Régulation directe adaptative et prédictive sur plusieurs pas de temps pour la commande floue de processus à plusieurs entrées et plusieurs sorties.

Frank Vanden Berghen,Edy Bertolissi, Antoine Duchâteau, Hugues Bersini
IRIDIA, Université Libre de Bruxelles
50, av. Franklin Roosevelt
1050 Brussels, Belgium
{fvandenb, eberto, aduchate, bersini}@iridia.ulb.ac.be

23 mars 2000

Résumé

Ce rapport présente les algorithmes d'apprentissage utilisés dans les régulateurs flous dévellopés à IRIDIA dans le cadre du projet FAMIMO (ESPRIT LTR Project 21911).

1 Introduction

Les régulateurs flous sont reconnus comme une solution viable face aux régulateurs classiques pour la commande de processus complexes, non-linéaire ou imparfaitement connus. La littérature propose déjà plusieurs approches pour leur conception ([**?**, **?**]). 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 IRIDIA dans le cadre du "projet FAMIMO".

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 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 (Pour plus de détails, voir le chapitre précédent). 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 [**?**, **?**]. Nous proposons une extension des résultats à la commande floue et une généralisation de la procédure.

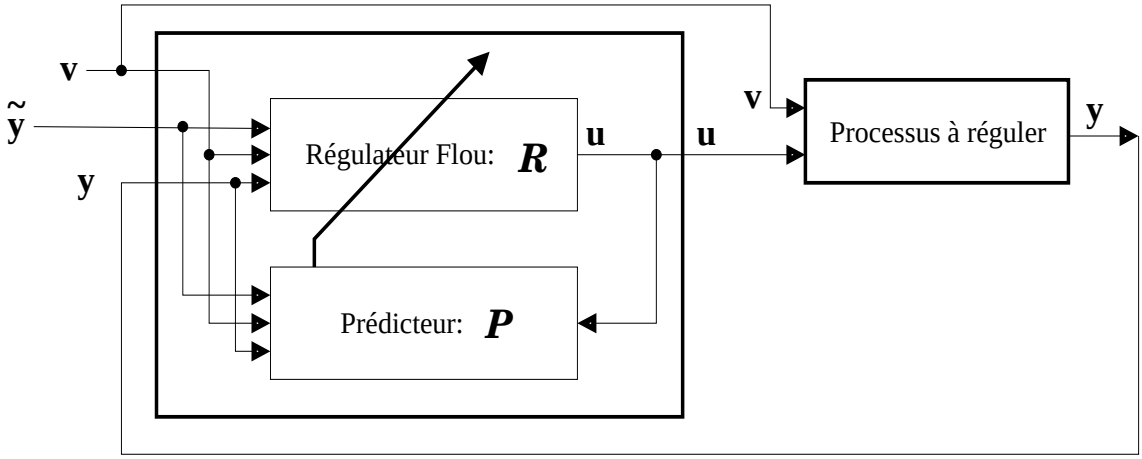

FIG. 1 – Représentation schématique du système de régulation.

Les système flous sont des approximateurs universels [**?**, **?**]. Cette propiété 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 [**?**, **?**].

Les techniques de régulation floue ont toutes un point commun : l'action de contrôle est obtenue par la combinaison de régulateurs simples (généralement linéraires) 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 "Direct Adaptive Fuzzy Control for MIMO Processes" [**?**], de "adaptive fuzzy controller for state-feedback optimal control" [**?**] et de "model based predictive control scheme" [**?**].

2 Description du système.

La figure 1 montre la structure en boucle fermée du système de contrôle. Celui est composé de :
– un processus à régler.
– un prédicteur P de ce processus.
– 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 régler un processus à n entrées et q sorties.

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

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

$$\begin{array}{lllll} \psi_i(t) = [& u(t-1) & , \ldots , & u(t-(n_{rs})_i - 1) & , \\ & v(t-1) & , \ldots , & v(t-(n_{rv})_i - 1) & , \\ & y(t) & , \ldots , & y(t-(n_{re})_i) &] \end{array} \tag{2}$$

avec $y(t-x) = [y_1(t-x_1) \ldots y_q(t-x_q)] \;\; (y, x \in \Re^q)$ et de même pour v(t-x) et u(t-x).

et 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 (idéallement $\tilde{y}(t) \simeq y(t)$).
$y(t) \in \Re^q$: entrées : la sortie du processus à au temps t.
$w(t)$: les paramètres du régulateur càd les centres (C_j) et variances (F_j) des ensembles flous et les paramètres L_j des conséquences de chaque règle floue (voir section 3.3.)
$R_i(\ldots)$: le système flou permettant de calculer u_i.
$n_{rs} \in \Re^{n_1 \times q}, n_{re} \in \Re^{n_1 \times n_1}, n_{rv} \in \Re^{n_1 \times n_2}$: ordres connus du système.

NOTE : associé à $y(t)$ nous trouvons n_{re} (voir 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 2.

