Titre: Title:	Asservissement en niveau et température d'un réservoir d'eau avec un contrôleur neuronal
Auteurs: Authors:	Angel Ruiz, & Ernesto Cornieles
Date:	1998
Type:	Rapport / Report
Référence: Citation:	Ruiz, A., & Cornieles, E. (1998). Asservissement en niveau et température d'un réservoir d'eau avec un contrôleur neuronal. (Technical Report n° EPM-RT-98-01). https://publications.polymtl.ca/9557/

Institution:	École Polytechnique de Montréal
Numéro de rapport: Report number:	EPM-RT-98-01
URL officiel: Official URL:	
Mention légale: Legal notice:	

Département de Génie Électrique
et de Génie Informatique

Section Automation et Systèmes

ASSERVISSEMENT EN NIVEAU ET TEMPÉRATURE D'UN RÉSERVOIR D'EAU AVEC UN CONTRÔLEUR NEURONAL

Par

Angel Ruiz
Ernesto Cornieles

projet dirigé par

Professeur Chahé Nerguizian

École polytechnique de Montréal
Mars 1998

Dépôt légal, Mars 1998
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Asservissement en niveau et température d'un
réservoir d'eau avec un contrôleur neuronal

Angel Ruiz
Ernesto Cornieles

Projet dirigé par Professeur Chahé Nerguizian

Département de Génie Électrique et
Génie Informatique, Section Automation et Systèmes

Pour se procurer une copie de ce document, s'adresser au:

Service des Éditions
École Polytechnique de Montréal
Case Postale 6079, Succursale Centre-Ville
Montréal (Québec) H3C 3A7
Téléphone: (514) 340-4711 ext. 4473
Télécopie: (514) 340-3734

Compter 0,10$ par page et ajouter 3,00$ pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat-poste au nom de l'École Polytechnique de Montréal.

Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canadiens.

ASSERVISSEMENT EN NIVEAU ET TEMPERATURE D'UN RESERVOIR D'EAU AVEC UN CONTROLEUR NEURONAL

Angel Ruiz, Ernesto Cornieles
Département de génie électrique et de génie informatique
Section Automation et Systèmes
Ecole Polytechnique de Montréal

SOMMAIRE

Ce document décrit la conception et le développement d'un contrôleur utilisant les réseaux de neurones artificielles (RNA) qui a pour fonction de commander par ordinateur le niveau et la température d'un réservoir d'eau. Un banc d'essai a été utilisé pour évaluer d'une façon expérimentale le potentiel offert par ce contrôleur. Cela a nécessité le développement d'un logiciel qui a permis d'effectuer la stratégie de contrôle.

REMERCIEMENTS

Nous remercions les professeurs Chahé Nerguizian, Romano DeSantis et Jules O'Shea de l'Ecole Polytechnique de Montréal et Maarouf Saad de l'Ecole de Technologie Supérieure pour leur accueil et la direction du projet.

Nous remercions également l'ensemble du personnel de l'Ecole Polytechnique de Montréal, et en particulier les techniciens de la section Automatique pour l'aide précieuse qu'ils ont apportée à ce travail.

Table de Matières

Liste des Symboles

NE	réseau de neurones qui copie la dynamique du procédé (Neuro Emulateur)
NC	réseau de neurones qui génère la commande du procédé (Neuro Contrôleur)
u(t)	commande appliquée au procédé
y(t)	sortie du procédé
$\hat{u}(t)$	commande estimée par le NC
$\hat{y}(t)$	sortie du procédé estimée par le NE
ε_t	erreur de traînage = référence - ÿ(t)
ε_u	erreur d'estimation du NC = u(t) - ü (t)
ε_y	erreur d'estimation du NE = y(t) - ÿ(t)
Xc	vecteur d'entrée pour NC
Xe	vecteur d'entrée pour NE
a_X	vecteur de sortie de la couche "X" d'un réseaux de neurones
δ^X	erreur propagée à la couche X
w^x	poids (weights) entre les entrées de la couche X et les sorties de la couche précédente
η	coefficient d'apprentissage
Δw^X	variation des valeurs des poids de la couche X
fx	fonction d'activation de la couche X
U1	signal de commande pour réguler le niveau
U2	signal de commande pour réguler la température
u1	commande pour la vanne d'eau chaude
u2	commande pour la vanne d'eau froide
pv1	mesure du niveau (en volts)
pv2	mesure de la température (en volts)
réf	consigne (en volts)

Table de Figures

1. DESCRIPTION DU SYSTEME

1.1 Introduction

L'objectif de ce projet est de mettre au point un contrôleur utilisant les réseaux des neurones artificiels pour l'asservissement et la régulation numériques du niveau et de la température de l'eau dans un réservoir.

Le réservoir utilisé a une capacité de 273 litres et sa section est constante. Ce réservoir est alimenté par deux arrivées d'eau. L'une pour l'eau chaude et l'autre pour l'eau froide. Les débits délivrés par ces deux entrées sont commandés par deux vannes indépendantes. C'est en agissant sur ces deux éléments du système que l'on va réaliser la régulation de niveau et de température.

On dispose de deux thermocouples qui permettent de connaître les températures des arrivées d'eau froide et d'eau chaude. Le réservoir dispose d'un robinet permettant de régler manuellement le débit de sortie. Un troisième thermocouple est fixé sur le robinet de sortie fournissant la température du mélange contenu à l'intérieur du réservoir, donc la température que l'on désire réguler. Le système est également équipé d'un capteur de pression qui est fixé dans la partie basse du réservoir afin de fournir une information sur le niveau d'eau que l'on veut réguler [7].

La cuve ayant une capacité relativement importante, la température de l'eau varie très lentement et n'est pas homogène à l'intérieur du réservoir. Les débits d'eau froide et d'eau chaude arrivants en surface mettent un certain temps pour se propager jusqu'à la sortie. L'asservissement de température est donc très délicat à effectuer avec précision. Pour augmenter la rapidité de réponse de la température, le réservoir dispose d'un agitateur.

1.2 Réalisation pratique des différents éléments du système

1.2.1 Réalisation du contrôle

Le contrôleur est réalisé à l'aide d'un PC équipé d'un microprocesseur *Intel Pentium®*, 166 Mhz. Pour pouvoir interpréter les signaux analogiques du système et pouvoir les commander, le micro-ordinateur utilise une carte entrée - sortie disposant de deux canaux numérique / analogique et de seize canaux analogique/numérique. Il s'agit d'une carte *Labmaster*. Elle est le moyen de communication entre l'ordinateur qui agit comme contrôleur, et le système.

Pour être efficace, le contrôleur doit avoir une information sur les grandeurs qu'il doit asservir. La carte *Labmaster®* fournit au contrôleur deux signaux discrets. L'un est représentatif du niveau d'eau contenue dans le réservoir, l'autre de la température du liquide. Nous appellerons *y1* la mesure du niveau du liquide et *y2* celle de la température.

Pour agir sur le système, le contrôleur utilise deux canaux numérique/analogique de la carte *Labmaster®*. L'un permet de commander la vanne d'eau froide, et l'autre celle d'eau chaude. Nous appellerons *u1* la commande d'eau chaude et *u2* la commande d'eau froide. Les interactions entre contrôleur et système sont ainsi parfaitement définies.

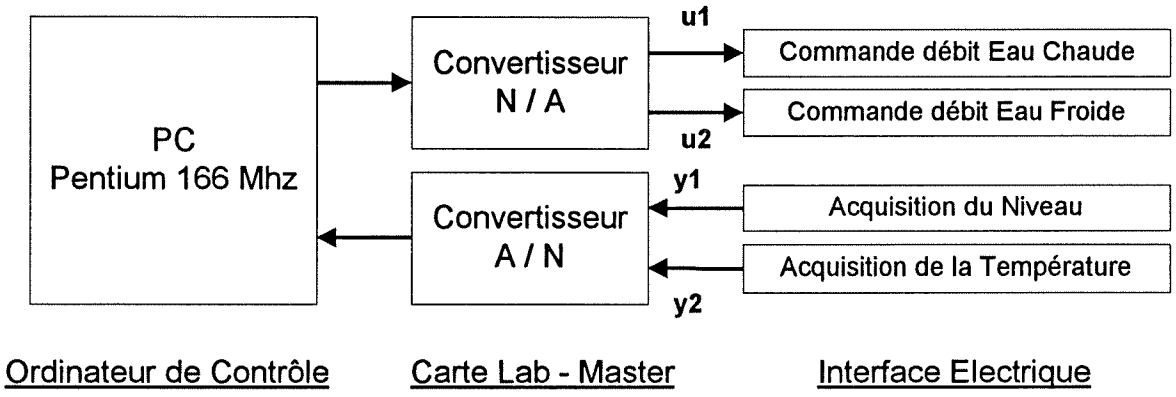

Figure 1 : Schéma d'interactions Entre Système et Contrôleur

1.2.2 Système complet

Les descriptions précédentes permettent d'établir le schéma complet des deux boucles d'asservissement suivant :

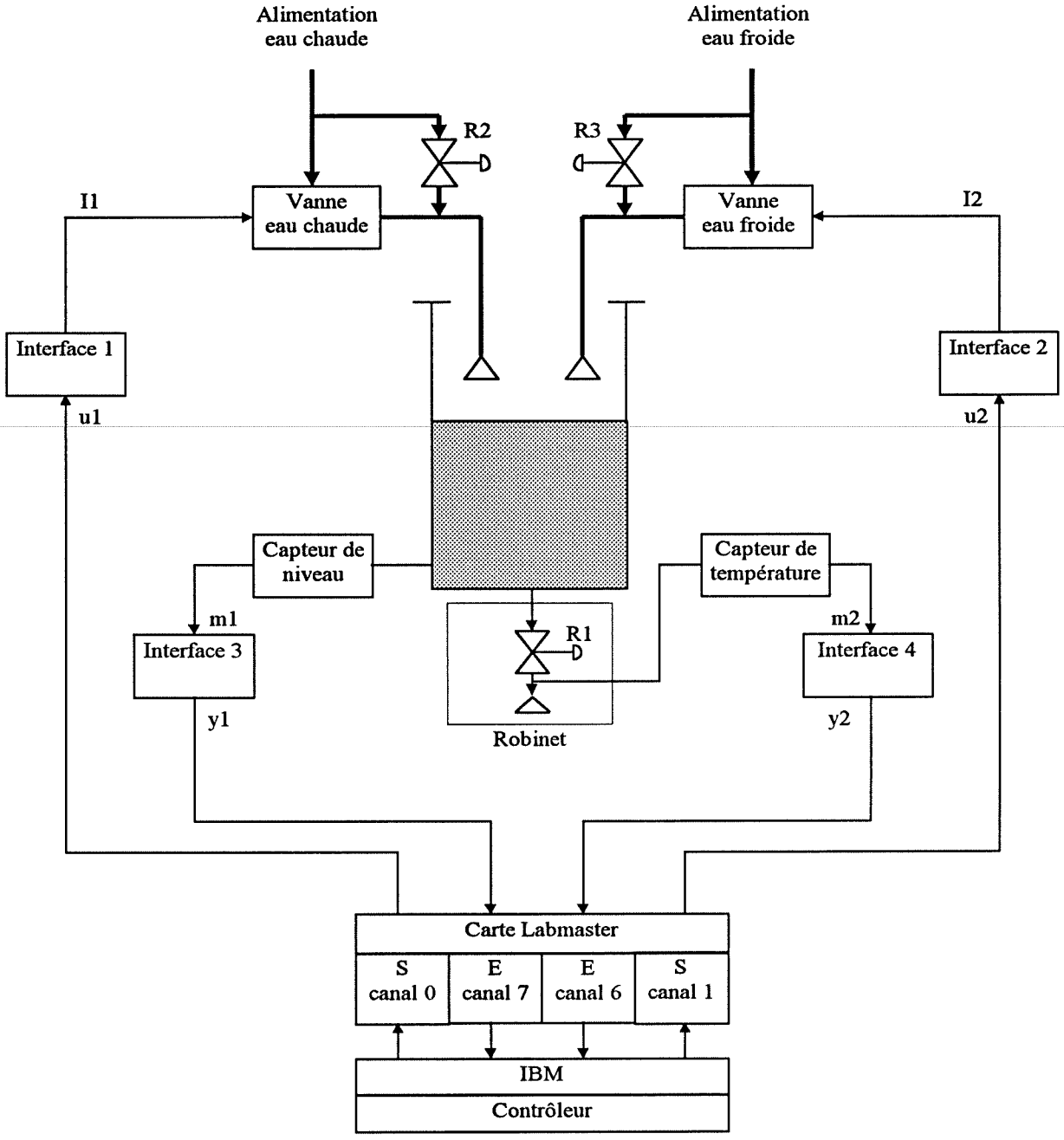

Figure 2 : Système Complet de Contrôle

2. RESEAUX DE NEURONES

2.1 Généralités

Malgré le fort potentiel des réseaux de neurones dans la conception de systèmes de commande, les applications pratiques sont, pour l'instant, très réduites.

