REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT de GENIE ELECTRIAUE

PROJET DE FIN D'ETUDE

En vue de l'obtention du diplôme d'ingénieur d'Etat en

AUTOMATIQUE

Utilisation des Algorithmes Génétiques en Identification et en Commande des systèmes

Présenté par :

M. HOCINE Abdelfettah
M. MAZOUNI Djalel Eddine

Proposé et dirigé par :

M. D. BOUKHETALA
M. F. BOUJEMA

Promotion juin 2001

Ecole nationale polytechnique
10, avenue Hassan Badi, BP 182 El Harrach Alger
Tel : 021 52 14 98 Fax : 021 52 29 73

Je dédie cet ouvrage

A mes parents, ma famille
A mes frères : Zakaria, Billel, Mohamed
A Kheir Edinne, Nabil et Mohamed
A tous mes amis.

Je dédie cet ouvrage

A ma mère

A mon père

A ma sœur

A la mémoire de mes grands-parents

A tous mes amis.

Abdelfettah

Nous tenon à exprimer nos sincères remerciements et notre profonde gratitude à nos promoteurs monsieur D. Bouketala et F.Boudjema pour l'assistance qu'ils nous ont témoignée tout au long de ce travail. Pour leur confiance, leurs encouragements et pour les conseils qu'ils ont apportés pour l'achèvement de ce projet.

Nous adressons nos sincères remerciements à messieurs les membres de jury qui nous ont fait l'honneur de juger ce travail.

Ainsi à tous ceux qui ont pu m'aider de prés ou de loin à réaliser cet ouvrage

Sommaire

Introduction générale

La physique, la biologie, mais aussi l'économie ou la sociologie sont couramment confrontées au problème classique de l'optimisation. D'une manière générale, une partie importante des développements mathématiques au cours notamment du XVIIIéme Siècle a été consacré à ce sujet

Les méthodes purement analytiques ont largement fait leurs preuves. Elles souffrent toutefois de faiblesse car le monde n'est que très exceptionnellement régi par des fonctions continues et dérivables auxquelles nous sommons habitués.

En automatique, Beaucoup de méthodes de synthèse de commande pour les systèmes physiques peuvent être transformées en un problème d'optimisation. En définissant un critère et en évaluant les performances de la commande selon ce critère.

Habituellement une bonne réponse du système est définie par une fonction continue qui est la somme quadratique de l'erreur entre la réponse réelle et la réponse désirée. Cette optimisation est généralement réalisée avec la *méthode du gradient* mais lorsque le régulateur ou le système sont non linéaires ou lorsque le critère n'est pas convexe les méthodes d'optimisation classique sont confrontées au problème des optimums locaux

D'autres méthodes non déterministes tel que *le recuit simulé, méthode du grimpeur* et *les Algorithmes évolutionnaires* [29], [33] et [36] combinant analyse mathématique et recherche aléatoire ont permis de dépasser cette limitation.

Les Algorithmes Evolutionnaires [35], Développer par John Holland puis par David Goldberg [20], sont des techniques de recherches d'optimums qui incluent la Programmation Evolutionnaire, la Stratégie Evolutionnaire et les Algorithmes Génétiques. Ces méthodes sont basées sur le principe de l'évolution naturelle des espèces, ils suscitent un intérêt considérable et croissant durant cette dernière décennie.

Les Algorithmes Génétiques sont des algorithmes robustes et de recherche globale et peuvent être appliquer sans recours à des informations a priori sur le problème. Pour cette raison et bien d'autre, les spécialistes en automatique ont réalisé rapidement l'étendue du potentiel des Algorithmes Génétiques. Plusieurs travaux ont été faits dans le domaine de l'identification et de l'optimisation des paramètres d'un régulateur particulier ou une structure de commande en mode off-line ou on-line [15], [21] et [22].

Dans notre travail, nous allons présenter les aspects théoriques et pratiques des Algorithmes Génétiques et nous les appliquerons à l'identification et à la commande des systèmes physiques.

Le chapitre I est consacré à la présentation de l'Algorithme Génétique (AG) ou seront énoncé les principes fondamentaux du fonctionnement de l'AG et les différentes étapes de son déroulement. Une étude exhaustive sur les opérateurs génétiques sera réalisée où nous présenterons plusieurs techniques avancés. Certaines applications des AGs en automatique seront présentés à la fin de se chapitre.

Le chapitre II est composé de trois parties. Dans la première partie, nous appliquons les AGs à l'identification paramétrique des processus par les modèles ARX et ARMAX. La deuxième partie sera consacré à l'application de AGs à la réduction d'ordre des modèles mathématiques. La dernière partie traitera l'utilisation des AGs pour la commande par retour d'état appliqué a un pendule inverse.

Dans le chapitre III, nous utiliserons les AGs pour l'optimisation des paramètres une loi de Commande Décentralisée à Structure Variable (CDSV) appliqué à un robot manipulateur

Tout au long de notre travail, des études comparatives entre les différentes techniques seront introduites.

Chapitre I

Algorithme Génétique

I.1 Introduction

En 1859, Charles Darwin, donne le premier l'idée simple d'une évolution naturelle des espèces basée sur un mécanisme de sélection et de reproduction. Depuis, la découverte de l'ADN a permis d'établir qu'à partir de l'information génétique, une évolution naturelle a abouti à l'immense variété des organismes biologiques actuels.

L'observation de la spécialisation des formes de vie montre que le génome d'un organisme peut modéliser l'environnement dans lequel il évolue de manière extrêmement précise. Un exemple cité par l'éthologue V. Dröscher [17] est celui de la chenille de Bombyx qui possède un abdomen dont la forme est celle d'une tête de serpent. Cette apparence lui permet d'effrayer son prédateur, l'oiseau pour lequel le serpent est une menace. Lors de l'évolution qui a abouti à cette espèce de papillon, la sélection a favorisé les chenilles dont la ressemblance, produite par le hasard des mutations et des hybridations, avec un serpent était la plus grande. Sous la simple influence d'une pression de sélection, il y a eu donc un encodage fidèle du modèle de serpent que possède l'oiseau, dans le génome génétique de la chenille. En d'autres termes, la mécanique aveugle de l'évolution naturelle a produit un type de chenille qui constitue une solution efficace au problème posé par le prédateur avien.

Les Algorithmes Génétiques (AG) sont des algorithmes fondés sur les théories darwiniennes (néo-darwinisme). Darwin a montré que l'apparition d'espèces distinctes se fait par le biais de la sélection naturelle de variations individuelles. Cette sélection naturelle est fondée sur la lutte pour la vie, due à une population tendant naturellement à s'étendre mais disposant d'un espace et de ressources finis. Il en résulte que les individus les plus adaptés tendent à survivre plus longtemps et à se reproduire plus aisément. Le terme " adapté " se réfère à l'environnement, que l'on peut définir comme étant l'ensemble des conditions

externes à un individu, ce qui inclut les autres individus. Les lois de variation (croisements et mutations) furent expliquées plus tard par Mendel, puis par la génétique moderne.

Les AGs sont conçus par analogie avec ce processus d'évolution biologique et tirent leur puissance des mêmes mécanismes.

C'est au début des années 60 que John Holland de l'Université du Michigan a commencé à s'intéresser à ce qui allait devenir les algorithmes génétiques. Ses travaux ont trouvé un premier aboutissement en 1975 avec la publication de *Adaptation in Natural and Artificial System.*

Holland poursuivait un double objectif : améliorer la compréhension des processus naturels d'adaptation, et concevoir des systèmes artificiels possédant des propriétés similaires aux systèmes naturels

L'originalité des travaux de Holland repose en particulier sur le fait qu'il n'a pas considéré les seules mutations comme source d'évolution mais aussi et surtout les phénomènes de croisement : c'est en croisant les solutions potentielles existant au sein du pool génétique que l'on peut se rapprocher de l'optimum.

L'idée fondamentale est la suivante : le pool génétique d'une population donnée contient potentiellement la solution ou plutôt une meilleure solution, à un problème adaptatif donné. Cette solution n'est pas exprimée car la combinaison génétique sur laquelle elle repose est dispersée chez plusieurs individus. Ce n'est que par l'association de ces combinaisons génétiques au cours de la reproduction que la solution pourra s'exprimer.

I.2 Formulation du problème d'optimisation

Les problèmes résolus par les algorithmes génétiques appartiennent en fait à la classe des problèmes dit d'optimisation. Nous sommes en effet en présence d'un critère, à optimiser pour des valeurs prises dans un espace de recherche donné. L'étude mathématique du problème d'optimisation est souvent limitée aux fonctions continues et différentiables. Ce n'est pas souvent le cas pour la fonction à optimiser.

En raison de l'analogie avec la théorie de l'évolution (survie des individus les mieux adaptés à leur environnement), l'algorithme génétique est naturellement formulé en terme de maximisation. Etant donné une fonction f réelle a une ou plusieurs variables, le problème d'optimisation sur l'espace de recherche E s'écrit de la manière suivante :

$$\max_{x \in E} \quad f(x) \tag{I.1}$$

De plus, la fonction à optimiser par un algorithme génétique doit avoir des valeurs positives sur l'ensemble du domaine E. Dans le cas contraire, il convient d'ajouter aux valeurs de f une constante positive $F_{\min}$ conformément à l'équation (I.1) .

$$\max_{x \in E} \quad f(x) + F_{\min} \tag{I.2}$$

Dans beaucoup de problème, l'objectif est exprimé sous la forme de minimisation d'une fonction d'évaluation g.

$$\min_{x \in E} \quad g(x) \tag{I.3}$$

Le passage du problème de minimisation à un problème de maximisation est obtenu par transformation de la fonction g. la transformation souvent utilisée est :

$$\max_{x \in E} \quad h(x) \tag{I.4}$$

avec :

$$h(x) = \begin{cases} G_{\max} - g(x) & Si\ G_{\max} \geq g(x) \\ 0 & Sinon \end{cases} \tag{I.5}$$

La fonction h n'est pas unique. En effet, il existe d'autre type de transformation. On rencontre notamment dans la littérature la fonction de transformation h suivante [20]

$$h(x) = \frac{1}{1 + g(x)} \tag{I.6}$$

I.3 Principe de fonctionnement

Les Algorithmes Génétiques sont des algorithmes de recherche de maximum, qui ne font que transposer ce que fait la nature à des systèmes artificiels. Ils simulent le phénomène d'évolution génétique s'appliquant aux chromosomes tel que Darwin les a observés. Donc, pour l'optimisation d'un dispositif, il suffit de faire l'analogie avec la nature : A chaque variable à optimiser x_i (x : Paramètres du dispositif) nous faisons correspondre un *gène*. Chaque dispositif est représenté par un individu doté d'un génotype constitué d'un ou plusieurs chromosomes où chaque chromosome est un ensemble de gène et nous appelons population un ensemble de N individus que nous allons faire évoluer[7], [20] et [24].

Tous les individus de cette population subiront différents opérateurs : sélection, croisement et mutation. Sous l'effet de ces opérateurs la population évoluera et donnera une nouvelle génération qui subira elle-même l'effet des opérateurs ainsi de suite jusqu'à la dernière génération. La sélection consiste à tirer des individus de la population selon la fonction d'évaluation, les individus sélectionnés forment la population intermédiaire. On tire aléatoirement deux individus de la population intermédiaire, en combinant les caractéristiques de ces deux individus on obtient un nouvel individu qui sera placé dans la génération suivante avec ses parents, c'est la reproduction. Les individus de la nouvelle génération subissent l'opérateur mutation qui tire aléatoirement un individu et le modifie aléatoirement.

- **Générer aléatoirement une population d'individu de taille donnée.**
- **Répéter jusqu'à n (nombre de générations)**
  1. **Evaluer les individus.**
  2. **Sélectionner les individus.**
  3. **Croiser deux individus pour créer deux nouveaux individus.**
  4. **Faire muter certains individus.**
  5. **Créer une nouvelle génération qui contient les individus sélectionnés et leurs descendent**

Figure (I.1) structure générale d'un Algorithme Génétique.

I.4 Les points essentiels des Algorithmes Génétiques

Ce qui fait l'attrait des AGs et leur capacité d'accumuler de l'information sur un espace de recherche initialement inconnu est d'exploiter cette information pour guider la recherche future dans des sous-espaces prometteurs [29].

L'AG possède quatre caractéristiques principales :

I.4.1 Le parallélisme

Les AGs travaillent en parallèle sur un certain nombre d'individus (solutions potentielles), et non pas sur un candidat unique. Le parallélisme est essentiel pour le mécanisme de recombinaison (si l'on travaille sur un seul point, on ne pourrait évidement pas définir de recombinaisons) n'oublions pas que la méthode de recherche n'est pas locale mais globale et distribuée sur tout l'espace de recherche. En un certain sens (exprimé par le théorème du schéma que nous allons voir par la suite), la population constitue une base de donnée contacte (mémoire) qui résume toute l'information acquise par la recherche jusqu'à la génération considérer ; le traitement parallèle est aussi très attrayant dans l'optique d'une mise en œuvre informatique sur machine parallèle.

I.4.2 Manipulation d'entités arbitraires

Les AGs manipulent des entités qui ne sont pas forcement numérique. En fait, un AG peut travailler sur n'importe quel espace de recherche, a condition que les points de cet espace de recherche soient toujours constituer d'un ensemble d'entités élémentaire (les gènes ou les caractéristiques) sur les quelles il est possible de définir des opérateurs de mutation et de croisement.

I.4.3 Utilisation minimale d'information à priori

Les AGs ne requiert de l'environnement qu'une mesure d'évaluation (ou de qualité) d'un individu. Ils ne reposent sur aucune autre information (par exemple des dérivés), ni hypothèse tel que la continuité et la différentiabilité de la fonction d'évaluation. En fait, certaines versions des AGs ne requiert de l'environnement qu'une capacité à classer les individus entre eux et rien d'autre.

I.4.4 Balance exploration exploitation

Les phases de sélection et de reproduction sont exécutées en utilisant des règles probabilistes plutôt que des règles déterministes. L'introduction du hasard a, entre autres buts, celui de maintenir les propriétés d'exploration lors de la recherche. Cela est non seulement bénéfique pour l'optimisation de fonction multi-modales (présentant plusieurs optima), mais aussi en cas de fonction non permanente (déplacement ou changement des optima au cours du temps), où le caractère adaptatif d'un algorithme prend alors de l'importance. En outre, certains versions des AGs (les AGs de base, avec sélection proportionnelle) réalisant automatiquement une distribution presque optimale des essais, ce qui signifie que les AGs gèrent le compromis exploration/exploitation d'une façon presque optimale [7].

I.5 La fonction d'évaluation

Le principe de fonctionnement d'un algorithme génétique repose sur la recherche du maximum de la fonction d'évaluation [22]. C'est une fonction qui mesure la qualité d'un individu en tant que solution au problème.

L'algorithme convergera vers un optimum de la fonction d'évaluation, quelle que soit sa définition. La pertinence de la solution dépendra donc de la pertinence de la fonction d'évaluation f qui doit donc traduire en langage mathématique le désir de l'utilisateur.

I.5.1 Objectif unique

Dans le cas d'un objectif unique, la définition de la fonction d'évaluation ne pose généralement pas de problème. Par exemple, si l'on se fixe l'objectif de trouver un dispositif dont le rendement est maximum, cette fonction sera égale au rendement. Le calcul de la fonction d'évaluation se fait en deux étapes. On commence par évaluer les caractéristiques des solutions potentielles en utilisant le modèle. Puis on calcule la fonction d'évaluation à partir de ces caractéristiques.

I.5.2 Objectifs multiples

Les problèmes d'optimisation doivent souvent satisfaire des objectifs multiples [33], dont certains sont concurrents. Une méthode classique consiste à définir des fonctions objectif f_i , traduisant chaque objectif à atteindre, et de les combiner au sein de la fonction d'évaluation. On établit ainsi un compromis. Le plus simple est de se ramener à une somme pondérée des fonctions objectives :

$$f = \sum_i \alpha_i f_i \tag{1.7}$$

Où les poids α_i doivent être tels que la fonction d'évaluation reste bornée dans l'intervalle [0, 1]. Remarquons que certains α_i peuvent être négatifs afin de tenir compte de certaines contraintes du problème. C'est à l'utilisateur de fixer convenablement les poids α_i. On peut souvent classer les objectifs par importance mais les poids seront généralement adaptés par tâtonnement, jusqu'à l'obtention d'une solution acceptable. Le processus d'optimisation a beau être automatisé, l'utilisateur doit donc quand même optimiser " à la main " la définition de la fonction d'adaptation.

A la place d'une somme, on peut également utiliser un produit du type :

$$f = \prod_i f_i^{\beta} \tag{1.8}$$

Il existe également des expressions plus complexes qu'on peut utiliser.

Il faut néanmoins être conscient des effets d'une telle combinaison des objectifs. En effet, deux solutions potentielles dont les fonctions objectif n'ont pas la même valeur peuvent aboutir à une même valeur de la fonction d'évaluation. De plus, un algorithme utilisant une telle approche ne convergera que vers une seule solution alors qu'il existe peut-être toute une famille de solutions remplissant les objectifs fixés. L'optimisation à objectifs multiples est un domaine de recherche très actif actuellement, de par les enjeux économiques et industriels auxquels il répond. Des concepts tels que les niches écologiques semblent prometteuses pour la résolution de ce genre de problème.

I.5.3 Contraintes

En pratique, la plupart des problèmes d'optimisation sont limités par les contraintes [15]. Les AGs peuvent manipuler les contraintes par plusieurs méthodes. La méthode la plus efficace et la plus directe est d'encastrer ces contraintes dans le codage des paramètres. Dans le cas ou il est impossible[22], une fonction de pénalité est utilisée pour assurer que la valeur de la fonction d'évaluation des individus invalide reflète qu'ils ont des basses performances. Toutefois, une fonction de pénalité appropriée n'est pas toujours facile à déterminer pour un problème donné et peut affecter l'efficacité de la recherche. Une autre approche est de considérer les contraintes comme des objectives à atteindre i.e. on se ramène à un problème multiobjectives

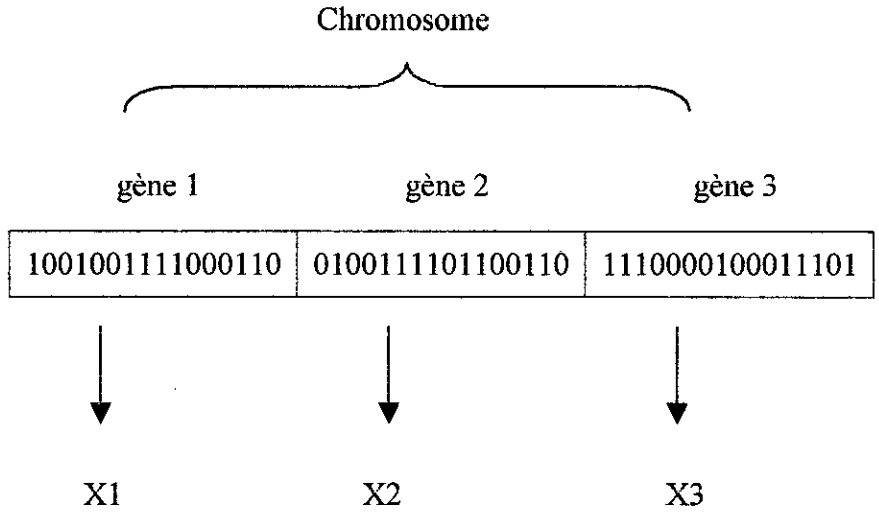
I.6 Codage décodage des variables

Le large domaine d'application de l'approche génétique résulte du fait que toute information, donc toute solution au problème posé au AGs, peut se représenter sous la forme d'une séquence de symboles. Le concepteur doit choisir lui-même le codage des solutions au problème qu'il désire résoudre. Il est ainsi possible d'appliquer un AG à des fonctions demandant plusieurs paramètres. Dans ce cas, il suffit au AGs de gérer une population dont les individus sont des concaténations de caractéristiques simples. Chaque solution du problème peut être représentée sous la forme d'un grand vecteur binaire dans lequel il suffit de décoder chaque paramètre

I.6.1 Représentation binaire des chromosomes

A l'instar de John H. Holland, les chromosomes seront représentés par des chaînes de bits [7]. Le parallèle structurel entre les bits et les bases azotées constituant l'ADN (Adénine, Cytosine, Guanine, Thymine) rend plus aisée la compréhension des transformations telles que le croisement et la mutation génétiques. Cette représentation est indépendante du problème traité et rend l'Algorithme Génétique d'autant plus robuste.

Cependant, nous ne sommes pas habitués à manipuler des nombres binaires. Cette représentation demande donc un effort supplémentaire: l'espace des solutions potentielles doit être transposé dans un espace de solutions binaires en entrée de l'algorithme, et la solution obtenue en sortie doit être reconvertie en une solution réelle afin de pouvoir être interprétée



Figure(I.2) représentation de la structure binaire d'un individu

I.6.1.1 Description du codage

Dans cette partie nous nous limitons au codage de variable réelle, nous considérons un espace de recherche fini

$$x_{i\,min} \le x_i \le x_{i\,max} \qquad \forall i \in [1,n] \tag{I.9}$$

Chaque chaîne de la population est codée sur un nombre fini de bits. La longueur de la chaîne l est fixée par l'utilisateur selon le domaine de variation et la précision demandée.

Pour l'optimisation d'une fonction à plusieurs variable $f(x_1,x_2,\ldots,x_n)$ on utilise un codage binaire concaténé qui consiste à coder chaque variable selon le choix de la longueur et construire les chaînes en concaténant les différents codes.

I.6.1.2 Description du décodage

Chaque chaîne doit être décodée pour pouvoir calculer la valeur de la fonction d'évaluation qui lui est associée.

Pour un codage binaire naturel $b_0,b_1,\ldots,b_{l-1}$ la valeur entière correspondante est donnée par :

$$N(x_i)=\sum_{j=0}^{l-1} b_j 2^j \tag{I.10}$$

le paramètre réel x_i de l'espace de recherche relatif à $N(x_i)$ est obtenu par interpolation linéaire

$$x_i = x_{i\min} + \frac{x_{i\max} - x_{i\min}}{2^l - 1}.N(x_i) \qquad (I.11)$$

la précision des paramètres par ce type de décodage est donnée par :

$$\varepsilon_i = \frac{x_{i\max} - x_{i\min}}{2^l - 1} \qquad (II.12)$$

I.6.2 Représentation réelle des chromosomes

Les nombres binaires sont pour nous bien moins évocateurs que les nombres réels. En outre, des difficultés surviennent pour exprimer la fonction d'évaluation, et traiter les problèmes à plusieurs variables.

Les fonctions d'évaluation conservent souvent leur forme réelle. Les chromosomes binaires sont alors convertis à chaque évaluation.

Les problèmes multi-variables sont ramenés à un problème mono-variable par concaténation des inconnues en un seul chromosome. A chaque évaluation, la chaîne de bits résultante est alors découpée en autant de sous-chaînes qu'il n'y a d'inconnues, ces sous-chaînes sont ensuite converties en nombres réels pour être appliquées à la fonction d'évaluation.

Ces opérations de conversion sont coûteuses en temps-machine, et sont répétées un grand nombre de fois à chaque génération, de plus n'oublions pas la perte de précision lorsqu'on fait la conversion. Nous discernons ici les limites de la représentation binaire.

La représentation réelle propose un compromis intéressant: elle élimine toutes les opérations de conversion, mais en contrepartie, elle rend l'algorithme génétique avancé plus dépendant du problème traité [8]. Dans ce type de représentation, un chromosome est un n-uplet de nombres réels, chacune des composantes correspondant alors à une inconnue du problème.

L'emploi des AGs codée réel semble plus pratique, rapide et précis. Dans ce cas les opérateurs agissent directement sur les variables. Ils sont surtout utilisés pour les larges domaines qui requièrent une représentation trop longue avec les AGs binaire. [29]

I.7 Opérateurs des algorithmes génétiques

I.7.1 Sélection des chromosomes

Une fois réalisée l'évaluation de la génération, on opère une sélection à partir de la fonction d'évaluation. Seuls les individus passant l'épreuve de sélection peuvent accéder à la génération intermédiaire (mating pool en terminologie anglo-saxonne) et s'y reproduire. Il existe plusieurs types de sélection parmi les quels:

I.7.1.1 la roulette de casino

La méthode de sélection naturelle la plus couramment employée pour l'algorithme génétique de base est dite "méthode de la roulette de casino" [20]. Chaque chromosome occupe un secteur de la roulette dont l'angle est proportionnel à sa fonction d'évaluation.

Un chromosome considéré comme bon aura un indice de qualité élevé, ainsi qu'un large secteur de la roulette, en conséquence aura plus de chances d'être sélectionné.

Exemple :

On considère une population de 7 individus avec les probabilités de sélection ps_i tel que leur somme est égale a 1 figure(I.3)

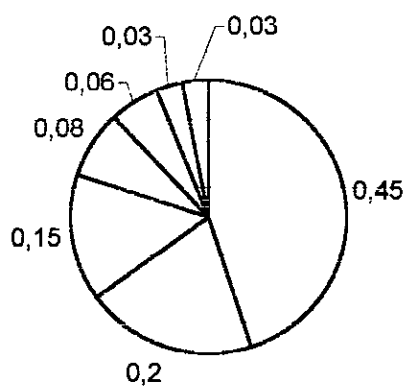


Figure (I.3) : représentation schématique d'une roulette de casino

I.7.1.2 N/2-elitisme

Les individus sont triés selon leur fonction d'évaluation. Seule la moitié supérieure de la population, correspondant aux meilleurs composants, est sélectionnée. Cette méthode peut induire une convergence prématurée de l'algorithme car la pression de sélection est trop forte. Il est en effet nécessaire de maintenir une diversité génétique suffisante dans la population, celle-ci constituant un réservoir de gènes pouvant être utile par la suite. En effet, tout individu peut transmettre à sa descendance des gènes (paramètres de composant) qui, une fois combinés avec d'autres, peuvent se révéler intéressants.

I.7.1.3 Sélection par tournoi

Deux individus sont choisis au hasard et combattent (on compare leurs fonctions d'évalution) pour accéder à la génération intermédiaire. Le plus adapté l'emporte avec une probabilité p tel que : $0.5 < p \leq 1$, que nous avons généralement prise égale à 1 (une valeur inférieure permet de réduire la pression de sélection si nécessaire). Cette étape est répétée jusqu'à ce que la population intermédiaire soit remplie ($N/2$ composants). Il est tout à fait possible que certains individus participent à plusieurs tournois : s'ils gagnent plusieurs fois, ils auront donc droit d'être copiés plusieurs fois dans la population intermédiaire, ce qui favorisera la pérennité de leurs gènes.

I.7.2 Croisement

Le phénomène de croisement est une propriété naturelle de l'ADN. C'est par analogie qu'ont été conçus les opérateurs de croisement dans les AGs.

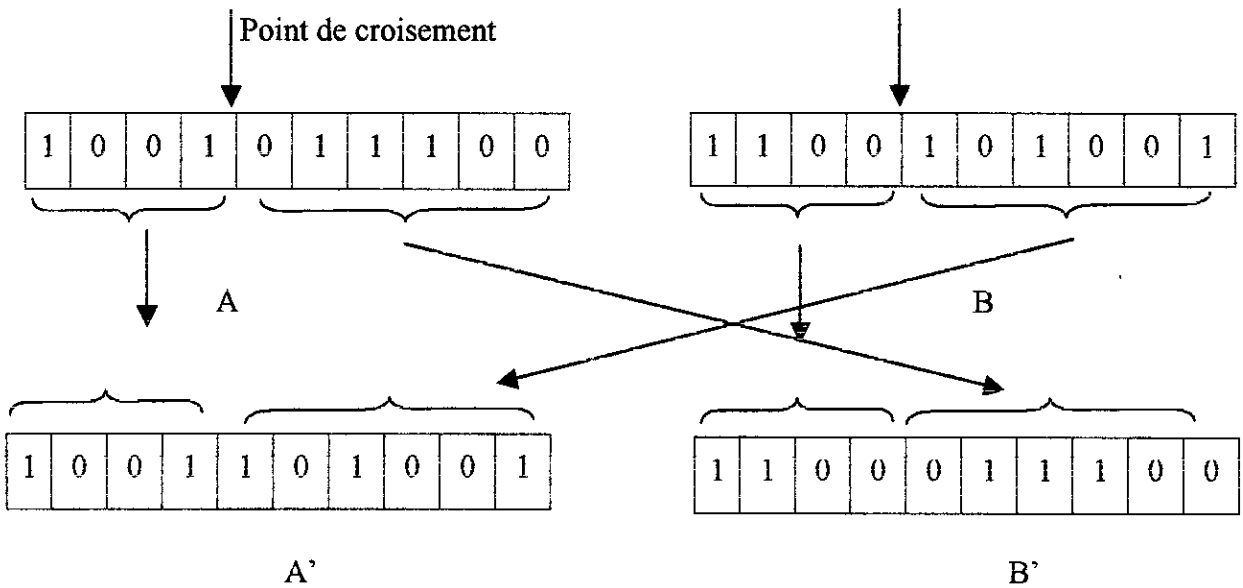
Les individus sélectionnés sont aléatoirement répartis en couples hermaphrodites. Les chromosomes (ensembles de paramètres) des parents sont alors copiés et recombinés de façon à former deux descendants possédant des caractéristiques issues des deux parents. On forme ainsi la génération t+1. L'information génétique globale est préservée, les gènes étant simplement transférés d'un individu à l'autre.

L'opérateur croisement favorise l'exploration de l'espace de recherche. Considérons deux gènes A et B pouvant être améliorés par mutation. Il est peu probable que les deux gènes améliorés A' et B' apparaissent par mutation dans un même individu. Mais l'opérateur de croisement permettra de combiner rapidement A' et B' dans la descendance de deux parents portant chacun un des gènes mutants. Il est alors possible que la présence simultanée des deux

gènes produit un individu encore plus adapté. L'opérateur de croisement assure donc le brassage du matériel génétique et l'accumulation des mutations favorables. En termes plus concrets, cet opérateur permet de créer de nouvelles combinaisons des paramètres des composants.

I.7.2.1 Croisement en un point

On choisit au hasard un point de croisement, pour chaque couple (A, B), les nouveaux chromosomes (A', B') sont crées en recombinant les parties des chromosomes initiaux. Notons que le croisement s'effectue directement au niveau binaire, et non pas au niveau des gènes. Un chromosome peut donc être coupé au milieu d'un gène.



