UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DU DOCTORAT EN GÉNIE ÉLECTRIQUE

PAR
SÉBASTIEN LESUEUR

ALGORITHMES À BASE DE RÉSEAUX DE NEURONES POUR LA COMMANDE DE SYSTÈMES DYNAMIQUES NON LINÉAIRES ET LEURS MODÈLES D'INTÉGRATION SUR SILICIUM

DÉCEMBRE 2004

Remerciements

Je tiens tout d'abord à exprimer toute ma reconnaissance aux membres du jury qui m'ont fait l'honneur d'évaluer cette thèse.

Je remercie également mon directeur de recherche, monsieur Daniel Massicotte, professeur à l'université du Québec à Trois-Rivières et directeur du laboratoire de signaux et systèmes intégrés.

Je remercie particulièrement mon codirecteur de recherche, monsieur Pierre Sicard, professeur à l'université du Québec à Trois-Rivières. Je lui exprime ma profonde gratitude pour la qualité de son enseignement et de ses interventions. Je le remercie aussi pour ses révisions méticuleuses des différentes versions de la thèse.

Je tiens à remercier chaleureusement ceux qui ont participé à ce projet de recherche, notamment dans le développement des circuits intégrés : Ahmed Ouamar, Mohamed Zebdi, Raymond Courteau et particulièrement Louis Lemire.

Enfin, au terme de mes études, j'adresse ma plus profonde reconnaissance à mes parents, qui m'ont toujours soutenu et encouragé.

Résumé

L'amélioration de la qualité de la commande de systèmes pour les applications modernes nécessite la prise en compte de tous les phénomènes physiques auxquels ces systèmes sont soumis. Ceci constitue un problème difficile pour les concepteurs qui se trouvent confrontés à deux problématiques contradictoires : (i) les algorithmes permettant de satisfaire les critères de commande des applications visées sont en général plus complexes et par conséquent plus difficiles à intégrer efficacement dans les processeurs ; (ii) l'Intégration à Très Grande Échelle (ITGE) permettant de satisfaire les critères d'implémentation privilégie les algorithmes plus simples à intégrer.

L'objectif de cette thèse est d'apporter une contribution scientifique au domaine de la commande de systèmes dynamiques non linéaires. Cette contribution vise deux domaines complémentaires en commande : (i) le développement d'algorithmes à base de RNA pour la commande de systèmes dynamiques non linéaires ; (ii) la modélisation au niveau de l'ITGE d'algorithmes de commande.

Les contributions scientifiques apportées par la présente thèse sont :

- de nouveaux algorithmes adaptatifs pour la rétro propagation à travers le modèle neuronal dans une structure de commande indirecte inverse. Ces algorithmes s'avèreront être, par rapport aux algorithmes de référence de la littérature, de meilleurs compromis entre les différents critères algorithmiques et d'ITGE,
- une architecture hautement parallèle pour l'ITGE de la loi de commande développée tout au long de la thèse, architecture intégrant le ou les nouveaux algorithmes développés,

- la modélisation de circuit CMOS pour l'ITGE de l'architecture proposée. La contribution à ce niveau sera surtout le circuit global obtenu, démontrant d'une part la faisabilité d'un ASIC analogique pour l'ITGE de la loi de commande proposée, et d'autre part la possibilité d'obtenir un circuit qui peut satisfaire à des applications en commande de systèmes dynamiques non linéaires.

Les résultats apportés par la thèse démontrent la pertinence de travaux de recherche reliant les domaines algorithmiques et d'ITGE pour des applications en commande. Ces résultats démontrent également la possibilité de développer conjointement les algorithmes de commande et des ASIC pour leur implémentation afin d'obtenir une solution pratique et efficace.

Table des matières

Liste des figures

Liste des tableaux

Liste des sigles et abréviations

ADN	Acide Désoxyribose Nucléique
AOP	Amplificateur Opérationnel
AOT	Amplificateur Opérationnel à Transconductance
ASIC	*Application Specific Integrated Circuit* (Circuit intégré à application spécifique)
BP	*Back Propagation* (Rétropropagation)
CAO	Conception Assistée par Ordinateur
CCT	Convertisseur Courant-Tension
CMFB	*Common Mode FeedBack* (Rétroaction en mode commun)
CMOS	*Complementary Metal-Oxyde Semiconductor* (Semiconducteur à metal-oxyde complémentaires)
CM-PWM	*Current-Mode Pulse Width Modulation* (Modulation de largeurs d'impulsions)
CMRR	*Common Mode Rejection Ratio* (Taux de réjection de mode commun)
CTC	Convertisseur Tension-Courant
DBP	*Dynamic Back Propagation* (Rétropropagation dynamique)
DRC	*Design Rule Check* (Règles de vérification de conception)
DSP	*Digital Signal Processing* (Traitement numérique de signal)
FPAA	*Field Programmable Analog Array* (Grille de circuits analogiques programmables)
FPGA	*Field Programmable Gate Array* (Grille de portes logiques programmables)
GaAs	Arsenure de Gallium
HP	Hewlett-Packard
ITGE	Intégration à Très Grande Échelle
LVS	*Layout Versus Schematic* (Schématique versus *layout*)

MIMO *Multiple Input Multiple Output* (Multiples sorties et multiples entrées)

NARMA *Nonlinear AutoRegressive Moving Average* (non linéaire autorégressif à moyenne mobile)

NARMAX Nonlinear AutoRegressive Moving Average with eXogenous inputs (non linéaire autorégressif à moyenne mobile et entrées auxiliaires)

NMOS *N-type Metal-Oxyde Semiconductor* (semi-conducteur métal-oxyde de type N)

OTA *Operational Transconductance Amplifier* (Amplificateur opérationnel à transconductance)

OTA-C *Operational Transconductance Amplifier and Capacitor* (Amplificateur opérationnel à transconductance et capacité)

PLD *Programmable Logic Device* (Circuit logique programmable)

PMOS *P-type Metal-Oxyde Semiconductor* (semi-conducteur métal-oxyde de type P)

PSRR *Power Supply Rejection Ratio* (Taux de réjection d'alimentation)

PWM *Pulse Width Modulation* (Modulation de largeurs d'impulsions)

RC *Resistance and Capacitor* (Résistance et capacité)

RN Réseau de Neurones

RNA Réseau de Neurones Artificiels

RNB Réseau de Neurones Biologiques

RP Rétro Propagation

RPD Rétro Propagation Dynamique

RPD1 Rétro Propagation Dynamique première approche

RPD2 Rétro Propagation Dynamique deuxième approche

RPS Rétro Propagation Statique

RTRL *Real Time Recurrent Learning* (Apprentissage récurrent en temps réel)

SBP *Static Back Propagation* (Rétro propagation statique)

SC *Switched Capacitor* (Capacité commutée)

SISO *Single Input Single Output* (Sortie unique et entrée unique)

TDLNN *Tapped Delay Line Neural Network* (Réseau de neurones avec ligne de délais retardés)

TSMC *Taiwan Semiconductor Matufacturing Company* (Fabrique de semi-conducteur de Taiwan)

VLSI *Very Large Scale Integration* (Intégration à très grande échelle)

Liste des symboles

y	sortie du système
$\mathbb{R}$	ensemble des nombres réels
$\mathbb{N}$	ensemble des nombres entiers naturels
$\Re$	opérateur de pseudo inversion approximative de N_I
ξ	erreur quadratique instantanée
η	taux d'adaptation de l'algorithme de rétro propagation
ρ	résistivité d'un matériau
I_{m_0}	hyper cube unitaire m_0-dimensionnel
$C(I)$	espace des fonctions continues sur I
ε	réel positif, seuil d'erreur d'approximation
α_i	constantes réelles du modèle d'approximation
β_i	constantes réelles du modèle d'approximation
w_{ij}	constantes réelles du modèle d'approximation
φ	fonction d'activation non linéaire d'un réseau de neurones multicouches
$\{\bullet\}$	séquence d'un ensemble d'échantillons du signal $\bullet$
y^*	valeur désirée ou idéale de y
y_d	valeur désirée ou idéale de y
$\mathbb{N}_n$	sous-ensemble contigu de $\mathbb{N}$

n_{cv} temps discrétisé après lequel la loi de commande a convergé

$w_{ij}^{(1)}$ poids synaptique du réseau N_I reliant la $j^{\text{ième}}$ entrée au $i^{\text{ième}}$ neurone caché N_i

$w_{1j}^{(2)}$ poids synaptique du réseau N_I reliant le $j^{\text{ième}}$ neurone caché au neurone de sortie N_s

$\hat{\bullet}$ valeur estimée de la variable $\bullet$

S_i somme interne du neurone N_i pour le réseau N_I

x_i sortie du neurone N_i pour le réseau N_I

x_s sortie du neurone N_s pour le réseau N_I

$v_{ij}^{(1)}$ poids synaptique du réseau N_C reliant la $j^{\text{ième}}$ entrée au $i^{\text{ième}}$ neurone caché N_i

$v_{1j}^{(2)}$ poids synaptique du réseau N_C reliant le $j^{\text{ième}}$ neurone caché au neurone de sortie N_s

$\sum_i$ somme interne du neurone N_i pour le réseau N_C

ζ_i sortie du neurone N_i pour le réseau N_C

ζ_s sortie du neurone N_s pour le réseau N_C

e_I erreur d'identification

e_u erreur de commande

z^{-1} délai temporel discret d'une période

e_C erreur de trajectoire

P_{N_I} ensemble ou sous-ensemble des paramètres du réseau N_I

ξ_t erreur quadratique totale

$\Delta\bullet$ variation de la valeur de la variable $\bullet$ entre deux instants d'échantillonnage

$O(\bullet)$ ordre de complexité de calcul d'un algorithme

ε_{dec} paramètre de décroissance des poids synaptiques

α paramètre de *momentum*

ξ_I critère quadratique pour le réseau N_I

η_I taux d'apprentissage du réseau N_I

ξ_C critère quadratique pour le réseau N_C

η_C taux d'apprentissage du réseau N_C

π variable du développement de RPD2

σ variable du développement de RPD2

j_a variable pour le changement d'indices dans le développement de RPD2

∇u dérivée de u par rapport au poids v dans RTRL

δ_{ki} symbole de Kronecker

ε_{qm} erreur quadratique moyenne sur un ensemble d'échantillons

$\overline{\varepsilon}_{qm}$ moyenne des valeurs de ε_{qm} obtenues pour différentes initialisation aléatoires des paramètres

$\varepsilon_{qm,i}$ erreur quadratique moyenne obtenue pour la i^{ieme} initialisation aléatoire des paramètres

ω vitesse angulaire du moteur

θ position angulaire du moteur

α constante positive pour l'évaluation du rendement de fabrication de circuits sur silicium

$\bullet^+$ borne positive d'une tension différentielle ou d'un courant différentiel

$\bullet^-$ borne négative d'une tension différentielle ou d'un courant différentiel

V_{DD} tension d'alimentation positive des circuits

V_{SS} tension d'alimentation négative des circuits

e_{ra} erreur relative

v_o tension de sortie

v_i tension d'entrée

v_{CMFB} tension de mode commun de l'AOT

μ_n mobilité des porteurs

C_{ox} capacité d'oxyde de grille

V_{eff} tension efficace

$\varphi'_{\bullet}$ dérivée de la fonction φ au point $\bullet$

R_{W} résistance d'une unité carrée de surface de matériau

Σ_i montage sommateur de courants

Σ_v montage sommateur de tensions

i, I	courant électrique
v, V	tension électrique
a	nombre de délais sur u dans le modèle du système
b	nombre de délais sur y dans le modèle du système
C	capacité
D	drain d'un transistor
$\mathbf{E}_C$	espace d'entrée de la fonction g
$\mathbf{E}_I$	espace d'entrée de la fonction f
F	approximation de la fonction f dans le théorème d'approximation universel
f	fonction non linéaire modélisant le système
FF	modèle avec NMOS rapide et PMOS rapide
FS	modèle avec NMOS rapide et PMOS lent
g	fonction non linéaire modélisant la loi de commande idéale
G	grille d'un transistor
i_b	courant de polarisation du multiplieur de Gilbert
I_D	courant de drain d'un transistor
J_{mot}	coefficient d'inertie du moteur
K	constante positive pour l'évaluation du rendement de fabrication de circuits sur silicium
k	constante de proportionalité dans l'expression du courant de sortie du multiplieur de Gilbert
L	longueur d'un transistor
l	longueur pour le calcul de la résistivité d'un matériau
M	nombre de neurones sur la couche cachée du réseau N_I
M	transistor
n	temps discrétisé
N	nombre de neurones sur la couche cachée du réseau N_C
N	type de transistor NMOS
N_C	réseau de neurones pour le contrôle du système
N_I	réseau de neurones pour l'identification du système
N_i	$i^{\text{ième}}$ neurone caché du réseau N_I ou du réseau N_C

N_s	neurone de sortie du réseau N_I ou du réseau N_C
P	type de transistor PMOS
R	résistance
R_{DS}	résistance drain-source d'un transistor
s	opérateur de Laplace
S	source d'un transistor
SF	modèle avec NMOS lent et PMOS rapide
s_i	$i^{\text{ième}}$ entrée du réseau N_I
SS	modèle avec NMOS lent et PMOS lent
T_f	couple total de frottements
t_i	$i^{\text{ième}}$ entrée du réseau N_C
T_s	période d'échantillonnage
TT	modèle standard de la technologie CMOS
u	signal de commande appliqué au système
V_{DS}	tension drain-source d'un transistor
v_{ech}	tension de réglage de la période d'échantillonnage
v_G	tension de grille d'un transistor
V_{GS}	tension grille-source d'un transistor
V_T	tension de seuil de conduction d'un transistor
W	largeur d'un transistor

Chapitre 1

Introduction

Le premier chapitre de la thèse vise à présenter la problématique de recherche, les objectifs, les contributions visées et la méthodologie de recherche. Nous y expliquerons pourquoi la mise en œuvre d'algorithmes de commande peut impliquer, pour certaines applications, une double problématique : (i) problématique algorithmique ; (ii) problématique d'intégration sur silicium. Cette double problématique mènera à la présentation des objectifs de recherche et des contributions scientifiques visées par la présente thèse. Nous décrirons alors la méthodologie de recherche qui sera utilisée et qui permettra d'atteindre les objectifs et les contributions scientifiques escomptées. Ce premier chapitre s'achèvera par une brève description de la structure de la thèse et des différents chapitres qui la composent.

1.1. Problématique

1.1.1. Une double problématique : commande et ITGE

Les tendances récentes dans l'industrie de la haute technologie et dans notre structure socioéconomique démontrent que la commande et le traitement d'information ont joué un rôle dominant dans les développements des dernières décennies. Ces tendances nous permettent également de penser, selon toute vraisemblance, que cette domination continuera de s'accroître dans les décennies à venir.

Les applications modernes dans le domaine de la commande visent des niveaux de performances toujours plus élevés. Citons par exemple les systèmes de positionnement de grande précision pour la manipulation de composantes (exemple : assemblage des pièces sur circuits imprimés, manipulation des gaufres des circuits intégrés, robots en médecine, etc.), les manipulateurs de grandes dimensions utilisés dans l'espace, etc. La conception, la réalisation et l'utilisation de tels systèmes nécessitent la convergence des efforts de recherche dans plusieurs domaines, principalement :

- la mécanique : utilisation et synthèse de nouveaux matériaux, développement des technologies de fabrication et d'assemblage,
- la théorie : développement de la théorie de la commande, des modèles nécessaires à la caractérisation et à la commande des systèmes, etc.,
- l'électronique : développement de circuits électroniques performants permettant une mise en œuvre efficace des algorithmes de traitement de signal dans une vaste gamme d'applications.

L'établissement de lois de commande pour des systèmes modernes est d'autant plus complexe que le comportement de ceux-ci possède généralement des caractéristiques particulières [Sweet & Good '85 ; Park *et al.* '96 ; Behera *et al.* '96] :

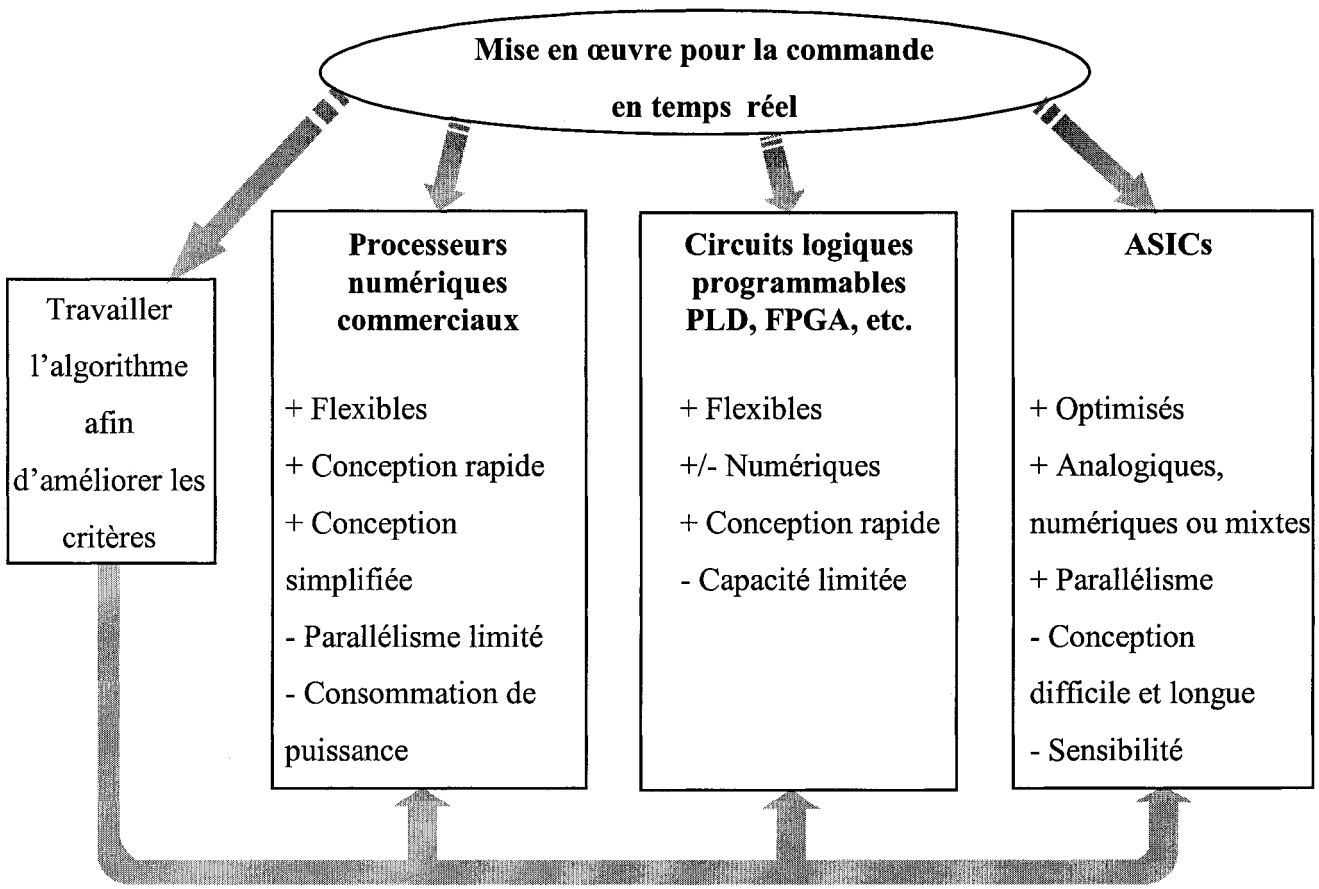
- non-linéarités : le système se comporte de façon non linéaire ; la théorie de commande des systèmes non linéaires est beaucoup moins uniforme que celle des systèmes linéaires, l'approche demeurant généralement propre à un système particulier ou limitée à une classe de systèmes soumis à des conditions très restrictives,
- dynamique : le comportement du système à un instant donné dépend non seulement de son entrée présente mais également de ses entrées et sorties passées. Ce comportement comprend donc un certain effet de mémoire qui doit être considéré dans l'élaboration de la loi de commande,
- variations de paramètres : certains paramètres du système peuvent varier pendant l'utilisation du système, par exemple en fonction de grandeurs d'influence telles que le temps, la température, etc.,

- modèle mathématique du système inconnu : dans certains cas, le modèle mathématique du système est partiellement ou totalement inconnu, ce qui rend la conception de la loi de commande beaucoup plus difficile.

Les algorithmes permettant d'apporter une solution satisfaisante au problème de commande de systèmes soumis à des caractéristiques si contraignantes requièrent généralement un traitement numérique comprenant un grand nombre d'opérations ou des opérations mathématiques très compliquées. L'efficacité de ce traitement dans les cas pratiques résulte d'un bon équilibre entre les exigences de la théorie et les possibilités de mise en pratique [Massicotte '95]. Dans la pratique, les algorithmes de commande peuvent être implantés dans différents types de circuits électroniques de calcul tels que présentés dans la figure 1.1.

Une première étape est, lorsque cela est possible, de tenter de retravailler l'algorithme de commande afin d'en faciliter l'ITGE. Quel que soit le résultat de cette étape et quel que soit le type d'implantation envisagé, une diminution de complexité est généralement bénéfique à l'ITGE. Ensuite, le choix du type de circuit dans lequel sera implanté l'algorithme de commande peut être fait dans l'une des trois catégories illustrées dans la figure 1.1. La première catégorie est constituée des processeurs numériques commerciaux les plus usuels : DSP (*angl. Digital Signal Processing*), microprocesseurs, microcontrôleurs, etc. Ce type de processeurs offre une grande flexibilité et une bonne facilité de mise en œuvre. Ces processeurs sont toutefois limités en vitesse par une architecture séquentielle et leur flexibilité est au dépend d'une surface d'intégration et d'une consommation de puissance plus élevées. À l'opposé, les ASIC (*angl. Application Specific Integrated Circuit*) sont des circuits spécifiquement conçus pour intégrer un algorithme particulier. Cette spécificité permet une plus grande exploitation du parallélisme et rend possible l'optimisation de critères d'ITGE tels que la vitesse, la surface d'intégration ou la consommation de puissance. Le type d'intégration peut être analogique, numérique ou mixte. La conception est souvent beaucoup plus difficile, surtout si le degré d'optimisation recherché est élevé. Les circuits logiques programmables représentent une catégorie intermédiaire aux deux précédentes. Ces circuits sont généralement composés d'un certain nombre de portes logiques élémentaires configurables permettant de réaliser des fonctions logiques plus ou moins complexes. Les outils logiciels pour la conception et la réalisation de circuits dans des FPGA

(*angl. Field Programmable Gate Array*) facilitent et accélèrent l'intégration. Cependant, le nombre de portes disponibles étant limité, la complexité de la fonction qu'il est possible d'intégrer est également limitée. Dans cette catégorie, ainsi que dans la catégorie des processeurs commerciaux, le traitement est exclusivement numérique. Notons toutefois que les FPAA (*angl. Field Programmable Analog Array*), équivalents analogiques des FPGA, sont en développement et commencent à voir le jour.



+ Avantage ; - Inconvénient ; +/- Avantage et inconvénient

Figure 1.1. Différents types de circuits électroniques de calcul pour la mise en œuvre d'algorithmes de commande

En général, les algorithmes de commande sont implantés dans des circuits de la première catégorie, c'est-à-dire celle des processeurs numériques commerciaux tels que les DSP ou autres microprocesseurs. Ceux-ci ont bénéficié de très grands efforts des manufacturiers afin d'obtenir de grandes vitesses de calculs tout en maintenant une grande flexibilité d'utilisation. De plus,

certains de ces processeurs traitent les calculs au format à virgule flottante et donc à très grande précision. Ces circuits, à usage général en traitement numérique du signal, ont permis et permettent encore de satisfaire la majorité des applications. Cependant, la dernière décennie a vu naître de nouvelles problématiques dans ce domaine.

La technologie d'intégration sur silicium, exprimée en μm, donne les dimensions du plus petit transistor réalisable. Ce chiffre est directement relié à la densité d'intégration puisque plus la technologie progresse, plus le nombre de transistors réalisables dans une même unité de surface est grand, c'est-à-dire plus la densité d'intégration augmente. Aujourd'hui, les technologies 0.18μm et 0.13μm sont utilisées et on parle déjà de technologie 0.1μm. De plus, la vitesse d'horloge des processeurs numériques est directement affectée par cette technologie et varie inversement à la diminution de la taille minimale des transistors. Ainsi, la progression continuelle des technologies a entraîné une progression continuelle de la vitesse d'horloge des processeurs, permettant des calculs de plus en plus rapides. Pendant les deux dernières décennies, cet axe de développement a été exploité au maximum, chaque nouvelle technologie donnant naissance à de nouvelles versions de processeurs, toujours plus denses et plus rapides, même sans évolution architecturale. Cette progression atteindra, dans un avenir très proche, les limites possibles de la technologie CMOS. En effet, certaines propriétés physiques des matériaux utilisés dans l'intégration sur silicium imposent des limites théoriques [Bhavnagarwata *et al.* '98 ; Wong *et al.* '99 ; Taur *et al.* '97]. Les différentes études menées à ce sujet ont déterminé la limite théorique de progression de la technologie d'intégration sur silicium selon le procédé actuel de type CMOS à environ 0.075μm. Les chercheurs ont aussi estimé que cette limite serait atteinte vers 2010, mais d'après le rythme d'évolution actuel, il n'est pas déraisonnable de penser qu'elle pourrait être atteinte encore plus tôt. La perspective d'atteindre ces limites a créé un regain d'intérêt pour d'autres axes de développement, notamment au niveau des architectures des processeurs.

D'un autre côté, la demande pour des circuits à faible consommation d'énergie ou de faible surface est de plus en plus importante [Sanchez-Sinencio & Silva-Martinez '00]. Les principales raisons à l'origine de cette demande peuvent être résumées comme suit :

- l'évolution de la technologie d'intégration aboutit à des circuits toujours plus denses et plus rapides qui consomment par conséquent beaucoup plus de puissance,

- la volonté de créer des systèmes autonomes dans certaines applications nécessite des circuits miniatures ou à faible consommation de puissance,
- le surcroît de consommation d'énergie s'accompagne d'une augmentation de la dissipation de chaleur et donc de problèmes de refroidissement des circuits,
- dans certains domaines, les réglementations liées à la protection de l'environnement imposent des limites de consommation de puissance.

Les concepteurs de circuits se retrouvent alors confrontés à un problème de taille : comment fabriquer des circuits toujours plus rapides malgré l'atteinte des limites des progrès des technologies d'intégration sur silicium et les contraintes de consommation de puissance ou de surface d'intégration ? Face à ce défi, certains chercheurs s'orientent vers la recherche d'autres matériaux ou d'autres procédés de fabrication permettant de repousser les limites des procédés actuels. Citons par exemple l'Arséniure de Gallium (GaAs) ou la fabrication de transistors à double grille. Citons également des axes de recherche pour le développement de nouveaux types de transistors dans le domaine moléculaire : transistor à molécules d'Acide Désoxyribose Nucléique (ADN), transistor à molécules de rotaxane, transistor quantique, etc. Une autre solution est de développer des processeurs dédiés spécifiquement aux applications visées, les ASIC. En développant un circuit dédié à une application, l'architecture peut être pensée afin d'optimiser la vitesse de calcul, la consommation de puissance ou la surface d'intégration selon le besoin. Ainsi, pour une technologie donnée et pour un algorithme particulier, un ASIC peut offrir des performances nettement supérieures à celles du meilleur des processeurs commerciaux réalisé dans la même technologie.

Finalement, la double problématique peut se résumer ainsi : comment réaliser l'ITGE d'algorithmes de commandes compliqués en satisfaisant les critères d'intégration des applications modernes (vitesse, consommation de puissance, surface) et ce malgré les limites technologiques ?

1.2. RNA : une solution attrayante

La double problématique algorithmique et d'ITGE a créé, depuis environ quinze ans, un regain d'intérêt pour les Réseaux de Neurones Artificiels (RNA), notamment dans le domaine de la commande de systèmes dynamiques non linéaires [Narendra '97].

1.2.1. Du point de vue algorithmique

Inspirés de la capacité évidente des réseaux de neurones biologiques du cerveau humain à résoudre des problèmes de traitement d'information très complexes, les RNA représentent une solution algorithmique attrayante pour la commande. En effet, du point de vue de traitement de l'information, le cerveau peut être perçu comme un organe capable de recevoir de grandes quantités d'informations de son environnement, d'effectuer des traitements très complexes et rapides sur ces informations - par exemple dans le traitement d'images - et de déclencher des actions appropriées aux informations reçues. Dans bien des cas, le traitement réalisé par le cerveau est beaucoup plus rapide et précis qu'un traitement similaire effectué sur le plus rapide des processeurs actuels. C'est une des raisons pour lesquelles, dans quelques domaines, des chercheurs s'intéressent à reproduire les capacités du cerveau humain. On peut par exemple imaginer un contrôleur de bras robotique articulé capable de commander et de mouvoir le bras dans un mouvement aussi précis et contrôlé que celui de l'homme. Le cerveau de l'homme recèle également d'autres capacités intéressantes telles que l'intuition, la conscience et d'autres propriétés abstraites et dont le fonctionnement précis au niveau biologique échappe encore en grande partie à la connaissance de la Science.

Les RNA sont des modèles synthétiques des réseaux de neurones biologiques établis par les chercheurs en biologie. En reproduisant ces modèles, les RNA possèdent sans aucun doute possible des capacités extraordinaires pour résoudre des problèmes de traitement de signal très complexes, même si bien sûr, les réseaux réalisables en pratique sont encore loin d'équivaloir les réseaux biologiques en termes de nombre de neurones ou de connexions synaptiques mis en œuvre.

Dans le cadre de la commande de systèmes dynamiques non linéaires, les RNA sont attrayants pour :

- leur capacité de modélisation de fonctions non linéaires,

- leur capacité d'adaptation,
- leur capacité de généralisation,
- leur tolérance aux pannes.

Ces différents points seront revus en détails dans la section 2.1 du deuxième chapitre.

1.2.2. Du point de vue de l'ITGE

Du point de vue de l'ITGE, les RNA sont bien adaptés au développement d'ASIC pour l'optimisation des critères d'intégration mentionnés dans la section 1.1.1 du premier chapitre.

Le parallélisme massif des RNA favorise une intégration dans des architectures hautement parallèles pour des vitesses de calcul accrues. La simplicité des opérations mathématiques mises en œuvre facilite la conception des circuits d'ITGE que ce soit dans le domaine numérique, analogique ou mixte.

Ainsi, beaucoup de chercheurs s'intéressent au problème du développement d'ASIC pour l'ITGE des RNA [Diotaveli *et al.* '00 ; Burton *et al.* '97 ; Bo *et al.* '99 ; Coué & Wilson '96 ; El-Masry *et al.* '97]. Pour les RNA de type perceptron multicouche, de nombreux circuits analogiques ou numériques ont été proposés et publiés dans la littérature. De nombreux circuits pour l'intégration de la partie de propagation des données dans un réseau ont été développés et mis en œuvre avec succès.

Cependant, l'obstacle majeur à l'ITGE des RNA demeure la partie d'adaptation des poids synaptiques et c'est ce sujet qui nécessite encore un important investissement de recherche. Le plus souvent, l'algorithme d'adaptation n'est pas considéré - par exemple lorsque l'apprentissage du réseau est réalisé hors ligne - ou bien intégré dans un processeur numérique connexe à la partie de propagation. Cette approche permet d'obtenir des circuits flexibles puisque l'algorithme d'adaptation peut être modifié par programmation du processeur, mais l'optimisation des critères d'ITGE est limitée aux caractéristiques d'un circuit numérique à usage général comme nous l'avons mentionné dans la section 1.1.1 (*cf.* figure 1.1). La double problématique algorithmique et d'ITGE fait de l'intégration de l'algorithme d'adaptation la difficulté majeure. Cette difficulté résulte d'exigences contraires des deux familles de critères : (i) les algorithmes d'adaptation plus performants sont généralement plus complexes et donc plus difficiles à intégrer ; (ii) les algorithmes d'adaptation moins complexes sont généralement plus faciles à intégrer mais aussi moins performants. De l'étude des publications dans ce domaine, il ressort un besoin de

développer des algorithmes d'adaptation qui offrent de bonnes performances en termes algorithmiques et qui puissent être intégrés efficacement dans des ASIC. Ce développement algorithmique qui est le sujet central de la thèse sera présenté dans le troisième chapitre.

1.2.3. Exemples d'applications

De nombreux travaux de recherche traitent de l'utilisation des RNA dans différents domaines d'applications. Nous présentons ci-après quelques exemples d'applications des RNA dans différents domaines. Ceci n'a pas pour but de présenter une revue exhaustive de l'application des RNA, mais simplement de montrer, par quelques exemples, la vaste étendue de leur applicabilité.

Citons par exemple des applications dans le domaine médical :

- prédiction d'hypertrophie ventriculaire [Elias *et al.* '97],
- reconnaissance d'arythmie cardiaque à partir d'électrocardiogrammes [Nabney *et al.* '01],
- stimulation fonctionnelle électrique pour l'aide à la locomotion de personnes handicapées [Abbas & Triolo '97],
- etc.

Citons également des applications en traitement d'images :

- contrôle d'évitement d'obstacles pour un véhicule [Nijhuis *et al.* '91],
- détection de défauts en conception de circuit d'ITGE [de Trémiolles *et al.* '97],
- etc.

Et bien d'autres applications en commande :

- contrôle de moteur à induction [Burton *et al.* '97 ; Ba-Razzouk *et al.* '97],
- contrôle de manipulateur robotique [Ge *et al.* '97],
- contrôle de transmission à courant alternatif [Chen et al. '00],
- commande de trajectoire d'un membre flexible [Gutierrez *et al.* '98],
- compensation de frottements [Kim & Lewis '00],
- etc.

Le grand nombre de publications scientifiques concernant l'application des RNA en commande témoigne d'un intérêt grandissant pour cet outil algorithmique. Les besoins actuels de circuits d'ITGE toujours plus rapides, de plus faible consommation de puissance et de surface d'intégration réduite, ainsi que l'intérêt évident des RNA dans le domaine de la commande justifient la pertinence de travaux de recherche dans l'ITGE des RNA, tant pour des applications présentes qu'en prévision d'applications futures.

1.3. Objectifs de recherche et contributions scientifiques

D'un point de vue général, le projet de recherche concerne le développement et la modélisation, dans une architecture dédiée, d'algorithmes adaptatifs à base de RNA pour la commande de systèmes dynamiques non linéaires. L'objectif général est de proposer une solution pratique et efficace pour la commande en temps réel de ces systèmes, solution satisfaisant aux contraintes des applications modernes en commande (*cf.* section 1.1.1 de ce chapitre).

Après l'analyse de la problématique de la section 1.1, l'objectif général de la thèse peut être décomposé en sous-objectifs :

- proposition de nouveaux algorithmes adaptatifs à base de RNA pour la commande de systèmes dynamiques non linéaires,
- validation des algorithmes proposés par des résultats de simulations mettant en évidence la fonctionnalité des algorithmes ainsi que leur apport en comparaison aux algorithmes de référence de la littérature,
- proposition d'une architecture hautement parallèle en vue d'une intégration sur silicium des algorithmes proposés,
- modélisation de l'architecture proposée dans des circuits d'ITGE,
- simulation des circuits d'ITGE proposés au niveau des dessins de masques sur silicium et évaluation des performances, incluant des simulations de la sensibilité des circuits aux défauts de fabrication,
- évaluation des performances de l'architecture au niveau de la simulation,
- énoncé de suggestions pour des travaux futurs basés sur la présente thèse.

La contribution scientifique apportée par la thèse peut être décomposée comme suit :

- de nouveaux algorithmes adaptatifs pour la rétro propagation à travers le modèle neuronal dans une structure de commande indirecte inverse. Ces algorithmes s'avèreront, par rapport aux algorithmes de référence de la littérature, de meilleurs compromis entre les différents critères algorithmiques et d'ITGE,
- une architecture hautement parallèle pour la modélisation d'ITGE de la loi de commande développée tout au long de la thèse, architecture intégrant le ou les nouveaux algorithmes développés,
- la modélisation de circuit CMOS pour l'ITGE de l'architecture proposée. La contribution à ce niveau sera surtout le circuit global obtenu, démontrant d'une part la faisabilité d'un ASIC analogique pour l'ITGE de la loi de commande proposée, et d'autre part la possibilité d'obtenir un circuit qui peut satisfaire à des applications en commande de systèmes dynamiques non linéaires.

1.4. Méthodologie

La solution générale à la double problématique décrite dans la section 1.1.1 peut être trouvée en suivant deux axes de recherche :

- développement algorithmique : développer des algorithmes permettant de commander de manière satisfaisante des systèmes ayant les caractéristiques particulières mentionnées dans la section 1.1.1 du premier chapitre,
- développement au niveau de l'ITGE : développement au niveau architectural ou au niveau des circuits de base en vue d'une intégration satisfaisante des algorithmes de commande développés.

Bien sûr, ces deux axes de développement sont étroitement liés. En effet, ce sont les opérations de l'algorithme de commande qui sont intégrées sur le silicium. Ainsi, toute caractéristique de ces opérations, par exemple le type des opérations mathématiques ou leur quantité, a une influence directe sur le circuit final d'ITGE. D'autre part, tout choix au niveau architectural ou au niveau des circuits d'ITGE a une influence directe sur les performances de

l'algorithme de commande (précision, vitesse de calcul, etc.). Les objectifs du projet seront donc atteints en travaillant conjointement sur les deux axes afin d'aboutir à un compromis efficace entre les contraintes liées à chacun des axes. Il est très important de toujours considérer les deux critères dans les étapes de développement. Ainsi, à chacune des étapes, nous devrons nous assurer de satisfaire aux critères de commande et d'ITGE avant de passer à l'étape suivante. En effet, si nous développons un algorithme de commande très performant en terme algorithmique sans nous soucier des critères d'ITGE, nous risquons, lors de l'étape de conception des circuits d'ITGE, de conclure que l'algorithme est trop complexe et qu'il faut donc revenir à l'étape de développement algorithmique. La figure 1.2 présente les différentes étapes qui composent la méthodologie de recherche utilisée ainsi que les correspondances avec les différents chapitres de la thèse.

Étape 1 - Définition du problème : il s'agit dans cette première étape de définir le problème traité dans la thèse et d'établir les objectifs de recherche. Cette étape est présentée dans les sections 1 et 2 du premier chapitre. À cette étape, nous exposons le besoin de tenir compte tant des critères algorithmiques que des critères d'ITGE afin d'aboutir à des circuits efficaces en vue d'une intégration sur le silicium de la loi de commande.

Étape 2 - Choix d'un outil algorithmique : une fois le problème posé, il faut choisir quel outil algorithmique sera utilisé comme base de développement de la loi de commande. Cette étape est cruciale puisque l'outil algorithmique représente en quelque sorte une passerelle entre les domaine algorithmique et d'ITGE. Le choix doit tenir compte à la fois des contraintes algorithmiques et des contraintes d'intégration. Ensuite, il faudra définir la structure de commande retenue pour la thèse. Notons qu'aucune contribution n'est envisagée dans la structure de commande même mais bien dans les algorithmes d'adaptation des RNA qui y seront inclus. Cette étape continuera avec la présentation des topologies des RNA utilisés ainsi que des hypothèses sur le système commandé et des contraintes d'utilisation de la loi de commande. Enfin, la stabilité étant un problème important en commande, nous exposerons une revue de littérature présentant brièvement l'état actuel de la recherche dans l'étude de stabilité des structures de commande indirecte inverse à base de RNA.

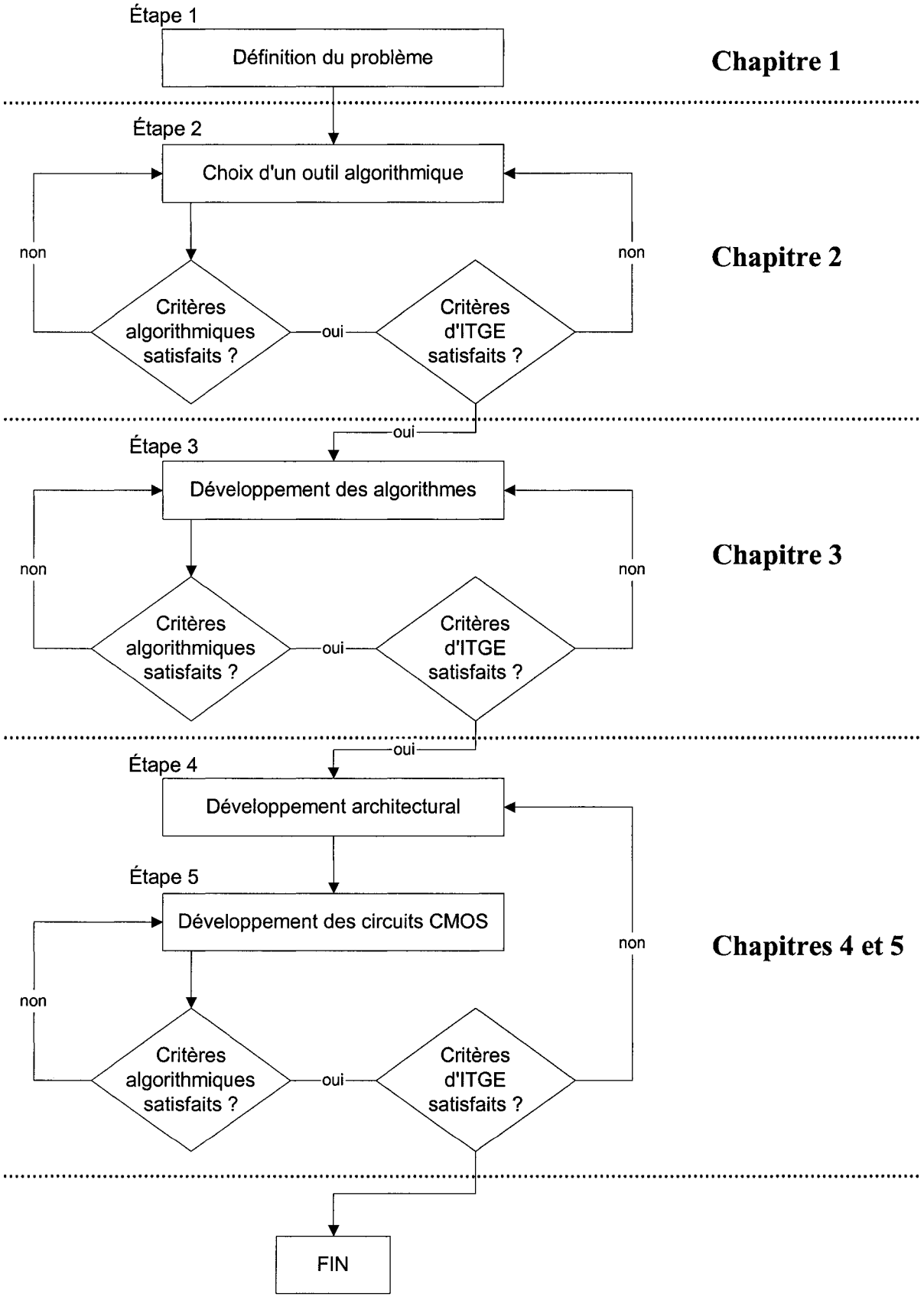


Figure 1.2. Méthodologie de recherche

Étape 3 - Développement de nouveaux algorithmes d'adaptation : une étude approfondie des algorithmes d'adaptation présentés dans la littérature devra être menée et aboutir à un ou plusieurs axes de développement de nouveaux algorithmes pour le type de RNA utilisé. Cette étape devra comprendre des simulations numériques des différents algorithmes et une étude comparative des résultats obtenus suivant les critères d'intérêt. L'étude comparative des résultats obtenus avec les nouveaux algorithmes proposés par rapport aux algorithmes de référence de la littérature permettra de mettre en évidence les avantages des nouveaux algorithmes proposés tant du point de vue de la commande que de l'ITGE et de conclure quant à l'algorithme à retenir pour les étapes suivantes.

Étape 4 - Développement architectural : avant de commencer à développer des circuits pour la réalisation des opérations mathématiques des algorithmes à intégrer, il faut représenter les circuits aux niveaux hiérarchiques les plus hauts. Ceci aboutira à une architecture contenant des blocs fonctionnels de haut niveau. Cette étape a une influence sur la mise en parallèle des opérations des algorithmes et aussi sur la consommation de puissance et sur la surface totale du circuit puisque le nombre de circuits nécessaires affecte ces caractéristiques.

Étape 5 - Modélisation des circuits CMOS : les blocs fonctionnels décrits dans la quatrième étape seront décomposés en fonctions élémentaires de plus bas niveau réalisant les fonctions mathématiques de base nécessaires à l'architecture développée dans la quatrième étape. L'étude bibliographique des possibilités d'intégration mènera aux choix relatifs aux types de circuits utilisés pour réaliser ces fonctions en tenant compte à la fois des critères de précision, de consommation de puissance et de surface d'intégration. Les dessins de masque sur silicium des différents circuits seront alors modélisés et simulés. Les résultats de simulations de la loi de commande complète au niveau des dessins de masques permettront de vérifier le bon fonctionnement de l'ensemble et d'évaluer les performances obtenues au niveau de la simulation. Finalement, des simulations aux coins (*angl. corners simulations*) permettront, au niveau de la simulation, d'évaluer la sensibilité des circuits aux défauts de fabrication. Ceci constituera le dernier niveau dans l'étape 5.

N. B. À certaines étapes, les critères devront être évalués de manière qualitative à défaut de pouvoir être évalués de manière quantitative. Par exemple, lors de la deuxième étape, l'évaluation des critères pour le choix de l'outil algorithmique repose surtout sur une évaluation qualitative.

1.5. Structure de la thèse

Les correspondances entre les différentes étapes décrites dans la section précédente et les différents chapitres de la thèse ont été mentionnées dans la figure 1.2. Les raisons du choix des RNA comme outil algorithmique seront détaillées dans le deuxième chapitre. La structure de commande ainsi que les topologies des RNA utilisés y seront également présentées. Ce chapitre s'achèvera par la présentation des hypothèses et des contraintes relatives au système commandé et à la loi de commande, ainsi qu'une revue de littérature concernant le problème de stabilité. Le troisième chapitre concernera les algorithmes d'adaptation. Après une revue des algorithmes de référence de la littérature, les nouveaux algorithmes développés y seront présentés ainsi que des résultats de simulations comparatifs mettant en évidence les avantages des nouveaux algorithmes apportés par cette thèse. Le quatrième chapitre traitera du développement de l'architecture pour la modélisation au niveau de l'ITGE de la loi de commande développée dans les deuxième et troisième chapitres. Après justification des différents choix technologiques, l'architecture pour la modélisation au niveau de l'ITGE de la loi de commande sera présentée. Dans le même chapitre, les circuits CMOS pour la modélisation des fonctions mathématiques de l'architecture proposée seront détaillés. Le cinquième chapitre présentera les résultats de simulations de la loi de commande complète au niveau des dessins de masques. Nous évaluerons alors les résultats obtenus tant au niveau des critères algorithmiques que des critères d'ITGE. Nous y présenterons également des résultats de simulations aux coins afin d'évaluer la sensibilité des circuits aux défauts de fabrication. Enfin, le sixième chapitre conclura la thèse avec les rappels des principaux résultats obtenus et les suggestions de travaux futurs.

Étant donné que les chapitres 2, 3 et 4 traitent de domaines bien distincts, l'état de la recherche relatif à chacun des domaines sera présenté dans chacun des trois chapitres, principalement au début.

Chapitre 2

Commande indirecte inverse à base de RNA

Ce chapitre présente la structure de commande utilisée dans la thèse. Après une justification détaillée du choix de l'utilisation des RNA comme outil de développement algorithmique, nous présenterons les différentes hypothèses de départ relatives au système commandé et à son contrôleur, hypothèses importantes dans le choix de la structure de commande et des topologies de réseaux utilisées. Ensuite, la structure générale de commande indirecte inverse à base de RNA sera exposée. Cette section comprendra la présentation des structures et des équations de propagation des réseaux utilisés ainsi que l'explication du rôle de chaque réseau. Les principes d'adaptation pour chacun des réseaux mis en œuvre dans la loi de commande seront également brièvement expliqués dans ce chapitre. Ensuite, nous reviendrons plus en détails sur les différentes hypothèses et conditions relatives à la loi de commande. Enfin, nous présenterons une brève revue de l'état de la recherche sur la stabilité des structures de commande indirecte inverse à base de RNA.

2.1. Introduction

La théorie du contrôle a été développée avec succès pour le traitement de systèmes dynamiques linéaires invariants dans le temps. Une multitude de méthodes et de théorèmes dans le domaine fréquentiel ou de la représentation d'état ont atteint un niveau auquel la conception de contrôleurs est devenue systématique, même en présence de perturbations et d'erreurs de modélisation [Sepulchre *et al.* '97]. Chacune de ces méthodes peut être utilisée pour atteindre la stabilisation, le suivi de trajectoire, l'atténuation de perturbations et d'autres objectifs de commande. La situation est radicalement différente dans le cas des systèmes non linéaires. Bien que certaines méthodologies de conception commencent à émerger, une approche aussi systématique que dans le cas des systèmes linéaires n'est pas envisageable. En effet, la grande diversité des phénomènes non linéaires suggère une diversité comparable des outils et des procédures de conception de contrôleurs pour ces systèmes. Même si, dans certains cas, les non linéarités peuvent être négligées afin de traiter le système comme quasi linéaire et utiliser les méthodes développées pour les systèmes linéaires, les exigences des applications modernes de la commande ne favorisent pas ce type d'approche. Citons par exemple les manipulateurs robotiques affectés par des phénomènes de frottements et de flexibilité articulaire [Armstrong & Canudas de Wit '96 ; Cetinkunt & Book '90], les phénomènes d'hystérésis dus aux retours de dents dans les engrenages de boîtes de transmission [Seidl et al. '95], la non linéarité des modèles de boîte de transmission [Tuttle & Seering '96], etc. Parmi les effets indésirables engendrés par ces phénomènes, citons l'usure prématurée des matériaux, les oscillations, les écarts de trajectoire, etc. Lorsque l'utilisation d'un système est sous-optimale, les phénomènes non linéaires peuvent éventuellement être négligés. Si par contre nous visons une utilisation optimale d'un système, une compensation ou une correction des erreurs engendrées par les phénomènes non linéaires est généralement nécessaire.

Outre les problèmes associés aux phénomènes non linéaires, ajoutons aussi que les systèmes non linéaires, tout comme leurs homologues linéaires, peuvent être dynamiques ou variants dans le temps. Ceci implique une certaine capacité de mémoire du contrôleur ainsi que la capacité de traiter des systèmes variants dans le temps, soit par commande robuste [Freeman & Kokotovic '96] soit par commande adaptative [Åström & Wittenmark '95]. Enfin, lorsqu'un modèle mathématique complet ou partiel du système à commander peut être établi, celui-ci peut

être utilisé dans la conception du contrôleur. Ce modèle constitue une information analytique essentielle sur le comportement du système commandé. Si ce modèle ne peut être établi, le problème de commande devient beaucoup plus complexe. L'approche dominante consiste alors à identifier ou estimer le modèle ou une partie du modèle et de l'utiliser ensuite dans la conception du contrôleur [Narendra & Parthasarathy '90 ; Åström & Wittenmark '95].

2.2. RNA comme outil algorithmique

La grande difficulté du problème de commande des systèmes non linéaires a amené un regain d'intérêt depuis le début des années 1990 pour les RNA. Afin de suivre la méthodologie présentée dans le premier chapitre, nous rappelons que nous devons tenir compte, dans le choix de l'outil algorithmique, des critères algorithmique et des critères d'ITGE. Dans les sections suivantes, nous présentons les avantages des RNA en termes algorithmiques et d'ITGE.

2.2.1. Avantages algorithmique des RNA pour la commande

D'un point de vue général, la citation suivante donne une bonne idée de l'intérêt porté aux RNA dans le domaine du traitement de signal [Hubel '88].

« *L'œil a souvent été comparé à une caméra. Il serait plus juste de le comparer à une caméra TV avec un trépied ajustable automatiquement, une machine avec autofocus qui s'ajuste automatiquement à l'intensité lumineuse, qui a une lentille autonettoyante, et connectée à un processeur avec des capacités de traitement parallèle si avancées que les ingénieurs commencent à considérer des stratégies similaires. Aucune invention de l'homme ne peut rivaliser avec l'œil.* »

Ainsi, inspirés des capacités extraordinaires des réseaux neuronaux biologiques de l'homme à traiter des problèmes de traitement d'informations très complexes et à commander des systèmes très perfectionnés, les chercheurs tentent de les reproduire de manière artificielle pour leurs applications. Le développement des RNA est motivé par cette analogie avec le cerveau

humain, preuve vivante que le calcul parallèle rapide et tolérant aux pannes est non seulement physiquement possible mais également très puissant.

Pour la commande en particulier, et en fonction des difficultés énoncées dans la section 2.1 de ce chapitre, les RNA ont des propriétés algorithmiques très attrayantes détaillées ci-après.

2.2.1.1. Modélisation de fonctions non linéaires

Un RNA constitué d'une interconnexion de neurones non linéaires est de fait non linéaire. Dans les années 1980, un certain nombre de preuves ont été formulées selon lesquelles les RNA de type perceptron multicouche sont capables de modéliser toute fonction non linéaire continue dans un ensemble compact à n'importe quel degré d'exactitude [Hornik & Stinchcombe '89 ; Funahashi '89]. C'est pour cette raison que les RNA de type perceptron multicouche sont qualifiés d'approximateurs universels. Le théorème d'approximation universel de fonctions non linéaires, qui peut être vu comme une extension naturelle du théorème de Weierstrass selon lequel toute fonction non linéaire continue dans un intervalle fermé de l'axe réel peut être approximée par une suite de polynômes, peut être écrit comme suit [Haykin '99] :

Théorème d'approximation universel :

« *Soit φ une fonction continue, monotone croissante, bornée et non constante. Soit I_{m_0} un hyper cube unitaire m_0-dimensionnel $[-1,1]^{m_0}$. Soit $C(I_{m_0})$ l'espace des fonctions continues sur I_{m_0}. Alors, pour toute fonction $f \in C(I_{m_0})$ et tout réel $\varepsilon > 0$, il existe des constantes réelles α_i, β_i et w_{ij} pour $i = 1,...m_1$ et $j = 1,...m_0$ telles que nous puissions définir*

$$F(x_1,...,x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + \beta_i\right) \qquad (2.1)$$

comme approximation de la fonction f, c'est-à-dire

$$\left|F(x_1,...,x_{m_0}) - f(x_1,...,x_{m_0})\right| < \varepsilon$$

pour tout $(x_1,...,x_{m_0})$ dans l'espace d'entrée de la fonction f. »

Soit un RNA avec une couche cachée, m_0 entrées notées $x_1,...,x_{m_0}$, w_{ij} les poids synaptiques connectés de la couche cachée, β_i les biais appliqués sur les neurones cachés, $\alpha_1,...,\alpha_{m_1}$ les poids synaptiques de la couche de sortie. Si la fonction d'activation des neurones de la couche cachée satisfait aux conditions sur φ et que la fonction d'activation des neurones de la couche de sortie est linéaire, ce RNA est un approximateur universel puisque sa sortie y exprimée par

$$y = \sum_{i=0}^{m_1} \alpha_i \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + \beta_i \right) \tag{2.2}$$

satisfait à la définition de la fonction F exprimée par l'équation (2.1).

Cette capacité de modélisation a également été étendue à l'identification (et au contrôle) de systèmes dynamiques non linéaires [Narendra & Parthasarathy '90 ; Mistry *et al.* '96 ; Waibel *et al.* '89 ; Levin & Narendra '96 ; Rovithakis & Christodoulou '95].

2.2.1.2. Capacité d'adaptation

Les RNA ont une capacité incorporée d'adapter les poids synaptiques selon les changements de leur environnement. Plus particulièrement, un réseau entraîné afin d'opérer dans un environnement déterminé peut être entraîné à nouveau pour traiter des variations de son environnement. De plus, si l'environnement est non stationnaire, le réseau peut être entraîné en temps réel (en ligne) pour s'adapter aux variations. Cette capacité d'adaptation couplée à la faculté de modélisation de systèmes dynamiques non linéaires est de première importance pour notre problème de commande puisqu'elle permet : (i) d'apprendre des modèles dynamiques non linéaires *a priori* inconnus à l'instant initial (en l'occurrence le modèle du système à commander et le modèle du contrôleur) ; (ii) de s'adapter aux variations du système ou de son environnement (variations de paramètres, changements de trajectoires désirées, etc.).

2.2.1.3. Capacité de généralisation

Les RNA sont très souvent utilisés pour réaliser une modélisation entrée-sortie de fonctions non linéaires. Dans l'apprentissage supervisé, le RNA apprend le modèle d'une fonction d'après un certain nombre de combinaisons d'entrées-sorties pour lesquelles les entrées et sorties correspondantes de la fonction sont connues. Dans une première phase dite

« d'apprentissage », les poids synaptiques du RNA sont ajustés jusqu'à obtention d'un état stationnaire pour lequel la qualité de modélisation pour l'ensemble des combinaisons présentées est satisfaisante. Le RNA présente alors la capacité de généraliser (interpoler), c'est-à-dire de conserver cette qualité de modélisation pour des combinaisons entrées-sorties non présentées lors de la phase d'apprentissage et ce, sans maintenir le processus d'adaptation des poids. Évidemment, la qualité de généralisation du réseau est dépendante des choix effectués à l'étape d'apprentissage, c'est-à-dire le temps d'apprentissage, l'algorithme utilisé, etc.

2.2.1.4. Tolérance aux pannes

Dire qu'un réseau de neurones est tolérant aux pannes, c'est dire qu'il peut maintenir la qualité de son traitement même après dégradation d'un ou plusieurs de ses composants. En raison de la nature distribuée de l'information dans un RNA, ces dégradations peuvent être nombreuses avant que la réponse globale du réseau ne soit affectée de manière significative. Par analogie, le cerveau humain possède la même propriété. Dès l'âge de 3 à 4 ans, le cerveau de l'homme commence à perdre des neurones ce qui ne l'empêche pas de continuer à fonctionner et même à apprendre encore. Cette propriété bien établie des RN n'est toutefois pas quantifiable. Pour être assuré de la tolérance aux pannes d'un réseau il est possible de sur dimensionner les RN utilisés afin de créer des redondances d'information [Haykin '99].

2.2.2. Avantages des RNA pour l'ITGE

En plus de posséder des propriétés très intéressantes pour la commande comme cela a été décrit dans la section précédente, les RNA sont d'excellents candidats à l'ITGE pour les raisons détaillées ci-après.

2.2.2.1. Parallélisme massif

Un RNA est par nature une interconnexion parallèle de neurones et de poids synaptiques. Ce parallélisme peut être mis à profit pour intégrer les RNA dans des structures d'ITGE hautement parallèles et ainsi augmenter la vitesse de calcul de l'ensemble. Notons qu'à ce niveau, une analogie intéressante peut être faîte entre les RNA et les circuits intégrés sur silicium qui réalisent des fonctions très complexes par interconnexion de nombreux éléments non linéaires

(transistors). Ce gain de vitesse permis par le parallélisme offre un autre avantage. Si la vitesse de calcul n'est pas l'élément crucial du circuit désiré, l'accélération des calculs due au parallélisme permet de réduire la vitesse nécessaire des éléments de base du réseau. Ainsi, des éléments plus lents mis en parallèle peuvent exécuter une tâche globale avec une vitesse suffisante. Cette vitesse de traitement plus faible des éléments de base du circuit implique qu'ils pourront très certainement être réalisés avec une consommation de puissance ou une surface d'intégration réduite.

2.2.2.2. Régularité

L'architecture d'un RNA est dite régulière puisqu'elle est constituée d'une interconnexion d'un grand nombre d'éléments de base identiques. Ainsi, la réalisation d'un circuit pour l'ITGE d'un RNA est simplifiée puisqu'il s'agit alors de réaliser un nombre limité de circuits pour les éléments de base (neurones et poids synaptiques) et de les connecter ensuite de façon régulière conformément à la topologie du réseau.

2.2.2.3. Simplicité des opérations mathématiques

À L'exception de la fonction d'activation des neurones qui peut être non linéaire, les opérations mathématiques mises en œuvre dans les RNA sont simples (additions et multiplications). Cette simplicité a une conséquence directe sur l'ITGE puisque des opérations mathématiques plus simples sont généralement plus faciles à intégrer dans le silicium. D'autre part, l'addition et la multiplication sont les opérations de base de nombreux algorithmes de traitement de signal. La littérature concernant l'ITGE de ces opérations est donc très abondante et constitue une base de connaissances intéressante pour le développement de nos circuits.

2.2.2.4. Tolérance aux pannes et aux défauts de circuits

Du point de vue de l'ITGE, la propriété de tolérance aux pannes s'applique encore. La dégradation d'un ou plusieurs circuits de l'ITGE d'un RNA n'empêche pas nécessairement son bon fonctionnement global. Ajoutons aussi que les RNA sont tolérants aux défauts dans les opérations mathématiques de base. Par exemple, comme nous le verrons plus en détails dans le quatrième chapitre, le réseau peut s'accommoder d'une imprécision sur la fonction d'activation

des neurones en autant que celle-ci conserve des propriétés conformes aux conditions du théorème d'approximation universel.

2.3. Hypothèses de travail

La première étape dans la conception d'un contrôleur est généralement le développement mathématique d'un modèle du système à contrôler. Ce modèle peut être établi en écrivant d'une manière analytique les équations physiques relatives aux éléments constituant le système. Malheureusement, il n'est pas toujours possible de procéder ainsi et dans certains cas le modèle reste partiellement ou totalement inconnu. Les RNA permettent de contourner ce problème en réalisant une approximation de type boîte noire du système. Cependant, même dans ce cas, il faut établir un modèle d'identification du réseau, c'est-à-dire spécifier quelles sont les sorties à identifier et quelles sont les entrées nécessaires au modèle. Une méthode générale et très utilisée est de modéliser le comportement du système par un modèle NARMAX (*angl. Nonlinear Auto Regressive Moving Average with eXogenous inputs*). Citons par exemple la modélisation de moteurs à combustion [John & Matthew '98] ou encore de systèmes cardiovasculaires [Vallverdu *et al.* '91]. Citons également l'utilisation de ce type de modèle dans des articles de référence sur la commande à base de RNA et dont le choix des modèles suit la même approche que celle que nous utiliserons [Cabrera & Narendra '99 ; Narendra & Parthasarathy '90 ; Levin & Narendra '96 ; Mistry *et al.* '96]. Dans cette thèse, nous limitons l'étude aux systèmes à une entrée et une sortie, systèmes communément appelés SISO (*angl. Single Input Single Output*). Toutefois, ceci n'a rien de restrictif pour les systèmes à plusieurs entrées et plusieurs sorties, appelés MIMO (*angl. Multiple Input Multiple Output*), pour lesquels les différentes équations présentées dans cette thèse peuvent être adaptées moyennant des développements mathématiques analogues.

