الجمهورية الجزائرية الديمقراطية الشعبية

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

ÉTUDE DE L'APPLICATION D'UN CONTRÔLEUR FLOU OPTIMISÉ PAR ALGORITHMES GÉNÉTIQUES A LA COMMANDE D'UN BRAS MANIPULATEUR DE ROBOT FLEXIBLE

PROJET DE FIN D'ÉTUDES PRÉSENTÉ EN VUE

DE L'OBTENTION DU DIPLÔME

D'INGÉNIEUR D'ÉTAT EN ÉLECTRONIQUE

Proposé par :
M M LOUDINI

Étudié par :
ISMAIL Mohamed amine

Dirigé par :
M M. LOUDINI
M C. LARBES

Septembre 2005

E.N.P 10, Avenue Hassen Badi –EL-HARRACH – ALGER

الجمهورية الجزائرية الديمقراطية الشعبية

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

ÉTUDE DE L'APPLICATION D'UN CONTRÔLEUR FLOU OPTIMISÉ PAR ALGORITHMES GÉNÉTIQUES A LA COMMANDE D'UN BRAS MANIPULATEUR DE ROBOT FLEXIBLE

PROJET DE FIN D'ÉTUDES PRÉSENTÉ EN VUE

DE L'OBTENTION DU DIPLÔME

D'INGÉNIEUR D'ÉTAT EN ÉLECTRONIQUE

Proposé par :
M M LOUDINI

Étudié par :
ISMAIL Mohamed amine

Dirigé par :
M M. LOUDINI
M C. LARBES

Septembre 2005

E.N.P 10, Avenue Hassen Badi –EL-HARRACH – ALGER

Résumé

Beaucoup de commandes conventionnelles appliquées à un système non linéaire, complexe, instable, sont difficiles à concevoir et donnent lieu à de piètres performances. Pour de tels systèmes, les contrôleurs par logique floue offrent des solutions souvent meilleures, la conception de ces contrôleurs non linéaires, utilise la connaissance de l'expert (connaissance humaine) et l'imprécision implicite ; En l'absence d'une telle connaissance, la conception des contrôleurs sera lente,non guidée, basée sur l'essai & l'erreur. Les algorithmes génétiques (AGs) procurent une manière de surmonter cette imperfection. Ces algorithmes emploient quelques concepts de la théorie de l'évolution pour rechercher des solutions optimales, dans un espace de solutions grand et complexe en un temps beaucoup plus rapide.

Mots clés

Algorithmes génétiques, contrôleurs par logique floue, robot flexible.

ملخص

كثير من خطط التحكم الكلاسيكية للأنظمة الغير خطية و المضطربة صعبة التصميم و نتائجها سيئة. في هده الحالة التحكم بالمنطق الغامض يصبح جيدا.هذا المتحكم يستعمل معرفة المتخصص و الأخطاء المتكررة, في غياب هده المعرفة تكون عملية البحث عن مثل هدا المتحكم غير منهجية.

تعتبر الخوارزمية الوراثية كوسيلة فعالة لاجتياز هدا المشكل. تستعمل هده الخوارزمية مبادئ نظرية التطور للبحث على حلول مبسطة في فضاء الحلول يكون كبير و معقد في وقت قصير.

المفتاح

الخوارزمية الوراثية, المتحكم الغامض, الذراع المرن.

Abstract

Most of conventional control applied on non-linear complex and instable systems are difficult to design and give unsatisfactory performance. For such systems, fuzzy logic controllers gives good solutions, the non-linear controllers design use the human expert knowledge and implicit imprecision, the construction of these controllers can be quick and effective based on trial-and-error.

Genetic algorithms provide a way of surmounting this shortcoming these algorithms use some of the concepts of evolutionary theory to find optimal solutions in large and complex solution space in short time.

Key word

Genetic algorithms, fuzzy logic controllers, flexible robot.

TABLE DES MATIÈRES

LISTE DES FIGURES & DES TABLEAUX

GLOSSAIRE

A	: La section droite de la poutre.
$C(i)$	: Position des sommets du triangle
C_m	: Couple moteur.
EI	: Élasticité du bras.
E	: Module d'élasticité.
f_i	: La valeur de la fonction d'évaluation ou fitness
G	: Module de cisaillement
I	: Moment d'inertie du bras.
Jc	: Moment d'inertie de la charge.
Ja	: Moment d'inertie de l'ensemble articulation+moteur.
J	: Critère de performance
K	: Facteur dépendant de la forme de la section.
L	: Longueur du bras.
L_s	: La longueur de la ligne des racines.
Mc	: Masse de la charge.
n	: Nombre de fonction d'appartenance
Pm	: Paramètre d'espacement
R_0	: Repère absolu.
R_1	: Repère du bras.
T_L	: Énergie cinétique de la liaison.
T_A	: Énergie cinétique de l'articulation.
T_C	: Énergie cinétique de la charge.
u (t)	: Tension couple du moteur

x	: Distance le long du bras.
Y (x , t)	: Déflexion du point x.
$\alpha(x, t)$	: Angle de flexion
$\beta(x, t)$	: Angle de cisaillement
γ	: Amortissement de l'air visqueux.
$\phi(x)$	: Fonctions modales
ρ	: Densité linéique du bras.
θs	: La pente de la ligne des racines.
$\theta(t)$	: Position angulaire d'un point si le bras été rigide.
ζ	: Amortissement de Kelving-Voight.

Introduction générale

1. INTRODUCTION

Ce travaille comporte deux parties :

- Modélisation d'un bras manipulateur flexible.
- Contrôle du bras considéré.

1.1. PREMIER PARTIE

Les performances surhumaines des robots, tel que l'infatigabilité, l'insensibilité, la précision, la rapidité d'action et la répétitivité des tâches, leurs permettent de remplacer l'homme dans beaucoup de domaines. Les premières générations de robots étaient conçus avec des articulations et bras rigides, ces derniers avaient une faible capacité de chargement (de l'ordre de 5 à 10% leurs poids). Cependant, des bras conçus avec des matériaux légers, dits : bras flexibles, ont une capacité de chargement beaucoup plus élevée.

1.1.1. Bras flexibles:

La légèreté d'un bras flexible offre un certain nombre d'avantages, tel qu 'une consommation réduite de l'énergie, une réduction du coût des actionneurs (actionneur petit), une réduction des dégâts en cas de collusion ainsi qu'une augmentation du rapport charge/masse.

L'année 1972 fut marquée par le début des travaux sur les bras flexibles, les recherches se sont intensifiées par la suite ; surtout dans les domaines de la modélisation et de la commande.

La complexité de la modélisation est due essentiellement à l'aspect flexible du bras (modèle dynamique non linéaire), au problème de couplage dans le cas des robots à plusieurs segments et au choix du nombre de modes de vibration qui décrivent correctement le modèle.

La commande des structures flexibles représente un challenge formidable pour nombre d'ingénieurs, l'allégement des structures mécaniques et les performances

demandées amènent à considérer les effets de la flexibilité dans la synthèse des lois de commande,

Cette prise en compte des modes flexibles dans la commande, introduit des problèmes nouveaux très complexes.

Il est à noter qu'il faut garder à l'esprit la nécessité d'obtenir des modèles suffisamment simples et utilisables pour la commande en temps réel.

Étant donné que le bras se compose principalement d'une simple poutre en aluminium encastrée dans un bloc rigide lié à l'axe d'un moteur électrique qui est l'organe de commande. Sa forme favorisant la flexibilité dans le plan horizontal et les effets de torsion et de la pesanteur étant négligés, le problème de commande est alors le suivant : positionner l'extrémité du bras en agissant uniquement sur l'entrée de commande ; le procédé a donc pour entrée la tension couple du moteur u (t) et pour sortie la position de l'extrémité y (t).

1.2. DEUXIÈME PARTIE

Dans cette partie, des détails sur les travaux concernant la synthèse de la commande, à savoir l'optimisation d'un contrôleur à logique floue en employant des algorithmes génétiques. Des explications détaillées de ces deux concepts seront présentées sous forme d 'une démarche exposant la façon avec laquelle ils peuvent être appliqués à la synthèse de la commande de notre bras flexible.

1.2.1. Logique floue

La logique floue est un système de logique qui emploie l'imprécision. Un être humain ne dira jamais qu'une table est de 1.456 mètres de long, mais plutôt qu'elle est d'environ un mètre et demi de longueur. La logique floue classifie les objets selon des catégories décrites par variables linguistiques telles que : « long », « rapidement », « lourd », « entre deux âges » ... etc. Les objets peuvent avoir des degrés d'appartenance à un ensemble flou variant entre « certainement pas un membre » (dénoté par 0) à « certainement un membre » (dénoté par 1). Par exemple, les objets peuvent avoir un degré moins certain d'appartenance tel que « pas vraiment un membre » (probabilité de 0.1) et « plutôt un membre » (probabilité de 0.9).

La commande par logique floue se compose des règles de commande qui imitent celles employées par des humains quand ils commandent ou actionnent des processus ; l'home dit : « Si vous devez aller un peu plus rapidement, poussez légèrement la pédale d'accélérateur ».

La commande floue peut être une manière particulièrement efficace de commande pour les systèmes non linéaires, seulement quand la connaissance humaine est disponible.

Les détails de la façon dont la logique floue et la commande par logique floue sont appliquées sont présentés dans le chapitre 3.

1.2.2. Algorithmes génétiques

Les algorithmes génétiques (AGs) sont utilisés pour l'optimisation des solutions d'un problème; particulièrement ceux qui sont analytiquement infranchissables. Ils sont inspirés, comme leurs nom l'indique, des concepts biologiques de la génétique et de l'évolution.

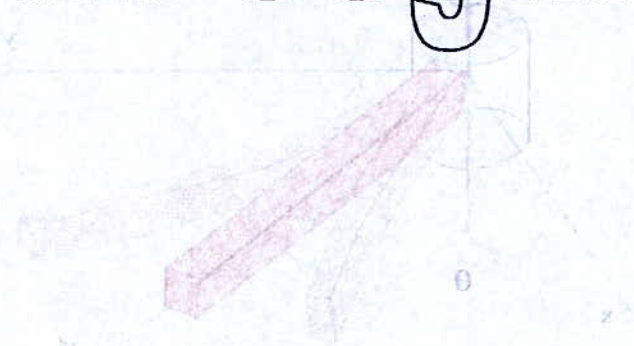
Le contrôleur le plus accompli que n'a jamais connu l'homme, le cerveau humain, ainsi que les nombreuses merveilles trouvées dans la nature, sont régis par le processus de l'évolution. Les principes derrière ce procédé fortement réussi devraient pouvoir fournir une certaine inspiration pour améliorer le processus d'étude. Les algorithmes génétiques emploient ces principes pour raffiner et optimiser les conceptions dont les paramètres agissent l'un sur l'autre d'une façon complexe. Les individus qui représentent les différentes solutions potentielles, sont préférentiellement choisis selon leur 'fitness' et donnent des caractéristiques, à des futures générations. Un couplage aura lieu entre ces individus avec l'espoir de partager les caractéristiques des individus réussis ; en conséquence des individus plus convenables peuvent être crées. La mutation se produit également pour injecter de nouveaux gènes dans une population. Comme dans la nature, la plupart des mutations sont nocives mais elles peuvent aider à améliorer le fitness des individus qu'elle affecte, c.-à-d. trouver de meilleures solutions.

1.2.3. *Vu d'ensemble sur les travaux déjà fait*

Dans le domaine de l'optimisation des contrôleurs par logique floue (CLF) en utilisant les algorithmes génétiques, un très grand nombre de recherche et travaux existent. Du coup, beaucoup d'approches et de méthodes ont été adoptées. Par exemple CHEN [17] applique les AG pour optimiser les règles d'inférence d'un CLF pour lequel les fonctions d'appartenance ont été déjà créées, tandis que Lee [6] a utilisé un AG qui détermine le nombre de fonctions d'appartenance et le nombre de règles floues. Dans les deux cas, ils utilisent le codage à valeurs réelles, tandis que Belarbi [7] emploie le codage binaire pour son approche (où ils met en application le CLF comme réseau neurologique et emploient l'AG pour former les poids), Park[24] utilise des paramètres caractéristiques pour automatiser la conception de CLF, cette méthode est utilisée pour ce projet également, en employant cette technique, un CLF peut-être conçu avec souplesse, où les nombres et les positions des fonctions d'appartenance et les règles d'inférence seront déterminées par un AG.

Dans ce mémoire les principes de base de la logique floue et des algorithmes génétiques sont expliqués. Ensuite, la stratégie de commande adoptée sera présentée avec les résultats obtenus. Mais avant tout une présentation et une analyse rentrant dans le cadre de la modélisation du système bras flexible, ainsi que le réglage classique de ce dernier seront illustrés.

Bras flexible à un segment



2. BRAS FLEXIBLE À UN SEGMENT

Dans ce chapitre on expose les détails de l'obtention d'une représentation d'état du système bras flexible, en passant par l'étude mécanique et l'étude énergétique, ensuite on montre l'étude du comportement du système en boucle ouverte et en boucle fermée utilisant un contrôleur classique (par retour d'état).

2.1. DESCRIPTION DU SYSTÈME

On considère que le mouvement est dû à au rotation θ et la déflexion, qui sont autour de l'axe OZ (Figure 2-1). On néglige la rotation autour de l'axe OY (Figure 2-2)qui est due à la gravité ainsi que la rotation autour de OX qui due à la torsion(Figure 2-3)

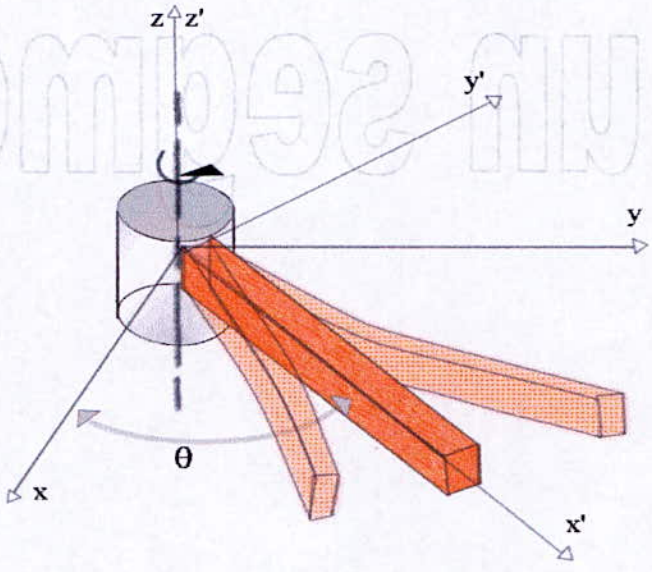


Figure 2-1 Déflection dans le plan XY

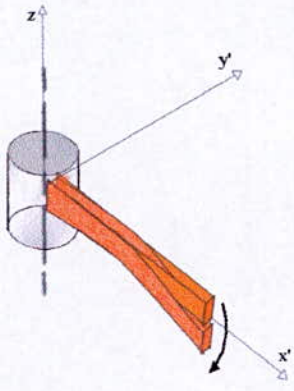


Figure 2-2 Déflection dans le plan XY

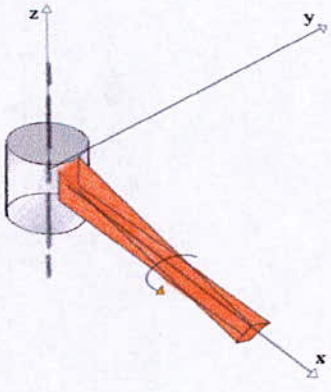


Figure 2-3 Torsion du bras autours de l'axe OX

Le bras flexible est représenté dans la Figure 2-4. Il pivote autour de l'axe OZ et est donc à un seul degré de liberté. Il est encastré à sa base par une articulation, et porte à son extrémité une charge.

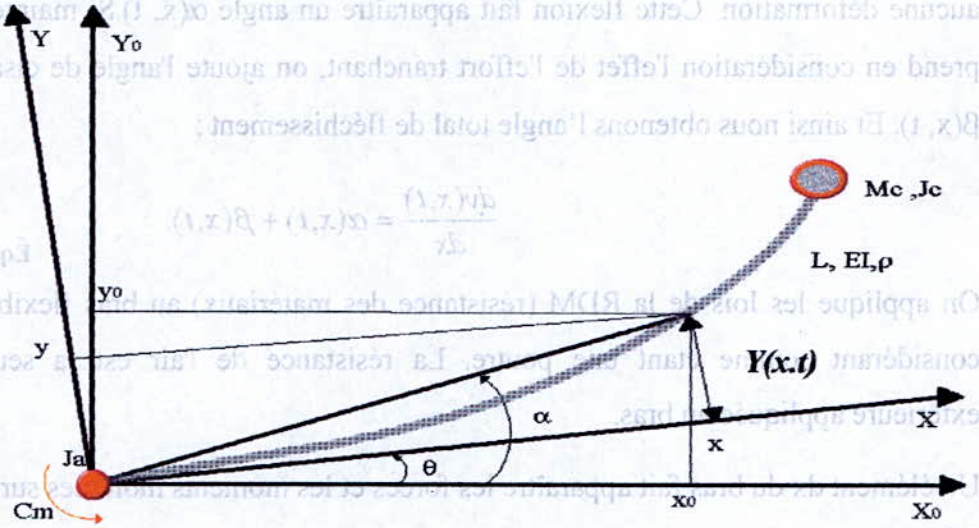


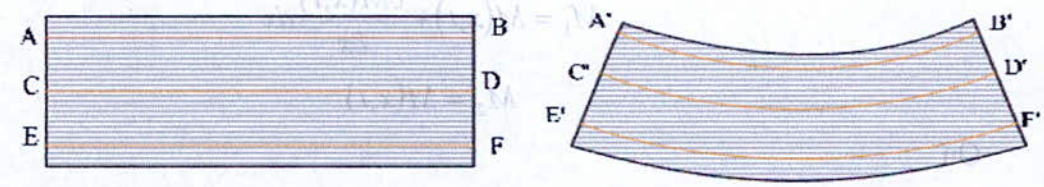
Figure 2-4 Bras flexible à un seul segment

Où

L	:	longueur du bras.
EI	:	élasticité du bras.
ρ	:	densité linéique du bras.
Ja	:	moment d'inertie de l'ensemble articulation+moteur.
Mc	:	masse de la charge.
Jc	:	moment d'inertie de la charge.
x	:	distance le long du bras.
θ(t)	:	position angulaire d'un point si le bras été rigide.
α(x,t)	:	Position angulaire d'un point du bras flexible.
y (x, t)	:	Déflection.
C_m	:	couple moteur.

2.2. ÉTUDE MÉCANIQUE

On se propose d'étudier un élément dx du bras, représenté par la figure 2-5.



AB=CD=EF
a) poutre avant déformation

A'B'<C'D'<E'F'
b) poutre après déformation

Figure 2-5 Élément dx à l'état d'équilibre (a) et en flexion (b)

AB>A'B' : la fibre AB subit une compression.
CD=C'D' : Fibre neutre ne subit aucune modification.
EF<E'F' : la fibre EF subit une traction.

Sous l'effet d'une flexion pure (absence de l'effort tranchant), la fibre neutre ne subit aucune déformation. Cette flexion fait apparaître un angle $\alpha(x, t)$.Si maintenant, on prend en considération l'effet de l'effort tranchant, on ajoute l'angle de cisaillement $\beta(x, t)$. Et ainsi nous obtenons l'angle total de fléchissement :

$$\frac{dy(x,t)}{dx} = \alpha(x,t) + \beta(x,t).$$

Équation 2-1

On applique les lois de la RDM (résistance des matériaux) au bras flexibles en le considérant comme étant une poutre. La résistance de l'air est la seule force extérieure appliquée au bras.

Un élément dx du bras fait apparaître les forces et les moments montrées sur la figure 2-6

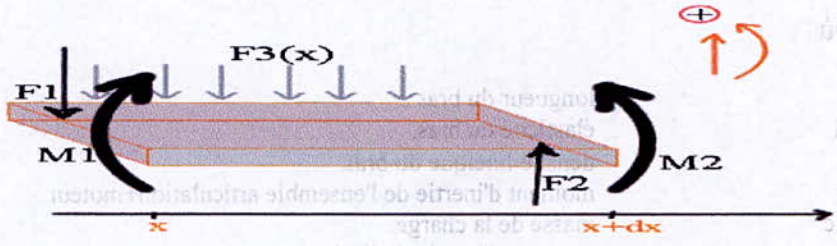


Figure 2-6 Efforts appliqués sur un élément dx

Avec :

$$F_1 = S(x,t)$$

$$F_2 = S(x,t) + \frac{\partial S(x)}{\partial x} dx$$

$$F_3 = \gamma \frac{dy(x,t)}{dt} . dx$$

$$M_1 = M(x,t) + \frac{\partial M(x,t)}{\partial x} dx$$

$$M_2 = M(x,t)$$

Où

F1 : L'effort tranchant du coté de la partie X.
F2 : L'effort tranchant du coté de la partie X+dX.
F3 : la résistance de l'air.
M1 : Le moment fléchissant sur la section X.
M2 : Le moment fléchissant sur la section X+dX.

- La somme des forces verticales à la section donne:

$$\left[S(x,t)+\frac{\partial S(x)}{\partial x}dx\right]-S(x,t)-\gamma\frac{\partial y(x,t)}{\partial t}dx=\rho\cdot A\cdot dx\cdot\frac{\partial^2 y(x,t)}{\partial t^2}$$

et donc

$$\frac{\partial S(x,t)}{\partial x}-\gamma\frac{\partial y(x,t)}{\partial t}=\rho\cdot A\cdot\frac{\partial^2 y(x,t)}{\partial t^2} \qquad \textbf{Équation 2-2}$$

avec

P : la densité du bras.

A : la section droite de la poutre.

γ : Amortissement de l'air visqueux.

- La somme des moments par rapport au point X+dX est donnée par :

$$\left[M(x,t)+\frac{\partial M(x,t)}{\partial x}dx\right]-M(x,t)-S(x,t)dx=\rho\cdot I\cdot dx\frac{\partial^2\alpha(x,t)}{\partial t^2}$$

et donc

$$\frac{\partial M(x,t)}{\partial x}-S(x,t)=\rho\cdot I\frac{\partial^2\alpha(x,t)}{\partial t^2} \qquad \textbf{Équation 2-3}$$

avec

I : moment d'inertie du bras.

- Le moment fléchissant et l'effort tranchant dépendent de la nature et de la forme du bras, ils sont donnés par les expressions suivantes :

$$M(x,t)=EI\frac{\partial\alpha(x,t)}{\partial x}+\zeta I\frac{\partial^2\alpha(x,t)}{\partial t\partial x} \qquad \textbf{Équation 2-4}$$

$$S(x,t)=kAG\beta(x,t)=kAG\left[\frac{\partial y(x,t)}{\partial x}-\alpha(x,t)\right] \qquad \textbf{Équation 2-5}$$

avec

E : module d'élasticité.

ζ : Amortissement de Kelving-Voight.

K : Facteur dépendant de la forme de la section.

G : Module de cisaillement.

Si on remplace le moment fléchissant et l'effort tranchant par leurs valeurs respectives dans les équations 2-2 et 2-3 on obtient

$$kAG(y_{xx}-\alpha_x)-\gamma\cdot y_t=\rho A y_{tt} \qquad \textbf{Équation 2-6}$$

$$EI\alpha_{xx} + \zeta I\alpha_{xxt} + kAG(y_x - \alpha) = \rho I\alpha_{tt} \qquad \textbf{Équation 2-7}$$

où

$$y_x = \frac{\partial y}{\partial x}, \text{ et } y_t = \frac{\partial y}{\partial t}$$

On tire α_x de 2-6 et on l'injecte dans 2-7 on obtient

$$a \cdot y_{xxxx} - b \cdot y_{xxtt} + c \cdot y_{tttt} + d \cdot y_{xxxxt} - e \cdot y_{xxt} - f \cdot y_{xxttt} + g \cdot y_t + h \cdot y_{ttt} + y_{tt} = 0 \qquad \textbf{Équation 2-8}$$

Où

$$a = \frac{EI}{\rho A}; \qquad b = \frac{1}{\rho A}\left[\rho I\left(1 + \frac{E}{kG}\right) + \frac{\zeta I\gamma}{kAG}\right];$$
$$c = \frac{\rho I}{kAG}; \qquad d = \frac{\zeta I}{\rho A}; \qquad e = \frac{EI}{\rho A^2 kG};$$
$$f = \frac{\zeta I}{kAG}; \qquad g = \frac{\gamma}{\rho A}; \qquad h = \frac{I\gamma}{kA^2 G}.$$

On peut vérifier facilement que si les termes d'amortissement sont nuls ($\gamma = 0, \zeta = 0$) l'équation 2-8 se ramène à l'équation classique de Timoshenko.

$$EI\frac{\partial^4 y(x,t)}{\partial x^4} + \rho A\frac{\partial^2 y(x,t)}{\partial t^2} - I\rho\left(1 + \frac{E}{k^2 G}\right)\frac{\partial^4 y(x,t)}{\partial x^2 \partial t^2} + \frac{I\rho}{k^2 G}\frac{\partial^4 y(x,t)}{\partial t^4} = 0 \qquad \textbf{Équation 2-9}$$

Et si de plus les effets du moment d'inertie de rotation sont négligés, on obtient l'équation d'Euler-Bernouilli :

$$\frac{\partial^4 y(x,t)}{\partial x^4} + \frac{\rho A}{EI}\frac{\partial^2 y(x,t)}{\partial t^2} = 0 \qquad \textbf{Équation 2-10}$$

Pour résoudre l'équation 2-8 on va utiliser la méthode standard de séparation des variables. Qui consiste à supposer que :

$$y(x,t) = \sum_{i=0}^{\infty} y_i(x,t) = \sum_{i=0}^{\infty} \phi_i(x) \cdot \delta_i(t) \qquad \textbf{Équation 2-11}$$

La détermination des fonctions $\phi_i(x) \cdot \delta_i(t)$ est faite à l'aide des conditions aux limites ainsi que quelque conditions d'orthogonalité.