Les réseaux de neurones artificielles (RNA) présentent un grand intérêt dans les systèmes de commande grâce aux propriétés suivantes :

- Traitement parallèle de l'information
- Capacité d'auto - apprentissage
- Comportement non linéaire

Ces caractéristiques permettent d'envisager des solutions aux problèmes typiques dans le domaine de contrôle, tels que :

- Comportement non linéaire des systèmes à contrôler
- Incertitude dans la modélisation des procédés réels
- Caractère non stationnaire des procédés

Parmi les différents types de réseaux proposés dans la littérature, les réseaux multicouches ont été choisis car :

- C'est la topologie la plus utilisée dans le milieu des applications de contrôle
- L'algorithme de rétro - propagation de l'erreur est bien connu et facile à implanter
- La structure multicouche permet de modéliser le système avec la technique des diagrammes de blocs

2.2 Cadre d'application

On ne fait pas de restriction sur la nature du procédé à commander. Ainsi, le système peut être linéaire ou non - linéaire, monovariable ou multivariable...

Une des habilitées la plus importante dans la théorie du « Neuro – contrôleur », est la possibilité de construire un régulateur sans la connaissance "à priori" du procédé lui même et ceci à partir des couples entrée / sortie. Malgré cette affirmation, il est fort souhaitable de disposer au moins d'une estimation de l'ordre du système.

2.3 Conception du système de contrôle

2.3.1 Principes

Au cours des dernières années, plusieurs structures basées sur les réseaux multicouches ont été considérées. La structure proposée est connue comme "Indirect Learning Architecture" dans la littérature [1] [2] [3] [6], et elle est représentée dans la figure 3. Les élements nommés Neuro – Emulateur et Neuro – Contrôleur, sont conçus et implantés à l'aide des réseaux de neurones artificielles. Ces deux éléments seront présentés dans les paragraphes suivants et ils sont la base de la structure général de contrôle du niveau et de la température du réservoir.

Dans un premier temps, les valeurs des poids des réseaux Neuro – Controleur (NC) et Neuro – Emulateur (NE) sont calculés à l'aide de la procédure dite hors ligne. Après, ces valeurs seront modifiés en ligne (en temps réel).

Dans tous les cas, l'algorithme de rétro – propagation de l'erreur [3] est à la base de ces deux procédures de calcul.

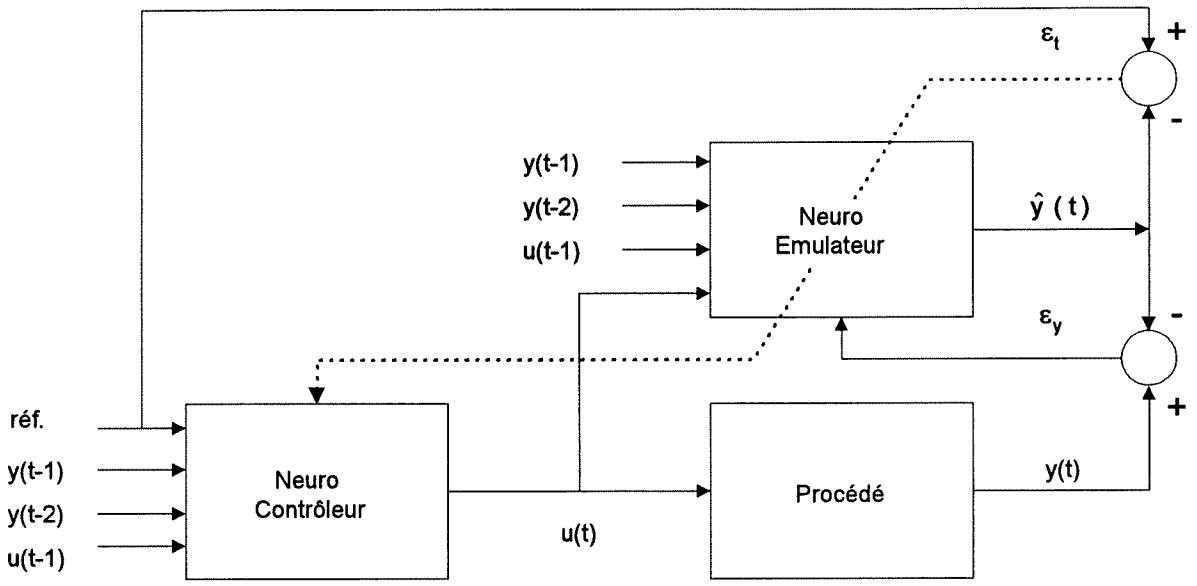

Figure 3 : Structure "Indirect Learning Architecture"

Les fonctions d'activation choisies sont :

- Tangente hyperbolique (bornes –1 à 1) pour les couches d'entrée et cachées. Comme avantage additionnel face aux différentes fonctions proposées dans la littérature, la fonction tangente hyperbolique est symétrique par rapport à l'origine [3].
- Linéaire pour les couches de sortie.

Les valeurs lues par la carte Labmaster® d'acquisition sont donc normalisées dans l'intervalle –1 à +1.

2.3.2 Structure du Neuro – Contrôleur (NC)

L'idée principale est de construire un modèle connexionniste qui reproduit la dynamique inverse du procédé à contrôler. Pour déterminer les valeurs des poids des connexions du NC c'est à dire, leur apprentissage, le principe est le suivant : on fournit au procédé une entrée u(t), qui produit une sortie y(t). La sortie à son tour est appliquée au neuro – contrôleur (NC) qui produit alors une sortie û(t) (voir figure 4, page 13).

Pour le Neuro – Contrôleur (NC), y(t) représente une consigne et u(t) est la commande permettant d'obtenir cette consigne. L'erreur (différence entre u et û) est rétro - propagée à travers le réseau. Cette structure est bien connue [1], [2], [3] et montrée à la figure 4.

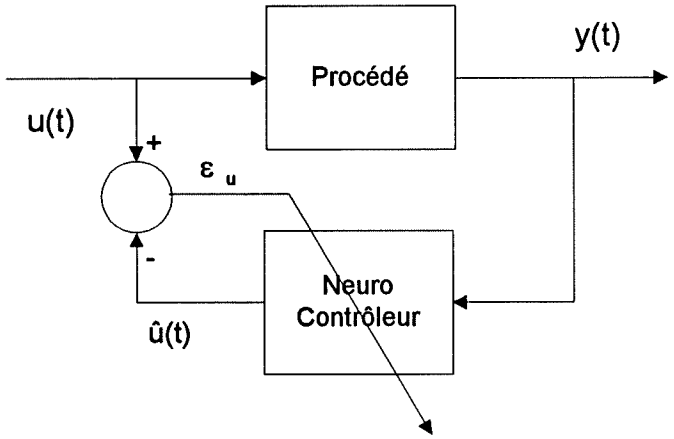

Figure 4 : Neuro – Contrôleur . Structure d'apprentissage

Comme on l'a déjà mentionné, la détermination des poids du réseau est réalisée à l'aide de l'algorithme de rétro – propagation de l'erreur hors ligne, ensuite ces poids sont actualisés en ligne (temps réel).

2.3.3 Structure du Neuro – Emulateur (NE)

L'apprentissage en ligne du Neuro – Contrôleur demande le calcul de la dérivée de l'erreur de sortie par rapport à la commande appliqué ($\delta\varepsilon / \delta u$) afin d'appliquer l'algorithme de rétro - propagation de l'erreur (*backpropagation*). Il est question d'évaluer cette "sensibilité" de la sortie du procédé par rapport à l'entrée d'une façon artificielle. Donc un deuxième réseau, connu comme Neuro – emulateur (NE) [1], [2], [3] est ajouté au système comme montré à la figure 3.

Le Neuro–Emulateur est une copie du procédé. L'apprentissage du neuro – emulateur (NE) est similaire à celui du Neuro – Contrôleur (NC) et le schéma de base est montré dans la figure 5.

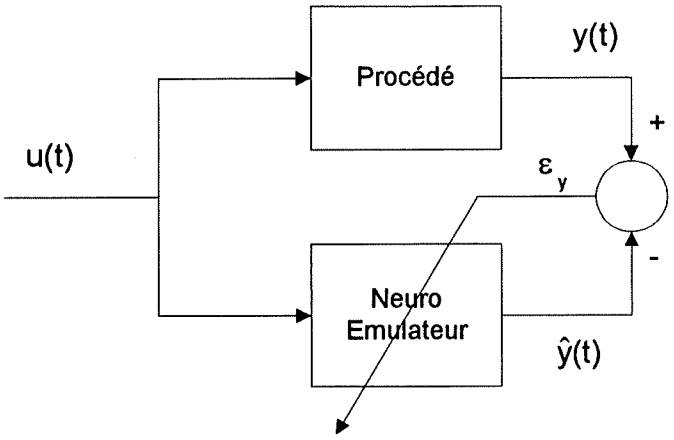

Figure 5 : Neuro Emulateur . Structure d'apprentissage

2.3.4 Apprentissage "hors ligne"

Pour l'application de l'algorithme de rétro – propagation de l'erreur sur les structures d'apprentissage montrées dans les figures 4 et 5, on doit construire un tableau avec les valeurs des couples entrée / sortie du procédé. A partir de ce fichier, une programmation en Matlab® faisant appel au fonction "trainbpx.m" déjà existante dans le toolbox Neural Networks, a permis le calcul des valeurs des poids des connexions des réseaux Emulateur et Contrôleur. La figure 6 illustre l'obtention du tableau entrée – sortie.

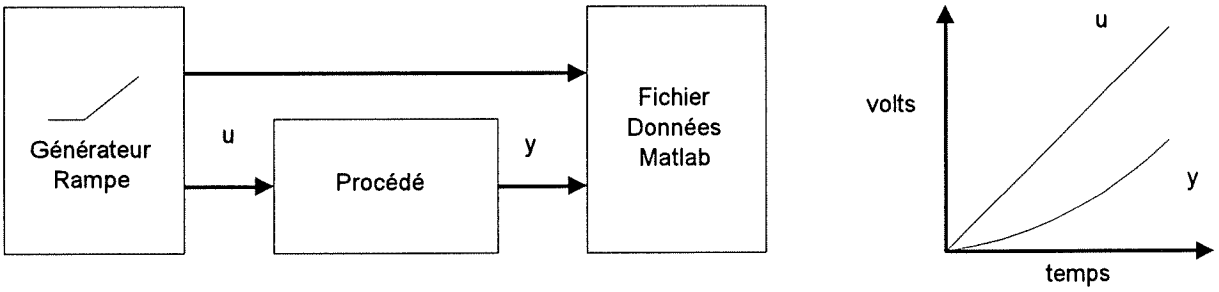

Figure 6 : Génération du Fichier Entrée / Sortie

On a appliqué une entrée de type rampe sur la commande du procédé afin de construire ce tableau entrée / sortie.

Le choix d'une entrée type rampe réside dans le fait qu'on désire obtenir des couples entrée/ sortie "dynamiques", où les valeurs de l'entrée varieraient pour couvrir toute la plage des valeurs de fonctionnement.

A partir du fichier entrée / sortie, quelques points (entre 20 et 30) ont été choisis en favorisant les plus significatifs pour générer un "ensemble d'apprentissage". Avec cet ensemble, et en utilisant la fonction "*trainbpx*" du toolbox Neural Networks de Matlab®, l'apprentissage hors ligne de plusieurs réseaux de différentes tailles a été réalisé jusqu'à l'obtention des erreurs quadratiques totales, pour l'ensemble d'apprentissage, acceptables.