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

$$y_i(t+1) = P_i[\phi_i(t)] \; 1 \leq i \leq q \tag{3}$$

$$\begin{array}{lllll} \phi_i(t) = [& y(t) & , \ldots , & y(t-(nps)_i) & , \\ & u(t-(n_{pde})_i + 1) & , \ldots , & u(t-(n_{pe})_i - (n_{pde})_i + 1) & , \\ & v(t-(n_{pdv})_i + 1) & , \ldots , & v(t-(n_{pd})_i - (n_{pdv})_i + 1) &] \end{array} \tag{4}$$

avec $y(t-x) = [y_1(t-x_1) \ldots y_q(t-x_q)] \;\; (y, x \in \Re^q)$ et de même pour v(t-x) et u(t-x).

et 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.
$nps \in \Re^{q \times q}, n_{pe} \in \Re^{q \times n_1}, n_{pv} \in \Re^{q \times n_2}$: ordres connus du système.
$n_{pde} \in \Re^{q \times n_1}, n_{pdv} \in \Re^{q \times n_2}$: délais connus du système.

NOTE : Les éléments des matrices n_{pde} et n_{pdv} doivent être supérieur à 1 pour avoir un réel sens physique.

EXEMPLE.

Soit un processus à 3 entrées (2 signaux de contrôles, 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}) :

- $n_{ps} = \begin{pmatrix} 2 & 3 \\ 1 & 0 \end{pmatrix}$: caractérise les signaux de sortie
- $n_{pe} = \begin{pmatrix} 2 & 1 & 2 \\ 3 & 0 & 1 \end{pmatrix}$: caractérise les signaux d'entrée
- $n_{pde} = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 1 & 2 \end{pmatrix}$: caractérise le délai sur les signaux d'entrée

Ce qui donne :

$$\begin{aligned} y_1(t+1) &= P_1[\ y_1(t), y_1(t-1), y_2(t), y_2(t-1), y_2(t-2), \\ &\qquad u_1(t-1), u_1(t-2), u_2(t), u_3(t-2), u_3(t-3)] \quad (5) \\ y_2(t+1) &= P_2[\ y_1(t), \\ &\qquad u_1(t), u_1(t-1), u_1(t-2), u_3(t-1)] \quad (6) \end{aligned}$$

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

$$\begin{aligned} u_1(t) &= R_1[\ u_1(t-1), u_1(t-2), u_2(t-1), u_3(t-1), u_3(t-2), \\ &\qquad y_1(t), y_1(t-1), y_2(t), y_2(t-1), y_2(t-2)] \quad (7) \\ u_2(t) &= R_2[\ u_1(t-1), u_1(t-2), u_1(t-3), u_3(t-1), \\ &\qquad y_1(t)] \quad (8) \end{aligned}$$

3 Algorithme d'apprentissage : Descente de gradient.

3.1 hypothèses requises.

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

1. Supposons qu'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 $H_p \geq H_c$.
2. Le régulateur flou est capable d'approximer cette loi de contrôle (1.), au degré de précision voulu dans le domaine voulu. (Les systèmes fuzzy sont des approximateurs universels.)
3. L'état du processus peut être reconstruit à tout moment à partir de ϕ.
4. La fonction P est continuellement différentiable par rapport à ses arguments. De plus,
$$\frac{\partial P}{\partial u_t} = \frac{\partial R}{\partial u_t} = \frac{\partial P}{\partial y_t} = \frac{\partial R}{\partial y_t} = 0 \ \forall t < k. \quad (9)$$
5. La vitesse d'adaptation des paramètres est suffisamment faible que pour pouvoir séparer la mesure de l'erreur d'une part, des effets de l'ajustement des paramètres d'autre part.
6. Le processus est asymptotiquement à minimum de phase.

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 :

$$J_k(w(k)) = \frac{1}{2}\sum_{t=k}^{k+H_p}(y(t)-\tilde{y}(t))^T Q_{k,t}(y(t)-\tilde{y}(t)) + \frac{1}{2}\sum_{t=k-1}^{k+H_p}(\Delta u(t))^T R_{k,t}(\Delta u(t)) \tag{10}$$

avec :

$$\Delta u(t) = u(t) - u(t-1) \tag{11}$$

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 brutale de la consigne et stabilisent le système. Généralement, $Q_{k,t} = I_q$ (I =matrice unité) et $R_{k,t} = 0.01\, I_{n_1}$.

Remarquez 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 régler pour simuler la boucle fermée sur un horizon H_p. Cette simulation est décrite sur la figure 2.

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 $u(t)$
- $y(t+1) = P(\phi(t))$

FIG. 2 – simulation de la boucle fermée.