Le modèle NARMAX d'un système causal pour le cas à une entrée et une sortie s'écrit :

$$y_{(n+1)} = f\left[u_{(n)}, u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)}\right] \tag{2.3}$$

où y est la sortie du système, u son entrée (commande), f est une fonction non linéaire modélisant le comportement non linéaire du système, a et b désignent les nombres de délais sur l'entrée et la sortie respectivement, et n est le temps discrétisé. Dans ce modèle, la non linéarité du système est

modélisée par la fonction f et la dynamique du système par les différents délais appliqués sur l'entrée et sur la sortie en rétroaction. Nous supposons pour la suite que le modèle du système commandé peut être décrit par l'équation (2.3) et que $f : \mathbb{R}^a \times \mathbb{R}^b \rightarrow \mathbb{R}$ est une fonction non linéaire continue.

L'objectif de commande est de réaliser un suivi exact de trajectoire après convergence de la loi de commande. Soit $\left\{y^*_{(n)}\right\}$ un ensemble continu de valeurs réelles, $y^*_{(n)}$ définit la valeur désirée de y à l'instant n et $\left\{y^*_{(n)}\right\}$ la trajectoire désirée pour y. L'ensemble des valeurs de n est noté $\mathbb{N}_n$ où $\mathbb{N}_n$ est $\mathbb{N}$ ou un sous-ensemble contigu de $\mathbb{N}$. Idéalement, nous souhaitons que les trajectoires réelles et désirées soient égales en tout temps après convergence de la loi de commande, ce qui serait un suivi de trajectoire parfait. Cet objectif de commande peut s'écrire sous la forme :

$$y_{(n)} = y^*_{(n)} \qquad \forall n > n_{cv} \tag{2.4}$$

où n_{cv} désigne l'instant à partir duquel la loi de commande est considérée avoir convergé.

Si le suivi exact de trajectoire ne peut être atteint, un objectif plus réaliste s'écrit :

$$\left|y_{(n)} - y^*_{(n)}\right| < \varepsilon \qquad \forall n > n_{cv} \tag{2.5}$$

où ε est une constante positive qui représente l'écart maximal toléré entre la trajectoire et la trajectoire désirée après convergence de la loi de commande.

En utilisant l'équation (2.3) et l'équation (2.4) réécrite à l'instant n+1, l'objectif de commande devient :

$$y^*_{(n+1)} = f\left[u_{(n)}, u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)}\right] \tag{2.6}$$

De cette expression nous pouvons déduire le modèle de la loi de commande :

$$u_{(n)} = g\left[u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)}, y^*_{(n+1)}\right] \tag{2.7}$$

où $g : \mathbb{R}^{a-1} \times \mathbb{R}^b \times \mathbb{R} \rightarrow \mathbb{R}$ est une fonction non linéaire réalisant une pseudo inversion de f.

2.4. Schéma de commande

Il existe à la fois de nombreuses topologies de RNA [Haykin '99] et de nombreuses structures de commande à base de RNA [Dash *et al.* '97 ; Narendra & Parthasarathy '90].

2.4.1. Choix d'une topologie de RNA

Le choix de la topologie de réseau doit d'abord être fait en fonction du problème algorithmique traité. Ensuite, en vue de satisfaire aussi les critères d'ITGE, les modèles neuronaux choisis doivent pouvoir être intégrés facilement dans le silicium, tant du point de vue de la topologie que du point de vue des opérations mathématiques mises en œuvre. Deux topologies de RNA sont principalement utilisées en commande : (i) les réseaux de type perceptron multicouche ; (ii) les réseaux récurrents.

Les réseaux de type perceptron multicouche sont utilisés pour leur capacité générale de modélisation de fonctions ou de systèmes non linéaires, conformément au théorème d'approximation universel décrit dans la section 2.2.1.1. Étant donné que les fonctions f et g décrivant les modèles du système et du contrôleur (équations (2.3) et (2.7)) sont, par hypothèse, non linéaires, le choix d'une telle topologie est tout à fait approprié.

D'un autre côté, les réseaux récurrents sont bien adaptés au traitement de systèmes dynamiques grâce à leur propriété de mémoire associative [Narendra & Parthasarathy '90]. D'ailleurs, en observant le modèle du contrôleur décrit par l'équation (2.7), nous remarquons que ce modèle est lui-même récurrent puisque le calcul de $u_{(n)}$ nécessite les valeurs passées de $u_{(n)}$. Le choix d'une topologie récurrente pour le contrôleur est donc approprié. Cependant, la récurrence du réseau ne permet pas de tenir compte de tous les termes dynamiques. En effet, dans le modèle (2.7), $y_{(n)}$ intervient ainsi que des valeurs passées de $y_{(n)}$. Il en est de même pour le modèle du système décrit par l'équation (2.3). Pour tenir compte de cet effet de mémoire dans nos réseaux, deux approches sont considérées. La première consiste à développer des modèles dynamiques pour les neurones eux-mêmes en ajoutant des connexions récurrentes dans le modèle interne du neurone [Yu & Gupta '92 ; Rao & Gupta '93 ; Sastry *et al.* '94]. Ceci peut par exemple être réalisé en définissant le modèle du neurone par une représentation d'état. Le nombre d'entrées du réseau est alors grandement diminué puisque seulement un échantillon du signal est nécessaire à

un instant donné. Cependant, l'ajout des récurrences dans le modèle du neurone augmente les risques d'instabilité et les équations d'adaptation sont plus difficiles à développer. La deuxième approche, plus simple, consiste à fournir à l'entrée du réseau autant de valeurs passées d'un signal qu'il est nécessaire pour la modélisation. Ceci peut être réalisé en appliquant à un échantillon du signal un certain nombre de délais comme ce sera présenté plus loin. Le type de réseau obtenu est communément appelé TDLNN (angl. *Tapped Delay Line Neural Network*). Cette approche constitue une façon simple d'incorporer les termes dynamiques dans le réseau. Un autre avantage important, comme nous le verrons dans le cas du réseau N_I, est que si la sortie elle-même n'est pas connectée en rétroaction vers l'entrée du réseau, la topologie obtenue n'est pas récurrente ce qui diminue les risques d'instabilité. Nous avons donc retenu cette approche pour les réseaux qui seront présentés par la suite.

2.4.2. Structure de commande indirecte inverse à base de RNA

2.4.2.1. Structure générale

Après avoir analysé les topologies de réseaux appropriées au problème traité dans cette thèse, il faut maintenant choisir une structure de commande à base de RNA, c'est-à-dire définir le nombre de réseaux utilisés, leur rôle, la façon de les connecter, etc. Comme nous l'avons déjà évoqué dans le premier chapitre, la littérature concernant l'application des RNA en commande est très vaste. Afin de limiter les possibilités de choix dans les structures, nous rappelons que le modèle du système est supposé partiellement ou entièrement inconnu. L'utilisation des réseaux de neurones est justifiée par cette absence de modèle comme nous l'avions mentionné dans la section 2.1. Dans ce cas, deux approches prédominent. Dans l'approche directe, un seul réseau est utilisé et contrôle le système. Ce réseau modélise directement le modèle inverse du système, c'est-à-dire la fonction g. L'approche indirecte quant à elle utilise deux réseaux, le premier sert à modéliser le système à commander et le deuxième est le contrôleur basé sur le modèle identifié. Le premier réseau est alors utilisé comme modèle mathématique du système et permet de déduire de façon analytique la loi d'adaptation du contrôleur. L'approche directe comporte les inconvénients suivants [Fukuda & Shibata '92 ; Behera *et al.* '96] :

- les algorithmes d'inversion directe sont plus sensibles aux erreurs de modélisation et requièrent généralement un traitement mathématique plus compliqué,

- si le modèle du contrôleur est variable, il est beaucoup plus difficile d'adapter le réseau à ces variations et la convergence est beaucoup plus lente,
- les variables du modèle telles que les dérivées de la sortie du système par rapport aux entrées ne sont pas accessibles,
- l'apprentissage par rétro propagation ne peut pas être utilisé directement puisque aucune information reliant directement l'entrée du système à sa sortie n'est disponible.

Nous avons donc choisi de retenir l'approche indirecte dans laquelle le modèle neuronal servira de base de connaissance du modèle du système pour le développement de la loi de commande. Nous partons donc de la structure générale de commande indirecte inverse présentée dans la figure 2.1 pour le développement de nos algorithmes [Fukuda & Shibata '92 ; Lesueur *et al.* '01]. Cette structure est connue dans la littérature sous le nom de commande indirecte inverse à base de RNA (*angl. indirect inverse control*). Un RNA (à droite sur la figure 2.1) sert à identifier ou estimer le modèle direct du système, c'est-à-dire la fonction f, tandis que l'autre RNA est le contrôleur du système ou l'estimé du modèle pseudo inverse du système, c'est-à-dire la fonction g. L'écart mesuré entre la sortie réelle et la sortie désirée définit l'erreur de trajectoire. Cette erreur est rétro propagée à travers le modèle neuronal direct pour déduire l'erreur commise sur la commande et adapter le contrôleur en conséquence, tel qu'illustré dans la figure 2.1. Cette approche peut être comparée à une méthode employée en commande et connue sous le nom de *backstepping* (pas vers l'arrière). Par la suite, nous présentons les équations de propagation directe des deux réseaux, c'est-à-dire les équations gouvernant la propagation des entrées des deux réseaux vers leurs sorties. Les équations mathématiques reliées à l'adaptation des réseaux propagent les signaux d'erreur dans le sens opposé à la propagation d'où le nom de rétro propagation. Ces équations feront l'objet du troisième chapitre.

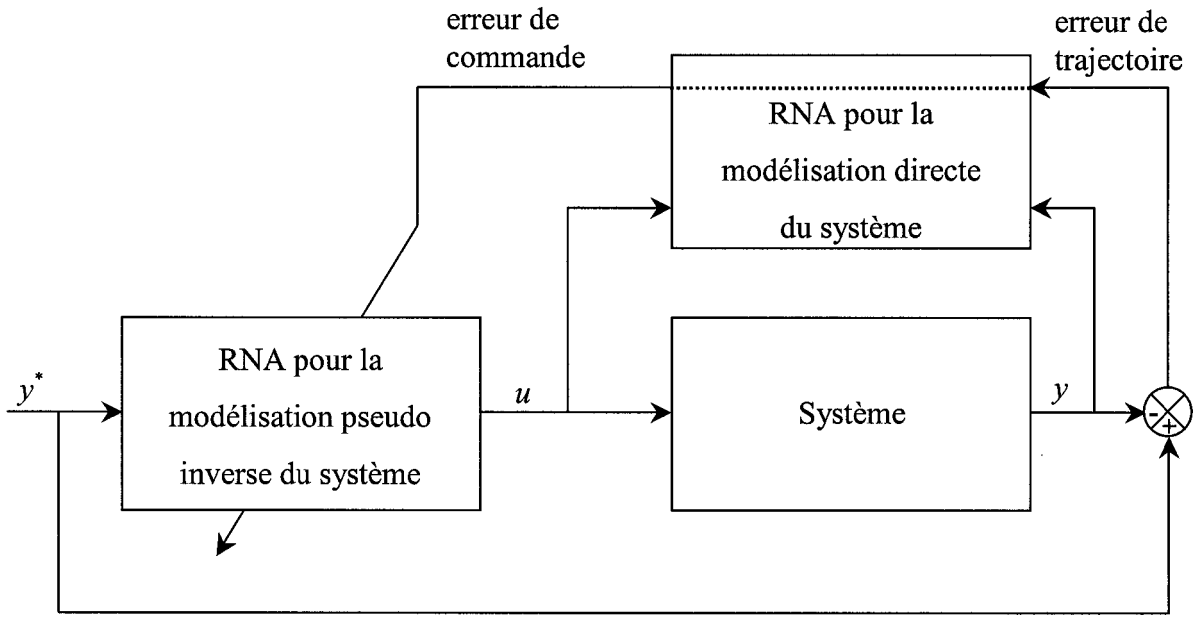


Figure 2.1. Structure générale de commande indirecte inverse à base de RNA

2.4.2.2. Modélisation directe

La modélisation directe vise à trouver un RNA capable d'identifier ou d'estimer le modèle direct du système. Pour cela, nous rappelons que, par hypothèse, le modèle du système peut être décrit par l'équation (2.3). Afin de bénéficier du théorème d'approximation universel des RNA, nous utilisons un RNA de type perceptron à une couche cachée. En supposant que $y_{(n)}$ et $u_{(n)}$ sont accessibles, une façon simple de réaliser le modèle souhaité est de choisir comme entrées du RNA les mêmes entrées que celles de la fonction f dans l'équation (2.3) ce qui correspond au type de réseau TDLNN mentionné dans la section 2.4.1 de ce chapitre.

Le réseau de modélisation directe, qui sera noté N_I, est détaillé dans la figure 2.2 dans le cas à une couche cachée. Il s'agit d'un perceptron multicouche. La première couche de poids synaptiques, qui est la couche cachée du réseau, est constituée des poids reliant les entrées s_i aux neurones N_i tandis que la deuxième couche est constituée des poids reliant les neurones cachés N_i au neurone de sortie N_s.

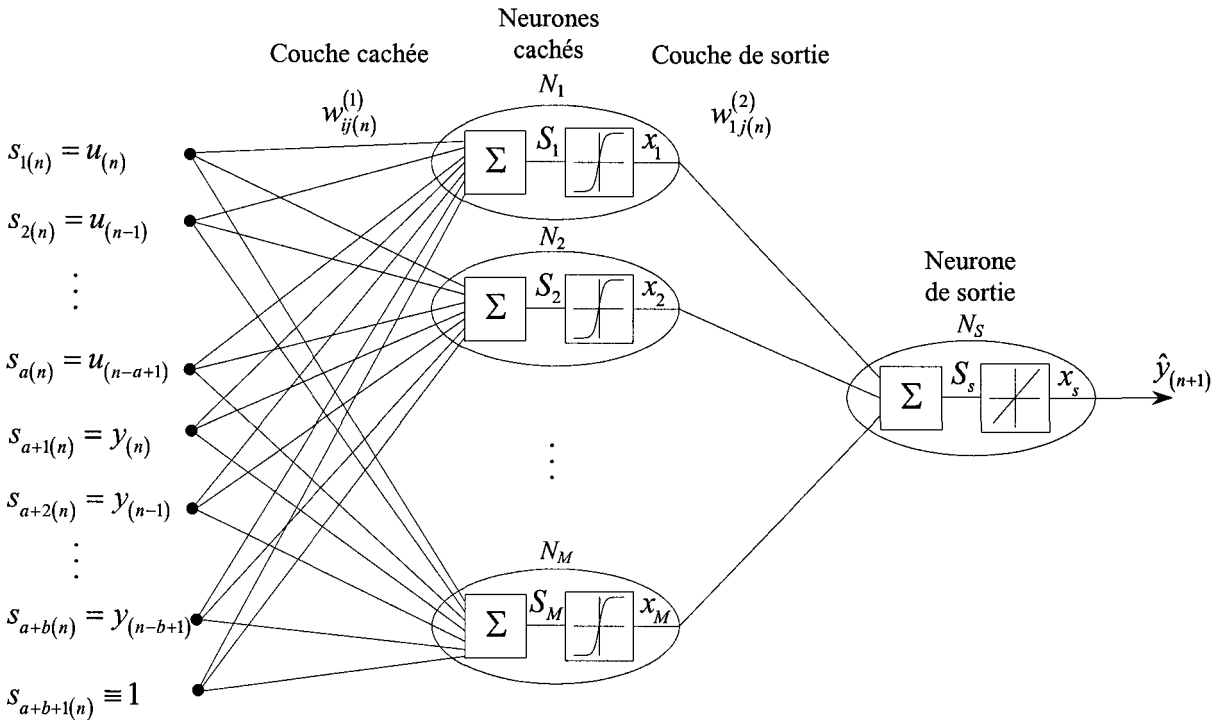


Figure 2.2. Réseau de modélisation directe N_I dans le cas à une couche cachée

Le choix du type de fonction d'activation est important car il a une grande influence sur la capacité de modélisation de fonctions non linéaires des RNA. Afin de modéliser une fonction non linéaire, le réseau doit au moins comporter des éléments non linéaires. Une fonction d'activation non linéaire simple est la fonction de seuil définie par $\varphi(\bullet) = sign(\bullet)$. Cette fonction est très simple tant au niveau de son calcul que de son intégration sur silicium. Cependant, une telle fonction limite grandement la capacité de généralisation d'un RNA et ne permet pas de satisfaire les conditions du théorème d'approximation universel. Afin de pouvoir modéliser une fonction non linéaire, la fonction d'activation doit respecter les conditions de la fonction φ mentionnées dans le théorème d'approximation universel. Le choix de la fonction tangente hyperbolique permet de satisfaire à ces conditions. Notons que ce choix n'est pas restrictif et que c'est l'allure de la fonction qui est importante. Parmi un éventail de fonctions possibles, la fonction tangente hyperbolique offre l'avantage d'avoir une dérivée simple à calculer comme nous le verrons dans le quatrième chapitre ainsi que d'avoir une sortie bipolaire.

Les neurones de la première couche ont une fonction d'activation tangente hyperbolique. Le neurone de sortie a une fonction d'activation linéaire et de gain unitaire. Dans ce réseau, s_i est la $i^{\text{ième}}$ entrée ; $w_{ij}^{(1)}$ est le poids synaptique reliant la $j^{\text{ième}}$ entrée au $i^{\text{ième}}$ neurone caché N_i ; $w_{1j}^{(2)}$ est le poids synaptique reliant le $j^{\text{ième}}$ neurone caché au neurone de sortie N_s ; S_i est la somme interne du neurone N_i ; x_i est la sortie du neurone N_i ; x_s est la sortie du neurone N_s et M est le nombre de neurones sur la couche cachée.

Les biais β_i du théorème d'approximation universel sont inclus sous la forme d'une entrée supplémentaire constante égale à l'unité et connectée aux différents neurones cachés par des poids synaptiques de valeurs ajustables. Le modèle de propagation directe que nous décrirons dans l'équation (2.9) est alors équivalent au modèle de la fonction F du théorème d'approximation universel.

Le modèle d'identification réalisé par le réseau N_I peut s'écrire

$$\hat{y}_{(n+1)} = \hat{f}\left[u_{(n)}, u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)}\right] \tag{2.8}$$

où $\hat{y}$ désigne la valeur estimée de y et $\hat{f}$ la fonction estimée de f .

Si les nombres de délais sur l'entrée et la sortie ne sont pas connus, il faut alors s'assurer de choisir des valeurs de ces paramètres au moins égales aux valeurs réelles. Le réseau associera des poids synaptiques de très faibles valeurs aux entrées supplémentaires qui ne sont pas nécessaires. Notons toutefois qu'il existe des méthodes basées sur les observations expérimentales d'un système permettant de déterminer avec suffisamment de précision les ordres.

L'équation de propagation du réseau N_I est

$$\hat{y}_{(n+1)} = \sum_{i=1}^{M} w_{1i(n)}^{(2)} x_{i(n)} \tag{2.9}$$

où

$$\begin{cases} x_{i(n)} = \varphi\left(S_{i(n)}\right) \\ S_{i(n)} = \sum_{j=1}^{a+b+1} w_{ij(n)}^{(1)} s_{j(n)} \\ \varphi(z) = \tanh(z) \end{cases}$$

Remarque : la fonction d'activation du neurone de sortie pourrait également être de type tangente hyperbolique. Ceci aurait pour avantage principal de limiter la valeur de sortie des réseaux grâce à la saturation de la fonction. Le fait de limiter par exemple la sortie du contrôleur permet de limiter le signal de commande physique que doit générer l'actionneur sur le système et donc de prévenir une usure prématurée voire un bris mécanique. Dans la suite, nous conserverons le choix de la fonction linéaire sans saturation en sortie pour éviter les problèmes d'adaptation de la plage dynamique de sortie des réseaux suivant les cas de simulations, c'est-à-dire le choix de la valeur de saturation de la fonction tangente hyperbolique.

2.4.2.3. Contrôle

Le réseau contrôleur, noté N_C, est également un réseau de type perceptron multicouche pour les raisons évoquées plus tôt. La topologie de ce réseau, détaillée dans la figure 2.3, est semblable à celle de N_I en adaptant les notations utilisées afin de réaliser le modèle du contrôleur décrit par l'équation (2.7). Le modèle d'identification réalisé par le réseau N_C peut s'écrire

$$u_{(n)} = \hat{g}\left[y^{*}_{(n+1)}, u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)} \right] \tag{2.10}$$

Les neurones de la première couche ont une fonction d'activation tangente hyperbolique. Le neurone de sortie a une fonction d'activation linéaire et de gain unitaire. Dans ce réseau, t_i est la $i^{\text{ième}}$ entrée du réseau ; $v_{ij}^{(1)}$ est le poids synaptique reliant la $j^{\text{ième}}$ entrée au $i^{\text{ième}}$ neurone caché N_i ; $v_{1j}^{(2)}$ est le poids synaptique reliant le $j^{\text{ième}}$ neurone caché au neurone de sortie N_s ; $\sum_i$ est la somme interne du neurone N_i ; ζ_i est la sortie du neurone N_i ; ζ_s est la sortie du neurone N_s et N est le nombre de neurones sur la couche cachée. Malgré les grandes similitudes entre les réseaux N_I et N_C, remarquons toutefois une différence importante entre les deux : le réseau N_C est récurrent puisque sa sortie $u_{(n)}$ est connectée à l'entrée. Notons que nous utilisons les mêmes notations pour désigner les neurones des deux réseaux N_I et N_C. Étant donné que ces notations ne sont utilisées que dans les figures 2.2 et 2.3, cela n'amènera pas de confusion par la suite.

L'équation de propagation du réseau N_C est

$$u_{(n)} = \sum_{i=1}^{N} v_{1i(n)}^{(2)} \zeta_{i(n)} \tag{2.11}$$

où

$$\begin{cases} \zeta_{i(n)} = \varphi\left(\Sigma_{i(n)}\right) \\ \Sigma_{i(n)} = \sum_{j=1}^{a+b+1} v_{ij(n)}^{(1)} t_{j(n)} \end{cases} \quad \text{et } \varphi \text{ a la même définition que dans le cas de } N_I.$$

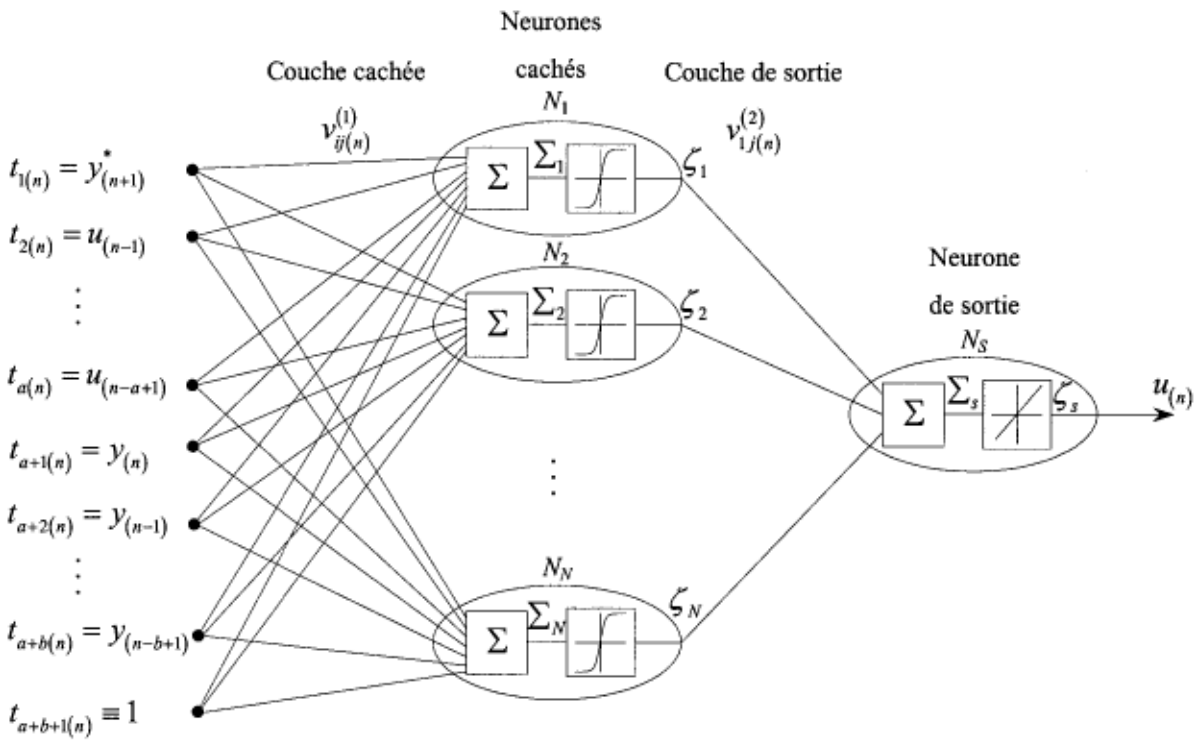


Figure 2.3. Réseau contrôleur N_C dans le cas à une couche cachée

2.4.3. Schéma de commande complet

Le schéma de commande complet incluant les deux réseaux est présenté dans la figure 2.4 et reprend exactement la structure décrite dans [Lesueur *et al.* '01]. Les entrées constantes et égales à l'unité qui représentent les biais ne sont pas indiquées dans cette figure. Le réseau

d'identification N_I utilise les entrées $u_{(n)}$ et $y_{(n)}$ ainsi qu'un certain nombre de valeurs passées de ces signaux pour réaliser le modèle d'identification décrit par l'équation (2.8). L'erreur d'identification, représentant l'erreur commise à l'instant n sur l'estimation de la sortie du système, est obtenue par :

$$e_{I(n)} = y_{(n)} - \hat{y}_{(n)} \tag{2.12}$$

Cette erreur est utilisée dans l'adaptation des poids synaptiques du réseau N_I avec les algorithmes qui seront présentés dans le prochain chapitre. Le but de cette adaptation est l'obtention d'un modèle direct du système suffisamment précis.

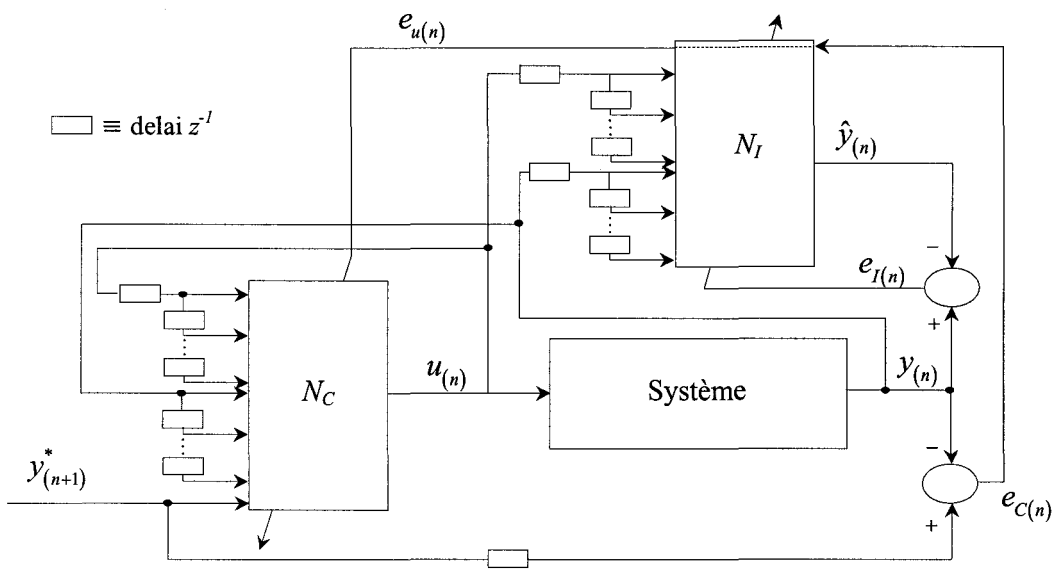


Figure 2.4. Schéma général de commande indirecte inverse avec deux réseaux de neurones

Le réseau contrôleur N_C utilise les entrées $u_{(n-1)}$ et $y_{(n)}$, un certain nombre de valeurs passées de ces signaux, et $y^*_{(n+1)}$ pour réaliser le modèle estimé de la fonction g décrit par l'équation (2.10). L'adaptation du réseau N_C n'est pas aussi directe que celle du réseau N_I. En

effet, la sortie idéale du contrôleur est inconnue. L'erreur commise sur sa sortie ne peut donc pas être directement calculée. Bien sûr, si cette sortie idéale était connue ou facile à obtenir, nous l'appliquerions directement à l'entrée du système. Puisque ce n'est pas le cas, nous procéderons de manière indirecte.

Tout d'abord, nous calculerons l'erreur commise sur la sortie du système par rapport à la sortie de référence. Cette erreur est notée e_C et est calculée à l'instant n par :

$$e_{C(n)} = y^*_{(n)} - y_{(n)} \tag{2.13}$$

Cette erreur représente l'écart entre la sortie obtenue à l'instant n et la valeur désirée de cette sortie au même instant. L'ensemble des valeurs de e_C sur un intervalle de temps donné représente l'erreur de trajectoire sur cet intervalle. Ce signal nous donne une information sur la qualité de la commande puisque, plus l'erreur de trajectoire est faible, meilleure est la commande.

Le principe est alors d'utiliser le modèle neuronal N_I afin de déduire l'erreur de sortie du contrôleur à partir de l'erreur de trajectoire e_C. En effet, considérons la chaîne directe contenant seulement les deux RNA de la figure 2.4. Dans le cas idéal, N_I est le modèle exact du système commandé et N_C le modèle pseudo inverse exact du système. Alors, la chaîne des deux réseaux réalise la fonction identité conduisant à l'égalité parfaite entre y et y^*. Dans la réalité, les tolérances de précision sur les modèles neuronaux introduisent des erreurs au long de la chaîne. L'erreur de commande, notée e_u est obtenue par :

$$e_{u(n)} = \Re\left\{e_{C(n)}, P_{N_{I(n)}}\right\} \tag{2.14}$$

où $\Re$ est l'opérateur de pseudo inversion approximative de N_I ($\Re$ sera élaboré dans le troisième chapitre) et $P_{N_{I(n)}}$ est l'ensemble ou un sous-ensemble des paramètres du réseau N_I.

L'ensemble $P_{N_{I(n)}}$ est défini par :

$$P_{N_{I(n)}} = \left\{w^{(1)}_{ij(n)}, w^{(2)}_{1i(n)}, s_{j(n)}\right\} \qquad i \in \{1,...,M\} \text{ et } j \in \{1,...,a+b+1\} \tag{2.15}$$

Une fois l'erreur e_u obtenue, les poids synaptiques du réseau N_C peuvent être adaptés par un algorithme d'adaptation du même type que celui utilisé pour N_I.

Les différentes étapes de calcul pour chaque période d'échantillonnage sont :

1. mesurer la sortie du système $y_{(n)}$,
2. calculer les erreurs d'identification et de trajectoire e_I et e_C,
3. calculer l'erreur de commande e_u,
4. adapter N_I selon e_I et N_C selon e_u,
5. échantillonner de nouvelles entrées pour les deux réseaux,
6. propager les entrées pour obtenir les nouvelles sorties des deux réseaux,
7. revenir à l'étape 1 pour le prochain échantillon.

2.5. Conditions d'existence de la loi de commande

La structure générale de commande indirecte inverse a été présentée dans la figure 2.1. La loi de commande du système peut être vue comme la composition des deux réseaux neuronaux : (i) N_I qui identifie le modèle direct du système et à travers lequel l'erreur de trajectoire est rétro propagée afin de déduire l'erreur de commande e_u ; (ii) N_C qui, en fonction de l'erreur e_u, fournit la commande au système. Il est alors nécessaire de se demander sous quelles conditions cette loi de commande existe. À l'image de la structure de commande, cette question peut être décomposée en deux sous-questions :

- quelles sont les conditions requises pour l'existence du réseau d'identification N_I ?
- quelles sont les conditions requises pour l'existence du réseau contrôleur N_C ?

2.5.1. Conditions d'existence du réseau N_I

La réponse à la première question est dictée par le théorème d'approximation universel énoncé dans la section 2.2.1.1. Notons d'abord que par hypothèse, le modèle du système peut être décrit par le modèle NARMAX conformément à l'équation (2.3). En adaptant le théorème d'approximation universel à notre réseau N_I, la réponse à la question d'existence de ce réseau

revient à la question d'approximation de la fonction f par la fonction $\hat{f}$ du réseau. En suivant le théorème d'approximation universel, les conditions d'existence de N_I s'énoncent comme suit :

Condition 1 : le modèle du système peut être décrit par un modèle NARMAX conformément à l'équation (2.3).

Condition 2 : le réseau d'approximation N_I a les mêmes entrées que la fonction f, c'est-à-dire que f et $\hat{f}$ ont le même espace d'entrée $\mathbf{E}_I = \left\{u_{(n)}, u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)}\right\}$.

Condition 3 : $\mathbf{E}_I \subset I_{a+b}$ où $I_{a+b} = [-1,1]^{a+b}$.

Condition 4 : $f \in C(I_{a+b})$ où $C(I_{a+b})$ est l'ensemble des fonctions continues sur I_{a+b}.

Condition 5 : φ est continue, monotone croissante, bornée et non constante sur $I_1 = [-1,1]$.

Dans ces conditions, le théorème d'approximation universel nous garantit qu'il existe des constantes réelles α_i, β_i et w_{ij} pour $i = 1,...m_1$ et $j = 1,...a+b$ telles que nous puissions définir

$$\hat{f}(x_1,...,x_{a+b}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{a+b} w_{ij} x_j + \beta_i\right) \tag{2.16}$$

comme approximation de la fonction f, c'est-à-dire

$$\left|\hat{f}(x_1,...,x_{a+b}) - f(x_1,...,x_{a+b})\right| < \varepsilon \qquad \forall \varepsilon > 0$$

pour tout $(x_1,...,x_{a+b})$ dans l'espace d'entrée de la fonction f.

Il reste à identifier les paramètres α_i, β_i et w_{ij} en les identifiant aux paramètres du réseau N_I :

$$\alpha_i = w_{1i}^{(2)} \qquad \text{pour } i = 1,...m_1 \text{ et } m_1 = M \tag{2.17}$$

$$\beta_i = w_{i\ a+b+1}^{(1)} \tag{2.18}$$

$$w_{ij} = w_{ij}^{(1)} \qquad \text{pour } i = 1,...m_1 \text{ et } j = 1,...a+b \tag{2.19}$$

2.5.2. Conditions d'existence du réseau N_C

La question de l'existence du réseau N_C peut être divisée en deux sous-questions :

- quelles sont les conditions d'existence de la fonction g ?

- quelles sont les conditions d'existence du réseau N_C en supposant que les conditions d'existence de la fonction g sont satisfaites ?

En supposant tout d'abord que la fonction g existe, la réponse à la deuxième sous-question est immédiate et suit exactement le même raisonnement dans le cas du réseau N_I. Les conditions s'écrivent donc :

Condition 1 : le modèle du contrôleur (réseau N_C) peut être décrit par un modèle NARMAX conformément à l'équation (2.7).

Condition 2 : le réseau d'approximation N_C a les mêmes entrées que la fonction g, c'est-à-dire que g et $\hat{g}$ ont le même espace d'entrée $\mathbf{E}_C = \left\{u_{(n-1)}, \cdots, u_{(n-a+1)}, y_{(n)}, y_{(n-1)}, \cdots, y_{(n-b+1)}, y^*_{(n+1)}\right\}$.

Condition 3 : $\mathbf{E}_C \subset I_{a+b}$ où $I_{a+b} = [-1,1]^{a+b}$.

Condition 4 : $g \in C(I_{a+b})$ où $C(I_{a+b})$ est l'ensemble des fonctions continues sur I_{a+b}.

Condition 5 : φ est continue, monotone croissante, bornée et non constante sur $I_1 = [-1,1]$.

Dans ces conditions, le théorème d'approximation universel nous garantit qu'il existe des constantes réelles α_i, β_i et w_{ij} pour $i = 1,...m_1$ et $j = 1,...a+b$ telles que nous puissions définir

$$\hat{g}(x_1,...,x_{a+b}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{a+b} w_{ij} x_j + \beta_i\right) \tag{2.20}$$

comme approximation de la fonction g, c'est-à-dire

$$\left|\hat{g}(x_1,...,x_{a+b}) - g(x_1,...,x_{a+b})\right| < \varepsilon \qquad \forall \varepsilon > 0$$

pour tout $(x_1,...,x_{a+b})$ dans l'espace d'entrée de la fonction g.

Il reste à identifier les paramètres α_i, β_i et w_{ij} en les identifiant aux paramètres du réseau N_C :

$$\alpha_i = v_{1i}^{(2)} \qquad \text{pour } i = 1,...m_1 \text{ et } m_1 = N \tag{2.21}$$

$$\beta_i = v_{i\,a+b+1}^{(1)} \tag{2.22}$$

$$w_{ij} = v_{ij}^{(1)} \qquad \text{pour } i = 1,...m_1 \text{ et } j = 1,...a+b \tag{2.23}$$

La deuxième sous-question est beaucoup plus difficile. Contrairement au cas de N_I pour lequel l'existence de f était une hypothèse de départ sur le système commandé, l'existence de g dépend des caractéristiques de la fonction f. En fait, répondre à la deuxième sous-question revient à trouver les conditions supplémentaires sur f pour que la fonction pseudo inverse de f existe. Ce problème est d'autant plus difficile à résoudre que la fonction f est supposée inconnue et qu'elle peut varier. L'approche quasi systématiquement utilisée est de baser l'existence de la fonction g sur les résultats de nombreuses simulations. Par exemple, si nous supposons que la trajectoire de référence est toujours de type sinusoïdale et que les gammes de fréquences et d'amplitudes sont suffisamment restreintes, il est possible de réaliser de grands nombres de simulations afin de couvrir au mieux l'ensemble des possibilités. C'est la méthode la plus simple mais elle ne garantit pas que la fonction g existe pour tous les cas. Il existe également plusieurs théorèmes d'existence de la fonction inverse d'un système dynamique non linéaire [Sandberg '80 ; Hirschorn '79]. Cette fois encore, le problème demeure de savoir comment mettre en œuvre ces théorèmes en pratique et de satisfaire les critères algorithmiques et d'ITGE. Étant donné qu'aucune contribution n'est visée par la thèse à ce niveau (*cf.* section 1.3 - objectifs de recherche et contributions), nous supposerons dans un premier temps que la fonction g existe et nous vérifierons dans l'étude des cas présentés dans cette thèse que cette hypothèse est vraie.

2.6. Stabilité

Établir les conditions de stabilité pour des structures adaptatives non linéaires est un véritable défi scientifique. Notons cette fois encore qu'aucune contribution scientifique n'est visée par la thèse à ce niveau (*cf.* section 1.3 - objectifs de recherche et contributions). Il y a deux obstacles principaux à l'étude de stabilité de structures de commande adaptatives à base de RNA :

- le caractère dynamique non linéaire des éléments mis en œuvre (système et réseaux) : la recherche dans le domaine de la stabilité de systèmes dynamiques non linéaires est beaucoup moins uniforme et moins développée que celle des systèmes dynamiques linéaires, l'approche demeurant généralement au cas par cas,
- l'adaptation en ligne des paramètres des réseaux oblige à incorporer dans l'étude de stabilité les algorithmes d'adaptation.

À la connaissance de l'auteur, aucun article de la littérature ne présente de résultats directement applicables pour déterminer les conditions de stabilité d'une structure telle que celle utilisée dans la thèse. Le but de cette section est de présenter une brève revue de littérature de l'état de la recherche sur ce problème. Les articles de la littérature font état de deux familles de méthodes : (i) les méthodes basées sur la théorie de Lyapunov ; (ii) les méthodes basées sur la théorie NLq. Plus particulièrement, quatre articles ont retenu notre attention pour leur plus grande pertinence en rapport avec la structure de commande utilisée dans cette thèse.

Dans [Nikravesh '96], le modèle du système à contrôler est supposé de la forme d'un modèle d'état. La condition de stabilité au sens de Lyapunov est garantie si les amplitudes des valeurs propres de la matrice Jacobienne du vecteur d'état demeurent à l'intérieur du cercle unitaire. Cette condition est écrite dans une structure d'inversion directe à base de RNA. Les calculs aboutissent à un ensemble d'équations dont les variables sont les valeurs propres de la matrice Jacobienne, valeurs propres directement reliées aux paramètres du réseau. Les auteurs ne précisent pas ensuite comment mettre en application cette condition.

Dans [Man *et al.* '99], la stabilité asymptotique est démontrée pour un réseau non récurrent (même type que le réseau N_I). Une fonction de Lyapunov de l'erreur est rétro propagée dans le réseau pour déterminer une loi d'adaptation stable des poids. L'application est limitée aux réseaux non récurrents.

Dans [Yerramalla *et al.* '03], une condition de stabilité basée sur la deuxième méthode de Lyapunov est décrite pour un réseau auto-organisant. La preuve est donnée que le réseau est auto-stabilisant. Les auteurs proposent l'idée de développer cette approche pour une utilisation en ligne mais aucun développement n'est proposé à ce sujet.

Dans [Suykens & Vandewalle '99], la théorie NLq est utilisée pour démontrer la stabilité absolue d'un réseau récurrent multicouche à temps continu. La condition de stabilité s'écrit sous la forme d'une inégalité matricielle, les matrices étant dépendantes des paramètres du réseau et de l'algorithme d'adaptation. Cette fois encore, l'utilisation de la condition est proposée pour des structures de commande utilisant des réseaux récurrents mais les auteurs ne précisent pas comment la mettre en application.

L'adaptation en ligne est vraiment l'obstacle majeur dans l'étude de stabilité de structures à bases de RNA. Dans une structure telle que celle utilisée dans la thèse, il est peu probable de pouvoir développer des algorithmes intrinsèquement stables. Comme cela est mentionné dans [Suykens & Vandewalle '99 ; Yerramalla *et al.* '03], la solution consiste plutôt à écrire les conditions de stabilité sous forme d'équations et d'utiliser ensuite un observateur de stabilité en ligne avec la loi de commande. Un premier problème à résoudre est alors de savoir quoi faire lorsque la condition de stabilité n'est plus satisfaite. Le deuxième problème important est relié à l'ITGE de l'ensemble. Si la stabilité de la loi de commande nécessite l'utilisation d'un observateur de stabilité, celui-ci doit également être inclus dans l'ITGE et par conséquent dans toute l'étude reliée à l'ITGE. Conformément à la méthodologie utilisée dans cette thèse, il faudrait alors, comme dans le cas des algorithmes d'adaptation des réseaux, développer la condition de stabilité conjointement avec son ITGE. Il pourrait en effet se présenter le cas où une loi de commande satisfaisante en termes algorithmique et d'ITGE ne le serait plus avec l'ajout d'un observateur de stabilité trop complexe.

2.7. Conclusion

Dans ce chapitre, nous avons présenté la structure globale à deux RNA de la loi de commande adaptative indirecte inverse. Les topologies et les équations de propagation régissant les deux réseaux ont également été présentées.

Afin de faciliter la compréhension des notations pour la suite des développements algorithmiques, le tableau 2.1 présente les correspondances de notation des différentes variables des deux réseaux N_I et N_C.

Tableau 2.1. Tableau des correspondances des variables entre les réseaux N_I et N_C

Variables/Réseau	N_I	N_C
Domaines pour i et j	$i \in \{1,...,M\}$; $j \in \{1,...,a+b+1\}$	$i \in \{1,...,N\}$; $j \in \{1,...,a+b+1\}$
Entrées	s_j	t_j
Sortie	$\hat{y}_{(n+1)}$	$u_{(n)}$
Poids couche cachée	$w_{ij}^{(1)}$	$v_{ij}^{(1)}$
Poids couche de sortie	$w_{1i}^{(2)}$	$v_{1i}^{(2)}$
Nombre de neurones cachés	M	N
Somme interne du neurone i	S_i	Σ_i
Sortie du neurone caché i	x_i	ζ_i
Erreur sur la sortie	e_I	e_u

Chapitre 3

Proposition d'algorithmes d'adaptation

Dans ce chapitre, nous détaillerons la problématique de développement d'algorithmes d'adaptation pour la loi de commande à base de RNA décrite dans le deuxième chapitre, plus particulièrement pour le type de topologie de RNA employé. La résolution de cette problématique mènera au développement de plusieurs algorithmes d'adaptation basés sur les principes de rétro propagation statique et dynamique. Notamment, deux nouveaux algorithmes apportés par la thèse et basés sur le principe de rétro propagation dynamique seront présentés. Finalement, les résultats de simulations présentés dans ce chapitre permettront, toujours selon la méthodologie de recherche présentée dans le premier chapitre, de déterminer l'algorithme qui sera retenu pour les étapes suivantes.

3.1. Introduction

Après avoir déterminé le schéma de commande et la topologie des réseaux mis en jeu, il faut choisir ou développer une méthode permettant d'adapter les paramètres des réseaux, c'est-à-dire les poids synaptiques.

L'adaptation des poids des réseaux est nécessaire pour les raisons suivantes :

- le comportement (modèle) du système et du contrôleur, et par conséquent les valeurs optimales des poids synaptiques pour leur modélisation, sont *a priori* inconnus à l'état initial,
- le comportement du système à contrôler peut varier dans le temps ou sous certaines grandeurs d'influence telles que la température (variations de paramètres). Les valeurs optimales des poids synaptiques pour la modélisation ne sont donc pas nécessairement constantes et peuvent elles-mêmes varier dans les mêmes conditions,
- le modèle pseudo inverse du système (contrôleur) peut dépendre de la trajectoire désirée spécifiée si le réseau contrôleur ne parvient pas à réaliser une approximation globale du modèle de la fonction g sur l'ensemble des trajectoires possibles. Dans ce cas, tout changement de trajectoire désirée peut entraîner de nouvelles valeurs optimales des poids du réseau contrôleur.

Rappelons tout d'abord que d'après la méthodologie de recherche présentée dans le premier chapitre, les algorithmes d'adaptation doivent être choisis en vue de satisfaire simultanément les critères algorithmiques et les critères d'ITGE. De nombreuses méthodes et algorithmes existent pour l'adaptation des RNA et la littérature dans ce domaine est très vaste [Haykin '99 ; Narendra & Parthasarathy '90 ; Cichocki & Unbehauen '93]. La capacité d'adaptation d'un RNA est de première importance puisque c'est cette propriété qui lui permet d'apprendre de son environnement et d'améliorer ses performances en fonction de cet apprentissage. La notion d'apprentissage est différente selon le domaine scientifique. Dans le domaine des RNA, nous utilisons une définition adaptée de [Mendel & McLaren '70] et qui définit l'apprentissage comme suit :

« *L'apprentissage est le procédé par lequel les paramètres libres d'un réseau de neurones sont adaptés d'après la stimulation de l'environnement du réseau. Le type d'apprentissage est déterminé par la manière précise dont les paramètres sont adaptés.* »

Dans la suite, nous utiliserons les termes « méthode » et « algorithme » pour deux significations différentes. Le terme « algorithme » désignera une séquence précise d'équations mathématiques

pour l'adaptation des poids tandis que le terme « méthode » désignera un principe, une idée ou une équation de base donnant naissance à un algorithme.

Il existe principalement cinq grandes familles de méthodes d'apprentissage pour les RNA : apprentissage à correction d'erreur, apprentissage basé sur la mémoire, apprentissage de Hebb, apprentissage compétitif et apprentissage de Boltzmann. Étant donné que nous avons statué sur l'utilisation de réseaux de type perceptron multicouche et que nous souhaitons faire un apprentissage en ligne, une méthode à correction d'erreur est plus appropriée et est retenue pour la suite [Haykin '99].

Pour illustrer cette méthode, considérons un neurone avec ses entrées $x_{k(n)}$ et sa sortie $y_{(n)}$. La sortie du neurone est comparée à la sortie de référence $y^*_{(n)}$, l'erreur $e_{(n)}$ est alors calculée. Ce signal d'erreur est utilisé dans une séquence d'ajustements réalisés sur les poids synaptiques connectés aux neurones. Le but de ces ajustements est de rendre le signal y plus proche de y^* dans les prochains instants, c'est-à-dire de diminuer l'erreur e. Au sens du filtrage optimal, il s'agit d'optimiser un critère (fonction de coût), généralement relié à l'erreur e. Il s'agit le plus souvent de la fonction d'énergie $\xi_{t(n)}$ aussi appelée erreur quadratique totale et définie par :

$$\xi_{t(n)} = \frac{1}{2}\sum_{i=0}^{n} e^2_{(i)} = \frac{1}{2}\sum_{i=0}^{n}\left(y^*_{(i)} - y_{(i)}\right)^2 \tag{3.1}$$

Cette fonction d'énergie est une fonction de tous les paramètres libres du réseau, c'est-à-dire des poids synaptiques et des biais. Afin de simplifier les algorithmes pour le traitement en temps réel et de limiter l'espace mémoire nécessaire, l'erreur quadratique instantanée est plus souvent utilisée :

$$\xi_{(n)} = \frac{1}{2}e^2_{(n)} = \frac{1}{2}\left(y^*_{(n)} - y_{(n)}\right)^2 \tag{3.2}$$

C'est cette fonction de coût qui sera utilisée pour la dérivation des équations de rétro propagation. Notons que cette méthode suppose la connaissance d'un signal de référence pour la sortie du neurone. Si ce signal n'est pas disponible, il faut trouver un moyen de le produire en fonction des informations disponibles du réseau.

3.2. Choix d'un algorithme d'adaptation

Il existe plusieurs algorithmes d'adaptation basés sur la méthode d'apprentissage à correction d'erreur. Cherchant toujours à satisfaire simultanément aux critères algorithmiques et d'ITGE, l'algorithme d'adaptation devrait au moins respecter les conditions suivantes :

- réaliser une adaptation des poids synaptiques permettant d'obtenir des modèles d'identification et de contrôle suffisamment précis,
- avoir une vitesse de convergence convenable relativement aux constantes de temps des systèmes commandés,
- répondre à une vaste gamme d'application,
- avoir une complexité de calcul réduite,
- pouvoir être parallélisé,
- pouvoir être intégré de façon simple dans le silicium.

Le choix de l'algorithme d'adaptation peut être très dépendant du type de mise en œuvre ou d'application visé. Différents algorithmes opèrent sur des topologies de RNA différentes en utilisant différentes techniques d'optimisation et dans des buts différents. De plus, les développements dans ce domaine continuent et de nouveaux algorithmes voient le jour régulièrement. Notons que nous avons volontairement omis les algorithmes non supervisés étant donné que ces algorithmes ne sont, à la connaissance de l'auteur de cette thèse, pas ou très peu utilisés pour le type d'application visé.

L'algorithme qui est au centre de la plupart des développements dans le domaine des RNA est l'algorithme de RP du gradient. Nous avons retenu cet algorithme pour les raisons suivantes :

- il répond à un grand nombre d'applications en commande,
- il comprend de manière intrinsèque un haut degré de parallélisme,
- il est relativement simple à intégrer dans le silicium.

Les inconvénients propres à cet algorithme sont principalement :

- sa vitesse de convergence réduite due à l'utilisation du principe du gradient,
- sa tendance à rester bloqué dans un minimum local de la fonction de coût,
- la nécessité de calculer les dérivées des fonctions d'activation des neurones.

Dans les sections suivantes, nous présentons l'algorithme de rétro propagation du gradient (*angl. Back Propagation* - BP) et discutons des différentes variantes de cet algorithme.

3.2.1. Rétro propagation du gradient de l'erreur

Un algorithme d'adaptation qui s'applique très naturellement aux RNA multicouches est l'algorithme communément appelé rétro propagation du gradient dans lequel les poids synaptiques sont adaptés couche après couche en procédant de la sortie vers l'entrée et en propageant en sens inverse du sens de propagation les gradients d'erreurs calculés dans le réseau [Rumelhart *et al.* '86 ; Haykin '99]. Cet algorithme est une formulation de la méthode d'optimisation du gradient adaptée aux RNA multicouches. Ainsi, l'adaptation d'un poids w connecté à un neurone N_k et dont la sortie est y_k est régie par :

$$\Delta w_{(n)} = w_{(n+1)} - w_{(n)} = -\eta \frac{\partial \xi_{k(n)}}{\partial w_{(n)}} \tag{3.3}$$

où ξ_k est l'erreur quadratique calculée sur la sortie y_k du neurone N_k conformément à l'équation (3.2) et η est le taux d'apprentissage de l'algorithme. Ce dernier, devant être positif et inférieur à 1, définit la vitesse d'adaptation des poids. Plus η est grand plus, pour une même valeur de gradient, la variation $\Delta w_{(n)}$ du poids w à l'instant n est grande. La complexité de calcul de cet algorithme pour un nombre de poids N à adapter est de l'ordre $O(N^2)$.

Bien sûr, il est important de noter que la plupart des défauts associés à l'algorithme de rétro propagation s'appliquent aussi aux autres algorithmes d'adaptation applicables aux RNA multicouches. Un des problèmes les plus importants est sa lenteur de convergence. Ce problème provient de la méthode de gradient descendant sur laquelle l'algorithme de rétro propagation est basé. Pour améliorer la vitesse de convergence, il est possible de choisir une valeur élevée de η, mais ceci a pour conséquence d'augmenter les risques d'instabilité et de limiter la profondeur des minima locaux que l'algorithme peut trouver [Haykin '99].

3.2.2. Les variantes de l'algorithme de rétro propagation du gradient

Plusieurs variantes de l'algorithme de RP ont été proposées afin de contourner ces problèmes associés à cet algorithme. Nous présentons ci-après un bref rappel des variantes les plus pertinentes en vue d'une utilisation en ligne.

3.2.2.1. Méthodes de deuxième ordre

Les méthodes de deuxième ordre sont des modifications de la méthode de gradient. Plusieurs méthodes ont été appliquées aux RNA : les méthodes de gradient conjugué et de quasi-Newton ainsi que l'algorithme de Levenberg-Marquardt [Hagan & Menhaj '94 ; Lehmann '94]. Ces méthodes, proposées afin d'améliorer la vitesse de convergence, peuvent être utilisées pour remplacer la méthode de gradient descendant comme base de développement de l'algorithme de RP. Cependant, les algorithmes de RP basés sur ces méthodes ont une complexité de calcul de $O\left(N^4\right)$, beaucoup plus élevée que dans le cas de l'algorithme de base.

3.2.2.2. Méthode de perturbation des poids

Une autre méthode, connue sous le nom de perturbation des poids est présentée dans [Jabri & Flower '92]. Cette méthode offre surtout l'avantage de limiter la sensibilité de l'algorithme d'adaptation aux défauts des circuits dans les implantations analogiques [Jabri & Flower '92 ; Montalvo *et al.* '97]. Cependant, la complexité de cet algorithme est encore de $O\left(N^4\right)$ et la plupart des opérations pour l'adaptation des poids sont sérielles, limitant la possibilité de paralléliser l'algorithme.

3.2.2.3. Méthode de décroissance des poids

Pour améliorer la capacité de généralisation, l'équation générale d'adaptation (3.3) peut être modifiée selon la méthode de décroissance des poids :

$$w_{(n+1)} = \left(w_{(n)} - \eta \frac{\partial \xi_{k(n)}}{\partial w_{(n)}} \right) \left(1 - \varepsilon_{dec}\right) \tag{3.4}$$

où $0 < \varepsilon_{dec} = 1$ est le paramètre de décroissance des poids [Fernandez-Redondo & Hernandez-Espinosa '00]. Cette méthode offre aussi l'avantage d'éviter les grandes valeurs de poids et donc de réduire la dynamique. Notons que cette variante serait, à partir d'un circuit d'ITGE de l'algorithme de RP de base, très facilement intégrable au niveau de l'ITGE.

3.2.2.4. Momentum

Une autre variante algorithmique est l'ajout d'un terme de *momentum* dans l'équation (3.3) comme décrit par l'équation (3.5).

$$\Delta w_{(n)} = -\eta \frac{\partial \xi_{k(n)}}{\partial w_{(n)}} + \alpha \Delta w_{(n-1)} \tag{3.5}$$

où $0 < \alpha < 1$ est le paramètre de *momentum*. Le terme additionnel de *momentum* a pour but de moyenner les variations aléatoires de poids dans l'algorithme et d'augmenter l'importance des variations consistantes de poids. Même si les améliorations apportées par cette variante peuvent être significatives, le coût au niveau de l'ITGE est important puisqu'il faut ajouter autant de mémoires supplémentaires que de poids dans le réseau. Une étude approfondie de l'effet de l'ajout du *momentum* est présentée dans [Haykin '99].

3.2.2.5. Taux d'adaptation dynamique

Le taux d'adaptation est un paramètre très important de l'algorithme d'adaptation : si sa valeur est trop grande, le gradient aboutit à des oscillations voire à l'instabilité ; si sa valeur est trop petite, le gradient converge très lentement. Le principe de taux d'adaptation dynamique est de modifier le taux pendant l'apprentissage. De nombreuses méthodes peuvent être développées suivant ce principe [Minghu *et al.* '00].

3.2.3. Rétro Propagation Statique (RPS) et Rétro Propagation Dynamique (RPD)

L'algorithme de rétro propagation étant depuis longtemps un élément crucial de développement algorithmique dans le domaine des RNA, son ITGE est devenue un domaine de recherche important.

Malgré les propriétés intéressantes de cet algorithme en vue de l'ITGE et son intérêt reconnu dans le domaine, une autre problématique a vu le jour ces dix dernières années et a été abordée par plusieurs chercheurs.

L'adaptation d'un RNA multicouche par l'algorithme de RP conventionnel ne permet, *a priori*, que l'apprentissage de fonctions ou de modèles statiques [Srinivasan *et al.* '94 ; Baldi '95]. C'est pourquoi certains auteurs ont renommé cet algorithme rétro propagation statique. Dans le cas de systèmes dynamiques, il faut donner une certaine mémoire au réseau, en

introduisant par exemple des délais temporels dans la structure synaptique tel que nous l'avons vu dans la figure 2.4 [Elman '90]. Certains chercheurs se sont alors demandés si l'algorithme de rétro propagation demeurait applicable à de tels réseaux.

Une première approche consiste à considérer toutes les entrées du réseau indépendantes, de sorte que le réseau puisse être considéré comme un réseau multicouche avec entrées indépendantes et que l'algorithme RPS puisse s'appliquer directement. Cependant, quelques chercheurs considèrent que la dépendance dans le temps de certaines entrées est une information temporelle supplémentaire qui peut être prise en compte dans l'algorithme afin d'améliorer les performances des structures à RNA pour des systèmes dynamiques.

Plusieurs méthodes ont alors été proposées afin de mieux prendre en compte la variable temps dans l'adaptation par rétro propagation : (i) rétro propagation à travers le temps [Werbos '90], rétro propagation à travers un modèle adjoint [Srinivasan *et al.* '94] et rétro propagation dynamique [Narendra & Parthasarathy '90 ; Williams & Zipser '89]. Ces différents travaux présentent des méthodes et des algorithmes d'apprentissage basés sur le principe de rétro propagation dynamique, permettant une meilleure prise en compte des termes dynamiques dans l'adaptation [Piché '94]. Malgré l'évidence des gains de performances que peuvent apporter ces algorithmes, la complexité de calcul est généralement plus grande, souvent trop grande pour envisager leur utilisation en temps réel [Lehmann '94].

Étant donné que nous souhaitons améliorer les performances de l'algorithme de rétro propagation de base et que nos RNA intègrent des délais temporels, nous sommes tout de même intéressés par le principe de rétro propagation dynamique. Constatant que les méthodes déjà publiées dans la littérature ne représentent pas de bons compromis entre les exigences des critères algorithmiques et d'ITGE, nous devons dans un premier temps mener un développement algorithmique nous permettant d'atteindre simultanément deux objectifs :

- proposer de nouveaux algorithmes basés sur la RPD permettant une amélioration significative de la qualité de la loi de commande comparativement aux algorithmes basés sur la RPS,
- obtenir un(des) algorithme(s) de plus faible complexité et plus facilement intégrable(s) sur silicium que les algorithmes déjà publiés dans la littérature.

3.3. Algorithmes d'adaptation

3.3.1. Principe général

Dans un premier temps, nous développerons toutes les équations d'adaptation en utilisant le principe de rétro propagation statique. Ceci nous donnera un algorithme de référence que nous noterons RPS. Ensuite, nous utiliserons le principe de rétro propagation dynamique à différents endroits et de différentes façons pour le développement d'autres algorithmes notés RPD1, RPD2, etc. L'application du principe de RPD pourra être faite à trois niveaux différents de la loi de commande :

- dans l'adaptation du réseau N_I,
- dans la rétro propagation à travers le modèle neuronal N_I pour le calcul de e_u,
- dans l'adaptation du réseau N_C.

3.3.2. Rétro Propagation Statique (RPS)

L'algorithme que nous appèlerons RPS correspond au principe de rétro propagation statique (conventionnel). Son utilisation est grandement répandue dans le domaine des RNA. Le terme statique est utilisé par opposition aux méthodes dites dynamiques [Narendra & Parthasarathy '91].

3.3.2.1. Adaptation du réseau N_I

Pour développer les équations d'adaptation du réseau N_I suivant la méthode RPS, nous partons de l'équation de gradient (3.3). Il faut alors évaluer les dérivées de la fonction de coût en fonction des différents poids synaptiques du réseau. Les équations de propagation directe du réseau N_I vues dans le deuxième chapitre peuvent être réécrites sous une forme compacte :

$$\hat{y}_{(n+1)} = \sum_{i=1}^{M} w_{1i(n)}^{(2)} x_{i(n)} = \sum_{i=1}^{M} w_{1i(n)}^{(2)} \tanh\left(\sum_{j=1}^{a+b+1} w_{ij(n)}^{(1)} s_{j(n)} \right) \tag{3.6}$$

Le critère quadratique pour le réseau N_I est défini par :

$$\xi_{I(n+1)} = \frac{1}{2} e^2_{I(n+1)} = \frac{1}{2}\left(y_{(n+1)} - \hat{y}_{(n+1)}\right)^2 \tag{3.7}$$

L'équation de gradient (3.3) adaptée au réseau N_I s'écrit :

$$\Delta w_{(n+1)} = w_{(n+1)} - w_{(n)} = -\eta_I \frac{\partial \xi_{I(n+1)}}{\partial w_{(n)}} \tag{3.8}$$

où η_I est le taux d'apprentissage de N_I et w représente $w_{ij}^{(1)}$ ou $w_{ij}^{(2)}$.

