Chapitre 4

Régulation directe adaptative et prédictive sur plusieurs pas de temps pour processus à plusieurs entrées et plusieurs sorties

4.1. Introduction

Les régulateurs flous sont reconnus comme une solution viable, face aux régulateurs classiques, pour la commande de processus complexes, non linéaires ou imparfaitement connus. La littérature propose déjà plusieurs approches pour leur conception [LIV 98, WAN 97]. Ce chapitre présente un régulateur flou adaptatif et prédictif sur plusieurs unités de temps, à plusieurs entrées et plusieurs sorties (MSDAFC : *Multiple Step Direct Adaptive Fuzzy Controller*). Un tel régulateur a été utilisé à l'IRIDIA dans le cadre du projet européen : "*Fuzzy Algorithm for the control of Multi-Input, Multi-Output processes* (FAMIMO)" (ESPRIT LTR project 21911).

Nous décrivons, dans ce chapitre, une technique de commande adaptative applicable à des systèmes dont la structure est connue mais dont les paramètres sont inconnus. Ces paramètres peuvent être regroupés dans un vecteur p. Si le système est linéaire et que p est connu, il est possible de trouver immédiatemment le vecteur des paramètres optimaux w^* du régulateur. Si le système est non linéaire, ou si p est inconnu, le vecteur w des paramètres du régulateur doit être ajusté en ligne.

Il existe deux approches distinctes pour la commande adaptative : l'approche directe et l'approche indirecte. Dans la commande adaptative indirecte, les paramètres du

---

Chapitre rédigé par Frank VANDEN BERGHEN et Hugues BERSINI.

processus sont estimés ($\hat{p}$) à chaque instant et les paramètres w du régulateur sont calculés sur base des $\hat{p}$. Les $\hat{p}$ sont assimilés aux vrais paramètres p du processus. Dans ce chapitre, nous nous intéressons à la commande adaptative directe. Les paramètres du régulateur sont ajustés afin de minimiser une fonction de coût donnée. Cet ajustement des paramètres est réalisé *via* un algorithme d'apprentissage ; nous utilisons la descente de gradient.

Le travail présenté ici est inspiré de travaux réalisés dans le cadre de la commande neuronale adaptative [REN 95, REN 96]. Nous proposons une extension des résultats à la commande floue et une généralisation de la procédure.

Les systèmes flous sont des approximateurs universels [CAS 95, YIN 94]. Cette propriété est une des hypothèses nécessaire à l'application de notre algorithme. Les systèmes flous peuvent donc être utilisés en tant que régulateur sous forme de "boîte noire" paramétrisable [REN 94, WAN 93].

Les techniques de régulation floue ont toutes un point commun : l'action de contrôle est obtenue par la combinaison de régulateurs simples (généralement linéaires) définis localement sur l'espace d'entrée du régulateur. L'action de contrôle définie par la conséquence de la règle s'applique dans la zone de l'espace d'entrée limitée par l'antécédent. Lorsque plusieurs règles sont activées en même temps, dans une certaine partie de l'espace, l'action de contrôle appliquée est obtenue par combinaison floue des diverses règles.

Le présent chapitre a été inspiré du travail réalisé par les auteurs de [BER 00], de [BER 96] et de [BAK 97].

4.2. Description du système

La figure 4.1 montre la structure en boucle fermée du système de contrôle. Celui-ci est composé :

– d'un processus à réguler ;

– d'un prédicteur P de ce processus ;

– d'un régulateur flou R.

A chaque appel du régulateur, celui-ci adaptera les paramètres des systèmes flous R_i qui le composent (un système flou par signal de contrôle u_i). Il calculera ensuite les signaux de contrôle (qui sont simplement la sortie des systèmes flous R_i).

Nous voulons réguler un processus à n entrées et q sorties.

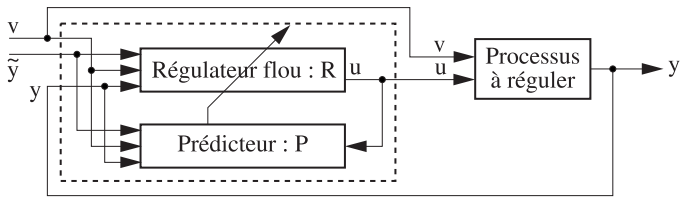

Figure 4.1. *Représentation schématique du système de régulation*

Le régulateur R se présente sous la forme d'une fonction floue entrée-sortie :

$$u_i(t) = R_i[\tilde{y}(t), \psi_i(t); w(t)] \quad 1 \leq i \leq n_1 \qquad [4.1]$$

$$\begin{aligned}\psi_i(t) = [&u(t-1), \ldots, u(t-(n_{rs})_i),\\ &v(t-1), \ldots, v(t-(n_{rv})_i),\\ &y(t), \ldots, y(t-(n_{re})_i+1)]\end{aligned} \qquad [4.2]$$

avec :

– $u(t) \in \Re^{n_1}$: sorties : actions de commande du régulateur au temps t ;

