Numéro d'ordre 18/2010 – M/EL

République algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari BOUMEDIENE
Faculté d'électronique et d'informatique
Laboratoire Robotique Parallélisme et Electro Energétique

جامعة هواري بومدين
للعلوم والتكنولوجيا
U S T H B

Mémoire de Magister

Préparé pour l'obtention du diplôme de magister d'état
Spécialité : Electronique
Option : Contrôle de processus et robotique

PAR

OUASSEL Yacine

Contribution des algorithmes évolutionnaires au contrôle force/position d'un bras manipulateur

Soutenu publiquement le 16 Mars 2010 devant le jury composé de :

Mme. ACHOUR Nouara	**M.C. à l'USTHB**	**Présidente**
Mr. TOUMI Redouane	**Prof. à l'USTHB**	**Directeur de mémoire**
Mr. FERGUENE Farid	**M.C. à l'USTHB**	**Examinateur**
Mr. BOUDJEMAA Fares	**Prof. à l'ENP**	**Examinateur**

Tables des matières

Introduction générale

La commande des robots manipulateurs constitue une des préoccupations majeures des recherches en robotique. En effet la majorité des tâches confiées aux robots sont délicates et exigent une très grande précision sous des trajectoires rapides. Durant ces dernières années, afin d'améliorer les performances des manipulateurs, des recherches avancées ont permis de faire émerger de nouvelles approches de commande appliquées aux robots manipulateurs, notamment celle basée sur l'intelligence artificielle ou plus particulièrement basée sur des algorithmes évolutionnaires.

Les travaux présentés dans ce mémoire ont pour principal objectif de montrer la capacité des algorithmes évolutionnaires à traiter le problème de la commande force/position des robots manipulateurs. Pour ce faire, nous avons choisi de diviser ce mémoire en cinq chapitres :

Le premier chapitre présente de manière générale la problématique du contrôle force/position, le concept de la compliance qui permet de traiter le problème de l'interaction robot/environnement est ainsi présenté suivi d'un état de l'art sur les différentes structures force/position réalisant ce type de compliance. Ce chapitre nous a permis de capitaliser nos connaissances afin de choisir la structure la plus intéressante qui est la commande en effort externe, cette dernière est choisie comme structure d'implémentation de notre approche de commande.

Le deuxième chapitre constitue une étude sur la structure de commande en effort externe. Ainsi, les caractéristiques principales de cette commande ont été présentées. Par la suite nous avons proposé une approche de commande reposant sur des outils de l'intelligence artificielle. Cette approche est basée sur l'utilisation d'algorithme évolutionnaire dont le rôle est d'optimiser les paramètres internes de notre structure en effort externe.

Le troisième chapitre est consacré à la présentation des éléments nécessaires à la compréhension des algorithmes évolutionnaires et en tout particulier les algorithmes génétiques. Nous décrivons même les raisons qui ont motivé notre choix.

Le quatrième chapitre est entièrement dédié à la présentation du système considéré par notre étude : c'est le robot manipulateur PUMA 560, une description de la tâche effectuée

par ce dernier ainsi qu'une description de l'environnement ont été faites. Ce chapitre comporte aussi les différents calculs de la trajectoire désirée, le modèle géométrique direct et inverse et le modèle cinématique direct et inverse.

Le cinquième chapitre est consacré à l'application de l'algorithme génétique sur notre structure en effort externe. Cette dernière comporte une loi de commande en effort à action intégrale et une loi de commande en position à partir d'un correcteur PID, les deux optimisées par l'algorithme génétique. Cette commande sera testée et évaluée sur notre manipulateur PUMA 560. Les résultats obtenus sont présentés et analysés afin de prouver l'efficacité de notre approche.

Chapitre 1 :

Etat de l'art sur les commandes force - position

1. Introduction

Les tâches que l'on souhaite robotiser ont souvent été auparavant réalisées par l'homme. Dans la plupart des cas, celui-ci était en contact manuel avec la tâche qu'il effectuait. Il gérait donc, inconsciemment ou non, les efforts de contact engendrés au cours de l'exécution de la tâche. Certaines de ces opérations tendent à être robotisées [SCH 88], soit pour décharger l'homme de tâches fastidieuses, soit parce que les tâches demandent une précision ou une dextérité, impossible à atteindre humainement.

Pour réaliser correctement ces tâches, le robot doit donc, comme l'homme, tenter de s'adapter en permanence aux efforts qu'il exerce sur son environnement. L'automatisation de ses tâches nécessite l'exécution de tâches où s'alternent des mouvements libres puis contraints. L'assemblage, le soudage, l'usinage ou l'ébavurage constituent des exemples typiques de ce type de tâches. Ces exemples montrent que les tâches mettant en contact le robot avec son environnement nécessitent une certaine souplesse pour ne pas détériorer les pièces en contact. Une solution possible consiste à mesurer les efforts qui s'exercent entre les objets manipulés et l'environnement, puis à exploiter ces mesures pour modifier en conséquence la trajectoire de référence. Ces tâches exigent alors que le robot soit commandé en position et en effort, d'où l'intérêt d'utiliser des commandes force-position. Ce concept de comportement flexible du robot à son environnement est désigné sous le terme de **compliance**.

2. Définition de la compliance

La compliance peut être définie comme étant la capacité d'un manipulateur à avoir un comportement souple en s'adaptant à son environnement pour la tâche qu'il a à accomplir [LIA 04].

Nous pouvons distinguer deux solutions méthodologiques à la mise en œuvre de la compliance :

- Une solution classique pour prendre en compte les forces de contact consiste à interposer entre l'organe terminal et l'effecteur un dispositif présentant une certaine élasticité [WHI 79], le dispositif le plus connu est le RCC (Remote Compliance

Center) développé par Nevins [NEV 77]. Cette technique porte le nom de **compliance passive**. Ce dispositif est simple et peu encombrant, mais ses applications restent limitées à cause de sa forte dépendance vis-à-vis de la tâche à accomplir et sa spécificité en fonction de la pièce à assembler, ce qui constitue un frein à sa généralisation.

- La deuxième solution basée sur la notion de déplacements gardés, consiste à équiper le robot de capteurs qui mesurent les efforts de contacts puis après analyse de ces efforts à définir une modification de la trajectoire de référence. On obtient ainsi un mouvement compliant en contrôlant par exemple les efforts de contact. Ce type de commande, qu'il est possible d'implanter sur la majorité des robots actuels à condition qu'ils soient équipés des capteurs appropriés, se nomme **compliance active**. Cette technique apparait comme la plus appropriée pour adapter le robot à différentes classes de tâches.

3. Comment obtenir un comportement compliant de la part d'un robot ?

Considérons un robot effectuant une tâche, en contact avec son environnement (figure 1.1) [PR 97] :

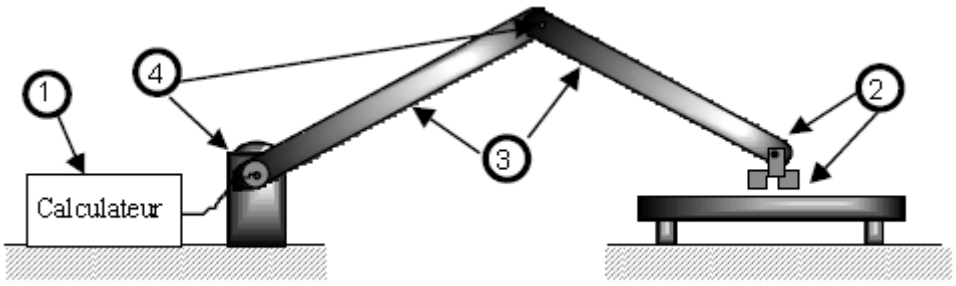

Figure 1.1 : *Localisation de la compliance dans la chaîne cinématique*

Nous pouvons schématiser cette chaine de la façon suivante (figure 1.2) :

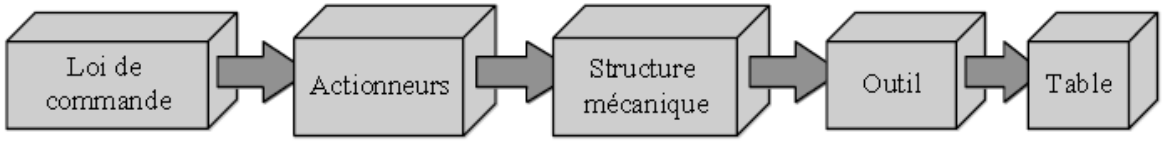

Figure 1.2 : *Schéma de la chaîne robot-environnement*

Un comportement globalement compliant sera obtenu en introduisant de la flexibilité dans cette chaine. Dans un premier temps, on peut tenter d'obtenir cette flexibilité logicielle uniquement (1).

Une deuxième solution (2), ne modifiant pas non plus la structure même du robot, consiste à ajouter un organe flexible au point de contact. Cet organe sera fixé en bout de robot, en tant que poignet compliant, soit sur l'environnement.

La troisième solution consiste à concevoir un robot possédant une structure mécanique volontairement souple (3), par l'utilisation de bras flexibles.

La dernière solution envisagée déporte la compliance dans les actionneurs (4), la souplesse de l'actionneur se répercutera ainsi au point de contact, par l'intermédiaire de la matrice jacobienne.

Dans le cadre de notre étude, nous allons s'intéresser à la solution logicielle. En effet, cela semble être une solution relativement simple et « propre » à mettre en œuvre, car elle n'implique pas de modification mécanique de l'ensemble robot - environnement.

4. La compliance logicielle : Commandes compliantes force - position

Le concept de compliance active nécessite de doter le robot de capteurs. Il utilise ces informations pour contrôler la trajectoire. Ceci conférera au robot un comportement compliant dans le sens où il minimise, ou tout au moins, limite les efforts de contact. Les différentes commandes réalisant ce type de compliance peuvent se résumer ainsi :

4. 1. Commande par retour d'effort explicite

Cette loi est née des travaux de Nevins et Whitney ([NEV 73] et [WHI 85]).

Avec cette méthode, le robot est commandé en position (ou en vitesse) et la mesure de l'effort est utilisée pour corriger le signal d'entrée. En fait, d'une manière générale, le robot est piloté en position ; il suit sa trajectoire de consigne. Dés qu'un effort extérieur apparait, c'est-à-dire dés que l'outil du robot rencontre son environnement, l'effort de contact apparaissait est alors détecté. Cette information en effort est ensuite utilisée pour modifier la commande par l'intermédiaire d'une matrice C_F ayant pour dimension l'inverse d'une raideur. Le déplacement ΔX, sera alors proportionnel à la force mesurée **:**

$$\Delta \mathbf{X} = \mathbf{C_F}\, \mathbf{F} \qquad (1.1)$$

Où :

$\mathbf{C_F}$ est la matrice diagonale de retour d'effort.

La valeur de l'élément $\mathbf{C_{fi}}$ sera faible si le degré de liberté correspondant est commandé en position et importante s'il est commandé en effort, de manière à pouvoir augmenter la compliance. On peut utiliser le même principe quand le robot est commandé en vitesse. Dans ce cas, la matrice de retour est une matrice d'amortissement.

On représente le principe de fonctionnement de cette méthode par le diagramme de la figure 1.3 :

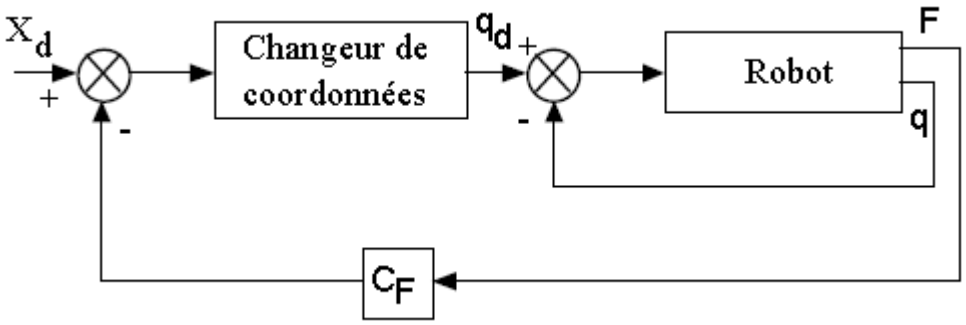

Figure 1.3 : *Principe de la commande par retour d'effort explicite*

Le principal avantage de la commande par retour d'effort est sa simplicité d'implantation et sa rapidité d'exécution, ce qui permet de l'utiliser dans l'espace cartésien, espace où est définie la tâche. Cette approche est adéquate lorsque appliquée dans un environnement homogène et stable : suivi de contour, polissage. Toutefois, elle comporte certains désavantages tant au niveau de sa structure que de l'approche utilisée. La présence du problème géométrique inverse à résoudre dans la boucle d'asservissement avec, en plus, la nécessité de résoudre ce problème en temps réel. On remarque aussi que la performance du système est directement reliée à celle du capteur de force; la présence de bruit vient fausser l'information, l'ajustement de la trajectoire s'en trouve affectée. De plus, cette commande est très sensible aux changements et discontinuités de l'environnement, une rigoureuse connaissance de l'environnement dans lequel le système évolue est donc très importante lors de l'implantation de cette commande.

4. 2. Commande en impédance

Ce type de commande a été introduit par Hogan en 1985 [HOG 85]. Quand le robot est en contact avec l'environnement, il y a interaction entre eux. Il doit donc y avoir compatibilité physique entre eux. L'idée de base de la commande en impédance repose sur le fait que si le robot se comporte comme une impédance mécanique, l'environnement se comportera comme une admittance mécanique, de sorte que le robot puisse faire face aux contraintes en position imposées par l'environnement.

Le vecteur de commande choisi pour piloter le robot à la fois en position et en vitesse, tout en maintenant un couple de consigne Γ_d est le suivant :

$$\Gamma - \Gamma_d = J^T [K_p(X_d - X) + K_v(\dot{X}_d - \dot{X})] \quad (1.2)$$

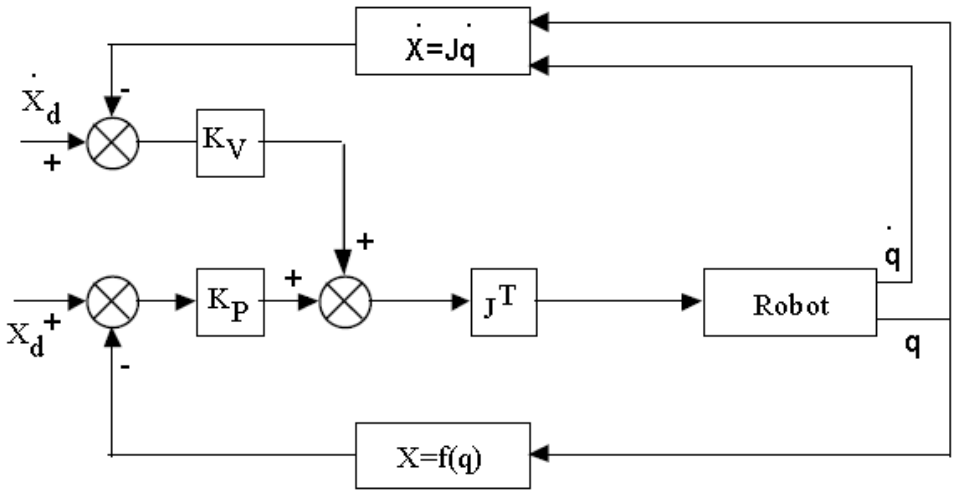

Figure 1.4 : *Principe de la commande en impédance*

La matrice $\mathbf{K_P}$ est choisie en fonction de la tâche compliante à réaliser. C'est une matrice diagonale dont les termes seront faibles lorsqu'ils correspondent à un degré de liberté commandé en effort et importants quand le degré de liberté est commandé en position. Les éléments de $\mathbf{K_V}$ déterminent l'amortissement dans chaque direction. La commande en impédance consiste donc à réguler l'interaction dynamique entre un robot et son environnement directement dans l'espace de la tâche en engendrant la commande précédente dans l'espace articulaire.

L'avantage de cette méthode est qu'elle ne nécessite pas l'inversion de la matrice jacobienne. Elle est capable de s'adapter à une classe d'environnement, mais sans contrôler à la fois la force et la position.

4. 3. Commande par raideur active

En 1980, Salisbury [SAL 80] propose de contrôler activement la raideur apparente d'un manipulateur. La raideur est changée de façon logicielle pour pouvoir s'adapter aux évolutions de la tâche. L'idée de cette commande consiste à contrôler simultanément le robot en force et en position.

Cette méthode définit, en statique, une fonction linéaire qui lie les forces d'interaction à la position finale du manipulateur, via une matrice de raideur en coordonnées cartésiennes.

$$\mathbf{F_C} = \mathbf{K_C}\,\Delta\mathbf{D_C} \qquad (1.3)$$

Avec :

$\mathbf{F_C}$: Effort de contact ;

$\mathbf{K_C}$: Matrice de raideur souhaitée ;

$\Delta\mathbf{D_C}$: Déplacement différentiel.

Le vecteur-couple différentiel des actionneurs est donné par la relation suivante :

$$\Delta\Gamma = \mathbf{J}^T\,\mathbf{F_c} \qquad (1.4)$$

Où :

J est la matrice jacobienne du manipulateur.

Donc :

$$\Delta\mathbf{D_c} = \mathbf{J}\Delta\mathbf{q} \qquad (1.5)$$

Où :

$\Delta\mathbf{q}$ représente le vecteur des variables articulaires.