Au vu de la complexité de fonction 2-8, la résolution par le principe de séparation est impossible. Pour cela nous nous considérons l'équation d'Euler-Bernouilli, 2-10, qui devient en utilisant 2-11 :

$$\frac{\phi''''(x)}{\phi(x)} + \frac{\rho A}{EI}\frac{\delta''(t)}{\delta(t)} = 0 \qquad \textbf{Équation 2-12}$$

Or, si deux fonctions qui dépendent de deux variables indépendantes sont égales, elles ne peuvent être que des constantes.

$$\frac{\phi''''(x)}{\phi(x)} = s = -\frac{\rho A}{EI}\frac{\delta''(t)}{\delta(t)}$$

Cela nous mène à diviser l'équation 2-12 en deux équations :

$$\delta''(t) + \frac{sEI}{\rho A}\delta(t) = 0 \quad \text{et donc } \omega^2 = \frac{sEI}{\rho A} \qquad \textbf{Équation 2-13}$$

$$\phi''''(x) - s\phi(x) = 0 \qquad \textbf{Équation 2-14}$$

Et si on pose $s = \lambda^4$

L'équation caractéristique sera

$$\lambda^4 = \frac{\omega^2 \rho A}{EI} \quad \text{et les racines seront}$$

$$\lambda_1 = \lambda = \sqrt[4]{\frac{\omega^2 \rho A}{EI}} \qquad \lambda_2 = -\lambda = -\sqrt[4]{\frac{\omega^2 \rho A}{EI}}$$

$$\lambda_3 = i\lambda = i\sqrt[4]{\frac{\omega^2 \rho A}{EI}} \qquad \lambda_4 = -i\lambda = -i\sqrt[4]{\frac{\omega^2 \rho A}{EI}}$$

La solution générale de l'équation 2-14 peut s'écrire

$$\phi(x) = Ae^{\lambda x} + Be^{-\lambda x} + Ce^{i\lambda x} + De^{-i\lambda x} \qquad \textbf{Équation 2-15}$$

Ce qui revient à écrire

$$\phi(x) = A_1 \sin(\lambda x) + A_2 \cos(\lambda x) + A_3 \sinh(\lambda x) + A_4 \cosh(\lambda x) \qquad \textbf{Équation 2-16}$$

Les conditions aux limites au point O (origine du bras) sont :

$$\begin{cases} \phi(0) = 0 \\ \phi'(0) = 0 \end{cases} \Rightarrow \begin{cases} A_2 + A_4 = 0 \\ \lambda(A_1 + A_3) = 0 \end{cases} \Rightarrow \begin{cases} A_2 = -A_4 \\ A_1 = -A_3 \end{cases}$$

Donc $\phi(x)$ devient

$$\phi(x) = A_1[\sin(\lambda x) - \sinh(\lambda x)] + A_2[\cos(\lambda x) - \cosh(\lambda x)] \qquad \textbf{Équation 2-17}$$

Les conditions aux limites au point x=1(normalisation par rapport à la longueur L) sont :

$$\begin{cases} EI\left.\dfrac{\partial^2 y(x,t)}{\partial x^2}\right|_{x=1} = -Jc\left.\dfrac{\partial^2}{\partial t^2}\left(\dfrac{\partial y(x,t)}{\partial x}\right)\right|_{x=1} \\ EI\left.\dfrac{\partial^3 y(x,t)}{\partial x^3}\right|_{x=1} = Mc\left.\dfrac{\partial^2 y(x,t)}{\partial t^2}\right|_{x=1} \end{cases}$$

$$\begin{cases} EI\phi''(x)\big|_{x=1}\delta(t) = -Jc\phi'(x)\big|_{x=1}\delta''(t) & \textbf{Équation 2-18} \\ EI\phi'''(x)\big|_{x=1}\delta(t) = Mc\phi(x)\big|_{x=1}\delta''(t) & \textbf{Équation 2-19} \end{cases}$$

$$\begin{cases} \phi''(x)\big|_{x=1} - \dfrac{Jc\,\omega^2}{EI}\phi'(x)\big|_{x=1} = 0 & \textbf{Équation 2-20} \\ \phi'''(x)\big|_{x=1} + \dfrac{Mc\omega^2}{EI}\phi(x)\big|_{x=1} = 0 & \textbf{Équation 2-21} \end{cases}$$

Comme

$$\begin{aligned} \phi(x) &= A_1(\sin\lambda x - \sinh\lambda x) + A_2(\cos\lambda x - \cosh\lambda x) \\ \phi'(x) &= \lambda\ [A_1(\cos\lambda x - \cosh\lambda x) - A_2(\sin\lambda x + \sinh\lambda x)] \\ \phi''(x) &= \lambda^2\ [-A_1(\sin\lambda x + \sinh\lambda x) - A_2(\cos\lambda x + \cosh\lambda x)] \\ \phi'''(x) &= \lambda^3\ [-A_1(\cos\lambda x + \cosh\lambda x) + A_2(\sin\lambda x - \sinh\lambda x)] \\ \phi''''(x) &= \lambda^4\,\phi(x) \end{aligned}$$

Les équations 2-20 et 2-21 deviennent

$$\begin{cases} A_1\left[(\sin\lambda+\sinh\lambda)+\dfrac{Jc\omega^2}{EI\lambda}(\cos\lambda-\cosh\lambda)\right]+A_2\left[(\cos\lambda+\cosh\lambda)-\dfrac{Jc\omega^2}{EI\lambda}(\sin\lambda+\sinh\lambda)\right]=0 \\ A_1\left[-(\cos\lambda+\cosh\lambda)+\dfrac{Mc\omega^2}{EI\lambda^3}(\sin\lambda-\sinh\lambda)\right]+A_2\left[(\sin\lambda-\sinh\lambda)+\dfrac{Mc\omega^2}{EI\lambda^3}(\cos\lambda-\cosh\lambda)\right]=0 \end{cases}$$

Pour trouver les A_i, le déterminant du système doit être nul, pour cela on trouve une infinité de valeurs de λ, qui correspondent aux fréquences modales. Les solutions sont données dans l'annexe A.1.

Si on dérive l'équation 2-18 et on, la remplace dans l'équation 2-19 on trouve

$$\frac{-Jc\phi''(1)}{Mc\phi(1)} = 1 \Rightarrow \phi''(1) + \frac{Mc}{Jc}\phi(1) = 0 \qquad \textbf{Équation 2-22}$$

Et après quelques développements on aboutie à

$$\frac{A_2}{A_1} = -\frac{(\sin\lambda + \sinh\lambda) - \dfrac{Mc}{\lambda^2 Jc}(\sin\lambda - \sinh\lambda)}{(\cos\lambda + \cosh\lambda) - \dfrac{Mc}{\lambda^2 Jc}(\cos\lambda - \cosh\lambda)} \qquad \textbf{Équation 2-23}$$

est donc $\phi(x)$ aura la forme :

$$\phi(x) = A_1\left[\sin(\lambda x) - \sinh(\lambda x) - \frac{(\sin\lambda + \sinh\lambda) - \frac{Mc}{\lambda^2 Jc}(\sin\lambda - \sinh\lambda)}{(\cos\lambda + \cosh\lambda) - \frac{Mc}{\lambda^2 Jc}(\cos\lambda - \cosh\lambda)}(\cos(\lambda x) - \cosh(\lambda x))\right]$$

Équation 2-24

La valeur de A_1 sera définie plus tard, en utilisant d'autres conditions expliquées dans l'annexeA.2

2.3. ÉTUDE MÉCANIQUE

Afin de trouver le modèle du système et puisque le mouvement est dû a l'addition de deux mouvements, la rotation et la déformation de la liaison, la méthode de Lagrange est la plus appropriée et la plus facile pour trouver l'équation différentielle du système, et pour cela il faut calculer les différentes énergies, cinétiques et potentielles.

Soit la représentation suivante du bras:

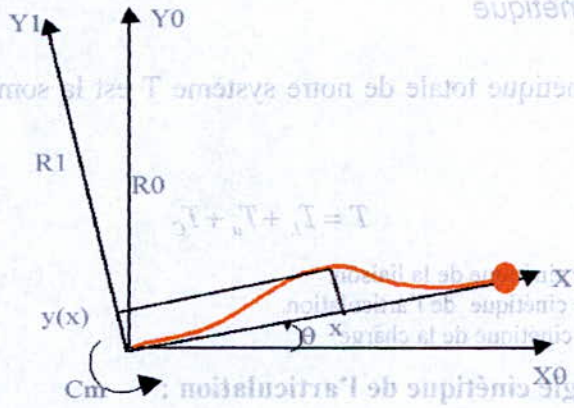


Figure 2-7, repérage cinématique du bras flexible

Sachant que :

R_0	: repère absolu.
R_1	: repère du bras.
θ (t)	: Rotation rigide.
Y (x , t)	: Déflexion du point x.

Soit A la matrice de passage de repère R_1 ver R_0.

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Cela veut dire que si ${}^1P = \begin{bmatrix} x \\ y(x) \end{bmatrix}$ les composantes du point P dans le repaire R_1, la composante dans le repaire R_0 sera :

$${}^0P = A.{}^1P$$ **Équation 2-25**

La vitesse de ce point par rapport au repaire absolu R_0 est :

$${}^0\dot{P} = \dot{A}.{}^1P + A.{}^1\dot{P}$$ **Équation 2-26**

Tel que

$$\dot{A} = \begin{bmatrix} -\dot{\theta}\sin\theta & -\dot{\theta}\cos\theta \\ \dot{\theta}\cos\theta & -\dot{\theta}\sin\theta \end{bmatrix} = \dot{\theta}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\dot{A} = \dot{\theta}\,S.A$$

où

$$S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2.3.1. *L'énergie cinétique*

L'énergie cinétique totale de notre système T est la somme des énergies cinétiques suivantes :

$$T = T_L + T_a + T_C$$ **Équation 2-27**

T_L : énergie cinétique de la liaison.
T_A : énergie cinétique de l'articulation.
T_C : énergie cinétique de la charge.

- **Energie cinétique de l'articulation :**

L'articulation ne subissant qu'une rotation, son énergie cinétique est donnée par :

$$T_a = \frac{1}{2} J_a \dot{\theta}^2$$ **Équation 2-28**

J_a moment cinétique de l'articulation

- **Energie cinétique de la liaison :**

$$T_L = \frac{1}{2}\int_0^1 \rho A\, {}^0\dot{P}^T\, {}^0\dot{P}\, dx$$ **Équation 2-29**

Si on développe en utilisant équation 2-26

$$\overset{0}{\dot{P}} = \dot{A}.\,{}^{1}P + A.\,\overset{1}{\dot{P}} \qquad \overset{0}{\dot{P}}^{T} = \dot{A}^{T}.\,{}^{1}P^{T} + A.\,\overset{1}{\dot{P}}^{T}$$

En utilisant les résultats suivants

$$A^{T} \cdot A = I$$

$$A^{T} \cdot \dot{A} = \dot{\theta} S$$

$$\dot{A}^{T} \cdot A = -\dot{\theta} S$$

Ou fait que le bras est inextensible, on a $\dot{X} = 0$, l'équation 2-29 devient :

$$\overset{0}{\dot{P}}^{T}\,\overset{0}{\dot{P}} = \left(x^2 + y^2\right)\dot{\theta}^2 + 2xy\dot{\theta} + \dot{y}^2$$

$$T_L = \frac{1}{2}\int_0^1 \rho A\left(x^2 + y^2\right)\dot{\theta}^2\,dx + \int_0^1 \rho A x y \dot{\theta}\,dx + \frac{1}{2}\int_0^1 \rho A \dot{y}^2\,dx$$

$$T_L = \frac{1}{2}\rho A\int_0^1 x^2\dot{\theta}^2\,dx + \int_0^1 \rho A x \sum_{i=0}^{\infty}\phi_i(x)\cdot\dot{\delta}_i(t)\dot{\theta}\,dx + \frac{1}{2}\int_0^1 \rho A\left(\sum_{i=0}^{\infty}\phi_i^2(x)\cdot\dot{\delta}_i^2(t)\right)dx$$

$$T_L = \frac{1}{2}\rho A\dot{\theta}^2\int_0^1 x^2dx + \rho A\dot{\theta}\sum_{i=0}^{\infty}\left(\dot{\delta}_i(t)\int_0^1 x\phi_i(x)dx\right) + \frac{1}{2}\rho A\sum_{i=0}^{\infty}\left(\dot{\delta}_i^2(t)\int_0^1 \phi_i^2(x)dx\right)$$

Si on pose

$$J_b = \int_0^1 x^2dx$$

$$Z_i = \rho A\int_0^1 \phi_i^2(x)dx$$

$$W_i = \rho A\int_0^1 x\phi_i(x)dx$$

T_L s'écrira comme suit

$$T_L = \frac{1}{2}J_b\dot{\theta}^2 + \dot{\theta}\sum_{i=0}^{\infty}\dot{\delta}_i(t)W_i + \frac{1}{2}\sum_{i=0}^{\infty}\dot{\delta}_i^2(t)Z_i \qquad \textbf{Équation 2-30}$$

- **Énergie cinétique de la charge :**

$$T_C = \frac{1}{2} Mc \overset{0}{\dot{P}}_c{}^T \overset{0}{\dot{P}}_c + \frac{1}{2} Jc \left(\dot{\theta} + \frac{\partial \dot{y}}{\partial x}\bigg|_{x=1} \right)^2$$

$$\overset{0}{\dot{P}}_c{}^T \overset{0}{\dot{P}}_c = \overset{0}{\dot{P}}{}^T \overset{0}{\dot{P}}\bigg|_{x=1} = \left(1 + y^2\right)\dot{\theta}^2 + 2\dot{y}\dot{\theta} + \dot{y}^2$$

$$T_C = \frac{1}{2} Mc \left[\left(1 + y^2\right)\dot{\theta}^2 + 2\dot{y}\dot{\theta} + \dot{y}^2 \right] + \frac{1}{2} Jc \left(\dot{\theta} + \frac{\partial \dot{y}}{\partial x} \right)^2 \bigg|_{x=1}$$

$$T_C = \frac{1}{2} Mc\dot{\theta}^2 + Mc\dot{\theta} \sum_{i=0}^{\infty} \phi_i(1) \cdot \dot{\delta}_i(t) + \frac{1}{2} Mc \sum_{i=0}^{\infty} \phi_i^2(1) \cdot \dot{\delta}_i^2(t) + \frac{1}{2} Jc\dot{\theta}^2 + \frac{1}{2} Jc \sum_{i=0}^{\infty} \phi_i'^2(1) \cdot \dot{\delta}_i^2(t) + Jc\dot{\theta} \sum_{i=0}^{\infty} \phi_i'(1) \cdot \dot{\delta}_i(t)$$

Équation 2-31

2.3.2. *L'énergie Potentielle :*

Le bras ne pouvant se mouvoir que dans le plan horizontal, les forces de gravité servant au fléchissement du bras dans le sens vertical sont ignorées. L'énergie potentielle se traduit donc par l'énergie élastique emmagasinée dans le bras. Elle est due au cisaillement et à la flexion, si on néglige le cisaillement et l'amortissement, l'expression de l'énergie potentielle sera :

$$U = \frac{1}{2} EI \int_0^1 \left(\frac{d^2 y(x,t)}{dx^2} \right)^2 dx$$

d'où

$$U = \frac{1}{2} EI \sum_{i=1}^{\infty} \left(\delta_i^2(t) \int_0^1 \phi_i''^2(x) dx \right)$$

an posant

$$K_i = EI \int_0^1 \phi_i''^2(x) dx$$

On obtient

$$U = \frac{1}{2} \sum_{i=1}^{\infty} \left(\delta_i^2(t) K_i \right)$$

Équation 2-32

2.3.3. *Le nombre de modes flexibles*

D'après l'annexe A.1 le nombre de fréquences modales est infini. Seulement, d'après les travaux de [25], l'étude des bras flexibles peut ce limiter au deux premiers modes flexibles. Ceci sera adopté dans tout ce qui suivre. On a alors.

$$T_{totale} = \frac{1}{2}[Mc + Jc + J_a + J_b]\dot{\theta}^2 + [Mc\phi_1(1) + Jc\phi_1'(1) + W_1]\cdot\dot{\delta}_1(t)\dot{\theta}$$
$$+ [Mc\phi_2(1) + Jc\phi_2'(1) + W_2]\cdot\dot{\delta}_2(t)\dot{\theta} + \frac{1}{2}[Mc\phi_1^2(1) + Jc\phi_1'^2(1) + Z_1]\cdot\dot{\delta}_1^2(t)$$
$$+ \frac{1}{2}[Mc\phi_2^2(1) + Jc\phi_2'^2(1) + Z_2]\cdot\dot{\delta}_2^2(t)$$

$$U = \frac{1}{2}\delta_1^2(t)K_1 + \frac{1}{2}\delta_2^2(t)K_2$$

2.3.4. *L'équation de Lagrange :*

Pour établir l'équation dynamique de notre système, on utilise l'équation de Lagrange :

$$L = T - U \qquad \textbf{Équation 2-33}$$

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}_i}\right) - \frac{dL}{dq_i} = \frac{dW}{dq_i} - \frac{dP}{d\dot{q}_i} \qquad \textbf{Équation 2-34}$$

Où

$q = [\theta(t), \delta(t), \delta(t)]$: Vecteur des coordonnées généralisées.

W est le travail dû aux forces appliquées, l'expression de dW est

$$dW = F(x,t)dy(x,t) + M(x,t)d\left[\frac{\partial y(x,t)}{\partial t}\right] \qquad \textbf{Équation 2-35}$$

Ou F(x,t) et M(x,t) représentent respectivement la force et le moment concentrés au point x.

Le bras n'étant soumis qu'au couple moteur à l'encastrement, F et M s'écrivent :

F (x, t) = 0 pour $0 \leq x \leq L$.
M (x, t) = 0 pour $0 < x \leq L$.
M (0, t) = Cm.

D'où, l'équation 2-35 devient

$$dW = C_m \sum_{i=1}^{2} \frac{d\phi_i(0)}{dx} . dq_i \Rightarrow \frac{dW}{dq_i} = C_m \sum_{i=1}^{2} \frac{d\phi_i(0)}{dx} \qquad \textbf{Équation 2-36}$$

L'amortissement est mis en évidence par l'expression de la fonction de dissipation de Rayleigh [1] :

$$P = \frac{1}{2} \sum_i \sum_j c_{ij} . \dot{q}_i(t) . \dot{q}_j(t) \qquad \textbf{Équation 2-37}$$

L'expression de la fonction de dissipation se simplifie par application de la propriété d'orthogonalité des fonctions pour donnée la forme suivante :

$$P = \frac{1}{2} \sum_i c_i . \dot{q}_i^{\,2}(t) \qquad \textbf{Équation 2-38}$$

c_i étant le facteur d'amortissement, défini dans l'annexe A.2.

Les calculs élaborés dans l'annexe A.2 permettent de trouver le modèle dynamique linéaire suivant :

$$M\ddot{q} + D\dot{q} + Kq = Q \qquad \textbf{Équation 2-39}$$

avec

$$M = \begin{pmatrix} J_T & V_1 & V \\ V & 1 & 0 \\ V & 0 & 1 \end{pmatrix} K = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \omega_1^2 & 0 \\ 0 & 0 & \omega_2^2 \end{pmatrix} D = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2\zeta_1\omega_1 & 0 \\ 0 & 0 & 2\zeta_2\omega_2 \end{pmatrix} Q = \begin{pmatrix} C_m \\ 0 \\ 0 \end{pmatrix}$$

et

$$J_T = Mc + Jc + J_a + J_b$$
$$V_1 = Mc\phi_1(1) + Jc\phi_1'(1) + W_1$$
$$V_2 = Mc\phi_2(1) + Jc\phi_2'(1) + W_2$$

Le passage à la représentation dans l'espace d'état donne lieu au système suivant :

$$\begin{cases} \dot{X}(t) = A\,X(t) + B\,U(t) \\ Y(t) = C\,X(t) + D\,U(t) \end{cases}$$

Où

$$X(t)=\begin{bmatrix}\theta(t)\\ \delta_1(t)\\ \delta_2(t)\\ \dot{\theta}(t)\\ \dot{\delta}_1(t)\\ \dot{\delta}_2(t)\end{bmatrix}$$

$$A=\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & \frac{V_1\omega_1^2}{\det} & \frac{V_2\omega_2^2}{\det} & 0 & \frac{2\zeta_1\omega_1V_1}{\det} & \frac{2\zeta_2\omega_2V_2}{\det}\\ 0 & \frac{\left(V_2^2-J_T\right)\omega_1^2}{\det} & -\frac{V_1V_2\omega_2^2}{\det} & 0 & \frac{2\zeta_1\omega_1\left(V_2^2-J_T\right)}{\det} & -\frac{2\zeta_2\omega_2V_1V_2}{\det}\\ 0 & -\frac{V_1V_2\omega_1^2}{\det} & \frac{\left(V_1^2-J_T\right)\omega_2^2}{\det} & 0 & -\frac{2\zeta_1\omega_1V_1V_2}{\det} & \frac{2\zeta_2\omega_2\left(V_1^2-J_T\right)}{\det}\end{bmatrix}$$

$$B=\begin{bmatrix}0\\ 0\\ 0\\ \frac{1}{\det}\\ -\frac{V_1}{\det}\\ -\frac{V_2}{\det}\end{bmatrix}$$

$$C=\begin{bmatrix}x & \phi_1(x) & \phi_2(x) & 0 & 0 & 0\end{bmatrix} \qquad D=[0]$$

avec

$$\det = J_T + V_1^2 + V_2^2$$

L'application numérique donne lieu à la fonction de transfert propre au système suivante (voir valeurs des paramètres physiques, l'annexe A.3) :

$$M(p)=\frac{5.7\cdot10^{-15}\mathrm{p}^5+0.46\mathrm{p}^4+0.713\mathrm{p}^3+974.4\mathrm{p}^2+858.4\mathrm{p}+253.5\cdot10^5}{\mathrm{p}^6+1.364\mathrm{p}^5+1893\mathrm{p}^4+1491\mathrm{p}^3+4.4\cdot10^5\mathrm{p}^2+253.510^5} \qquad \textbf{Équation 2-40}$$

2.4. LE MODÈLE SIMULINK

En utilisant SIMULINK, un modèle du système, (figure 2-8) a été créé. Dans ce modèle, l'action de commande est une entrée et le vecteur x est la sortie. Cette représentation permet de connaître les variations internes des variables du système.

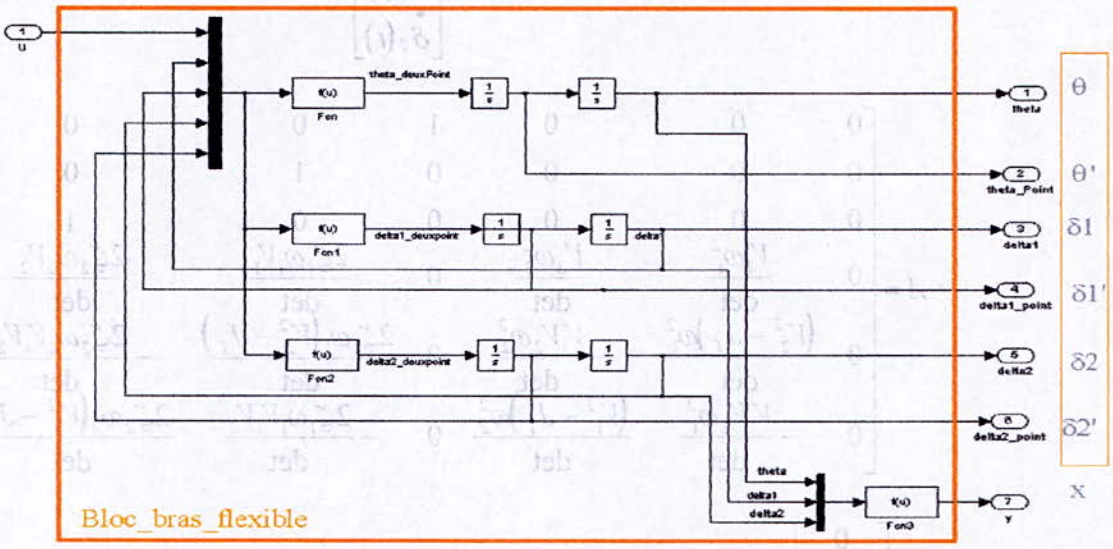


Figure 2-8 Bloc du bras flexible

2.5. RÉPONSE EN BOUCLE OUVERTE

Les différentes réponses en boucle ouverte (impulsionnelles et indicielles) sont obtenues en utilisant le modèle SIMULINK suivant:

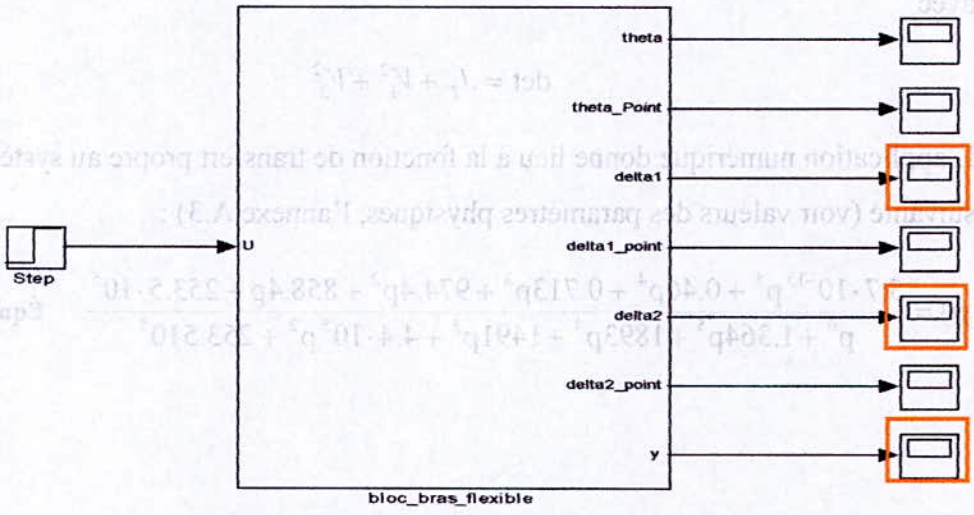


Figure 2-9 Modèle SIMULINK pour la réponse en boucle ouverte

Les simulations se sont déroulées pendant 12 secondes en utilisant les paramètres physiques de l'annexe A.3.