– $v(t) \in \Re^{n_2}$: entrées : perturbations observables du système ;

– $\tilde{y}(t) \in \Re^{q}$: entrées : la consigne voulue au temps t (idéalement $\tilde{y}(t) \simeq y(t)$) ;

– $y(t) \in \Re^{q}$: entrées : la sortie du processus au temps t ;

– $w(t)$: les paramètres du régulateur c'est-à-dire les centres (C_j) et variances (F_j) des ensembles flous et les paramètres L_j des conséquences de chaque règle floue (voir paragraphe 4.3.3.) ;

– $R_i(\ldots)$: le système flou permettant de calculer u_i ;

– $n_{rs} \in \Re^{n_1}, n_{re} \in \Re^{n_1}, n_{rv} \in \Re^{n_1}$: ordres connus du système.

REMARQUE.– Associé à $y(t)$ nous trouvons n_{re} (voir [4.2]). n_{re} est un ordre associé à une entrée du régulateur.

Des délais n_{rde}, n_{rdv} peuvent être intégrés au régulateur, mais ils présentent peu d'intérêt, c'est pourquoi ils ne sont pas repris dans [4.2].

Nous utilisons un prédicteur P sous forme de système flou qui se présente comme suit :

$$y_i(t+1) = P_i[\phi_i(t)], \quad 1 \leq i \leq q \qquad [4.3]$$

$$\begin{aligned}\phi_i(t) = [&y(t), \ldots, y(t-(n_{ps})_i+1),\\ &u(t-(n_{pde})_i+1), \ldots, u(t-(n_{pe})_i-(n_{pde})_i+2),\\ &v(t-(n_{pdv})_i+1), \ldots, v(t-(n_{pd})_i-(n_{pdv})_i+2)]\end{aligned} \qquad [4.4]$$

avec :

– $u(t) \in \Re^{n_1}$: entrées : actions de commande du régulateur au temps t ;

– $v(t) \in \Re^{n_2}$: entrées : perturbations observables du système ;

– $y(t) \in \Re^q$: sorties : prédiction pour le temps t ;

– $P_i(\ldots)$: le système flou permettant de prédire y_i ;

– $n_{ps} \in \Re^q$, $n_{pe} \in \Re^q$, $n_{pv} \in \Re^q$: ordres connus du système ;

– $n_{pde} \in \Re^q$, $n_{pdv} \in \Re^q$: délais connus du système.

REMARQUE.– Les éléments des vecteurs n_{pde} et n_{pdv} doivent être supérieurs à 1 pour avoir un réel sens physique.

EXEMPLE.– Nous voulons réguler un processus à 3 entrées (2 signaux de commande, 1 perturbation observable) et 2 sorties. Si nous incluons le seul signal non contrôlable $v_1(t)$ dans les $u(t)$, nous obtenons ($u_3(t) := v_1(t)$) (pas de n_{pv} et de n_{pdv}) :

– $n_{ps} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$: caractérise les signaux de sortie de P ;

– $n_{pe} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$: caractérise les signaux d'entrée de P ;

– $n_{pde} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$: caractérise le délai sur les signaux d'entrée de P.

Ce qui donne :

$$\begin{aligned} y_1(t+1) = P_1[&y_1(t), y_1(t-1), y_1(t-2), y_2(t), y_2(t-1), y_2(t-2), \\ &u_1(t-1), u_1(t-2), u_2(t-1), u_2(t-2), u_3(t-1), u_3(t-2)] \\ y_2(t+1) = P_2[&y_1(t), y_2(t), u_1(t), u_2(t), u_3(t)] \end{aligned}$$

Soit un régulateur à 3 entrées et 2 sorties. Si nous prenons $n_{re} = n_{pe}$, $n_{rs} = n_{ps}$ (voir ci-dessus) (et $n_{rde} = 0$) nous avons :

$$\begin{aligned} u_1(t) = R_1[&u_1(t-1), u_1(t-2), u_2(t-1), u_2(t-2), u_3(t-1), u_3(t-2)) \\ &y_1(t), y_1(t-1), y_1(t-2), y_2(t), y_2(t-1), y_2(t-2)] \\ u_2(t) = R_2[&u_1(t-1), u_2(t-1), u_3(t-1), y_1(t), y_2(t)] \end{aligned}$$

4.3. Algorithme d'apprentissage : descente de gradient

4.3.1. *Hypothèses requises*

Avant de procéder aux calculs, nous devons faire les hypothèses suivantes :

– il existe une série de signaux de contrôle unique $[u_t, u_{t+1}, \ldots, u_{t+H_c}]$ qui guident y vers $\tilde{y}$ au temps $t + H_c$. H_c est l'horizon de contrôle. Le régulateur doit avoir un horizon de prédiction $H_p \geq H_c$;

– le régulateur flou est capable d'approximer cette loi de contrôle, au degré de précision voulu, dans le domaine voulu (les systèmes flous sont des approximateurs universels) ;

– l'état du processus peut être reconstruit à tout moment à partir de ϕ ;

– la fonction P est continuellement différentiable par rapport à ses arguments. De plus :

$$\frac{\partial P}{\partial u_t} = \frac{\partial R}{\partial u_t} = \frac{\partial P}{\partial y_t} = \frac{\partial R}{\partial y_t} = 0 \quad \forall t < k \qquad [4.5]$$

– la vitesse d'adaptation des paramètres est suffisamment faible pour pouvoir séparer la mesure de l'erreur, d'une part, et des effets de l'ajustement des paramètres d'autre part ;

– le processus est asymptotiquement à minimum de phase.