On obtient alors la relation suivante dans l'espace des variables articulaires :

$$\Delta\Gamma = \mathbf{K_q}\,\Delta\mathbf{q} \qquad (1.6)$$

Où

$\mathbf{K_q}=\mathbf{J}^T \mathbf{K_C}\, \mathbf{J}$;

$\mathbf{K_q}$ représente la matrice de raideur dans l'espace articulaire. La commande a pour principe de superposer des forces de biais sur le comportement raide précédemment décrit. Cela permet d'appliquer des forces indépendantes de la position. Le réglage de la raideur (donc de la compliance) se fera par l'intermédiaire du gain $\mathbf{K_F}$.

$$\Gamma = K_q(q_d - q) + K_v(\dot{q}_d - \dot{q}) + J^T(F_d + K_F(F_d - F)) \qquad (1.7)$$

Cette loi de commande est représentée par le diagramme fonctionnel de la figure 1.5 :

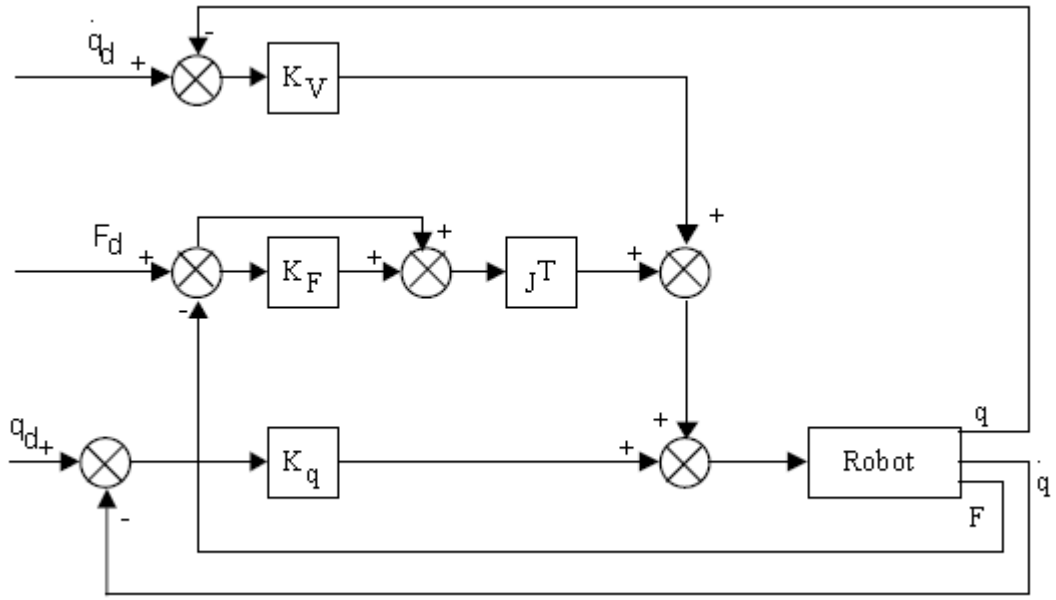

Figure 1.5 : *Principe de la commande par raideur active*

Le principal avantage de la commande par raideur active est sa simplicité d'implantation. Cette commande est appropriée pour des tâches d'assemblage ou des tâches où le contact

s'effectue à faible vitesse, mais elle manque de flexibilité face aux incertitudes de l'environnement. De plus, elle ne peut être utilisée au voisinage d'une singularité puisque dans une telle configuration, certains efforts cartésiens nuls correspondent à des efforts articulaires non-nuls [WAL 91]. Enfin, la commande s'effectuant dans l'espace articulaire, il est nécessaire de calculer le problème géométrique inverse. La présence de solutions multiples à ce problème fait que la trajectoire doit être planifiée avec soin, hors ligne, ce qui limite la facilité et la flexibilité d'interaction entre l'usager et la machine.

4. 4. Commande hybride

La commande en effort et la commande en position sont deux notions duales. Lorsque le robot est contraint par l'environnement suivant toutes les directions aucun déplacement de l'organe terminal n'est possible, il ne peut qu'exercer des efforts sur l'environnement. Par contre, lorsque le manipulateur opère dans l'espace libre, l'absence de tout contact empêche la création d'efforts. Il apparait donc que l'espace de commande doit être subdivisé en deux sous espaces complémentaires, l'un contenant les degrés de liberté commandés en position et l'autre les degrés de liberté pilotés en force.

Ce type de commande dite commande hybride force position consiste à asservir simultanément des consignes d'effort et des consignes de position afin d'obtenir la trajectoire désirée et à appliquer à l'environnement les forces souhaitées.

La synthèse de cette commande a été formalisée par Mason [MAS 81]. Les structures de commande proposées dans la littérature se répartissent selon deux grandes classes :

- L'approche associée à la première classe calcule les couples à appliquer aux articulations en fonction des consignes de position et de force (figure A).
- L'approche utilisée dans la seconde classe considère qu'à tout déplacement de l'effecteur du robot, un effort est associé. L'erreur de force détermine l'incrément de position qui vient corriger la trajectoire nominale de position (figure B).

Pour assurer l'orthogonalité des commandes issues de chaque boucle, Mason [MAS 81] a montré qu'il est nécessaire d'inclure dans ces structures une matrice de sélection. Cette matrice S qui est diagonale permet de désigner le type de commande nécessaire pour chaque degré de liberté. L'élément Si sera égal à 1 si le degré de liberté est commandé en position et 0 s'il est commandé en effort.

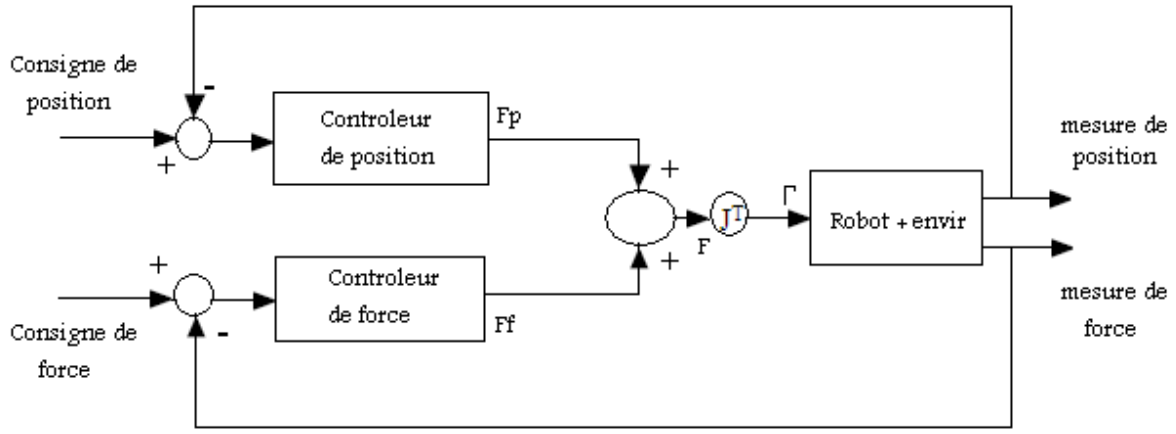

Figure A : *Approche de commande associée à une addition des forces*

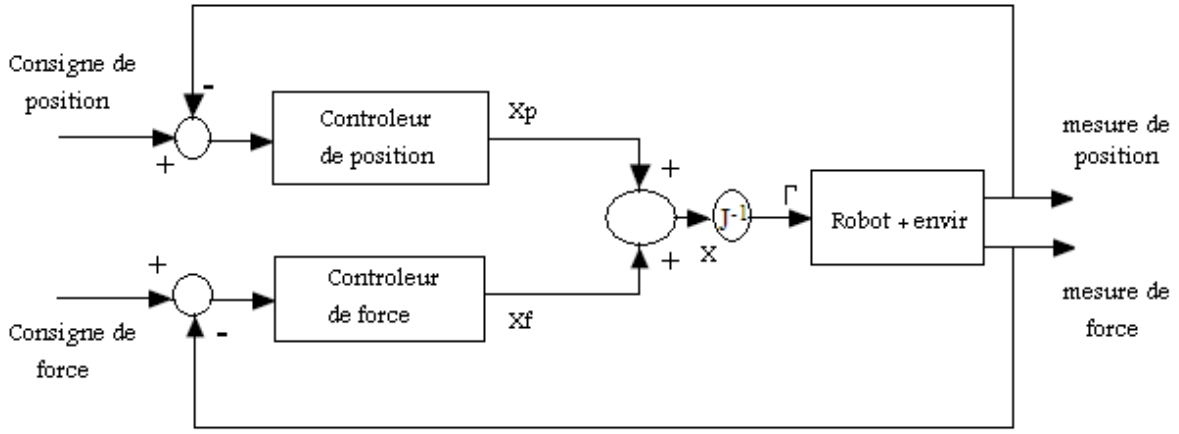

Figure B : *Approche de commande associée à une addition des déplacements*

Cette technique de commande est bien adaptée aux applications nécessitant des commandes différentes dans différentes directions. Par exemple, l'ébavurage nécessite une commande en position dans l'axe radial et une commande en effort dans l'axe transversal à la surface à ébavurer. Un inconvénient de cette stratégie de commande est qu'elle requiert l'inversion de la matrice jacobienne.

L'approche hybride permet de commander en force et en position, mais elle nécessite un modèle géométrique détaillé de l'environnement. D'autre part, les données provenant des capteurs passent par une sélection liée à la tâche à exécuter.

4. 4. 1. Principales structures hybrides forces position

A présent nous allons exposer quelques structures hybrides les plus représentatives proposées dans la littérature :

- La structure de CRAIG et RAIBERT ;
- La structure de REBOULET et ROBERT ;
- La structure de KHATIB.

4. 4. 1. 1. Structure de CRAIG et RAIBERT

Cette structure a été présentée par Craig et Raibert [RAI 81]. Son architecture se base essentiellement sur la notion d'orthogonalité de MASON c.à.d. qu'elle inclut dans sa structure une matrice de sélection qui permet de désigner le type de commande nécessaire pour chaque degré de liberté.

Le schéma global de la figure 1.6 fait apparaitre deux boucles parallèles et complémentaires, ces boucles de force et de position disposent chacune de son propre contrôleur. Une matrice de sélection S dans l'une des boucles et sa matrice complémentaire I-S dans l'autre boucle assure l'orthogonalité des commandes issues de chaque asservissement.

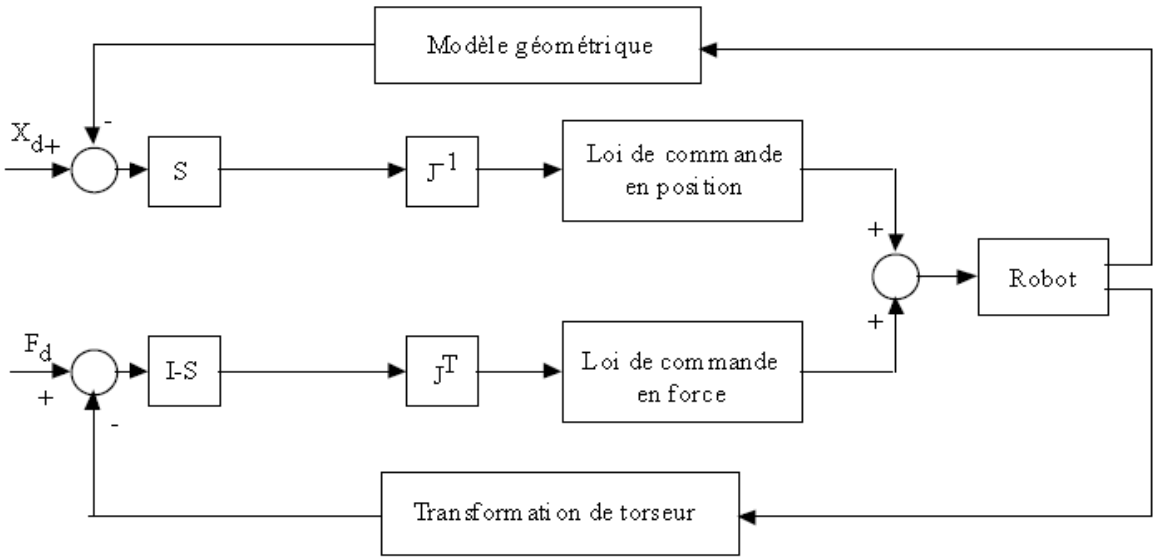

Figure 1.6 : *Structure de commande de Craig et Raibert*

4. 4. 1. 2. Structure de REBOULET et ROBERT

La structure proposée par Reboulet et Robert [REB 85] qui est illustrée figure 1.7 entre dans la deuxième classe de structures de commande hybride où l'on effectue un calcul d'erreur en position ou en vitesse pour chaque boucle. Elle utilise tout comme la première structure la formalisation de tâche de Mason.

Via l'inverse de la raideur estimée, l'erreur de force devient une erreur de position. Ces erreurs sont formées dans l'espace de tâches alors que le correcteur, commun aux deux boucles, est implémenté dans l'espace généralisé. Son entrée est une erreur en position dans l'espace articulaire. Des translations et des rotations nécessaires à la transformation géométriques sont introduites pour passer du repère de référence au repère de la tâche.

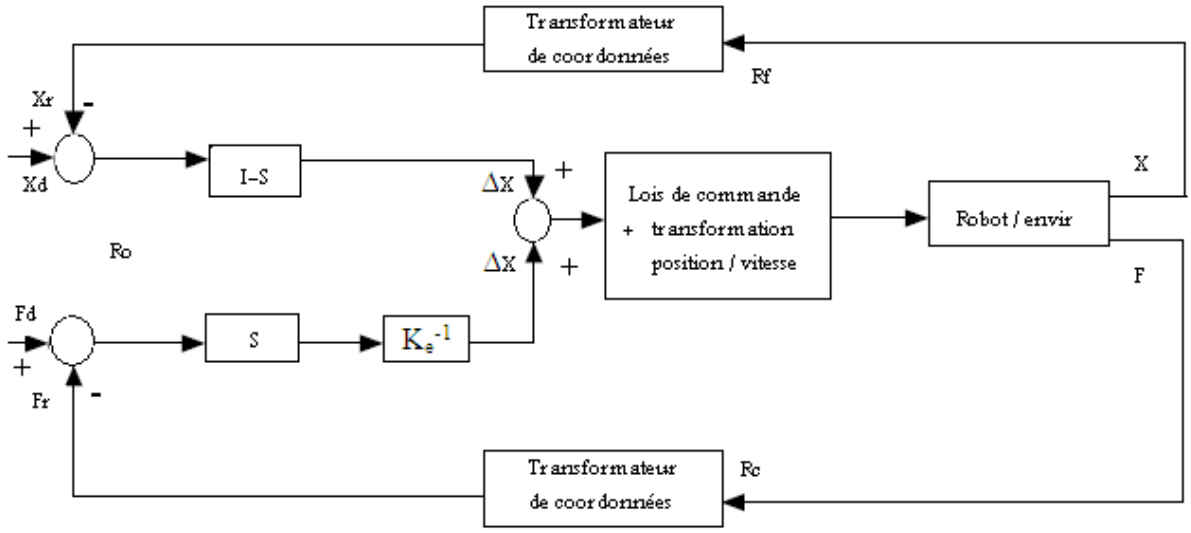

Figure 1.7 : *Structure de commande de Reboulet et Robert*

An et Hollerbach [AN 87] ont étudié la stabilité d'un tel système et montrent que le comportement du robot dépend de la façon dont les transformations cinématiques sont effectuées dans la chaine d'asservissement ainsi que de la dynamique du robot pour ce qui concerne le calcul des correcteurs. Cette étude a ainsi permis d'identifier une nouvelle forme d'instabilité qui n'apparait pas seulement aux points singuliers, mais qui dépend aussi de certaines configurations de la tâche.

