UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

MÉMOIRE PRÉSENTÉ À
UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN ÉLECTRONIQUE INDUSTRIELLE

PAR
ABDELKRIM MEFTAH

APPLICATION DES RÉSEAUX DE NEURONES DANS LE DOMAINE DE L'ÉLECTRONIQUE DE PUISSANCE

JUIN 1997

RÉSUMÉ

Dans ce travail de recherche on aborde l'application des réseaux de neurones artificiels dans le domaine de l'électronique de puissance dont les systèmes sont en général non linéaires. On présente une brève introduction sur les réseaux de neurones et ainsi que la validation des modèles plus ou moins réels des convertisseurs utilisés dans ce travail. Deux applications principales portent sur les convertisseurs de puissance. La première application montre qu'un réseau de neurones est capable d'apprendre une limitation du courant. La deuxième application est la régulation de la tension de sortie d'un redresseur triphasé survolteur à facteur de puissance unitaire. Deux types de régulateurs sont proposés, un régulateur à réseau de neurones statique et un régulateur à réseau de neurones dynamique. Une méthode de conception de ce dernier est proposée qui peut être généralisée pour tous les convertisseurs de ce type. Une comparaison de ces régulateurs permet de voir la supériorité des performances du régulateur dynamique en temps de réponse et en robustesse. Enfin une étude de l'implantation dans un processeur numérique est présentée. En réalisant des blocs de quantification dans l'environnement SIMULINK®, on détermine le nombre de bits, le type de quantification et la représentation. Ainsi, un besoin en quantité de mémoire a été déterminé pour l'implantation.

REMERCIEMENTS

Je ne saurais finir ce mémoire sans exprimer mes plus sincères remerciements:

-à mon directeur de recherche Daniel Massicotte, de l'Université du Québec à Trois-Rivières (UQTR) et à mon codirecteur Ziwen Yao, chercheur scientifique au sein de la chaire de recherche industrielle, Hydro-québec - CRSNG. Qu'ils trouvent ici le témoignage de mes plus chaleureux remerciements pour l'honneur et le privilège qu'ils m'ont faits en acceptant de diriger mon travail de recherche de même que pour la part de responsabilités qu'ils m'ont accordée et pour leurs conseils, leurs suggestions et leur aide financière.

-au professeur V. Rajagopalan, de l'Université du Québec à Trois-Rivières (UQTR), titulaire de la chaire de recherche industrielle, Hydro-québec - CRSNG sur les procédés Efficaces Électrothermiques, pour ses conseils et son aide financière.

-au professeur Ahmed Cheriti, de l'UQTR, pour ses conseils techniques et l'aide inestimable qu'il m'a apportée. Il m'est difficile de témoigner ici toute ma gratitude et ma reconnaissance pour ses réelles qualités humaines, qu'il trouve ici l'expression de ma sincère amitié.

-aux membres du corps professoral de la maîtrise en électronique industrielle pour m'avoir transmis leurs connaissances en particulier Monsieur A.Lakhsasi.

-Je remercie également mes frères A. Ba-razzouk, N. Chouhaid, A. Maçbahi et S. Messaoudi, pour leur support moral et technique.

-Et, à mon père, ma mère et surtout ma femme Souad et mes enfants Kaouthar, Abdelhak et Abdelhakim pour leur aimable compréhension.

TABLE DES MATIÈRES

pages

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SYMBOLES ET DES ABRÉVIATIONS

b	nombre de bits
b_i	seuil du neurone i
C	capacité de sortie
CA	courant alternatif
CC	courant continu
C_f	capacité du filtre d'entrée
D	rapport cyclique
$D_{1,\ldots,6}$	diodes du pont redresseur
E	erreur quadratique globale
e_k	erreur entre la valeur désirée et la sortie k
e_q	erreur absolue maximale
e_{rq}	erreur relative maximale
f_p	fréquence de commutation
f_s	fréquence du réseau électrique
$f(x)$	fonction d'activation
$i_{b,R}$	courant dans l'inductance L_b
i_{ch}	courant dans la charge
i_D	courant de diode
i_{Lb}	courant dans l'inductance L_b
i_s	courant de source
$i_{N,R}$	courant de source de la phase R
i_Q	courant dans le transistor
L_b	inductance du survolteur et de l'abaisseur
L_c	inductance critique

L_f	inductance du filtre
P	vecteur d'entrée du réseau de neurones (apprentissage)
PG	vecteur d'entrée du réseau de neurones (généralisation)
q	pas de quantification
Q	transistor
r_c	résistance interne de C
r_{ch}	variation de R_{ch}
R_{ch}	résistance de charge
r_{Cf}	résistance interne de C_f
r_{Lb}	résistance interne de l'inductance L_b
r_{Lf}	résistance interne de L_f
RNA	réseaux de neurones artificiels
s_i	sortie du neurone i
T	vecteur de sortie désirée (apprentissage)
TG	vecteur de sortie désirée (généralisation)
T_p	période de commutation
u(k)	entrée du système à l'instant k
$U_{N,R}$	tension entre phases
V_c	tension de commande
V_{ch}	tension de charge
V_{cq}	tension de commande quantifiée
V_D	tension aux bornes de la diode
$V_{N,R}$	tension simple du réseau
V_Q	tension aux bornes du transistor
$V_{réf}$	tension de référence
V_s	tension de source

W_{ij}	poids du réseau neurone
x_j	entrée élémentaire du neurone i
x_q	signal quantifié
X_{max}	maximum du signal x
X_{min}	minimum du signal x
y(k)	sortie du système à l'instant k
z_i	entrée totale du neurone i

INTRODUCTION GÉNÉRALE

Depuis l'apparition des diodes et les thyristors au début des années soixantes, le domaine de l'électronique de puissance s'intéresse aux dispositifs permettant la conversion d'un type de courant alternatif ou continu en un autre CA ou CC. Les convertisseurs CC-CC et CA-CC prennent une grande part dans les applications industrielles. Cependant, le fonctionnement de ces convertisseurs affectent la qualité d'onde du courant de source, ce qui engendre un mauvais facteur de puissance. Plusieurs topologies ont été proposées pour satisfaire les nouvelles normes spécifiant le maximum de la distorsion du courant source et la valeur absolue des harmoniques de celui-ci [4-6]. Une autre préoccupation qui tient l'attention des chercheurs est la conception des régulateurs pour ces systèmes qui sont non linéaires. Ces régulateurs doivent assurer le réglage de la tension de charge ainsi qu'un facteur de puissance unitaire.

Dès les années 90, quelques applications des réseaux de neurones artificiels (RNA) ont vu le jour dans le domaine de l'électronique de puissance. L'intelligence artificielle est une approche issue à la fois des sciences de traitement de l'information et de la neurobiologie [8-11]. L'effort effectué par les neurobiologistes, les mathématiciens et les informaticiens a donné naissance à ce domaine et plusieurs algorithmes sont disponibles permettant l'apprentissage d'un comportement ou une tâche donnée. Les RNA permettent de résoudre certains problèmes reconnus comme difficiles tels que l'optimisation, la classification, la reconnaissance des formes, le diagnostic et la précision. Les premières applications des réseaux de neurones dans l'électronique de puissance sont apparues dans la correction de facteur de puissance [19-20] et la commande des machines électriques [21-23].

Dans ce travail de recherche, nous allons montrer quelques applications des réseaux de neurones dans le contrôle et la protection des convertisseurs de puissance. Trois montages sont choisis, deux convertisseurs de types continu - continu (CC-CC) et le troisième de type alternatif - continu (CA-CC). Ces montages sont des systèmes non linéaires.

Le chapitre 1 présente la modélisation et la validation des convertisseurs à étudier, deux convertisseurs CC-CC dont le premier est un montage abaisseur et le second est un montage survolteur. Le troisième est un convertisseur CA-CC qui est un redresseur triphasé survolteur à facteur de puissance unitaire que n'a pas été l'objet d'un réglage par les RNA. Pour les deux premiers, une étude théorique montre les différents fonctionnements en régime continu et discontinu. Le troisième montage fonctionne toujours en régime discontinu. Une validation de ces modèles est faite dans l'environnement MATLAB®/SIMULINK®. Ce chapitre permet la préparation des trois montages qui seront contrôlés par les réseaux de neurones.

Le chapitre 2 présente des généralités sur les réseaux de neurones artificiels (RNA). On commence par une présentation du fondement biologique qui est la base des RNA. Ensuite une formulation mathématique d'un neurone, ainsi que l'architecture et les types de ces réseaux, une brève présentation de l'algorithme de rétropropagation et à la fin quelques applications des RNA dans l'électronique de puissance sont présentées.

Le chapitre 3 présente l'application des réseaux de neurones comme régulateur statique (sans prendre en considération la dynamique du système) pour les trois montages. L'apprentissage et la généralisation sont faites sur MATLAB® et la validation du régulateur statique avec chaque modèle est faite sur l'environnement SIMULINK®.

Le chapitre 4 présente l'application d'un régulateur dynamique à réseaux de neurones pour le montage redresseur triphasé. Ce régulateur diffère du premier par

l'apprentissage de la dynamique du système et ce dernier est entraîné par un modèle en réseaux de neurones du système à contrôler. Une comparaison entre les deux régulateurs est effectuée à la fin de ce chapitre.

Le chapitre 5 présente la réalisation de quatre blocs de quantification dans l'environnement SIMULINK® et une étude de l'effet de la quantification numérique sur le modèle d'un neurone. On commence par la présentation des quatre quantifications à virgule fixe, ensuite la programmation et l'implantation de ces derniers dans l'environnement MATLAB®/SIMULINK®. Enfin une comparaison entre un neurone quantifié en virgule fixe et un neurone à virgule flottante est réalisée ainsi qu'un choix du nombre de bits et une mise en oeuvre sont faites pour l'implantation du régulateur dynamique .

CHAPITRE 1

MODÉLISATION ET VALIDATION DES CONVERTISSEURS DE PUISSANCE DANS L'ENVIRONNEMENT MATLAB®/SIMULINK®

1.1 Introduction

Dans les équipements industriels, pour alimenter les charges sous tension continue convenablement, on a besoin d'une alimentation continue généralement régulée pour une gamme de variation de la tension de source et de la résistance de charge. Dans ce chapitre, nous allons faire l'étude des trois montages: deux montages de type convertisseurs CC-CC et le troisième de type convertisseur triphasé du courant CA-CC.

Les convertisseurs CC-CC sont le hacheur abaisseur et le hacheur survolteur. Le premier permet d'obtenir une tension continue variable inférieure à la tension de source. Le deuxième est un montage survolteur comme indiqué par son nom. Dans la pratique, ces montages fonctionnent en haute fréquence permettant la réduction de la taille des composants réactifs du montage.

Le troisième montage est de type convertisseur CA-CC. C'est le redresseur triphasé survolteur mono-interrupteur à facteur de puissance unitaire. Ce dernier fonctionne en mode discontinu avec une commande non linéaire. Ceci est une topologie typique parmi plusieurs topologies de redresseurs à facteur de puissance unitaire. Ces montages ont des entrées-sorties non linéaires dues aux éléments du montage et du mode de fonctionnement continu ou discontinu.

Dans ce chapitre nous présentons l'étude des trois montages choisis ainsi que la simulation de ces derniers. Premièrement, on va présenter le fonctionnement de chaque montage en régimes continu et discontinu. La simulation de ces derniers prennent en considération toutes les pertes dans les éléments du montage.

1.2 Hacheur abaisseur

1.2.1 Schéma du montage

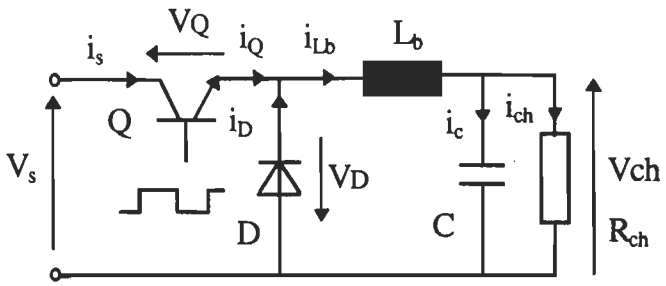


Figure 1.1: Montage du hacheur abaisseur idéal

Le hacheur abaisseur est un convertisseur CC-CC. Il est alimenté par une source de tension généralement à travers un pont redresseur monophasé ou une batterie. Il débite sur un récepteur de courant et permet la variation de la tension de sortie V_{ch} de zéro à V_s.

1.2.2 Principe de fonctionnement

Ce hacheur peut fonctionner en deux modes: continu et discontinu. La continuité ou la discontinuité du courant dans l'inductance L_b de sortie détermine le mode de fonctionnement. Ces régimes de fonctionnement dépendent fortement de la charge et des éléments du montage [1-2].

2.2.2.1 Régime continu

Intervalle (0<t<DT)

Le transistor Q est en conduction, la diode est bloquée. la figure 1.2 montre le schéma équivalent du hacheur abaisseur (la figure 1.5 montre les formes d'ondes correspondantes). La tension de source V_s est supérieure à la tension de charge V_{ch}, et le courant dans l'inductance augmente linéairement de I_m (min) à I_M (max), permettant la charge de la capacité de sortie.

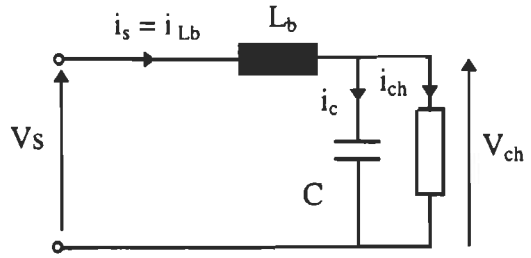


Figure 1.2: Circuit équivalent lorsque le transistor est en conduction

L'équation de tension est la suivante:

$$V_s - V_{ch} = L_b \frac{di_s}{dt} \tag{1.1}$$

L'intervalle de conduction du transistor:

$$t_{on} = L_b \frac{I_M - I_m}{V_s - V_{ch}} \tag{1.2}$$

Cet intervalle est caractérisé par le stockage de l'énergie électrique sous forme magnétique dans l'inductance.

Intervalle (DT<t<T)

Le transistor est bloqué à l'instant DT jusqu'à T. La diode est en conduction et le courant dans l'inductance ne peut avoir une discontinuité. Cette inductance change de polarité ce qui permet la polarisation en directe de la diode et cette dernière commence à conduire. Dans cet intervalle, on n'a qu'une décharge partielle de l'inductance.

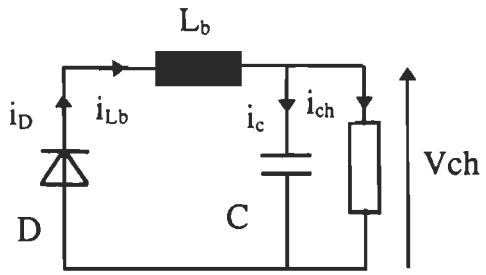


Figure 1.3: Schéma équivalent lorsque le transistor est bloqué

Le rapport entrée-sortie est le suivant:

$$V_{ch} = D.V_s \tag{1.3}$$

1.2.2.2 Régime discontinu

Ce régime de fonctionnement est caractérisé par une discontinuité du courant dans l'inductance L_b, c'est-à-dire que celle-ci se décharge complètement avant la fin de la période.

Intervalle (0<t<DT)

C'est le même fonctionnement que le régime continu sauf que le courant I_m est égale à zéro car l'inductance est complètement déchargée. La valeur de l'inductance critique est la suivante:

$$L_c = \frac{R(1-D)}{2f_s} \tag{1.4}$$

Si L_b est inférieur à L_c, le régime est discontinu, sinon il est continu. Le circuit équivalent est le même que celui de la figure 1.2.

Intervalle (DT<t<t_2)

Dans cet intervalle, l'inductance se décharge complètement. Le circuit équivalent est le même que celui de la figure 1.3.

Intervalle ($t_2<t<T$)

Le courant est nul dans l'inductance, la capacité C se décharge dans la résistance R_{ch} (la charge). Le transistor et la diode sont tous les deux bloqués. La figure 1.4 montre le schéma équivalent du circuit dans cet intervalle.

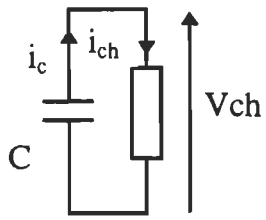


Figure 1.4: Circuit équivalent lorsque la diode et le transistor sont bloqués

Le rapport des tensions entrée-sortie est le suivant [2]:

$$\frac{V_{ch}}{V_s} = \frac{2}{1+\sqrt{1+4L_b(1-D)/D^2L_c}} \tag{1.5}$$

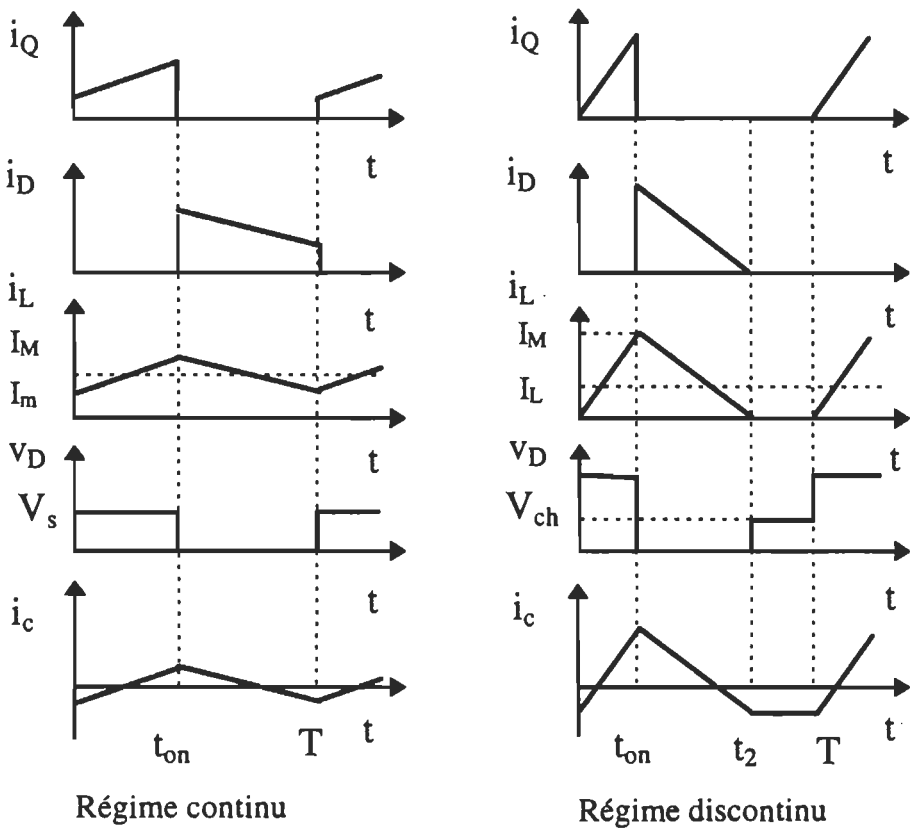


Figure 1.5:Formes d'ondes pour un abaisseur en régimes continu et discontinu

1.2.3 Simulation du montage

Le montage est simulé dans l'environnement MATLAB®/SIMULINK®. Les composants passifs, comme la capacité et l'inductance, sont modélisés par un élément idéal en série avec une résistance r qui représente les pertes dans ces éléments. Les semi-conducteurs sont modélisés pour une résistance en série avec une inductance, la résistance est très faible à la conduction et très forte au blocage et enfin une inductance permet de prévoir le régime transitoire à l'ouverture et à la fermeture du semi-conducteur [3].

L'utilisation de l'environnement SIMULINK® se fait en trois phases:

- Mise en équations (courant et tension) du montage à simuler.
- Construction du montage sous forme de blocs.
- Choix des paramètres de simulation et le lancement de celle-ci.

Les paramètres du hacheur sont donnés dans le tableau 1.1:

Tableau 1.1: Paramètres des hacheurs

Tension de source, Vs	24V +10% -15%
Inductance, L_b	1mH
Résistance interne, r_{Lb}	0.05Ω
Capacité de sortie, C	300μF
Résistance interne, r_c	0.05Ω
Fréquence de commutation, fs	5 kHz
Fréquence d'échantillonnage	5kHz

1.2.3.1 Modèle du hacheur abaisseur

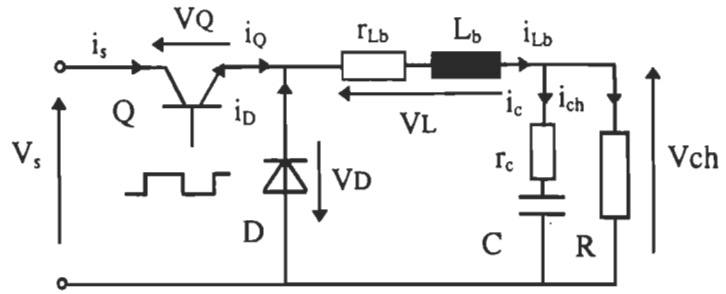


Figure 1.6: Schéma du hacheur abaisseur utilisé en simulation

La méthodologie permettant la mise en équation du modèle (voir figure 1.6) en MATLAB®/SIMULINK® consiste à choisir une équation de tension et une équation de courant alternativement en prenant en considération les conventions du sens du courant et la tension dans les modèles des composants.

Les équations de ce hacheur sont les suivantes :

$$V_Q = V_S + V_D \tag{1.6}$$

$$i_{L_b} = i_Q + i_D \tag{1.7}$$

$$V_D = -\left(V_{ch} + V_{L_b}\right) \tag{1.8}$$

$$i_C = i_{L_b} - \frac{V_{ch}}{R} \tag{1.9}$$

Les figures 1.7 et 1.8 montrent le schéma en bloc du montage abaisseur dans l'environnement MATLAB®/SIMULINK®.

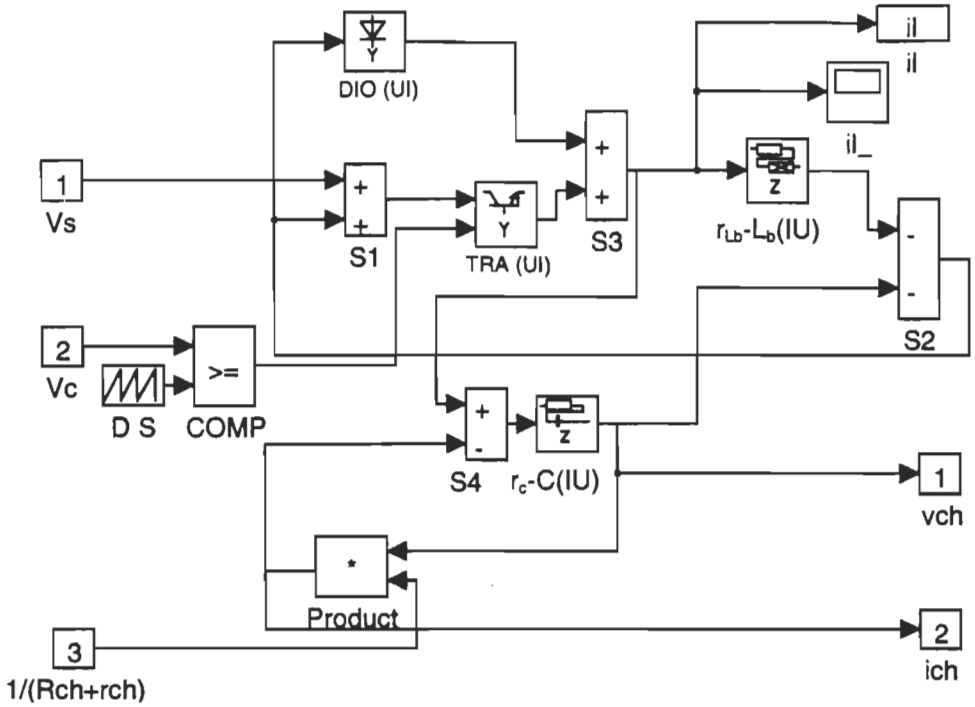


Figure 1.7: Schéma bloc du modèle de l'abaisseur dans SIMULINK®

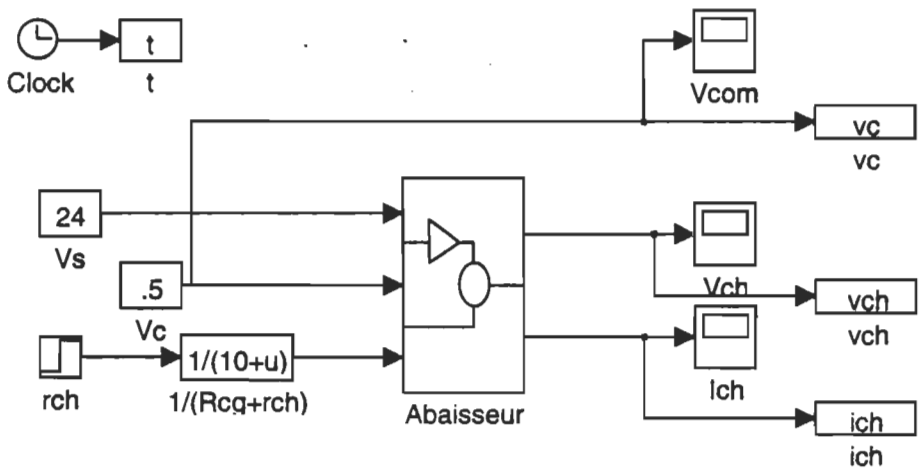


Figure 1.8: Schéma bloc complet du montage abaisseur dans SIMULINK®

1.2.3.2 Résultats de simulation

Pour ce montage, la tension de source est de 24V nominale avec +10% et -15%. Deux simulations sont faites montrant le régime continu et le régime discontinu.

La figure 1.9 met en évidence le fonctionnement discontinu de l'abaisseur. Le courant dans l'inductance L_b est nul après la décharge complète de celle-ci. La tension d'alimentation est de 24V avec une tension V_c de commande égale à 0.4V et une résistance de charge égale à 50Ω.

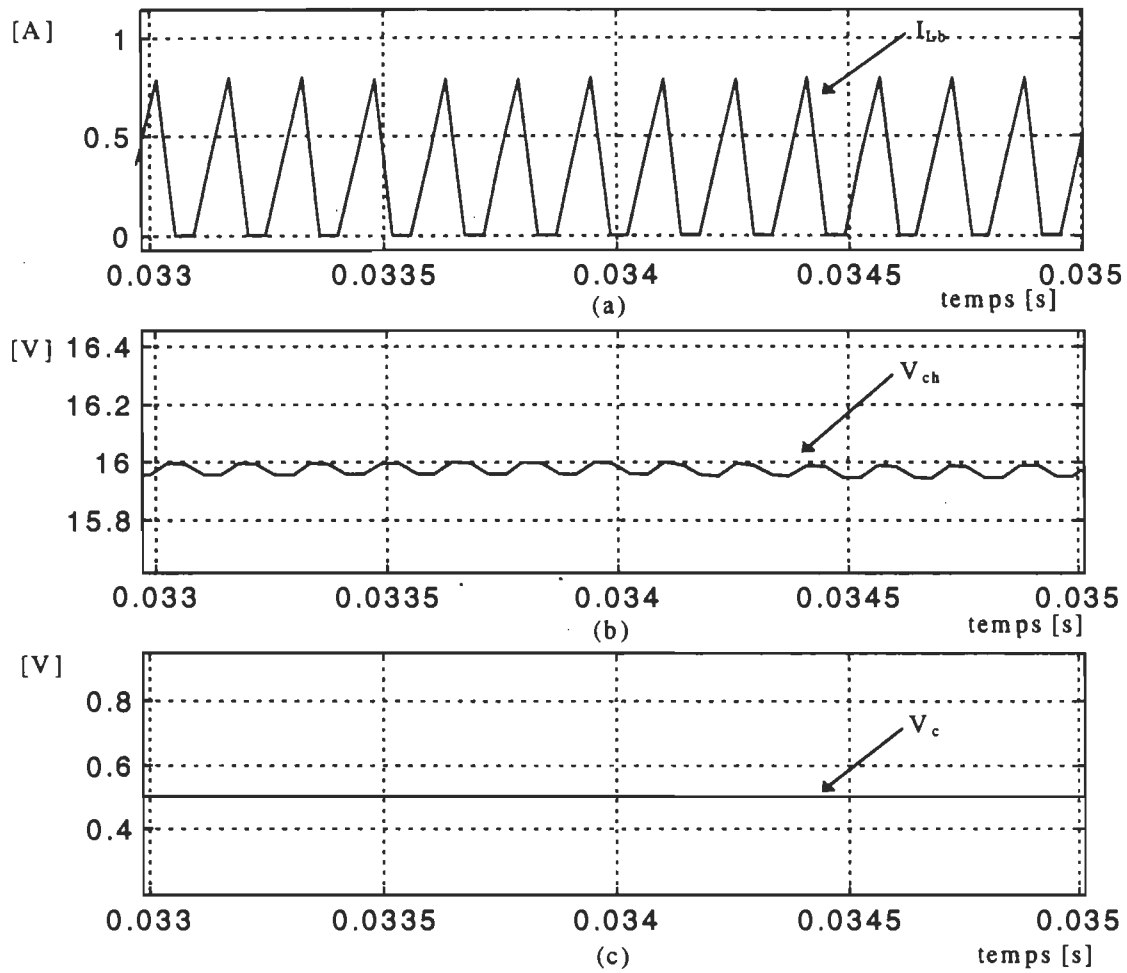


Figure 1.9 : Fonctionnement de l'abaisseur en régime discontinu:

a) courant dans L_b, b) tension de charge et c) tension de commande

Le régime continu est caractérisé par l'établissement d'un courant d'inductance L_b dont la valeur instantanée est toujours positive. La figure 1.10 montre l'allure du ccourant I_{Lb} dans l'inductance, la tension V_{ch} ainsi que la tension de commande V_c de ce fonctionnement.

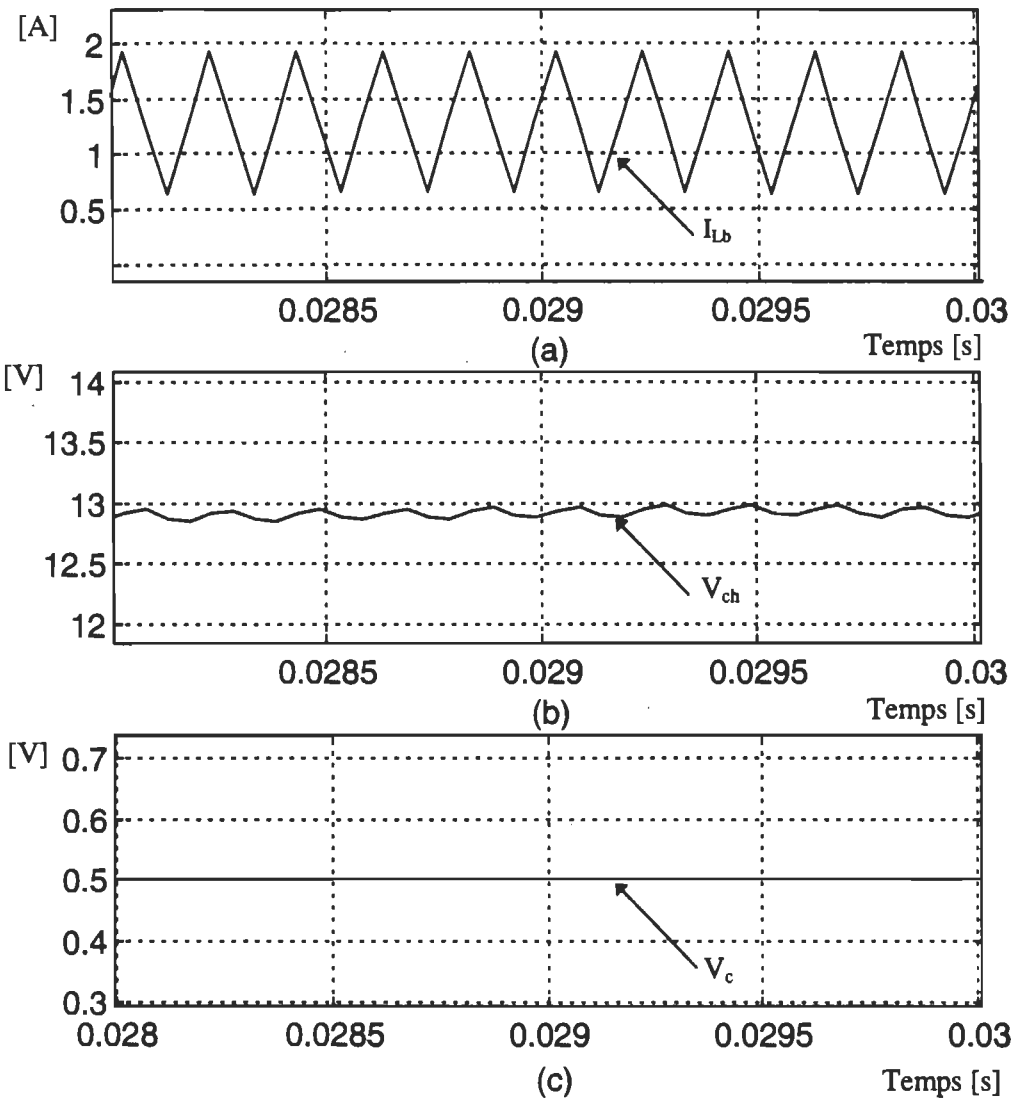


Figure 1.10: Fonctionnement de l'abaisseur en régime continu

a) courant dans L_b, b) tension de sortie et c) tension de commande

1.3 Hacheur survolteur

1.3.1 Schéma du montage

Ce montage est généralement appelé un hacheur survolteur ou un montage "boost". Il est capable de donner une tension de charge V_{ch} supérieure à la tension de source V_s. Les composants qui constituent ce montage sont les mêmes que pour l'abaisseur utilisé dans la section 1.2. La figure 1.11 montre le hacheur survolteur.

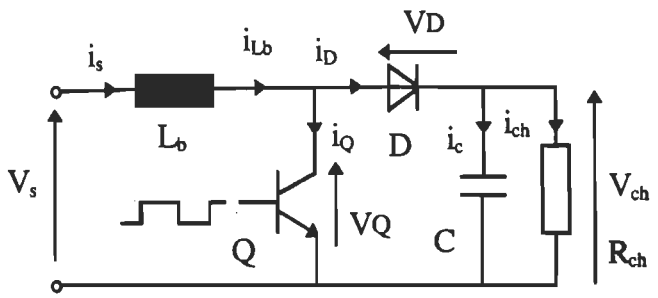


Figure 1.11: Montage du hacheur survolteur idéal

1.3.2 Principe de fonctionnement

Ce montage, comme son nom l'indique, permet d'avoir une tension supérieure ou égale à la tension de source. Les mêmes remarques que pour le hacheur abaisseur concernant les deux régimes de fonctionnement continu et discontinu s'appliquent [1-2]. L'allure des courbes de courants et tensions sont représentés dans la figure 1.15 pour les deux régimes.

1.3.2.1 Régime continu

Intervalle (0<t<DT):

Dans cet intervalle, le transistor Q est conducteur, par contre la diode D est bloquée (polarisée en inverse), la source V_s est en court-circuit à travers l'inductance L_b. Le courant augmente linéairement de la valeur minimale I_m à la valeur maximale

I_M en même temps que la capacité se décharge dans la résistance de charge R_{ch}. La figure 1.12 montre le circuit équivalent de cet intervalle.

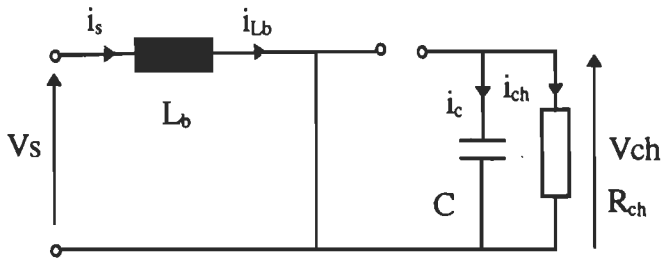


Figure 1.12: Circuit équivalent lorsque le transistor conduit

Cet intervalle est caractérisé par la charge de l'inductance L_b.

$$V_s = \frac{I_m - I_M}{t_{on}} L_b \qquad (1.10)$$

Intervalle ($DT<t<t_2$) :

Le circuit équivalent est montré sur la figure 1.13. Le transistor est bloqué et la diode sera en conduction car l'inductance inverse sa polarité et il y a transfert d'énergie stockée dans l'inductance vers la sortie (capacité C et la charge R_{ch}) via la diode jusqu'à la fin de cet intervalle. La sortie voit une source équivalente à $V_s + V_L$ qui est supérieure à la tension de source V_s, ce qui justifie la survoltage de la tension. Le courant décroît de I_M à I_m et le cycle recommence.

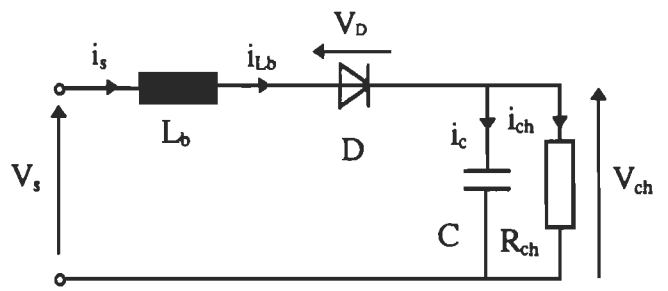


Figure 1.13: Circuit équivalent lorsque le transistor est bloqué

L'équation de tension de cet intervalle est la suivante :

$$V_s - V_{ch} = \frac{I_m - I_M}{toff} L_b \tag{1.11}$$

La caractéristique entrée-sortie est la suivante :

$$V_{ch} = \frac{V_s}{1-D} \tag{1.12}$$

où, D est le rapport cyclique

1.3.2.2 Régime discontinu

Ce régime est caractérisé par la décharge complète de l'inductance L_b avant la fin de la période T et les remarques du montage abaisseur sont valables pour ce régime. Dans les deux premiers intervalles, le fonctionnement est le même que le régime continu sauf que le courant I_m est nul.