4.3.2. *Algorithme*

Nous pouvons maintenant développer un algorithme de descente de gradient permettant de trouver/optimiser les paramètres w du régulateur.

La fonction de coût à minimiser est :

$$J_k(w(k)) = \frac{1}{2} \sum_{t=k+1}^{k+H_p} (y(t) - \tilde{y}(t))^T Q_{k,t} (y(t) - \tilde{y}(t)) + \frac{1}{2} \sum_{t=k}^{k+H_p-1} (\Delta u(t))^T R_{k,t} (\Delta u(t)) \qquad [4.6]$$

avec :

$$\Delta u(t) = u(t) - u(t-1) \qquad [4.7]$$

Les matrices $Q_{k,t} \in \Re^{q \times q}$ pondèrent les erreurs $(y(t) - \tilde{y}(t))$ de suivi de la consigne. Les matrices $R_{k,t} \in \Re^{n_1 \times n_1}$ empêchent des variations $(\Delta u(t))$ trop brutales

de la consigne et stabilisent le système. Généralement, $Q_{k,t} = I_q$ (I = matrice unité) et $R_{k,t} = 0{,}01 I_{n_1}$.

Remarquons que la fonction de coût fait intervenir des valeurs pour $y(t)$ et $u(t)$ avec $t > k$. C'est-à-dire, elle utilise des valeurs inconnues des variables. Nous devons prédire l'évolution de ces variables. Nous utiliserons P qui est un prédicteur du processus à réguler pour simuler la boucle fermée sur un horizon H_p. Cette simulation de la boucle fermée est pour $t = k$ jusqu'à $k + H_p$:

– construire $\psi(t)$;

– $u(t) = R(\tilde{y}(t), \psi(t); w(t))$;

– construire $\phi(t)$ en tenant compte des saturations sur $u(t)$;

– $y(t+1) = P(\phi(t))$.

A chaque pas de temps k, le régulateur réduit l'erreur J_k grâce à une descente de gradient dans l'espace des paramètres :

$$w(k) = w(k-1) - \eta \frac{\delta J_k}{\delta w} \qquad [4.8]$$

avec :

– $w(k) \in \Re$: la valeur à l'instant k du paramètre en cours d'optimisation. Ce paramètre est choisi parmi l'ensemble des paramètres des systèmes flous R_i qui composent le régulateur ;

– $\eta \ll 1$: la vitesse d'apprentissage.

Nous avons :

$$w(k) = w(k-1) - \eta \left(\sum_{t=k+1}^{k+H_p} (y(t) - \tilde{y}(t))^T Q_{k,t} \frac{\delta y(t)}{\delta w} + \sum_{t=k}^{k+H_p-1} (\Delta u(t))^T R_{k,t} \frac{\delta(\Delta u(t))}{\delta w} \right) \qquad [4.9]$$

$$\begin{aligned} \frac{\delta u(t)}{\delta w} &= \frac{\delta R(\tilde{y}(t), \psi(t); w)}{\delta w} \\ &= \left[\frac{\delta R_1(\tilde{y}(t), \psi_i(t); w)}{\delta w} \cdots \frac{\delta R_{n_1}(\tilde{y}(t), \psi_i(t); w)}{\delta w} \right]^T \\ &= \left[\frac{\delta R_1(t)}{\delta w} \cdots \frac{\delta R_{n_1}(t)}{\delta w} \right]^T \end{aligned} \qquad [4.10]$$

$$\frac{\delta R_i(t)}{\delta w} = \frac{\delta u_i(t)}{\delta w} \qquad [4.11]$$

$$= \frac{\partial R_i}{\partial w} + \frac{\partial R_i}{\partial \psi_i(t)} \frac{\delta \psi_i(t)}{\delta w}$$

$$= \frac{\partial R_i}{\partial w} + \sum_{j=1}^{q} \sum_{k=0}^{(n_{re})_i - 1} \frac{\partial R_i}{\partial y_j(t-k)} \frac{\delta y_j(t-k)}{\delta w}$$

$$+ \sum_{j=1}^{n_1} \sum_{k=0}^{(n_{rs})_i - 1} \frac{\partial R_i}{\partial u_j(t-k-1)} \frac{\delta u_j(t-k-1)}{\delta w} \qquad [4.12]$$