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

$$w(k) = w(k-1) - \eta\frac{\delta J_k}{\delta w} \tag{12}$$

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 :

$$w(k) = w(k-1) - \eta\left(\sum_{t=k}^{k+H_p}(y(t)-\tilde{y}(t))^T Q_{k,t}\frac{\delta y(t)}{\delta w} + \sum_{t=k-1}^{k+H_p}(\Delta u(t))^T R_{k,t}\frac{\delta(\Delta u(t))}{\delta w}\right) \tag{13}$$

$$\begin{aligned}\frac{\delta u(t)}{\delta w} &= \frac{\delta \boldsymbol{R}(\tilde{y}(t), \psi(t); w)}{\delta w} \\ &= \left[\frac{\delta \boldsymbol{R}_1(\tilde{y}(t), \psi_i(t); w)}{\delta w} \cdots \frac{\delta \boldsymbol{R}_{n_1}(\tilde{y}(t), \psi_i(t); w)}{\delta w}\right]^T \\ &= \left[\frac{\delta \boldsymbol{R}_1(t)}{\delta w} \cdots \frac{\delta \boldsymbol{R}_{n_1}(t)}{\delta w}\right]^T \end{aligned} \tag{14}$$

$$\frac{\delta \boldsymbol{R}_i(t)}{\delta w} = \frac{\delta u_i(t)}{\delta w} \tag{15}$$

$$\begin{aligned} &= \frac{\partial \boldsymbol{R}_i}{\partial w} + \frac{\partial \boldsymbol{R}_i}{\partial \psi_i(t)} \frac{\delta \psi_i(t)}{\delta w} \\ &= \frac{\partial \boldsymbol{R}_i}{\partial w} + \sum_{j=1}^{q} \sum_{k=0}^{(n_{re})_{i,j}-1} \frac{\partial \boldsymbol{R}_i}{\partial y_j(t-k)} \frac{\delta y_j(t-k)}{\delta w} + \\ &\qquad \sum_{j=1}^{n_1} \sum_{k=0}^{(n_{rs})_{i,j}-1} \frac{\partial \boldsymbol{R}_i}{\partial u_j(t-k-1)} \frac{\delta u_j(t-k-1)}{\delta w} \end{aligned} \tag{16}$$

$$\begin{aligned}\frac{\partial y(t+1)}{\partial w} &= \frac{\partial \boldsymbol{P}(\phi(t))}{\partial w} \\ &= \left[\frac{\partial \boldsymbol{P}_1(\phi(t))}{\partial w} \cdots \frac{\partial \boldsymbol{P}_q(\phi(t))}{\partial w}\right]^T \\ &= \left[\frac{\partial \boldsymbol{P}_1(t)}{\partial w} \cdots \frac{\partial \boldsymbol{P}_q(t)}{\partial w}\right]^T \end{aligned} \tag{17}$$

$$\frac{\delta \boldsymbol{P}_i(t)}{\delta w} = \frac{\delta y_i(t+1)}{\delta w} \tag{18}$$

$$\begin{aligned} &= \frac{\partial \boldsymbol{P}_i}{\partial \phi(t)} \frac{\delta \phi(t)}{\delta w} \\ &= \sum_{j=1}^{q} \sum_{k=0}^{(nps)_{i,j}-1} \frac{\partial \boldsymbol{P}_i}{\partial y_j(t-k)} \frac{\delta y_j(t-k)}{\delta w} + \\ &\qquad \sum_{j=1}^{n_1} \sum_{k=0}^{(n_{pe})_{i,j}-1} \frac{\partial \boldsymbol{P}_i}{\partial u_j(t-k-(n_{pde})_{i,j}+1)} \frac{\delta u_j(t-k-(n_{pde})_{i,j}+1)}{\delta w} \end{aligned} \tag{19}$$

Les deux équations importantes sont 16 et 19. A l'intérieur de ces deux équations est cachée une récurrence. En effet, elles font intervenir des termes en $\frac{\delta y}{\delta w}$ et en $\frac{\delta u}{\delta w}$ qui doivent être développés suivant respectivement 15 et 18.

Remarquons aussi la notation employée pour écrire les dérivées dans les équations 16 et 19. Cette notation fait apparaître clairement la différence entre les dérivées partielle (∂) 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 calul trop importante, nous commencerons par calculer les $\frac{\delta y(t)}{\delta w}$ et les $\frac{\delta u(t)}{\delta w}$ à partire de $t = k$ car à ce moment, le calcul est très simple d'après l'équation 9. Ensuite, nous calculons les dérivées pour $t > k$ sur base des dérivées précédentes soigneusement gardées en mémoire. Il est hors de question d'effectuer les calculs selon un algorithme récursif (càd 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.

3.3 Dérivées d'un système flou.

Les équations 16 et 19 font intervenir les dérivées partielles d'un système flou par rapport à un paramètre interne et par rapport à une de ses entrées(gradient). Cette section détaille le calcul de ces dérivées.