Figure(I.4) : représentation schématique du croisement en 1 point

I.7.2.2 Croisement en deux points

Le croisement en deux points est bâti sur le même principe que celui en un point. Seulement, on choisit au hasard deux points de croisement pour décomposer les chromosomes.

Notons que d'autres formes de croisement existent, le croisement en *k* points jusqu'au cas limite du croisement uniforme.

I.7.2.2 Croisement uniforme

Le croisement uniforme est radicalement différent du croisement en un point. Chaque gène du descendant est crée à partir du gène correspondant de l'un des parents. Les gènes sont choisis selon un vecteur appelé vecteur *masque de croisement* [8]. Lorsqu'il y a un 1 dans le vecteur masque, le gène est copié à partir du premier parent et lorsqu'il y a un 0 le gène est copié à partir du second parent comme il est montré sur la figure (I.5). Le procédé est répété avec d'autres parents pour produire le second descendant, un nouveau vecteur masque est généré aléatoirement pour chaque paire de parents.

Le descendant est une mosaïque de gènes des parents. Le nombre de point de croisement effectif n'est pas fixé.

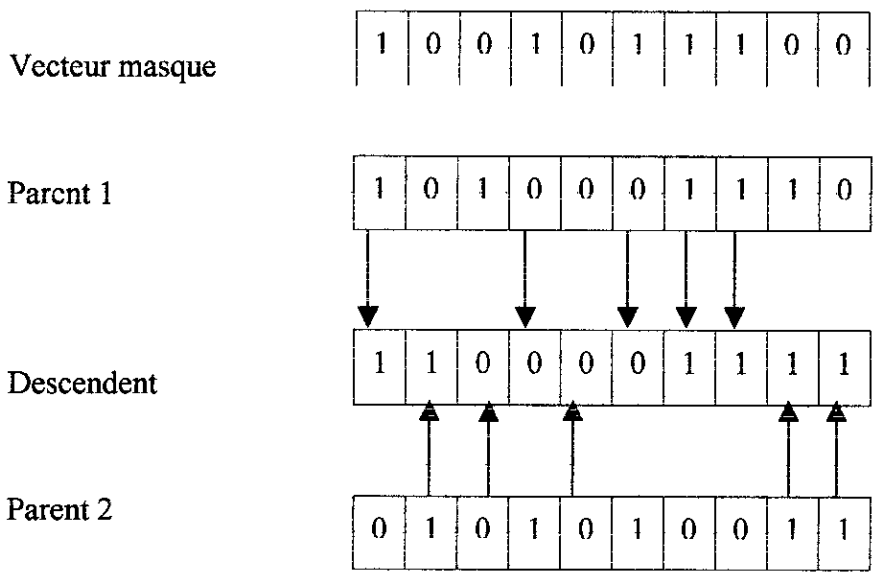


Figure (I.5) schéma représentant le croisement uniforme

I.7.2.3 Croisement arithmétique

Le croisement arithmétique est propre à la représentation réelle[29]. Il s'applique à une paire de chromosomes et se résume à une moyenne pondérée des variables des deux parents

Soient $[a_i, b_i, c_i]$ et $[a_j, b_j, c_j]$ deux parents, et p un poids appartenant à l'intervalle [0, 1] alors les deux descendants sont obtenus par les combinaisons suivantes :

$$[p.a_i+(1-p).a_j,\ p.b_i+(1-p).b_j,\ p.c_i+(1-p).c_j]$$

$$[p.a_j+(1-p).a_i,\ p.b_j+(1-p).b_i,\ p.c_j+(1-p).c_i]$$

Si nous considérons que p est un pourcentage, et que i et j sont nos deux parents, alors le descendant i est constitué à p% du parent i et à (100-p)% du parent j, et réciproquement pour le descendant j.

Notons qu'il est possible de définire autres types de croisement réel suivant la manière du codage et l'espace de recherche. De nombreux exemples de croisement réel ont été proposés dans la littérature des algorithmes génétiques [8] et [29] citons par exemple :

Moyenne : on calcule la moyenne arithmétique des gènes des deux chromosomes parents

Moyenne géométrique : on calcule la racine carrée du produit des deux gènes.

Extension : on calcule la différence entre les deux gènes, et on la rajoute à l'un des parents et on la retranche à l'autre.

Biaisé : on produit un chromosome dans la zone voisine du meilleur des deux chromosomes initiaux.

I.7.3 Mutation

Nous définissons une mutation comme étant l'inversion d'un bit dans un chromosome Cela revient à modifier aléatoirement la valeur d'un paramètre du dispositif. Les mutations jouent le rôle de bruit et empêchent l'évolution de se figer. Elles permettent d'assurer une recherche aussi bien globale que locale, selon les poids et le nombre des bits mutés. De plus, elles garantissent mathématiquement que l'optimum global peut être atteint car une population trop petite peut s'homogénéiser à cause des erreurs stochastiques : les gènes favorisés par le hasard peuvent se répandre au détriment des autres. Cet autre mécanisme de l'évolution, qui existe même en l'absence de sélection, est connu sous le nom de *dérive génétique*. Du point de vue du dispositif, cela signifie que l'on risque alors d'aboutir à des dispositifs qui ne seront pas forcément optimaux. Les mutations permettent de contrebalancer cet effet en introduisant constamment de nouveaux gènes dans la population.

I.7.3.1 Mutation binaire

La mutation binaire s'applique à un seul chromosome. Un bit du chromosome est tiré au hasard avec une probabilité Pm. Sa valeur est alors inversée.

Il existe une variante où plusieurs bits peuvent muter au sein d'un même chromosome. Un test sous le taux de mutation est effectué non plus pour le chromosome mais pour chacun de ses bits: en cas de succès, un nouveau bit tiré au hasard remplace l'ancien.

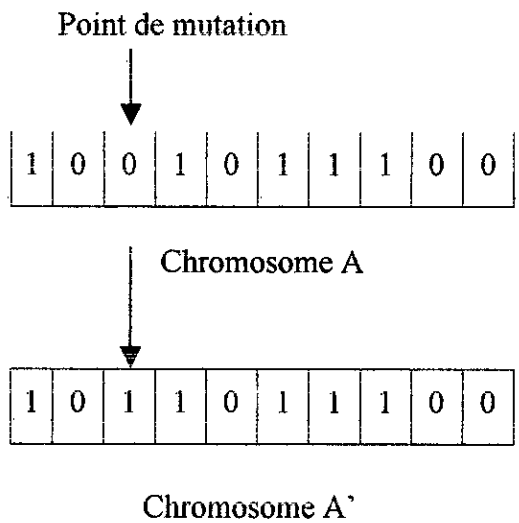


Figure (I. 6) représentation schématique d'une mutation binaire.

I.7.3.2 Mutation réelle

La mutation réelle ne se différencie de la mutation binaire que par la nature de l'élément qu'elle altère: ce n'est plus un bit qui est inversé, mais une variable réelle tirée sur l'intervalle de définition (espace de recherche) qui remplacera l'ancienne valeur. Quelques mutations adaptées pour la représentation réelle sont : [8]

- **Remplacement aléatoire** : la valeur du gène est remplacée par une autre valeur tirée aléatoirement.
- **Creep** : on ajoute une petite valeur aléatoire positive ou négative au gène
- **Creep géométriques** : on multiplie la valeur du gène par une quantité inférieur à 1.

I.7.3.3 Mutation non uniforme

La mutation non uniforme possède la particularité de retirer les éléments qu'elle altère dans un intervalle de définition variable et de plus en plus petit. Plus nous avançons dans les générations, moins la mutation n'écarte les éléments de la zone de convergence. Cette mutation adaptative offre un bon équilibre entre l'exploration du domaine de recherche et un affinement des individus. Le coefficient d'atténuation de l'intervalle est un paramètre de cet opérateur.

I.8 Fondement théorique

I.8.1 Modélisation des Algorithmes Génétiques

Pour étudier de manière formelle les algorithmes génétiques, la première tâche a été de leurs enlever la métaphore biologique qui les entoure. La *théorie des schémas* [7] et la mise en relation des algorithmes génétiques avec les processus stochastiques. C'est la base de la réponse à la question fondamentale de convergence [30].

Modéliser un algorithme génétique est un problème complexe car dans la pratique, les opérateurs de croisement et de mutation, ainsi que les méthodes de sélection, sont très variés. On se limitera dans cette étude au cas des séquences binaires[11] et [30]. De plus, nous nous limiterons au croisement à un point et à la mutation uniforme. L'algorithme génétique ainsi défini est dit *canonique.*

L'essentiel des recherches est basé sur des algorithmes génétiques à séquences binaires pour une raison bien simple. En effet, l'étude des séquences binaires a permis l'élaboration de la Théorie des Schémas. C'est la première étape dans notre processus de modélisation.

I.8.2 Notion de schémas

La Théorie des Schémas fournit un cadre très utile pour déterminer des propriétés intéressantes des algorithmes génétiques canoniques.

Enonçons premièrement quelques définitions nécessaires à la poursuite du raisonnement:

Définition1 :

on appelle Séquence ω de longueur $l(\omega)$ une suite ω tel que

$$\omega = a_1 a_2 \cdots a_l \tag{I.12}$$

$$\text{avec} \quad \forall\ i \in [\![1\ ,l\ ]\!], a_i \in V\ = \{0\ ,1\ \}$$

$$\text{exemple}\ :\ \omega = 100110010$$

Dans la pratique, les séquences de bits utilisées sont de longueur fixée. L'espace de recherche (l'ensemble de toutes les séquences de longueur donnée) est donc de cardinal fini.

Une séquence est donc une suite finie de bits.

Définition2 :

on appelle Schéma H de longueur $l(H)$ une suite H tel que

$$H = a_1 a_2 \cdots a_l \tag{I.13}$$

$$\text{avec} \quad \forall\ i \in [\![1\ ,l\ ]\!], a_i \in V^{+} = \{0\ ,1\ ,*\ \}$$

$$\text{exemple}:\ H = *0**1**10$$

Les bits marqués par * peuvent être 0 ou 1. Le schéma H = 10*1 admet 2 séquences 1011 et 1001. Ces deux séquences sont ses instances.

Définition3 :

On dit qu'une séquence $\omega = a_1 a_2 \cdots a_l$ est une *INSTANCE du Schéma H*,

Si pour tout i tel que $b_i \neq *$, on a : $a_i = b_i$

Exemple : ω = 100110010 est une instance de H = *0**1**10

Définition4 :

On appelle I position fixe(resp position libre) d'un schéma H,

Si a_i = 1 ou a_i = 0 (resp. si a_i = *)

A partir de ces définitions fondamentales, on peut définir l'ordre d'un schéma H, noté $o(H)$, qui correspond au nombre de positions fixes de H

De même, la longueur fondamentale, notée d(H), est égale à la distance entre la première et la dernière position fixe d'un schéma.

On remarquera notamment que le nombre d'instances différentes d'un schéma H est donné par :

$$n\ instances = 2^{l(H)-o(H)} \tag{I.14}$$

On notera $f(\omega)$ l'adaptation (fonction d'évaluation) d'une séquence donnée ω.

Définition5 :

L'adaptation d'un schéma $f(H)$ *est* la moyenne des adaptations de ses instances, elle est donnée par :

$$f(H) = \frac{\sum_{i=1}^{2^{l(H)-o(H)}} f(\omega_i)}{2^{l(H)-o(H)}} \tag{I.15}$$

où les $\omega_{i\ i}$ sont les instances de H.

I.8.3 Effet des opérateurs génétiques

Le formalisme des schémas, va nous permettre d'étudier les effets de la reproduction, des croisements et des mutations. Nous allons pour cela nous appuyer sur le calcul des probabilités. En effet, derrière la métaphore biologique, les opérations de reproduction et les opérateurs de croisement et mutation cachent un modèle probabiliste.

Les générateurs de nombres aléatoires utilisés dans l'implémentation en témoignent. Rappelons aussi que la population initiale est définie aléatoirement.

I.8.3.1 Effet de la sélection.

Intéressons-nous dans un premier temps à l'opération de reproduction ou plutôt de sélection. Les individus dont l'adaptation est la plus forte ont plus de chance de se reproduire. Pour mettre en oeuvre ce système, la plupart des ouvrages font référence à une *Roulette de Casino*. On fait affaire à une variable aléatoire discrète, dite d'alternative généralisée. Ses

réalisations fournissent les individus choisis et sa densité de probabilité est caractérisée par la fonction d'adaptation de chaque séquence.

La probabilité de reproduction d'une séquence ω_i ; de la population est donnée par :

$$p_i=\frac{f(\omega_i)}{\sum_{j=1}^{n} f(\omega_j)} \qquad (I.16)$$

Où n est le nombre d'individus de la population.

Si on note ξ (H,t) le nombre de séquences représentant le schéma H à l'instant t dans la population, on obtient alors à l'instant t+1.

$$\xi(H,t+1)=\xi(H,t).n.\frac{f(H)}{\sum_{j=1}^{n} f(\omega_i)} \qquad (I.17)$$

Cette notation est un abus d'écriture dans le sens où nous ne pouvons connaître que l'espérance mathématique du nombre d'instances de H à l'instant t+1. Cette notation se trouvant dans la plupart des ouvrages, nous la conserverons.

Si on note la moyenne des adaptations des individus de la population par :

$$\bar{f}_{population} =\frac{\sum_{j=1}^{n} f(\omega_i)}{n} \qquad (II.18)$$

et on pose $c_i(H)=\frac{f(H)}{\bar{f}_{population}}-1$, on peut alors écrire :

$$\xi(H,t+1)=(1+c_1(H)).\ \xi(H,,t) \qquad (II.19)$$

Conclusion

Nous avons ainsi prouvé que plus un schéma dispose d'une bonne adaptation $f(H)$, plus le nombre des instances de ce schéma à la génération suivante à de chance d'être important. Au contraire, les schémas faibles ont touts les chances de disparaître.

I.8.3.2 Effet du croisement

Comme nous l'avons dit précédemment, nous nous limiterons au croisement à 1 point. Dans ses conditions, nous allons étudier la probabilité de survie d'un schéma H lors d'un croisement. Il est impossible de déterminer directement la probabilité pour qu'au terme d'un

croisement, une instance de H reste une instance de H. On peut tout au plus minorer cette valeur.

Revenons à la définition de la longueur fondamentale qui donne la longueur entre la première et la dernière position fixe.

Par exemple, H=**100**,donne $\delta(H)$=5-3=2

On remarque que toute instance de H restera une instance de H si le lieu de croisement est inférieur à 2 ou supérieur à 5. Le site de croisement est choisi au hasard selon une loi uniforme. La probabilité de croisement (dans la pratique un taux de croisement fourni par l'utilisateur) est notée p_c. La probabilité p_s de survie d'un schéma H vérifie alors :

$$p_s \geq 1 - p_c \frac{\delta(H)}{l-1} \quad \text{(I.20)}$$

I.8.3.3 Effet de la mutation

Traitons maintenant l'effet de la mutation.

Pour un schéma H donné, seule la modification d'une position fixe par mutation entraîne la destruction du schéma. Avec une probabilité de mutation d'un bit dans une séquence donnée par p_m, la probabilité de survie d'un schéma est :

$$p_{s'} = (1 - p_m)^{o(H)} \quad \text{(I.21)}$$

où o(H) est l'ordre du schéma.

Pour $p_m << 1$, ce qui est le cas en pratique (généralement $p_m < 0.1$), on a alors

$$p_{s'} = 1 - p_m.o(H) \quad \text{(I.22)}$$

où o(H) est l'ordre du schéma.

Finalement, les effets conjugués de la mutation et du croisement nous fournissent alors le résultat final :

$$\xi(H,t+1) \geq \xi(H,t).(1+c_l(H).(1-p_c\frac{\delta(H)}{l-1}-p_m.o(H)) \quad \text{(II.23)}$$

La conséquence générale de cette formule, présentée comme *le Théorème des schémas*, est la suivante:

> ***Th*** *: Les schémas de longueur fondamentale et d'ordre plus petit que les autres sont favorisés lors de la génération d'une nouvelle population.*

Ce résultat théorique est d'un intérêt pratique évident pour qui souhaite améliorer le choix de représentation des individus. En effet, on remarque qu'il est préférable de coder les données les plus adaptées avec des séquences d'ordre et de longueur fondamentale faible. Donc, une connaissance à priori de la forme de la solution doit être prise en compte, comme heuristique, dans le choix de la représentation.

I.9 La convergence prématurée de la population et ses solutions

L'un des problèmes que l'on rencontre fréquemment lors de l'utilisation d'un AG est la convergence prématurée de la population vers un optimum local de l'espace de recherche. La solution à ce problème n'est pas triviale puisqu'il ne se pose que lorsque l'on connaît la forme de l'espace de recherche, c'est à dire que l'on sait que l'on a raté l'optimum global. Dans d'autres cas, il se peut que l'espace de recherche possède plusieurs optima globaux alors que l'AG ne converge que vers l'un d'entre eux, bien qu'il soit souhaitable d'obtenir toutes les solutions possibles au problème que l'on cherche à résoudre.

Il existe différentes recettes qui permettent de limiter l'uniformisation rapide de la population [17].

- Transformation linéaire : linear scaling
- Utilisation de la variance : sigma truncation
- Simulation de niches : crowding
- La fragmentation de la population
- L'unification des individus

- Les procédures de sélection

I.9.1 Transformation linéaire : Linear scaling

Cette transformation vise à diminuer artificiellement l'écart entre les individus en utilisant une équation linéaire du type :

$$f_{transformée}=a.f_{originale}+b \tag{I.24}$$

Où les valeurs de a et b sont choisies de manière à assurer que la fonction d'évaluation originale et la fonction d'évaluation transformée auront la même moyenne ($f_{t\,moy}=f_{o\,moy}$) et à ce que la valeur maximale de la fonction d'évaluation transformée est un multiple (en générale le double) de cette moyenne.

$$f_{t\,moy}=B.f_{o\,moy} \qquad 1 \leq B \leq 2 \tag{I.25}$$

Donc à assurer que les individus dont la valeur est moyenne auront une chance de produire un descendant égale à 1. Toutefois, comme D. Goldberg le fait remarquer[20], cette équation produit des valeurs négatives lorsque la valeur moyenne des individus est proche de la force maximale, ce qui est le cas lorsque la population a convergée vers une solution. Si le remède le plus simple à ce défaut est de transformer toutes les valeurs négatives en 0, il est également possible d'effectuer un prétraitement pour éliminer les valeurs négatives.

Il est également possible de faire d'autres transformations suivant les problèmes rencontrés [23].

Il peut être intéressant de faire une transformation logarithmique de la fonction objective. Par exemple

$$f'(x)=b-\log(f(x)) \tag{I.26}$$

Cette fonction permet de limiter les écarts de la fonction d'évaluation dans les premières générations (limitant ainsi le risque de convergence prématurée). De plus, elle permet d'amplifier les écarts de la fonction d'évaluation entre les individus pour lesquelles $f(x)$ est faible (que l'on retrouve en grand nombre lorsque l'algorithme commence à converger).

Une autre possibilité, lorsque l'on cherche à maximiser $f(x)$ est d'utiliser une transformation par élévation à une puissance donnée (power laws mapping)

$$f'(x) = [a.f(x) + b]^{\gamma} \quad (I.27)$$

La valeur de γ dépend du problème à traiter. Il peut être intéressant de faire varier le paramètre γ durant l'exécution de l'algorithme génétique de manière à adapter les écarts de valeurs sélectives comme désiré

I.9.2 Utilisation de la variance : Sigma truncation

Cette technique, introduite par S. Forrest consiste à transformer la force d'un individu par :

$$f' = f - (\bar{f} - c \cdot \sigma) \quad (I.28)$$

Où c est choisie comme un multiple (entre 1 et 3) de l'écart type σ de la population. Dans cette technique les valeurs négatives sont automatiquement mises à zéro.

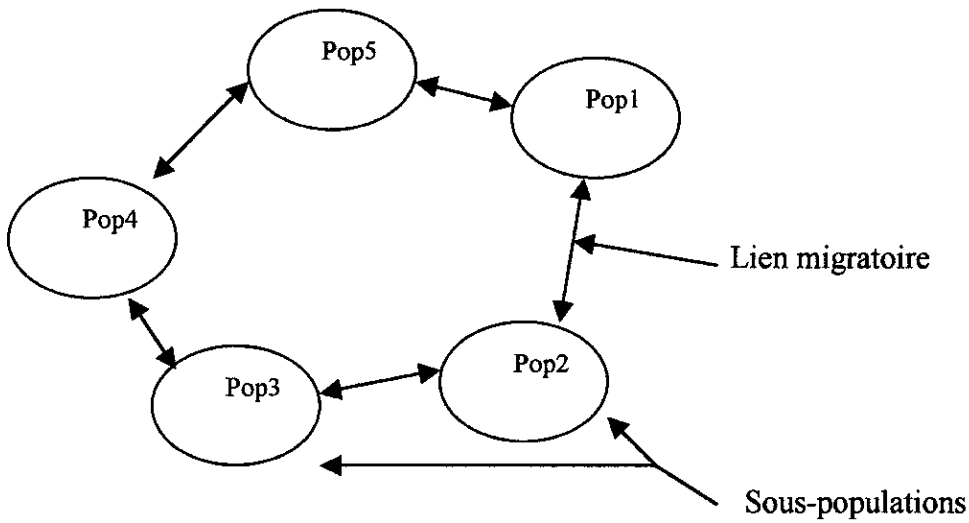
I.9.3 Simulation de niches : Crowding

Dans la nature, lorsqu'une espèce occupe une niche écologique donnée, elle n'entre pas directement en compétition avec une autre espèce dédiée à une tout autre niche. Or dans l'AG de base que nous avons décrit, il n'y a pas réellement de possibilité de création d'espèces puisqu'un individu (qui correspond à une suite de caractéristique spécifique) peut être remplacé, lors de la phase de reproduction, par le descendant d'un individu fort différent. La simulation de niche permet de simuler l'effet d'une compétition à l'intérieur de sous-populations de la population globale de l'AG. Lors de la sélection des individus destinés à être éliminés, il suffit de sélectionner aléatoirement une petite sous population (contenant traditionnellement deux ou trois individus) et de choisir dans cette population l'individu qui ressemble le plus à l'individu nouvellement créé.

I.9.4 La fragmentation de la population

Il est également possible d'utiliser un ensemble de petites populations au lieu d'une population unique. Dans cette optique, si l'une de ces populations converge prématurément, cela n'affecte en rien la recherche effectuée par les autres populations. De plus, nous pouvons faire migrer d'une population à l'autre les individus localement performants. Bien évidemment, la fréquence de ces migrations doit être suffisamment faible pour éviter de faire converger prématurément l'ensemble des populations vers un même type d'individu. Comme

le montre la figure(II.6), chaque population utilise la même fonction d'évaluation et produit, si l'espace de recherche le permet, une espèce d'individu spécifique.



Figure(I.6) Un AG réparti en sous-populations.

I.9.5 L'unification des individus

Etant donné que pour certains problèmes la fonction d'évaluation est la partie de l'AG la plus coûteuse en temps de traitement, l'unification des individus est un nouveau type d'AG : à chaque cycle, deux individus sont choisis pour la reproduction. Les opérateurs de croisement et de mutation sont identiques à ceux de l'AG classique, mais les descendants nouvellement créés ne sont introduits dans la population que s'ils diffèrent d'individus déjà existants dans la population. Si cette méthode permet d'éviter l'évaluation d'individus dupliqués, elle préserve également la diversité génétique de la population.

I.9.6 Les procédures de sélection

Lors de la phase de reproduction de la population, la procédure de sélection classique consiste à utiliser la valeur d'évaluation d'un individu pour lui attribuer une probabilité de sélection. Afin de réduire la vitesse de convergence de la population, une première solution consiste à sélectionner un premier parent en fonction de sa fonction d'évaluation et à choisir l'autre parent de manière totalement aléatoire. Toutefois, le choix d'une procédure de sélection dépend du problème auquel l'AG est confronté. Dans le cas où le problème serait simple, en l'absence de maximums locaux, une convergence rapide est avantageuse alors que dans le cas inverse une convergence lente est préférable.

Il existe d'autre procédure de sélection [2] et [4]. On peut citer :

- Sélection pure ou Elitiste
    - Pure : C'est une technique qui impose une durée de vie d'une génération seulement pour chaque individu. i.e. les parents sont capables de reproduire des nouveaux individus pendant une seule génération. Ensuite, ils meurent et ils seront remplacer par leurs descendants.
    - Elitiste : les parents, ou seulement une partie, peuvent participer à la sélection avec leurs descendants. Dans ce cas, on peut avoir une vie illimitée de quelques individus.
- Sélection générationnel ou steady-stat
    - Générationnel : L'ensemble de parents sera fixé et les membres de la nouvelle génération seront reproduit uniquement par cet ensemble.
    - Steady-stat : Un descendent remplace directement un des parents si ses performances son meilleures. Ainsi, l'ensemble des parents peut être changer à chaque étape de reproduction.

Remarque

On peut faire des combinaisons avec les techniques précédentes pour générer des nouvelles formes de sélection.

I.10 Contrôle des paramètres de l'Algorithme Génétique

Nous touchons là au délicat problème du réglage des paramètres de l'algorithme. Celui-ci doit être optimisé pour chaque type de problème traité, ce qui constitue une part importante du travail de l'utilisateur. Les caractéristiques de l'algorithme doivent être adaptées à la topologie du paysage dessiné par la fonction d'évaluation. L'ensemble problème-méthodes-paramètres constitue donc un tout. En témoigne certaines études où les paramètres d'un Algorithme Génétique sont réglés et optimisés par un autre Algorithme Génétique [20]. Dans la pratique, les méthodes et paramètres des AGs sont tout d'abord réglés approximativement par tâtonnement avec des fonctions de n variables couramment utilisées pour tester les

algorithmes d'optimisation. Le temps de calcul de ces fonctions étant minime, on peut ainsi régler rapidement les paramètres.

I.10.1 Taille de la population

Une population trop petite évoluera probablement vers un optimum local peu intéressant. Une population trop grande sera inutile car le temps de convergence sera excessif. La taille de la population doit être choisie de façon à réaliser un bon compromis entre temps de calcul et qualité du résultat [1].

Il faut être conscient que la taille de population dépend de la puissance de calcul dont on dispose, des méthodes utilisées (sélection, opérateurs génétiques...), du nombre de variables considérées et de la fonction d'évaluation. Si la fonction à optimiser comporte peu d'optima locaux et un optimum global net, la population nécessaire sera plus petite que dans le cas d'une fonction beaucoup plus compliquée comportant de nombreux optima locaux.

I.10.2 Taux de sélection

Le mécanisme de sélection dans un AG joue un rôle très important. D'une part il conduit la recherche vers les solutions optimales et d'autre part, il garantit la diversité génotype de la génération. C'est une relation directe entre la vitesse de convergence et la probabilité de trouver l'optimum global, qui est l'un des plus grands problèmes dans l'optimisation par algorithme génétique.

Plusieurs techniques ont été introduite pour le choix du taux de sélection [5]. Pour décrire ces techniques par des relations formelles on utilise les notations suivantes :

$$p^t = (a_1^t, \ldots, a_\lambda^t) \in I^\lambda \tag{I.29}$$

Avec P^t : population à la génération numéro t,

λ : la taille de population.

I^λ : l'espace des individus a_i^t

On définit aussi le rang d'un individu dans le cas d'un problème de maximisation par :

$$\forall i \in \{1, \ldots, \lambda\} : rang\ (a_i^t) = i \Leftrightarrow \forall j \in \{1, \ldots, \lambda - 1\}, f(a_j^t) \geq f(a_{j+1}^t) \tag{I.30}$$

$f(a_i^t)$: fonction d'évaluation

Par conséquent on peut utiliser l'indice i comme le rang de l'individu et l'élément a_1^t est le meilleur individu de P^t

$P_s(a_i^t)$ est la probabilité de sélection associer à l'individu i

Sélection proportionnelle

C'est la méthode la plus utiliser dans les AGs standards, la probabilité de sélection est donnée par :

$$ps\ (a_i^t) = \frac{f(a_i^t)}{\sum_{j=1}^{\lambda} f(a_j^t)} \qquad (I.31)$$

Sélection par classement linéaire

$$ps\ (a_i^t) = \frac{1}{\lambda}\left(\eta\ \max - (\eta\ \max - \eta\ \min)\frac{i-1}{\lambda-1}\right) \qquad (I.32)$$

avec $\eta\ \min = 2 - \eta\ \max$ et $1 \leq \eta\ \max \leq 2$

Sélection par classement uniforme

$$ps\ (a_i^t) = \begin{cases} 1/\mu & , 1 \leq i \leq \mu \\ 0 & , \mu < i \leq \lambda \end{cases} \qquad (I.33)$$

Remarque :

Dans la première relation, la probabilité de sélection dépend de la fonction d'évaluation alors que dans les deux dernières elle dépend du rang i

les techniques de sélection peuvent être classer suivant les critères :

- Sélection dynamique ou statique
    - Dynamique : la probabilité de sélection dépend de la valeur de la fonction d'évaluation pour la sélection proportionnelle par conséquent elle change au cours des générations.

- Statique : elle dépend uniquement du rang de la fitness qui va donner une valeur fixe pendant toutes les générations.

- Sélection extinctive ou préservative

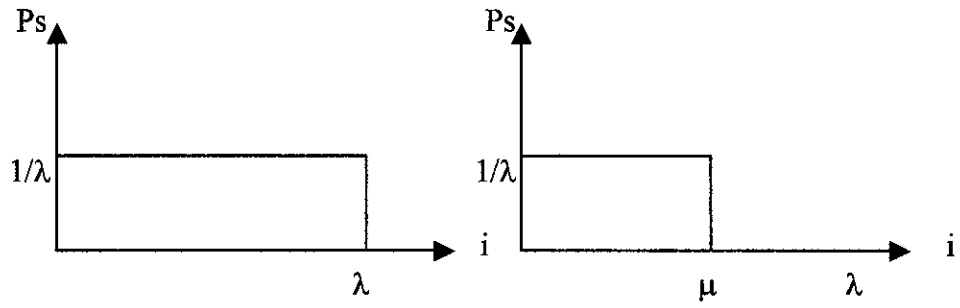
  - Préservative : C'est une sélection qui garantie une probabilité de sélection non nulle pour chaque individu, c'est à dire que tout individu a une chance de contribuer à la création des nouveaux descendants de la prochaine génération.