Les réseaux ainsi appris étaient censés de réaliser la dynamique inverse du procédé (Neuro - Contrôleur) et de copier le procédé (Neuro - Emulateur).

2.3.5 Apprentissage en ligne

Une fois les valeurs des poids des connexions du Neuro – Contrôleur et du Neuro – Emulateur calculées à l'aide de la procédure hors ligne, elles sont actualisées afin de minimiser respectivement l'erreur de traînage réf(t) - $\hat{y}(t)$ et l'erreur d'émulation y(t) - $\hat{y}(t)$ (apprentissage en ligne).

2.3.6 Choix des Entrées

Les réseaux de neurones multicouches ont un fort caractère statique. Très utilisés dans les problèmes de classification, quand on présente une entrée au réseau, celui-ci génère la sortie. L'ordre ou séquence de présentation des entrées n'a pas d'influence sur les sorties. En revanche, on est concerné par un système dynamique. Un bon choix de variables retardées à fournir au réseau est très important.

Plusieurs auteurs dans la littérature [3] proposent d'utiliser autant de signaux retardés que l'ordre du système à contrôler. En particulière, on a réalisé plusieurs expériences afin de déterminer à chaque fois la structure la plus performante sans y tenir compte du modèle déjà connu du système et on a déterminé que la réponse obtenue avec 2 signaux retardés est la plus satisfaisante.

Afin d'optimiser la taille des réseaux à implanter, on a limité au début des expériences le nombre de couches à 3 (1 couche cachée) et le nombre de neurones par couche à 15. Après l'apprentissage hors ligne et validation des résultats, ces paramètres se sont révélés suffisants et on a même réduit la taille du Neuro - Emulateur à 2 couches. Si les poids obtenus pour chaque neurone d'une même couche se ressemblaient entre eux, on réduisait encore le nombre de neurones dans cette couche. L'intérêt de réduire au maximum la taille des réseaux est évident : étant donné une période d'échantillonnage fixée à 0.5 seconde (période utilisée dans des expériences précédentes [7]) le temps disponible pour l'actualisation des réseaux et la génération de la commande est donc aussi limité. L'algorithme d'apprentissage en ligne devrait tenir compte de cette contrainte.

2.3.7 Algorithme d'apprentissage en ligne

Basé sur l'approche de Tanomaru & Omatu [3], on a programmé sur Matlab® une fonction (RM.M) qui reçoit comme paramètres la référence, la sortie du système ainsi que ce même signal retardée, et génère la commande pour le procédé (voir les figures 7 et 8).

L'algorithme utilise les poids des réseaux qui ont été calculés hors ligne et enregistrés dans le fichier weights.mat puis ils sont adaptés à chaque échantillon de l'expérience avant de les ré - enregistrer. L'approche théorique utilisée est la suivante, soient :

Vecteur d'entrée au NC :	$Xc(t) = [\, réf(t), y(t-1), y(t-2), \ldots, y(t-p+1), u(t-1), u(t-2), \ldots, u(t-q)]$	(1)
Sortie du NC :	$U(t) = fc\,[Xc(t)]$	(2)
Vecteur d'entrée au NE :	$Xe(t) = [u(t), u(t-1), \ldots, u(t-q), y(t-1), \ldots, y(t-p+1)]$	(3)
Sortie du NE :	$\ddot{y}(t) = fe\,[Xe(t)]$	(4)

Si on considère qu'à l'instant *t* les valeurs disponibles sont :

{ réf(t), ..., réf(t - i) } , { y(t), ..., y(t - i) } et { u(t-1), ...,u(t -1-i) }

il est donc possible de calculer les erreurs:

$$e(t) = réf(t) - y(t)$$
$$e(t - i) = réf(t -1) - y(t - i) \qquad (5)$$

et propager ces erreurs à travers le NE pour actualiser les poids du NC selon l'algorithme de rétro - propagation de l'erreur.

Mais, comme les poids des réseaux sont actualisés à chaque instant d'échantillonnage, les valeurs :*u(t -1), ...,u(t-1-i)* gardées en mémoire ne sont pas les mêmes que si elles étaient générées actuellement. Il est aussi possible de recalculer les valeurs *u(t-1), ...,u(t-1-i)* et ainsi que, les valeurs de sortie de chaque couche, soient NC et NE à partir des données *réf(t), ..., réf(t-i), y(t), ..., y(t-i)* et les poids actuels.

On est maintenant en condition d'appliquer l'algorithme d'actualisation pour l'apprentissage du NC à partir de l'erreur d'estimation $\varepsilon_t = réf(t) - \hat{y}(t)$ et, également l'apprentissage du NE à partir de l'erreur $\varepsilon_y = y(t) - \hat{y}(t)$. Alors, on construit les différentes couples d'erreur :

$y(t) - \hat{y}(t)$, $y(t-1) - \hat{y}(t-1)$, ..., $y(t-i) - \hat{y}(t-i)$ et $réf(t) - \hat{y}(t)$, $réf(t-1) - \hat{y}(t-1)$, ..., $réf(t-i) - \hat{y}(t-i)$

Il est aussi possible d'effectuer plusieurs actualisations (« epochs ») par période d'échantillonnage, et même de choisir différentes fréquences d'actualisation pour le Neuro - Emulateur et le Neuro - Contrôleur or ces possibilités n'ont pas été explorées.

En pratique, étant donné la contrainte de temps fixée pour la période d'échantillonnage, on a pris $i = 2$ et, pour les expériences de contrôle simultané de niveau et de température, $i = 0$. On a réalisé dans toutes les expériences une seule itération par période d'échantillonnage.

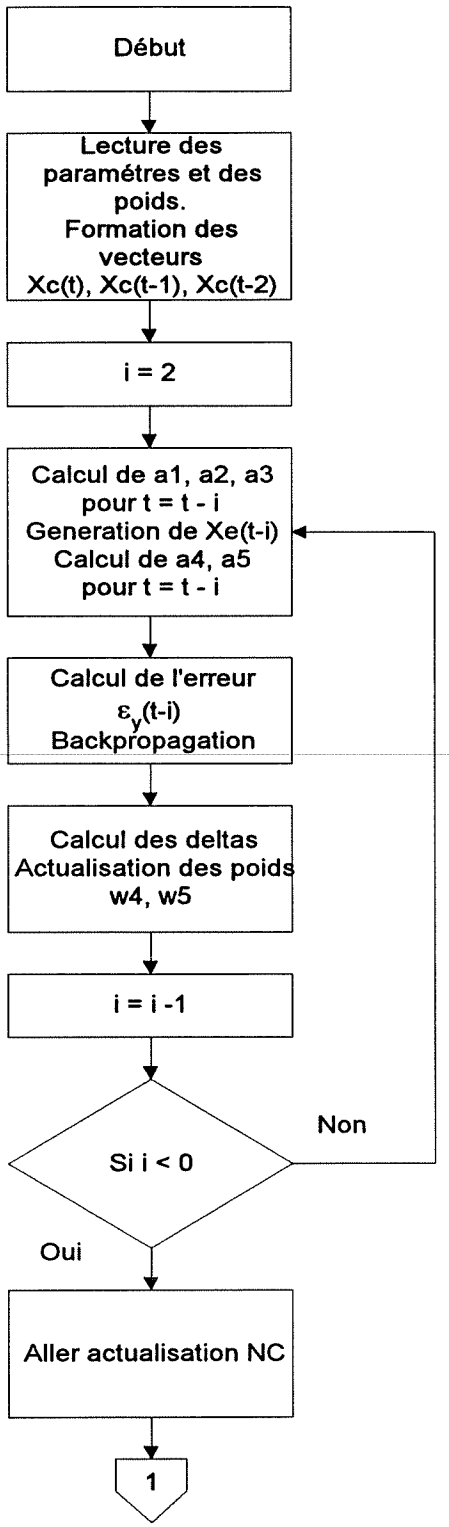

Figure 7 : Algorithme d'apprentissage des Réseaux (1)

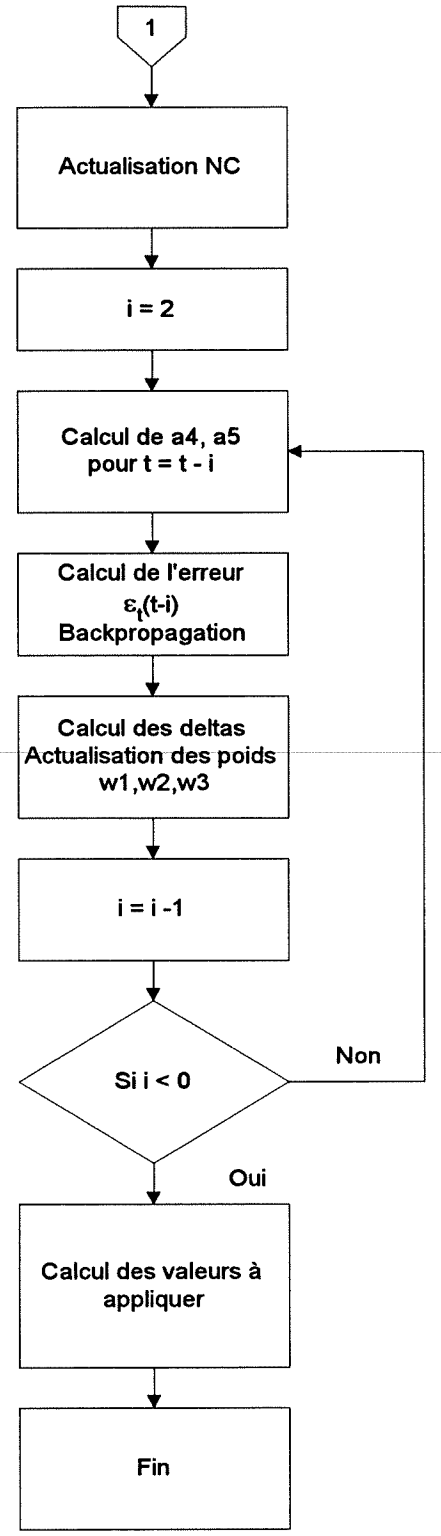

Figure 8 : Algorithme d'apprentissage des Réseaux (Suite)

2.3.8 Algorithme de "Back-propagation". Apprentissage en ligne du NC

Comme première approche, on travail avec la règle Delta [1] [2] [3] [5], sans "momentum". Ainsi, appliquée sur la structure de la figure 9.