4. 4. 1. 3. Structure de KHATIB

Khatib propose une formulation de la commande hybride force-position dans l'espace opérationnel. Tous les niveaux de cette structure sont placés dans le repère de référence comme indiquer sur la figure 1.8 :

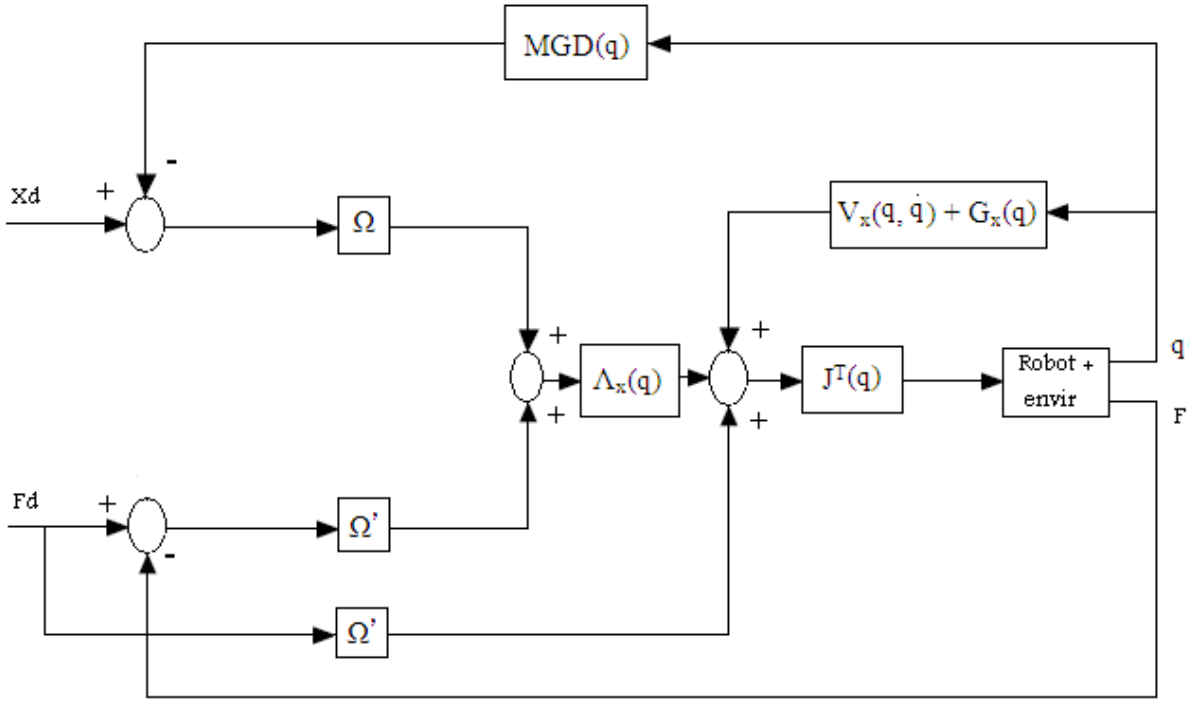

Figure 1.8 : *Structure de commande de Khatib*

Avec :

$\Lambda_x(q)$: Matrice inertielle ;

Ω : Matrice de spécification de la tâche.

La consigne de position sera directement la trajectoire sur la surface et la consigne de force un vecteur normal à tout instant à cette surface.

Khatib a introduit d'un coté, la matrice inertielle **$\Lambda_x(q)$** seulement dans l'asservissement de position pour découpler les déplacements du point terminal dus aux commandes issues du contrôleur de position dans l'espace opérationnel. D'un autre coté, il introduit une matrice de spécification de la tâche qui est constituée de la matrice de sélection et des rotations directes et inverses qui permettent le passage entre le repère de référence et le repère des contraintes.

4. 5. Commande en effort externe

Dans la structure de commande externe formalisée par Shutter et Van Brussel les deux boucles d'asservissement ne sont pas en parallèle, mais on a plutôt une hiérarchisation de la boucle d'effort sur la boucle de position [PER 03, 91] [SCH 88].

Le principe de cette structure est la suivante : A partir de la consigne $\mathbf{F_d}$ et des efforts mesurés $\mathbf{F_r}$ au niveau de l'effecteur , on détermine l'erreur d'effort $\Delta\mathbf{F}$ que l'on convertit en incrément de position $\Delta\mathbf{P}$ grâce à la loi de commande en effort, puis on ajoute cet incrément de position à la consigne de position pour avoir la consigne désirée totale.

La figure 1.9 représente le schéma de la structure de commande en effort externe.

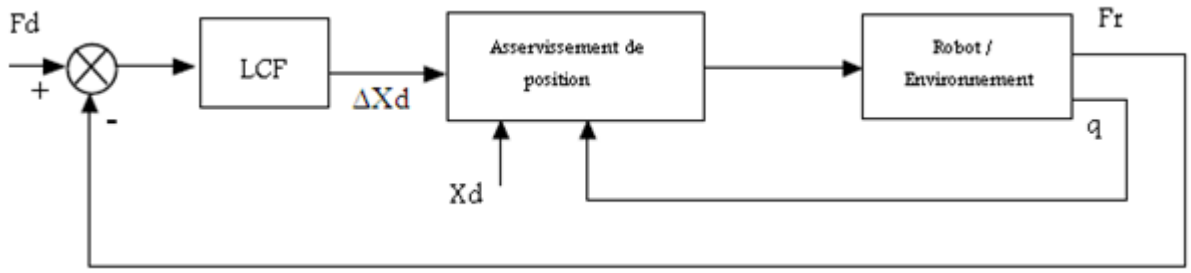

Figure 1.9 : *Principe de la commande en effort externe*

L'idée originale de cette commande est donc de convertir l'effort en déplacement que l'on va ajouter à la consigne de position au lieu d'avoir deux boucles indépendantes dont on va additionner les commandes.

La commande en effort externe présente l'avantage de ne pas faire appel à une matrice de sélection et de faire appel à moins de transformations ce qui diminue le temps de calcul. Lorsque le robot entre en contact avec un environnement de raideur donnée et qu'un choc se produit l'amplitude de l'effort d'interaction est inférieur dans le cas de cette commande [LAD 00].

Véronique Perdereau [PER 91] montre également dans son travail que pour un même réglage de la dynamique, la réponse en effort du contrôle hybride externe est plus rapide et plus précise que celle du contrôle de Craig et Raibert. N. Saadia [SAA 97] a elle aussi démontré que la commande en effort externe donne de bons résultats vu sa simplicité de mise en œuvre, cependant, son principal inconvénient réside dans la difficulté de spécifier les trajectoires désirées en effort et en position. De plus, elle nécessite comme les autres structures, la connaissance d'un modèle mathématique précis de la dynamique du robot et celui de son interaction avec l'environnement.

4. 6. Autres commandes

Le contrôle simultané en force et en position peut également être obtenu par commande parallèle [PRE 97]. La commande parallèle combine les avantages des deux méthodes décrites précédemment. Ici, la commande en force est calculée de façon à prévaloir sur la commande en position en cas de conflit ; les déviations de la tâche planifiée donnent priorité à la commande en effort, en augmentant les gains de cette dernière. Par exemple, on peut utiliser une commande Proportionnelle-Intégrale en effort et une commande Proportionnelle-Dérivée en position. Cette technique de commande a été rendue adaptative par Siciliano et Villani [SIC 96] en agissant sur les dynamiques du modèle ; elle améliore la poursuite de trajectoire, quand les paramètres sont mal estimés.

Nous pouvons aussi citer la commande compliante partagée [PRE 97]. Elle est utilisée dans le domaine de la téléopération ; les commandes classiques précédemment décrites fonctionnent bien quand le transfert prend moins d'une seconde entre l'opérateur et le manipulateur. Par contre, des retards de transmission importants (applications spatiales) causent des problèmes d'instabilité. La commande compliante partagée traite localement les forces de contact sans attendre la réaction de l'opérateur à ces efforts. La commande est partagée en ce sens que l'opérateur garde son contrôle manuel classique d'un coté et de l'autre, le robot a une fonction compliante autonome.

5. Conclusion

Nous avons introduit au début de ce chapitre la notion de compliance qui permet de traiter le problème d'interaction robot-environnement, par la suite nous avons vu les différentes commandes qui réalisent ce type de compliance, certaines de ces dernières font apparaitre un certain nombre de difficultés qui compliquent leur mise en œuvre. On va opter dans le cadre de notre travail la structure qui nous semble la plus intéressante qui n'est autre que la commande externe vu que :

- La réponse en effort est plus rapide ;
- Elle fait appel à moins de transformations géométriques ce qui réduit considérablement le temps de calcul ;
- Elle n'utilise pas de matrice de sélection, une simple commutation de consignes suffit lors de la transition espace libre/espace contraint, la structure reste donc constante ;

Dans le chapitre suivant nous allons détailler cette structure et proposer une approche de commande pour notre structure en effort externe.

Chapitre 2 :

Structure et approche de commande proposées

1. Introduction

Dans le chapitre précédent, nous avons présenté un ensemble de travaux qui s'insèrent dans le domaine général de la conception de lois de commande force/position des robots manipulateurs. Une étude critique présentant les avantages et les inconvénients de chacune de ces approches a été faite.

Nous proposons ici un chapitre assez synthétique permettant de définir la structure de commande choisie et une nouvelle approche de commande à laquelle nous l'introduisons dans notre structure de commande.

2. Synthèse de la structure de commande en effort externe

Le schéma de la commande est constitué, comme le montre la figure 2.1, de deux contrôleurs distincts, mais imbriqués. L'asservissement de force est rebouclé sur un asservissement de position, ce qui se traduit par une hiérarchisation de la boucle d'effort sur celle de la position [PER 03a, 91] [SCH 88].

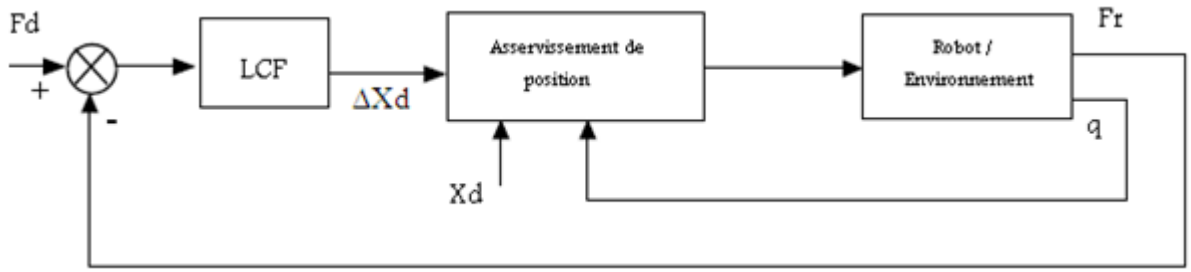

Figure 2.1 : *Structure de commande en effort externe*

Nous allons détailler chaque bloc de la structure de commande en effort externe afin de mieux comprendre son fonctionnement.

2.1. Le contrôleur d'effort

Le principe, rappelons le, est d'imposer une consigne d'effort $\mathbf{F_d}$, de récupérer l'information sur l'effort d'interaction avec l'environnement $\mathbf{F_r}$ grâce à un capteur d'effort et ensuite de calculer l'erreur d'effort $\Delta\mathbf{F= F_d - F_r}$ qui sera convertie en un incrément de déplacement $\Delta\mathbf{X}$ grâce à la loi de commande en effort. Cet incrément de position auquel on rajoute la consigne de position, constitue la consigne de position globale qui attaquera notre contrôleur de position.

2.1.1. Comment mesurer l'effort d'interaction ?

On détermine les efforts de contact du robot avec son environnement par la modélisation de la déformation d'un ressort de raideur $\mathbf{K_e}$.

L'effort mesuré sera alors : $\mathbf{F_r=K_e.\ (X-X_{obs})}$ (2.1)

Avec :

$\mathbf{X}$: position de l'effecteur dans le repère de référence ;

$\mathbf{X_{obs}}$: position de l'obstacle dans le repère de référence

2.2. Le contrôleur de position

2.2.1. L'asservissement en position

L'asservissement de position peut être cartésien dans certains cas comme il peut être articulaire dans d'autre cas, c'est-à-dire que pour un asservissement cartésien la loi de commande prend en compte l'erreur cartésienne au lieu de l'erreur articulaire pour l'asservissement articulaire.

Les transformations utilisées dans un asservissement cartésien sont internes à la boucle de position comme le montre la figure 2.2 :

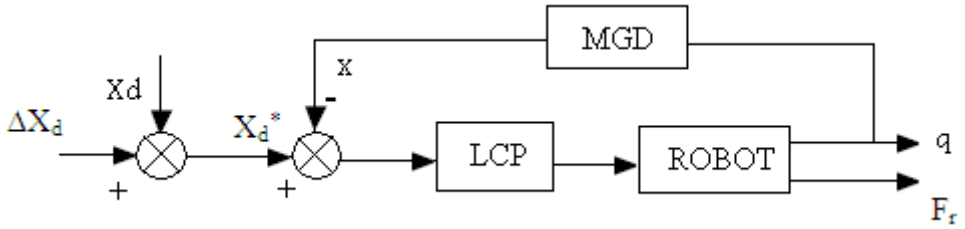

Figure 2.2 : *Structure de commande externe avec asservissement cartésien de la position*

Où :

MGD : Représente le modèle géométrique direct ;

LCP : Représente la loi de commande en position.

Dans notre travail l'asservissement est articulaire, la loi de commande en position prend en compte l'erreur $\Delta \mathbf{q}=\mathbf{q_d}-\mathbf{q}$ dans l'espace articulaire. Les consignes sont transformées de l'espace cartésien vers l'espace articulaire par le biais du modèle géométrique inverse **MGI**. Cette transformation peut se faire de deux façons différentes :

- La première méthode consiste à exprimer l'incrément de position issu de la loi de commande en effort dans l'espace articulaire à travers la jacobienne inverse, par la suite on l'additionne avec la consigne de position articulaire obtenu après transformation de la consigne de position dans l'espace cartésien grâce au modèle géométrique inverse comme le montre la figure 2.3 :

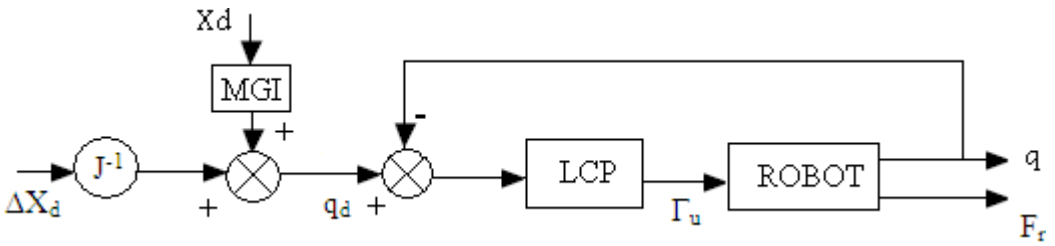

Figure 2.3 : *Structure de commande externe avec asservissement articulaire de la position (transformation indépendante des consignes)*

Où :

MGI : Représente le modèle géométrique inverse.

Cette méthode présente certains inconvénients du à l'utilisation de l'inverse de la matrice jacobienne lorsque cette dernière n'est pas inversible.

- Une deuxième méthode consiste à additionner l'incrément de position issu de la boucle d'effort avec la consigne de position pour former une consigne de position globale que l'on exprime dans l'espace articulaire par le biais du modèle géométrique inverse seulement. Le schéma de principe est le suivant :

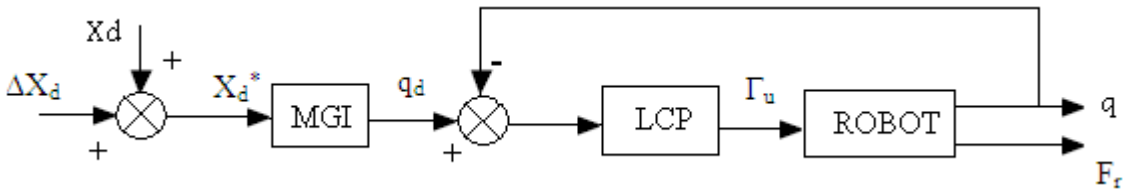

Figure 2.4 : *Structure de commande externe avec asservissement articulaire de la position (transformation globale des consignes)*

On a opté dans notre travail pour cette deuxième méthode car seul le modèle géométrique inverse intervient dans les calculs.

3. Schéma global de la structure de commande en effort externe

La figure 2.5 représente le schéma global de la structure de commande externe :

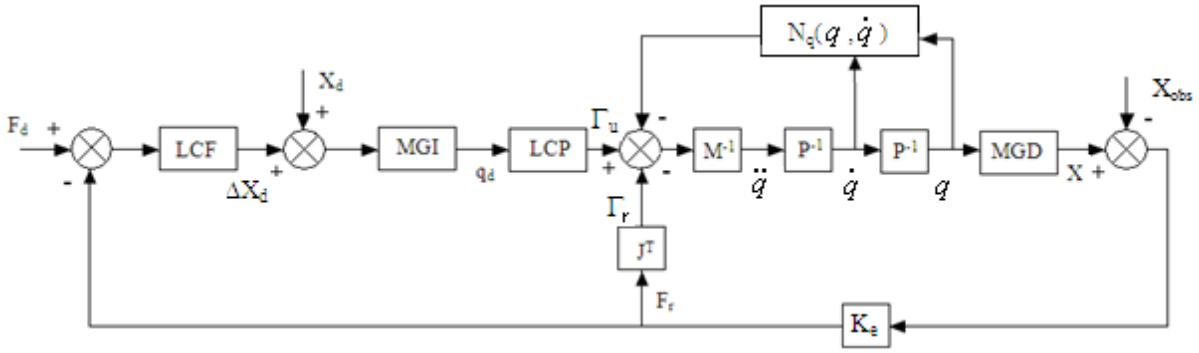

Figure 2.5 : *Schéma global de structure de commande en effort externe*

Le bloc $\boldsymbol{N_q(q,\dot{q})}$ regroupe la somme des matrices des termes centrifuges et de Coriolis ainsi que les frottements secs et visqueux tel que :

$$\boldsymbol{N_q(q,\dot{q}) = B(q,\dot{q}).\dot{q} + G(q) + \Gamma_p} \tag{2.2}$$

Où :

X : représente la position cartésienne mesurée ;

X_{obs} : représente la position cartésienne de l'obstacle dans le repère de référence ;

X_d : position désirée ;

ΔX : incrément de position issu de la boucle d'effort ;

q_d : représente la consigne de position articulaire ;

$\ddot{q}$ **:** représente l'accélération angulaire mesurée ;

$\dot{q}$ **:** représente la vitesse angulaire mesurée ;

q **:** représente la position angulaire mesurée ;

Γ_r : couple issu de l'interaction avec l'environnement ;

Γ_u : couple de commande à appliquer au robot ;

M^{-1} : Matrice d'inertie du robot.