L'équation de l'inductance critique L_c est la suivante [2]:

$$L_c = \frac{RD(1-D)^2}{2f_s} \tag{1.13}$$

Le rapport de conversion en boucle ouverte[2]:

$$\frac{V_{ch}}{V_s} = \frac{1+\sqrt{1+4DL_c/(1-D)^2}}{2} \tag{1.14}$$

Intervalle (t_2<t<T)

Le courant est nul dans les deux semi-conducteurs, le montage est équivalent à un circuit R-C et la charge R_{ch} n'est alimentée que par la capacité. Le circuit équivalent est montré sur la figure 1.14.

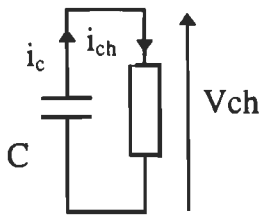


Figure 1.14: Circuit équivalent du survolteur pour l'intervalle $t_2<t<T$

Remarque

Une protection est nécessaire au niveau de la source pour protéger le montage contre les surcharges. Par contre le montage abaisseur peut être protégé par le transistor puisqu'il se trouve en série avec la source.

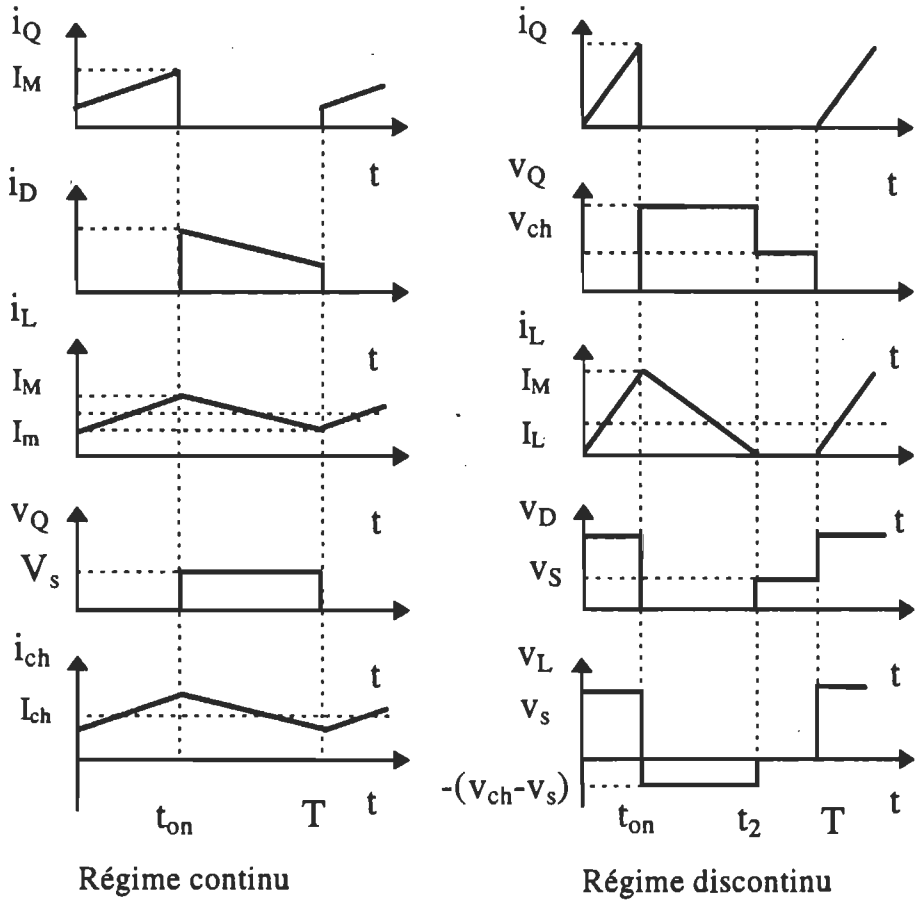


Figure 1.15: Formes d'ondes pour un survolteur en régime continu et discontinu

1.3.3 Simulation du montage

Les mêmes modèles et les mêmes paramètres sont utilisés pour les deux hacheurs sauf que la tension d'alimentation est fixée à 20V pour le hacheur survolteur. La figure 1.16 montre le schéma de ce montage.

Les équations de ce hacheur sont les suivantes :

$$V_Q = V_S - V_{Lb} \tag{1.15}$$

$$i_{Lb} = i_Q + i_D \tag{1.16}$$

$$V_D = V_Q - V_{ch} \tag{1.17}$$

$$i_C = i_D - \frac{V_{ch}}{R_{ch}} \tag{1.18}$$

1.3.3.1 Modèle du hacheur survolteur

La figure 1.17 montre le schéma bloc du montage survolteur. Les mêmes remarques sont appliquées pour le hacheur survolteur.

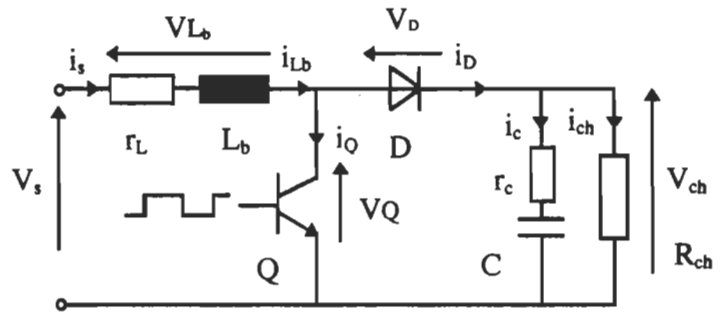


Figure 1.16: Schéma du montage survolteur utilisé en simulation

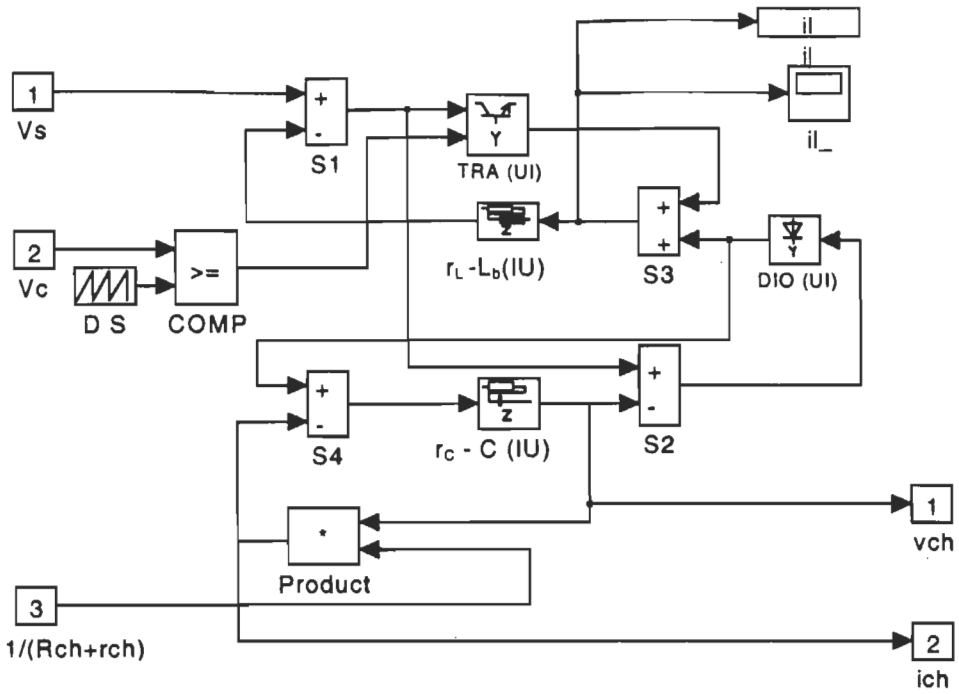


Figure 1.17:Schèma du modèle survolteur en bloc SIMULINK®

1.3.3.2 Résultats de simulation

Ce montage fonctionne en régime continu et la tension d'alimentation est de 20V nominale fixe. La figure 1.18 montre l'allure des courbes de courant dans l'inductance, la tension de charge ainsi que la tension de commande. Le courant dans l'inductance L_b est continu et la tension de charge est supérieure à celle de la source.

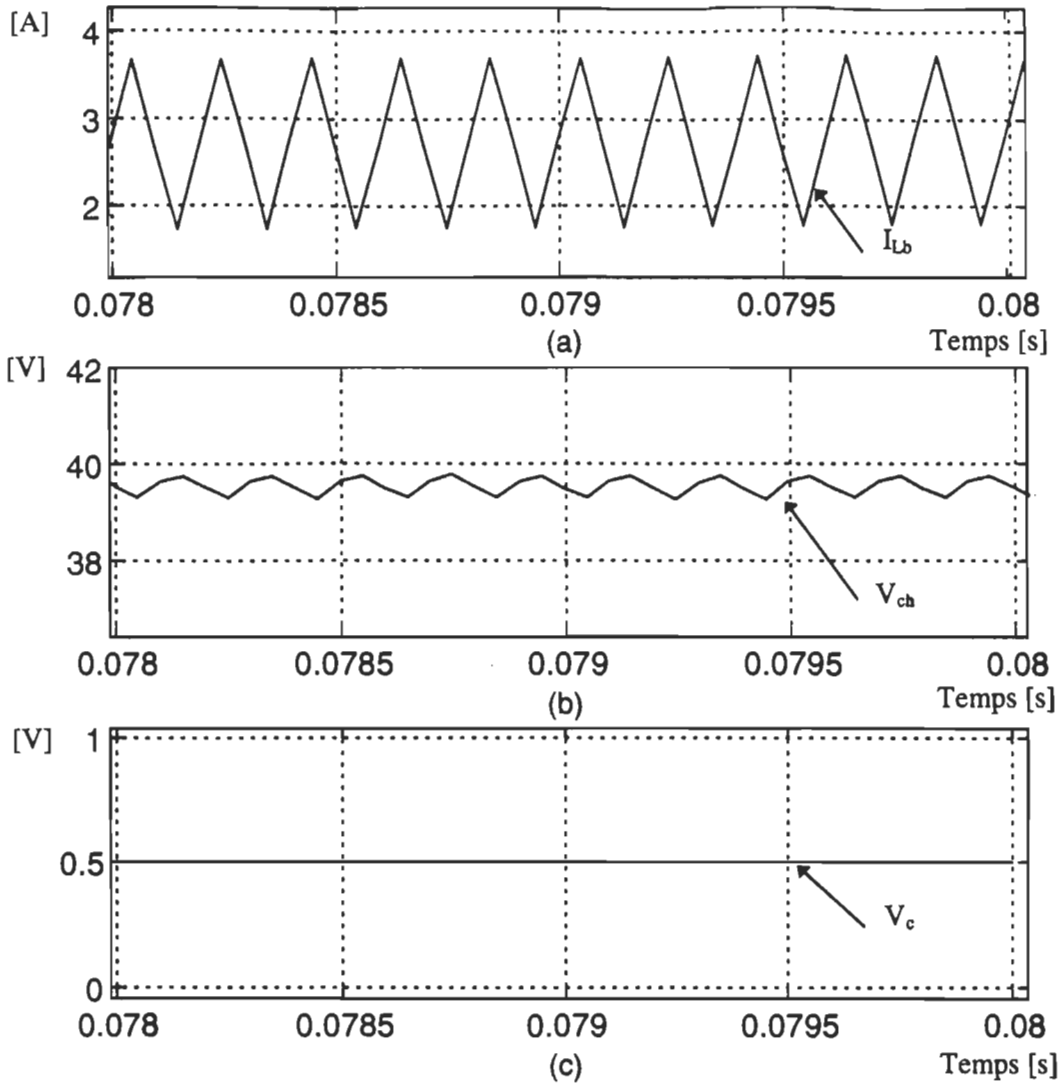


Figure 1.18 : Fonctionnement en régime continu du survolteur

a) courant dans L_b, b) tension de charge et c) tension de commande

1.4 Redresseur triphasé survolteur mono-interrupteur à facteur de puissance unitaire

Parmi les nombreuses causes de distorsion harmonique à travers le réseau électrique d'alimentation, les convertisseurs statiques de puissance tiennent une grande place, et parmi ces convertisseurs, on trouve les redresseurs à diodes. L'ajout d'un montage survolteur à la sortie permet d'améliorer le facteur de puissance (FP) et

la distorsion harmonique afin de rencontrer les normes IEEE 519-1992 et IEC-555 [4].

1.4.1 Schéma du montage

Ce montage est constitué d'un pont triphasé à diodes, d'un hacheur survolteur branché à la sortie du pont alimentant une charge résistive dont l'inductance est distribuée sur les trois phases à l'entrée du pont et d'un filtre L_f-C_f qui est inséré entre la source alternative et les inductances du survolteur. le filtre L_f-C_f sert à éliminer les harmoniques de haute fréquence car ce montage fonctionne en mode discontinu [4]. Le régulateur R(s) permet le réglage de la tension de sortie. Le schéma du montage survolteur est montré sur la figure 1.19.

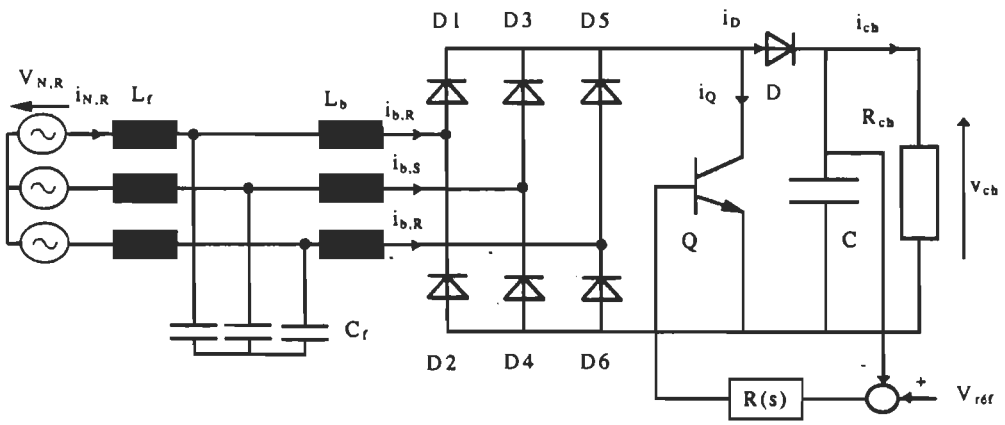


Figure 1.19: Schéma du redresseur triphasé survolteur à FP unitaire

1.4.2 Principe de fonctionnement

Pour avoir un facteur de puissance unitaire, il faut que le montage fonctionne en mode discontinu, c'est-à-dire à chaque période de hachage, les trois inductances du survolteur doivent être complètement déchargées (voir figure 1.20). Ce convertisseur fonctionne à une fréquence très élevée et la commande est autonome (sans synchronisation sur le réseau). La tension du réseau est supposée être constante

pendant la période de hachage et le fonctionnement idéal de ce montage est étudié sur un intervalle de $\pi/6$ avec $u_{N,R}>0$, $u_{N,T} \leq 0$ et $u_{N,S} \leq 0$ caractérisé par quatre intervalles [3-5].

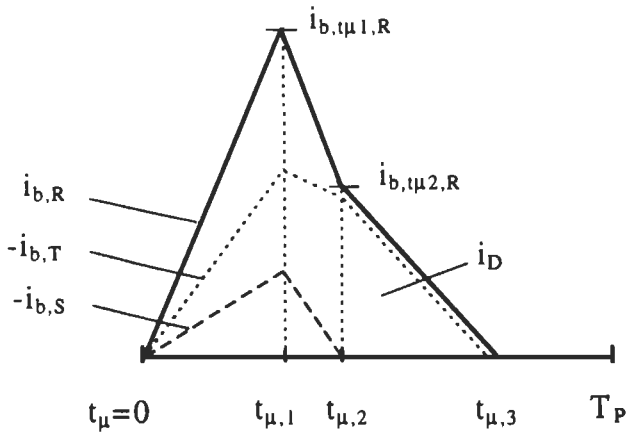


Figure 1.20: Forme du courant dans les inductances L_b sur une période T_p

Intervalle : $t_\mu \in [0,t_{\mu 1}]$

Lorsque le transistor Q est conducteur et la diode D est bloquée, le court-circuit du pont conduit à la charge des trois inductances L_b avec une pente constante U/L_b car la fréquence de commutation $f_p>>f_n$ (60Hz). Les diodes D1, D4 et D6 sont conductrices car elles sont polarisées en direct, et le condensateur C fournit le courant à la charge R_{ch} (voir figure 1.21).

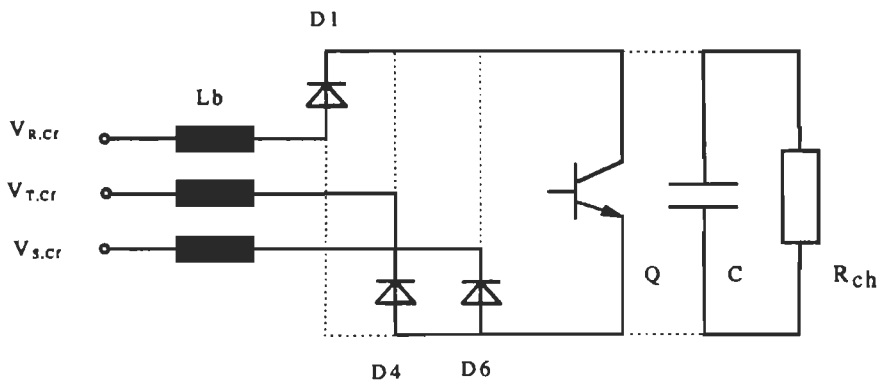


Figure 1.21: Séquence de fonctionnement pour $t_\mu \in [0,t_{\mu 1}]$

Intervalle: $t_\mu \in [\, t_{\mu 1}, t_{\mu 2}]$

Le transistor Q est bloqué, la diode D se met à conduire car elle est polarisée en direct. Les trois inductances L_b se déchargent dans le condensateur de sortie C et la résistance de la charge R_{ch}. Les diodes D1, D4 et D6 du pont restent en conduction puisque les inductances ne permettent pas des discontinuités du courant (voir la figure 1.22).

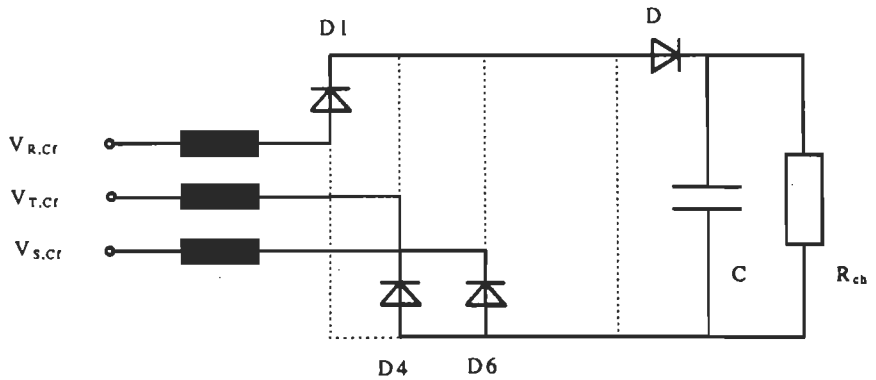


Figure 1.22: Séquence de fonctionnement pour $t_\mu \in [t_{\mu 1}, t_{\mu 2}]$

Intervalle : $t_\mu \in [\, t_{\mu 2}, t_{\mu 3}]$

Le transistor Q est toujours bloqué, l'inductance L_b de la phase T est complètement déchargée et la diode D4 sera bloquée à la fin de cet intervalle. La décharge se poursuit de la même façon que l'intervalle précédent (voir figure1.23).

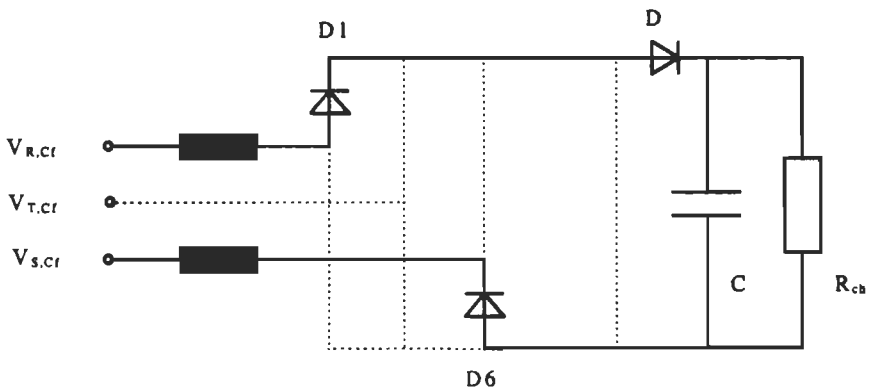


Figure 1.23: Séquence de fonctionnement pour $t_\mu \in [t_{\mu 2}, t_{\mu 3}]$

Intervalle : $t_\mu \in [\, t_{\mu 3}, T_P]$

Toutes les inductances sont déchargées et tous les semi-conducteurs sont bloqués (le transistor et les diodes), les courants à l'entrée du pont sont nuls et la capacité C de sortie fournit l'énergie à la résistance de charge R_{ch} (voir la figure 1.24).

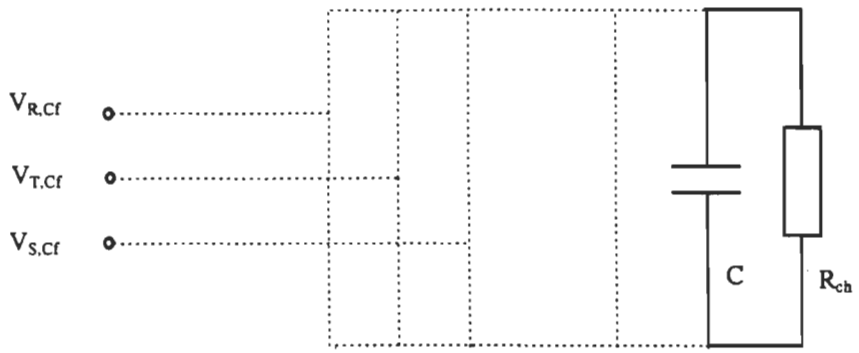


Figure 1.24: Séquence de fonctionnement pour $t\mu \in [\, t\mu 3, T_P]$

1.4.3 Simulation du montage

1.4.3.1 Modélisation du montage

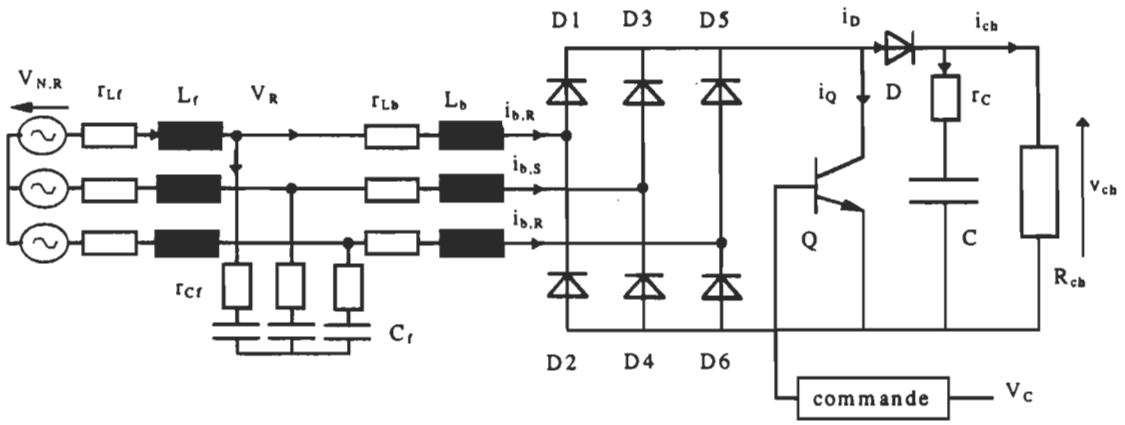


Figure 1.25: Schéma du montage redresseur utilisé en simulation

La figure 1.25 montre le montage à simuler qui est très proche de la réalité. Les pertes sont prises en considérations dans tous les éléments du montage et ce dernier est divisé en trois parties :

- La source, le filtre et l'inductance du survolteur
- Le pont de diode
- L'unité de commutation et la charge

On connecte ensemble les trois blocs pour donner le schéma bloc du montage présenté sur la figure 1.26.

La modélisation de la source, du filtre L_f-C_f et de l'inductance L_b, est basée sur les équations suivantes :

$$v_{N,R} = v_{Cf} + v_{Lf} \tag{1.19}$$

$$i_{Lf} = i_{Cf} + i_{N,R} \tag{1.20}$$

$$v_R = v_{Cf} - v_{Lb} \tag{1.21}$$

La figure 1.26 montre le schéma bloc de la source, du filtre et de l'inductance L_b en triphasé (pour les détails des blocs de la figure 1.26 se réfère à [3]).

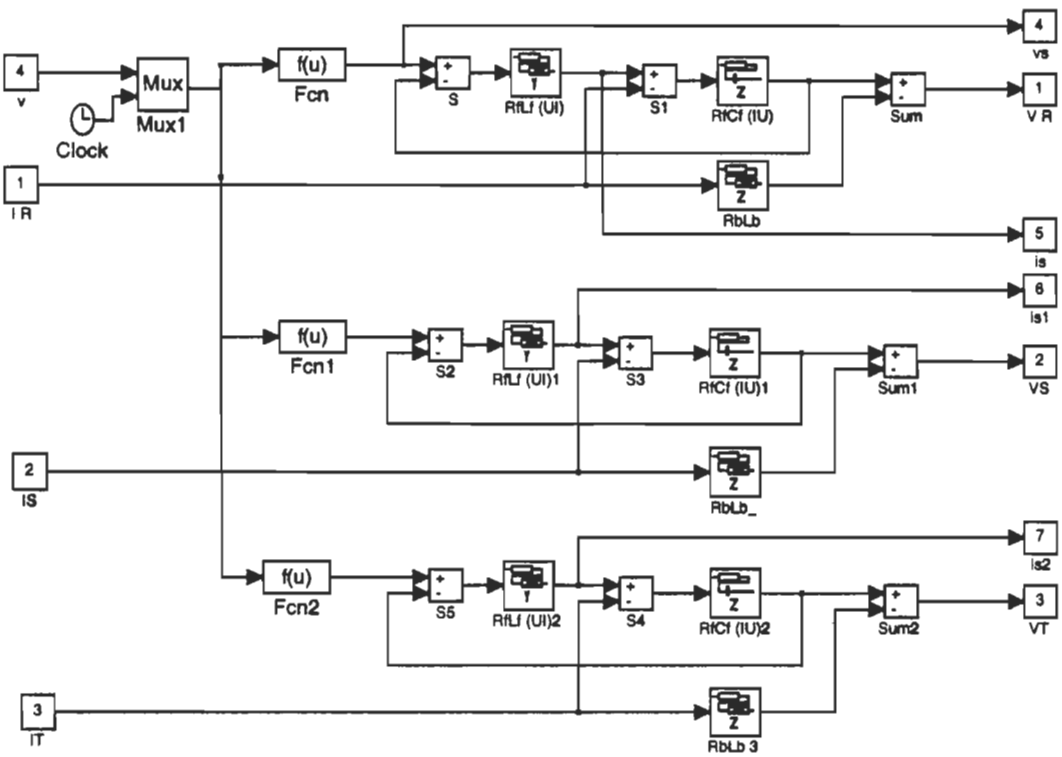


Figure 1.26: Schéma bloc de la source, du filtre et de l'inductance L_b

Nous avons utilisé le modèle du pont triphasé à diodes déjà existant dans la bibliothèque de SIMUSEC [3], enfin l'unité de commutation est modélisée de la même façon que le survolteur étudié dans le paragraphe précédent.

Pour le premier bloc (source + filtre + l'inductance):

* les courants de source triphasés et les tensions de source sont les entrées,
* les courants des trois inductances L_b sont les sorties de ce bloc.

Pour le second bloc soit celui du pont à diode:

* les courants des trois inductances L_b et le courant redressé sont les entrées,
* les courants alternatifs $i_{N,R}$, $i_{N,T}$ et $i_{N,S}$ et la tension redressée sont les sorties du pont à diode.

Pour le bloc de l'unité de commutation:

* la tension redressée du pont à diode, la commande et le courant de charge sont les entrées,
*la tension de charge et le courant redressé sont les sorties de ce bloc.

Après la réalisation de ces trois parties, nous avons connecté ces blocs entre eux en insérant les paramètres de chaque composant. La figure 1.27 montre le schéma du montage en bloc SIMULINK®. Les paramètres de ce redresseur sont donnés au tableau 1.2.

Tableau 1.2: Paramètres du redresseur

Tension de source, Vs	120 V +10% -15%
Tension de sortie Vch	500 V
Inductance, L_b	0.54 mH
Résistance interne, r_{Lb}	0.01 Ω
Capacité de sortie, C	250 µF
Résistance interne, r_c	0.005 Ω
Inductance, L_f	12 mH
Résistance interne, r_{Lf}	0.05 Ω
Capacité de filtre, C_f	130 µF
Résistance interne, r_{Cf}	0.001 Ω
Résistance de la charge, R	50 Ω
Fréquence de commutation fs	2 kHz
Puissance de sortie	5 kw

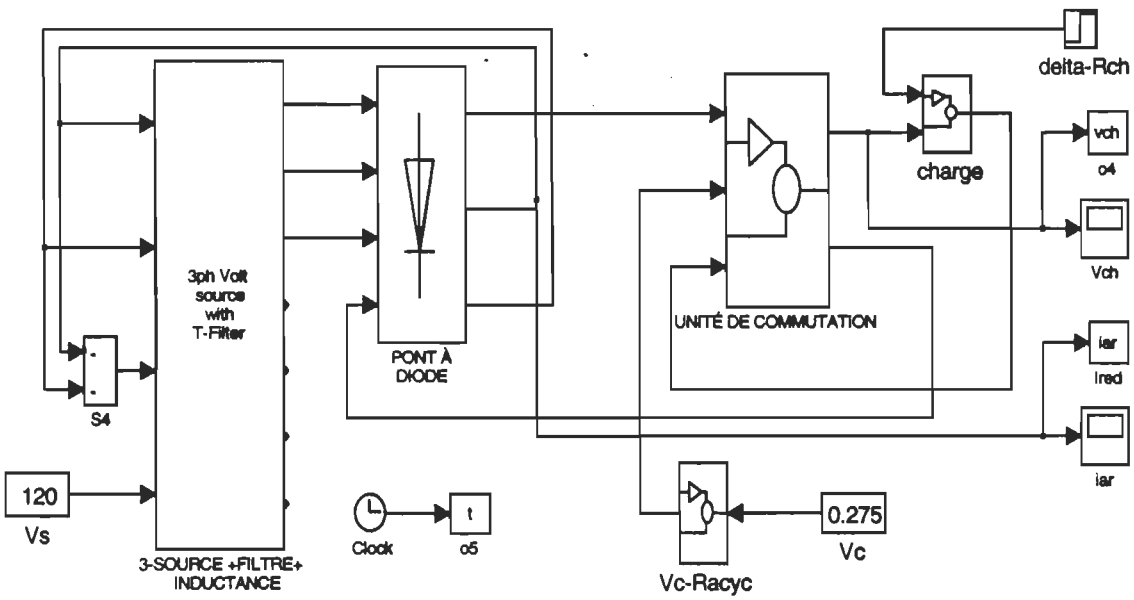


Figure 1.27: Schéma bloc du redresseur triphasé mono-interrupteur

1.4.3.2 Résultats de simulation

Après la simulation du montage, le fonctionnement est conforme à l'étude analytique. La figure 1.28 montre la forme des courants $i_{b,R}$ $-i_{b,T}$ et $-i_{b,S}$ dans les inductances L_b pour une période de commutation T_p. Les allures des courbes de courants dans les inductances L_b sont identiques à celles de la figure 1.20.

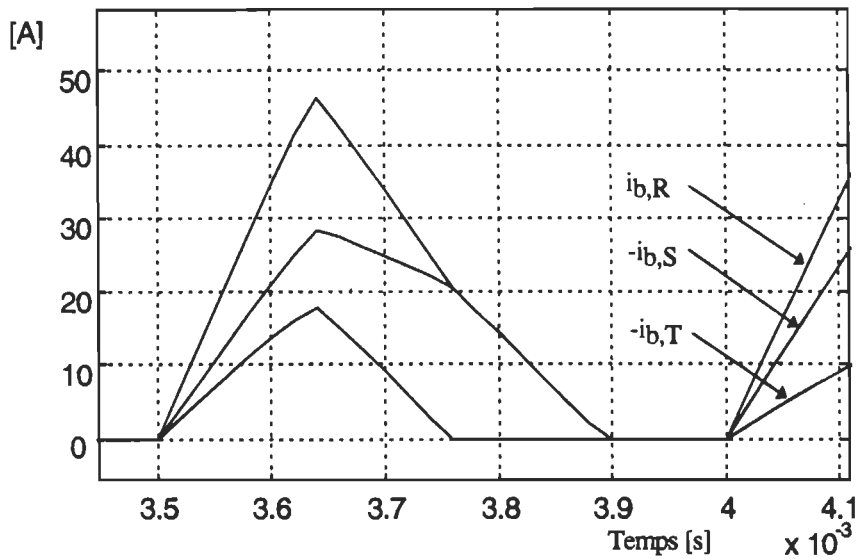


Figure 1.28: Forme des courants dans les inductances L_b

La figure 1.29 montre l'allure des courbes de la tension de charge, le courant dans une des inductances du survolteur ainsi que le courant et la tension de source (multipliée par un facteur d échelle de 0.25). L'analyse harmonique (voir figure 1.30) du courant de source au fonctionnement nominale montre bien que la distorsion totale des harmoniques est inférieure à 5% et le facteur de puissance est de 0.9992.