$$\frac{\partial y(t+1)}{\partial w} = \frac{\partial P(\phi(t))}{\partial w}$$

$$= \left[\frac{\partial P_1(\phi(t))}{\partial w} \cdots \frac{\partial P_q(\phi(t))}{\partial w} \right]^T$$

$$= \left[\frac{\partial P_1(t)}{\partial w} \cdots \frac{\partial P_q(t)}{\partial w} \right]^T \qquad [4.13]$$

$$\frac{\delta P_i(t)}{\delta w} = \frac{\delta y_i(t+1)}{\delta w} \qquad [4.14]$$

$$= \frac{\partial P_i}{\partial \phi(t)} \frac{\delta \phi(t)}{\delta w}$$

$$= \sum_{j=1}^{q} \sum_{k=0}^{(nps)_i - 1} \frac{\partial P_i}{\partial y_j(t-k)} \frac{\delta y_j(t-k)}{\delta w}$$

$$+ \sum_{j=1}^{n_1} \sum_{k=0}^{(n_{pe})_i - 1} \frac{\partial P_i}{\partial u_j(t-k-(n_{pde})_i+1)} \frac{\delta u_j(t-k-(n_{pde})_i+1)}{\delta w} \qquad [4.15]$$

Les deux équations importantes sont [4.12] et [4.15]. A l'intérieur de ces deux équations est cachée une récurrence. En effet, elles font intervenir des termes en $\delta y/\delta w$ et en $\delta u/\delta w$ qui doivent être développés suivant respectivement [4.11] et [4.14].

Remarquons, aussi, la notation employée pour écrire les dérivées dans les équations [4.12] et [4.15]. Cette notation fait apparaître, clairement, la différence entre les dérivées partielles (∂) et les dérivées totales(δ). Lors d'une dérivée partielle (∂), la récurrence s'arrête. Pour une dérivée totale (δ), nous devons continuer le développement, ce qui représente une lourde charge de calcul. L'essentiel du temps de calcul lors de l'adaptation est utilisée pour trouver ces dérivées (δ).

Si nous voulons éviter une charge de calcul trop importante, nous commençons par calculer les $\delta y(t)/\delta w$ et les $\delta u(t)/\delta w$ à partir de $t = k$ car, à ce moment, le calcul est très simple d'après l'équation [4.5]. Ensuite, nous calculons les dérivées pour $t > k$ sur la base des dérivées précédentes soigneusement gardées en mémoire. Il est hors de question d'effectuer les calculs selon un algorithme récursif (c'est-à-dire en partant de $t = k + H_p$ et ensuite en « remontant » vers $t = k$).

Nous venons de transformer un algorithme récursif (lourde charge de calcul) en un algorithme itératif (faible charge de calcul), au prix d'une consommation mémoire somme toute faible.

4.3.3. *Dérivées d'un système flou*

Les équations [4.12] et [4.15] font intervenir les dérivées partielles d'un système flou par rapport à un paramètre interne et par rapport à une de ses entrées (gradient). Ce paragraphe détaille le calcul de ces dérivées.

Soit un système flou de Tagaki-Sugeno R à une sortie ($y \in \Re$), n entrées ($u \in \Re^n$) et r règles avec $\mathcal{E}_j (j = 1, \ldots, r)$, un ensemble flou de forme gaussienne, caractérisé par son centre $C_j \in \Re^n$ et par son étalement $F_j \in \Re^{n \times n}$.

$$\begin{cases} \text{si } (U \text{ est dans } \mathcal{E}_j) \text{ alors } y_j = \sum_{i=1}^{n} U_i . L_{i,j} + L_{n+1,j}, \\ a_j = (\mu_{\mathcal{E}_j}(U)) = \frac{1}{e^{(U-C_j)^T F_j (U-C_j)}}, \end{cases} \qquad j = 1, \ldots, r \qquad [4.16]$$

$$y = \frac{\sum_{j=1}^{r} a_j y_j}{\sum_{j=1}^{r} a_j} \qquad [4.17]$$

Si nous calculons analytiquement le gradient de R, nous obtenons :

$$\frac{\partial R}{\partial u_k}(U) = \frac{1}{\sum_{j=1}^{r} a_j} \sum_{j=1}^{r} \left[a_j \cdot L_{j,k} - \left(2. \sum_{s=1}^{n} (F_j)_{k,s} \cdot (U - C_j)_s \right) (y_j - R(U)) \right] \qquad [4.18]$$

La figure 4.2 est une représentation graphique d'un système flou à une entrée u et une sortie y, composé de deux règles floues. La fonction d'appartenance de la première règle est $\mu_1(x)$ et sa sortie locale est $y_1 = a_1u + h_1$. Nous pouvons voir sur la partie gauche du schéma 4.2 le résultat obtenu en utilisant [4.18].

