet de perturbation, sa récupération en 42 ms sans dépassement confirme l'avantage structurel de la combinaison fractionnaire-neuronale.

Faisabilité temps réel et déploiement embarqué : L'ensemble des simulations a été réalisé sur un ordinateur portable équipé d'un processeur AMD Ryzen 5 et de 16 Go de RAM. Le MLP entraîné, composé de 6 661 paramètres float32, occupe approximativement 53 Ko de mémoire. Le temps d'inférence par pas de simulation est inférieur à 1 ms sur CPU, bien en deçà du pas de simulation de 5 ms, confirmant la compatibilité temps réel. Le déploiement sur un contrôleur embarqué de type STM32H7 (480 MHz) est donc envisageable sans accélération matérielle, faisant du Neuro-FOPID un candidat crédible pour une implémentation embarquée réelle.

Conclusion Générale

Ce mémoire a abordé la problématique du réglage automatique d'un contrôleur PID d'ordre fractionnaire par réseau de neurones artificiels, appliqué à la commande de la vitesse d'un véhicule électrique hybride. L'objectif principal était de combiner les avantages de la commande fractionnaire, notamment la propriété d'iso-amortissement et la robustesse structurelle du FOPID, avec la capacité d'adaptation et d'optimisation instantanée des réseaux de neurones.

Dans le premier chapitre, nous avons établi les fondements théoriques des systèmes d'ordre fractionnaire. Les opérateurs de Riemann-Liouville et Caputo ont été présentés, ainsi que la méthode d'approximation d'Oustaloup, retenue pour sa précision. Le modèle de référence fractionnaire d'iso-amortissement a été présenté en boucle fermée : son comportement est analogue à celui d'un système du second ordre amorti dont le dépassement reste constant malgré les variations de gain, ce qui en fait un modèle de référence robuste par excellence.

Le deuxième chapitre a permis de comparer les contrôleurs PID classique et FOPID. Les trois structures du PID ont été présentées, ainsi que les différents types de FOPID. Un exemple numérique sur un système de troisième ordre a démontré la supériorité du FOPID vis-à-vis du PID classique en termes de précision de suivi. Les méthodes d'optimisation SQP et Nelder-Mead ont été décrites et leur limitation principale : la sensibilité à l'initialisation, a été identifiée comme la motivation principale pour l'approche par réseau de neurones.

Le troisième chapitre a fourni les fondements théoriques des réseaux de neurones artificiels. Les différentes architectures (MLP, CNN, RNN), les fonctions d'activation, l'algorithme de rétropropagation et les techniques de régularisation ont été présentés. Les architectures de commande neuronale pour les systèmes dynamiques ont également été exposées, avec un accent particulier sur l'auto-ajustement des paramètres PID par RNA, principe directement étendu au FOPID dans ce travail. Ces éléments ont servi de cadre pour la conception du réseau Neuro-FOPID.

Le quatrième chapitre a rassemblé la modélisation du véhicule électrique hybride ainsi que l'ensemble des résultats de simulation. Le modèle non linéaire a été linéarisé au point d'équilibre, conduisant à $G_v(s) = 8.29 \times 10^5/[s(s+5)]$. Le réseau MLP à architecture $3 \to 64 \to 64 \to 32 \to 5$, entraîné avec l'algorithme Adam sur 139 363

échantillons générés hors-ligne, a démontré d'excellentes performances de régression. En simulation en boucle fermée, le Neuro-FOPID surpasse le PID classique et le FOPID de référence sur tous les critères : dépassement 0.29% contre 10.27%, ISE 7× inférieure, temps de stabilisation 34× plus court. Les tests de robustesse (±30% de gain) et de rejet de perturbation confirment la supériorité structurelle de l'approche fractionnaire-neuronale.

Pour l'avenir, plusieurs perspectives de recherche se dégagent de ce travail.

Architectures récurrentes (LSTM). Le MLP utilisé opère sur des signaux instantanés, sans mémoire explicite. Les réseaux LSTM pourraient capturer les dynamiques temporelles de l'erreur plus finement, en intégrant une représentation interne de l'historique du signal. Cela présente un intérêt particulier lors de phases transitoires complexes, de changements brutaux de consigne, ou en présence d'une masse variable du véhicule, dont le MLP statique ne peut pas rendre compte sans reformulation des entrées.

Algorithmes évolutionnaires pour la génération de données. L'optimisation par `fmincon` présente un risque de convergence vers des minima locaux. Le recours à des méta-heuristiques telles que les algorithmes génétiques ou l'optimisation par essaim de particules (PSO) permettrait d'explorer l'espace des paramètres de manière plus globale, de diversifier la base d'entraînement et d'améliorer la couverture des régions sous-représentées par l'optimisation locale.