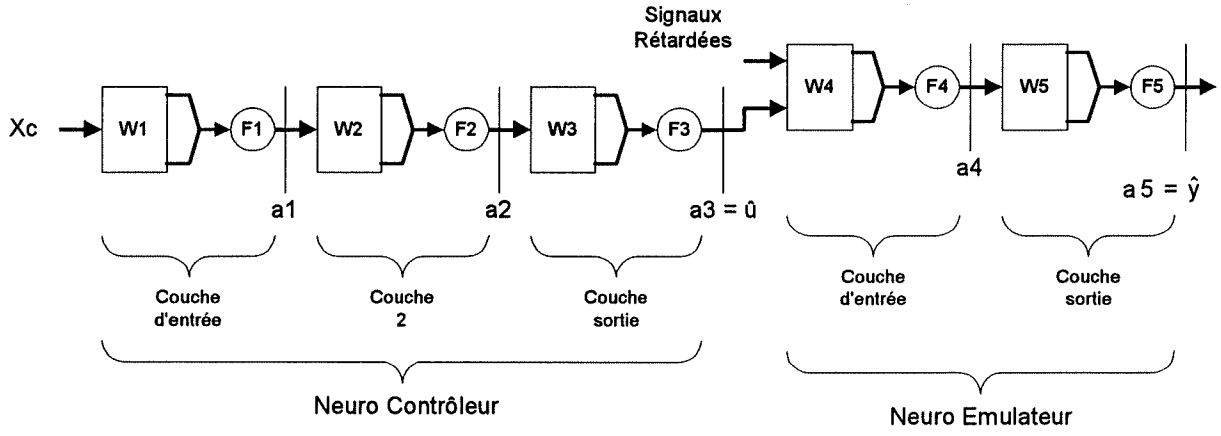

Figure 9 : Structures du Neuro - Emulateur et Neuro - Contrôleur

Les matrices des poids calculées lors des apprentissages hors ligne ont les dimensions suivantes : W1 = [10 x 4] ; W2 = [5 x 10] ; W3 = [1 x 5] ; W4 = [15 x 4] et W5 = [1 x 15] et les fonctions d'activation sont linéaire et tangente hyperbolique pour F5, F3 et F4, F2 , F1 respectivement. Le développement de l'algorithme de rétro – propagation de l'erreur donne l'ajustement des poids pour chaque couche à partir de l'erreur de traînage (Δw) et on obtient :

$$\Delta w^5 = \eta * \varepsilon \tag{6}$$

$$\delta^4 = \left[f'(a_4) * \varepsilon\right] = \left[1 - (a_4 * a_4)\right] * \varepsilon \tag{7}$$

$$\Delta w^4 = \eta * \delta^4 * a_3 \tag{8}$$

$$\delta^3 = \left[f'(a_3) * \sum \delta^4 * w^4\right] = a_3 * \sum \delta^4 * w^4 \tag{9}$$

$$\Delta w^3 = \eta * \delta^3 * a_2 \tag{10}$$

$$\delta^2 = \left[f'(a_2) * \sum \delta^3 * w^3\right] = \left[1 - (a_2 * a_2)\right] * \sum \delta^3 * w^3 \tag{11}$$

$$\Delta w^2 = \eta * \delta^2 * a_1 \tag{12}$$

$$\delta^1 = \left[f'(a_1) * \sum \delta^2 * w^2\right] = \left[1 - (a_1 * a_1)\right] * \sum \delta^2 * w^2 \tag{13}$$

$$\Delta w^1 = \eta * \delta^1 * Xc \tag{14}$$

Où δ^x est l'erreur propagée vers la couche « x » , η est le coefficient d'apprentissage et f' est la dérivée de la fonction d'activation correspondante.

En remplaçant "ε" par ε_t ou ε_y, il est possible d'actualiser les poids du NC ou NE respectivement.

2.4 Expériences

La méthodologie de travail a été :

Construction des réseaux

- Génération de fichier entrée – sortie du procédé (test en boucle ouverte)
- Normalisation des données et sélection d'un nombre de pairs réduit (entre 20 et 30 échantillons)
- Apprentissage du Neuro – Emulateur
- Validation
- Optimisation de la taille du réseau si possible
- Apprentissage du Neuro - Contrôleur
- Validation
- Optimisation de la taille du réseau si possible
- Génération du fichier weights.mat, où sont stockés les poids des réseaux

Réservoir

Tout en ignorant la modélisation mathématique du procédé, on s'est servi des expériences en boucle ouverte pour la conception des contrôleurs. Système, capteurs, interface électrique et actionneurs sont considérés comme un seul ensemble et, d'après notre approche, une "boîte noire" . Pour s'y faire, une rampe est appliquée comme signal de commande. Ainsi, des fichiers dynamiques ont été générés, la valeur de la commande variant d'un point à l'autre.

Trois expériences ont été effectuées : niveau, température, niveau et température. L'idée est de concevoir un régulateur pour le contrôle du niveau, un deuxième pour la température et, en tenant compte des interactions entre les deux variables, réguler le niveau et la température.

L'aspect le plus intéressant, c'est à dire, la réalisation d'une commande sur le deux variables demande la conception d'un réseau linéaire qui prendra comme entrées les commandes générées par les contrôleurs de niveau et de température et fournira les commandes pour les vannes d'eau chaude (u1) et d'eau froide (u2). Ce nouveau réseau a aussi pour mission le découplage, dans la mesure du possible, entre le deux variables à contrôler.

A différence des réseaux précédents, les poids du réseau linéaire ne seront pas modifiés lors des expériences. Les valeurs des poids sont fixées à partir des valeurs du découpleur déjà conçu et implanté lors des travaux précédents sur le réservoir [7].

3. - RÉSULTATS EXPÉRIMENTAUX

On va étudier le développement des tests pour illustrer le comportement de l'algorithme pour la commande de niveau et de température d'un réservoir. Ces tests ont pour but de faire apparaître la capacité du système de contrôle neuronal. Les références sont données en Volts. Pour avoir une idée de la variation du niveau, il faut savoir que 2 Volts de variation en sortie du capteur de niveau correspondent à 5 centimètres de variation du niveau dans le réservoir. Toutes les expériences sont réalisées avec un niveau de repos de 0.5m (0 Volts), une référence en niveau de 0.55m (2 Volts) et le robinet de sortie ouvert à une position fixe.

3.1 Logiciel de contrôle

Le contrôleur neuronal déjà analysé est implanté dans le fichier RN.M. Il s'agit d'un programme écrit avec le langage commercial Matlab®. La liste du logiciel se trouve en annexe. Le programme RN.M fait aussi appel aux routines de communication avec la carte d'acquisition de données *LabMaster*®.

Données en entrées modifiables par l'utilisateur :

- Type d'expérience (Niveau, Température, Niveau + Température)
- Introduction d'une perturbation sur le système (Oui / Non)
- Introduction d'un retard sur la commande (Oui / Non et si Oui retard exprimé en moitié de secondes)
- Alpha(rapport d'eau froide/eau chaude)
- Les références en niveau et température exprimées en Volts
- La période d'échantillonnage du contrôleur
- Le nombre d'échantillons sur lesquels va porter l'expérience. On remarque que le produit de la fréquence d'échantillonnage avec le nombre d'échantillons nous donne la durée de l'expérience

Toutes ces variables ont une valeur par défaut et l'utilisateur du logiciel n'a pas à initialiser toutes les données à chaque utilisation.

Informations retournées par le logiciel :

Lorsque l'expérience est terminée, le logiciel nous donne l'évolution des variables à contrôler (niveau et température) et de la commande de chaque vanne dans le temps. L'utilisateur peut également sauvegarder ces mêmes données sous forme de fichier utilisable par Matlab®. Le logiciel offre aussi la possibilité d'enregistrer les poids des réseaux après l'expérience. Dans le cas où l'utilisateur ne choisit pas cet option, les poids stockés dans le fichier weights.mat seront réutilisés lors du démarrage de la prochaine expérience. La figure 10 présente l'organigramme de l'algorithme général de l'asservissement.

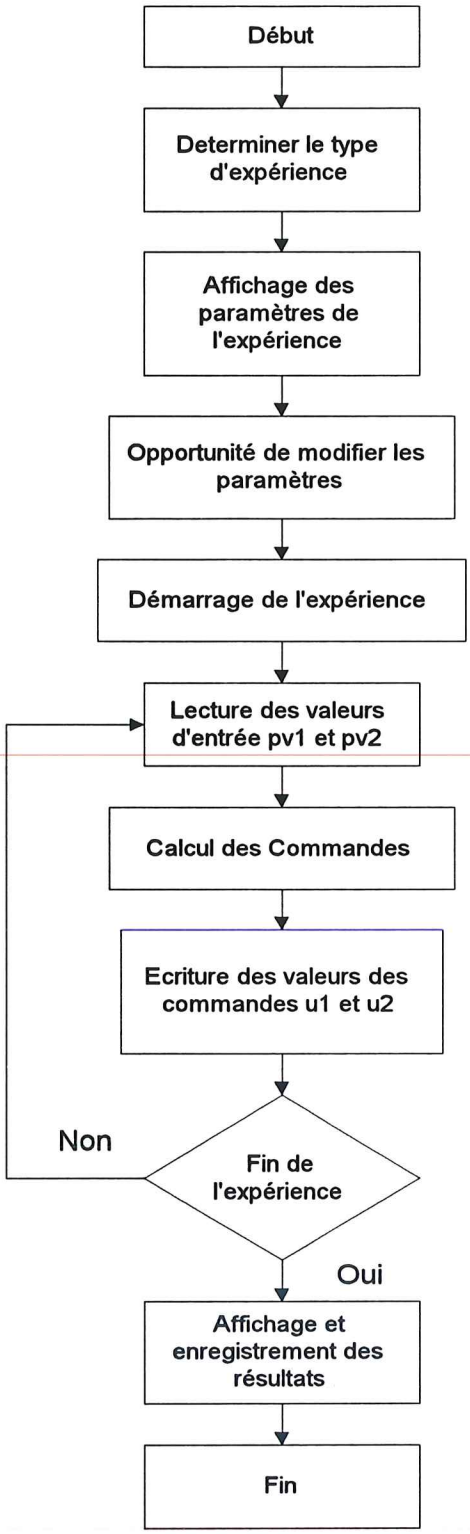

Figure 10 : Algorithme du Logiciel de Contrôle

3.2 Contrôle du niveau

La commande du niveau générée par le contrôleur est nommée *U1*. Elle est appliquée sur le découpleur afin de fournir les deux commandes *u1* et *u2*, commandes d'eau chaude et d'eau froide respectivement. Le découpleur est, dans ce cas ci, un facteur d'échelle qui a pour objectif d'économiser la consommation d'eau chaude. Ainsi, le rapport choisi de façon expérimentale étant *Alpha* = 2 [7], les équations des variables *u1* et *u2* sont :

$$u1 = \left[1 - \frac{Alpha}{1 + Alpha}\right] * U1 = \frac{1}{3} * U1$$

$$u2 = \left[\frac{Alpha}{1 + Alpha}\right] * U1 = \frac{2}{3} U1$$

Noter que, le système étant réel, il est normal de retrouver des écarts entre les réponses lors des différentes expériences car les variations de la pression du réseau d'air comprimé pour les vannes sont importantes pendant la journée. Le régulateur doit faire face à cette contrainte. L'ensemble des tests proposés vont être réalisés avec le même contrôleur afin de tester sa robustesse et versatilité.

3.2.1 Contrôle du niveau seul

Objectif : Mise en évidence des performances de fonctionnement de la régulation de niveau par le contrôle neuronal.

Modalités : Dans cette série de tests, on ne se préoccupe pas à garder la valeur de la température. Le robinet de sortie est ouvert au point fixé et on fixe les valeurs de repos des vannes de manière à équilibrer le niveau à 0.5m (0 volts). Lorsque le système est stable, c'est à dire que le niveau est stationnaire, on peut lancer l'expérience. La valeur de consigne est fixée à 2 Volts.

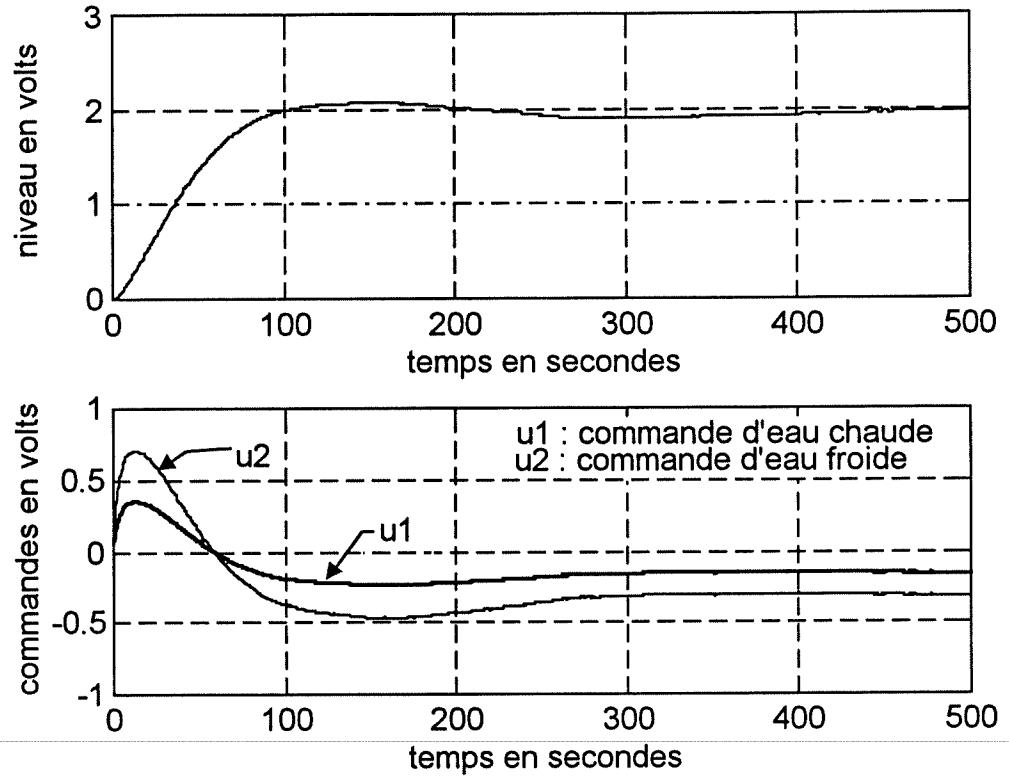

Figure 11 : Test de régulation du niveau

Résultats et commentaires : Le contrôleur offre une bonne performance en rapidité et précision. Il existe un léger dépassement mais l'erreur de traînage est annulé. On peut remarquer quand même des oscillations amorties autour de la valeur finale. Les signaux de commande ne sont pas agités et leurs formes sont identiques mais affectées par le rapport fixe du découpleur. On constate une bonne performance du contrôleur.