En appliquant la règle de perturbation en chaîne, l'équation (3.8) peut être développée comme suit :

$$\Delta w_{(n+1)} = -\eta_I \frac{\partial \xi_{I(n+1)}}{\partial e_{I(n+1)}} \frac{\partial e_{I(n+1)}}{\partial \hat{y}_{(n+1)}} \frac{\partial \hat{y}_{(n+1)}}{\partial w_{(n)}} \tag{3.9}$$

Les deux premières dérivées dans l'équation (3.9) sont obtenues très simplement en utilisant l'équation (3.7) :

$$\begin{cases} \dfrac{\partial \xi_{I(n+1)}}{\partial e_{I(n+1)}} = e_{I(n+1)} \\ \dfrac{\partial e_{I(n+1)}}{\partial \hat{y}_{(n+1)}} = -1 \end{cases} \tag{3.10}$$

L'équation (3.9) devient donc :

$$\Delta w_{(n+1)} = \eta_I e_{I(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial w_{(n)}} \tag{3.11}$$

Il reste alors à évaluer la dérivée de la sortie du réseau $\hat{y}_{(n+1)}$ par rapport aux différents poids synaptiques, ce calcul sera effectué à partir de l'équation compacte de propagation (3.6).

Nous avons choisi un réseau N_I à deux couches, c'est-à-dire une couche cachée et une couche de sortie. Étant donné que les poids des deux couches n'interviennent pas de la même façon dans l'équation de propagation, nous les distinguons dans le calcul de la dérivée de la fonction de coût.
Pour la couche de sortie :

$$\Delta w^{(2)}_{1j(n+1)} = \eta_I e_{I(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial w^{(2)}_{1j(n)}}$$

$$= \eta_I e_{I(n+1)} \frac{\partial}{\partial w^{(2)}_{1j(n)}} \left(\sum_{k=1}^{M} w^{(2)}_{1k(n)} x_{k(n)} \right)$$

$$= \eta_I e_{I(n+1)} \sum_{k=1}^{M} \frac{\partial}{\partial w^{(2)}_{1j(n)}} \left(w^{(2)}_{1k(n)} x_{k(n)} \right) \qquad j \in \{1,...,M\} \qquad (3.12)$$

Considérant qu'il n'y a pas de dépendance entre $x_{k(n)}$ et $w^{(2)}_{1j(n)}$, et que $\partial w^{(2)}_{1k(n)} / \partial w^{(2)}_{1j(n)}$ est non nul et égal à 1 seulement pour $k = j$, l'équation (3.12) devient

$$\Delta w^{(2)}_{1j(n+1)} = \eta_I e_{I(n+1)} x_{j(n)} \qquad j \in \{1,...,M\} \qquad (3.13)$$

Ceci définit l'équation d'adaptation des poids de la couche de sortie du réseau N_I dans l'algorithme RPS.
Pour la couche cachée :

$$\Delta w^{(1)}_{ij(n+1)} = \eta_I e_{I(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial w^{(1)}_{ij(n)}}$$

$$= \eta_I e_{I(n+1)} \frac{\partial}{\partial w^{(1)}_{ij(n)}} \left(\sum_{k=1}^{M} w^{(2)}_{1k(n)} x_{k(n)} \right)$$

$$= \eta_I e_{I(n+1)} \frac{\partial}{\partial w^{(1)}_{ij(n)}} \left(\sum_{k=1}^{M} \left(w^{(2)}_{1k(n)} \tanh \left(\sum_{l=1}^{a+b+1} w^{(1)}_{kl(n)} s_{l(n)} \right) \right) \right)$$

$$= \eta_I e_{I(n+1)} \sum_{k=1}^{M} \left\{ w^{(2)}_{1k(n)} \frac{\partial}{\partial w^{(1)}_{ij(n)}} \left(\tanh \left(\sum_{l=1}^{a+b+1} w^{(1)}_{kl(n)} s_{l(n)} \right) \right) \right\}$$

$$= \eta_I e_{I(n+1)} \sum_{k=1}^{M} \left\{ w^{(2)}_{1k(n)} \left(1 - \tanh^2 \left(\sum_{l=1}^{a+b+1} w^{(1)}_{kl(n)} s_{l(n)} \right) \right) \frac{\partial}{\partial w^{(1)}_{ij(n)}} \sum_{l=1}^{a+b+1} w^{(1)}_{kl(n)} s_{l(n)} \right\}$$

$$= \eta_I e_{I(n+1)} \sum_{k=1}^{M} \left\{ w_{1k(n)}^{(2)} \left(1 - \tanh^2 \left(\sum_{l=1}^{a+b+1} w_{kl(n)}^{(1)} s_{l(n)} \right) \right) \sum_{l=1}^{a+b+1} \frac{\partial}{\partial w_{ij(n)}^{(1)}} \left(w_{kl(n)}^{(1)} s_{l(n)} \right) \right\} \qquad (3.14)$$

où $i \in \{1,...,M\}$ et $j \in \{1,...,a+b+1\}$.

Considérant qu'il n'y a pas de dépendance entre $s_{l(n)}$ et $w_{ij(n)}^{(1)}$, et que $\partial w_{kl(n)}^{(1)} / \partial w_{ij(n)}^{(1)}$ est non nul et égal à 1 seulement pour $k = i$ et $l = j$, l'équation (3.14) devient :

$$\Delta w_{ij(n+1)}^{(1)} = \eta_I e_{I(n+1)} w_{1i(n)}^{(2)} \left(1 - x_{i(n)}^2 \right) s_{j(n)} \qquad i \in \{1,...,M\}, j \in \{1,...,a+b+1\} \qquad (3.15)$$

Ceci définit l'équation d'adaptation des poids de la couche cachée du réseau N_I dans l'algorithme RPS. Notons que la fonction tangente hyperbolique de l'équation (3.14) a été remplacée par $x_{i(n)}$ conformément à l'équation de sortie du $i^{\text{ème}}$ neurone caché.

Finalement, l'adaptation des poids du réseau N_I selon la méthode RPS est complètement définie par les équations (3.13) et (3.15).

3.3.2.2. Adaptation du réseau N_C

Les équations d'adaptation du réseau N_C selon la méthode RPS peuvent être obtenues directement des équations développées pour le réseau N_I en utilisant les correspondances de notations du tableau 2.1.

$$\Delta v_{1j(n+1)}^{(2)} = \eta_C e_{u(n+1)} \zeta_{j(n)} \qquad j \in \{1,...,N\} \qquad (3.16)$$

$$\Delta v_{ij(n+1)}^{(1)} = \eta_C e_{u(n+1)} v_{1i(n)}^{(2)} \left(1 - \zeta_{i(n)}^2 \right) t_{j(n)} \qquad i \in \{1,...,N\}, j \in \{1,...,a+b+1\} \qquad (3.17)$$

Nous rappelons que e_u représente l'erreur sur la sortie du réseau N_C.

3.3.2.3. Calcul de e_u

Les équations d'adaptation du réseau N_C nécessitent l'évaluation de l'erreur e_u comme le montrent les équations (3.16) et (3.17). C'est d'ailleurs le calcul de ce terme qui est au centre de la proposition des nouveaux algorithmes apportés par cette thèse. Étant donné que nous ne pouvons évaluer directement l'erreur en sortie du contrôleur en l'absence de signal de référence

pour cette sortie, nous utilisons l'erreur de trajectoire e_C. Notons que cette erreur représente une bonne information sur la qualité de la commande. En effet, étant donné que notre objectif de commande est le suivi de trajectoire, plus e_C est faible plus la commande est bonne. C'est donc sur cette erreur que nous basons le critère quadratique pour l'adaptation de N_C

$$\xi_{C(n+1)} = \frac{1}{2} e^2_{C(n+1)} = \frac{1}{2}\left(\dot{y}_{(n+1)} - y_{(n+1)}\right)^2 \tag{3.18}$$

Le principe général est alors de considérer N_I et N_C comme deux parties d'un même réseau, dans notre cas à quatre couches, et d'adapter les poids de N_C en rétro propageant l'erreur e_C à travers les différentes couches de ce réseau, c'est-à-dire d'abord à travers N_I, ensuite à travers N_C.
De même que pour N_I, la loi du gradient pour l'adaptation des poids du réseau N_C s'écrit :

$$\Delta v_{(n+1)} = -\eta_C \frac{\partial \xi_{C(n+1)}}{\partial v_{(n)}} \tag{3.19}$$

où η_C est le taux d'apprentissage de N_C et v représente $v^{(1)}_{ij}$ ou $v^{(2)}_{1j}$.

Nous utilisons cette fois encore le principe de dérivation en chaîne pour développer l'expression du gradient dans (3.19).

$$\begin{aligned} \Delta v_{(n+1)} &= -\eta_C \frac{\partial \xi_{C(n+1)}}{\partial e_{C(n+1)}} \frac{\partial e_{C(n+1)}}{\partial y_{(n+1)}} \frac{\partial y_{(n+1)}}{\partial v_{(n)}} \\ &= \eta_C e_{C(n+1)} \frac{\partial y_{(n+1)}}{\partial v_{(n)}} \end{aligned} \tag{3.20}$$

À la différence du réseau N_I, nous obtenons une équation avec une dérivée qui n'est pas directement calculable dans l'équation de propagation du réseau. En effet, les équations de propagation du réseau N_C ne donnent pas de relation entre $y_{(n+1)}$ et les poids $v_{(n)}$. Pour établir une relation entre ces deux variables, nous utilisons l'hypothèse suivante :

$$\frac{\partial y_{(n+1)}}{\partial v_{(n)}} \approx \frac{\partial \hat{y}_{(n+1)}}{\partial v_{(n)}} \tag{3.21}$$

L'hypothèse formulée dans l'équation (3.21) peut être justifiée en plusieurs étapes :

- rappelons d'abord que d'après le théorème d'approximation universel décrit dans le deuxième chapitre, si f satisfait aux conditions du théorème, il existe un RNA multicouches capable de modéliser f dans son espace d'entrée à n'importe quel degré d'exactitude [Hornik & Stinchcombe '89 ; Funahashi '89],
- en appliquant la règle de dérivation en chaîne, l'équation (3.21) peut s'écrire :

$$\frac{\partial y_{(n+1)}}{\partial u_{(n)}}\frac{\partial u_{(n)}}{\partial v_{(n)}} \approx \frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}}\frac{\partial u_{(n)}}{\partial v_{(n)}}$$

$$\Leftrightarrow \frac{\partial y_{(n+1)}}{\partial u_{(n)}} \approx \frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}}$$

- selon le théorème d'approximation universel, les sorties des modèles réels et estimés du système peuvent être rendues aussi proches que désirées. En supposant f de classe C^2, la convergence des dérivées est assurée et les dérivées des sorties réelles et estimées par rapport à l'entrée $u_{(n)}$ peuvent donc également être rendues aussi proches que désirées,
- le choix des paramètres du réseau et des paramètres d'adaptation pour l'obtention de ce niveau de précision dépendent du système considéré et des trajectoires de référence. Ces choix permettant de satisfaire à la condition (3.21) doivent donc être élaborés pour chaque système considéré ou au mieux pour une classe de systèmes non linéaires.

En injectant (3.21) dans (3.20), nous obtenons une nouvelle expression du gradient :

$$\Delta v_{(n+1)} = \eta_C e_{C(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial v_{(n)}} \tag{3.22}$$

Nous appliquons une nouvelle fois la dérivation en chaîne :

$$\Delta v_{(n+1)} = \eta_C e_{C(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}}\frac{\partial u_{(n)}}{\partial v_{(n)}} \tag{3.23}$$

Rappelons l'équation analogue obtenue pour le réseau N_I :

$$\Delta w_{(n+1)} = \eta_C e_{I(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial w_{(n)}}$$

Par analogie, nous déduisons de (3.23) l'erreur équivalente pour la sortie u du réseau N_C :

$$e_{u(n+1)} = e_{C(n+1)} \frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}} \tag{3.24}$$

Ainsi, nous obtenons une équation d'adaptation similaire à celle du réseau N_I :

$$\Delta v_{(n+1)} = \eta_C e_{u(n+1)} \frac{\partial u_{(n)}}{\partial v_{(n)}} \tag{3.25}$$

L'expression de $\partial \hat{y}_{(n+1)} / \partial u_{(n)}$ peut être obtenue en dérivant l'équation de propagation (3.6) :

$$\frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}} = \frac{\partial}{\partial u_{(n)}} \left(\sum_{i=1}^{M} w_{1i(n)}^{(2)} \tanh \left(\sum_{j=1}^{a+b+1} w_{ij(n)}^{(1)} s_{j(n)} \right) \right) \tag{3.26}$$

En considérant que seules les entrées $s_{j(n)}$ peuvent dépendre de $u_{(n)}$ et en utilisant la règle de dérivation composée pour la fonction tangente hyperbolique, il vient :

$$\frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}} = \sum_{i=1}^{M} w_{1i(n)}^{(2)} \left(1 - x_{i(n)}^2\right) \left(\sum_{j=1}^{a+b+1} w_{ij(n)}^{(1)} \frac{\partial s_{j(n)}}{\partial u_{(n)}} \right) \tag{3.27}$$

Rappelons que :

$$s_{j(n)} = \begin{cases} u_{(n-j+1)} & \text{si} \quad 1 \le j \le a \\ y_{(n+a-j+1)} & \text{si} \quad a < j \le a+b \\ 1 & \text{si} \quad j = a+b+1 \end{cases} \tag{3.28}$$

Selon le principe de RPS, nous dérivons (3.28) par rapport à $u_{(n)}$ comme suit :

$$\frac{\partial s_{j(n)}}{\partial u_{(n)}} = \begin{cases} 1 & \text{si} \quad j = 1 \\ 0 & \text{sinon} \end{cases} \tag{3.29}$$

Il ne reste qu'à injecter (3.29) dans (3.27) pour obtenir l'expression finale de $\partial \hat{y}_{(n+1)} / \partial u_{(n)}$:

$$\frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}} = \sum_{i=1}^{M} w_{1i(n)}^{(2)} \left(1 - x_{i(n)}^{2}\right) w_{i1(n)}^{(1)} \tag{3.30}$$

Finalement, nous injectons cette expression dans (3.24) pour obtenir l'expression finale de e_u :

$$e_{u(n+1)} = e_{C(n+1)} \sum_{i=1}^{M} w_{1i(n)}^{(2)} \left(1 - x_{i(n)}^{2}\right) w_{i1(n)}^{(1)} \tag{3.31}$$

Ceci définit l'expression de e_u obtenue avec la méthode RPS.

3.3.3. Rétro Propagation Dynamique (RPD)

3.3.3.1. Premier algorithme : RPD1

En observant l'équation (3.31), nous constatons que dans l'expression de e_u obtenue avec la méthode RPS, l'erreur de trajectoire n'est rétro propagée que dans la partie supérieure du réseau N_I, comme si l'erreur sur la sortie n'était due qu'à l'entrée $u_{(n)}$. La figure 3.1 donne une illustration de cette rétro propagation de l'erreur de trajectoire. Les traits en gras représentent les différents chemins de rétro propagation à travers le modèle neuronal N_I.

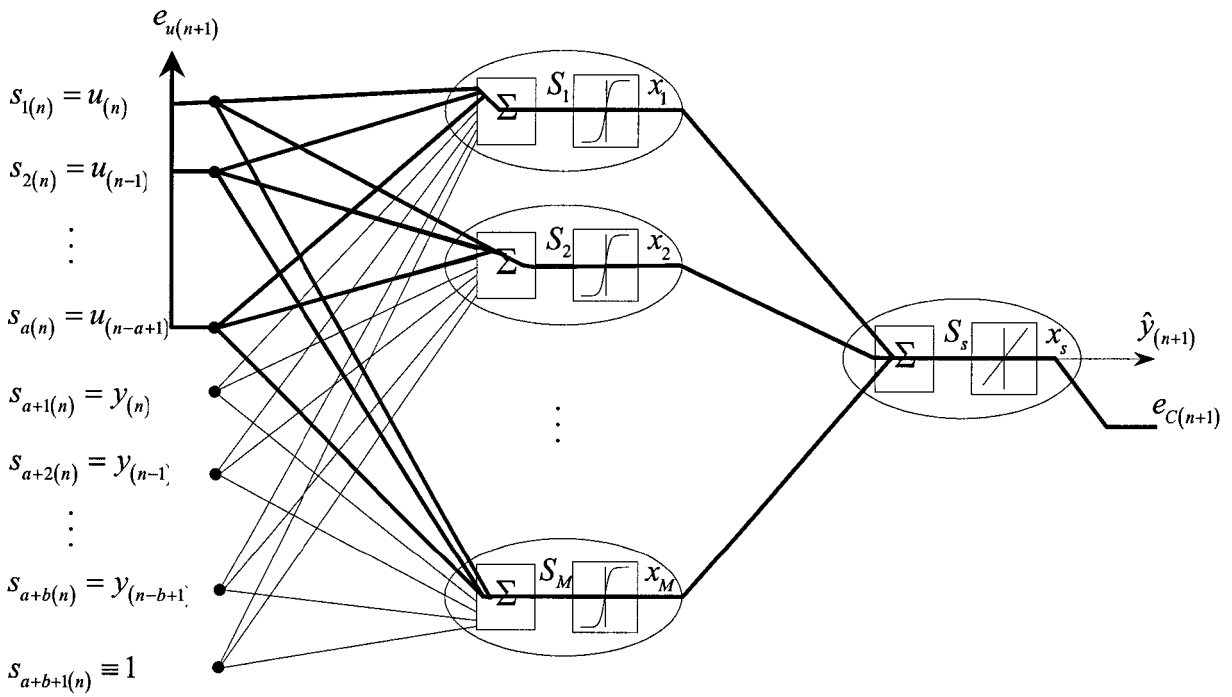


Figure 3.1. Illustration de la rétro propagation de l'erreur de trajectoire dans l'algorithme RPS

Cependant, si nous observons le modèle dynamique du système à contrôler, modèle défini par l'équation (2.3), nous constatons que la sortie du système, et par conséquent l'erreur sur cette sortie, dépend de $u_{(n)}$ mais aussi d'un certain nombre de valeurs passées de $u_{(n)}$. Pour en tenir compte, nous proposons de modifier l'expression de l'erreur $e_{u(n+1)}$ comme suit :

$$e_{u(n+1)} = e_{C(n+1)} \sum_{l=1}^{a} \frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n-l+1)}} \tag{3.32}$$

Cette formulation de e_u permet de réaliser une sommation des influences des différentes entrées de type $u_{(n-k+1)}$ où $k \in \{1,...,a\}$ et constitue l'originalité de l'algorithme RPD1. Notons qu'il serait également possible d'appliquer des coefficients multiplicateurs différents à chacun des termes de la somme pour réaliser un filtrage temporel des influences des différentes entrées. Cependant, le fait de laisser tous les termes avec un coefficient unitaire est très avantageux au niveau de l'ITGE puisqu'il n'est pas nécessaire d'implanter de coefficients de filtrage.

En procédant de la même façon que pour la méthode RPS, cette nouvelle formulation de e_u conduit à une expression modifiée de (3.27) :

$$\frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n)}} = \sum_{i=1}^{M} \left\{ w_{1i(n)}^{(2)} \left(1 - x_{i(n)}^{2}\right) \left(\sum_{j=1}^{a+b+1} w_{ij(n)}^{(1)} \sum_{l=1}^{a} \frac{\partial s_{j(n)}}{\partial u_{(n-l+1)}} \right) \right\} \tag{3.33}$$

Nous devons également adapter (3.29) comme suit :

$$\frac{\partial s_{j(n)}}{\partial u_{(n-l+1)}} = \begin{cases} 1 & \text{si} \quad j = l \\ 0 & \text{sinon} \end{cases} \qquad 1 \le j \le a \tag{3.34}$$

Finalement, la nouvelle expression de $e_{u(n+1)}$ est :

$$e_{u(n+1)} = e_{C(n+1)} \sum_{k=1}^{M} \left(w_{1k(n)}^{(2)} \left(1 - x_{k(n)}^{2}\right) \sum_{l=1}^{a} w_{kl(n)}^{(1)} \right) \tag{3.35}$$

La figure 3.2 donne une illustration de la rétro propagation de l'erreur de trajectoire dans ce nouvel algorithme.

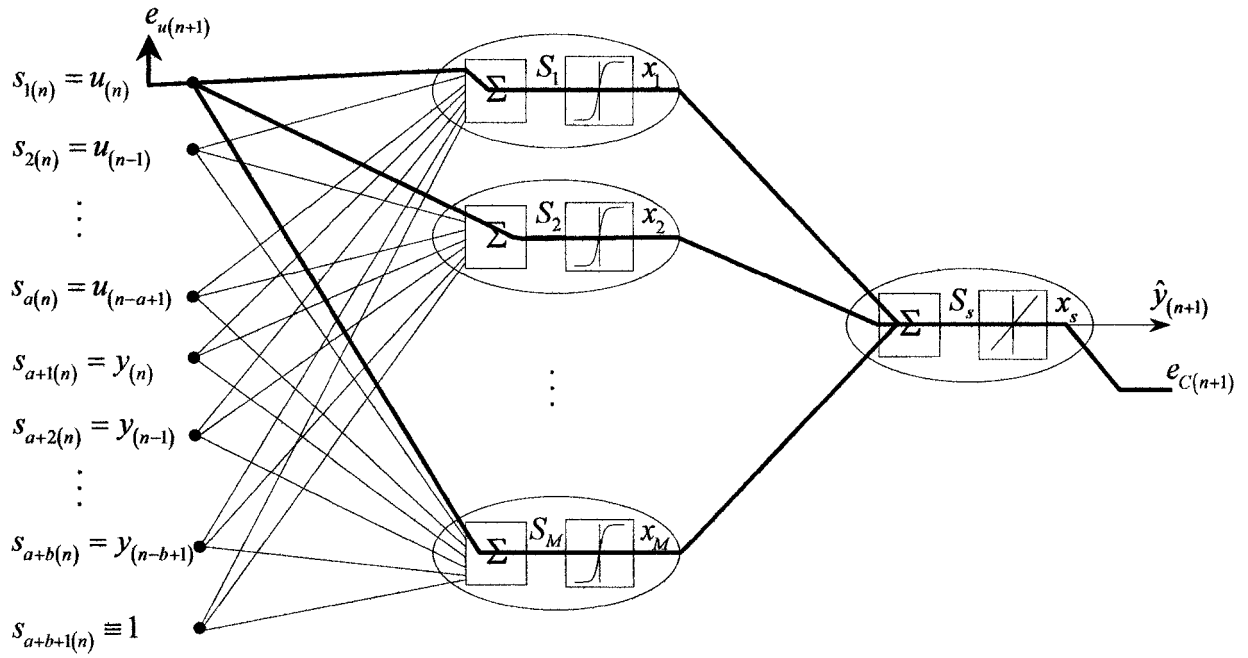


Figure 3.2. Illustration de la rétro propagation de l'erreur de trajectoire dans l'algorithme RPD1

Nous remarquons bien sur la figure 3.2 que l'erreur de trajectoire est maintenant rétro propagée vers toutes les entrées $u_{(n-k+1)}$ du réseau ($1 \le k \le a$). Ceci a pour but de considérer davantage d'information sur les termes dynamiques et donc d'améliorer la commande du système.

3.3.3.2. Deuxième algorithme : RPD2

Dans ce deuxième algorithme, il s'agit encore de revoir le calcul de $e_{u(n+1)}$. Plus particulièrement, nous revenons sur l'équation (3.34) :

$$\frac{\partial s_{j(n)}}{\partial u_{(n-l+1)}} = \begin{cases} 1 & \text{si} \quad j = l \\ 0 & \text{sinon} \end{cases}$$

Effectivement, il est trivial d'écrire :

$$\frac{\partial u_{(n)}}{\partial u_{(n)}} = 1; \frac{\partial u_{(n-1)}}{\partial u_{(n-1)}} = 1; \cdots; \frac{\partial u_{(n-a+1)}}{\partial u_{(n-a+1)}} = 1$$

Cependant, d'autres dérivées de l'équation (3.34) sont non nulles et ont été négligées dans l'algorithme RPD1. Par exemple, le terme $\partial s_{j(n)}/\partial u_{(n-l+1)}$ pour $j=1$ et $l=2$ s'écrit $\partial u_{(n)}/\partial u_{(n-1)}$. Dans l'équation (3.34) nous avons annulé ce terme. En observant le modèle du contrôleur défini par l'équation (2.7), il apparaît que $u_{(n)}$ dépend de $u_{(n-1)}$ et donc que le terme $\partial u_{(n)}/\partial u_{(n-1)}$ est *a priori* non nul. De la même façon, un terme tel que $\partial y_{(n)}/\partial u_{(n-1)}$ ne peut être considéré nul conformément au modèle du système défini par l'équation (2.3). Nous reprenons alors le développement des équations sans négliger ces termes. Ceci constitue l'originalité de l'algorithme RPD2.

Pour faciliter l'écriture des équations récurrentes qui apparaîtront dans le développement de RPD2, nous posons :

$$\pi^k_{(n+1)} = \frac{\partial \hat{y}_{(n+1)}}{\partial u_{(n-k+1)}} \qquad 1 \le k \le a \tag{3.36}$$

et

$$\sigma^k_{(n+1)} = \frac{\partial u_{(n)}}{\partial u_{(n-k+1)}} \qquad 1 \le k \le a \tag{3.37}$$

Une expression équivalente à (3.32) est alors obtenue avec la variable π :

$$e_{u(n+1)} = e_{C(n+1)} \sum_{l=1}^{a} \pi^l_{(n+1)} \tag{3.38}$$

Dérivons en détails le calcul de $\pi^l_{(n+1)}$ à partir de l'expression obtenue dans (3.33) :

$$\pi^l_{(n+1)} = \sum_{i=1}^{M} \left(w^{(2)}_{1i(n)} \left(1 - x^2_{i(n)}\right) \sum_{j=1}^{a+b+1} w^{(1)}_{ij(n)} \frac{\partial s_{j(n)}}{\partial u_{n-l+1}} \right) \qquad 1 \le l \le a \tag{3.39}$$

En séparant les entrées s_j de type $u_{(n-l+1)}$ et $y_{(n-l+1)}$, (3.39) devient :

$$\pi^l_{(n+1)} = \sum_{i=1}^{M} \left(w^{(2)}_{1i(n)} \left(1 - x^2_{i(n)}\right) \left(\sum_{j=1}^{a} w^{(1)}_{ij(n)} \frac{\partial u_{(n-j+1)}}{\partial u_{(n-l+1)}} + \sum_{j=a+1}^{a+b} w^{(1)}_{ij(n)} \frac{\partial y_{(n+a-j+1)}}{\partial u_{(n-l+1)}} \right) \right) \quad 1 \le l \le a \tag{3.40}$$

Les dérivées de y par rapport à u ne sont pas calculables directement puisque nous ne connaissons pas le modèle exact du système. Cette fois encore, comme nous l'avions établi pour l'algorithme RPS, nous utilisons l'approximation définie par (3.21). L'équation (3.40) devient alors :

$$\pi^{l}_{(n+1)} = \sum_{i=1}^{M}\left(w^{(2)}_{1i(n)}\left(1-x^{2}_{i(n)}\right)\left(\sum_{j=1}^{a} w^{(1)}_{ij(n)} \frac{\partial u_{(n-j+1)}}{\partial u_{(n-l+1)}} + \sum_{j=a+1}^{a+b} w^{(1)}_{ij(n)} \frac{\partial \hat{y}_{(n+a-j+1)}}{\partial u_{(n-l+1)}} \right)\right) \qquad 1 \le l \le a \quad (3.41)$$

La troisième somme peut être simplifiée en remplaçant j par $j+a$ et (3.41) devient :

$$\pi^{l}_{(n+1)} = \sum_{i=1}^{M}\left(w^{(2)}_{1i(n)}\left(1-x^{2}_{i(n)}\right)\left(\sum_{j=1}^{a} w^{(1)}_{ij(n)} \frac{\partial u_{(n-j+1)}}{\partial u_{(n-l+1)}} + \sum_{j=1}^{b} w^{(1)}_{ij_a(n)} \frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-l+1)}} \right)\right) \quad (3.42)$$

où $j_a = j + a$.

En considérant les modèles des deux réseaux, les conditions pour avoir des dérivées non nulles dans (3.42) s'écrivent :

$$\frac{\partial u_{(n-j+1)}}{\partial u_{(n-l+1)}} \neq 0 \qquad \text{si} \qquad n-j+1 \ge n-l+1 \quad (3.43)$$

$$\frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-l+1)}} \neq 0 \qquad \text{si} \qquad n-j+1 > n-l+1 \quad (3.44)$$

ou encore :

$$\frac{\partial u_{(n-j+1)}}{\partial u_{(n-l+1)}} \neq 0 \qquad \text{si} \qquad j \le l \quad (3.45)$$

$$\frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-l+1)}} \neq 0 \qquad \text{si} \qquad j < l \quad (3.46)$$

Nous pouvons alors réduire (3.42) en injectant les conditions précédentes, il vient

$$\pi^{l}_{(n+1)} = \sum_{i=1}^{M}\left(w^{(2)}_{1i(n)}\left(1-x^{2}_{i(n)}\right)\left(\sum_{j\le l} w^{(1)}_{ij(n)} \frac{\partial u_{(n-j+1)}}{\partial u_{(n-l+1)}} + \sum_{j<l} w^{(1)}_{ij_a(n)} \frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-l+1)}} \right)\right) \quad 1 \le l \le a \qquad (3.47)$$

Dans cette équation, nous voyons apparaître des termes π et σ à l'intérieur des deuxième et troisième sommes :

$$\pi^{l}_{(n+1)} = \sum_{i=1}^{M}\left(w^{(2)}_{1i(n)}\left(1-x^{2}_{i(n)}\right)\left(\sum_{j\le l} w^{(1)}_{ij(n)} \sigma^{l-j+1}_{(n-j+2)} + \sum_{j<l} w^{(1)}_{ij_a(n)} \pi^{l-j}_{(n-j+1)} \right)\right) \quad 1 \le l \le a \qquad (3.48)$$

Nous voyons déjà apparaître une récurrence en π dans (3.48). Il reste encore à évaluer l'expression des termes σ pour compléter l'algorithme RPD2. Étant donné l'expression donnée dans (3.37), le calcul de $\sigma^{k}_{(n+1)}$ doit être effectué avec l'équation de propagation de N_C.

Nous utilisons la forme compacte de cette équation de propagation :

$$u_{(n)} = \sum_{i=1}^{N} v^{(2)}_{1i(n)} \zeta_{i(n)} = \sum_{i=1}^{N}\left\{ v^{(2)}_{1i(n)} \tanh\left(\sum_{j=1}^{a+b+1} v^{(1)}_{ij(n)} t_{j(n)} \right)\right\} \qquad (3.49)$$

Notons que $\sigma^{1}_{(n+1)}$ est très simple à évaluer puisque :

$$\sigma^{1}_{(n+1)} \equiv 1 \qquad (3.50)$$

Le calcul de $\sigma^{k}_{(n+1)}$ pour $k \ge 2$ est détaillé ci-après.

$$\sigma^{k}_{(n+1)} = \frac{\partial}{\partial u_{(n-k+1)}}\left(\sum_{i=1}^{N}\left\{ v^{(2)}_{1i(n)} \tanh\left(\sum_{j=1}^{a+b+1} v^{(1)}_{ij(n)} t_{j(n)} \right)\right\}\right) \quad k \ge 2 \qquad (3.51)$$

Seules les entrées $t_{j(n)}$ peuvent dépendre de $u_{(n-k+1)}$, donc :

$$\sigma^{k}_{(n+1)} = \sum_{i=1}^{N}\left\{ \frac{\partial}{\partial u_{(n-k+1)}}\left(v^{(2)}_{1i(n)} \tanh\left(\sum_{j=1}^{a+b+1} v^{(1)}_{ij(n)} t_{j(n)} \right)\right)\right\} \quad k \ge 2 \qquad (3.52)$$

En séparant les entrées t_j de type $u_{(n-l+1)}$ et $y_{(n-l+1)}$ et en appliquant la dérivation composée de la fonction tangente hyperbolique, (3.52) devient :

$$\sigma_{(n+1)}^{k}=\sum_{i=1}^{N}\left(v_{1i(n)}^{(2)}\left(1-\zeta_{i(n)}^{2}\right)\left(\sum_{j=2}^{a}\left(v_{ij(n)}^{(1)}\frac{\partial u_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)+\sum_{j=a+1}^{a+b}\left(v_{ij(n)}^{(1)}\frac{\partial y_{(n+a-j+1)}}{\partial u_{(n-k+1)}}\right)\right)\right)\qquad k\geq 2 \quad (3.53)$$

La troisième somme peut être simplifiée en remplaçant j par $j+a$, (3.53) devient

$$\sigma_{(n+1)}^{k}=\sum_{i=1}^{N}\left(v_{1i(n)}^{(2)}\left(1-\zeta_{i(n)}^{2}\right)\left(\sum_{j=2}^{a}\left(v_{ij(n)}^{(1)}\frac{\partial u_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)+\sum_{j=1}^{b}\left(v_{ij_a(n)}^{(1)}\frac{\partial y_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)\right)\right)\qquad k\geq 2 \quad (3.54)$$

Notons que l'indice j dans la deuxième somme commence à 2 puisque la première entrée du réseau est la sortie de référence y^{*} qui est indépendante des entrées. Nous utilisons à nouveau l'hypothèse formulée dans (3.21) pour développer l'équation (3.54).

$$\sigma_{(n+1)}^{k}=\sum_{i=1}^{N}\left(v_{1i(n)}^{(2)}\left(1-\zeta_{i(n)}^{2}\right)\left(\sum_{j=2}^{a}\left(v_{ij(n)}^{(1)}\frac{\partial u_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)+\sum_{j=1}^{b}\left(v_{ij_a(n)}^{(1)}\frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)\right)\right)\qquad k\geq 2 \quad (3.55)$$

En considérant les modèles des deux réseaux, les conditions pour avoir des dérivées non nulles dans (3.55) s'écrivent :

$$\frac{\partial u_{(n-j+1)}}{\partial u_{(n-k+1)}}\neq 0 \qquad \text{si} \qquad n-j+1\geq n-k+1 \quad (3.56)$$

$$\frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-k+1)}}\neq 0 \qquad \text{si} \qquad n-j+1> n-k+1 \quad (3.57)$$

ou encore :

$$\frac{\partial u_{(n-j+1)}}{\partial u_{(n-k+1)}}\neq 0 \qquad \text{si} \qquad j\leq k \quad (3.58)$$

$$\frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-k+1)}}\neq 0 \qquad \text{si} \qquad j< k \quad (3.59)$$

Nous pouvons alors réduire (3.55) en injectant les conditions précédentes, il vient :

$$\sigma^{k}_{(n+1)} = \sum_{i=1}^{N}\left(v^{(2)}_{1i(n)}\left(1-\zeta^{2}_{i(n)}\right)\left(\sum_{2\le j\le k}\left(v^{(1)}_{ij(n)}\frac{\partial u_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)+\sum_{j<k}\left(v^{(1)}_{ij_a(n)}\frac{\partial \hat{y}_{(n-j+1)}}{\partial u_{(n-k+1)}}\right)\right)\right) \qquad k\ge 2 \quad (3.60)$$

Dans cette équation, nous voyons à nouveau apparaître des termes π et σ à l'intérieur des deuxième et troisième sommes :

$$\sigma^{k}_{(n+1)} = \sum_{i=1}^{N}\left(v^{(2)}_{1i(n)}\left(1-\zeta^{2}_{i(n)}\right)\left(\sum_{2\le j\le k}\left(v^{(1)}_{ij(n)}\sigma^{k-j+1}_{(n-j+2)}\right)+\sum_{j<k}\left(v^{(1)}_{ij_a(n)}\pi^{k-j}_{(n-j+1)}\right)\right)\right) \qquad k\ge 2 \qquad (3.61)$$

Ceci complète le calcul de $e_{u(n+1)}$ pour l'algorithme RPD2. Nous voyons apparaître dans (3.61) une autre récurrence en σ et π.

Afin de faciliter la compréhension du lecteur de l'algorithme RPD2, considérons l'exemple où $a=b=4$. Dans ce cas, nous devons calculer à l'instant $n+1$ quatre valeurs de π et trois de σ. Le tableau 3.1 présente dans la première colonne les variables à calculer à l'instant $n+1$ et dans la deuxième colonne la liste des dépendances déduites des équations (3.61) et (3.48).

Tableau 3.1. Méthode RPD2 dans le cas où $a=b=4$: variables à calculer à l'instant $n+1$ et leurs dépendances

Variable	Dépend de
$\pi^{1}_{(n+1)}$	$\left\{\sigma^{1}_{(n+1)}\right\}$
$\pi^{2}_{(n+1)}$	$\left\{\sigma^{2}_{(n+1)};\sigma^{1}_{(n)};\pi^{1}_{(n)}\right\}$
$\pi^{3}_{(n+1)}$	$\left\{\sigma^{3}_{(n+1)};\sigma^{2}_{(n)};\sigma^{1}_{(n-1)};\pi^{2}_{(n)};\pi^{1}_{(n-1)}\right\}$
$\pi^{4}_{(n+1)}$	$\left\{\sigma^{4}_{(n+1)};\sigma^{3}_{(n)};\sigma^{2}_{(n-1)};\sigma^{1}_{(n-2)};\pi^{3}_{(n)};\pi^{2}_{(n-1)};\pi^{1}_{(n-2)}\right\}$
$\sigma^{2}_{(n+1)}$	$\left\{\sigma^{1}_{(n)};\pi^{1}_{(n)}\right\}$
$\sigma^{3}_{(n+1)}$	$\left\{\sigma^{2}_{(n)};\sigma^{1}_{(n-1)};\pi^{2}_{(n)};\pi^{1}_{(n-1)}\right\}$
$\sigma^{4}_{(n+1)}$	$\left\{\sigma^{3}_{(n)};\sigma^{2}_{(n-1)};\sigma^{1}_{(n-2)};\pi^{3}_{(n)};\pi^{2}_{(n-1)};\pi^{1}_{(n-2)}\right\}$

Ceci implique pour RPD2 un certain nombre de mémoires supplémentaires dont il faudra tenir compte dans l'évaluation de la complexité de cet algorithme en termes de critères d'ITGE.

Les équations de calcul des termes π et σ dans le cas particulier où $a = b = 4$ sont :

$$\pi^1_{(n+1)} = \sum_{i=1}^{M} w^{(2)}_{1i(n)} \left(1 - x^2_{i(n)}\right) w^{(1)}_{i1(n)} \sigma^1_{(n+1)}$$

$$\pi^2_{(n+1)} = \sum_{i=1}^{M} w^{(2)}_{1i(n)} \left(1 - x^2_{i(n)}\right) \left(w^{(1)}_{i1(n)} \sigma^2_{(n+1)} + w^{(1)}_{i2(n)} \sigma^1_{(n)} + w^{(1)}_{i5(n)} \pi^1_{(n)} \right)$$

$$\pi^3_{(n+1)} = \sum_{i=1}^{M} w^{(2)}_{1i(n)} \left(1 - x^2_{i(n)}\right) \left(w^{(1)}_{i1(n)} \sigma^3_{(n+1)} + w^{(1)}_{i2(n)} \sigma^2_{(n)} + w^{(1)}_{i3(n)} \sigma^1_{(n-1)} + w^{(1)}_{i5(n)} \pi^2_{(n)} + w^{(1)}_{i6(n)} \pi^1_{(n-1)} \right)$$

$$\pi^4_{(n+1)} = \sum_{i=1}^{M} w^{(2)}_{1i(n)} \left(1 - x^2_{i(n)}\right) \left(w^{(1)}_{i1(n)} \sigma^4_{(n+1)} + w^{(1)}_{i2(n)} \sigma^3_{(n)} + w^{(1)}_{i3(n)} \sigma^2_{(n-1)} + w^{(1)}_{i4(n)} \sigma^1_{(n-2)} + w^{(1)}_{i5(n)} \pi^3_{(n)} \right.$$
$$\left. + w^{(1)}_{i6(n)} \pi^2_{(n-1)} + w^{(1)}_{i7(n)} \pi^1_{(n-2)} \right)$$

$$\sigma^2_{(n+1)} = \sum_{i=1}^{M} v^{(2)}_{1i(n)} \left(1 - \zeta^2_{i(n)}\right) \left(v^{(1)}_{i1(n)} \sigma^1_{(n)} + v^{(1)}_{i5(n)} \pi^1_{(n)} \right)$$

$$\sigma^3_{(n+1)} = \sum_{i=1}^{M} v^{(2)}_{1i(n)} \left(1 - \zeta^2_{i(n)}\right) \left(v^{(1)}_{i2(n)} \sigma^2_{(n)} + v^{(1)}_{i3(n)} \sigma^1_{(n-1)} + v^{(1)}_{i5(n)} \pi^2_{(n)} + v^{(1)}_{i6(n)} \pi^1_{(n-1)} \right)$$

$$\sigma^4_{(n+1)} = \sum_{i=1}^{M} v^{(2)}_{1i(n)} \left(1 - \zeta^2_{i(n)}\right) \left(v^{(1)}_{i2(n)} \sigma^3_{(n)} + v^{(1)}_{i3(n)} \sigma^2_{(n-1)} + v^{(1)}_{i4(n)} \sigma^1_{(n-2)} + v^{(1)}_{i5(n)} \pi^3_{(n)} \right.$$
$$\left. + v^{(1)}_{i6(n)} \pi^2_{(n-1)} + v^{(1)}_{i7(n)} \pi^1_{(n-2)} \right)$$

3.3.3.3. Troisième algorithme : RTRL

Dans la troisième approche, il s'agit d'intégrer le principe de rétro propagation dynamique au niveau de l'adaptation des poids du contrôleur. Le calcul de $e_{u(n+1)}$ est effectué selon les équations de RPD2 définies dans la section précédente. Rappelons l'équation (3.25) pour l'adaptation des poids du réseau N_C :

$$\Delta v_{(n+1)} = \eta_C e_{u(n+1)} \frac{\partial u_{(n)}}{\partial v_{(n)}}$$

Dans les différentes approches présentées jusqu'ici, le terme $\partial u_{(n)}/\partial v_{(n)}$ a été évalué selon le principe de rétro propagation statique en considérant le réseau N_C de la même façon que le réseau N_I. Cependant, le réseau N_C possède une propriété particulière : il est récurrent. Cette récurrence est due à la rétroaction de la sortie $u_{(n)}$ vers l'entrée du réseau. Considérant cette particularité des réseaux récurrents, quelques chercheurs ont élaboré des algorithmes particuliers à ce type de réseaux [Hertz *et al.* '91]. Un de ces algorithmes a retenu beaucoup d'attention pour sa possibilité d'utilisation en ligne et son amélioration de la vitesse de convergence par rapport à l'algorithme RPS, l'algorithme d'apprentissage récurrent en temps réel (*angl. Real Time Recurrent Learning - RTRL*) [Williams & Zipser '89]. Même si les auteurs n'ont pas utilisé le même nom, le principe de cet algorithme est d'appliquer la rétro propagation dynamique dans le calcul de $\partial u_{(n)}/\partial v_{(n)}$. Étant donné que nous utilisons l'algorithme tel que présenté dans [Williams & Zipser '89], nous ne présentons dans cette section que les équations finales de calcul de $\partial u_{(n)}/\partial v_{(n)}$ en distinguant les deux couches du réseau.

Pour simplifier les équations et mieux faire apparaître les récurrences, nous ajoutons les notations suivantes :

$$\nabla u^{(2)}_{1j(n)} = \frac{\partial u_{(n)}}{\partial v^{(2)}_{1j(n)}} \qquad j \in \{1,...,N\} \tag{3.62}$$

$$\nabla u^{(1)}_{ij(n)} = \frac{\partial u_{(n)}}{\partial v^{(1)}_{ij(n)}} \qquad i \in \{1,...,N\}, j \in \{1,...,a+b+1\} \tag{3.63}$$

Alors, les équations d'adaptation des poids pour les deux couches du réseau N_C s'écrivent :

$$\nabla u^{(2)}_{1j(n)} = \zeta_{l(n)} + \sum_{k=1}^{N}\left(v^{(2)}_{1k(n)}\left(1-\zeta^2_{k(n)}\right)\sum_{l=2}^{a} v^{(1)}_{kl(n)} \nabla u^{(2)}_{1j(n-l+1)}\right) \qquad j \in \{1,...,N\} \tag{3.64}$$

$$\nabla u^{(1)}_{ij(n)} = \sum_{k=1}^{N} v^{(2)}_{1k(n)}\left(1-\zeta^2_{k(n)}\right)\left(\delta_{ki} t_{j(n)} + \sum_{l=2}^{a}\left(v^{(1)}_{kl(n)} \nabla u^{(1)}_{ij(n-l+1)}\right)\right)$$

$$i \in \{1,...,N\} \; j \in \{1,...,a+b+1\} \tag{3.65}$$

où δ_{ki} est le symbole de Kronecker défini par :

$$\delta_{ki} = \begin{cases} 1 & \text{si } k = i \\ 0 & \text{sinon} \end{cases} \tag{3.66}$$

3.3.3.4. Récapitulatif des quatre algorithmes

Le tableau 3.2 dresse un récapitulatif des équations finales d'adaptation pour les quatre algorithmes présentés. Les équations sont divisées en trois groupes : adaptation de N_I, adaptation de N_C et calcul de e_u.

Tableau 3.2. Récapitulatif des équations pour les quatre algorithmes d'adaptation

Algorithme	Équations		
	N_I	N_C	e_u
RPS	(3.13) (3.15)	(3.16) (3.17)	(3.31)
RPD1	(3.13) (3.15)	(3.16) (3.17)	(3.35)
RPD2	(3.13) (3.15)	(3.16) (3.17)	(3.38) (3.48) (3.61)
RTRL	(3.13) (3.15)	(3.25) (3.62) (3.63) (3.64) (3.65) (3.66)	(3.31)

Le tableau 3.3 présente, pour chacun des quatre algorithmes, les méthodes utilisées (c'est-à-dire RPS ou RPD) dans l'adaptation de N_I et N_C ainsi que dans le calcul de e_u.

Tableau 3.3. Méthodes utilisées dans l'adaptation de N_I et N_C ainsi que dans le calcul de e_u pour les quatre algorithmes

Algorithme	Méthode		
	N_I	e_u	N_C
RPS	RPS	RPS	RPS
RPD1	RPS	RPD*	RPS
RPD2	RPS	RPD	RPS
RTRL	RPS	RPS	RPD

* avec certains termes négligés

L'originalité des algorithmes proposés par la thèse est d'utiliser la méthode RPD dans la rétro propagation à travers le modèle neuronal. Ceci aboutit aux deux nouveaux algorithmes RPD1 et RPD2 qui sont les contributions algorithmiques de la thèse.

3.4. Résultats de simulations

3.4.1. Méthode de comparaison des algorithmes

Rappelons que le but de ce chapitre est d'aboutir à un algorithme satisfaisant tant du point de vue algorithmique que du point de vue de l'ITGE. Il s'agit donc d'évaluer de manière comparative les quatre algorithmes présentés dans la section 3.3. Cette évaluation permettra de retenir le ou les algorithmes qui seront les meilleurs compromis entre les différents critères. Ce ou ces algorithmes feront l'objet du développement architectural et de circuits d'ITGE du quatrième chapitre. Dans les sections précédentes, nous avons présenté quatre algorithmes d'adaptation pour les réseaux N_I et N_C : RPS, RPD1, RPD2 et RTRL. Les résultats de simulations présentés dans cette section sont décomposés en deux parties, correspondant aux deux familles de critères d'intérêt. Dans la section 3.4.2, les résultats présentés concernent les critères algorithmiques, principalement la qualité de commande obtenue mais également la sensibilité des algorithmes vis-à-vis du taux d'adaptation du contrôleur. Dans la section 3.4.3, les résultats concernent les critères d'ITGE, principalement l'évaluation de la complexité de calcul des algorithmes et les quantités de mémoires nécessaires.

3.4.2. Évaluation suivant les critères algorithmiques

3.4.2.1. Méthode de comparaison

Dans cette partie, il s'agit d'abord d'évaluer la qualité de la loi de commande complète suivant l'utilisation des méthodes RPS, RPD1, RPD2 ou RTRL pour l'adaptation des réseaux. La qualité de la loi de commande est surtout donnée par la qualité du suivi de trajectoire qui, rappelons le, constitue l'objectif de commande tel que cela avait été défini au deuxième chapitre.

Ainsi, dans chaque exemple de simulation, nous calculerons les erreurs quadratiques moyennes obtenues pour les quatre algorithmes en fonction de la valeur de η_C qui demeure le

paramètre ajustable crucial du contrôleur. Pour chaque valeur de η_C, la loi de commande sera simulée et l'erreur quadratique moyenne calculée par :

$$\varepsilon_{qm} = \frac{1}{N_s} \sum_{n=0}^{N_s-1} \xi_{C(n)} \tag{3.67}$$

où N_s représente ici le nombre d'échantillons utilisés dans le calcul de l'erreur quadratique moyenne. Aussi, étant donné que les poids initiaux du contrôleur sont choisis de manière aléatoire, il peut y avoir des différences entre les résultats obtenus pour différentes valeurs initiales. Chaque courbe présentée par la suite sera donc la moyenne de trente courbes de ε_{qm} obtenues pour la même simulation avec des initialisations différentes des poids synaptiques. La variable ainsi représentée est définie par :

$$\overline{\varepsilon}_{qm} = \frac{1}{30} \sum_{i=1}^{30} \varepsilon_{qm,i} \tag{3.68}$$

où $\varepsilon_{qm,i}$ est la valeur de ε_{qm} obtenue pour la $i^{\text{ème}}$ simulation. Ceci permet simplement de moyenner les variations aléatoires et donc de lisser les courbes d'erreurs quadratiques obtenues. Pour chacun des systèmes que nous considérerons, la vérification de la continuité est triviale. Il a également été vérifié, lors des simulations, que les différentes variables des fonctions f et g évoluent dans $[-1,1]$.

En ce qui concerne le choix des systèmes pour les simulations comparatives, nous adoptons la même approche que dans les articles de référence [Narendra & Parthasarathy '90 ; Levin & Narendra '96 ; Cabrera & Narendra '99 ; Sastry *et al.* '94], c'est-à-dire l'utilisation de systèmes synthétiques. Ce choix peut être fait parmi une infinité de choix possibles. Nous présenterons dans cette section les résultats de simulations obtenus pour quatre systèmes synthétiques. Ces systèmes ont été choisis de façon à créer une bonne variété des caractéristiques suivantes :

- ordre du système,
- type de non linéarité,
- variables affectées par les non linéarités.

Les deux premiers systèmes que nous étudierons sont des systèmes standards bien connus dans la littérature [Narendra & Parthasarathy '90 ; Cabrera & Narendra '99]. Les deux autres sont des systèmes d'ordre plus élevé.

Notons que beaucoup d'autres simulations ont été réalisées par l'auteur, simulations caractérisées par :

- d'autres systèmes,
- variations de certains paramètres des modèles,
- variations de la période d'échantillonnage,
- variations des nombres de neurones sur les couches cachées des deux réseaux,
- variations des autres paramètres de simulations.

Ainsi, de très nombreuses combinaisons de paramètres ont été envisagées. Les résultats obtenus représentent environ cinquante familles de courbes telles que celles qui seront présentées pour les quatre systèmes retenus dans cette section. Ces résultats ont tous corroboré les conclusions qui seront présentés dans la section 3.4.2.8, sans ajouter d'élément nouveau par rapport aux résultats présentés dans cette section.

Ensuite, nous présenterons des résultats de simulations obtenus pour la commande d'un moteur à courant continu avec frottements non linéaires. Cette partie aura pour but de démontrer de manière plus concrète l'applicabilité de la loi de commande à des systèmes réels.

3.4.2.2. Paramètres de simulations

Deux paramètres caractérisent principalement les réseaux utilisés : le nombre de poids synaptiques sur la couche cachée et le taux d'adaptation η. Il est difficile de connaître les valeurs optimales de ces paramètres, c'est-à-dire les valeurs qui donneront les meilleurs résultats en vue des critères considérés.

Le nombre de poids synaptiques est très dépendant de la complexité du modèle que doit réaliser le réseau, tant au niveau du nombre d'entrées nécessaires et du nombre de neurones sur la couche cachée qu'au niveau de la complexité de la fonction à modéliser elle-même (dans notre cas f ou g). Certains principes peuvent nous guider dans le choix du nombre de neurones cachés. Par exemple, ce nombre devrait être au moins égal au nombre d'entrées du réseau et il devrait être

d'autant plus élevé que la complexité de la fonction à modéliser est grande. D'autre part, comme nous l'avions mentionné dans le deuxième chapitre, les RNA sont tolérants aux pannes lorsqu'ils sont correctement dimensionnés. Afin d'augmenter cette tolérance aux pannes, il est possible de choisir un nombre de neurones cachés plus grand que nécessaire afin de créer des redondances d'informations. Il faut toutefois être prudent et ne pas choisir des valeurs déraisonnablement élevées puisque chaque neurone caché supplémentaire implique des poids supplémentaires pour le connecter aux entrées et sorties du réseau. Ces poids supplémentaires impliquent une complexité de calcul plus élevée au niveau de la propagation directe et de la rétro propagation pour l'adaptation. Finalement, le choix d'une bonne valeur pour le nombre de neurones cachés est un choix difficile. Ce choix revient au concepteur qui doit, en fonction du système à contrôler et de son expérience, trouver une valeur adéquate. Pour la suite, nous prendrons un nombre de neurones cachés égal au double du nombre d'entrées, choix empirique bien connu de la littérature et vérifié par de nombreuses simulations réalisées par l'auteur de la thèse.

Le taux d'adaptation des réseaux est un autre paramètre crucial de la loi de commande. D'une manière générale, un taux plus faible garantit la stabilité mais aussi une convergence plus lente alors qu'un taux élevé garantit une convergence plus rapide mais aussi de plus grands risques d'instabilité. Cette fois encore, il s'agit de trouver un bon compromis. La vitesse de convergence nécessaire dépend aussi du problème traité. Si nous considérons un système avec variations de paramètres, plus les paramètres varient rapidement plus le taux d'adaptation doit être élevé afin d'adapter rapidement le réseau aux variations. Notons également que le taux d'adaptation est aussi relié à la période d'échantillonnage. Une fréquence d'échantillonnage plus élevée signifie un plus grand nombre d'échantillons par unité de temps ce qui donne donc plus de périodes à l'algorithme pour converger. Dans ce cas, le taux d'adaptation peut être choisi moins élevé. De plus, même si de faibles valeurs du taux d'adaptation peuvent engendrer des vitesses de convergence plus faibles, celles-ci garantissent néanmoins de meilleures chances de trouver des *minima* locaux plus profonds [Haykin '99]. Comme pour le nombre de neurones cachés, le choix doit être fait judicieusement par le concepteur.

Le tableau 3.4 présente les paramètres utilisés dans les simulations. Le taux d'adaptation η_C du contrôleur a une valeur variable. Les poids du réseau contrôleur N_C sont initialisés avec des valeurs aléatoires de moyenne nulle et uniformément réparties dans

l'intervalle $[-0.01, 0.01]$. Afin d'accélérer la convergence dans tous les cas de simulations et de satisfaire aux conditions d'utilisation de l'hypothèse (3.21), les poids du réseau N_I sont initialisés différemment. Nous procéderons à une phase préalable d'identification du système en présence d'une entrée de commande connue. Les poids du réseau N_I sont initialisés avec des valeurs aléatoires de moyenne nulle et uniformément réparties dans l'intervalle $[-0.01, 0.01]$ et une phase d'identification préalable est menée sur 40000 échantillons (c'est-à-dire une durée de 4s). Les valeurs de poids obtenues à la fin de cette phase d'identification préalable constituent les valeurs initiales pour la simulation complète avec le contrôleur. La variable T_s désigne la période d'échantillonnage.

Tableau 3.4. Paramètres de simulations

Paramètre	Valeur
T_s	0.0001s
η_I	0.02
M	$2*(a+b+1)$
N	$2*(a+b+1)$
η_C	variable

3.4.2.3. Premier système

Le premier système est caractérisé par le modèle NARMAX :

$$y_{(n+1)} = u_{(n)} + u_{(n-1)} + \frac{y_{(n)} y_{(n-1)} \left(2.5 + y_{(n)}\right)}{1 + y_{(n)}^2 + y_{(n-1)}^2} \tag{3.69}$$

La fonction pseudo inverse peut être facilement déduite dans ce cas :

$$u_{(n)} = y^*_{(n+1)} - \frac{y_{(n)} y_{(n-1)} \left(2.5 + y_{(n)}\right)}{1 + y_{(n)}^2 + y_{(n-1)}^2} - u_{(n-1)} \tag{3.70}$$

La phase préalable d'apprentissage du réseau N_I est réalisée en appliquant au système le signal :

$$u_{(n)} = 0.1 * \sin\left(4n\pi T_S\right) \tag{3.71}$$

La trajectoire de référence est définie par

$$y^*_{(n)} = 0.05 * \sin\left(4n\pi T_S\right) \tag{3.72}$$

La figure 3.3 présente les courbes de $\overline{\varepsilon}_{qm}$ obtenues dans la simulation de la loi de commande complète pour ce système et cette référence.

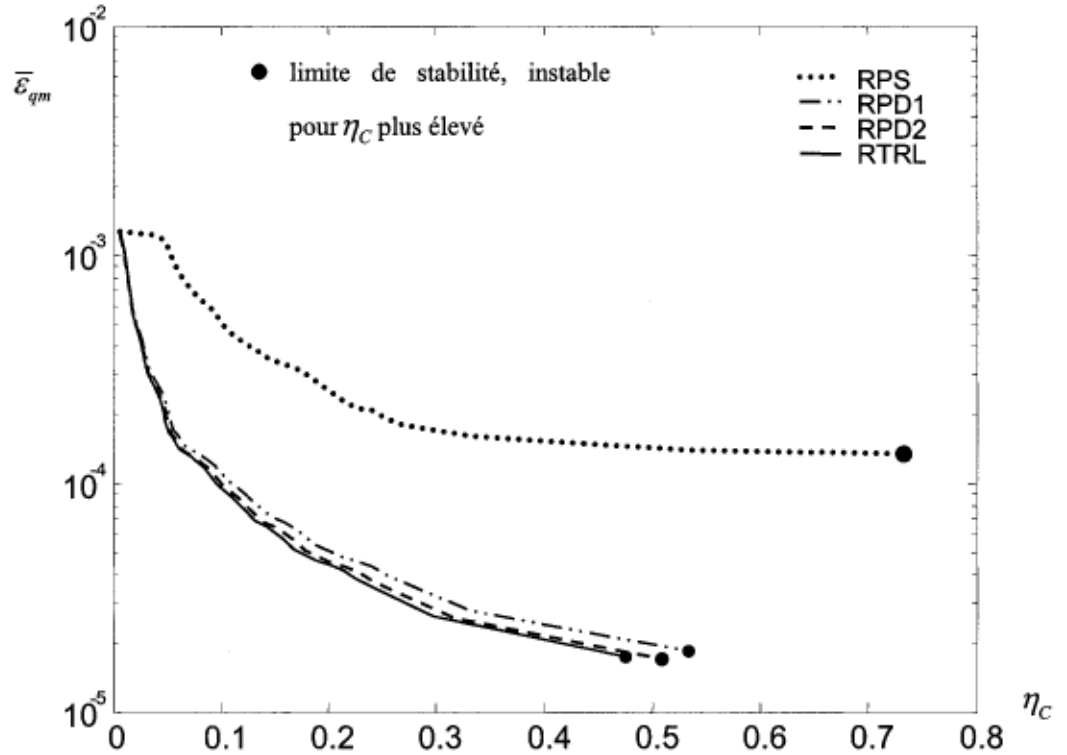


Figure 3.3. Erreur quadratique moyenne pour les quatre algorithmes en fonction de η_C : premier système

3.4.2.4. Deuxième système

Le deuxième système est caractérisé par le modèle NARMAX :

$$y_{(n+1)} = \frac{y_{(n)} y_{(n-1)} y_{(n-2)} u_{(n-1)} \left(y_{(n-2)} - 1\right)}{1 + y^2_{(n-2)} + y^2_{(n-1)}} + u_{(n)} \tag{3.73}$$

La fonction pseudo inverse peut être facilement déduite dans ce cas :

$$u_{(n)} = y^*_{(n+1)} - \frac{y_{(n)}y_{(n-1)}y_{(n-2)}u_{(n-1)}\left(y_{(n-2)}-1\right)}{1+y^2_{(n-2)}+y^2_{(n-1)}} \tag{3.74}$$

La phase préalable d'apprentissage du réseau N_I est réalisée en appliquant au système le signal :

$$u_{(n)} = 0.5*\sin\left(4n\pi T_S\right) \tag{3.75}$$

La trajectoire de référence est définie par :

$$y^*_{(n)} = 0.5*\sin\left(12n\pi T_S\right) \tag{3.76}$$

La figure 3.4 présente les courbes de $\overline{\varepsilon}_{qm}$ obtenues dans la simulation de la loi de commande complète pour ce système et cette référence.