  - Extinctive : contrairement a la première, quelques individus n'ont pas le droit de participer à la création des nouveaux descendants, i.e. ils ont une probabilité de sélection nulle. Dans la majorité des cas se sont les plus mauvais individus. Elle est appelée sélection extinctive à droite. Bien que dans des cas spéciaux, les meilleurs individus sont prévus de la reproduction pour éviter la convergence prématurée. Dans ce cas, elle est appelée sélection extinctive à gauche.

Remarque :

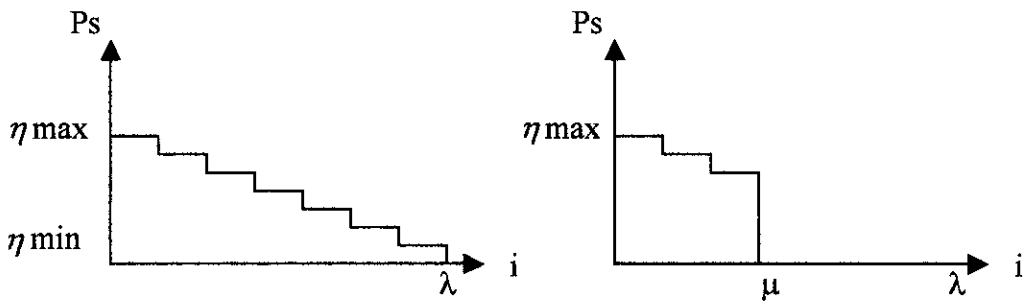
Bien sur, dans tous les cas précédents la relation $\sum_{i=1}^{\lambda} ps(a_i)=1$ doit être vérifiée.

En regardant ces classifications, on peut dire que la sélection proportionnelle est une sélection dynamique préservative, la sélection linéaire est une sélection statique préservative et la sélection uniforme est une sélection statique extinctive.



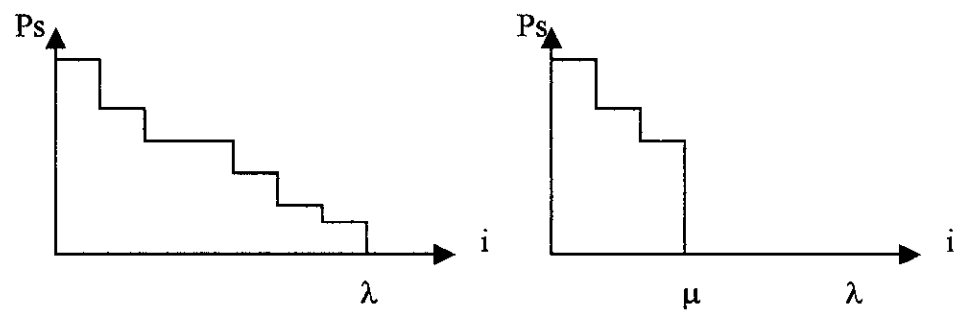
a) Preservative b) Extinctive

1) Sélection statique uniforme



a) Preservative b) Extinctive

2) Sélection statique linéaire



a) Preservative b) Extinctive

3) Sélection dynamique proportionnelle

Figure(I.7) : probabilité de sélection suivant plusieurs techniques

I.10.3 Taux de mutation

Plusieurs travaux expérimentaux ont été faits pour déterminer la meilleure probabilité de mutation (pm) d'un algorithme génétique [1], [2] et [5]. Ces études permettent de trouver

des moyens pour augmenter le succès de l'opérateur mutation qui va accélérer la vitesse de convergence et d'autre part, il y aura une grande chance d'éviter les optimums locaux.

On distingue trois grande classe de mutation :

Probabilité de mutation constante

En générale elle est choisie préalablement tel que $0.005 < Pm < 0.01$, ou par l'une des équations :

$$pm = \frac{1.75}{\lambda . \sqrt{L}} \quad , \quad pm = \frac{1}{L} \tag{I.34}$$

Avec λ : la taille de la population

L : longueur d'individu

Cette probabilité reste constante pondant toutes les générations et elle est applicable à tous les individus

Probabilité de mutation variable

Les probabilités de mutation variables sont proposées pour la première fois par Holland [5], mais il n'a pas détaillé comment le taux de mutation diminue suivant les générations. Plus tard, plusieurs lois ont été données [3] et qui ont permit d'augmenter les performances de l'algorithme génétique. Elles consistent à diminuer le taux de mutation au cours des générations jusqu'à pm = 0.

Par exemple : à la génération t on a :

$$pm\ (t) = \sqrt{\frac{a}{b}}\ \frac{\exp\left(\frac{-gt}{2}\right)}{\lambda\sqrt{L}} \tag{I.35}$$

ou a, b et g sont des constantes

Probabilité de mutation modulable

Autres approches consistent à moduler le taux de mutation au cours des générations, i.e. la qualité des individus générer par la mutation est utilisée pour adapter le taux de mutation comme suit :

Si les individus sont améliorés par l'opérateur mutation, alors le taux de mutation est augmenté sinon il est diminué.

Remarque :

Plusieurs recherches ont démontré que l'utilisation d'une mutation adaptative participe à la convergence de l'algorithme.

La sélection, le croisement et la mutation sont liés entre eux. Alors que la mutation permet l'apparition de nouveaux gènes par la modification d'un chromosome, le croisement diffuse les gènes existants lors du renouvellement de la population. Cette balance entre l'exploration par mutation et l'exploitation par croisement et sélection doit être soigneusement respectée car un taux de mutation trop élevé entraîne la destruction de gènes avant qu'ils n'aient eu la chance d'être assemblés par croisement afin de former des structures valables. De l'autre côté, si le taux de sélection excessif, la population sera uniformisée trop rapidement et la population convergera alors prématurément vers un type d'individu probablement sous-optimal.

I.11 Les Algorithmes Génétiques et le calcul parallèle

Dans un AG classique, le calcul de la fonction d'évaluation d'un individu ne dépend que des caractéristiques contenues par celui-ci. En conséquence, il est tout à fait raisonnable d'envisager un calcul parallèle [15]. Enfin, étant donné que selon J. Holland [11], un AG traite un nombre de schéma proportionnel au cube de la taille de la population, il apparaît souhaitable de doter cet AG d'une très grande population. Toutefois, lorsque la taille de la population est élevée le temps de convergence de l'AG, lorsqu'il est traité par une machine séquentielle, augmente au-delà du raisonnable et il devient nécessaire de paralléliser ce calcul.

Nous présentons deux approches qui utilisent des architectures parallèles [17] fort différentes:

- Un AG pour une machine SIMD (Single Instruction, Multiple Data)
- Le calcul sur machine MIMD (Multiple Instruction Multiple Data)

I.11.1 Un Algorithme Génétique pour une machine SIMD

Une machine (Single Instruction, Multiple Data) SIMD est constituée d'un ensemble de processeurs élémentaires (ou PEs, *Processing Elements* en anglais) interconnectés qui exécutent simultanément les instructions envoyées par un contrôleur séquentiel.

Piet Spiessens et Bernard Manderick [15] ont implémenté un AG sur une machine massivement parallèle SIMD, le DAP (Distributed Array of Processors) est composé d'une grille torique de processeurs élémentaires qui traitent chacun un individu de l'AG.

L'algorithme implémenté par ces deux auteurs sur le DAP est tel que chaque processeur génère et évalue un seul individu. Les opérateurs sélection et croisement s'effectuent entre processeurs voisins, l'opérateur mutation est appliqué avec une probabilité pm pour chaque processeur.

L'intérêt de l'approche est qu'un DAP 510 (sur lequel cet algorithme a été implémenté) possède une grille de 32x32 processeurs, ce qui permet de traiter en parallèle une population de 1024 individus. Il est également important de noter que cet algorithme tire parti de l'architecture du DAP en exploitant l'efficacité des connections entre processeurs voisins.

A l'utilisation, cet algorithme permet la visualisation de la progression des génomes artificiels sur la grille de processeurs. Etant donné la nature locale des interactions entre processeurs, il n'y a pas de convergence prématurée mais plutôt une lente diffusion des caractéristiques des individus au sein de la grille qui permet la formation progressive de grappes d'individus, correspondant à l'exploration de différents points de l'espace de recherche.

I.11.2 Le calcul sur machine MIMD

Il est également possible de tirer parti d'une architecture MIMD [10] Ainsi, H. Muhlenbein, M. Schomisch et J. Born [15] ont développé un AG parallèle (qu'ils nomment PGA) sur un réseau de processeurs.

Le PGA est dédié à la minimisation de fonctions dont les solutions peuvent s'exprimer sous la forme d'un vecteur de nombres réels. La représentation binaire des solutions traitées

par le PGA est tout simplement celle des nombres flottants que la machine utilise pour ses calculs arithmétiques.

Le PGA est original à plusieurs titres :

La représentation des individus :

Chaque individu possède plusieurs chromosomes. Chaque chromosome est la représentation, en virgule flottante, d'un des nombres de la solution ainsi codée. L'opérateur de croisement opère sur le contenu de chromosomes homologues (de même indice) des individus mises en jeux. Un autre opérateur, la recombinaison, est chargé d'échanger les chromosomes homologues d'un individu en préservant leur contenu.

La fragmentation en sous-populations :

Chacune des sous-populations évolue en isolation, pendant un nombre de cycle fixé, sur un processeur particulier.

La migration :

Les canaux de communication entre processeurs sont utilisés pour transférer les meilleurs individus d'un processeur à l'autre.

La topologie du réseau :

Ce n'est pas une simple grille, mais une sorte d'échelle dans laquelle chaque processeur communique avec quatre autres processeurs.

L'optimisation locale :

Les individus d'une population peuvent être soumis à une procédure d'ascension de gradient ou par un autre algorithme d'optimisation.

Encore une fois, nous nous trouvons confrontés à une situation dans laquelle il nous est impossible de déterminer à priori une architecture de réseau de processeurs qui sera adaptée au traitement de tous les problèmes. Idéalement, cette architecture doit dynamiquement se reconfigurer en fonction de la nature du problème traité. Cette capacité d'auto organisation, pour un réseau de processeurs, si elle est pour le moment utopique en terme d'implémentation

effective, sera d'autant plus nécessaire qu'augmentera le nombre de processeurs mis en jeu. Car s'il est encore possible d'adapter à la main l'architecture d'un réseau constitué d'une dizaine de processeurs à un problème spécifique, il devient impossible de spécifier une architecture si le nombre de processeurs devient très grand.

I.12 Application des Algorithmes Génétiques en commande

On peut classer les applications des AGs en commande dans deux grandes catégories [15]:

- Synthèse et dimensionnement en mode off-line
- Réglage et adaptation en mode on-line

Dans l'application off-line, on peut utiliser les AGs comme un moyen de recherche ou d'optimisation. Par exemple : le calcul d'une loi de commande convenable qui satisfait un critère de performance donné pour un processus connu ou la recherche des paramètres optimaux d'un régulateur particulier ou d'une structure de commande.

Dans l'application on-line, les AGs peuvent être utiliser comme un mécanisme d'apprentissage pour identifier les caractéristiques d'un processus inconnu ou un système non stationnaire ou pour le réglage d'un contrôleur adaptatif.

Dans le reste de cette partie, quelques applications récentes d'AGs dans la commande et l'identification sont considérées où les méthodes conventionnelles sont inaptes ou non disponibles.

I.12.1 Régulation et commande des systèmes

L'une des applications les plus courantes d'AGs est l'optimisation paramétrique. Dans la commande des systèmes, beaucoup de problèmes peuvent être transformés en un problème d'optimisation. Les méthodes d'optimisation conventionnelle peuvent aboutir à des solutions locales, par ce qu'ils sont forcées de considérer seulement des petites régions de l'espace de recherche qui est influencé par les valeurs initiales. Les AGs sont capables d'échantillonner l'espace de la solution entier et par conséquent peuvent produire des solutions qui sont plus globales. En outre, les approches révolutionnaires ont la capacité de trouver des solutions dans beaucoup de régions différentes de l'espace de la recherche simultanément. Donc plus de

renseignements concernant la nature du problème de commande peuvent être découverts pendant la recherche.

Il a été montré par plusieurs chercheurs à travers l'optimisation paramétrique que les AGs sont des stratégies efficaces dans la commande off-line des processus. Krishnakumar et Goldberg (1992) et Bramlette et Cousin (1989) [15] ont démontré comment l'optimisation par AG peut être utilisée pour déterminé des structures du contrôleur dans les applications aérospatiales en moins de temps (concernent la fonction d'évaluation) que les autres méthodes tel que LQR "linear quadratic regulator" et autres.

Varsek (1993) [15] a démontré comment les AGs peut être utilisé dans le choix et le réglage d'une structure de contrôle, tandis que dans la robotique Gleghorn (1989) [15] a démontré comment les AGs peut être utilisé pour les problèmes de détermination des trajectoires (navigation) dans les environnements stationnaires et non-stationnaire [18]. Murray-smith et Sharman (1995) [15] ont appliqué les AGs pour la synthèse d'un régulateur flou par mode glissant "fuzzy-sliding mode" appliqué à un système hydraulique. Gray et Li (1995) [21] ont développé une fonction d'évaluation multiobjective qui prend en considération toutes les performances : stabilité du système, précision et rapidité[27].

I.12.2 Commande robuste

Une approche pour synthétiser une commande robuste est par l'utilisation du placement de vecteurs propres "eigenstructure assignment". Patton et Liu (1994) [15] ont développé une approche hybride qui combine les AGs et l'optimisation par la méthode du gradient. Leur plan a été appliqué à la détermination d'une commande d'un avion. Ils ont employé une représentation réelle dans la minimisation d'une fonction d'évaluation basée sur une combinaison de la sensibilité et la sensibilité complémentaire du système en boucle fermé. Dans l'étape de l'évaluation de l'AG, chaque individu est amélioré par un pas d'un algorithme (DFP) concernent la commande robuste et les individus résultants sont évalués d'après la fonction d'évaluation. Le reste de l'AG traite la population par les opérateurs habituelle. Il est à noter que cette approche fait plein usage de la liberté offerte par le placement de vecteurs propres pour trouver un régulateur qui minimise le critère de performance.

Dans une autre approche, Dakev (1995) [15] emploient un AG dans une procédure spéciale " the loop shaping design procedure" pour trouver les fonctions poids pour un contrôleur robuste d'un système critique MIMO.

L'AG emploie une représentation structurée [15] qui autorise la recherche de plusieurs fonctions poids et des paramètres identifiés simultanément. Le problème de commande considéré était un système de suspension pour un véhicule soulevé par une force magnétique (EMS) et le problème de commande a été basé sur la recherche des fonctions poids qui donnent une forme souhaitée des valeurs singulières de la fonction de transfert en boucle ouverte du problème H_∞ par facteurs coprimes (premiers) normalisés du système nominal. Ceci permettra de satisfaire le critère de performance en boucle fermée.

La technique type grimpeur "Hill climbing techniques" avait été proposé comme une solution potentielle pour une telle combinaison des problèmes d'optimisation. Cependant, l'approche des AGs a été trouvée avantageuse si le domaine des poids était grand et au-delà du champ d'application des approches conventionnelles. En particulier, il est à noter que les AGs permettent de trouver des contrôleurs répandant au critère H_∞ d'ordre inférieur à ceux synthétisé par les méthodes classiques.

I.12.3 Identification

Beaucoup de problèmes de commande, traitement du signal et les procédures d'apprentissage peuvent être transformer sous forme de problème d'identification des processus où l'objectif est de déterminer un modèle convenable à partir d'un ensemble de données entrées-sorties. Le modèle résultant peut être utilisé pour la prédiction et le contrôle d'une " boîte noire ".

Il existe plusieurs méthode pour l'identification des systèmes linéaire mais en pratique la plupart des systèmes sont non linéaires, la complexité de l'identification des systèmes non linéaire particulièrement lorsqu'il n'y a pas d'information initiale ou bien si on ne connaît par la structure du modèle, a contribué au manque relatif d'attention porter à ce domaine particulier.

Une approche prospère à ce problème est la méthode LSR (Least Squares Regression) Chen (1989) [15] c'est une méthode qui permet de trouver un ensemble convenable de termes non - linéaires pour le système en question. Cependant, la complexité du problème et l'augmentation de l'espace de recherche fait qu'une étude exhaustive n'est pas toujours faisable ce qui montre les limites de l'application.

Fonseca (1993) [15] a implémenté une identification d'un système non – linéaire par une méthode hybride entre les algorithmes génétiques et l'algorithme LSR. Il a divisé la

population en sous-ensembles et la sélection est effectuée au niveau de chaque sous-ensemble. L'AG est utilisé pour sélectionner un nombre fixe de termes à partir d'un ensemble de termes non-linéaires possibles et l'algorithme LSR est utilisé pour identifier les paramètres de ces termes. Comme les AGs opèrent avec une population de solution, ils produisent une famille de modèles qui peuvent être évaluer selon différents critères avant de choisir un modèle final.

Comparer avec les approches conventionnelles qui cherchent les termes par itérations en construisant de plus en plus des modèles complexes, L'approche des AGs conduit vers une recherche globale est robuste. Ainsi les AGs ont la capacité d'être plus efficace dans l'identification pour l'obtention d'un model de structure convenable.

La stratégie de l'identification génétique peut être rendue plus efficace en permettant à la recherche d'être conduite sur un nombre de termes variables. Le problème peut être lancer comme un multiobjective en considérant les termes et leur nombre simultanément. Pour adapter une population avec des individus composer de nombre de termes variable, il faut utiliser une structure de représentation approprier. En utilisant une représentation hiérarchique, les gênes des individus peuvent être utiliser pour activer ou désactiver des termes spécifiques pour un modèle. Cette approche est appliquer avec succès par Roberts et Wade (1993) [15]. Dans un sens similaire, une représentation sous forme d'arbre est utilisée, les sous arbres peuvent être échangés entre les individus pendant le croisement sans le besoin de réévaluer cette partie.

I.12.4 Temps-réel et commande adaptative

Deux problèmes majors sont rencontrés dans l'utilisation des AGs dans la commande en temps réel : le temps d'exécution et l'exigence de produire une loi de commande satisfaisante à chaque génération.

- Le problème de temps d'exécution peut être résolu par un des algorithmes spécifiques :

Algorithme Génétique parallèle : les individus sont regroupés en sous-populations et on utilise des opérateurs locaux et quelques modifications appropriées. Chaque sous-ensembles est traité à part sur une machine séquentielle. Un traitement global à la fin de chaque génération permet de déterminer la solution optimale.

Algorithme Génétique incrémental : il produit seulement un ou deux descendants à chaque génération par conséquent il a l'avantage de réduire la mémoire nécessaire et le temps d'exécution, mais il se peut qu'il ne produise pas des bons individus

Micros Algorithmes Génétiques : ils emploient des populations très petites et ils ont la capacité de produire plusieurs descendants mais il y aura un manque de diversité génétique.

- L'obligation de produire une loi de commande satisfaisante à chaque génération est le problème le plus difficile. Pour cela l'utilisation des algorithmes génétique en temps réel est limiter dans les applications où les régulateurs sont peu sensibles aux variations. On peut avoir un large domaine de paramètres convenables qui donne une loi de commande satisfaisante. Ainsi, c'est possible qu'un algorithme génétique produise des lois de commande consécutives sans accepter les grands changements des paramètres. Malgré cela, plusieurs plans prospères ont été développés. Porter et Jones (1992) ont développé une approche génétique pour ajuster un régulateur PID digitale. Ils ont noté que cette réalisation est plus simple qu'un calcule préalable des paramètres du régulateur par une technique d'optimisation, même dans les cas les plus simples.

Une approche relative à l'utilisation on – line des AGs est celle du contrôleur flou. Karr(1992) [15] a expliqué comment les AGs peuvent être utilisés pour déterminer le régulateur flou d'un système dynamique. Dans cette approche les AGs sont utilisés pour optimiser les fonctions d'appartenance du régulateur. Il est à noter que les règles floues tendent à rester constante, même si on a un grand nombre de condition et que les foncions d'appartenance doivent être adaptée à cette situation.

I.13 Conclusion

Dans ce chapitre nous avons présenté les Algorithmes Génétiques. C'est un algorithme d'optimisation itératif de recherche globale et parallèle.

Il est intéressant d'utiliser les Algorithmes Génétiques dans le cas où les espaces de recherche seraient importants et/ou accidentés. Le principe de la recherche consiste à échantillonner l'espace en construisant une population d'individus, à évaluer ses différents individus grâce à une fonction d'évaluation et à partir de ces évaluations, à construire une nouvelle population. Ce mécanisme permet une recherche très efficace car il utilise

implicitement les similarités entre les individus pour extraire de l'information sur les régions de l'espace où la fonction d'évaluation présente de fortes valeurs.

Nous avons étudié le fonctionnement des Algorithmes Génétiques et les différents éléments qui les constituent. A travers le théorème des schémas, nous avons vu l'influence des opérateurs de l'AG: sélection, croisement et mutation, sur la convergence en suite nous avons exposé différentes techniques pour éviter la convergence prématurée de l'algorithme en réalisant un compromis entre l'exploration de l'espace de recherche et l'exploitation des résultats.

Les Algorithmes Génétiques nécessitent une grande puissance de calcul et consomment beaucoup de temps, pour remédier à ce problème nous avons exposé deux approches de calcul parallèle par un réseau de processeurs pour les utilisations des AGs en temps réel.

Chapitre II

Application des Algorithmes Génétiques

II.1 Identification

L'identification est une approche expérimentale qui permet la détermination des caractéristiques dynamiques d'un procédé dont la connaissance est nécessaire pour la conception et la mise en œuvre d'un système performant de régulation. [25]

Le signal d'entrée utilisé pour l'identification doit être centré et riche en fréquence. En pratique, ce signal est fourni par l'utilisation d'une *séquence binaire pseudo - aléatoire.* (SBPA)

La longueur de la SBPA est choisie d'une manière a ce que la durée de l'impulsion maximale t_{im} soit supérieure au temps de montée t_M du procédé. La durée maximale d'une impulsion étant $N.T_e$ (N : nombre de cellules du registre de décalage qui génère la SBPA, T_e la période d'échantillonnage) il résulte la condition :

$$T_{im}=N.T_e>t_m \tag{II.1}$$

II.1.1 Identification par les Algorithmes Génétique

Comme nous l'avons vue auparavant, les Algorithmes Génétiques peuvent être utilisés dans plusieurs domaines dont celui de l'identification. Dans ce paragraphe nous examinons les performances de l'identification par Algorithmes Génétiques. On applique cette méthode pour l'identification de deux types de processus, le processus autoregressif à entrée exogène (ARX) et le processus autoregressif à moyenne ajustée à entrée exogène (ARMAX). Pour chaque processus on détermine les paramètres optimaux à l'aide des AGs.

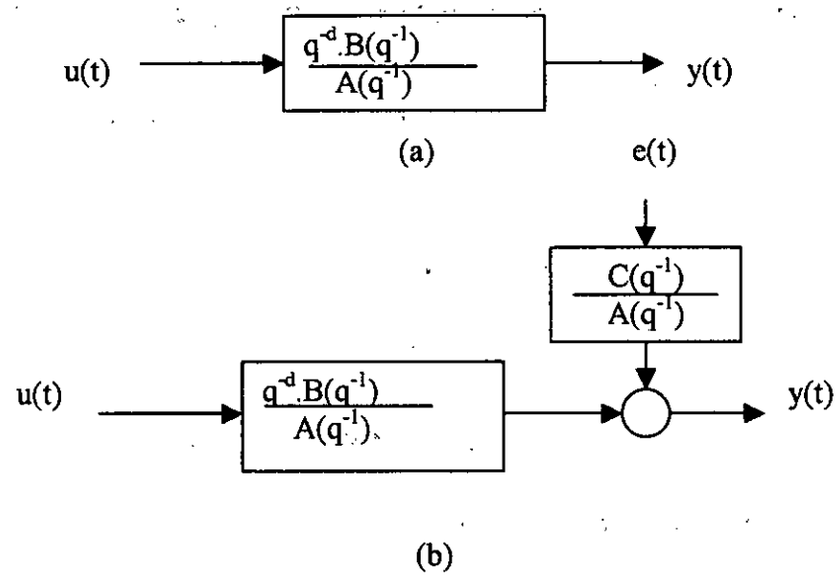
Les fichiers entrée-sortie (E/S) utilisés pour l'identification par la structure ARX, ARMAX contiennent respectivement 31 et 255 enregistrement. L'entrée est une SBPA générée par des registres de longueur N=5 et N=8. Deux fichiers, générés à partir de modèles

connus, sont utilisés pour l'identification pour les deux structures ARX et ARMAX. Des perturbations stochastiques ont été rajoutées au deuxième fichier pour l'identification par un processus ARMAX.

Deux techniques de sélection sont utilisées, sélection linéaire et N/2-élitisme. Une comparaison empirique est faite pour déterminer l'efficacité de chaque technique.

II.1.2 Models ARX et ARMAX

Les modèles ARX, ARMAX sont des représentations linéaires discrètes d'un processus. leurs structures générales sont représentées respectivement sur les figures II.1 (a) et (b)



Figure(II.1) schéma représentant les structure ARX (a), ARMAX (b)

avec : u(t) et y(t) sont l'entrée et la sortie, e(t) est un bruit blanc de moyenne nulle est de variance unitaire et q^{-1} est l'opérateur retard dans le temps.

A(q-1), B(q-1),et C(q-1) sont des polynômes de q^{-1} tels que

$$A(q^{-1})=1+a_1.q^{-1}+a_2.q^{-2}+\cdots+a_n.q^{-n} \qquad \text{(II.2)}$$

$$B(q^{-1}) = b_1.q^{-1} + b_2.q^{-2} + \cdots + b_n.q^{-n} \qquad \text{(II.3)}$$

$$C(q^{-1})=1+c_1.q^{-1}+c_2.q^{-2}+\cdots+c_n.q^{-n} \qquad \text{(II.4)}$$

On remarque aussi, Pour la courbe de la moyenne lors d'une sélection N/2-élitisme, que la moyenne de la fonction d'évaluation de la population est toujours en progression contrairement a celle de la sélection linéaire qui présente des chutes dans la moyenne

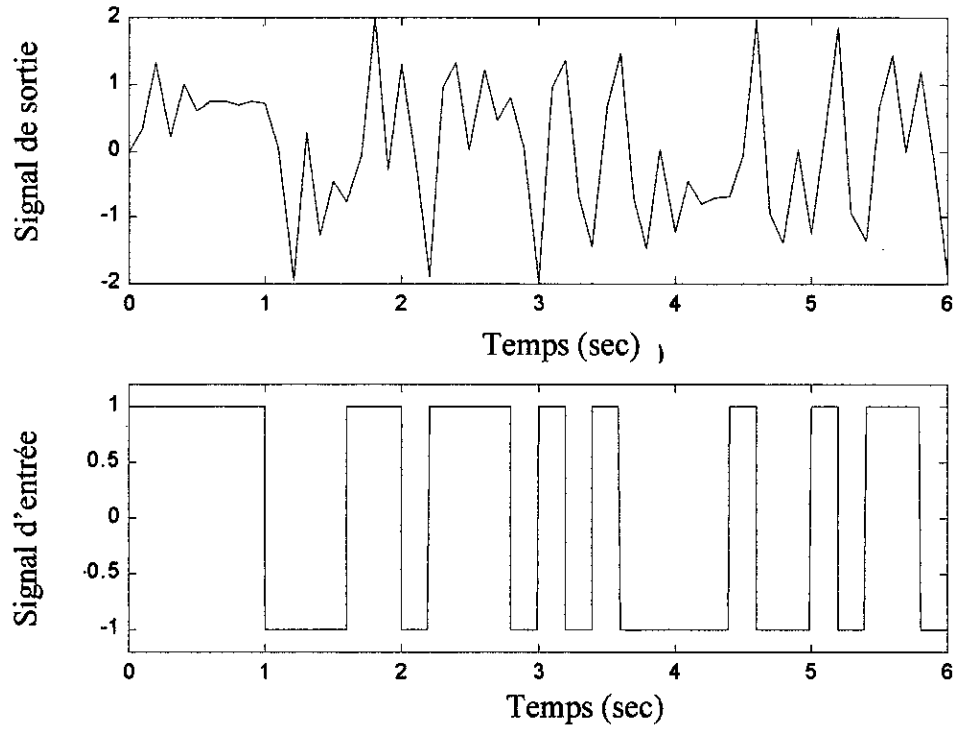
La convergence de l'AG avec sélection N/2-élitisme est plus rapide car seulement les meilleurs individus sont choisis pour la reproduction par contre la méthode linéaire donne une chance aux individus moins performants d'être sélectionné causant les chutes de moyenne ce qui ralentie la convergence.

paramètres	vrais	Par AG	
		N/2-élitisme	Linéaire
a1	1	1.0052	0.9929
a2	1/3	0.3336	0.3325
b1	1/3	0.3335	0.3424
b2	4/3	1.3422	1.3259
Somme des erreurs quadratiques		0.0131	0.0198
Fonction d'évaluation		0.9870	0.9805

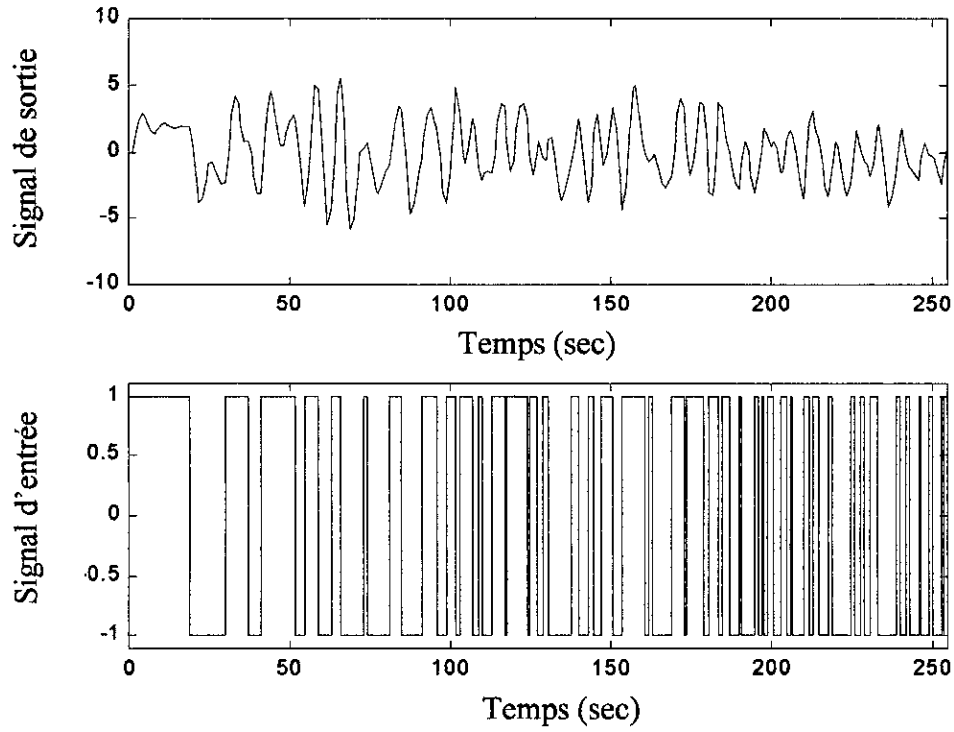
Tableau (II.1) : Paramètres du modèle ARX obtenu par les deux techniques de sélection

paramètres	vrais	Par AG	
		N/2-élitisme	Linéaire
a1	-1	-1.0007	-0.9922
a2	0.7	0.6963	0.6941
b1	0.8	0.8123	0.7882
b2	0.5	0.4965	0.5059
c1	1	1.0385	0.9863
c2	-0.2	-0.2622	-0.2235
Somme des erreurs quadratiques		0.0202	0.0196
Fonction d'évaluation		0.9802	0.9807

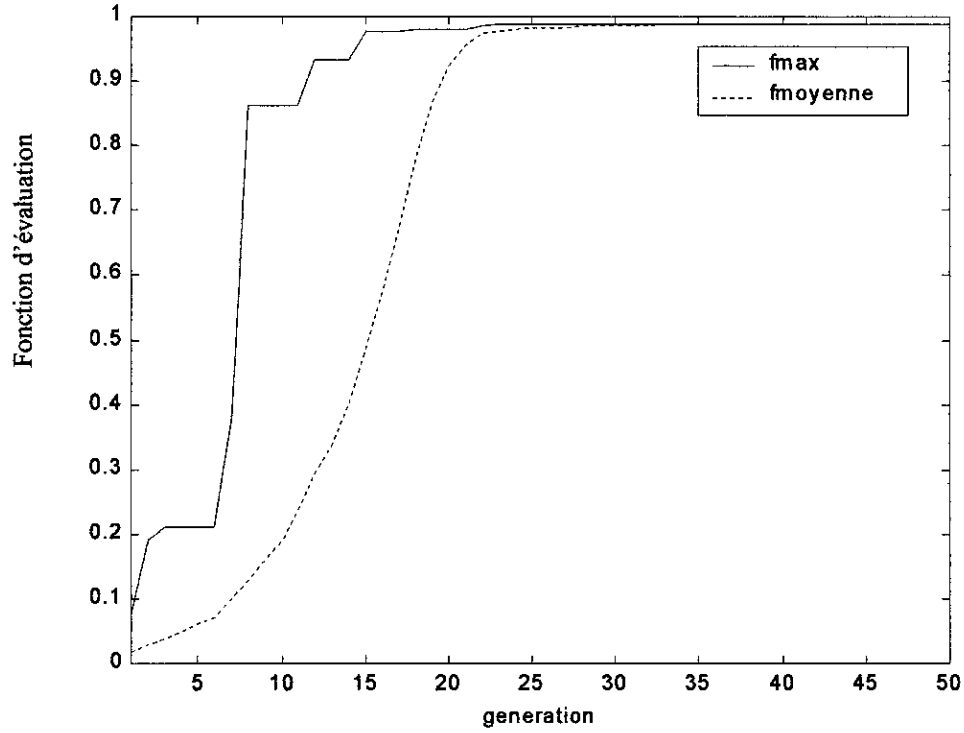
Tableau (II.2): Paramètres du modèle ARMAX obtenu par les deux techniques de sélection.