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

$$\begin{cases} \text{si } (U \text{ est dans } \mathcal{E}_j) \text{ alors } y_j = \sum_{i=1}^{n} U_i.L_{i,j} + L_{n+1,j}, \\ a_j = (\mu_{\mathcal{E}_j}(U)), \end{cases} \qquad j = 1, \ldots, r \tag{20}$$

$$y = \frac{\sum_{j=1}^{r} a_j y_j}{\sum_{j=1}^{r} a_j} \tag{21}$$

Si nous posons : $D_j = (U - C_j)^T F_j (U - C_j)$ ($D_j \in \Re$ est une distance de U à C_j), nous obtenons les fonctions d'appartenance suivantes :

- type "Gaussian" : $\mu_{\mathcal{E}_j}(U) = \frac{1}{e^{D_j}}$
- type "InverseDist" : $\mu_{\mathcal{E}_j}(U) = \frac{1}{D_j^{\frac{1}{m-1}}}$

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

$$\frac{\partial R}{\partial u_k}(U) = \frac{1}{\sum_{j=1}^{r} a_j} \sum_{j=1}^{r} \left[a_j.L_{j,k} + \left(2.\Gamma(D_j). \sum_{s=1}^{n} (F_j)_{k,s}.(U - C_j)_s \right) (y_j - R(U)) \right] \tag{22}$$

La figure 3 est une représentation graphique d'un système flou à une entrée u et une sortie y, composé de 2 règles floues. La fonction d'appartenance de la première règle est $\mu_1(x)$ et sa sortie locale est $y_1 = a_1 u + h_1$. Nous pouvons voir sur la partie gauche du shéma 3 le résultat obtenu en utilisant 22.

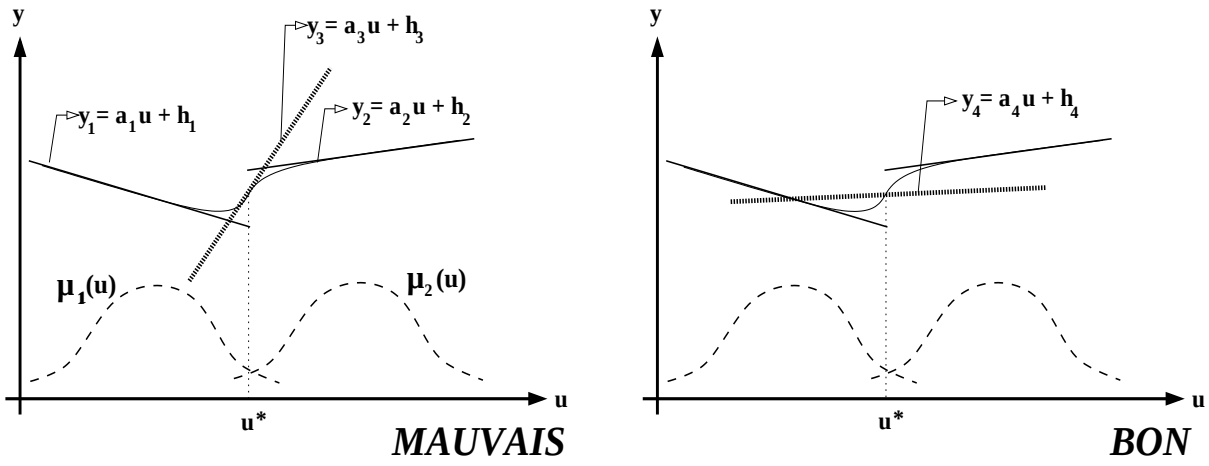

FIG. 3 – Deux définitions pour le calcul du gradient en u^* d'un système flou.

La méthode correcte pour le calcul du gradient (vue sur la partie droite de 3) consiste à calculer la combinaison floue des gradients locaux (de chaque règle). Ce qui donne :

$$\frac{\partial R}{\partial u_k}(U) = \frac{1}{\sum_{j=1}^{r} a_j} \sum_{j=1}^{r} [a_j . L_{j,k}] \tag{23}$$

La dérivée d'un système flou par rapport à un de ses paramètres ne pose pas de problème.
Si nous posons $\Gamma(D)$:

- type "Gaussian" : $\Gamma(D) = -1$
- type "InverseDist" : $\Gamma(D) = \frac{-1}{(m-1)D^{\frac{2-m}{m-1}}}$

Nous obtenons les dérivées suivantes :

$$\frac{\partial R}{\partial L_{k,l}}(U) = \frac{a_k}{\sum_{j=1}^{r} a_j} . \begin{cases} U_l & \text{si } (l \leq n) \\ 1 & \text{si } (l = n+1) \end{cases} \tag{24}$$

$$\frac{\partial R}{\partial (C_k)_l}(U) = \frac{-2.a_k}{\sum_{j=1}^{r} a_j} \Gamma(D_k) \left(\sum_{s=1}^{n} (F_k)_{l,s} . (U - C_k)_s \right) (y_k - R(U)) \tag{25}$$

$$\frac{\partial R}{\partial (F_k)_{l,m}}(U) = \frac{a_k}{\sum_{j=1}^{r} a_j} \Gamma(D_k) (U - C_k)_l . (U - C_k)_m (y_k - R(U)) \tag{26}$$

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 IRIDIA sur 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.

3.5 Autre algorithme d'apprentissage : "Levenberg-Marquardt".

Voir aussi [**?**].