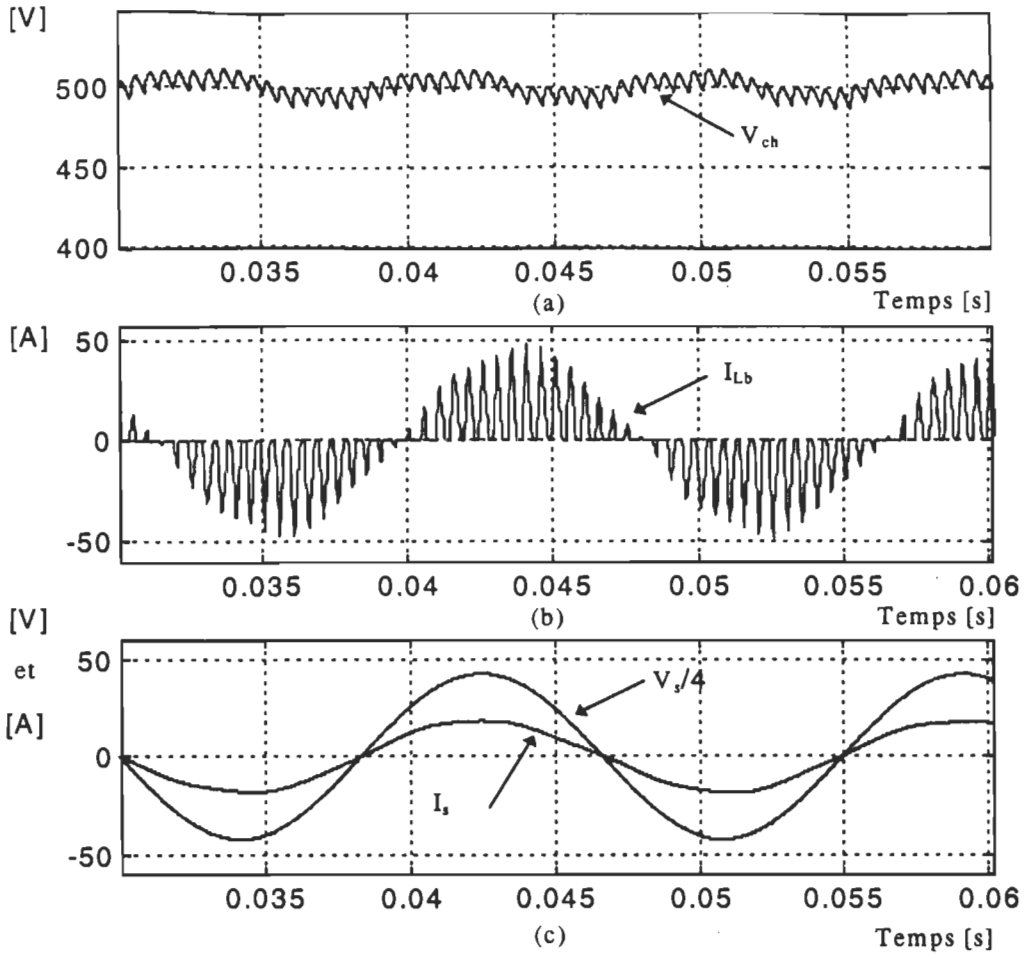


Figure 1.29: Tensions V_{ch} ,Vs/4 et courants i_{Lb} et i_s: a) tension de charge, b) courant dans L_b et c) courant et tension de source

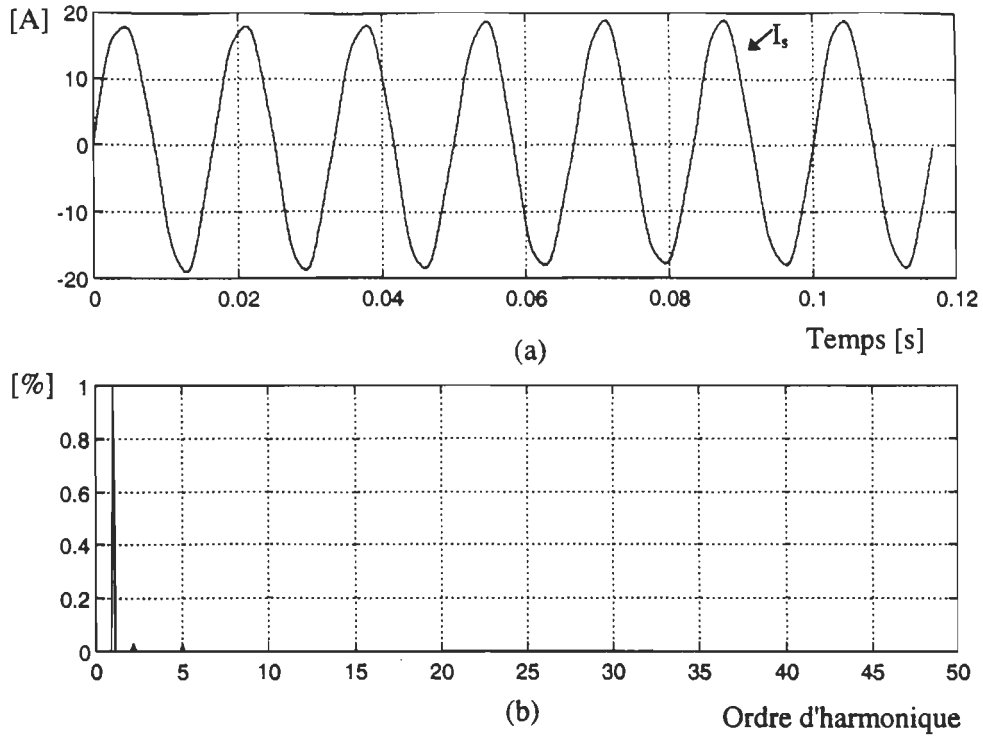


Figure 1.30: Analyse harmonique de courant de source I_s: (a) courant de source et (b) analyse harmonique relative au fondamental

Valeur moyenne du courant $i_s = 100\,\mu A$

Valeur efficace du courant i_s = 13.0840A

Valeur maximale du fondamental du courant i_s = 18.4896A

Valeur efficace du fondamental du courant i_s = 13.0741A

Facteur de distorsion du courant i_s = 0.9992

Facteur de déplacement du signal = 1.0000

Facteur de puissance du signal = 0.9992

Distorsion totale des harmoniques (THD) = 3.89%

1.5 Conclusion

Dans ce chapitre, nous avons fait une étude théorique des trois montages en montrant leur fonctionnement dans différents régimes. Les montages utilisés sont proches des montages réels en considérant toutes les pertes dans les composants passifs et actifs. Nous avons expliqué brièvement la mise en bloc de ces montages.

Les résultats de simulation montrent bien le bon fonctionnement des régimes continu et discontinu des deux convertisseurs CC-CC. Le troisième convertisseur CA-CC est caractérisé par un fonctionnement en régime discontinu qui assure un facteur de puissance unitaire et une distorsion totale des harmoniques inférieure à 5% aux conditions nominales.

CHAPITRE 2
RÉSEAUX DE NEURONES

2.1 Introduction

Les études des réseaux de neurones artificiels (RNA) remontent des années 1940. Grâce aux du développement des recherches sur le cerveau et la disponibilité des outils de simulation, Mc Culloch et Pitts [8] étudièrent un ensemble de neurones formels interconnectés. Ce réseau développé à l'époque permettait d'effectuer quelques opérations logiques simples. Jusqu'aux années 1980, la recherche était freinée par la limitation théorique du perceptron. Après cette époque, Hopfield [9] lança de nouveau la recherche dans ce domaine après avoir montré l'analogie entre les RNA et les systèmes physiques (année 82).

Les RNA sont des assemblages fortement connectés d'unités de calcul (neurone formel). Le neurone formel est une formulation mathématique simplifiée d'un neurone biologique. Les RNA ont la capacité de mémorisation, de généralisation et surtout d'apprentissage qui est le phénomène le plus important.

Après les années 1990, quelques articles ont vu le jour dans le domaine de l'électronique de puissance, parmi ces applications, la commande des machines électriques et des convertisseurs de puissance. Dans ce chapitre, on va présenter quelques généralités sur les réseaux de neurones artificiels.

2.2 Fondements biologiques

2.2.1 Le cerveau

Le système nerveux central est composé d'un ensemble d'éléments de base appelés neurones. Ces neurones sont fortement interconnectés entre eux et on dit que le cerveau en posséderait environ cent milliards. C'est le centre de régulation et de communication de l'organisme; nos pensées, nos actions, nos émotions attestent son

activité. Le système nerveux remplit trois fonctions étroitement liées: il reçoit de l'information sur le changement qui se produit tant à l'extérieur qu'à l'intérieur de l'organisme; il traite l'information et détermine l'action à entreprendre à tout moment; et enfin il fournit une réponse motrice (commande) qui active les muscles ou les glandes [7].

2.2.2 Le neurone

Le neurone est l'élément de base du cerveau (voir figure 2.1). Il possède des fonctions spécialisées, comme recevoir des signaux en provenance d'autres neurones, intégrer ces signaux, produire un influx nerveux, le conduire et le transmettre à une autre cellule capable de le recevoir.

2.2.3 Structure

Chaque neurone est constitué des éléments suivants: le corps cellulaire, les dendrites et l'axone. Le corps cellulaire contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la synthèse des enzymes et des autres molécules qui assurent la survie du neurone. Les dendrites sont des petites ramifications de quelques dixièmes de microns de diamètre et d'une longueur de quelque dizaines de microns. Enfin l'axone qui est la fibre nerveuse sert de moyen de transport pour les signaux émis par le neurone. Sa longueur varie d'un millimètre à plus d'un mètre, il se ramifie à son extrémité pour permettre la communication avec d'autres cellules.

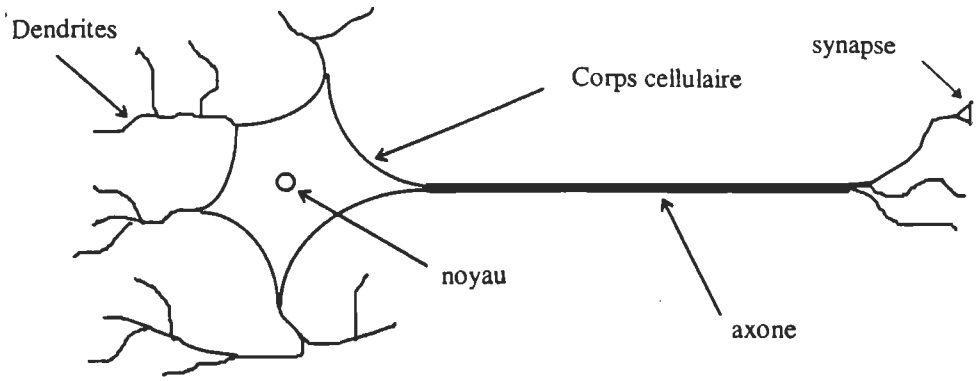


Figure 2.1:Un neurone biologique simple

La communication entre neurones est très complexe. Le neurone émet une impulsion nerveuse du corps cellulaire, c'est une décharge électrique qui prend chemin le long de l'axone vers les terminaisons axonales. Là, elle affectera tous les neurones reliés à ce neurone émetteur par l'intermédiaire de jonctions (appelées synapses) entre les terminaisons axonales et les dendrites des autres neurones. La synapse est le point de contact entre les neurones où le signal électrique est converti en un signal biologique. Une substance biologique appelée neurotransmetteur traverse la courte distance entre les deux cellules et excite les neurones qui la reçoivent. Les impulsions excitatrices convergent en un temps très court sur un même neurone, celui-ci produira en général à son tour une impulsion nerveuse et ainsi de suite. L'effet inverse existe, un neurone peu inhibé un autre neurone pour réduire sa tendance à produire des impulsions nerveuses [8-9].

2.3 Le neurone formel

2.3.1 Modèle de Mac Culloch et Pitts

Le première modélisation de neurone formel date des années quarante. Elle a été présentée par Mc Culloch et Pitts [8], c'est un automate à seuil dont l'unité de sortie est binaire, la fonction d'activation utilisée est une fonction seuil. Ce neurone est

actif si la sortie est à 1 et inactif si la sortie est à 0. Ce modèle ne peut résoudre que des problèmes simples tels que les fonctions logiques ET, OU, NON etc.

2.3.2 Le modèle généralisé

Le neurone formel est la cellule fondamentale d'un réseau de neurones artificiels. Par analogie avec le neurone biologique, le neurone formel doit être apte à accomplir les trois tâches suivantes: collecter, traiter les données qui viennent des neurones émetteurs et transmettre les messages aux autres neurones. La figure 2.2 présente un neurone formel simple.

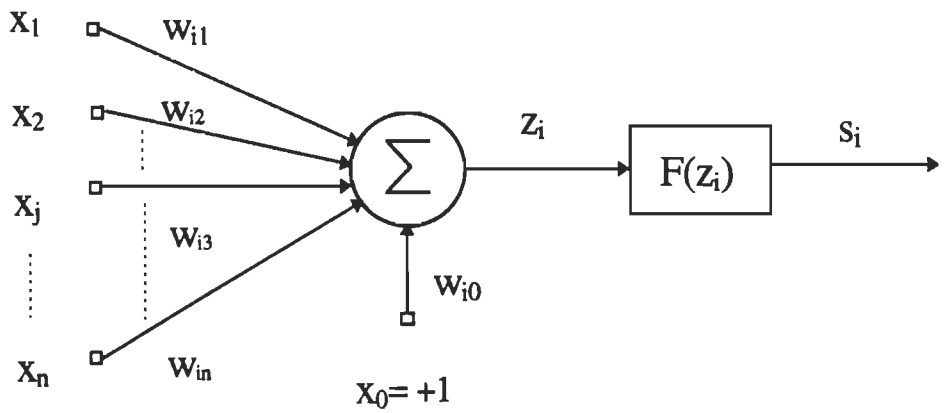


Figure 2.2: Formalisation mathématique d'un neurone

Le neurone formel possède:

- plusieurs entrées pour collecter les messages provenant des autres neurones formels, le signal x_j provenant du neurone j connecté au neurone i doit être pondéré par le poids W_{ij} (l'indice i réfère le neurone récepteur et l'indice j réfère au neurone émetteur). Si le produit du poids et de l'entrée est positif, le neurone sera excitateur, et si le produit du poids et de l'entrée est négatif, le neurone sera

inhibiteur. W_{i0} est appelé biais et sert à ajuster la fonction d'activation, le signal d'entrée correspondant est x_0 et est égal à +1.

- la sommation des signaux constitue une combinaison linéaire des signaux d'entrées xj pondérés par les poids W_{ij}.
- une fonction mathématique limite l'amplitude de la sortie du neurone et est appelée fonction d'activation. Cette dernière limite la sortie en un intervalle fini de valeurs, en général $[0,1]$ ou $[-1,1]$.

La formulation mathématique d'un neurone peut alors s'écrire comme suit [9]:

$$y_i = f(z_i) = f\left(\sum_{j=1}^{n} W_{ij} x_j - b_i\right) \qquad (2.1)$$

où

x_j entrée élémentaire du neurone i,

z_i entrée totale du neurone i,

y_i sortie du neurone i,

Wij poids de connexion reliant le neurone i à l'entrée x_j,

b_i seuil du neurone i, qui est égale $-W_{i0}$,

f(x) fonction de décision non linéaire, dite d'activation,

n nombre de neurones j connectés au neurone i.

2.3.3 Fonction d'activation

Les fonctions d'activation utilisées dans les modèles connexionnistes d'aujourd'hui sont variées. On peut identifier quatre principaux types de fonctions les plus connues [10-11,15]: binaire à seuil, rampe avec saturation, sigmoïde et gaussienne.

Fonction binaire à seuil

$$f(x)=\begin{cases}0 & si \le 0\\ 1 & si \ge 0\end{cases} \qquad (2.2)$$

Cette fonction d'activation a été utilisée par McCulloch et Pitts, le seuil introduit une non linéarité dans le comportement du neurone, c'est le modèle tout ou rien (voir figure 2.3a).

Fonction à rampe avec saturation

$$f(x)=\begin{cases}-1 & si \quad x \le -1\\ x & si \quad |x|<1\\ +1 & si \quad x \ge 1\end{cases} \qquad (2.3)$$

Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil: entre ses deux bornes, elle confère au neurone une combinaison linéaire de l'entrée. A la limite, la fonction linéaire est équivalente à la fonction seuil (voir figure 2.3b).

Fonction sigmoïde

$$f(x)=\frac{1}{1+e^{-\alpha x}} \qquad (2.4)$$

où $\alpha > 0$, en général $\alpha = 1$, qui maintient la sortie dans l'intervalle [0,1].

La fonction sigmoïde est une fonction continue qui maintient la sortie dans l'intervalle [0,1]. Son avantage principal est l'existence de sa dérivée en tout point. Elle est employée en général dans le perceptron multicouches(voir figure 2.3c).

Fonction Gaussienne

$$f_i(x)=e^{\frac{-d_i(\chi)^2}{\sigma_i^2}} \qquad (2.5)$$

avec $d_i(\chi)=\|x-c_i\|$

c_i: est la position du centre de la fonction appelé noyau,

$d_i(\chi)$: la distance entre le vecteur x et le centre c_i de la fonction -noyau,

σ_i : la taille du champ récepteur.

Elle est souvent employée dans les réseaux RBF ("Radial Basis Function") où chaque neurone est conçu pour résoudre de préférence un ensemble de valeurs (voir figure 2.3d).

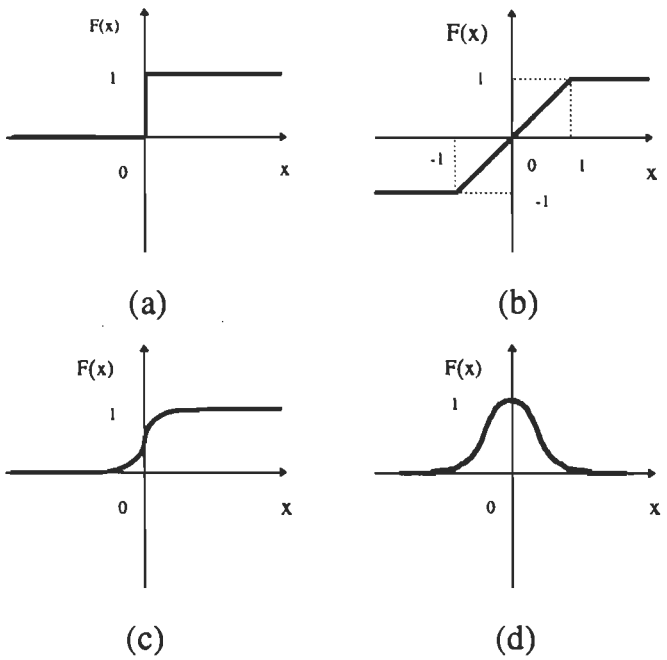


Figure 2.3: Fonction d'activation binaire avec seuil (a), rampe avec saturation (b), sigmoïde (c), gaussiénne (d).

2.4 Architectures des réseaux de neurones

L'architecture des réseaux de neurones artificiels peut aller d'une connectivité totale, où tous les neurones sont reliés les uns aux autres, à une connectivité locale, où les neurones ne sont reliés qu'à leurs plus proches voisins. Il y a plusieurs classifications des réseaux de neurones suivant le nombre de couches, le mode de connexion ou le type de couches. Principalement, nous pouvons identifier quatre classes d'architectures de réseaux [10] et [12].

2.4.1 Réseau à couche simple

Le réseau à couche est un réseau dont les neurones sont organisés en couches, la forme la plus simple est un réseau à une couche. Tous les signaux d'entrées sont propagés des noeuds d'entrées vers la couche de neurones de sortie.

Le nombre de neurones d'entrée (noeuds) et de neurones de sortie est en général lié au problème à résoudre. Les entrées seront propagées à travers la matrice des poids W pour ensuite obtenir la réponse de sortie (voir la figure 2.4).

$$W = \begin{bmatrix} W(1,1) & W(1,2) & . & . & W(1,R) \\ W(2,1) & W(2,2) & . & . & W(2,R) \\ . & . & . & . & . \\ . & . & . & . & . \\ W(S,1) & W(S,2) & . & . & W(S,R) \end{bmatrix} \qquad b = \begin{bmatrix} b(1) \\ b(2) \\ . \\ . \\ b(S) \end{bmatrix} \tag{2.6}$$

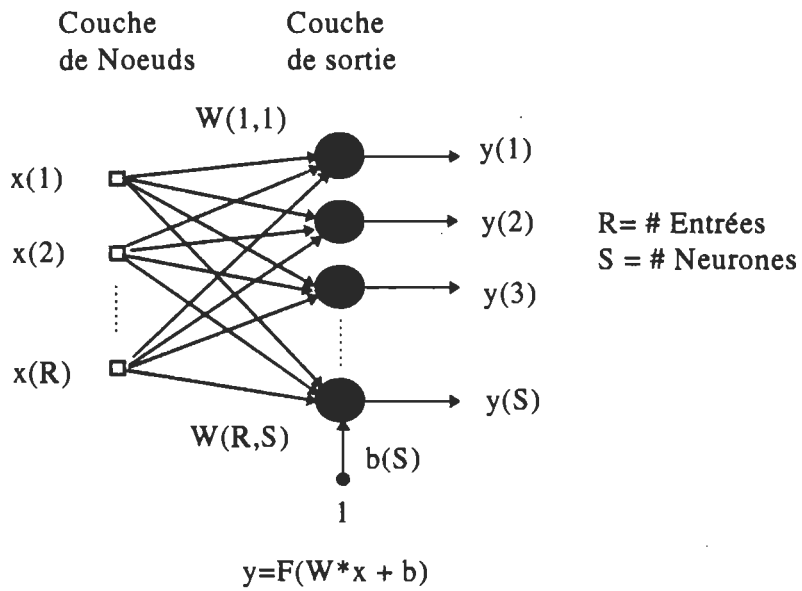


Figure 2.4: Réseau à couche simple

2.4.2 Réseaux multicouches

La seconde classe est caractérisée par une ou plusieurs couches. À chaque couche correspond une matrice de poids **W** et un vecteur de biais **b**, et on peut avoir aussi des fonctions d'activation différentes pour chaque couche. Ces réseaux comportent une ou plusieurs couches cachées et une couche de sortie. La fonction des couches cachées intervient entre les entrées **x** et la couche de sortie **y**. Elle permet de résoudre des problèmes plus complexes que le réseau à couche simple.

La figure 2.5 montre un réseau à deux couches, une cachée avec S1 neurones et une couche de sortie avec S2 neurones. La connexion entre les neurones peut être totale ou partielle (d'une couche à la couche suivante).

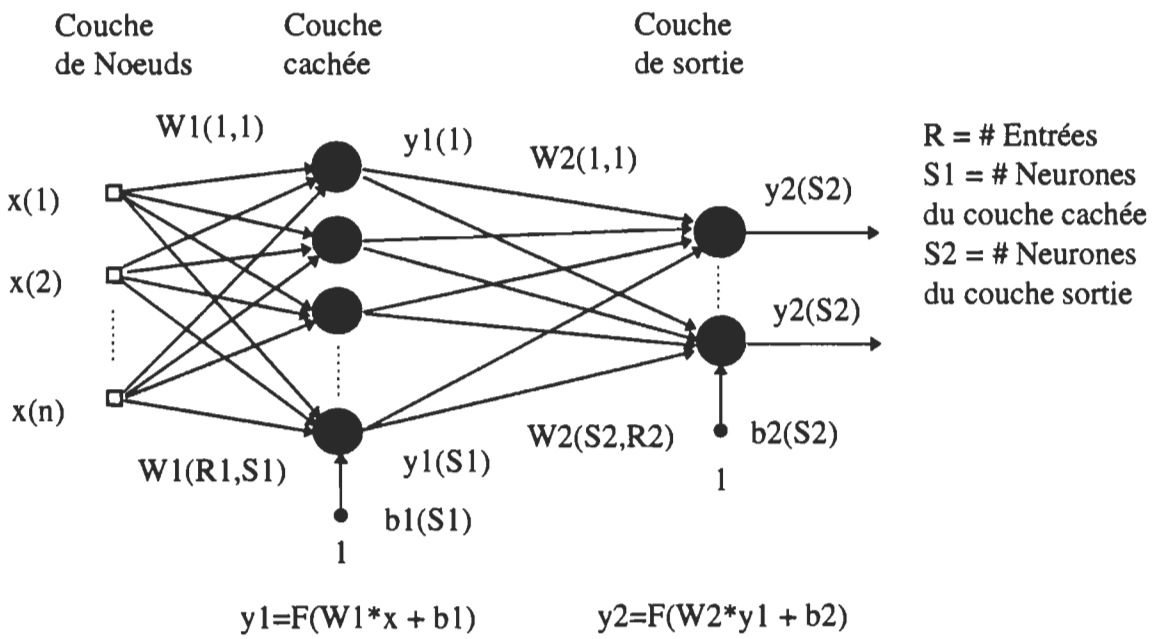


Figure 2.5: Réseau à deux couches (une couche cachée)

2.4.3 Réseaux récurrents

Nous appelons réseaux récurrents les réseaux pouvant comporter des boucles. Ces réseaux se distinguent des autres réseaux par la connexion des sorties de neurones avec leurs entrées. La sortie d'un neurone peut être connectée avec l'entrée du même neurone ou avec les entrées des autres neurones. L'importance de ces réseaux est qu'ils permettent d'apprendre la dynamique des systèmes, c'est-à-dire qu'ils peuvent imiter le comportement temporel en insérant des délais dans les boucles, ces dernières sont entre l'entrée et la sortie du réseau ou dans des couches internes. La figure 2.6 illustre un réseau récurrent dont les sorties sont bouclées avec les entrées de tous les neurones.

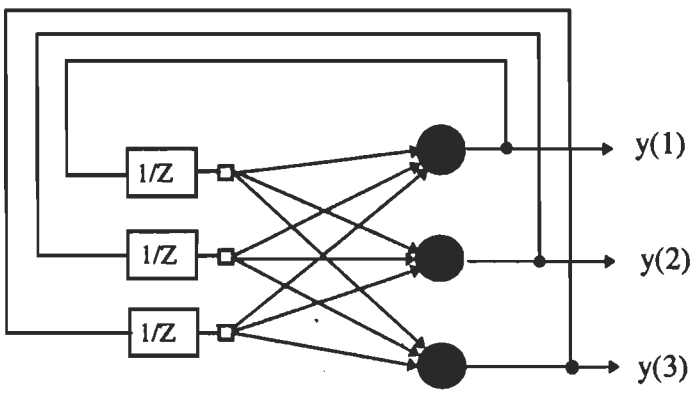


Figure 2.6: Réseau récurrent

2.4.4 Structure "Lattice"

C'est un réseau dont les neurones sont organisés en lignes et en colonnes, toutes les entrées de neurones sont reliées aux noeuds d'entrées. Ces réseaux peuvent être d'une, deux ou plusieurs dimensions, la figure 2.7 montre un réseau de lattice à deux dimensions, de 3 par 2.

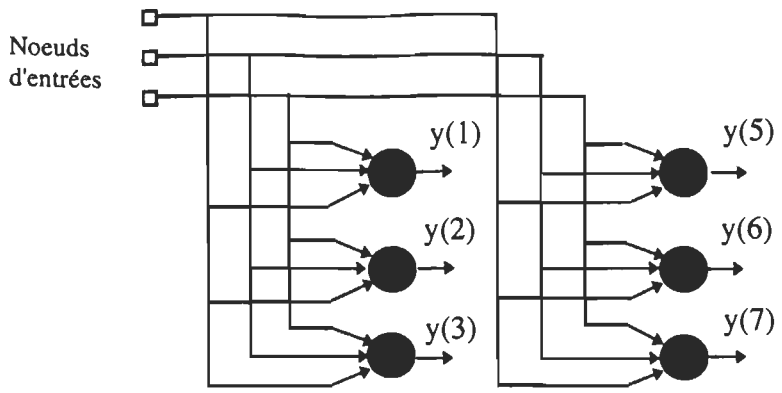


Figure 2.7: "Lattice" à deux dimensions (3 neurones par 2)

2.5 Apprentissage de réseaux de neurones

On peut définir l'apprentissage comme la capacité d'emmagasiner des informations (acquisition des propriétés) qui peuvent être rappelées par la suite. Les connaissances d'un réseau connexionniste sont mémorisées dans les poids de connexions qui seront déterminés lors de l'apprentissage. Le but de l'apprentissage pour un réseau est de trouver un ensemble de poids synaptiques qui minimisent l'erreur entre la sortie du réseau et le résultat désiré. C'est la caractéristique principale des réseaux de neurones [8], [10] et [12].

2.5.1 Modes d'apprentissages

Les modes d'apprentissages connexionnistes sont:

- Supervisé: des associations entrée/sortie sont présentées au réseau par un professeur, c'est-à-dire que l'apprentissage dispose d'un comportement de référence vers lequel il tente de faire converger le réseau.
- Non supervisé: cet apprentissage concerne les réseaux ne disposant que des informations d'entrées et qui les regroupent en fonction d'un critère interne dans le but de créer des classes.

- Par coeur: l'information se mémorise parfaitement et définitivement. Dans ce cas l'erreur est quasi-nulle entre la sortie du réseau et la sortie désirée.
- En ligne: le réseau est en phase d'apprentissage continuelle, à chaque nouvelle action le réseau apprend et s'ajuste en réalisant la tâche qui lui est assignée.
- Hors ligne: le réseau est entraîné durant la phase d'apprentissage à l'aide d'un ensemble de données qui permettent de fixer les poids de connexions. Une fois les poids calculés, le réseau est fonctionnel et il peut mener la tâche qui lui a été assignée sans modifications des poids.

2.5.2 Procédure d'apprentissage

En général, l'apprentissage se fait sur une période relativement longue, durant laquelle les données d'entrées et les données de sorties peuvent être présentées au réseau un grand nombre de fois. Du point de vue calcul, l'objectif de l'apprentissage est de minimiser l'erreur quadratique globale [9].

$$E = \frac{1}{2}\sum_{k} e_k^2 \tag{2.7}$$

E: l'erreur quadratique globale

e_k: l'erreur entre la valeur désirée et la sortie x

k: nombres d'itération.

À chaque itération, il faut trouver un Δw_{kj} pour ajuster de nouveau les poids.

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k) \tag{2.8}$$

Il y a quatre étapes dans la procédure d'apprentissage.

1 - Initialisation des poids du réseau;

2 - Présentation des données et propagation d'activation;

3 - Calcul d'erreur

4 - Calcul du vecteur de corrélation.

Les étapes 2-3-4 sont répétées jusqu'à la fin de l'apprentissage (voir figure 2.8).

En général, on divise les données d'entrées en deux parties distribuées aléatoirement. Une banque d'apprentissage avec laquelle on entraîne le réseau et une banque de généralisation pour le test du réseau à la fin de l'apprentissage. L'arrêt du processus d'apprentissage s'effectue lorsque les performances en test sont optimales, c'est-à-dire avant que le réseau rentre dans l'apprentissage par coeur [9].

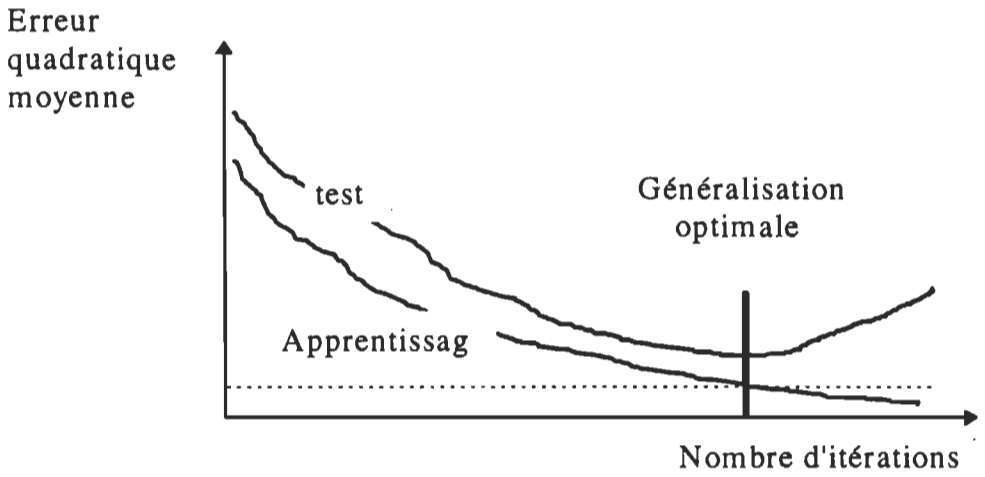


Figure 2.8: Évolution typiques de l'erreur quadratique moyenne sur les données d'apprentissage et de test

2.5.3 Problèmes d'apprentissage

Malgré la diversité des algorithmes d'apprentissage il n'y a pas de méthodes systématiques du choix des paramètres d'apprentissage, tel que le choix de l'architecture du réseau, le nombre de neurones, le nombre de couches, le choix des paramètres internes de l'algorithme comme l'erreur quadratique et le nombre d'itérations. Nous pouvons citer quelques problèmes concernant l'apprentissage [9] et [12]:

- **Insuffisance de la règle d'apprentissage:** En principe, les réseaux de neurones sont capables de calculs élaborés, en revanche rien ne garantit que la règle d'apprentissage soit capable de tirer profit du plein potentiel du réseau.
- **Minima locaux:** Ce problème est rencontré lorsque l'apprentissage converge vers une solution sous optimale. Ce type de problème est difficile à résoudre car généralement la surface d'erreur est inconnue.
- **Choix du nombre de neurones:** Le nombre de neurones cachés est particulièrement important parce qu'il détermine la capacité de calcul du réseau. Un nombre insuffisant de neurones cachés peut compromettre la capacité du réseau à résoudre le problème. Inversement, trop de neurones permettent au réseau d'apprendre par coeur au détriment des performances de généralisation. Même les paramètres propres à l'algorithme d'apprentissage sont difficiles à choisir comme par exemple le pas d'apprentissage.
- **Surapprentissage:** C'est lorsque l'apprentissage d'un réseau reflète trop les particularités du problème au détriment de la tâche réelle.
- **Mauvais échantillonnage:** La matrice d'apprentissage ne reflète pas toujours adéquatement la tâche. Il en résulte que le réseau généralise mal.

2.6 Algorithme de rétropropagation

Cet algorithme que l'on désigne couramment par "Back-propagation" est le plus populaire parmi les deux techniques d'apprentissage des réseaux multicouches. La deuxième technique est l'apprentissage de la machine de Boltzman. L'algorithme de rétropropagation est basé sur la généralisation de la règle de Widrow-Hoff [8] en utilisant une fonction d'activation sigmoïde. Le réseau utilisé est un réseau à couches tel que défini à la section 2.3 où chaque neurone est connecté à l'ensemble des neurones de la couche suivante. Le principe de cet algorithme est la propagation d'un

signal provenant des noeuds d'entrées vers la sortie et ensuite on propage l'erreur commise de la sortie vers les couches internes jusqu'à l'entrée [8] et [12].

2.6.1 Formulation

Pour un exemple de données à apprendre, on note **x** le vecteur d'entrées et $\mathbf{y_d}$ le vecteur de sortie désirée. Supposant que notre réseau a **n** noeuds d'entrée et m neurones de sortie, on a donc:

$$x = (x_1, x_2, \ldots x_n) \text{ et } y_d = (y_{d1}, y_{d2}, \ldots y_{dm}) \tag{2.9}$$

On note $y = (y_1, y_2, \ldots y_m)$ le vecteur des sorties obtenues effectivement à l'issue de la propagation avant du vecteur **x**. On cherche à minimiser l'erreur quadratique entre les sorties désirées et les sorties obtenues, cette erreur étant considérée comme une fonction des poids des connexions:

$$E(W) = \sum_{i=1,m} (y_{di} - y_i)^2 \tag{2.10}$$

La règle de modification des poids à la présentation numéro k de l'exemple **x** est:

$$W_{ij}(k) = W_{ij}(k-1) - e(k).d_i.o_j \tag{2.11}$$

où d_i est calculé de proche en proche de la couche de sortie à la couche d'entrée:

$$d_i = 2.(y_i - y_{di}).f'(l_i) \text{ pour la couche de sortie} \tag{2.12}$$

$$d_i = \sum_h y_h.W_{hi}.f'(l_i) \text{ pour les couches cachées} \tag{2.13}$$

où *h* parcourt les neurones vers lesquels le neurone i envoie une connexion

f: est la fonction sigmoïde d'un neurone, *f'* est sa dérivée,

o_j: est la sortie du neurone j,

l_i: est l'entrée du neurone i, $l_i = \sum_j W_{ij}.o_j$

e(k): est le pas du gradient à l'étape k.

2.6.2 Algorithme de Levenberg - Marquardt

Nous retrouvons trois variantes de l'algorithme de rétropropagation disponibles dans le toolbox des réseaux de neurones de MATLAB®: Backpropagation standard et Faster Backpropagationsont deux algorithmes basés sur la descente du gradient, et l'algorithme de Levenberg - Marquardt basé sur l'approximation de Newton. Cette troisième technique est plus puissante que la descente du gradient .

L'algorithme de Levenberg - Marquardt converge plus rapidement que les autres algorithmes sités, mais il demande plus de mémoire lorsque le réseau devient grand. La règle de correction des poids est la suivante [14] et [16].

$$\Delta W = \left(J^T J + \mu I\right)^{-1} J^T e \qquad (2.15)$$

J: la matrice jacobienne des dérivées de l'erreur,

μ: un scalaire,

e: le vecteur d'erreur.

Lorsque μ est grand, l'algorithme se rapproche de la méthode de descente du gradient. Par contre si μ est petit l'algorithme se rapproche de la méthode de Gauss-Newton. La procédure de l'algorithme est la suivante:

1- Présentation du vecteur d'entrée en propageant celui-ci jusqu'à la sortie et calcul de l'erreur quadratique,
2- calcul de la matrice jacobienne,
3- calcul de ΔW pour corriger les poids du réseau,
4- calcul des nouveaux poids, ajustement du paramètre μ et vérification de la convergence vers l'erreur demandée.

Cet algorithme converge plus rapidement par rapport aux autres algorithmes lorsque le réseau de neurones contient quelques centaines de poids. Son inconvénient majeur est qu'il demande beaucoup de mémoire de la part des calculateurs.

2.7 Différentes classes des réseaux de neurones

Chaque architecture définit la forme externe du réseau et contient plusieurs classes, ces derniers spécifient une applications données.

2.7.1 Perceptron

Le perceptron est la forme la plus simple d'un réseau de neurones, il modélise la perception visuelle. Il comprend trois principaux éléments: la rétine, les cellules d'association et les cellules de décision. La fonction d'activation utilisée dans ce réseau est de type tout ou rien (0 ou 1). L'apprentissage du perceptron va se faire suivant la règle de Hebb, il n'y a qu'une seule couche de poids modifiables entre les cellules d'association et les cellules de décision. Le perceptron est limité dans ses applications; premièrement, il ne peut être applicable que dans la classification dont les variables sont linéairement séparables, deuxièmement la sortie ne peut être que 0 ou 1 [8].

2.7.2 Réseau linéaire

Cette classe de réseaux diffère du perceptron, car elle possède un neurone dont la fonction d'activation est linéaire. La règle d'apprentissage utilisée est nommée règle de Widrow-hoff ou delta qui permet d'effectuer une descente de gradient de l'erreur sur une mesure d'erreur quadratique. Ces techniques d'estimation sont dites des moindres carrés "Least Mean Square". Ce type de réseau est appelé Adaline "ADAptive LINear Element". Les domaines d'application comprennent le traitement de signal, le contrôle et le radar. L'avantage de ce réseau est qu'il converge sur un seul minimum si la solution existe, sinon l'ajout de couches n'a aucun effet. Parmi ces inconvénients, il est limité à une couche de sortie et ne peut résoudre que les problèmes dont la relation entrées - sorties est linéaire [11-12,14].

2.7.3 Perceptron multicouches

Cette classe est la plus importante des réseaux de neurones, car elle représente la généralisation du perceptron monocouche avec une fonction d'activation de type sigmoïde et une ou plusieurs couches cachées. Le vecteur d'entrée se propage dans le réseau de couche en couche jusqu'à la sortie, l'entraînement de celui-ci se fait avec l'algorithme le plus populaire, soit l'algorithme de rétro-propagation de l'erreur "error Back-propagation algorithm". Ce réseau est caractérisé par

1- le modèle du neurone incluant la non linéarité à la sortie (fonction sigmoïde).
2- le réseau comporte une ou plusieurs couches cachées.
3- le plus grand nombre de connections permettant de résoudre la majorité des problèmes.