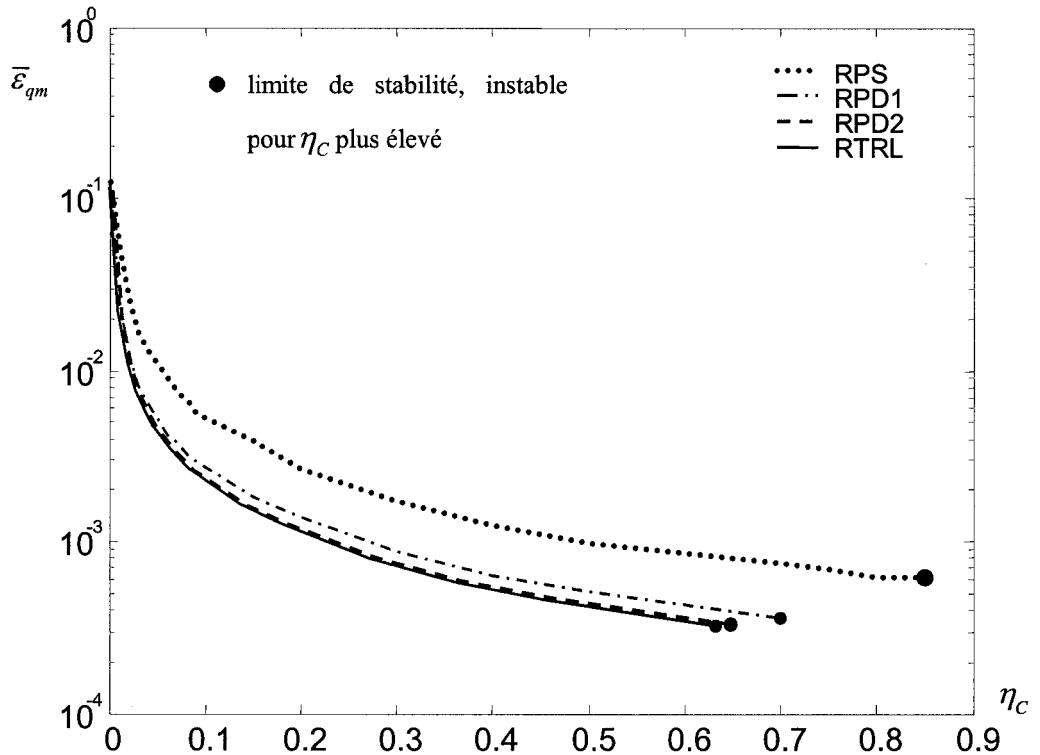


Figure 3.4. Erreur quadratique moyenne pour les quatre algorithmes en fonction de η_C : deuxième système

3.4.2.5. Troisième système

Le troisième système est caractérisé par le modèle NARMAX :

$$y_{(n+1)} = 0.2u_{(n)} + 0.4\sin\left(u_{(n-1)}\right) + 0.31u_{(n-2)} - 0.2u_{(n-2)}u_{(n-3)}$$
$$-1.85y_{(n)} - 0.145y_{(n-1)} + 1.34y_{(n-2)} + 0.63y_{(n-3)} \tag{3.77}$$

La fonction pseudo inverse peut être facilement déduite dans ce cas :

$$u_{(n)} = 5*\left(y^{*}_{(n+1)} - 0.4\sin\left(u_{(n-1)}\right) - 0.31u_{(n-2)} + 0.2u_{(n-2)}u_{(n-3)}\right)$$
$$+5*\left(1.85y_{(n)} + 0.145y_{(n-1)} - 1.34y_{(n-2)} - 0.63y_{(n-3)}\right) \quad (3.78)$$

La phase préalable d'apprentissage du réseau N_I est réalisée en appliquant au système le signal

$$u_{(n)} = \sin\left(4n\pi T_S\right) \quad (3.79)$$

La trajectoire de référence est définie par :

$$y^{*}_{(n)} = 0.6*\sin\left(4n\pi T_S\right) - 0.4*\sin\left(12n\pi T_S\right) \quad (3.80)$$

La figure 3.5 présente les courbes de $\overline{\varepsilon}_{qm}$ obtenues dans la simulation de la loi de commande complète pour ce système et cette référence.

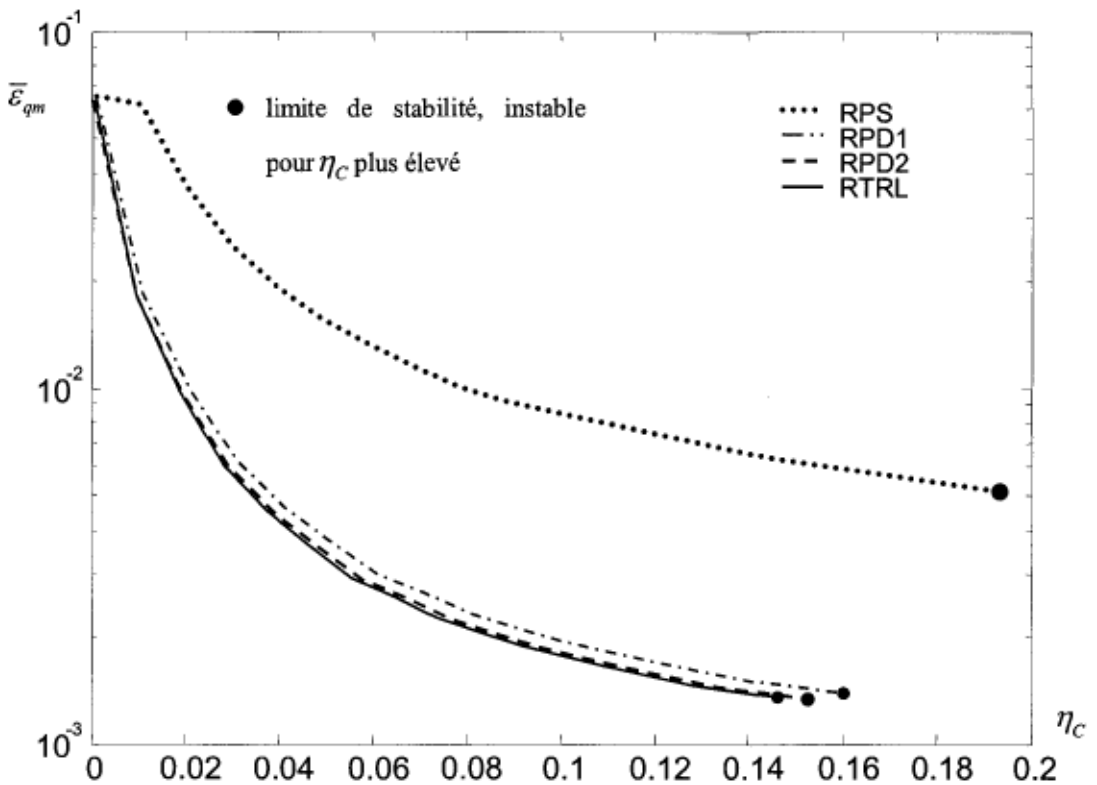


Figure 3.5. Erreur quadratique moyenne pour les quatre algorithmes en fonction de η_C : troisième système

3.4.2.6. Quatrième système

Le quatrième système est caractérisé par le modèle NARMAX :

$$\begin{aligned} y_{(n+1)} &= 0.2u_{(n)} + 0.4\sin\left(u_{(n-1)}\right) + 0.1u_{(n-2)}u_{(n-3)} - 0.5u_{(n-4)} + 0.345u_{(n-5)} \\ &+0.45u_{(n-6)} - 1.63u_{(n-7)} - 0.576y_{(n)} + 1.41y_{(n-1)} + 0.619y_{(n-2)} - 0.587y_{(n-3)} \\ &-0.14y_{(n-4)} + 0.065y_{(n-5)}\left(1 - 0.10\sin\left(y_{(n-5)}\right)\right) - 0.0038y_{(n-6)} - 0.0001y_{(n-7)} \end{aligned} \quad (3.81)$$

La fonction pseudo inverse peut être facilement déduite dans ce cas :

$$\begin{aligned} u_{(n)} &= 5*\left(y^*_{(n+1)} - 0.4\sin\left(u_{(n-1)}\right) - 0.1u_{(n-2)}u_{(n-3)} + 0.5u_{(n-4)} - 0.345u_{(n-5)}\right) \\ &+5*\left(-0.45u_{(n-6)} + 1.63u_{(n-7)} + 0.576y_{(n)} - 1.41y_{(n-1)} - 0.619y_{(n-2)} + 0.587y_{(n-3)}\right) \\ &+5*\left(0.14y_{(n-4)} - 0.065y_{(n-5)}\left(1 - 0.10\sin\left(y_{(n-5)}\right)\right) + 0.0038y_{(n-6)} + 0.0001y_{(n-7)}\right) \end{aligned} \quad (3.82)$$

La phase préalable d'apprentissage du réseau N_I est réalisée en appliquant au système le signal :

$$u_{(n)} = \sin\left(4n\pi T_S\right) \quad (3.83)$$

La trajectoire de référence est définie par :

$$y^*_{(n)} = 0.5*\sin\left(12n\pi T_S\right) \quad (3.84)$$

La figure 3.6 présente les erreurs quadratiques moyennes obtenues dans la simulation de la loi de commande complète pour ce système et cette référence.

3.4.2.7. Commande d'un moteur à courant continu avec frottements non linéaires

À titre d'application concrète de la loi de commande développée dans la thèse, considérons le moteur à courant continu dont le modèle schématique est présenté dans la figure 3.7.

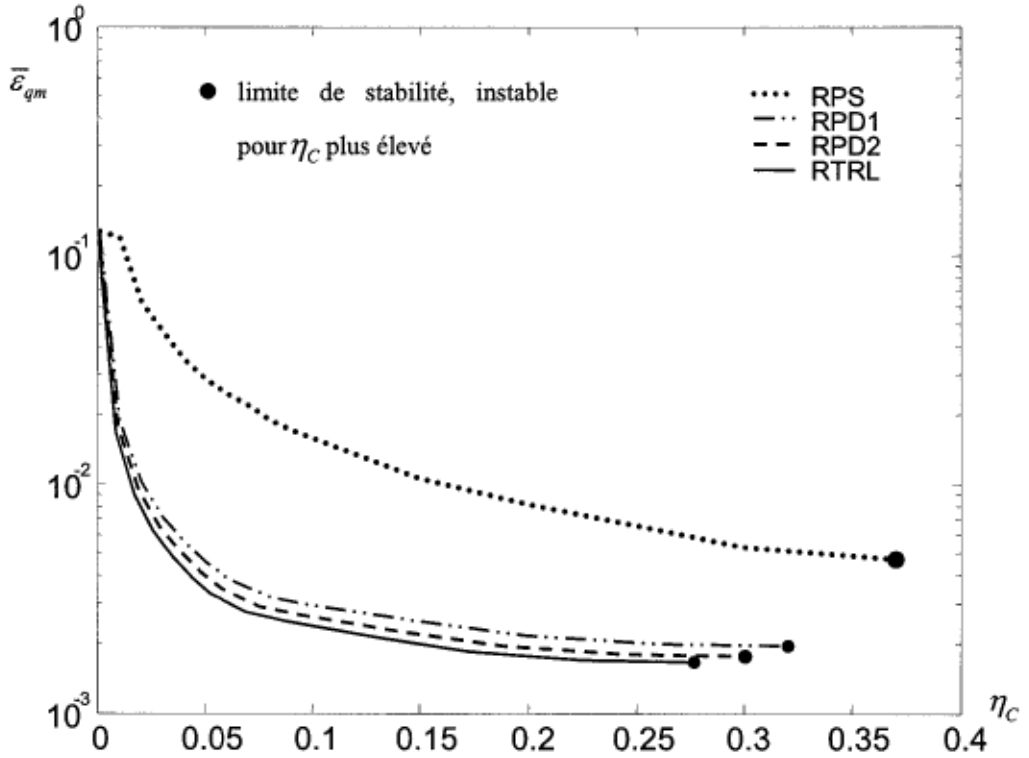
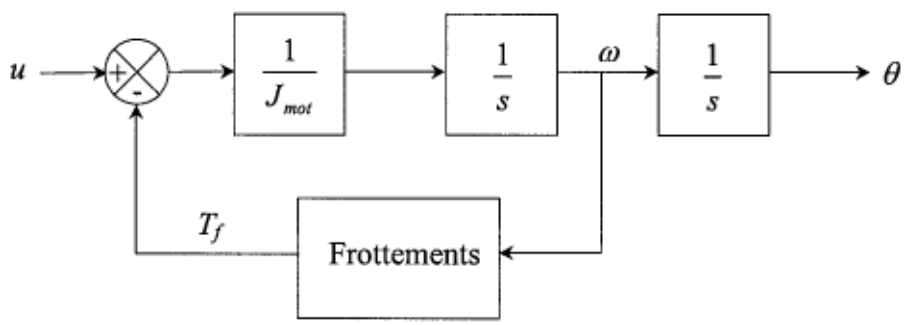


Figure 3.6. Erreur quadratique moyenne pour les quatre algorithmes en fonction de η_C : quatrième système



J_{mot} - coefficient d'inertie du moteur

s - opérateur de Laplace

ω - vitesse angulaire

θ - position angulaire

u - commande du moteur

T_f - couple total de frottements

Figure 3.7. Modèle schématique du moteur à courant continu avec frottements non linéaires

Le modèle analytique est décrit par les équations suivantes [Lesueur '99 ; Armstrong & Canudas de Wit '96] :

$$\theta = \frac{1}{J_{mot} s^2}\left(u - T_f\right) \tag{3.85}$$

$$T_f = \begin{cases} F_c sign(\omega) + F_v \omega + F_s sign(\omega) \exp\left(-\omega^2 / \omega_s^2\right) & \text{si} \quad |\omega| \geq \varepsilon \\ K\omega & \text{si} \quad |\omega| < \varepsilon \end{cases} \tag{3.86}$$

$$K = \frac{1}{\varepsilon}\left(F_c + F_v \varepsilon + F_s \exp\left(-\varepsilon^2 / \omega_s^2\right)\right) \; ; \; 0 < \varepsilon \ll 1 \tag{3.87}$$

F_c, F_v et F_s sont les coefficients de frottement de Coulomb, visqueux et statique respectivement et ω_s est la vitesse caractéristique de frottement statique.

Le modèle NARMAX discret du système est supposé de la forme :

$$\theta_{(n+1)} = f\left[u_{(n)}, u_{(n-1)}, \theta_{(n)}, \theta_{(n-1)}\right] \tag{3.88}$$

La phase préalable d'apprentissage du réseau N_I est réalisée en appliquant au système le signal

$$u_{(n)} = 0.4 \sin\left(2n\pi T_S\right) \tag{3.89}$$

Deux trajectoires de référence sont utilisées :

$$\theta^*_{1(n)} = 0.4 \sin\left(4\pi n T_s\right) \tag{3.90}$$

$$\theta^*_{2(n)} = 0.3 \sin\left(2\pi n T_s\right) + 0.2 \sin\left(8\pi n T_s\right) \tag{3.91}$$

Les valeurs de paramètres pour les simulations sont présentées dans le tableau 3.5.

Tableau 3.5. Valeurs des paramètres de simulations du moteur à courant continu avec frottements non linéaires

Paramètre	Valeur	Paramètre	Valeur
F_v (N.m.s.rad^{-1})	0.0491	ε (rad.s^{-1})	0.001
F_s (N.m)	0.025	J_{mot} (N.m.rad^{-1}.s^2)	3.6*10^{-4}
F_c (N.m)	0.023	ω_s (rad.s^{-1})	0.095

Pour ce système, la marge de stabilité de η_C s'est avérée très réduite pour les deux algorithmes et les deux références. Ceci est très probablement dû à l'instabilité naturelle du système. Nous présentons donc dans les figures 3.8 à 3.11 les meilleurs résultats obtenus avec les deux algorithmes et les deux références, c'est-à-dire les résultats obtenus avec les plus grandes valeurs de η_C avant instabilité. Notons que nous n'avons pas représenté les résultats obtenus avec les algorithmes RPD2 et RTRL. Dans ce cas encore, nous avons obtenu des résultats proches de ceux obtenus avec RPD1. Par conséquent, pour ne pas surcharger les figures et conserver une bonne lisibilité, nous n'avons pas inclus ces résultats.

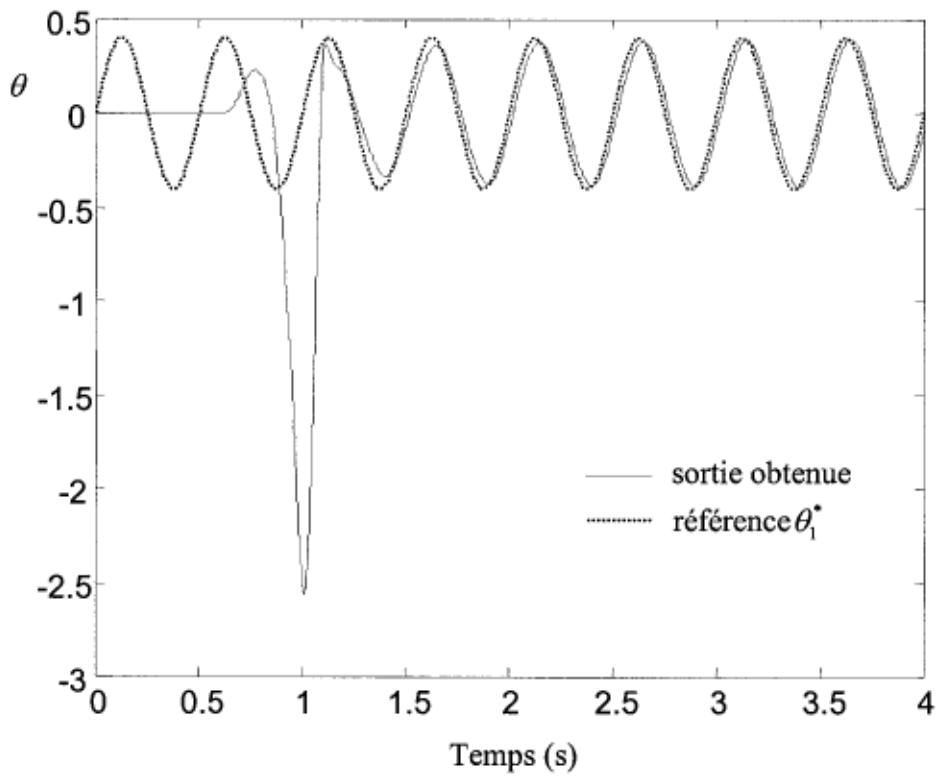


Figure 3.8. Sortie obtenue avec l'algorithme RPS pour la référence θ_1^* ($\eta_C = 0.016$) ; moteur CC avec frottements

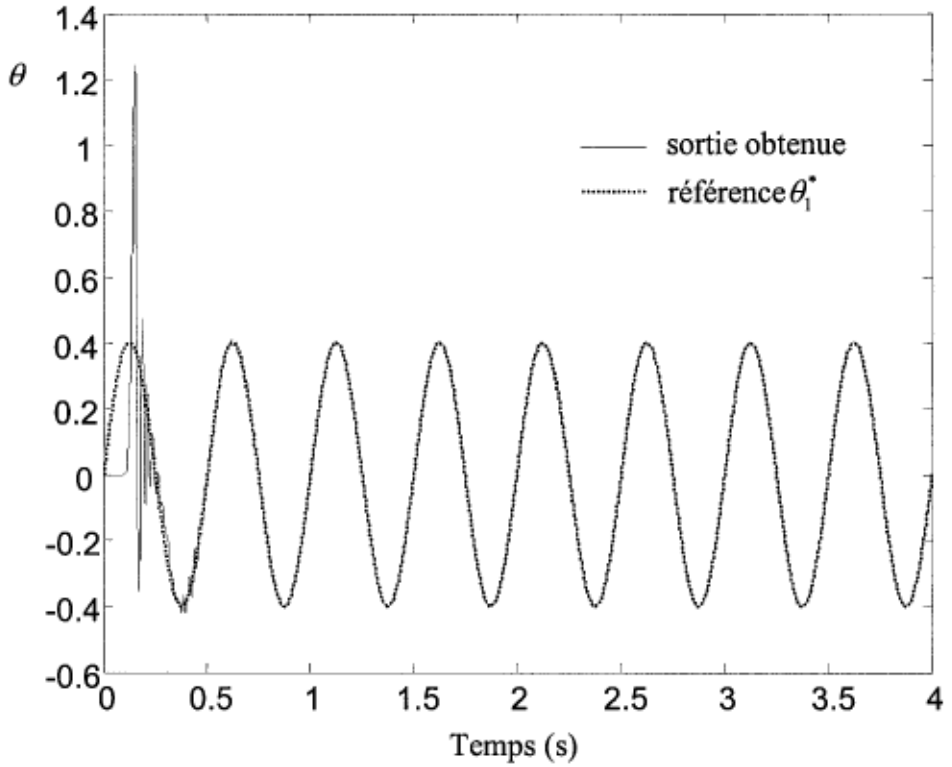


Figure 3.9. Sortie obtenue avec l'algorithme RPD1 pour la référence θ_1^* ($\eta_C = 0.015$) ; moteur CC avec frottements

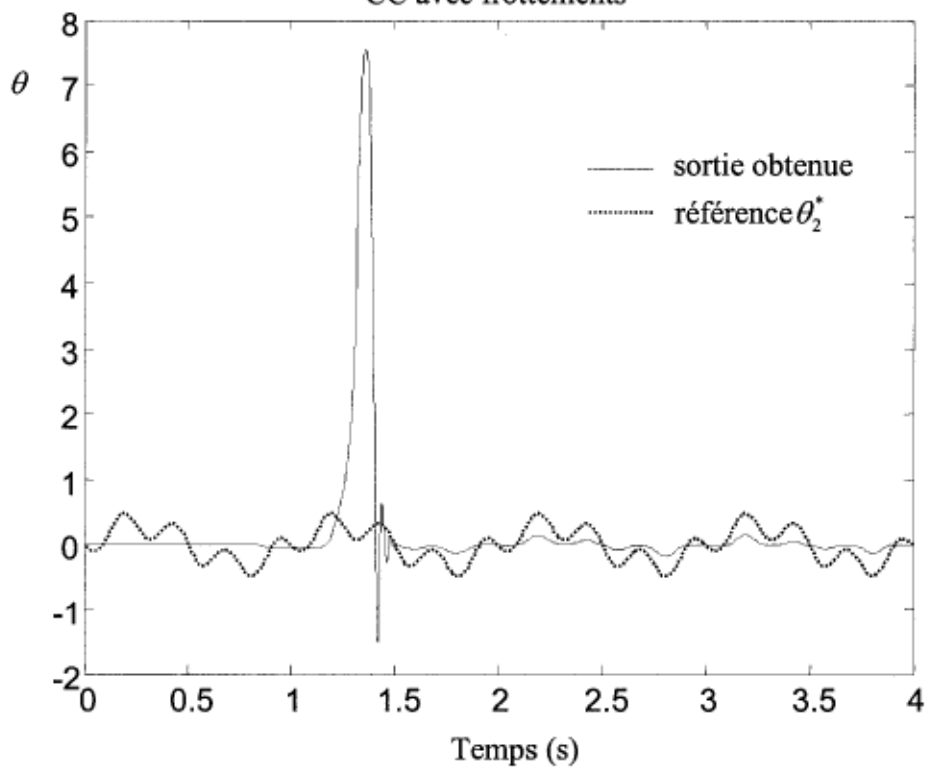


Figure 3.10. Sortie obtenue avec l'algorithme RPS pour la référence θ_2^* ($\eta_C = 0.013$) ; moteur CC avec frottements

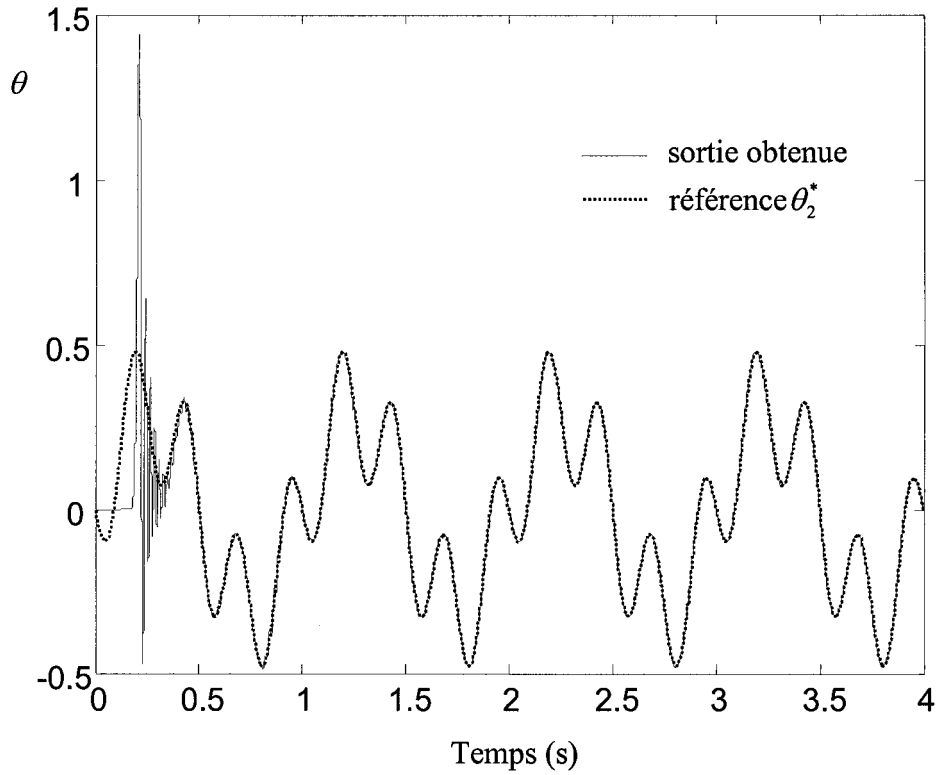


Figure 3.11. Sortie obtenue avec l'algorithme RPD1 pour la référence θ_2^* ($\eta_C = 0.015$) ; moteur CC avec frottements

3.4.2.8. Discussion

D'une manière générale, les résultats présentés dans les figures 3.3 à 3.11 démontrent une diminution de l'erreur quadratique moyenne avec les algorithmes basés sur la rétro propagation dynamique. Dans chaque cas, nous pouvons aussi constater que les courbes obtenues pour RPD2 et RTRL sont très proches l'une de l'autre. L'écart entre RPS et RPD1 est en général très significatif, un peu moins important pour le deuxième système. Le cas du moteur continu avec frottements non linéaires est très intéressant et met en évidence une grande différence de comportement des algorithmes RPS et RPD1. En effet, pour la deuxième référence (*c.f.* figure 3.10), l'algorithme RPS ne parvient pas à suivre la trajectoire de référence. L'augmentation du taux d'apprentissage pour accélérer sa convergence rend très vite l'algorithme instable. De plus, les résultats obtenus avec la première référence montrent que même après convergence, une erreur stationnaire très visible subsiste avec l'algorithme RPS (*cf.* figure 3.8).

L'amélioration apportée par les algorithmes basés sur la rétro propagation dynamique confirme la meilleure prise en compte des termes dynamiques, comme cela avait été prévu dans la section 3.2.3. L'écart entre RPD1 et RPD2 (ou RTRL) est visible mais nettement moins

significatif dans les courbes présentées. Le tableau 3.6 présente les valeurs minimales de $\overline{\varepsilon}_{qm}$ obtenues pour les quatre systèmes synthétiques avec les quatre algorithmes.

Tableau 3.6. Erreurs quadratiques moyennes minimales obtenues pour les quatre systèmes synthétiques avec les quatre algorithmes

Algorithme / Système	Système 1	Système 2	Système 2	Système 4
RPS	$1.35*10^{-4}$	$8.41*10^{-4}$	$5.01*10^{-3}$	$4.66*10^{-3}$
RPD1	$1.85*10^{-5}$	$4.34*10^{-4}$	$1.41*10^{-3}$	$1.95*10^{-3}$
RPD2	$1.73*10^{-5}$	$4.01*10^{-4}$	$1.35*10^{-3}$	$1.83*10^{-3}$
RTRL	$1.75*10^{-5}$	$3.98*10^{-4}$	$1.37*10^{-3}$	$1.77*10^{-3}$

Un avantage important des algorithmes basés sur la rétro propagation dynamique par rapport à l'algorithme basés sur la rétro propagation statique est d'obtenir de meilleurs résultats pour des valeurs plus faible de η_C. L'algorithme de rétro propagation réalise une approximation de la trajectoire dans l'espace des poids synaptiques calculée par la méthode du gradient descendant. Plus le taux d'apprentissage est faible, plus les changements opérés sur les poids synaptiques d'une itération à l'autre sont petits, ce qui a pour conséquence d'obtenir une trajectoire plus lisse. En contrepartie, des changements plus petits sur les poids synaptiques d'une itération à l'autre signifient une convergence plus lente de l'algorithme puisqu'il faut un plus grand nombre d'itérations pour atteindre des valeurs optimales de poids. Ceci est un inconvénient majeur bien connu de l'algorithme de rétro propagation conventionnel. Par le développement algorithmique présenté dans cette thèse, nous démontrons qu'il est possible de développer des algorithmes permettant d'atteindre des vitesses de convergence plus élevées en conservant de faibles valeurs de η_C. Cette amélioration peut être comparée de ce point de vue à l'ajout du terme de *momentum* dans les équations de rétro propagation. Rappelons cependant que la méthode d'ajout du *momentum* est beaucoup plus coûteuse en termes de complexité de calcul et de mémoire requise que l'algorithme RPD1 proposé dans cette thèse.

Les courbes des figures 3.3 à 3.6 sont présentées dans l'entière zone de stabilité des algorithmes, c'est-à-dire pour toutes les valeurs de η_C qui n'aboutissent pas à la divergence de la

loi de commande. Il est donc important de noter que la zone de stabilité de méthode RPS est plus large que celles des trois autres algorithmes. Ceci semble à première vue être un avantage de la méthode RPS. Cependant, la différence de largeur de la zone de stabilité n'est pas très grande et l'amélioration des résultats avec l'algorithme RPS dans la zone de stabilité supplémentaire est souvent négligeable. De plus, dans le cas du moteur à courant continu avec frottements non linéaires, nous avons pu constater que les zones de stabilité des deux algorithmes RPS et RPD1 étaient quasiment identiques. Dans une application pratique, le taux d'apprentissage est généralement choisi constant. Il suffira alors de choisir une valeur du taux bien située dans la zone de stabilité.

Le tableau 3.7 présente les temps de stabilisation à 5% des quatre systèmes synthétiques utilisés dans les sections 3.4.2.3 à 3.4.2.6 pour une entrée de type échelon unitaire. Ceci nous permet de constater que la loi de commande développée permet de commander, avec une fréquence d'échantillonnage de 100KHz, une gamme de systèmes allant de systèmes moyennement rapides, c'est-à-dire ayant des temps de stabilisation de 5 à 10ms, jusqu'à des systèmes plus rapides, c'est-à-dire ayant des temps de stabilisation de quelques dixièmes de milliseconde. La gamme des systèmes ayant un temps de stabilisation de quelques dixièmes de milliseconde est très intéressante puisqu'elle correspond à des systèmes de hautes performances tels que les moteurs miniatures utilisés dans un grand nombre d'applications modernes.

Tableau 3.7. Temps de stabilisation à 5% des quatre systèmes synthétiques utilisés dans les sections 3.4.2.3 à 3.4.2.6 pour une entrée de type échelon unitaire

Système (section)	Temps de stabilisation à 5% (ms)
3.4.2.3	0.7
3.4.2.4	0.5
3.4.2.5	6.8
3.4.2.6	5.5

3.4.3. Évaluation des critères d'ITGE

3.4.3.1. Méthode de comparaison

Étant donné que les différences entre les algorithmes ne portent que sur l'adaptation des réseaux, l'étude comparative des critères d'ITGE peut être limitée aux équations d'adaptation. Dans cette section, la comparaison est surtout basée sur trois critères : le type d'opérations mathématiques mises en jeu, la quantité de calculs (nombre d'opérations arithmétiques) et le nombre de mémoires nécessaires.

3.4.3.2. Types d'opérations arithmétiques

En observant les différentes équations définissant les quatre algorithmes présentés, les opérations arithmétiques nécessaires sont :

- additions,
- multiplications,
- soustractions,
- tangente hyperbolique.

Ce sont les opérations arithmétiques de base considérées les plus simples à intégrer, à l'exception de la fonction hyperbolique, et cela donne un autre aspect très attrayant des RNA. Les quatre algorithmes utilisent les mêmes types d'opérations arithmétiques. Il n'y a donc pas de différence de ce point de vue.

3.4.3.3. Complexité de calcul

La complexité de calcul désigne ici la quantité d'opérations arithmétiques de base nécessaire à chacun des algorithmes. Cette quantité dépend non seulement de l'algorithme choisi mais également du nombre d'entrées du réseau (calculé à partir des paramètres a et b) et du nombre de poids des réseaux (calculé à partir des paramètres M et N). Le tableau 3.8 présente le nombre d'additions et de multiplications nécessaires pour les quatre algorithmes en fonction des quatre paramètres a, b, M et N.

Tableau 3.8. Complexité de calcul des quatre algorithmes d'adaptation présentés : nombre de multiplications et d'additions en fonction des paramètres a, b, M et N

Algorithme	Multiplications	Additions
RPS	$(M+N)(5a+5b+7)+3M+1$	$(M+N)(a+b+1)+M$
RPD1	$(M+N)(5a+5b+7)+3M+1$	$(M+N)(a+b+1)+M(a+1)-1$
RPD2	$1+a(2M+1)+(a-1)^2 M+$ $(a-1)(aN+1)+(M+N)(5a+5b+7)$	$(a-1)+N(a-1)(3+N)+$ $a^2M+3aM+(M+N)(a+b+1)$
RTRL	$5M(a+b+2)+1+2N+N^2(a+2)$ $+N(a+b+1)(aN+3N+2)$	$M(a+b+1)+2M-1+N(a-1)$ $+2N^2+N^2(a+b+1)(a+2)$

Les résultats du tableau sont obtenus en considérant la dérivée de la fonction hyperbolique réalisée par :

$$\frac{d\tanh(z)}{dz}=1-\tanh^2(z) \tag{3.92}$$

Les fonctions tangentes hyperboliques étant déjà réalisées dans la partie de propagation directe des réseaux, le calcul de la dérivée nécessite donc une soustraction (considérée comme une addition) et une multiplication pour la mise au carré.

Le tableau 3.9 présente les évaluations numériques pour les quatre systèmes des sections 3.4.2.3 à 3.4.2.6.

Tableau 3.9. Évaluation numérique des nombres de multiplications et additions pour les quatre systèmes des sections 3.4.2.3 à 3.4.2.6

Algorithme	Additions				Multiplications			
	Syst. 1	Syst. 2	Syst. 3	Syst. 4	Syst. 1	Syst. 2	Syst. 3	Syst. 4
RPS	110	156	342	1190	571	805	1747	6019
RPD1	129	179	413	1461	571	805	1747	6019
RPD2	331	445	1965	12961	614	856	2222	10046
RTRL	2279	3851	18395	199715	3321	5485	23617	232017

Bien sûr, les chiffres présentés dépendent beaucoup des valeurs des quatre paramètres a, b, M et N. Une comparaison des complexités de calcul en fonction des quatre paramètres serait exhaustive et non nécessaire. En effet, plusieurs éléments peuvent nous amener à ne considérer qu'un seul paramètre.

Tout d'abord, en vue d'une application pratique, les nombres de délais sur l'entrée et la sortie (a et b) doivent être au moins égaux aux valeurs définissant le modèle NARMAX du système. Remarquons que rien n'oblige un système réel à avoir un modèle où ces deux paramètres sont égaux. Dans le cas général, nous considérerons donc que les réseaux sont dimensionnés pour avoir des nombres de délais suffisamment grands et par simplicité nous choisirons a et b égaux. Il faudra donc choisir cette valeur commune au moins égale au maximum des nombres de délais a et b des modèles envisagés.

Considérons maintenant les paramètres M et N. Il est difficile voire impossible de déterminer de façon analytique le nombre de poids optimal pour chaque réseau. Cependant, l'expérience de l'auteur des nombreuses simulations réalisées ainsi que les considérations empiriques bien connues de la littérature nous ont amené à choisir un nombre de neurones égal à deux fois le nombre d'entrées total du réseau. Finalement, les deux dernières considérations nous amènent à choisir :

$$\begin{cases} a = b \\ M = N = 2(a+b+1) = 4a+2 \end{cases} \tag{3.93}$$

Dans ce cas, les résultats du tableau 3.8 peuvent être réécrits en fonction d'un seul des quatre paramètres, par exemple a, comme présenté dans le tableau 3.10.

Tableau 3.10. Complexité de calcul pour les quatre algorithmes présentés en fonction du paramètre a ($M = N = 4a+2$ et $a = b$)

Algorithme	Multiplications	Additions
RPS	$80a^2+108a+35$	$16a^2+20a+6$
RPD1	$80a^2+108a+35$	$20a^2+22a+5$
RPD2	$8a^3+80a^2+100a+30$	$20a^3+42a^2+5a-7$
RTRL	$32a^4+160a^3+272a^2+196a+49$	$32a^4+112a^3+164a+98a+19$

Les figures 3.12 et 3.13 présentent les courbes correspondant aux résultats du tableau 3.10. L'axe des ordonnées est logarithmique pour une meilleure visibilité. Ces courbes nous permettent de visualiser l'évolution des écarts de complexité en fonction du paramètre a.

3.4.3.4. Autres critères

Un autre critère à considérer dans l'ITGE d'algorithmes de traitement de signal est la quantité de mémoire requise. En effet, certains algorithmes nécessitent la mémorisation de données pour les calculs, en particulier dans les équations récurrentes. Dans notre cas, les poids synaptiques de l'instant n doivent être mémorisés jusqu'à l'instant $n+1$ pour pouvoir adapter les poids à cet instant. En ce qui concerne la comparaison des quatre algorithmes, la quantité de poids synaptiques à mémoriser est la même. Par contre, les algorithmes RPD2 et RTRL nécessitent des mémoires supplémentaires non nécessaires dans RPS et RPD1. Le tableau 3.11 présente les quantités de mémoires supplémentaires nécessaires pour ces deux algorithmes en fonction du paramètres a dans le cas particulier décrit par l'équation (3.93).

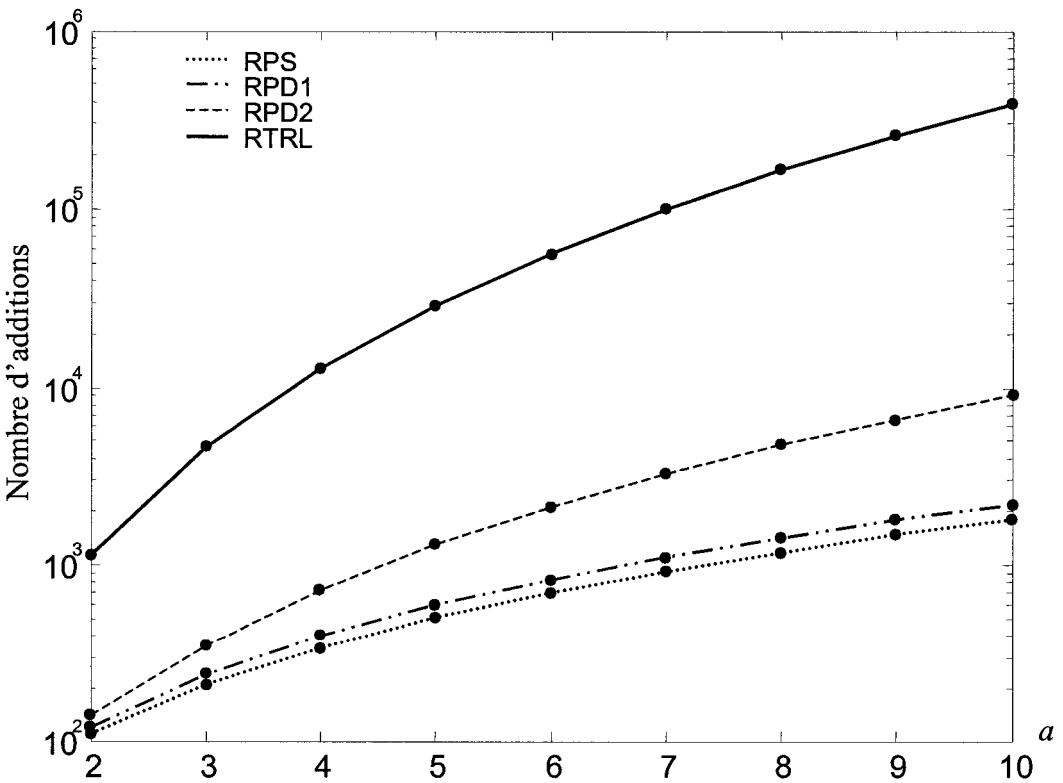


Figure 3.12. Courbe de complexité de calcul des quatre algorithmes en fonction du paramètre a : nombre d'additions

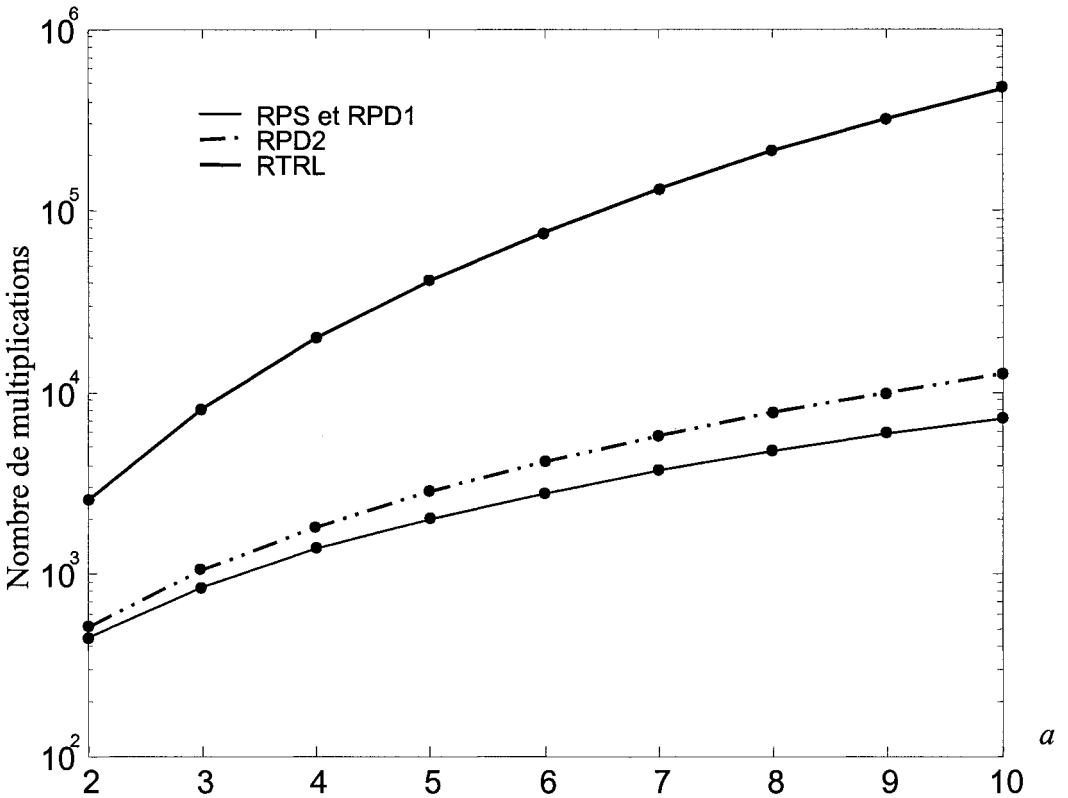


Figure 3.13. Courbe de complexité de calcul des quatre algorithmes en fonction du paramètre a : nombre de multiplications

Tableau 3.11. Nombre de mémoires supplémentaires par rapport à RPS pour les algorithmes RPD2 et RTRL en fonction du paramètre a

Algorithme	Mémoires supplémentaires
RPD2	$2a^2 - a$
RTRL	$8a^3 + 6a^2 - 9a - 4$

Tous les algorithmes présentés présentent une rétroaction de la sortie vers l'entrée. La récursivité entraîne une plus grande sensibilité des algorithmes vis-à-vis des imperfections dans les opérations mathématiques. L'algorithme de rétro propagation conventionnel (RPS) est notamment reconnu pour sa sensibilité aux tensions de décalage (*angl. offset*) dans les intégrations analogiques ou aux effets de quantification sur les poids synaptiques dans les intégrations numériques [Montalvo *et al.* '92 ; Withagen '94]. La présence de ces tensions de décalage peut créer une accumulation dans la rétroaction et aboutir à l'instabilité.

Nous devrons donc nous assurer de maintenir les tensions de décalage à des niveaux suffisamment bas pour éviter ce problème.

Cette fois encore, les algorithmes RPD2 et RTRL se distinguent par des récursivités supplémentaires dans le calcul des termes π, σ et ∇u. Ceci ajoute donc une sensibilité supplémentaire aux tensions de décalage ou auxeffets de quantification au niveau des signaux représentant ces dernières variables.

3.4.3.5. Discussion

Les tableaux 3.7 à 3.10 nous ont permis de comparer les quatre algorithmes en termes de complexité de calcul. Cette fois, la tendance d'amélioration est opposée puisque plus nous allons de RPS vers RTRL, plus la complexité de calcul est élevée. Notons tout de suite la marge très importante entre RTRL et les trois autres algorithmes. Étant donné que RPD2 offre des résultats très proches pour une complexité nettement inférieure, nous excluons d'emblée RTRL pour la suite du développement. Cette conclusion quant à la complexité de RTRL est d'ailleurs mentionnée par d'autres auteurs [Lehmann '94]. Les tableaux 3.7 à 3.10 et les figures 3.12 et 3.13 nous ont également montré comment évolue la complexité de calcul des quatre algorithmes en fonction des paramètres *a*, *b*, *M* et *N*. Portons une attention particulière au tableau 3.10 qui nous permet de voir plus facilement les différences entre les quatre algorithmes. Nous constatons que les méthodes RPD2 et RTRL ont une complexité de $O\left(a^3\right)$ et $O\left(a^4\right)$ respectivement alors que les deux autres ont une complexité de $O\left(a^2\right)$. Les différences d'ordres expliquent pourquoi les courbes s'écartent très vite dans les figures 3.12 et 3.13 lorsque *a* augmente. Ces figures nous permettent également de constater que les courbes de RPS et RPD1 sont très proches même lorsque $a = 10$.

Les deux méthodes RPD2 et RTRL nécessitent des mémoires supplémentaires comme cela a été présenté dans le tableau 3.11. Ceci est une contrainte à l'ITGE puisque ces mémoires supplémentaires impliquent des circuits supplémentaires pour les réaliser et donc une surface totale d'intégration accrue.

3.5. Conclusion

Dans ce chapitre, nous avons présenté quatre algorithmes pour l'adaptation en ligne des deux RNA mis en œuvre dans le schéma de commande global. Les résultats de simulations nous ont permis d'évaluer comparativement les quatre algorithmes selon les critères algorithmiques et d'ITGE. Pour clore ce chapitre, nous devons conclure quant à l'algorithme (ou les algorithmes) à retenir pour l'ITGE. Nous avions exclu RTRL d'emblée lors de l'étude des critères d'ITGE, il ne reste donc que trois choix. Si nous comparons les courbes d'erreurs obtenues pour RPD1 et RPD2 et que, d'un autre côté, nous regardons la complexité de calcul de ces deux algorithmes, la différence de qualité ne justifie pas la différence de complexité. De plus, RPD2 nécessite des mémoires supplémentaires qui augmenteraient la surface d'intégration de l'ensemble. Nous retenons donc RPD1 préférablement à RPD2. Notons également que d'un point de vue architectural, les équations des algorithmes RPD2 et RTRL sont beaucoup plus complexes. Il serait donc beaucoup plus difficile de développer une architecture pour l'intégration de l'un ou l'autre de ces algorithmes. Quant à l'algorithme RPS, il est très intéressant pour l'ITGE mais la qualité de commande est moindre par rapport à RPD1 comme nous l'avons vu dans les figures 3.3 à 3.11.

Finalement, nous retenons l'algorithme RPD1 pour l'intégration. Cet algorithme représente le meilleur compromis entre les critères de commande et les critères d'intégration sur silicium. Le développement algorithmique aboutit à un algorithme qui, contrairement aux algorithmes de référence de la littérature, permet une amélioration significative des résultats avec une complexité de $O(a^2)$, ce qui satisfait nos objectifs établis dans la section 3.2.3. Nous avons également pu constater que la loi de commande, avec une fréquence d'échantillonnage de 100KHz, permet de commander des systèmes rapides ayant des temps de stabilisation à 5% de quelques dixièmes de milliseconde. Ceci démontre son applicabilité à une large gamme de systèmes de hautes performances. Notons que la différence entre RPS et RPD1 se résume à l'ajout d'une somme dans le calcul de e_u. Les circuits développés pour l'ITGE de RPD1 pourraient donc facilement être utilisés pour réaliser, en un minimum de temps, l'ITGE de RPS.

Chapitre 4

Modélisation d'ITGE de la loi de commande

Dans ce chapitre, les architectures et circuits CMOS pour la modélisation, au niveau de l'ITGE, de la loi de commande développée dans les deuxième et troisième chapitres seront présentés. Après justification des différents choix technologiques d'intégration, les modèles des circuits réalisant les opérations arithmétiques de base de la loi de commande seront présentés. Ces circuits de base seront ensuite assemblés dans des architectures hautement parallèles pour la modélisation des blocs fonctionnels principaux de la loi de commande. Ces blocs permettront finalement la modélisation et la simulation au niveau de l'ITGE de la loi de commande complète présentée dans cette thèse, incluant l'algorithme d'adaptation RPD1 retenu pour cette étape.

4.1. Introduction : pourquoi des circuits dédiés analogiques ?

Les RNA ainsi qu'un grand nombre d'algorithmes de traitement de signal peuvent être implémentés dans des circuits numériques tels que les DSP et autres microprocesseurs. Ces derniers bénéficient de recherches intensives des manufacturiers et voient leurs architectures et leurs performances évoluer très rapidement. Dans les applications scientifiques du traitement de signal, les DSP connaissent un succès grandissant grâce à leurs performances toujours croissantes

et à de nouvelles architectures de traitement à virgule flottante permettant des calculs à la fois rapides et de grande précision.

Dans un tel contexte, il est intéressant de se demander pourquoi un bon nombre de chercheurs continuent à s'intéresser au développement d'ASIC. Comment penser que de tels circuits, surtout dans le cas des ASIC analogiques, peuvent rivaliser avec les DSP actuels ? En réalité, dans le cas général, ils ne peuvent pas rivaliser. Il existe toutefois certains avantages des ASIC, qu'ils soient analogiques ou non, qui peuvent être mis à profit dans certaines applications particulières et qui motivent le travail des chercheurs dans ce domaine. Les principaux avantages peuvent être trouvés dans l'optimisation de critères tels que le parallélisme, la consommation de puissance et la surface d'intégration.

4.1.1. Parallélisme

En quête de traitement d'information toujours plus rapide, plusieurs approches sont possibles.

Une première approche consiste à utiliser des processeurs plus rapides. Les processeurs commerciaux tels que les DSP ont une architecture de type Von Neumann. Cette architecture est basée sur une unité de traitement très rapide qui traite les instructions de manière séquentielle. La croissance de vitesse de ces processeurs passe par l'augmentation de la fréquence d'horloge et approche aujourd'hui des limites technologiques comme nous l'avions mentionné dans la section 1.1.1 du premier chapitre.

Dans l'attente d'une éventuelle nouvelle technologie d'intégration, une deuxième approche permettant des améliorations de performances significatives est le parallélisme. En répartissant les opérations d'un algorithme sur plusieurs processeurs travaillant simultanément, il est possible d'obtenir de grandes améliorations soit en terme de temps d'exécution total de l'algorithme soit en terme de débit (c'est-à-dire le nombre de fois que l'algorithme peut être exécuté par unité de temps) [Guyennet '01]. D'ailleurs, les constructeurs de DSP eux-mêmes s'orientent vers cette solution puisque les derniers DSP sur le marché intègrent plusieurs unités de calcul capables d'exécuter plusieurs instructions en parallèle. Cependant, la répartition des opérations peut être difficile et tous les algorithmes ne peuvent être parallélisés à un degré intéressant.

De par leur structure intrinsèquement parallèle, les RNA sont de bons candidats à ce type d'implémentation. Le développement d'ASIC à architectures parallèles, analogiques ou numériques, permet alors de réaliser des intégrations très efficaces. Même si, en général, les circuits de base analogiques sont plus lents que leurs homologues numériques, ils peuvent parfois être intégrés sur de plus petites surfaces. Ainsi, des systèmes massivement parallèles peuvent être intégrés efficacement dans des circuits analogiques et le parallélisme donne un potentiel de calcul très rapide.

Bien sûr, pour des calculs de très grande précision, il est préférable d'utiliser des circuits numériques, notamment des circuits opérant au format à virgule flottante. Dans notre cas, nous ne cherchons pas la plus grande précision mais une précision suffisante nous permettant de maintenir les résultats algorithmiques présentés dans le troisième chapitre. Plusieurs auteurs ont d'ailleurs établi que la précision offerte par des circuits analogiques est suffisante pour la plupart des systèmes à RNA [Hollis & Paulos '90 ; Tarassenko *et al.* '93].

4.1.2. Consommation de puissance

Pendant longtemps, les développements des circuits intégrés ont été concentrés vers l'augmentation des vitesses de traitement des microprocesseurs, la consommation de puissance étant un sujet de peu d'intérêt. Comme nous l'avions mentionné dans la section 1.1.1 du premier chapitre, la progression de la technologie d'intégration sur silicium selon le procédé CMOS a permis une évolution rapide de la vitesse d'horloge des microprocesseurs. Du même coup, la progression de la technologie a engendré une augmentation importante de la densité d'intégration, souvent exprimée en nombre de transistors par unité de surface d'une puce. Un circuit ayant une vitesse de fonctionnement plus élevée et une densité d'intégration plus élevée consomme plus de puissance. Même si cet aspect a été négligé pendant des années, la consommation de puissance sans cesse croissante des circuits est progressivement devenue un problème. Aujourd'hui, l'équipement informatique et électronique représente une partie importante de la consommation de puissance des pays développés. Cette part représente par exemple 5 à 10% pour les États-Unis et continue de s'accroître avec l'augmentation des exigences de l'informatique.

La réduction de la consommation de puissance a plusieurs raisons d'être :

- la demande pour des systèmes autonomes portables (ordinateur, téléphone, appareil pour l'audition, stimulateur cardiaque, etc.) est en hausse,
- la capacité de stockage d'énergie dans une pile est limitée, limitant donc la durée d'autonomie des systèmes portables,
- les dispositifs de refroidissement des circuits engendrent des coûts et des encombrements d'espace importants,
- la haute puissance engendre des défauts dans les composants,
- la réduction de consommation d'énergie est nécessaire à la protection de l'environnement.

D'une manière générale, la consommation de puissance dépend des voltages et des courants présents dans un circuit [Sanchez-Sinencio & Andreou '99]. L'idée la plus importante est alors de tenter de diminuer ces voltages et ces courants afin de diminuer la puissance totale consommée par le circuit. Plus précisément, les principales sources de consommation de puissance dans un circuit intégré CMOS sont :

- les charges et les décharges de capacités parasites pendant les commutations des transistors,
- les faibles impédances entre les bornes positives et négatives de l'alimentation pendant les commutations,
- les courants de fuite dus aux jonctions polarisées en inverse et à la conduction en dessous du seuil des transistors,
- la consommation statique.

La consommation totale d'un circuit peut être exprimée mathématiquement en fonction de plusieurs variables, notamment la fréquence d'horloge et la tension d'alimentation. Réduire la consommation totale équivaut alors à diminuer l'un ou l'autre de ces facteurs. La réduction de fréquence n'étant généralement pas souhaitée, les microprocesseurs ont vu leur tension d'alimentation baisser jusqu'à 1.8 V et on parle déjà de nouvelles générations fonctionnant autour de 1V. Bien sûr, des niveaux de tensions si bas posent d'importants problèmes de conception, notamment en raison des tensions de seuil des transistors. Cet axe de développement atteint lui

aussi des limites incontournables de la technologie CMOS. D'autre part, même avec des voltages faibles, les fréquences d'horloges élevées impliquent des consommations importantes au moment des commutations. Notons qu'avec des circuits analogiques sans horloge comme nous utiliserons par la suite, les différentes consommations dues aux commutations n'existent plus, la consommation statique devenant la plus importante.

4.1.3. Surface d'intégration

Dans certaines applications, notamment pour les systèmes portatifs, la surface d'intégration des circuits est un autre élément crucial. De même que pour la consommation de puissance, la conception d'ASIC permet de créer des circuits spécialement optimisés en ce sens. La réduction de surface d'intégration peut être faite dans deux optiques. La première consiste à créer des circuits de base plus petits afin d'en intégrer le plus possible dans une même puce. Dans notre cas, il s'agirait par exemple de réduire au maximum la surface d'intégration des circuits de neurones, des poids, et leurs connexions afin d'être capable d'intégrer des réseaux ayant de très grands nombres de neurones et de poids. Dans une deuxième optique, il s'agit plutôt d'intégrer un circuit donné dans la plus petite surface possible afin de l'implanter dans un environnement réduit.

En termes de surface d'intégration, il n'est pas toujours facile de comparer les circuits analogiques et numériques. La représentation d'un signal en analogique ne nécessite qu'un fil (ou deux dans la représentation différentielle), tandis qu'en numérique il y aura autant de fils que de bits utilisés pour la représentation. La surface de connexion est alors moindre en analogique. Souvent, pour une même fonction arithmétique, un circuit numérique utilise plus de transistors qu'un circuit analogique. Cependant, les dimensions des transistors utilisés dans les circuits analogiques sont généralement plus grandes. Dans la réalisation de certaines fonctions de calcul, les circuits analogiques sont très avantageux en terme de surface d'intégration. Nous verrons par exemple dans la section 4.4 comment intégrer simplement, à l'aide de six transistors, la fonction tangente hyperbolique en analogique. Pour cet exemple précis, les circuits analogiques offrent un avantage indéniable en terme de surface d'intégration et ceci est généralement le cas pour la réalisation de fonctions non linéaires.

La surface d'intégration d'un circuit a également un impact direct sur le coût de fabrication du circuit. Parmi les coûts relatifs à la fabrication d'un microcircuit, la surface d'intégration est le seul élément sur lequel le concepteur peut intervenir.

Rappelons la formule d'évaluation du coût de fabrication d'un circuit intégré dans un procédé CMOS [Henessy & Patterson '92] :

$$coût\ \ d'un\ \ circuit = \frac{coût\ \ de\ \ la\ \ tranche}{nombres\ \ de\ \ circuits\ \ par\ \ tranche * rendement} \tag{4.1}$$

Il est clair que plus la puce est petite plus le nombre de puces par tranche est élevé et donc le coût du circuit faible. De plus, le rendement de fabrication est inversement proportionnel à la surface du circuit mise à un exposant au moins égal à deux :

$$rendement = \frac{K}{surface\ \ du\ \ circuit^{\alpha}} \tag{4.2}$$

où K est une constante positive et α est une constante positive supérieure ou égale à deux.

Il est finalement possible de démontrer que le coût de fabrication est inversement proportionnel à une puissance au moins égale à trois de la surface du circuit [Henessy & Patterson '92]. La surface du circuit a donc une répercussion très importante sur le coût final de chaque unité produite du circuit.

4.1.4. Limitations

Les circuits dédiés analogiques présentent certains inconvénients importants qui peuvent limiter leur application, notamment sous les contraintes du domaine industriel :

- temps de conception plus long,
- bibliothèques de circuits beaucoup moins riches que pour les circuits numériques dans les logiciels de Conception Assistée par Ordinateur (CAO),
- sensibilité au bruit,

- sensibilité au procédé de fabrication (c'est-à-dire les défauts des circuits dus au procédé même),
- difficulté d'obtenir des circuits de grande précision,
- application limitée à la tâche pour laquelle le circuit a été conçu.

Ainsi, il est évident que ce type de circuit ne peut être destiné à un usage général auquel cas il perdrait tout avantage face à ses homologues numériques. Un ASIC analogiques doit donc demeurer dédié à une application bien spécifique et être optimisé pour cette application en vue d'optimiser un ou plusieurs critères d'ITGE.

4.2. ASIC analogiques pour l'ITGE des RNA : état de la recherche

Au niveau de l'intégration sur silicium des RNA, le développement a commencé depuis le début des années 90. L'approche dominante pour l'intégration était de réaliser un circuit d'ITGE de la partie de propagation directe d'un réseau et de laisser des entrées externes pour les valeurs des poids synaptiques, lesquelles étaient programmées à partir d'un ordinateur. Progressivement, cette solution a laissé place à une autre approche visant à intégrer dans un même circuit la partie de propagation directe et la partie d'adaptation. Cette dernière décennie, pour les raisons invoquées dans la section 4.1, quelques chercheurs ont commencé à relever à un nouveau défi : l'intégration complètement analogique des RNA avec adaptation en ligne. En raison des caractéristiques non idéales des circuits analogiques, l'adaptation en ligne peut permettre dans une certaine mesure de compenser certains défauts de fabrication [Castro *et al.* '93 ; Eberhardt *et al.* '88]. Ceci sera notamment illustré par des résultats d'analyse de sensibilité des circuits présentés dans la section 5.6 du cinquième chapitre. L'ITGE d'algorithmes d'adaptation en ligne pour les RNA est difficile dans le domaine analogique. Dans la plupart des cas, les circuits analogiques concernent la partie de propagation des données dans le réseau tandis que les équations d'adaptation sont implantées dans des circuits numériques connexes [Ghosh *et al.* '94 ; Hollis & Paulos '94]. Cette approche permet de tirer profit de la flexibilité et de la précision des circuits numériques pour le calcul des poids. Par exemple, si un algorithme d'adaptation est programmé dans un DSP, il est facile de modifier l'algorithme ou d'en implémenter un autre en

modifiant simplement le programme du DSP. Cependant, cette approche est contraire à la direction prise dans cette thèse et décrite dans la section 4.1. Dans les paragraphes qui suivent, nous présentons l'état de la recherche par les publications les plus pertinentes concernant l'ITGE complètement analogique avec adaptation en ligne des RNA. Nous avons inclus certaines références qui ne traitent pas de la partie d'adaptation mais dont l'intégration de la partie de propagation est tout de même très intéressante.

Dans [Montalvo *et al.* '97], une approche complètement analogique basée sur le principe de perturbation des poids est proposée. Dans cette méthode, des perturbations de faibles amplitudes sont appliquées sur les poids synaptiques et les influences sur la sortie du réseau sont mesurées. Un poids est alors adapté en fonction de l'influence de la perturbation de ce poids sur l'erreur en sortie. La méthode proposée offre surtout l'avantage d'insensibiliser les circuits d'adaptation aux défauts de fabrication des composants puisque les influences des poids sont directement mesurées dans le circuit. Ceci constitue un avantage intéressant du point de vue de l'ITGE mais les inconvénients au niveau algorithmique présentés dans le troisième chapitre (*cf.* section 3.2.2.2) nous ont amené à ne pas retenir cette méthode. Au niveau de l'ITGE, les auteurs ont fait fabriquer un circuit avec 64 synapses et 8 neurones. Le circuit obtenu offre l'avantage d'être configurable au niveau du nombre de poids et de neurones par couche et consomme relativement peu de puissance. Cependant, les neurones sont linéaires, limitant les applications pour les systèmes non linéaires.