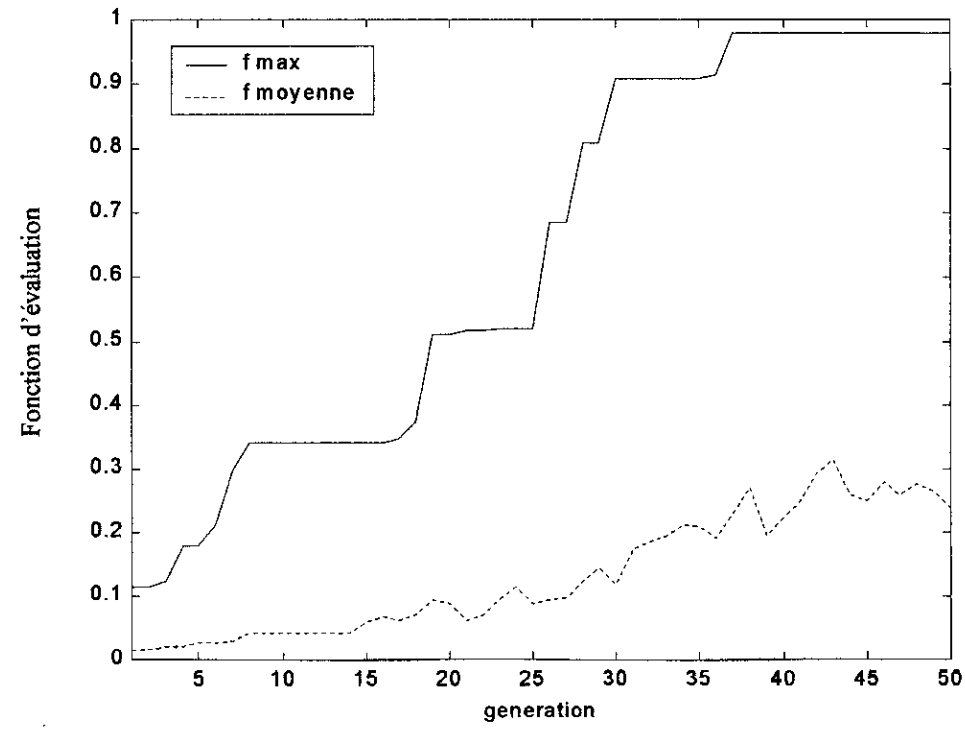
Figure(II.2) : Le fichier entrer sortie utiliser pour l'identification
Par une structure ARX



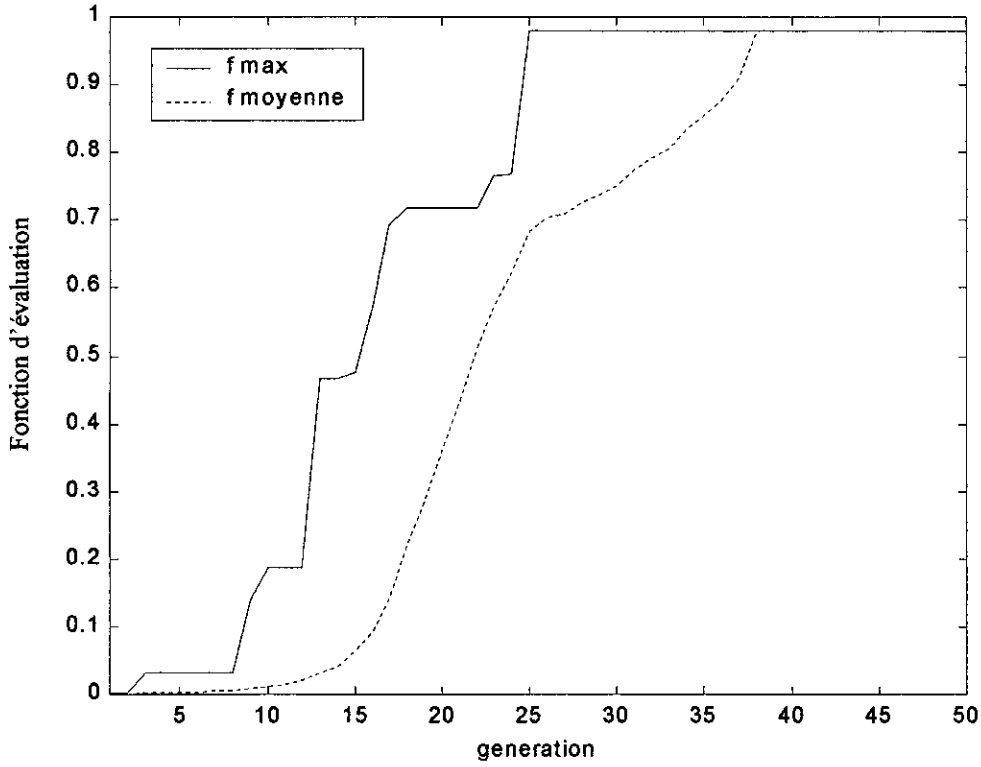
Figure(II.3) Le fichier entrer sortie utiliser pour l'identification
Par une structure ARMAX



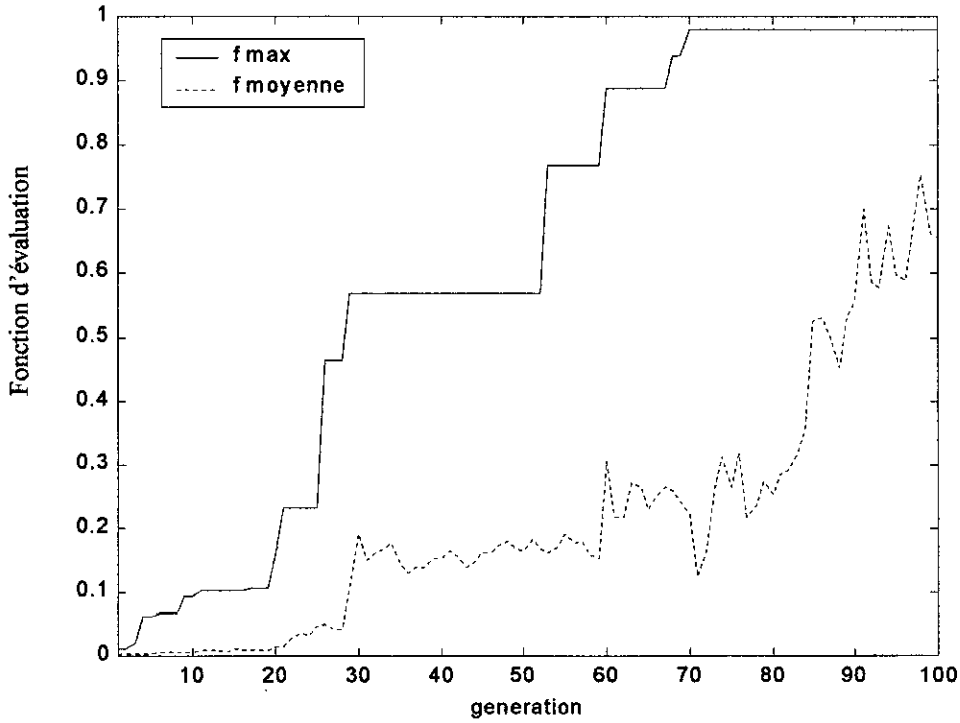
Figure(II.4) Evolution de la fonction d'évaluation fmax et sa moyenne avec sélection N/2-élitisme appliquée pour la structure ARX



Figure(II.5) Evolution de la fonction d'évaluation fmax et sa moyenne avec sélection linéaire appliquée pour la structure ARX



Figure(II.6) Evolution de la fonction d'évaluation fmax et sa moyenne avec sélection N/2-élitisme appliquée pour la structure ARMAX



Figure(II.7) Evolution de la fonction d'évaluation fmax et sa moyenne Avec sélection linéaire appliquée pour la structure ARMAX

II.1.4 Conclusion

Dans cette partie nous avons vu l'application des AGs à l'identification paramétrique des deux modèles ARX et ARMAX. Cette approche est intéressante par les résultas obtenus d'autant plus que les différents essais ont abouti à des résultats similaires et proche des vrais paramètres

Pour cette application, la sélection N/2-élitisme et la sélection linéaire ont été utilisée dans la conception des AGs. A travers les résultas, Nous avons pu distinguer les différences entre ces deux techniques. En effet, la sélection linéaire favorise l'exploration de l'espace de recherche en gardant la diversité génétique dans le but d'éviter les optimums locaux au détriment de la rapidité de convergence. La sélection N/2-élitisme permet une meilleure exploitation des individus et converge plus rapidement vers le maximum et à la fin, la population est constituée du même individu.

II.2 Réduction d'ordre

La réduction d'ordre d'un modèle sert à simplifier la conception de la commande. Il existe plusieurs méthodes de réduction d'ordre développés par différentes techniques. En générale, elles sont basées sur la minimisation d'un critère quadratique sur l'erreur de sortie. Ces méthodes son souvent difficiles et nécessitent un calcul énorme par la résolution d'un système d'équations non-linéaires ou par une procédure itérative. De la difficulté de trouver des solutions à travers les méthodes classiques, réside l'avantage de l'utilisation des AGs pour cette application.

La réduction d'ordre d'un modèle stable de dimension n consiste à trouver un autre modèle stable réduit de dimension p inférieur à n et qui décrit exactement le comportement du même processus.

II.2.1 Réduction d'ordre par Algorithme Génétique

Dans ce paragraphe, nous étudions la réduction d'ordre par Algorithme Génétique. On applique cette méthode pour réduire l'ordre d'une fonction de transfert et celui d'une équation d'état. Les paramètres optimaux des modèles réduits sont déterminés par l'algorithme génétique.

Dans cette application, deux techniques de mutation sont utiliser, mutation constante et mutation variable. Une comparaison empirique est faite pour déterminer l'influence de la probabilité de mutation sur l'évolution de l'Algorithme Génétique.

II.2.2 Description de la méthode

La méthode employée consiste à transformer le problème de réduction d'ordre en un problème d'identification. En effet, un fichier entrée – sortie est utilisé à partir du modèle initialement connu qu'on veut réduire et on utilise l'AGs pour la recherche d'un autre modèle d'ordre inférieur qui peut générer le même fichier.

II.2.3 Réduction d'ordre d'une fonction de transfert

La fonction de transfère w(s) d'ordre 3 du model choisi, est donnée par l'équation 4.7.

$$w(s)=\frac{s+2}{s^3+2.s^2+3.s+4} \qquad \text{(II.7)}$$

On désir trouver une fonction de transfert ŵ(s), équivalente à w(s), d'ordre 2 dont la forme est donné par l'équation II.8 et dont la réponse a un échelon unitaire coïncide avec celle de w(s).

$$\hat{w}(s)=\frac{b_0+b_1.s}{a_0s^2+a_1.s+a_2} \tag{II.8}$$

II.24 Réduction d'ordre d'une équation d'états

L'équation d'état sous forme compagne de commande du même modèle ci-dessus est donné par :

$$\begin{cases} \dot{x}=Ax+bu \\ y=c^T x \end{cases} \tag{II.9}$$

Avec :

$$A=\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -3 & -2 \end{bmatrix}, \quad b=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{et} \quad c^T=\begin{bmatrix} 2 & 1 & 0 \end{bmatrix} \tag{II.10}$$

Dans ce cas, on désir trouver une équation d'état sous forme compagne de commande dont la matrice $\hat{A}$ et les vecteurs $\hat{b}$ et $\hat{c}$ s'écrivent sous les formes :

$$\hat{A}=\begin{bmatrix} 0 & 1 \\ a_0 & a_1 \end{bmatrix}, \quad \hat{b}=\begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{et} \quad \hat{c}^T=\begin{bmatrix} c_0 & c_1 \end{bmatrix} \tag{II.11}$$

II.2.5 Paramètres de l'Algorithme Génétique

L'AG suivant est utilisé pour les deux applications, réduction d'ordre de la fonction de transfert et de l'équation d'état.

- Les variables à optimiser sont séparément :

  Le numérateur et le dénominateur de la fonction $\hat{w}(s)$

  Les composants de la matrice $\hat{A}$ et le vecteur $\hat{c}$ du système d'état.

- Chaque paramètre est codé en binaire sur 16bits dans l'intervalle [0, 10]
- La population contient 100 individus et le nombre de générations est de 50
- La fonction d'évaluation est donnée par :

$$f = \frac{1}{1+J} \tag{II.12}$$

$$J = a\left|g-\hat{g}\right| + \sum_{k=0}^{n} e(k)^2 \tag{II.13}$$

Avec: $g,\hat{g}$ sont respectivement les gains statiques de w(s) et ŵ(s),

La fonction d'évaluation et le critère vérifient les inégalités suivantes :

$$0 < j < \infty. \text{ Et } 0 < f < 1 \tag{II.14}$$

l'erreur e est donnée par : $e = y - \hat{y}$ (II.15)

y et $\hat{y}$ sont respectivement les réponses à un échelon unitaire du système initiale et du système réduit.

- La sélection linéaire est appliquée dans cet algorithme.

Les résultats obtenus par l'AG en utilisant une probabilité de mutation constante pm = 0.01, et une probabilité de mutation variable donnée par l'équation II.16 sont présentés aux tableaux II.3 et II.4

$$pm = 0.5.\exp\left(-\frac{5t}{ng}\right) \tag{II.16}$$

Avec t: indice de la génération et ng le nombre total de génération.

Les figures de (II.8) à (II.11) représentent les réponses à un échelon unitaire du système initial et des systèmes réduits pour les deux types de mutation et les figures(II.12) et (II.13) représentent l'évolution des fonctions d'évaluation.

En comparant les critères des deux applications à partir des tableaux (II.3) et (II.4), on remarque que les résultats obtenus par une mutation variable sont meilleurs que ceux obtenus par une mutation constante.

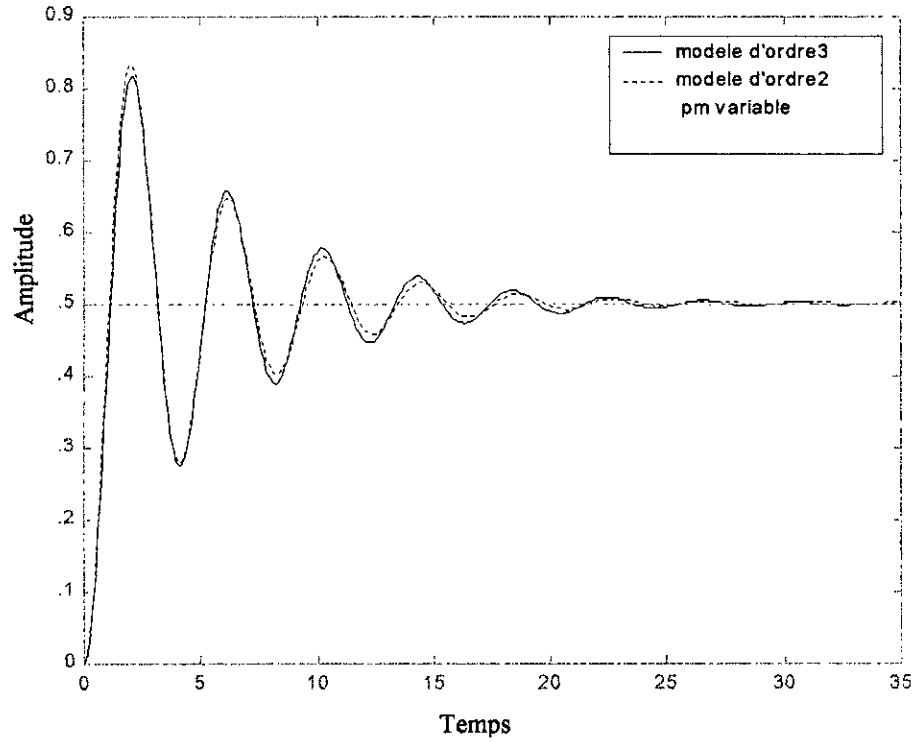
De plus on remarque à partir de la figure (II.13) que l'algorithme utilisant une probabilité de mutation constante converge vers un optimum local.

Paramètres	Par AG	
	Pm=0.01	Pm variable
a0	2.1552	3.3939
a1	1.0009	1.3400
a2	5.2528	7.9592
b0	0.2117	0.0951
b1	2.6222	3.9903
Somme quadratique des erreurs	0.1763	0.03820
Fonction d'évaluation	0.8501	0.9632

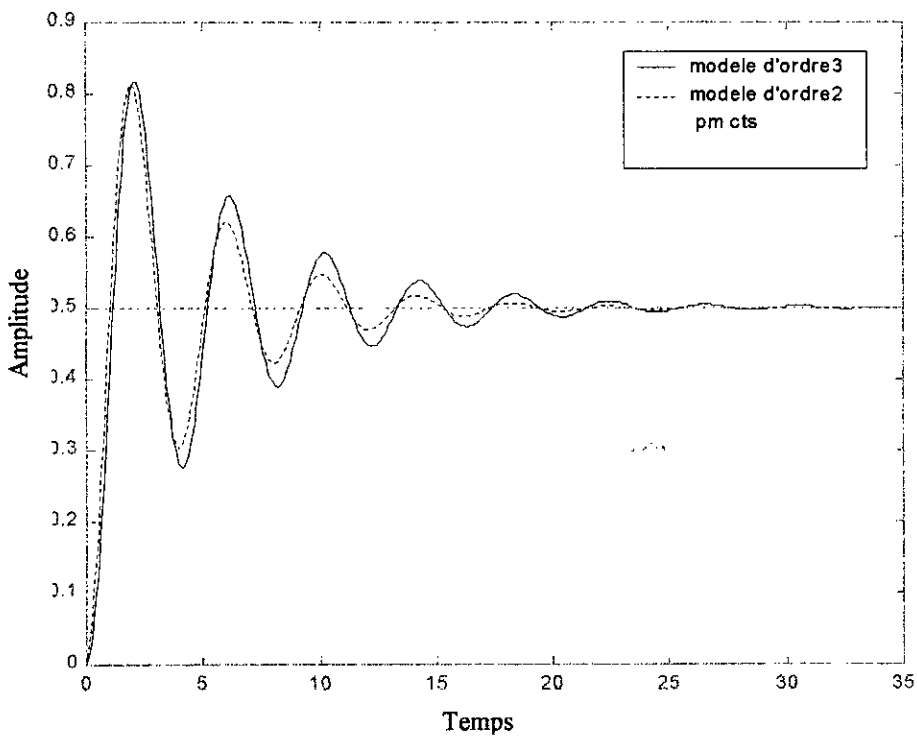
Tableau(II.3): les paramètres de la fonction de transfert $\hat{w}(s)$ obtenus par les deux types de mutation

Paramètres	Par AG	
	Pm=0.01	Pm variable
a0	2.0356	2.3156
a1	0.4135	0.3262
c0	0.9974	1.1588
c1	0.1561	0.0786
Somme quadratique des erreurs	0.3911	0.1211
Fonction d'évaluation	0.7189	0.8920

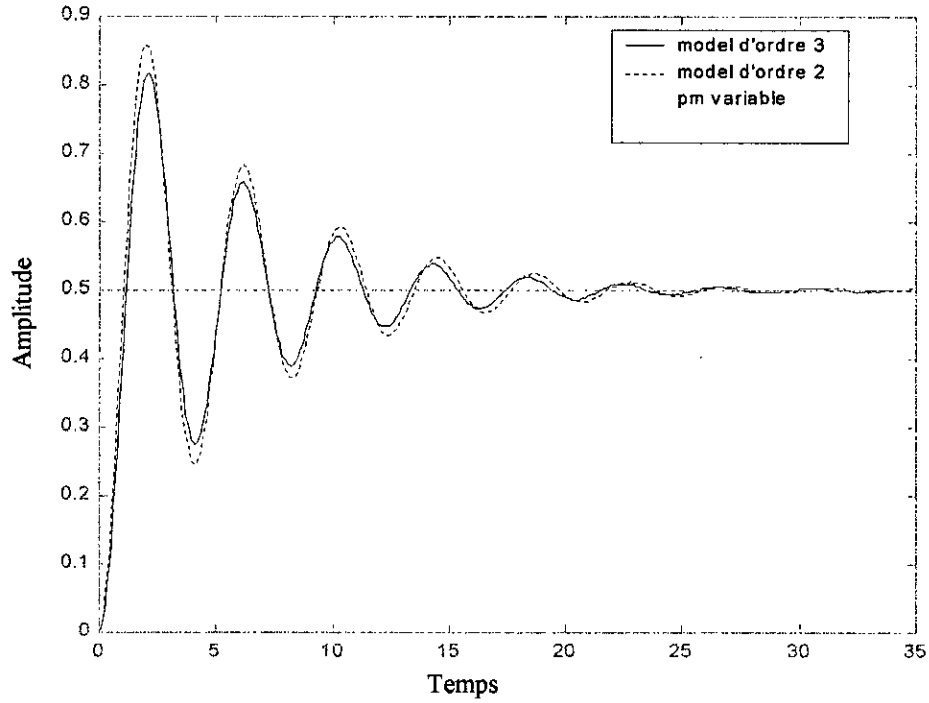
Tableau(II.4): les paramètres de l'équation d'état obtenus par les deux types de mutation



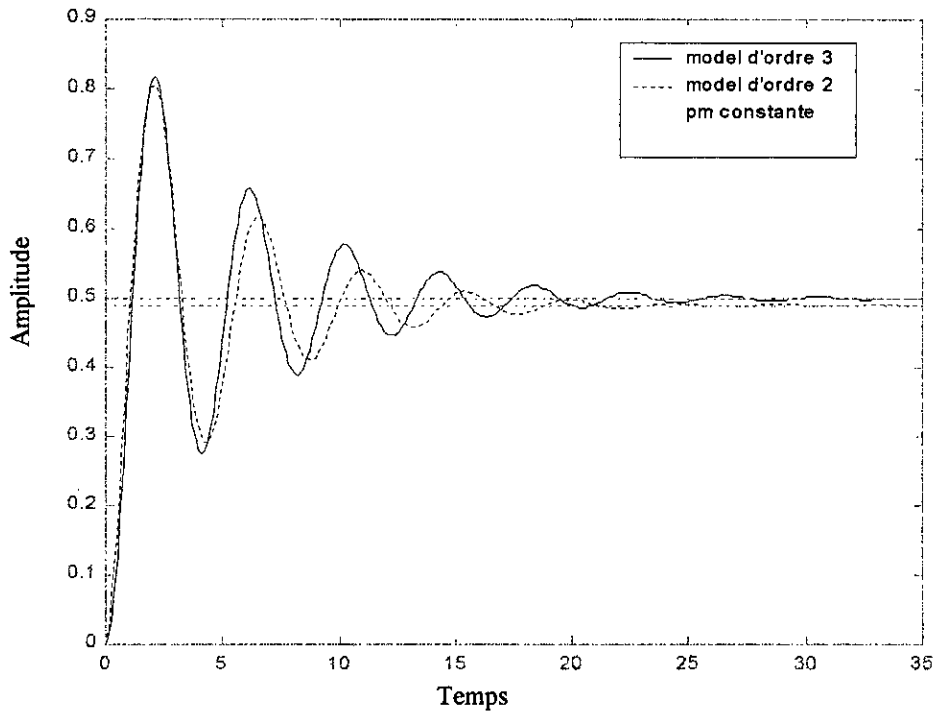
Figure(II.8) : réponse à un échelon unitaire de la fonction w(s)
(s) par une mutation variable $\hat{w}$ et la fonction réduite



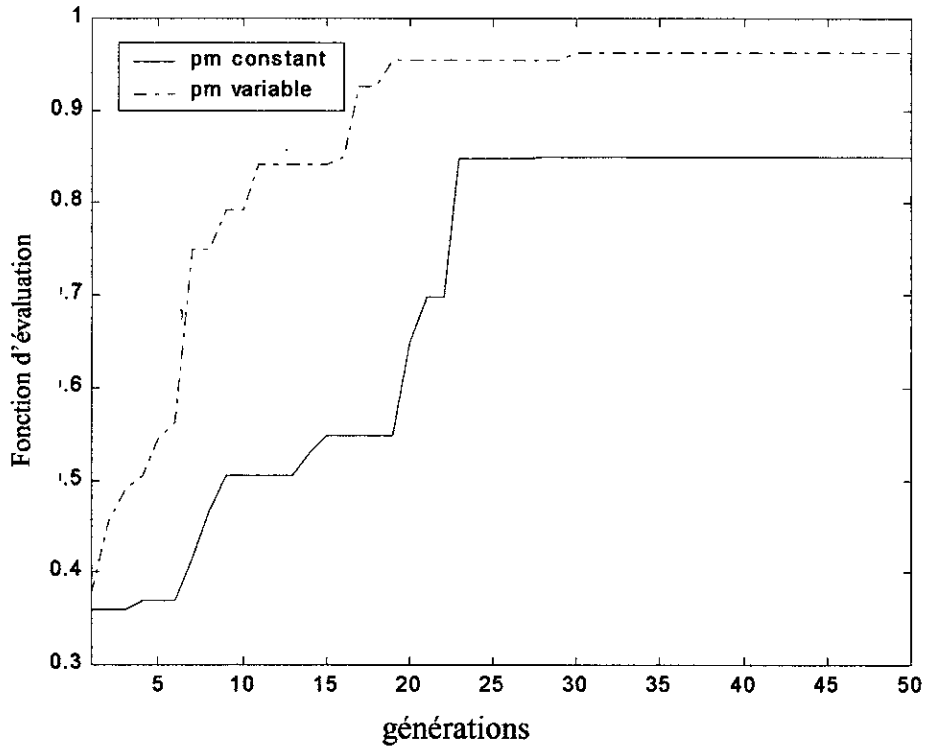
Figure(II.9) : réponse à un échelon unitaire de la fonction w(s)
(s) par une mutation constante $\hat{w}$ et la fonction réduite



Figure(II.10) : réponse à un échelon unitaire de l'équation d'état et l'équation d'état réduite par une mutation variable



Figure(II.11) : réponse à un échelon unitaire de l'équation d'état et l'équation d'état réduite par une mutation constate



Figure(II.12) : l'évolution de la fonction d'évaluation de l'AG appliqué pour la réduction d'une fonction de transfert

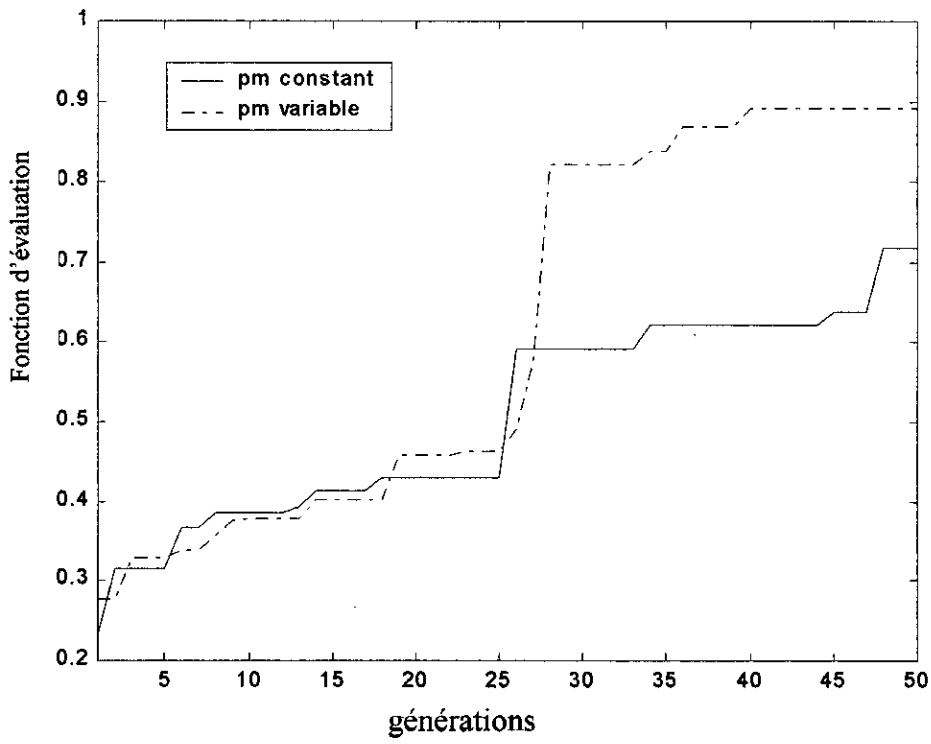


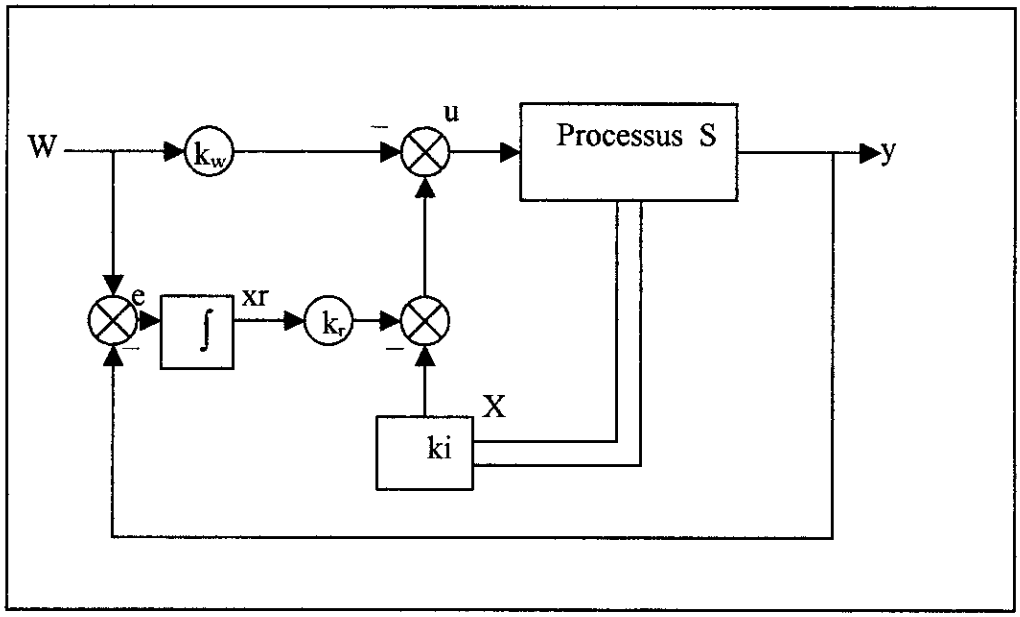
Figure (II.13) : l'évolution de la fonction d'évaluation de l'AG appliqué pour la réduction d'une équation d'état

II.2.6 conclusion

Dans cette partie, nous avons utilisé deux types de mutation pour l'application des AGs à la réduction d'ordre. Nous avons constaté que l'utilisation d'une mutation à probabilité variable décroissante est plus avantageuse car elle permet au début de l'algorithme, lorsque la probabilité de mutation est grande, une meilleure exploration de l'espace de recherche ce qui contribue à éviter les optimums locaux. Dans les dernières générations, la probabilité de mutation devient faible ce qui permet une meilleure exploitation des individus.