Même avec les avantages des couches cachées et la performance de l'algorithme d'apprentissage, il reste plusieurs problèmes non réglés comme le choix du nombre de couches, le nombre de neurones par couche et le problème des minimums locaux où le réseau peut converger [14].

2.7.4 Réseau RBF

Les réseaux RBF ("Radial Basis Function") sont des réseaux à couches qui ont comme origine une technique d'interpolation nommée méthode d'interpolation RBF. Ce réseau comporte une seule couche cachée dont la fonction d'activation est appelée fonction-noyau ou gaussienne et une couche de sortie avec une fonction d'activation linéaire. La méthode RBF est particulière par ses réponses utiles pour un domaine de valeurs restreint. La réponse de la fonction-noyau est maximale au noyau et décroît généralement de façon monotone avec la distance qui existe entre le vecteur d'entrée et le centre de la fonction-noyau. Afin d'approximer un comportement donné, les fonctions-noyau sont assemblées pour couvrir par leurs champs récepteurs l'ensemble des données d'entrée. Ces fonctions sont ensuite

pondérées et la somme de leurs valeurs est calculée pour produire la valeur de sortie. Les réseaux RBF sont capables de calculs puissants. L'apprentissage est plus rapide et plus simple mais demande beaucoup de neurones par rapport aux réseaux multicouches. De plus, ils s'avèrent davantage insensibles à la destruction de leurs poids. Le domaine d'application est vaste, comme le traitement d'image, la reconnaissance de forme, les radars et surtout dans les problèmes de classification [11-12,14].

2.7.5 Réseaux récurrents

Cette classe de réseaux se caractérise de trois façons: la non linéarité des unités élémentaires (neurones) du réseau, la symétrie synaptique des connexions et enfin l'utilisation abondante des boucles dans le réseau. Il y a deux importants réseaux récurrents:

1- Réseau de Hopfield: ce réseau est basé sur les principes de la physique statistique qui est fondamentalement une mémoire adressable par son contenu. Le neurone est basé sur le modèle de Mc Culloch et Pitts tous interconnectés et la règle d'apprentissage utilisée est la règle de Hebb. Plusieurs domaines d'application sont possibles, en particulier: les mémoires associatives et l'économie. La principale limitation est qu'il n'y a pas de couches cachées.

2- Réseau de Elman: ce réseau a deux couches; la première est bouclée (la sortie est reliée à l'entrée à travers un délai). La fonction d'activation est de type tangente-sigmoïde pour la première et linéaire pour la couche de sortie. Ce réseau peut résoudre la plupart des phénomènes temporaires. Son domaine d'application constitue la détection et l'identification [14].

2.7.6 Réseaux compétitifs

Ces réseaux utilisent la compétition comme mode d'interaction prédominant, ce phénomène de compétition est le contraire de la coopération. Les neurones sont reliés par des liens inhibiteurs, c'est-à-dire que leur activité est antagoniste, ceci permet notamment de filtrer l'entrée d'informations parasites, ce qui facilite leur traitement. Il y a trois importants types de réseaux compétitifs: le LVQ ("Learning Vector Quantization"), le SOM de Kohonen ("Self-Organizing Map") et Le réseau ART "Adaptive Resonance Theory". Son domaine d'application est la classification [14].

2.8 Application des réseaux de neurones dans l'électronique de puissance

Dans le domaine de l'électronique de puissance, les montages génèrent des formes d'ondes de tensions et courants complexes à cause de plusieurs facteurs tels que le mode de fonctionnement, la fréquence du réseau, la fréquence de commutation de la commande, les types de charges à alimenter, etc. Ces montages sont fortement non linéaires. Ceci poussa plusieurs chercheurs à trouver quelques applications des réseaux de neurones pour contrôler et estimer les paramètres électriques des convertisseurs et les paramètres mécaniques des machines électriques de l'électronique de puissance.

Les premières applications se trouvaient dans la commande des machines électriques, comme le réglage de la vitesse, du couple et la commande vectorielle. Ensuite, quelques applications ont vu le jour dans le contrôle des formes d'ondes comme le courant de source, la tension de sortie des onduleurs et la correction du facteur de puissance, ainsi que l'estimation des formes d'ondes pour le diagnostic de ces montages [17-27].

2.8.1 Régulateur statique

Le régulateur statique est un modèle inverse statique qui ne prend pas en considération la dynamique du système. Le réseau de neurones établira une relation entre les paramètres d'entrées et de sorties pour simuler l'inverse du système réel, afin de permettre de linéariser le système. On obtiendra l'identité ($X= A*A^{-1}X$) formée par le système inverse et le système réel à contrôler. La banque d'apprentissage doit comprendre toute la plage de fonctionnement du système à commander afin de permettre une bonne généralisation et un bon fonctionnement du réglage du système.

2.8.2 Régulateur dynamique

Ce type de régulateur est plus performant que le modèle inverse statique, car il permet une amélioration dans la réponse dynamique du système. Il existe plusieurs méthodes proposées dans la littérature comme le contrôle direct, indirect, adaptatif, etc. [19-22]. Deux méthodes sont utilisées pour concevoir un régulateur à réseau de neurones dynamiques, la méthode directe et la méthode indirecte, cette dernière est utilisée, elle comprend deux étapes: la première est de concevoir un modèle à l'aide d'un réseau de neurones du système réel. La seconde est l'entraînement de régulateur avec ce dernier. Pour l'apprentissage de la dynamique, il n'y a pas de méthodes systématiques pour déterminer le nombre de retards ou les paramètres à boucles, soit pour le modèle ou le régulateur.

2.8.3 Commande des machines électriques

À cause de la non linéarité des machines électriques et de leur constante de temps qui est relativement plus grande que celle des convertisseurs, les premières applications des réseaux de neurones ont vu le jour dans ce domaine.

Pour que le fonctionnement d'un système d'entraînement à vitesse variable soit robuste et stable (stabilité mécanique et électrique), la machine doit fonctionner dans le régime linéaire (sans saturation du flux). L'avantage des réseaux de neurones est de permettre le réglage de la vitesse même s'il y a une variation des paramètres non accessibles de la machine comme la variation de la résistance interne [21-23]. En outre, des applications de type commande vectorielle des machines à induction sont sur le point de se développer.

2.9 Conclusion

Ce chapitre a porté sur les fondements des réseaux de neurones artificiels en commençant par la formulation mathématique d'un neurone. Ensuite on a mentionné à propos de ces réseaux de neurones leur principe biologique, l'architecture, leurs classes, l'apprentissage et les problèmes liés à leur utilisation ont été soulignées. Enfin quelques applications qui ont vu le jour en électronique de puissance. Il apparaît que jusqu'à aujourd'hui, l'entraînement des réseaux de neurones demeure encore plutôt un art qu'une science. Ceci constituera la base de nos applications de ces RNA dans l'électronique de puissance.

CHAPITRE 3
RÉGULATEUR STATIQUE

3.1 Introduction

Les convertisseurs de l'électronique de puissance sont des systèmes non linéaires dus à la commutation et au mode de fonctionnement, ce qui pose des difficultés à déterminer les fonctions de transfert. Les réseaux de neurones présentent des caractéristiques intéressantes pour solutionner des problèmes complexes qui sont difficiles à résoudre par les méthodes classiques. Les RNA ne demandent aucun calcul ni équations préalables d'un système à régler. Seules les données d'entrées et de sorties du système sont nécessaires, ce qui nous amène à introduire l'utilisation des RNA dans ce domaine. Ces réseaux sont, en effet, capables de faire l'interpolation et la mémorisation des données apprises. Dans ce chapitre nous allons montrer quelques applications de la commande par le modèle inverse.

3.2 Objectifs

Les convertisseurs sont soumis à plusieurs perturbations telles que la variation de charge et la tension d'alimentation. Le bon fonctionnement de ces montages demande un réglage de la tension de charge et la protection contre les surcharges. L'objectif de ce travail consiste à concevoir un régulateur à base des RNA pour ces deux applications. Le régulateur statique sera appliqué pour les trois convertisseurs étudiés dans le chapitre 1.

3.3 Régulateur statique

Les réseaux de neurones sont constitués d'un grand nombre d'unités de traitement travaillant en parallèle. Ils établissent une fonction qui peut être non linéaire entre un domaine d'entrée et un domaine de sortie par l'apprentissage des données

représentatives. On définit le modèle inverse en réseau de neurones d'un système comme une boite noire dont les paramètres entrées de celle-ci correspondent aux sorties du système et vice versa. Les perturbations doivent être comme des entrées pour le modèle inverse. Une linéarisation du système est possible et nous permet d'obtenir un régulateur statique. Ce type de régulateur ne prend pas en considération le temps. En effet toutes les valeurs d'apprentissage sont prises en régime permanent. Les convertisseurs de l'électronique de puissance étudiés ont comme paramètres d'entrées la tension de commande et comme paramètres de sortie le courant ou la tension.

Dans notre cas, c'est le réglage de la tension de charge qui nous intéresse. Le modèle inverse a comme paramètres d'entrées la tension de référence et les perturbations et comme paramètre de sortie la tension de commande. La figure 3.1 montre le principe du modèle inverse qui sera appliqué aux trois montages vus au chapitre 1.

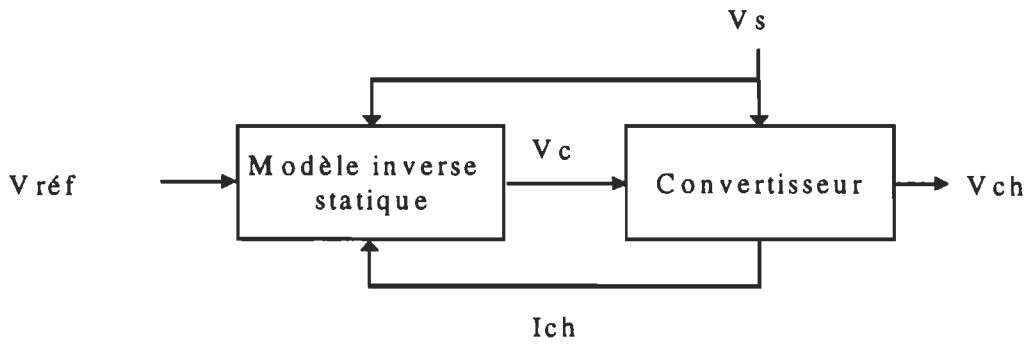


Figure 3.1: Schéma de réglage par modèle inverse

3.3.1 Choix du réseau de neurones

La notion des réseaux à couches s'applique à la majorité des types de réseaux connus aujourd'hui. La plupart des réseaux proposés plus récemment sont structurés en couches comme le perceptron, l'Adaline, etc. Le développement du perceptron multicouches marqua un grand tournant dans la recherche connexioniste. C'est un

réseau qui contient au moins une couche cachée et il est capable de faire l'approximation de toute fonction continue. Le modèle inverse utilisé pour contrôler les trois convertisseurs n'a pas besoin de comportement dans le temps car l'apprentissage se fait par des données en régime permanent [9] et [12]. Tous ces facteurs nous ramènent à choisir le réseau perceptron multicouche. La figure 3.2 montre le schéma d'un réseau de neurones en bloc SIMULINK®.

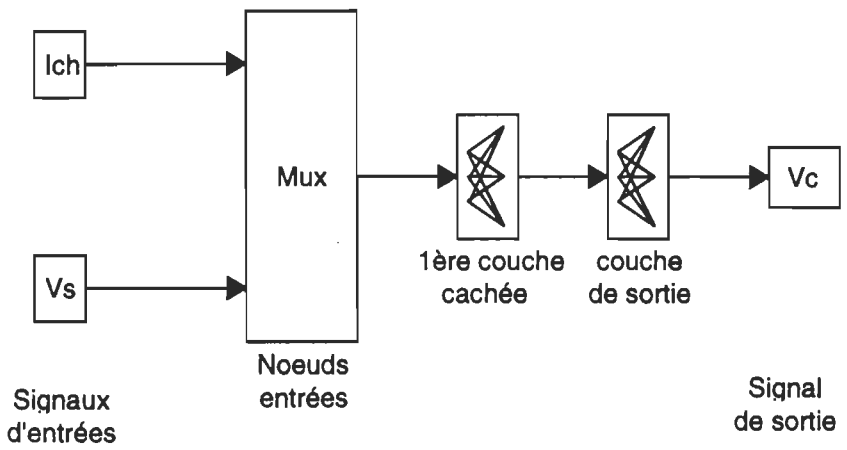


Figure 3.2: Réseau de neurones en bloc SIMULINK®

3.3.2 Apprentissage du réseau

L'apprentissage utilisé dans cette application est appelé apprentissage supervisé. On présente au réseau un vecteur d'entrée, et un vecteur de sortie désiré vers lequel l'apprentissage tente de faire converger le réseau.

3.3.2.1 Préparation des données

La première étape importante est la préparation des données d'entrée et de sortie désirées. La banque d'apprentissage doit refléter adéquatement la tâche à accomplir pour éviter un mauvais échantillonnage et ne doit pas être petite car elle peut causer une mauvaise généralisation. Dans notre cas, les données d'entraînement doivent

comprendre le fonctionnement non linéaire dû aux régimes continu et discontinu pour les deux hacheurs et seulement le régime discontinu pour le redresseur triphasé. La méthode de préparation des données d'entraînement est la suivante :

1- Réaliser plusieurs simulations en variant la tension de commande, la tension de source et la charge sur plusieurs échelons de façon à obtenir un régime permanent entre deux échelons.
2- Une fois le régime permanent atteint, on sauvegarde les données du courant, de la tension de charge et de la tension de source.
3- Créer une matrice de données qui sera divisée en deux matrices appelées banque d'apprentissage et banque de généralisation.
4- Choisir les données pour former la matrice de généralisation de façon à pouvoir valider le réseau obtenu après apprentissage. La taille est de 10 à 20% de la banque d'apprentissage.

3.3.2.2 Choix du nombre de neurones

Il n'y a pas de règle systématique pour le choix de nombre de neurones des couches cachées. La méthode suivie est de commencer par un petit nombre et de l'augmenter progressivement jusqu'à l'obtention de l'erreur désirée. Pour les neurones d'entrées (noeuds) et les neurones de sorties, le nombre est fixé par les données du système à régler.

3.2.2.3 Choix des paramètres de l'algorithme d'apprentissage

L'algorithme choisi parmi les trois, disponible dans "Toolbox Neural network" de MATLAB® est l'algorithme de Levenberg-Marquardt. Ce dernier converge plus rapidement que "Backpropagation standard" et "Faster Backpropagation". On économise le temps d'apprentissage surtout lorsque le réseau comporte quelques dizaines de poids. Il y a huit paramètres à choisir pour cet algorithme. En général,

on choisit que les trois premiers qui sont le pas d'affichage, le nombre d'itérations et l'erreur quadratique globale en prenant le reste des paramètres par défaut [14].

3.3.2.4 Procédure d'apprentissage

Dans ce paragraphe, on va donner la structure d'un programme d'apprentissage supervisé disponible dans l'environnement MATLAB® :

- chargement des données d'apprentissage,
- normalisation des données par rapport à la valeur maximale,
- choix du nombre de neurones,
- initialisation des poids de façon aléatoire,
- choix des paramètres de l'algorithme d'apprentissage Trainlm [14],
- appel de l'algorithme Trainlm [14],
- test avec les valeurs apprises et non-apprises et
- sauvegarde de la matrice des poids.

Si l'erreur globale est inférieure à un seuil, on aura l'arrêt de l'apprentissage sinon on continu l'apprentissage en prenant les poids de la dernière itération. L'initialisation des poids est effectuée une seule fois au début de l'apprentissage.

Après avoir présenter la méthodologie suivie pour l'application du modèle inverse statique. Une application d'un réglage de la tension de charge V_{ch} est faite pour les trois convertisseurs cités au chapitre 1.

3.4 Hacheur abaisseur

Pour ce montage, deux objectifs sont visés, soit la régulation de la tension de charge contre la perturbation de la source, et la protection du montage contre les surcharges. La charge est fixée à 15Ω. La régulation de la tension de sortie est indispensable à tout montage soit à faible puissance (alimentations à découpage) ou à grande puissance (commande des machines).

La limitation du courant est nécessaire pour protéger le montage contre l'échauffement ou la destruction des composants comme par exemple le transistor et la diode de puissance qui sont les plus sensibles à la surcharge. Pour cela il faut ajouter des données de protection aux données d'apprentissage pour permettre au réseau de neurones d'apprendre la limitation de courant.

3.4.1 Préparation des données

Données d'apprentissage pour le réglage: la préparation des données d'apprentissage par simulation demande beaucoup de temps, car on a besoin de faire plusieurs simulations pour avoir les combinaisons entres les paramètres d'entrées du régulateur.

Pour ce montage, la perturbation étudiée concerne la tension de source V_s dont la valeur est de 24V et qui varie de -15% à +10%, par pas de 1V. La tension de commande V_c varie de 40% à 65% et sa valeur maximale est de 1V. Pour chaque tension de source on fait varier la tension de commande d'un pas de 5%. Le nombre total de simulations à effectuer est de 48.

V_s = [20 21 22 23 24 25 26 27]

V_c = [0.4 0.45 0.5 0.55 0.6 0.65]

Données d'apprentissage pour la protection: On ajoute aux données de réglage les données de protection. La limitation du courant de charge est fixée à 1A. Cette dernière est imposée de manière à ce que quelque soit la tension de charge ou de source, le courant I_{ch} ne dépasse pas 1A. À chaque fois que le courant dépasse cette limite on impose la valeur zéro à la tension de commande V_c. La banque de données contient le vecteur courant de charge égale à 1A, V_c égale à zéro et une combinaison de la tension de charge V_{ch} et de source V_s.

Les données d'apprentissage sont formées de la tension de charge, du courant de charge, de la tension de source et de la tension de commande [V_{ch} I_{ch} V_s V_c]. Ces données comportent les données de réglage et de protection. On les divise en deux parties, une d'apprentissage et une autre de généralisation.

*** Banque d'apprentissage:**

Les paramètres d'entrées du réseau sont données par la matrice : P=[V_{ch} V_s I_{ch}]

Le vecteur de sortie désirée est donné par ce vecteur T=[V_c]

$$P=\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\begin{matrix} V_{ch} & V_s & I_{ch} \end{matrix}\begin{bmatrix} 7.96 & 20 & 0.531 \\ - & - & - \\ 17.28 & 27 & 0.97 \\ 10.5 & 20 & 1 \\ - & - & - \\ 15.5 & 27 & 1 \end{bmatrix} \qquad T=\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\begin{matrix} V_c \end{matrix}\begin{bmatrix} 0.4 \\ - \\ 0.65 \\ 0 \\ - \\ 0 \end{bmatrix} \tag{3.1}$$

*** Banque de généralisation:**

Les paramètres d'entrés du réseau sont PG=[V_{ch} V_s I_{ch}]

Le vecteur de sortie désiré TG=[V_c]

Les Banques PG et TG sont formées de la même manière que P et T.

3.4.2 Apprentissage du réseau

Le réseau de neurones est un réseau 3-15-1 : 3 noeuds, 15 neurones en couche cachée et un neurone de sortie. L'erreur quadratique globale est fixée à 10^{-7}. Après plusieurs simulations l'algorithme converge jusqu'à cette erreur, ce qui donne l'arrêt de l'apprentissage.

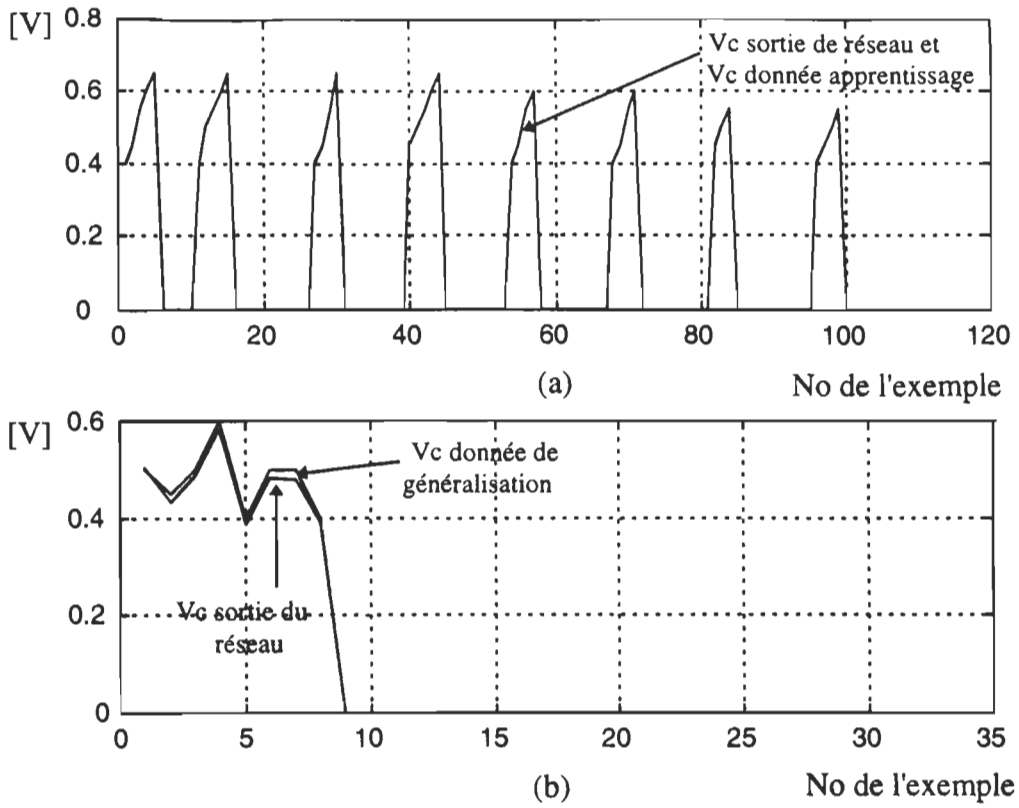


Figure 3.3: Test d'apprentissage: a)données apprises et b) données non-apprises

La figure 3.3 montre le test d'apprentissage des données apprises et non-apprises. Les deux courbes sur le graphique (a) montrent que le réseau a bien appris les données d'apprentissage. Le graphique (b) montre le résultat de généralisation pour les données non-apprises. Ceci montre que le réseau après apprentissage a le pouvoir de généraliser même pour des données non-apprises.

3.4.3 Simulation du montage abaisseur

La simulation du montage commandé par le régulateur statique a été réalisée dans l'environnement SIMULINK®. Les données du courant I_{ch} et les tensions V_s et $V_{réf}$ sont normalisées de la même façon comme pour l'apprentissage. L'ajout d'un échantillonneur bloqueur permet de bloquer les données pour un temps égal à la période de commutation ce qui correspond à une situation réelle lors de la mise en

oeuvre dans un processeur numérique. La figure 3.4 montre la commande d'un abaisseur régulé par réseau de neurones statique.

Régulation de la tension de charge:

Premièrement, on désire réguler de la tension de charge à 12V en régime discontinu. Après simulation, le courant dans l'inductance est discontinu et la tension

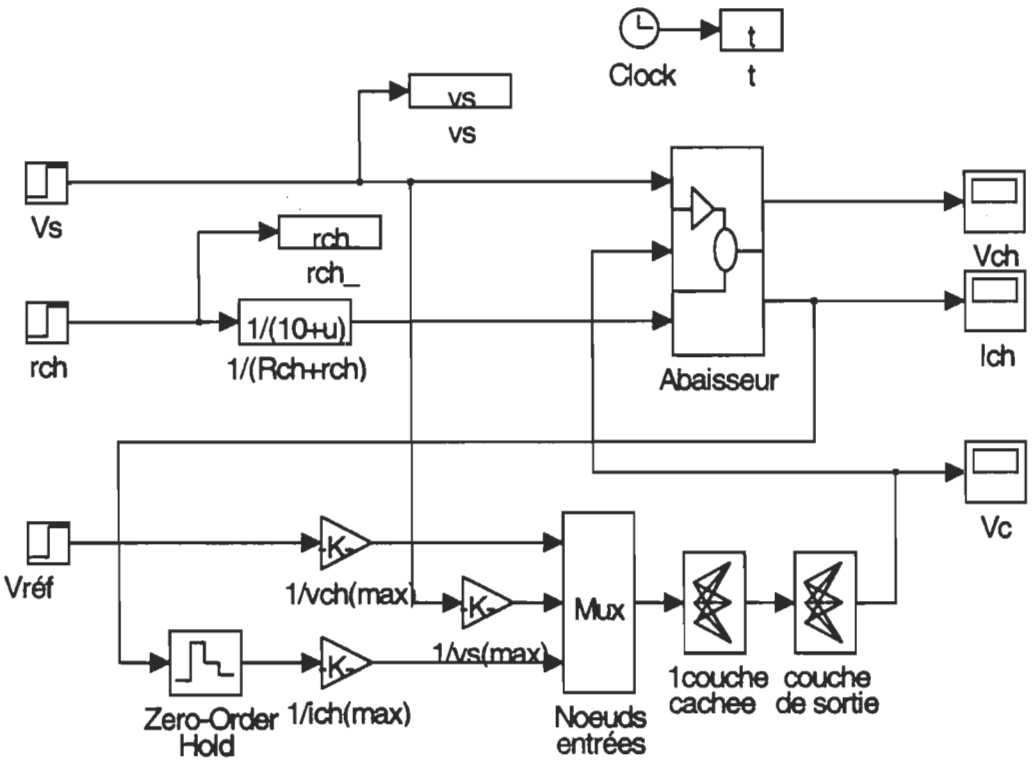


Figure 3.4: Schéma de commande de l'abaisseur par régulateur statique

de charge est légèrement supérieure à 12V car, il s'avère moins précis de prendre les valeurs du régime discontinu, car la qualité du régulateur dépend de la qualité des données d'apprentissage. La figure 3.5 montre la courbe du courant dans l'inductance et la tension de charge à la sortie de l'abaisseur.

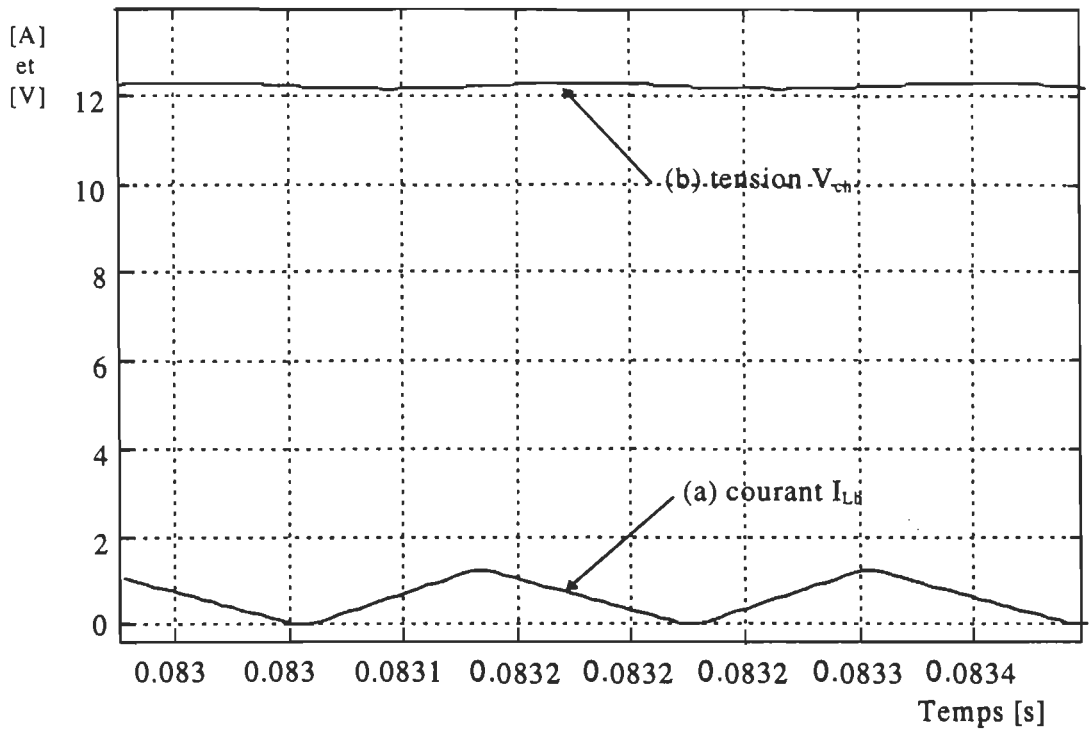


Figure 3.5: Fonctionnement en régime discontinu de l'abaisseur

a) courant dans l'inductance L_b, b) la tension de charge

La figure 3.6 montre une perturbation due à la tension de source V_s et la tension de charge V_{ch} reste stable à 12V avec de petites oscillations amorties. Le modèle inverse répond pratiquement à la vitesse d'une période d'échantillonnage et donne une tension de commande convenable pour garder la tension de charge à la consigne.

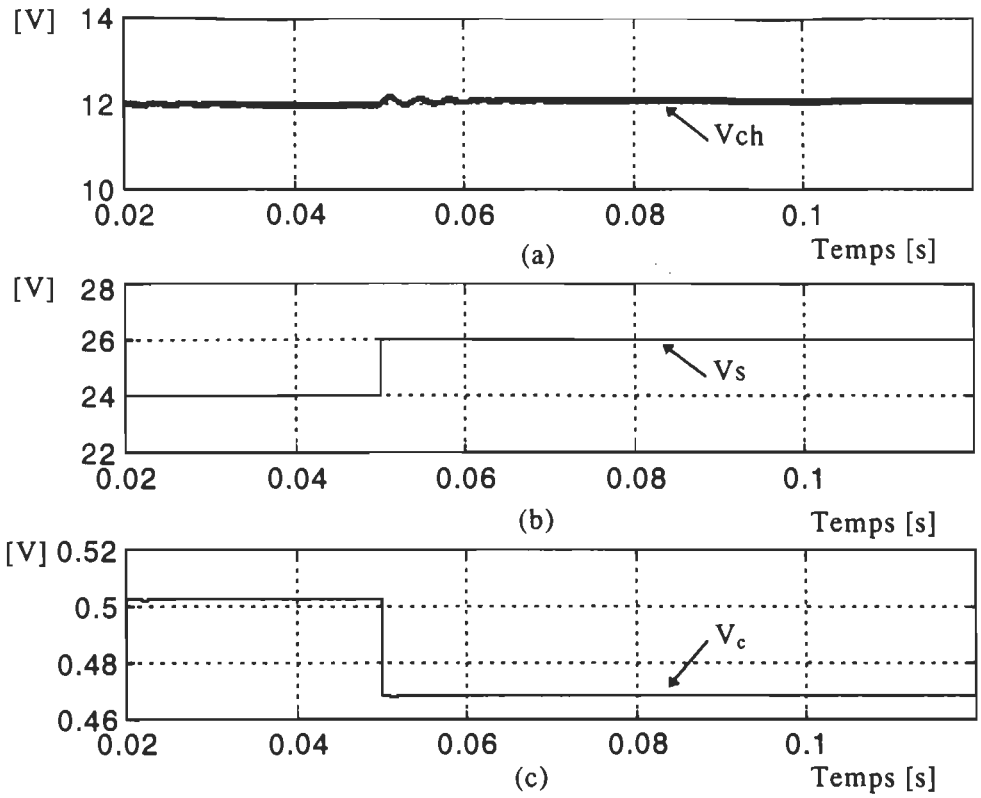


Figure 3.6: Réponse du système à une perturbation de la tension de source:
a) tension de charge, b) tension de source et c) tension de commande

La figure 3.7 montre un changement de consigne $V_{réf}$ de 10V à 12V. La tension V_{ch} suit la consigne et se stabilise à 12V après quelques oscillations amorties. Le modèle inverse répond immédiatement avec une tension de commande convenable pour qu'il stabilise la tension de charge à la consigne. On peut remarquer que, la tension de commande a commencé à descendre brusquement lorsque la protection a réagi pour limiter le courant dans la charge à 1A.

D'après ces résultats le modèle inverse peut jouer le rôle d'un régulateur statique. Ce régulateur ne prend pas en considération la dynamique du système car il n'a appris que les données du régime permanent. Ceci explique les oscillations du régime transitoire. Par contre, ce type de régulateur permet d'obtenir une erreur quasiment nulle en régime permanent du fait qu'il est capable de tenir compte de la

forte non linéarité du système. L'erreur est fonction de la qualité des données d'apprentissage et de la qualité de généralisation du réseau de neurones.

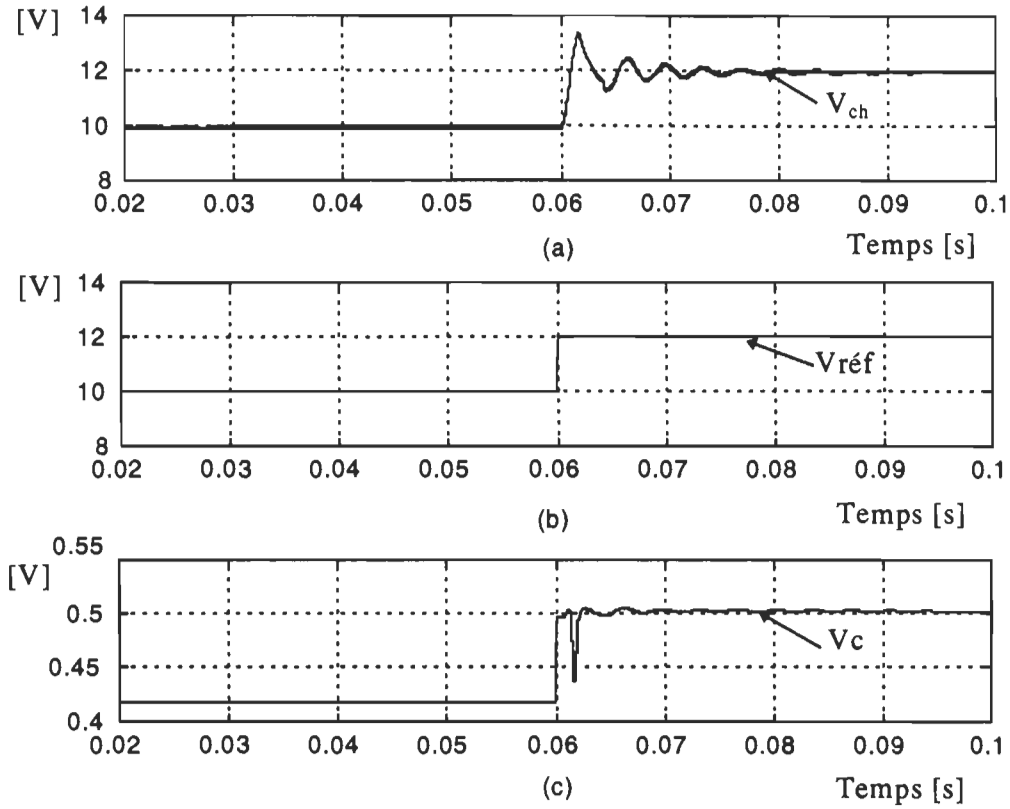


Figure 3.7: Réponse du système à un changement de consigne:
a) tension de charge, b) tension référence c) tension de commande

Limitation du courant de charge

Dans cette partie, nous allons montrer une limitation de courant dans ce convertisseur. Comme on a mentionné précédemment, la chute du signal au niveau de la de commande V_c à 0.43 (figure 3.7) correspond à la détection d'un courant supérieur à 1A.

La figure 3.8 montre le résultat de simulation pour une variation de la résistance de charge de ±5 ohms pour une charge de 15 ohms. Le courant dans la charge est limite à 1A et la tension de charge diminue à une valeur inférieure à la tension $V_{réf}$.

Cette limitation n'est pas recommandée car la tension de commande comporte une grande ondulation et un temps de réponse plus long que celle de la limitation analogique, ce qui augmente les contraintes sur le transistor par rapport à la protection analogique. Cette application montre le pouvoir du réseau de neurones de faire une protection contre les surcharges.