Dans [El-Masry *et al.* '97], une autre approche basée sur les modulations de largeur d'impulsions en mode courant (*angl. Current-Mode Pulse Width Modulation - CM-PWM)* est proposée. La modulation de largeur d'impulsions est utilisée pour représenter les données et réaliser les opérations de base du réseau. Cette approche originale permet une implantation simple et modulaire d'un réseau multicouche mais l'adaptation des poids n'est pas considérée.

Dans [Coué & Wilson '96], un circuit pour l'ITGE analogique d'un réseau multicouche est présenté. Le circuit global est composé de modules élémentaires constitués à partir de circuits de base très simples : multiplieur analogique, sommation en mode courant, etc. Certains des circuits de base ressemblent beaucoup à ceux qui seront employés dans cette thèse. Même si

l'adaptation du réseau n'est pas considérée, les résultats expérimentaux prouvent la faisabilité de l'ensemble et la fonctionnalité des circuits de base du réseau.

Dans [Cauwenberghs '96], la méthode de perturbation des poids est utilisée pour l'intégration d'un réseau multicouche récurrent. Même si nous avons exclu la méthode de perturbation des poids, les résultats expérimentaux du circuit présentés dans cet article démontrent la fonctionnalité d'un réseau récurrent analogique avec apprentissage en ligne pour l'apprentissage de fonctions non linéaires (sinusoïdes). Le circuit obtenu consomme 1.2mW pour 56 synapses et a une surface de $5mm^2$.

Dans [Morie & Amemiya '94], une variante de l'algorithme de rétro propagation appelée *constrastive backpropagation* est proposée afin de limiter la sensibilité de l'algorithme de base aux tensions de décalage (*angl. offset*). Un circuit est présenté et fabriqué pour l'intégration d'un réseau multicouche avec adaptation continue en ligne par l'algorithme proposé. Les neurones sont non linéaires avec fonction d'activation de type tangente hyperbolique. Le circuit obtenu pour 81 poids et 9 neurones consomme 320mW et a une surface de $49mm^2$. Les résultats expérimentaux démontrent la fonctionnalité du circuit. Cependant, l'algorithme est basé sur le principe de perturbation des poids dont les inconvénients ont déjà été mentionnés dans le troisième chapitre.

L'étude bibliographique relative aux ASIC analogiques pour l'ITGE des RNA permet finalement de constater que :

- l'intégration de la partie d'adaptation en ligne est peu réalisée voire quasi inexistante dans le domaine analogique,
- les résultats expérimentaux de quelques publications démontrent la faisabilité d'ASIC analogiques fonctionnels en pratique,
- les résultats expérimentaux de quelques publications démontrent la possibilité de réaliser des circuits à basse consommation de puissance ou de faible surface d'intégration,
- la sensibilité au bruit n'est pas considérée dans les articles répertoriés par l'auteur de cette thèse.

4.3. Méthodologie d'implantation

4.3.1. Représentation des données

D'une manière générale, l'intégration d'algorithmes dans le silicium est limitée par les degrés de liberté et les contraintes de la technologie utilisée. L' intégration d'algorithmes dans des circuits électroniques nécessite la représentation des données et des variables de l'algorithme sous forme de signaux électriques. Dans le domaine analogique, les principaux choix de représentations, appelées aussi modes de représentation, sont :

- mode courant : les données sont représentées par la valeur d'un courant électrique,
- mode tension : les données sont représentées par la valeur d'une tension électrique,
- mode fréquence : les données sont représentées par la fréquence d'un signal électrique (exemple : modulation PWM).

Les raisons du choix de l'une ou l'autre des représentations dépendent des données de l'algorithme et des opérations mathématiques mises en jeu. La représentation en mode courant permet une dynamique des variables plus importante alors que la représentation en mode tension est plus adaptée à la distribution d'un signal d'une source vers plusieurs circuits différents. De plus, certaines opérations mathématiques favorisent naturellement l'une ou l'autre des représentations comme nous le verrons au long de ce chapitre. La représentation en fréquence offre une plus faible sensibilité au bruit mais réduit considérablement la vitesse des circuits. D'autre part, les circuits basés sur ce principe font large usage de comparateurs qui rendent les algorithmes plus sensibles aux tensions de décalages [El-Masry *et al.* '97]. Les représentations en mode courant et en mode tension permettent de plus grandes vitesses mais sont plus sensibles au bruit. Nous optons finalement pour ce choix, le problème de sensibilité au bruit étant résolu d'une autre manière qui sera présentée dans la section 4.3.2. Les tensions représentant des variables seront comprises dans l'intervalle $[-1;1]$ Volts tandis que les courants représentant des variables dépendront des courants admissibles par les circuits utilisés.

4.3.2. Circuits différentiels

Comme nous l'avons mentionné dans les sections 4.1 et 4.2, un des inconvénients majeurs des circuits analogiques, comparativement aux circuits numériques, est la sensibilité au bruit, surtout lorsque les représentations en mode courant ou en mode tension sont utilisées. Dans la section 4.2, nous avons pu constater que ce problème n'était pas envisagé par les chercheurs du domaine de l'ITGE complètement analogique des RNA. L'utilisation de circuits différentiels permet d'apporter une solution simple et efficace à ce problème [Johns & Martins '97]. Dans un circuit différentiel, une donnée ou une variable est représentée non plus par un courant ou par une tension sur un fil mais sur deux fils comme illustré par la figure 4.1.

(a) (b)

Figure 4.1. Mode de représentation différentiel : (a) en courant ; (b) en tension.

Un des deux fils représente la borne dite positive et l'autre la borne dite négative. La valeur de la donnée ou de la variable est représentée par la différence de courant ou de tension entre les deux fils selon le mode de représentation. Ainsi, en supposant que le bruit agit statistiquement de façon égale sur les deux fils (si ceux-ci sont proches dans le circuit), l'influence de ce bruit s'annule dans la différence. Plus précisément, les principaux avantages des circuits différentiels sont :

- de très grands taux de réjection de bruit de mode commun (*angl. Common Mode Rejection Ratio – CMRR*) et d'alimentation (*angl. Power Supply Rejection Ratio – PSRR*) [Johns & Martins '97 ; Lesueur *et al.* '01],
- atténuation importante des niveaux de distorsions fréquemment rencontrés avec les circuits analogiques [Sanchez-Sinencio & Andreou '99 ; Baker *et al.* '98],

- une dynamique des variables doublée en mode tension : dans une représentation en mode tension unique, lorsque $V_{DD} = 1.65\text{V}$ et $V_{SS} = -1.65\text{V}$ par exemple, la valeur positive maximale dans la représentable est 1.65V. En représentation différentielle, la valeur théorique maximale positive est 3.3V correspondant à $V_{DD} - V_{SS}$.

4.3.3. Surface d'intégration et consommation de puissance

Pour chacune des fonctions de base qui composeront le circuit final, il existe des circuits possibles et déjà publiés dans la littérature. Dans cette thèse, l'objectif majeur est d'obtenir un modèle de circuit qui soit un bon compromis entre les différents critères de précision, de surface d'intégration et de consommation de puissance. Ainsi, les choix des circuits seront effectués dans le sens de ce compromis. Dans certains cas, l'optimisation de plusieurs critères s'avère contradictoire, il faut alors opter pour le meilleur compromis ou bien effectuer le choix suivant le critère le plus important dans l'application visée.

4.3.4. Précision

La précision algorithmique n'est pas l'unique objectif dans ce travail. Dans chaque circuit, l'objectif est d'obtenir une précision suffisante permettant de maintenir la qualité des résultats algorithmiques obtenus dans le troisième chapitre. Les calculs dans les circuits analogiques sont limités en précision. D'après O'Leary, cette limite de précision est de l'ordre de 1% [O'Leary '91]. D'après cette référence et d'après nos essais de simulations, un niveau de précision de 1% des fonctions de base de l'algorithme, c'est-à-dire une erreur relative maximale de $\pm 1\%$ sur ces fonctions, est suffisante pour éviter une dégradation significative des résultats.

Soit y la sortie d'un circuit de base et y_d la sortie idéale pour le même circuit, nous définissons l'erreur relative e_{ra} par :

$$e_{ra(n)} = \frac{y_{(n)} - y_{d(n)}}{\left| y_{d(n)} \right|} \qquad \text{pour} \quad y_{d(n)} \neq 0 \tag{4.3}$$

Pour $y_{d(n)} = 0$, e_{ra} est obtenue par prolongation continue. Nous avons volontairement utilisé une erreur relative signée afin de faciliter la visualisation de la symétrie de certains circuits comme nous le verrons dans le cinquième chapitre.

4.3.5. Originalités des circuits proposés

Dans la section 4.4, nous présenterons les différents circuits pour la modélisation au niveau de l'ITGE de la loi de commande développée dans la thèse. Certains circuits de base proviennent d'articles de la littérature et ont été directement utilisés comme composants dans la réalisation de différents blocs fonctionnels. La modélisation de circuits CMOS analogiques pour l'intégration d'une loi de commande adaptative est déjà une originalité puisque la littérature dans ce domaine est très limitée. De plus, des circuits proposés dans la thèse et non publiés dans la littérature présentent les originalités suivantes :

- circuits complètement différentiels : tous les circuits de base utilisés sont complètement différentiels. Dans [Spencer & Sanchez-Sinencio '99], un circuit CMOS complètement différentiel est proposé pour l'ITGE analogique d'un réseau de neurones avec adaptation en ligne. Cependant, il s'agit d'un type de réseau auto-organisant avec des neurones linéaires, type de réseau bien différent de celui utilisé dans la thèse. Notons aussi que dans [Titus & Gopalan '02], l'utilisation d'un sommateur-amplificateur différentiel est suggérée pour l'ITGE analogique des RNA,
- adaptation analogique en ligne : les exemples déjà publiés de circuits analogiques avec adaptation en ligne pour les RNA utilisent la méthode de perturbation des poids ou la représentation PM-CWM [El-Masry *et al.* '97 ; Jabri & Flower '92 ; Montalvo *et al.* '97]. Nous n'avions pas retenu ces méthodes en raison des inconvénients algorithmiques et d'ITGE qu'elles présentent (*cf.* sections 3.2.2.2 et 4.3.1). Le développement algorithmique présenté dans le troisième chapitre nous a permis d'obtenir un algorithme de complexité plus faible que la méthode de perturbation des poids et qui permet de mieux exploiter le parallélisme intrinsèque des RNA,
- adaptation par intégration à temps continu : l'adaptation est réalisée avec des intégrateurs de type OTA-C. Ces intégrateurs sont très utilisés dans la conception de filtres analogiques, notamment dans les applications en vidéo [Lee *et al.* '99 ;

Bhattacharyya & Dey '99]. Nous proposons d'utiliser ce principe pour la modélisation des équations d'adaptation en ligne des poids synaptiques des réseaux mis en œuvre dans la loi de commande.

4.4. Circuits et fonctions de base

La modélisation au niveau de l'ITGE de la loi de commande décrite dans le troisième chapitre nécessite un certain nombre de circuits de base :

- amplificateur opérationnel, élément de base pour la réalisation de nombreuses fonctions analogiques,
- multiplieur,
- sommateur-soustracteur,
- convertisseur courant-tension permettant de passer d'un mode de représentation à l'autre,
- fonction tangente hyperbolique et sa dérivée,
- circuit d'adaptation des poids synaptiques,
- délai d'une période.

4.4.1. Amplificateur Opérationnel à Transconductance (AOT)

L'amplificateur opérationnel est un circuit de base fondamental dans la conception de nombreux circuits analogiques. Parmi les nombreuses topologies et variantes d'amplificateurs disponibles dans la littérature, notre choix s'est porté sur l'Amplificateur Opérationnel à Transconductance (AOT) présenté dans [Baker *et al.* '98] et illustré par la figure 4.2.

Par rapport à l'amplificateur opérationnel standard, l'AOT utilisé permet surtout d'obtenir les caractéristiques suivantes :

- amplification complètement différentielle c'est-à-dire deux entrées différentielles pour une sortie différentielle,
- asservissement de la polarisation afin d'assurer une tension de mode commun constante en sortie,
- opération en mode transconductance, c'est-à-dire que c'est le courant produit par l'étage différentiel d'entrée qui attaque directement l'étage de sortie, contrairement

aux topologies plus communes où ce courant développe une tension à travers une charge active,

- plage d'utilisation sur toute la plage dynamique de tension (*angl. rail to rail*).

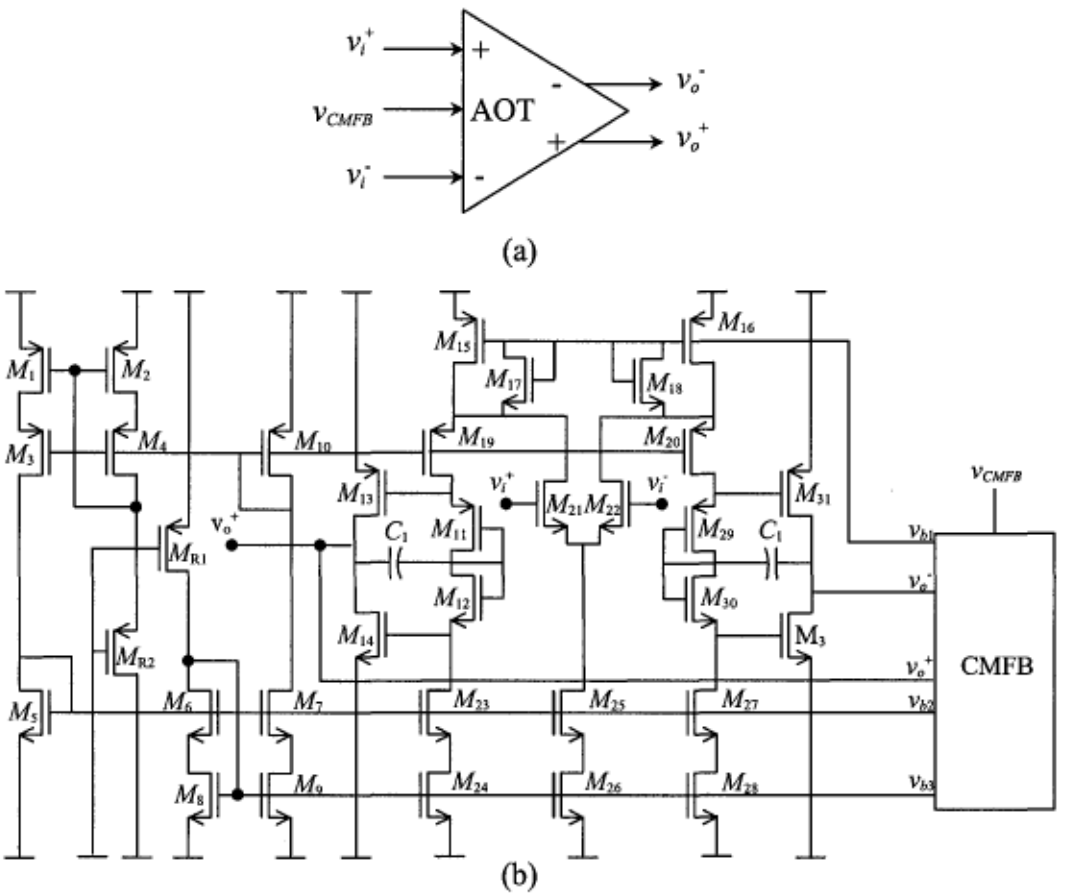


Figure 4.2. Amplificateur Opérationnel à Transconductance (AOT) : (a) symbole ; (b) schématique

De plus amples détails sur les rôles des différents étages de l'AOT sont donnés en annexe A. Notons la présence dans la figure 4.2 d'un circuit d'asservissement de mode commun (*angl. Common Mode FeedBack - CMFB*). Ce circuit, détaillé en annexe B est nécessaire pour l'asservissement de l'amplificateur. Étant donné qu'il n'y a pas de référence à la masse dans la représentation différentielle, ce circuit d'asservissement permet de maintenir la tension différentielle de sortie de l'AOT autour d'une tension de mode commun v_{CMFB} [Baker *et al.* '98]. L'objectif majeur de ce circuit est surtout d'éviter que la tension de mode commun ne dérive progressivement vers la tension d'alimentation positive ou négative.

Remarque : la tension de mode commun du circuit CMFB peut au besoin être non nulle comme nous le verrons plus loin.

4.4.2. Multiplieur

L'opération de multiplication est surtout présente dans le poids synaptique dont la fonction est de multiplier le signal d'entrée par la valeur du poids. Les entrées ainsi que les poids sont des signaux qui doivent être distribués à travers l'architecture du réseau. En conséquence, ces signaux sont représentés par des tensions conformément aux conclusions de la section 4.3.1. Les sorties des poids synaptiques connectées à un même neurone sont additionnées avant de passer dans la fonction tangente hyperbolique du neurone. Il est donc préférable de choisir un multiplieur ayant une sortie en mode courant afin de réaliser de façon très simple l'opération d'addition selon la loi de Kirchoff. Enfin, les entrées et la sortie du multiplieur doivent pouvoir prendre des valeurs positives et négatives telles qu'imposées par les algorithmes décrits dans le troisième chapitre. Il faut donc choisir un multiplieur dit à quatre quadrants [Sanchez-Sinencio & Silva-Martinez '00]. Le problème le plus important avec ce type de multiplieur est de conserver une bonne linéarité dans toute la plage d'utilisation. L'idée principale est d'utiliser une caractéristique non linéaire permettant de réaliser la multiplication et d'éliminer ensuite les termes non linéaires. De plus amples détails sur ce principe sont donnés en annexe C.

Pour notre multiplieur, nous avons retenu la topologie présentée dans la figure 4.3. Cette topologie permet d'annuler les termes non linéaires et par conséquent d'obtenir une excellente caractéristique de linéarité telle que démontrée dans [Sanchez-Sinencio & Silva-Martinez '00].
En écrivant les expressions des courants de drains des transistors M_1 à M_4, la sortie différentielle du multiplieur peut être exprimée par :

$$i_{out} = i_o^+ - i_o^- = \mu_n C_{ox} \frac{W}{L} v_x v_y \tag{4.4}$$

où W/L est le rapport largeur/longueur des transistors M_5 à M_8 et les tensions différentielles des entrées du multiplieur sont définies par l'équation (4.5).

$$\begin{cases} v_x = v_x^+ - v_x^- \\ v_y = v_y^+ - v_y^- \end{cases} \tag{4.5}$$

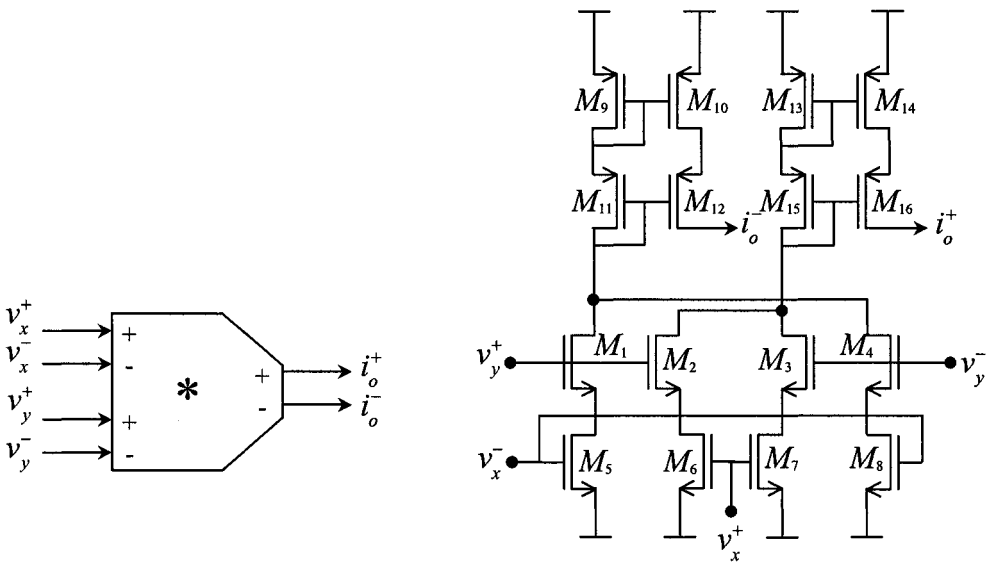


Figure 4.3. Multiplieur différentiel linéaire à quatre quadrants : (a) symbole ; (b) schématique

Afin d'obtenir une meilleure linéarité du multiplieur, les tensions différentielles v_x et v_y doivent avoir des tensions de mode commun égales en valeur absolue et opposées en signe. Étant donné la plage de représentation des tensions sur [-1,1], les tensions de mode commun choisies sont de 0.5V pour v_x et –0.5V pour v_y. Les circuits fournissant les entrées des multiplieurs doivent satisfaire ces conditions. Cela peut être réalisé en utilisant la tension de mode commun v_{CMFB} de l'AOT. Par exemple, si la sortie d'un AOT est connectée à l'entrée v_x du multiplieur, il suffit d'imposer $v_{CMFB} = 0.5\text{V}$ et l'AOT maintiendra la tension de mode commun à 0.5V.

4.4.3. Sommateur-soustracteur

Les équations de propagation et d'adaptation décrites dans les deuxième et troisième chapitres comprennent également des opérations d'additions et de soustractions. Lorsque cela est possible, il est préférable de représenter les signaux à additionner en mode courant. Ainsi, la sommation est réalisée de façon très simple en utilisant la loi de Kirchoff. La figure 4.4 illustre ce principe pour la sommation des entrées d'un neurone, les N entrées sont des courants différentiels i_k où $k = 1,\text{L}\ ,N$ et la sortie du circuit sommateur est exprimée par :

$$i_{out} = i_o^+ - i_o^- = \sum_{k=1}^{N} i_k = \sum_{k=1}^{N} \left(i_k^+ - i_k^- \right) \tag{4.6}$$

Pour cette opération, l'avantage de l'analogique en terme de surface d'intégration est très évident.

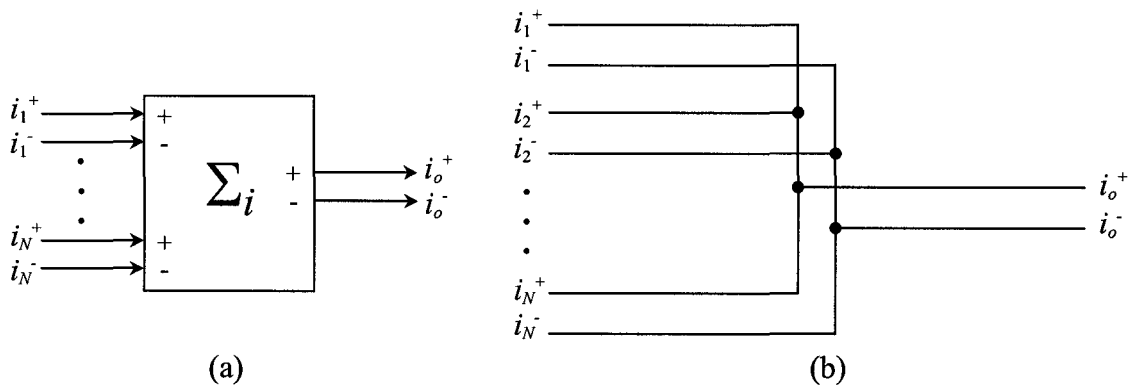


Figure 4.4. Principe d'addition et de soustraction en mode courant selon la loi de Kirchoff : (a) symbole ; (b) schématique

Le montage de la figure 4.4 peut être directement adapté afin de réaliser la soustraction en mode courant en inversant les bornes de l'entrée à soustraire, c'est-à-dire en réalisant la sommation de l'entrée opposée. Si les signaux à additionner ou à soustraire sont en mode tension, la solution la plus simple est d'utiliser un montage sommateur à amplificateur tel que présenté dans la figure 4.5.

La sortie de ce montage soustracteur est :

$$v_{out} = v_o^+ - v_o^+ = \frac{R_2}{R_1}\left(v_1 + v_2 \right) = \frac{R_2}{R_1}\left(v_1^+ - v_1^- + v_2^+ - v_2^- \right) \tag{4.7}$$

Les résistances R_1 et R_2 peuvent être égales mais au besoin le rapport de ces deux résistances permet un gain sur la tension différentielle de sortie. Cette fois encore, la soustraction de deux tensions peut être réalisée avec le même montage en inversant les bornes de l'entrée à soustraire.

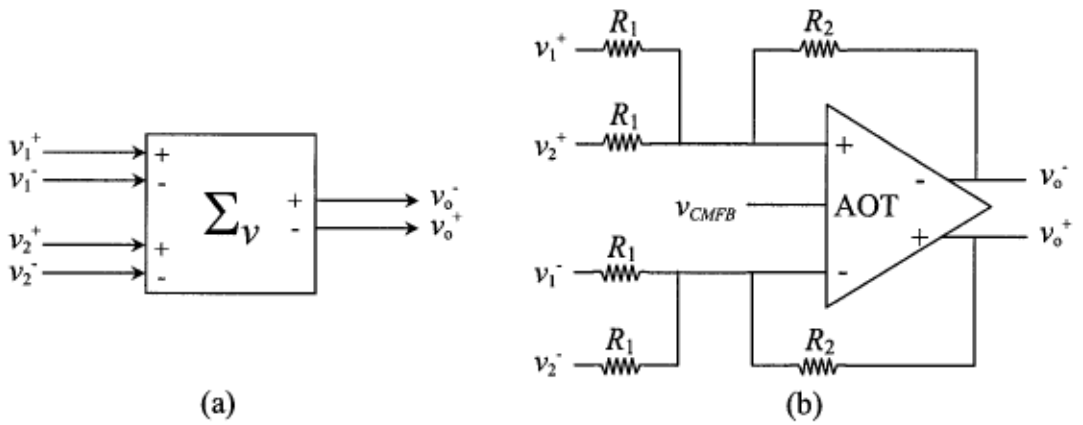


Figure 4.5. Montage sommateur à amplificateur pour l'addition ou la soustraction en mode tension : (a) symbole ; (b) schématique

4.4.4. Convertisseur courant-tension

Étant donné que certaines opérations sont effectuées en mode courant et d'autres en mode tension, une Conversion Courant-Tension (CCT) peut s'avérer nécessaire. Une façon simple de réaliser la CCT est d'utiliser l'AOT avec une résistance en rétroaction sur chacune des sorties de l'amplificateur. Ce principe est présenté dans la figure 4.6.

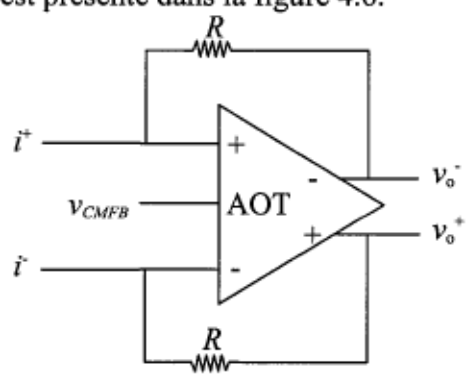


Figure 4.6. Principe de conversion courant-tension avec un AOT et deux résistances en rétroaction

Si l'impédance d'entrée de l'AOT est suffisamment grande, les courants d'entrées i^+ et i^- passent quasiment entièrement dans les résistances de part et d'autre de l'amplificateur et la tension différentielle en sortie est :

$$v_o^+ - v_o^- = R\left(i^+ - i^-\right) \tag{4.8}$$

Ainsi, le gain de conversion est directement donné par la valeur de la résistance. Pour réaliser la résistance au niveau des dessins de masques, on associe des résistances élémentaires calculées à partir de la valeur ohmique par unité de surface du matériau utilisé [Hastings '00]. Ce principe est présenté en détails en annexe D. Cependant, les tolérances (pourcentage d'erreur) données par les manufacturiers sur ces valeurs ohmiques sont élevées : jusqu'à 20% d'erreur sur la valeur ohmique du carré unitaire de surface du matériau. Sur un circuit tel que celui de la figure 4.6, cette erreur se reporterait directement sur la sortie ce qui n'est pas acceptable compte tenu du critère de précision énoncé dans la section 4.3.4.

La solution à ce problème peut être de remplacer simplement la résistance par un transistor fonctionnant dans sa partie linéaire [Johns & Martins '97]. En effet, la courbe de courant de drain en fonction de la tension drain-source d'un transistor dans sa partie linéaire est similaire à celle d'une résistance. Le circuit modifié est présenté dans la figure 4.7.

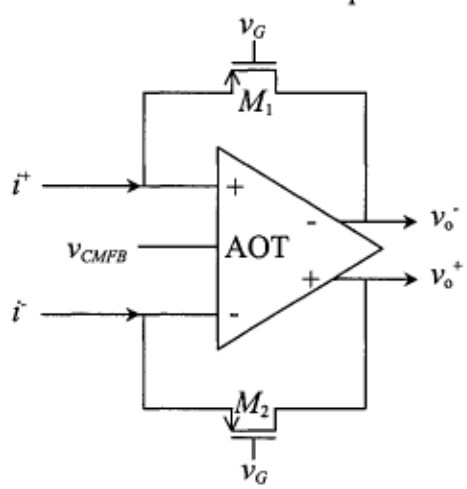


Figure 4.7. Conversion courant/tension avec des transistors connectés en résistances

La résistance équivalente du transistor est donnée par [Johns & Martins '97] :

$$R_{DS} = \frac{\partial V_{DS}}{\delta i_{DS}} = \frac{1}{\mu C_{ox}\left(\frac{W}{L}\right)\left(V_{GS} - V_T\right)} \tag{4.9}$$

Pour obtenir une valeur constante de la résistance, il faut maintenir la tension V_{GS} constante. Ceci est possible en imposant d'une part la tension de grille v_G et d'autre part en connectant la source des transistors aux entrées de l'AOT qui maintient ses entrées constantes et égales en tension dans ce montage.

Deux conditions supplémentaires doivent être satisfaites :

$$V_{GS} > V_T \tag{4.10}$$

$$V_{DS} > V_{eff} = V_{GS} - V_T \tag{4.11}$$

L'équation (4.10) est la condition de conduction du transistor, V_T étant la tension de seuil du transistor, et l'équation (4.11) représente la condition nécessaire pour que le transistor soit en saturation. Les deux équations sont écrites pour un transistor de type NMOS et doivent être réécrites au besoin pour un transistor PMOS [Johns & Martins '97].

Un autre problème survient dans l'utilisation du montage de la figure 4.7. Lorsque le gain de conversion doit être élevé, la valeur de la résistance équivalente des deux transistors doit également être élevée. Nous avons pu constater par simulation que lorsqu'un CCT est connecté en aval d'un multiplieur, ce qui en sera notre principale utilisation, si la résistance équivalente du CCT dépasse une certaine valeur, le courant de sortie du multiplieur commence à chuter. Ceci dénote un problème d'adaptation d'impédance entre les deux circuits. De plus, la valeur admissible de la résistance équivalente dépend du nombre de multiplieurs ayant leur sortie connectée à une même entrée de CCT, les impédances de sortie des différents multiplieurs s'additionnant pour donner l'impédance équivalente à l'entrée du CCT. Par exemple, à l'entrée d'un neurone, un certain nombre d'entrées sont multipliées par des poids synaptiques. Les courants résultant des différents multiplieurs sont additionnés selon la loi de Kirchoff (*cf.* figure 4.4) et le courant total est converti en tension par un CCT. Le nombre d'entrées d'un neurone étant dépendant de l'application, l'impédance vue en amont à l'entrée du CCT est, elle aussi, variable. Pour contourner ces deux problèmes, nous adoptons les deux solutions suivantes :

- ajout d'un deuxième étage de gain dans le circuit de la figure 4.7 afin de limiter la valeur nécessaire de résistance équivalente des deux transistors M_1 et M_2,
- ajustement des dimensions des deux transistors M_1 et M_2, donc de la résistance équivalente, en fonction du nombre de multiplieurs connectés à l'entrée du CCT :

si le nombre de multiplieurs connectés augmente alors diminuer la résistance équivalente à M_1 et M_2.

Le circuit final du CCT avec les deux étages est présenté dans la figure 4.8.

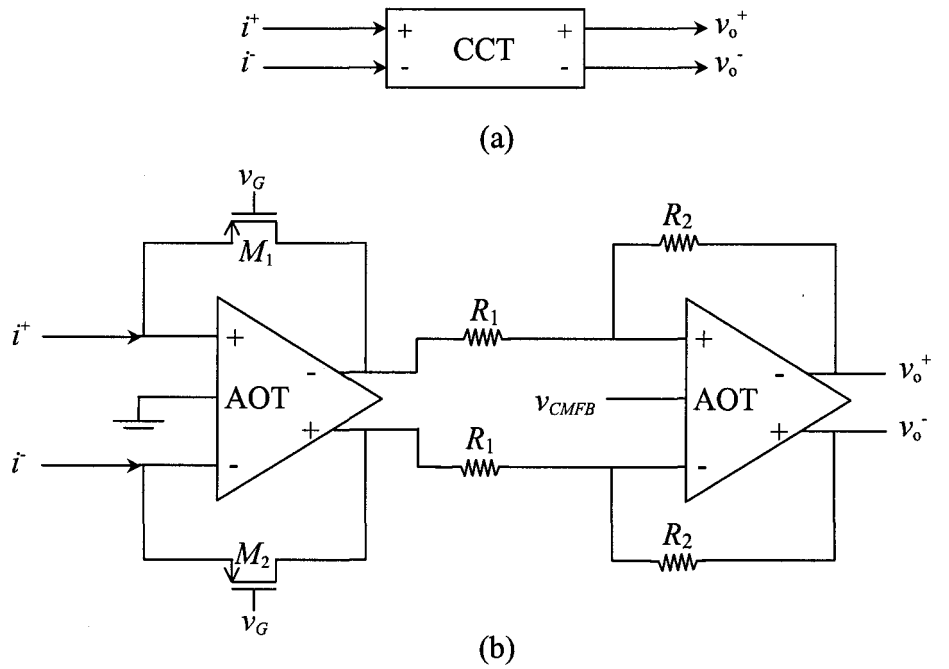


Figure 4.8. Conversion courant-tension avec deux étages : (a) symbole ; (b) schématique

Dans la suite, le symbole du CCT désignera un circuit similaire à celui de la figure 4.8 mais différentes valeurs des dimensions de M_1, M_2, R_1, R_2 et différentes valeurs de v_G et v_{CMFB} seront possibles. Les valeurs des paramètres pour chacun des circuits seront présentées dans le cinquième chapitre.

Remarque : le problème d'imprécision sur la valeur de la résistance au niveau des dessins de masques n'a pas été mentionné pour les montages sommateurs des figures 4.5 et 4.8. En fait, le problème survient très peu pour ces montages car il s'agit de rapport de résistances et que dans ce cas une précision beaucoup plus importante peut être atteinte comme nous le verrons dans le cinquième chapitre.

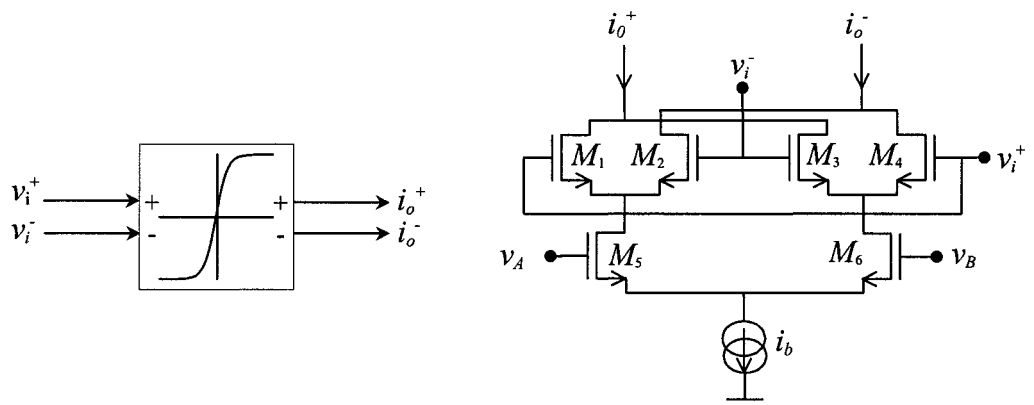
4.4.5. Fonction tangente hyperbolique et sa dérivée

Dans le deuxième chapitre, nous avons statué sur l'utilisation d'une fonction tangente hyperbolique comme fonction d'activation des neurones de la couche cachée. Rappelons aussi que c'est surtout la forme de cette fonction qui est importante ainsi que le calcul de sa dérivée pour l'adaptation des poids. Une façon simple de réaliser la fonction tangente hyperbolique est d'utiliser le multiplieur de Gilbert présenté dans la figure 4.9 [Coué & Wilson '96].

Ce circuit a été utilisé dans plusieurs travaux sur l'ITGE de RNA multicouches, certains auteurs présentant des résultats expérimentaux après fabrication [Coué & Wilson '96]. Le circuit est composé de trois paires différentielles balancées. La paire M_5-M_6 sert à répartir le courant de polarisation i_b entre les paires M_1-M_2 et M_3-M_4. Ces dernières produisent à leurs drains respectifs un courant qui obéit à une loi de tangente hyperbolique et sont connectées de sorte que leurs courants de drains soient soustraits deux à deux pour donner les deux courants de sortie i_o^+ et i_o^-. Les tensions v_A et v_B permettent d'assurer la conduction des transistors M_5-M_6 et d'ajuster la répartition des courants de drains dans les deux branches. La sortie différentielle de ce circuit donne directement une allure tangente hyperbolique exprimée par [Coué & Wilson '96]

$$i_{out} = i_o^+ - i_o^- \; ; \; i_b \tanh\left(k\left(v_i^+ - v_i^-\right)\right) \tag{4.12}$$

où i_b est le courant de polarisation du multiplieur et k est une constante dépendante des paramètres des transistors.



(a) (b)

Figure 4.9. Multiplieur de Gilbert pour la réalisation de la fonction tangente hyperbolique : (a) symbole ; (b) schématique

La sortie de ce circuit étant un courant, il faudra ajouter un CCT pour obtenir la tension de sortie du neurone. À partir de cette tension, la dérivée peut être obtenue de manière simple en considérant la relation suivante :

$$\frac{d\tanh(z)}{dz} = 1 - \tanh^2(z) \tag{4.13}$$

La dérivée de la fonction hyperbolique peut donc être obtenue par une multiplication et une soustraction. Ce principe est illustré par la figure 4.10.

En connectant les entrées du multiplieur comme indiqué dans la figure 4.10, la sortie est un courant différentiel directement proportionnel à $-\tanh^2(z)$. Il ne reste alors qu'à convertir ce courant avec un CCT et à ajouter une tension différentielle de 1V. Cette tension de 1V est ajoutée par des courants $i^+_{(0.5V)}$ et $i^-_{(0.5V)}$, courants générant une tension différentielle de 1V supplémentaire dans le CCT.

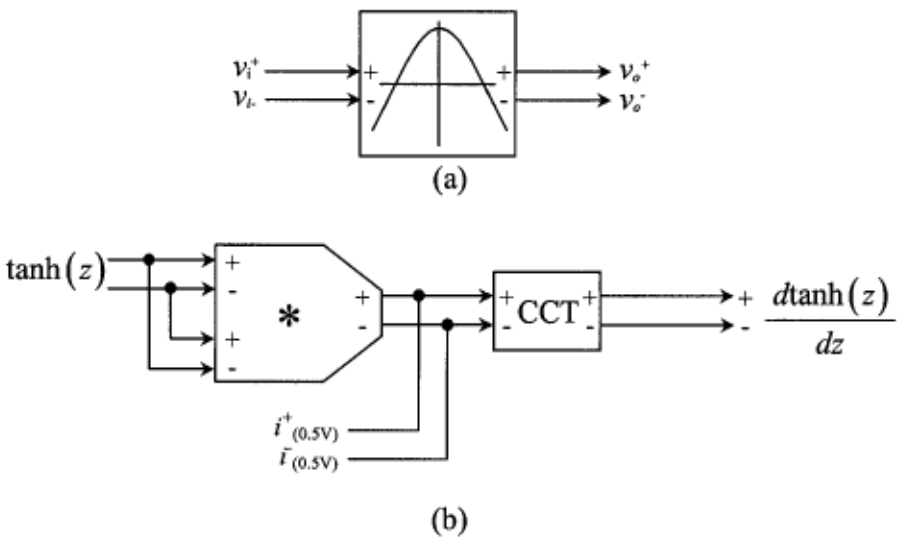


Figure 4.10. Circuit pour le calcul de la dérivée de la fonction tangente hyperbolique : (a) symbole ; (b) schématique

4.4.6. Adaptation des poids

Suite à la section 4.2 où nous avons présenté une synthèse de l'état de la recherche dans le développement d'ASIC analogiques pour l'ITGE des RNA, nous proposons une nouvelle approche pour l'adaptation en ligne des poids synaptiques :

- le circuit d'un poids synaptique comprend deux parties : la propagation et l'adaptation. Ainsi, l'adaptation est réalisée localement à l'intérieur de chaque poids et de façon complètement parallèle à la propagation directe,
- l'adaptation est réalisée à temps continu : les circuits d'adaptation utilisent des signaux continus dans le temps et sans signaux de synchronisation (horloge),
- les circuits d'adaptation, comme les circuits de propagation directe, sont complètement différentiels.

D'une manière générale, les équations d'adaptation d'un poids w développées dans le troisième chapitre peuvent s'écrire sous une forme unique :

$$w_{(n+1)} = w_{(n)} + s_{(n)} \tag{4.14}$$

où $s_{(n)}$ représente la quantité ajoutée à la valeur du poids w à l'instant n pour obtenir la nouvelle valeur à l'instant $n+1$.

En divisant par la période d'échantillonnage, en arrangeant les termes et en utilisant une approximation du premier ordre - les termes de plus hauts degrés sont négligés en raison de la très faible valeur de la période d'échantillonnage relativement à la dynamique du système commandé - cette équation devient :

$$w_{(n+1)} \approx w_{(0)} + \frac{1}{T_s} \int_{t=0}^{t=n+1} s_{(t)} dt \tag{4.15}$$

où $w_{(0)}$ est la valeur initiale du poids w, c'est-à-dire sa valeur à l'instant 0. Cette transformation permet une modélisation simple de l'équation d'adaptation, sans horloge et sans circuit de mémoire. Elle réalise en même temps une transformation de la loi de commande discrète en loi de commande continue.

Il existe principalement trois techniques de réalisation d'intégration continue dans des circuits analogiques : capacité commutée (*angl. Switched Capacitor* - SC), transconductance-capacité (gm-C) et filtres actifs à résistance et capacité (RC) [Sanchez-Sinencio & Andreou '99 ; Johns & Martins '97].

La technique de capacité commutée est une manière simple de réaliser des intégrateurs ayant une bonne linéarité et une plage dynamique intéressante. Un inconvénient est le besoin d'une horloge pour la commutation, horloge dont la fréquence doit être au moins égale au double de la fréquence maximale du signal à intégrer pour éviter les recouvrements de fréquences. D'autre part, cette technique souffre du problème d'injection de charges et de courants de court-circuit pendant les commutations, comme nous l'avions mentionné dans la section 4.1.2 [Sanchez-Sinencio & Andreou '99].

La technique utilisant un élément de transconductance et une capacité (gm-C) est une autre façon de réaliser l'intégration à temps continu. Toutefois, cette technique s'applique à l'intégration de tensions. Dans notre cas, une intégration de courant est désirée, il serait alors possible d'ajouter un bloc de CCT pour attaquer l'intégrateur à transconductance en tension, mais l'ajout d'un CCT signifie l'ajout de deux AOT alors que la prochaine technique d'intégration que nous allons voir n'en utilise qu'un.

Finalement, nous avons retenu la technique de filtre actif à résistance et capacité. La structure est composée d'un élément de résistance combiné à un amplificateur avec une capacité en rétroaction. L'utilisation de résistances peut engendrer de grandes surfaces d'intégration sur le silicium et de grandes incertitudes. Il est alors préférable, comme pour le CCT, d'utiliser un transistor dans sa partie linéaire pour réaliser une résistance. Un autre problème important associé à cette technique est la réalisation d'amplificateurs à faible distorsion. L'utilisation d'un AOP classique par exemple ne donne pas de bons niveaux de distorsion. Ce problème peut être contourné en utilisant pour amplificateur un AOT à faible distorsion tel que celui que nous avons présenté dans la section 4.4.1.

Nous proposons finalement le circuit d'adaptation des poids synaptiques détaillé dans la figure 4.11. L'intégrale de l'équation (4.15) est réalisée par l'intégrateur à base d'AOT de la partie gauche du circuit tandis que la partie droite réalise la sommation avec la valeur initiale du poids. Notons que le produit R_iC_i de l'intégrateur doit être égal à la période d'échantillonnage T_s pour satisfaire l'équation (4.15). Cette condition peut nécessiter de très grandes valeurs de R_i ou C_i. Pour éviter cela, une partie de la constante de temps peut être introduite dans le gain du CCT situé en amont de l'intégrateur ou encore dans le gain du montage sommateur de la partie droite de la figure 4.11, ce qui permet du même coup de diminuer ce gain

et de rendre ce circuit plus petit. Notons dans cette figure que l'entrée est en mode tension. La résistance du produit R_iC_i est réalisée par des transistors pour les raisons invoquées dans la section 4.4.4. Ceci permet également de rendre ajustable la période de l'intégrateur en variant la tension de grille v_{G1} (*cf.* figure 4.8) en respectant bien sûr les conditions décrites dans la section 4.4.4. Le rapport des résistances R_1 et R_2 dans le sommateur permet d'ajuster au besoin le niveau de tension en sortie tandis que la tension v_{CMFB} permet d'ajuster la tension de mode commun dépendant du montage situé en aval.

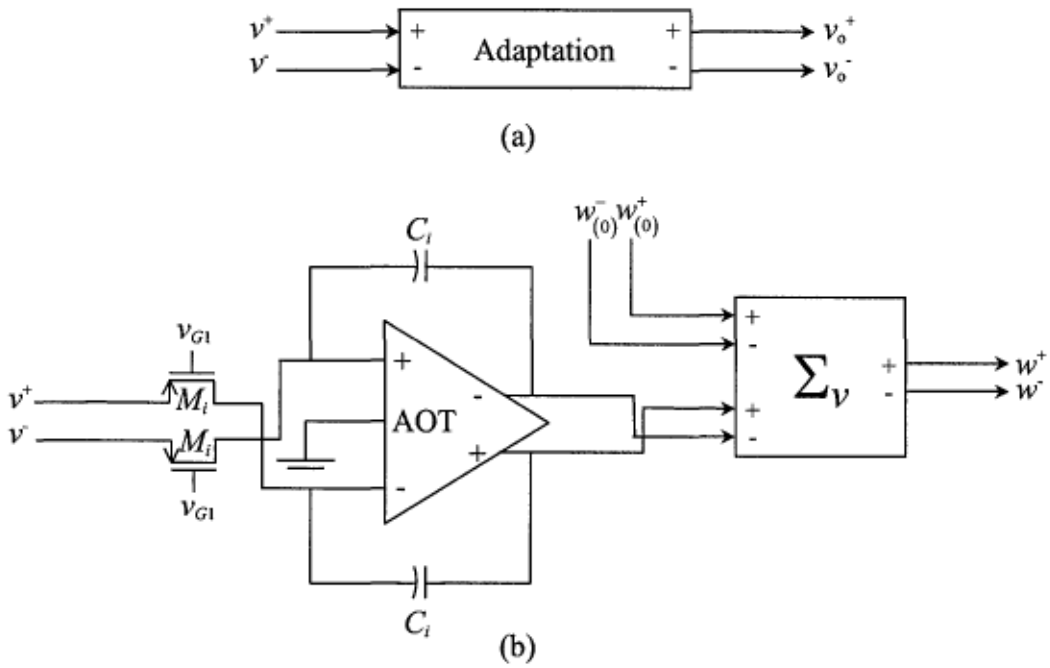
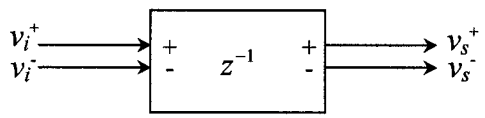


Figure 4.11. Circuit différentiel d'intégration à temps continu pour l'adaptation des poids : (a) symbole ; (b) schématique

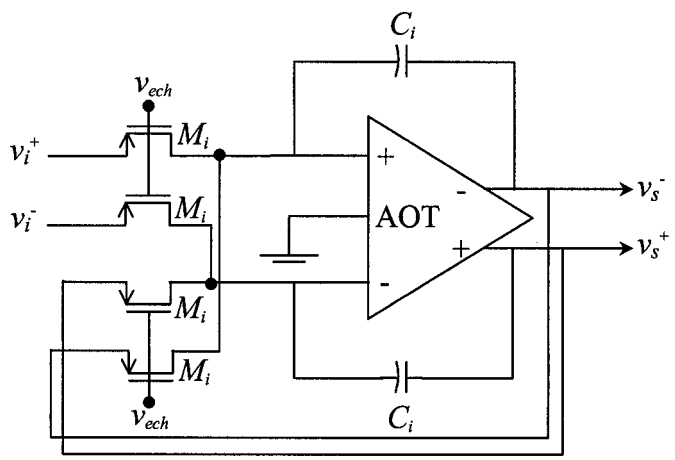
4.4.7. Délai

Les topologies de RNA décrites dans le deuxième chapitre comportent des délais d'une période appliqués à certains signaux (*cf.* figure 2.4). Une façon simple de réaliser un délai analogique à temps continu est d'utiliser un montage intégrateur en rétroaction tel qu'illustré par la figure 4.12. L'intégrateur est le même qu'utilisé dans le circuit d'adaptation de la figure 4.11.

L'utilisation de l'AOT en boucle fermée pour la réalisation du circuit de délai permet d'obtenir une bonne précision dans la gamme des fréquences des signaux auxquels sont appliqués les délais. La tension v_{ech} permet de régler la constante d'intégration comme décrit dans la section 4.4.6 et sous les mêmes conditions. Cette tension représente donc un moyen simple d'ajuster la valeur du délai dépendant de la période d'échantillonnage souhaitée.



(a)



(b)

Figure 4.12. Circuit différentiel de délai analogique à intégration à temps continu

4.5. Architecture complète

4.5.1. Réseaux N_I et N_C

L'architecture des réseaux N_I et N_C au niveau de l'intégration sur silicium est similaire à celle décrite dans le deuxième chapitre au niveau de la structure de commande. Nous y distinguons deux blocs fonctionnels de base : (i) neurone ; (ii) poids synaptique. Les différents

circuits présentés dans cette section utilisent les noms des variables du réseau N_I. Des circuits analogues pour le réseau N_C peuvent être obtenus en adaptant convenablement les noms de variables (*cf.* Tableau 2.1).

4.5.1.1. Neurone

Le circuit complet d'un neurone caché N_i est présenté dans la figure 4.13 où l'on retrouve différents circuits de base présentés dans la section 4.4.

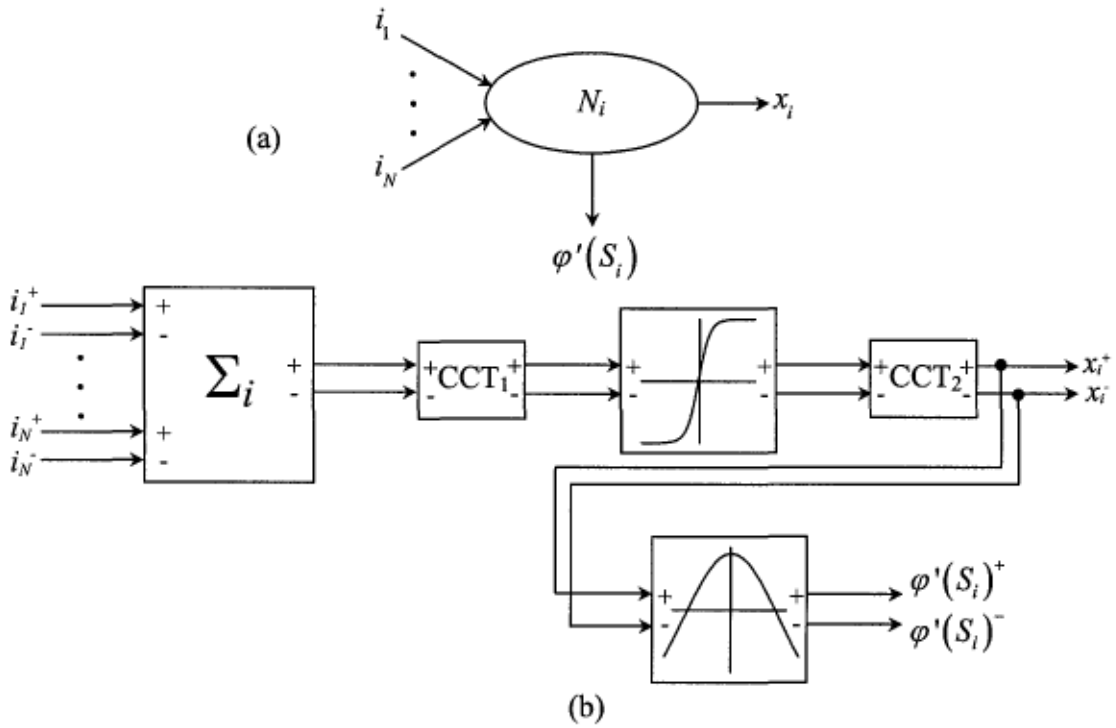


Figure 4.13. Schéma complet d'un neurone caché : (a) symbole ; (b) schématique

La sommation des entrées est réalisée en mode courant, un premier CCT (CCT_1) est nécessaire pour entrer une tension au circuit de tangente hyperbolique et un autre CCT (CCT_2) est nécessaire en sortie pour obtenir finalement la sortie du neurone en tension. Cette tension est aussi retournée à l'entrée du bloc qui calcule la dérivée de la fonction tangente hyperbolique conformément à ce que nous avions vu dans la section 4.4.5.

La figure 4.14 présente le circuit du neurone de sortie. Étant donné que la fonction d'activation de ce neurone est linéaire et de gain unitaire, la dérivée est toujours égale à l'unité. Le circuit de tangente hyperbolique et celui de sa dérivée ne sont donc plus nécessaires dans ce circuit.

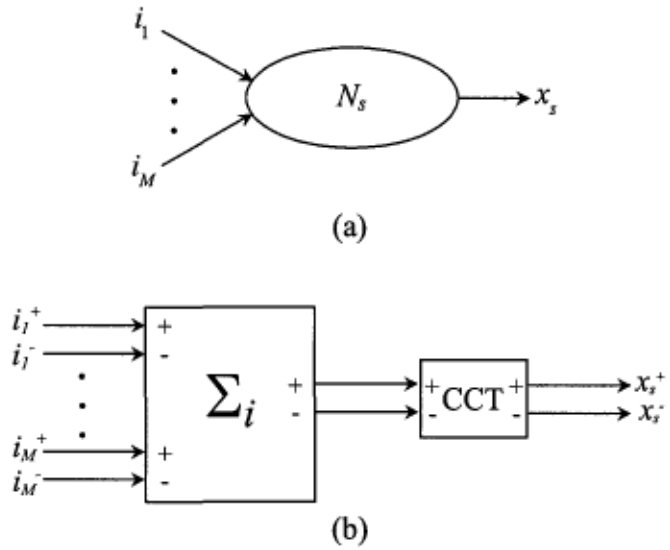


Figure 4.14. Schéma complet du neurone de sortie : (a) symbole ; (b) schématique

Remarque : les signaux dans les symboles de neurones sont représentés sur un fil même s'ils demeurent différentiels. À ce niveau de hiérarchie et à tous les niveaux supérieurs, les signaux sont représentés sur un fil pour simplifier les figures, la représentation différentielle étant sous-entendue.

4.5.1.2. Poids synaptique

Les figures 4.15 et 4.16 présentent les schémas blocs respectifs des poids synaptiques des couches de sortie et cachée. Pour faciliter la compréhension des figures, nous rappelons les équations d'adaptation des poids du réseau N_I. Les équations pour le réseau N_C sont obtenues en adaptant les notations conformément aux correspondances présentées dans le tableau 2.1.

$$w_{1j(n+1)}^{(2)} = w_{1j(n)}^{(2)} + \eta_I e_{I(n+1)} x_{j(n)} \tag{4.16}$$

$$w_{ij(n+1)}^{(1)} = w_{ij(n)}^{(1)} + \eta_I e_{I(n+1)} w_{1i(n)}^{(2)} \varphi'\left(S_{i(n)}\right) s_{j(n)} \tag{4.17}$$

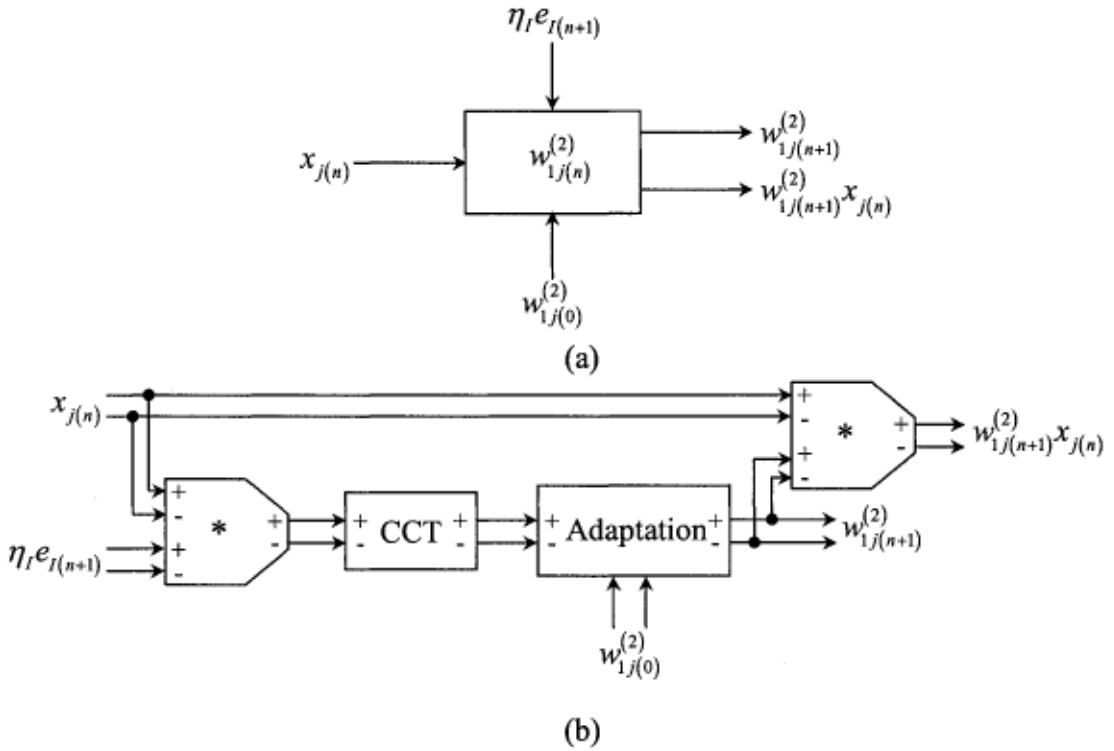


Figure 4.15. Schéma complet d'un poids synaptique de sortie : (a) symbole ; (b) schématique

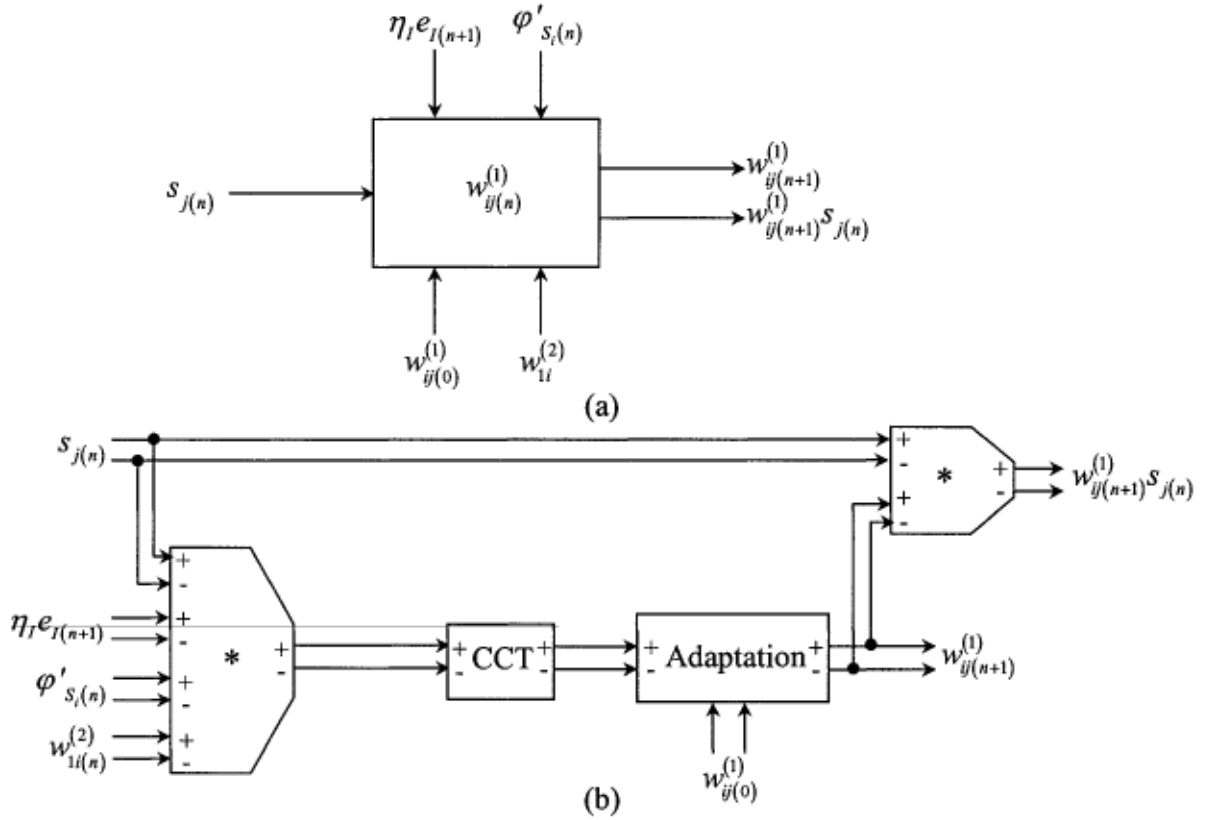


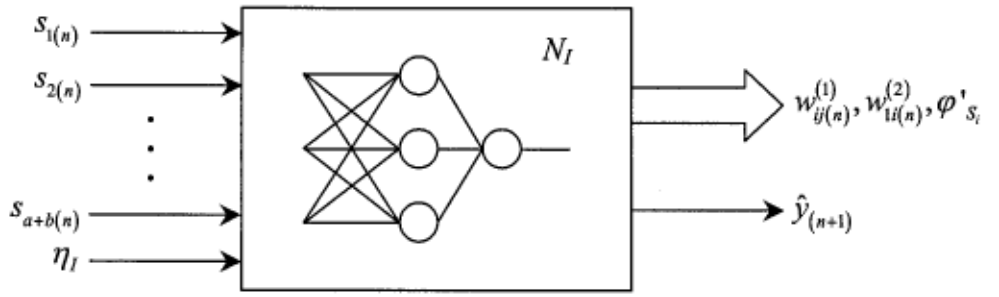
Figure 4.16. Schéma complet d'un poids synaptique caché : (a) symbole ; (b) schématique

Dans la suite nous utiliserons la notation $\varphi'_{S_i} = \varphi'\left(S_{i(n)}\right)$ pour simplifier les figures. La présence de $w^{(2)}_{1j(0)}$ et de $w^{(1)}_{ij(0)}$ dans les circuits est due à la transformation des équations d'adaptation selon une intégration continue exposée dans la section 4.4.6. Les valeurs des poids $w^{(2)}_{1j}$ sont également sorties des poids synaptiques de sortie puisqu'elles sont nécessaires à l'adaptation des poids de la couche cachée. Les poids de la couche cachée ont deux entrées supplémentaires conformément à l'équation (4.17).