II.3 Réglage par retour d'état

Le réglage d'état est une méthode moderne qui est introduite dans le domaine de réglage industriel[14]. Dans beaucoup de cas, elle donne une bonne qualité de réglage par rapport aux méthodes classiques. Elle est basée sur le principe de contre réaction des variables d'états. La structure générale du réglage d'état, présenté sur la figure (II.14), est une structure composée de la contre réaction d'état, du régulateur intégrateur et de l'intervention directe de grandeur de consigne W



Figure(II.14) : schéma de la structure du réglage d'état

Le processus S est un système décrit par des équations d'états. Le vecteur X est alors son vecteur d'état. Les valeurs ki ,kr et kw sont des constants à déterminer.

La commande u est donnée par l'équation :

$$u = k_w . w - k_i^T X + kr . x_r \qquad \text{(II.17)}$$

avec :

$$x_r = \frac{1}{Ti}\int (w - y)dt \qquad \text{(II.18)}$$

D'habitude, pour la détermination des coefficients de la contre réaction, on fera appel au principe du placement des pôles, mais cette méthode nécessite un modèle linéaire. Le choix a priori des pôles n'est pas toujours facile et si on n'obtient pas une allure satisfaisante pour la

réponse indicielle, il faut modifier le choix des pôles et refaire les calcules jusqu'à ce qu'on aboutisse à une solution convenable.

II.3.1 Conception du retour d'état par Algorithme Génétique

Dans l'application suivante nous allons proposer une nouvelle approche pour synthétiser le réglage par retour d'état. On applique cette méthode à un système non-linéaire dont on ne peut pas appliquer le principe du placement de pôle. On fait appelle aux AGs pour déterminer les paramètres ki, kw et kr de la commande. Le système utilisé dans notre application et le pendule inverse.

II.3.1.1 Pendule inverse

Le pendule inverse est un système non linéaire d'ordre deux représenté par l'équation différentielle suivante :

$$ml^2\ddot{\theta} - mgl \sin\theta = u - b\dot{\theta} \qquad \text{(II.19)}$$

θ Angle que fait le pendule avec la verticale

g Accélération gravitationnelle = 9.81 m/s^2

m Masse = 2 Kg

l Longueur du pendule = 0.5 m

b Coefficient d'amortissement = 0.01 Nm^2/s

La commande est appliquée à ce système en utilisant la trajectoire désiré $w(t)$ définie par :

$$w(t) = 0.5\cos(3\pi t) \qquad \text{(II.20)}$$

Comme $\theta(0) = 0$ rad et $w(0) = 0.5$ rad l'erreur initiale $e(0)$ est toujours $e(0) = 0.5$ rad.

II.3.1.2 Paramètres de l'algorithme génétique

Les paramètres de l'AG sont les suivants :

- Les variables à optimiser sont les constantes : Kw, Kr et Ki i= 1, 2,
- Chaque paramètre est codé en binaire sur 8 bits sur l'intervalle [0, 100],

- La population contient 200 individus et le nombre de génération est de 50,

- Le critère à minimiser est $j=\sum_{k=0}^{n}\left(e^2(k)+\alpha\left|u(k)\right|\right)$, (II.21)

- La fonction d'évaluation est donnée par : $f=\frac{1}{1+j}$, (II.22)

avec : $0<j<\infty$. $0<f<1$, L'erreur est donnée par : $e=w-\theta$, (II.23)

- La probabilité de mutation variable *pm* et donné par

$$pm=0.5.\exp(-\frac{5t}{ng}) \quad \text{(II.24)}$$

- La sélection utilisée est la sélection linéaire.

Les paramètres de la commande obtenus par Algorithme génétique et qui optimise notre critère sont indiqués dans le tableau

Kw	*K1*	*K2*	*Kr*	*T*
62.4656	54.9891	62.3804	25.9432	26.4578

Tableau (II.5): les paramètres optimaux de la commande

La valeur du critère *j* est égale à : 0.3122

La valeur de maximale de la fonction d'évaluation est égale à 0.7620

II.3.2 Robustesse de la commande

L'étude de la robustesse d'une loi de commande consiste à étudier son comportement pour des conditions et des paramètres autres que ceux qui ont servis à sa conception. Le modèle considéré pour la détermination des paramètres est caractérisé par l'angle initial θ (0), la masse *m* et la langueur du pendule *l*. le test de robustesse consiste à faire varier les paramètres du système.

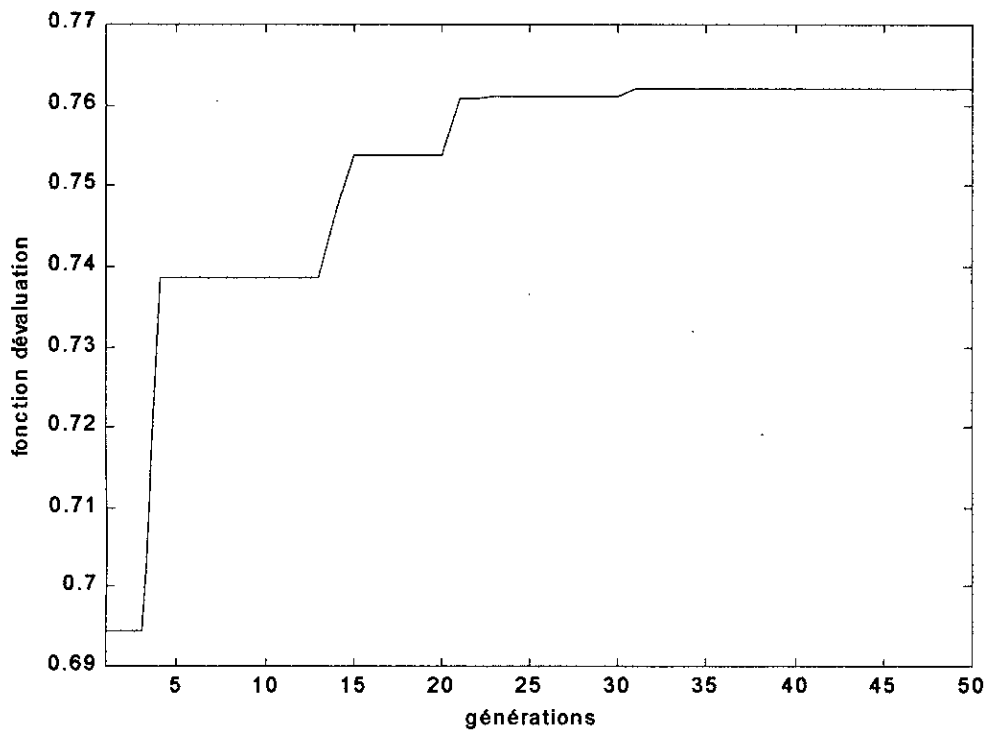
La figure (II.15) représente l'évolution de la fonction d'évaluation de l'Algorithme Génétique.

Les figures (II.16-18) représentent les résultats de simulation du modèle pour lequel les paramètres du régulateur sont optimisés.

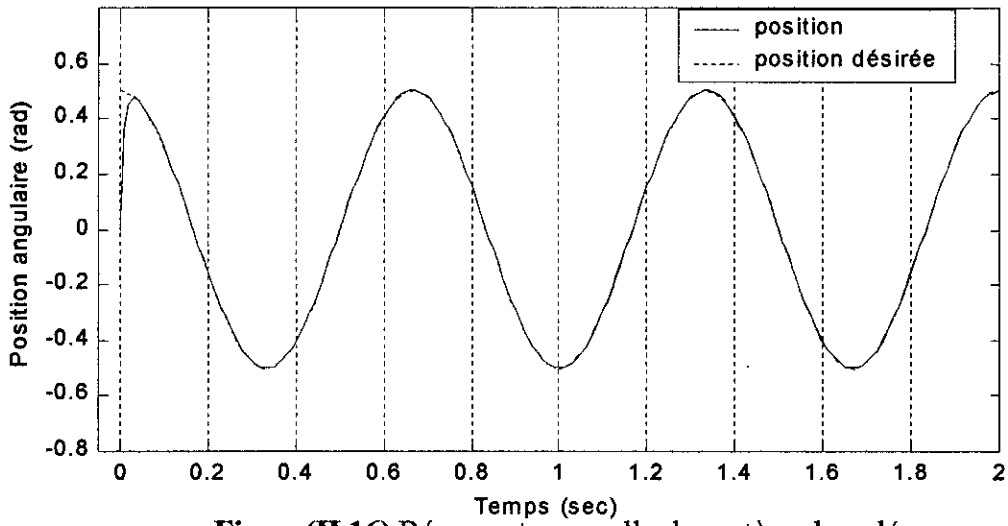
Les figures (II.19-21) et (II.22-24) représentent les résultats de simulation pour deux tests de robustesse.

Le premier test consiste à faire varier la masse (de 2 Kg à 1.8 Kg) et la longueur de pendule (de 0.5 m a 0.51 m) à listant t = 1 sec.

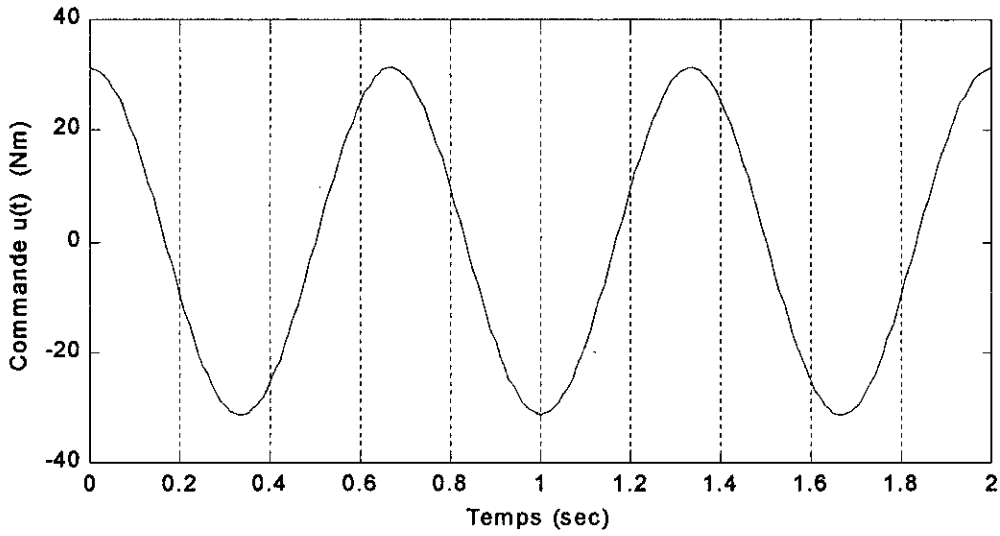
Le deuxième test consiste à changer la valeur initiale du pendule (de la position 0 à –0.3 rad)



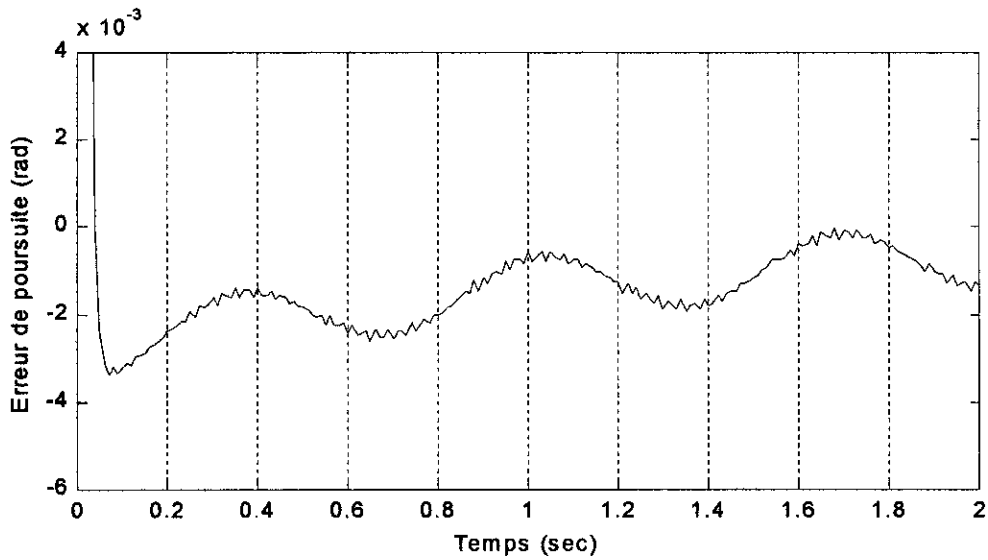
Figure(II.15) Evolution de la fonction d'évaluation



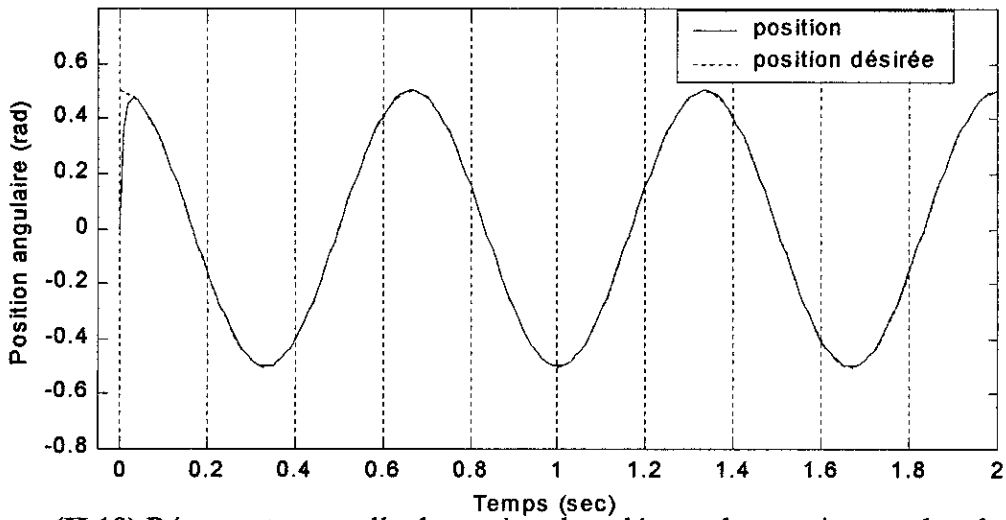
Figure(II.16) Réponse temporelle du système bouclé



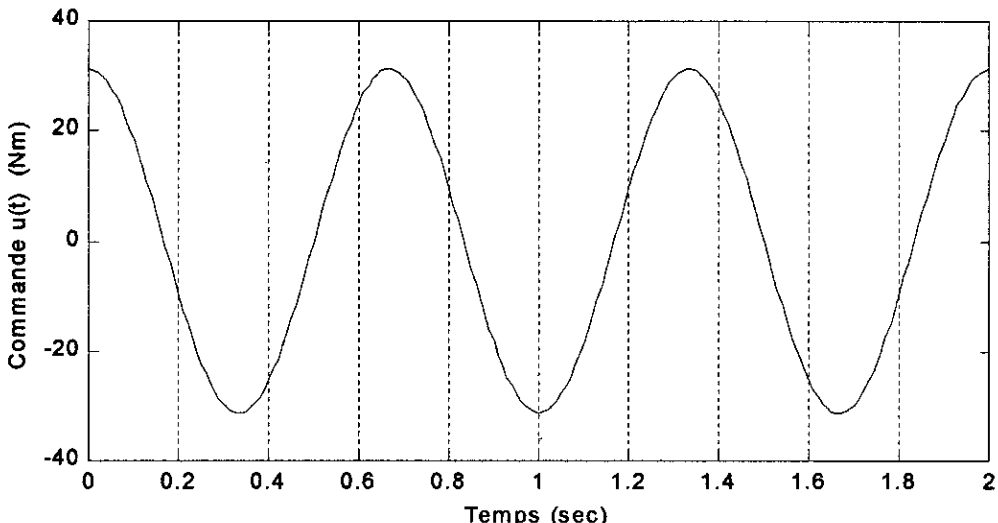
Figure(II.17) Commande développée par le régulateur



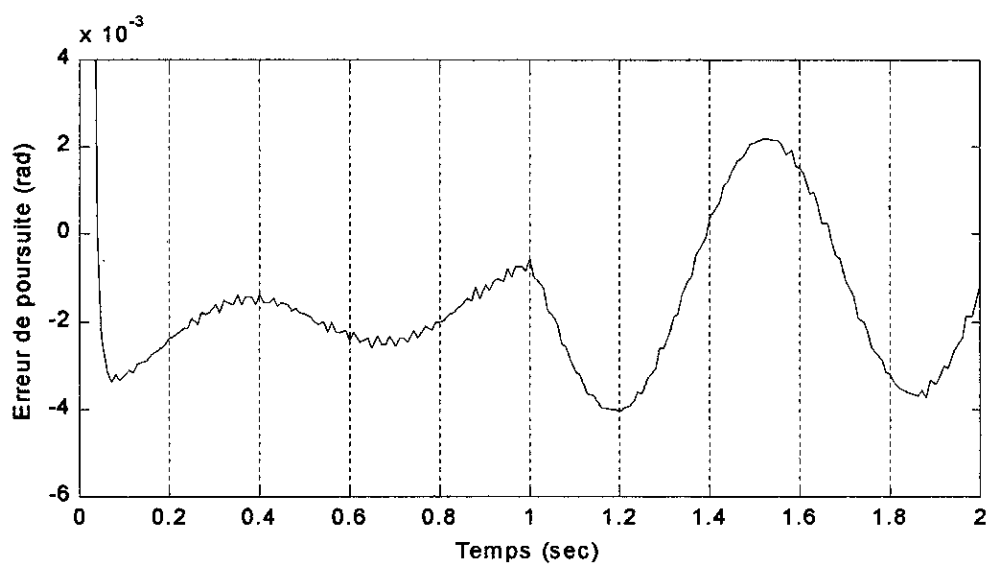
Figure(II.18) Erreur de poursuite



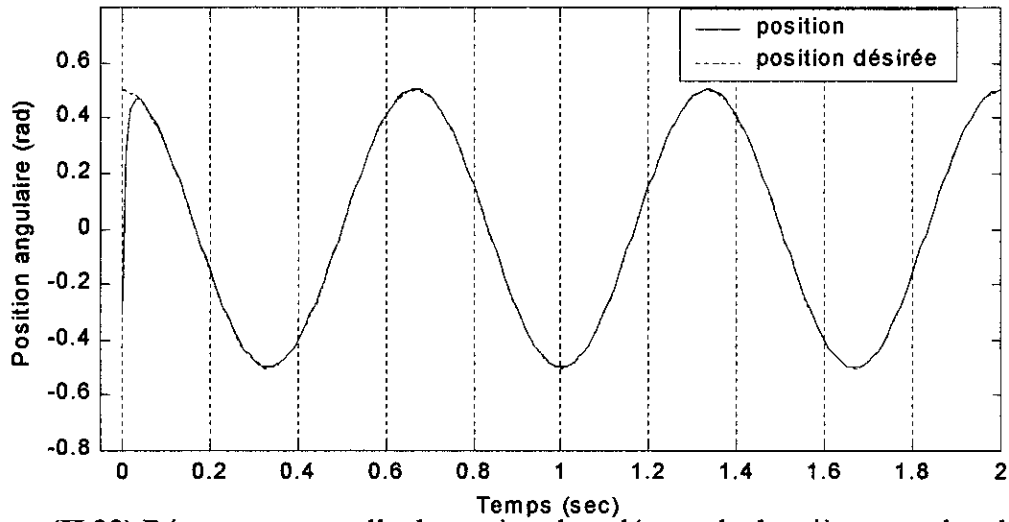
Figure(II.19) Réponse temporelle du système bouclé pour le premier test de robustesse



Figure(II.20) Commande développée par le régulateur pour le premier test de robustesse



Figure(II.21) : Erreur de poursuite pour le premier test de robustesse



Figure(II.22) Réponse temporelle du système bouclé pour le deuxième test de robustesse

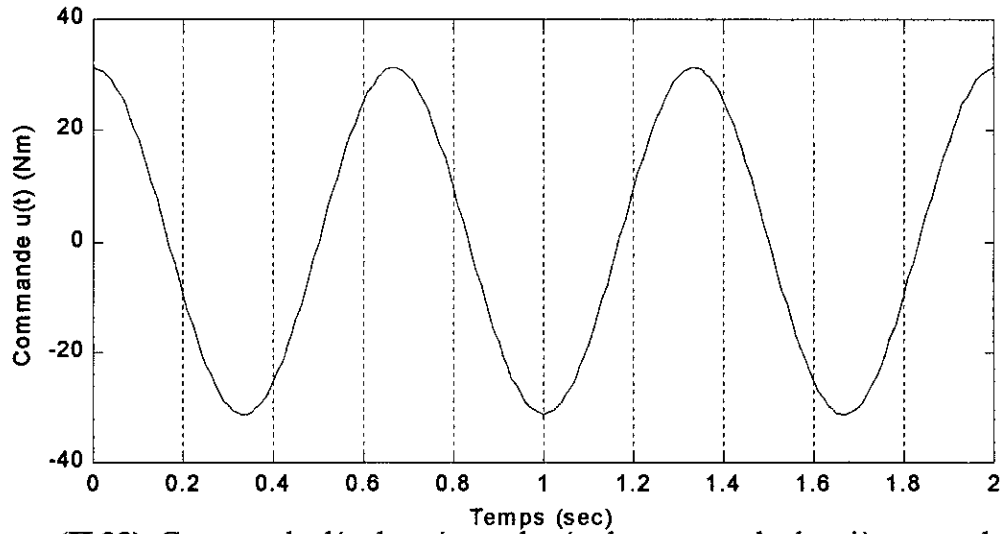


figure (II.23) Commande développée par le régulateur pour le deuxième test de robustesse

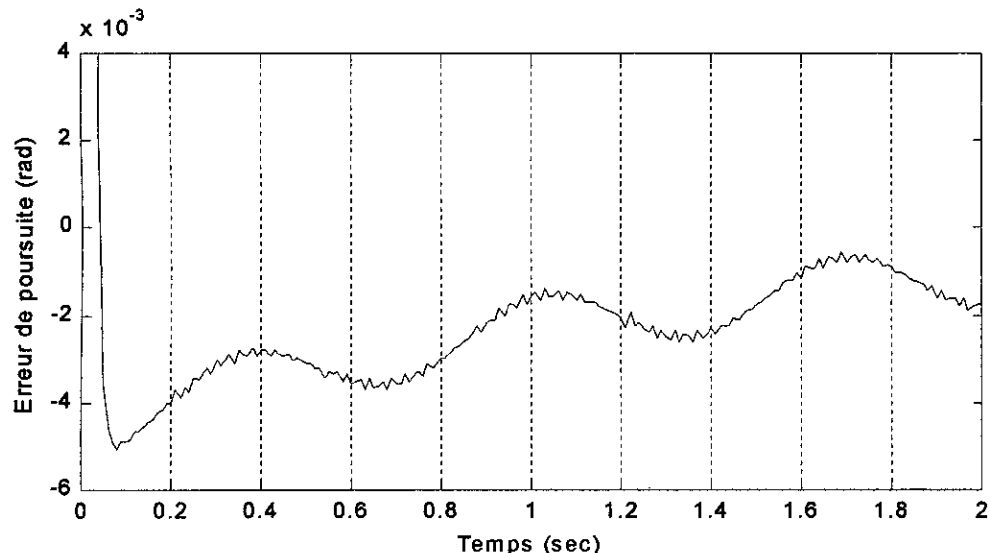


figure (II.24) Erreur de poursuite pour le deuxième test de robustesse

II.3.3 Conclusion

Dans cette partie, nous avons proposé une approche pour la détermination des gains du retour d'état à l'aide des Algorithmes Génétiques. Nous avons pu synthétiser un réglage par retour d'état pour un système non linéaire sans recourir à la linéarisation, alors que d'autres méthodes de synthèse l'exigent comme le placement de pôles.

Cette technique, testée pour la régulation d'un pendule inverse, aboutit à de bonnes performances. Le pendule suit la trajectoire désirée avec une très bonne précisions, l'erreur de poursuite est négligeable de l'ordre de 10^{-3} rad avec une commande acceptable. Les tests de robustesse que nous avons effectués ont prouvé l'aptitude de cette approche à donner une bonne robustesse au régulateur vis a vis des variations paramétriques du système, car les paramètres du régulateur étant optimaux une variation des paramètres du système ou des conditions initiales n'affecte pas les performances.

Chapitre III

Optimisation des paramètres de la loi CDSV par Algorithme Génétique

III.1 Commande décentralisée à structure variable

Ce sont les travaux de recherche effectuée par Phillipov en 1960, sur les équations à second membre discontinu qui ont donné naissance à l'idée de système de commande à structure variable. Par la suite plusieurs travaux ont été effectués par l'équipe du professeur S.V Emelyanov et publié en 1962 et par le professeur Utkin en 1978, sur cette classe particulière de système de commande. Cette commande consiste à amener la trajectoire d'état du système bouclé vers une surface de glissement et à la faire commuter à l'aide d'une logique de commutation autour de la surface jusqu'au point d'équilibre (régime glissant réel)

Les systèmes de commande à structure variable sont des systèmes non-linéaires discontinus où la structure de commande varie entre deux valeurs, la commutation se fait suivant le signe d'une certaine fonction non-linéaire appelée surface de glissement, de façon à réduire l'ordre de ce système et à forcer le point représentatif de son mouvement à rester sur cette surface d'où le nom de régime glissant idéal, ceci dans le but d'obtenir une meilleure stabilité et précision que celle obtenu généralement avec les méthodes classiques.[12]

La commande à structure variable n'affecte plus ce nouveau régime qui est commandé uniquement par une commande appelée commande équivalente. Cette dernière est indépendante de la variation des paramètres et des perturbations d'où l'invariance de la dynamique du système dans le mode de glissement (qui est gouverné uniquement par le choix des coefficients de la surface de glissement), le système commandé est alors complètement insensible aux paramètres incertains et aux perturbations.

La commande décentralisée est utilisée pour simplifier la synthèse des commandes surtout pour les systèmes dynamiques non-linéaires. Des recherches faites par des spécialistes

confirment que la commande décentralisée satisfait plus de contraintes que la commande centralisée, à savoir : [13]

- La réduction du coût de l'implémentation des lois de commande.
- La minimisation du taux d'information utilisée pour la génération des lois de commande et d'autres types de contraintes selon la nature du système.
- L'augmentation de la fiabilité des contrôleurs.