L'ensemble des propriétés de la structure de commande en effort externe peut être résumé dans l'organigramme suivant :

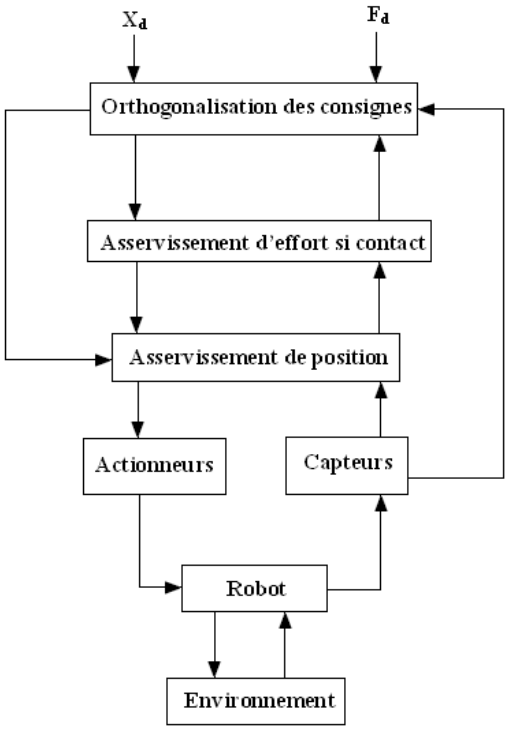

Figure 2.6 : *Structure de la commande en effort externe*

4. Approche de loi de commande

Quelle que soit la nature du système de commande considéré (continu ou discret), l'exécution d'une loi de commande revient à amener le système contrôlé d'un état stable particulier à un autre état stable en respectant des contraintes et en optimisant certains paramètres. Pour ce faire, on a choisi d'introduire dans notre structure de commande en effort externe une approche évolutionnaire qui fait partie du champ de l'Intelligence Artificielle (IA).

L'objectif de ce travail de thèse est de montrer la capacité des algorithmes évolutionnaires pour traiter le problème de la commande des systèmes compliants actifs dont les robots constituent un exemple représentatif. Notre approche de commande qui est basée sur l'utilisation d'algorithme évolutionnaire de type génétique (AG) a été mise en œuvre sur le robot manipulateur PUMA 560 qui effectue une tâche de soudure de deux pièces.

Notre structure de commande en effort externe comporte deux lois de commandes :

- Loi de commande en position basée sur un contrôleur de type proportionnel-intégrateur-dérivé (PID) optimisé par un AG ;
- Loi de commande en effort basée sur un contrôleur à action intégral optimisé par un AG.

L'algorithme génétique a pour rôle d'optimiser les paramètres de nos deux contrôleurs d'effort et de position afin de rendre notre système plus performant en lui apportant de la précision, de la rapidité, et de la stabilité.

5. Conclusion

L'analyse présentée dans ce chapitre a eu pour objectif de proposer à la fois une structure de commande et une nouvelle approche de commande qui soient bien adaptées pour traiter le problème du contrôle en force/position des systèmes compliants.

Le chapitre suivant propose des notions fondamentales sur les algorithmes évolutionnaires afin de mieux comprendre l'approche de commande par algorithme génétique utilisée par nos contrôleurs.

Chapitre 3 :

Les algorithmes génétiques

1. Introduction

Pour résoudre les problèmes d'optimisation impliqués dans nos deux lois de commande en effort et de position, nous utilisons une technique d'optimisation stochastique : les algorithmes génétiques. Dans ce chapitre, nous décrivons les principes de ces algorithmes et nous explicitons les raisons qui ont motivé le choix des algorithmes génétiques.

2. Les Algorithmes évolutionnaires

L'évolution a donné lieu à une extraordinaire diversité de formes vivantes aux capacités étonnantes. Différents organismes, adaptés aux conditions particulières de leur environnement, forment des populations coopérantes ou en compétition qui évoluent au gré de variations et de la sélection. Le plan de la croissance des organismes, « encodé » dans les génomes, varie de génération en génération. Les individus qui finissent par dominer sont ceux dont les capacités spécifiques s'accordent le mieux aux conditions environnementales. Voilà, très résumés, les principes de l'évolution et de l'adaptation des formes vivantes sur Terre. On peut s'en inspirer pour développer des concepts et des stratégies informatiques aptes à résoudre des tâches d'apprentissage et à traiter des problèmes d'optimisation pour les systèmes intelligents artificiels. Parmi ces approches, les plus intéressantes sont celles qui utilisent les **algorithmes évolutionnaires.** Elles jouent un rôle croissant comme alternative à la programmation « classique » et comptent déjà à leur actif de nombreux succès, notamment en matière de robotique et d'intelligence artificielle.

3. Généralités

Les algorithmes évolutionnaires ou AE font partie du champ de l'Intelligence Artificielle (IA) inspirée par " l'intelligence " de la Nature. Intelligence que l'on peut définir de la façon suivante : *" the capability of a system to adapt its behaviour to meet its goals in a range of environments "* [FOG 95].

Les AE s'inspirent du modèle de la théorie de l'évolution proposée par Darwin pour faire évoluer une population initiale, Nous parlerons donc *d'individus* (solutions potentielles), de *population*, de *gènes* (variables), de *chromosomes* , de *parents*, de *descendants*, de *reproduction*, de *croisement*, de *mutations*, etc. Et nous nous appuierons constamment sur des analogies avec les phénomènes biologiques.

Les AE constituent une approche originale, ils forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficiles pour lesquels on ne connaît pas de méthode classique plus efficace [SOR, WYL 93]. Le but n'est pas forcément de trouver la meilleure solution (celle qui minimise l'erreur globale) mais de trouver une bonne solution (une solution représentant un bon minima local) en un temps acceptable.

4. Organigramme d'un AE

La figure 3.1 présente l'organigramme d'un AE. Il s'agit de simuler l'évolution d'une population d'individus divers (généralement tirée aléatoirement au départ) à laquelle on applique différents opérateurs (recombinaisons, mutations…) et que l'on soumet à une sélection, à chaque génération. Si la sélection s'opère à partir de la fonction d'adaptation, alors la population tend à s'améliorer [BAC 96, BAC 97]. Un tel algorithme ne nécessite aucune connaissance du système.

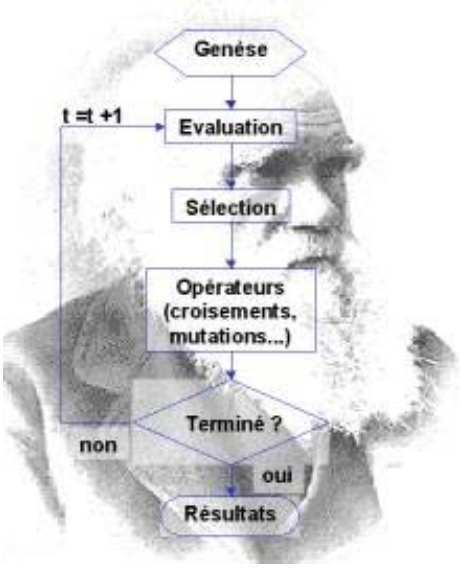

Figure 3.1 : *Organigramme d'un Algorithme Evolutionnaire*

5. Types d'algorithmes évolutionnaires

Trois types d'AE ont été développés isolément et à peu près simultanément, dans les années 60, par différents scientifiques : les *Algorithmes Génétiques*, les *Stratégies d'Evolution*, et la *Programmation Evolutionnaire*. Présentant des différences marquées à l'origine, ils tendent de plus en plus à se confondre suite à leurs emprunts respectifs [BAC 97]. Dans les années 90, ces trois champs ont commencé à sortir de leur isolement et ont été regroupés sous le terme anglo-saxon d'*Evolutionnary Computation*. Ainsi en avril 1997, la revue Transactions on Evolutionary Computation a vu le jour [FOG 97].

Notons que les AE incluent également la *Programmation Génétique* qui consiste à faire évoluer le code d'un logiciel afin qu'il remplisse au mieux certaines tâches. Citons enfin le domaine de la Vie Artificielle où l'on tente de reproduire les mécanismes de la vie dans la mémoire d'un ordinateur afin de mieux comprendre l'organisation et l'évolution du vivant.

Parmi les AE que nous venons de citer, nous avons choisi de traiter des Algorithmes Génétiques (AG). En effet, ils nous paraissaient concilier au mieux puissance, généralité et facilité de programmation. Leur particularité est qu'ils sont fondés sur le Néo-Darwinisme, c'est-à-dire l'union de la théorie de l'évolution et de la génétique moderne. Ainsi, les variables sont généralement codées en binaire (par analogie avec les quatre lettres de l'alphabet génétique) sous forme de gènes dans un chromosome. Des opérateurs génétiques (croisement, mutation) sont appliqués à ces chaînes binaires que sont les chromosomes [BAC 96].

6. Les Algorithmes génétiques

6.1. Historique

Les premiers travaux dans ce domaine ont commencé dans les années cinquante, lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis, entre 1960 et 1970, John Holland [HOL 75], sur la base des travaux précédents, développe les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique.

Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des algorithmes génétiques sur des problèmes réels de grande taille. La parution en 1989 de l'ouvrage de référence écrit par D.E. Goldberg [GOL 89] qui décrit l'utilisation de ces algorithmes dans le cadre de résolution de problèmes concrets a permis de mieux faire connaître ces derniers dans la communauté scientifique et a marqué le début d'un nouvel intérêt pour cette technique d'optimisation, qui reste néanmoins très récente.

6.2. Généralités

Les algorithmes génétiques ou AG (*Genetic Algorithm*) sont un outil d'optimisation d'inspiration biologique, leur fonctionnement est basé sur des mécanismes d'évolutions proposées par Charles Darwin [DAR 59] :

- Dans chaque environnement, seules les espèces les mieux adaptées perdurent au cours du temps, les autres étant condamnées à disparaître ;
- Au sein de chaque espèce, le renouvellement des populations est essentiellement dû aux meilleurs individus de l'espèce.

Les AG consistent à faire évoluer une population d'individus sur plusieurs générations en les croisant entre eux et en ne gardant que ceux qui résolvent le mieux un problème donné [GOL 89]. On parlera ainsi d'individu dans une population et bien souvent l'individu sera résumé par un seul chromosome (individu haploïde). Les chromosomes sont eux-mêmes constitués de gènes qui contiennent les caractères héréditaires de l'individu. On retrouvera aussi les principes fondamentaux de l'évolution naturelle, à savoir les principes de sélection, de croisement, de mutation, etc.

Dans le cadre de l'optimisation, chaque individu représente un point de l'espace d'état auquel on associe la valeur du critère à optimiser. On génère ensuite une population d'individus aléatoirement pour laquelle l'algorithme génétique s'attache à sélectionner les meilleurs individus tout en assurant une exploration efficace de l'espace d'état. Les algorithmes génétiques diffèrent des algorithmes classiques d'optimisation et de recherche essentiellement par quatre points fondamentaux :

1. Les algorithmes génétiques utilisent un codage des éléments de l'espace de recherche et non pas les éléments eux-mêmes.
2. Les algorithmes génétiques recherchent une solution à partir d'une population de points et non pas à partir d'un seul point.
3. Les algorithmes génétiques n'imposent aucune régularité sur la fonction étudiée (continuité, dérivabilité, convexité…). C'est un des gros atouts des algorithmes génétiques.
4. Les algorithmes génétiques ne sont pas déterministes, ils utilisent des règles de transition probabilistes.

La robustesse est une des caractéristiques principales des algorithmes génétiques : ils permettent de fournir une ou plusieurs solutions de « bonne » qualité (pas nécessairement optimales, mais suffisantes en pratique) à des problèmes très variés. En effet, l'heuristique de l'évolution est en quelque sorte « universelle », et très peu d'informations suffisent pour résoudre un problème quelconque. C'est ainsi qu'ils ont donné de bons résultats dans le problème du voyageur de commerce [HSL 93], dans le domaine médical [FSJP 93] ou dans les problèmes de contrôle du trafic aérien [DASF 94]. Ils sont également efficaces sur des problèmes pour lesquels il n'existe pas - encore - d'algorithme de résolution ou dont la taille est rédhibitoire pour les méthodes classiques.

Ces algorithmes sont de plus en plus utilisés dans l'industrie car ils sont particulièrement adaptés aux problèmes d'optimisation comportant de nombreux paramètres [MAG 06], ils sont bien adaptés à des problèmes où l'espace de recherche est grand et contient de nombreux minima locaux. Ils demandent par contre un bon codage des paramètres dans le génotype (individu) et une fonction d'évaluation efficace [MON, FLO 03].

6.3. Applications des algorithmes génétiques

Les applications des AG sont multiples, on peut citer les suivantes:

- Optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), optimisation d'emplois du temps, optimisation de design, optimiser des réseaux (câbles, fibres optiques, mais aussi eau, gaz…), des circuits VLSI [BEA 93a], des antennes [REI 97], des commutateurs optiques adiabatiques ont été optimisés à l'aide des Stratégies d'Evolutions (autres AE) chez SIEMENS AG [MOO 97] ;
- Traitement d'image (alignement de photos satellites, reconnaissance de suspects…) ;
- Apprentissage des réseaux de neurones [REN 95] ;
- Contrôle de systèmes industriels [BEA 93a] : contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire…) car la population peut s'adapter à des conditions changeantes. En particulier, ils supportent bien l'existence de bruit dans la fonction à optimiser ;
- Modéliser le comportement animal ou à déterminer la configuration d'énergie minimale d'une molécule;
- Trouver les paramètres d'un modèle petit-signal à partir des mesures expérimentales [MEN 97] ;
- On envisage l'intégration d'AG dans certaines puces électroniques afin qu'elles soient capables de se reconfigurer automatiquement en fonction de leur environnement *(Evolving Hardware* en anglais*).*

6.4. Principes généraux des algorithmes génétiques

6.4.1. Architecture d'un algorithme génétique

La figure 3.2 représente l'architecture d'un algorithme génétique.

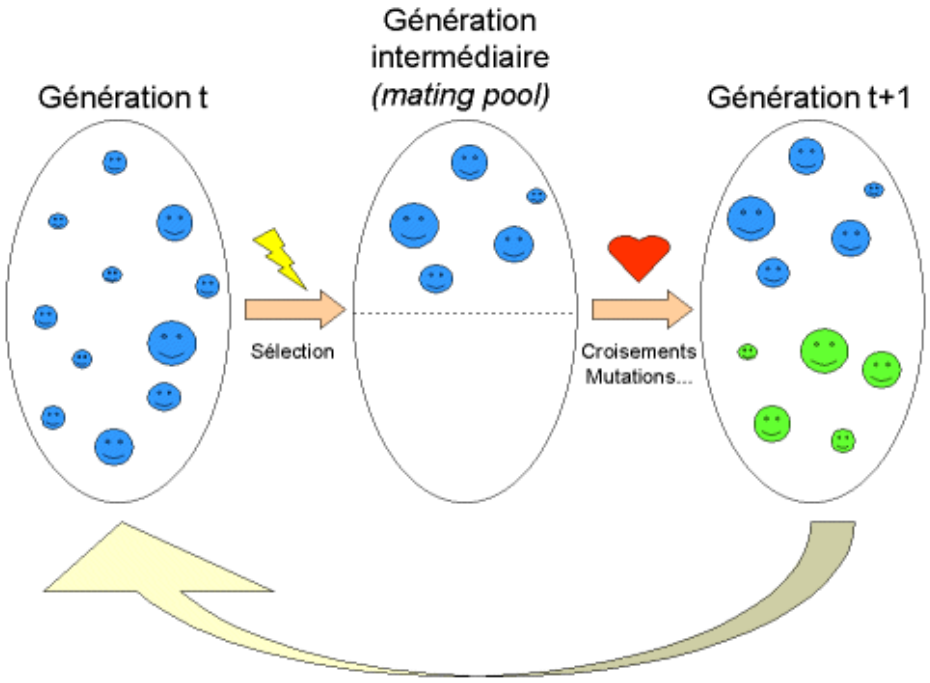

Figure 3.2 : *Architecture d'un algorithme génétique*

Les différentes étapes se résument alors à :

1. Sélectionner les individus les mieux adaptés, seuls jugés aptes à survivre :
    - évaluer le degré d'adaptation de chaque individu à son environnement ;
    - sélectionner des paires de génotypes en fonction de leurs adaptations à leur environnement.

2. Appliquer des opérateurs de diversification de la population car il est prouvé qu'une population qui vivrait en cercle fermé et serait xénophobe et raciste dégénèrerait et disparaitrait :

   - l'opérateur de croisement ;
   - l'opérateur de mutation.

3. Développer les génotypes pour obtenir les phénotypes de la nouvelle génération, puis recommencer le cycle.

6.4.2. Conception d'un algorithme génétique

6.4.2.1. Codage des variables

Le codage a pour intérêt de permettre de créer des opérateurs génétiques (sélection, croisement, mutation. Etc.). L'utilisation d'un algorithme génétique nécessite le choix préalable d'un code génétique représentant le problème à traiter. Le choix de ce codage est essentiel car il va déterminer essentiellement les performances de l'algorithme. Le code est représenté sous forme d'une chaîne de bits ou de caractères, chaine analogue à un chromosome.