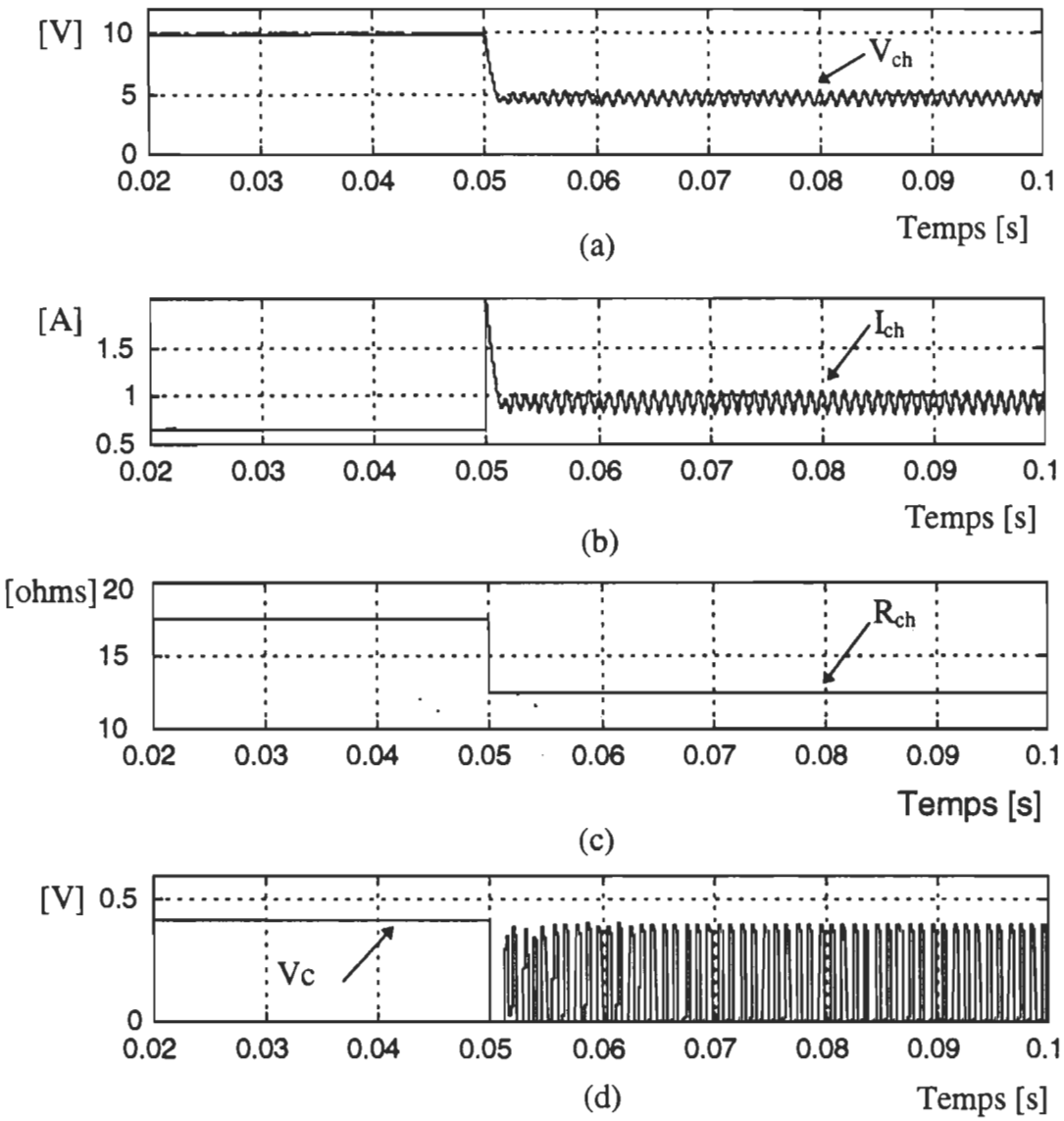


Figure 3.8:Résultats de simulation pour une limitation de courant à 1A: a) tension de charge, b) courant de charge, c) résistance de charge et d) tension de commande

3.5 Hacheur survolteur

Le hacheur survolteur à rapport cyclique constant est un convertisseur instable c'est-à-dire qu'il ne peut pas fonctionner à vide, plus la charge est faible plus la tension de charge augmente. La même méthode d'apprentissage du réseau de neurones de la section 3.4 est utilisée, seulement au lieu que la perturbation vienne de la source, elle vient de la charge.

3.5.1 Apprentissage et préparation des données

L'apprentissage du réseau de neurones se fait de la manière suivante: on fait varier la résistance de charge de 10 à 50 ohms par pas de 10 ohms pour chaque valeur de la tension de commande Vc. Le modèle inverse est entraîné seulement contre la variation de la charge. Après apprentissage, le hacheur sera commandé par le réseau de neurones. La figure 3.9 montre le schéma du hacheur élévateur commandé par le modèle inverse qui joue le rôle d'un régulateur statique.

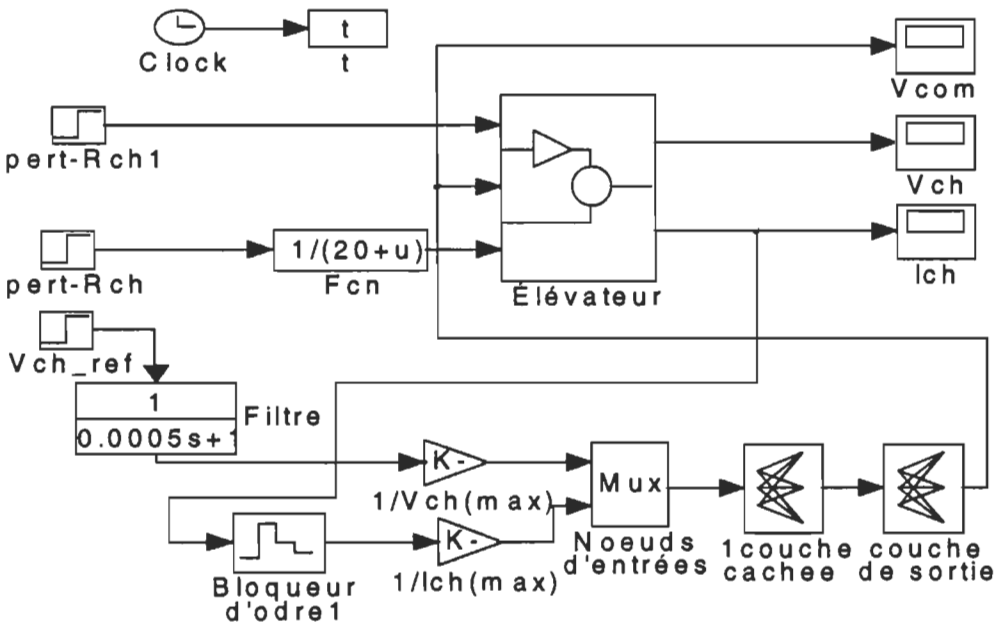


Figure 3.9: Schéma de commande du survolteur par régulateur statique

3.5.2 Simulation du montage

Plusieurs simulations ont été faites pour des changements de consigne et des variations de charge. Ces valeurs sont choisies aléatoirement pour tester le bon fonctionnement du régulateur statique. La figure 3.10 nous montre le résultat de simulation du survolteur pour deux perturbations de la charge (le premier échelon ajoute 20 ohms et le deuxième soustrait 10 ohms), la valeur nominale de la résistance de charge est de 20 ohms.

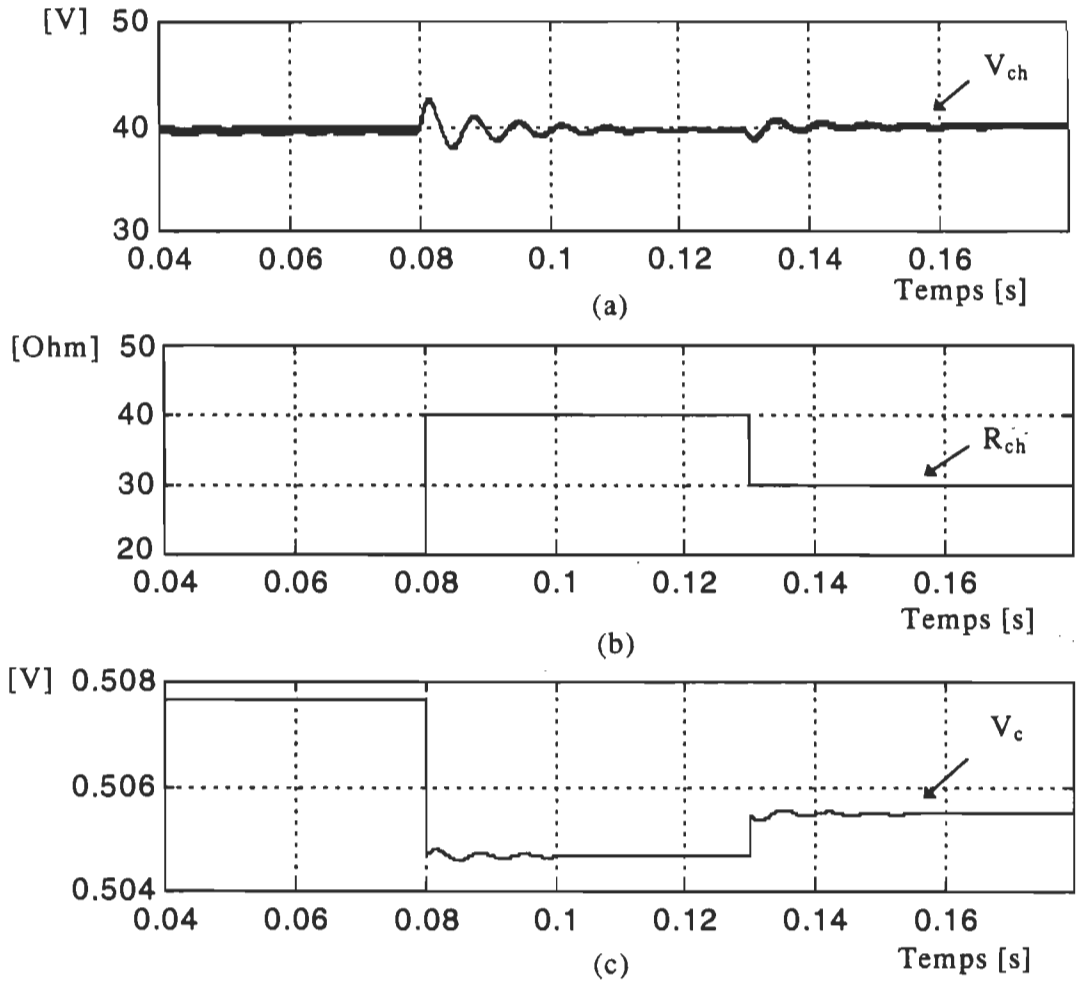


Figure 3.10: Résultats de simulation pour une variation de charge:

a) tension de charge, b) résistance de charge et c) tension de commande

On remarque que la tension de charge est toujours maintenue à 40V après quelques oscillations, car le régulateur n'a pas appris la dynamique du système.

La deuxième simulation est faite pour un changement de consigne (de 40V à 50V). Nous avons ajouté un filtre pour diminuer le premier dépassement au niveau de la tension de charge V_{ch}, son rôle est de diminuer indirectement la montée rapide de la tension de commande V_c. La figure 3.11 nous montre le résultat de simulation pour un changement de consigne $V_{réf}$.

Dans cette deuxième application, on a présenté l'utilisation d'un réseaux de neurones comme régulateur statique. Ce dernier est le modèle inverse du système (survolteur) à l'aide d'un RNA 2-15-1 (15 neurones cachés et un neurone de sortie).

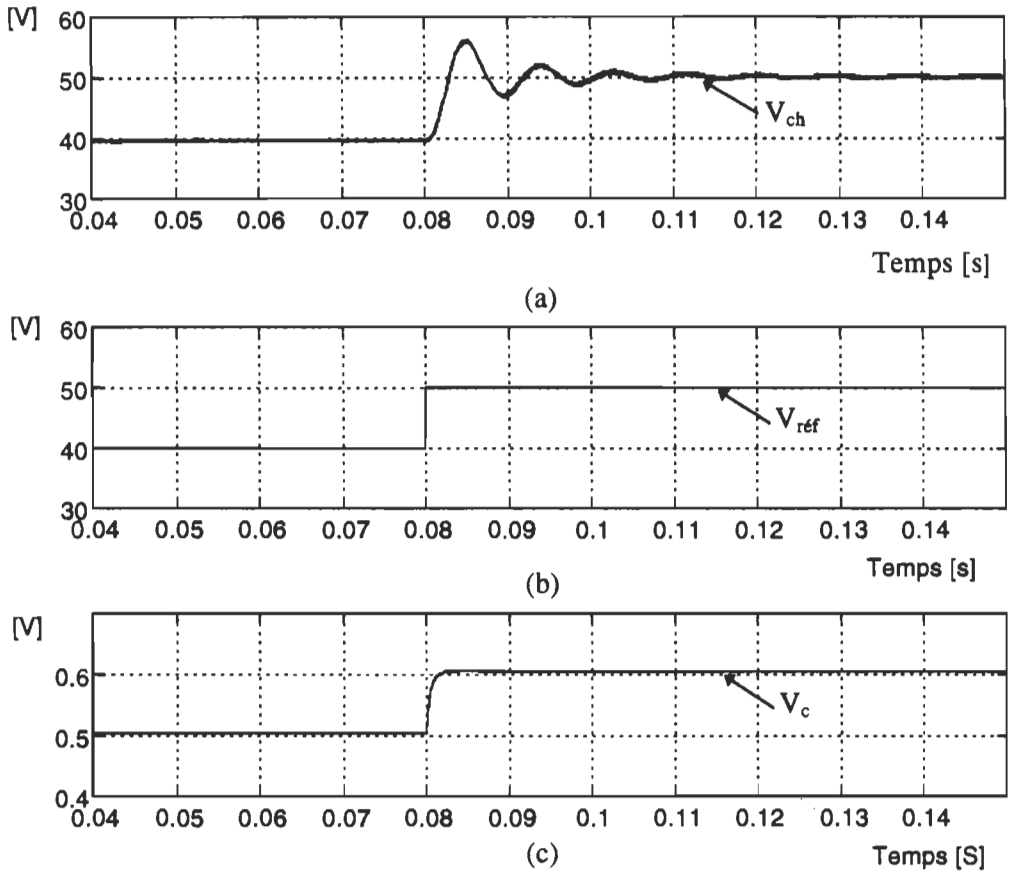


Figure 3.11: Résultats de simulation pour une variation de la tension de source: a) tension de charge, b) résistance de charge et c) tension de commande.

3.6 Redresseur triphasé élévateur mono-interrupteur à FP unitaire

Ce convertisseur, comme mentionné dans le chapitre 1 fonctionne en régime discontinu. Ce régime permet au facteur de puissance d'être unitaire. Dans cette application, nous ferons un réglage de la tension de charge contre les perturbations dues au réseau électrique. La charge est supposée être maintenue constante.

3.6.1 Apprentissage et préparation des données

Les données d'apprentissage sont la tension de charge V_{ch}, l'amplitude de la tension de réseau électrique V_s et la tension de commande V_c comme sortie désirée. Les plages de variation des données sont: de +15% à -10% de 120V pour la tension du réseau électrique et de 0.2V à 0.4V pour la tension V_c.

On fait plusieurs simulations en faisant toutes les combinaisons de la tension de commande V_c et de la tension du réseau électrique V_s. À chaque échelon de l'une ou de l'autre des tensions, le régime permanent doit être établi pour chaque valeur de la tension de charge. La préparation des données d'apprentissage et de généralisation est réalisée de la même manière que pour les autres montages.

V_c=[0.22 0.24 0.26 0.28 0.3 0.32 0.34 0.36 0.38 0.40]

V_s=[102 105 108 111 114 117 119 121 124 127 130 132]

La figure 3.12 montre le résultat de test d'apprentissage pour des données apprises et non-apprises. Le premier test (données apprises) montre deux courbes confondues celle des valeurs apprises et la réponse du réseau de neurone. Le deuxième test (données non-apprises) montre aussi deux courbes presque confondues, celles des données de généralisation et la sortie du réseau de neurones pour ces données sauf pour quelques valeurs. Nous pouvons conclure que l'apprentissage est satisfaisant.

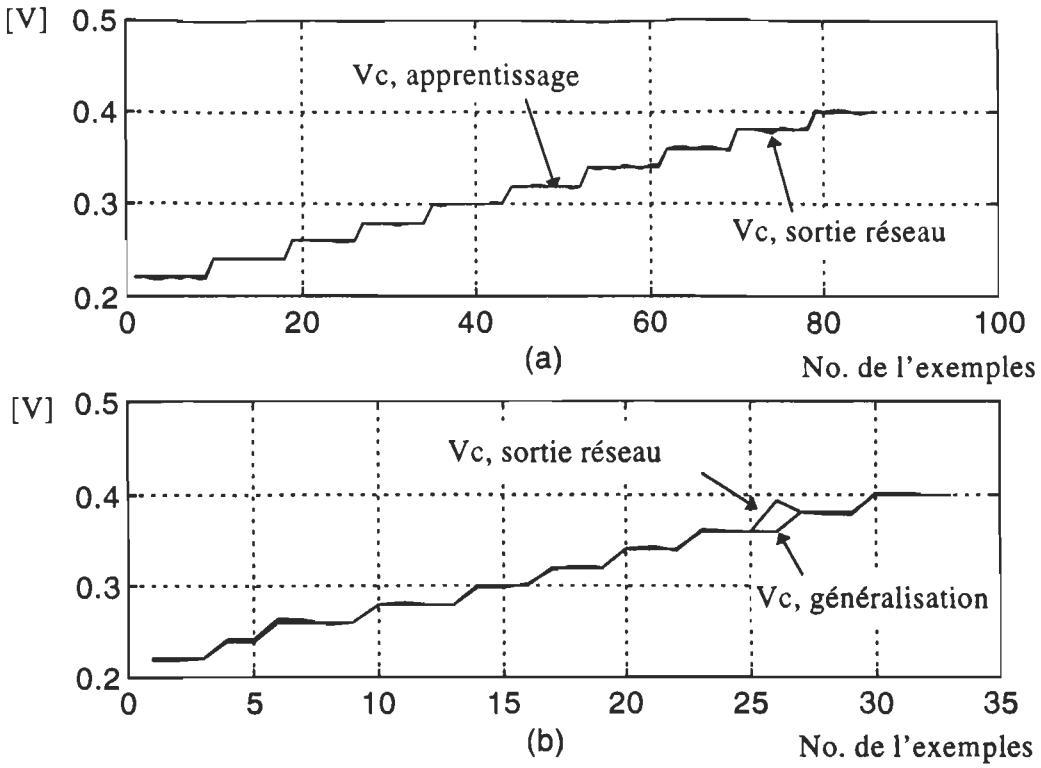


Figure 3.12: Test d'apprentissage: a) données apprises et b) données non-apprises

3.6.2 Simulation du montage

Le réseau de neurones comporte deux noeuds dont les entrées sont l'amplitude de la tension du réseau électrique V_s et la tension de référence $V_{réf}$. La sortie du réseau de neurones donne la tension de commande V_c. La figure 3.13 présente le schéma du montage en bloc SIMULINK® commandé par le réseau de neurones.

Le but de la première simulation est de réguler la tension de charge V_{ch} à 500V. Le réseau de neurones est testé pour une chute brusque de la tension du réseau électrique V_s (125V à 115V) et l'établissement de cette dernière à la valeur nominale qui est 120V.

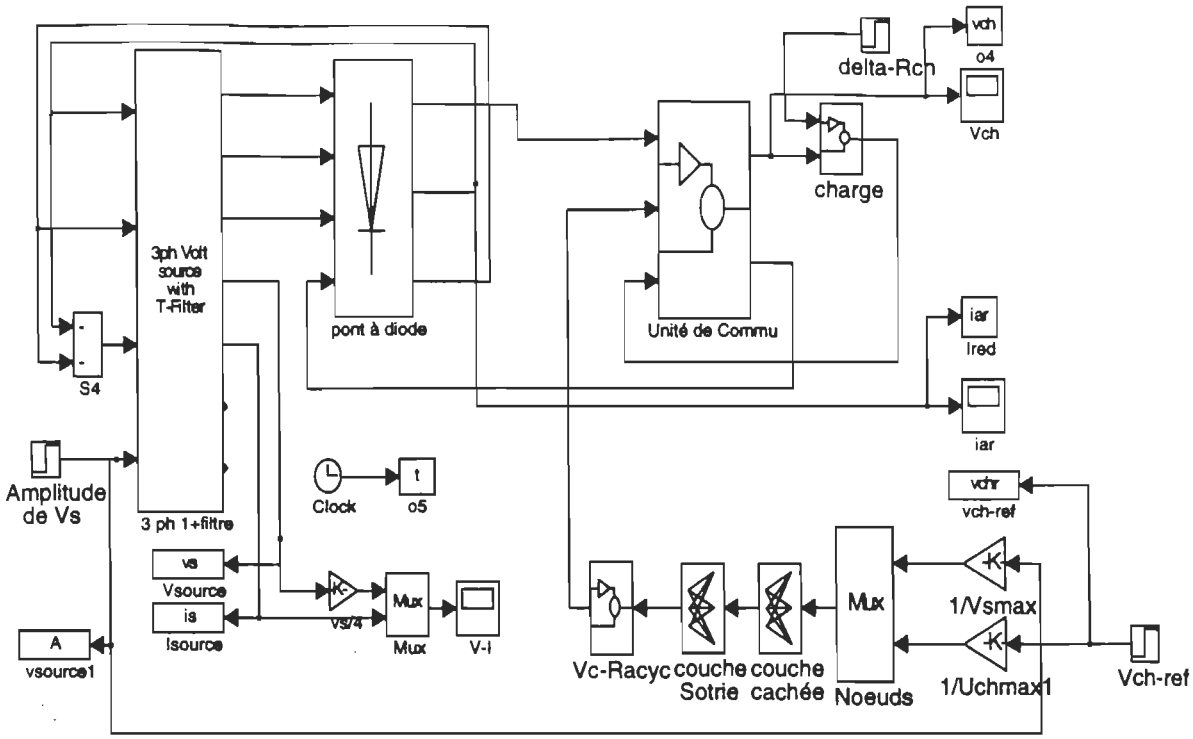


Figure 3.13: Schéma du montage commandé par régulateur statique

Ces trois valeurs de tensions ne font pas partie des données d'apprentissage. La figure 3.14 montre le résultat de simulation de ce réglage. La courbe de la tension du réseau électrique est réduite de quatre fois pour pouvoir la superposer avec courbe du courant I_s. D'après les résultats de simulation, la tension de V_{ch} reste stable à 500V, le facteur de puissance est presque unitaire et le courant dans l'inductance est discontinu.

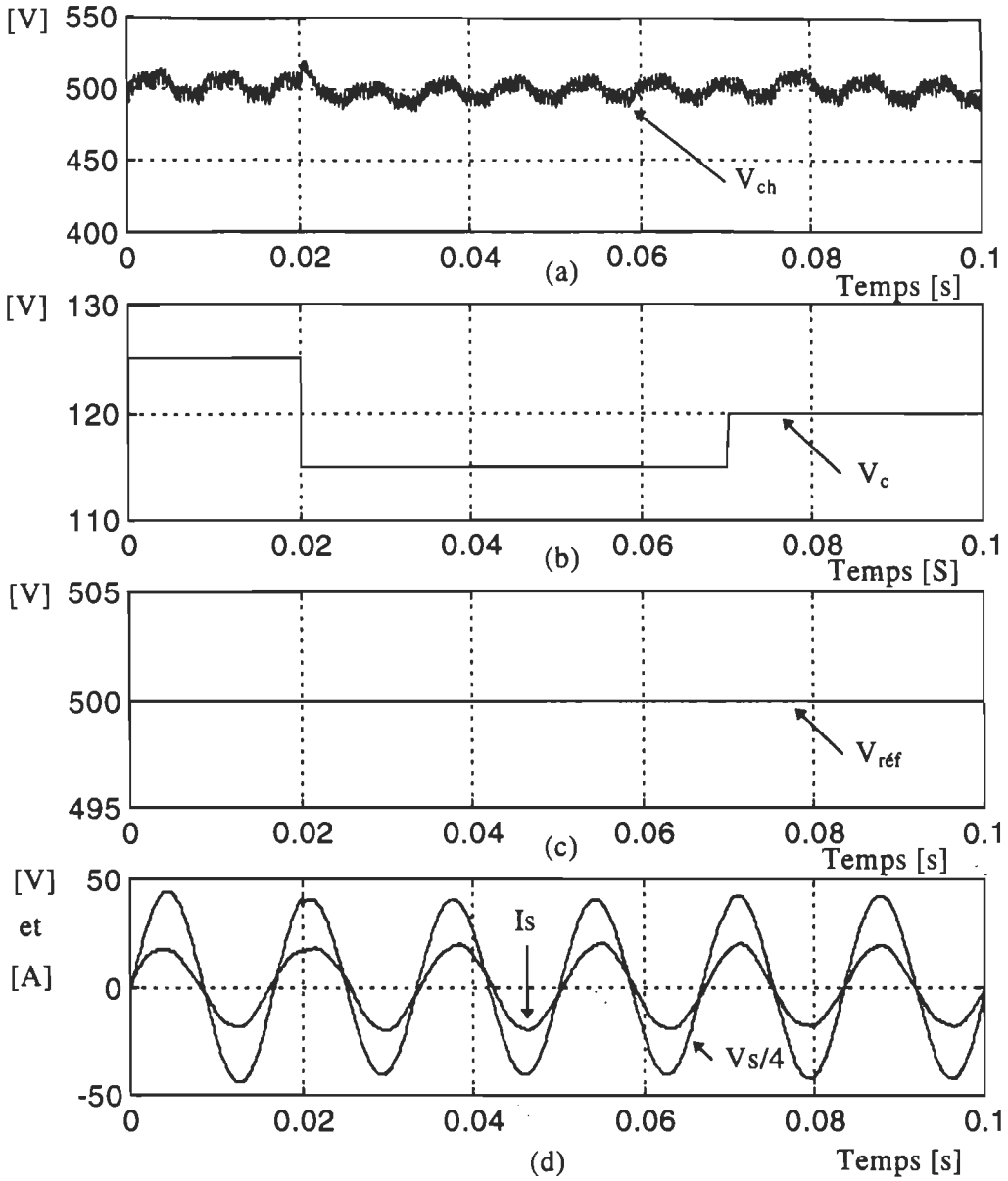


Figure 3.14: Résultats de simulation pour une variation de la tension de source: a) tension de charge, b) tension de source, c) tension de référence, et d) courant et tension de source.

La deuxième simulation permet de tester le réseau de neurones pour une variation de la tension de référence $V_{réf}$. Le premier échelon varie de 480V à 520V, c'est à dire de -4% à +4% de la tension de charge nominale. La tension V_{ch} suit la référence et

se stabilise à 520V, ensuite on applique un deuxième échelon rendant la tension de consigne à la tension nominale de 500V.

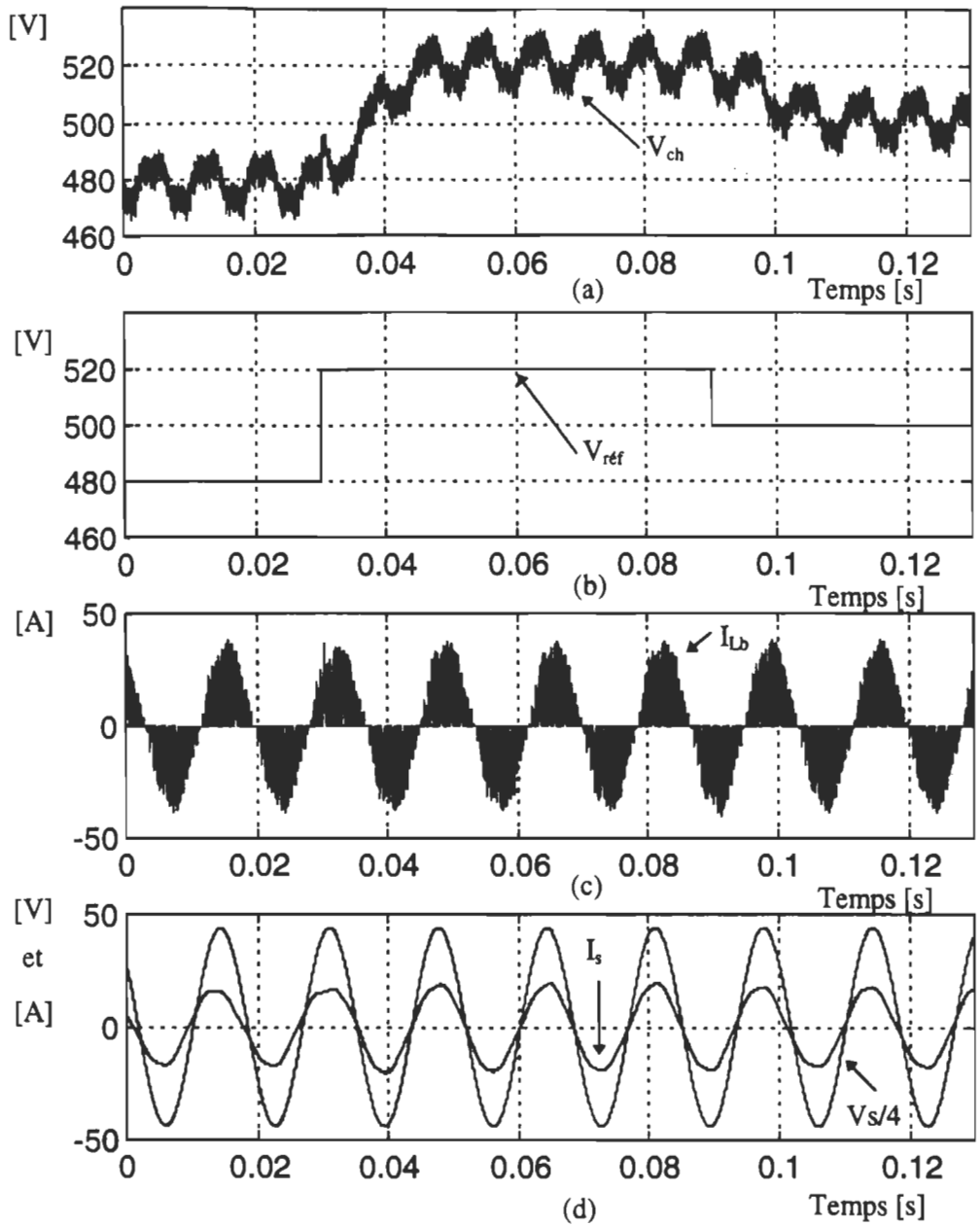


Figure 3.15: Résultat de simulation pour un changement de consigne: a) tension de charge, b) tension de référence, c) courant dans L_b, et d) courant et tension de source

La figure 3.15 nous montre la tension de charge V_{ch} qui suit la consigne, le courant discontinu dans les inductances ainsi que la tension et le courant du réseau électrique (la tension V_s est réduite de quatre fois).

3.7 Conclusion

Dans ce chapitre, nous avons montré l'application du modèle inverse comme régulateur statique pour la tension de charge et comme protection contre les surcharges. Pour les deux premiers montages, on a appliqué le réseau de neurones pour réguler la tension de charge contre les perturbations dues à la charge et à la source ainsi qu'une limitation de courant par le même réseau de neurones. Pour le dernier montage, une régulation de la tension de sortie est accomplie en gardant un facteur de puissance unitaire. On peut donc conclure que le modèle inverse joue le rôle d'un régulateur statique donnant des résultats satisfaisants compte tenu de la forte non linéarité des montages. Cependant, ce modèle avec la commande statique ne fournit pas la dynamique désirée pour le système.

CHAPITRE 4

RÉGULATEUR DYNAMIQUE

4.1 Introduction

Dans les dernières années, beaucoup de travaux ont vu le jour dans le domaine du contrôle par les réseaux de neurones. Ces derniers sont caractérisés par une commande dynamique d'un système. La conception des régulateurs classiques demande une analyse mathématique et présente des difficultés pour le contrôle d'un système non linéaire. L'utilisation des réseaux de neurones donne une grande flexibilité pour résoudre ces types de difficultés.

Notre objectif dans ce chapitre est de concevoir un régulateur dynamique à base d'un réseau de neurones pour le montage redresseur triphasé survolteur vu dans le chapitre 3. Le régulateur à réseau de neurones doit assurer une bonne dynamique de sortie à la réponse d'une référence donnée. Il y a plusieurs types de régulateurs à réseau de neurones (adaptatif, modèle de référence, optimal,...) et l'entraînement se fait soit en ligne ou hors ligne [18, 24-25].

4.2 Différentes classes de régulateurs

La commande d'un système par réseau de neurones est basée sur un nombre élevé d'information traitée et adaptée afin de réguler le système non linéaire. Considérons un système avec une entrée et une sortie (SISO: Single Input Single Output) [18], décrit par l'équation suivante:

$$y(k+1) = f\left[y(k), y(k-1), \ldots, y(k-m-1), u(k), u(k-1), \ldots, u(k-n-1)\right] \qquad (4.1)$$

où

y: sortie du système,

u: entrée du système,

k: l'instant d'échantillonnage k

m: retard sur la sortie et

n: retard sur l'entrée

Le régulateur permet de forcer la sortie à suivre la référence r(k+1) pour minimiser l'erreur e(k+1) à travers le processus d'apprentissage défini comme suit:

$$e(k+1) = r(k+1) - y(k+1) \tag{4.2}$$

Le contrôle direct et indirect sont les deux principales classes de contrôleurs.

4.2.1 Contrôle direct

La figure 4.1 montre le principe d'un contrôle direct d'un système. Le régulateur à réseau de neurones permet de donner une commande au système par les retards au niveau de la commande et de la sortie permettant au régulateur d'apprendre la dynamique du système. À l'initialisation, les poids du régulateur à réseau de neurones sont aléatoires donnant ainsi une erreur sur la commande u(k) et générant une erreur sur la sortie y(k+1). L'erreur e(k+1) est utilisée pour corriger les poids du régulateur à réseau de neurones. Plus le nombre d'itérations augmente, plus le réseau de neurone devient performant et l'erreur e(k+1) diminue [18, 28].

Pour corriger les poids du régulateur, on a besoin de l'erreur $e_c(k)$ à la sortie du régulateur.

$$e_c(k) = d(k) - u(k) \tag{4.3}$$

où

d(k) est la commande désirée qui minimise l'erreur e(k+1).

Seule l'erreur à la sortie du système e(k+1) est disponible. Donc l'erreur de la commande peut s'écrire de la façon suivante:

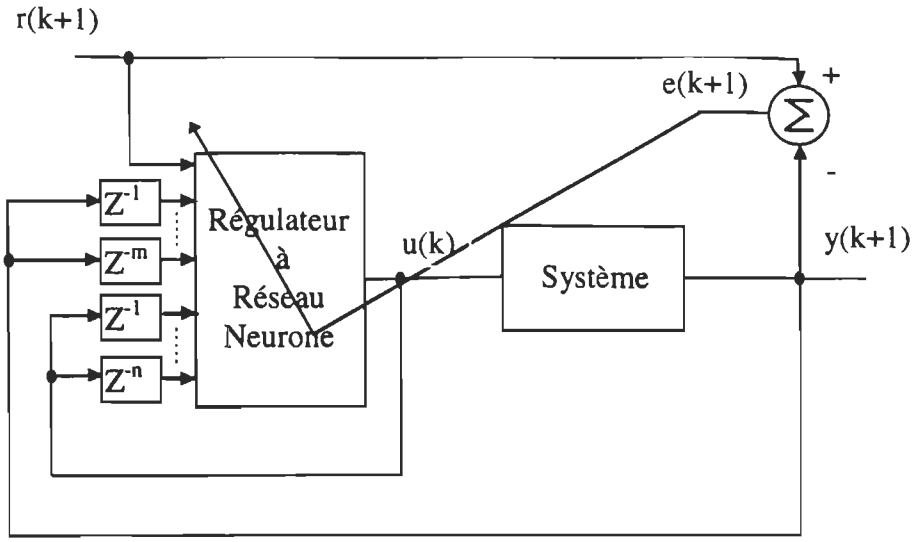


Figure 4.1: Schéma de contrôle direct

$$e_c(k+1) = \frac{\partial y(k+1)}{\partial u(k+1)} \tag{4.4}$$

Pour faire ce type de contrôle, le jacobien du système doit être connu, c'est-à-dire que la sortie du système en fonction de l'entrée doit être connue.

4.2.2 Contrôle indirect

Dans le cas où le système n'est pas bien défini, le contrôle indirect s'impose. La figure 4.2 montre le schéma de principe d'un contrôle indirect [18, 24-28].Ce type de contrôle a besoin de deux réseaux de neurones: le premier sert de modèle au système permettant de propager l'erreur e(k+1) de la sortie du modèle à son entrée pour trouver l'erreur sur la commande afin de corriger les poids du deuxième réseau de neurones jouant le rôle de régulateur.