2.5.1. *Réponse impulsionnelle*

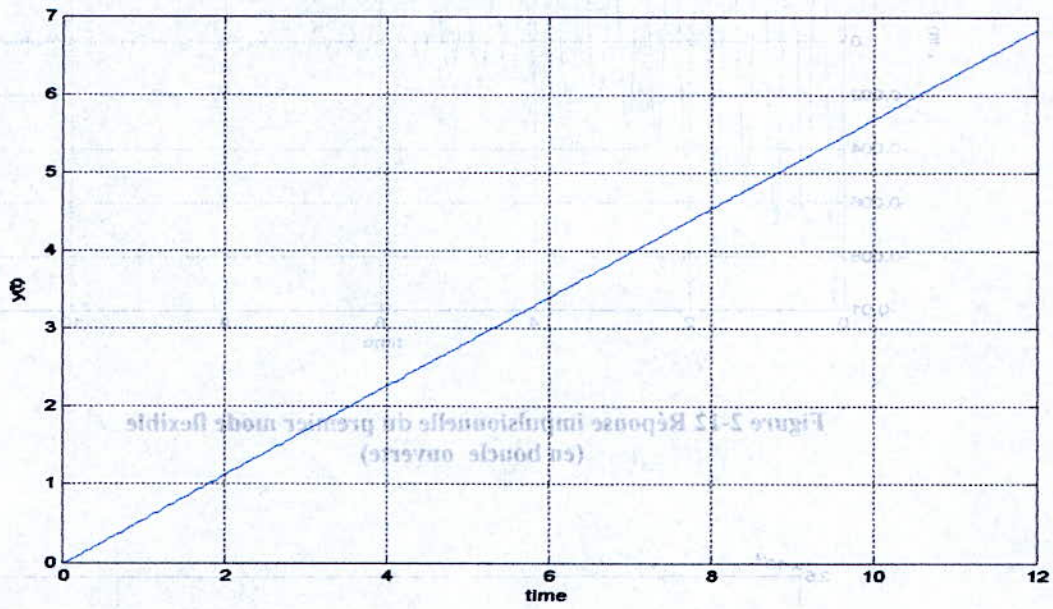


Figure 2-10 Réponse impulsionnelle en position de l'extrémité du bras (en boucle ouverte)

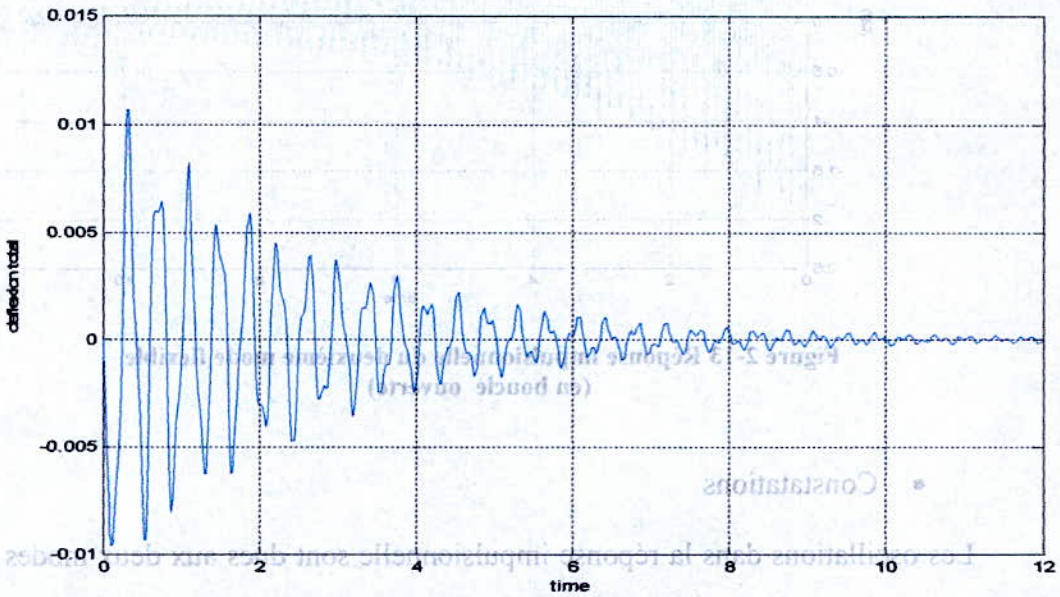


Figure 2-11 Réponse impulsionnelle de la déflexion totale (en boucle ouverte)

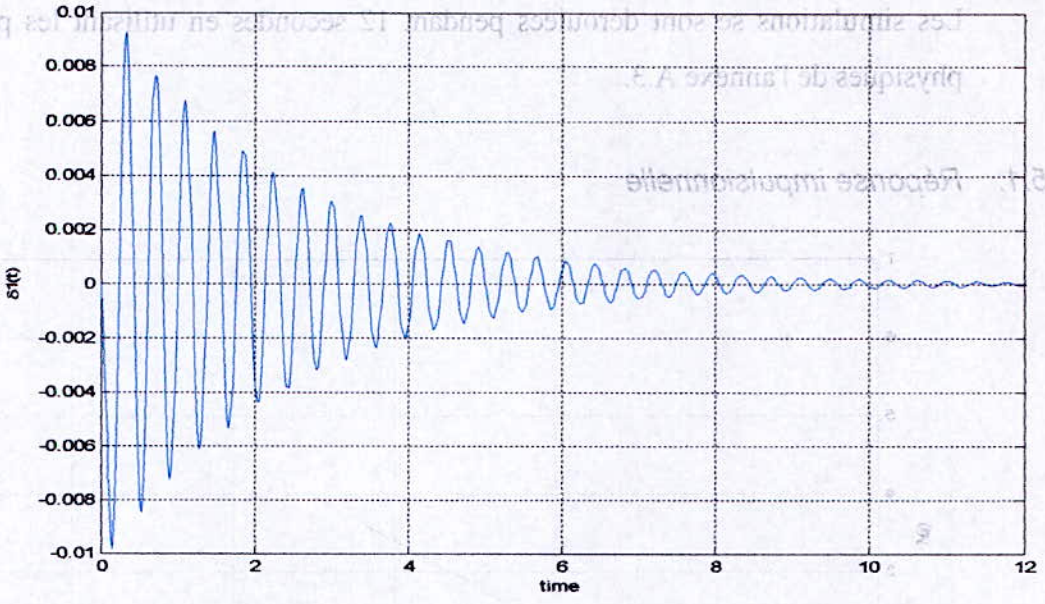


Figure 2-12 Réponse impulsionnelle du premier mode flexible (en boucle ouverte)

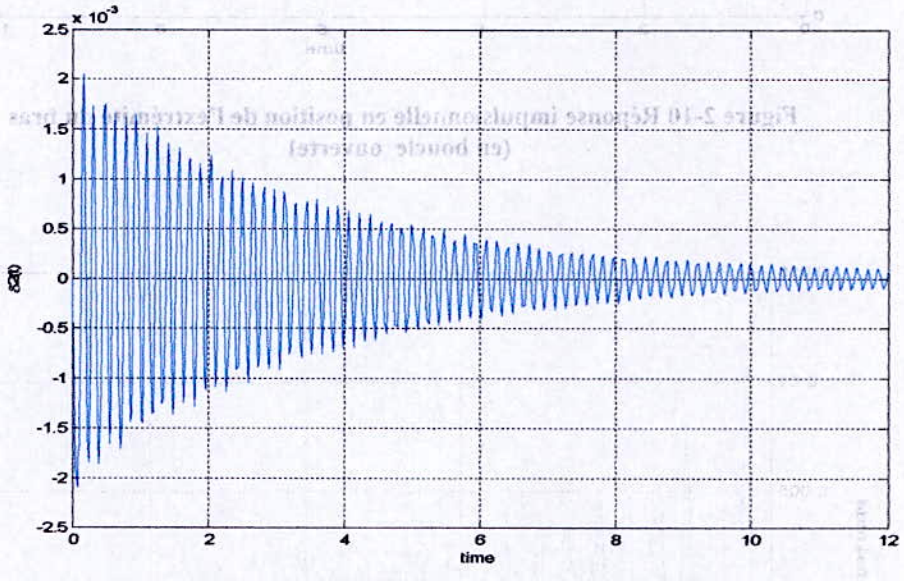


Figure 2-13 Réponse impulsionnelle du deuxième mode flexible (en boucle ouverte)

- Constatations

Les oscillations dans la réponse impulsionnelle sont dues aux deux modes flexibles, et ξ1 ξ2 provoque l'amortissement de ces oscillations.

2.5.2. *Réponse indicielle*

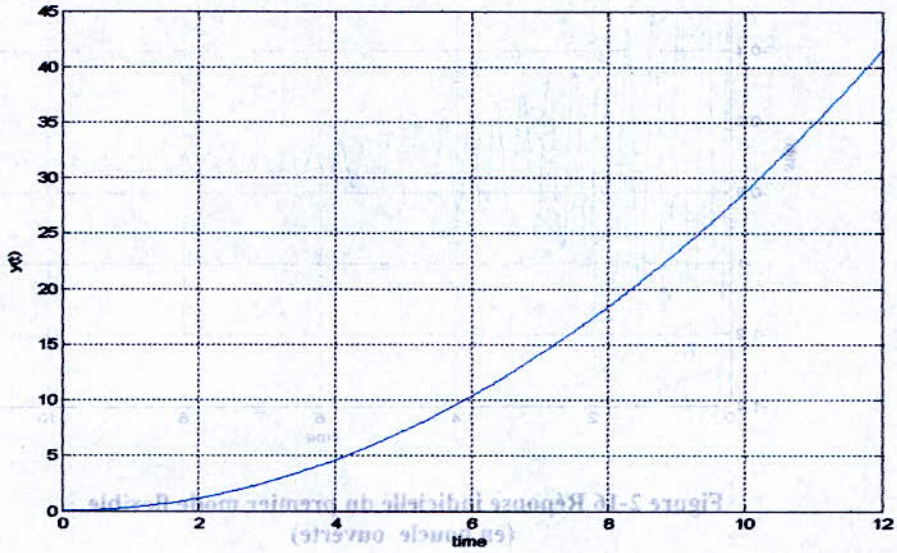


Figure 2-14 Réponse indicielle en position de l'extrémité du bras (enboucle ouverte)

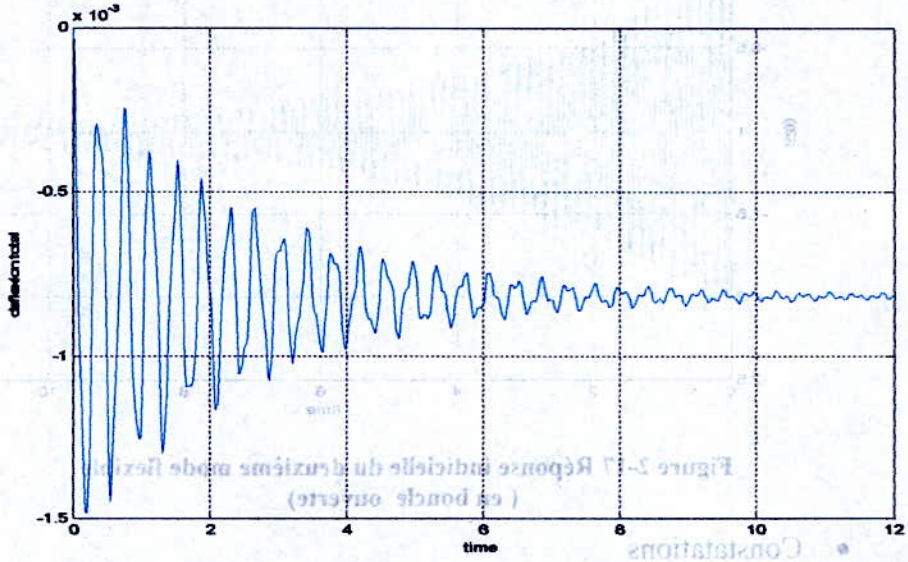


Figure 2-15 Réponse indicielle de la déflexion totale (en boucle ouverte)

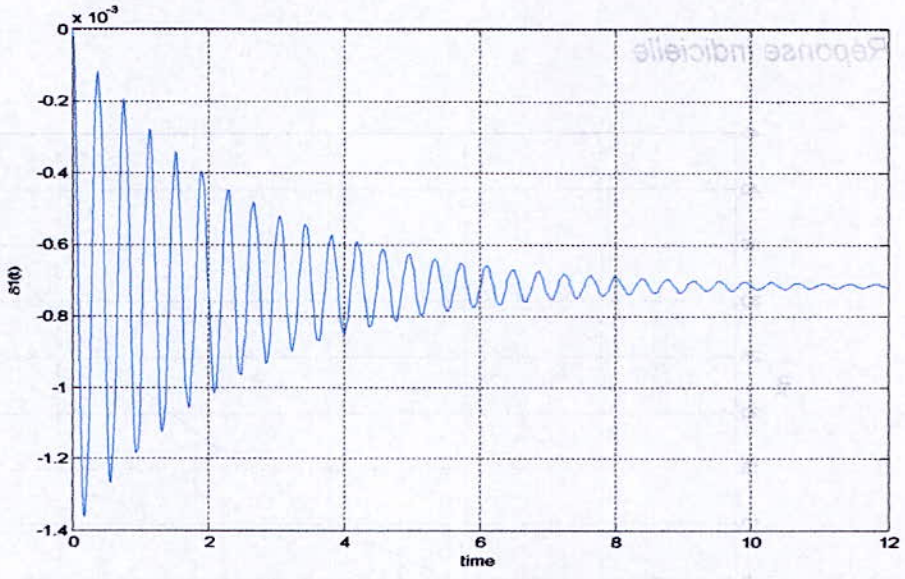


Figure 2-16 Réponse indicielle du premier mode flexible (en boucle ouverte)

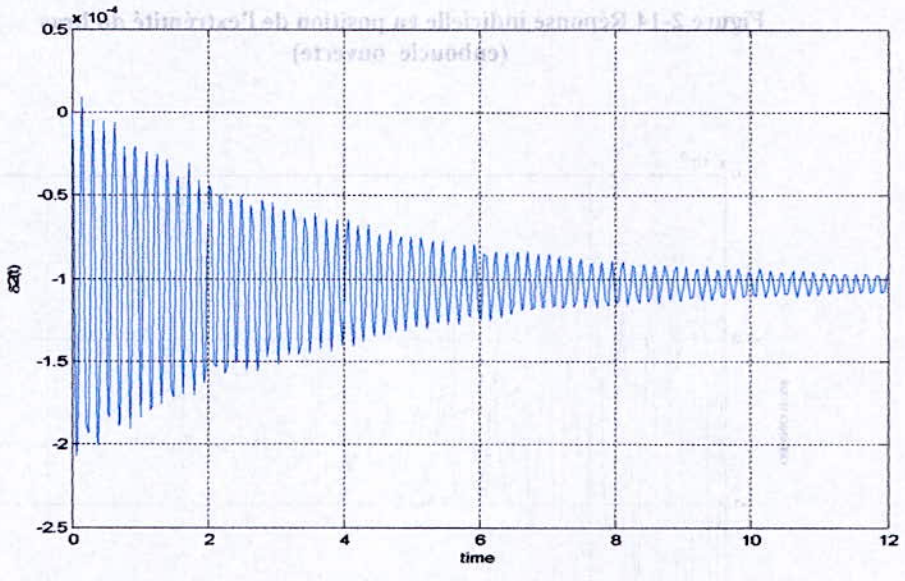


Figure 2-17 Réponse indicielle du deuxième mode flexible (en boucle ouverte)

- Constatations

Les oscillations ont disparu dans la réponse indicielle car le mode rigide a une amplitude plus grande que le mode flexible. On remarque une déflexion négative.

Les tests nous ont permis d'avoir un bon aperçu sur le comportement dynamique en boucle ouverte de notre système flexible en vue de sa commande.

2.6. RÉPONSE EN BOUCLE FERMÉE

Pour illustrer l'application des méthodes de commande conventionnelles, on a choisi d'utiliser la méthode de retour d'état, le modèle d'état déjà établi est utilisé.

2.6.1. *Retour d'état*

Pour commander le système tout en minimisant la commande, ou plutôt, avoir une commande réalisable sans abîmer notre actionneur, un contrôleur par retour d'état sera conçu et son schéma est montré dans la figure 2-18.

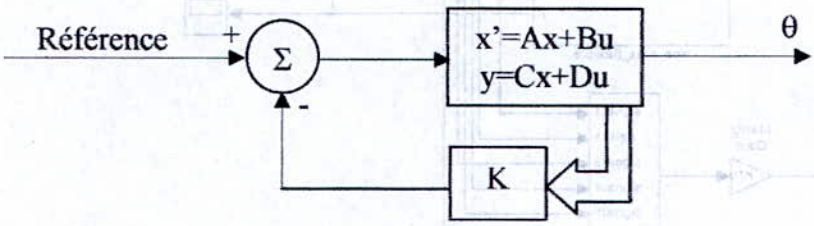


Figure 2-18 Contrôle par retour d'état

Pour concevoir le contrôleur, la méthode Linéaire Quadratique (LQR) est utilisée. En utilisant l'équation d'espace d'état, cette méthode trouve le K optimal basé sur la lois de retour d'état.

$$u = -Kx \qquad \textbf{Équation 2-41}$$

où un critère J doit être minimisé

$$J = \int \left(x' Qx + u' Ru\right) d\tau \qquad \textbf{Équation 2-42}$$

Q est une matrice poids indiquant l'importance relative de chaque état du système, **R** est une matrice poids qui indique l'importance relative des entrées. Puisqu'il y a seulement une entrée dans le système (mono variable), **R** prend la valeur 1. Pour calculer la matrice K, on fait appel à la fonction *LQR* du *CONTROL SYSTEMS TOOLBOX* de Matlab.

Le poids est placé de sorte que θ ,δ1 et δ2, aient une importance relative de 5000 comparée à l'action de commande.

Le vecteur K obtenu est :

[1.0000 -0.6172 -3.1774 0.4531 -0.4268 -0.4366]

Le contrôleur par retour d'état est présenté par le model SIMULINK figure 2-19, les différentes réponses obtenues seront représentées par la suite.

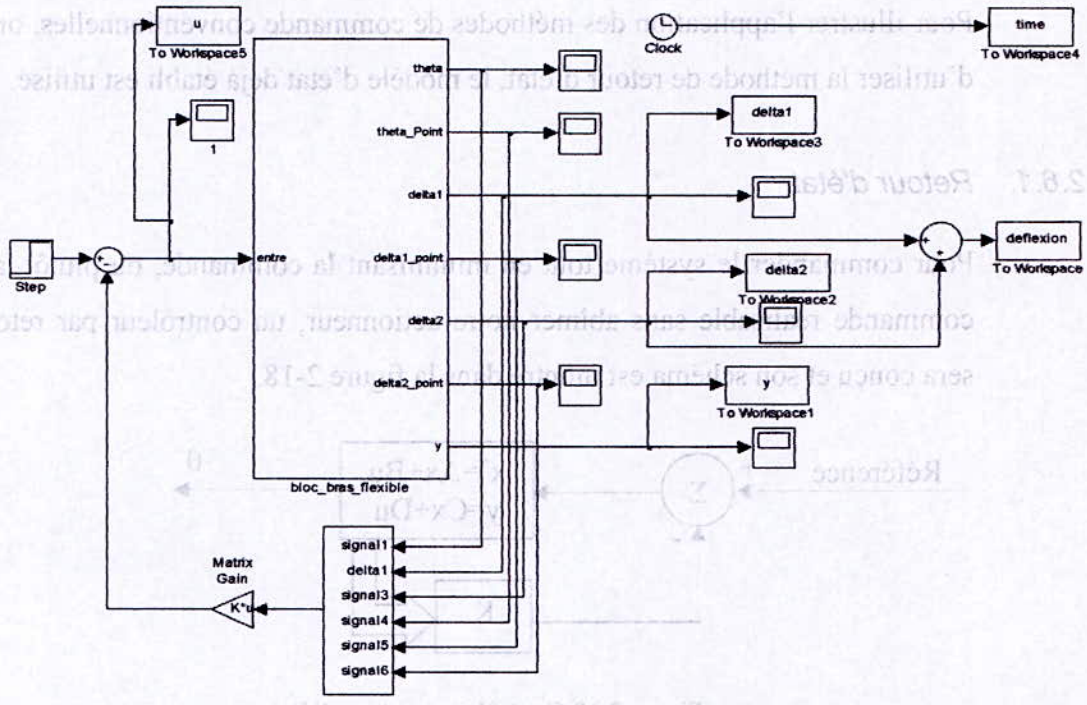


Figure 2-19 Réglage par retour d'état

2.6.2. *Réponse impulsionnelle*

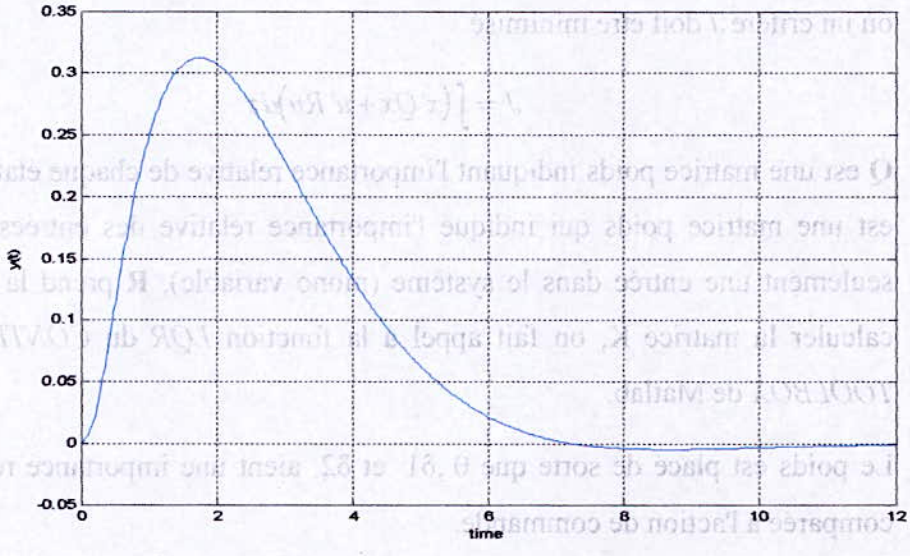


Figure 2-20 Réponse impulsionnelle en position de l'extrémité du bras (retour d'état)

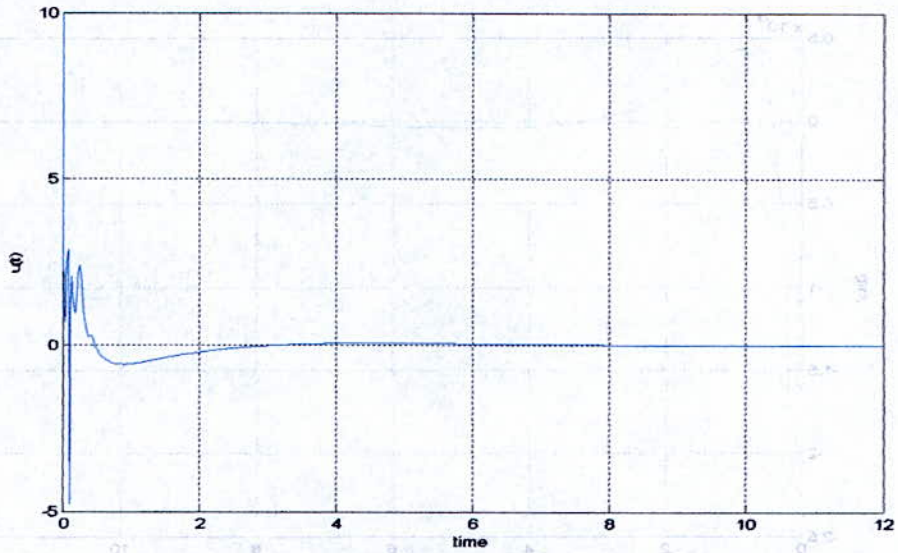


Figure 2-21 Commande pour la réponse impulsionnelle (retour d'état)

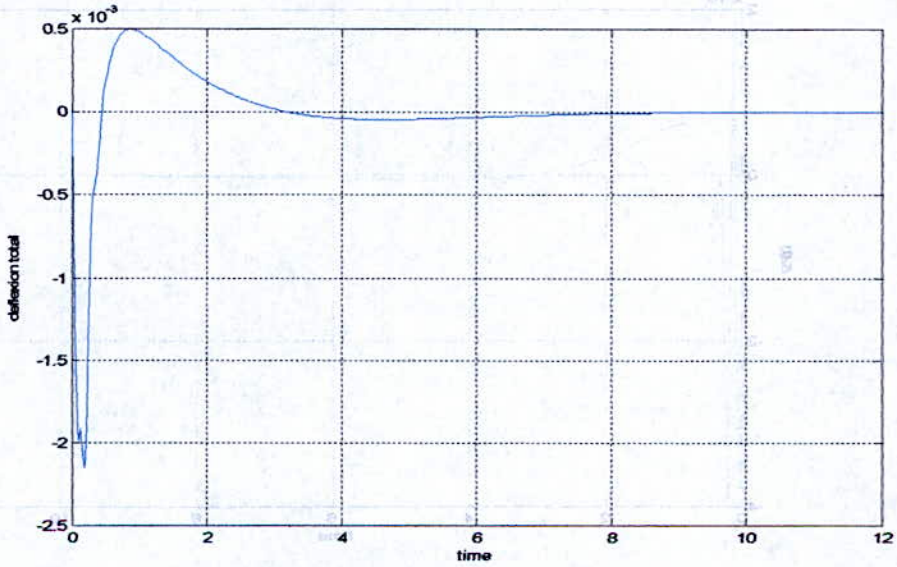


Figure 2-22 Réponse impulsionnelle de la déflexion totale (retour d'état)