La première étape est de définir et de coder convenablement le problème. A chaque variable d'optimisation ***xi*** (à chaque paramètre du dispositif), nous faisons correspondre un ***gène***. Nous appelons ***chromosome*** un ensemble de gènes. Chaque dispositif est représenté par un ***individu*** doté d'un génotype constitué d'un ou plusieurs chromosomes. Nous appelons ***population*** un ensemble de ***N*** individus que nous allons faire évoluer.

D'un point de vue informatique, nous pouvons utiliser dans l'algorithme un codage binaire. C'est-à-dire qu'un gène est par exemple un entier long.

Un chromosome est un tableau de gènes (figure 3.3). Un individu est un tableau de chromosomes. La population est un tableau d'individus. Notons qu'on pourrait aussi utiliser d'autres formes de codage (réel, codage de Gray...) [DAV 91].

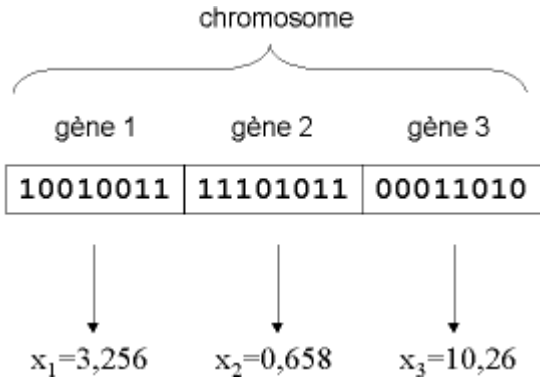

Figure 3.3 : *Codage des variables d'optimisation xi*

On aboutit à une structure présentant cinq niveaux d'organisation (figure 3.4), d'où résulte le comportement complexe des AG :

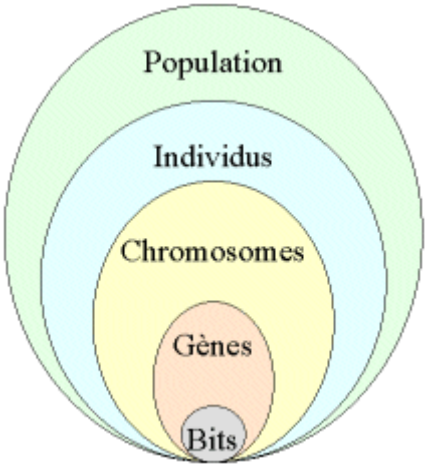

Figure 3.4 : *Niveaux d'organisation de l' AG*

Un des avantages du codage binaire est que l'on peut ainsi facilement coder toutes sortes d'objets : des réels, des entiers, des valeurs booléennes, des chaînes de caractères… Cela nécessite simplement l'usage de fonctions de codage et décodage pour passer d'une représentation à l'autre. L'inconvénient majeur du codage binaire réside dans la difficulté à

l'adapter dans le cas de problème d'optimisation de grande dimension ou à haute précision numérique.

6.4.2.2. Genèse de la population

Cette étape consiste à générer la population initiale, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer. On pourrait prendre des individus régulièrement répartis dans l'espace. Néanmoins, une initialisation aléatoire est plus simple à réaliser : les valeurs *gi* des gènes sont tirées au hasard selon une distribution uniforme.

Dans la figure suivante, on se rend compte sans difficulté que le cas 1 est plus favorable à la recherche de l'optimum que le cas 2, car la population 1 est mieux distribuée dans l'espace d'état (les individus sont figurés par les croix, l'optimum par le cercle).

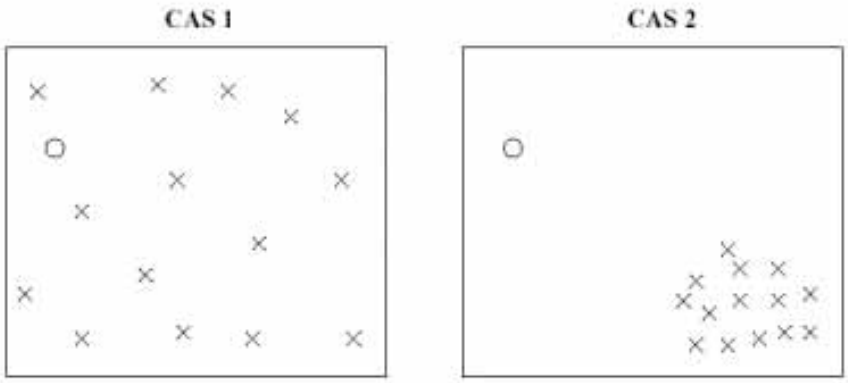

Figure 3.5 : *Exemple de Population initiale dans l'espace d'état*

A présent que nous disposons d'une population d'individus aléatoirement répartis, il nous faut être capable d'entretenir la diversité de la population au cours des générations (nous appelons *génération* la population à un instant *t* donné), afin d'entretenir le processus d'exploration de l'espace d'état : c'est le rôle des opérateurs de sélection, de croisement et de mutation, et ce jusqu'à qu'un critère d'arrêt soit satisfait. Différents critères d'arrêt de l'algorithme peuvent être choisis : nombre de générations fixé, fitness seuil, limite de convergence de la population, population qui n'évolue plus suffisamment, etc.

6.4.2.3. Sélection

La sélection, comme son nom l'indique, permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer partiellement les mauvais. Elle consiste à choisir quels seront les individus qui seront croisés pour former la nouvelle génération. Cette sélection est dépendante de la fonction d'adaptation (fitness) des individus. Un exemple très utilisé est celui de *la roue de la fortune* [GOL 94] où chaque individu est tiré au hasard en fonction de son fitness. Plus le fitness est grand, plus l'individu a de chances de se faire sélectionner.

Fonction d'évaluation et fonction *fitness* : Pour calculer le coût d'un point de l'espace de recherche, on utilise une ***fonction d'évaluation***, elle mesure le degré d'adaptation d'un individu à son environnement en estimant sa performance [GOL 94]. L'évaluation d'un individu ne dépendant pas de celle des autres individus, le résultat fourni par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus ayant le meilleur coût en fonction de la population courante : c'est le rôle de la fonction *fitness*. Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population [TOL 03]. Cependant, en général, on ne fait pas la différence entre fonction fitness et fonction d'évaluation.

On présente ci dessous les techniques de sélection les plus utilisées [GOL 89, MIC 94] :

- ***N* /2-élitisme :** les individus sont triés selon leur fonction d'adaptation. Seule la moitié supérieure de la population, correspondant aux meilleurs composants, est sélectionnée. Cette méthode induisait une convergence prématurée de l'algorithme : 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 utiles par la suite. En effet, tout individu peut transmettre à sa descendance des gènes qui, une fois combinés avec d'autres, peuvent se révéler intéressants.

- **Sélection par tournoi :** deux individus sont choisis au hasard et combattent (on compare leurs fonctions d'adaptation) pour accéder à la génération intermédiaire. Le plus adapté l'emporte avec une probabilité, 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 génération 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 génération intermédiaire, ce qui favorisera la pérennité de leurs gènes.

- **Roulette wheel selection** (ou sélection par roulette de casino) : elle consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fitness. Ces segments sont ensuite concaténés sur un axe gradué que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné, et on reproduit l'individu correspondant. Avec cette technique, les bons individus seront plus souvent sélectionnés que les mauvais, et un même individu pourra avec cette méthode être sélectionné plusieurs fois. Néanmoins, sur des populations de petite taille, il est difficile d'obtenir exactement l'espérance mathématique de sélection à cause du faible nombre de tirages (le cas idéal d'application de cette méthode est bien évidemment celui où la population est de taille infinie). On aura donc un biais de sélection plus ou moins fort suivant la dimension de la population.

Individus	*fitness*	*fitness relative*
1	15	0.1
2	30	0.2
3	45	0.3
4	22.5	0.15
5	7.5	0.05
6	30	0.2

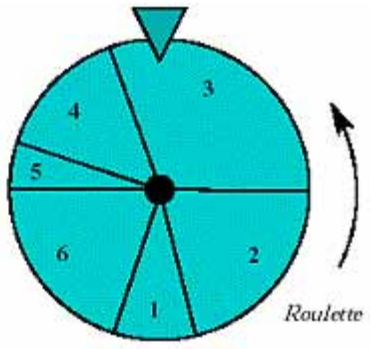

Figure 3.6 : *Exemple d'application de la " Roulette wheel selection "*

- **Stochastic remainder without replacement selection** : elle évite ce genre de problème, car une partie de la population est sélectionnée de manière purement déterministe. On associe à chaque individu le rapport ri de sa fitness sur la moyenne des fitness puis on prend sa partie entière **E(ri)** qui indique le nombre de fois à reproduire l'individu i. On assure ainsi un nombre exact de représentants pour la génération suivante, ce qui élimine le biais. Cependant, les individus faibles (fitness inférieure à la fitness moyenne) sont invariablement éliminés avec cette méthode, ce qui est mauvais, car ceux-ci occupent des positions dans l'espace d'état qui associées avec d'autres peuvent nous rapprocher du sous-domaine contenant l'optimum. On associe donc à la sélection déterministe un principe de sélection aléatoire basée sur une *roulette wheel selection* exécutée sur tous les individus affectés de nouvelles finess **ri-E(ri)**

Individus	*fitness*	r_i	$E[r_i]$	$r_i - E(r_i)$	p_i
1	15	0.6	0	0.6	1/5
2	30	1.2	1	0.2	1/15
3	45	1.8	1	0.8	4/15
4	22.5	0.9	0	0.9	3/10
5	7.5	0.3	0	0.3	1/10
6	30	1.2	1	0.2	1/15

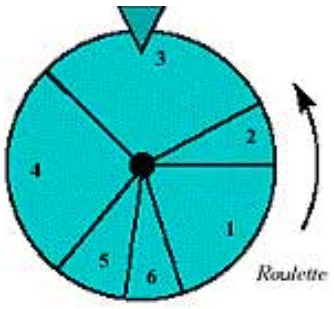

Figure 3.7 : *Exemple d'application de la " Stochastic remainder without replacement selection "*

On ajoute également fréquemment un principe d'élitisme dans le processus de sélection destiné à conserver systématiquement le ou les meilleurs individus de la population courante dans la génération suivante, sans lui faire subir de croisement ou de mutation qui pourraient le détruire. Ce principe confère aussi à l'algorithme génétique la propriété de croissance monotone de la fitness du meilleur individu au cours des générations. Les résultats issus de la théorie des schémas montrent que le principe de la *roulette wheel selection* offre le meilleur compromis entre exploration de l'espace de recherche et exploitation des informations obtenues.

6.4.2.4. Croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes [QP 93, SYS 89]. Une fois la génération intermédiaire remplie, les individus sont aléatoirement croisés deux à deux pour donner les individus de la génération suivante. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants, les chromosomes 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'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 produise un individu encore plus adapté [DES 96]. 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.

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 AG. Nous allons voir deux méthodes de croisement classiques :

- **Slicing crossover (Croisement en un point) :** on choisit au hasard un point de croisement dans chacun des parents (figure 3.8). On échange ensuite les deux sous-chaînes de chacun des chromosomes, ce qui produit deux enfants. 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.

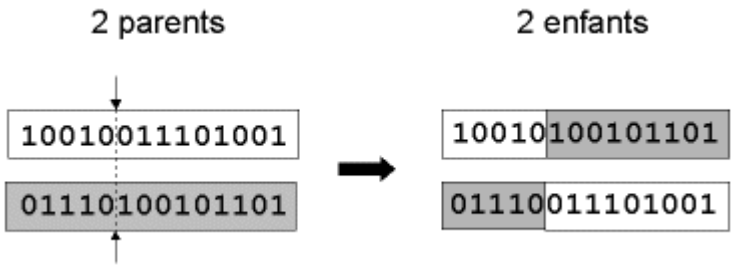

Figure 3.8 : *Croisement en 1 point*

Ce mécanisme présente l'inconvénient de privilégier les extrémités des individus. Et selon le codage choisi, il peut générer des fils plus ou moins proches de leurs parents. Pour éviter ce problème, on peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, 4, etc. [BG 91]. On parle alors de ***k-point slicing crossover.***

- **2-point slicing crossover (Croisement en deux points) :** on choisit au hasard deux points de croisement (figure 3.9).

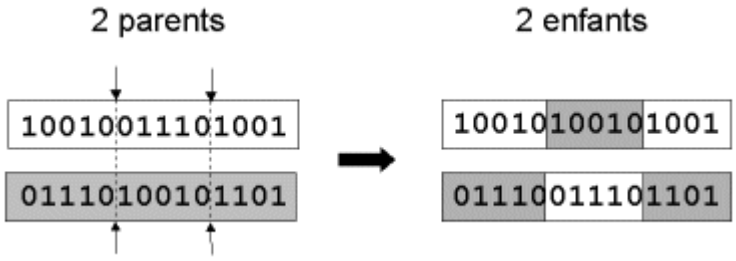

Figure 3.9 : *Croisement en 2 points*

En général, ce type de croisement est considéré comme plus efficace que le précédent [BEA 93b].

6.4.2.5. Mutation

Afin de sortir des minima locaux dans lesquels l'algorithme pourrait stagner, une mutation est faite, avec une faible probabilité. Celle-ci a généralement lieu en swappant deux parties d'un individu ou en remplaçant une valeur du tableau représentant l'individu, par exemple en inversant un bit dans un chromosome (figure 3.10).

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 afin d'éviter une convergence prématurée vers un maximum local, en maintenant une diversité de solution. De plus, elles garantissent mathématiquement que l'optimum global peut être atteint.

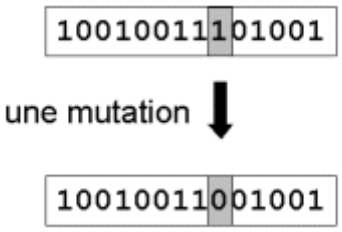

Figure 3.10: *Mutation dans un chromosome*

D'autre part, 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 [BEA 93b].

Comment réaliser notre opérateur mutation ? De nombreuses méthodes existent. Souvent la probabilité de mutation P_m par bit et par génération est fixée entre 0,001 et 0,01. On peut prendre également $P_m = 1/l$ où l est le nombre de bits composant un chromosome. Il est

possible d'associer une probabilité différente à chaque gène. Et ces probabilités peuvent être fixes ou évoluer dans le temps.

Il existe également un principe de mutation adaptative (figure 3.11) qui permet d'optimiser le taux de mutation en codant ce dernier dans la structure du chromosome [BAC 92]. Cette technique est inspirée de la formulation biologique de Wills: Si dans un environnement stable il est préférable d'avoir un taux de mutation faible, la survie d'une espèce dans un environnement subissant une évolution rapide nécessite un taux de mutation élevé permettant une adaptation rapide. Les taux de mutation d'une espèce dépendent donc de leur environnement [WIL 91].

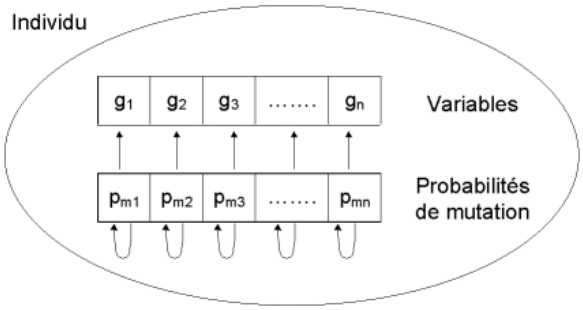

Figure 3.11 : *Principe de la mutation auto-adaptative*

6.5. Limitations des algorithmes génétiques

Les algorithmes génétiques sont des outils efficaces pour une classe de problèmes très large. De plus, ils permettent de traiter des problèmes où la fonction à optimiser ne présente aucune propriété de continuité ou de dérivabilité, par exemple. Néanmoins, les sections précédentes mettent en avant un certain nombre de limitations à leur sujet :

1. Ils sont moins efficaces qu'un algorithme déterministe spécifique (lorsqu'il en existe un) dédié à un problème donné.
2. Les nombreux paramètres qui les contrôlent sont délicats à régler (probabilités de croisement et de mutation notamment), ainsi que le codage des chromosomes qui peut faire varier radicalement la vitesse de convergence.

3. Afin de garantir la robustesse des algorithmes évolutifs, le calcul d'un très grand nombre de fitness (parfois de l'ordre de plusieurs centaines de milliers) est généralement nécessaire avant l'obtention d'une bonne solution. Ce nombre de calculs important peut s'avérer problématique lorsque le coût de calcul (ressources systèmes ou temporelles) de la fitness est important, lorsqu'on travaille en grande dimension sur des fonctions à complexité importante par exemple.

6.6. Pourquoi avoir choisi les algorithmes génétiques?

La première qualité des algorithmes génétiques et non la moindre, est leur simplicité. Un algorithme génétique complet (hormis la fonction d'évaluation) ne représente que quelques centaines de lignes de code. C'est évidement un avantage déterminant pour le développement d'applications.

Les algorithmes génétiques ne nécessitent aucune connaissance du problème, c'est leur deuxième qualité, ils ont la capacité de trouver des solutions optimales d'un problème donné, tout en ne requérant que peu d'information sur ces dernières.

Leur troisième qualité est la généricité. Il suffit, en théorie, de changer la fonction d'évaluation pour pouvoir appliquer le même algorithme génétique à différents problèmes d'optimisation. En pratique, évidemment, cette affirmation doit être nuancée et il peut arriver que la fonction à optimiser entraine de légères modifications ou adaptations du code de l'algorithme génétique lui même. Le peu de modifications à faire, et la simplicité des algorithmes génétiques font qu'il est très facile de modifier un algorithme génétique développé pour un problème d'optimisation pour l'utiliser pour un autre.