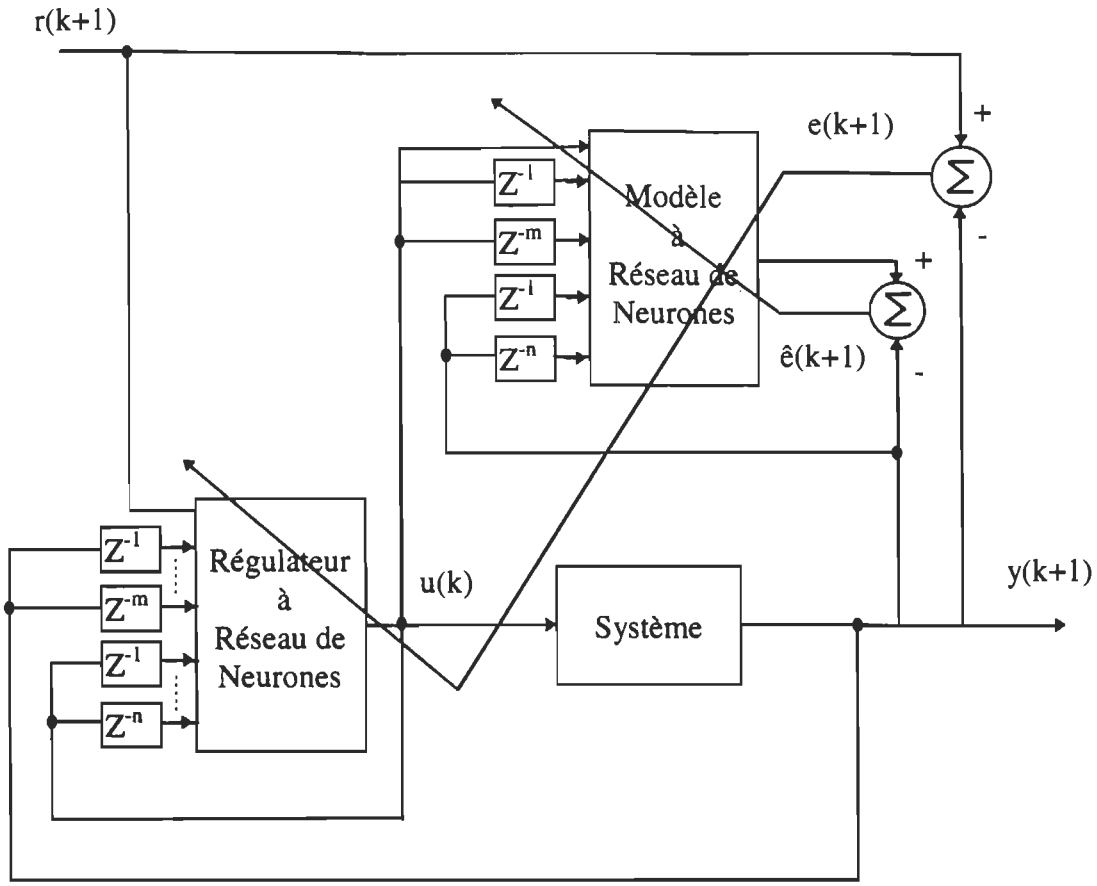


Figure 4.2: Schéma de contrôle indirect

4.3 Les réseaux récurrents à couches

Les réseaux récurrents à couches intègrent à la fois les caractéristiques du réseau perceptron multicouches et des réseaux récurrents. Les perceptrons multicouches ne sont capables de présenter que des comportements statiques ou dynamiques très simples [12]. Les réseaux récurrents ont en général des comportements dynamiques plus élaborés que le PMC, car la sortie dépend des entrées précédentes. Les réseaux récurrents à couches ont leurs neurones groupés en couches et possèdent des boucles récurrentes qui permettent des comportements dynamiques plus complexes. Les règles d'apprentissage de ces réseaux considèrent généralement les réseaux comme non récurrents. Il y a deux techniques connues d'apprentissage capables d'entraîner

les réseaux récurrents à couches: la rétropropagation dans le temps et l'apprentissage en temps réel [12, 24].

4.3.1 Rétropropagation dans le temps

Cette technique a été proposée à l'origine par Rumelhart et Werbos. Elle est appelée la rétropropagation dans le temps "Back-Propagation Through Time" (BPTT). La méthode consiste à "déplier" le réseau récurrent autant de fois qu'il y a eu de pas de temps dans le traitement et à partager les poids entre les pas de temps. Les neurones au temps t sont alors distincts des neurones temps t+1, cela élimine la récurrence et une rétropropagation classique peut alors être effectuée en propageant l'erreur des neurones les plus récents vers les plus anciens. La figure 4.3 montre un réseau récurrent déplié dans le temps [10-29].

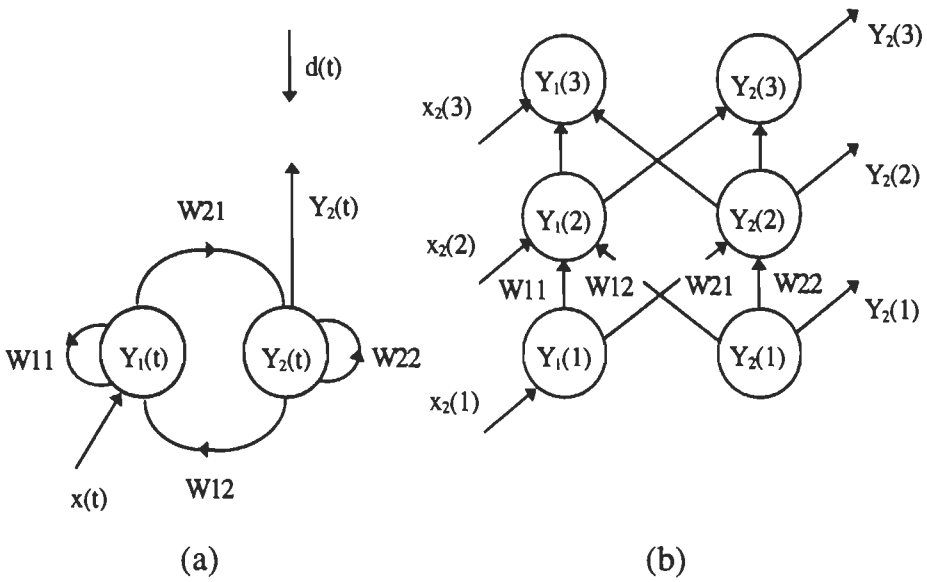


Figure 4.3: a) réseau récurrent, b) le même "déplier" dans le temps

4.3.2 Apprentissage en temps réel

Cette technique d'apprentissage fut développée par Williams et Zipser et appelée apprentissage en temps réel "Real-Time Recurrent Learning" (RTRL). L'appellation RTRL désigne que la modification des poids se fait à chaque pas de temps au lieu de

la fin de celui-ci. Cette technique demande beaucoup de paramètres et de mémoires [12, 22].

4.4 Outils logiciels pour l'identification et le contrôle

Pour l'obtention des résultats nous avons utilisé des outils logiciels fonctionnant sur le logiciel MATLAB® développé à l'Université Technique de Denmark [30-31]. Ce sont deux outils, dont le premier sert à l'identification des systèmes et le deuxième sert pour la conception des régulateurs, sont basés sur les réseaux de neurones et fonctionnent sur le logiciel MATLAB®.

Ces outils contiennent des programmes et des fonctions permettant l'entraînement et l'évaluation du modèle du système. Ces programmes sont basés sur des réseaux multicouches. En somme, les deux programmes sont des algorithmes d'apprentissage en temps réel et ils permettent la conception des régulateurs de processus non linéaires.

4.5 Conception du régulateur dynamique

En général, les systèmes électroniques de puissance sont des systèmes non linéaires comme mentionné au chapitre 3. Cette non linéarité expliquée au chapitre 1 est due à plusieurs facteurs tels que la fréquence de commutation, le mode de fonctionnement, etc. Le montage redresseur survolteur dont on veut réguler la tension de sortie n'est pas bien connu sous forme d'équation mathématique, c'est-à-dire que son jacobien est inconnu. Cela nous amène à utiliser le contrôle indirect qui requiert l'utilisation de deux réseaux de neurones. En effet, le premier permet d'identifier le modèle du système qui servira à l'entraînement du deuxième réseau qui constituera le régulateur dynamique [18-20].

4.5.1 Identification du système

Il y a trois étapes pour identifier un système [30-32]: préparation des données d'entrée sortie du système, sélection de la structure pour estimation du régulateur et validation du modèle.

4.5.1.1 Préparation des données

La préparation des données exige d'abord la simulation du système avec plusieurs échelons de tension de commande pour obtenir plusieurs transitions de la tension de sortie. La tension de sortie est filtrée pour enlever les ondulations causées par la fréquence de commutation. Le réseau de neurones apprend des constantes de temps du système sans la constante de temps due à la fréquence de commutation.

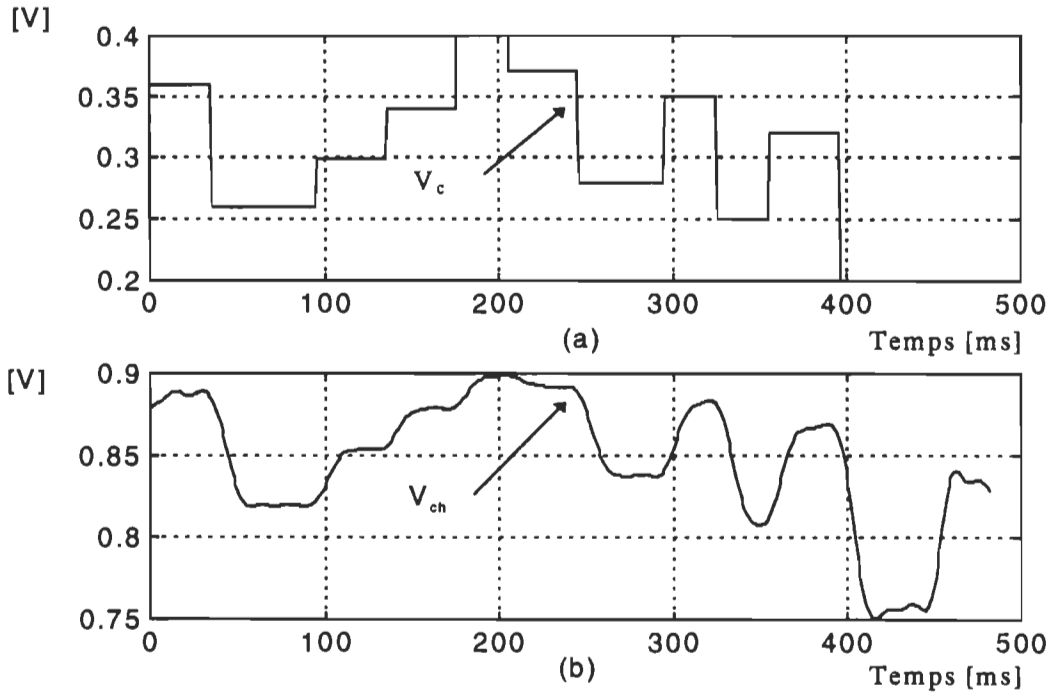


Figure 4.4: Données d'entrée sortie du redresseur

a) tension de commande et b) tension de charge

La figure 4.4 montre les données d'entrées et sorties du redresseur représentées par la tension de commande V_c et la tension de sortie V_{ch} après filtrage du bruit de commutation.

4.5.1.2 Sélection de la structure du modèle et estimation

Le choix de la structure consiste à trouver le vecteur d'entrée et à déterminer l'architecture du modèle. On estime le modèle par la fonction autorégressive "nnarx" pour identifier le redresseur. Ce modèle utilise l'algorithme de rétropropagation de Levenberg-Marquardt pour l'évaluation des poids du réseau de neurones selon les entrées u(t) et sorties y(t). Le modèle "nnarx" peut s'exprimer selon les équations suivantes:

$$\varphi(t) = \left[y(t-1)\ldots y(t-n_a), u(t-n_k)\ldots u(t-n_b-n_k+1)\right]^T \quad (4.5)$$

$$\hat{y}(t|\theta) = g(\varphi(t),\theta) \quad (4.6)$$

où

$\varphi(t)$ vecteur de retour,

$\hat{y}(t|\theta)$ signal de sortie du modèle,

θ matrice de poids,

g fonction réalisée par le réseau de neurone,

n_a retards sur la sortie,

n_b retards sur l'entrée et

n_k pas de temps mort.

Par l'analyse des données d'entrées et de sorties du montage redresseur triphasé survolteur, nous observons un comportement se rapprochant de celui d'un système de premier ordre. Ceci nous permet de proposer un retard n_a=1 sur la tension de

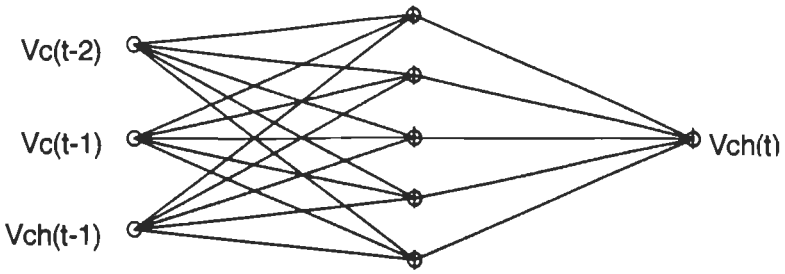


Figure 4.5: Structure du modèle

sortie et un retard n_b=2 sur la tension de commande avec n_k=0. L'architecture du modèle montré à la figure 4.5 définit le nombre de neurones et le nombre de couches pour ce modèle. On prendra cinq neurones sur la couche cachée avec une fonction d'activation de type tangente hyperbolique et un neurone de sortie dont la fonction d'activation est de type linéaire. La tension est normalisée par rapport à 600V. L'apprentissage du réseau de neurones se fait par l'appelle de la fonction "nnarx". L'arrêt de l'apprentissage est fait à une erreur quadratique égale 10^{-6}. La figure 4.6 montre un résultat typique d'apprentissage.

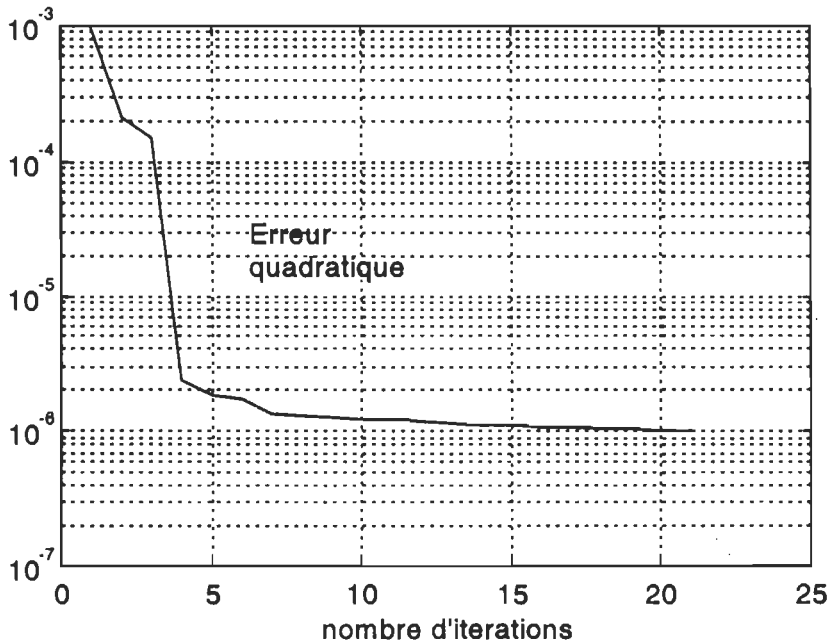


Figure 4.6: Erreur quadratique

4.5.1.3 Validation du modèle

Le modèle a été évalué en étudiant l'erreur de prédiction pour des couples d'entrée et de sortie non apprises (généralisation). La figure 4.7 montre un exemple de validation du modèle.

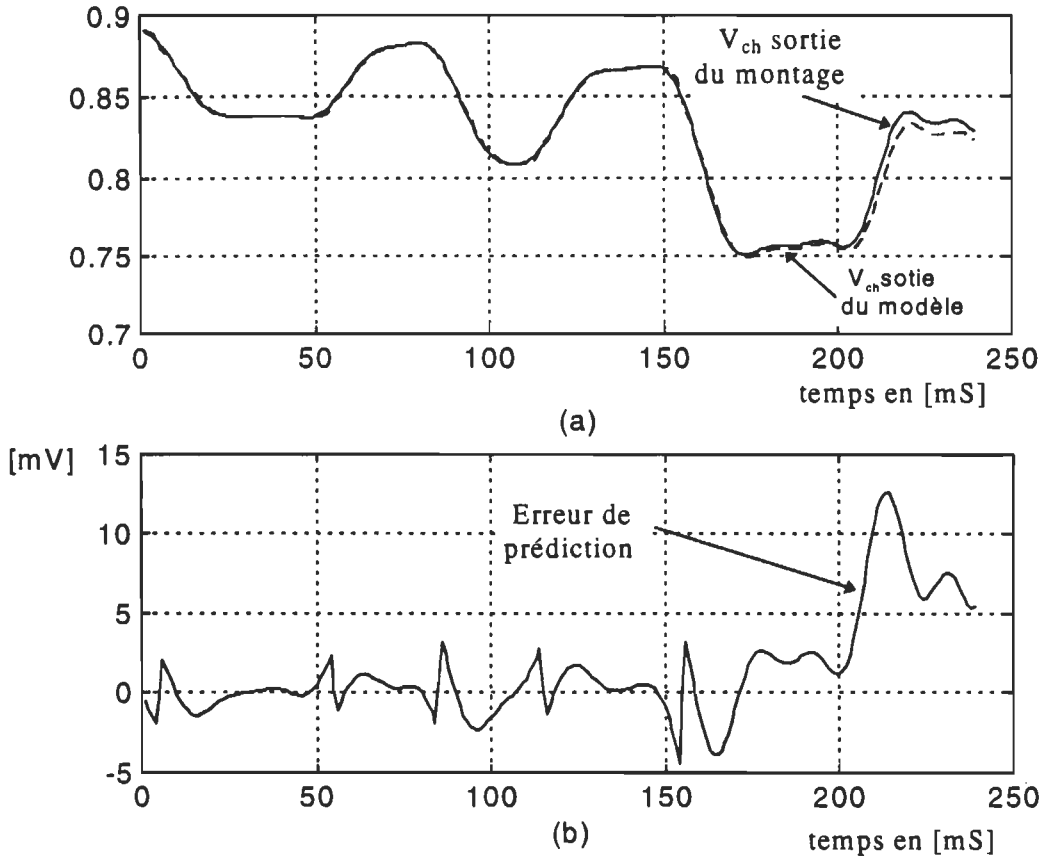


Figure 4.7: Résultat de validation du modèle, tension normalisée par rapport à 600V
a) V_{ch} (trait plein) et V_{ch} du modèle (trait pointillé) et b) erreur de prédiction

Le modèle peut généraliser même pour des données non apprises. La figure 4.7 montre une erreur de prédiction maximale de 1.5% qui est équivalente à 7.2V pour une tension nominale de sortie de 500 V.

4.5.2 Conception du régulateur

Comme mentionné au début, nous avons appliqué un contrôle optimal pour notre redresseur triphasé. Ce régulateur est basé sur le modèle inverse du système dont le principe est de considérer le système mentionné dans l'équation (4.1) par son modèle inverse donné par [28, 33].

$$\hat{u}(k+1) = \hat{f}^{-1}\left[y(k+1), y(k), \ldots, y(k-m-1), u(k), u(k-1), \ldots, u(k-n-1)\right] \qquad (4.7)$$

où,

m=1 et n=1,

u(k): tension de commande,

y(k+1): tension de référence

De cette équation, on remplace la sortie y(k+1) par la référence r(k+1). Le principe de l'algorithme d'entraînement du régulateur optimal est basé sur la minimisation de la fonction critique suivante [31]:

$$J(\theta) = \sum_{t} \left(r(t) - y(t)\right)^2 + \rho\left(u(t)\right)^2 \text{ avec } . \rho \geq 0 \qquad (4.8)$$

ρ est un facteur de pénalité.

4.5.2.1 L'entraînement du régulateur

Le régulateur optimal est entraîné par un algorithme en ligne et il a besoin d'un modèle du système à réguler. L'apprentissage se fait par l'appel de la fonction "opttrain", cette fonction est initialisée par le fichier "optrinit". Ce dernier contient toutes les données pour l'apprentissage du régulateur optimal (le modèle du système, la structure du régulateur, le signal de référence, les paramètres de simulation, etc.). La modification des poids du régulateur se fait à chaque pas d'itération. Une époque de simulation correspond à une ou plusieurs périodes du signal de référence, dans notre cas l'époque correspond à 2 périodes. Le signal de référence est formé de

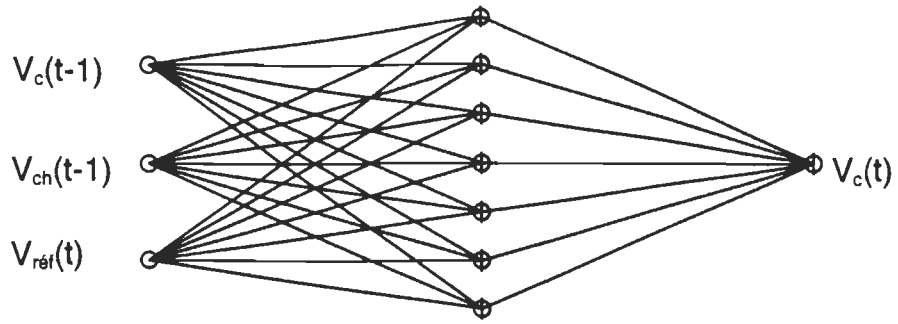


Figure 4.8: Structure du régulateur

plusieurs échelons permettant au régulateur d'apprendre le réglage de la tension de sortie du redresseur. La structure (voir figure 4.8) de ce régulateur est la même que le modèle du redresseur sauf il contient 7 neurones dans la couche cachée [33] .

À chaque époque (cycle) d'apprentissage, le régulateur améliore sa commande pour que la sortie V_{ch} suive tout près la référence $V_{réf}$. Entre l'époque 2 et 4, le régulateur optimise la tension de commande à chaque pas d'itération. Nous avons remarqué d'après la figure 4.9 que la tension de commande du régulateur dépasse les valeurs limites de la tension de commande $(0.2 \leq Vc \leq 0.4)$ de notre système, les dépassements au niveau de la commande permettent à la tension de charge de suivre la référence et de minimiser le temps de réponse du système. L'étape suivante sera la validation et la généralisation du fonctionnement du régulateur.

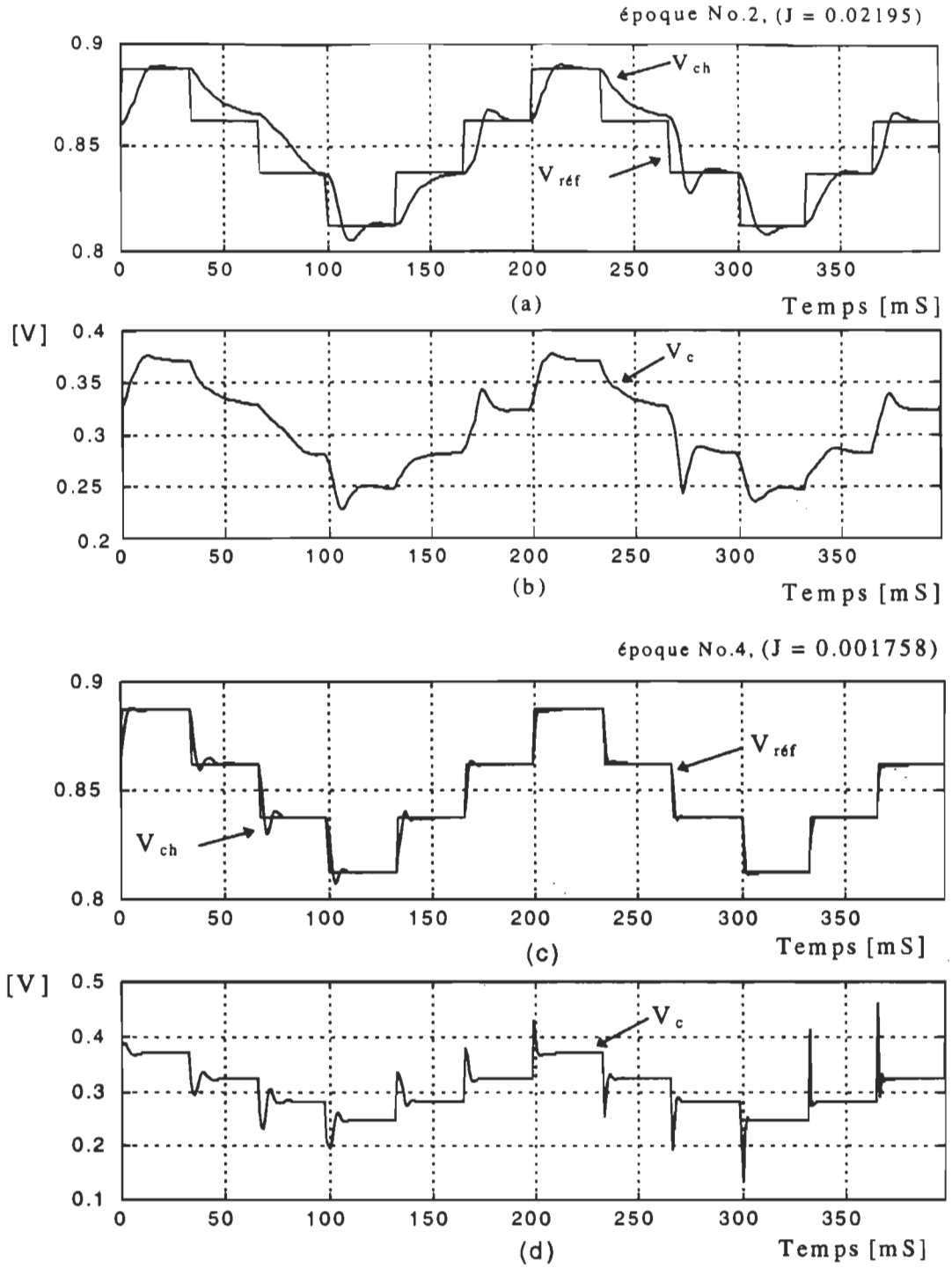


Figure 4.9: Résultats de l'apprentissage

époque No.2 : a) $V_{réf}$ (référence) et V_{ch} (modèle) et b)V_c : tension de commande

époque No.2 : c) $V_{réf}$ (référence) et V_{ch} (modèle) et d)V_c : tension de commande

4.5.2.2 Validation du régulateur

Cette étape sera celle du test du régulateur par un signal de référence $V_{réf}$ différent du signal d'apprentissage. La validation sera effectuée par la fonction "invcon". Cette fonction est initialisée par le fichier "optinit". Ce dernier contient presque les mêmes données du fichier d'initialisation de l'entraînement. Il permet le chargement de la structure du régulateur, le nouveau signal de référence, le modèle du système et la durée de simulation.

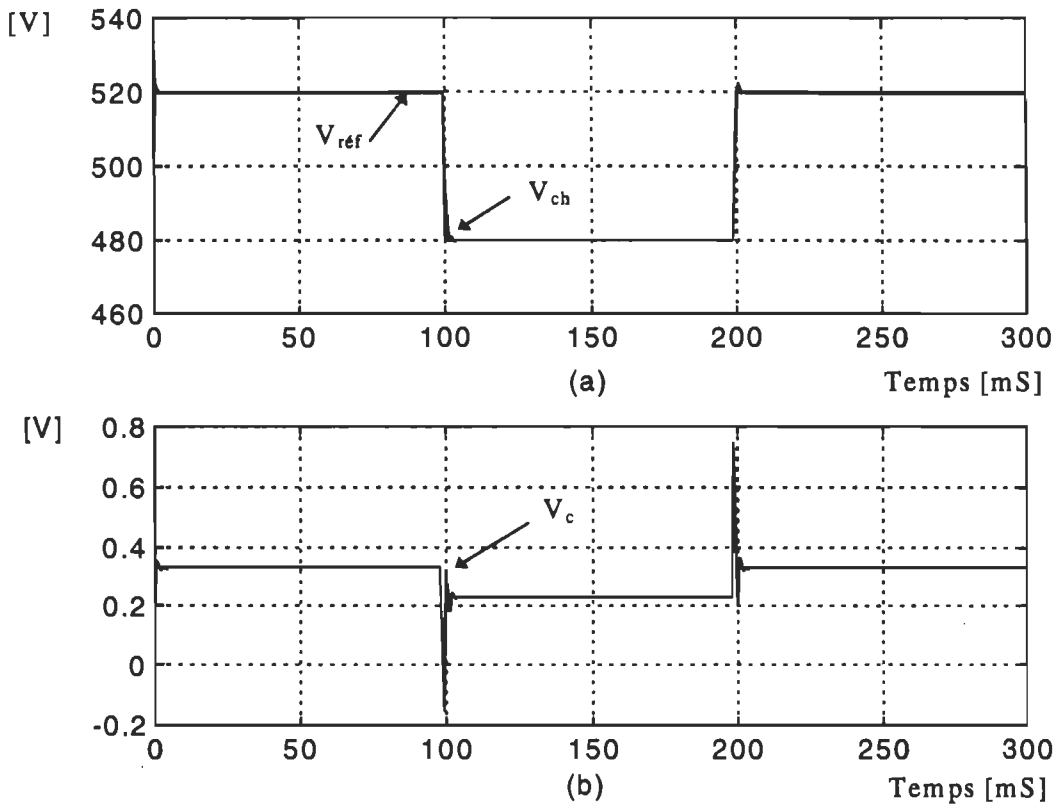


Figure 4.10: Validation de l'apprentissage du régulateur

a) tensions: $V_{réf}$ et V_{ch}(modèle) et b) tension de commande V_c

D'après la figure 4.10 le régulateur commande parfaitement le système. La sortie du modèle suit le signal de référence sauf que la commande dépasse les limites de la

tension de commande du système. Cela nous oblige de faire une limitation à la sortie du régulateur qui détériorera un peu la dynamique de la réponse du système.

4.6 Comparaisons des régulateur dynamique et statique

Pour faire une comparaison entre le régulateur dynamique et le régulateur statique, nous allons vérifier les performances d'une part sur le modèle en réseau de neurones et d'autre part sur le montage simulé par blocs SIMULINK®.

4.6.1 Réponse du modèle à un échelon de consigne

Pour montrer l'avantage du régulateur dynamique par rapport au régulateur statique vu dans le chapitre 3, nous commandons le modèle du redresseur par les deux régulateurs.

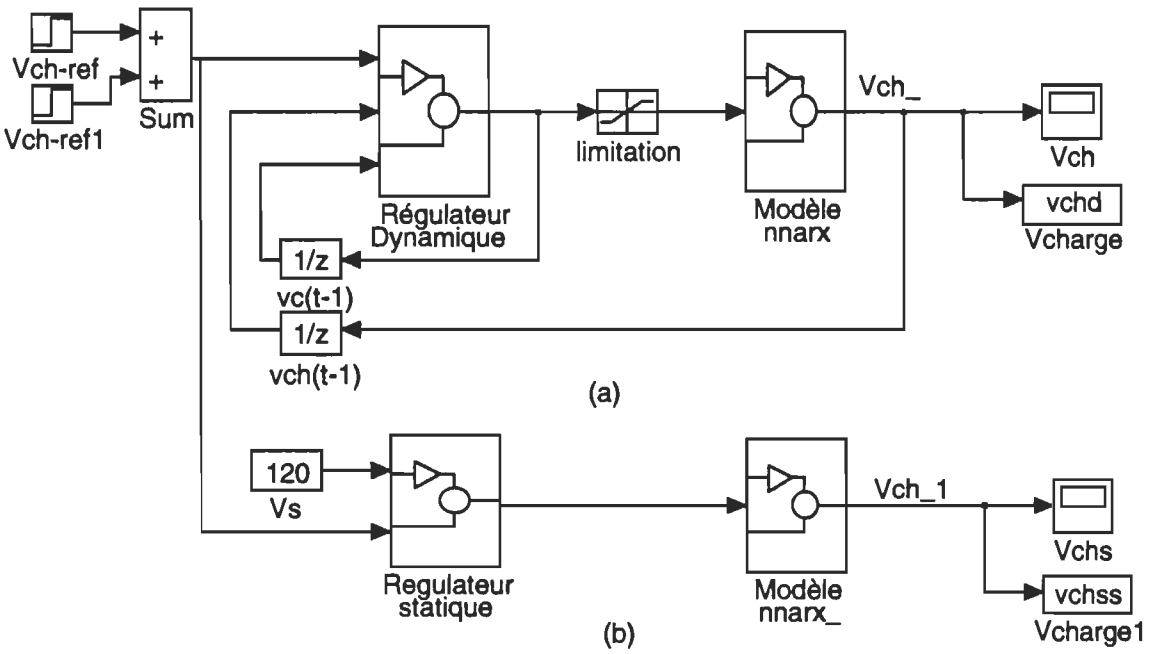


Figure 4.11: Schéma du montage de test

La figure 4.11 montre le schéma bloc en SIMULINK® pour simuler la comparaison entre le régulateur dynamique et statique sur le modèle du redresseur triphasé. Un bloc de limitation est inséré entre le régulateur dynamique et le modèle pour éviter une commande excessive au niveau du régulateur dynamique tel que vu à la section 4.5.2.1.

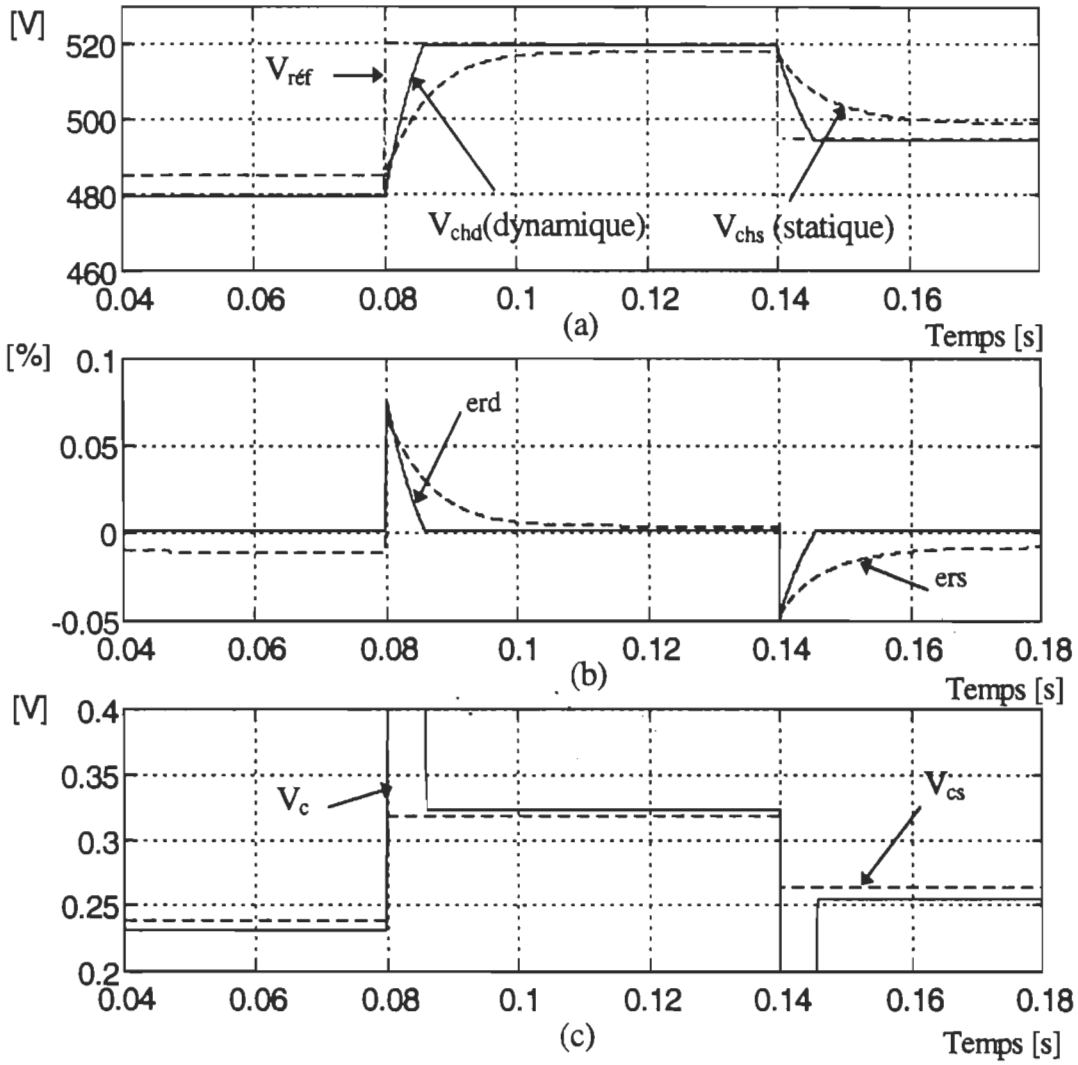


Figure 4.12: Comparaison entre les régulateurs dynamique et statique V_{chd} (trait continu, dynamique), $V_{réf}$ (-.) et V_{chs} (-- ,statique), a) tension de charge et de référence, b) erreur relative et c) tension de commande

La figure 4.12 montre les résultats de simulation pour une référence d'un échelon de 40V. La courbe en trait interrompu représente la réponse du modèle avec le régulateur dynamique et la courbe en trait discontinu représente la réponse du même modèle avec le régulateur statique. L'avantage du régulateur dynamique est qu'il permet une commande anticipée de façon à obtenir une réponse plus rapide et une meilleure annulation de l'erreur au régime permanent. Le régulateur dynamique est meilleur de point de vu temps de réponse et précision.

4.6.2 Réponse du système à un échelon de consigne

Après la comparaison de la qualité de réglage sur le modèle du système, on doit vérifier ces qualités sur le montage de simulation implanté en blocs SIMULINK®. La simulation, telle que décrite à la section 1.4, qui est très proche de la réalité. Le même montage est commandé par les deux régulateurs. Quant à celui qui est commandé par le régulateur dynamique, on a besoin de deux échantillonneurs - bloqueurs pour considérer les retards de la tension de commande et de la tension de charge. Un filtre est utilisé pour la tension de sortie avec une fréquence de coupure de 800Hz . La figure 4.13 montre le résultat de simulation pour la même tension de référence d'un échelon de tension de 40V. On peut voir que la réponse de la sortie du montage commandé par le régulateur dynamique est quelque peu plus rapide par rapport au régulateur statique. Par contre, dans les deux cas, l'erreur en régime permanent de la valeur moyenne est quasiment nulle.