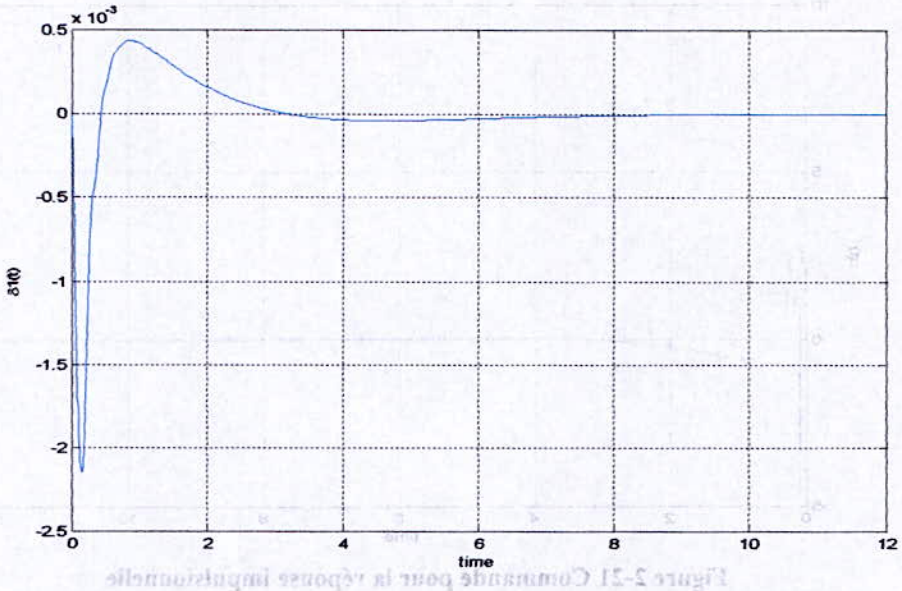


Figure 2-23 Réponse impulsionnelle de premier mode flexible (retour d'état)

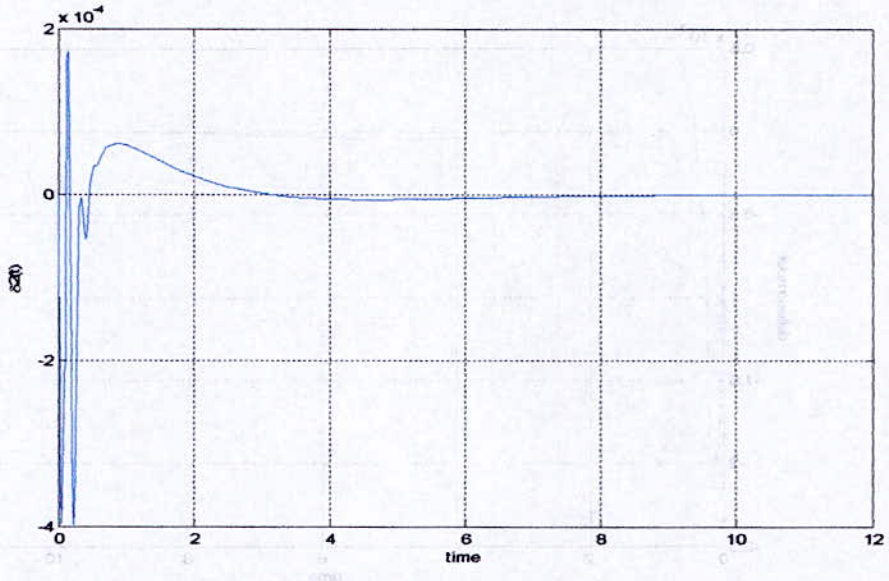


Figure 2-24 Réponse impulsionnelle de deuxième mode flexible (retour d'état)

2.6.3. *Réponse indicielle*

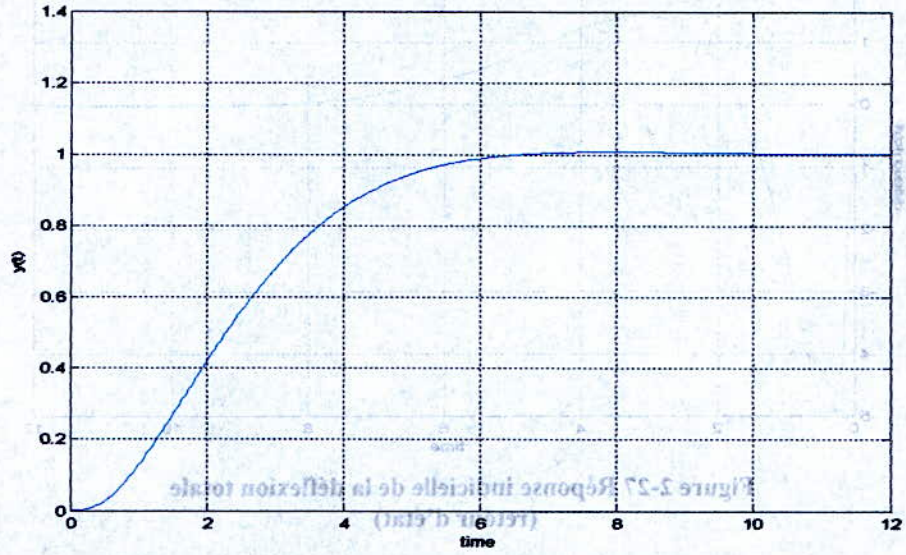


Figure 2-25 Réponse indicielle en position de l'extrémité du bras (retour d'état)

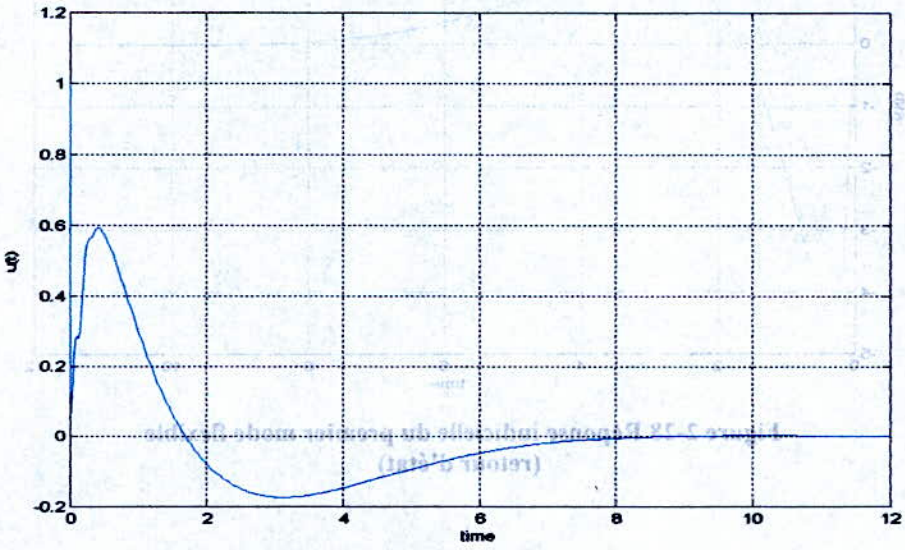


Figure 2-26 Commande pour la réponse indicielle (retour d'état)

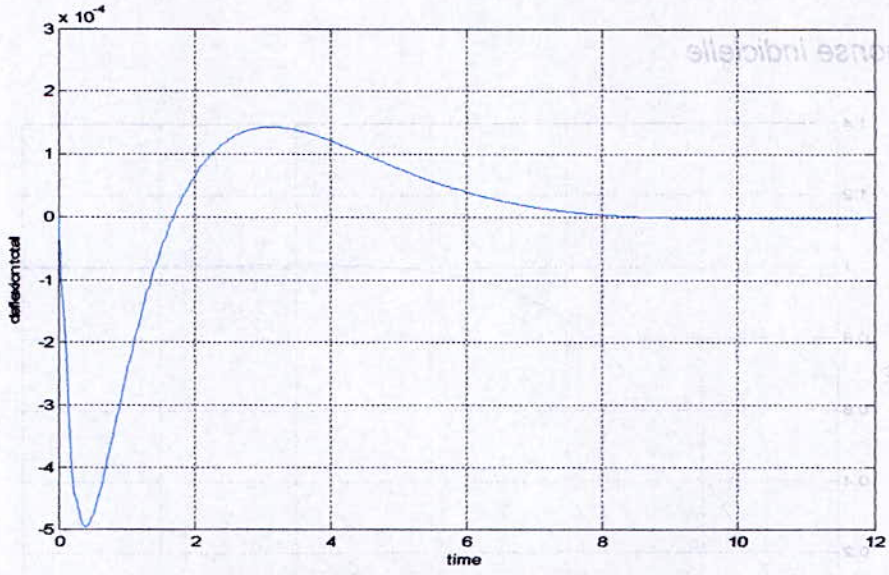


Figure 2-27 Réponse indicielle de la déflexion totale (retour d'état)

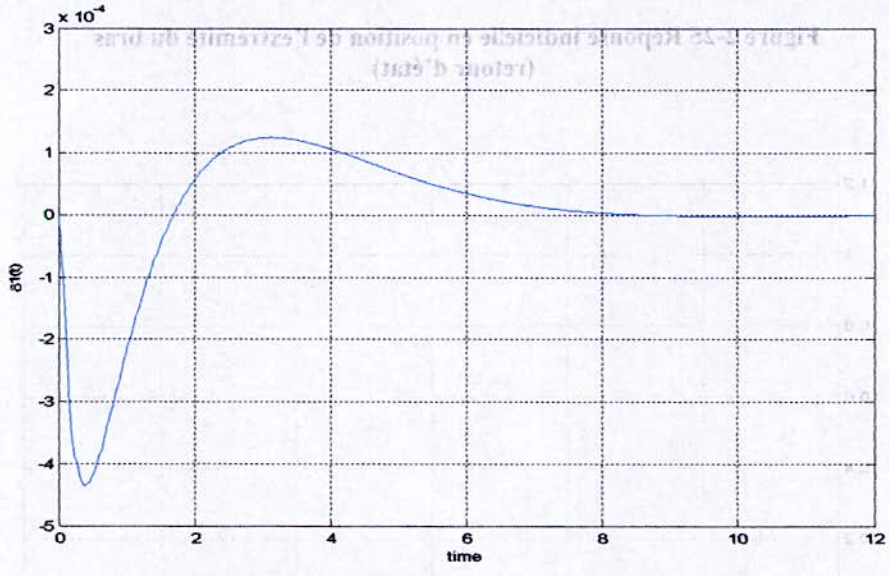


Figure 2-28 Réponse indicielle du premier mode flexible (retour d'état)

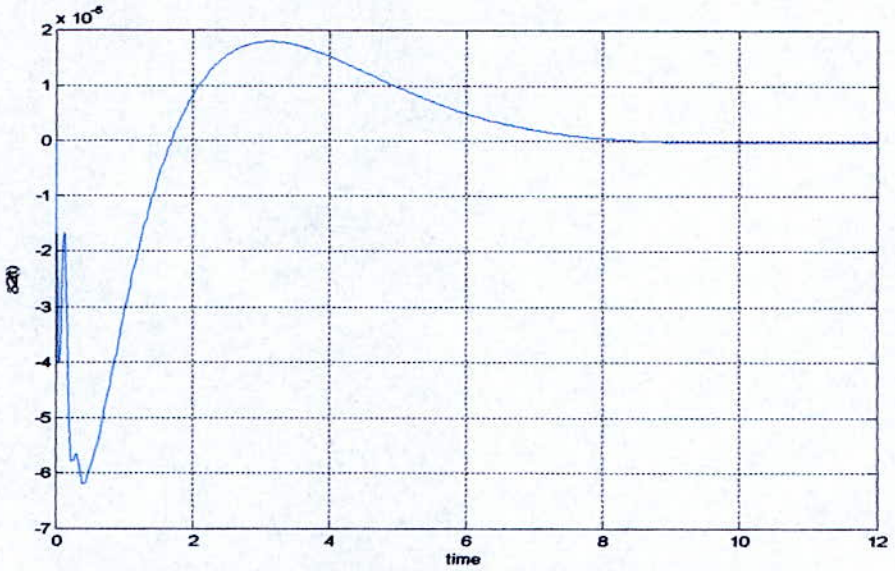


Figure 2-29 Réponse indicielle du deuxième mode flexible (retour d'état)

- Constatations

Les oscillations sont réduites de façon remarquable, mais la réponse reste très lente. Ce contrôleur ne réalise pas des performances satisfaisantes. De là on pense à utiliser un contrôleur non linéaire.

On se propose d'étudier les contrôleurs par logique floue. Ceux-ci pourraient répondre à nos attentes. Dans le chapitre suivant, une courte introduction au contrôle par logique floue est abordée.

LOGIQUE FLOUE ET COMMANDE FLOUE

3. LOGIQUE FLOUE ET COMMANDE FLOUE

La commande par logique floue [4] [5] est une méthode de contrôle non linéaire basée sur les principes de la logique floue développée par Zadeh en 1965[2] [3]. Elle essaye d'appliquer les connaissances d'un utilisateur expérimenté à la conception d'un contrôleur. Dans ce chapitre, les principes de base de la logique floue sont présentés, ainsi que les différents aspects de son utilisation on commande.

3.1. LOGIQUE FLOUE

La logique floue est basée sur la théorie des **ensembles flous** où les variables peuvent avoir des degrés d'appartenance à des ensembles ; à la différence de la théorie des ensembles la plus familière où une variable est un membre d'un ensemble ou ne l'est pas. Le degré auquel une variable appartient à un ensemble flou peut varier entre 0 et 1.

Un univers de discours défini la gamme entière des ensembles flous auxquels une variable peut appartenir. Chaque ensemble sur cet univers de discours est désigné sous le nom d'une **fonction d'appartenance.** Il est souvent décrit en utilisant « **une variable linguistique** ». Dans un univers de discours, une variable a un degré de vérité qui varie entre 0 et 1.

La logique floue emploie des règles avec des antécédents et des conséquences pour produire des sorties à partir des entrées. Les antécédents sont les entrées qui sont employées dans le processus décisionnel dites les parties « **Si** » des règles. Les conséquences sont les implications des règles dites les parties « **alors** » des règles.

3.1.1. Exemple de logique floue

Un exemple d'un ensemble flou est l'ensemble des personnes qui pourraient être décrites en tant que jeunes. La plupart des personnes conviendraient à dire que n'importe quelle personne âgée entre zéro et vingt ans pourrait être décrite en tant que jeune, tandis que si elle a plus de quarante ans, elle serait décrite en tant que 'pas du tout jeune'. Les âges entre vingt et quarante ans sont un secteur gris (ni vieux ni jeunes). Cependant, plus l'âge de la personne est au voisinage de vingt ans, plus elle

pourrait être décrite en tant que jeune facilement. Cet ensemble flou est montré sur le schéma ci-dessous. Selon cet ensemble flou une personne âgée de quinze ans est certainement jeune, c.-à-d. que leur degré de vérité à l'ensemble flou des jeunes est 1. Alors qu'une personne âgée de trente ans est moins définie comme étant jeune, c.-à-d. que leur degré de vérité est inférieur à 1.

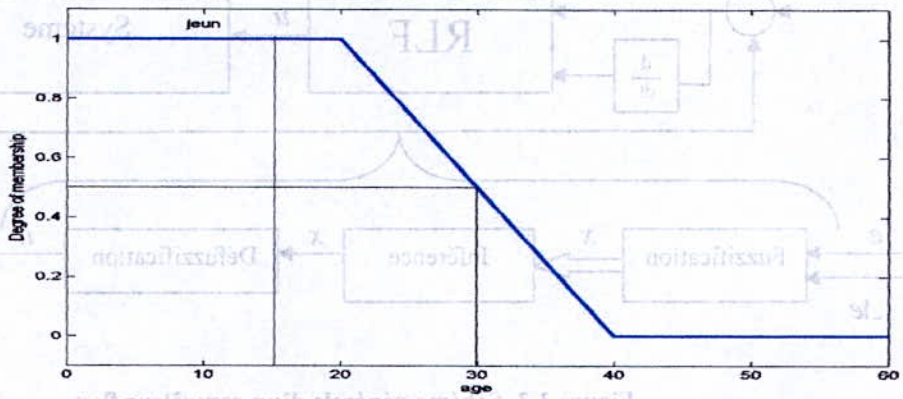


Figure 3-1 L'ensemble flou 'jeunes'

Dans cet exemple, L'univers de discours d'une variable est la gamme des valeurs que cette variable peut prendre. Beaucoup d'ensembles flous peuvent être définis sur un univers de discours, une variable simple peut avoir l'appartenance à plus d'un seul ensemble flou. Par exemple, sur le schéma ci-dessous nous revenons à notre exemple «âge ». En examinant l'univers de discours sur lequel, en plus des jeunes, on a les ensembles des vieux et ceux d'âge moyen. Ici nous pouvons voir qu'une personne âgée de 35 ans à l'appartenance à deux ensembles, jeunes et âge moyen avec des degrés de vérité de 0.26 et de 0.38 respectivement.

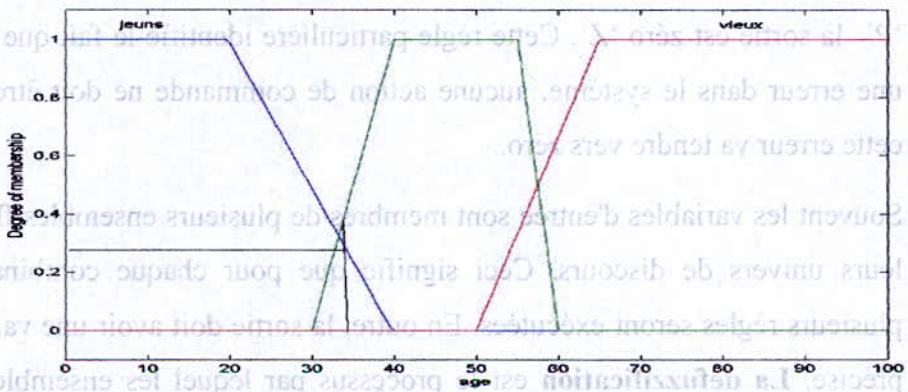


Figure 3-2 Univers de discours 'âge'

3.2. COMMANDE PAR LOGIQUE FLOUE

La commande par logique floue applique cette dernière à la commande des processus en utilisant différentes méthodes, habituellement l'erreur et le changement d'erreur sont utilisés comme entrées du CLF et la sortie sera la commande du système

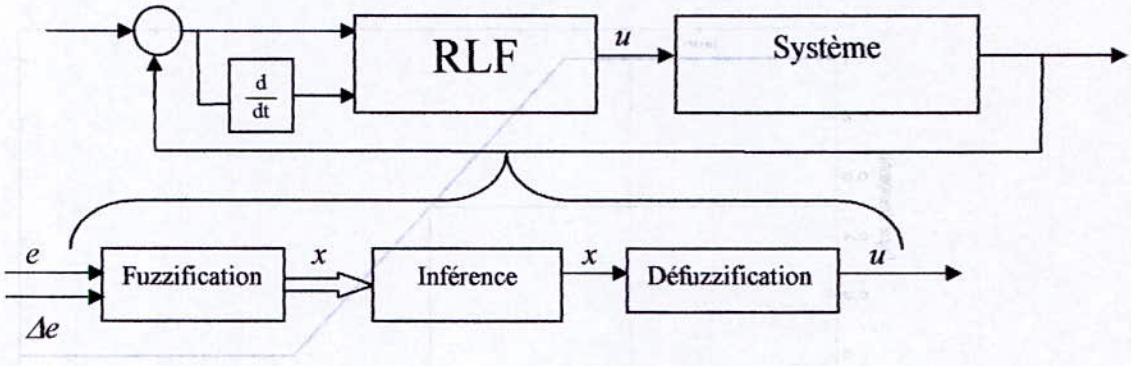


Figure 3-3 Schéma générale d'un contrôleur flou

La fuzzification transforme les variables d'entrée au variable linguistique. Les variables linguistiques utilisées sont souvent appelées « négatif grand», « positif », « zéro »...etc. **Les règles d'inférence** typiques pour deux entrées, une sortie et trois fonctions d'appartenance par variable sont montrées dans le tableau suivent :

DE \ E	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Tableau 3-1 Table d'inférence

Exemple : si l'erreur 'E' est négative 'N' et le changement de l'erreur 'DE' est positif 'P' la sortie est zéro 'Z'. Cette règle particulière identifie le fait que bien qu'il y ait une erreur dans le système, aucune action de commande ne doit être appliquée car cette erreur va tendre vers zéro.

Souvent les variables d'entrée sont membres de plusieurs ensembles flous définis sur leurs univers de discours. Ceci signifie que pour chaque combinaison d'entrée, plusieurs règles seront exécutées. En outre, la sortie doit avoir une valeur numérique précise. **La defuzzification** est le processus par lequel les ensembles des diverses règles lancées sont combinés pour produire la sortie.

3.2.1. *Illustration*

Pour illustrer ceci, en prend l'exemple suivant.

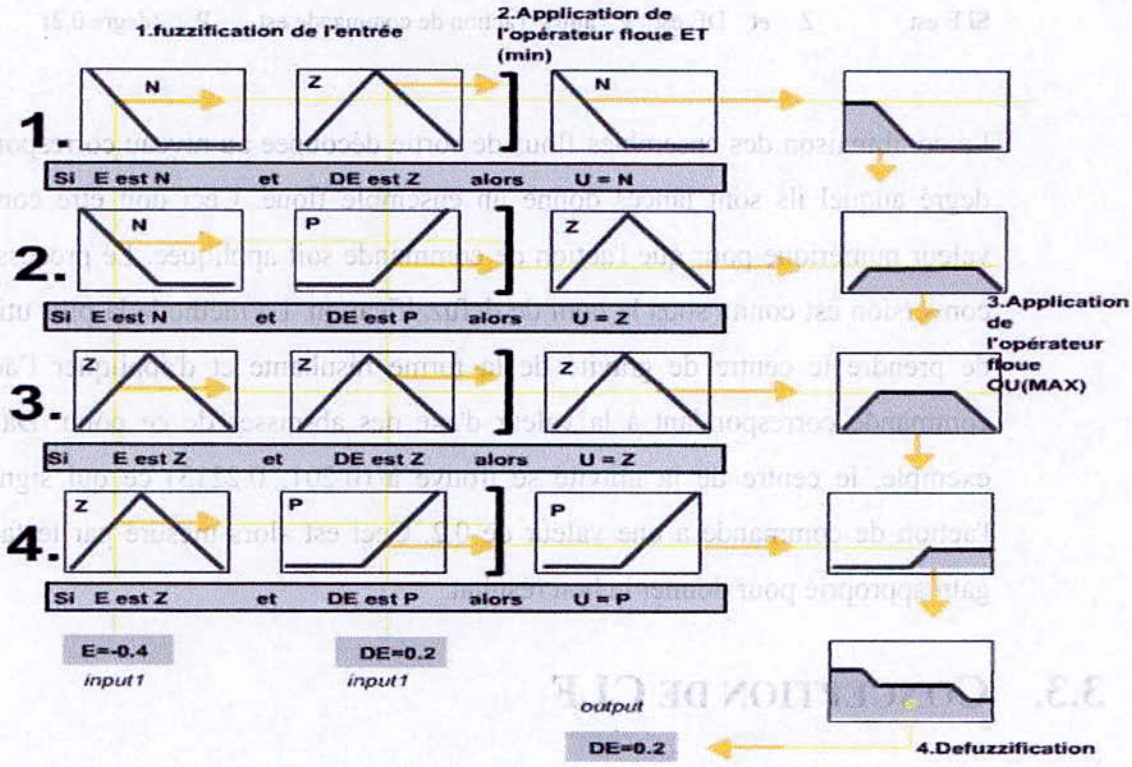


Figure 3-4 Schéma d'un contrôleur par logique floue

Le contrôleur par logique floue, CLF, possède deux entrées, erreur, changement d'erreur et une sortie. L'univers de discours pour chaque variable est normalisé de sorte que sa valeur se trouve toujours entre – 1 et 1. (Des gains sont appliqués à chaque variable pour donner la gamme appropriée). Il y a trois fonctions d'appartenance pour chaque variable et qui sont équidistant. Les règles d'inférence sont déjà données dans le **tableau 3-1**

Si l'erreur d'entrée est 0.4, ceci signifie qu'elle est membre de deux ensembles Z et P, avec un degré de vérité égale à 0.2 et 0.8 respectivement (voir la figure 3-4). Si le changement d'erreur à l'entrée est de – 0.2, alors ce dernier est membre de Z avec un degré de 0.6 et de N avec un degré de 0.4. Quatre règles sont lancées dans cette situation et le degré auquel chacune est exécutée est indiqué par le degré minimum des deux entrées qui l'ont lancée. (C'est comme ça que l'opération booléenne ET est effectuée dans la logique floue). Les règles lancées et les degrés auxquels elles sont exécutées sont :

Si E est	N	**et**	DE est	Z,	**alors** l'action de commande est	N	(degré 0.4)
Si E est	N	**et**	DE est	P,	**alors** l'action de commande est	Z	(degré 0.2)
Si E est	Z	**et**	DE est	Z,	**alors** l'action de commande est	Z	(degré 0.6)
Si E est	Z	**et**	DE est	P,	**alors** l'action de commande est	P	(degré 0.2)

La combinaison des ensembles flous de sortie découpée au niveau correspondant au degré auquel ils sont lancés donne un ensemble floue. Ceci doit être converti en valeur numérique pour que l'action de commande soit appliquée. Le processus de la conversion est connu sous le nom de defuzzification. La méthode la plus utilisée est de prendre le centre de gravité de la forme résultante et d'appliquer l'action de commande correspondant à la valeur d'axe des abscisses de ce point. Dans notre exemple, le centre de la gravité se trouve à (0.201, 0.2213) ce qui signifie que l'action de commande a une valeur de 0.2. Ceci est alors mesuré par le facteur de gain approprié pour donner le bon résultat.

3.3. CONCEPTION DE CLF

Quand on conçoit un contrôleur par logique floue (CLF), une connaissance experte du processus à commander peut être employée pour établir les fonctions d'appartenance et les règles d'inférence. Malheureusement il n'y a aucune procédure générale pour concevoir un CLF, vu que le temps est exigu et que beaucoup d'erreurs d'essais, peuvent être rencontrées lors de la réalisation d'un CLF. Dans cette approche traditionnelle, concevoir un CLF peut être un processus laborieux et long. Ces CLF tendent également à être non exportable et adaptable à d'autres applications.

Il serait judicieux de chercher à optimiser les différents paramètres concernant un CLF dont le choix reste plutôt subjectif. On se propos de faire appel aux algorithmes génétiques. Ce sont des algorithmes peuvent être employés pour chercher dans un grand espace de possibilité des solutions optimales. Leurs principes de base sont expliqués dans le chapitre suivant.