Cette méthode est toujours basée sur une minimisation de fonction de coût J. Supposons que cette fonction J est aproximée par :

$$J(w) \simeq J(w^{(l)}) + (w - w^{(l)}) \cdot \nabla J(w^{(l)}) + \frac{1}{2}(w - w^{(l)}) \cdot D \cdot (w - w^{(l)}) \tag{27}$$

avec $w \in \Re^m$: un vecteur qui contient les m paramètres de J que nous voulons optimiser.
∇J : le gradient de J.
$D = \nabla^2 J$: la matrice Hessienne de J.

De 27, nous avons :

$$\nabla J(w) = \nabla J(w^{(l)}) + D \cdot (w - w^{(l)}) \tag{28}$$

Si $J(w^*)$ est un minimum alors $\nabla J(w^*) = 0$, ce qui combiné avec 27 et 28 donne :

$$\begin{aligned} w^* - w^{(l)} &= -D^{-1} \cdot \nabla J(w^{(l)}) \\ \Longleftrightarrow w^* &= w^{(l)} - D^{-1} \cdot \nabla J(w^{(l)}) \\ \\ w^{(l+1)} &= w^{(l)} - D^{-1} \cdot \nabla J(w^{(l)}) \end{aligned} \tag{29}$$

Si J est une parabole à m dimensions, l'approximation 27 se transforme en égalité stricte et l'équation 29 nous amène directement (en 1 itération) au minimum de la parabole. En pratique cela ne sera pas toujours le cas, nous devrons alors procéder, comme vu à la section 3.2 précédente, par descente de gradient :

$$w^{(l+1)} = w^{(l)} - \eta \cdot \nabla J(w^{(l)}) \tag{30}$$

Nous pouvons facilement calculer la matrice Hessienne D. La seule raison pour ne pas utiliser 29 est que son utilisation augmente J, signalant par cela que 27 n'est pas une bonne approximation de J.

3.5.1 Calcul de la matrice Hessienne D.

La fonction de coût réelle à minimiser est (voir 13) :

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

ce qui donne :

$$(\nabla J_k)_i = \frac{\delta J_k}{\delta w_i} = \sum_{t=k}^{k+H_p} (y(t) - \tilde{y}(t))^T Q_{k,t} \frac{\delta y(t)}{\delta w_i} + \sum_{t=k-1}^{k+H_p} (\Delta u(t))^T R_{k,t} \frac{\delta(\Delta u(t))}{\delta w_i} \tag{31}$$

$$\begin{aligned} D_{i,j} &= (\nabla^2 J_k)_{i,j} = \frac{\delta^2 J_k}{\delta w_i \delta w_j} \\ &= 2\sum_{t=k}^{k+H_p}(\frac{\delta y(t)}{\delta w_j} Q_{k,t} \frac{\delta y(t)}{\delta w_i} + (y(t)-\tilde{y}(t))^T Q_{k,t} \frac{\delta^2 y(t)}{\delta w_j \delta w_i}) + \\ & \quad 2\sum_{t=k-1}^{k+H_p}(\frac{\delta(\Delta u(t))}{\delta w_j} R_{k,t} \frac{\delta(\Delta u(t))}{\delta w_i} + (\Delta u(t))^T R_{k,t} \frac{\delta^2(\Delta u(t))}{\delta w_j \delta w_i}) \end{aligned} \tag{32}$$

Les dérivées secondes de 32 peuvent être ignorées. En effet, elles sont négligeables par rapport aux dérivées premières. De plus, le terme devant celles-ci est $(y(t) - \tilde{y}(t))$ qui est la mesure aléatoire de l'erreur. Cette erreur n'est en général pas corrélée avec le système et peut être positive ou négative. C'est pourquoi les dérivées secondes tendent à s'annuler quand elles sont sommées avec $(y(t) - \tilde{y}(t))$ en coefficient.

L'équation 32 devient :

$$\begin{aligned} D_{i,j} &= (\nabla^2 J_k)_{i,j} = \frac{\delta^2 J_k}{\delta w_i \delta w_j} \\ &= 2\sum_{t=k}^{k+H_p}(\frac{\delta y(t)}{\delta w_j} Q_{k,t} \frac{\delta y(t)}{\delta w_i}) + \\ & \quad 2\sum_{t=k-1}^{k+H_p}(\frac{\delta(\Delta u(t))}{\delta w_j} R_{k,t} \frac{\delta(\Delta u(t))}{\delta w_i}) \end{aligned} \tag{33}$$

Les dérivées dans cette équation sont calculées de la même manière que pour l'algorithme de descente de gradient.