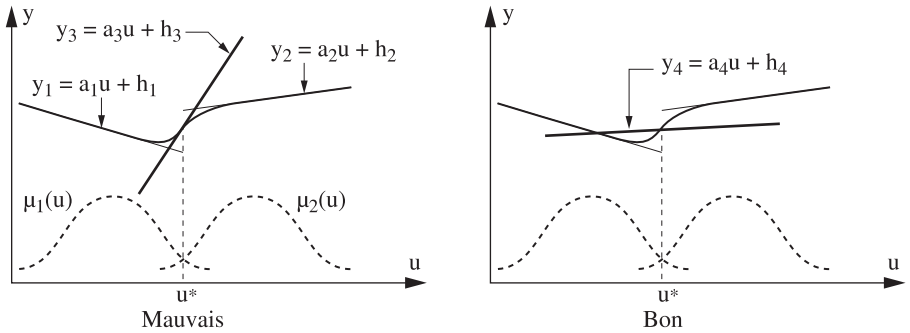

Figure 4.2. *Deux définitions pour le calcul du gradient en u^* d'un système flou*

La méthode correcte pour le calcul du gradient (vue sur la partie droite de la figure 4.2) consiste à calculer la combinaison floue des gradients locaux (de chaque règle). Ce qui donne :

$$\frac{\partial R}{\partial u_k}(U) = \frac{1}{\sum_{j=1}^{r} a_j} \sum_{j=1}^{r} \left[a_j . L_{j,k} \right] \qquad [4.19]$$

La dérivée d'un système flou par rapport à un de ses paramètres ne pose pas de problème. Nous obtenons les dérivées suivantes :

$$\frac{\partial R}{\partial L_{k,l}}(U) = \frac{a_k}{\sum_{j=1}^{r} a_j} . \begin{cases} U_l & \text{si } (l \leq n) \\ 1 & \text{si } (l = n+1) \end{cases} \qquad [4.20]$$

$$\frac{\partial R}{\partial (C_k)_l}(U) = \frac{2 \cdot a_k}{\sum_{j=1}^{r} a_j} \left(\sum_{s=1}^{n} (F_k)_{l,s} \cdot (U - C_k)_s \right) (y_k - R(U)) \qquad [4.21]$$

$$\frac{\partial R}{\partial (F_k)_{l,m}}(U) = \frac{-a_k}{\sum_{j=1}^{r} a_j} (U - C_k)_l \cdot (U - C_k)_m (y_k - R(U)) \qquad [4.22]$$

4.3.4. *Remarque sur le prédicteur*

Il est possible d'employer comme prédicteur, non pas un système flou, mais n'importe quelles fonctions décrivant le processus. Pour pouvoir appliquer l'algorithme

d'apprentissage, il est cependant nécessaire de pouvoir calculer le gradient de ce prédicteur. Des tests ont été réalisés à l'IRIDIA sur la base de prédicteurs linéaires $AU = Y$, de prédicteurs *Lazy* et de prédicteurs basés sur des *Lookup-Tables*.

Le régulateur est fortement sensible à la qualité du prédicteur. C'est un point faible de notre technique.

4.4. Applications en simulation

L'approche MSDAFC a été testée sur deux systèmes différents : un *toy problem* et un traitement des eaux usées. Ce dernier est utilisé par tous les partenaires du projet FAMIMO [YOU 96].

4.4.1. *Toy problem*

Le système à réguler est le suivant (système non linéaire, à non-minimum de phase) :

$$y_{k+1} = y_k + sin(3 * u_{k-1}) - 3u_k \qquad [4.23]$$

Un régulateur basé sur une prédiction d'une unité de temps dans le futur ne pourra contrôler ce système car celui-ci est à non-minimum de phase. Il est donc nécessaire d'utiliser un algorithme qui peut prédire le comportement du système sur plusieurs occurences dans le futur.

Le processus est modélisé grâce à un système flou Takagi-Sugeno. Celui-ci a été identifié en utilisant l'algorithme de Gustafson-Kessel [GUS 79] pour calculer la position des centres et les conséquences de chaque règle floue. L'identification a été réalisée sur une base de 5 000 points, obtenus en excitant le processus avec un signal pseudo-aléatoire sinusoïdal. Le régulateur est, lui, constitué de 9 règles floues. Les conséquences ($L_{i,j}$) ont été initialisées à 0. La vitesse d'apprentissage est de $\eta = 0{,}0005$. Un horizon de prédiction de 5 pas en avant a été utilisé pour le calcul de la mise à jour des paramètres des conséquences des règles floues du régulateur. Le résultat de la simulation est présenté sur la figure 4.3.

Dans cet exemple, il est intéressant de noter la forme typique de la réponse d'un système à non-minimum de phase. A chaque changement de consigne, le système part dans la direction opposée à la consigne. Ce système, impossible à réguler en utilisant une prédiction à un pas, est facilement contrôlé en utilisant une prédiction sur plusieurs pas.