ALGORITHMES GENETIQUES

4. ALGORITHMES GÉNÉTIQUES

Les algorithmes génétiques sont des méthodes fiables et robustes pour chercher des solutions dans des espaces immenses. Ils sont inspirés de la théorie biologique d'évolution de la sélection naturelle [9].

4.1. BASES DES ALGORITHMES GÉNÉTIQUES

Un chromosome est une chaîne codée de valeurs utilisées pour l'optimisation des paramètres. Ces chromosomes peuvent se composer de chaînes à valeurs réelles ou binaires. Souvent un des défis principaux de conception d'un algorithme génétique, utilisé pour chercher une solution à un problème, est de trouver une manière appropriée de coder les paramètres.

Un ensemble de solutions potentielles, appelé une **population**, est créé. Chaque membre de cet ensemble est nommé **individu.** L'évaluation des populations est le fruit de l'application des paramètres décodés à partir du chromosome au problème (l'objectif qui doit être optimisé). Les points qu'un individu acquiert en accomplissant la tâche invoquée s'appellent **fitness**.

Après que le fitness de chaque individu soit calculé, un procédé connu sous le nom de **sélection** est exécuté. Des individus sont choisis pour contribuer à la création de la prochaine génération, la probabilité de la sélection étant liée au fitness de l'individu.

Une fois que la sélection est faite, un **croisement** aura lieu entre des paires d'individus choisis. Les chaînes de deux individus seront mélangées. De cette façon, on crée de nouveaux individus qui contiennent des caractéristiques différentes ; des individus relativement réussis.

Une troisième opération qui se produit est la **mutation ;** elle est un changement aléatoire dans le chromosome. Elle est généralement exécutée avec une probabilité relativement basse. La mutation s'assure que la probabilité de recherche dans une partie donnée de l'espace des solutions n'est jamais zéro.

Il existe beaucoup de manières pour l'application de ces différentes opérations. Différents algorithmes peuvent être employés pour chaque opération avec des degrés

variables de probabilité. Les algorithmes (les plus connus) de ces opérations sont maintenant examinés et leurs effets sur l'exécution de l'AG sont étudiés.

4.2. ALGORITHMES DE SÉLECTION

Un algorithme notoire de sélection est le prélèvement stochastique avec remplacement, généralement connu sous le nom d'algorithme «**Roue de Loterie**». Cette méthode fonctionne d'une manière analogue à une roue de Loterie. Chaque individu dans une population est assigné à une part de la roue, la taille de la part étant proportionnelle au fitness de l'individu. La taille est donnée par la formule :

$$Ps_i = f_i \Big/ \left(\sum_{j=1}^{N} f_j \right) \qquad \textbf{Équation 4-1}$$

Où : f_i : est la valeur de la fonction d'évaluation pour l'individu i (le fitness).

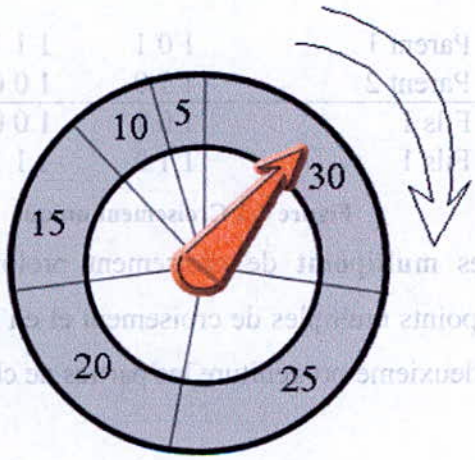


Figure 4-1 Roue de Loterie

Un indicateur est tourné (un nombre aléatoire produit) et l'individu pointé est sélectionné. Ceci continue, jusqu'à ce que le nombre requis d'individus construit une population. La probabilité d'un individu de la sélection est ainsi liée à son fitness. L'inconvénient de cette approche est qu'elle ne garantie pas la représentation des individus les mieux adaptés dans la prochaine génération.

La sélection **stochastique du reste** est un autre algorithme très connu qu'on retrouve souvent. Dans cette méthode l'espérance du nombre de fois choisi est calculée pour chaque individu. Les parties de nombre entier de cette espérance pour chaque individu sont assignées de manière déterministe et les restes partiels sont assignés

comme dans la sélection de 'Roue de Loterie'. Par exemple, un individu dont l'espérance du nombre de fois choisi est de 2.4, il serait sûr d'être sélectionné deux fois et la probabilité d'être sélectionné une troisième foi sera 0.4. Cette approche réduit le désaccord lié à l'algorithme de 'Roue de Loterie' et s'assure que tous les individus avec le fitness au dessus de la moyenne seront représentés dans la génération suivante.

4.3. ALGORITHMES DE CROISEMENT

Une fois que la sélection est finie le croisement est exécuté. Des individus sont couplés en mélangent leurs chromosomes pour créer de nouveaux individus. L'algorithme de croisement fondamental est connu en tant que **croisement unique**. Un point le long de la chaîne est choisi et les chaînes sont permutées dans ce point.

Parent 1	1 0 1	1 1 1 0 0
Parent 2	1 1 0	1 0 0 1 0
Fils 1	1 0 1	1 0 0 1 0
Fils 1	1 1 0	1 1 1 0 0

Figure 4-2 Croisement unique

Les algorithmes **multipoint** de croisement prolongent le croisement simple en choisissant les points multiples de croisement et en leurs assignant alternativement à la première ou deuxième progéniture les parties de chaîne entre ces points

Parent 1	1 0	1 1 1	1 0 0
Parent 2	1 1	0 1 0	0 1 0
Fils 1	1 0	0 1 0	1 0 0
Fils 1	1 1	1 1 1	0 1 0

Figure 4-3 Croisement multipoint

Le **croisement uniforme** est un autre algorithme de croisement. Cette fois un masque aléatoire de 1 et de 0 de la même longueur que les chaînes de parents est produit. Si un bit dans le masque est 1, alors le bit correspondant dans le premier fils viendra du premier parent et le deuxième parent donnera ce bit au deuxième fils. Si le bit de masque est 0 le premier parent contribue au deuxième fils et le deuxième parent au premier fils.

Parent 1	1 0 1 1 1 1 0 0
Parent 2	1 1 0 1 0 0 1 0
masque	0 1 0 0 1 1 0 1
Fils 1	1 0 0 1 1 1 1 0
Fils 1	1 1 1 1 0 0 0 0

Figure 4-4 Croisement avec masque (uniforme)

Le croisement uniforme est le plus turbulent des algorithmes de croisement, c'est à dire. il est le plus susceptible d'engendrer les bits voisins qui contribueront d'une manière positive au fitness de l'individu parent. En même temps, le croisement uniforme tient compte d'une recherche plus étendue de l'espace de solution malgré qu'il y ait sensiblement plus de progéniture potentielle en utilisant cette méthode.

4.4. ALGORITHMES DE MUTATION

Pour des codages binaires une seule méthode existe pour la mutation. Pour chaque bit on génère un nombre aléatoire, si ce dernier est plus petit que la probabilité indiquée de mutation, alors le bit est renversé. Cette probabilité de mutation est généralement petite et elle est constante durant toute l'exécution de L'AG. Cependant, une très bonne astuce existe, elle utilise une variation de la probabilité de mutation durant l'exécution de l'algorithme, et ceci en commençant par un taux relativement élevé et en diminuant de façon continue pendant que l'AG progresse. Cette méthode permet à l AG de rechercher plus les solutions potentielles au départ et de s'accrocher à la meilleure solution en convergeant vers elle.

L'algorithme de mutation peut être plus complexe lors d'un codage réel.

Différents algorithmes sont employés pour la mutation, certains sont donnés comme suit :

- **Mutation uniforme** : on choisi une valeur aléatoire pour le changement.
- **Mutation limite** : le changement est placé à sa limite inférieure ou supérieure.
- **Mutation non uniforme** : le changement est assigné à une valeur basée sur une courbe en cloche qui devient progressivement plus étroite pendant que l'AG progresse. Ceci s'assure que pendant que l'on converge vers la solution, l'intervalle dans lequel une variable subit une mutation se réduit.

4.5. ÉLITISME

Les solutions optimales peuvent être perdues à cause du croisement et de la mutation, en plus de l'absence de garantie pour que ces opérations préservent le meilleur fitness. Une stratégie élitiste consiste à conserver d'une génération à l'autre, dans la population, au moins l'individu ayant la meilleure adaptation. Dans ces modèles, le meilleur individu d'une population est sauvé avant que les opérations aient lieu. Après que la nouvelle population se soit formée et évaluée, on l'examine pour voir si cette meilleure structure a été préservée. Sinon, la copie sauvée est réinsérée de nouveau dans la nouvelle population à la place du membre le plus faible. L'AG effectue les opérations sur cette même nouvelle population.

4.6. CODAGE

Les algorithmes génétiques peuvent être exécutés en utilisant les codages binaires ou à valeurs réelles. Avec le codage binaire, chaque paramètre est converti en chaîne binaire. Ces chaînes sont attachées et les opérations génétiques sont effectuées sur cette chaîne enchaînée. Par contre les paramètres du codage à valeurs réelles sont maintenus dans leur vrai format. Dans la pratique Les deux formes de codages sont employées.

Les avantages d'employer un format binaire sont que les alphabets binaires tiennent compte d'un plus grand prélèvement de l'espace de solution et que ce format traite un plus grand nombre de combinaisons.

Parfois le codage réel peut être plus efficace. Par exemple, si un certain paramètre pouvait prendre cinq valeurs, il devrait être codé en utilisant trois bits. Cependant, ceci mène à huit combinaisons possibles, trois d'entre elles sont superflues. Dans ce cas-ci l'emploi d'un alphabet de cinq lettres (réel) donne un codage plus efficace.

4.7. TEST DES PARAMÈTRES DE L'AG

Beaucoup de facteurs parviennent à changer la façon dont un AG est exécuté, tel que la taille de population, la probabilité de mutation et la probabilité de croisement et d'autres. Dans le but de l'étude de ces facteurs, le programme *multim.m* est crée (figure 4-5).

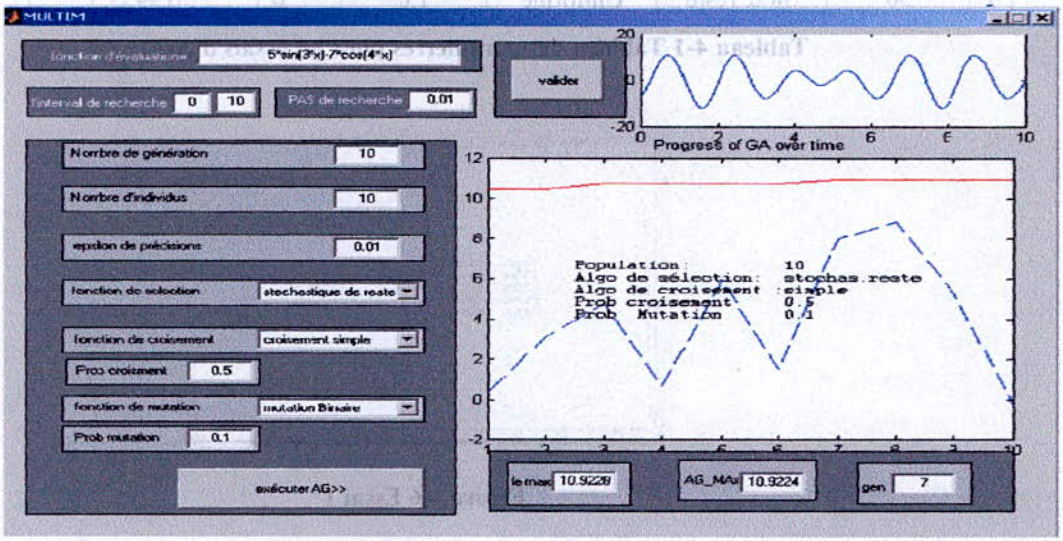


Figure 4-5 MULTIM.m

Pour la fonction d'évaluation on écrit une fonction avec beaucoup de maximums, de différentes valeurs, cette fonction est définie dans un intervalle choisi. L'AG recherche le maximum dans cet espace de solution. Il existe plusieurs maximums locaux, l'AG qui les évite, en tombant sur le vrai maximum, est sauvegardé (ses paramètres) pour être utilisé par la suite dans l'optimisation d'un contrôleur par logique floue.

Pour faire l'étude, on prend la fonction d'évaluation « 3sin (5x)- 4cos (7x)» dans l'intervalle [0,10] avec un pas de 0.01; on prend 100 générations, la mutation binaire et on fait varier les facteurs suivants :

- La taille de la population entre 10, 50 et 100.
- L'algorithme de sélection **stochastique du reste et Roue de Loterie**
- L'algorithme de croisement ; **unique et uniforme**
- Probabilité de croisement entre 0.5 et 1.0
- Probabilité de mutation entre 0.01 et 0.1.

Le tableau suivant montre les essais sur l'AG avec leurs résulta

	La taille de la population	L'algorithme de sélection	L'algorithme de croisement	Probabilité de croisement	Probabilité de mutation	fitness	génération
1	10	Stoc. reste	Uniforme	1.0	0.01	6.9293	44
3	100	Stoc. reste	Uniforme	1.0	0.01	6.9455	55
2	50	Stoc. Rem.	Uniforme	1.0	0.01	6.9455	18
4	50	Roue Loterie	Uniforme	1.0	0.01	6.9455	21
5	50	Stoc. reste	unique	1.0	0.01	6.9455	37
6	50	Stoc. reste	Uniforme	0.5	0.01	6.9455	25
7	50	Stoc. reste	Uniforme	1.0	0.1	6.9455	11

Tableau 4-1 Tableau des paramètres pour les essais d'AG

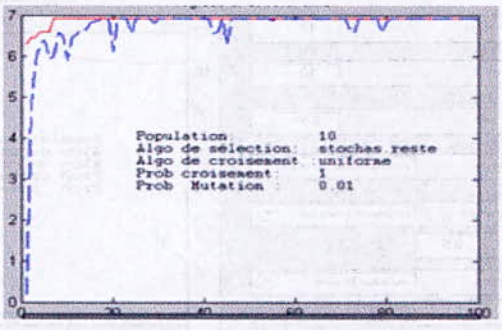


Figure 4-6 Essai 1

Dans les cent premières générations, seulement l'essai 1 (**Figure 4-6**) n'a pas pu trouver la valeur maximale réelle. L'exécution a convergé à la valeur finale avant la cinquantième génération cela nous conduit à dire que l'exécution a collée à un maximum local. Ceci suggère que la taille de la population soit très importante pour une exécution d'un algorithme génétique.

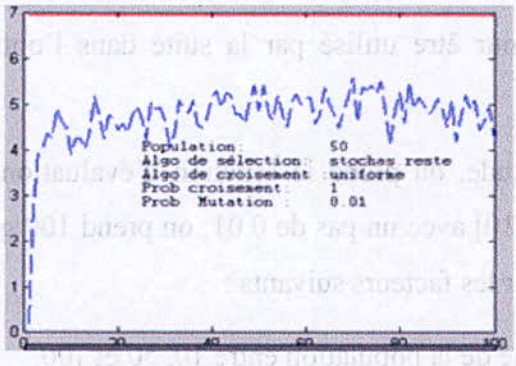


Figure 4-7 Essai 2

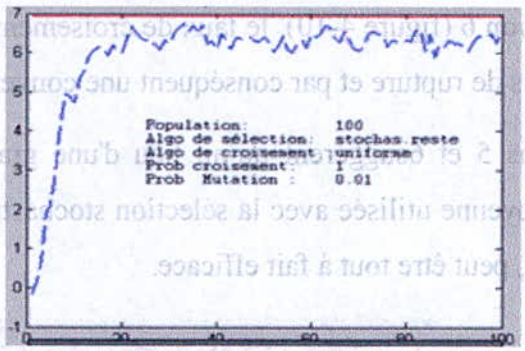


Figure 4-8 Essai 3

Les exécutions 2 et 3 diffèrent dans leur taille de population. La moyenne du fitness de L'exécution 3 et 2 (Figure 4-8, 4-7) est plus grande que celle d'une recherche aléatoire, mais pour l'exécution 2 la moyenne fitness est assez basse pour dire que l'AG fasse une recherche significative de l'espace de solution. Si l'AG converge très rapidement (prématurément), il pourrait facilement bloquer dans un maximum local.

En prenant les paramètres de l'exécution 2 comme référence, on a étudié l'effet de changer les autres paramètres.

Figure 4-9 Essai 5

Dans l'exécution 5 (figure4-9), l'algorithme de croisement change de l'uniforme à un croisement unique. Ceci a l'effet d'augmenter le taux auquel la moyenne du fitness approche le fitness maximum, prouvant que le croisement unique est moins agitateur que le croisement uniforme.

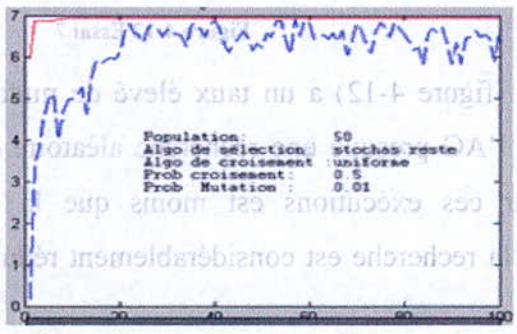


Figure 4-10 Essai 6

Dans l'exécution 6 (figure 4-10), le taux de croisement est changé en 0.5. Ceci mène encore à moins de rupture et par conséquent une convergence plus vite.

Les exécutions 5 et 6suggèrent qu'au lieu d'une grande taille de population, une population moyenne utilisée avec la sélection stochastique de reste et les taux élevés de croisement, peut être tout à fait efficace.

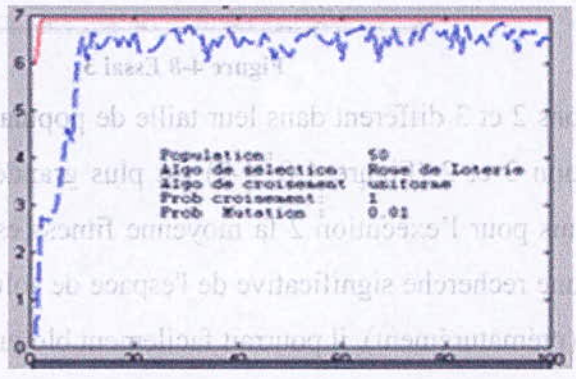


Figure 4-11 Essai 4

L'algorithme de sélection de Roue de Loterie est essayé dans l'exécution 4 (Figure 4-11). Ceci mène également à la convergence rapide. En raison du désaccord réduit dans des probabilités de sélection, les individus moins adaptés contribuent plus dans l'algorithme stochastique de sélection de reste que dans la Roue de Loterie. Ceci devrait être confirmé avec plus de recherche.

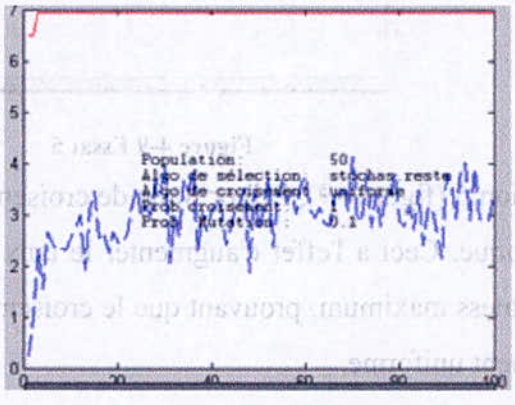


Figure 4-12 Essai 7

L'exécution 7 (figure 4-12) a un taux élevé de mutation (10%). Le taux élevé de mutation rend l'AG presque une recherche aléatoire car la moyenne du fitness des individus pour ces exécutions est moins que la moitié du maximum, c.-à-d. l'efficacité de la recherche est considérablement réduite. On dit que cette recherche est aléatoire.

Après avoir gagné une bonne connaissance de la façon dont les divers paramètres affectent l'exécution d'un AG. Celui-ci peut maintenant être appliqué à la tâche principale qui est d'optimiser un CLF. Mais avant, une manière d'automatiser la conception du CLF sous Matlab, doit être au préalable réalisée. Ce problème est affiné, et résolu dans le prochain chapitre.

CONCEPTION D'UN CLF SOUS MATLAB

5. CONCEPTION D'UN CLF SOUS MATLAB

Dans ce chapitre, on aborde les détails de la conception d'un CLF de façon automatique. Les hypothèses utilisées et les contraintes rencontrées pour une telle conception, sont expliquées ci-dessous.

5.1. HYPOTHÈSES ET CONTRAINTES

Pour appliquer un CLF au bras flexible, certaines propriétés du système sont exploitées pour faciliter la conception. La symétrie du système exploitée nous conduit à adopter :

- Des fonctions d'appartenance symétriques par rapport à l'axe des ordonnées.
- Des règles d'inférence également symétriques.

Les autres conditions et hypothèses qui se présentent à la conception du CLF sont :

- Tous les univers de discours sont normalisés pour se trouver entre – 1 et 1. Les gains externes au CLF donnent les valeurs appropriées aux variables.
- Seulement des triangles sont employés pour les fonctions d'appartenance.
- Le nombre d'ensembles flous est un nombre entier impaire supérieur a 1 (3, 5, 7, 9). En combinaison avec la condition de symétrie, ceci signifie que la fonction centrale d'appartenance pour toutes les variables aura son sommet à zéro.
- Les sommets formant la base d'un triangle correspondent aux sommets des triangles adjacents, ceci assure que la valeur de n'importe quelle variable d'entrée est un membre de deux ensembles flous. De plus, on est assuré que si le degré d'appartenance d'une variable à un ensemble est 1, elle n'appartiendra à aucun autre ensemble adjacent.
- On suppose que les premières et dernières fonctions d'appartenance ont leurs sommet au – 1 et 1 respectivement ; Ceci peut être justifié par le fait que le changement de gain aura l'effet semblable du changement des positions de ces sommets

5.2. CONCEPTION DES FONCTIONS D'APPARTENANCE

En appliquant ces conditions, la conception des fonctions d'appartenance peut être décrite en utilisant deux paramètres :

- Le nombre de fonctions d'appartenance.
- Le positionnement des sommes des triangles.

La méthode suivie pour la conception est inspirée des travaux de [24] et [8], où les sommets *C (i)* sont décrits par l'équation:

$$C(i) = sign(i)\left(\frac{|i|}{\left(\frac{n-1}{2}\right)}\right)^{Pm} \text{ tel que } \quad i = \left[-\frac{n-2}{2}, \ldots, 0, \cdots, \frac{n-2}{2}\right] \qquad \textbf{Équation 5-1}$$

et où

n : nombre de fonction d'appartenance.

Pm : paramètre d'espacement

L'espacement des centres varie suivant Pm, si Pm=1 les fonctions d'appartenances restent équidistantes, si Pm est supérieure à 1 les fonctions se regroupent au centre, si Pm est inférieure à 1 les sommets se regroupent à l'extérieur, voir Figure 5-1.

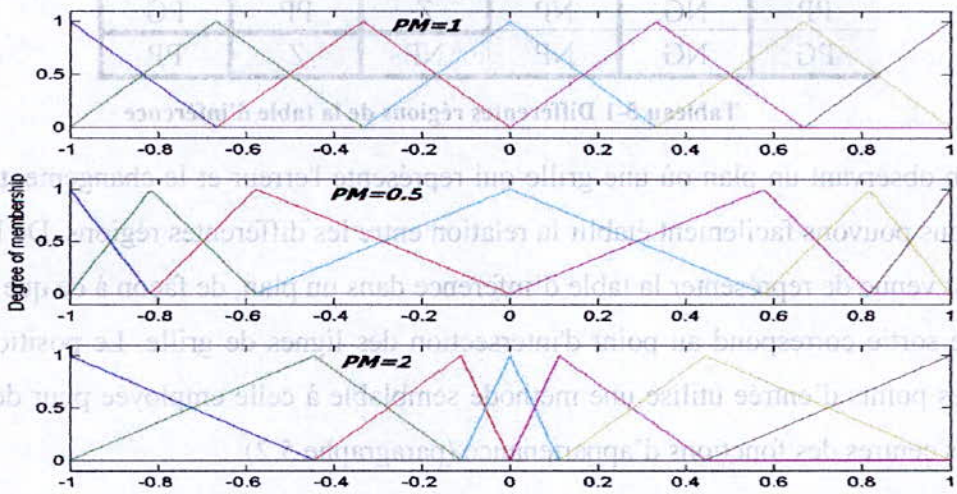


Figure 5-1 Effet du paramètre d'espacement Pm.

Cette méthode rend la conception plus simple, mais, elle réduit le nombre de CLF possibles. Tout de même, cette méthode permet au CLF de répondre aux exigences de conception.

Un CLF identifie une fonction d'appartenance à partir des trois paramètres qui sont les abscisses des sommets du triangle. La fonction *Appartenance.m* calcule ces paramètres pour chaque fonction d'appartenance en utilisant comme entrée le nombre et l'exposant de l'espacement de ces fonctions (Pm et n).

5.3. CONCEPTION DES RÈGLES D'INFÉRENCE

La conception des règles d'inférence est inspirée des idées de Park [24]. La construction des règles emploie des paramètres d'espacement pour chaque variable, un angle spécifique et le nombre de fonctions d'appartenances des entrées et de la sortie.

Suivant le tableau, l'espace entier de la table de règle peut être divisé en multiples régions qui ont la même sortie. Suite à cette division, le nombre de régions obtenues correspond au nombre de fonctions d'appartenance de la sortie.

	NG	NP	Z	PP	PG
NG	NP	Z	PP	PP	PG
NP	NG	NP	Z	PP	PG
Z	NG	NP	Z	PP	PG
PP	NG	NP	Z	PP	PG
PG	NG	NP	NP	Z	PP

Tableau 5-1 Différentes régions de la table d'inférence

En observant un plan où une grille qui représente l'erreur et le changement d'erreur, nous pouvons facilement établir la relation entre les différentes régions. De là, l'idée est venue de représenter la table d'inférence dans un plan, de façon à ce que la valeur de sortie correspond au point d'intersection des lignes de grille. Le positionnement des points d'entrée utilise une méthode semblable à celle employée pour déterminer les centres des fonctions d'appartenance (paragraphe 5.2).