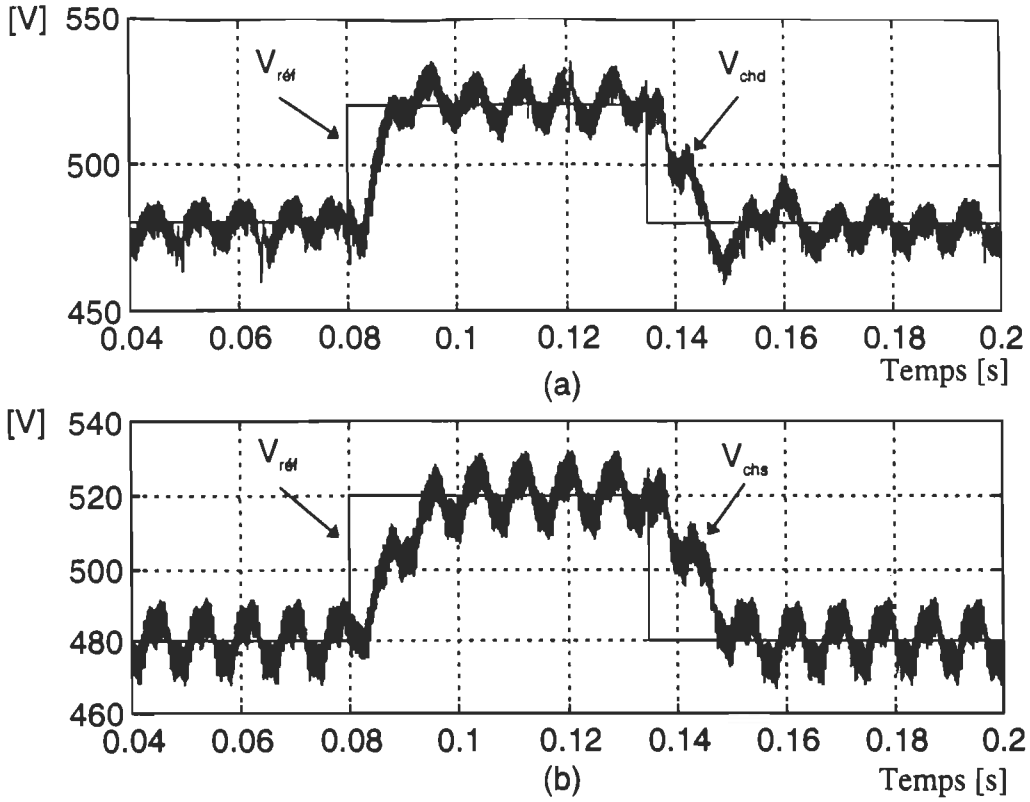


Figure 4.13: Réponse du système pour un échelon de 40V

a) régulateur dynamique : tensions $V_{réf}$ et V_{chd},

b) régulateur statique : tensions $V_{réf}$ et V_{chs},

Le régulateur dynamique doit garantir un facteur de puissance unitaire, la tension de commande ne doit pas dépasser 0.4V, pour cela il est obligatoire d'insérer un bloc de saturation entre le régulateur et le système, son inconvénient réside dans une perte au niveau du temps de réponse du système. La figure 4.13 montre la forme d'onde de la référence et la tension de sortie du montage ainsi que le courant et la tension du réseau multiplié par un facteur de 0.2. On peut remarquer une légère perte dans la qualité d'onde de courant et des ondulations dû au filtrage de la tension de sortie (voir la figure 4.14).

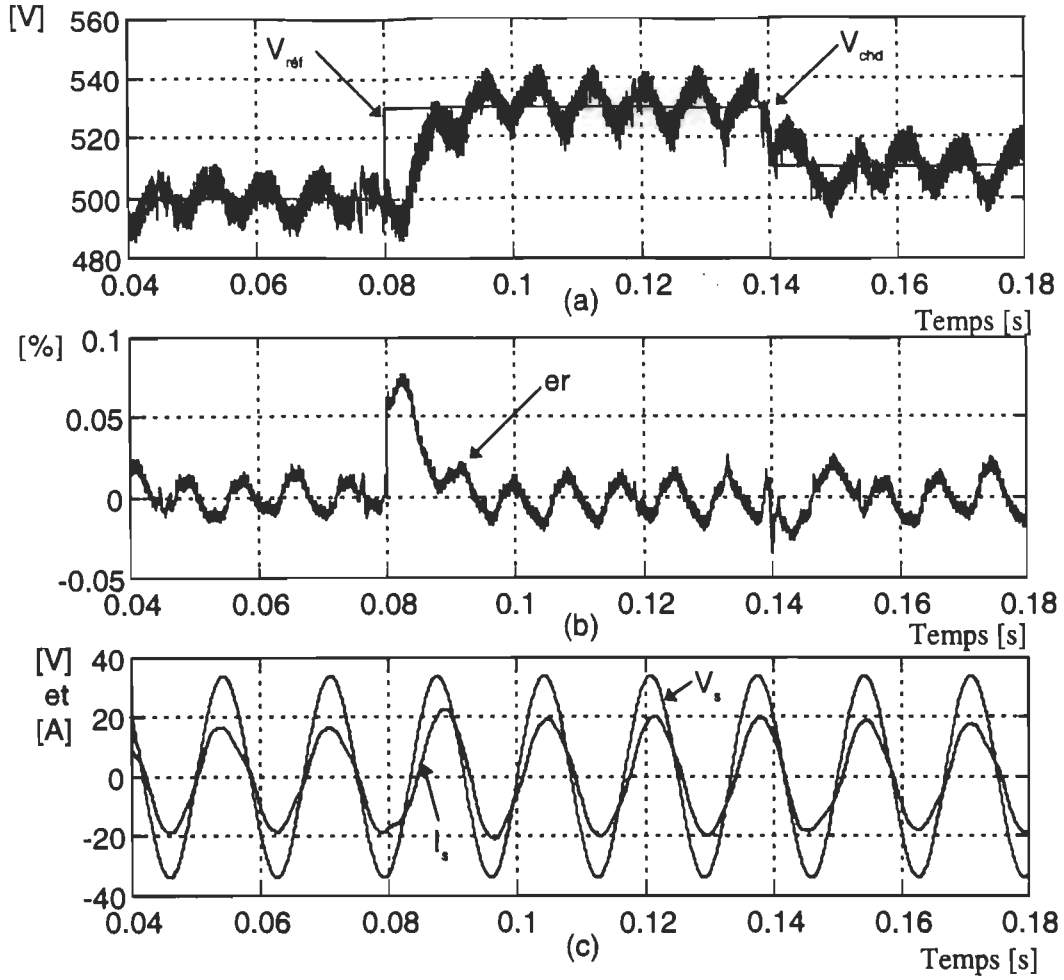


Figure 4.14: Résultats de simulation pour deux échelons de la référence du régulateur dynamique a) tensions $V_{réf}$ et V_{chd} b) erreur relative c) tension et Vs/500 et Is de source

4.6.3 Réponse du système à des perturbations de la source

Pour le régulateur dynamique, nous avons appliqué les pires conditions que peut rencontrer le montage dans les conditions réelles. La figure 4.15 montre la réponse du système pour deux perturbations de la source d'alimentation. Rappelant que ce régulateur a eu d'apprentissage qu'avec un modèle dont la source est de 120V. Nous allons montrer la réaction du régulateur dynamique à deux perturbations de la

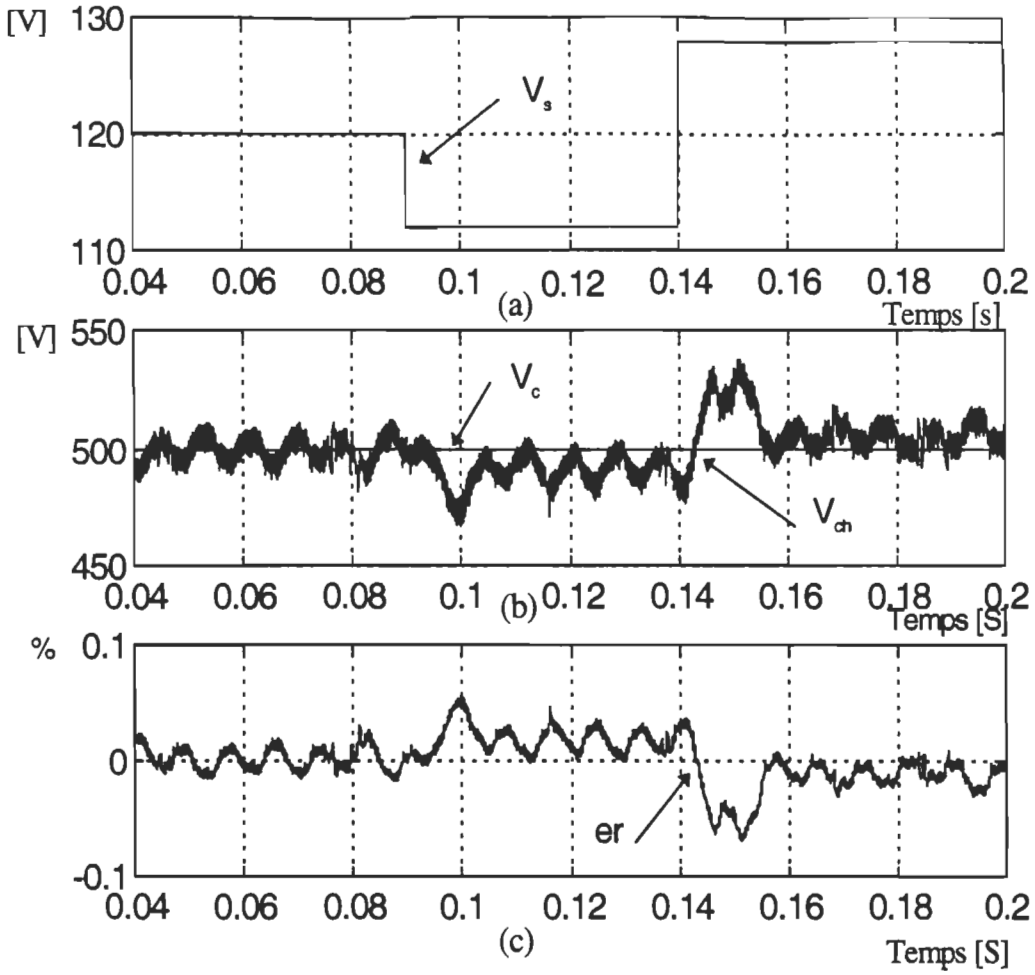


Figure 4.15: Résultats de simulation d'une perturbation de la tension source

a) tension de source, b) tensions de charge et c) de référence et l'erreur relative

tension de source V_s. L'erreur relative est calculée par rapport à la tension de charge V_{ch} nominale de 500V. Le premier échelon de 6.67% simule une chute de tension de la source passant de 120V à 112V et la deuxième échelon de 13.33% simule une augmentation de la source passant de 112V à 128V. Dans les deux cas, le régulateur dynamique a corrigé la tension de charge avec une erreur relative inférieure à 2%, la correction sur la tension de charge est bien visible pour le deuxième échelon en la ramenant à sa valeur de consigne.

4.6.4 Réponse du système à des perturbations de la charge

Dans ce cas, le système est soumis à deux perturbations de charge. La première est une augmentation de 20% de la charge, la seconde est diminution de 30%. Les résultats de simulation montrent que l'erreur relative moyenne est inférieure à 2%. Le régulateur dynamique corrige l'erreur due à la variation de la résistance de charge même si ce dernier n'a pas appris la variation de la charge lors de l'apprentissage (voir figure 4.16). Ce montage est généralement conçu pour fonctionner sous une charge fixe ou une faible variation de celle-ci. Le montage doit fonctionner toujours en régime discontinu pour garder le facteur de puissance presque unitaire.

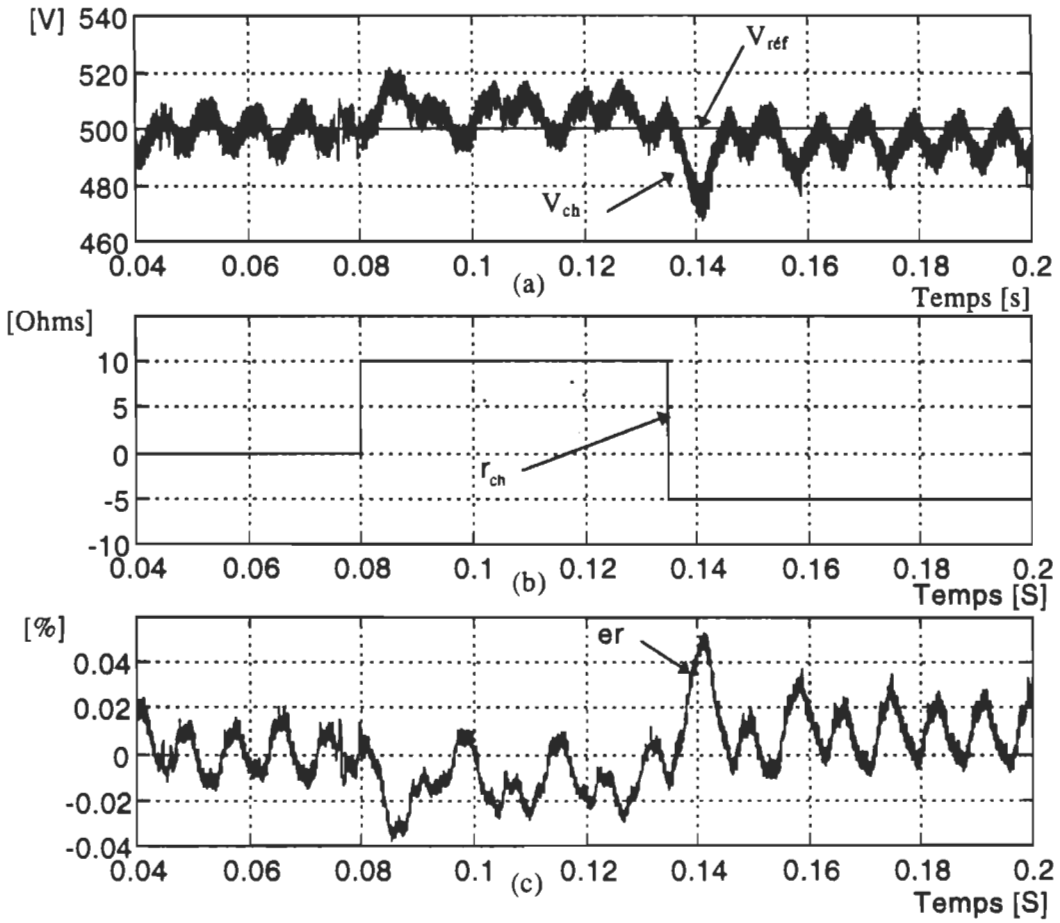


Figure 4.16: Résultats de perturbations de la charge: a) tension de charge et de référence b) r_{ch}: variation de R_{ch} et c) l'erreur relative

4.7 Conclusion

Dans ce chapitre nous avons montré la méthode de conception d'un régulateur dynamique à réseau de neurones artificiel. En électronique de puissance, la majorité des systèmes non linéaires sont peu connus du point de vue analytique. La meilleur façon de concevoir un régulateur c'est d'utiliser la méthode de contrôle indirect, cette dernière a besoin de deux réseaux de neurones Le premier permet à l'identification du modèle du système qui servira à l'apprentissage du second réseau, soit le régulateur. Une comparaison est faite entre les deux régulateurs statique et dynamique. Le régulateur statique garantie mieux un facteur de puissance unitaire par rapport au régulateur dynamique, par contre l'inconvénient du régulateur statique est que son temps de réponse est plus grande et qu'il ne garantie pas une erreur nulle. En conclusion, le régulateur dynamique assure une rapidité du système, une erreur nulle et une robustesse à toutes les perturbations.

CHAPITRE 5

EXIGENCE D'UNE MISE EN OEUVRE DANS UN PROCESSEUR NUMÉRIQUE

5.1 Introduction

Les systèmes numériques de traitement de l'information opèrent sur des nombres et font appel à un ordinateur ou un processeur numérique. Les signaux doivent subir une conversion analogique - numérique (A/N) pour être traités dans le processeur et une conversion numérique - analogique (N/A) pour émettre le résultat sous forme de signal analogique.

Les processeurs numériques de traitement des données ne peuvent manipuler que des nombres binaires (0 ou 1). Chaque signal est représenté par un nombre fini de valeurs. Nous retrouvons deux types de représentations, à virgule fixe et à virgule flottante. La présentation on virgule flottante permet une plus grande exactitude dans le traitement des données comparativement à la représentation en virgule fixe. Par contre, elle est coûteuse en temps de calcul et en surface d'intégration.

Des erreurs peuvent être néfastes pour la commande à cause de la quantification des signaux en valeurs numérique et les opérations arithmétiques dans les processeurs. Par conséquent, Il est intéressant de prévoir ces erreurs de calculs par simulation pour optimiser l'implantation des algorithmes [34-37].

Dans le domaine de l'électronique de puissance, la fréquence de commutation des convertisseurs est très élevée et souvent supérieure à 20 kHz. Ce qui oblige la commande à être rapide afin de fournir un temps de réponse adéquat surtout dans le cas des convertisseurs. Il devient donc nécessaire d'effectuer des calculs en virgule fixe pour une mise en oeuvre réaliste au niveau coût et performance.

Ce chapitre consiste d'une part à comprendre et simuler dans l'environnement de SIMULINK® les différentes méthodes de quantification des signaux d'autre part à prévoir les erreurs de calculs et d'optimiser la mise en oeuvre des algorithmes. Une comparaison sera effectuée entre un neurone quantifié en virgule fixe et une neurone en virgule flottante. Finalement l'influence du nombre de bits de la quantification du régulateur dynamique, proposé au chapitre 4 basé sur un réseau de neurones sera évaluée.

5.2 La quantification en virgule fixe

La quantification est une règle de correspondance entre le nombre infini des valeurs possibles du signal d'entrée x(t) et un nombre fini de valeurs assignées au signal de sortie x_q (t).

La représentation proprement dite est que chaque nombre doit être représenté par un mot binaire de **b** bits en laissant un bit pour le signe. La représentation en virgule fixe est une suite de valeurs de représentation équidistantes sur l'axe réel. Le nombre de points et la distance entre des points voisins sont arbitraires. En général, on choisit l'ensemble symétrique par rapport à l'origine. Le nombre de valeurs représentables détermine le nombre de bits nécessaire pour leur représentation. La dynamique des signaux numériques se définit dans l'intervalle [-1 ,+1].

La représentation par signe et par module: est la représentation la plus usuelle. Elle consiste à écrire le nombre sur b bits en réservant un bit pour le signe. Le module sera représenté sur les (b-1) bits restant. La figure 5.1 montre cette représentation. La valeur réelle de x devient une fois représentée [34]:

$$x_q = (-1)^{C_0} \sum_{i=1}^{b-1} C_i 2^{-i} \tag{5.1}$$

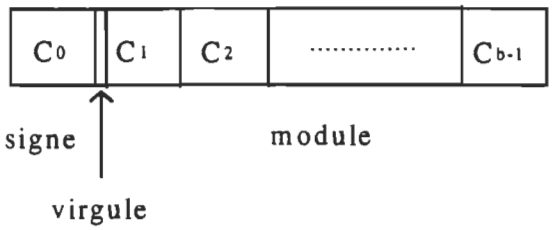


Figure 5.1: Représentation par signe et module

On remarque qu'on ne représente que n_i+1 n_i-1, par contre la valeur zéro est représentée à deux reprises. La dynamique est donc représentée par $\left[-1+2^{-(b-1)}, 1-2^{-(b-1)}\right]$.

Le pas de quantification définit la distance q entre deux valeurs quantifiées successives. Il est fonction du nombre de bits et dépend de la représentation $[X_{\max}, X_{\min}]$ suivante:

$$q = \frac{X\max - X\min}{2^b - 2} \tag{5.2}$$

La représentation par complément à 2: est largement utilisé pour une représentation numérique des signaux bipolaires. Il facilite en particulier les opérations arithmétiques d'addition et de soustraction de nombres négatifs. Si M est un nombre positif, le nombre négatif -M est représenté par le code binaire du complément à 2 de M défini par $M' = 2^b - M$.

$$x_q = C_0 + \sum_{i=1}^{b-1} C_i 2^{-i} \tag{5.3}$$

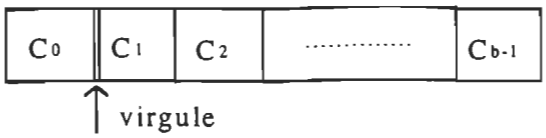


Figure 5.2: Représentation en virgule fixe par complément à 2

La valeur +1 reste sans représentation, par contre la valeur 0 possède une représentation unique, ce qui est un avantage certain. La dynamique exacte est donc $\left[-1,+1-2^{-(b-1)}\right]$ et le pas de quantification par complément à 2 devient:

$$q = \frac{X\max - X\min}{2^b - 1} \tag{5.4}$$

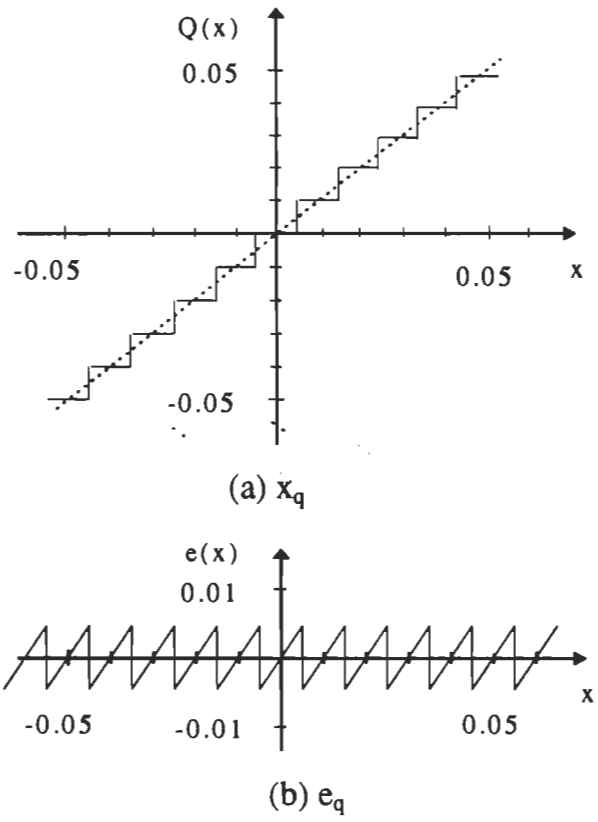


Figure 5.3: Caractéristique de quantification par arrondi:
a) x quantifié et b) erreur d'arrondi

La quantification la plus correcte consiste à choisir la valeur représentable la plus proche de la valeur donnée. À l'intérieur de la dynamique, on appelle cette prescription la loi de quantification par arrondi. Cette loi conduit à une erreur de quantification minimale. Dans ce cas l'erreur d'arrondi est bornée (voir figure 5.3).

$$|e(x)| = |x - xq| \leq \frac{q}{2} \qquad (5.5)$$

La quantification par troncature consiste à choisir la valeur représentable la plus proche de la valeur dont le module est inférieur à celui de la donnée. Cette méthode de quantification est appelée la troncature du module. Sa caractéristique est représentée à la figure 5.4.

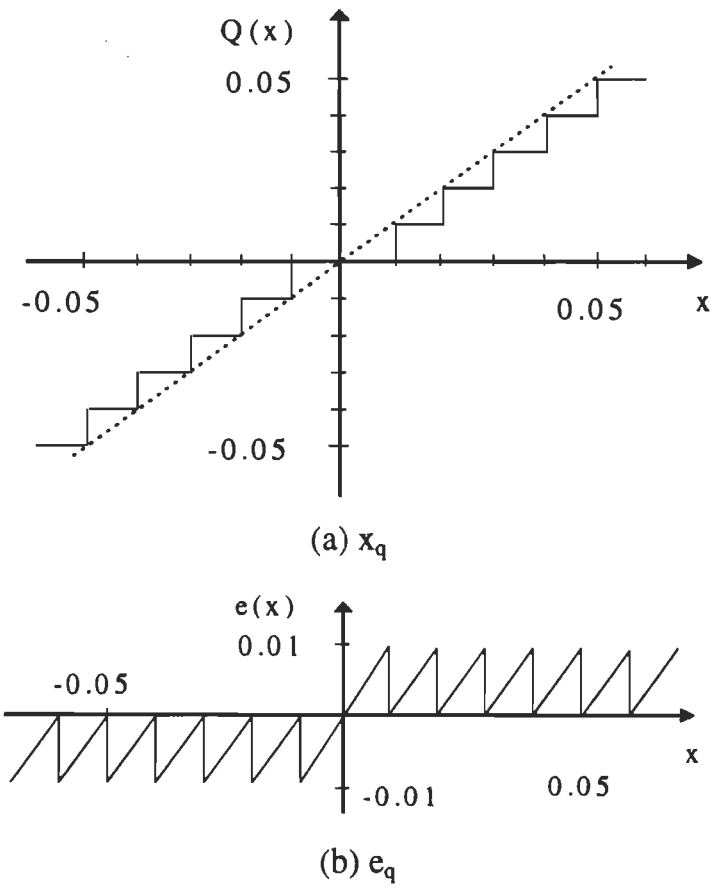


Figure 5.4 : Quantification par troncature du module

a) x quantifié et b) erreur de troncature

Les valeurs négatives sont représentées par la valeur la plus proche dont le module est supérieur à celui de la valeur donnée. La caractéristique de la quantification par troncature et par complément à 2 et l'erreur correspondante sont représentées à la figure 5.5.

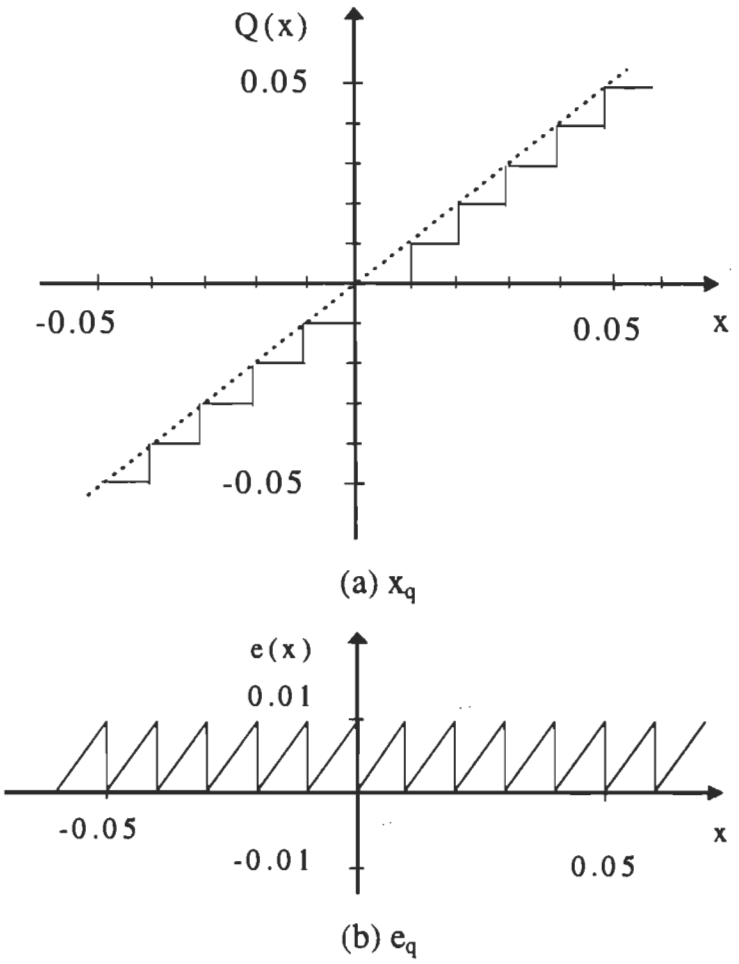


Figure 5.5: Quantification par troncature par complément à 2
a) x quantifié et b) son erreur

Les lois de quantification ne sont spécifiées de manière complète que si l'on indique [34]:

- le type de représentation (virgule fixe ou virgule flottante)

- la dynamique et /ou le pas de quantification (virgule fixe), ou le nombre de bits de l'exposant et la représentation des exposants négatifs (virgule flottante)
- le nombre de bits (virgule fixe) ou le nombre de bits de la mantisse (virgule flottante)
- la loi de représentation des nombres négatifs (signe et module ou complément à 2).
- la loi de quantification à l'intérieur de la dynamique (arrondi ou troncature)
- la loi de dépassement (saturation ou modulaire)

5.3 Réalisation des blocs de quantification

La quantification en virgule fixe permet d'obtenir quatre types de quantifications: la quantification arrondi par complément à 2, la quantification par troncature par complément à 2, la quantification arrondi par signe et module et la quantification par troncature par signe et module [38]. Le programme principal "digital.m" qui réalise la quantification arrondi par complément à 2 a été réalisé par D. Massicotte dans le logiciel MATLAB®. Ce programme a été modifié de telle façon à obtenir les quatre méthodes de quantification.

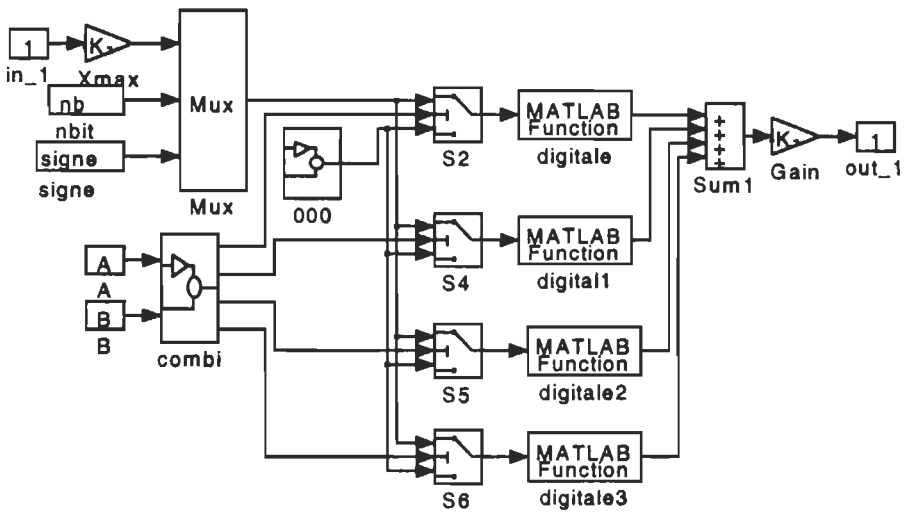


Figure 5.6: Schéma bloc de la quantification en virgule fixe

Le signal sera normalisé par rapport à sa valeur maximale, on doit choisir le nombre de bits, signé ou non et le couple A (0 ou 1) et B (0 ou 1) pour les différentes quantifications. À chaque couple A et B, on sélectionne avec une logique combinatoire le programme correspondant à la quantification désirée. Le schéma de la figure 5.6 est masquée pour donner un seul bloc appelé quantification. Pour diminuer le temps de simulation, chaque quantification a été réalisée dans un bloc à part présenté à la figure 5.7.

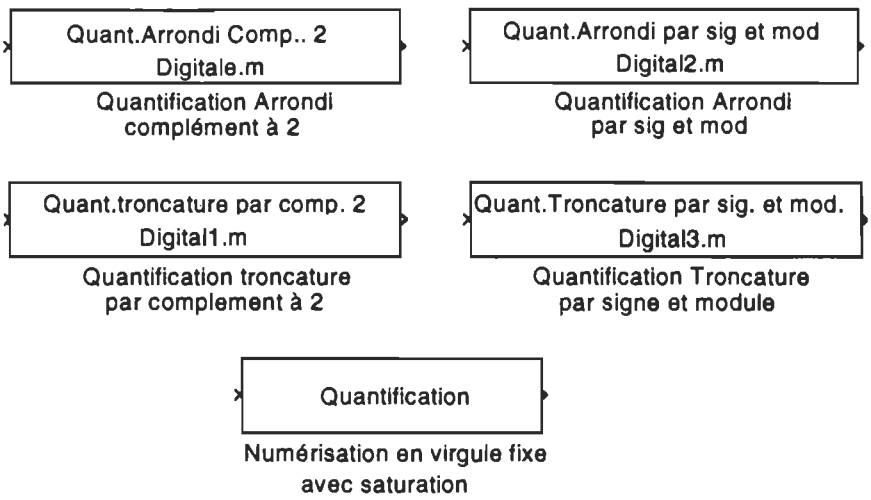


Figure 5.7: Différentes blocs de quantification

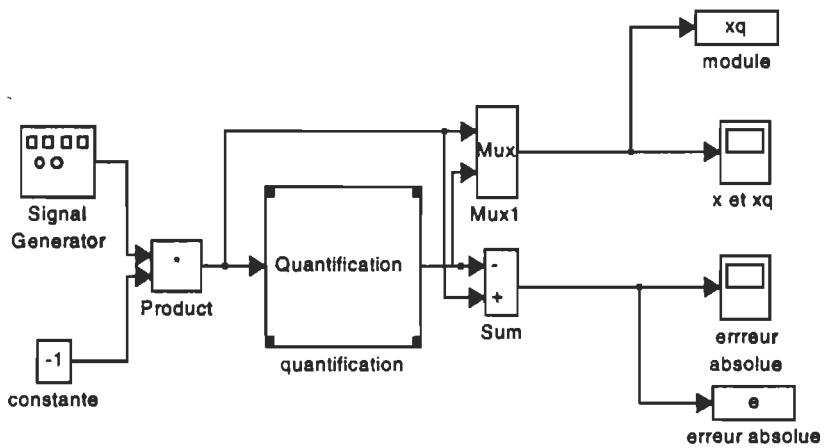


Figure 5.8: Schéma bloc du montage test

Test et validation des blocs de quantification

Le schéma bloc du montage test de la figure 5.8 permet de tester le bloc quantification. L'entrée du bloc est un signal analogique et la sortie est le signal quantifié. Ce bloc masqué nous donne la possibilité de choisir les configurations suivantes:

nb = le nombre de bits (supérieur à 2),

Signe = 1 signé ou 0 non signé,

Xmax = la valeur max. du signal,

A = 0 pour la représentation par signe et par module ou 1 pour la représentation par complément à 2,

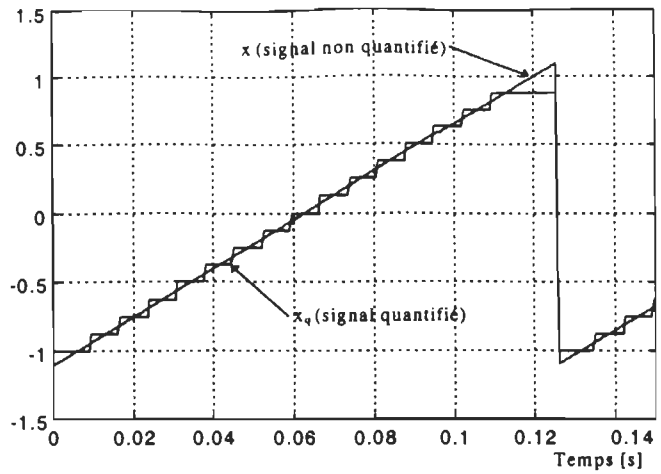
B = 0 pour une quantification par Troncature ou 1 pour une quantification par Arrondi.

La figure 5.9 montre les signaux réel et quantifié ainsi que l'erreur correspondante pour la quantification arrondi par complément à 2, les résultats sont satisfaisants et concordent avec la théorie.

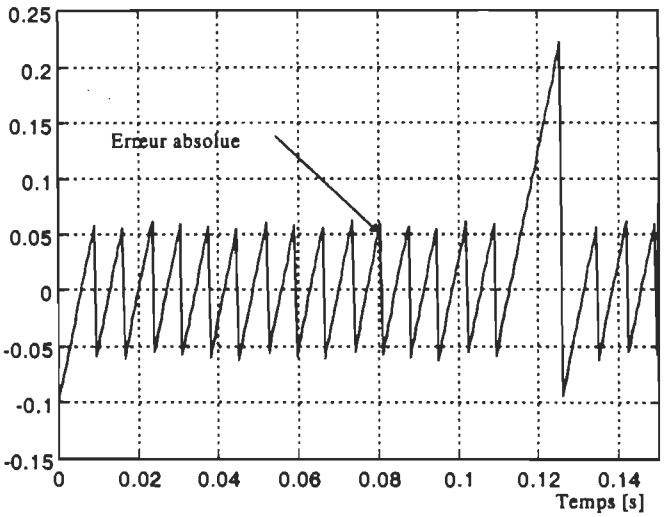
5.4 Résultat de la quantification d'un neurone

Nous allons montrer dans cet exemple l'influence du choix de nombre de bits sur l'erreur relative de la sortie d'un neurone quantifié par rapport à un neurone en virgule flottante.

Nous avons choisi les valeurs de poids positif et négatif, deux de ces poids sont très faibles et presque nuls. La quantification choisit est la quantification arrondi par signe et par module avec la valeur maximale du signal égale à Xmax =1. Le modèle du neurone en bloc SIMULINK® est à la figure 5.10.



(a)



(b)

Figure 5.9: Quantification Arrondi par complément à 2:
a) signaux réel et quantifié et b) l'erreur correspondante

À chaque opération effectuée, la valeur doit être quantifiée, les poids sont normalisés par rapport au poids maximal Wmax=10. La figure 5.11 montre le schéma bloc de la comparaison d'une neurone en virgule en virgule fixe (quantifié) et un neurone en virgule flottante, les signaux d'entrée pour les deux neurones sont

choisis arbitrairement trois sinusoïdes et deux constantes une négative et l'autre positive.