3.5.2 Algorithme.

nous posons :

$$\beta_i = -\frac{\delta J_k}{\delta w_i} \quad \alpha_{i,j} = \frac{\delta^2 J_k}{\delta w_i \delta w_j} \tag{34}$$

L'équation 29 peut être réécrite :

$$\begin{aligned} w^{(l+1)} &= w^{(l)} - D^{-1} \cdot \nabla J(w^{(l)}) \\ \iff D \cdot (w^{(l+1)} - w^{(l)}) &= -\nabla J(w^{(l)}) \\ \iff \sum_{j=1}^{m} \alpha_{i,j} \delta w_j &= \beta_i,\ 1 \leq i \leq m \end{aligned} \tag{35}$$

avec $\delta w = w^{(l+1)} - w^{(l)}$

L'équation 30 (descente de gradient) peut être réécrite :

$$\begin{aligned} w^{(l+1)} &= w^{(l)} - \eta \cdot \nabla J(w^{(l)}) \\ \Longleftrightarrow \frac{1}{\eta}\,\delta w_i &= \beta_i,\ 1 \leq i \leq m \end{aligned} \tag{36}$$

L'équation 36 est équivalente à l'équation 35 avec

$$\begin{cases} \alpha_{i,j} = \frac{1}{\eta},\ i = j \\ \alpha_{i,j} = 0,\ i \neq j \end{cases}$$

En général η est très petit $\Rightarrow \frac{1}{\eta}$ est très grand. Pour passer graduellement de l'équation 35 à l'équation 36, il suffit de rendre la diagonale de $\alpha_{i,j}$ de plus en plus prépondérante.

Nous pouvons maintenant écrire l'algorithme de Levenberg-Marquart :

1. Prendre une valeur modeste pour ζ ($\zeta = .1$).
2. Calculer $J(w)$.
3. Résoudre le système d'équation :

$$\sum_{j=1}^{m} \tilde{\alpha}_{i,j}\,\delta w_j = \beta_i,\ 1 \leq i \leq m \tag{37}$$

   avec :

$$\tilde{\alpha}_{i,j} = \begin{cases} \alpha_{i,j} + \zeta & , i = j \\ \alpha_{i,j} & , i \neq j \end{cases} \tag{38}$$

4. si $J(w + \delta w) \geq J(w)$ alors augmenter ζ d'un facteur 10 et revenir au point 3. (mauvaise itération)
5. si $J(w+\delta w) < J(w)$ alors diminuer ζ d'un facteur 10, mettre à jour $w \leftarrow w+\delta w$ et revenir au point 3. (bonne itération)

FIG. 4 – algorithme de Levenberg-Marquardt.

3.5.3 Remarque.

Lorsque nous appliquons cet algorithme pour la recherche des paramètres du régulateur, nous limitons le nombre de mauvaises itérations ($= mi$) pour des raisons de vitesse. Nous limitons aussi le nombre de bonnes itérations ($= bi$) pour ne pas avoir un "overfiting" pour un point de fonctionnement qui ne serait pas représentatif du comportement idéal du régulateur.

Lorsque nous nous trouvons près du minimum de J, la matrice $\alpha_{i,j}$ sera presque entièrement composée de 0. La probabilité d'avoir une matrice singulière est grande. Si nous tombons sur une matrice singulière, nous multiplions ζ par 10 ($= zf$) et reprenons le calcul. Pour des raisons de vitesse nous limitons ζ à des

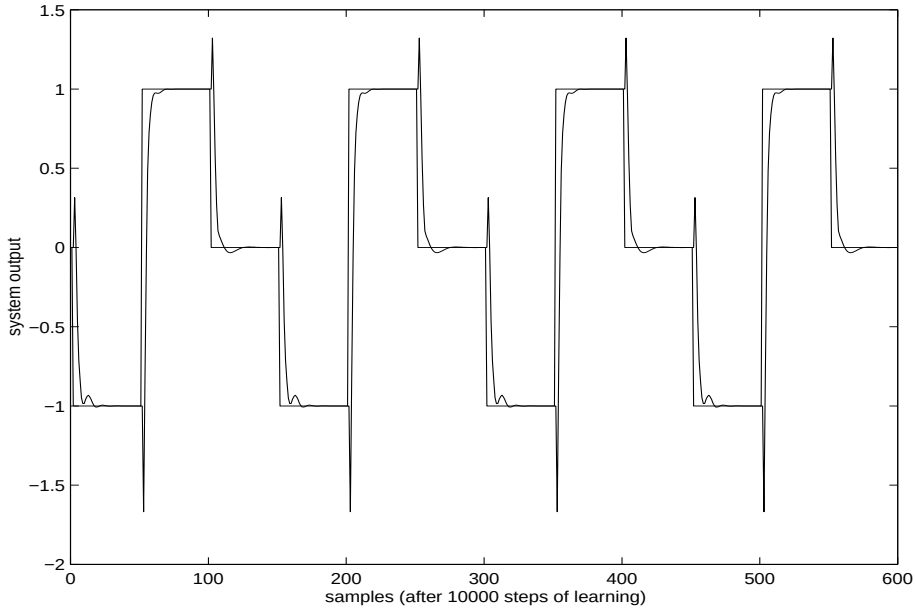

FIG. 5 – Performance du MSDAFC sur un système à non-minimum de phase après 10000 unités de temps

valeurs inférieures à 10^5 ($= zm$).

La vitesse d'apprentissage en terme d'unité de temps de simulation est grandement améliorée. En contrepartie, chaque pas de simulation prend beaucoup plus de temps de calcul que pour un algorithme basé sur une simple descente de gradient. De plus, cet algorithme exige de résoudre un système d'équation souvent fort proche de la singularité. Nous rencontrons alors des instabilités numériques. Celles-ci rendent le bon fonctionnement du régulateur difficile.