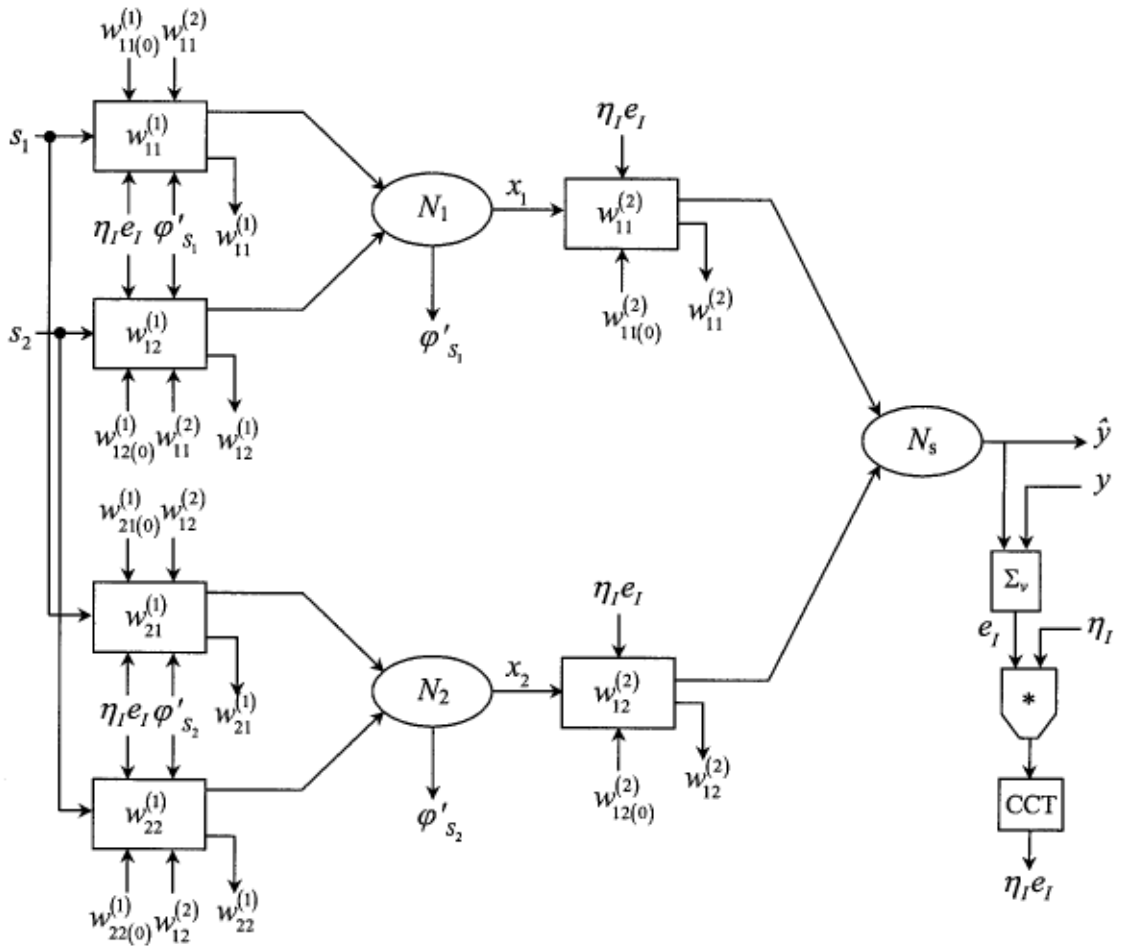
4.5.1.3. Schémas blocs de N_I et N_C

Les circuits complets des réseaux N_I et N_C sont obtenus en combinant adéquatement les circuits de base présentés dans les sections précédentes. L'architecture du réseau N_I est représentée dans la figure 4.18 dans le cas simple d'un réseau à deux entrées et deux neurones cachés afin de simplifier le dessin. Les signaux nécessaires à l'adaptation des poids y sont également représentés. Pour ne pas trop compliquer la figure et la rendre plus lisible, les signaux différentiels sur deux fils sont représentés sur un fil et certaines connections entre les différents éléments ne sont pas dessinées. Dans la figure 4.18, le montage sommateur de sortie est configuré en soustracteur pour le calcul de l'erreur d'identification e_I.

La figure 4.19 présente le schéma bloc du réseau N_C. Comme cela a déjà été mentionné, l'architecture du réseau N_C est similaire à celle du réseau N_I. Notons toutefois que dans le cas du réseau N_C, l'erreur utilisée pour l'adaptation des poids n'est plus calculée à la sortie du réseau mais provient du circuit qui réalise le calcul de cette erreur e_u et qui sera présenté plus loin.

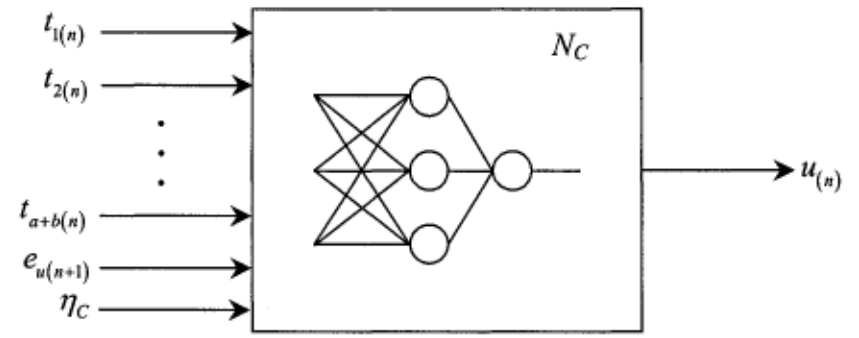


(a)

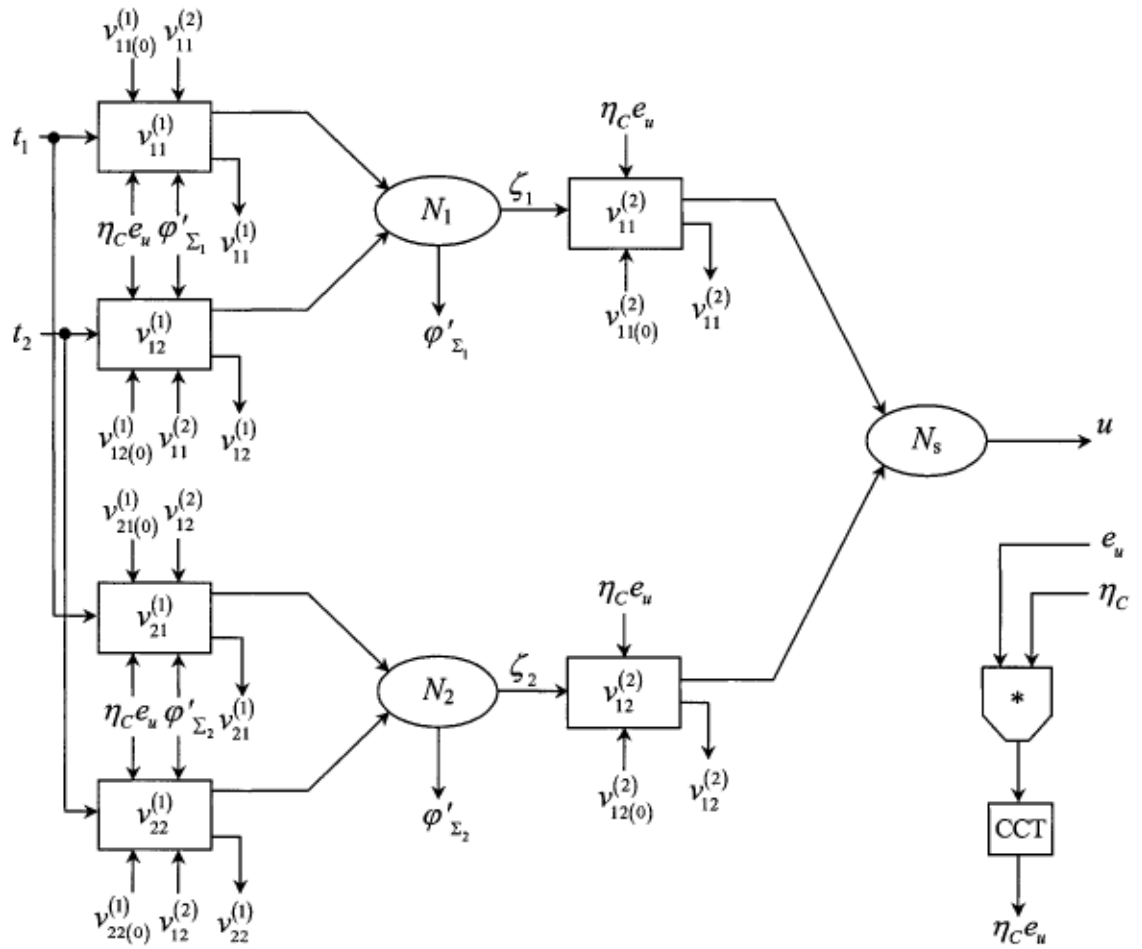


(b)

Figure 4.17. Schéma bloc du réseau N_I pour deux entrées et deux neurones cachés : (a) symbole ; (b) schématique



(a)



(b)

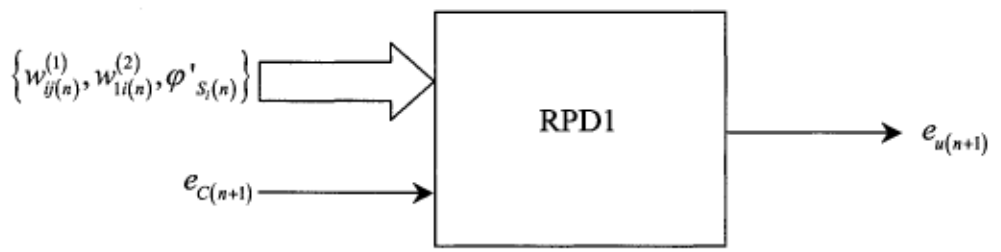
Figure 4.18. Schéma bloc du réseau N_C : (a) symbole ; (b) schématique pour deux entrées et deux neurones cachés.

4.5.2. Intégration de l'algorithme RPD1

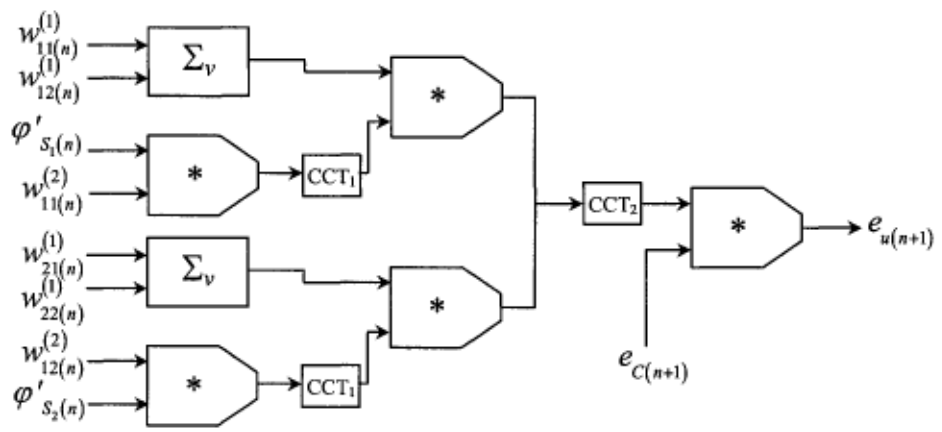
Nous rappelons l'expression de l'erreur e_u selon l'algorithme RPD1 développée dans le troisième chapitre :

$$e_{u(n+1)} = e_{C(n+1)} \sum_{k=1}^{M} \left(w_{1k(n)}^{(2)} \varphi' \left(S_{k(n)} \right) \sum_{l=1}^{a} w_{kl(n)}^{(1)} \right) \tag{4.18}$$

Cette expression est simplement une combinaison de sommes et de produits réalisés par les circuits élémentaires décrits dans ce chapitre. Notons que les variables nécessaires à l'évaluation de cette expression, à l'exception de l'erreur e_C, sont sorties des poids de la couche cachée et de la couche de sortie du réseau N_I comme illustré dans la figure 4.18. Le schéma bloc du circuit pour le calcul de e_u selon RPD1 est présenté dans la figure 4.20 dans le où $a = 2$ et $M = 2$.



(a)



(b)

Figure 4.19. Schéma bloc du circuit de calcul de l'erreur e_u selon RPD1 dans le où $a = 2$ et $M = 2$: (a) symbole ; (b) schématique

Dans ce cas, l'équation (4.18) peut être développée comme suit :

$$e_{u(n+1)} = e_{C(n+1)}\left[w_{11(n)}^{(2)}\varphi'_{S_{1(n)}}\left(w_{11(n)}^{(1)} + w_{11(n)}^{(1)}\right) + w_{12(n)}^{(2)}\varphi'_{S_{2(n)}}\left(w_{21(n)}^{(1)} + w_{22(n)}^{(1)}\right)\right] \tag{4.19}$$

L'équation (4.19) nous permet de constater que le calcul de $e_{u(n+1)}$ par la méthode RPD1 est très simple.

4.5.3. Schéma bloc complet

Le schéma bloc complet de la structure de commande à deux réseaux a été présenté dans la figure 2.4 au deuxième chapitre. Ce schéma bloc reste valable en remplaçant les réseaux et les autres blocs par les circuits correspondants et présentés au long de ce chapitre. La figure 4.21 présente le schéma bloc global.

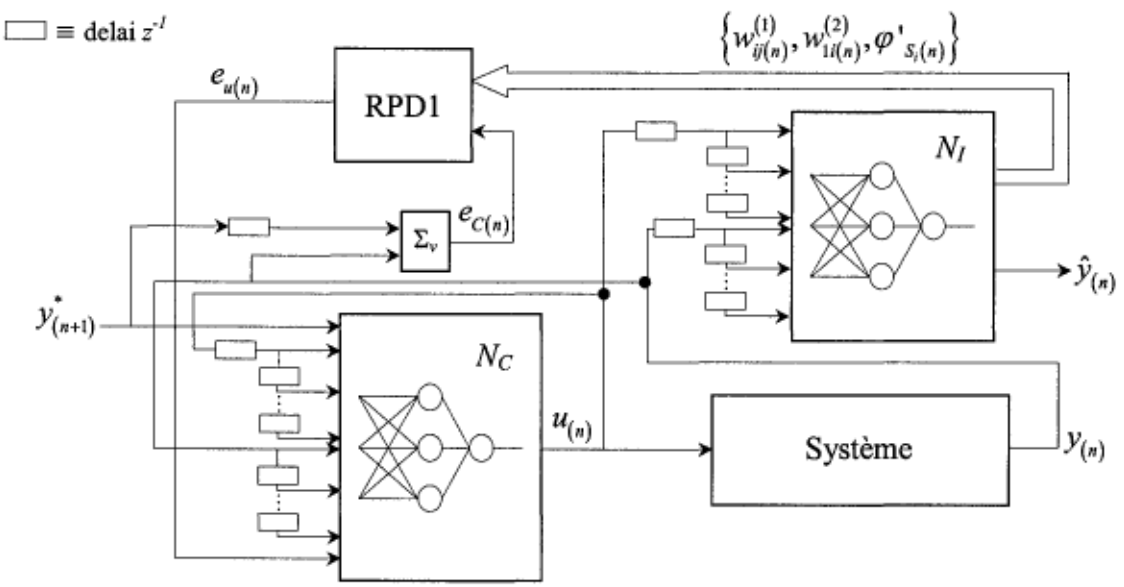


Figure 4.20. Schéma bloc complet de la loi de commande

4.6. Conclusion

Dans ce chapitre, nous avons tout d'abord justifié les différents choix technologiques et architecturaux en vue des critères d'ITGE visés. Les principaux choix que nous avons établis sont :

- modéliser un ASIC complètement analogique pour l'intégration de la loi de commande,
- tirer un maximum de profit du parallélisme inhérent des RNA et des algorithmes d'adaptation associés,
- utiliser une représentation différentielle des données pour une plus grande immunité au bruit,
- utiliser de circuits fonctionnant à temps continu.

Alors, nous avons présenté les différents modèles de circuits CMOS analogiques pour la modélisation au niveau de l'ITGE de la loi de commande développée dans les deuxième et troisième chapitres. Ces circuits ont été développés sous formes de modules de base qui peuvent être assemblés simplement pour réaliser les architectures des réseaux mis en jeu dans la loi de commande. Le circuit global comprend également les circuits de base nécessaires à l'adaptation en ligne des RNA selon l'algorithme RPD1 développé dans le troisième chapitre et retenu suivant les conclusions de ce même chapitre.

Certains circuits de base utilisés sont des circuits déjà publiés dans la littérature. Cependant, la thèse apporte aussi les contributions suivantes :

- un circuit différentiel complètement analogique avec adaptation continue en ligne pour la modélisation d'ITGE d'un poids synaptique,
- un circuit différentiel complètement analogique pour la modélisation d'ITGE d'un neurone non linéaire avec fonction tangente hyperbolique incluant le calcul de la dérivée de la fonction d'activation,
- un circuit différentiel complètement analogique pour la modélisation d'ITGE de l'algorithme RPD1,
- un circuit différentiel complètement analogique à temps continu pour la modélisation d'ITGE de la loi de commande développée dans la thèse.

Chapitre 5

Résultats de simulations

5.1. Introduction

5.1.1. Environnement de simulation

Le logiciel utilisé pour les simulations est Cadence®. Les dessins de masques respectent les règles de dessin de la technologie TSMC 0.35µm. Les dessins de masques des différents circuits présentés dans le quatrième chapitre sont obtenus en associant les dessins de masques de base des transistors, résistances et capacités réalisés avec les méthodes présentées en annexe D. Les dimensions des masques de base doivent être ajustées aux valeurs qui seront présentées dans ce chapitre pour chaque circuit.

5.1.2. Méthode d'évaluation

Le but final de ce chapitre est d'obtenir les résultats de simulations de la loi de commande complète au niveau des dessins de masques. Afin d'en arriver là, nous procédons en plusieurs étapes :

- simulation des circuits de base,
- simulation du réseau d'identification N_I seul (sans contrôleur),
- simulation de la loi de commande complète.

Dans chacune de ces étapes, les résultats de simulations des dessins de masques seront comparés, lorsque cela est possible, aux résultats de simulations des fonctions idéales dans Matlab®, ces derniers nous servant de résultats de référence pour l'évaluation de la précision. À ce stade, il s'agit surtout d'évaluer la précision obtenue avec les différents circuits, l'évaluation des autres critères d'ITGE (c'est-à-dire la consommation de puissance et la surface d'intégration) sera présentée dans la section 5.7. Le but de la comparaison avec les résultats obtenus dans Matlab® est d'utiliser la très grande précision de calcul de ce logiciel comme référence de précision.

5.2. Dessins de masques sur silicium

La simulation des dessins de masques est le dernier niveau de simulation numérique avant d'entreprendre les étapes finales de mise en fabrication du circuit. Les logiciels actuels de conception de circuits en microélectronique permettent à ce niveau de nombreuses analyses : effet des capacités parasites, sensibilité des circuits au procédé de fabrication, sensibilité aux variations de température, etc. Les simulations des dessins de masques correspondent alors au niveau le plus proche du circuit fabriqué que nous puissions modéliser. Cette étape nous garantit les meilleures chances, au niveau de la simulation, que le circuit fabriqué corresponde aux spécifications initiales. Les étapes menant à la fabrication d'un circuit intégré selon [Baker *et al.* '98] peuvent se décomposer en trois parties (*cf.* annexe F). La première mène à la réalisation des circuits sous forme de schématiques et à leur simulation. Cette étape, à l'exception des simulations, a été présentée dans le quatrième chapitre. Lorsque les résultats de simulations des schématiques sont satisfaisants, l'étape suivante est la modélisation des dessins des masques et la simulation du circuit global à ce niveau de modélisation. Enfin, lorsque les résultats de simulations des dessins de masques rencontrent les spécifications du concepteur, la dernière étape est la mise en fabrication du circuit, d'abord d'un prototype pour fins de vérifications matérielles puis la mise en fabrication à plus grande échelle si le prototype est satisfaisant. Notons que dans ce chapitre, nous ne présenterons que des résultats de simulations au niveau des dessins de masques. En effet, les différentes simulations que nous avons menées nous ont permis de constater que les résultats de simulations des schématiques et des dessins de masques étaient toujours très proches. Nous avons donc volontairement omis la présentation des résultats au niveau des schématiques. La conception et la simulation des dessins de masques nécessitent plusieurs étapes. Notons que pour

augmenter les chances de réussite et simplifier la conception, il est préférable de commencer par les dessins de masques des circuits de base et de les assembler ensuite pour réaliser progressivement les circuits de niveaux hiérarchiques plus élevés. Les différentes étapes de conception et de simulation des dessins de masques sont présentées en annexe D. La première étape consiste à modéliser les dessins de masques dans le logiciel. Il faut ensuite vérifier si les dessins réalisés satisfont les règles de dessin de la technologie. En effet, les dessins de masques sont soumis à des règles très précises, règles directement reliées au procédé de fabrication (dimensions maximales admissibles, espacement à respecter entre les matériaux, etc.). Lorsque les règles de dessin sont satisfaites, l'étape suivante consiste à extraire les capacités parasites. Cette étape vise à modéliser dans les dessins des masques les capacités parasites qui seront effectivement présentes dans le circuit fabriqué. Ensuite, l'étape de vérification des dessins de masques versus les schématiques nous permet de savoir si les dessins de masques réalisés correspondent effectivement aux schématiques de départ. Finalement, les simulations des dessins de masques du circuit peuvent être réalisées afin de vérifier si ceux-ci donnent les résultats escomptés.

Les différentes techniques de réalisation des dessins de masques des éléments électroniques de base, c'est-à-dire transistors, résistances et capacités, sont présentées en annexe D. L'exemple du dessin des masques de l'AOT est donné en annexe G.

5.3. Simulations des circuits de base

Dans cette section, nous présentons les résultats de simulations des circuits de base seuls. À ce niveau, comme nous l'avions déjà mentionné dans la section 4.3.4, nous nous sommes fixé l'objectif d'atteindre une précision d'au moins 1%, c'est-à-dire que l'erreur relative sur les fonctions de base décrites dans la section 4.4 demeure entre –1% et 1%.

5.3.1. AOP

Le circuit de l'AOP est présenté en annexe B. Ce circuit est nécessaire pour le circuit de compensation (CMFB) de l'AOT comme cela est montré dans l'annexe B. Les valeurs des dimensions des transistors pour la simulation de l'AOP sont regroupées dans le tableau 5.1.

Tableau 5.1. Dimensions des transistors de l'AOP

Transistor	*W/L* [µm/µm]	Transistor	*W/L* [µm/µm]
M_1	3.1/0.5	M_6	3.2/5.5
M_2 et M_3	12.5/0.5	M_7 et M_8	25/0.5
M_4 et M_5	35/0.5	M_9	50/0.5

La capacité C a pour valeur numérique 0.6µF.

Les simulations des dessins de masques de l'AOP ont donné les principales caractéristiques suivantes :

- un gain différentiel en boucle ouverte de 68dB,
- une marge de phase de 120°,
- une fréquence de transition de 60MHz.

5.3.2. AOT

Le schématique de l'AOT a été présenté dans la figure 4.2 au quatrième chapitre. Les différents paramètres nécessaires à la simulation de l'AOT sont présentés dans le tableau 5.2. La tension de mode commun est un paramètre qui dépend du circuit dans lequel l'AOT est utilisé. Lorsque la valeur de cette tension n'est pas indiquée, elle est nulle. Les courbes de réponses de gain et de phase en fréquence de l'AOT sont présentées en annexe H. L'AOT est principalement caractérisé par un gain DC de 90.4dB, une fréquence à gain unitaire de 9.74MHz et une marge de phase de 90.1°. Les analyses transitoires ont montré un taux de variation maximal de sortie de 3.9V/µs et un temps de saturation de sortie de 0.35µs. La plage dynamique de sortie s'étend sur l'intervalle $[-1,1.63]$ V pour chaque sortie de l'AOT. Ceci signifie donc une plage dynamique différentielle de sortie dans l'intervalle $[-2.63,2.63]$ V.

Tableau 5.2. Paramètres de simulation de l'AOT

Transistor	*W/L* [µm/µm]	Transistor	*W/L* [µm/µm]
M_1 à M_4	5.4/1	M_{15} et M_{16}	2.35/1
M_{R1} et M_{R2}	0.4/3.85	M_{17} et M_{18}	1/0.8
M_5	0.52/1	M_{19} et M_{20}	0.35/1
M_6 à M_9	3/1	M_{21} et M_{22}	10/0.6
M_{10}	0.4/1	M_{23} à M_{28}	0.57/1
M_{11} et M_{29}	14/1	M_{33} et M_{34}	4.27/0.8
M_{12} et M_{30}	3/1	M_{35} et M_{36}	5/0.4
M_{13} et M_{31}	50/0.4	M_{37} à M_{42}	6.2/1
M_{14} et M_{32}	23.5/0.4		
Paramètre	Valeur	Paramètre	Valeur
C_1	0.24µF	R	40kΩ
C_2	0.48µF		

5.3.3. Multiplieur

Le schématique du multiplieur a été présenté dans la figure 4.3. Les dimensions des transistors de ce circuit sont présentées dans le tableau 5.3.

Tableau 5.3. Dimensions des transistors du multiplieur

Paramètre	*W/L* [µm/µm]	Paramètre	*W/L* [µm/µm]
M_1 à M_4	10/7	M_9 à M_{12}	47/0.5
M_5 et M_8	0.4/7	M_{13} et M_{16}	47/0.5

La caractéristique principale du multiplieur à vérifier est la linéarité. Comme nous l'avions vu dans l'équation (4.7), c'est la différence des courants de sortie du multiplieur qui est proportionnelle au produit des deux tensions d'entrées v_x et v_y. Une bonne linéarité du multiplieur signifie que le coefficient de proportionnalité doit être le plus constant possible quelles que soient

les entrées du multiplieur. Une méthode d'évaluation courante dans ce cas est de faire varier une des deux entrées sur toute la plage de tension et de laisser l'autre entrée constante. Dans ce cas, la réponse de sortie du multiplieur devrait être représentée par une droite. La figure 5.1 présente les courbes de linéarité obtenues pour le multiplieur en variant v_x sur toute la plage de tension et en laissant v_y constante. Plusieurs valeurs réparties sur la plage dynamique de tension ont été considérées pour v_y.

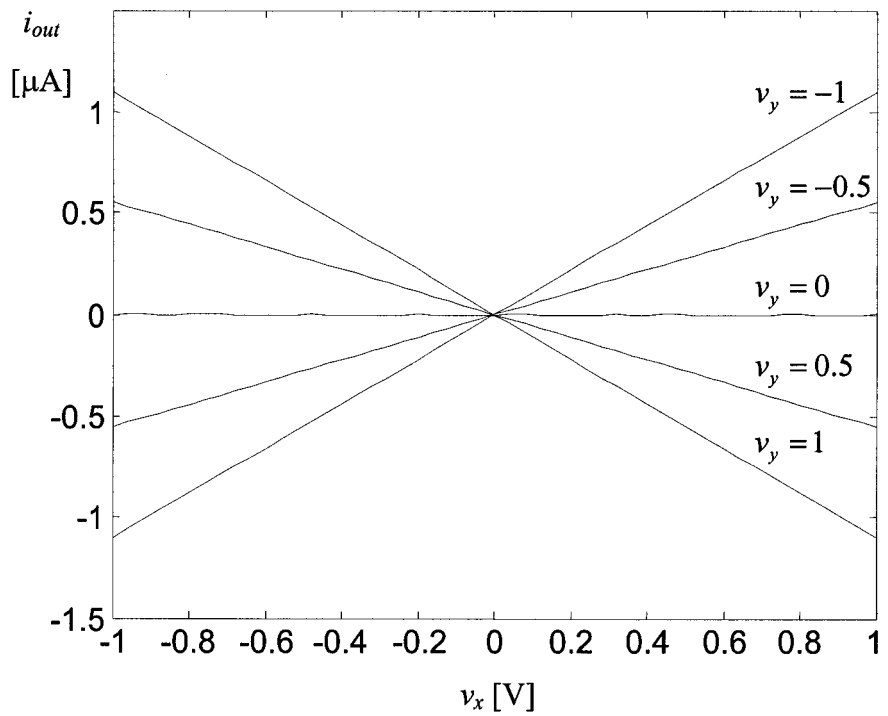


Figure 5.1. Courbes de linéarité du multiplieur à v_x variable et v_y constante

La figure 5.2 présente maintenant les courbes de linéarité obtenues en faisant varier v_y et en laissant cette fois v_x constante.

Les figures 5.1 et 5.2 nous permettent de constater visuellement la bonne linéarité du multiplieur sur toute la plage de tension, aussi bien sur v_x que sur v_y. Pour ces courbes, il n'y a pas véritablement de courbe de référence car la constante de proportionnalité idéale pourrait être prise à n'importe quel endroit des courbes.

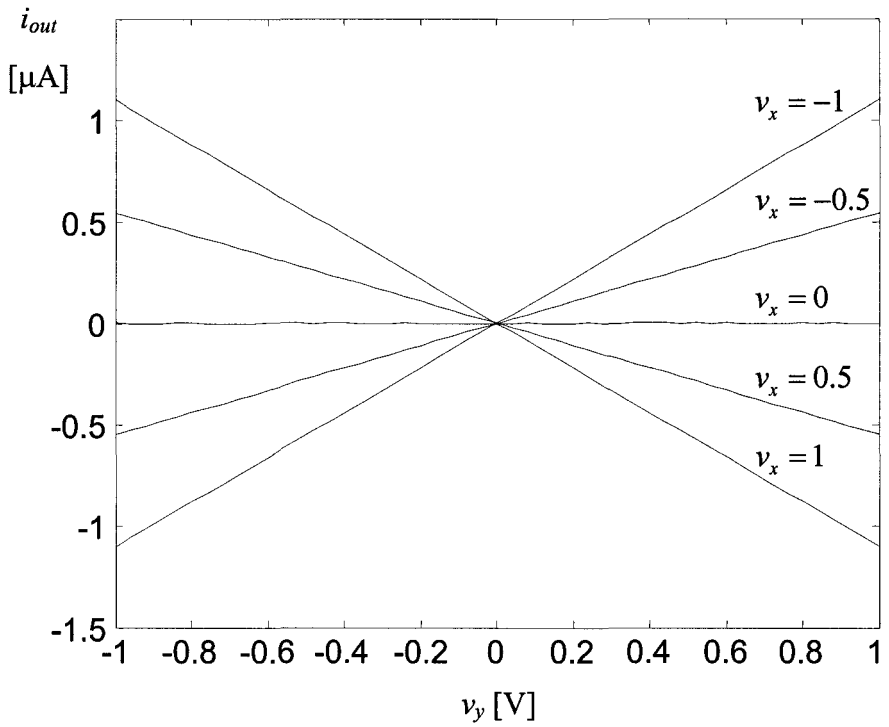


Figure 5.2. Courbes de linéarité du multiplieur à v_y variable et v_x constante

Afin d'évaluer quantitativement l'erreur de linéarité du multiplieur, nous prenons pour référence la constante de proportionnalité obtenue pour la droite joignant le maximum d'une courbe avec son minimum. L'erreur relative est alors calculée pour chaque courbe selon l'équation (4.3). Les figures 5.3 et 5.4 présentent respectivement les résultats obtenus à v_x constante et v_y constante. Notons que les cas où $v_x = 0$ ou $v_y = 0$ ne sont pas représentés pour des raisons évidentes. Ces figures nous permettent également de constater la très bonne symétrie du multiplieur. En effet, si nous observons par exemple la figure 5.3, les courbes d'erreurs obtenues pour les valeurs opposées de v_x sont bien symétriques par rapport à l'axe des abscisses. Les résultats des figures 5.3 et 5.4 permettent de constater que les erreurs relatives calculées pour le multiplieur pour les différents cas envisagés demeurent inférieures en valeur absolue à 0.006%. Ceci démontre une très bonne linéarité du multiplieur qui satisfait à notre critère de précision.

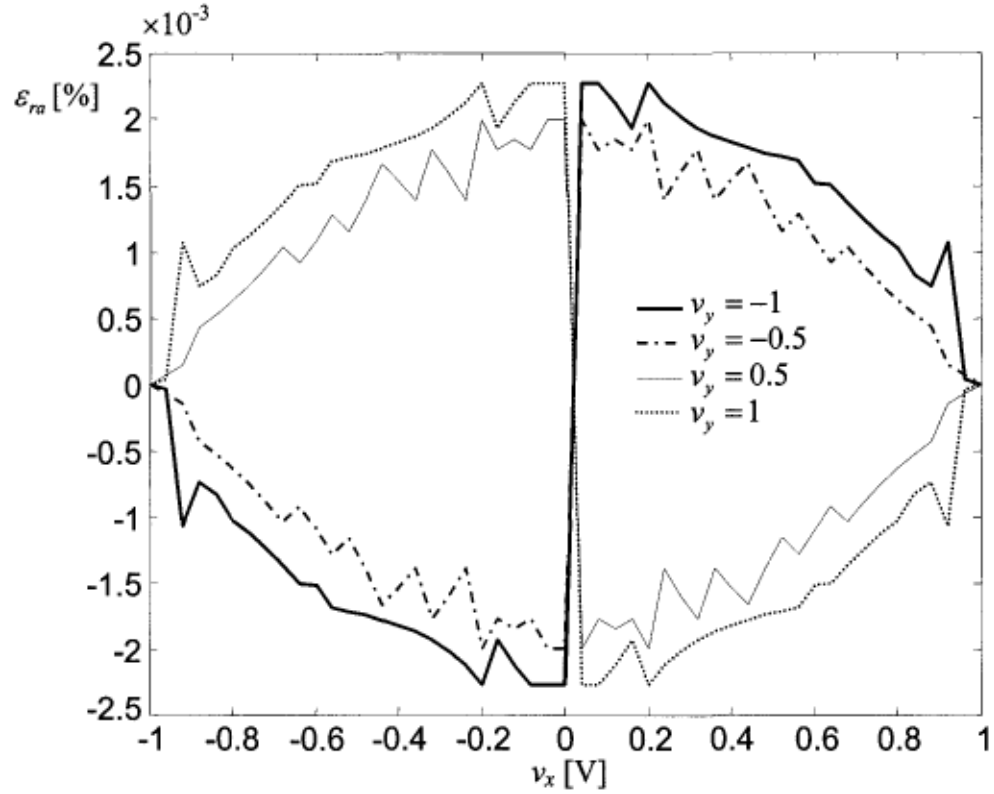


Figure 5.3. Erreurs relatives sur la sortie du multiplieur calculées pour v_x variable et v_y constante

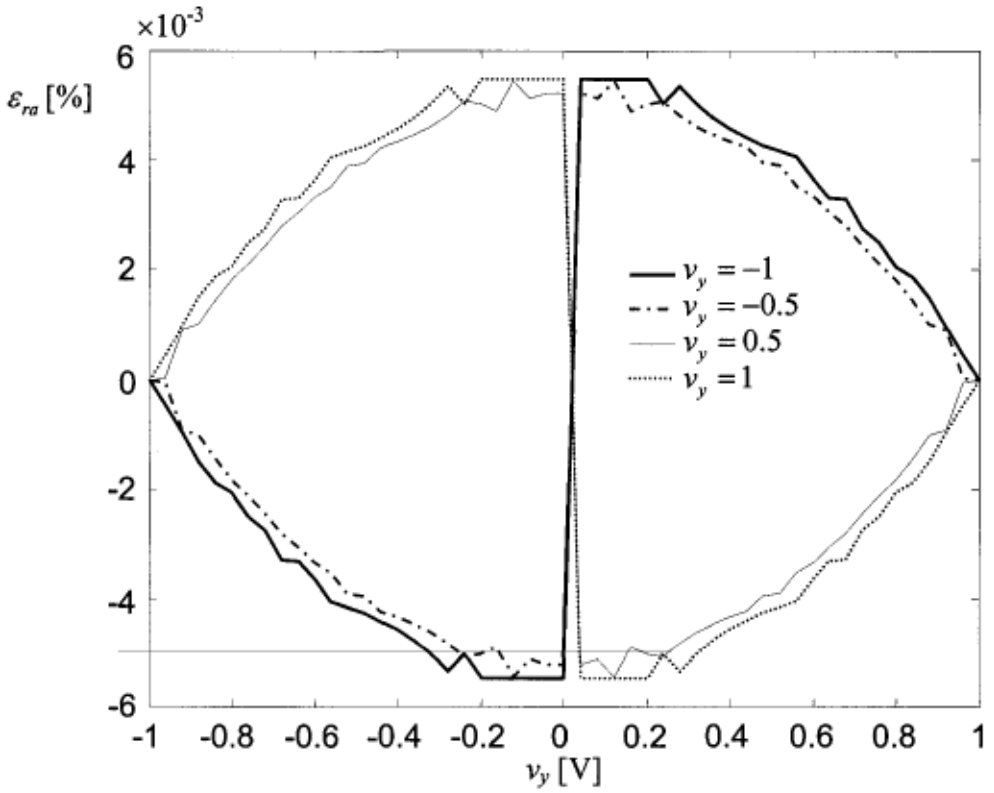


Figure 5.4. Erreurs relatives sur la sortie du multiplieur calculées pour v_y variable et v_x constante

5.3.4. CCT

Le circuit général de CCT a été présenté dans la figure 4.8. Comme nous l'avions mentionné, certains paramètres du circuit de CCT peuvent dépendre du nombre de multiplieurs connectés en aval. Le tableau 5.4 présente les paramètres des différents CCT mis en œuvre dans les figures du quatrième chapitre. Les valeurs utilisées des paramètres des CCT permettent de couvrir l'ensemble des exemples qui seront présentés dans les sections 5.4 et 5.5 de ce chapitre.

Tableau 5.4. Paramètres des différents CCT mis en oeuvre

Figures	Paramètres					
	M_1 [W/L]	M_2 [W/L]	R_1 [kΩ]	R_2 [kΩ]	v_{CMFB} [V]	v_G [V]
4.10	0.6/2	0.6/2	0.5	7.78	0.5	1.65
4.13, CCT_1	0.7/0.8	0.7/0.8	0.5	3.58	0	1.65
4.13, CCT_2	0.6/2	0.6/2	0.5	0.83	0.5	1.65
4.14	0.7/0.8	0.7/0.8	0.5	22.15	0	1.65
4.15	0.6/2	0.6/2	0.5	7.78	0.5	1.65
4.16	0.7/0.8	0.7/0.8	0.5	3.58	0	1.65
4.17 et 4.18	0.6/2	0.6/2	0.5	7.78	0.5	1.65
4.19, CCT_1 et CCT_2	0.6/2	0.6/2	0.5	7.78	0.5	1.65

Notons que la tension v_G a été fixée à 1.65V dans tous les cas pour simplifier les dessins des masques.

Afin de vérifier le bon fonctionnement des CCT pour les différentes configurations, nous avons utilisé le schéma de simulation de la figure 5.5. La tension v_x est constante égale à 1V tandis que la tension v_y est une rampe de tension croissante de –1V jusqu'à 1V. Ainsi, nous devrions idéalement obtenir en sortie une copie du signal d'entrée v_y qui nous servira de référence pour le calcul d'erreur.

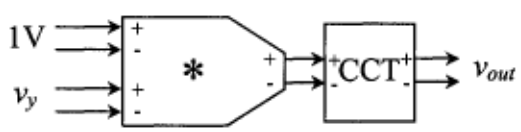


Figure 5.5. Schéma de simulation des CCT

Les figures 5.6 et 5.7 présentent les deux courbes d'erreurs relatives obtenues pour les paramètres des deux premières lignes du tableau 5.4.

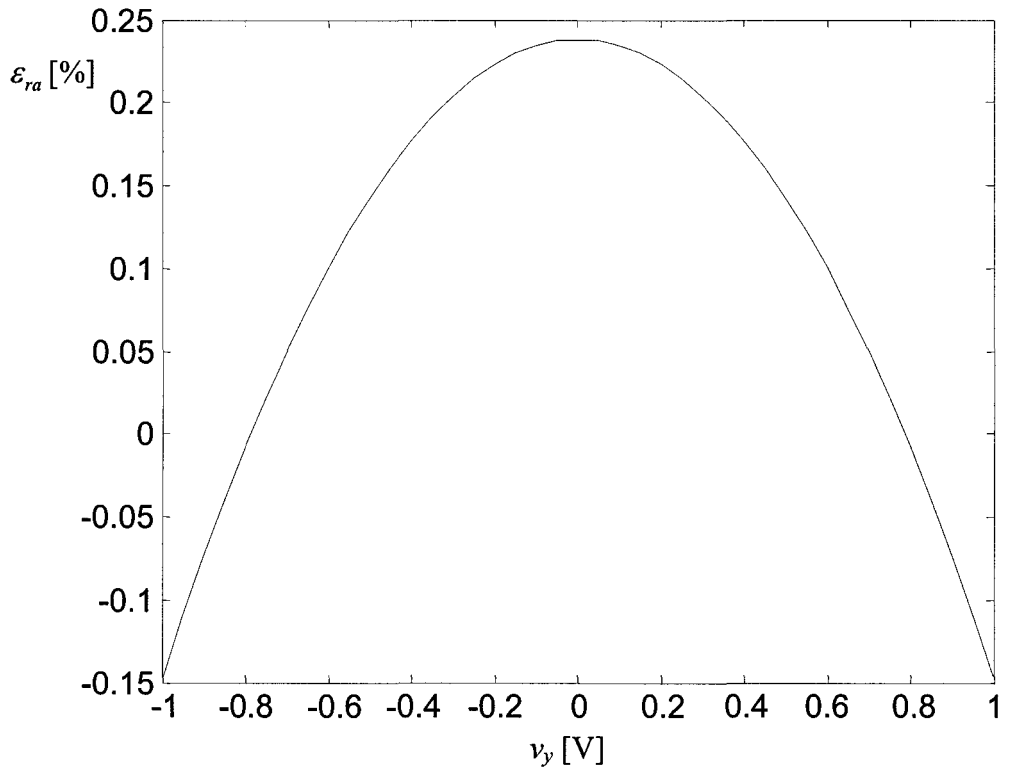


Figure 5.6. Erreur relative sur la sortie du CCT avec les paramètres de la première ligne du tableau 5.4

Les résultats des figures 5.6 et 5.7 nous permettent de constater que les erreurs relatives pour les deux cas considérés demeurent comprises entre -0.3 et 0.25% ce qui satisfait amplement notre critère de précision. Nous avons aussi vérifié par simulations que c'était le cas pour toutes les autres configurations du tableau 5.4.

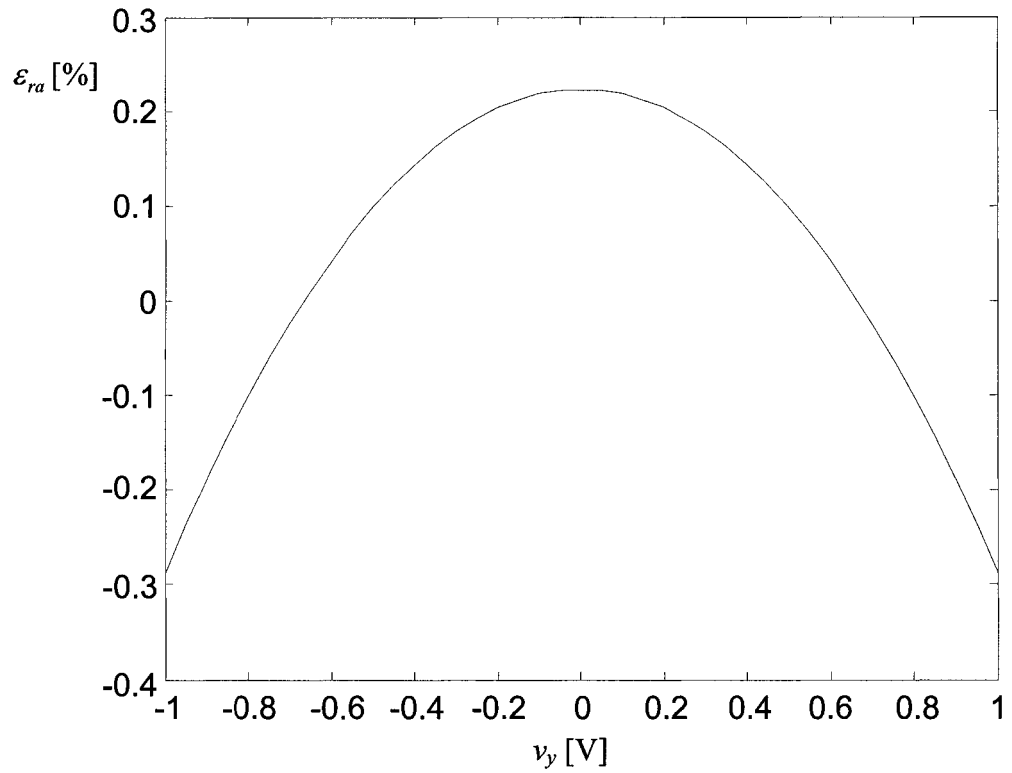


Figure 5.7. Erreur relative sur la sortie du CCT avec les paramètres de la deuxième ligne du tableau 5.4

5.3.5. Fonction tangente hyperbolique et sa dérivée

Les schématiques des circuits de la fonction tangente hyperbolique et de sa dérivée ont été respectivement présentés dans les figures 4.10 et 4.11. Le tableau 5.5 présente les paramètres de simulations pour ces deux circuits qui sont référencés par le numéro de figure.

Tableau 5.5. Paramètres de simulation pour la fonction tangente hyperbolique et sa dérivée

Figure	Paramètres	Valeur
4.10	M_1 à M_6 [µm/µm]	25/0.5
4.10	i_b [µA]	10
4.11	R_1 et R_2 (sommateur) [kΩ]	0.5

La figure 5.8 présente la sortie obtenue pour la fonction hyperbolique. La sortie de référence est obtenue d'après l'équation (4.15) en remplaçant les paramètres par les valeurs numériques :

$$i_{out} \approx 10 tanh\left(12.7\left(v_i^+ - v_i^-\right)\right) \quad [\mu A] \tag{5.3}$$

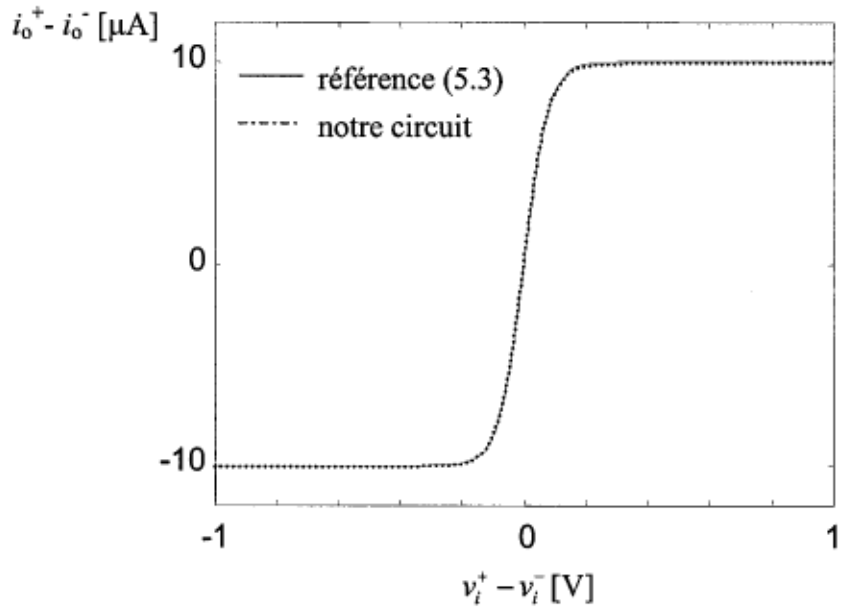


Figure 5.8. Sortie de la fonction tangente hyperbolique : référence et sortie de notre circuit

La figure 5.9 donne la courbe d'erreur relative obtenue sur cette sortie.

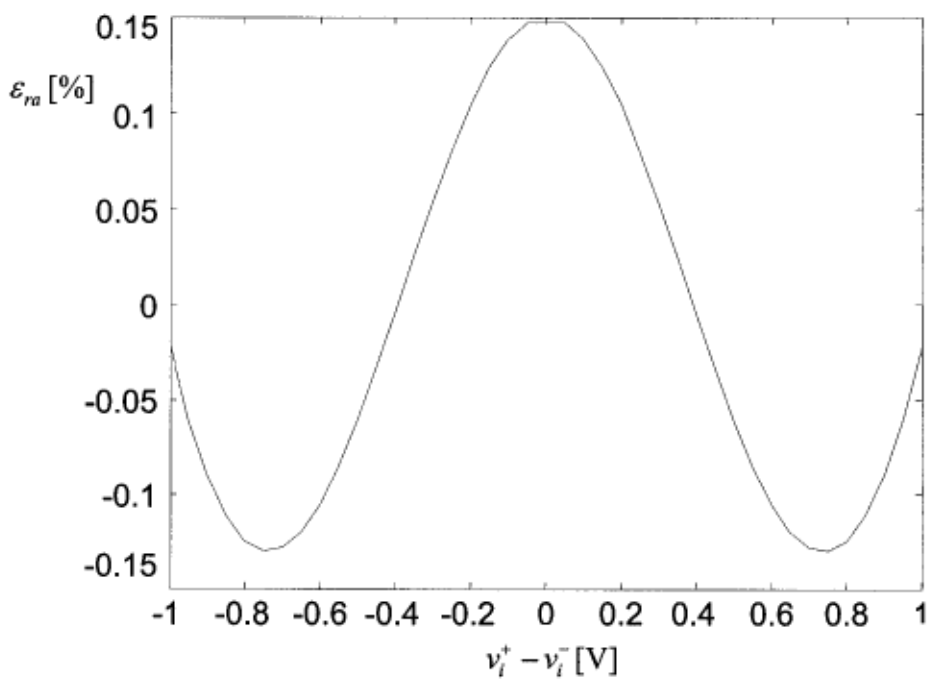


Figure 5.9. Erreur relative sur la sortie de la fonction tangente hyperbolique

La figure 5.10 présente l'erreur relative obtenue sur la dérivée de la fonction tangente hyperbolique.

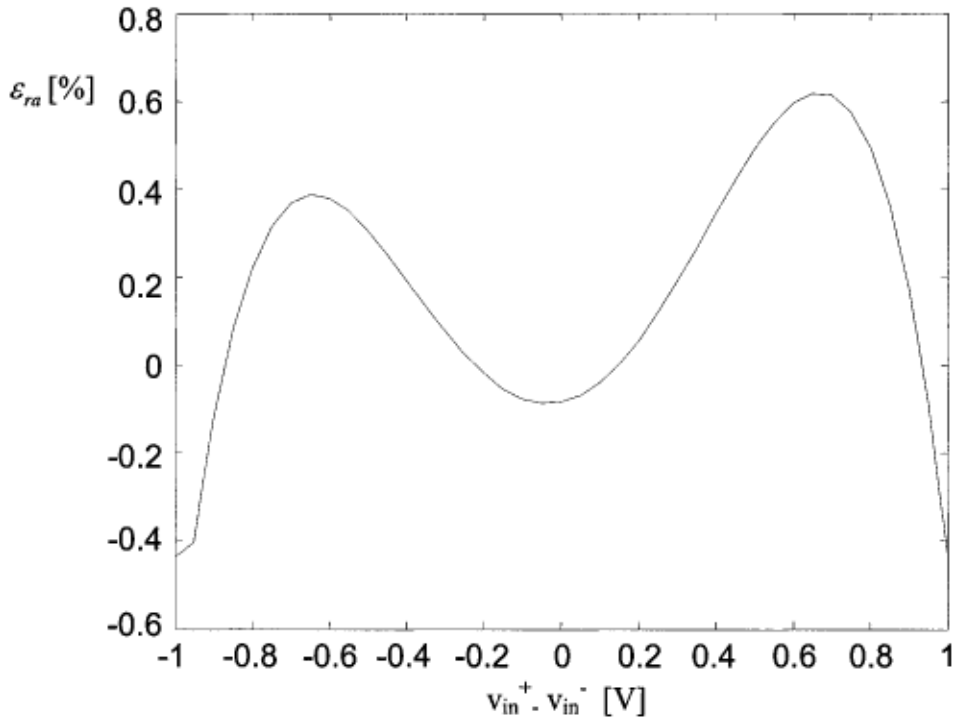


Figure 5.10. Erreur relative sur la dérivée de la fonction tangente hyperbolique

Les résultats permettent de constater que l'erreur relative sur la fonction tangente hyperbolique reste comprise entre –0.15% et 0.15% et que l'erreur relative sur sa dérivée reste comprise entre –0.5% et 0.6%.

5.3.6. Autres paramètres de simulations

Le tableau 5.6 présente les autres paramètres de simulations des circuits qui sont référencés par le numéro de figure.

La tension v_{ech} a été fixée à 1.65V afin de simplifier les dessins de masques. Cependant, comme nous l'avions mentionné dans le quatrième chapitre, cette tension peut servir de paramètre externe d'ajustement de la période d'échantillonnage équivalente à la constante d'intégration.

Tableau 5.6. Autres paramètres de simulation

Figure	Paramètres	Valeurs
4.11	v_{G1} [V]	1.65
	M_i [W/L]	0.5/12
	C_i [pF]	1.01
	R_1 [kΩ] dans Σ_v	3
	R_2 [kΩ] dans Σ_v	6
4.12	M_i [W/L]	0.5/12
	C_i [pF]	1.01
	v_{ech} [V]	1.65

5.4. Résultats de simulations

Dans cette section, nous présentons des résultats de simulations de la loi de commande, incluant l'algorithme RPD1, au niveau des dessins de masques. Nous utilisons les deux systèmes présentés dans les sections 3.4.2.3 et 3.4.2.7. Le système synthétique présenté dans la section 3.4.2.3 représente une gamme d'application très intéressante pour des systèmes rapides. Le modèle du moteur à courant continu avec frottements non linéaires de la section 3.4.2.7 représente quant à lui un exemple concret de système largement répandu dans l'industrie. Le but est de démontrer, au niveau de la simulation des dessins de masques, la fonctionnalité du circuit pour l'identification et le contrôle de ces deux systèmes.

5.4.1. Premier système : système synthétique

5.4.1.1. Identification

Comme nous l'avions mentionné dans la section 3.4.2.3, une phase préalable d'identification sans contrôleur est nécessaire afin d'éviter les problèmes d'instabilité au démarrage. Lors de cette phase, le système est soumis à une entrée sinusoïdale définie par :

$$u_{(n)} = 0.5\sin(2\pi n T_s) \tag{5.5}$$

Les paramètres utilisés sont les mêmes que dans la section 3.4.2.3.

La figure 5.11 présente les sorties estimées par l'algorithme programmé dans Matlab® et par notre circuit analogique. La figure 5.12 présente les erreurs d'estimation e_I correspondantes.

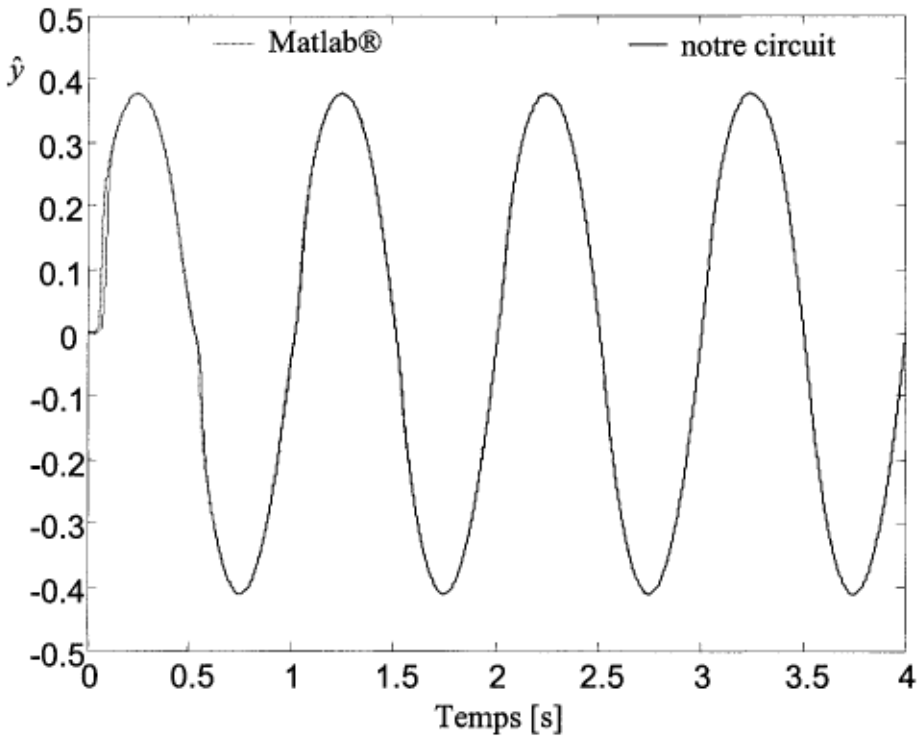


Figure 5.11. Comparaison des sorties estimées obtenues avec Matlab® et notre circuit, premier système

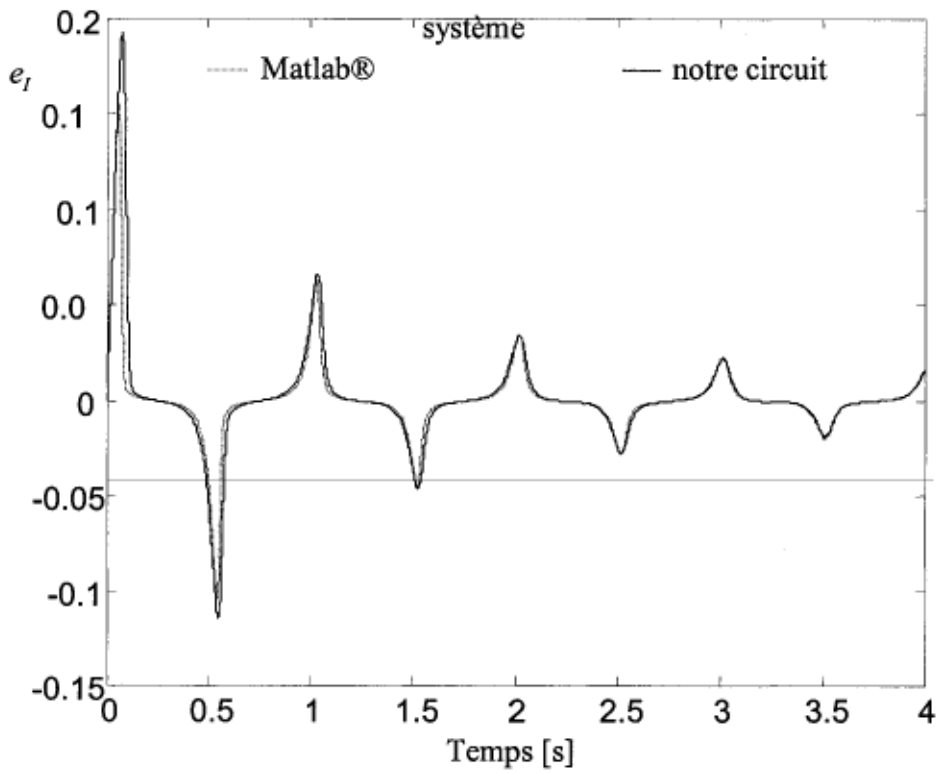


Figure 5.12. Comparaison des erreurs d'estimation obtenues avec Matlab® et notre circuit, premier système

5.4.1.2. Contrôle

Pour la partie contrôle, nous utilisons les deux trajectoires de référence définies par :

$$y^*_{1(n)} = 0.5\sin(4\pi nT_s) \tag{5.6}$$

$$y^*_{2(n)} = 0.3\sin(4\pi nT_s) + 0.2\sin(8\pi nT_s) \tag{5.7}$$

La figure 5.13 présente les sorties obtenues avec Matlab® et notre circuit analogique pour la première trajectoire de référence. La figure 5.14 présente les erreurs de commande correspondantes. Les figures 5.15 et 5.16 présentent les résultats analogues obtenus pour la deuxième trajectoire de référence. Les valeurs des paramètres de simulations sont les mêmes que celles présentées dans la section 3.4.2.3 et $\eta_C = 0.3$ pour les deux trajectoires de référence.