3.2.2 Contrôle du niveau en présence d'une perturbation

Objectifs : Mise en évidence des performances de fonctionnement de la régulation de niveau pour le contrôle neuronal et sa capacité de réponse face à une perturbation sur le système .

Modalités : Dans les mêmes conditions que l'expérience précédente, une fois assuré du régime permanent (1000 secondes) on applique une perturbation (échelon de 1 volt d'amplitude) sur le signal de commande de la vanne d'eau froide.

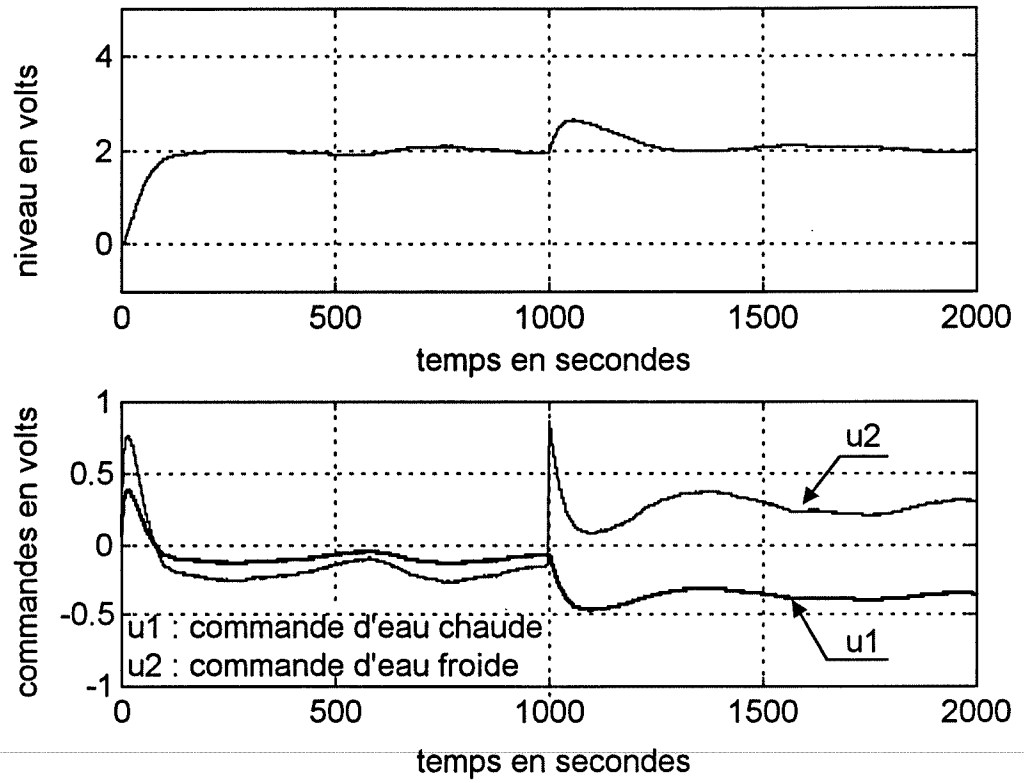

Figure 12 : Contrôle du niveau en présence d'une perturbation

Résultats et commentaires : La réponse initiale du système est similaire à celle obtenue dans l'expérience précédente. La perturbation introduite provoque une augmentation du débit d'entrée au réservoir. Le contrôleur répond très vite en réduisant le débit global d'entrée pour neutraliser la perturbation et récupérer la consigne et la stabilité. On constate une variation de la pression d'air d'alimentation des vannes autour de 800 secondes de l'expérience.

3.2.3 Robustesse du contrôleur en présence d'une perturbation et d'un retard

Objectif : Etudier la robustesse du contrôleur neuronal sous les conditions d'opération visant à reproduire la réalité industrielle, c'est à dire, retard sur la commande et perturbation.

Modalités : Le régulateur utilisé ici est identique à celui utilisé précédemment. Par rapport au cas précédent, un retard de 20 secondes a été introduit sur la commande.

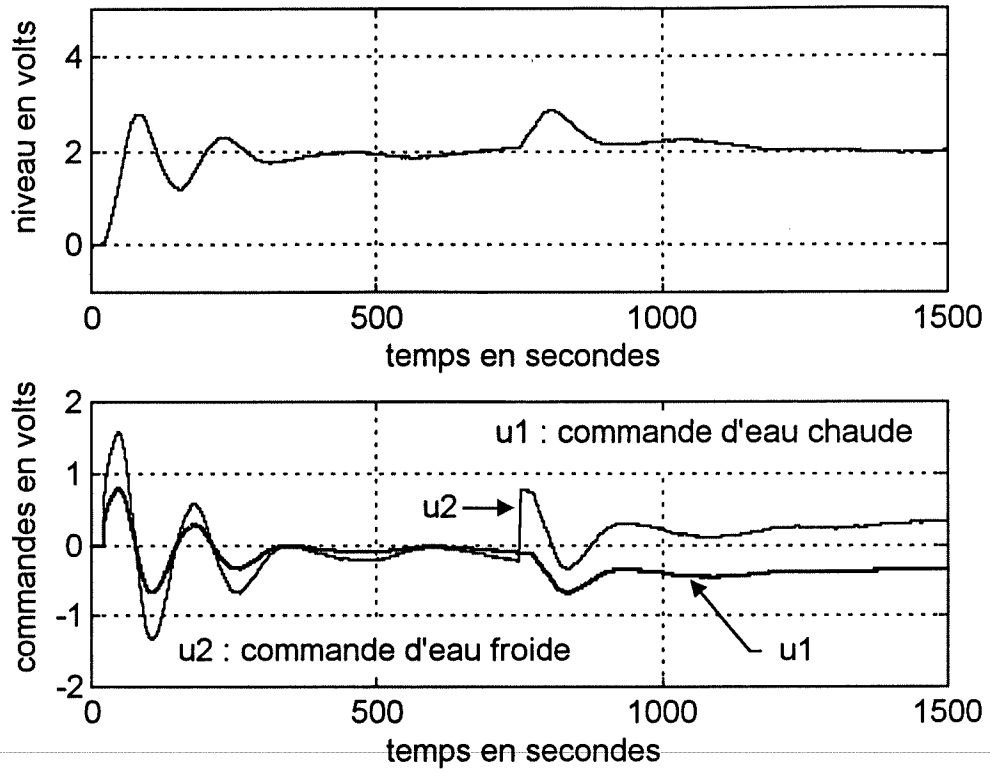

Figure 13 : Robustesse du contrôleur en présence d'une perturbation et d'un retard

Résultats et commentaires : On observe les oscillations types d'un système avec retard, mais le contrôleur réagit encore bien à la perturbation. On peut conclure cette étape en remarquant comme aspect positif la robustesse du régulateur et, dans le coté négatif, la difficulté montrée à maîtriser le retard .

3.3 Contrôle de la température

Parallèlement au cas précèdent, un contrôleur qui aura pour objectif la commande de la température a été conçu. Le contrôleur génère le signal de commande *U2* qui, une fois appliqué sur le découpleur, fournit *u1* et *u2*, commandes des vannes d'eau chaude et d'eau froide respectivement. La valeur expérimentale du *Beta1*, coefficient d'eau chaude, est 0.36 et celle de *Beta2*, coefficient d'eau froide, -0.64. trouvée lors des travaux précédents [7].

Donc, $u1 = Beta1 * U2$

$u2 = Beta2 * U2$

C'est à remarquer que le système physique possède un retard naturel d'environ 20 secondes dû au temps nécessaire pour le mélange et l'homogénéisation de l'eau arrivant avec toute la masse d'eau déjà stockée dans le réservoir [7].

La mesure de la température est assurée par un thermocouple qui présente une sensibilité trop élevée. Un filtre numérique du 1er ordre avec une constante de temps de 20 secondes a été conçu expérimentalement afin de lisser la réponse de la température (de l'ensemble capteur – interface électrique) [7]. Cette constante de temps fournie la meilleur performance de la réponse.

Comme dans le cas du contrôle du niveau, un seul contrôleur a été soumis à l'ensemble des expériences.

3.3.1 Contrôle de la température seule

Objectif : Illustrer le comportement de la température avec un contrôleur neuronal, le niveau étant réglé manuellement à 0.5m (136 Litres).

Modalités : Les tests réalisés pour l'asservissement de température sont de même type que ceux réalisés pour l'asservissement de niveau. Pour ces tests, on ouvre le robinet de sortie au point fixé puis on choisit le point de repos pour les débits d'eau froide et d'eau chaude de façon à équilibrer le niveau à 0.5m et à fixer la température de repos souhaitée. La consigne désirée étant fixée à 6 volts, on démarre l'expérience.

Pour s'assurer des bonnes conditions de fonctionnement, on stabilise manuellement le niveau à l'aide du potentiomètre de régulation d'alimentation de la vanne d'eau chaude et d'eau froide.

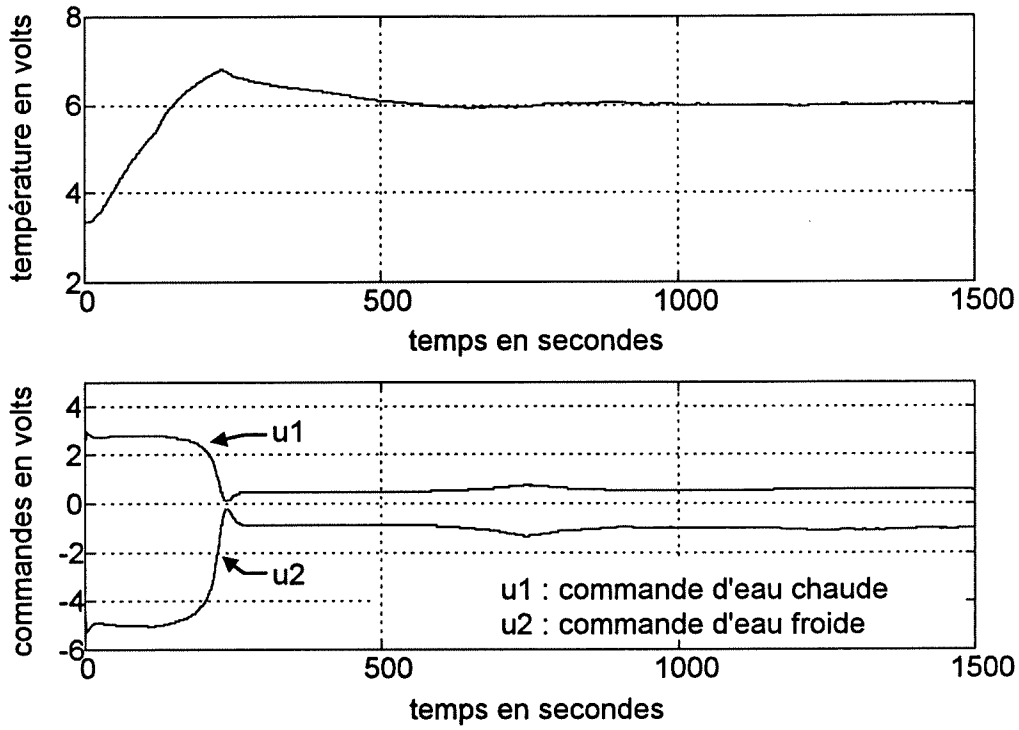

Figure 14 : Contrôle de la température

Résultats et commentaires : On remarque un fort dépassement que, malgré les différentes expériences, n'a pas pu être réduit. Cette contrainte a comme conséquence une mauvaise performance en rapidité. Le changement de pente dans la première partie du transitoire de la réponse est témoin des variations de la pression de l'eau arrivant au réservoir car, pour une commande de valeur constante, il y a une variation du débit d'entrée.