4.4.2. *Le problème du traitement des eaux usées*

Un système biologique constitué d'un bioréacteur continu, à ciel ouvert, utilisé pour le traitement des eaux usées dans l'industrie du papier, est la cible de notre seconde

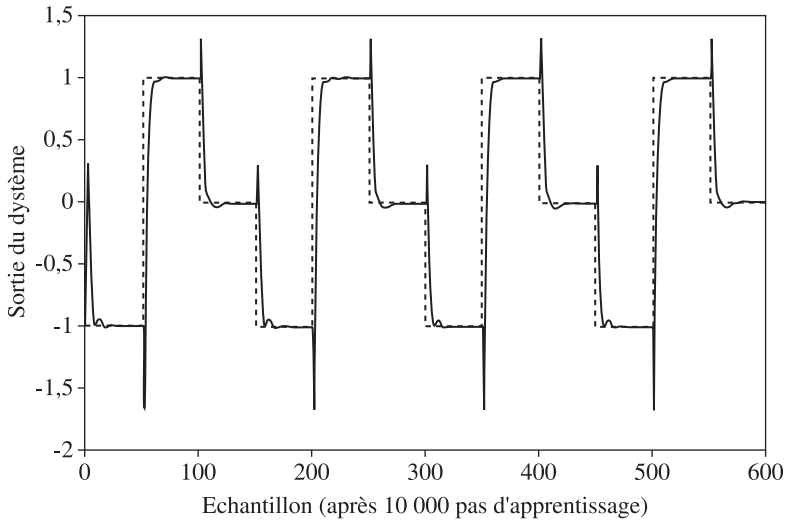

Figure 4.3. *Performance du MSDAFC sur un système à non-minimum de phase après 10 000 unités de temps*

étude expérimentale. Ce réacteur contient une certaine masse de bio-organismes et deux substrats : le *xenobiotic polluant substrate* et l'*energetic substrate*. Nous avons à notre disposition un modèle simulink du réacteur développé par le LAAS, laboratoire d'analyse et d'architecture des systèmes du CNRS [YOU 96]. Nous devons réguler la concentration résiduelle de *xenobiotic polluant substrate* et la prolifération des bio-organismes (biomasse), pour empêcher toutes nuisances bactériologiques.

La concentration du *energetic substrate* ne doit pas être régulée car celui-ci est facilement dégradable. Le processus est caractérisé par sa haute non-linéarité, sa dynamique changeante et des problèmes de couplage entre les variables. De plus, nous ne disposons pas de capteur pour connaître la masse du *xenobiotic polluant substrate*. Celle-ci est donc obtenue grâce à un observateur asymptotique. Le volume d'eau dans le réacteur est maintenu constant. Il est possible de réguler la concentration en substrats en jouant sur les deux débits d'eau entrant dans le réacteur. La première arrivée d'eau provient d'un réservoir d'eau pure et la seconde provient d'un réservoir d'eau polluée (voir figure 4.4).

Soit :

– $c(t)$ [g/l] : (entrée ; à réguler) concentration de la biomasse. Les micro-organismes dégradent les deux substrats présents dans le réacteur ;

– $s(t)$ [g/l] : (à réguler) concentration du *xenobiotic polluant substrate* ;

– $e(t)$ [g/l] : (entrée) concentration du *energetic substrate* ;

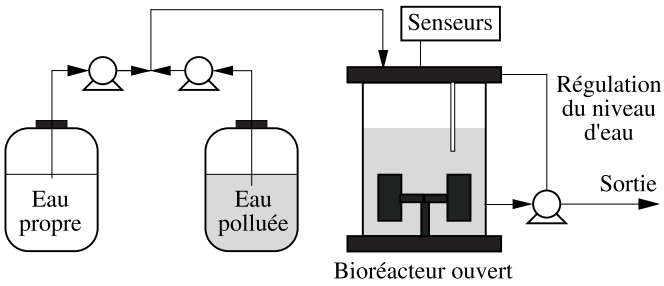

Figure 4.4. *Représentation schématique du bioréacteur pour le traitement des eaux usées*

– $u_1(t)$ [l/s] : (sortie) débit d'eau propre entrant dans le réacteur ;

– $u_2(t)$ [l/s] : (sortie) débit entrant d'eau polluée. Cette eau polluée contient une certaine concentration maximale de substrats : s_a^{max} [g/l] et e_a^{max} [g/l] ;

– μ_{sm} et μ_{em} : le taux de croissance spécifique maximum des deux substrats ;

– $Y_{c/s}$ et $Y_{c/e}$: coefficients liés correspondant à la conversion des substrats en biomasse ;

– K_s et K_e : constantes de Michaelis-Menten correspondant à l'affinité de la population microbienne pour chaque substrat ;

– a_s et a_e : constantes inhibitoires : a_e est l'influence inhibitoire du *energetic substrate* sur la dégradation du *xenobiotic polluant substrate* (et *vice versa* pour a_s).