Enfin, et surtout, les algorithmes génétiques sont performants pour traiter des problèmes d'optimisation où l'espace de recherche est très grand et où l'ensemble des solutions satisfaisantes est important. Ils ne sont pas du tout adaptés pour trouver une solution unique optimale. Ils sont, par contre, très performants pour découvrir rapidement une solution acceptable (sachant qu'il y en a beaucoup) parmi un nombre colossal de solutions potentielles.

En résumé, les raisons pour lesquelles nous avons choisi d'utiliser les algorithmes génétiques sont les suivantes :

- Simplicité de programmation ;
- Non nécessité d'une connaissance du problème ;
- Généricité ;
- Adapté au problème d'optimisation (très grand espace de recherche et grand nombre de solutions).

7. Conclusion

Dans ce chapitre nous avons vu que les algorithmes génétiques (**AG**s) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (*chromosomes*) initiales arbitrairement choisies. On évalue leur performance (*fitness*) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante.

Avant d'exposer l'approche de commande par algorithme génétique, nous allons présenter dans le chapitre suivant le système robot/environnement pris comme exemple d'application.

Chapitre 4 :

Présentation du système Robot/Environnement

1. Introduction

En vue de toute commande la disponibilité d'un modèle du système est indispensable, concernant notre étude un modèle général adopté pour un robot en chaîne ouverte simple est donné par l'équation suivante [CRA 89] :

$$\boldsymbol{A(q)\ddot{q}+C(q,\dot{q})+G(q)+F(q,\dot{q})=\Gamma} \tag{4.1}$$

Où :

$\boldsymbol{A(q)}$ est la matrice (n x n) d'inertie.

$\boldsymbol{C(q,\dot{q})}$ est le vecteur (n x 1) des couples de Coriolis et centrifuges.

$\boldsymbol{G(q)}$ est le vecteur (n x l) des termes de la gravité.

$\boldsymbol{F(q,\dot{q})}$ est le vecteur (n x 1) des frottements secs et visqueux.

$\boldsymbol{\Gamma}$ est le vecteur des forces/couples articulaires appliquées.

$\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ et $\ddot{\boldsymbol{q}}$ représentent respectivement les vecteurs des positions, vitesses et accélérations articulaires.

Lorsque le robot doit interagir avec l'environnement. L'interaction est modélisée en considérant un modèle de comportement de l'environnement. Supposons que cet environnement manifeste un comportement d'un système du second ordre de type masse-amortissement-ressort [KAZ 86]. Une forme du second ordre de l'environnement admet que ce dernier a ses propres modes oscillatoires, mais simplifie l'analyse générale en considérant uniquement le premier mode. Ainsi, un tel modèle est plus restrictif qu'un simple modèle permettant de réagir sur le robot par un effort mesurable. Cependant, une représentation spécifique des composantes dynamiques de l'environnement permet une meilleure compréhension de l'interaction [VOL 90]. Le modèle choisi de l'environnement est décrit par l'équation suivante :

$$\boldsymbol{m_e\ddot{x}+b_e\dot{x}+k_e(x-x_0)=F} \tag{4.2}$$

Où **F** est la force exercée par le robot sur l'environnement, $\mathbf{m_e}$ la masse de l'environnement, $\mathbf{b_e}$ le coefficient d'amortissement, $\mathbf{k_e}$ la constante de raideur de l'environnement. $\mathbf{x_0}$ est sa position d'équilibre lorsque l'effort qu'il subit est nul. $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$ et $\ddot{\boldsymbol{x}}$ sont respectivement, la position, la vitesse et l'accélération à la fois du robot et de l'environnement [VOL 90] ; ceci suppose qu'au-delà de la position d'équilibre x_0, le robot et l'environnement sont directement en contact (mouvement sans rebonds) ou encore que l'adhérence des deux parties est instantanée.

Le modèle du système robot/environnement retenu est, en tenant compte de l'effort exercé sur l'environnement (4.2), le suivant :

$$\boldsymbol{A(q)\ddot{q}+C(q,\dot{q})+G(q)+F(q,\dot{q})+J^T F=\Gamma} \tag{4.3}$$

Où **J** est la matrice jacobienne du robot et la relation (4.3) est le modèle qui est adopté pour l'accomplissement des tâches compliantes.

Le problème peut être formulé autrement : un robot qui accomplit une tâche impliquant une interaction avec l'environnement est soumis à deux forces : celle des couples actionneurs et celle appliquée par l'environnement sur le robot qui n'est que la réaction de l'environnement à l'effort exercé par le robot. Le modèle du système est donc le suivant :

$$\boldsymbol{A(q)\ddot{q}+C(q,\dot{q})+G(q)+F(q,\dot{q})=\Gamma-J^T F} \tag{4.4}$$

En simulation les deux modèles (4.3) et (4.4) avec la relation (4.2) sont équivalents et donnent les mêmes résultats du moment que les seules entrées commandables sont celles des couples actionneurs.

Géométriquement, les contraintes affectent généralement certaines directions de l'espace opérationnel associées à certains degrés de liberté du système sans en affecter les autres directions et degrés de liberté associés. Cette association entre l'espace opérationnel et les degrés de liberté du système est réalisée à l'aide d'un repère dit de compliance, Ce dernier peut être lié, selon la tâche à accomplir, à l'organe terminal, à l'environnement ou à l'objet manipulé.

2. Présentation du Système

Le robot manipulateur utilisé dans cette étude est le Puma 560 d'Unimation. C'est un robot à six degrés de liberté dont toutes les articulations sont rotatives (type 6R).

Pour simuler et commander le Puma 560 nous utilisons un modèle explicite élaboré par B. Armstrong, O. Khatib el J. Burdick présenté dans [ARM 86]. Le modèle considéré pour effectuer leur analyse est le suivant :

$$\boldsymbol{A(q)\ddot{q} + B(q)[\dot{q}\,\dot{q}] + C(q)[\dot{q}^2] + G(q) = \Gamma} \qquad (4.5)$$

Où :

$\boldsymbol{A(q)}$ est la matrice (n x n) d'inertie.

$\boldsymbol{B(q)}$ est la matrice (n x n(n-1)/2) des couples de Coriolis.

$\boldsymbol{C(q)}$ est la matrice (n x n) des couples centrifuges.

$\boldsymbol{G(q)}$ est le vecteur (n x l) des termes de la gravité.

Γ est le vecteur des forces/couples articulaires appliqués.

$\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ et $\ddot{\boldsymbol{q}}$ représentent respectivement les vecteurs de, positions, vitesses et accélérations articulaires.

$[\dot{q}\,\dot{q}]$ est une notation du vecteur (n (n- 1)/2 x 1) des produits des vitesses articulaires.

$[\dot{q}^2]$ est une notation du vecteur (n x 1) des carrés des vitesses articulaires.

Ce modèle est dit équation de l'espace de configuration puisque les différentes matrices ne dépendent que des positions articulaires du robot. Les paramètres nécessaires pour le calcul des éléments de ces matrices sont les masses des liaisons, la position des centres de gravité des différentes liaisons et les termes d'inertie. Les éléments de ces matrices sont donnés en annexe 1.

Les masses les liaisons sont données dans le tableau 4.1. La liaison 1 n'est pas incluse puisqu'elle n'a pas été détachée de la base.

Liaison	**Masse**
Liaison 2	17,40
Liaison 3	4,80
Liaison 3 avec poignet	6,04
Liaison 5	0,34
Liaison 6	0,09
Liaison 4	0,82
Poignet	2,24

Tableau 4.1 : *Masse des liaisons [Kg ; ±0,01 +1%]*

Les coordonnées des centres de gravité sont exprimées dans les repères attachés aux liaisons (tableau 4.2). Ces repères sont attachés aux liaisons selon la notation de Denavit - Hartenberg modifiée.

Liaison	$\mathbf{r_x}$	$\mathbf{r_y}$	$\mathbf{r_z}$
Liaison 2	0,068	0,006	-0,016
Liaison 3	0	-0,070	0,014
Liaison 3 avec poignet	0	-0,143	0,014
Liaison 4	0	0	-0,019
Liaison 5	0	0	0
Liaison 6	0	0	0,032
Poignet	0	0	-0,064

Tableau 4.2 : *Centre de gravité des liaisons [M ; ±0,003]*

Dans cette notation le repère i est lié à la liaison i et l'axe Z_i est confondu avec l'axe de rotation de l'articulation i (tableau 4.3) (figure 4.1).

i	$\boldsymbol{\alpha_{i-1}}$ **(degrès)**	$\mathbf{q_i}$	$\mathbf{a_{i-1}}$ **(mètres)**	$\mathbf{d_i}$ **(mètres)**
1	**0**	q_1	0	0
2	-90	q_2	0	0.2435
3	0	q_3	0,4318	-0.0934
4	90	q_4	-0.0203	0.4331
5	-90	q_5	0	0
6	90	q_6	0	0

Tableau 4.3 : *Paramètres de Denavit-Hartenberg modifiés*

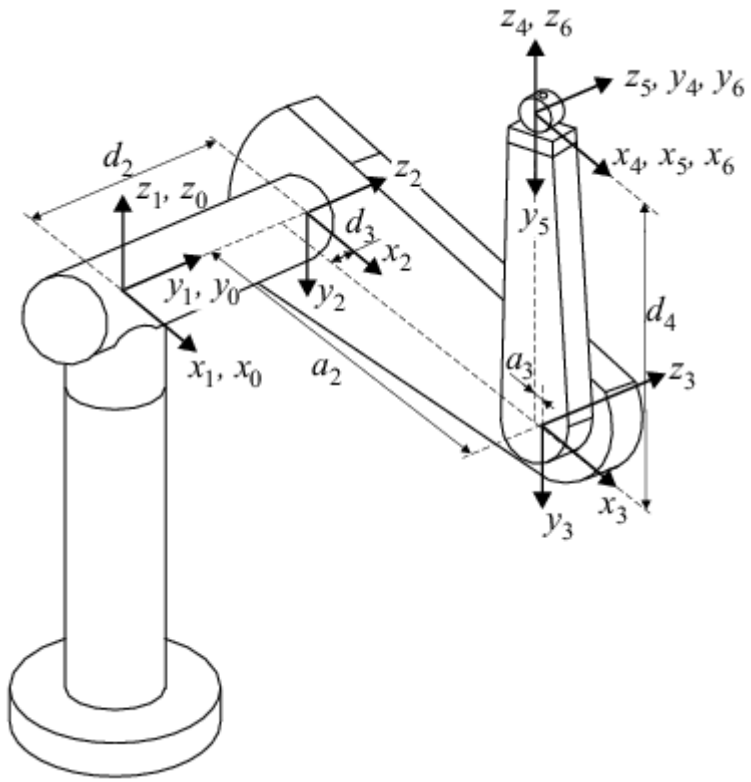

Figure 4.1 : *Le Puma560 en position zéro et les systèmes de coordonnées associés*

Dans le tableau 4.4 sont donnés les moments d'inertie et les termes d'inertie des actionneurs. Pour chaque liaison le système de coordonnées des termes d'inertie est placé au centre de gravité.

Liaison	**I_{xy}**	**I_{yy}**	**I_{zz}**	**I_{moteur}**
Liaison 1	-	-	0,35	1,14 (±0,27)
Liaison 2	0,130 (±3%)	0,524 (±5%)	0,539 (±3%)	4,71 (±0,54)
Liaison 3	0,066	0,0125	0,086	0,83 (±0,09)
Liaison 3 avec poignet	0,192 (±4%)	0,0154 (±5%)	0,212 (±4%)	-
Liaison 4	$1,80\ 10^{-3}$	$1,80\ 10^{-3}$	$1,30\ 10^{-3}$	0,200 (±0,016)
Liaison 5	$0,30\ 10^{-3}$	$0,30\ 10^{-3}$	$0,40\ 10^{-3}$	0,179 (±0,014)
Liaison 6	$0,15\ 10^{-3}$	$0,15\ 10^{-3}$	$0,04\ 10^{-3}$	0,193 (±0,015)

Tableau 4.4 : *Termes d'inertie diagonaux des liaisons et inertie des actionneurs*

Les couples actionneurs maximaux et minimaux d'action et le rapport de réduction des engrenages sont donnés dans le tableau 4.5.

	Art 1	**Art 2**	**Art 3**	**Art 4**	**Art 5**	**Art 6**
Rapport de réduction	62,61	107,36	53,69	76,01	71,91	76,73
Couple maximal [N-m]	97,6	186,4	89,4	24,2	20,1	21,3
Couple minimal [N-m]	6,3	5,5	2,6	1,3	1,0	1,2

Tableau 4.5 : *Paramètres des actionneurs*

Les termes de frottements secs et visqueux sont décrits selon [ARM 88], pour l'articulation i, comme suit :

$$\begin{aligned} &si \quad \dot{q}_i < 0 \longrightarrow F_i(t) = S_i^- + V_i^- \dot{q}_i \\ &si \quad \dot{q}_i > 0 \longrightarrow F_i(t) = S_i^+ + V_i^+ \dot{q}_i \end{aligned} \tag{4.6}$$

$\dot{q}_i$ est la vitesse angulaire de l'articulation i. Lorsque la vitesse articulaire est nulle $\mathbf{S_i = 1}$ [CRA 86] ce qui implique que $\mathbf{F_i = 1}$ puisque les frottements visqueux sont nuls. Les frottements secs $\mathbf{S_i}$ et visqueux $\mathbf{V_i}$ des trois premières articulations sont donnés ci-après:

$$\begin{matrix} S_1^- = -8.26 \\ S_1^+ = 8.43 \end{matrix} ; \quad \begin{matrix} S_2^- = -11.34 \\ S_2^+ = 12.77 \end{matrix} ; \quad \begin{matrix} S_3^- = -5.57 \\ S_3^+ = 5.93 \end{matrix}$$

$$\begin{matrix} V_1^- = 3.45 \\ V_1^+ = 4.94 \end{matrix} ; \quad \begin{matrix} V_2^- = 8.53 \\ V_2^+ = 7.67 \end{matrix} ; \quad \begin{matrix} V_3^- = 3.02 \\ V_3^+ = 3.27 \end{matrix}$$

L'indice supérieur (- ou +) indique le signe de la vitesse articulaire correspondante.

3. Description de l'environnement

L'environnement consiste en un panneau dont la surface est lisse (mouvement sans frottement) composé d'une plaque d'aluminium coupée en deux morceaux qui repose dessus, le tout fixé sur une table immobile d'une raideur considérablement élevée par rapport à celle du panneau. L'environnement est incliné d'un angle φ_0 et présente des caractéristiques vérifiant le modèle donné dans la relation (4.2) où la raideur est donnée par $\mathbf{K_e = 10000\ [N/m]}$.

4. Description de la Tâche

La tâche à accomplir par l'effecteur est de fusionner ces deux morceaux métalliques, il a donc pour rôle de souder ces deux pièces de bout en bout, de ce fait, nous aboutissons à générer une trajectoire composée de deux phases de mouvements: une phase libre où l'effecteur se positionne sur la plaque par un mouvement libre et une autre phase contrainte où l'effecteur rentre en contact avec son environnement (opération de soudure).

La première phase, où le milieu sera libre, se déroule de bas en haut autour d'un demi-cercle de centre ${}^0(0.6459, 0.1501, 0.2184)$ et de rayon r = 0.15 décrit dans le plan vertical (XOZ) où ${}^0y = 0.1501$.

La deuxième phase sera caractérisée par l'entrée en contact du robot avec son environnement, elle se déroule de haut en bas le long de la surface du panneau qui relie les deux extrémités du demi-cercle décrit dans la figure 4.2 :

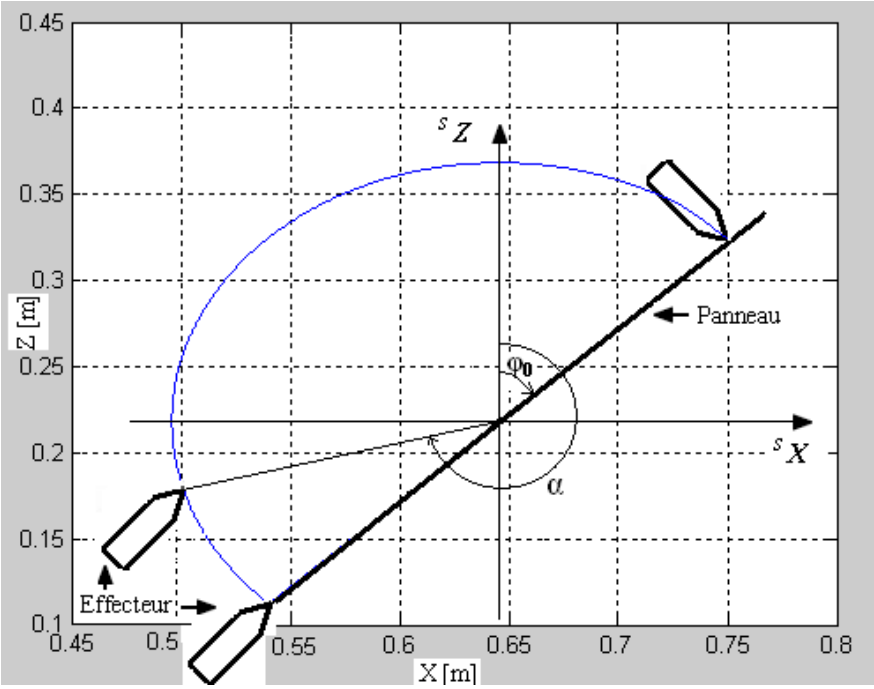

Figure 4.2 : *Trajectoire à parcourir*

5. Génération des Trajectoires

La trajectoire est décrite dans l'espace cartésien par rapport au repère de la base pour les deux phases selon une loi polynomiale du troisième degré [BE 05].