3.3.2 Contrôle de la température en présence d'une perturbation

Objectif : Mise en évidence de la capacité de réponse du contrôleur neuronal face à une perturbation sur le système .

Modalités : Dans les mêmes conditions que l'expérience précédente, une fois assuré du régime permanent (1000 secondes) on applique une perturbation artificielle (échelon de -3 volts) sur la commande de température qui se traduit, après l'action du découpleur, en un échelon de -1 volt sur la commande d'eau chaude et de 2 volts sur la commande d'eau froide.

La température du réservoir tendant à diminuer, le contrôleur doit réagir pour compenser les nouvelles conditions d'opération.

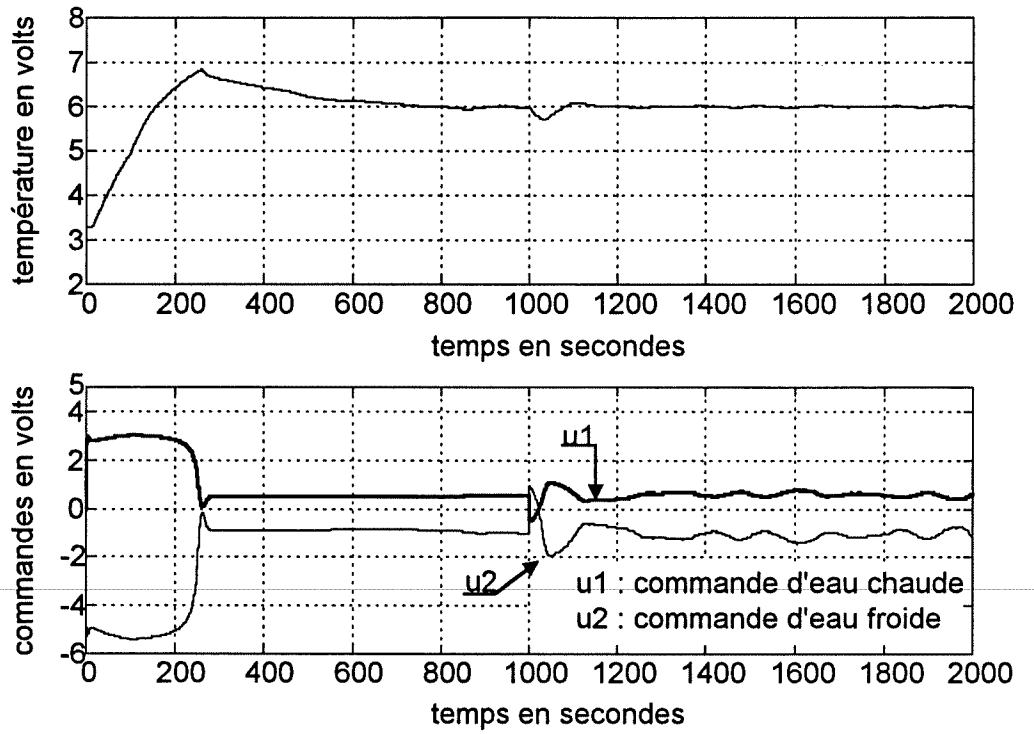

Figure 15 : Contrôle de la température en présence d'une perturbation

Résultats et commentaires : Le contrôleur offre une bonne réponse en régulation réagissant rapidement et efficacement à la perturbation. On remarque aussi la qualité des signaux de commande.

3.3.3 Robustesse du contrôleur en présence d'une perturbation et d'un retard

Objectif : Etudier la robustesse du contrôleur neuronal sous les conditions d'opération visant à reproduire la réalité industrielle, c'est à dire, retard sur la commande et perturbation.

Modalités : Le régulateur utilisé et la perturbation introduite sont identiques à ceux utilisés lors de l'expérience précédente. De plus, au retard naturel du système (20 secondes), un retard sur la commande de 30 secondes additionnelles a été introduit.

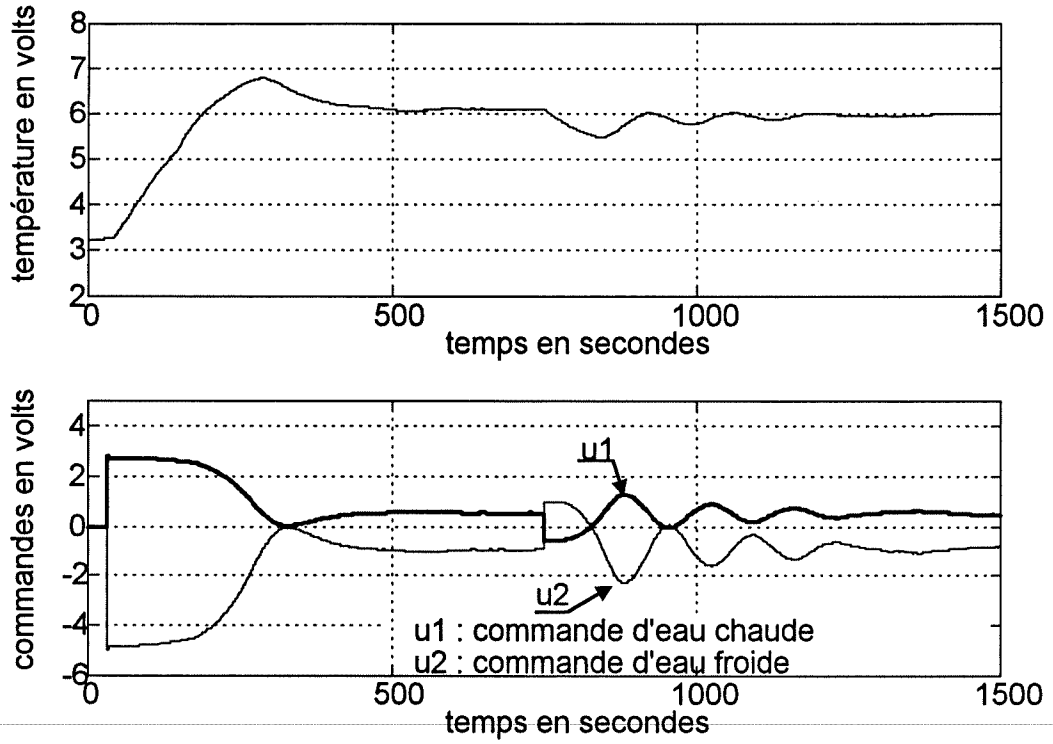

Figure 16 : Robustesse du contrôleur en présence d'une perturbation et d'un retard

Résultats et commentaires : Le retard appliqué ne détériore pas en excès les performances obtenues lors de l'expérience précédente. On observe les oscillations sur la température dans le procès de récupération après la perturbation. En général, on constate un bon comportement en régulation et une performance moins brillante en asservissement.

3.4 Test des deux boucles scalaires

3.4.1 Deux boucles scalaires seules

Objectifs : Les expériences suivantes ont pour but d'illustrer le comportement de deux contrôleurs conçus séparément mais utilisés simultanément à l'aide du découpleur. Les interactions entre les deux variables à contrôler étant très fortes, cette série d'essais a comme objectif additionnel de tester l'efficacité du découpleur proposé.

Modalités des test : Les expériences sont réalisées sous les mêmes conditions qu'aux parties 3.2 et 3.3 (mêmes contrôleurs, mêmes paramètres et mêmes consignes) sauf que maintenant, les deux boucles fonctionnent de façon simultanée. Le robinet de sortie est ouvert à son point fixé et on fixe le débit de repos pour les vannes d'eau chaude et d'eau froide de manière à stabiliser le niveau à 0.5m et obtenir la température de repos désirée. Ces réglages ne seront pas modifiés pendant l'expérimentation.

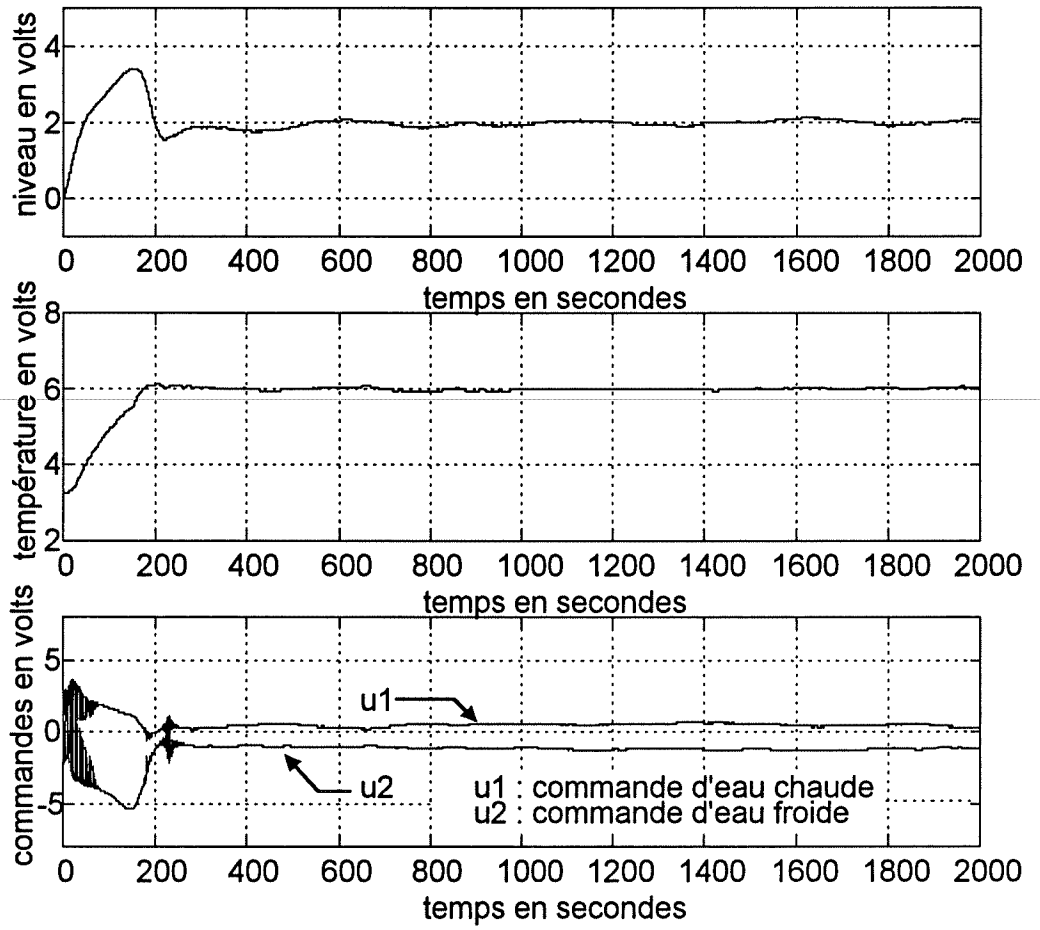

Figure 17 : Contrôle des deux boucles scalaires

Résultats et commentaires : La stratégie de contrôle développée est correcte : le regulateur ouvre la vanne d'eau chaude et ferme celle d'eau froide pour faire monter le niveau et la température ; une fois la consigne de niveau atteint, le débit est réduit afin de minimiser le dépassement du niveau. L'objectif de température est aussi réussi, le régulateur fixe le débit global (eau froide + eau chaude) pour garder le niveau désiré.

La forte interaction entre les deux variables provoque le dépassement observé dans le niveau. On remarque un excellent contrôle sur la température. Les commandes sont, par contre, très agitées au début de l'expérience car la recherche simultanée des deux objectifs présente un conflit. Une fois les consignes rapprochées, les commandes se stabilisent.