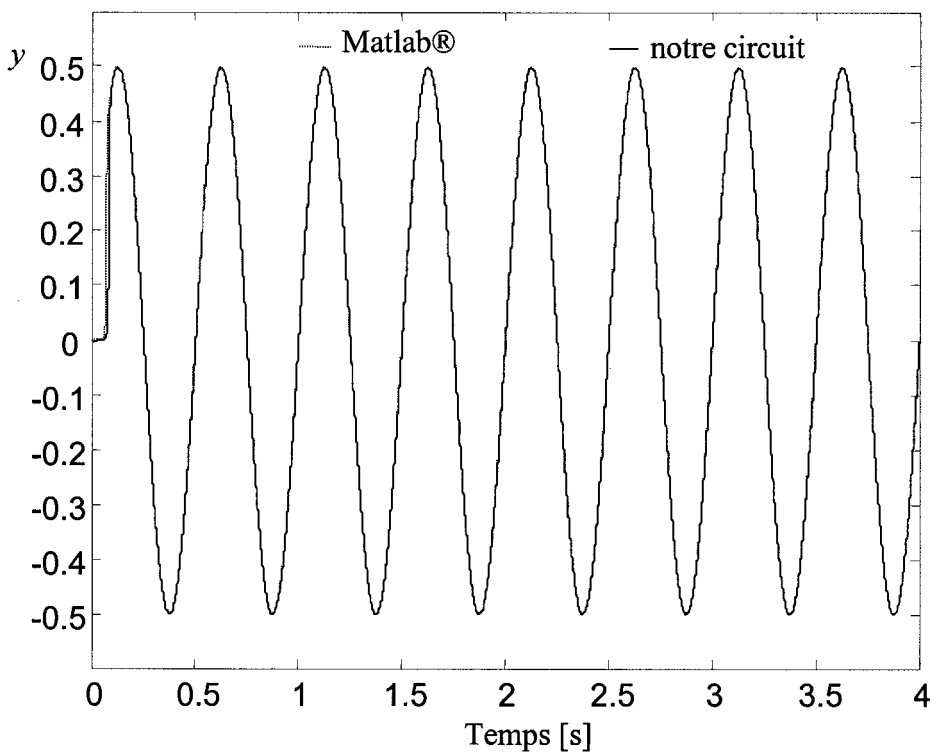


Figure 5.13. Comparaison des sorties obtenues avec Matlab® et notre circuit pour la première trajectoire de référence, premier système

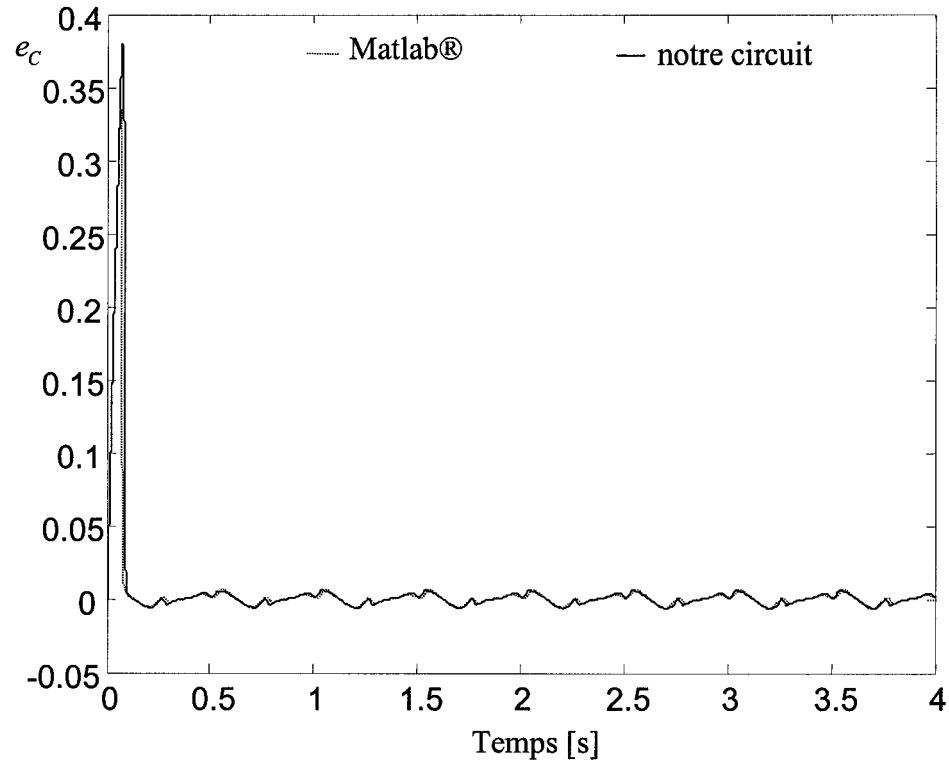


Figure 5.14. Comparaison des erreurs de commande obtenues avec Matlab® et notre circuit pour la première trajectoire de référence, premier système

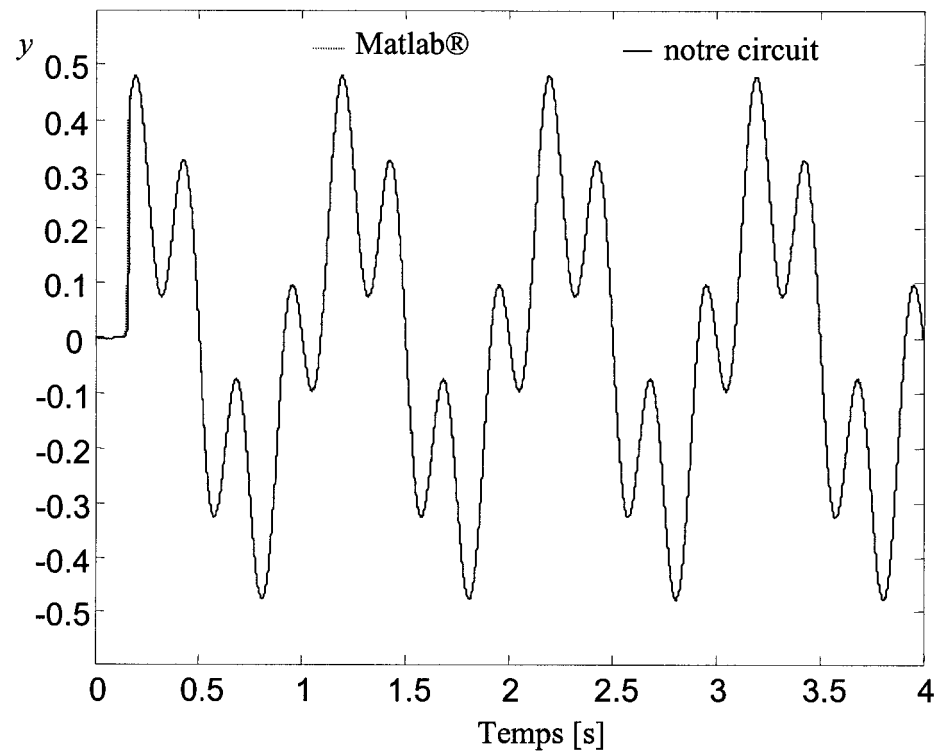


Figure 5.15. Comparaison des sorties obtenues avec Matlab® et notre circuit pour la deuxième trajectoire de référence, premier système

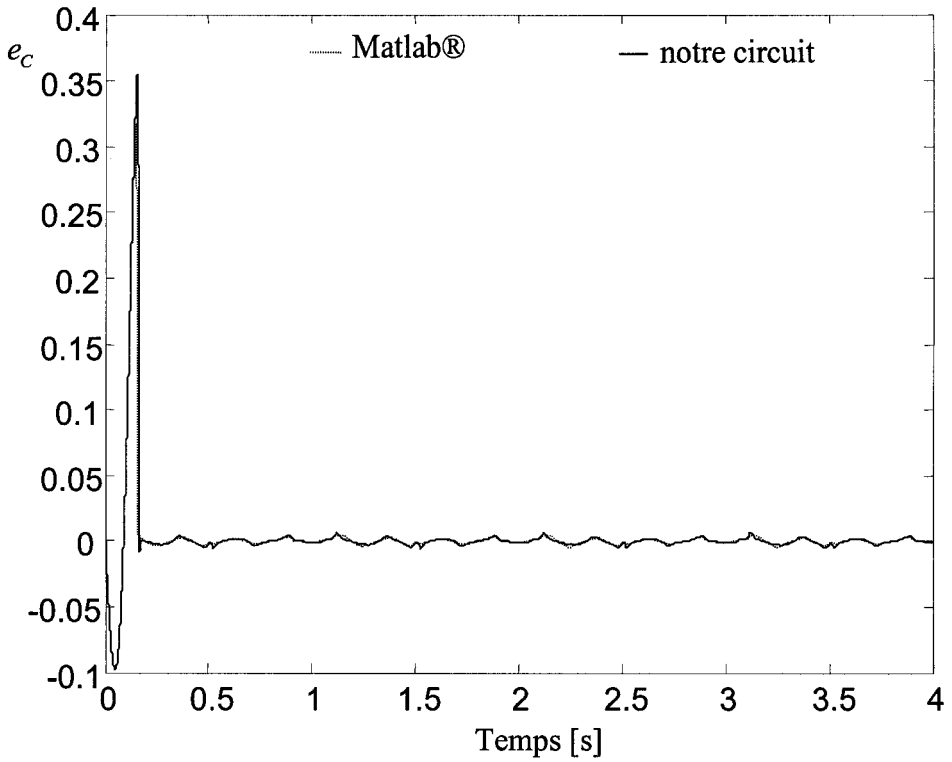


Figure 5.16. Comparaison des erreurs de commande obtenues avec Matlab® et notre circuit pour la deuxième trajectoire de référence, premier système

5.4.2. Deuxième système : moteur à courant continu avec frottements non linéaires

5.4.2.1. Identification

Comme nous l'avions mentionné dans la section 3.4.2.7, une phase préalable d'identification sans contrôleur est nécessaire afin d'éviter les problèmes d'instabilité au démarrage. Lors de cette phase, le système est soumis à une entrée sinusoïdale définie par

$$u_{(n)} = 0.1\sin\left(2\pi n T_s\right) \tag{5.5}$$

Les paramètres utilisés sont les mêmes que dans la section 3.4.2.7. La figure 5.17 présente les sorties estimées par l'algorithme programmé dans Matlab® et par notre circuit analogique. La figure 5.18 présente les erreurs d'estimation e_I correspondantes.

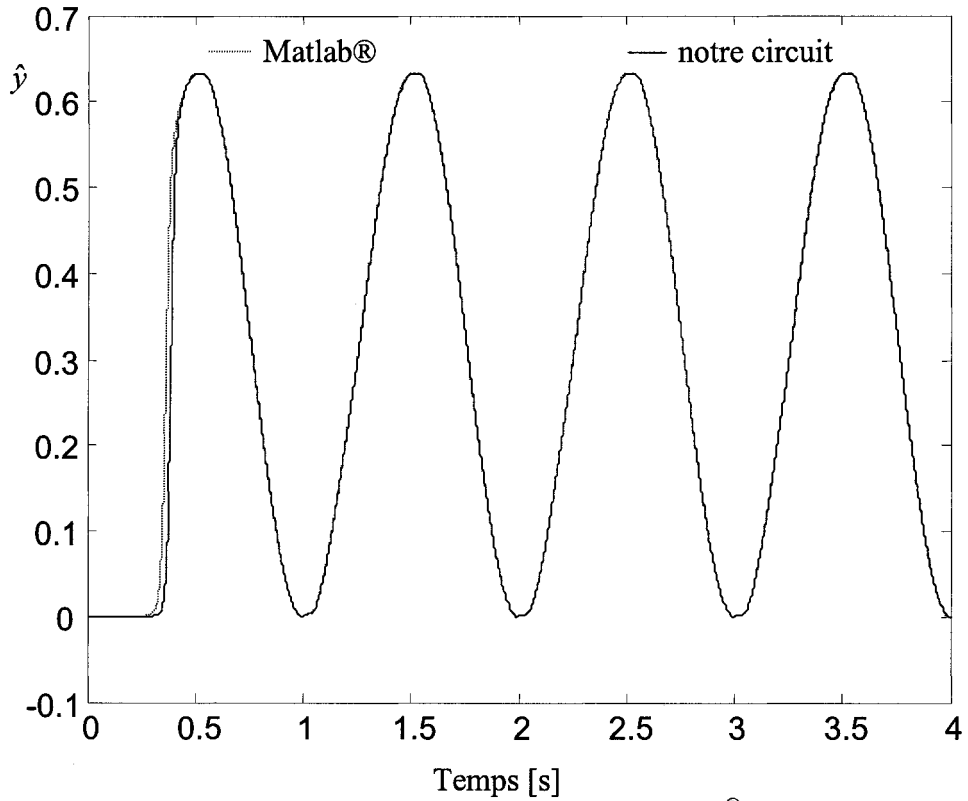


Figure 5.17. Comparaison des sorties estimées obtenues avec Matlab® et notre circuit, moteur CC avec frottements

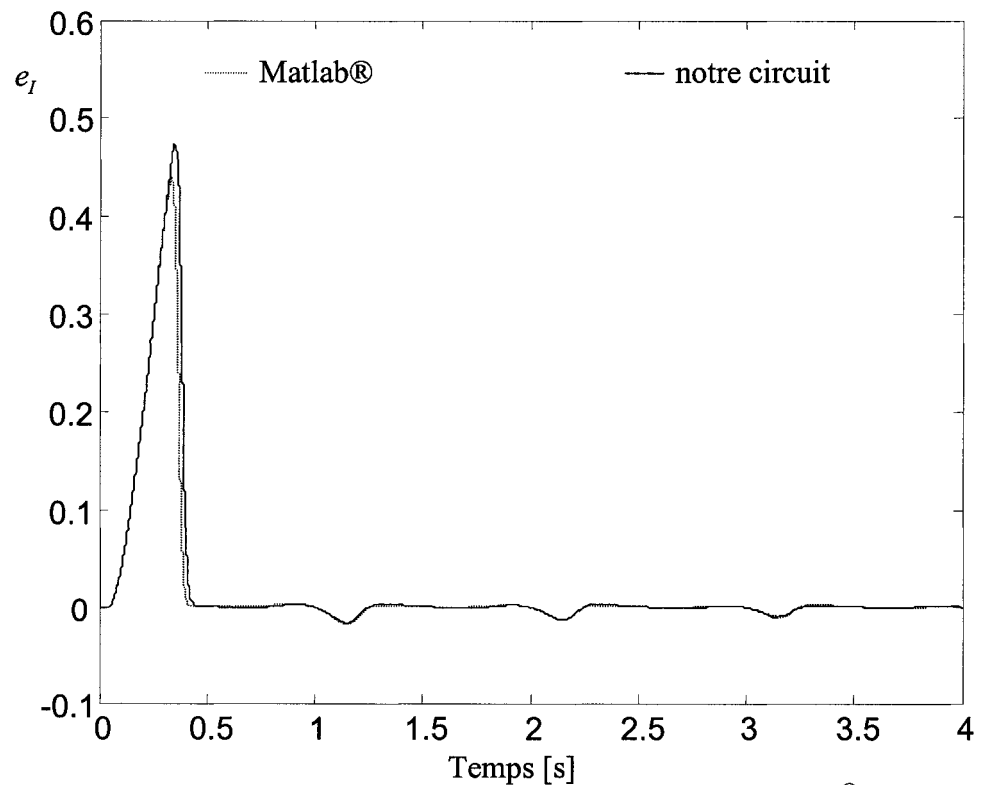


Figure 5.18. Comparaison des erreurs d'estimation obtenues avec Matlab® et notre circuit, moteur CC avec frottements

5.4.2.2. Contrôle

Pour la partie contrôle, nous utilisons les deux trajectoires de référence décrites dans la section 3.4.2.7. La figure 5.19 présente les sorties obtenues avec Matlab® et notre circuit analogique pour la première trajectoire de référence. La figure 5.20 présente les erreurs de commande correspondantes. Les figures 5.21 et 5.22 présentent les résultats analogues obtenus pour la deuxième trajectoire de référence. Les valeurs des paramètres de simulations sont les mêmes que celles présentées dans la section 3.4.2.7 et $\eta_C = 0.015$ pour les deux trajectoires de référence.

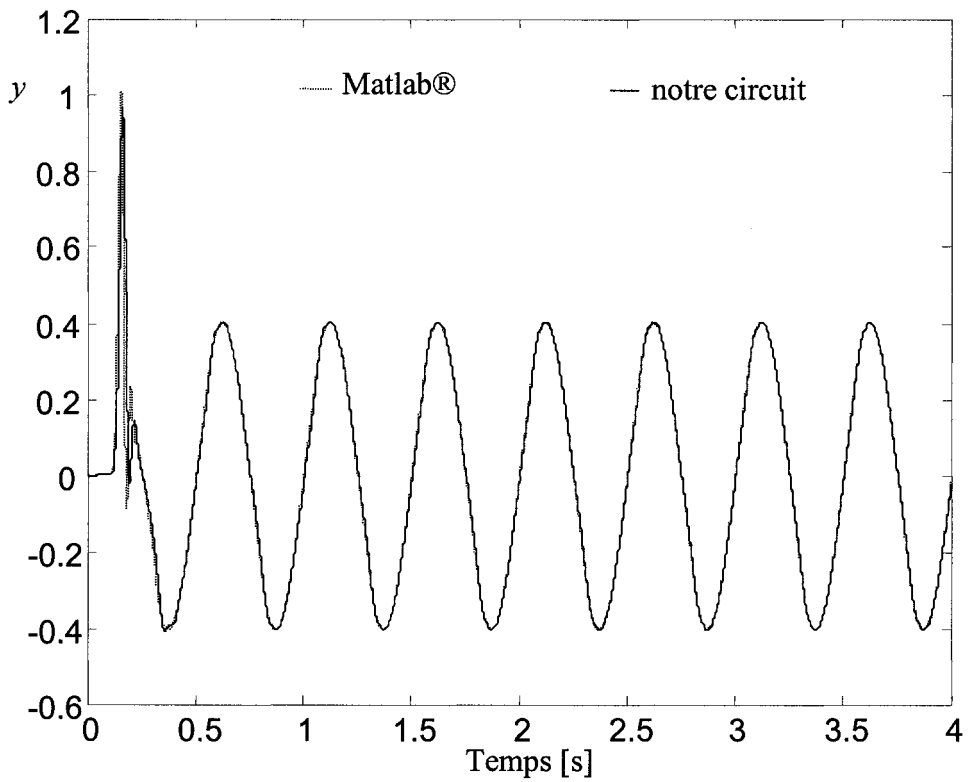


Figure 5.19. Comparaison des sorties obtenues avec Matlab® et notre circuit pour la première trajectoire de référence, moteur CC avec frottements

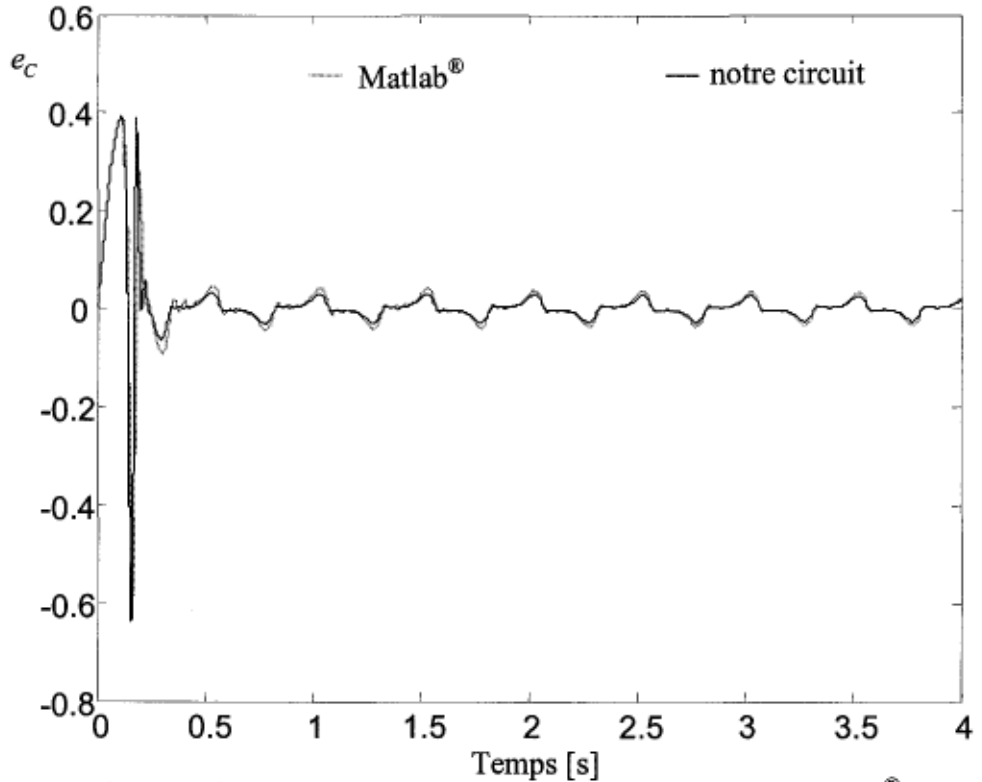


Figure 5.20. Comparaison des erreurs de commande obtenues avec Matlab® et notre circuit pour la première trajectoire de référence, moteur CC avec frottements

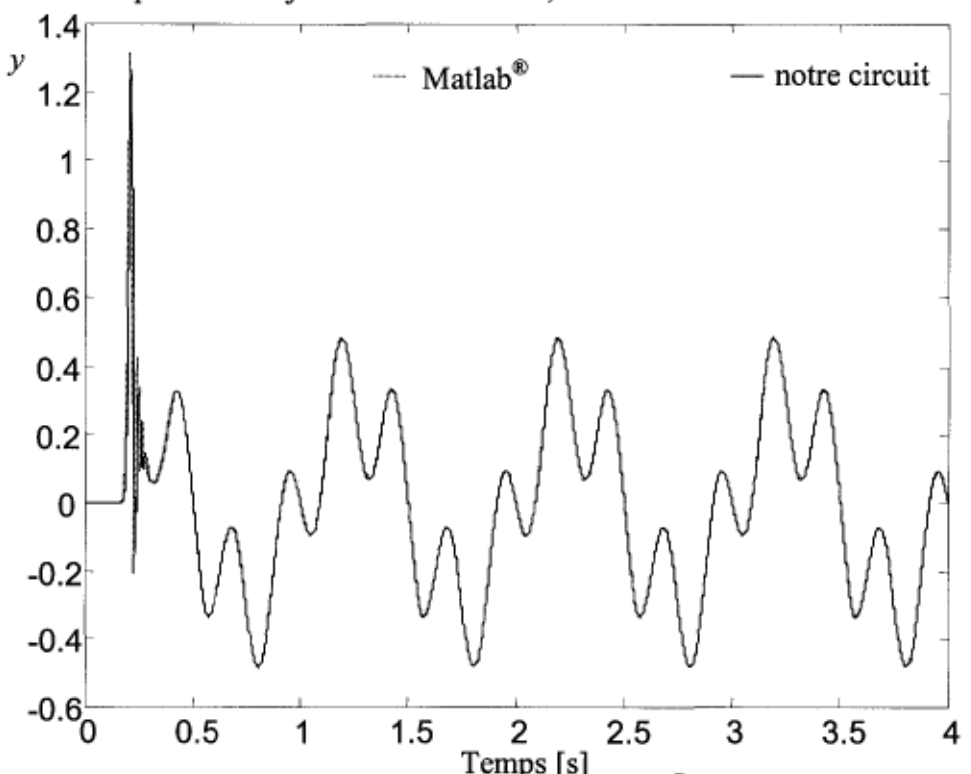


Figure 5.21. Comparaison des sorties obtenues avec Matlab® et notre circuit pour la deuxième trajectoire de référence, moteur CC avec frottements

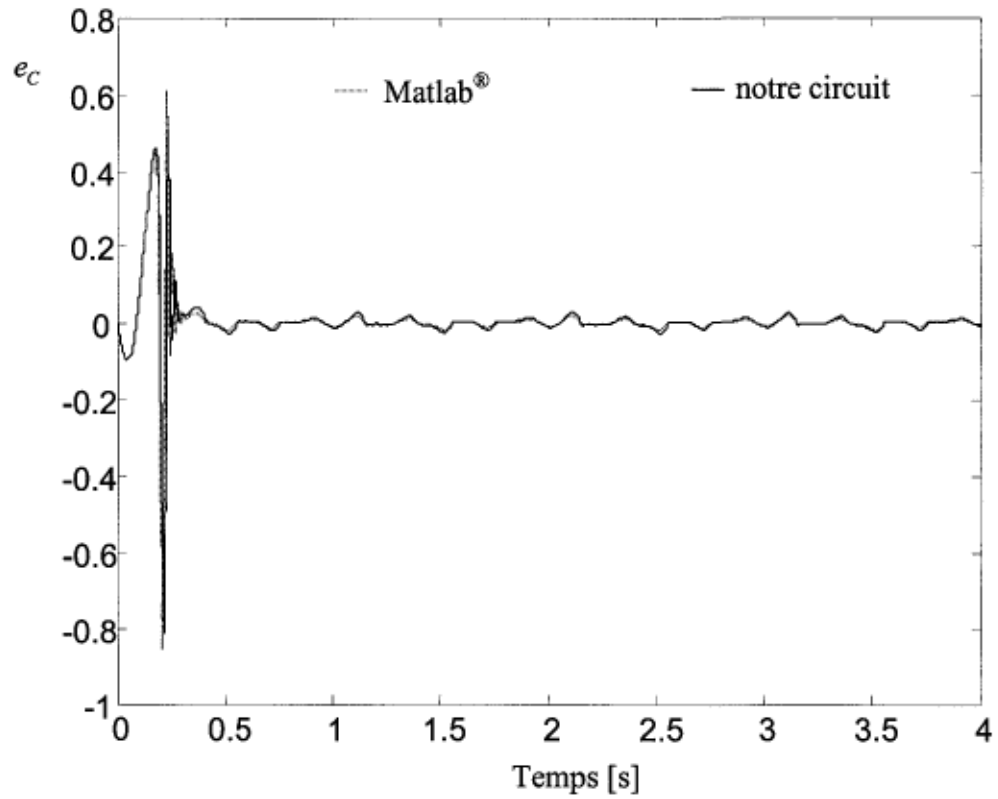


Figure 5.22. Comparaison des erreurs d'estimation du deuxième système par Matlab® et notre circuit pour la deuxième trajectoire de référence, moteur CC avec frottements

5.5. Sensibilité des circuits aux défauts de fabrication

L'augmentation du rendement de fabrication des circuits intégrés est très importante, spécialement pour les géométries de plus faibles dimensions. Par exemple, il est prédit que le rendement de fabrication n'excédera pas 60% pour les technologies de 0.1μm et moins. Une grande partie des facteurs influençant le rendement de fabrication sont hors du contrôle du concepteur. Cependant, certaines analyses existent afin de vérifier, au niveau de la simulation, la sensibilité d'un circuit au procédé de fabrication, à la température et aux variations d'alimentation. Citons les trois méthodes communément utilisées : les simulations aux coins (*ang. corners simulations*), les analyses de Monte Carlo et la modélisation de surface de réponse (*ang. response surface modeling*). Les simulations aux coins représentent la méthode la plus utilisée. Il s'agit de déterminer les modèles extrêmes, *i.e* les coins, des élément électroniques de base (c'est-à-dire transistors) du procédé de fabrication et de simuler le circuit en utilisant ces

différents modèles. Le principal inconvénient de cette méthode est de nécessiter les modèles des coins. Ces modèles sont liés au procédé de fabrication. Cependant, dans la technologie utilisée, les modèles sont disponibles, par exemple dans les librairies du logiciel Cadence®.

L'étude de sensibilité est très importante, spécialement pour les circuits analogiques. Pour un circuit analogique tel que celui présenté dans la thèse, trois facteurs peuvent affecter de manière importante le fonctionnement du circuit :

- le procédé de fabrication qui impose notamment des variations des caractéristiques des transistors,
- la température,
- les variations de l'alimentation.

Au niveau des deux premiers points, nous avons effectué les simulations des différents coins disponibles dans le logiciel Cadence® pour la technologie CMOS 0.35µm, et ce, pour différentes valeurs de la température. Les cinq différents coins sont obtenus en combinant les modèles extrêmes des transistors de type NMOS et PMOS :

- « TT » : modèle standard de la technologie,
- « SS » : NMOS lent et PMOS lent,
- « SF » : NMOS lent et PMOS rapide,
- « FS » : NMOS rapide et PMOS lent,
- « FF » : NMOS rapide et PMOS rapide.

Pour chaque modèle de coin, la simulation du circuit pour la commande du système synthétique de la section 5.4.1 avec la deuxième référence a été réalisée pour différentes valeurs de la température. Dans chaque cas, nous avons considéré les résultats satisfaisants lorsque la courbe de sortie du système ne présente pas, après convergence de la loi de commande, d'écart relatif absolu supérieur à 1% par rapport au modèle « TT » à 25°C qui nous sert de référence. Cette condition s'écrit :

$$\left| \frac{y_{(n)} - y_{(n),TT,25}}{y_{(n),TT,25}} \right| \leq 0.01 \qquad \forall n > n_{cv} \qquad (5.6)$$

où $y_{(n),TT,25}$ est la valeur de $y_{(n)}$ pour le modèle TT à 25°C et $y_{(n)}$ désigne dans cette équation la valeur de $y_{(n)}$ pour le modèle de coin et la température considérés.

Le tableau 5.7 présente les résultats obtenus.

Tableau 5.7. Résultats de simulations des différents coins pour différentes valeurs de la température : 'S' = satisfaisant ; 'E' = échec.

Modèle de coin	Température (°C)					
	5	15	25	35	45	60
TT	E	S	S	S	S	S
SS	E	S	S	S	S	E
SF	E	S	S	S	S	E
FS	E	S	S	S	S	E
FF	E	S	S	S	S	S

Ces résultats nous permettent de constater que le circuit présente, au niveau de la simulation, une robustesse aux variations du procédé de fabrication et de la température lui permettant de maintenir une bonne qualité de fonctionnement pour les modèles extrêmes (coins) et dans une gamme de températures allant approximativement de 10 à 50°C.

De plus, le PSRR (ang. Power Supply Rejection Ratio) du circuit supérieur à 120dB permet une grande insensibilité du circuit aux variations de l'alimentation.

5.6. Évaluation des circuits

Dans cette section, nous évaluons, au niveau de la simulation, les critères d'ITGE qui n'ont pas encore été évalués : la surface d'intégration et la consommation de puissance. Comme nous l'avons vu dans ce chapitre, certains paramètres des circuits peuvent dépendre de la dimension des réseaux. Pour notre évaluation, nous considérons le cas du moteur à courant continu avec frottements non linéaires traité dans la section 5.4.2. Les paramètres de simulations sont donnés dans les tableaux 5.1 à 5.6. Les circuits de base constituant le réseau N_C étant les

mêmes que ceux constituant le réseau N_I, les paramètres de simulations sont les mêmes. Le circuit de calcul de e_u selon l'algorithme RPD_1 dans ce cas de simulation ($a = b = 2$) est présenté dans la figure 5.23. Notons également que dans ce cas, aucun biais n'est nécessaire, l'entrée supplémentaire constante et égale à l'unité a donc été retirée.

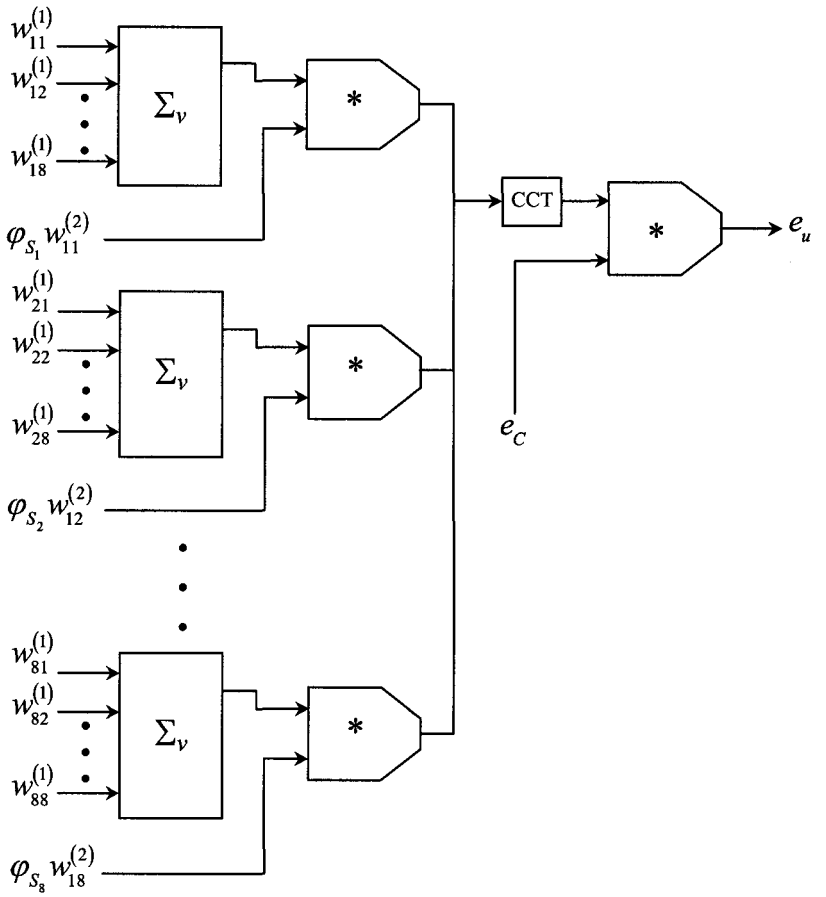


Figure 5.23. Circuit de calcul de e_u selon RPD1 dans le cas où $a = b = 2$

Notons que le sommateur de tension reprend le circuit présenté dans la figure 4.5 mais adapté au cas à huit entrées. Les résistances dans les sommateurs sont toutes égales à 0.5kΩ. Les valeurs des paramètres du CCT de la figure 5.23 sont les mêmes que celles du CCT de la figure 4.16 pour le cas à huit entrées (*cf.* tableau 5.4). Le tableau 5.8 présente le nombre de blocs de base de chaque type nécessaires dans le cas étudié. Dans ce tableau et ceux qui suivront, le numéro de figure correspondant au circuit est indiqué à droite du nom du circuit entre parenthèses.

Tableau 5.8. Nombres de blocs de base nécessaires dans le cas étudié

Bloc de base	N_I (4.18)	N_C (4.19)	Calcul e_u (5.23)
Poids caché (4.17)	40	40	0
Poids de sortie (4.16)	8	8	0
Neurone caché (4.14)	8	8	0
Neurone de sortie (4.15)	1	1	0
Multiplieur (4.3)	1	1	5
Sommateur (4.5)	1	0	2
CCT (4.8)	1	1	3

5.6.1. Circuits de base

Le tableau 5.9 présente les surfaces d'intégration et les puissances consommées de chacun des circuits de base utilisés obtenues avec le simulateur. Pour les CCT, le premier nombre entre parenthèses donne le nombre de multiplieurs connectés à l'entrée. Tous ces circuits ont été présentés dans le quatrième chapitre. Cette fois encore, nous distinguons les différents cas pour les CCT comme nous l'avions fait dans le tableau 5.4. Pour le montage sommateur nous distinguons également le cas à deux entrées du cas à huit entrées tel qu'utilisé dans la figure 5.23. La consommation de l'AOT a été mesurée pour le montage sommateur de la figure 4.5 avec toutes les résistances égales à 0.5kΩ. En fonction des surfaces d'intégration des circuits de base présentées dans le tableau 5.9, le tableau 5.10 présente les surfaces d'intégration totales des blocs de base du tableau 5.8. En fonction des puissances consommées des circuits de base présentées dans le tableau 5.9, le tableau 5.11 présente les consommations de puissance totales pour les blocs de base du tableau 5.8.

Tableau 5.9. Surface d'intégration et consommation de puissance des circuits de base

Circuit / Figure	Consommation de puissance [mW]	Surface d'intégration [μm^2]
AOT (4.2)	0.85	18840
Multiplieur (4.3)	0.055	850
Sommateur 2 entrées (4.5)	0.85	18860
Sommateur 8 entrées (4.5)	0.89	18905
CCT (4.8, 4.11, 4.16, 4.17 et 5.23)	1.76	37728
CCT_1 (4) (4.14)	1.76	37749
CCT_2 (4.14)	1.76	37692
CCT (8) (Fig. 4.15)	1.76	37723
Tangente hyperbolique (4.10)	0.03	838
Dérivée de tanh (4.11)	0.90	18875
Adaptation des poids (4.12)	1.78	37695
Délai (4.13)	0.86	18900

Tableau 5.10. Surfaces d'intégration des blocs de base présentés dans le tableau 5.8 pour N_I, N_C et e_u

Bloc de base	Surface unitaire [μm^2]	Nombre d'unités			Surface totale [μm^2]		
		N_I	N_C	e_u	N_I	N_C	e_u
Poids caché (4.17)	77973	40	40	0	3118920	3118920	0
Poids de sortie (4.16)	77128	8	8	0	617024	617024	0
Neurone caché (4.14)	95145	8	8	0	761160	761160	0
Neurone de sortie (4.15)	37723	1	1	0	37723	37723	0
Multiplieur (4.3)	850	1	1	5	850	850	4250
Sommateur (4.5)	18860	1	0	2	18860	0	37720
CCT (4.8)	37728	1	1	3	37728	37728	113184

Tableau 5.11. Consommation de puissance des blocs de base présentés dans le tableau 5.8 pour N_I, N_C et e_u

Bloc de base	Consommation unitaire [mW]	Nombre d'unités			Consommation totale [mW]		
		N_I	N_C	e_u	N_I	N_C	e_u
Poids caché (4.17)	3.70	40	40	0	148.0	148.0	0
Poids de sortie (4.16)	3.64	8	8	0	29.12	29.12	0
Neurone caché (4.14)	4.45	8	8	0	35.60	35.60	0
Neurone de sortie (4.15)	1.76	1	1	0	1.76	1.76	0
Multiplieur (4.3)	0.055	1	1	5	0.055	0.055	0.27
Sommateur (4.5)	0.85	1	0	2	0.85	0	1.70
CCT (4.8)	1.76	1	1	3	1.76	1.76	5.28

5.6.2. Architecture complète

Le tableau 5.12 présente les consommations de puissance et les surfaces d'intégration totales des trois blocs principaux de la loi de commande obtenues avec le simulateur. Nous avons également ajouté les blocs de délais et le sommateur supplémentaires nécessaires conformément à la figure 4.21.

Tableau 5.12. Consommation de puissance et surfaces d'intégration totales des trois blocs principaux de la loi de commande

Bloc	Consommation totale [mW]	Surface totale [mm^2]
N_I	217.14	4.59
N_C	216.29	4.57
calcul e_u	7.25	0.16
8 délais	6.88	0.15
Sommateur	0.85	0.04
Total	448.41	9.51

5.7. Conclusion

Dans ce chapitre, l'objectif était de valider, au niveau de la simulation, la fonctionnalité des circuits CMOS présentés dans le quatrième chapitre pour la modélisation au niveau de l'ITGE de la loi de commande développée dans la thèse.

Les résultats obtenus (*cf.* sections 5.4.1 et 5.4.2) ont permis de valider, au niveau de la simulation des dessins de masques, la fonctionnalité du circuit global pour la commande de deux systèmes. Le cas du système synthétique de la section 5.4.1 est très intéressant puisque, rappelons le, ce système a une constante de temps de l'ordre de 0.5ms, démontrant la fonctionnalité du circuit pour la commande d'une gamme de systèmes rapides. D'un autre côté, le cas du moteur à courant continu avec frottements non linéaires représente un cas concret d'application à un modèle réel très répandu dans l'industrie.

Au niveau de l'étude de sensibilité du circuit, nous avons mené les simulations des différents coins de la technologie pour différentes valeurs de la température. Les résultats de simulations obtenus ont montré une bonne robustesse de notre circuit sur une plage de températures allant approximativement de 10 à 50°C. Toutefois, il est difficile d'évaluer à quel point la robustesse provient du circuit lui-même ou de l'algorithme de commande. Nous avons mentionné à plusieurs reprises la capacité des RNA de compenser certains défauts des opérations arithmétiques de base mises en œuvre. Nous avons en effet constaté, en analysant les résultats de simulations de certains coins à certaines températures, que l'influence des variations du procédé et de la température devenait nettement visible sur certaines opérations arithmétiques de base (*e.g.* la fonction tangente hyperbolique et sa dérivée). Cette influence s'avère pourtant, jusqu'à un certain point, compensée par l'adaptation des paramètres des réseaux. Ceci reste toutefois très difficile à quantifier.

L'évaluation du circuit global a abouti à une consommation de puissance totale de l'ordre de 448mW et une surface d'intégration d'environ 9.5mm^2. Il est important de noter que ces chiffres sont les résultats estimés obtenus avec le simulateur.

Chapitre 6

Conclusion

6.1. Bilan des objectifs atteints et des contributions scientifiques

6.1.1. Objectifs atteints

L'objectif général de la thèse était de proposer une solution pratique et efficace pour la commande en temps réel de systèmes dynamiques non linéaires, solution satisfaisant aux contraintes des applications modernes en commande mentionnées dans la section 1.1 du premier chapitre. Nous rappelons les sous-objectifs de recherche énoncés dans section 1.3 du premier chapitre :

O1) proposition de nouveaux algorithmes adaptatifs à base de RNA pour la commande de systèmes dynamiques non linéaires,

O2) validation des algorithmes proposés par des résultats de simulations mettant en évidence la fonctionnalité des algorithmes proposés ainsi que leur apport en comparaison aux algorithmes de référence de la littérature,

O3) proposition d'une architecture hautement parallèle en vue d'une intégration sur silicium des algorithmes proposés,

O4) modélisation de l'architecture proposée dans des circuits d'ITGE,

O5) simulation des circuits proposés au niveau des dessins de masques sur silicium et évaluation des performances, incluant des simulations de la sensibilité des circuits aux défauts de fabrication,

O6) évaluation des performances du circuit au niveau de la simulation,

O7) énoncé de suggestions pour des travaux futurs basés sur la présente thèse.

L'atteinte de ces objectifs de recherche peut être résumée comme suit :

O1) deux nouveaux algorithmes RPD1 et RPD2 ont été proposés dans une loi de commande adaptative indirecte inverse (*cf.* sections 3.3.3.1 et 3.3.3.2, troisième chapitre),

O2) les algorithmes proposés ont été validés par des résultats de simulations démontrant d'une part que les algorithmes proposés sont fonctionnels et d'autre part que l'algorithme RPD1 est le meilleur compromis entre les critères algorithmiques et d'ITGE comparativement à RPD2 et aux algorithmes de référence de la littérature, c'est-à-dire RPS et RTRL (*cf.* section 3.4, troisième chapitre),

O3) une architecture hautement parallèle pour l'implémentation de la loi de commande complète incluant l'algorithme RPD1 a été proposée (*cf.* sections 4.4 et 4.5, quatrième chapitre),

O4) l'architecture proposée a été modélisée au niveau de l'ITGE par des circuits CMOS analogiques (*cf.* sections 4.4 et 4.5, quatrième chapitre),

O5) les dessins de masques pour la réalisation sur silicium de l'architecture ont été modélisés. Les résultats de simulations au niveau des dessins de masques de la loi de commande complète ont permis d'une part de vérifier la fonctionnalité de la loi de commande à ce niveau et d'autre part d'évaluer le circuit global modélisé en termes de surface d'intégration et de consommation de puissance (*cf.* sections 5.3 et 5.4, cinquième chapitre). Les résultats de simulations aux coins pour plusieurs valeurs de la température ont également été présentés afin de vérifier, au niveau de la simulation, la sensibilité du circuit au procédé de fabrication et à la température (*cf.* sections 5.5, cinquième chapitre),

O6) les performances du circuit ont été évaluées dans les sections 5.4, 5.5 et 5.6 du cinquième chapitre,

O7) un certain nombres de suggestions pour des travaux futurs reliés à la présente thèse sont présentées dans la section 6.2 de ce chapitre.

6.1.2. Contributions scientifiques

Les contributions scientifiques principales apportées par la thèse sont :

C1) deux nouveaux algorithmes adaptatifs RPD1 et RPD2 pour la rétro propagation à travers le modèle neuronal dans une structure de commande adaptative indirecte inverse. Ces nouveaux algorithmes permettent, par rapport aux algorithmes de référence de la littérature, un gain tant du point de vue des critères algorithmiques que du point de vue des critères d'intégration sur silicium,

C2) un circuit différentiel complètement analogique avec adaptation continue en ligne pour la modélisation d'un poids synaptique,

C3) un circuit différentiel complètement analogique pour la modélisation d'un neurone non linéaire avec fonction tangente hyperbolique incluant le calcul de la dérivée de la fonction d'activation,

C4) un circuit différentiel complètement analogique pour la modélisation de l'algorithme RPD1,

C5) un circuit différentiel complètement analogique à temps continu pour la modélisation au niveau de l'ITGE de la loi de commande développée dans la thèse.

6.1.3. Applications des résultats obtenus

D'un point de vue général, le travail de recherche présenté dans cette thèse vise des applications dans le domaine de la commande de systèmes dynamiques non linéaires, applications telles que celles citées dans la section 1.3 du premier chapitre. Comme nous l'avons mentionné à plusieurs reprises dans la thèse, le développement d'ASICs pour ce genre d'application ne peut être justifié que par les contraintes de l'application elle-même. Les circuits intégrés dédiés tirent leurs avantages de l'optimisation qui peut être faîte sur le circuit en vue de l'intégration d'un algorithme particulier et d'objectifs particuliers liés à l'application.

6.2. Suggestions de travaux futurs

6.2.1. Au niveau algorithmique

Un point important dans le développement d'une loi de commande est l'étude de stabilité. Dans les applications pratiques, il est toujours préférable d'utiliser une loi de commande dont la stabilité peut être démontrée au moins dans les conditions particulières de son utilisation pour une application donnée. L'établissement d'une preuve de stabilité pour une loi de commande telle que développée dans nos travaux est une tâche très difficile pour plusieurs raisons, notamment :

- parce que ceci est généralement le cas avec des systèmes non linéaires et que pour de tels systèmes les preuves de stabilité ne sont établies que pour un système particulier ou au mieux pour une classe très restreinte de systèmes,
- parce que la loi de commande comprend des rétroactions,
- parce que la loi de commande comprend, outre le système commandé lui-même, des éléments dynamiques.

La solution à ce problème pourrait être trouvée en développant, en laboratoire, une loi de commande stable basée sur l'algorithme RPD1 pour un système réel. Les expérimentations sur le système réel pourraient permettre de réduire la complexité du problème de stabilité pour ce cas particulier.

6.2.2. Au niveau de l'ITGE

L'évaluation de la surface d'intégration totale et de la consommation de puissance totale du circuit présentée dans le cinquième chapitre nous a permis de constater qu'une grande partie est due à l'AOT. Bien sûr, nous savions d'avance qu'étant le circuit le plus utilisé, ce serait probablement le cas. Cependant, nous estimons que le circuit de l'AOT est loin d'être optimal et qu'il y a une grande amélioration possible tant du point de vue de la surface d'intégration que de la consommation de puissance de ce circuit de base. Une amélioration significative de ce circuit aura un impact important sur le circuit global de la loi de commande. Ceci pourrait être obtenu en optimisant l'AOT avec son architecture présentée dans la section 4.4 ou encore en proposant un nouveau circuit d'AOT.

Enfin, la fabrication d'un prototype de circuit et son expérimentation en laboratoire à la commande d'un système réel seront deux étapes supplémentaires importantes vers la mise en pratique du circuit.

6.3. Avenir des ASIC analogiques pour l'ITGE des RNA

Bien que l'ITGE des RNA dans des circuits dédiés analogiques soit un défi sur lequel de plus en plus de chercheurs commencent à se pencher, incluant la présente thèse, un investissement de recherche important est encore nécessaire avant que ce domaine ne parvienne à maturité. Nous avons énoncé tout au long de la thèse que la recherche est nécessaire tant du point de vue algorithmique que du point de vue de l'ITGE afin de satisfaire les applications modernes. Nous pensons que deux axes de recherche sont importants dans ce domaine. D'abord le développement d'algorithmes adaptatifs en vue d'une ITGE comme ceci a été fait dans nos travaux, ensuite dans des phases d'expérimentations intensives permettant de caractériser les algorithmes implantés dans des circuits analogiques au niveau physique.

Les résultats présentés dans le présent document ainsi que ceux obtenus par d'autres équipes de recherche démontrent un réel potentiel pour l'intégration de RNA dans des ASIC analogiques et la pertinence de travaux de recherche dans ce domaine. Le développement d'ASIC analogiques pour des applications de commande est un domaine récent qui ne bénéficie pas encore de toute l'expérience acquise par exemple dans le domaine numérique. Ce domaine nécessite encore à notre avis de nombreux travaux sur les plans théorique et pratique avant d'envisager une mise en application pratique courante de ce type de circuit.

Bibliographie

[Abbas & Triolo '97] J. J. Abbas, R. J. Triolo, « Experimental Evaluation of an Adaptive Feedforward Controller for Use in Functional Neuromuscular Stimulation Systems », *IEEE Transactions on Rehabilitation Engineering*, Vol. 5, No. 1, pp. 12-22, 1997.

[Armstrong & Canudas de Wit '96] B. Armstrong, C. Canudas de Wit, « Friction modeling and compensation » dans *The Control Handbook*, Chap. 77, W.S. Levine, editor, *IEEE Press*, 1996.

[Åström & Wittenmark '95] Åström, K. J., Wittenmark B. « Adaptive control – 2nd edition », *Addison Wesley series in electrical engineering, Control engineering*, 1995.

[Baker *et al.* '98] R. J. Baker, H. W. Li, D. E. Boyce, « CMOS Circuit Design and Simulation », *IEEE Press*, 1998.

[Baldi '95] P. Baldi, « Gradient Descent Learning Algorithm Overview: A General Dynamical Systems Perspective », *IEEE Trans. on Neural Networks*, Vol. 6, No. 1, pp. 182-95, 1995.

[Ba-Razzouk *et al.* '97] A. Ba-Razzouk, A. Cheriti, G. Olivier, P. Sicard, « Field-Oriented Control of Induction Motors Using Neural Network Decouplers », *IEEE Trans. on Power Electronics*, Vol. 12, No. 4, pp. 752-63, juillet 1997.

[Baruch *et al.* '02] I. S. Baruch, J. M. Flores, F. Nava, I. R. Ramirez, B. Nenkova, « An Advanced Neural Network Topology and Learning, Applied for Identification and Control of a D.C. Motor », *Proc. of the First Int. IEEE Symposium on Intelligent Systems*, pp. 289-95, septembre 2002.

[Behera *et al.* '97] L. Behera, S. Chaudhury, M. Gopal, « Neuro-adaptive hybrid controller for robot-manipulator tracking control », *IEE Proc. on Control Theory and Applications*, Vol. 143, No. 3, pp. 270-5, 1997.

[Behera *et al.* '96] L. Behera, M. Gopal, S. Chaudhury, « On Adaptive Trajectory Tracking of a Robot Manipulator Using Inversion of its Neural Emulator », *IEEE Trans. On Neural Networks*, Vol. 7, No. 6, pp. 1401-14, novembre 1996.

[Bhattacharyya & Dey '99] A.B. Bhattacharyya, S. Dey « Sub-circuit analysis for power supply rejection ratio in regulated cascode operational transconductance amplifiers and filters », *Proc of the twelfth International Conference on VLSI Design*, pp. 164-8, 1999.

[Bhavnagarwata *et al.* '98] A. J. Bhavnagarwata, B. Austin, J. D. Meindl, « Minimum supply voltage for bulk Si CMOS GSI », *Proc of the 1998 International Symposium on Low Power Electronics and Design*, pp. 100-2, 1998.

[Bo & Frenzel '02] L. Bo, J. F. Frenzel, « A CMOS neuron for VLSI circuit implementation of pulsed neural networks », *Proc. of the IEEE Annual Conference of the Industrial Electronics Society*, Vol. 4, pp. 3182-5, 2002.

[Bo *et al.* '00] G. M. Bo, D. D. Caviglia, M. Valle, « An on-chip learning neural network », *Proc. of the IEEE INNS-ENNS Int. Joint Conference on Neural Networks*, Vol. 4, pp. 66-71, 2000.

[Bo *et al.* '99] G. M. Bo, D. D. Caviglia, H. Chiblè, M. Valle, « An Experimental Analog CMOS Self-Learning Chip », *Proc. of the 7th Int. Conf. On Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, MicroNeuro'99, pp. 135-39, 1999.

[Burton *et al.* '97] B. Burton, F. Kamran, R. G. Harley, T. G. Habetler, T. G. Brooke, M. A. Poddar, « Identification and Control of Induction Motor Stator Currents Using Fast On-Line Random Training of a Neural Network », *IEEE Trans. on Industry Applications*, Vol. 33, No. 3, mai/juin 1997.

[Cabrera & Narendra '99] J. B. D. Cabrera, K. S. Narendra, « Issues in the Application of Neural Networks for Tracking Based on Inverse Control », *IEEE Trans. on Automatic Control*, Vol. 44, No. 11, pp. 2007-27, novembre 1999.

[Cadence '99] « Layout Design and Simulation Using Virtuoso/Diva/Analog Artist », *Cadence Tutorials*, Royal Military College of Canada, juillet 1999.

[Castro *et al.* '93] H. A. Castro, S. M. Tam, M. A. Holler, « Implementation and Performance on an Analog Nonvolatile Neural Network », *Analog Integrated Circuits and Signal Processing*, Vol. 4, pp. 97-113, 1993.

[Cauwenberghs '96] G. Cauwenberghs, « An Analog VLSI Recurrent Neural Network Learning a Continuous Time Trajectory », *IEEE Trans. on Neural Networks*, Vol. 7, No. 2, pp. 346-61, mars 1996.

[Cetinkunt & Book '90] S. Cetinkunt, W. J. Book, « Performance Limitations of Joint Variable-Feedback Controllers Due to Manipulator Structural Flexibility », *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 2, 219-31, avril 1990.

[Chen et al. '00] D. Chen, R. R. Mohler, L. Chen, « Synthesis of Neural Controller Applied to Flexible AC Transmission Systems », *IEEE Trans. on Circuits and Systems – Part I: Fundamental Theory and Applications*, Vol. 47, No. 3, pp. 376-88, mars 2000.

[Cichocki & Unbehauen '93] A. Cichocki, R. Unbehauen, « Neural Networks for Optimization and Signal Processing », *John Wiley & Sons Ltd. & B. G. Teubner*, Stuttgart, 1993.

[Coué & Wilson '96] D. Coué, G. Wilson « A Four-Quadrant Subthreshold Mode Multiplier for Analog Neural-Network Applications », *IEEE Trans. on Neural Networks*, Vol. 7, No. 5, septembre 1996.

[Dash *et al.* '97] P. K. Dash, S. K. Panda, T. H. Lee, J. X. Xu, « Fuzzy and Neural Controllers for Dynamic Systems : an Overview », *Proc. of* the *Int. Conf. on Power Electronics and Drive Systems*, Vol. 2, pp. 632-36, 1997.

[Diotaveli *et al.* '00] F. Diotaveli, M. Valle, G. M. Bo, D. D. Caviglia, « A VLSI architecture for weight perturbation on-chip learning », *Proc. of the IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks*, Vol. 4, pp. 219-24, 2000.

[Eberhardt *et al.* '88] S. Eberhardt, A. Moonpenn, A. Thakoor, « Considerations for Hardware Implementations of Neural Networks », *Proc. of the 22nd Asilomar Conference on Signals, Systems and Computers*, pp. 649-53, 1988.

[Elias *et al.* '97] A. Elias, L. Leija, C. Alvarado, P. Hernandez, A. Gutierrez, « Personal Computer System for ECG Recognition in Myocardial Infarction Diagnosis based on an Artificial Neural Network », *Proc. of the 19th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, Vol. 3, pp. 1095-96, 1997.

[Elman '90] J. L. Elman, « Finding structure in time », *Cognitive Science 14*, pp. 179-211, 1990.

[El-Masry *et al.* '97] E. I. El-Masry, H-K. Yang, M. A, Yakout, « Implementation of Artificial Neural Networks Using Current-Mode Pulse Width Modulation Technique », *IEEE Trans. on Neural Networks*, Vol. 8, No. 3, pp. 532-47, mai 1997.

[Fernandez-Redondo & Hernandez-Espinosa '00] M. Fernandez-Redondo, C. Hernandez-Espinosa, « On the combination of weight-decay and input selection methods », *Proc. of the IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks*, Vol. 1, pp. 191-96, 2000.

[Freeman & Kokotovic '96] R. A. Freeman, P. V. Kokotovic, « Robust Nonlinear Control Design State-Space and Lyapunov Techniques », Birkhauser Boston, 1996.

[Fukuda & Shibata '92] T. Fukuda, T. Shibata, « Theory and Applications of Neural Networks for Industrial Control Systems », *IEEE Trans. on Industrial Electronics*, Vol. 39, No. 6, décembre 1992.

[Funahashi '89] K. Funahashi, « On the approximate realization of continuous mappings by neural networks », *Neural Networks*, Vol. 2, pp. 183-92, 1989.

[Ge *et al.* '97] S. S. Ge, C. C. Hang, L. C. Woon, « Adaptive Neural Network Control of Robot Manipulators in Task Space », *IEEE Trans. on Industrial Electronics*, Vol. 44, No. 6, pp. 746-52, décembre 1997.

[Ghosh *et al.* '94] J. Ghosh, P. Lacour, S. Jackson, « OTA-Based Neural Network Architectures with On-Chip Tuning of Synapses », *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 41, No.1, pp. 49-57, janvier 1994.

[Glad & Ljung '00] T. Glad, L. Ljung, « Control theory: multivariable and nonlinear methods », *Taylor & Françis*, Londres, 2000.

[Gutierrez *et al.* '98] L. B. Gutierrez, F. L. Lewis, J. A. Lowe, « Implementation of a Neural Network Tracking Controller for a Single Flexible Link: Comparison with PD and PID Controllers », *IEEE Trans. on Industrial Electronics*, Vol. 45, No.2, pp. 307-18, avril 1998.

[Guyennet '01] H. Guyennet, « Programmation et systèmes parallèles : tendances actuelles », *Collection Techniques et sciences informatiques*, édition Hermès, Paris, 2001.

[Hagan & Menhaj '94] M. T. Hagan, M. B. Menhaj, « Training Feedforward Networks with the Marquardt Algorithm », *IEEE Trans. on Neural Networks*, Vol. 5, No. 6, novembre 1994.

[Han & Sanchez-Sinencio '98] G. Han, E. Sanchez-Sinencio, « CMOS Transconductance Multipliers : A Tutorial », *IEEE Trans. on Circuits and Systems II : Analog & Digital Signal Processing*, Vol. 45, No. 12, pp. 1550-63, décembre 1998.

[Hastings '00] A. Hastings, « The art of analog layout », Prentice Hall, 2000.

[Haykin '99] S. Haykin, « Neural Networks : a Comprehensive Foundation – 2nd edition», Prentice Hall, 1999.

[Henessy & Patterson '92] J. L. Henessy, D. A. Patterson, « Architectures des ordinateurs : une approche quantitative », McGraw Hill, 1992.

[Hertz *et al.* '91] J. Hertz, A. Krogh, R. G. Palmer, « Introduction to the Theory of Neural Computation », *Addison-Wesley Publishing Company*, 1991.

[Hischorn '79] R. M. Hischorn, « Invertibility of nonlinear control systems », *SIAM Journal Control and Optimization*, Vol. 17, No. 2, pp. 289-97, 1979.

[**Hollis & Paulos '94**] P. W. Hollis, J. J. Paulos, « A Neural Network Algorithm Tailored for VLSI Implementation », *IEEE Trans. on Neural Networks*, Vol. 5, No. 5, septembre 1994.

[Hollis & Paulos '90] P. W. Hollis, J. J. Paulos, « The *Effects of Precision Constraints in a Back-propagation Learning Network », Neural* Computation, Vol. 2, No. 3, pp. 363-73, 1990.

[Hornik & Stinchcombe '89] K. Hornik, M. Stinchcombe, « Multilayered feedforward neural networks are universal approximators », *Neural Neworks*, Vol. 2, pp. 359-66, 1989.

[Hubel '88] D. Hubel, « Eye, Brain and Vision », New-York : *Scientific American Library*, 1988.

[Jabri & Flower '92] M. Jabri, M. Flower, « Weight Perturbation : An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks », *IEEE Trans. on Neural Networks*, Vol. 3, No. 1, pp. 154-7, 1992.

[Jain & Vemuri '99] L. C. Jain, V. R. Vemuri, « Industrial Applications of Neural Networks », *CRC Press, International Series on Computational Intelligence*, 1999.

[Jin & Gupta '99] L. Jin, M. M. Gupta, « Stable Dynamic Backpropagation Learning in Recurrent Neural Networks », *IEEE Trans. on Neural Networks*, Vol. 10, No. 6, pp. 1321-33, novembre 1999.

[Johns & Martins '97] D. A. Johns, K. Martin, « Analog Integrated Circuit Design », *John Wiley & Sons. Inc.*, 1997.

[John & Matthew '98] W. John, A. F. Matthew, « NARMAX Modeling and Robust Controller Design of Internal Combustion Engines », *Proc. of the American Control Conference*, pp. 957-61, Philadelphie, PEN, juin 1998.

[Karr '99] C. L. Karr, « Practical Applications of Computational Intelligence for Adaptive Control », CRC Press, 1999.

[Kim & Lewis '00] Y. H. Kim, F. L. Lewis, « Reinforcement Adaptive Learning Neural-Net Based Friction Compensation Control for High Speed and Precision », *IEEE Trans. on Control Systems Technology*, Vol. 8, No. 1, pp. 118-26, janvier 2000.

[Kung & Hwang '89] S. Y. Kung, J. N. Hwang, « Neural network architectures for robotic applications », *IEEE Trans. on Robotics and Automation*, Vol. 5, No. 5, p. 641-657, 1989.

[Lapedes & Farber '97] A. Lapedes, R. Farber, « How Neural Nets Work », *Proc. of the Neural Information Processing Systems Conference*, New-York, pp. 442-56, 1997.

[Lee *et al.* '99] K-J. Lee, W-C. Wang, K-S. Huang, « Current-Mode Testable Design of Operational Transconductance Amplifier-Capacitor Filters », *IEEE Trans. on Circuits and Systems-II :Analog and Digital Signal Processing*, Vol. 46, No. 4, pp. 401-13, avril 1999.

[Lehmann '94] T. Lehmann, « Hardware Learning in Analogue VLSI Neural Networks », Ph.D. Thesis, Technical University of Denmark, 1994.