Grâce à l'algorithme de Levenberg-Marquardt, nous ne devons plus spécifier la vitesse d'apprentissage η, malheureusement d'autres paramètres (mi, bi, zf, zm) tout aussi difficile à ajuster apparaîssent.

4 Applications en simulation.

L'approche MSDAFC a été testée sur deux systèmes différents : un "toy problem" et un traîtement des eaux usées. Ce dernier est utilisé par tous les partenaires du projet FAMIMO (ESPRIT LTR Project 21911) [**?**].

4.1 Le "toy problem".

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

$$y_{k+1} = y_k + sin(3 * u_{k-1}) - 3u_k \tag{39}$$

Un régulateur basé sur une prédiction de une unité de temps dans le futur ne pourra régler 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.

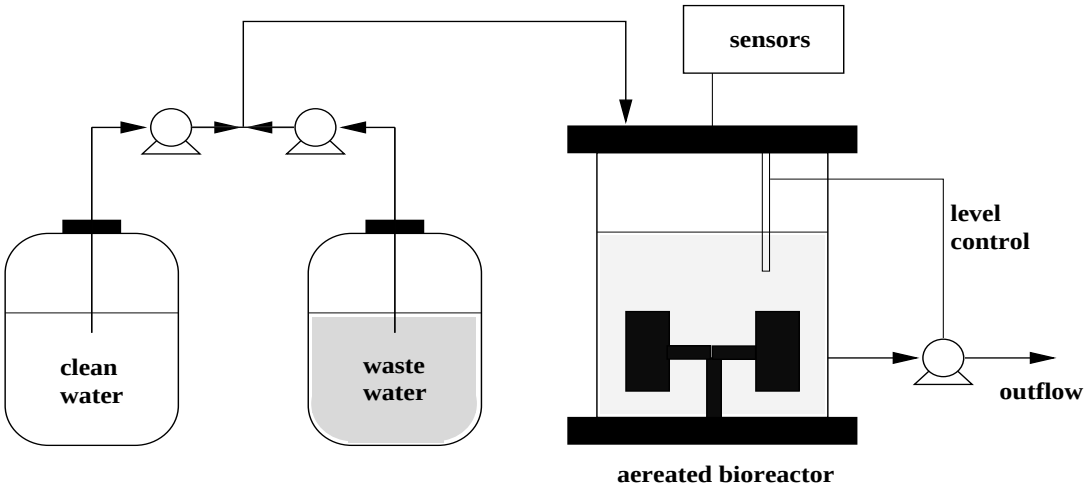

FIG. 6 – Representation schématique du bioréacteur pour le traîtement des eaux usées

Le processus est modélisé grâce à un système flou Takagi-Sugeno. Celui-ci a été identifié en utilisant l'algorithme de Gustafson-Kessel [**?**] pour calculer la position des centres et les conséquences de chaque règle floue. L'identification a été réalisée sur base de 5000 points obtenus en excitant le processus avec un signal pseudo aléatoire sinusoïdal. Le régulateur est, lui, constitué de 9 règles floues. Les conséquences ont été initialisées a 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 5.

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 régler en utilisant une prédiction de un pas, est facilement réglé en utilisant une prédiction sur plusieurs pas.

4.2 Le problème du traîtement des eaux usées.

Un système biologique constitué d'un bioréacteur continu à ciel ouvert utilisé pour le traîtement des eaux usées dans l'industrie du papier, est la cible de notre seconde étude expérimentale. Ce réacteur contient une certaine masse de bio-organismes et 2 substrats : le "xenobiotic polluant substrate" et "l'energetic substrate". Nous avons à notre disposition un modèle simulink du réacteur développé par le LAAS[**?**]. Nous devons régler la concentration résiduelle de "xenobiotic polluant substrate" *et* la prolifération des bio-organismes (biomasse) pour empêcher toutes nuisances bactériologiques. La concentration de "energetic substrate" ne doit pas être réglée car celui-ci est facilement dégradable. Le processus est caracterisé 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 de "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 régler la concentration en substrats en jouant sur les 2 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 6).