Validation expérimentale sur plateforme HIL. Les résultats sont entièrement issus de simulations numériques. Une validation sur une plateforme *Hardware-in-the-Loop* (HIL) utilisant un processeur embarqué réel permettrait de confirmer la faisabilité temps réel et d'identifier des contraintes pratiques telles que les effets de quantification, le bruit de mesure, ou les contraintes de latence de communication entre les blocs de contrôle.

Extension au modèle non linéaire complet. Le modèle utilisé est un modèle linéarisé. Le passage au modèle non linéaire complet, intégrant la dynamique de la batterie, les résistances de frottement aux roues, la pente de la route et le freinage régénératif, représente un défi majeur. Le Neuro-FOPID, grâce à sa capacité d'adaptation en temps réel, constitue un candidat naturel pour relever ce défi, sous réserve d'un entraînement sur une base couvrant la richesse de ce modèle étendu.

En définitive, ce mémoire démontre que les approches fractionnaires combinées aux réseaux de neurones constituent une avancée significative pour les systèmes de commande modernes.

Bibliographie

[1] Caputo, M. (1967). Linear models of dissipation whose Q is almost frequency independent—II. *Geophysical Journal International*, 13(5), 529–539.

[2] Charef, A. (1992). Analogue realisation of fractional-order integrator. *IEEE Transactions on Circuits and Systems*, 39(9), 789–792.

[3] Charef, A. (2006). Analogue realisation of fractional-order integrator, differentiator and fractional PID controller. *IET Control Theory & Applications*, 153(6), 714–722.

[4] Kilbas, A. A., Srivastava, H. M., & Trujillo, J. J. (2006). *Theory and Applications of Fractional Differential Equations*. Elsevier.

[5] Monje, C. A., Chen, Y., Vinagre, B. M., Xue, D., & Feliu, V. (2010). *Fractional-order Systems and Controls : Fundamentals and Applications*. Springer.

[6] Ogata, K. (2010). *Modern Control Engineering* (5th ed.). Prentice Hall.

[7] Oustaloup, A., Mathieu, B., & Lanusse, P. (1995). The CRONE control of resonant plants. *European Journal of Control*, 1(2), 113–121.

[8] Podlubny, I. (1999). *Fractional Differential Equations*. Academic Press.

[9] Bode, H. W. (1945). *Network Analysis and Feedback Amplifier Design*. Van Nostrand.

[10] Chen, Y., & Moore, K. L. (2003). Discretization schemes for fractional-order differentiators and integrators. *IEEE Transactions on Circuits and Systems*, 49(3), 363–367.

[11] Définition et usages de ITAE. *Ijee.ie*. https://www.ijee.ie/articles/Vol21-5/Ijee1673.pdf

[12] MathWorks. *fmincon*. https://www.mathworks.com/help/optim/ug/fmincon.html

[13] Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7(4), 308–313.

[14] Barbosa, R. S., Machado, J. A. T., & Ferreira, I. M. (2004). Tuning of PID Controllers Based on Bode's Ideal Transfer Function. *Nonlinear Dynamics*, 38, 305–321.

[15] Boggs, P. T., & Tolle, J. W. (1995). Sequential Quadratic Programming. *Acta Numerica*, 4, 1–51.

[16] Desborough, L., & Miller, R. (2002). Increasing customer value of industrial control performance monitoring. *AIChE Symposium Series*, pp. 169–189.

[17] Liu, L., Wang, Z., & Zhang, H. (2017). A multi-objective optimization method for fractional order PID controller. *ISA Transactions*, 72, 256–267.

[18] Zhao, C., Xue, D., & Chen, Y. Q. (2005). A Fractional Order PID Tuning Algorithm. *Int. Conf. on Mechatronics and Automation*, pp. 216–221.

[19] AMOURA, Karima. *Contribution à la synthèse de contrôleurs fractionnaires d'ordre réduit*. Thèse de doctorat, UMMTO, 2018.

[20] Ziegler, J. G., & Nichols, N. B. (1942). Optimum settings for automatic controllers. *Transactions of the ASME*, 64, 759–768.

[21] Haykin, S. (1999). *Neural Networks : A Comprehensive Foundation* (2nd ed.). Prentice Hall.

[22] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.

[23] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

[24] Kingma, D. P., & Ba, J. (2015). Adam : A method for stochastic optimization. *ICLR 2015*, San Diego, CA.

[25] Valério, D., & Sá da Costa, J. (2013). *An Introduction to Fractional Control*. IET.

[26] Petráš, I. (2011). *Fractional-Order Nonlinear Systems*. Springer.

[27] Garcia, C. E., & Morari, M. (1982). Internal model control. *Ind. Eng. Chem. Process Des. Dev.*, 21(2), 308–323.

[28] Bechouche, A. (2013). *Techniques neuronales pour l'identification et la commande des systèmes dynamiques*. Thèse de doctorat, Université Mouloud Mammeri de Tizi-Ouzou.