3.4.2 Contrôle des deux boucles en présence d'une perturbation

Objectif : Mise en évidence de la capacité de réponse du contrôleur neuronal face à une perturbation sur le système .

Modalités : Dans les mêmes conditions que l'expérience précédente, une fois assuré du régime permanent (1000 secondes), on applique une perturbation (échelon de 1 volt) sur la commande d'eau froide. Deux effets simultanés sont déclenchés : La température du réservoir tendant à diminuer et le niveau du réservoir à augmenter.

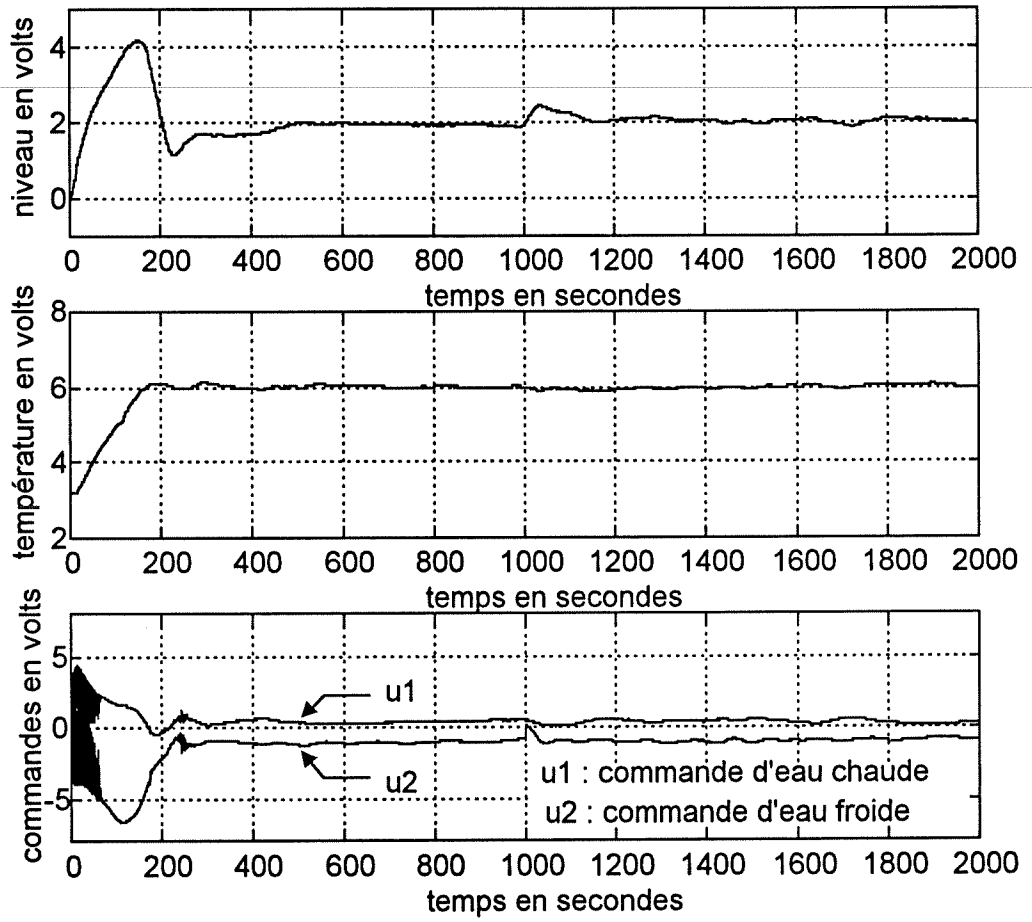

Figure 18 : Contrôle des deux boucles en présence d'une perturbation

Résultats et commentaires : Le contrôleur répond de façon très efficace face à la perturbation, mettant en évidence son bon comportement en régulation. On constate un dépassement plus élevé par rapport a l'expérience précédente due a l'augmentation de la pression d'air du système.

3.4.3 Robustesse du contrôleur en présence d'une perturbation et d'un retard

Objectif : Etudier la robustesse du contrôleur neuronal sous les conditions d'opération visant à reproduire la réalité industrielle, c'est à dire, retard sur la commande et perturbation.

Modalités : Les régulateurs utilisés sont identiques à ceux utilisés lors des expériences de niveau et de température. De plus, au retard naturel du système (20 secondes), un retard sur la commande de 20 secondes additionnelles a été ajouté.

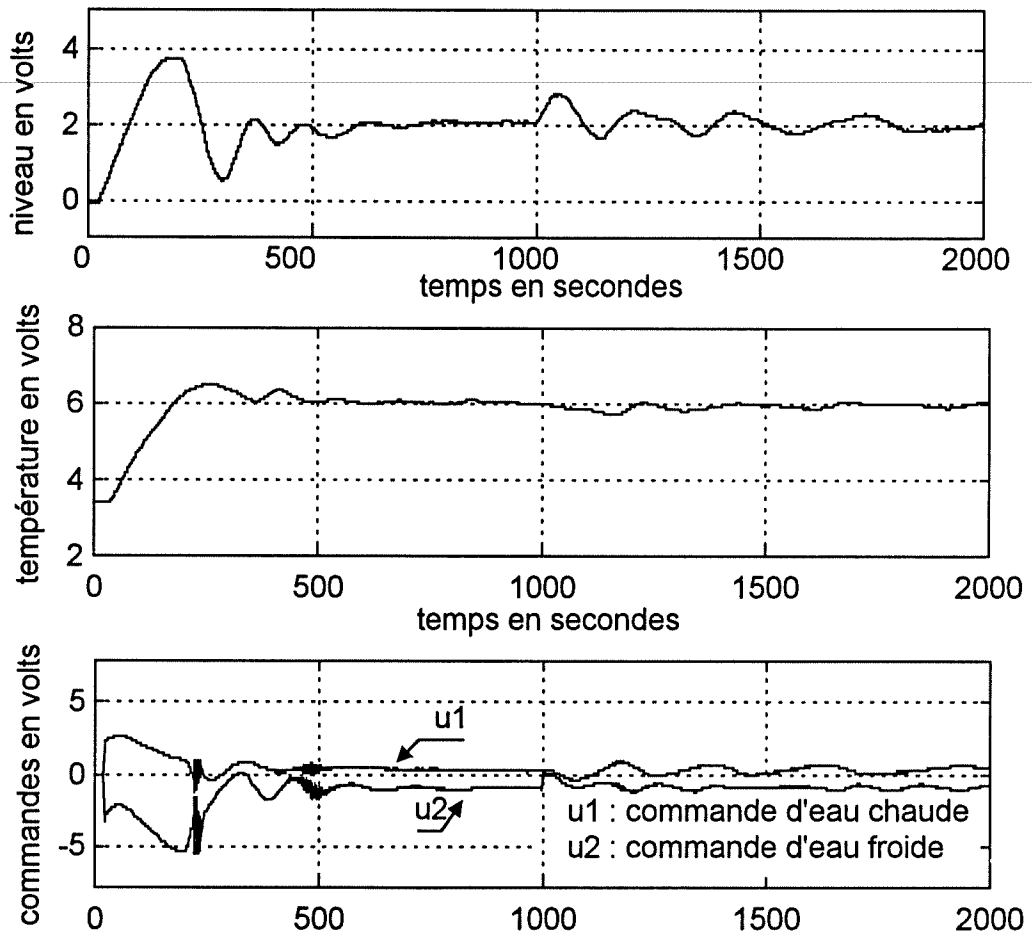

Figure 19 : Robustesse du contrôleur en présence d'une perturbation et retard

Résultats et commentaires : Le contrôleur réussit à stabiliser les variables de sortie. La performance en asservissement montre la forte interaction entre les deux variables (le niveau et la température). La recherche de la stratégie de contrôle optimal provoque l'agitation locale des commandes.

3.5 Conclusion sur les expériences et simulations

Les résultats montrent que il est possible de concevoir et implanter des systèmes de contrôle neuronal performants.

Des difficultés de conception ont été rencontrées, notamment concernant le dimensionnement des différents réseaux (choix de la taille : nombre de couches et de neurones), qui ont été résolues de façon expérimentale.

L'algorithme d'apprentissage en ligne a aussi présenté de difficultés de conception et d'implantation sur le banc d'essai. Cette partie est la contribution la plus importante de ce travail et représente le point clé pour le développement du contrôleur neuronal.

L'ensemble d'expériences proposées fait une étude complète des performances du contrôleur neuronal. L'analyse des performances obtenues peut être résumée de la façon suivante :

Pour la commande des boucles indépendantes, même si les contrôleurs offrent un fonctionnement correct, leurs performances (rapidité en particulier) ne sont pas optimales. Malgré cette contrainte, la robustesse et l'immunité aux perturbations des contrôleurs sont remarquables. L'introduction d'un retard sur la commande provoque l'apparition d'oscillations très difficiles à maîtriser. Les commandes sont dans tous les cas convenants (pas de saturation).

Pour la commande simultanée des deux boucles, les performances ont été améliorées de façon importante. Les résultats sont plus que acceptables et le comportement général des contrôleurs permet d'envisager une utilisation industrielle.

On voudrait ajouter que, étant donné le temps très limité investi dans ce projet, les résultats sont positifs et encourageants. Certainement, les travaux démarrés doivent être poursuivis mais on ne doute pas que les réseaux de neurones font parti présentement des outils de l'automaticien.

4. Difficultés techniques et remarques

Si bien que les travaux précédents sur le réservoir [7] ont servi à la mise au point de la plupart des détails techniques, notamment ceux reliés aux interfaces électriques et logiciels accessoires (horloge temps réel, lecture et écriture sur la carte LabMaster® ...), lors du développement de ce projet on a rencontré plusieurs difficultés techniques.

Ainsi, la lecture de la température de l'eau dans le réservoir a présenté un problème important de bruit, car la sensibilité du thermocouple est trop importante. Pour résoudre ce problème, on a ajouté un filtre numérique de premier ordre (constante de temps de 20 secondes) dans le logiciel de contrôle qui nous a permis de lisser le signal fourni par le capteur.

On a bien retrouvé dans notre système physique un bon exemple de système non stationnaire, car en effet, le système n'était jamais le même :

- La pression d'alimentation de l'eau variait au cours de la journée
- La température de l'eau chaude variait aussi de façon importante (on a mesuré des températures allant de 97 jusqu'à 120 degrés Fahrenheit
- La température de l'eau froide augmentait aussi avec l'arrivée de l'été mais, de façon bizarre, et probablement à cause d'un système de réfrigération, certains journées spécialement chaudes on trouvait que la température de l'eau froide avait baissée.

Malgré ces variations, aucun ajustement a été réalisé sur les contrôleurs.

Côté équipement, la récente acquisition d'un *PC Pentium®* 166 Mhz avec 32 Mo de mémoire vive a vraiment permis la réalisation pratique du projet, car la puissance de calcul demandé par l'algorithme neuronal est très importante.

5. Bibliographie

[1] Psaltis D., Athanasios S., Yamamura A. , ***"A multilayered Neural Network Controller"*** . IEEE Control Systems Magazine Vol. 8 pp. 17-21, 1988.

[2] Widrow B. et Nguyen, D.H. , ***"Neural Networks for self-learning control"***. IEEE Control Systems Magazine, Vol. 10, pp. 18-23, 1990.

[3] Omatu S. , Khalid M. and Yusof R., ***"Neuro-Control and its Applications"***. SPRINGER, London, 1996.

[4] Narenda, K.S. et Parthasarathy K., ***"Identification and Control of Dynamical Systems Using Neural Networks"*** IEEE Transactions on Neural Networks, Vol. 1, pp. 4-27, 1990.

[5] Youji, I. , Hideaki, S. et Hidekatsu, T., ***"A non linear Regulator Design in the presence of systems uncertainties using multilayered Neural Networks"*** . IEEE Transactions on Neural Networks. Vol. 2, pp. 410-417, 1991.

[6] Werbos P. J. , ***"Backpropagation Through Time : What it does and how to do it"*** . Proc. IEEE, vol. 10, pp. 1550 – 1560, 1990.

[7] Cornieles, E. , Bougeret, C., ***"Comparaison Expérimentale de différentes Techniques de Réglage du Régulateur PID et PID Dual Loop"***, Ecole Polytechnique de Montréal, Génie Electrique, Section Automatique, EPM/RT-97/19.