Le temps total de notre trajectoire est égale à 5s avec : $T = t_1 + t_2$ d'où $t_1 = 1$ s et $t_2 = 4$s.
Notre incrément de temps est égal à 1ms donc on aura 5000 échantillons à calculer pour élaborer notre trajectoire.

5.1. Génération de la première phase

Dans la première phase l'orientation de l'organe terminal et la position de son extrémité sont données en fonction d'un angle **α** tel que décrit dans la figure 4.2. Initialement l'orientation de l'effecteur est telle que l'organe terminal soit parallèlement alignée avec la surface de contact, et selon une loi linéaire par rapport à **α**, l'orientation varie

pour qu'à la fin de la première phase l'effecteur soit perpendiculaire à la surface de contact : $\varphi \in [\varphi_0, \varphi_0 + \pi/2]$

L'orientation de l'effecteur par rapport au repère de la base est décrite par :

$$\varphi = \frac{1}{2}\alpha + \frac{1}{2}(\varphi_0 - \pi) \tag{4.7}$$

La position cartésienne par rapport au repère de la base est donnée par :

$$\begin{aligned} x &= x_{Sorg} + r\sin(\alpha)\ ; \\ y &= y_{Sorg} = d_2 + d_3\ ; \\ z &= z_{Sorg} + r\cos(\alpha)\,. \end{aligned} \tag{4.8}$$

L'angle **α** est une fonction polynomiale cubique du temps ayant des valeurs entre $[\varphi_0 + \pi, \varphi_0 + 2\pi]$:

$$\alpha(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{4.9}$$

Avec :

$$a = \varphi_0 + \pi, \qquad a_1 = 0, \qquad a_2 = 3\frac{\pi}{t_1^2}, \qquad a_3 = -2\frac{\pi}{t_1^3}.$$

$\mathbf{t_1}$ marque le temps final de la première phase.

Les vitesses et les accélérations sont obtenues par dérivation par rapport au temps :

$$\begin{aligned} \dot{\alpha}(t) &= a_1 + 2a_2 t + 3a_3 t^2 \\ \ddot{\alpha}(t) &= 2a_2 + 6a_3 t \end{aligned} \tag{4.10}$$

D'où les vitesses et les accélérations cartésiennes de l'effecteur:

$$\begin{aligned}
&\dot{\varphi} = \frac{1}{2}\dot{\alpha}, && \dot{x} = r\dot{\alpha}\cos(\alpha), && \dot{z} = -r\dot{\alpha}\sin(\alpha).\\
&\ddot{\varphi} = \frac{1}{2}\ddot{\alpha}, && \ddot{x} = r\ddot{\alpha}\cos(\alpha) - r\dot{\alpha}^2\sin(\alpha), && \ddot{z} = -r\ddot{\alpha}\sin(\alpha) - r\dot{\alpha}^2\cos(\alpha).
\end{aligned} \tag{4.11}$$

5.2. Génération de la deuxième phase

Dans la deuxième phase on voudrait que l'organe terminal reste perpendiculaire à la surface jusqu'à la fin de la tâche. L'axe de déplacement est orienté par le vecteur $^{P}\boldsymbol{Z}$. Le déplacement le long de la surface est régi par une loi polynomiale cubique comme dans la première phase ayant des valeurs entre **[r, -r]** :

$$z_p(t) = b_0 + b_1(t-t_1) + b_2(t-t_1)^2 + b_3(t-t_1)^3 \qquad avec : t \geq t_1 \tag{4.12}$$

Avec :

$$b_0 = r, \qquad b_1 = 0, \qquad b_2 = -6\frac{r}{t_2^2}, \qquad b_3 = 4\frac{\pi}{t_2^3}.$$

$\mathbf{t_2}$ marque le temps final de la deuxième phase.

Les vitesses el les accélérations sont:

$$\begin{aligned}
&\dot{z}_p(t) = b_1 + 2b_2(t-t_1) + 3b_3(t-t_1)^2\\
&\ddot{z}_p(t) = 2b_2 + 6b_3(t-t_1)
\end{aligned} \tag{4.13}$$

L'orientation désirée étant constante, ses dérivées sont nulles :

$$\varphi = \varphi_0 + \frac{\pi}{2}, \qquad \dot{\varphi} = 0, \qquad \ddot{\varphi} = 0. \tag{4.14}$$

Les positions, les vitesses et les accélérations de l'organe terminal sont données par :

$$\begin{aligned} &x = x_{Sorg} + z_p \sin(\varphi_0), && y = d_2 + d_3, && z = z_{Sorg} + z_p \cos(\varphi_0) \\ &\dot{x} = \dot{z}_p \sin(\varphi_0), && \dot{y} = 0, && \dot{z} = \dot{z}_p \cos(\varphi_0) \\ &\ddot{x} = \ddot{z}_p \sin(\varphi_0), && \ddot{y} = 0, && \ddot{z} = \ddot{z}_p \cos(\varphi_0) \end{aligned} \qquad (4.15)$$

Les positions, vitesses et accélérations cartésiennes désirées durant les deux phases sont présentées à figure 4.3 :

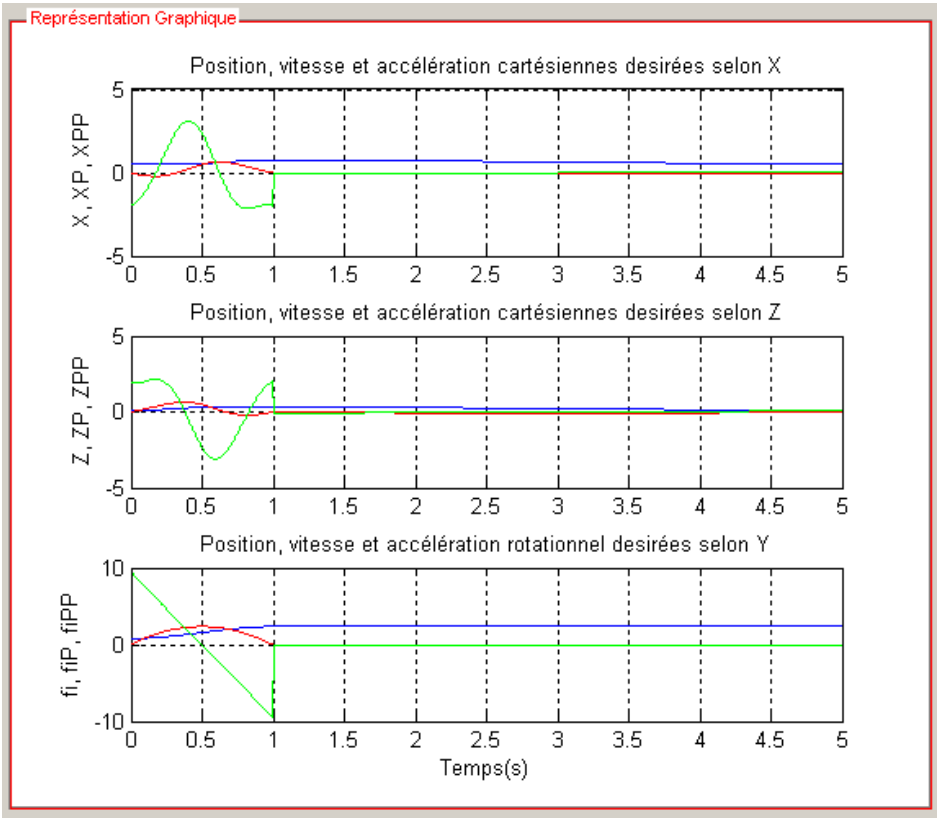

Figure 4.3 : *Trajectoire cartésienne désirée*

6. Géométrie et Cinématique Inverses

Souvent, il est commode de décrire la trajectoire désirée dans l'espace articulaire pour effectuer une correction en position et/ou en vitesse, Aussi, le recours aux modèles géométrique et cinématique inverses est nécessaire.

6.1. Le modèle géométrique directe

Le modèle géométrique directe (MGD) permet de transformer la position du robot de l'espace articulaire vers l'espace opérationnel. Du fait que le mouvement désiré (figure 4.2) est contenu dans un plan vertical et pour éviter que le robot ne soit redondant vis à vis de la tâche, il suffit de verrouiller les articulations 1 ,4 et 6 ($q_1=q_4=q_6=0$). Dans ce cas, le problème est planaire et les transformations décrivant le MGD se réduisent à :

$$ {}^{0}T_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^{1}T_{2} = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^{2}T_{3} = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}T_{4} = \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & -1 & -d_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^{4}T_{5} = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^{5}T_{6} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_{6} = \begin{bmatrix} c_{235} & 0 & s_{235} & a_3c_{23} + d_4s_{23} + a_2c_2 \\ 0 & 1 & 0 & d_2 + d_3 \\ -s_{235} & 0 & c_{235} & -a_3s_{23} + d_4c_{23} - a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad {}^{6}T_{E} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_{E} = \begin{bmatrix} c_{235} & 0 & s_{235} & a_3c_{23} + d_4s_{23} + a_2c_2 + l_6s_{235} \\ 0 & 1 & 0 & d_2 + d_3 \\ -s_{235} & 0 & c_{235} & -a_3s_{23} + d_4c_{23} - a_2s_2 + l_6c_{235} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

La transformation ${}^0T_E$ montre que l'orientation de l'organe terminal correspond à une rotation autour de ${}^0\mathbf{Y}$ d'un angle φ et que les rotations autour de ${}^0\mathbf{X}$ et ${}^0\mathbf{Z}$ sont nulles:

$$\varphi = q_2 + q_3 + q_5 \tag{4.16}$$

Nous tirons aussi la relation entre la position de l'extrémité de l'effecteur et l'origine du repère attaché au poignet {W} :

$$\begin{aligned} x_{Worg} &= x_E - l_6 \sin(\varphi) \ ; \\ y_{Worg} &= y_E = d_2 + d_3 \ ; \\ z_{Worg} &= z_E - l_6 \cos(\varphi). \end{aligned} \tag{4.17}$$

6.2. Le modèle géométrique inverse

L'opération inverse (i.e., exprimer une position décrite dans l'espace opérationnel dans l'espace articulaire) est réalisée en utilisant le modèle géométrique inverse (MGI). En utilisant la méthode développée par Pieper [KHA 86], les résultats obtenus sont les suivants :

- La position de la 3éme articulation est donnée par :

$$q_3 = a\tan 2\left(K, \sqrt{a_3^2 + d_4^2 - K^2}\right) - a\tan 2\left(a_3, d_4\right) \tag{4.18}$$

Où :

$$K = \frac{x_{Worg}^2 + z_{Worg}^2 - a_2^2 - a_3^2 - d_4^2}{2a_2}$$

- Celle de la 2éme articulation correspond à :

$$q_2 = a\tan 2\left(K_2 x_{Worg} - K_1 z_{Worg}, K_1 x_{Worg} + K_2 z_{Worg}\right) \tag{4.19}$$

Où :

$$K_1 = -a_3 \cos(q_3) + d_4 \sin(q_3) + a_2$$
$$K_2 = d_4 \cos(q_3) - a_3 \sin(q_3)$$

- La 5ème articulation est déterminée par :

$$q_5 = \varphi - (q_2 + q_3) \tag{4.20}$$

Les positions articulaires issues de la trajectoire désirée décrite dans l'espace opérationnel sont représentées dans la figure 4.4 :

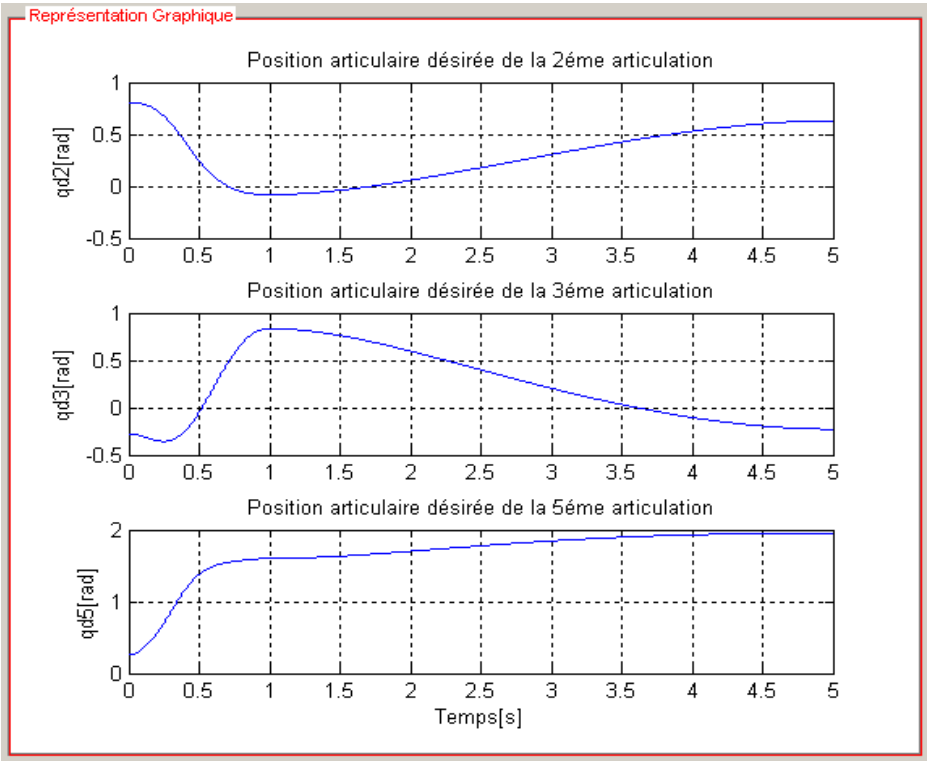

Figure 4.4 : *Positions articulaires désirées*

6.3. Le modèle cinématique directe

Le modèle cinématique direct (MCD) décrit les vitesses opérationnelles en fonction des vitesses articulaires. Il est défini par :

$$\dot{X} = J(q)\dot{q} \tag{4.21}$$

Où **J(q)** désigne la matrice jacobienne du robot. Le problème revient au calcul du jacobien, lequel est obtenu en dérivant le modèle géométrique direct. De l'expression de ${}^{0}T_{E}$ il est directement déduit que :

$$\begin{aligned} X_E &= a_3c_{23} + d_4s_{23} + a_2c_2 + l_6s_{235} \\ Y_E &= d_2 + d_3 \\ Z_E &= -a_3s_{23} + d_4c_{23} - a_2s_2 + l_6c_{235} \\ \varphi &= q_2 + q_3 + q_5 \end{aligned} \tag{4.22}$$

En dérivant ces équations il vient :

$$\begin{aligned} \dot{X}_E &= \left[-(a_2s_2 + a_3s_{23}) + d_4c_{23} + l_6c_{235}\right]\dot{q}_2 + \left[-a_3s_{23} + d_4c_{23} + l_6c_{235}\right]\dot{q}_3 + l_6c_{235}\dot{q}_5 \\ \dot{Y}_E &= 0 \\ \dot{Z}_E &= -\left[a_2c_2 + a_3c_{23} + d_4s_{23} + l_6s_{235}\right]\dot{q}_2 - \left[a_3c_{23} + d_4s_{23} + l_6s_{235}\right]\dot{q}_3 - l_6s_{235}\dot{q}_5 \\ \dot{\varphi} &= \dot{q}_2 + \dot{q}_3 + \dot{q}_5 \end{aligned} \tag{4.23}$$

En mettant cette relation sous forme matricielle on obtient :

$$\begin{bmatrix} \dot{X}_E \\ \dot{Z}_E \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} -(a_2s_2 + a_3s_{23}) + d_4c_{23} + l_6c_{235} & -a_3s_{23} + d_4c_{23} + l_6c_{235} & l_6c_{235} \\ -\left[a_2c_2 + a_3c_{23} + d_4s_{23} + l_6s_{235}\right] & -\left[a_3c_{23} + d_4s_{23} + l_6s_{235}\right] & -l_6s_{235} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \end{bmatrix} \tag{4.24}$$

6.4. Le modèle cinématique inverse

Le modèle cinématique inverse (MCI) exprime les vitesses articulaires en fonction des vitesses opérationnelles. Ceci est réalisé, dans le cas régulier (i.e., le jacobien est carré et défini), en inversant la matrice jacobienne telle que :

$$\dot{q} = J^{-1}(q)\,\dot{X} \tag{4.25}$$

Donc notre système devient :

$$\begin{bmatrix} \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} \frac{-(a_3c_{23} + d_4s_{23})}{a_2} & \frac{a_3s_{23} - d_4c_{23}}{a_2} & \frac{l_6(a_3c_5 - d_4c_5)}{a_2} \\ \frac{a_2c_2 + a_3c_{23} + d_4s_{23}}{a_2} & \frac{-(a_2s_2 + a_3s_{23} - d_4c_{23})}{a_2} & \frac{-l_6(a_2c_{35} + a_3c_5 - d_4s_5)}{a_2} \\ -c_2 & s_2 & -a_3s_3 + d_4c_3 + l_6c_{35} \end{bmatrix} \begin{bmatrix} \dot{X}_E \\ \dot{Z}_E \\ \dot{\varphi} \end{bmatrix} \tag{4.26}$$