Les régions indiquées dans la table d'inférence peuvent être également montrées dans le plan, et cela en les séparant avec des lignes, la figure 5-2 illustre tout ça.

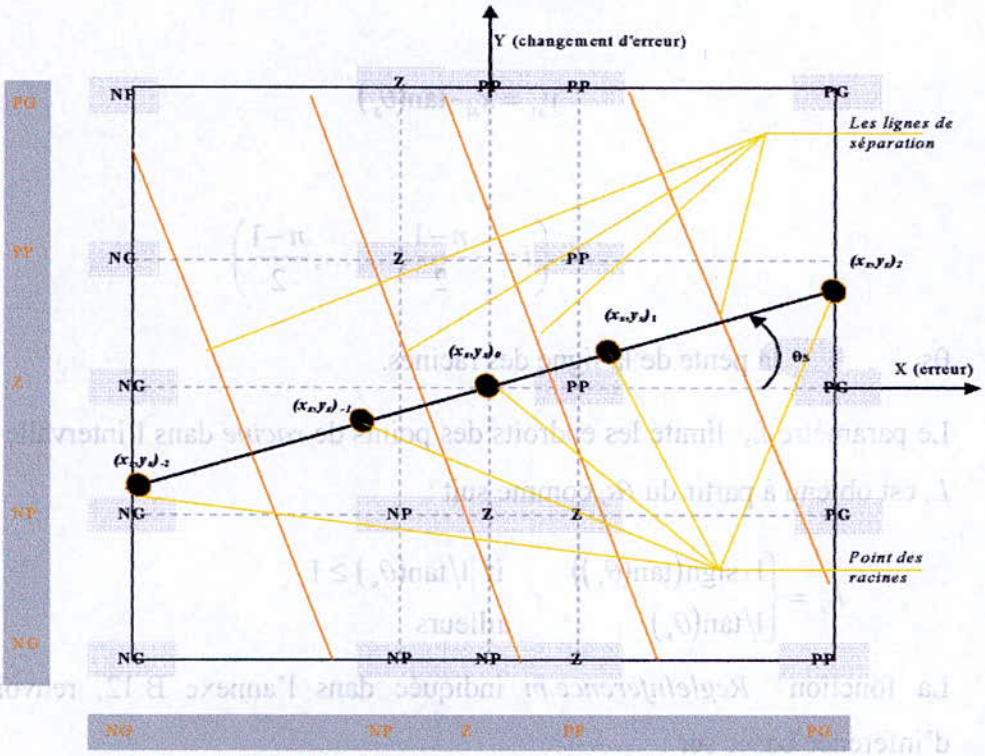


Figure 5-2 Les racines et la grille pour la construction des règles d'inférence

Pour avoir des paramètres de conception plus simples, on suppose que les lignes de séparation sont droites et parallèles. Si cela est vérifié, une ligne qui passe par l'origine et qui est orthogonale aux lignes de séparation existe, cette ligne s'appelle ligne des racines. Dans cette ligne, on pose des points de racine, leurs noms correspond au nom des variables linguistique de sorties, touts ça pour dire qu'une règle d'inférence qui est représentée par un point d'intersection de lignes de grille, prendra comme sortie la variable linguistique du point de racine le plus proche du point d'intersection de grille. Ceci signifie que la table des règles peut être obtenue en déterminant les points de racine.

Par conséquent, les paramètres des règles d'inférence peuvent être obtenus à partir des positions des points de racine. Les positions des points de racine sont déterminées en suivant la même procédure utilisée pour la détermination des sommets des fonctions d'appartenance :

$$x_{si} = L_s \cdot \mathrm{sign}(i) \cdot \left(\frac{|i|}{\frac{n-1}{2}} \right) \qquad \textbf{Équation 5-2}$$

$$y_{si} = x_{si} \cdot \tan(\theta_s) \qquad \textbf{Équation 5-3}$$

$$\left(i = -\frac{n-1}{2}, \ldots, \ldots, \frac{n-1}{2} \right)$$

θs : la pente de la ligne des racines.

Le paramètre L_s limite les endroits des points de *racine* dans l'intervalle du plan, ce L_s est obtenu à partir du θs, comme suit :

$$L_s = \begin{cases} 1 \cdot \text{sign}(\tan(\theta_s)) & \text{if } |1/\tan(\theta_s)| \geq 1 \\ 1/\tan(\theta_s) & \text{ailleurs} \end{cases} \qquad \textbf{Équation 5-4}$$

La fonction *RegleInference.m* indiquée dans l'annexe B.12, renvoie la règle d'inférence basée sur :

- Le nombre de fonctions d'appartenance par variable,
- Les paramètres d'espacement pour chaque variable
- L'angle pour la ligne de racine.

Au début, on calcule les coordonnés des points de racine et les coordonnés des points d'intersection de la grille, en suite, en mesurant la distance pour chaque point de racine avec les points d'intersection, on choisit la plus courte. Les sorties de chaque règle sont donc trouvées. Les entrées et les sorties seront alors retournées dans une matrice dont le format est exigé par *'Fuzzy Logic Toolbox'* de Mtlab. [12][13].

5.4. CONCEPTION DU CONTRÔLEUR PAR LOGIQUE FLOU

La fonction *RegulateurLF.m* fabrique le système flou d'inférence (FIS) en employant les deux fonctions *Appartenance.m* et *RegleInference.m*.

Le code pour *RegulateurLF.m* est montré dans l'annexe B.13. Cette fonction appelle *Appartenance.m* pour obtenir les paramètres des fonctions d'appartenance de chaque variable. Alors une règle d'inférence appropriée est créée pour chaque variable de sortie en appelant *RegleInference.m*, qui renvoie une matrice de règles dont le format est exigé par *'Fuzzy Logic Toolbox'* de Mtlab. Ces informations sont écrites de

manière à créer le FIS. Il est a noter que seules des fonctions triangulaires d'appartenance peuvent être créées en utilisant *RegulateurLF.m*.

Avec tous ces programmes un FIS complet peut être engendré en utilisant seulement quelques paramètres. C'est idéal pour l'utilisation d'un AG, car celui-ci utilise ces simples paramètres pour améliorer les caractéristiques du CLF.

Mais pour déterminer la fonction que l'AG devra maximiser (fitness), nous utilisons d'abord cet AG pour la conception d'un PID, pour chercher le bon fitness. Le prochain chapitre illustre les procédures de cette réalisation.

COMMANDE GENETIQUE PID

6. COMMANDE GÉNÉTIQUE PID

Dans ce chapitre, on cherche à appliquer les AGs à la conception d'un PID pour la commande du bras flexible. Les diverses expériences seront associées à différents critères de performance, et ce dans le but d'adopter le critère le plus représentatif de notre problématique.

6.1. DESCRIPTION DU TRAVAIL

Pour appliquer un algorithme génétique à la conception d'un contrôleur PID, différents moyens d'évaluation et de conception sont exigés. Cette évaluation doit être exécutée de façon rapide pour qu'un AG puisse traiter un grand nombre de combinaisons possibles.

Les **fonctions d'évaluation** sont des fonctions appelées par un AG pour calculer le fitness d'un ensemble de paramètres. Les paramètres sont transmis à la fonction d'évaluation, qui les traite et renvoie une valeur décrivant à quel point ces mêmes paramètres ont accomplis leur tâche.

Pour ce projet la fonction d'évaluation *ControlPID_Sim.m* a été écrite. Ceci est montré dans l'annexe B.15. Cette fonction extrait les paramètres appropriés à partir du chromosome et les emploie pour fixer les trois gains du contrôleur PID ; Kp, Ki et Kd.

La fonction d'évaluation appelle le modèle de SIMULINK, illustré dans la figure 6-1. Où l'erreur, la dérivée de la commande et le temps sont retournés durant toute la durée de la simulation pour le calcul du fitness.

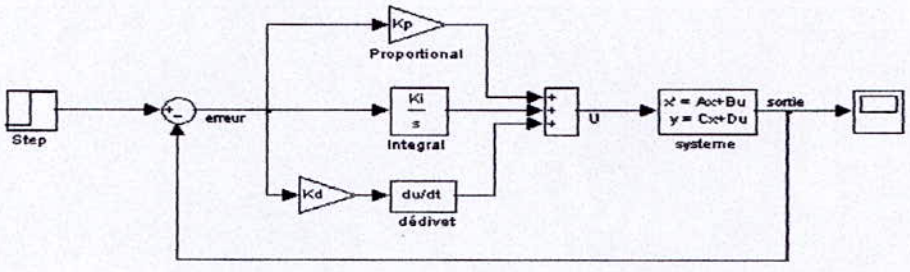


Figure 6-1 système de commande par contrôleur PID

Le fitness est une valeur calculée à base d'un critère de performance. L'objectif du contrôleur étant : la capacité du bras flexible à suivre la consigne de l'entrée avec une

erreur minimale, il peut être exprimé en termes de minimisation des critères à usage courant tels que IAE (intégrale des erreurs absolues), ISE (l'intégrale des erreurs carrées) et ITAE (intégrale des erreurs absolues multipliées par le temps)....Chacun de ces critères a ses propres performances, à savoir :

- IAE : pour un amortissement moyen et une réponse transitoire peu oscillatoire.
- ISE : pour minimiser l'énergie du signal.
- ITAE : il regroupe les deux mais il n'est utilisé que pour la simulation en raison de la complexité des calculs.

Si on opte pour ITAE on prend:

$$J_{ITAE} = \int_0^{\infty} t|e(t)|dt \quad \text{donc en discret } J_{ITAE} = \sum_{k=1}^{n} kT \cdot |e(k)|$$

Puisque cette fonction est minimale pour le meilleur individu, la fonction du fitness doit être l'inverse, car le meilleur individu doit avoir le fitness le plus grand.

$$fitness = \frac{1}{J_{ITAE}}$$

Mais l'AG optimise seulement pour satisfaire la condition sur l'erreur, cela mène dans la plupart des cas à des commandes non réalisables.

Après des nombreux essais on a trouvé nécessaire d'incorporer une pénalité dans le fitness pour les solutions qui produisent de grandes variations de la commande *u*, pour cela le critère suivant est élaboré :

$$J = \sum_{i=1}^{n} \alpha \cdot |e(i)| + \beta \cdot |u(i) - u(i-1)|$$

où

α , β sont des facteurs à trouver.

Le vrai défi est de trouver des valeurs pour α , β qui donnent de très bonnes réponses, tout en ayant une commande réalisable.

Après plusieurs recherches on a trouvé des réponses qui sont représentées dans les figures 6-1et 6-2

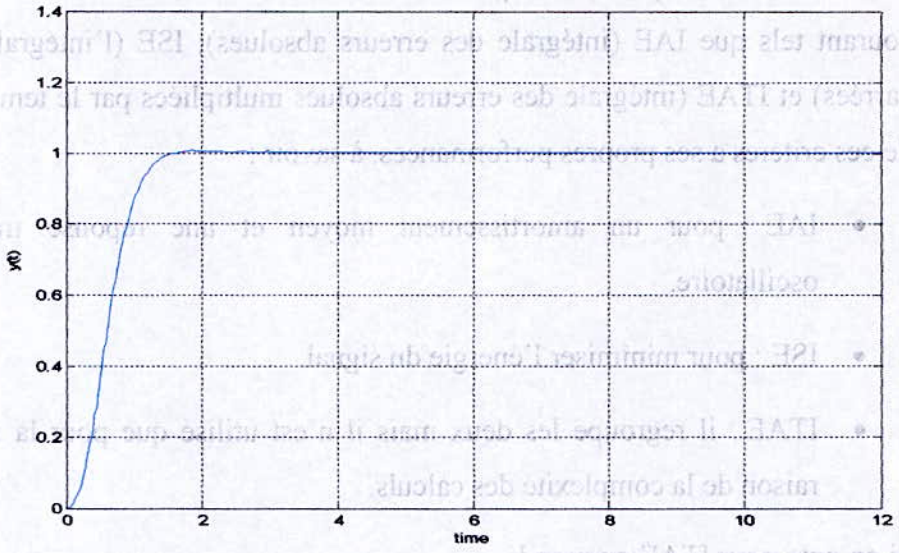


Figure 6-2 Réponse indicielle de l'extrémité du bras (PID-AG)

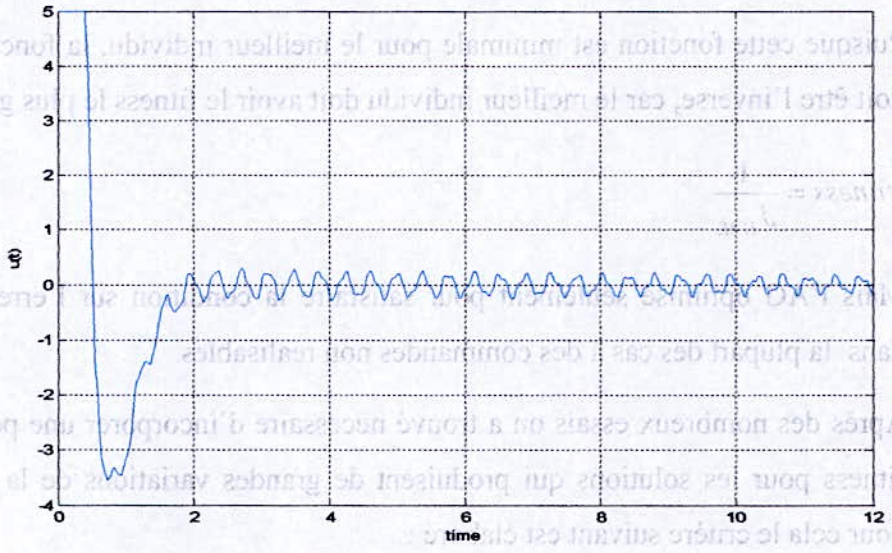


Figure 6-3 commande pour la réponse indicielle (PID-AG)

Ces réponses correspondent aux valeurs de Kp, Kd et Ki, respectivement, 24.5, 11.1 et 0.1. Ces résultats sont obtenus en prenant, $\alpha=5\ 10^5$, et $\beta=100$.

Ainsi on a résolu le problème du fitness pour notre système. Par la suite nous utilisons ce critère de performance pour la recherche d'un contrôleur par logique floue.

COMMANDE PAR CLF OPTIMISÉ PAR L'AG

7. COMMANDE PAR CLF OPTIMISÉ PAR L'AG

Dans ce qui suit, on applique les AGs pour concevoir le meilleur CLF pour la commande de notre bras flexible, en utilisant le critère de performance trouvé dans le chapitre précédent

7.1. MODÈLE DE SIMULINK

Le modèle SIMULINK employé est montré sur la Figure 7.1. L'erreur et le changement de l'erreur sont pondérés par les gains appropriés (ces paramètres sont également fixés par l'AG) et le résultat est limité de sorte qu'il se situe dans l'intervalle [-1, 1]. Ces entrées sont introduites dans le CLF et la sortie du CLF est alors pondérée par un autre gain, la commande générée est appliquée au système dont les réponse est la position de l'extrémité du bras flexible.

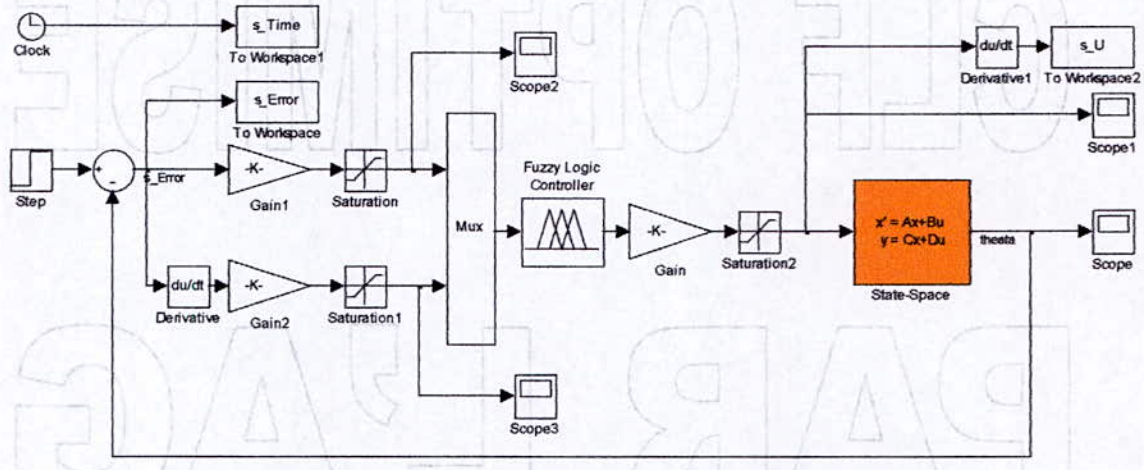


Figure 7-1 Modèle SIMULINK pour le contrôle à logique floue

7.2. CODAGE DES PARAMÈTRES

L'exécution d'un AG nécessite l'affectation d'une plage de variation pour chaque variable. Pour notre travail, les paramètres, leur plages de variation ainsi que leur pas de précision sont donnés dans le tableau 7-1. Nous utilisons le codage binaire car nous estimons que ce dernier permet d'avoir une recherche plus complète dans l'espace de solutions.

Paramètre	Intervalle	Pas de précision	Nombre de bits pour le codage
Nombre de fonction d'appartenance	[3 , 9]	2	2
Espacement des fct d'appartenance	[0.1 , 1]	0.01	7
Euissance pour L'espacement	[-1 , 1]	2	1
Espacement des règles d'inférence	[0.1 , 1]	0.01	7
Puissance pour les règles	[-1 , 1]	2	1
Angle des règles d'inférences	[0 , 2π]	π /521	11
Gain de sortie et de l'entrée	[0 , 200]	0.1	11

Tableau 7-1 Paramètres utilisés pour le codage

Le nombre des fonctions d'appartenance est limité aux nombres entiers impairs entre trois et neuf, car la plupart des CLF rencontrés dans la littérature utilisent cette marge. Ceci constitue une contrainte qui réduit les cas possibles. Mais, le vrai avantage est que cette information peut être saisie dans deux bits par variable.

Pour l'espacement, deux paramètres séparés sont employés. Le premier est la grandeur qui varie dans l'intervalle [0.1 – 1.0] avec un pas de 0.01, le second est une puissance par laquelle la grandeur doit être élevée ; elle prend seulement les valeurs – 1 ou -1. La puissance détermine si la concentration des fonctions d'appartenance est au centre ou à l'extrémité. La précision proposée pour la grandeur est 0.01, cela signifie que 8 bits sont employés au total pour chaque paramètre d'espacement.

Le gain des variables d'entrée et de sortie de varie dans l'intervalle [0 – 1000]. Ces valeurs ont été trouvées après quelques essais.

Cette manière de coder nous donne un chromosome de 98 bits pour chaque individu. Ceci signifie qu'il y a 2^{98} solutions (cas possibles) soit environ $3x10^{30}$. Une partie très petite représente les contrôleurs stables. Ceci indique que l'espace de recherche est très grand. Si l'AG réussi à trouver des solutions optimales dans un si grand espace et cela sans aucune connaissance à priori, l'AG serait un instrument très puissant.

7.3. EXÉCUTION DE L'ALGORITHME GÉNÉTIQUE

PROGRAMME.M est le programme principal où les paramètres de l'AG sont fixés. Le code source de ce programme est donné dans l'annexe B.6. Les fonctions Matlab *Execution_AG.m* et *initialiser_AG.m* sont des versions modifiées de **GAOT** [18] et **GAT** [16], elles sont exposées dans les annexe B.10 et B.8 respectivement.

On a modifié ce code :

- Pour éliminer les bugs découverts au cours de la conversion du binaire en valeurs réelles et vice-versa.
- Pour changer la façon de l'exécution du programme, en utilisant une interface graphique. **Figure7-2**

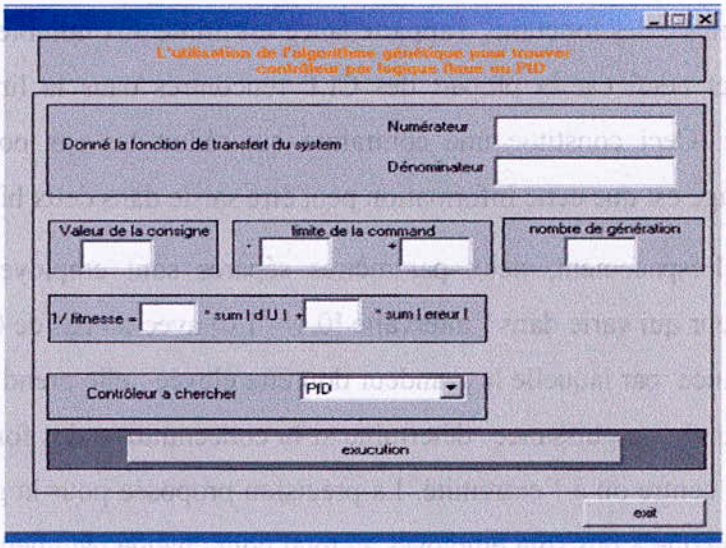


Figure 7-2 Interface graphique du programme principal

Dix exécutions d'AG ont été réalisées pour cent générations. En changeant la probabilité de croisement et la probabilité de mutation, le meilleur contrôleur a été trouvé pendant $9^{ème}$ exécution. Cette exécution a été prolongée pour deux cents générations, mais le meilleur fitness n'a pas vraiment augmenté. Les détails du CLF trouvé par l'AG sont présentés sur **la figure 7-3, le tableau 7-2** et **la figure 7-4**. Le nombre des fonctions d'appartenances assignées à l'erreur, sa dérivée et la commande sont respectivement cinq, sept et sept. Pour la deuxième entrée, les fonctions d'appartenance sont concentrées autour du centre, alors qu'elles sont concentrées vers les extrémités pour la sortie. Le graphe représenté sur **la figure 7.5**

montre la relation entrées-sortie dans l'espace, on remarque qu'elle est à tendance linéaire et qu'il n'existe pas de pics. Les contrôleurs par logique floue linéaires permettent la création de ce genre de graphe; le but d'un contrôleur linéaire de deux entrées, sera toujours d'obtenir un plan dans l'espace.

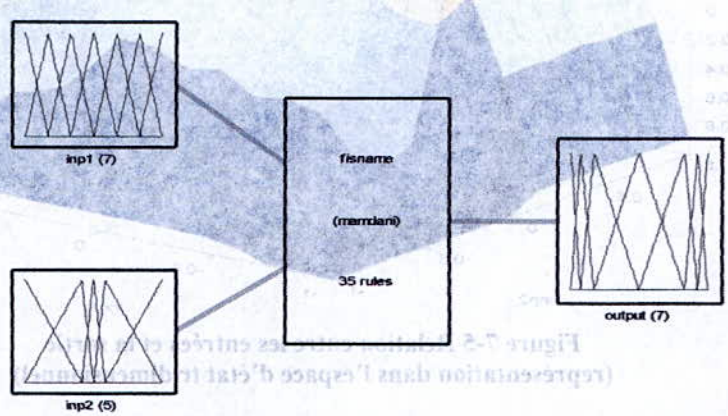


Figure 7-3 Schéma générale du CLF optimisé

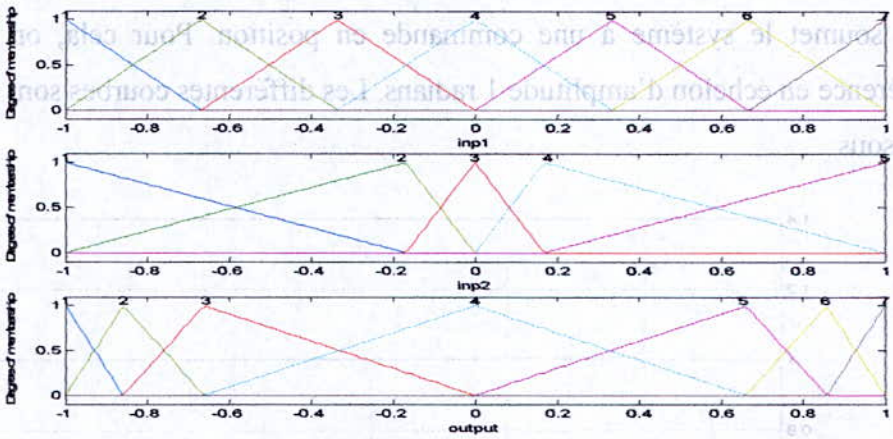


Figure 7-4 Fonctions d'appartenance pour le CLF optimisé

		Variation d'erreur				
		NG	NP	Z	PP	PG
Erreur	NG	NG	NG	NG	NM	NM
	NM	NG	NM	NM	PM	PM
	NP	NM	NM	NP	PM	PM
	Z	NM	NM	Z	PM	PM
	PP	NM	NM	PP	PM	PM
	PM	NM	NM	PM	PM	PG
	PG	PM	PM	PG	PG	PG

Tableau 7-2 Règles d'inférence pour le CLF optimisé

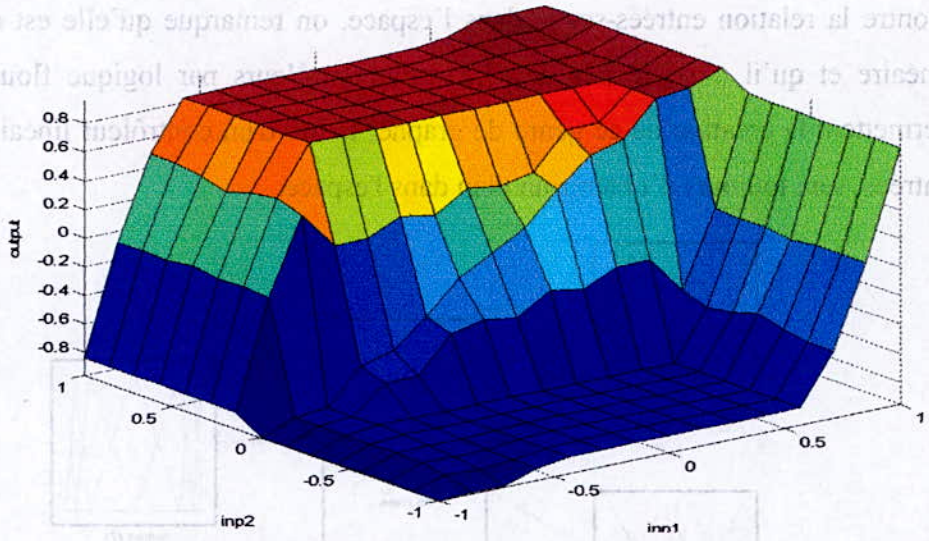


Figure 7-5 Relation entre les entrées et la sortie (représentation dans l'espace d'état tridimensionnel)

7.4. RÉPONSE INDICIELLE

On soumet le système à une commande en position. Pour cela, on choisie une référence en échelon d'amplitude 1 radians. Les différentes courbes sont montrées ci-dessous.