L'objectif de la commande décentralisée est de synthétiser pour chaque sous-système (le système global est décomposé en n sous-systèmes) une loi de commande locale en se basant sur les informations locales du sous-système. Le plus important problème de cette technique est d'assurer la stabilité et les performances du système global.

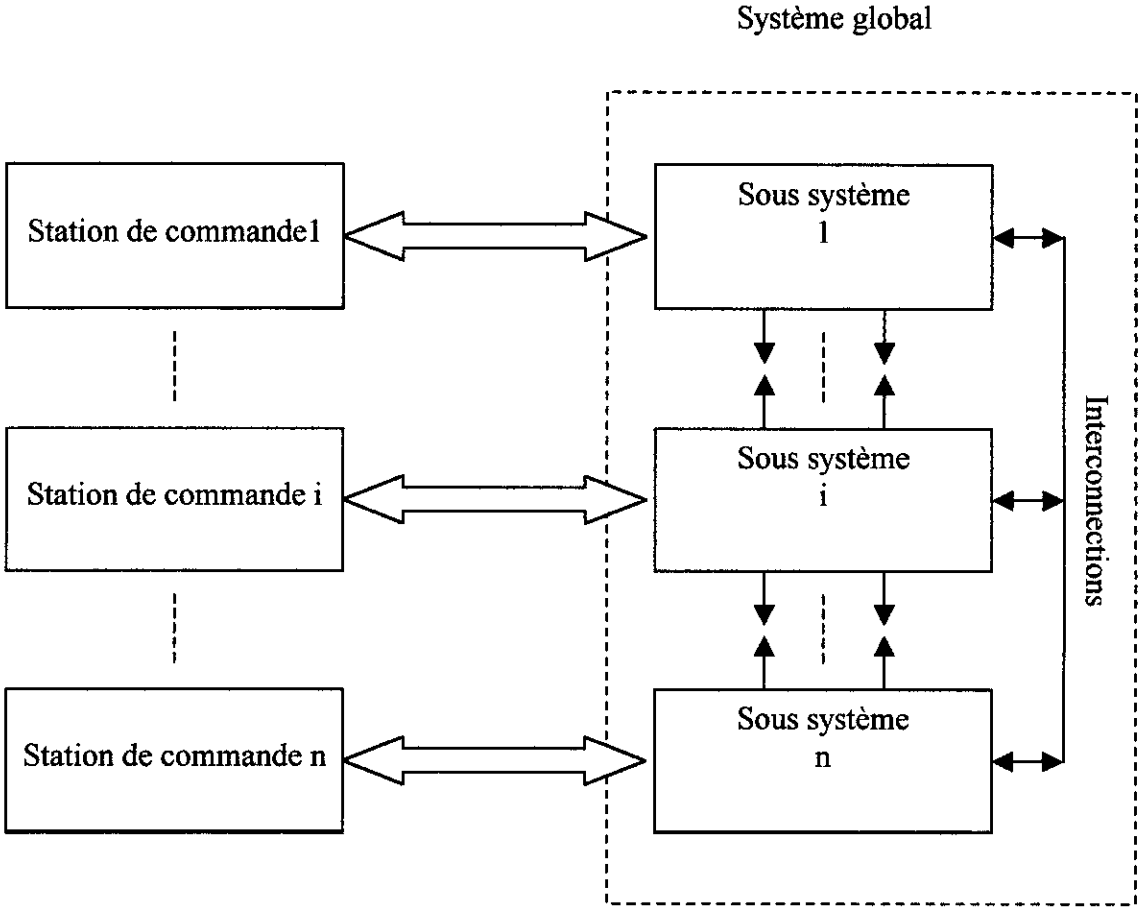


Figure (III.1) Structure de la commande décentralisée d'un système interconnecté

Une technique de commande décentralisée à structure variable (CDSV) a été développée à l'issue des travaux de Madani [26]. Elle est appliquée au modèle du robot manipulateur PUMA560.

Le modèle utilisé est non-linéaire fortement interconnecté entre ces sous-systèmes. La synthèse de cette commande pour chaque sous-système est effectuée en trois parties :

- Sélection d'une surface de glissement locale stable avec une forme générale non -- linéaire et variable dans le temps
- Construction de la commande décentralisée à structure variable.
- Introduction d'un secteur de glissement pour remédier au problème de la discontinuité de la commande locale.

Les paramètres de la surface de glissement et de la commande décentralisée sont optimisés par l'utilisation des Algorithmes Génétiques.

III.2 Conception de la loi CDSV pour le robot manipulateur PUMA 560 par Algorithme Génétique

Jusqu'à maintenant, nous avons employé les algorithmes génétiques pour la détermination d'un petit ensemble de paramètres. Il est plus intéressant de voir l'efficacité des AGs pour des problèmes plus complexes.

Dans cette partie, nous allons utiliser les AGs pour la détermination de l'ensemble de paramètre de la CDSV. C'est un ensemble très grand, il comporte les paramètres des surfaces de glissement locales et des régulateurs décentralisés. Pour examiner les performances de cette approche, nous avons fait plusieurs essais pour optimiser les paramètres de la CDSV par AG pour la commande du robot manipulateur PUMA 560 en considérant seulement les trois premières articulations.

Dans un premier temps, on fixe les paramètres de la surface de glissement donnés dans [26] et on détermine le reste des paramètres en utilisant deux types de critères, un critère qui est la somme quadratique des erreurs de trajectoires et le deuxième, est une pondération entre la somme quadratique des erreurs et des commandes.

Ensuite, on détermine tous les paramètres y compris les paramètres des surfaces de glissement locales

Dans ce type de problème le choix du critère à minimiser dépend des performances désirer. Une étude empirique est faite pour comparer l'influence du critère sur les performances.

III.2.1 modèle dynamique du robot manipulateur PUMA 560

Dans cette application, on considère seulement les trois articulations essentielles du robot manipulateur PUMA 560. Les trois autres articulations sont simplement maintenues à la position zéro. Le modèle dynamique des trois premières articulations de ce robot est donné par l'équation :

$$M(q)\ddot{q}+B(q,\dot{q})\dot{q}+G(q)=u-mJ^{T}(q)\left[J(q)\ddot{q}+\dot{J}(q,\dot{q})\dot{q}+g\right] \qquad \text{(III.1)}$$

avec $q^{T}=[q_1,q_2,q_3]$ positions angulaires des trois articulations, m est la charge utile et $g^{T}=[0\ \ 0\ \ 9.81]$

les matrices M, B, G et J seront données dans l'annexe.

III.2.2 La commande décentralisée à structure variable

L'objectif de la commande décentralisée à structure variable est de générer une commande locale $u_i(t)$ du sous-système i afin d'assurer le maintien du régime de glissement local. D'autre part, la trajectoire d'état $q_i(t)$ de chaque sous-système suit la trajectoire d'état désirer $q_{id}(t)$. La dynamique de l'erreur de poursuit $\tilde{q}_i(t)=q_{id}(t)-q_i(t)$ dépend de la surface de glissement locale σ_i. Cette dernière doit être choisit d'une façon telle que dans le régime glissant idéal, l'erreur de poursuite converge vers zéro. Autrement dit, la surface de glissement doit être asymptotiquement stable.

L'expression de la commande locale $u_i(t)$ correspondante a l'articulation i est donnée par l'équation (III.2) [26]

$$u_i(t)=\sum_{j=0}^{2}\Psi_{ij}^{0}(t).\,q_{di}^{(j)}+\sum_{k=0}^{1}\Phi_{ik}^{0}(t).\,q_i^{(j)}+\gamma_i^{0}(t)$$

$$+\mathrm{sgn}\,(\sigma_i).\left\{\sum_{j=0}^{2}\hat{\Psi}_{ij}\left|q_{di}^{(j)}\right|+\sum_{k=0}^{1}\hat{\Phi}_{ik}\left|q_i^{(j)}\right|+\hat{\gamma}_i\right\} \qquad \text{(III.2)}$$

avec : $i = 1,2,3 \quad j = 0,1,2$ et $k = 0,1$

$$\Psi_{ij}^{0}(t)=\overline{\Psi}_{ij}+X_{ij}.q_{di}^{(j)}.\sigma_i \qquad \text{(III.3)}$$

$$\Phi_{ik}^{0}(t)=\overline{\Phi}_{ik}+\rho_{ik}.q_{di}^{(k)}.\sigma_i \qquad \text{(III.4)}$$

$$\gamma_i^{0}(t)=\overline{\gamma}_i+\mu_i\sigma_i \qquad \text{(III.5)}$$

$$\sigma_i=\dot{\tilde{q}}+A_{i1}(\tilde{q})+A_{i2}(z_i)-\dot{\tilde{q}}_i(0)-A_{i1}(\tilde{q}_i(0)) \qquad \text{(III.6)}$$

$$z_i=\int_0^t\tilde{q}_i(\tau)\,d\tau \quad \text{où} \quad \tilde{q}_i(t)=q_{di}(t)-q_i(t) \qquad \text{(III.7)}$$

$$A_{i1}(\tilde{q}_i)=\lambda_i.\tilde{q}_i+\alpha_i\tanh(\beta_i.\tilde{q}_i) \qquad \text{(III.8)}$$

$$A_{i2}(z_i)=\eta_i.z_i(1+z_i^2) \qquad \text{(III.9)}$$

Les paramètres et les constantes de la commande : $\overline{\psi}_{ij}, \hat{\psi}_{ij}, \overline{\Phi}_{ik}, \hat{\Phi}_{ik}, \overline{\gamma}_i, \hat{\gamma}_i$ et X_{ij}, ρ_{ik}, μ_i avec les paramètres de la surface de glissement $\lambda_i, \alpha_i, \beta i, \eta_i$ sont à déterminer par l'Algorithme Génétique. Sachant que j = 0,1,2 et k = 0,1 il y aura 22 paramètres à déterminer pour chaque articulation

Le schéma de principe de cette commande locale $u_i(t)$ est représenté sur la figure(III.2). Elle comporte deux parties :

- La première partie caractérise un changement de structure par l'intervention directe de la grandeur de consigne : position, vitesse et accélération (PD^2 feedforward), un retour d'état variable (PD feedback) et un signale variable.
- La deuxième partie est le produit du singe de la surface de glissement locale avec l'ensemble de la valeur absolue des grandeurs de consigne : position, vitesse et accélération (PD^2 ABS feedforward), un retour de la valeur absolue du vecteur d'état (PD ABS feedback) plus un signale constant.

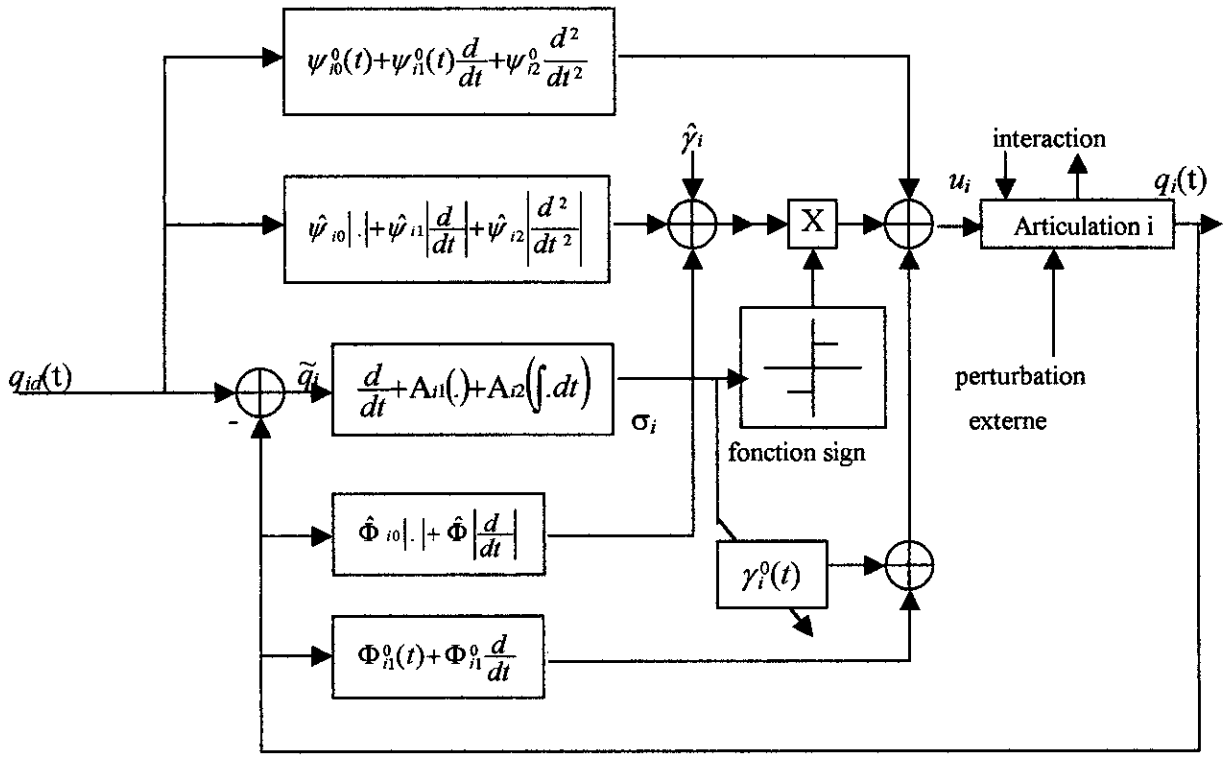


Figure (III.2) : Schéma de la commande décentralisé à structure variable

III.2.3 Introduction du secteur de glissement

La commande CDSV est discontinue autour de la surface de glissement locale, elle conduit vers une commande forte avec des variations brusque de haute fréquence. Ce phénomène est indésirable en pratique. Pour y remédier, on introduit une couche bornée prés de la surface de glissement en remplaçant la fonction sgn(σ_i) utilisée dans la commande locale par une fonction d'approximation continue sous forme sigmoïde qui est variable par rapport à l'erreur de poursuit, donnée par l'équation : [26]

$$M(\sigma_i)=\frac{\sigma_i}{|\sigma_i|+\delta_{i0}+\delta_{i1}|\tilde{q}_i|} \qquad \text{(III.10)}$$

Avec : δ_{i0} une petite constante positive, δ_{i1} une grande constante positive.

III.3 Paramètres de l'Algorithme Génétique

Les éléments de l'AG sont les suivants :

- Le codage des paramètres et un codage réel.
  - $\overline{\psi}_{ij}, \hat{\psi}_{ij}, \hat{\Phi}_{ik}, \bar{\gamma}_i, \hat{\gamma}_i \; X_{ij}, \rho_{ik}$ et μ_i : appartiennent à l'intervalle [0, 30],
  - $\overline{\Phi}_{ik}$: appartiennent à l'intervalle [-30, 0],
  - $\lambda_i, \alpha_i, \beta_i$ et η_i : appartiennent à l'intervalle [0, 50],
- Les opérateurs de l'AG sont :
  - Croisement en un point
  - Sélection par roulette de casino.
  - Mutation adaptative donnée par l'équation :

$$pm = \left(1 - \frac{t}{ng}\right)^2 \qquad \text{(III.11)}$$

- Le critère à minimiser est :

$$J = \sum_{i=1}^{3} \sum_{j=0}^{n} \alpha_i . e(j)^2 + \beta_i . u(j)^2 , \quad \alpha_i, \beta_i \in [0,1] \qquad \text{(III.12)}$$

La population contient 20 individus et le nombre de générations maximal est de 30.

Remarques :

Pour le choix de la fonction d'évaluation d'un Algorithme Génétique, on peut souvent classer les objectifs par importance mais les poids seront généralement adaptés par tâtonnement, jusqu'à l'obtention d'une solution acceptable. Le processus d'optimisation a beau être automatisé, l'utilisateur doit donc quand même optimiser à la main la définition de la fonction d'évaluation.

Les constantes α_i et β_i du critère J, sont choisis d'une manière à ramener les valeurs de la sommes quadratique des erreurs et la somme quadratique des commandes vers un même ordre de grandeur tout en satisfaisant les performances désirées.

Les singes des paramètres des surfaces de glissement et des commandes décentralisées sont choisis d'une manière à assurer la stabilité des surfaces de glissement et du système en boucle fermée. [26].

III.4 Résultats de simulation

La trajectoire de référence choisie pour chaque articulation du robot PUMA 560 est de type cycloïde [31]. Le but est de faire varier la position $q_i(t)$ de l'articulation I de la position initiale $q_{di}(0)$ jusqu'à la position finale $q_{di}(t_f)$ dans un temps de mouvement égal à t_f suivant une trajectoire définie par la l'équation (III.13). Cette trajectoire assure une continuité de la position, la vitesse et l'accélération désirées, et les vitesses de départ et d'arrivée des articulations sont nulles.

$$q_{di}(t)=\begin{cases} q_{di}(0)+\dfrac{\Delta_i}{2\pi}\left(2\pi\dfrac{t}{t_f}-\sin\left(2\pi\dfrac{t}{t_f}\right)\right) rad \ . si\ 0\le t\le t_f \\ q_{di}(t_f)\ rad \quad si\ t_f\le t \end{cases} \qquad \text{(III.13)}$$

avec : $\Delta_i=q_{di}(t_f)-q_{di}(0)$ le déplacement local.

Les différentes articulations se déplacent respectivement des positions {-50°, 135°, 135°} aux positions {45°, -85°, 30°} en un temps de mouvement t_f égale à 1.5 sec. Ces conditions excitent toute la dynamique de ce robot.

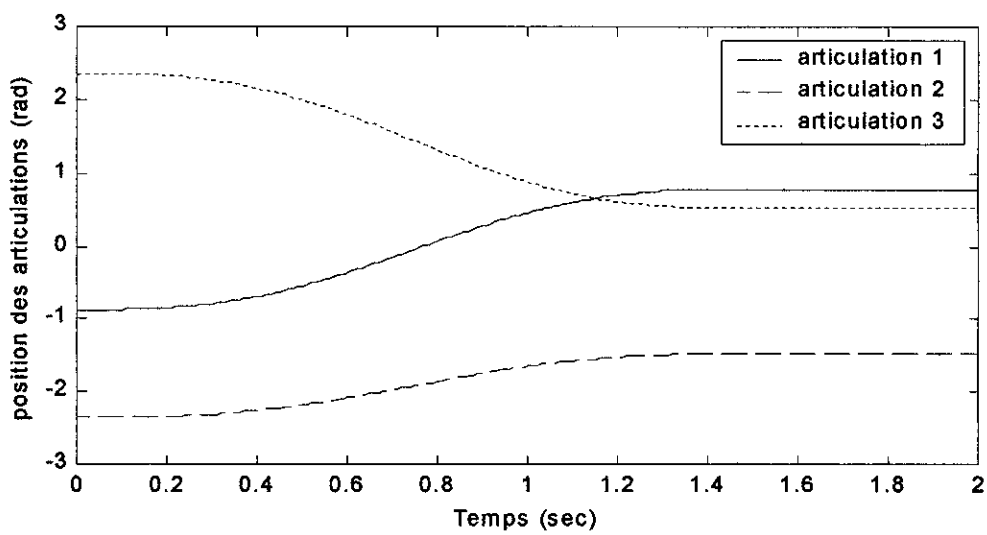


Figure (III.3) : trajectoires désirées des articulations

Toutes les simulations présentent un test de robustesse via une chute soudaine d'une masse de 5 Kg à t = 0.75 sec.

Les paramètres de la commande sont optimisés pour un secteur de glissement utilisant les paramètres : $\delta_{10},\delta_{20},\delta_{30}=0.05$ et $\delta_{11},\delta_{21},\delta_{31}=10$ de l'équation (III.10)

Pour le premier essai un critère ***J1*** uni-objectif est utilisé pour minimiser les erreurs en prenant $\beta_i = 0$, de l'équation (III.12). Les résultats obtenus sont présentés sur les figures (III6)et(III7).

Un deuxième type de critère ***J2*** et ***J3*** multi-objectif, dont les valeurs de α_i et β_i de l'équation (III.12) sont adaptées pour satisfaire les performances désirées, qui minimise les erreurs et les commandes sont utilisés respectivement pour les essais 2 et 3.

	α_1	α_2	α_3	β_1	β_2	β_3
*J*1	$5/3\ 10^{-4}$	$5/3\ 10^{-4}$	$5/3\ 10^{-4}$	0	0	0
*J*2	0.1	0.2	0.01	$2\ 10^{-12}$	$2\ 10^{-12}$	$2\ 10^{-12}$
*J*3	1	1	1	10^{-9}	10^{-9}	10^{-9}

Tableau (III.1) : Constates des critères *J*i.

Le Tableau (III.2) contient les paramètres déterminés dans [26], les résultats obtenus avec ces paramètres sont représentés sur les figure (III.4) et (III.5). En vue d'une comparaison avec nos résultats.

Le tableau (III.6) comporte la somme quadratique des erreurs et des commandes des articulations ainsi que leurs sommes pour chaque essai. A travers ce tableau nous allons évaluer les performances de notre algorithme.

paramètres	Articulations		
	i =1	i = 2	i = 3
$\overline{\psi}_{i0}$	6	10	7
$\overline{\psi}_{i1}$	8	7	4
$\overline{\psi}_{i2}$	2	4	2
$\overline{\Phi}_{i0}$	-6	-10	-7
$\overline{\Phi}_{i1}$	-4	-4	-3
$\overline{\gamma}_i$	0	0	0
$\hat{\psi}_{i0}$	5	5	6
$\hat{\psi}_{i1}$	4	5	3
$\hat{\psi}_{i2}$	1	3	1
$\hat{\Phi}_{i0}$	5	5	6
$\hat{\Phi}_{i1}$	6	4	4
$\hat{\gamma}_i$	10	30	15

Paramètres des régulateurs décentralisés (**a**)

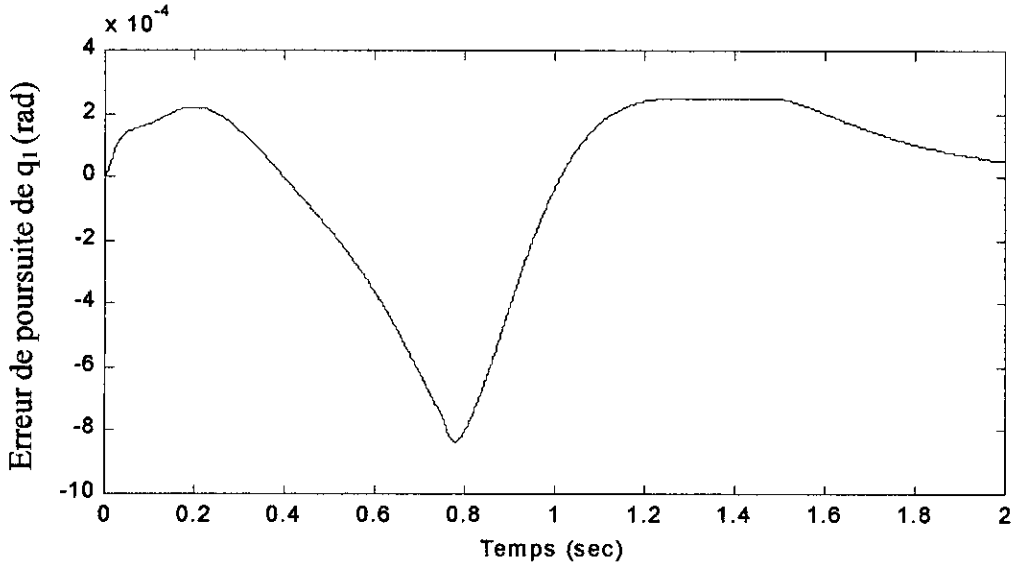
constantes	Articulations		
	i=1	i=2	i=3
X_{i0}	20	15	10
X_{i0}	20	15	10
X_{i0}	20	15	10
ρ_{i0}	20	15	10
ρ_{i1}	20	15	10
μ_i	20	15	10

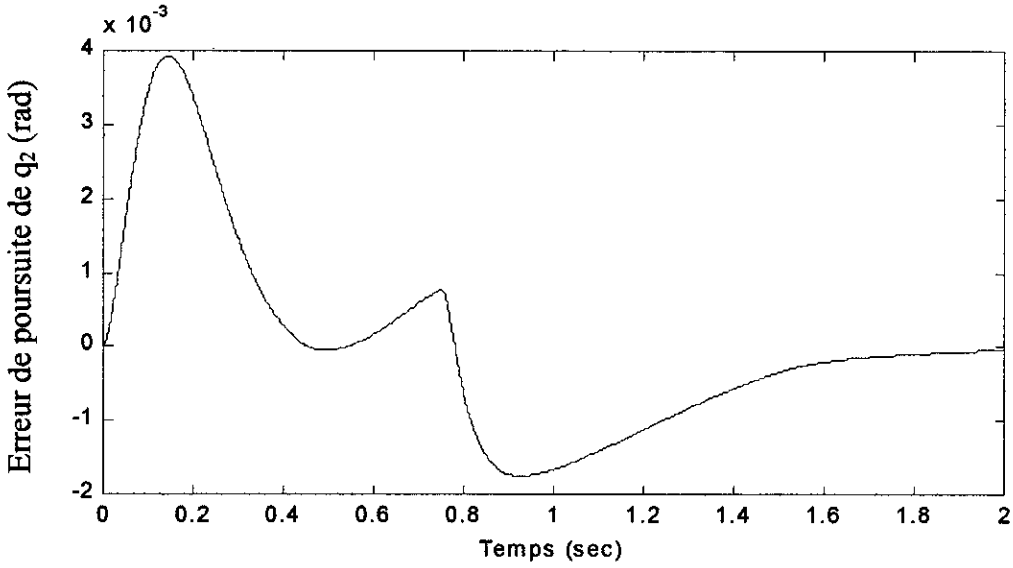
(b) Les constants X, ρ et μ

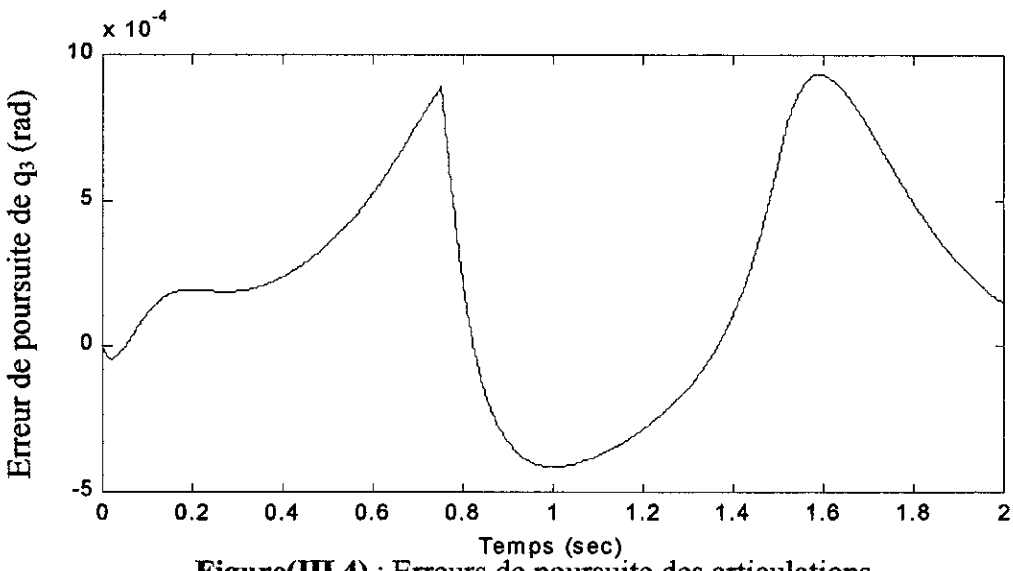
constantes	Articulations		
	i=1	i=2	i=3
λ_i	5	5	5
α_i	3	3	3
β_i	4	4	4
η_i	50	50	50

(c) Paramètres de la surface de glissement

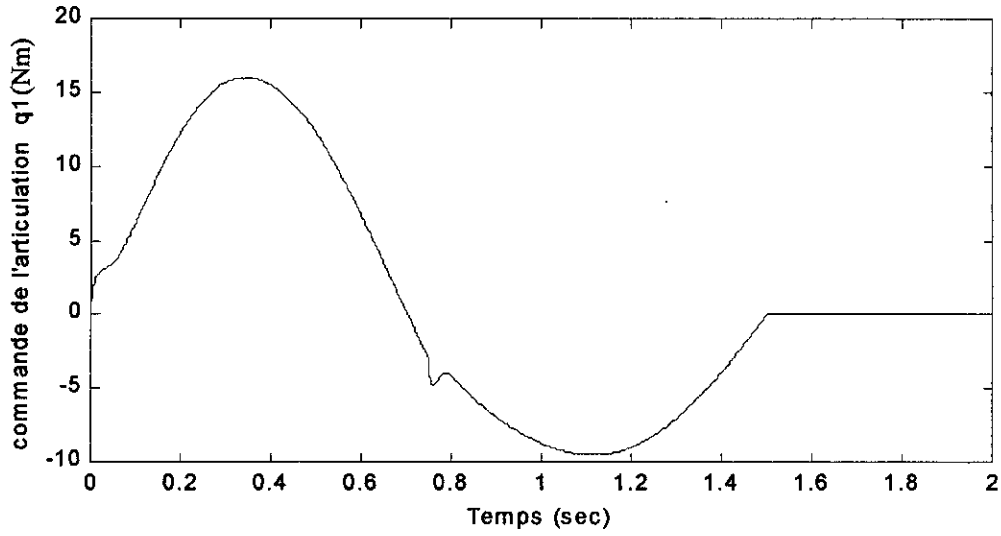
Tableau (III.2) Paramètres de la loi CDSV selon [26]