6. ANNEXE 2 : LISTING DU LOGICIEL

```
% Programme rn.m : Ce programme utilise comme algoritme de regulation deux
% réseaux de neurones multicouche.
% Les poids des réseaux se trouvent dans le fichier weights.mat dans le même
% repertoire que le programme principal dans l'ordinateur du banc d'essai

% INITIALISATION

clear

% Parametres du controleur
% ========================

clear
echo on
nb_pts=4000;
T=0.5;
ref1=2;
ref2=6;
echo off

F=input('Asservissement de 1.-Niveau 2.-Température 3.-Niveau et
temperature=');
P=input('Echelon de perturbation sur le niveau/température: 1.-oui 0.-nom=');
ret=input('Retard sur le niveau/température, 1.-oui, 0.-nom=');

if ret==1,
retard=input('valeur de retard*0.5 en secondes=');
end


% OPPORTUNITE DE CHANGEMENT
fprintf('************************************************************\n');
fprintf('  Si vous voulez changer les donnes, taper maintenant\n')
fprintf('  les variables desirees et taper break, enter.\n')
fprintf('  Pour demarrer l''experience taper break, enter.\n')
fprintf('************************************************************\n')

keyboard

%  Decoupleur
alpha1=2;
alpha11=1/(1+alpha1);
alpha12=alpha1*alpha11;
alpha21=0.36;
alpha22=-.64;

% Initialisation des variables

pv1 = zeros(1,nb_pts);
pv2 = zeros(1,nb_pts);
U1 = zeros(1,nb_pts);
U2 = zeros(1,nb_pts);
consigne1 = ref1*ones(1,nb_pts);
consigne2 = ref2*ones(1,nb_pts);
pert=0;
pv3(1)=lire_an(6);
pv2(1)=pv3(1);
```

```
fprintf('SOYEZ PATIENT!!!!\n');
est=0;
fprintf('---1---2---3---4---5---6---7---8---9---0\n');
% Initialisation du convertisseur A/N N/A
init_lab;

% ***** BOUCLE DE REGULATION *****

dep_horl;           % Demarrage horloge temps reel
load weights;       % Lecture des poids reseaux
for i=2:nb_pts,     % Boucle principal

   dep_horl;
   if (100*i/nb_pts)>(est+2.45) est=est+2.5;
   fprintf('*');
   end

   pv1(i) = lire_an(7);    % Lecture du niveau
   pv3(i) = lire_an(6);    % Lecture de la temperature

% Filtre numerique pour la température

      pv2(i)=(T*pv3(i)+20*pv2(i-1))/(T+20);

% Creation du vecteur v = r(t) y(t-10) y(t-20) u(t-10)

      v3=0;
      v4=0;
      v5=0;
      if i>5,
            v3=pv1(i-5);
            v5=U1(i-5);
      end
      if i>10
            v4=pv1(i-10);
      end

      if F==1 | F==3,   % Control du Niveau
      v1=consigne1(i);
      v2=pv1(i);
      v=[v1;v2;v3;v4;v5];

   % Reseau Niveau

   % Function complete pour Nc=3 couches et Ne=2 couches
   % Recoit le vecteur v(5x1) = r(t) y(t) y(t-5) y(t-10) u(t-5)
   % Update des poids du Nc puis Ne avec 1 echantillon.
   % Utilise le fichier weights avec W1-W5 et B1-B5 plus les vecteurs p et t.
   % Entrees pour le Nc : r(t) y(t-5) y(t-10) u(t-5)
   % Entrees pour le Ne : u(t) y(t-5) y(t-10) u(t-5)

   v=v/10;                          % Normalisation des entrees
   p=[v(1,1);v(3,1);v(4,1);v(5,1)]; % Nc = r(t) y(t-5) y(t-10) u(t-5)
   t=v(2,1);                        % Vecteur target t = y(t)
   u=[0;p(2,:);p(3,:);p(4,:)];      % Ne = a3(t) y(t-5) y(t-10) u(t-5)
```

```
% Update NE

    a1=tsig(W1*p,B1);
    a2=tsig(W2*a1,B2);
    a3=(W3*a2+B3);
    u(1,1)=a3;
    a4=tsig(W4*u,B4);
    a5=W5*a4+B5;

    % Erreur et calcul des deltas

    e=t-a5;              % Erreur = y(t)-a5(t)
    d5=e;
    d4=(ones-(a4.*a4)).*(W5'*d5);

    % Update des poids du NE

    lr=0.025;
    W5=W5+lr*(d5*a4');
    B5=B5+lr*d5;
    W4=W4+lr*(d4*u');
    B4=B4+lr*d4;

% Update NC

   a4=tsig(W4*u,B4);
   a5=(W5*a4)+B5;

   % Erreur et calcul des deltas

   e=(p(1,1)-a5);          % Erreur = r(t)-a5(t)
   d5=e*3;
   d4=(ones-(a4.*a4)).*(W5'*d5);
   d3=d4'*W4(:,1);
   d2=(ones-(a2.*a2)).*(W3'*d3);
   d1=(ones-(a1.*a1)).*(W2'*d2);

   % Update des poids du Nc

   lr =2;
   D3=lr*(d3*a2');
   W3=W3+D3;
   C3=lr*d3;
   B3=B3+C3;
   D2=lr*(d2*a1');
   W2=W2+D2;
   C2=lr*d2;
   B2=B2+C2;
   D1=lr*(d1*p');
   W1=W1+D1;
   C1=lr*d1;
   B1=B1+C1;
```

```
   % Valeurs a appliquer

      a1=tsig(W1*p,B1);
      a2=tsig(W2*a1,B2);
      a3=(W3*a2+B3);
      U1(i)=a3;

% Sauvegarde des valeurs

      k=.997;
      B1=B1-C1*k;
      B2=B2-C2*k;
      B3=B3-C3*k;
      W1=W1-D1*k;
      W2=W2-D2*k;
      W3=W3-D3*k;

      End        % Fin boucle de control niveau

if F==2 | F==3,  % Control de la temperature

      v3=0;
      v4=0;
      v5=0;
      if i>5,
            v3=pv2(i-5);
            v5=U2(i-5);
      end
      v1=consigne2(i);
      v2=pv2(i);
      v=[v1;v2;v3;v4;v5];

% ************************************************************************

% Function complete pour Nc=3 couches et Ne=2 couches
% Recoit le vecteur v(5x1) = r(t) y(t) y(t-10) y(t-20) u(t-10)
% Update des poids du Nc puis Ne avec les dernier echantillon.
% Retourne la prochaine valeur de commande.
% Utilise le fichier weights.mat avec W1-W5 et B1-B5 plus les vecteurs p et t.
% Entrees pour le Nc : r(t) y(t-10) y(t-20) u(t-10)
% Entrees pour le Ne : u(t) y(t-10) y(t-20) u(t-10)

% Creation des vecteurs Xc et Xe

v=v/10;
q=[v(1,1);v(3,1);v(4,1);v(5,1)]; % Nc = r(t) y(t-1) y(t-2) u(t-1)
t=v(2,1);              % Vecteur target t = y(t)
u=[0;p(2,:);p(3,:);p(4,:)];      % Ne = a3(t) y(t-1) y(t-2) u(t-1)

% Update NE

   a6=tsig(W6*q,B6);
   a7=tsig(W7*a6,B7);
   a8=(W8*a7+B8);
   u(1,1)=a8;
   a9=tsig(W9*u,B9);
   a10=W10*a9+B10;
```

```
    % Erreur et calcul des deltas

    e=t-a10;          % Erreur = y(t)-a5(t)
    d10=e;
    d9=(ones-(a9.*a9)).*(W10'*d10);

    % Update des poids du NE

    lr=0.025;
    W10=W10+lr*(d10*a9');
    B10=B10+lr*d10;
    W9=W9+lr*(d9*u');
    B9=B9+lr*d9;

% Update NC

    a9=tsig(W9*u,B9);
    a10=(W10*a9)+B10;

    % Erreur et calcul des deltas

    e=(q(1,1)-a10);         % Erreur = r(t)-a5(t)
    d10=e*2;
    d9=(ones-(a9.*a9)).*(W10'*d10);
    d8=d9'*W9(:,1);
    d7=(ones-(a7.*a7)).*(W8'*d8);
    d6=(ones-(a6.*a6)).*(W7'*d7);

    % Update des poids du Nc

    lr =2;
    D8=lr*(d3*a2');
    W8=W8+D8;
    C8=lr*d8;
    B8=B8+C8;
    D7=lr*(d7*a6');
    W7=W7+D7;
    C7=lr*d7;
    B7=B7+C7;
    D6=lr*(d6*q');
    W6=W6+D6;
    C6=lr*d6;
    B6=B6+C6;

% Valeurs a appliquer

    a6=tsig(W6*q,B6);
    a7=tsig(W7*a6,B7);
    a8=(W8*a7+B8);
    U2(i)=a8;
    k=.997;
    B6=B6-C6*k;
    B7=B7-C7*k;
    B8=B8-C8*k;
    W6=W6-D6*k;
    W7=W7-D7*k;
    W8=W8-D8*k;
    end          % Fin du control de la temperature
```

```
%**** Boucle de perturbation *****

if P==1 & i>=nb_pts/2;
pert=1;
end

%**** Boucle de retard ****

if F==1,
u11(i)=alpha11*U1(i);
u12(i)=alpha12*U1(i)+pert;
UU1(i)=0;
if ret==1,
   if i>retard,
      UU1(i)=U1(i-retard);
   end
   u11(i)=alpha11*UU1(i);
   u12(i)=alpha12*UU1(i)+pert;
   end

u21(i)=0;
u22(i)=0;
end
if  F==2,
     u21(i)=alpha21*(U2(i)+pert);
     u22(i)=alpha22*(U2(i)+pert);
     UU2(i)=0;
   if ret==1
      if i>retard,
         UU2(i)=U2(i-retard);
      end
      u21(i)=alpha21*(UU2(i)+pert);
      u22(i)=alpha22*(UU2(i)+pert);
   end
u12(i)=0;
u11(i)=0;
end
if F==3,
     u21(i)=alpha21*U2(i);
     u22(i)=alpha22*U2(i);
     u11(i)=alpha11*U1(i);
     u12(i)=alpha12*U1(i)+pert;
     UU1(i)=0;
     UU2(i)=0;
     if ret==1
      if i>retard
         UU1(i)=U1(i-retard);
         UU2(i)=U2(i-retard);
      end
     u21(i)=alpha21*UU2(i);
     u22(i)=alpha22*UU2(i);
     u11(i)=alpha11*UU1(i);
     u12(i)=alpha12*UU1(i)+pert;
     end
end
```

```
%**** Ecriture des valeurs calcules **************

u1=u11+u21;
u2=u12+u22;
ecr_na (0, u1(i));
ecr_na (1, u2(i));
att_horl (1000 * T);
end    % Fin de la boucle principal

% ***** Retour aux offsets des vannes *****

ecr_na (0, 0);
ecr_na (1, 0);

% *****  AFFICHAGE REPONSE *****

temps = [1:1:nb_pts] * T;
subplot(221),plot(temps,pv1)
title('Comportement du niveau')
xlabel('Temps en secondes')
ylabel('Volts')
grid
subplot(222),plot(temps,u1)
title('Commmande d"eau chaude')
xlabel('Temps en secondes')
ylabel('Volts')
grid
subplot(223),plot(temps,pv2)
title('Comportement de la temperature')
xlabel('Temps en secondes')
ylabel('Volts')
grid
subplot(224),plot(temps,u2)
title('Commande d"eau froide')
xlabel('temps(secondes)')
ylabel('Volts')
grid

% ***** Possibilite d'enregistrer les resultats *****

E=input('Vous voulez enregistre les donnees(o/n):','s');
if E=='o',
nom='a:\test1'
keyboard
eval(['save ' nom ' pv1 pv2  u1 u2 temps ']);
end

% ***** Possibilite d'enregistrer les poids *****

E=input('voulez-vous enregistrer les poids(o/n):','s');
if E=='o',
save weights B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 W1 W2 W3 W4 W5 W6 W7 W8 W9 W10
end
```