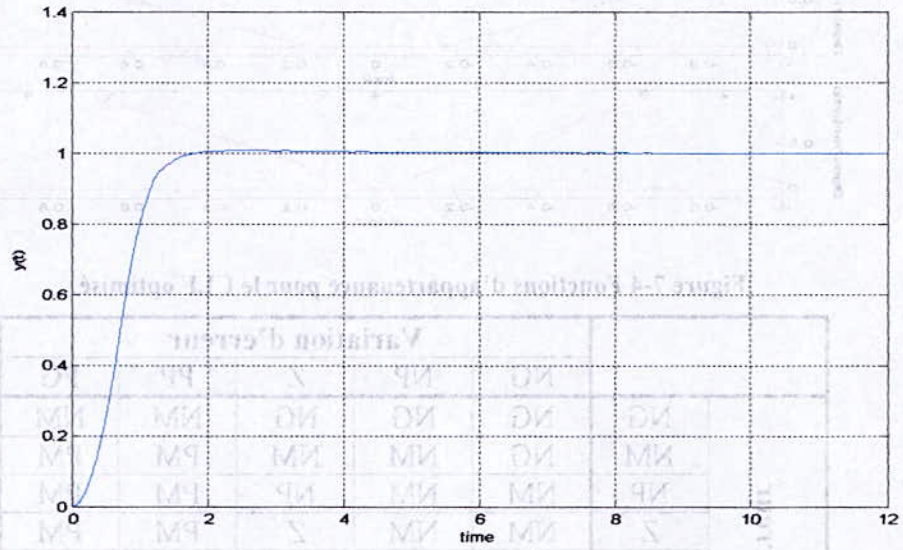


Figure 7-6 Réponse indicielle en position de l'extrémité du bras (CLF)

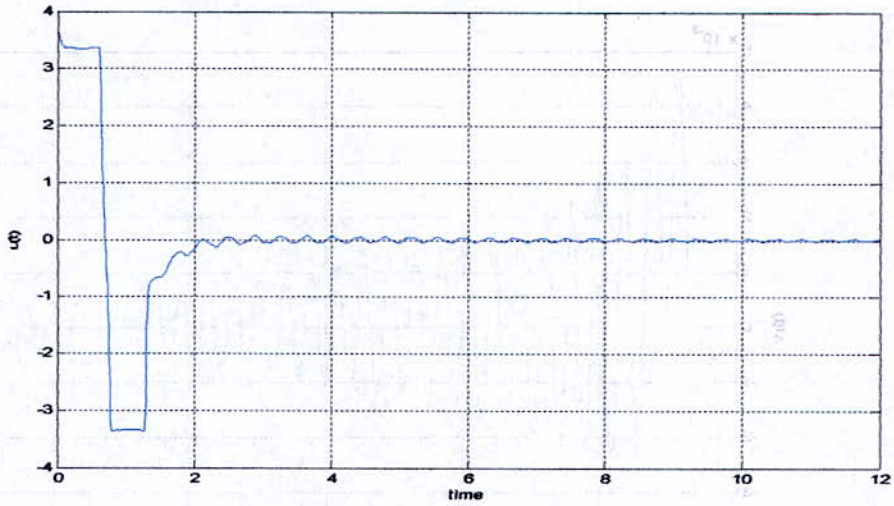


Figure 7-7 Commande générée pour la réponse indicielle (CLF)

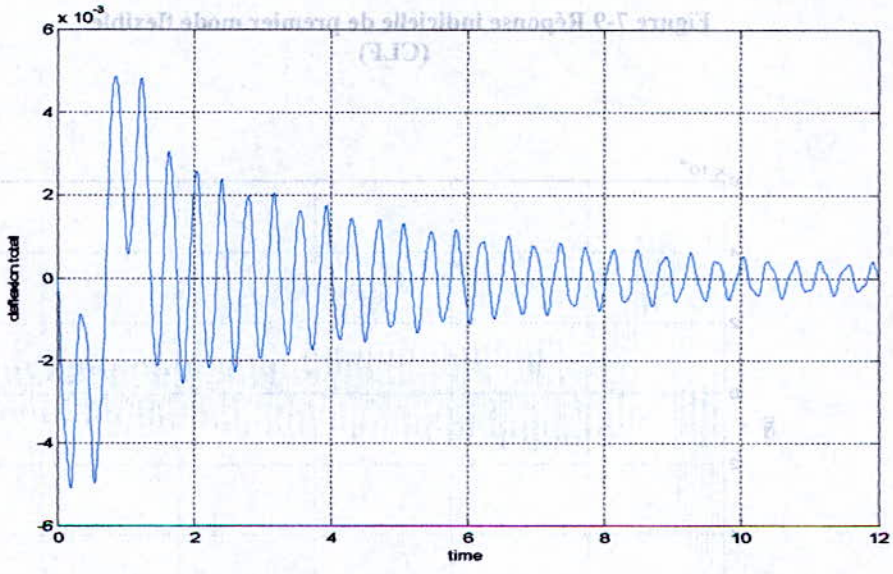


Figure 7-8 Réponse indicielle de déflexion totale (CLF)

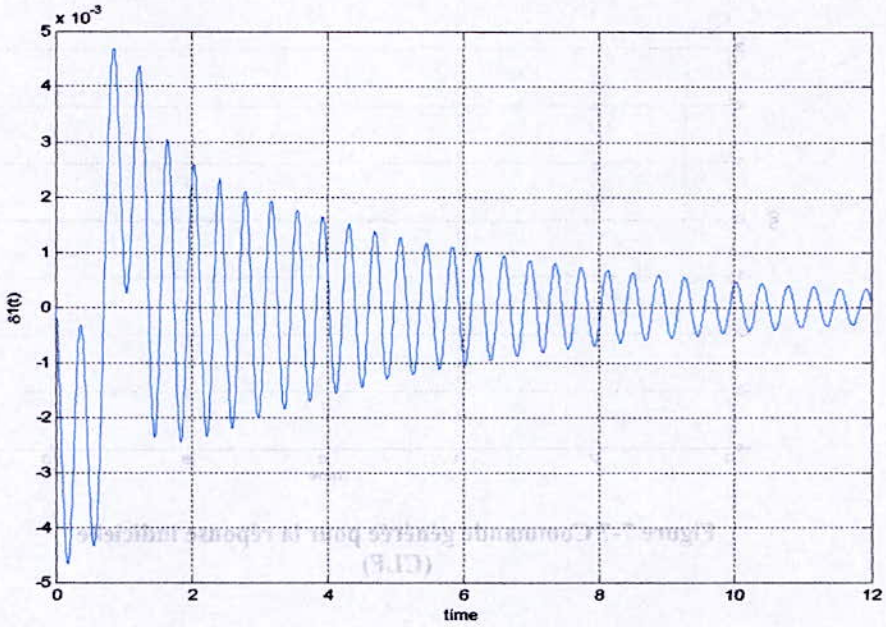


Figure 7-9 Réponse indicielle de premier mode flexible (CLF)

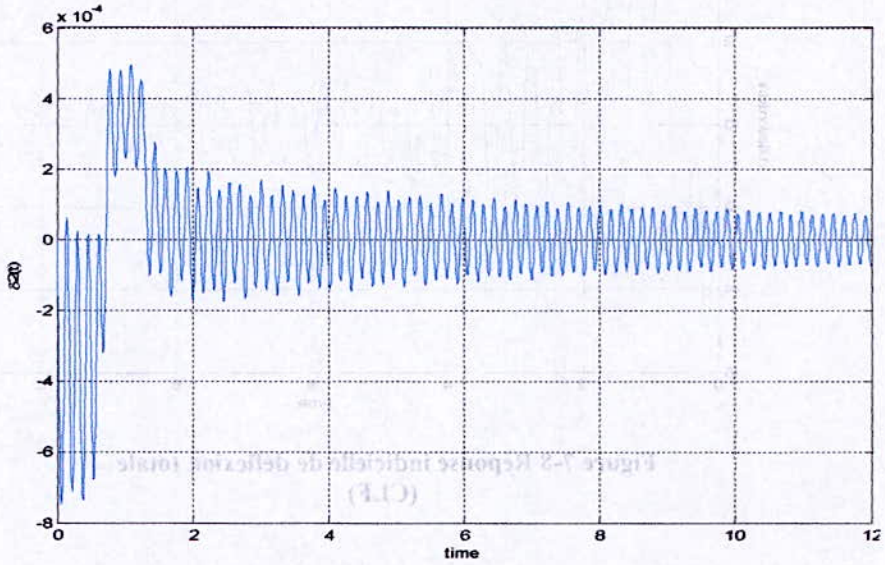


Figure 7-10 Réponse indicielle de deuxième mode flexible (CLF)

7.5. POURSUITE DE TRAJECTOIRE

Le problème de poursuit d'une trajectoire désirée par l'organe terminal est l plus importante difficulté pour les robot flexible

Pour notre application, nous avons sélectionné le trajectoire a (t), en radient, suivant :

$$\alpha(t)=\begin{cases}1 & \text{pour } 0 \le t < 2\\ 2 & \text{pour } 2 \le t < 4\\ 3 & \text{pour } 4 \le t < 6\\ 2 & \text{pour } 6 \le t < 10\end{cases}$$

Les résultas du test sont illustrés par la figure 7-11

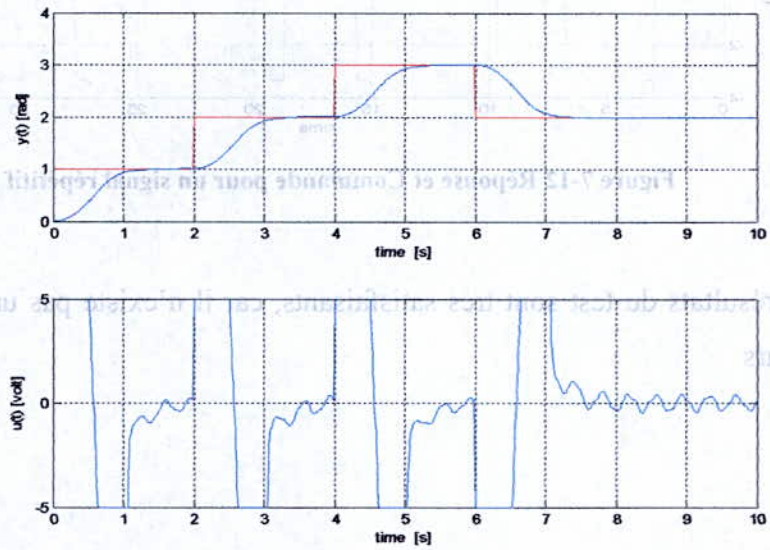


Figure 7-11 Réponse et Commande pour le test poursuite

Les résultats de poursuit sont acceptables, mais en remarque que la commande n'est pas vraiment bonne. L'exécution de cette tâche de poursuite est généralement difficile pour les bras flexibles, nos résultas sont relativement bons.

7.6. TEST DE RÉPÉTITIVITÉ

Il est très intéressant d'avoir un aperçu sur le comportement du système lors de l'exécution des taches répétitives. Pour cela on soumet le système à un signal carré, les résultats sont donnés dans la figure suivante.

Ce test consiste à donner un signal de consigne carré périodique variant entre deux valeurs différentes. Il a pour but de quantifier une éventuelle accumulation d'erreurs statiques lorsque le bras effectue un mouvement de va et vient entre deux positions. Pour notre application, nous avons choisies les position 0 et 1 radian.

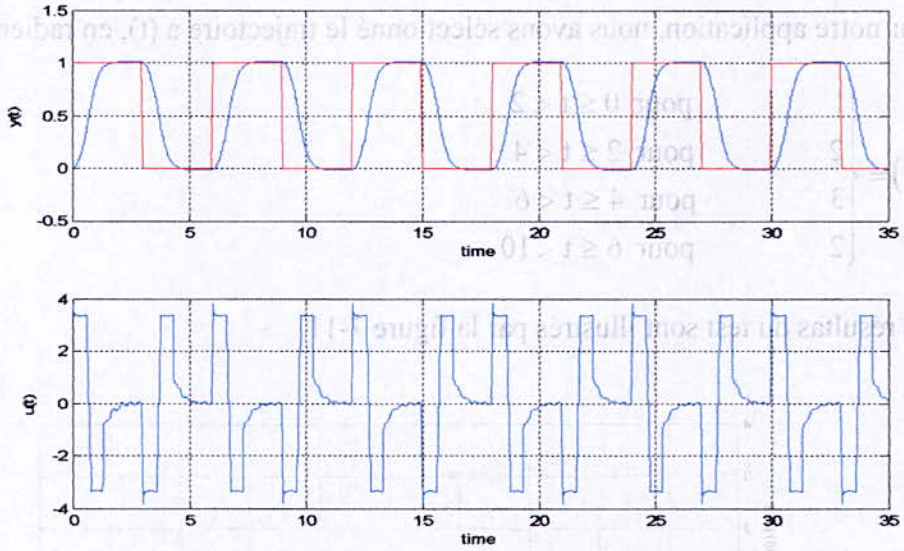


Figure 7-12 Réponse et Commande pour un signal répétitif

Les résultats du test sont très satisfaisants, car il n'existe pas une accumulation des erreurs

Les différents résultats réalisés attestent de la capacité de la stratégie de contrôle adoptée à prendre en charge la commande du système objet de notre étude. Ceci nous amène à affirmer que l'optimisation par algorithmes génétiques avère une méthodologie d'optimisation très efficace et offre un outil puissant pour le système de contrôleurs flous dont les paramètres constitutifs sont fortement empreints de subjectivité.

conclusion générale

8. CONCLUSION GÉNÉRALE

Les travaux exposés, dans ce mémoire, ont porté sur la modélisation, l'analyse et l'application d'une nouvelle technique de commande en position et on poursuite de trajectoire d'un bras manipulateur a une seule liaison flexible. Une large recherche a été effectuée concernent l'application des algorithmes génétiques pour l'optimisation des contrôleurs par logique floue. Les résultats de cet effort aient été entièrement satisfaisants, ils ont pus répandre à nos attentes.

8.1. SUGGESTIONS POUR LES TRAVAUX FUTURS

Ce travail gagnerait beaucoup à être enrichi et étudié selon les suggestions suivantes :

- Réaliser une étude plus approfondie de la façon dont les divers paramètres et codages affectent l'exécution de l'AG dans l'optimisation du CLF. Ce travail étant trop vaste, il faudrait accorder beaucoup plus de temps pour plus d'investigations dans cette partie.
- Étudier l'impact des hypothèses du chapitre 5 sur les contrôleurs, tel que l'emploie des fonctions d'appartenance non triangulaire ainsi que l'utilisation d'un nombre pair de ces ensembles.
- Essayez d'optimiser une partie du code de l'annexe B de sorte que ces algorithmes puissent fonctionner plus vite.
- Appliqué les algorithmes génétique dans le cadre d'une commande en temps réel.

8.2. CONCLUSION

Les algorithmes génétiques se sont avérés comme des outils de recherche puissants, ils peuvent réduire le temps et l'effort impliqués dans les systèmes de conception pour lesquels aucun procédé méthodique de conception n'existe.

Les contrôleurs par logique floue peuvent fournir une commande plus efficace que les contrôleurs linéaires classiques pour les systèmes non linéaires, car il y a plus de flexibilité. La conception traditionnelle des CLF exige la connaissance de l'expert, mais dans notre travail nous avons montré que nous pouvons détourner ce problème, car l'AG que nous proposons est capable de construire un contrôleur efficace d'une façon rapide et sure sans aucune connaissance du système. Cette technique peut mener à une plus large utilisation des CLF, alors que le procédé traditionnel rendait leur utilisation inadéquate.

BIBLIOGRAPHIE

[1] **M. Loudini**
"Modélisation, analyse et méthodologies de commande floue d'un bras de robot flexibles"
Thèse magister LCP-DER de génie électrique, ENP, Alger, 1997.

[2] www.cs.Berkeley.edu/~zadeh
Site perso de professeur Lotfi Zadeh, université de la Californie, Berkeley.

[3] **L. A. Zadeh**
"Soft computing and fuzzy logic,"
IEEE Trans. Software, vol. 11, pp. 48–56, June 1994

[4] **C.C. Lee**
"Fuzzy logic control systems: Fuzzy logic controller- Part I'
IEEE Transaction; on Systems, Man and Cybernetics," Vol. 20, No. 2, pp. 404-418, 1990.

[5] **C.C. Lee**
"Fuzzy logic in control systems: F- logic controller - Part II,"
IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, pp. 419-435, 1990.

[6] **M. Lee, A. Takagi,**
"Integrating Design Stages of Fuzzy Systems Using Genetic Algorithms"
IEEE, 2nd Conference international of fuzzy systems,, San Francisco, 1993; 612-617

[7] **K. Belarbi, F. Titel**
"Genetic algorithm for the design of a class of fuzzy controllers: An alternative approach"
IEEE Transactions on fuzzy Systems, Vol 8, pp 398-405, 2000.

[8] **F. Cheong, R. Lai**
"Constraining the Optimization of a Fuzzy Logic Controller Using an Enhanced Genetic Algorithm"
IEEE Transactions on Systems, Man and Cybernetics -Part B: Cybernetics, Vol 30, No.1, Feb 2000

[9] **D. Naranker**
"Introduction to Genetic Algorithms"
URL: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html

[10] **P. Jennings,**
"Mechanical Engineering in Real Time Computer Systems Fuzzy Logic"
URL: http://www.digital-bauhaus.com/html/fuzzylogic.html

[11] **A. Abreu, L. Custodio C. Pinto-Ferreira,**
"Fuzzy Modelling: a Rule Based Approach"
IEEE 5th Conference international of fuzzy systems, pp. 162-168, 1996.

[12] **Matlab help**
"Fuzzy Logic Toolbox – User Guide for use with Matlab"

[13] **Matlab help**
"Using Matlab Graphics"

[14] **J. Foran**
"Optimisation of a Fuzzy Logic Controller Using Genetic Algorithms"
M.Eng Project Report Summer 2002 DUBLIN city university school of electronic engineering

[15] **A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca**
"Genetic Algorithm Toolb oxFor use with Matlab v5 – User Guide"
Université de Sheffield, Royaume-Uni, 1994.

[16] **A. Chipperfield, P. Fleming , H. Polemic,C. Fonseca**
"Genetic Algorithm Toolbox User's Guide Version 1.2"
Dept of Automatic Control and Systems Engineering, University of Sheffield, 1995

[17] **C. Chen , C Wong**
"Self-generating rule-mapping fuzzy controller design using a genetic algorithm"
IEEE Proc.-Control Theoy Appl. No. 2, pp 149, March 2002

[18] **C. Houck, J. Joines. M.Kay.**
"A genetic algotithm for function optimisation: A Matlab implementation. Mathematical Software, 1996"
http://www.eos.ncsu.edu/eos/service/ie/research/kay_res/GAToolBox/gaot

[19] **H Ishibuchi and T Yamamoto,**
"Fuzzy rule selection by multi-objective genetic local search algorithms & rule evaluation measures in data mining"
IEEE Transactions on Systems Man, Cybernetics

[20] **J. P. Byrne**
"GA-Optimisation of a Fuzzy Logic Controller"
M.Eng Project Report, School of Electronic Engineering, D.C.U.

[21] **K. Deb, A. Pratap, S. Agarwal, T. Meyarivan,**
"A fast and elitist multiobjective genetic algorithm: NSGA-II,"
IEEE Trans. Evol. Comput. vol. 6, pp. 182–197, Apr. 2002.

[22] **X. Zhihong R. Guang**
"Optimization Design of TS-PID Fuzzy Controllers Based on Genetic Algorithms"
IEEE Proceedings of the 5" World Congress on Intelligent Control and Automation, 2004

[23] **D.A. Linkens, H.O. Nyongesa**
"Genetic algorithms for fuzzy control Part 1 : Offline system development and application"
IEEE Proc.-Control Theory Appl., Vol. I42, No. 3, May I995

[24] **Y.J. Park., H.S. Cho, D.H. Cha**
"Genetic Algorithm-Based Optimization of Fuzzy Logic Controller Using Characteristic Parameters"
IEEE Conference international' Evolutionary Computation', 1995, pp 831- 836

[25] **G. Hastings , W. Book**
"Verification of a linear dynamic model for flexible robotic manipulators"
IEEE Conference international of robotics and Automation, April 1986, pp 1024-1029.

A. ANNEXE A

A.1. LES FRÉQUENCES MODALES

La résolution du système ci-dessous exige que le déterminant du système soit nul

$$\begin{cases} A_1\left[(\sin\lambda+\sinh\lambda)+\dfrac{Jc\omega^2}{EI\lambda}(\cos\lambda-\cosh\lambda)\right]+A_2\left[(\cos\lambda+\cosh\lambda)-\dfrac{Jc\omega^2}{EI\lambda}(\sin\lambda+\sinh\lambda)\right]=0 \\ A_1\left[-(\cos\lambda+\cosh\lambda)+\dfrac{Mc\omega^2}{EI\lambda^3}(\sin\lambda-\sinh\lambda)\right]+A_2\left[(\sin\lambda-\sinh\lambda)+\dfrac{Mc\omega^2}{EI\lambda^3}(\cos\lambda-\cosh\lambda)\right]=0 \end{cases}$$

Il existe une infinité de solutions. Les quatre premiers modes sont montrés dans le tableau suivant

Mode	λ	ω
0	0	0
1	1.2479	1.5572
2	4.0271	16.217
3	7.1136	50.6037
4	10.1958	103.9535

A.2. LES FORMES MODALES

$$\begin{aligned} L = \quad & \frac{1}{2}\left[Mc+Jc+J_a+J_b\right]\dot{\theta}^2+\left[Mc\phi_1(1)+Jc\phi_1'(1)+W_1\right]\cdot\dot{\delta}_1(t)\dot{\theta} \\ & +\left[Mc\phi_2(1)+Jc\phi_2'(1)+W_2\right]\cdot\dot{\delta}_2(t)\dot{\theta}+\frac{1}{2}\left[Mc\phi_1^2(1)+Jc\phi_1'^2(1)+Z_1\right]\cdot\dot{\delta}_1^2(t) \\ & +\frac{1}{2}\left[Mc\phi_2^2(1)+Jc\phi_2'^2(1)+Z_2\right]\cdot\dot{\delta}_2^2(t)-\left[\frac{1}{2}\delta_1^2(t)K_1+\frac{1}{2}\delta_2^2(t)K_2\right] \end{aligned}$$

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}_i}\right)-\frac{dL}{dq_i}=\frac{dW}{dq_i}-\frac{dP}{d\dot{q}_i}$$

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}}\right)-\frac{dL}{dq}=\begin{bmatrix}\left[Mc+Jc+J_a+J_b\right]\ddot{\theta}+\left[Mc\phi_1(1)+Jc\phi_1'(1)+W_1\right]\cdot\ddot{\delta}_1(t)+\left[Mc\phi_2(1)+Jc\phi_2'(1)+W_2\right]\cdot\ddot{\delta}_2(t)\\ \left[Mc\phi_1(1)+Jc\phi_1'(1)+W_1\right]\cdot\ddot{\theta}+\left[Mc\phi_1^2(1)+Jc\phi_1'^2(1)+Z_1\right]\cdot\ddot{\delta}_1(t)+\delta_1(t)K_1\\ \left[Mc\phi_2(1)+Jc\phi_2'(1)+W_2\right]\cdot\ddot{\theta}+\frac{1}{2}\left[Mc\phi_2^2(1)+Jc\phi_2'^2(1)+Z_2\right]\cdot\ddot{\delta}_2(t)+\delta_2(t)K_2\end{bmatrix}$$

$$\frac{dW}{dq_i}=\begin{bmatrix}C_m\\0\\0\end{bmatrix}\qquad\frac{dP}{d\dot{q}}=\sum_i c_i.\dot{q}_i(t)=\begin{bmatrix}0&0&0\\0&d1&0\\0&0&d2\end{bmatrix}$$

On trouve l'équation suivante

$$M\ddot{q}+D\dot{q}+Kq=Q$$

Où

$$\mathrm{M}=\begin{bmatrix}\left[Mc+Jc+J_a+J_b\right]&\left[Mc\phi_1(1)+Jc\phi_1'(1)+W_1\right]&\left[Mc\phi_2(1)+Jc\phi_2'(1)+W_2\right]\\ \left[Mc\phi_1(1)+Jc\phi_1'(1)+W_1\right]&\left[Mc\phi_1^2(1)+Jc\phi_1'^2(1)+Z_1\right]&0\\ \left[Mc\phi_2(1)+Jc\phi_2'(1)+W_2\right]&0&\left[Mc\phi_2^2(1)+Jc\phi_2'^2(1)+Z_2\right]\end{bmatrix}$$

$$\mathrm{K}=\begin{bmatrix}0&0&0\\0&K_1&0\\0&0&K_2\end{bmatrix}=\begin{pmatrix}0&0&0\\0&\omega_1^2&0\\0&0&\omega_2^2\end{pmatrix}$$

$$D=\begin{pmatrix}0&0&0\\0&2\zeta_1\omega_1&0\\0&0&2\zeta_2\omega_2\end{pmatrix}\qquad Q=\begin{pmatrix}C_m\\0\\0\end{pmatrix}$$

Les amplitudes des fonctions modales sont arbitraires. Une méthode de normalisation basée sur la simplification de la matrice M, où on pose [1]

$$\begin{cases}Mc\phi_1^2(1)+Jc\phi_1'^2(1)+Z_1=1\\ Mc\phi_2^2(1)+Jc\phi_2'^2(1)+Z_2=1\end{cases}$$