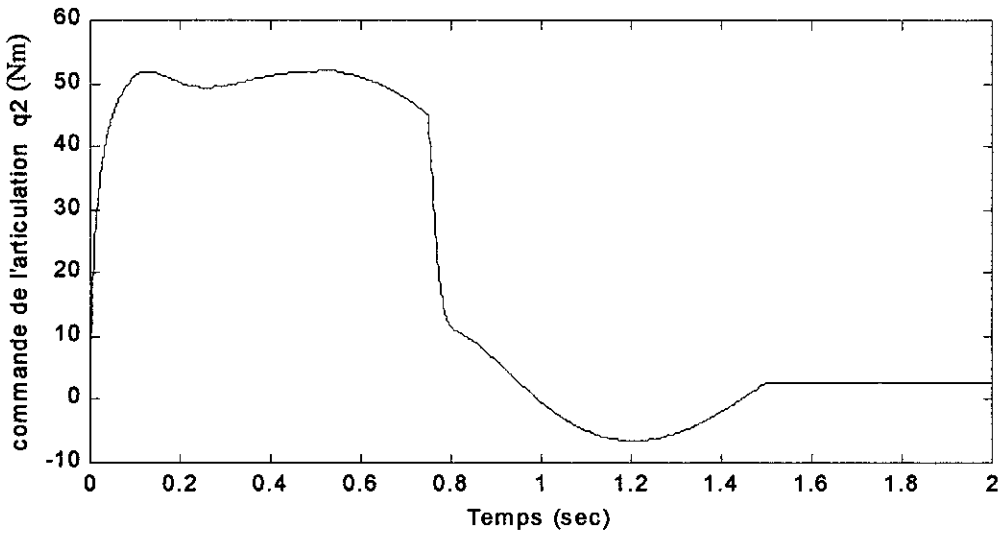






Figure(III.4) : Erreurs de poursuite des articulations





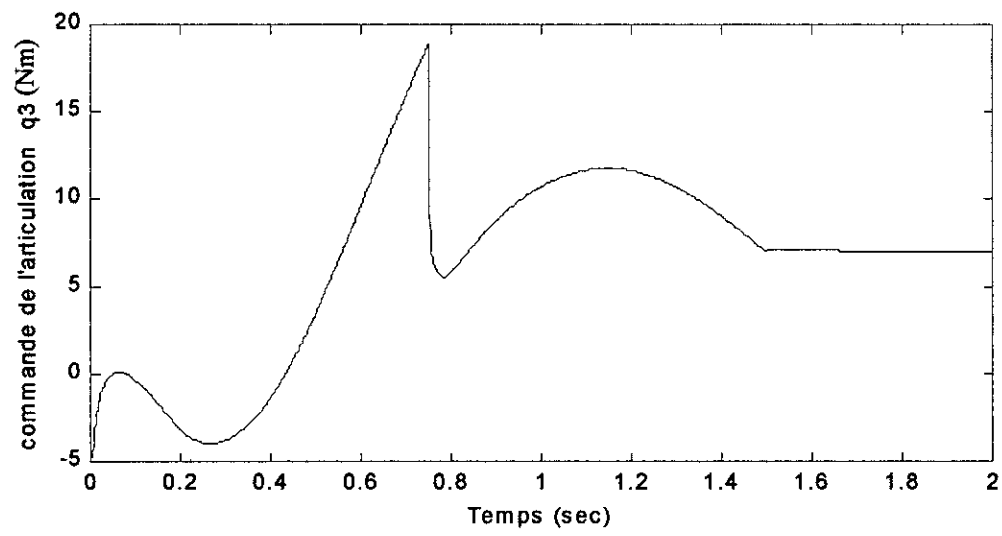


Figure (III.5) : Commandes développées aux articulations

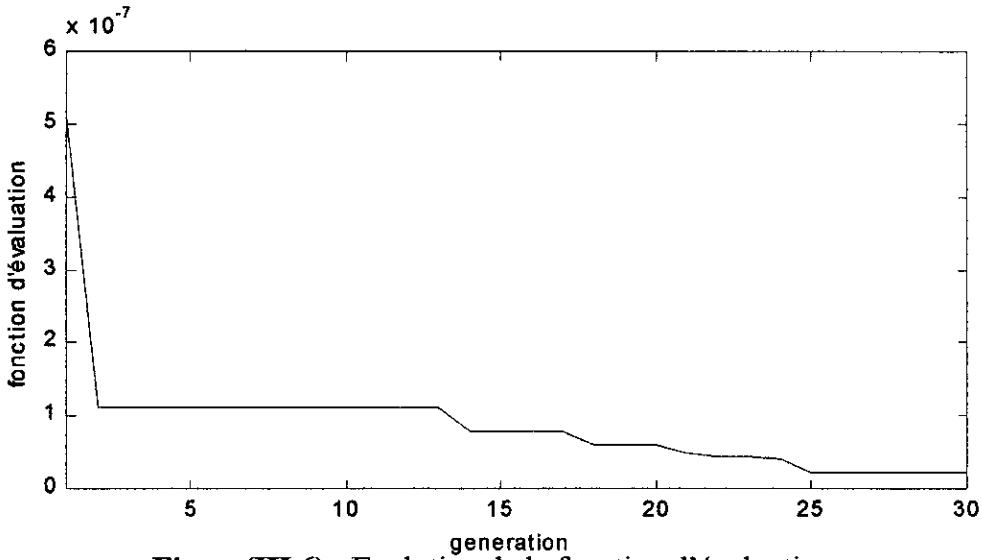
paramètres	Articulations		
	i =1	i = 2	i = 3
$\overline{\psi}_{i0}$	2.9381	16.8861	16.9998
$\overline{\psi}_{i1}$	15.6496	29.4401	19.0385
$\overline{\psi}_{i2}$	1.5282	16.4250	1.2511
$\overline{\Phi}_{i0}$	-8.6540	-11.7650	-17.3295
$\overline{\Phi}_{i1}$	-19.4951	-3.9526	-25.0818
$\overline{\gamma}_i$	2.4739	6.3445	7.3576
$\hat{\psi}_{i0}$	21.1349	25.5060	17.3180
$\hat{\psi}_{i1}$	15.0641	10.4370	25.8805
$\hat{\psi}_{i2}$	19.1018	23.9516	21.1852
$\hat{\Phi}_{i0}$	11.450	25.6755	22.2159
$\hat{\Phi}_{i1}$	27.0314	3.9110	9.3032
$\hat{\gamma}_i$	29.5227	15.1725	29.3003

(a) Paramètres des régulateurs décentralisés

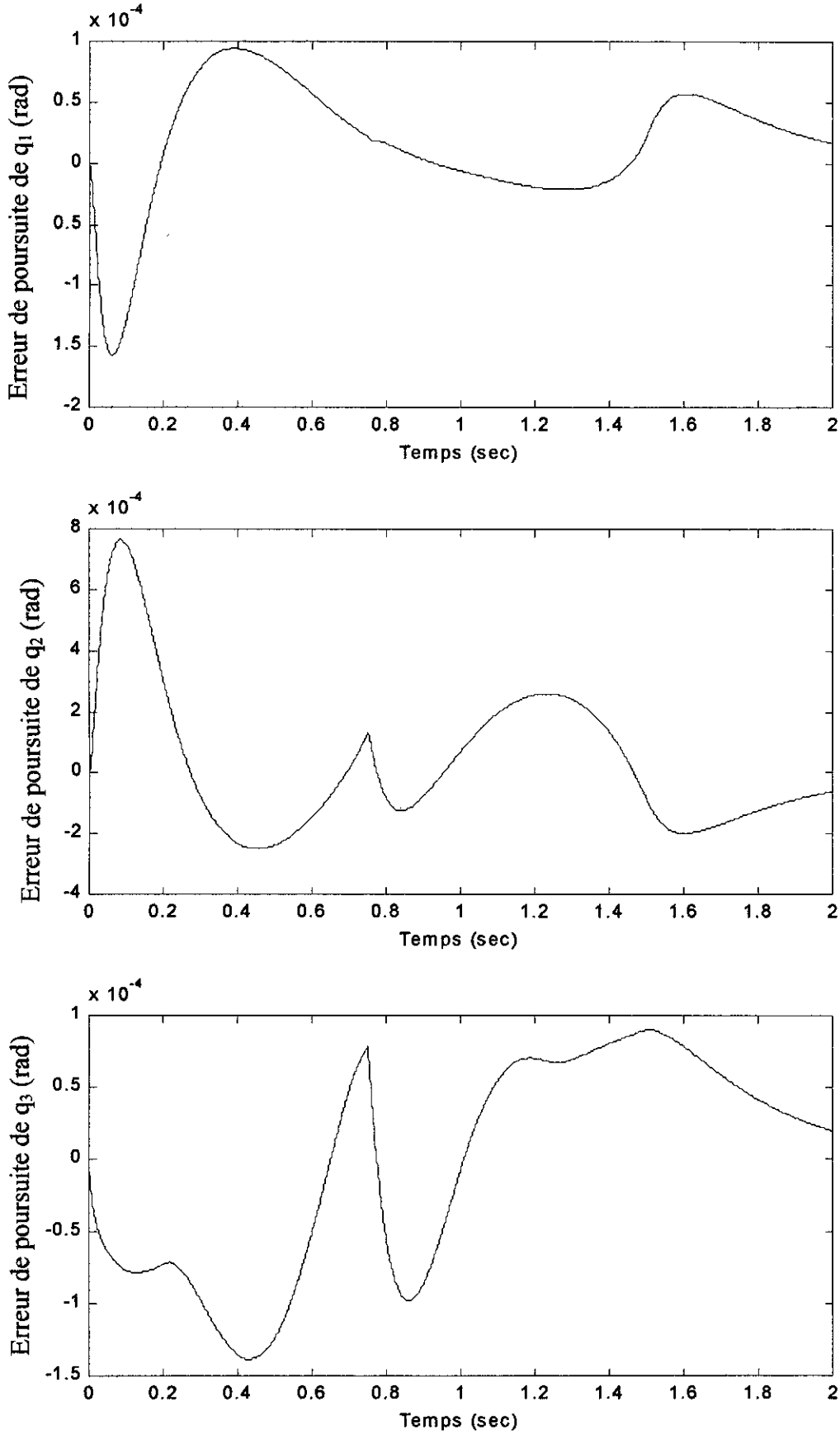
constantes	Articulations		
	i=1	i=2	i=3
X_{i0}	6.0053	27.7023	24.1579
X_{i0}	16.7332	13.2808	7.7001
X_{i0}	9.2308	5.8377	4.8854
ρ_{i0}	20.3728	22.1345	22.0636
ρ_{i1}	29.9021	11.9170	2.5788
μ_i	4.7228	13.4571	15.1657

(b) Les constants X, ρ et μ

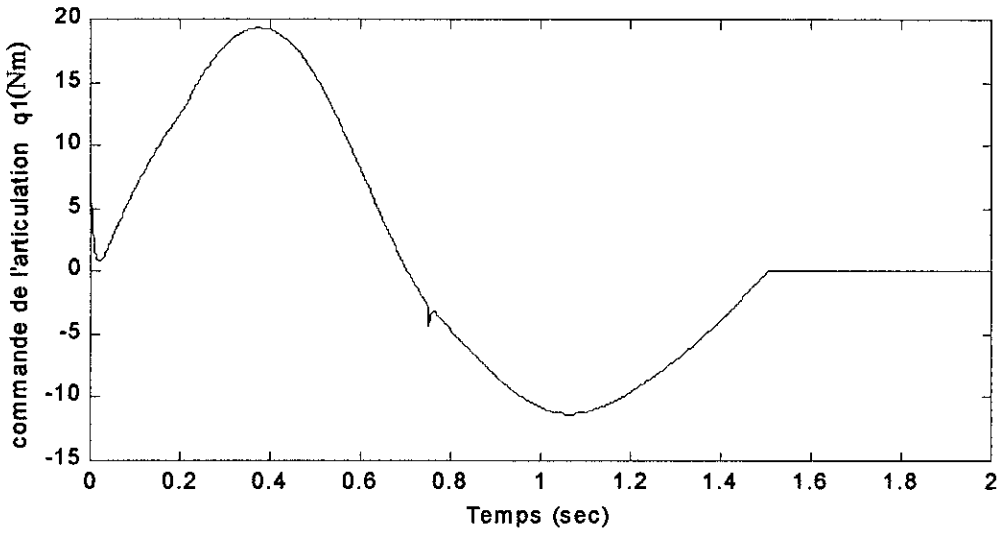
Tableau (III.3) les paramètres de la loi CDSV optimisés par l'Algorithme Génétique avec le critère $J1$

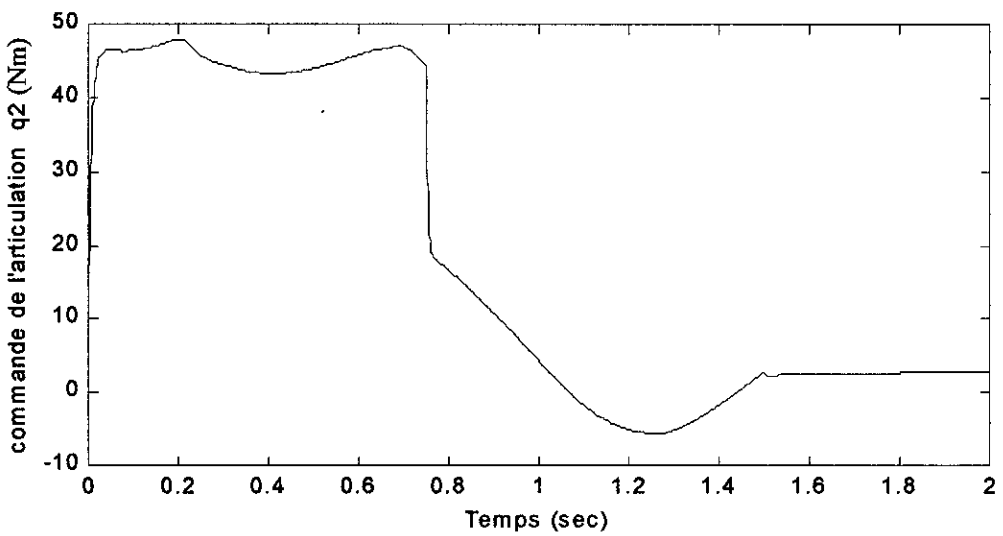


Figure(III.6) : Evolution de la fonction d'évaluation.



Figure(III.7) : Erreurs de poursuite des articulations





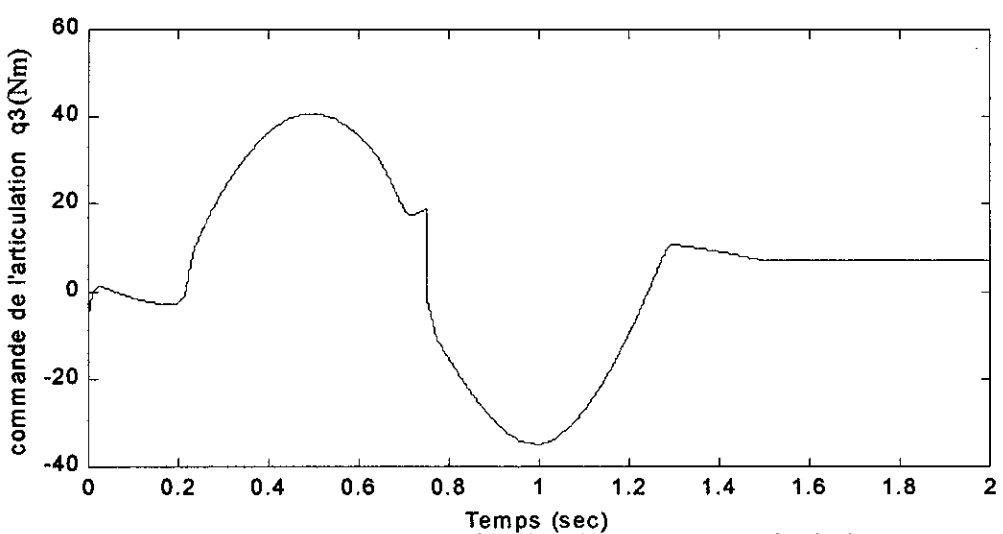


Figure (III.8) : Commandes développées aux articulations

paramètre	articulation		
	i =1	i = 2	i = 3
$\overline{\psi}_{i0}$	25.0521	24.7119	11.5638
$\overline{\psi}_{i1}$	0.9601	20.4792	17.2750
$\overline{\psi}_{i2}$	1.1473	9.9993	9.2236
$\overline{\Phi}_{i0}$	-18.9932	-27.4373	-8.4221
$\overline{\Phi}_{i1}$	-5.4154	-2.8895	-13.0781
$\overline{\gamma}_i$	10.0841	16.4213	22.8941
$\hat{\psi}_{i0}$	28.3910	24.2509	9.0700
$\hat{\psi}_{i1}$	16.4871	26.0413	4.7172
$\hat{\psi}_{i2}$	8.4409	12.2648	25.8361
$\hat{\Phi}_{i0}$	18.1696	22.1416	17.2785
$\hat{\Phi}_{i1}$	29.4200	26.2320	11.2055
$\hat{\gamma}_i$	29.4200	11.5453	27.3035

Paramètres des régulateurs décentralisés (**a**)

constantes	Articulations		
	i=1	i=2	i=3
X_{i0}	10.3723	12.3013	11.4405
X_{i0}	7.6094	1.6209	2.3426
X_{i0}	21.6433	2.0935	5.3842
ρ_{i0}	10.1810	13.8988	26.6730
ρ_{i1}	29.7489	1.9898	25.2585
μ_i	24.7510	26.1825	21.1859

(b) Les constants X, ρ et μ

Tableau (III.4) Paramètres de la loi CDSV optimisés par l'Algorithme Génétique avec critère ***J2***

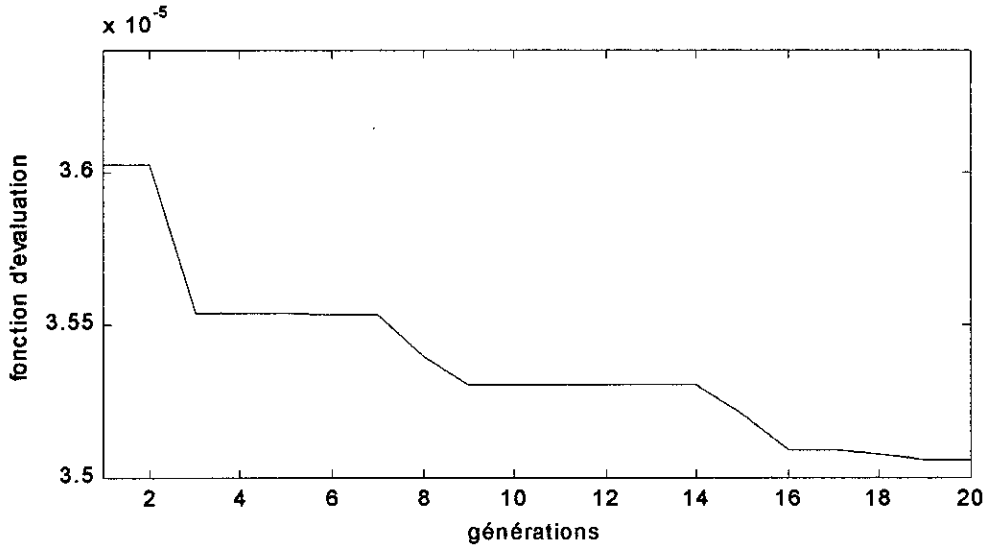
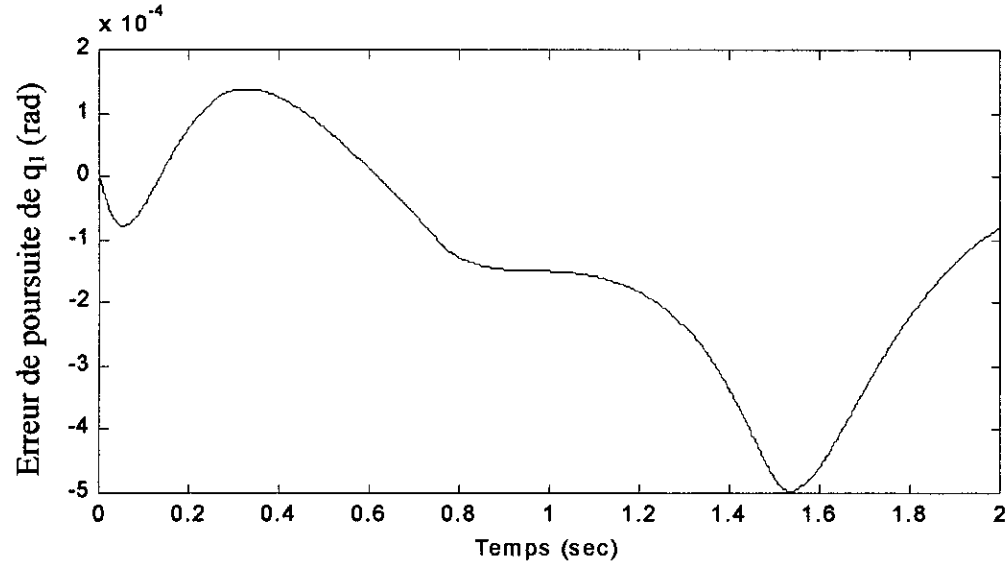
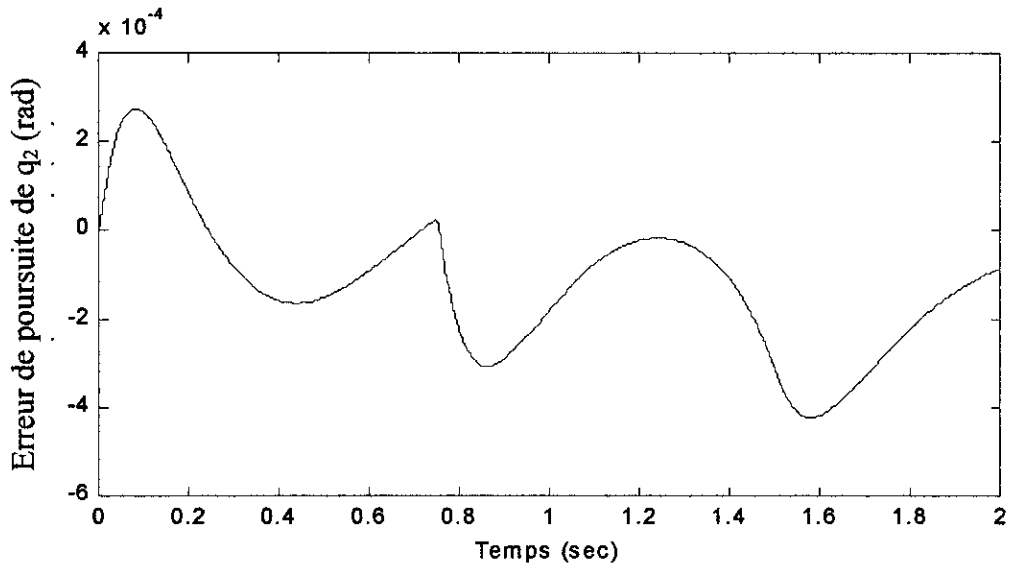
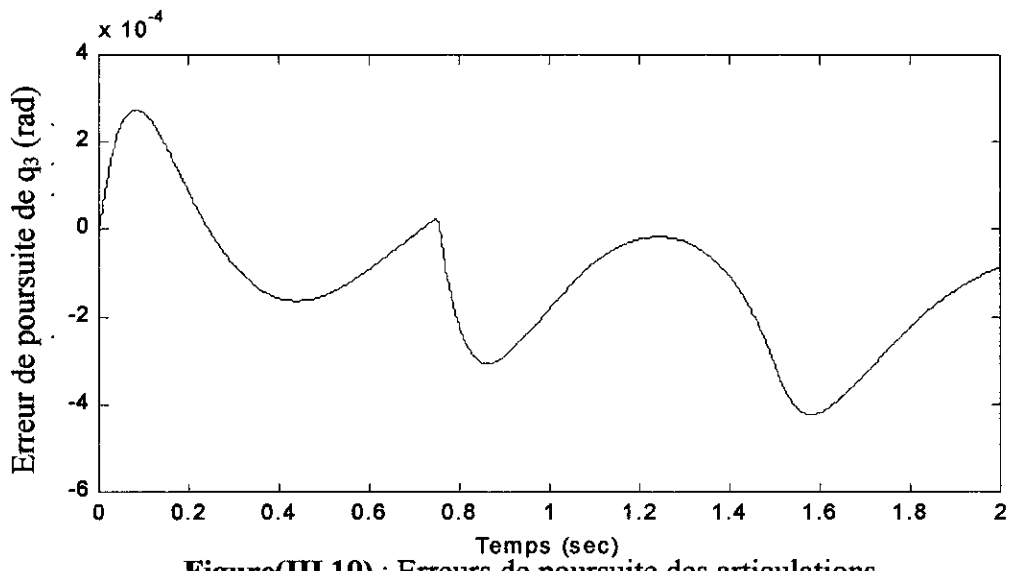


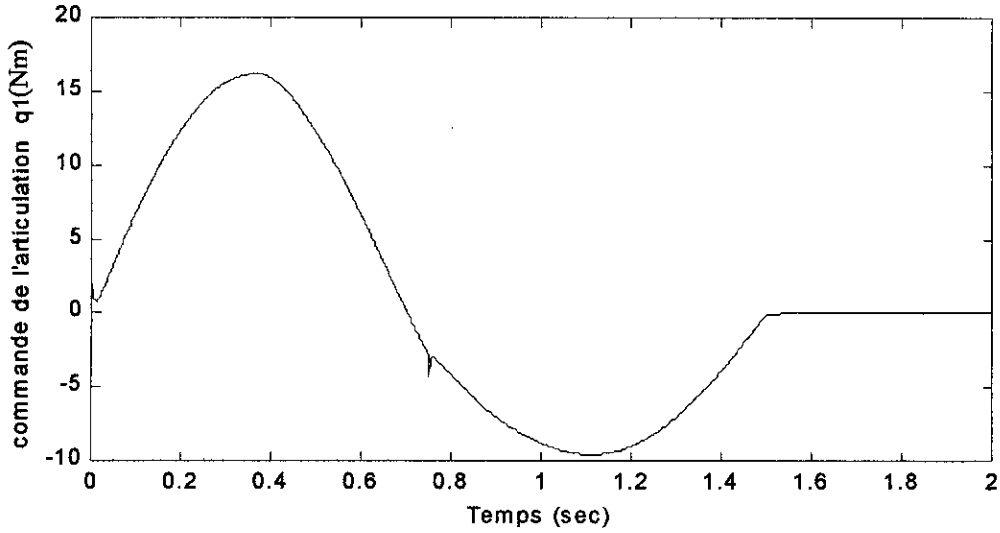
Figure (III.9) : Evolution de la fonction d'évaluation

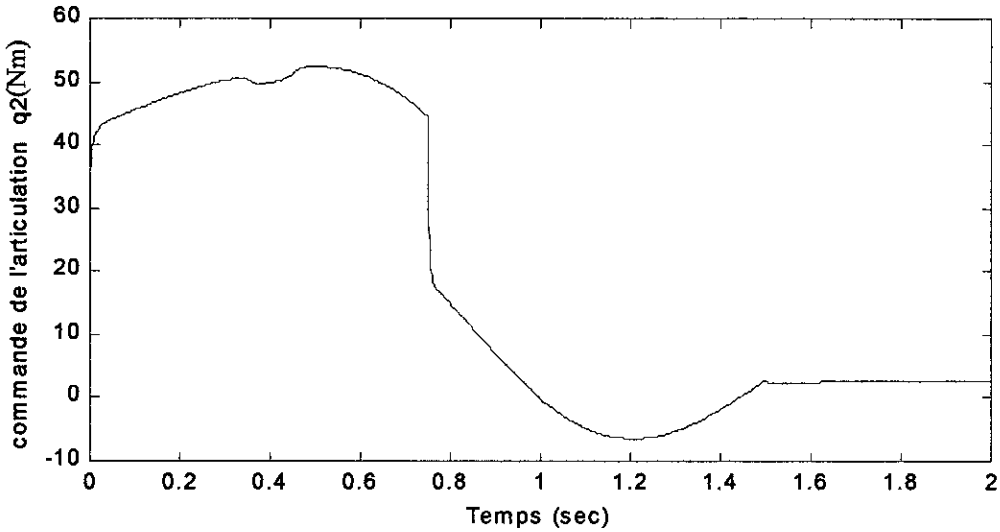






Figure(III.10) : Erreurs de poursuite des articulations





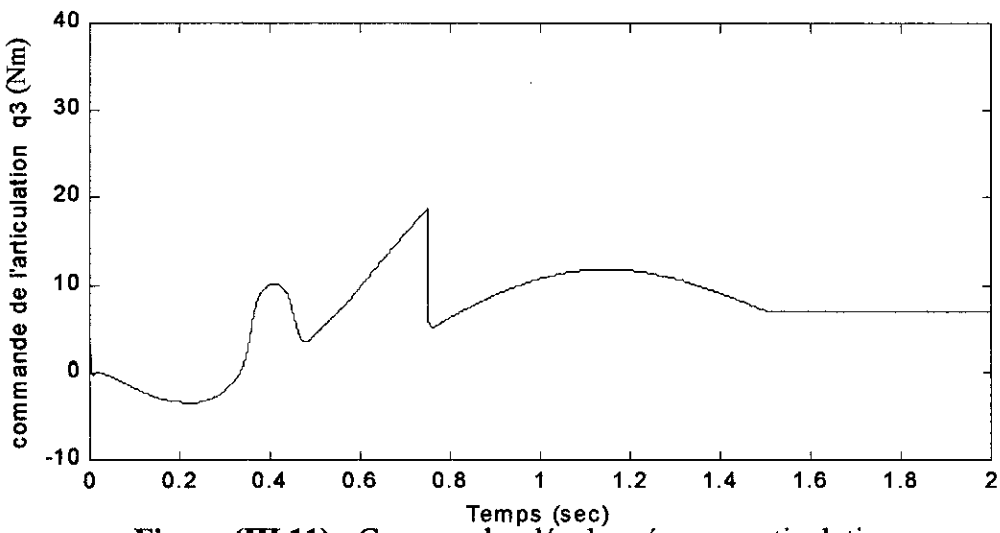


Figure (III.11) : Commandes développées aux articulations

paramètres	articulation		
	i =1	i = 2	i = 3
$\overline{\psi}_{i0}$	24.4634	3.6594	22.0900
$\overline{\psi}_{i1}$	15.8460	10.2474	15.7014
$\overline{\psi}_{i2}$	23.2730	6.8844	4.8582
$\overline{\Phi}_{i0}$	-20.1370	-14.9532	-21.8337
$\overline{\Phi}_{i1}$	-12.1608	-8.8183	-3.8333
$\overline{\gamma}_i$	13.2068	5.5234	17.9497
$\hat{\psi}_{i0}$	19.6542	21.7271	17.9497
$\hat{\psi}_{i1}$	13.1796	12.2295	22.0431
$\hat{\psi}_{i2}$	11.9077	18.3109	29.8559
$\hat{\Phi}_{i0}$	5.5121	10.0477	14.8641
$\hat{\Phi}_{i1}$	29.1391	28.6879	2.4520
$\hat{\gamma}_i$	0.3195	29.9021	27.4093