Les équations du système sont :

$$\begin{cases} \dot{c}(t) = \left\{ \mu_{sm} \dfrac{s(t)}{K_s + s(t) + a_e e(t)} + \mu_{em} \dfrac{e(t)}{K_e + e(t) + a_s s(t)} \right\} \\ \qquad \times c(t) - u_1(t)c(t) - u_2(t)c(t) \\ \dot{s}(t) = -\dfrac{u_{sm}}{Y_{c/s}} \dfrac{s(t)}{K_s + s(t) + a_e e(t)} c(t) - u_1(t)s(t) + \left(s_a^{max} - s(t)\right) u_2(t) \\ \dot{e}(t) = -\dfrac{u_{em}}{Y_{c/e}} \dfrac{e(t)}{K_e + e(t) + a_s s(t)} c(t) - u_1(t)e(t) + \left(e_a^{max} - e(t)\right) u_2(t) \end{cases} \quad [4.24]$$

Le processus est modélisé grâce à un système flou de Takagi-Sugeno. Celui-ci a été identifié en utilisant l'algorithme de Gustafson-Kessel [GUS 79] pour calculer la position des centres et les conséquences de chaque règle floue. L'identification a été réalisée sur une base de 5 000 points obtenus en excitant le processus avec un signal pseudo-aléatoire sinusoïdal. Le prédicteur pour la biomasse est constitué de quatre

règles floues. Le prédicteur pour la concentration en *xenobiotic substrate* est constitué de deux règles floues. Le régulateur est, lui, constitué de deux systèmes flous (un système pour chaque sortie) chacun constitué d'une règle floue. Les conséquences ($L_{i,j}$) des règles du régulateur ont été initialisées à 0.

L'adaptation a été réalisée en utilisant trois horizons de prédiction différents : 1, 3 et 10. Les résultats avec un horizon de 1 sont trop médiocres et ne seront pas présentés. Sur les figures 4.5 et 4.6, on peut voir le résultat de la simulation pour les autres horizons. La vitesse d'apprentissage est fonction de l'horizon de prédiction. Elle vaut $\eta = 0{,}05$ pour un horizon de 3 et $\eta = 0{,}005$ pour un horizon de 10. Seuls les paramètres des conséquences des règles floues du régulateur sont mis à jour. Les capteurs à l'intérieur du réacteur sont perturbés par un bruit d'amplitude compris entre -2 % et $+2$ % de la valeur de sortie. Des variations des constantes cinétiques biologiques ont été appliquées pour pouvoir tenir compte de la nature imprécise de la dynamique du système.

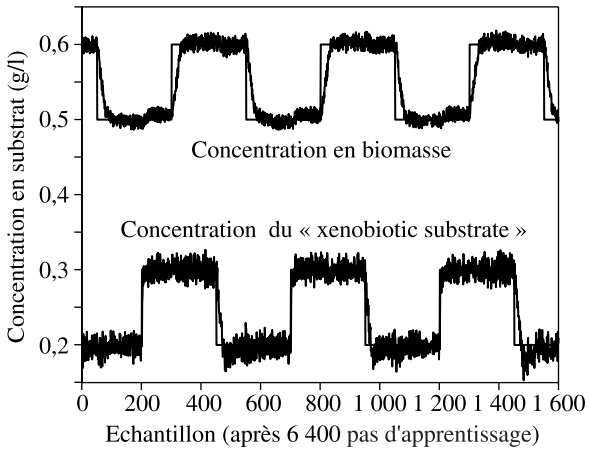

Figure 4.5. *Performance du MSDAFC avec un horizon de prédiction de 3 sur le bioréacteur après 6 400 unités de temps*

Nous pouvons constater sur les figures 4.5 et 4.6 qu'après un temps d'apprentissage suffisament long (6 400 unités de temps), les deux systèmes donnent de bons résultats et cela malgré des conditions fort bruitées. La principale différence entre les deux algorithmes tient dans le temps nécessaire pour apprendre la loi de contrôle optimale. La méthode basée sur un horizon de prédiction de 10 « apprend » plus vite que celle basée sur un horizon de 3 car, à chaque unité de temps, elle regarde plus loin et peut donc retirer plus d'informations du système. Malheureusement, elle exige plus de puissance de calcul (à chaque unité de temps, il faut plus de temps pour mettre à jour les paramètres du régulateur que lorsque l'horizon est de 3). Cela implique que, bien que l'apprentissage soit plus rapide, la simulation pourra prendre plus de temps que sa concurrente.