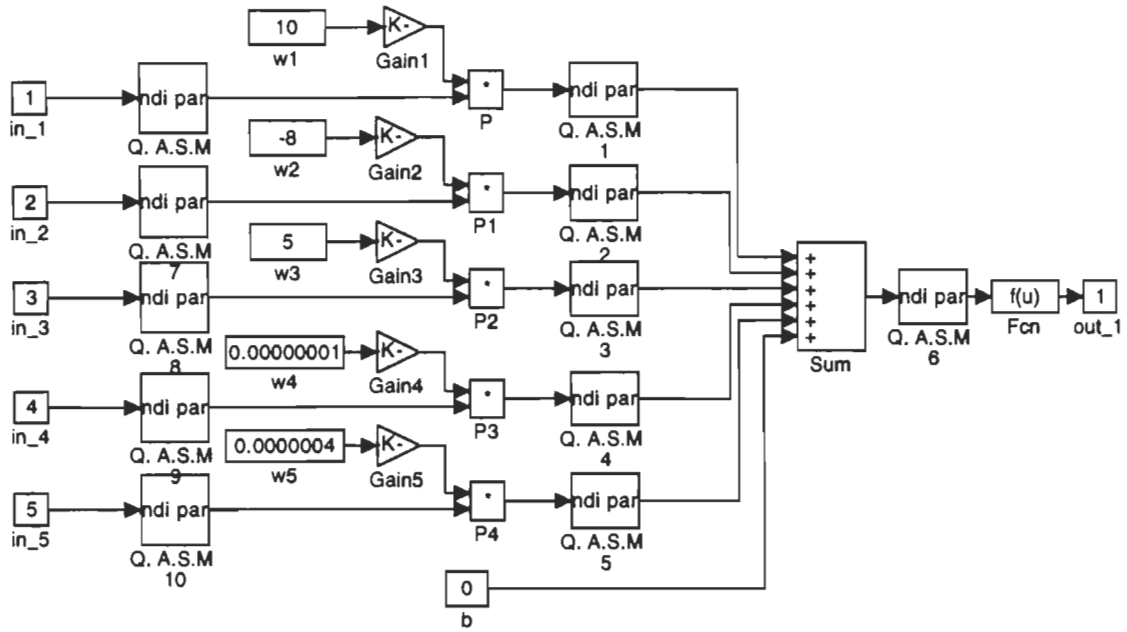


Figure 5.10: Modèle du neurone quantifié

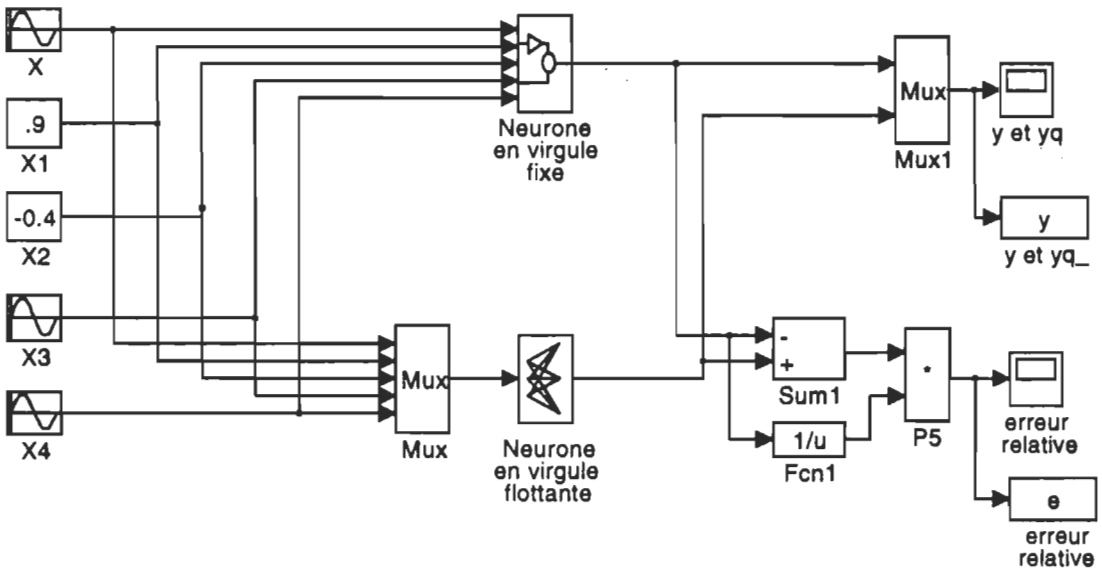


Figure 5.11: Schéma de comparaison entre un neurone en virgule fixe et un neurone en virgule flottante

Après simulation, on constate que l'erreur relative diminue quant le nombre de bits augmente (voir tableau 5.1). La sortie du neurone en virgule fixe Y_q est presque confondue avec la sortie du neurone en virgule flottante Y.

Tableau 5.1: Résultats de quantification d'un neurone

Nombre de bits	L'erreur relative
4	0.800
8	0.060
12	0.006

Les figures 5.12 et 5.13 montrent les deux sorties Yq (en escalier) et Y en virgule flottante et l'erreur relative correspondante pour un nombre de bits égal à 8.

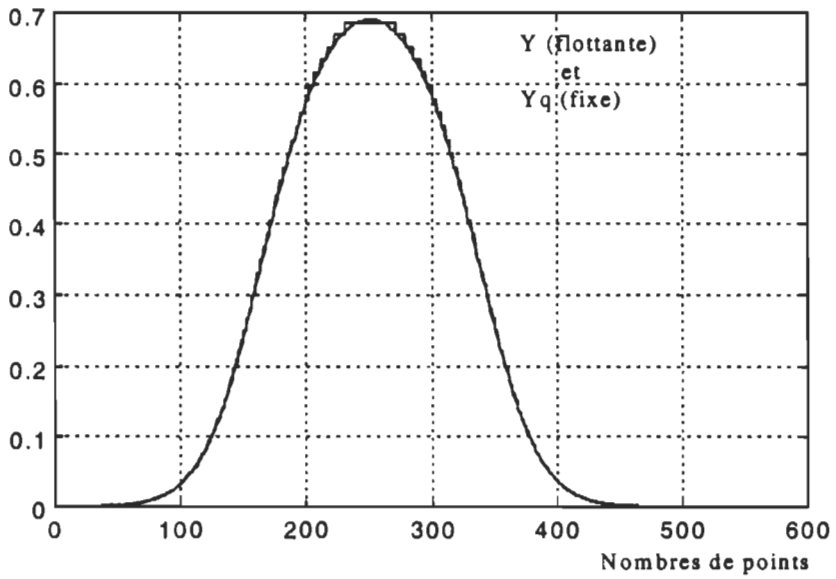


Figure 5.12: Réponse d'un neurone en virgule fixe Yq (en escaliers)
et d'un neurone en virgule flottante Y (en trait continu) pour Nb=8 et Xmax=1

5.5 Résultats de quantification du régulateur dynamique

Dans ce dernier aspect nous allons discuter l'effet de la quantification pour le régulateur dynamique (chapitre 4). Ce dernier consiste en un réseau de neurones à une couche cachée (7 neurones) dont la fonction d'activation est la tangente hyperbolique, la couche de sortie contient un neurone de type linéaire.

La méthode d'évaluation de l'erreur de quantification est la suivante: on propage l'entrée p qui contient les signaux réels échantillonnés provenant du convertisseur A/N soit la tension V_{ch} (à l'instant t-1), la tension de référence $V_{réf}$ et la tension de commande V_c (à l'instant t-1). À la sortie, on retrouve la tension de commande V_c (à l'instant t) pour un calcul en virgule fixe et en virgule flottante. La comparaison de ces deux valeurs détermine l'erreur introduite par la quantification. Si le niveau d'erreur n'est pas acceptable, une correction est apportée au nombre de bits utilisé. Un programme développé en MATLAB® a été utilisé pour effectuer la simulation numérique (voir ANNEXE E).

Tableau 5.2: Résultats de quantification du régulateur dynamique

Nombre de bits (Nbit)	erreur absolue max. e_q(V)	erreur relative max. à e_{rq}
8	0.15	0.360
12	0.008	0.020
16	0.0008	0.002

La simulation est faite pour un nombre de bits égal à 8, 12 et 16, la tension de commande est limitée entre 0.2 à 0.4V pour satisfaire le facteur de puissance unitaire. D'après le tableau 5.2, l'erreur relative diminue de 36% pour 8 bits à 0.2% pour 16 bits. La figure 5.14 montre les résultats de simulation de la quantification

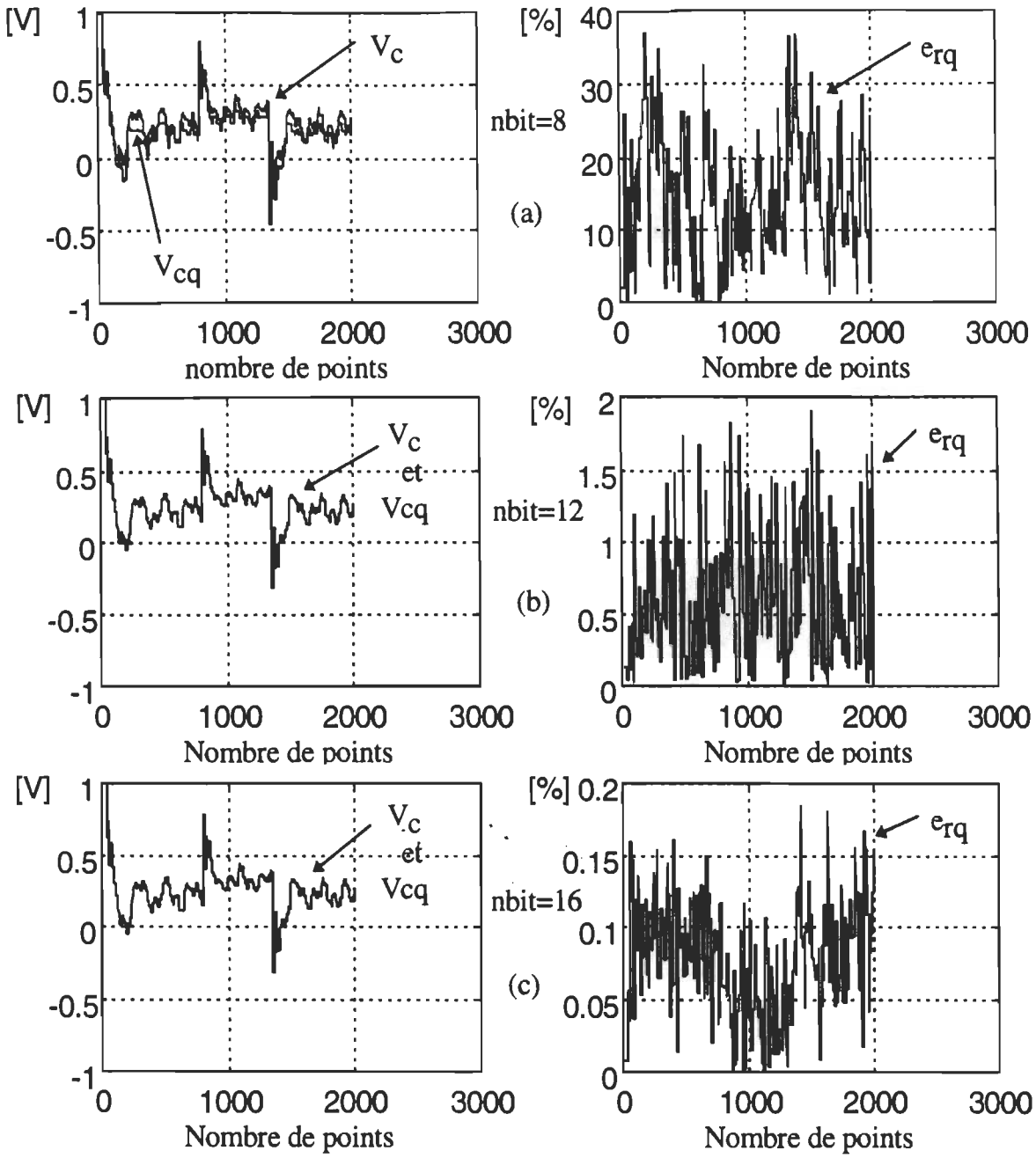


Figure 5.13: Résultats de simulation de la quantification du régulateur dynamique: tensions de commande quantifiée et non quantifiée et l'erreur relative, (a) 8 bits, (b)12 bits et 16 bits

pour le signal de commande V_c de la sortie du régulateur à réseau de neurones ainsi l'erreur relative de quantification. La figure 5.14 montre le résultat de simulation pour trois quantifications soit 8, 12 et 16 bits. Les courbes sont les tensions V_{cq} (virgule fixe) et V_c (virgule flottante) et l'erreur absolue e_q.

D'autres facteurs peuvent êtres pris en considération comme le temps d'exécution, la fréquence d'échantillonnage etc. Nous pouvons choisir une implantation à 16 bits car l'erreur sur le quatrième chiffre de la tension de commande n'influe pas la tension de charge.

5.6 Résumé des besoins pour la mise en oeuvre

Pour faire l'implantation du régulateur à réseau de neurones dans un processeur (voir figure 5.15), il faut déterminer le temps de calcul, la précision et la quantité de mémoire pour stocker les poids et la fonction d'activation. La complexité des calculs doit être étudiée, ce qui nous ramène à calculer le nombre d'opérations arithmétiques à effectuer. Dans le cas du régulateur dynamique étudie au chapitre 4 a besoin de 28 multiplications et 20 additions. Concernant la fonction d'activation, nous avons stocké les données de la fonction sous forme d'une table et si nécessaire appliqué une interpolation linéaire entre les données de la table afin d'augmenter la précision.

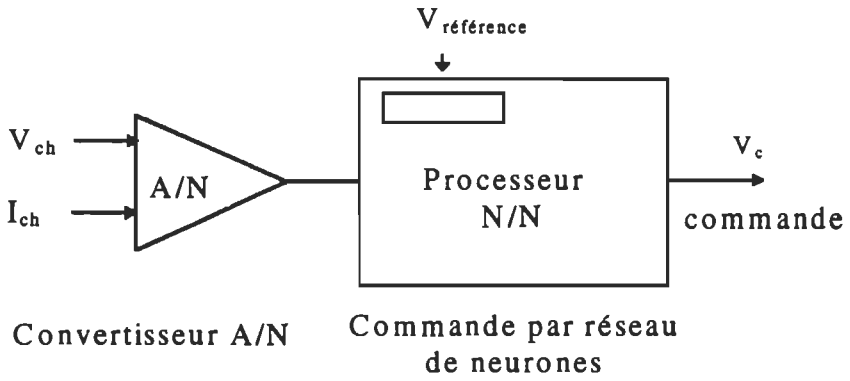


Figure 5.14:Schéma fonctionnel du traitement

L'effet de la quantification pour les montages avec régulateur statique à base de réseau de neurones du chapitre 3 a été réalise par M. A. Lassonde et un résultat convenable est obtenu avec une mise en oeuvre en virgule fixe sur 16 bits [37]. Le tableau 5.3 montre le nombre d'opérations et la quantité de mémoire qu'on en a besoin pour l'implantation de chaque régulateur.

Tableau 5.3: nombres d'opérations et quantité de mémoire:

(s) statique, (d) dynamique

Montage	multiplication	addition	mémoire
abaisseur (s)	60	44	76 + 128
survolteur (s)	45	29	61 + 128
Redresseur (s)	45	29	61 + 128
Redresseur (d)	28	20	36 + 128

Par exemple, pour le mise en oeuvre d'un régulateur dynamique, le choix du processeur doit se faire sur les besoins suivants:

*des mots de 16 bits, mémoires de 256 données de 16 bits,

*opérations arithmétique exécutées en virgule fixe arrondi par complément à 2 par saturation,

*pour un processeur avec une fréquence de fonctionnement de 10 MHz, le temps pour exécuter une opération arithmétique doit être au moins inférieure à $2.8\mu s$, soit $\frac{28 multiplications}{10 MHz}$

5.7 Conclusion

Pour implanter un algorithme dans un processeur, il faut connaître le type du quantification (Arrondi, ou Troncature), la représentation (Complément à 2 ou signe et module), le nombre de bits, signé ou non et la valeur de la saturation. Le nombre

de bits est le paramètre le plus important, car il peut fausser les calculs. Ce travail nous a permis de concevoir quatre blocs dans l'environnement SIMULINK®, des différentes quantifications pour une représentation en virgule fixe. Ces blocks seront insérés après chaque opération arithmétique pour prévoir les erreurs de quantification du calcul du processeur. Une quantification est faite sur un neurone permet le choix de nombre de bits à choisir et ainsi pour le régulateur dynamique. Enfin une mise en oeuvre de l'implantation celui-ci conçu dans le chapitre 4 a été faite.

CONCLUSION GÉNÉRALE

Dans ce travail de recherche, nous avons étudié trois convertisseurs: un convertisseur de type CC-CC appelé abaisseur, un convertisseur de même type appelé survolteur et un convertisseur de type CA-CC qui est appelé redresseur triphasé survolteur mono-interrupteur à facteur de puissance unitaire. Ces convertisseurs de puissance sont en général des systèmes fortement non linéaires, ce qui rend plus difficile la conception des régulateurs par les méthodes classiques. L'objectif de ce travail a été de montrer l'avantage de l'application des réseaux de neurones pour ces convertisseurs.

La contribution dans cette approche, vient des nouveaux points apportés de sorte que ces trois montages sont modélisés de façon à obtenir des montages proches de la réalité.

Deux types de régulateurs à base des réseaux de neurones artificiels sont utilisés pour assurer à la fois le réglage de la tension de charge et le facteur de puissance. Le premier est un régulateur statique et le second est un régulateur dynamique. Tous les deux sont basés sur le principe du modèle inverse du système à régler.

Le régulateur statique est appliqué pour les trois convertisseurs. L'apprentissage de ce régulateur est effectué en régime permanent. Le principe de la conception est de perturber le système et de mesurer la réponse en régime permanent en le considérant comme une boîte noire avec des entrées/sorties. Les résultats obtenus montrent que ce type de régulateur assure une commande pour chaque consigne imposée. Une robustesse est garantie pour les perturbations apprises. La précision de ce régulateur dépend de la qualité des données d'apprentissage. En particulier, il assure un facteur de puissance unitaire pour le redresseur survolteur.

Une protection est faite sur le montage abaisseur limitant le courant à 1 A. Cette dernière est assurée par le même régulateur à réseau de neurones. Cette méthode de

protection pourrait être généralisée pour tous les convertisseurs de ce type en ajoutant une banque de valeurs de la protection aux données d'apprentissage du régulateur.

Le régulateur dynamique est basé sur le contrôle indirecte du système, c'est-à-dire que celui-ci a besoin d'un modèle en réseau de neurones du système servant à l'entraînement du régulateur. Cette méthode a été utilisée pour la conception du régulateur pour le montage redresseur triphasé survolteur à facteur de puissance unitaire. Ce régulateur est plus robuste que le régulateur statique vis-à-vis des perturbations (sans l'apprentissage de ces dernières) et donne une amélioration du temps de réponse et une erreur en régime permanent plus faible par rapport au régulateur statique. Par contre un régulateur dynamique ne peut assurer un facteur de puissance meilleur que le régulateur statique. Cette méthode de conception pourrait être généralisée pour tous les convertisseurs de ce type.

Les blocs de quantification sont réalisés sous MATLAB®/SIMULINK®. Ceux-ci permettent d'étudier l'effet de la quantification des réseaux de neurones. Les besoins pour la mise en oeuvre des régulateurs à base des réseaux de neurones dans un processeur numérique: nombre de bits nécessaire pour atteindre la précision désirée en pratique, le type de quantification, le type de présentation, la quantité de mémoires pour stocker les données et la vitesse de calcul pour exécuter l'algorithme.

Les avantages de l'application des réseaux de neurones artificiels (RNA) pour les corrections de facteur de puissance sont: la non-nécessité de la fonction de transfert du système à étudier ou un calcul préalable, la généralisation de la commande même pour certaines des situations non apprises dans la bande d'apprentissage, la simplicité de l'implantation dans les processeurs numériques.

Quelques précautions doivent être prises lors de la conception d'un régulateur à réseau de neurones pour une tâche donnée:

-le filtrage des signaux d'apprentissages pour éviter l'apprentissage de la constante de temps due à la commutation.

-la bande d'apprentissage doit être suffisamment riche pour avoir une bonne généralisation,

-les données doivent refléter la dynamique du système pour obtenir une identification correcte du système à régler.

Ce travail peut donner des perspectives pour l'enrichissement de la bibliothèque de quantification, l'amélioration de la protection par les réseaux de neurones ainsi que la réalisation pratique de ces régulateurs.

BIBLIOGRAPHIE

[1] R. Bausière, F. Labrique, G. Seguier, "Les Convertisseurs de l'Électronique de Puissance" La conversion continu - continu, Volume 3 Technique et Documentation (Lavoisier), Paris 1987

[2] S. S. Ang, "Power-Switching Converters", by Marcel Dekker, inc. USA, 1995.

[3] A. Ba-razzouk, A. Pittet, V. Rajagopalan, "Conception Assistée par Ordinateur des Systèmes Électronique de Puissance, le Logiciel SIMUSEC," Rapport interne, Groupe de recherche en Électronique industrielle, Mai 1993.

[4] J. W. Kolar, H. Ertl, F. C. Zach, "A Comprehensive Design a Three-Phase High-Frequency Single-Switch Discontinuous-Mode Boost Power Facteur Correction Based on Analyticaly Derived Normalised Converter Compenent Rating," IEEE Conf.IAS'93, Vol.2, pp 931-938.

[5] J. W. Kolar, H. Ertl, F. C. Zach, "Space Vector - Based Analytical Analysis of the input Current Distortion of a Three-Phase Discontinuous-Mode Boost Rectifier System," IEEE Trans. On P. E. Vol. 10, No. 6, Nov. 1995, pp733-745.

[6] E. H. Ismail, R. Erickson, "Single-Switch 3Ø PWM Low Harmonic Rectifiers," IEEE Trans. On Power Electronics, Vol. 11, No. 2, March 1996, pp 338-346.

[7] E. N. Marieb, "Anatomie et Physiologie Humaines" édictions du Renouveau Pédagogique Inc. Canada 1993.

[8] P. Davalo, "Des Réseaux de Neurones " Éditions Eyrolles, Paris 1989.

[9] J. F. Jodouin, "Les Réseaux de Neurones" Principes et définitions Hernés, Paris, 1994.

[10] M. H.Hassoun, "Fundamentals of Artificial Neural Networks" MIT, USA, 1995.

[11] S.Haykon, "Neural Networks" A comprehensive Foundation , Macmillan college Publishing Company Newyork,

[12] J. F. Jodouin, "Les Réseaux Neuromimétriques Principes et définitions" Hernés, Paris, 1994.

[13] Y. Kamp, M. Hasker, "Réseaux de Neurones Récursifs pour Mémoires Associatives" Presses Polytechniques et universitaires romondes, Lausanne, Suisse, 1990.

[14] H. Demuth, M. Beale "Neural Network Toolbox" The MATHWORKS inc. january 1994.

[15] P. S. Neelakanta et D. F. de Groff "Neural Network Modeling Satatistic mechanics and cybernectic perspectives" CRC press, Inc. USA 1994.

[16] M. T. Hagan et M. B. Menhaj, "Training Feedforward Neural Networks With the Marquardt Algorithm" IEEE Transactions on Neural Networks, Vol. 5, NO. 6, November 1994, pp 989-993.

[17] M. H. Kim, M.G. Simoes, B.K. Bose, "Neural Network-Based Estimation of Power Electronic Waveforms" IEEE Transactions on Power Electronics, Vol. 11, NO. 2 March 1996, pp 383-389.

[18] H-C. Chan, K. T. Chau, C. C. Chan, "A Neural Network Controller for Switching Power Converter" IEEE PESC'93, pp 887-892.

[19] D.S.L Simonetti, G. Bevilacqua, P. Mejia et J. Uceda, "A Neural Control of Power Factor Preregulators" IEEE IECON 1993, pp 971-976.

[20] Allan Insleay, Navid R. Zargari, Géza Joos, "A Neural Network Controlled Unity Power Factor Three Phase PWM Rectifier" IEEE PESC'94, pp 577-582.

[21] M. A. El-Sharkawi, "Neural Network Application to High Performance Electrical Drives Systems" IEEE IECON 1995, pp 44-49.

[22] J. Bates, M. E. Elbuluk, D. S.Zinger, "A Neural Network Control of a Chopper-Fed DC Motor" IEEE PESC'93, pp 893-899.

[23] J. B. Patton, "Brushless DC Motor Control Using a General Regression Neural Network" IEEE IAS'95, pp 1422-1427.

[24] K. S. Narendra, K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks" IEEE Transactions on Neural Networks, Vol. 1, NO. 1, March 1990, pp 4-27.

[25] A. U. Levin, K. S. Narendra, "Control of Nonlinear Dynamical Systems Using Neural Networks-Part II: Obsevability, Identification, and Control" IEEE Transactions on Neural Networks, Vol. 7, NO. 1, January 1996, pp 30-42.

[26] J. G. Kuschewski, S. Hui, S. H. Zak, "Application of Feedforward Neural Networks to Dynamical System Identification and Control" IEEE Transactions on Control, Systems Technology, Vol. 1, NO. 1, March 1993, pp 37-49.

[27] S. I. Mistry, S-L Chang, and S. S. Nair "Idirect Control of a class of Nonlinear Dynamical Systems" IEEE Transactions on Neural Networks, Vol. 7, NO. 4, July 1996, pp 1015-1023.

[28] A. M. S. Zalzala and A. S. Morris "Neural Network for Robotic Control: Theory and Application," Ellis Horwood, Great Britain 1996, pp 989-993.

[29] P. J. Werbos " Backprobagation Through Time," Proceeding IEEE,Vol. 78, No. 10, October 1990, pp 1555-1560.

[30] M. Norgaard " Neural Network Based Identification Toolbox," Tech. Report. 95E-773, Department of Automation, Technical University of Denmark 1995.

[31] M. Norgaard "Neural Network Based Control System Design Toolbox," Tech. Report. 96E-830, Department of Automation, Technical University of Denmark 1996.

[32] L. Ljung "System Identification Toolbox User's Guide" The mathworks Inc. May 1995.

[33] L. Ljung "Control System Toolbox User's Guide" The mathworks Inc. May 1995.

[34] M. Kunt, "Techniques Modernes de Traitement Numérique des Signaux" collection électricité, Volume 1, Lausanne 1991.

[35] R. L. Tokheim, "Les Microprocesseurs 1", cours et problèmes, Paris,1984.

[36] F. de Coulon, "Théorie et Traitement des Signaux", Traité d'électricité, Dunod, 1984.

[37] J. S. Alarie, "Application des Réseaux de Neurones au Filtre de Kalman: programmation dans le DSP56000", projet d'activités de synthése, UQTR, Avril 1996.

[38] A. Meftah, "Réalisation des Blocs de Quantification des Nombres en Virgule Fixe" mini-projet cours GEI6021, UQTR, 1996.

ANNEXES

ANNEXE A

Listage du fichier d'apprentissage du régulateur statique pour le montage survolteur

```
% Algorithme d'apprentissage pour le régulateur statique
% Montage survolteur
% Meftah Abdelkrim

clear
clg

% chargement de la banque d'apprentissage
load c:\meftah\projet\data.mat

% chargement des poids de la dernière itération
load c:\meftah\projet\Weight.mat


AP(:,2)=AP(:,2)/max(AP(:,2));     % Apprentissage
AP(:,3)=AP(:,3)/max(AP(:,3));     % Apprentissage

G(:,2)=G(:,2)/max(G(:,2)); % Généralisation
G(:,3)=G(:,3)/max(G(:,3)); % Généralisation

P= [AP(:,2) AP(:,3)]';            % Apprentissage
T= [AP(:,1)]';                    % Apprentissage

PG= [G(:,2) G(:,3)]';             % Généralisation
TG= [G(:,1)]';                    % Généralisation

% choix de nombre de neurones
R=2;                              % Nbre d'entrées
S1=15;                            % Nbre de neurones cachés
S2=1;                             % Nbre de neuronss de sortie

%initialisation des poids
%[W1,B1] = nwlog(S1,R);
%[W2,B2] = nwlog(S2,S1);

%les paramètres de l'algorithme trainlm
        TP(1) =5;                 % Affichage
        TP(2) =100;               % Maximum number of epochs to train
        TP(3) =1e-7;              % Sum-squared error goal
        TP(4) =1e-5;              % Minimum gradient
```

```
TP(5) =0.001;          % Initial value for MU
TP(6) =10;             % Multiplier for increasing MU
TP(7) =0.0001;         % Multiplier for decreasing MU
TP(8) =1e10;           % Maximum value for MU

[W1,B1,W2,B2,TE,TR] = trainlm(W1,B1,'logsig',W2,B2,'logsig',P,T,TP);

% Sortie avec les données apprises
A=simuff(P,W1,B1,'logsig',W2,B2,'logsig');
% Sortie avec les données non-apprises
AG=simuff(PG,W1,B1,'logsig',W2,B2,'logsig');

save c:\meftah\projet\Weight.mat W1 W2 B1 B2

clg;

subplot(211)
t=1:length(A);
plot(t,T,t,A)
title('Test avec les données apprises')

subplot(212)
t=1:length(AG);
plot(t,TG,t,AG)
title('Test avec les données non-apprises')
```

ANNEXE B

Listage du fichier de Filtrage des données

```
% filtrage des données pour le l'apprentissage du modèle du système
% Meftah Abdelkrim

        %%%% acquisition des données de simulation %%%%
        %% i j k l m n
i=demuu;
n=dem17;
        %%% Calcul de la longueur du vecteur %%%
s=length(i)
ss=length(n)
        %%% enlevement du transitoire %%%
i=i(35:s,:);
n=n(35:ss,:);
        %%% préparation de la matrice d'apprentissage et de généralisation %%%
        %%% [vs vc vch vs(-1) vc(-1) vch(-1)] %%%

%A=[i;j;k;l;m];
A=[i];
B=[n];
a=A(:,3);      % Vch
aa=B(:,3);     % vch

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%% Filtrage numérique d'un signal %%%%%%%%%%%%
%  a= signal 1 a filtré
%  b= point de coupure ( le choisir avant l'exécution du programme)
%  c= la longueur du signal ou size de a

%  aa= signal 2 a filtré
%  bb= point de coupure ( le choisir avant l'exécution du programme)
%  cc= la longueur du signal ou size de aa
x=a;
xx=aa;
c=length(x)
cc=length(xx)
y=fft(x);
yy=fft(xx);
plot(sqrt(real(y).*real(y)+imag(y).*imag(y)));grid;zoom
```

```
pause
plot(sqrt(real(yy).*real(yy)+imag(yy).*imag(yy)));grid
pause
y1=([y(1:b);zeros((c-2*b)-1,1);y(c-b:c)]);
yy1=([yy(1:bb);zeros((cc-2*bb)-1,1);yy(cc-bb:cc)]);
x1=ifft(y1);
xx1=ifft(yy1);
t=1:1:c;
tt=1:1:cc;
plot(t,real(x1),t,x);grid
pause
plot(tt,real(xx1),tt,xx);grid
pause
af=real(x1);                    %signal filtré
aaf=real(xx1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A(:,3)=af;
B(:,3)=aaf;

A=[A(:,1) A(:,2) A(:,3)];
B=[B(:,1) B(:,2) B(:,3)];

AA=A';
BB=B';
        %%%  préparation de la matrice delay z=-1  %%%
AA=delaysig(AA,1,1);
BB=delaysig(BB,1,1);
% matrice cologne z=-1

AA=AA';
BB=BB';
        %%%  préparation de la matrice finale  %%%
AP=[A AA];
G=[B BB];

sap=length(AP)
sg=length(G)

AAPP=AP(2:sap-5,:);
```

```
GG=G(2:sg-5,:);
plot(AAPP(:,3));grid
pause
plot(GG(:,3));grid
```

ANNEXE C

Listage du programme d'apprentissage et de validation du modèle NNARX pour le redresseur survolteur mono-interrupteur

```
% conception du modèle NNARX du redresseur en réseau de neurone
%Meftah Abdelkrim

close all

load c:\meftah\Nnctrl\expmr3c.mat;

Y=Y/600;
 N2=length(U);
 N1=floor(N2/2);

 y1 = Y(1:N1)';
  u1 = U(1:N1)';
 y2 = Y(N1+1:N2)';
 u2 = U(N1+1:N2)';

 subplot(211)
 plot(u1);
xlabel('Temps [ms]')
title('Vc et Vch [V]')
 subplot(212)
 plot(y1);
xlabel('Temps [ms]')
 % sélection de la structure du modèle

 W1f=rand(5,4);
 W2f=rand(1,6);

 drawnet(W1f,W2f,eps,['y(t-1)';'u(t-1)';'u(t-2)'],'yhat(t)');

% paramètres d'entraînement
 maxiter = 1000;
 stop_crit=1e-6;
 lambda=0.01;
 D=1e-4;
 NetDeff = ['HHHHH'
         'L----'];
 NN = [1 2 1];

 trparms=[maxiter stop_crit lambda D];
```

```
[W1f,W2f,NSSEvec,iteration,lambda]=nnarx(NetDeff,NN,W1f,W2f,trparms,y1,u1);

save mforward NN NetDeff W1f W2f

 % Validation du modèle NNARX du système

 [Yhat,NSSE]=nnvalid('nnarx',NetDeff,NN,W1f,W2f,y2,u2);

 drawnow
end
```

ANNEXE D

Listage du programme d'apprentissage et validation du régulateur dynamique

```
% OPTIMAL CONTROL
% régulateur dynamique
% Meftah abdelkrim

% chargement du modèle du système
 load mforward
% L'archeticture du régulateur
drawnet(W1f,W2f,eps,['y(t-1)';'u(t-1)';'u(t-2)'],'yhat(t)');
% L'archeticture du régulateur dynamique
drawnet(ones(7,4),ones(1,8),eps,['r(t+1)';'  y(t)';'u(t-1)'],'u(t)');

% Entraînement du régulateur dynamique
  moptrain
  save moptctrl W1c W2c NetDefc NN

% Validation du régulateur dynamique
  moptcon
  close(2)
  subplot(411)
  plot([0:samples-1],[ref_data*600 y_data*600]); grid
 %axis([0 samples -2 2])
  axis([0 samples min(480) max(530)])
  title('Reference and output signal')
  subplot(412)
  plot([0:samples-1],u_data); grid
  axis([0 samples 0.15 0.45])
 %axis([0 samples min(u_data) max(u_data)])
```

```
title('Control signal')
xlabel('time (samples)')
drawnow
end
```

ANNEXE E

Listage du programme de quantification d'un réseau de neurones artificiels (régulateur dynamique)

```
% Quantification d'un réseau de neurones artificiel
% Nbit= 8, 12 et 16 bit
% 1er couche tanh , 2eme couche lin
% Abdelkrim Meftah  le 30/.03/97 à UQTR
% ce programme fait appelle à mresq et qdegital

load dataq
p=p';

%%%%%%  CALCUL D'ERREUR POUR NBIT=8 %%%%%%%%%%%%

% calcul de la sortie du réseau pour une
% quantification à virgule fixe ( tension de commande )

   Nb_bit=8;
   dq=mres2q(p,w1,b1,w2,b2,Nb_bit);

% calcul de la sortie du réseau pour
% virgule flottante ( tension de commande )

   Nb_bit=inf;
   dqinf=mres2q(p,w1,b1,w2,b2,Nb_bit);

subplot(321)
plot(dq);grid;hold
plot(dqinf)
title('Vcq nbit=8 et Vc [V]')
xlabel('nombre de points')
subplot(322)
plot(abs(dq-dqinf)*100/0.4);grid
title(' eq en [V], nbit=8')
xlabel('Nombre de points')

%%%%%%%%%%  CALCUL D'ERREUR POUR NBIT=12  %%%%%%%%%%

% calcul de la sortie du réseau pour une
% quantification à virgule fixe (tension de commande)

   Nb_bit=12;
   dq=mres2q(p,w1,b1,w2,b2,Nb_bit);
```

```
% calcul de la sortie du réseau pour
% virgule flottante ( tension de commande )

   Nb_bit=inf;
   dqinf=mres2q(p,w1,b1,w2,b2,Nb_bit);

subplot(323)
plot(dq);grid;hold
plot(dqinf)
title('Vcq nbit=12 et Vc [V]')
xlabel('Nombre de points')
subplot(324)
plot(abs(dq-dqinf)*100/0.4);grid
title(' eq en [V],nbit=12')
xlabel('Nombre de points')

%%%%%%%%%  CALCUL D'ERREUR POUR NBIT=16  %%%%%%%%%

% calcul de la sortie du réseau pour une
% quantification à virgule fixe (tension de commande)

   Nb_bit=16;
   dq=mres2q(p,w1,b1,w2,b2,Nb_bit);

% calcul de la sortie du réseau pour
% virgule flottante ( tension de commande )

   Nb_bit=inf;
   dqinf=mres2q(p,w1,b1,w2,b2,Nb_bit);

subplot(325)
plot(dq);grid;hold
plot(dqinf)
title('Vcq nbit=16 et Vc [V]')
xlabel('Nombre de points')
subplot(326)
plot(abs(dq-dqinf)*100/0.4);grid
title(' eq en [V],nbit=16')
xlabel('Nombre de points')
```