(a) Paramètres des régulateurs décentralisés

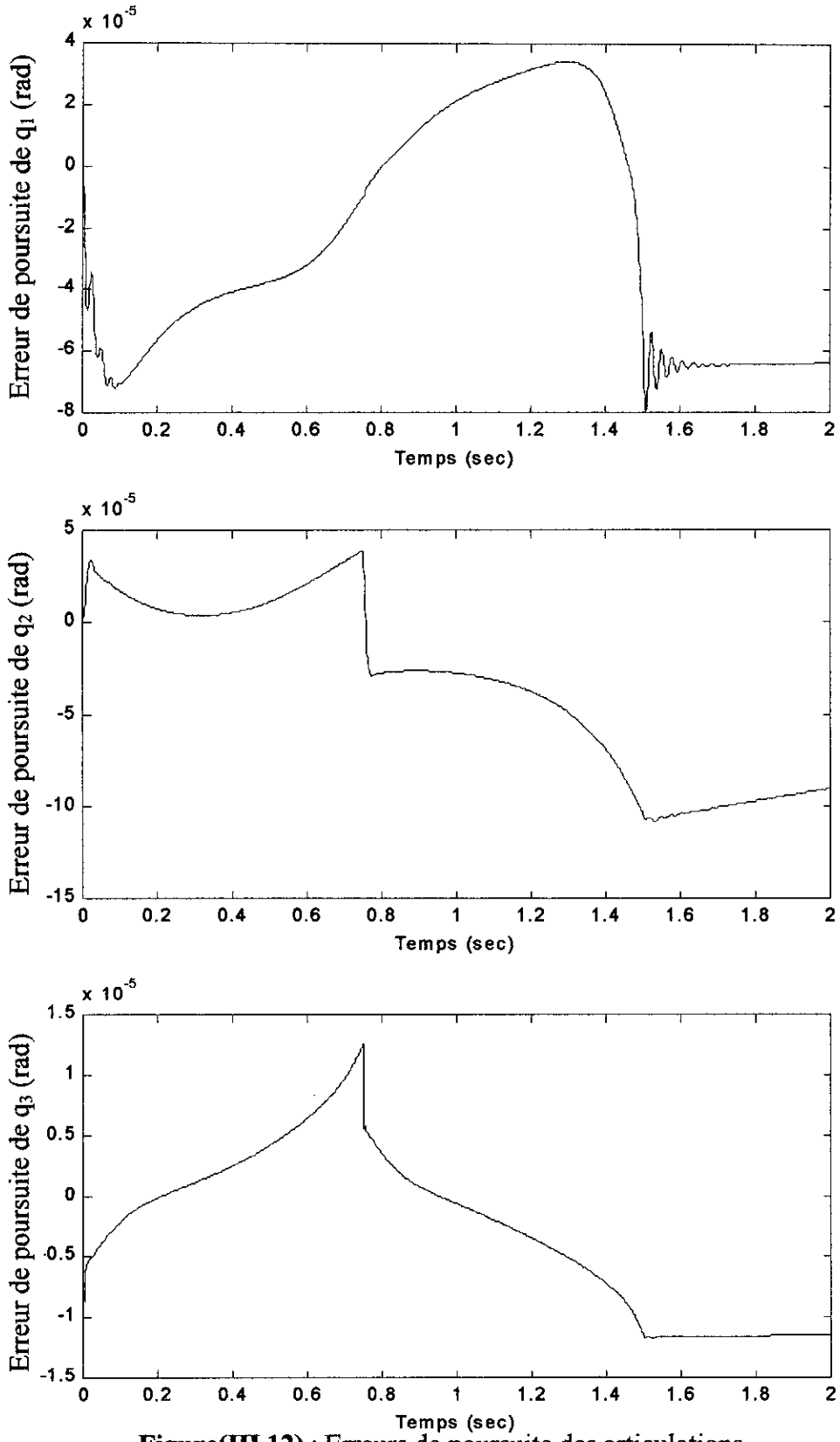
constantes	Articulations		
	i=1	i=2	i=3
X_{i0}	2.3424	20.0249	10.7051
X_{i0}	6.0082	5.2191	21.7284
X_{i0}	27.4842	16.5666	10.6851
ρ_{i0}	18.8342	27.7996	5.3662
ρ_{i1}	8.1009	27.6662	15.0264
μ_i	23.2452	20.7153	17.5393

(b) Les constants X, ρ et μ

constantes	Articulations		
	i=1	i=2	i=3
λ_i	4.5708	28.0419	24.3561
α_i	46.4287	8.1614	48.7399
β_i	31.3327	12.8221	25.7787
η_i	46.9526	46.9425	40.1572

(c) Paramètres de la surface de glissement

Tableau (III.5) les paramètres de la loi CDSV optimisés par l'Algorithme Génétique avec le critère $\boldsymbol{J3}$



Figure(III.12) : Erreurs de poursuite des articulations

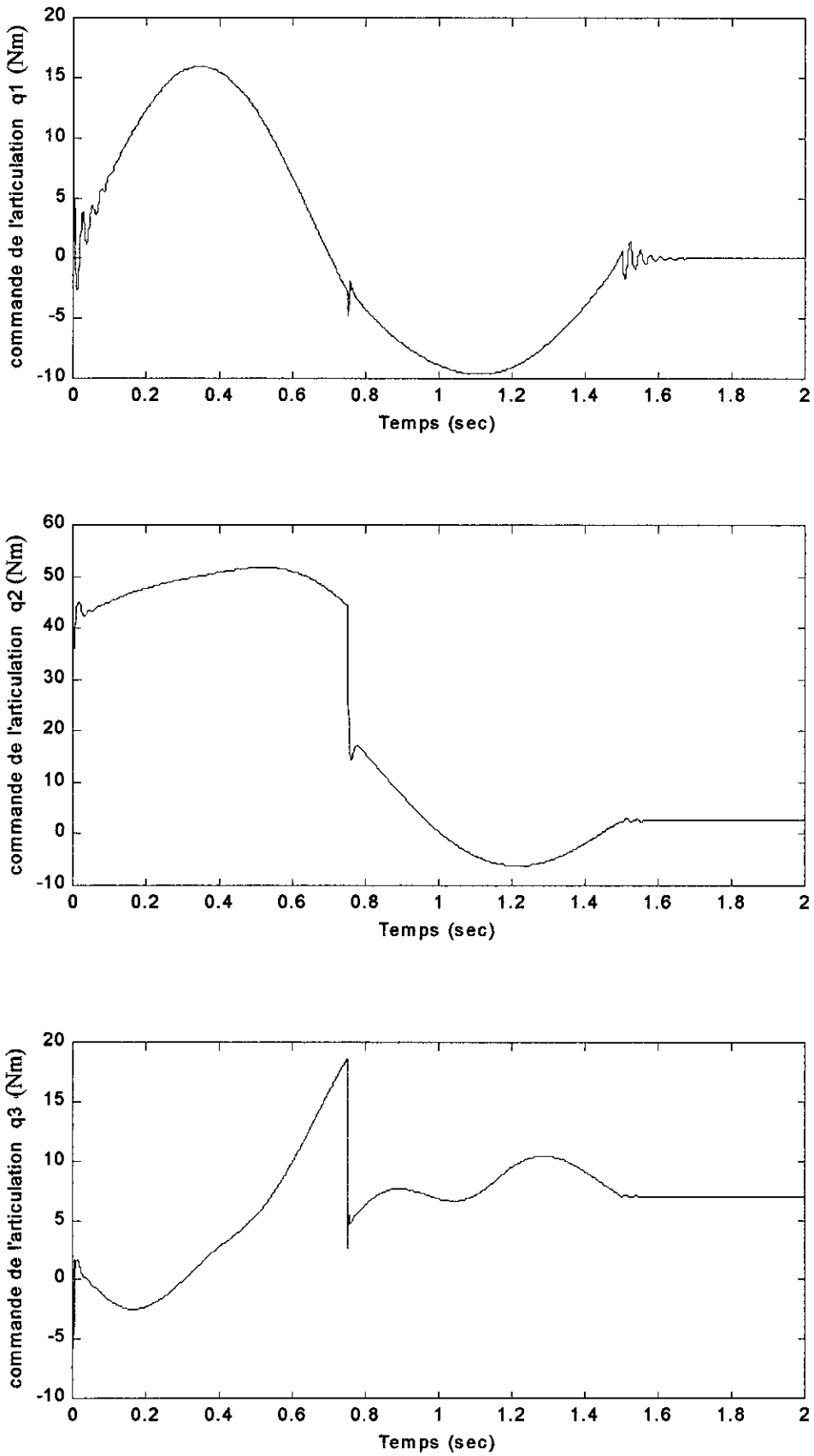
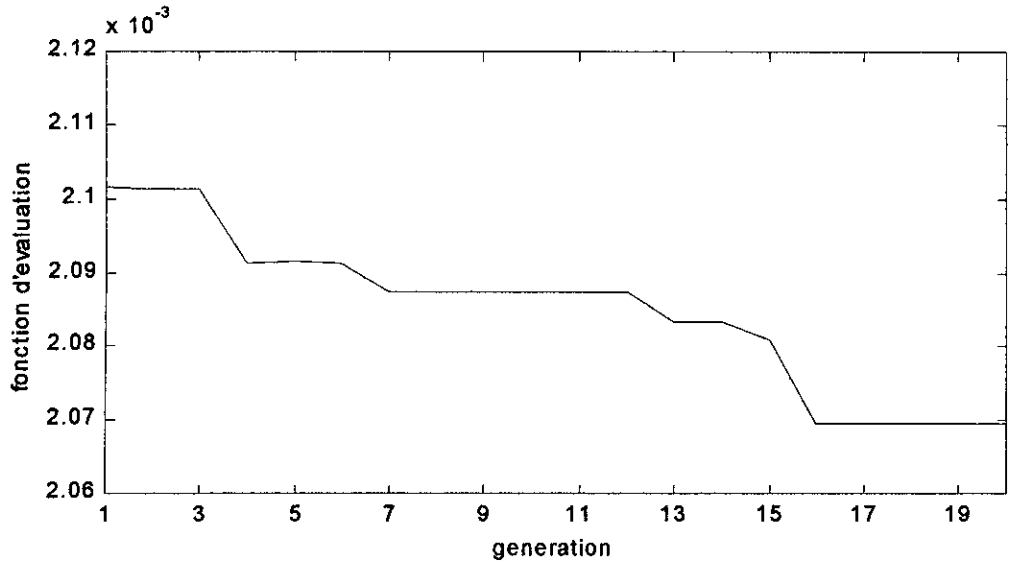


Figure (III.13) : Commandes développées aux articulations



Figure(III.14) : Evolution de la fonction d'évaluation.

	Somme quadratique des erreurs des articulations				Somme quadratique des commandes des articulations			
critères	1	2	3	Somme	1	2	3	Somme
	1.8007 e-4	3.6282 e-3	4.0153 e-4	4.210 e-3	1.2512 e+5	1.8834 e+6	1.3755 e+5	2.1460 e+6
$J1$	5.7138 e-6	1.1351 e-4	1.1149 e-5	1.3037 e-4	1.6854 e+5	1.5877 e+6	8.7798 e+5	2.6342 e+6
$J2$	9.6066 e-5	7.9085 e-5	5.4604 e-4	7.2120 e-4	1.2704 e+5	1.8309 e+6	1.4651 e+5	2.1044 e+6
$J3$	4.0758 e-6	6.8112 e-6	1.0197 e-7	1.0988 e-5	1.2661 e+5	1.8163 e+6	1.1561 e+5	2.0582 e+6

Tableau(III.6) : tableau de comparaison

On remarque du tableau (III.6) que la somme des erreurs par le critère $J1$ et inférieure à celle de $J2$ par contre la somme des commandes de $J1$ est supérieure par rapport à $J2$.

Pour l'essai 3 ($J3$) les performances deviennent meilleures que celle des deux autres essais. Les erreurs sont plus petites avec des commandes inférieures aux commandes des essais précédents.

III.5 Conclusion

Dans cette partie, nous avons synthétisé les paramètres d'une loi de commande décentralisée à structure variable à l'aide des Algorithmes Génétiques. Pour l'optimisation des paramètres de cette commande plusieurs essais ont été faits. Dans un premier temps on a fixé les paramètres des surfaces de glissement, et on a introduit deux types de critères, un critère uni-objectif qui minimise l'erreur et un critère multi-objectif qui minimise l'erreur et la commande. Les résultats obtenus montrent que les performances sont liées au choix du critère. Ainsi le critère multi-objectif réalise un compromis entre la minimisation de l'erreur et la minimisation de la commande.

L'optimisation de la surface de glissement avec les paramètres des régulateurs variables a permis d'obtenir de meilleurs résultats sur l'erreur, car la surface de glissement agit directement sur la dynamique de l'erreur.

Conclusion générale

Dans notre travail, nous avons présenté les différents aspects des Algorithmes Génétiques et les possibilités qu'ils nous offrent en matière d'optimisation de par leur grande capacité à résoudre des problèmes complexes et multi-modale.

La plupart des problèmes en automatique peuvent être transformés en un problème d'optimisation dont les AGs sont un moyen très adéquat pour le résoudre.

Plusieurs applications ont été présentées dans ce sens, où différentes techniques de conception d'AGs ont été utilisées pour chaque application.

Nous avons appliqué les AGs à l'identification paramétrique des processus par les modèles ARX et ARMAX et à la réduction d'ordre des modèles mathématiques ensuite nous avons appliqué les AGs à la régulation d'un pendule inverse par retour d'état et à l'optimisation des paramètres une loi de Commande Décentralisée à Structure Variable (CDSV) appliqué à un robot manipulateur.

Les applications réalisées dans les domaines de l'identification et de la commande, nous ont permis de faire les constatations suivantes :

- L'optimisation par Algorithme Génétique permet d'obtenir de bons résultats malgré la complexité du problème.
- L'utilisation des AGs permet de satisfaire plusieurs objectives en même temps.
- Les AGs peuvent être utilisés pour des problèmes complexe dans les quels un grand nombre de paramètres intervient.
- Les AGs nécessitent une grande puissance de calcul durant un temps énorme.

- Les AGs sont applicables dans des problèmes où les méthodes conventionnelles sont impuissantes.

- Les AGs sont applicables a tout problème dont on peut:

  - Coder les solutions du problème par une chaîne finie.

  - Evaluer chaque chaîne (trouver une fonction d'évaluation).

- Le choix du type de codage est lié à l'application et à l'espace de recherche.

- La taille de la population dépend de la puissance de calcul dont on dispose, des méthodes utilisées (sélection, opérateurs génétiques...), du nombre de variables considérées et de la fonction d'évaluation.

- Pour choisir la fonction d'évaluation, il faut classer les objectifs par ordre d'importance et adapter les poids jusqu'à l'obtention d'une solution acceptable

- Pour mettre en œuvre avec succès un AG, il faut choisir avec soin les techniques de sélection et de mutation.

Annexe

Modèle Dynamique du Robot manipulateur

Le modèle dynamique général d'un robot manipulateur rigide à n degrés de liberté peut être représenter par un système d'équation différentielle non-linéaire du second-ordre donné sous forme matricielle comme suit : [37]

$$M(q).\ddot{q} + B(q,\dot{q})\dot{q} + K(q,\dot{q})q + G(q) = u_p + u \qquad (1)$$

avec $q \in R^n$, $\dot{q} \in R^n$, $\ddot{q} \in R^n$ représente respectivement les position, les vitesse et les accélération articulaires et :

$M(q) \in R^{n \times n}$: matrice d'inertie,

$G(q) \in R^n$: vecteur de forces ou couples dus aux forces de gravitation,

$B(q,\dot{q})\dot{q}+K(q,\dot{q})q \in R^n$: englobe les couples dus aux forces de coriolis et centrifuges,

u, u_p : sont respectivement, les vecteurs de forces ou couples moteurs et perturbation externe.

Les éléments de *M, B, K, G* et *H* sont généralement des fonctions très compliquées et non-linéaires par rapport aux coordonnées générales du manipulateur.

Modèle dynamique du robot manipulateur PUMA560

Le modèle dynamique des trois premières articulations du robot manipulateur PUMA560 est donné par l'équation (2). Les trois autres articulations sont simplement maintenues à la position zéro. Ce manipulateur réalise trois mouvements de rotation, le premier suivant l'axe vertical, le second et le troisième suivent deux axes horizontaux caractérisés par ces trois coordonnées généralisés $q_1\ q_2\ q_3$.

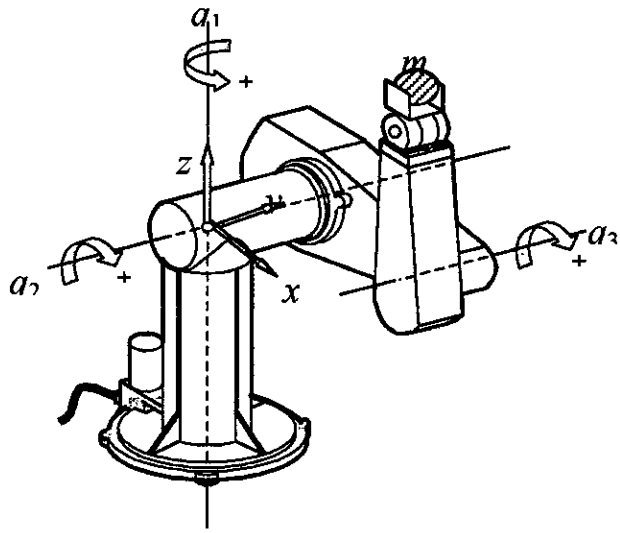


Figure 3. Robot PUMA560 à la position d'origine.*

$$M(q)\ddot{q}+B(q,\dot{q})\dot{q}+G(q)=u-mJ^{T}(q)\left[J(q)\ddot{q}+\dot{J}(q,\dot{q})\dot{q}+g\right] \qquad (2)$$

avec m : charge = 5 Kg

J : matrice Jacobienne,

g : vecteur de gravité = [0 0 9.81].

On considère les abréviations des fonctions trigonométriques suivante :

$C_i = cos\,(q_i)$,

$C_{ij} = cos\,(q_i + q_j)$,

$C_{ijk} = cos\,(q_i + q_j + q_k)$,

$S_i = sin\,(q_i)$,

$S_{ij} = sin\,(q_i + q_j)$,

$S_{ijk} = sin\,(q_i + q_j + q_k)$,

Les expressions des matrices M, B, G et du vecteur J seront :

* Dessine par M. R. Madani

Matrice d'Inertie *M*

$m_{11} \approx 2.57 + 1.38\,C_2^2 + 0.30\,S_2 S_3 + 0.744\,C_2 S_{23}$,

$m_{12} = m_{21} \approx 0.69\,S_2 - 0.134\,C_{23} + 0.0238\,C_2$,

$m_{13} = m_{31} \approx -0.134\,C_{23} - 0.00397\,S_{23}$,

$m_{22} \approx 6.79 + 0.744\,S_3$,

$m_{23} = m_{32} \approx 0.333 + 0.372\,S_3 - 0.011\,C_3$,

$m_{33} \approx 1.16$.

Facteurs de Coriolis et Centrifuge *B*

$b_{11} \approx (-2.76\,S_2 C_2 + 0.744\,C_{223} + 0.6 S_2 C_3$

$-0.0213\,(1 - 2S_2 S_3))\dot{q}_2$,

$b_{12} \approx (0.69\,C_2 + 0.134\,S_{23} - 0.0238\,S_2)\dot{q}_2$

$+(0.267\,S_{23} - 0.00758\,C_{23})\dot{q}_3$,

$b_{13} \approx (0.744\,C_2 C_{23} + 0.6 S_2 C_3 + 0.022\,S_{23} C_2$

$-0.0213\,(1 - 2S_2 S_3))\dot{q}_1$

$+0.5(0.0267\,S_{23} - 0.00758\,C_{23})\dot{q}_3$,

$b_{21} \approx -0.5(-2.76\,S_2 C_2 + 0.744\,C_{223} + 0.6 S_2 C_3$

$-0.0213\,(1 - 2S_2 S_3))\dot{q}_1$,

$b_{22} \approx (0.022\,S_3 + 0.744\,C_3)\dot{q}_3$,

$b_{23} \approx 0.5(0.022\,S_3 + 0.744\,C_3)\dot{q}_3$,

$b_{31} \approx -0.5(0.744\,C_2 C_{23} + 0.6 S_2 C_3 + 0.022\,S_{23} C_2$

$-0.0213\,(1 - 2S_2 S_3))\dot{q}_1$,

$b_{32} \approx -0.5(0.022\,S_3 + 0.744\,C_3)\dot{q}_2$,

$b_{33} = 0.$

Vecteur gravitationnel G

$g_1 = 0$,

$g_2 \approx -37.2C_2 - 8.4S_{23} + 1.02S_2$,

$g_3 \approx -8.4S_{23} + 0.25\,C_{23}$.

Matrice Jacobienne J

$J_{11} = -S_1(a_2C_2 + a_3C_{23}) - (d_2 + d_3)C_1 - d_4S_1S_{23}$,

$J_{12} = -C_1(a_2S_2 + a_3S_{23}) - d_4C_1C_{23}$,

$J_{13} = -a_3C_1S_{23} + d_4C_1C_{23}$,

$J_{21} = -C_1(a_2C_2 + a_3C_{23}) - (d_2 + d_3)S_1 + d_4C_1S_{23}$,

$J_{22} = -S_1(a_2S_2 + a_3S_{23}) + d_4S_1C_{23}$,

$J_{23} = -a_3S_1S_{23} + d_4S_1C_{23}$,

$J_{31} = 0$,

$J_{32} = -(a_2C_2 + a_3C_{23}) - d_4S_{23}$,

$J_{33} = -a_3C_{23} - d_4S_{23}$,

Où : $a_2 = 0.4319$ m,

$a_3 = -0.0203$ m,

$d_2 = 0.2435$ m,

$d_3 = -0.0934$ m,

$d_2 = 0.4331$ m.

Référence Bibliographique

[1] T. BÄCK, F. HOFFMCISTCR et H.P. SCHWERFEL *A Survey of Evolution Strategies.* International conference on genetic algorithme 1991.

[2] T. BÄCK, F. HOFFMCISTCR et H.P. SCHWERFEL *Extended Selection mechanisms in Genetique Algorithme.* seconde International conference on genetic algorithme 1991.

[3] T. BÄCK *Optimal Mutation Rates in Gentic Search.*department of computer science university of Dortmund, international conference on génétic algorithme 1993.

[4] T. BÄCK *Optimisation by means of Genetic Algorithmes.* department of computer science university of Dortmund 1991

[5] T. BACK *Self-Adaptation in Genetic Algorithms.* In proceedings of the first European Conference on Artificial Life, December 11-13, Paris, France, 1991.

[6] T. BACK *The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm.* department of computer science university of Dortmund 1993

[7] D. BEASLEY, D.R. BULL et R.R. MARTIN *An Overview of Genetic Algorithms :Part 1, Fundamentals.* university computing 15(2) pp58-69 1993

[8] D. BEASLEY, D.R. BULL et R.R. MARTIN *An Overview of Genetic Algorithms :Part 2, Research topics.*University computing 15(4) pp170-181, 1993

[9] D. BEASLEY, D.R. BULL et R.R. MARTIN A Sequential Niche Technique for Multimodal Function Optimization. Evolutionary Computation 1(2), pp101-125, 1993.

[10] T.C. BELDING The Distributed Genetic Algorithm revisited. Eshelman (1995). Proceedings of the sixth International Conference on genetic algorithms. Morgan Kaufman, San Francisco.

[11] A.BERTONI et M. DORIGO *Implicit Parallelism in Genetic Algorithms.* Artificial Intelligence (61)2, p 307-314 April 1993

[12] M. BOUCHOUCHA *Conception d'un Contrôleur a Logique Floue basée sur la théorie des Modes Glissants* Thèse de magistère, Ecole militaire polytechnique 1999.

[13] D. BOUKHETALA *Commande Décentralisée des Systèmes Structurés.* Thèse magister, Ecole nationale polytechnique 1993.

[14] H. BÜHLER *Conception des Systèmes Automatiques. p*resses polytechniques romandes 1988.

[15] A. CHIPPERFIELD et P. FLEMING *An Overview Of Evolutionary Algorithms for Control Systems Engineering.* Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield. 1996.

[16] N. DURAND et J.M. ALLIOT et F. MEDIONI *Neural Nets trained by Genetic Algorithms for collision avoidance.*1998

[17] N. DRAKOS *Synthèse de comportement animaux individuels et collectifs par Algorithmes Génétiques.* These de doctorat, Computer Based Learning Unit, University of Leeds. Traduit par R.DUMEUR 1995.

[18] D. FLORINO et S. NOLFL *God save the Red Queen ! Competition in Co-Evolutionary Robotic.* center for neuro-mimetic systems, laboratory of microcomputing, Swiss Federal Institute of Technology. 1998.

[19] A.FRICK, C. KESKIN et V. VOGELMANN *Combination and Integration of declarative approaches.* university of Karlsruhe 1996.

[20] D. E. GOLDBERG *Algorithmes Génétique, exploration et apprentissage automatique.* Paris, addison wesley1994.

[21] G.J. GRAY, Y.LI, D.J. MURRAY-SMITH et K.C.SHARMAN *Identification of Nonlinear Control Laws using Genetic Programming* university of Glasgow, technical report TR-Control-970313. 1997.

[22] G.J. GRAY, Y.LI, D.J. MURRAY-SMITH et K.C.SHARMAN *Specification of Control System Fitness Function using Constraints for Genetic Algorithms based design methods* university of Glasgow, technical report TR-Control-950329. 1995.

[23] M. KERMAT *Optimisation Globale avec les Algorithmes Génétique* Ecole supérieure d'électricité de Paris supelec rapport N° sup-0497-20. 1997.

[24] S. KHURI, T. BACK et J. HEITKÖTTER *An Evolutionary Approach to Combinatorial Optimization problems.* In D. Cismar, editor, proceeding of the 22nd Annual ACM Computer Science conference, pages 66-73, phoenix, 1994 ACM Press, New York

[25] J. D. LANDAU *Identification et Commande des systèmes.* Paris Hermes1988

[26] R. MADANI *Différentes Approches de Commande Décentralisée a Structure Variable appliquée en Robotique.* Thèse de magistère Ecole nationale polytechnique 2000.

[27] L MARET *Régulation Automatique.* Presses polytechniques romandes 1987.

[28] K.C. NG, Y.LI, D.J. MURRAY-SMITH et K.C.SHARMAN *Genetic Algorithm applied to Fuzzy Sliding Mode Controller design.* university of Glasgow, technical report TR-Control-950317. 1995.

[29] J. M. RENDERS *Algorithmes Génétiques et Réseaux de Neurones, Application a la commande de processus.* Paris, Hermès 1995.

[30] G. RUDOLPH *Convergence Analysis of Canonical Genetic Algorithms* University of Dortmund, Germany. Projet N° 107 004 91 Parallel Computing.

[31] H. SERAJI *Decentralized Adaptive Control of Manipulator, Theory, simulation and experimentation* .IEEE trans. Robotic and Automation, RA-5, 183-201.

[32] N.N. SCHRAUDOLPH et R.K. BELEW *Dynamic Parameter Encoding for Genetic Algorithms. C*omputer Science and Engineering Department, university of California, San Diego, technical report CS 90-175, 1992.

[33] E-G. TALBI *Métaheuristiques pour l'optimisation combinatoire multi-objectif:Etat de l'art.* rapport technique N° PE 98-757.33, laboratoire d'informatique fondamentale de Lille, Université des Sciences et Technologie de Lille, 1998.

[34] K.C. TAN, Y.LI, D.J. MURRAY-SMITH et K.C.SHARMAN *System Identification and Linearisation using Genetic Algorithms with Simulated Annealing.* university of Glasgow, technical report TR-Control-950316.

[35] G.WANG, E.D. GOODMAN et W.F. PUNCH *Toward the optimization of a class of black box optimization Algorithms.* Genetic Algorithms Research and Application Group (GARAGe). Michigan state university 1997.

[36] D.WHITLEY *A Genetic Algorithm Tutorial.* Technical Report Cs-93-103, Colorado Stat University 1993.

[37] K. YEUNG et Y.P. CHEN *A New Controller Design for Manipulators Using the Theory of Variable Structure Systems* IEEE Transaction on Automatic Control. VOL 33, NO 2, 1988

ملخص:

العمل المقدم في هذه المذكرة هو دراسة للخوارزميات الجينية مطبقة في مجال التعرف والتحكم في الأنظمة. في البداية قمنا بدراسة شاملة ومعمقة للخوارزميات الجينية مع توضيح مختلف التقنيات المستعملة فيها ثم طبقنا هذه للخوارزميات في مجال التعرف وتقليص درجة الأنظمة ثم استعملناها لإيجاد عوامل التغذية الخلفية بواسطة متغيرات الحالة. في النهاية استعملنا الخوارزميات الجينية لايجاد عوامل قانون التحكم اللامركزي ذو البنية المتغيرة مطبق على الذراع الآلي PUMA 560 .

كلمات مفاتيح:

الخوارزميات الجينية، التعرف، تقليص الدرجة، التغذية الخلفية بواسطة متغيرات الحالة و التحكم اللامركزي ذو البنية المتغيرة

RESUME :

Le travail présenté dans cette thèse est une étude des Algorithmes Génétiques (AGs) appliqués dans les domaine de l'identification et de la commande des systèmes. Dans un premier temps nous avons fait une étude exhaustive des AGs en présentant plusieurs techniques. Ensuite, nous avons appliqué les AGs pour l'identification des systèmes puis pour la réduction d'ordre des models mathématique par la suite nous avons utilisé les AGs pour la détermination des gains du retour d'état pour la commande d'un système non linéaire. En fin, nous avons employé les AGs pour optimiser les performances de la commande décentralisée à structure variable appliquée au robot manipulateur PUMA 560.

Mots clefs :
Algorithmes Génétiques, identification, réduction d'ordre, retour d'état, Commande Décentralisée à Structure Variable.

ABSTRACT:

The work presented in this thesis is a study of Genetic Algorithms (GAs) applied in the field of identification and control systems design In a first time we do an exhaustive study of GAs while presenting several technique. Then, we applied GAs in the identification of systems and in order reduction of mathematical models, after we use GAs to find gains of state feedback applied to nonlinear system. Finely we employ GAs to optimize performances of the Decentralized Variable Structure Control applied to the robot manipulator PUMA 560.

Keywords :
Genetic Algorithms, identification, order reduction, state feedback, Decentralized Variable Structure Control.