[Lesueur *et al.* '03a] S. Lesueur, D. Massicotte et P. Sicard, « Adaptive Control of Nonlinear Dynamic Systems Based on Dynamic Backpropagation Neural Networks », soumis à *EURASIP Journal on Applied Signal Processing*, 2003.

[Lesueur *et al.* '03b] S. Lesueur, Louis Lemire et D. Massicotte, « A Full-Differential Analog Neural Network With On-Line Backpropagation Learning », *Proc. Of the Northeast Workshop on Circuits and Systems*, pp. 81-4, Montréal, Canada, juin 2003.

[Lesueur *et al.* '02] S. Lesueur, Louis Lemire et D. Massicotte, « Réseau de neurones analogique intégré complètement différentiel avec adaptation en ligne par rétropropagation », Congrès Canadien en Génie Électrique et Informatique (CCECE), Toronto, Ont., Canada, mai 2002.

[Lesueur *et al.* '01] S. Lesueur, D. Massicotte, P. Sicard, « Indirect Inverse Adaptive Control of Nonlinear Dynamic Systems Using Neural Networks and Dynamic Back Propagation », Int. Symposium on Circuits and Systems (ISCAS'01), Sydney, Australie, mai 2001.

[Lesueur '99] S. Lesueur, « Implantation en technologie ITGE (VLSI) d'une loi de commande pour une articulation flexible », mémoire de maîtrise (M.Sc.A), Université du Québec à Trois-Rivières, Département de Génie Électrique, 1999.

[Levin & Narendra '96] A. U. Levin, K. S. Narendra, « Control of Nonlinear Dynamical Systems Using Neural Networks – Part II : Observability, Identification, and Control », *IEEE Trans. on Neural Networks*, Vol. 7, No. 1, pp. 30-42, janvier 1996.

[Liu & Asada '96] S. Liu, H. Asada, « Teaching and Learning of Deburring Robots Using Neural Networks », Neural Networks Applications, *IEEE Technology Update Series*, 1996.

[Lyiong *et al.* '95] C. Lyiong, J. C. Balda, K. J. Olejniczak, « Implementation of a High-Performance Induction Motor Drive Using a Three-Phase Inverter Module, a PWM ASIC and a DSP », *Proc. of the Applied Power Electronics Conference and Exposition*, Vol. 1, No. 1, pp. 217-23, 1995.

[Man *et al.* '99] Z. Man, S. K. Phooi, H. R. Wu, « Lyapunov stability-based adaptive backpropagation for discrete time system », *Proc. of the Fifth Int. Symp. On Processing and its Applications*, Vol. 2, pp. 661-64, 1999.

[Massicotte '98] D. Massicotte, « A Systolic VLSI Implementation of Kalman-filter-based Algorithms for Signal Reconstruction », *IEEE Int. Conf. on Acoustics Speech and Signal Processing*, Vol. 8, pp. 3029-032, Seattle, USA, Mai 1998.

[Massicotte '95] D. Massicotte, « Une approche à l'implantation en technologie VLSI d'une classe d'algorithmes de reconstitution des signaux », Thèse de doctorat (Ph.D.) - École Polytechnique de Montréal, 1995.

[McCulloch & Pitts '43] W. S. McCulloch, W. Pitts, « A logical calculus of the ideas immanent in nervous activity », *Bulletin of Mathematical Biophysics 5*, pp. 115-33, 1943.

[Mendel & McLaren '70] J. M. Mendel, R. W. McLaren, « Reinforcement-learning control and pattern recognition systems », *Adaptive, Learning, and Pattern Recognition Systems : Theory and Applications*, Vol. 66, pp. 287-318, New-York, 1970.

[Minghu *et al.* '00] J. Minghu, Z. Xiaoyan, T. Xiaofang, L. Biqin, R. Qiuqi, J. Mingyan, « A fast hybrid algorithm of global optimization for feedforward neural networks », *Proc. of the 5th Int. Conf. on Signal Processing*, Vol. 3, pp. 1609-12, 2000.

[Minsky & Papert '69] M. Minsky, S. Papert, « Perceptrons : An Introduction to Computational Geometry », *MIT Press*, 1969.

[Mistry *et al.* '96] S. I. Mistry, S. Chang, S. Nair, « Indirect Control of a Class of Nonlinear Dynamic Systems », *IEEE Trans. on Neural Networks*, Vol. 7, No. 4, pp. 1015-23, juillet 1996.

[Montalvo *et al.* '97] A. J. Montalvo, R. S. Gyurcsik, J. J. Paulos, « Toward a General-Purpose Analog VLSI Neural Network with On-Chip Learning », *IEEE Trans. on Neural Networks*, Vol. 8, No. 2, pp. 413-23, mars 1997.

[Montalvo *et al.* '92] A. J. Montalvo, R. S. Gyurcsik, J. J. Paulos, « On-Chip Learning in the Analog Domain with Limited Precision », *Proc. International Symposium on Circuits and Systems*, Vol. 1, pp. 196-201, 1992.

[Morie & Amemiya '94] T. Morie, Y. Amemiya, « An All-Analog Expandable Neural Network LSI with On-Chip Backpropagation Learning », *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 9, pp. 1086-93, septembre 1994.

[Nabney *et al.* '01] I. T. Nabney, D. J. Evans, J. Tenner, L. Gamlyn, « Benchmarking Beat Classification Algorithms », Computers in Cardiology Conference, pp. 529-32, 2001.

[Narendra '97] K. S. Narendra, « Neural networks for real-time control », *Proc. of the 36th conf. On Decision and Control*, pp. 1026-31, 1997.

[Narendra & Parthasarathy '91] K. S. Narendra, K. Parthasarathy, « Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks », *IEEE Trans. on Neural Networks*,Vol. 2, No. 2, pp. 252-62, 1991.

[**Narendra & Parthasarathy '90**] K. S. Narendra, K. Parthasarathy, « Identification and Control of Dynamical Systems Using Neural Networks », *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, pp. 4-26, 1990.

[Nijhuis *et al.* '91] J. Nijhuis, B. Hofflinger, S. Neuber, A. Siggelkow, L. Spaanenburg, « A VLSI Implementation of a Neural Car Collision Avoidance Controller », *Proc. of the Int. Joint Conf. on Neural Network*, IJCNN'91, Vol. 1, pp. 493-9, 1991.

[O'Leary '91] P. O'Leary, « Practical Aspects of Mixed Analogue and Digital Design », in *Analogue Digital ASIC's Circuit Techniques, Design Tools ans Applications*, R. S. Soin, F. Maloberti, J. Franca, Eds., *IEE Circuits and Systems (3) Series*, pp. 213-38, London, 1991.

[Park *et al.* '96] Y-M Park, M-S. Choi, K-L. Lee, « An Optimal Tracking Neuro-Controller for Nonlinear Dynamic Systems », *IEEE Trans. on Neural Networks*, Vol. 7, No. 5, pp. 1099-110, septembre 1996.

[Piché '94] S. W. Piché, « Steepest Descent Algorithms for Neural Networks Controllers and Filters », *IEEE Trans. on Neural Networks*, Vol. 5, No. 2, pp. 198-212, 1994.

[Pineda de Gyvez & Pradhan '99] J. Pineda de Gyvez, D. K. Pradhan, « Integrated Circuit Manufacturability – The Art of Process and Design Integration », *IEEE Press*, 1999.

[Rao & Gupta '93] D. H. Rao, M. M. Gupta, « Dynamic Neural Units and Function Approximation », *IEEE Conf. on Neural Networks*, pp. 743-8, San Francisco, avril 1993.

[Rovithakis & Christodoulou '95] G. A. Rovithakis, M. A. Christodoulou, « Direct Adaptive Regulation of Unknown Nonlinear Dynamical Systems via Dynamic Neural Networks », *IEEE Trans. on System, Mans and Cybernetics*, Vol. 25, No. 12, décembre 1995.

[Roy & Prasad '89] K. Roy, S. C. Prasad, « Low-Power CMOS VLSI Circuit Design », John Wiley & Sons, 1989.

[Rumelhart *et al.* '86] D. E. Rumelhart, G. E. Hinton, R. J. Williams, « Learning Internal Representations by Error Propagation », Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Vol. 1, *MIT Press*, 1986.

[Sanchez-Sinencio & Silva-Martinez '00] E. Sanchez-Sinencio, J. Silva-Martinez, « CMOS transconductance amplifiers, architectures and active filters : a tutorial », *IEE Proc. Circuits Devices Systems*, No. 1, février 2000.

[Sanchez-Sinencio & Andreou '99] E. Sanchez-Sinencio, G. A. Andreou, « Low-Voltage / Low-Power Integrated Circuits and Systems », *IEEE Press Series on Microelectronic Systems*, 1999.

[Sandberg '80] I. W. Sandberg, « Global inverse function theorems », *IEEE Trans. on Circuits and Systems*, Vol. 27, No. 11, 1980.

[Sastry *et al.* '94] P. S. Sastry, G. S. Santharam, K. P. Unnikrishnan, « Memory Neuron Networks for Identification and Control of Dynamical Systems », *IEEE Trans. on Neural Networks*, Vol. 5, No. 2, pp. 306-19, mars 1994.

[Seidl et al. '95] D.R. Seidl, S.L. Lam, J.A. Putman, R.D. Lorenz, « Neural network compensation of gear backlash hysteresis in position-controlled mechanisms », *IEEE Trans. on Industry Applications*, Vol. 31, No. 6, pp. 1475-83, novembre 1995.

[Sepulchre *et al.* '97] R. Sepulchre, M. Jankovic, P. Kokotovic, « Constructive Nonlinear Control », Springer-Verlag London Limited, 1997.

[Spencer & Sanchez-Sinencio '99] R.G. Spencer, E. Sanchez-Sinencio « A fully-differential CMOS implementation of Oja's learning rule in a dual-synapse neuron for extracting principal components for face recognition », *Proc. of the 42nd Symposium on Circuits and Systems*, Vol. 2, pp. 1102-04, 1999.

[Srinivasan *et al.* '94] B. Srinivasan, U. R. Prasad, N. J. Rao, « Back Propagation Through Adjoints for the Identification of Nonlinear Dynamic Systems Using Recurrent Neural Models », *IEEE Trans. on Neural Networks*, Vol. 5, No. 2, pp. 213-27, 1994.

[Suvkens & Vandewalle '99] J. A. K. Suvkens, J. Vandewalle, « Continuous time NLq theory : absolute stability criteria », *Proc. of the Int. Joint Conf. on Neural Networks*, Vol. 3, pp. 1481-84, 1999.

[Sweet & Good '85] L. M. Sweet, M. C. Good, « Redefinition of the Robot Motion Control Problem », *IEEE Control Systems Magazine*, pp. 18-25, août 1985.

[Tarassenko *et al.* '93] L. Tarassenko, J. Tombs, G. Cairns, « On-Chip Learning with Analog VLSI Neural Networks », *Int. Journal of Neural Systems*, Vol. 4, No. 4, pp. 419-26, 1993.

[Taur *et al.* '97] Y. Taur, D. A. Buchanan, W. Chen, D. J. Frank, K. E. Ismail, S-H. Lo, G. A. Sai-Halasz, R. G. Viswanathan, H-J. C. Wann, S. J. Wind, H-S. Wong, « CMOS scaling into the nanometer regime », *Proc. IEEE*, Vol. 85, No. 4, pp. 486-504, avril 1997.

[Titus & Gopalan '02] A.H. Titus, A. Gopalan, « A differential summing amplifier for analog VLSI systems », *Proc. of the Int. Symposium on Circuits and Systems*, Vol. 4, pp. 57-60, 2002.

[Tomberg & Kashi '90] J. E. Tomberg, K. K. Kaski, « Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms », *IEEE Journal of Solid States Circuits*, Vol. 25, No. 5, pp. 1277-86, octobre 1990.

[de Trémiolles *et al.* '97] G. de Trémiolles, P. Tannhof, B. Plougonven, C. Demarigny, K. Madani, « Visual probe mark inspection, using hardware implementation of artificial neural networks, in VLSI production », *Proc. of the Int. Conf. on Artificial and Natural Neural Networks*, Espagne, juin 1997.

[Tuttle & Seering '96] T. D. Tuttle, W, P, Seering, « A Nonlinear Model of a Harmonic Drive Gear Transmission », *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 3, pp. 368-74, 1996.

[Vallverdu *et al.* '91] M. Vallverdu, M. J. Korenberg, P. Caminal, « Model Identification of the Neural Control of the Cardiovascular System using NARMAX Models », *Proc. of the Computers in Cardiology Conference*, pp. 585-88, 1991.

[Waibel *et al.* '89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang, « Phoneme Recognition Using Time-Delay Neural Networks », *IEEE Trans. on Acoustics Speech and Signal Processing*, Vol. 37, No. 3, pp. 329-39, mars 1989.

[Werbos '90] P. J. Werbos, « Backpropagation Through Time: What it does and how to do it », *Proc. IEEE*, Vol. 78, No. 10, octobre 1990.

[Williams & Zipser '89] R. J. Williams, D. Zipser, « A learning algorithm for continually running fully recurrent networks », *Neural Computation 1*, pp. 270-80, 1989.

[Wishart & Harley '93] M. T. Wishart et R. G. Harley, « Identification and control of an induction machine using artificial neural networks », *Conf. Rec. IEEE IAS Annual Meeting*, Toronto, Ont. Canada, pp. 867-74, octobre 1993.

[Withagen '94] H. Withagen, « Implementing Backpropagation with Analog Hardware? », *Proc. Int. Conf. on Neural Networks*, Orlando, 1994.

[Wong *et al.* '99] H-S. P. Wong, D. J. Frank, P. M. Salomon, C. H. J. Wann, J. J. Welser, « Nanoscale CMOS », *Proc. IEEE*, Vol. 87, No. 4, pp. 537-70, 1999.

[Yerramalla *et al.* '03] S. Yerramalla, E. Fuller, M. Mladenovski, B. Cukic, « Lyapunov Analysis of Neural Network Stability in an Adaptive Flight Control System », to appear in *Proc. of the 6th Annual Symposium on Self-Stabilizing Systems*, 2003.

[Yu & Gupta '92] C. Yu, M. M. Gupta, « A Dynamic Neural Network for Associative Memory », *IRIS and PRECARN Second Annual Conference*, Montréal, Canada, juin 1992.

Annexes

Annexe A : détails sur les rôles des différents étages de l'AOT

Le circuit complet de l'AOT présenté dans la figure 4.2 peut être décomposé en plusieurs étages :

- l'étage de polarisation d'entrée, c'est-à-dire la partie gauche du circuit constituée des transistors M_{R1}, M_{R2} et M_1 à M_9. Cet étage assure la polarisation en entrée de l'AOT. Ce circuit assure un courant de polarisation constant sur une large plage de tension de sortie de l'étage, ce type de circuit est communément appelé *wide swing,*
- l'étage d'entrée, c'est-à-dire la paire différentielle M_{21}-M_{22},
- l'étage cascode réalisé par M_{19} du côté positif et M_{20} du côté négatif et qui sert à isoler les étages d'entrée et de sortie,
- la polarisation de l'étage cascode réalisée par M_{15}-M_{17} du côté positif et M_{16}-M_{18} du côté négatif,
- l'étage de sortie réalisé par M_{13}-M_{14} du côté positif et M_{31}-M_{32} du côté négatif. Cet étage est un inverseur configuré en convertisseur courant-tension *via* un condensateur de rétroaction locale,
- la polarisation basse réalisée par les transistors M_{23} à M_{28} qui imposent le courant de la paire différentielle d'entrée et le courant de repos des nœuds de sortie des étages cascodes.

Les transistors M_{R1} et M_{R2} sont des transistors connectés en résistances comme décrit dans la section 4.4.4.

Annexe B : détails du circuit de CMFB

Le circuit de CMFB est présenté dans la figure B.1.

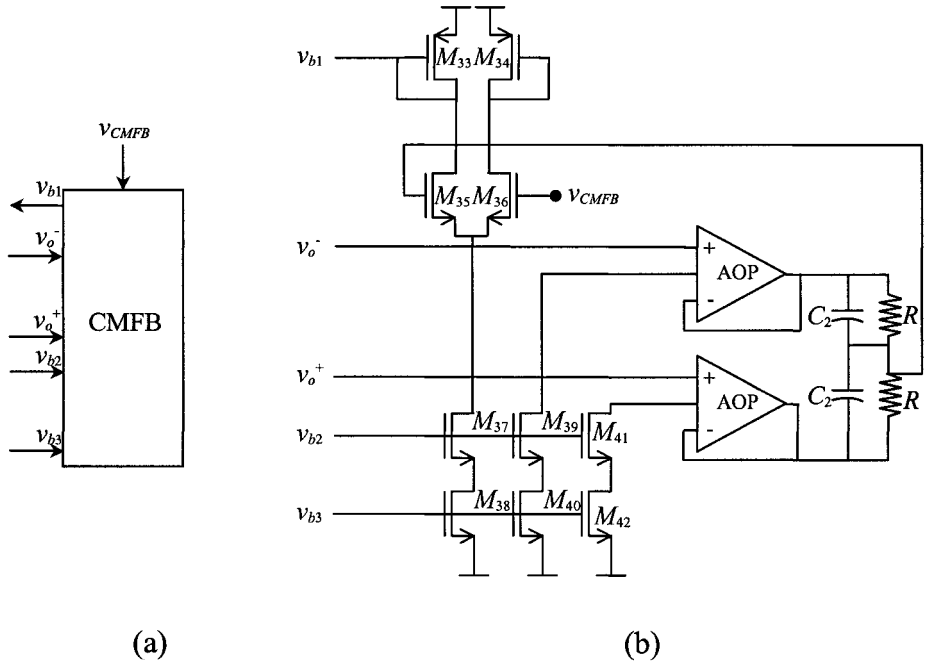


Figure B.1. Circuit d'asservissement en mode commun (CMFB) ; (a) symbole (b) schématique

Ce circuit fonctionne de manière opposée à un amplificateur différentiel, c'est-à-dire qu'il rejette la tension différentielle et est sensible aux tensions de mode commun. Les caractéristiques requises afin d'assurer la stabilité et la précision de la boucle d'asservissement réalisée par le CMFB sont :

- une réponse en fréquence du circuit d'asservissement supérieure à celle de l'amplificateur lui-même,
- un gain unitaire et une marge de phase de 180° sur la plage de fréquence désirée.

L'amplificateur opérationnel (AOP) est un amplificateur classique à deux étages de gain tel que décrit dans [Johns & Martins '97] et présenté dans la figure B.2. Le premier étage est constitué d'un amplificateur différentiel à sortie unique. Le deuxième étage est un amplificateur à source commune possédant une charge active. La capacité sert à assurer la stabilité de l'AOP lorsque celui-ci est connecté avec une boucle de rétroaction. Le transistor M_6 est utilisé comme résistance.

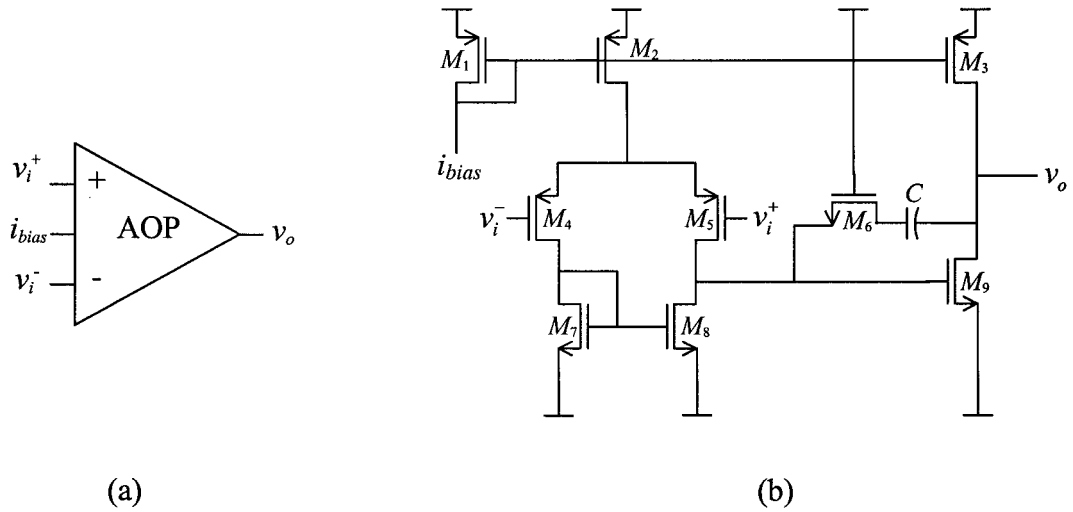


Figure B.2. L'AOP : (a) symbole ; (b) schématique

Annexe C : principe de multiplication permettant d'éliminer les termes non linéaires

Pour faciliter la compréhension du lecteur de la section 4.4.2, la caractéristique de courant de drain (I_D) en fonction de la tension drain source (V_{DS}) d'un transistor NMOS est représentée dans la figure C.1.

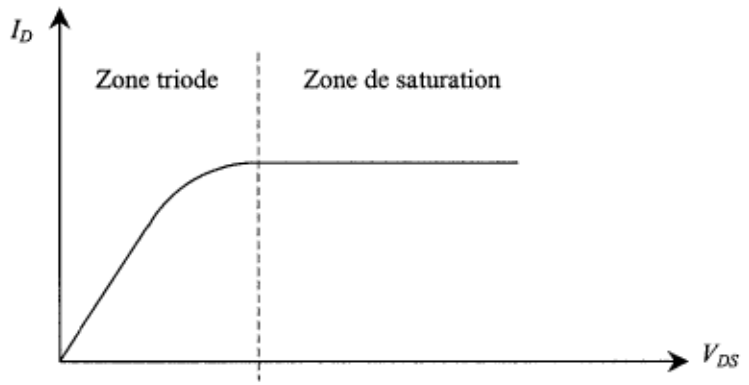


Figure C.1. Caractéristique générale courant/tension d'un transistor NMOS

Dans la caractéristique du transistor nous distinguons deux zones : (i) zone triode ; (ii) zone de saturation. Dans la zone triode, le courant de drain peut être exprimé par une équation non linéaire :

$$I_D = \mu_n C_{ox} \frac{W}{L}\left(V_{GS} - V_T - \frac{V_{DS}}{2}\right)V_{DS}$$

où μ_n est la mobilité des porteurs ; C_{ox} est la capacité d'oxyde de grille ; W est la largeur du transistor ; L est la longueur du transistor ; V_T est la tension de seuil de conduction et V_{GS} est la tension grille-source. Lorsque la tension V_{DS} est faible, la caractéristique du transistor devient quasi linéaire et peut être exprimée par :

$$I_D \approx \mu_n C_{ox} \frac{W}{L}\left(V_{GS} - V_T\right)V_{DS}$$

Enfin, dans la zone de saturation, le transistor atteint le maximum de courant qu'il est capable de conduire, courant de saturation $I_{D_{SAT}}$ d'une valeur constante exprimée par :

$$I_{D_{SAT}} = \mu_n C_{ox} \frac{W}{L}\left(V_{GS} - V_T\right)^2$$

Annexe D : dessins de masques des éléments électroniques de base

Transistors

Dans les circuits analogiques, les transistors ont généralement de plus grandes dimensions que dans les circuits numériques. Le dessin des masques d'un transistor de faible dimension est réalisé très simplement par l'intersection d'une couche de polysilicium et d'une couche active de matériau de type P ou N suivant le type du transistor (PMOS ou NMOS). Le dessin des masques de transistors de grandes dimensions ne peut être réalisé avec cette technique [Hastings '00]. Une technique largement utilisée consiste alors à considérer le transistor équivalent à une série de transistors de plus faibles dimensions et connectés en parallèle comme illustré par la figure F.1. Les notations D, G et S désignent respectivement le drain, la grille et la source des transistors.

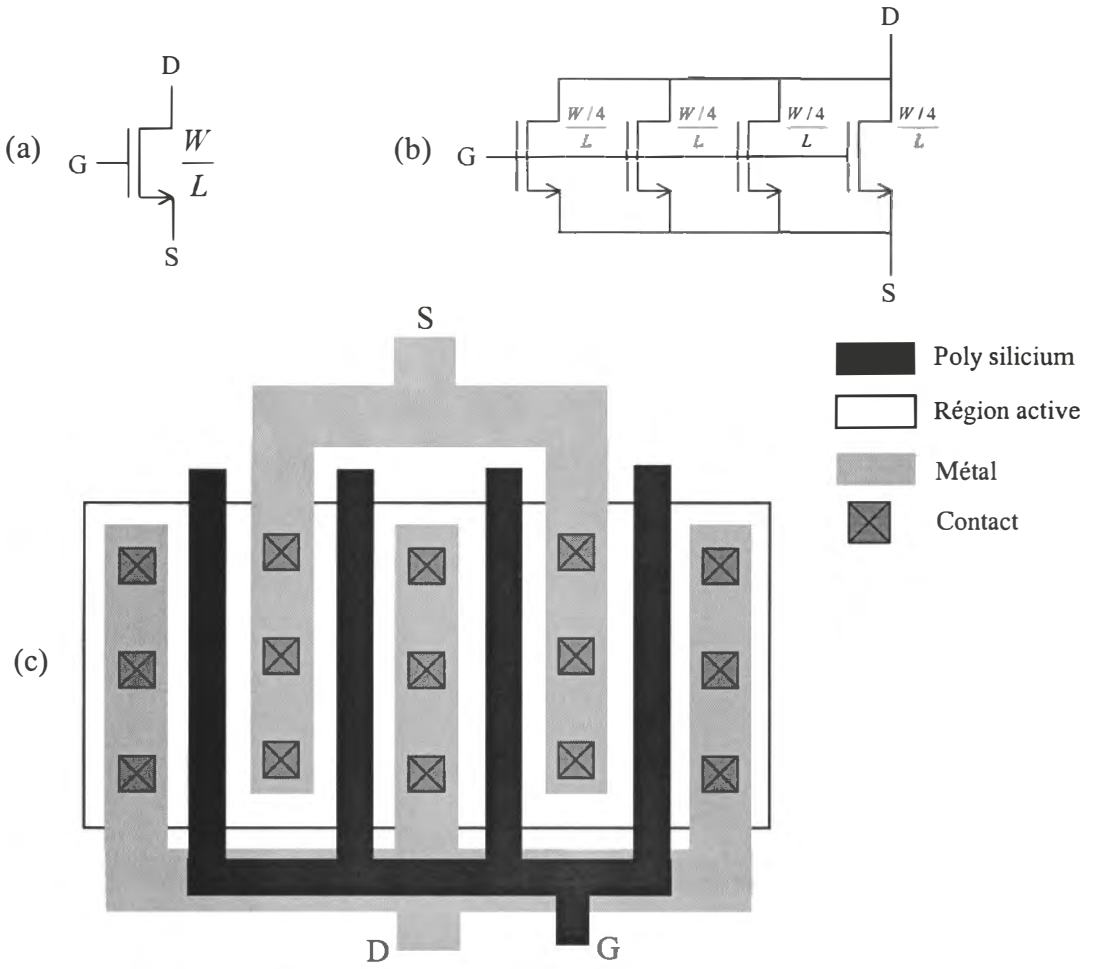


Figure F.1. Transistor de grande dimension en utilisant quatre transistors connectés en parallèle : (a) schématique ; (b) schématique équivalent avec quatre transistors ; (c) dessin des masques

Le dessin des masques comprend quatre doigts correspondant aux dessins des masques des quatre transistors élémentaires. Les transistors élémentaires ayant la même longueur, le transistor équivalent a aussi cette longueur mais sa largeur est égale à la somme des largeurs des quatre transistors.

Lorsque deux transistors de mêmes dimensions doivent être dessinés (transistors balancés), non seulement chacun des transistors doit être dessiné selon ce principe mais aussi les doigts constituant les transistors doivent être entrelacés selon une disposition appelée « centroïde commun ». Cette configuration est illustrée par la figure F.2 dans le cas de deux transistors M_1 et M_2 dont les sources sont interconnectées (paire différentielle) [Johns & Martins '97].

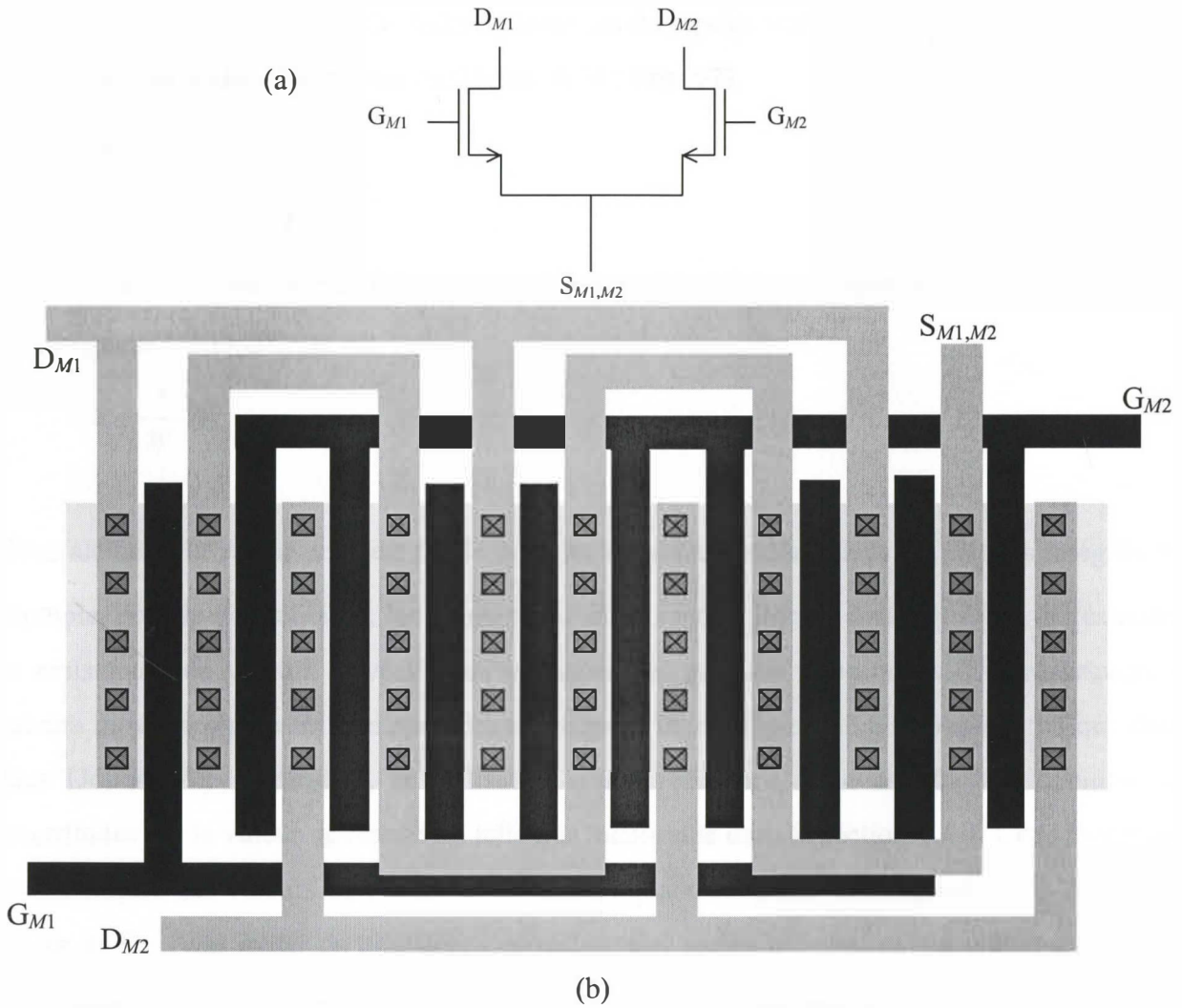


Figure F.2. Deux transistors balancés connectés en paire différentielle (sources connectées) : (a) schématique ; (b) dessin des masques selon la disposition de centroïde commun

Cette disposition favorise l'élimination des erreurs provoquées par l'effet de gradient à travers un microcircuit. Dans la figure F.2, D_{M1} désigne le drain de M_1, D_{M2} le drain de M_2, G_{M1} la grille de M_1, G_{M2} la grille de M_2 et $S_{M1,\,M2}$ la source commune à M_1 et M_2. La légende est la même que dans la figure F.1.

Résistances

Les résistances intégrées peuvent être réalisées en utilisant la propriété de résistivité d'une large gamme de matériaux conducteurs. Un choix très répandu est le polysilicium utilisé pour la réalisation des grilles de transistors et qui offre l'avantage d'être un matériau déposé et gravé lors du procédé de photolithographie. Indépendamment du type de matériau utilisé, la résistance d'une unité carrée de surface est donnée par [Johns & Martins '97]

$$R_{\mathrm{W}} = \frac{\rho}{l}$$

où ρ est la résistivité du matériau et l son épaisseur. La résistance totale R pour une largeur W et une longueur L du matériau est

$$R = \frac{L}{W} R_{\mathrm{W}}$$

Généralement, la valeur de R_{W} est faible pour les matériaux utilisés dans les circuits intégrés. Par exemple, pour le polysilicium, les valeurs courantes sont de l'ordre de 20Ω. Ainsi, la réalisation de résistances de valeurs élevées peut nécessiter des surfaces importantes. Des résistances de valeurs moyennes peuvent être réalisées en utilisant la topologie dite « serpentin » tel que décrit dans [Johns, 97]. Cependant, la réalisation d'une résistance seule est toujours soumise aux incertitudes sur la valeur de résistivité telle que mentionné dans la section 4.4.4. Cette incertitude peut atteindre des valeurs de l'ordre de 20% impliquant donc une erreur possible de 20% sur la valeur de R_{W}. Pour éviter ce problème, les résistances seules nécessaires ont été remplacées par des transistors connectés en résistances. Il ne reste alors que des rapports de résistances à réaliser comme pour le montage sommateur de la figure 4.5. Dans le cas des rapports de résistances, le problème d'incertitude relatif à la résistivité du matériau utilisé est nettement moins important. En

effet, dans un rapport de résistances, c'est le rapport des dimensions géométriques et non la valeur précise de R_w qui déterminera précisément la valeur du rapport. La précision devient alors une question de précision des géométries sur le circuit final. Bien sûr, même si c'est la précision du rapport des résistances qui prime, les valeurs elles-mêmes des résistances doivent être relativement proches des valeurs désirées pour des raisons d'adaptation d'impédances le plus souvent.

La figure F.3 présente le dessin des masques d'une résistance sous la forme d'un serpentin. Notons que pour le calcul de la valeur de la résistance, les résistances de contacts et de courbures doivent être prises en comptes comme décrit dans [Johns & Martins '97].

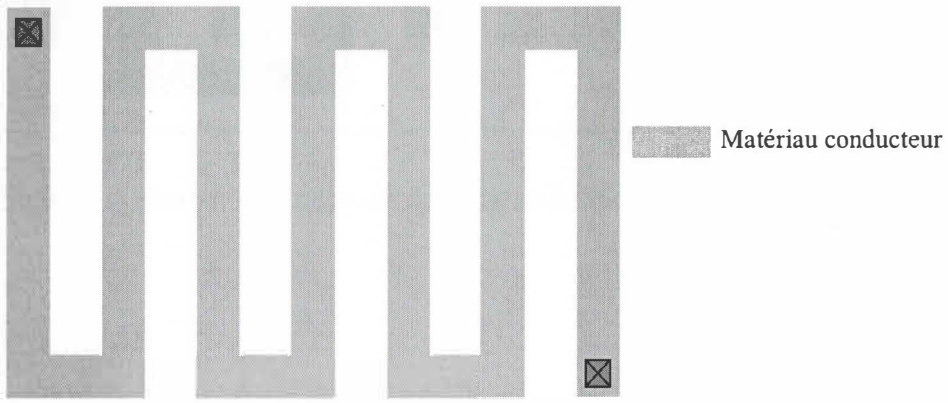


Figure F.3. Dessin des masques d'une résistance intégrée sous la forme d'un serpentin

La figure F.4 présente le dessin des masques de deux résistances égales R_1 et R_2. L'entrelacement des serpentins permet une annulation des effets de gradients mentionnés plus haut et de compenser les effets d'impédances entre les deux résistances (impédance mutuelle). Les doigts constituant les deux résistances sont connectés par du métal de faible résistivité. Deux résistances factices (*angl. dummy*) sont ajoutées pour satisfaire les conditions aux limites.

Pour réaliser deux rapports de résistances égaux R_2/R_1, par exemple pour le montage sommateur, il suffit alors de réaliser les deux résistances R_1 tel que dans la figure F.4 et les deux résistances R_2 sur le même principe en modifiant les dimensions géométriques pour obtenir le bon rapport R_2/R_1. Enfin, il est conseillé d'ajouter une couche protectrice sous l'ensemble formant les résistances, couche connectée à une alimentation propre, afin de protéger les résistances des bruits de substrat. Cette couche protectrice peut être un puits N ou P et permet d'éliminer les bruits

pouvant être induits par le substrat à cause du couplage capacitif entre le substrat et la structure de la résistance.

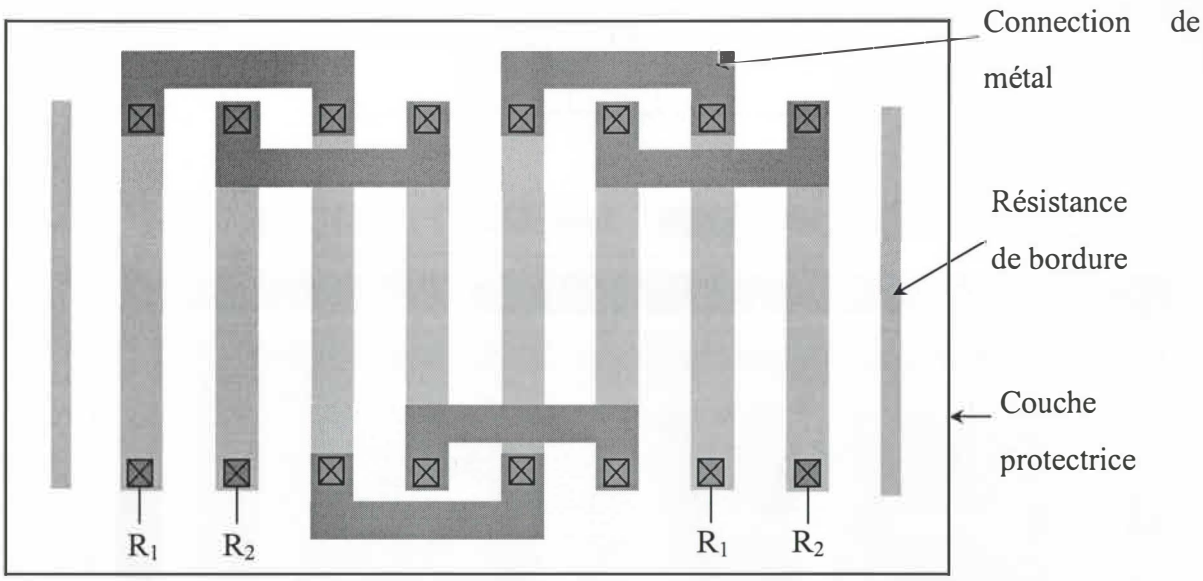


Figure F.4. Dessin des masques de deux résistances égales R_1 et R_2

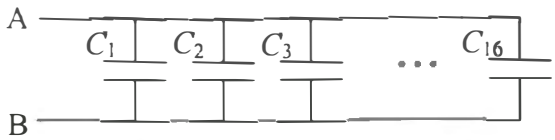
Capacités

Les principales sources d'erreurs dans la réalisation de capacités intégrées sont dues d'une part au phénomène appelé *overetching* qui rend les surfaces réelles plus petites que celles des masques, d'autre part au gradient d'épaisseur d'oxyde à travers le microcircuit [Johns & Martins '97]. Le premier, habituellement dominant, peut être réduit en réalisant les grands condensateurs à l'aide de condensateurs élémentaires de plus petites surfaces connectés en parallèle. Cette fois encore, l'intérêt est de réaliser des rapports de capacités pour lesquels les techniques vues pour le dessin des masques de résistances s'appliquent. La figure F.5 présente le dessin des masques d'un condensateur constitué de seize condensateurs élémentaires connectés en parallèle. Les condensateurs élémentaires sont réalisés par les surfaces parallèles de polysilicium Poly1 et Poly2 en regard.

Suivant le même principe, deux capacités balancées peuvent être réalisées en entrelaçant les condensateurs dans une disposition centroïde commun telle qu'illustrée dans la figure F.5. En

plus de cette disposition, plusieurs principes doivent être appliqués pour l'obtention d'un balancement précis.

(a)



(b)

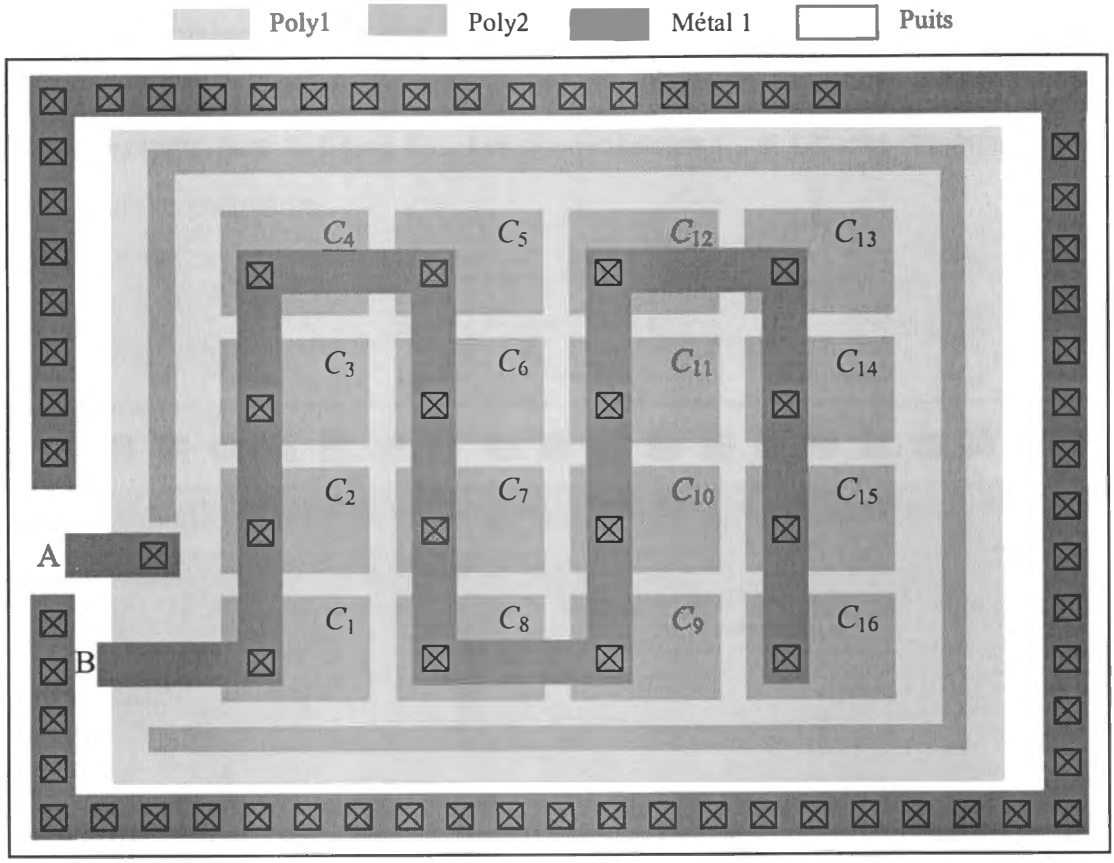


Figure F.5. Réalisation d'un condensateur en connectant seize condensateurs élémentaires en parallèle : (a) schématique ; (b) dessins de masques

Habituellement, la base des condensateurs sera la première couche de polysilicium mais pour certaines technologies c'est la région d'implantation ionique qui est utilisée. Cette région est habituellement commune à beaucoup de condensateurs de dimensions unitaires. L'interconnexion des plateaux supérieurs de la capacité, qui sont presque toujours en polysilicium, peut souvent être réalisée en utilisant le métal 1 avec des contacts de polysilicium. Pour certaines technologies

ce n'est pas possible et les plateaux doivent être interconnectés par des contacts de polysilicium sur les côtés. Ces zones entreront en contact avec le métal 1 dans une région où la base de polysilicium est absente. Les capacités parasites de ces zones devraient être balancées autant que possible. Ce balancement nécessite souvent l'ajout de zones de contacts supplémentaires qui ne sont pas reliées. Un autre principe est d'assurer que les conditions limites autour du condensateur à dimension unitaire sont satisfaites. Ceci est accompli en ajoutant du polysilicium supplémentaire au plateau autour des bornes extérieures des condensateurs de dimensions unitaires. Les différents principes énoncés se retrouvent dans le dessin des masques de deux condensateurs présenté dans la figure F.6. Les condensateurs C_1 et C_2 sont constitués chacun de huit condensateurs élémentaires.

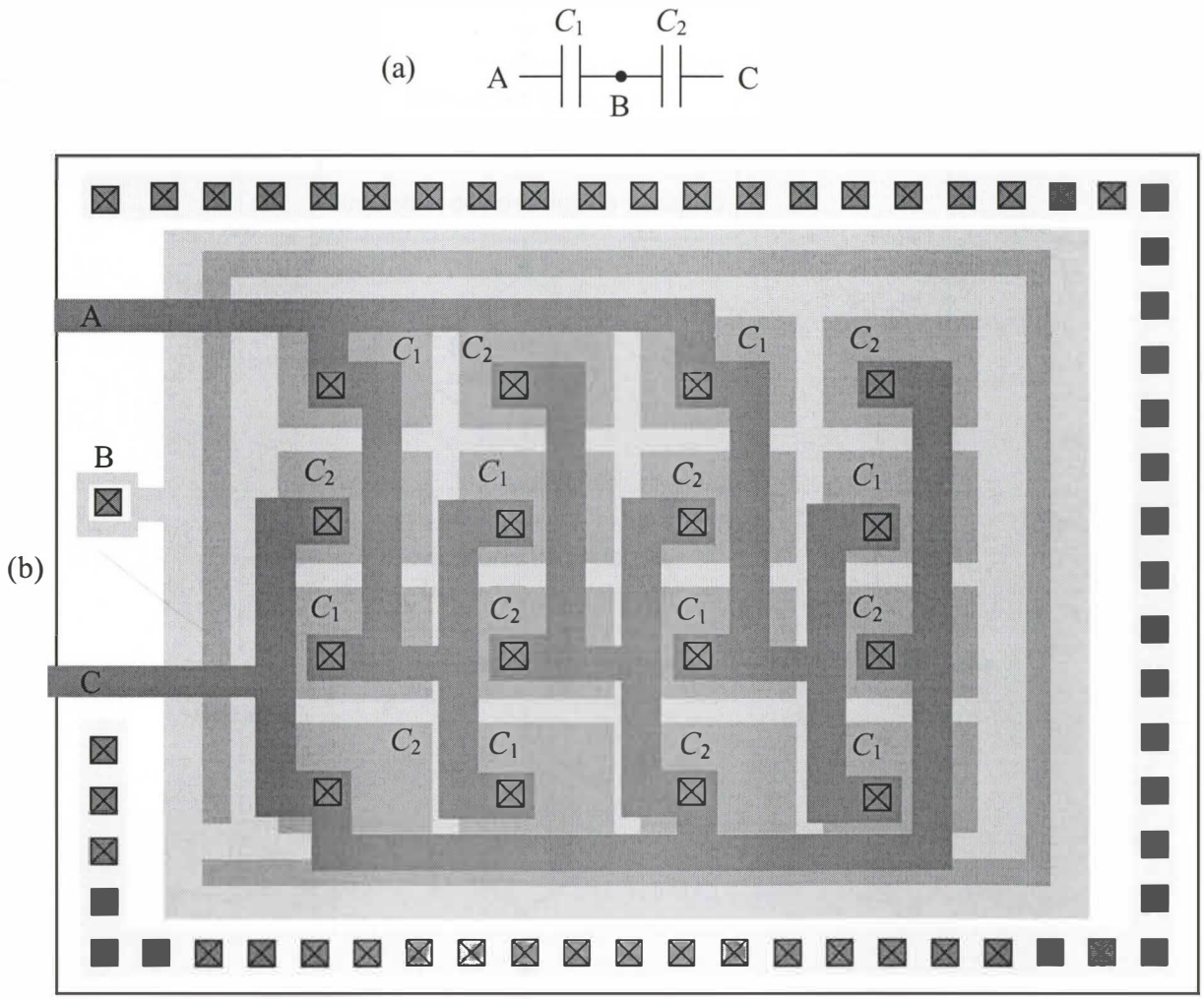


Figure F.6. Réalisation de deux condensateurs C_1 et C_2 avec la configuration de centroïde commun : (a) schématique ; (b) dessins de masques

Annexe E : Diagramme de conception d'un circuit intégré selon [Baker, 98]

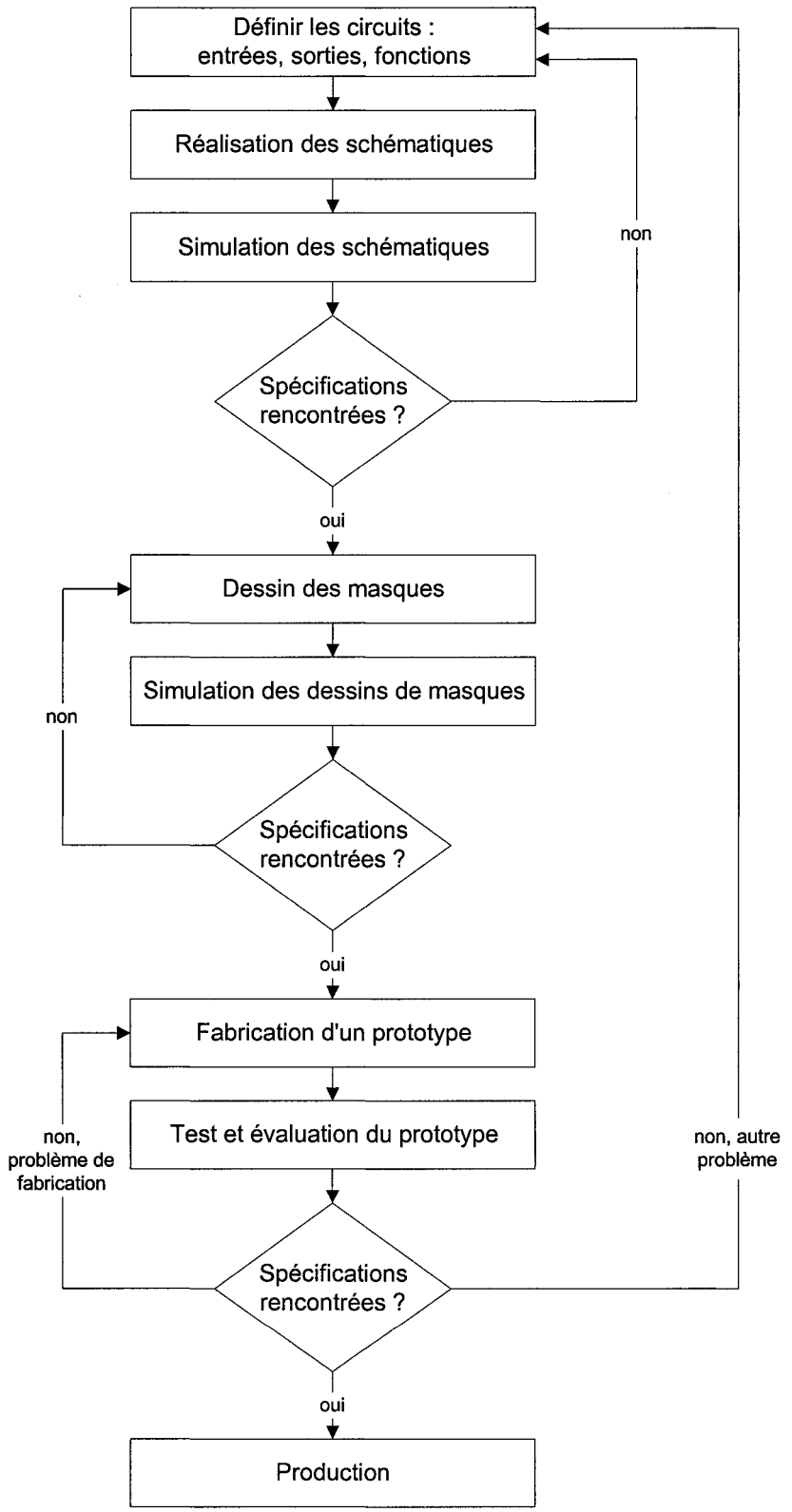


Figure D.1. Diagramme de conception d'un circuit intégré selon [Baker, 98]

Annexe F : Diagramme de réalisation des dessins de masques selon [Baker, 98]

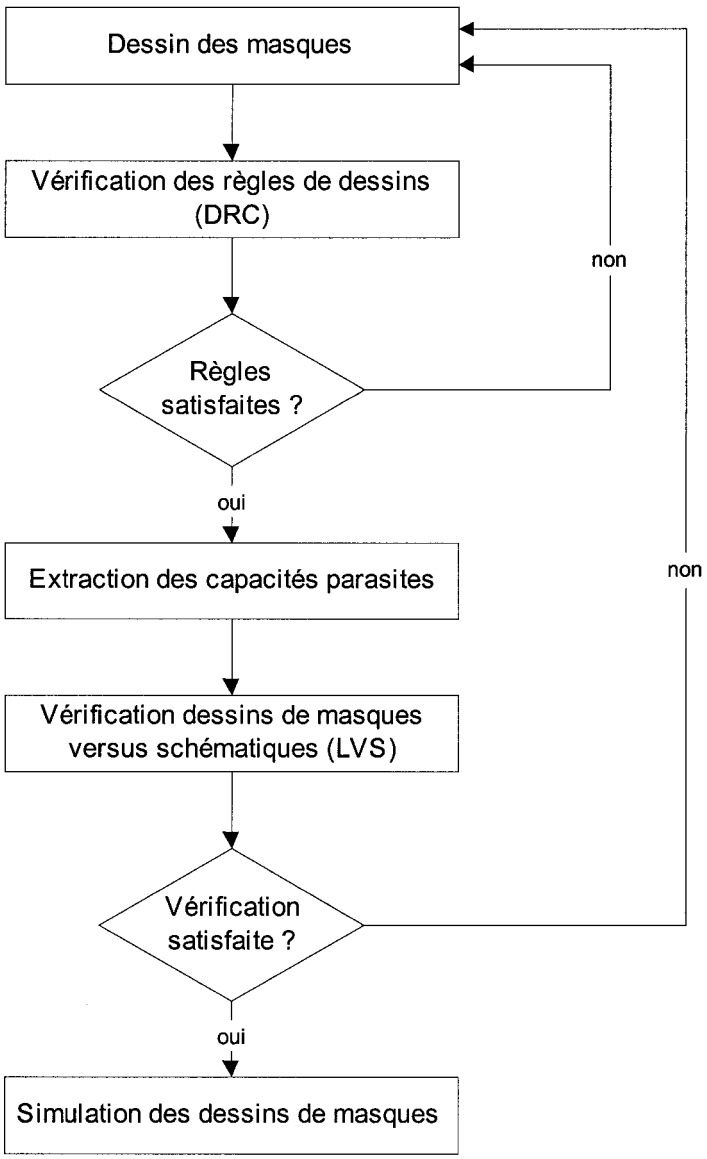


Figure E.1. Diagramme de réalisation des dessins de masques selon [Baker, 98]

Annexe G : exemple de dessins de masques : l'AOT

À partir des dessins de masques des éléments de base (transistors, résistances et capacités), tous les dessins de masques des différents circuits peuvent être réalisés en connectant les éléments correctement dimensionnés conformément aux schématiques présentés dans le quatrième chapitre. À titre d'exemple, le dessin complet des masques de l'AOT est présenté dans la figure G.1. Nous pouvons y distinguer les éléments de base présentés en annexe D. Cette vue nous permet aussi de constater la proportion de surface importante occupée par les condensateurs.

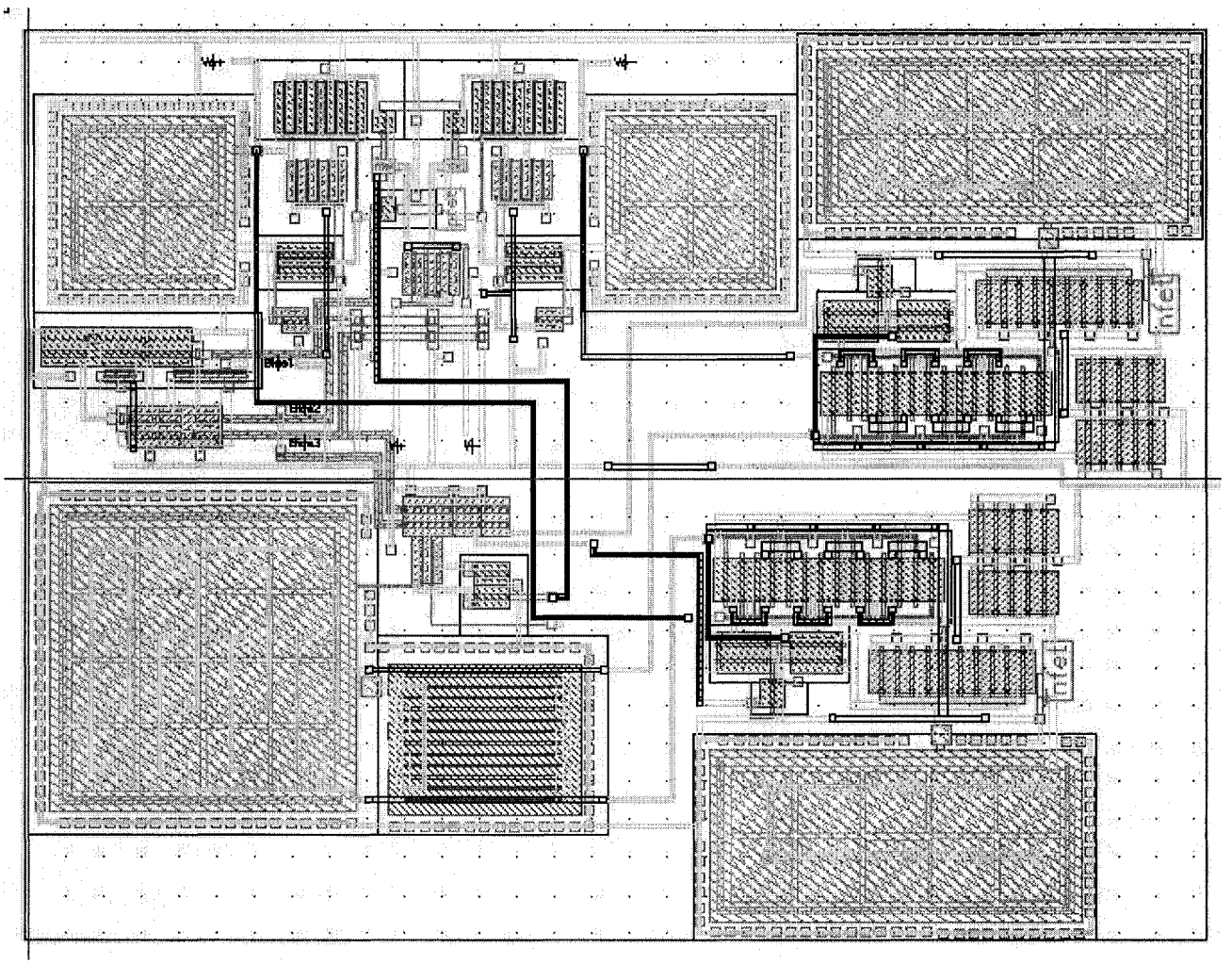


Figure G.1. Exemple de dessins de masques : l'AOT

Annexe H : courbes de gain et de phase de l'AOT

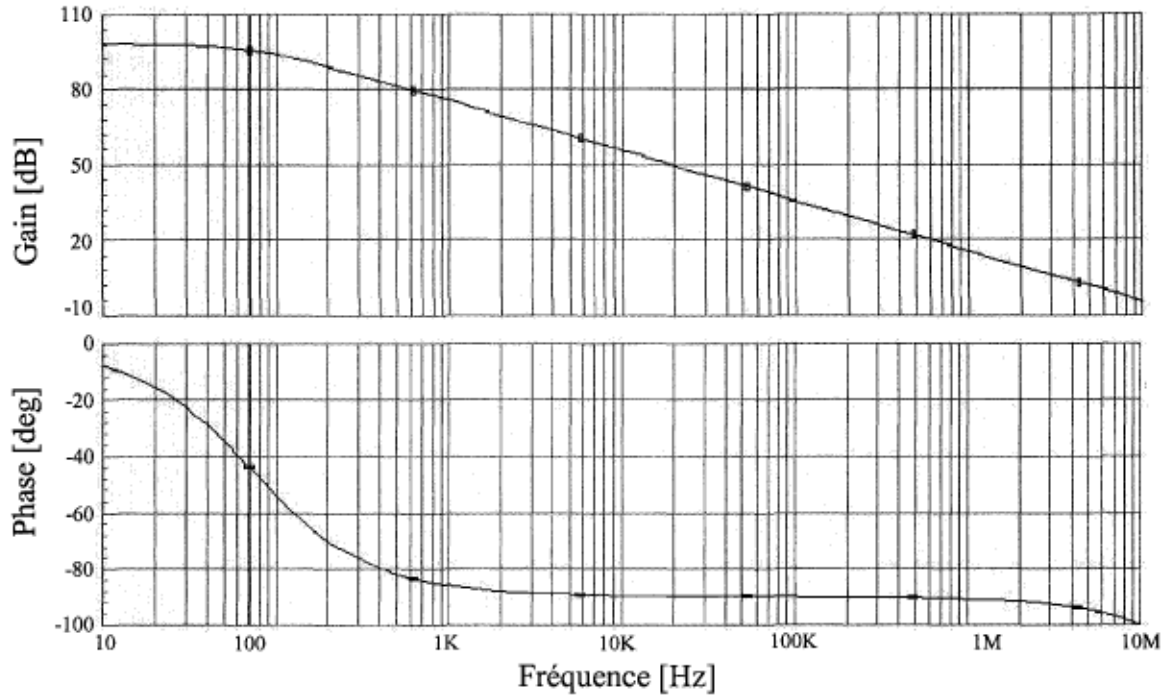


Figure H.1. Courbes de gain et de phase de l'AOT en fonction de la fréquence