Permet de tirer la valeur de A1et aboutir aux formes modales suivantes

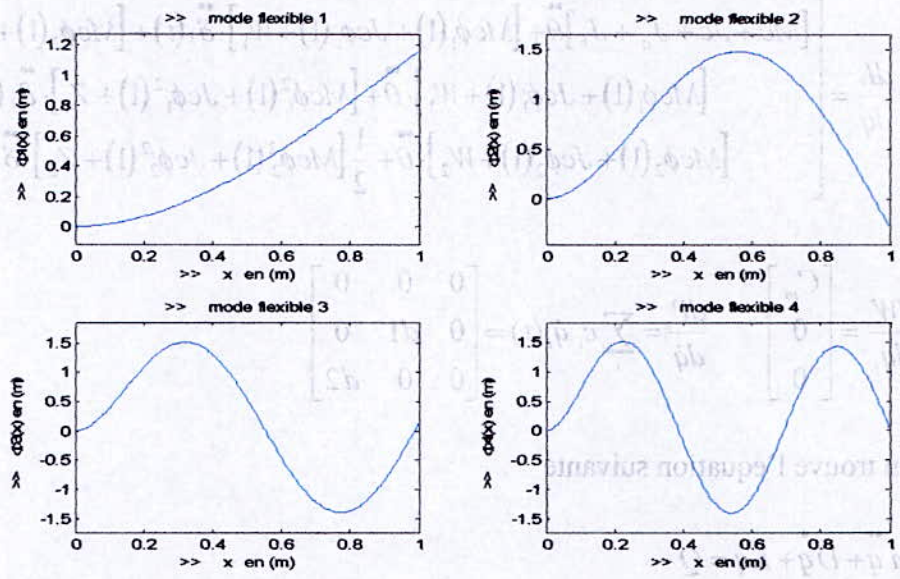


Figure A-0-1 Formes des fonctions modales

A.3. PARAMÈTRES PHYSIQUES DU SYSTÈME

Sous-système	Grandeur	Valeur	unité
Articulation + moteur	J_A	0.4	$Kg.m^2$
Bras	L	1	m
	ρ	1	Kg/m
	EI	1	$N.m^2$
	J_B	1/3	$Kg.m^2$
La charge	M_c	1	Kg
	J_c	$5*10^{-3}$	$Kg.m^2$

B. ANNEXE B

B.1. CROISESIMPLE

```
%*********************************************************

function [fils1,fils2] = CroiseSimple(Parent1,Parent2)

%-------------------------------------------------------------
%croisement simple fait le croisement simple a partir
%de 2 parent aux fils
%on utilisant un point de croisement de valeur aléatoire
%-------------------------------------------------------------

%                       numbre de variables

numVar = size(Parent1,2)-1;

%                       création de point

cPoint = round(rand * (numVar-2)) + 1;

%                       Produisez des fils.

fils1 = [Parent1(1:cPoint) Parent2(cPoint+1:numVar+1)];
fils2 = [Parent2(1:cPoint) Parent1(cPoint+1:numVar+1)];

%*********************************************************
```

B.2. CROISEMASQUE

```
%*************************************************************

function [fils1,fils2] = CroiseMasque(Parent1,Parent2)

%-------------------------------------------------------------
%croisement masque fait le croisement uniforme a partir
%de 2 parent aux fils
%on utilisant un masque de valeur binaire aléatoire
%-------------------------------------------------------------

%                       Obtenez le nombre de variables.

numVar = size(Parent1,2)-1;

%                       Produisez d'un masque et  son complément.

masque1 = round(rand(1,numVar));
masque2 = ~masque1;

%                       Assignez l'espace pour des fils.

fils1 = zeros(1,numVar+1);
fils2 = zeros(1,numVar+1);

%                       Produisez des fils.

fils1(1:numVar) = (Parent1(1:end-1).*masque1+Parent2(1:end-1).*masque2);
fils2(1:numVar) = (Parent1(1:end-1).*masque2+Parent2(1:end-1).*masque1);

%*************************************************************
```

B.3. MUTBINAIRE

```
%*********************************************************
```

```
function [parent] = MutBinaire(parent,ProbMut)

%-------------------------------------------------------
fait la mutation des nombre binaire de façon aléatoire
%-------------------------------------------------------

%              number de variables

numVar = size(parent,2)-1;

%              création d'un vecteur aléatoire et le compare
%              avec la propabilité de mutation

rN=rand(1,numVar)<ProbMut;
parent=[abs(parent(1:numVar) - rN) parent(numVar+1)];

%*******************************************************
```

B.4. STOCREMSELEC

```
%*******************************************************

function newPop = stocRemSelec(oldPop)

%-------------------------------------------------------
%Choix stochastique de reste de STOCREMSELEC une La Goldberg p.123.
%TCet algorithme retourne une population la même taille que l'un po passé.
%newPop    newPop :   La nouvelle population choisie parmi la vieille.
%oldPop    oldPop  :  La population à partir dont la nouvelle population
%                     doit être choisie.
%-------------------------------------------------------

popSize = size(oldPop,1);

 %Espérance de choix.

expec = oldPop(:,end)/mean(oldPop(:,end));

%Partie de nombre entier d'espérance.

intexp = floor(expec);

%Partie partielle d'espérance.

fracexp = expec - intexp;

%Disque de prises des individus choisis.

choice = zeros(popSize,1);

%Disque dont la position doit être remplie.

count = 1;

%parties de nombre entier de llocate.

for j = 1:popSize
   alloc = intexp(j);
   while alloc >0
      choice(count) = j;
      alloc = alloc -1;
      count = count+1;
   end
end

%Assignez les parties partielles.
 %Nombre d'endroits restants

nremain = popSize - count+1;.

while nremain >0
```

```
        %Produisez des nombres aléatoires

    randNums = rand(popSize,1);.

        %Comparez au fract. exp. du choix pour chaque Ind.

    selec = fracexp > randNums;

    numSelec = sum(selec); %Le nombre a choisi ce rond  .

    if numSelec>nremain %If more were selected than places are available
       aaaaa = cumsum(selec); %Ceci donne un billet différent à chaque entrée.
       randNums = rand(numSelec,1);
       [tmp ind] = sort(randNums); %Choisissez les billets de gain au hasard  .
       for i = 1:nremain %Trouvez les gagnants et donne les leur prix  .
          choice(count) = find((aaaaa == ind(i))& (selec(:,1) == 1));
          fracexp(ind(i)) = 0;
          count = count+1;
       end
       nremain = 0;
    else %Autrement assignez les endroits à tout ceux choisis.
       nremain = nremain-numSelec;
       for i = 1:popSize
          if selec(i) == 1
             choice(count) = i;
             count = count+1;
             fracexp(i) = 0;
          end
       end
    end
end

%Brouillez la population choisie  .
[ans, shuf] = sort(rand(popSize,1));
newPop= oldPop(choice(shuf),:);
%*********************************************************
```

B.5. ROULETTE

```
%*********************************************************

function[newPop] = roulette(oldPop)

%---------------------------------------------------------
%roulette is the traditional selection function with kthe
%probability of surviving equal to the fittness of i / sum
% of the fittness of all individuals
%
%function[newPop] = roulette(oldPop,options)
%newPop  - the new population selected from the oldPop
%oldPop  - the current population
%---------------------------------------------------------

%Get the parameters of the population

numVars = size(oldPop,2);
numSols = size(oldPop,1);

%Generate the relative probabilites of selection

totalFit = sum(oldPop(:,numVars));
prob=oldPop(:,numVars) / totalFit;
prob=cumsum(prob);

%Generate random numbers

rNums=sort(rand(numSols,1));

%Select individuals from the oldPop to the new

fitIn=1;newIn=1;
while newIn<=numSols
  if(rNums(newIn)<prob(fitIn))
```

```
    newPop(newIn,:) = oldPop(fitIn,:);
    newIn = newIn+1;
  else
    fitIn = fitIn + 1;
  end
end
%*****************************************************
```

B.6. PROGRAMME

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  le script pour placer des limites et des paramètres        de l'AG          %
%      pour l'optimisation de RLF, pour n'importe quelle systè                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%echo off
%clear

global GainFLOU RLF     %Les variables globales requises
                        %par l'exucution de modele SIMULINK


    %   fonction d'évaluation

FctEvaluation = 'ControlFlou_sim';

    %   Limite inférieure du population.

Limit_Inf = [3 3 3 0.1 0.1 0.1 -1 -1 -1 0.1 0.1 0.1 -1 -1 -1  0    0    0   0 ]';

    %   Limite supérieure du population.

Limit_sup = [9 9 9   1  1   1   1  1  1  1   1   1    1  1  1 2*pi 100 100 100 ]';

    %   les gammes

gammes = [Limit_Inf Limit_sup];

    %   pas de précisions

pas = [2 2 2 0.01 0.01 0.01 2 2 2 0.01 0.01 0.01  2 2 2 pi/512 0.1 0.1 0.1 ]';

    %   Le codage binaire

bits = CalculBits(gammes,pas);

    %   Nombre de génération

ngens = 100;

    %   Nombre d'individus dans une population

PopTaille =20;

    %   epsilon pour deux solutions différentes.

epsilon = 1e-6;

    %   Créez une population initiale aléatoire pour l'AG

   PopInitiale= initialiser_AG(PopTaille,gammes,bits,FctEvaluation);

   %    Fonction d'arrêt
FctStop = 'stop_maxGen';

   %    nom de la fonction de selection  %roulette ou stocRemSelec

Fctchoix = 'stocRemSelec';

   %    om de la fonction de croisement % CroiseMasque ou CroiseSimple

FctCroisement = ['CroiseMasque'];
ProbCrois = 1.0;
```

```
   %     nom de la fonction de mutation %MutBinaire

FctMutation = 'MutBinaire';
ProbMut =0.1 ;

   %     éxécuter AG
[solution PopFinale bpop trace] = Execution_AG (gammes,bits,FctEvaluation, ...
PopInitiale, epsilon, FctStop,ngens,Fctchoix,FctCroisement,ProbCrois,FctMutation,ProbMut);

eff = ControlFlou_sim(solution);

figure
plotfis(RLF)
figure
subplot(3,1,1)
plotmf(RLF,'input',1)
subplot(3,1,2)
plotmf(RLF,'input',2)
subplot(3,1,3)
plotmf(RLF,'output',1)

save latest.mat solution PopFinale bpop trace;

%*******************************************************
```

B.7. CALCULBITS

```
%*************************************************************

function [bits,prec] = CalculBits(gammes,pas)

%-------------------------------------------------------
% CalculBits  donne le nombre de bit sur quelle valeur binaire
% serai codée
%pas           le pas pour parcourir l'intervalle
%gamme  l'intervalle
%bits   nombre de bite
% prec précision entre deux valeurs consécutives
%-------------------------------------------------------

%           calcule de des bits

range = (gammes(:,2) - gammes(:,1))';
bits=floor(log2(range ./ pas'))+1;

%             la precision

prec = range./(2.^bits-1);

%*******************************************************
```

B.8. INITIALISER_AG

```
%*********************************************************************

function [pop] = initialiser_AG(PopTaille, gammes, bits, FctEvaluation)

%---------------------------------------------------------------------
%La production d'une population initiale de valeur aléatoire
%---------------------------------------------------------------

pop = round(rand(PopTaille,sum(bits)+1));
gammes = gammes(:,1:2);

for i=1:PopTaille
    x=b2f(pop(i,:),gammes,bits);
    eval(['v=' FctEvaluation '(x);']);
    pop(i,end) = v;
end
```

```
%*****************************************************************
```

B.9. B2F

```
%*****************************************************************

function [fval] = b2f(bval,gammes,bits)

%-----------------------------------------------------------------
%b2f calcul la conversion binaire real
%bval          la valeur binaire
%gamme         l'intervalle
%bits          nombre de bite
%-----------------------------------------------------------------

%              calcule de  rang de la variable

scale=(gammes(:,2)-gammes(:,1))'./(2.^bits-1);
numV=size(gammes,1);
cs=[0 cumsum(bits)] ;

%                     boucle de conversion

for i=1:numV
  a=bval((cs(i)+1):cs(i+1));
  fval(i)=sum(2.^(size(a,2)-1:-1:0).*a)*scale(i)+gammes(i,1);
end

%*****************************************************************
```

B.10. EXECUTION_AG

```
%*****************************************************************

function [solution,PopFinale,bPop,traceInfo] = Execution_AG(gammes,bits,FctEvaluation,...
        PopInitiale,epsilon,FctStop,ngens,Fctchoix,FctCroisement,ProbCrois,FctMutation,ProbMut)


   gammes = gammes(:,1:2);

%numVars+fittness

   XLongueur  = size(PopInitiale,2);

%Nombre de variables

   numVar       = XLongueur-1;

%Nombre d'individus dans pop.

   PopTaille      = size(PopInitiale,1);

%matrice secondaire de population.

   PopFinale       = zeros(PopTaille,XLongueur);

% La meilleure valeur dans Pop de début.

   BestVal_1         = max(PopInitiale(:,XLongueur));

 %Le nombre de fois mieux a changé.

   bFoundIn     = 1;

%arrêtez avec l'évolution simulée.

   stop         = 0;

%Nombre de génération courant.
```

```
gen          = 1;


figure
while(~stop)

     %%%%%%%%%%%%%%%%%%%%%%%%%Elitisme

     %Mieuxre de POP courant

     [BestVal_2,bindx] = max(PopInitiale(:,XLongueur));.
     best =  PopInitiale(bindx,:);

     %génération courante.

     traceInfo(gen,1)=gen;

     %BLa meilleure qualité.  est fittness

     traceInfo(gen,2)=PopInitiale(bindx,XLongueur);
     traceInfo(gen,3)=mean(PopInitiale(:,XLongueur));
     traceInfo(gen,4)=std(PopInitiale(:,XLongueur));

     hold off
     plot(traceInfo(:,1),traceInfo(:,2),'r')
     hold on
     plot(traceInfo(:,1),traceInfo(:,3),'b')

     %Si nous avons un nouveau  best sol

     if ( (abs(BestVal_2 - BestVal_1)>epsilon) | (gen==1))
         bPop(bFoundIn,:)=[gen b2f(PopInitiale(bindx,1:numVar),gammes,bits)...
                           PopInitiale(bindx,XLongueur)];

     %       Nombre de mise à jour de changements

         bFoundIn=bFoundIn+1;

     %       Mettez à jour les meilleures

         BestVal_1=BestVal_2;

     end


     %%%%%%%%%%%%%%%%%%%%%%%selection

     PopFinale = feval(Fctchoix,PopInitiale);

     %%%%%%%%%%%%%%%%%%%%%%%FIN selection


     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CROISEMENT
     cp=find(rand(PopTaille,1)<ProbCrois==1);

     if~isempty(cp)

         if rem(size(cp,1),2)
                cp=cp(1:(size(cp,1)-1));
         end

            %Produisez d'une liste aléatoire pour choisir que les individus obtiennent
            % appareillés pour le croisement

         list = rand(size(cp));

      %DEVEZ EXAMINER QUE DES INDIVIDUS SONT CHOISIS ALÉATOIREMENT.

         for j=1:size(cp,1)
               [tmp, a] = max(list);
               list(a) = 0;
               [tmp, b] = max(list);
               list(b) = 0;
               a = cp(a) ;
               b = cp(b);
               [PopFinale(a,:) PopFinale(b,:)] = feval(FctCroisement,PopFinale(a,:)
```

```
                                                    ,PopFinale(b,:));
            end
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%FIN CROISEMENT

        %%%%%%%%%%%%%%%%%%%%%%%%%%MUTATION

        for j=1:PopTaille
                PopFinale(j,:) = feval(FctMutation,PopFinale(j,:),ProbMut);
                x=b2f(PopFinale(j,:),gammes,bits);
                eval(['v=' FctEvaluation '(x);']);
                PopFinale(j,end)= v;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%FIN MUTATION

        %Permutez les populations

        gen=gen+1;

        % Gardez la meilleure solution et Remplacez

        PopInitiale=PopFinale;
        [BestVal_2,bindx] = min(PopInitiale(:,XLongueur));

        %le plus mauvais. ellitisme

        PopInitiale(bindx,:) = best;


        %%%%%%%%%%%%%%%%%%%%%%%%%%LE STOP
              %Voyez si l'AG est arrêt.

        stop=feval(FctStop,[gen ngens]);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FIN boucle


    PopFinale = PopInitiale;
    [BestVal_2,bindx] = max(PopInitiale(:,XLongueur));
    x=PopInitiale(bindx,:);
    x=b2f(x,gammes,bits);
    bPop(bFoundIn,:)=[gen b2f(PopInitiale(bindx,1:numVar),gammes,bits)...
            PopInitiale(bindx,XLongueur)];

    solution=x;

    traceInfo(gen,1)=gen;                                   %current generation
    traceInfo(gen,2)=PopInitiale(bindx,XLongueur);          %Best fittness
    traceInfo(gen,3)=mean(PopInitiale(:,XLongueur));        %Avg fittness
    traceInfo(gen,4)=std(PopInitiale(:,XLongueur));         %Std. Deviation
    hold off
    plot(traceInfo(:,1),traceInfo(:,2),'r')
    hold on
    plot(traceInfo(:,1),traceInfo(:,3),'b')

%*****************************************************************
```

B.11. APPARTENANCE

```
%*****************************************************************

function FonctAppartenance  = Appartenance (n_FonAppar,p)

%-----------------------------------------------------------------
%Appartenance calcul Fonction d'Appartenance pour une variable floue.
%Seulement fonction triangulaire peut être créé.
%Les extrémités d'une base d'un triangle coïncidents
%avec les sommets des triangles adjacents.
%
%n_FonAppar   Le nombre de fonctionne d'appartenance
```

```
%p              Un paramètre indique l'espacement des centres.
%               p=1 indique l'espacement égal,
%                p<1 indique comprimé aux extrémités
%               P>1 indique la compression au centre.
%-------------------------------------------------------------------

%               Ordre des fonctions d'appartenance.

n = (n_FonAppar-1)/2;

%                 Calculer l'espacement des centres.

c = zeros(n,1);
for i = 1:n;
   c(i) = (i/n)^p;
end

%               Assigner les positions centrals.

centres = zeros(n_FonAppar,1);

for i = 1:n_FonAppar
   if i < n+1
      centres(i) = -c(n-i+1);
   elseif i == n+1
      centres(i) = 0;
   else
      centres(i) = c(i-n-1);
   end
end

%             Créer les fonctions d'appartenances triangulaires
%                  baser sur ces sommets (les bases sont coïncidents)

pt = zeros(n_FonAppar,3);

for i = 1:n_FonAppar
   if i == 1
      pt(i,1) = 2*centres(i)-centres(i+1);
   else
      pt(i,1) = centres(i-1);

   end
   if i ==n_FonAppar
      pt(i,3) = 2*centres(i)-centres(i-1);
   else
      pt(i,3) = centres(i+1);
   end
   pt(i,2) = centres(i);
end

FonctAppartenance  = zeros(1,n_FonAppar*3);
for i = 1:n_FonAppar
   FonctAppartenance (3*i-2:3*i) = pt(i,:);
end

%************************************************************
```

B.12. REGLEINFERENCE

```
%************************************************************

function Regle = RegleInference(inp,n_FonAppar,p_s,theta_s)

%-----------------------------------------------------------
%RegleInference créent une matrice de règle d'inférence
%
%Regle                  La matrice de règle d'inférence à
%                       employer par un FIS
%inp            Nombre de variables d'entrée
%n_FonAppar     Le nombre de fonctionne d'appartenance
%p_s            Un vecteur des paramètres indiquant
%                       l'espacement desracine et de la grille.
```

```
%theta_s              Indique la pente de la ligne de graine.
%-------------------------------------------------------

%              Ordre des fonctions d'appartenance.

mfOrder = (n_FonAppar-1)/2;
outOrder = mfOrder(end);

%              Positions de point de racine

c = zeros(outOrder,1);
for i = 1:outOrder;
   c(i) = (i/outOrder)^p_s(end);
end

n_out = n_FonAppar(end);

%                    calcule des coordonne des Racine ;
%                    elles doivent être indiquées dans
%                    l'espace dimensionnel xy

co_ords = zeros(inp,n_out);

%Indiquez la x-position des co_ords

for j = 1:n_out
   if j < outOrder+1
      co_ords(1,j) = -c(outOrder-j+1);
   else
      if j == outOrder+1
         co_ords(1,j) = 0;
      else
         co_ords(1,j) = c(j-outOrder-1);
      end
   end
end

%                    multiplication par la ponte

for k = 2:inp
   co_ords(k,:) = co_ords(1,:)*tan(theta_s(k-1));
end

%              Normalisez les cordonné pour être entre - 1 et 1.

norm_fact = max(max(abs(co_ords)));
co_ords = co_ords./norm_fact;

 %             les cordonné des point de la grille

no_rules = prod(n_FonAppar(1:inp));
c = zeros(inp,(max(n_FonAppar(1:inp))-1)/2);
centres = zeros(inp,max(n_FonAppar(1:inp)));
antecedents = zeros(no_rules,inp);

%              Espacez dehors les grille-points dans chaque dimension.

for i = 1:inp
   for j = 1:n_FonAppar(i);
      c(i,j) = (j/((n_FonAppar(i)-1)/2))^p_s(i);
  end
end

for i = 1:inp
   for j = 1:n_FonAppar(i)
      if j < mfOrder(i)+1
         centres(i,j) = -c(i,mfOrder(i)-j+1);
      elseif j == mfOrder(i)+1
         centres(i,j) = 0;
      else
         centres(i,j) = c(i,j-mfOrder(i)-1);
     end

  end
end

%Créez chaque combinaison possible des antécédents.
```

```
for i = 1:no_rules
   for j = 1:inp
      if j == inp
         antecedents(i,j) = mod(i,n_FonAppar(inp));
      else
         antecedents(i,j) = mod(ceil(i/prod(n_FonAppar(j+1:inp))),n_FonAppar(j));
      end

      if antecedents(i,j) ==0
         antecedents(i,j) = n_FonAppar(j);
     end
  end
end


%calcule le la Distance entre  de chaque Racine et point de la grille.

region = zeros(no_rules,n_out);
point = ones(no_rules,inp); %Co-ordinates of each grid-point

%Calculez la distance de chaque grille-point à chaque graine-point.

for i = 1:no_rules
   for j = 1:inp
      point(i,j)=centres(j,antecedents(i,j));
   end
   for j = 1:n_out
      region(i,j) = sum((point(i,:)-co_ords(:,j)').^2);
   end
end

consequent = zeros(no_rules,1);

%Pour assurer la pleine rotation des règles ajustez la région trouvant l'algorithme selon le
régime.

temp = mean(theta_s); %Trouvez l'angle moyen  .
temp = mod(temp,2*pi); %Tracez de nouveau à entre les degrés 0->360  .

%If regime is in left-hand half-plane

if 180*temp/pi>90 & temp*180/pi<=270
   t2 = -1;
else
   t2 =1;
end

flip =1;

%Si le régime est dans le moitié-avion à gauche.

for i = 1:no_rules
   index = find(region(i,:)==min(region(i,:)));
   if size(index) == [1 1]
      consequent(i) = index;
   else %if grid-point is equidistant from two seed-points
      if flip ==1
         consequent(i) = index(1);
      else
         consequent(i) = index(2);
      end
      flip = -flip;
   end

   if t2 == -1 %Trouvez la région dans laquelle chaque grid_pt se situe.

      consequent(i) = n_out+1-consequent(i);
   end
end


Regle = [antecedents consequent ones(no_rules,2)];


%*****************************************
```

B.13. REGULATEURLF

```
%*********************************************************

function RLF = RegulateurLF(inp,out,n,pm,ps,theta_s)

%-------------------------------------------------------
%RegulateurLF  :      Créent un FIS basé sur des paramètres d'entrés.
%
%inp         Nombre de variables d'entrée.
%out            Nombre de variables de rendement.
%n              Vecteur de colonne de la longueur inp+out contenant
%               le nombre de fonctions d'adhésion par variable.
%pm          Vecteur de colonne de la longueur inp+out indiquant comment les fonctions
%            d'adhésion sont écartées pour chacun variable.
%ps          Matrix de la taille inp+1 X out indiquant comment les sont formée.
%theta_s        Matrix de la taille inp-1 X dehors.
%
%-------------------------------------------------------


%%%%%%%%%%%%% Obtenez les paramètres de fonction d'appartenance%%%%%%%%

mfpar = zeros(inp+out,3*max(n));
for i = 1:inp+out
   mfpar(i,1:3*n(i)) = Appartenance(n(i),pm(i));
end

%ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


%%%%%%%%Création de la matrice des Règles d'inférence %%%%%%%%%%%%%%

Regle= RegleInference(inp,[n(1:inp); n(inp+1)],ps(:,1),theta_s(:,1));

%ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


%%%%%%%%%%%%%%%%%%%%%%%%%%%% creation de FIS %%%%%%%%%%%%%%%%%%

RLF = newfis('fisname');
for i = 1:inp
   varname = ['inp' int2str(i)];
   range = [mfpar(i,2) mfpar(i,n(i)*3-1)];
   RLF = addvar(RLF,'input',varname,range);
end
varname = 'output';
range = [mfpar(inp+1,2) mfpar(inp+1,n(inp+1)*3-1)];
RLF = addvar(RLF,'output',varname,range);

%ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

%%%%%%%%%%%%%%%%%%%%%%%%%Au commencement seulement des fonctions%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%d'appartenance doivent être laissées.%%%%%%%%%%

for i=1:inp
   for j = 1:n(i)
      mfname = int2str(j);
      RLF = addmf(RLF,'input',i,mfname,'trimf',mfpar(i,((j*3)-2):(j*3)));
   end
end

for j = 1:n(inp+1)
    mfname = int2str(j);
    RLF = addmf(RLF,'output',1,mfname,'trimf',mfpar(inp+1,((j*3)-2):(j*3)));
End

%ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


RLF = addrule(RLF, Regle);
```

```
%*****************************************************************
```

B.14. CONTROLEURPID_SIM

```
%*************************************************
function  eff = ControlPID_sim(chrom)
%*************************************************

global Gain
 Gain = chrom';
t =12;
sim('syst_pid9',t);

eff = 1/(sum(abs(s_Error))+(10^3)*sum(abs(s_U)));%4;

%*****************************************
```