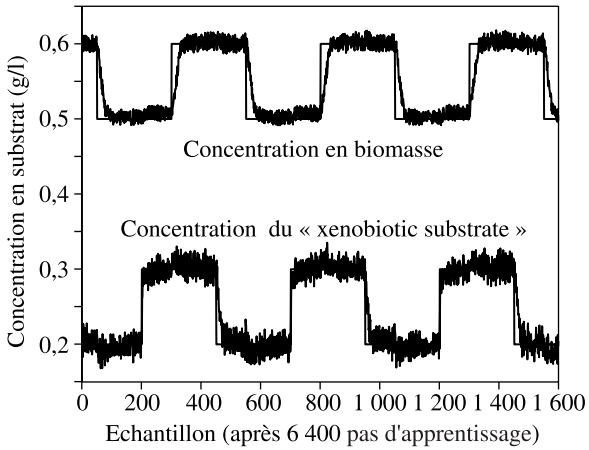

Figure 4.6. *Performance du MSDAFC avec un horizon de prédiction de 10 sur le bioréacteur après 6 400 unités de temps*

Nous avons aussi utilisé un algorithme d'apprentisssage de type « Levenberg-Marquardt ». Le temps d'apprentissage du régulateur (le temps mis pour arriver à un contrôle du processus satisfaisant) avec l'algorithme de Levenberg-Marquardt est plus court que pour un algorithme basé sur une simple descente de gradient. Une fois l'apprentissage terminé, « Levenberg-Marquardt » ne donne pas des résultats de contrôle significativement meilleurs que la « descente de gradient ».

4.5. Conclusion

Les algorithmes adaptatifs présentés ici sont très proches des algorithmes présents dans les régulateurs neuronaux. Ces algorithmes adaptatifs peuvent fonctionner à partir de paramètres initiaux aléatoires ; mais, si le régulateur de départ est déjà correctement initialisé, l'adaptation est plus rapide et donne de meilleurs résultats. Il est facile, dans le cas de systèmes flous, de retirer de l'information d'un expert et de l'intégrer dans un régulateur. En effet, la structure d'un système flou est facilement compréhensible pour un humain. L'initialisation partielle d'un régulateur est donc possible. Cette facilité ne se retrouve pas, par exemple, dans les réseaux de neurones. C'est pourquoi les systèmes flous sont d'excellents candidats pour réaliser des régulateurs dits « boîtes noires ».

Grâce à sa faculté d'apprentissage constant, le MSDAFC est capable de résorber rapidement des perturbations accidentelles qui modifieraient le comportement du processus à régler.

Il reste certains problèmes à résoudre, notamment la grande sensibilité du processus d'apprentissage face à la qualité du prédicteur du processus. Nous pensons qu'il est possible de perfectionner l'algorithme de manière à le rendre plus robuste.

4.6. Bibliographie

[BAK 97] BAKUŠKA R., Fuzzy Modeling and Identification, PhD thesis, TU Delft, Delft, Pays-Bas, 1997.

[BER 96] BERSINI H., « Adaptive fuzzy control for the state feedback optimal control », *Proceedings of the $XIII^{th}$ IFAC World Congress*, San Francisco, juillet 1996.

[BER 00] BERTOLISSI E., DUCHATEAU A., BERSINI H., BERGHEN F.V., « Direct Fuzzy Control for MIMO Processes », *Proceedings FUZZ-IEEE 2000*, San Antonio, Texas, 7-10 mai 2000.

[CAS 95] CASTRO J., « Fuzzy logic controllers are universal approximtors », *IEEE Trans. on Systems, Man and Cybernetics*, vol. 25, p. 629-635, 1995.

[GUS 79] GUSTAFSON D.E., KESSEL W.C., « Fuzzy clustering with a fuzzy covariance matrix », *Proc. IEEE CDC*, San Diego, Etats-Unis, p. 761–776, 1979.

[LIV 98] LIVCHITZ M., AVERSHITZ A., SOUDAK U., KANDEL A., « Development of an automated fuzzy-logic-based expert system for unmanned landing », *Fuzzy Sets and Systems*, vol. 2, p. 145-159, 1998.

[REN 94] RENDERS J.M., SAERENS M., BERSINI H., On the stability of direct adaptive fuzzy controllers ; Rapport IRIDIA/ULB, Bruxelles, 1994.

[REN 95] RENDERS J.M., SAERENS M., BERSINI H., dans Hunt K.J., Irvin G.R., Warwick K., (Eds.), « Adaptive neurocontrol of a certain class of MIMO discrete-time processes based on stability theory », *Neural Network Engineering in Dynamic Control Systems*, Springer Verlag, p. 43-60, 1995.

[REN 96] RENDERS J.M., SAERENS M., BERSINI H., « Neurocontrol based on the backpropagation algorithm », dans M.M. Gupta, N.K. Sinha (dir.) *Intelligent Control Systems*, p. 292-326, IEEE Press, 1996.

[WAN 93] WANG L.-X., « Stable adaptive fuzzy control of nonlinear systems », *IEEE Trans. on Fuzzy Systems*, vol. 1, p. 146-155, 1993.

[WAN 97] WANG L.-X., « Combining mathematical model & heuristics into controllers, an adaptive fuzzy control approach », *Fuzzy Sets and Systems*, vol. 89, p. 151-156, 1997.

[YIN 94] YING H., « Sufficient conditions on general fuzzy systems as function approximators », *Automatica*, vol. 30, p. 521-525, 1994.

[YOU 96] YOUSSEF C.B., Filtrage, estimation et commande adaptive d'un procédé de traitement des eaux usuées, PhD thesis, Institut national polytechnique, Toulouse, 1996.