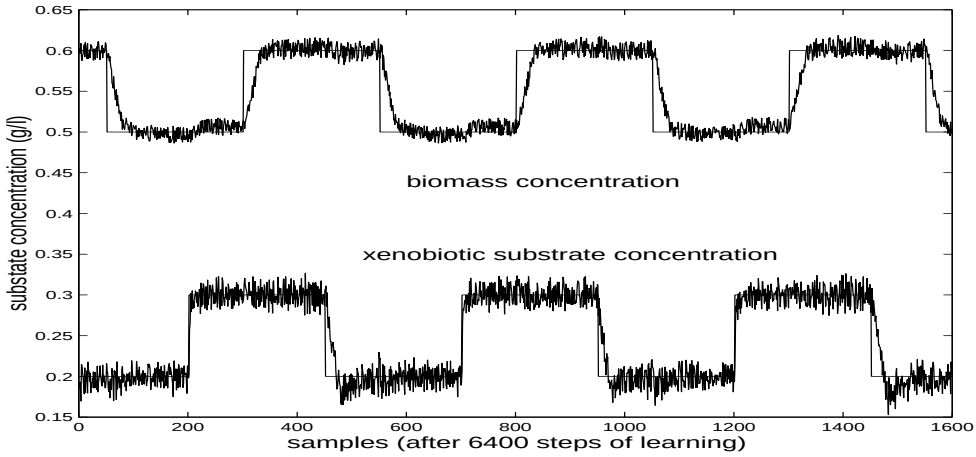

FIG. 7 – Performance du MSDAFC avec un horizon de prédiction de 3 sur le bioréacteur après 6400 unités de temps

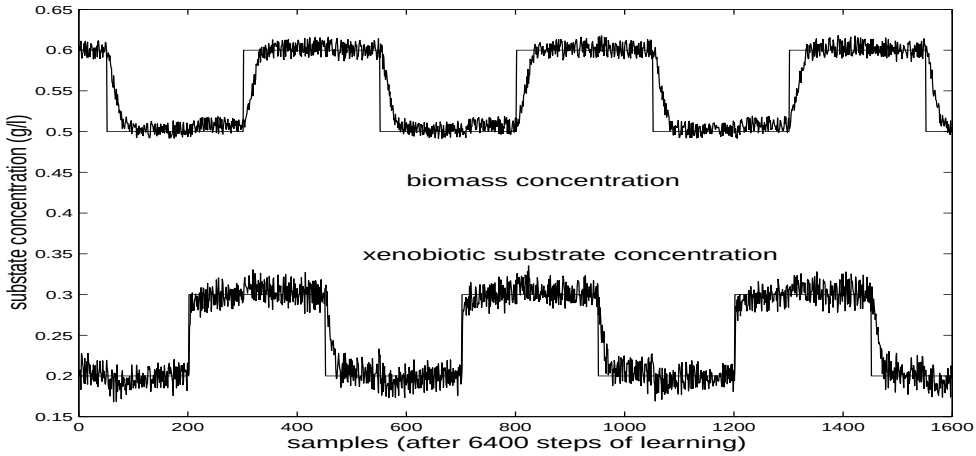

FIG. 8 – Performance du MSDAFC avec un horizon de prédiction de 10 sur le bioréacteur après 6400 unités de temps

soit c(t) [g/l] : (entrée ; à régler) concentration de la biomasse. Les micro-organismes dégradent les 2 substrats présents dans le réacteur.

s(t) [g/l] : (à régler) concentration du ”xenobiotic polluant substrate”.

e(t) [g/l] : (entrée) concentration du ”energetic substrate”

$u_1(t)$ [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 2 substrats.

$Y_{c/s}$ et $Y_{c/e}$: coefficients liés correspondant à la convertion 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 :

$$\begin{cases} \dot{c}(t) = \left\{ \mu_{sm} \frac{s(t)}{K_s + s(t) + a_e e(t)} + \mu_{em} \frac{e(t)}{K_e + e(t) + a_s s(t)} \right\} c(t) - u_1(t) c(t) - u_2(t) c(t). \\ \dot{s}(t) = -\frac{u_{sm}}{Y_{c/s}} \frac{s(t)}{K_s + s(t) + a_e e(t)} c(t) - u_1(t) s(t) + (s_a^{max} - s(t))\, u_2(t) \\ \dot{e}(t) = -\frac{u_{em}}{Y_{c/e}} \frac{e(t)}{K_e + e(t) + a_s s(t)} c(t) - u_1(t) e(t) + (e_a^{max} - e(t))\, u_2(t) \end{cases} \quad (40)$$

Le processus est modélisé grâce à un système flou Takagi-Sugeno. Celui-ci a été identifié en utilisant l'algorithme de Gustafson-Kessel [**?**] pour calculer la position des centres et les conséquences de chaque règle floue. L'identification a été réalisée sur base de 5000 points obtenus en excitant le processus avec un signal pseudo aléatoire sinusoïdal. Le prédicteur pour la biomasse est constitué de 4 règles floues. Le prédicteur pour la concentration en "xenobiotic substrate" est constitué de 2 règles floues. Le régulateur est, lui, constitué de 2 systèmes flous (un système pour chaque sortie) chacun constitué de 1 règle floue. Les conséquences des règles du régulateurs ont été initialisées à 0.

L'adaptation a été réalisée en utilisant 3 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 7 et 8, vous pouvez 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% and $+2\%$ 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.

Nous pouvons constater sur les figures 7 et 8 qu'après un temps d'apprentissage suffisament long (6400 unités de temps), les deux systèmes donnent de bons résultats et cela malgré des conditions forts 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'information hors du système. Malheureusement, elle exige plus de puissance de calcul (A 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, elle pourra prendre plus de temps de simulation que sa concurrente.

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) pour "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 meilleur que la "descente de gradient".

5 Conclusions

Les algorithmes adaptitifs 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.