Où delta est le déterminant du jacobien: $\Delta = -a_3s_3 + d_4c_3$

Les vitesses articulaires correspondant aux vitesses opérationnelles désirées sont représentées à la figure 4.5 :

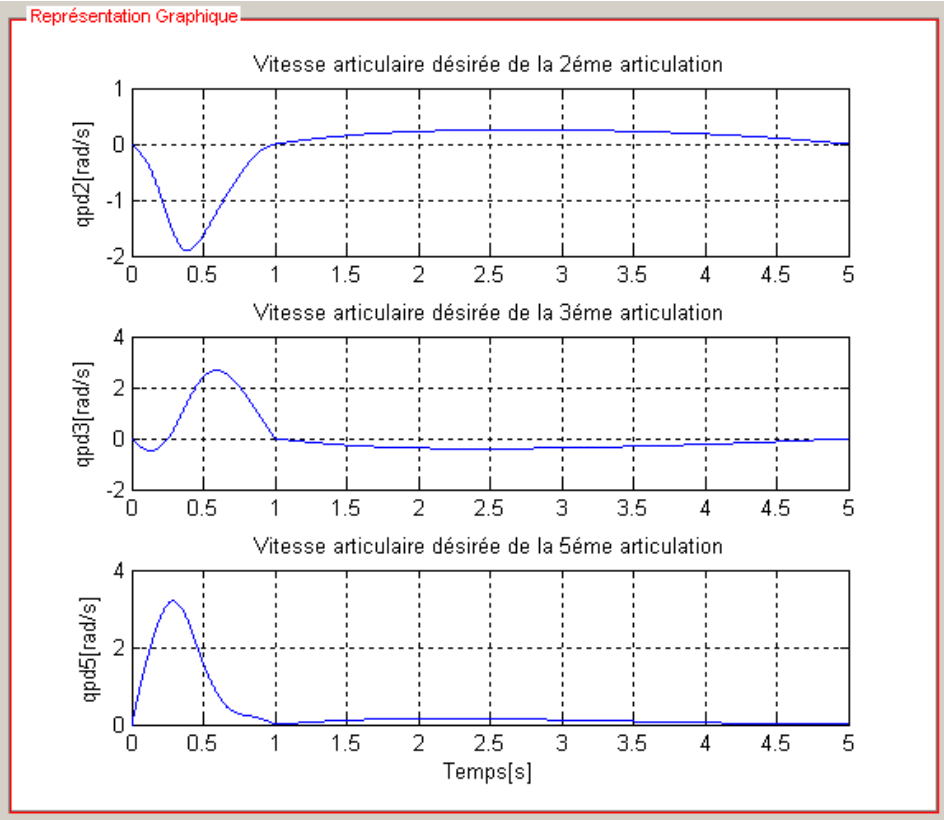

Figure 4.5 : *Vitesses articulaires désirées*

7. Conclusion

Nous avons vu dans ce chapitre le système robot/environnement pris comme exemple d'application, ainsi que la tâche à accomplir. Nous avons même vu les calculs concernant la génération de la trajectoire, le modèle géométrique direct et inverse et le modèle cinématique direct et inverse.

L'approche et l'application des algorithmes génétiques au problème spécifique qui nous intéresse, à savoir la mise en œuvre de nos contrôleurs optimisés sur notre structure de commande en effort externe, serons abordés dans le prochain chapitre.

Chapitre 5 :

Application de l'AG sur la structure de commande en effort externe

1. Introduction

Dans ce chapitre, nous allons mettre en œuvre la structure de commande en effort externe et la mettre en application sur notre tâche que nous avons présenté au 4éme chapitre.

Notre structure de commande comporte une loi de commande en effort à action intégral et une loi de commande en position à partir d'un correcteur PID, les deux optimisées par un algorithme génétique.

Nous avons commencé en premier lieu à implémenter et tester notre contrôleur de position et par la suite introduire notre contrôleur d'effort.

2. Loi de commande en position

Notre loi de commande en position sera élaborée à partir d'un correcteur de type PID optimisé par un algorithme génétique. Les couples de commande sont déterminés par la relation suivante :

$$\Gamma = K_p(q_d - q) + K_v(\dot{q}_d - \dot{q}) + K_i \int_{t_0}^{t} (q_d - q)\partial\tau + G \qquad (5.1)$$

Où :

$\mathbf{K_p}$, $\mathbf{K_v}$, et $\mathbf{K_i}$ sont des matrices diagonales définies positives de dimension (n x n) dont les éléments représentent respectivement les gains proportionnels, dérivés et intégrals des articulations ;

q_d et $\dot{q}_d$ sont les vecteurs des positions et vitesses articulaires désirées ;

G représente le vecteur de compensation de la gravité.

Le schéma de cette commande est présenté à la figure 5.1 :

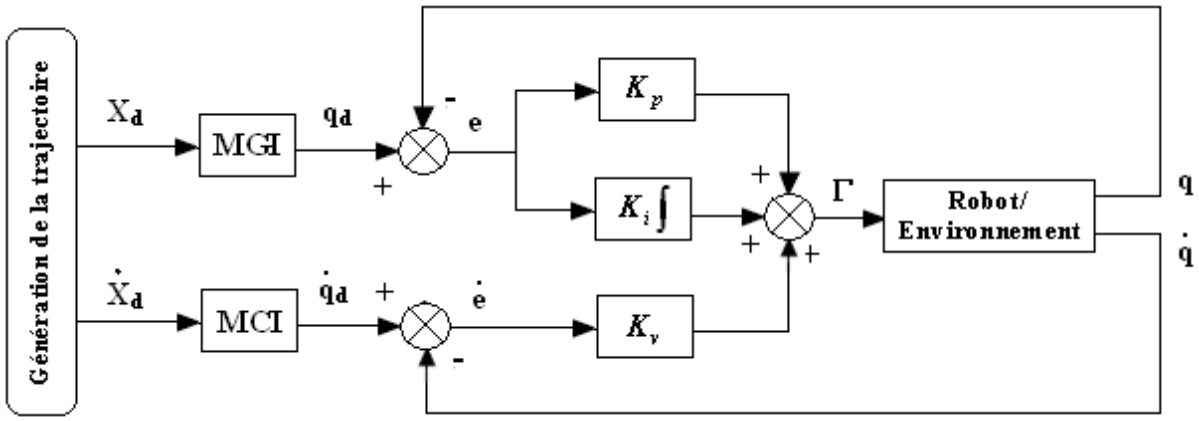

Figure 5.1 : *Loi de commande en position par correction PID*

La précision de notre système asservi peut être représentée par l'erreur quadratique moyenne **(EQM)** entre la trajectoire désirée et celle calculée. L'erreur quadratique moyenne est une performance que l'on calcule en réalisant la moyenne des carrés des erreurs sur toute la trajectoire, elle est donnée par l'équation suivante [KLR 92]:

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2 \qquad (5.2)$$

Où :

$e_i = X_d - X_c$ est l'erreur vraie ;

n est le nombre de mesures.

Plus l'EQM est petite et plus le système est précis, par conséquent on aura un bon suivi de trajectoire.

En essayant de régler les trois matrices diagonales $\mathbf{K_p}$, $\mathbf{K_v}$ et $\mathbf{K_i}$ de notre contrôleur, on a remarqué que l'EQM dépend principalement du choix de ces matrices et qu'un mauvais paramétrage de ces dernières nous mène à une EQM importante et à un mauvais suivi de trajectoire. Le résultat de notre simulation est illustré à la figure 5.2 :

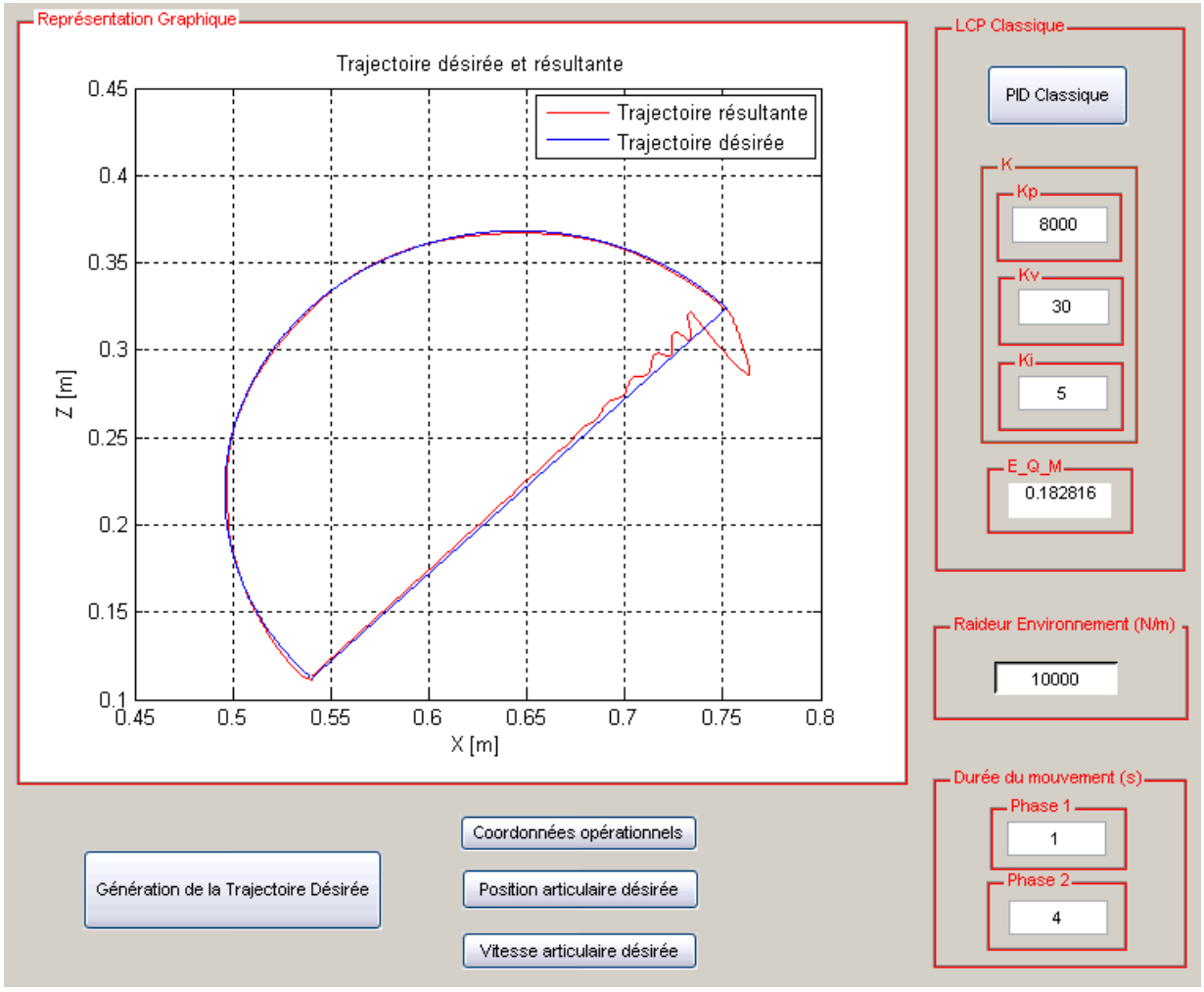

Figure 5.2 : *Poursuite de la trajectoire avec une correction PID classique*

Le contact établi avec l'environnement au moment de l'impact n'est pas maintenu, le robot réalise des rebonds sur l'environnement (figure 5.2), ces rebonds sont dus à la dynamique de l'interaction (c.à.d. l'accélération et la vitesse du robot) au moment du contact. Néanmoins, ces rebonds diminuent progressivement en amplitude jusqu'à la fin de la tâche.

En l'absence d'un contrôleur d'effort (qu'on va introduire ultérieurement) et en sachant que l'effort exercé sur l'environnement ne dépend que de l'erreur et des dérivées de la position de l'outil aux moments où ce dernier est supposé adhérer à l'environnement, la figure 5.3 montre qu'au moment de l'impact l'effort d'interaction est important et dès que le contact est rompu cet effort tend à s'annuler.

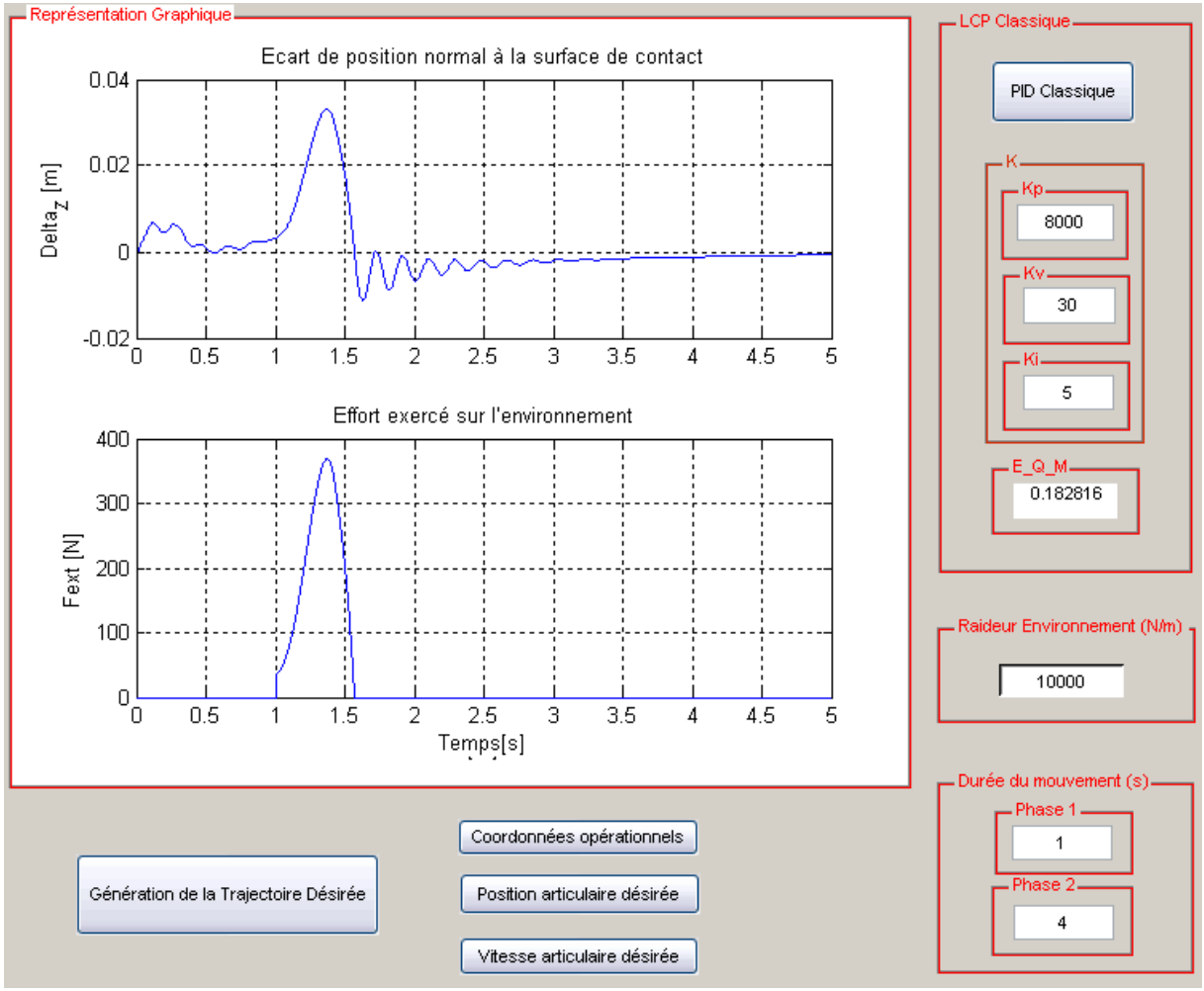

Figure 5.3 : *Ecart normal à la surface de contact et effort exercé sur l'environnement*

Remarque

Lors de notre simulation, la position et la vitesse articulaire ont été retrouvés en intégrant la dynamique du manipulateur par la méthode Runge-Kutta du 4iéme ordre (annexe 2).

La dynamique du manipulateur a été prise en considération en calculant la matrice d'inertie, les matrices des termes centrifuges et de Coriolis, la matrice gravitationnelle ainsi que les frottements secs et visqueux (annexe 1). De même pour l'influence de l'environnement où le couple d'effort, issu de l'interaction de l'effecteur avec la surface métallique, a été modélisé puis rebouclé vers l'entrée de notre système.

3. Application de l'algorithme génétique (AG) sur la loi de commande en position (LCP)

Le paramétrage du contrôleur PID est très important pour que le manipulateur puisse bien suivre sa consigne, c'est principalement à lui de déterminer l'allure de la trajectoire calculée et de voir si elle correspond à la trajectoire désirée.

Pour bien montrer la contribution des algorithmes évolutionnaires au contrôle force-position de notre bras manipulateur PUMA560, on a optimisé le PID par un algorithme génétique. Les variables à optimiser sont les trois matrices $\mathbf{K_p}$, $\mathbf{K_v}$ et $\mathbf{K_i}$.

La figure 5.4 représente le schéma d'asservissement la loi de commande en position optimisée par un algorithme génétique :

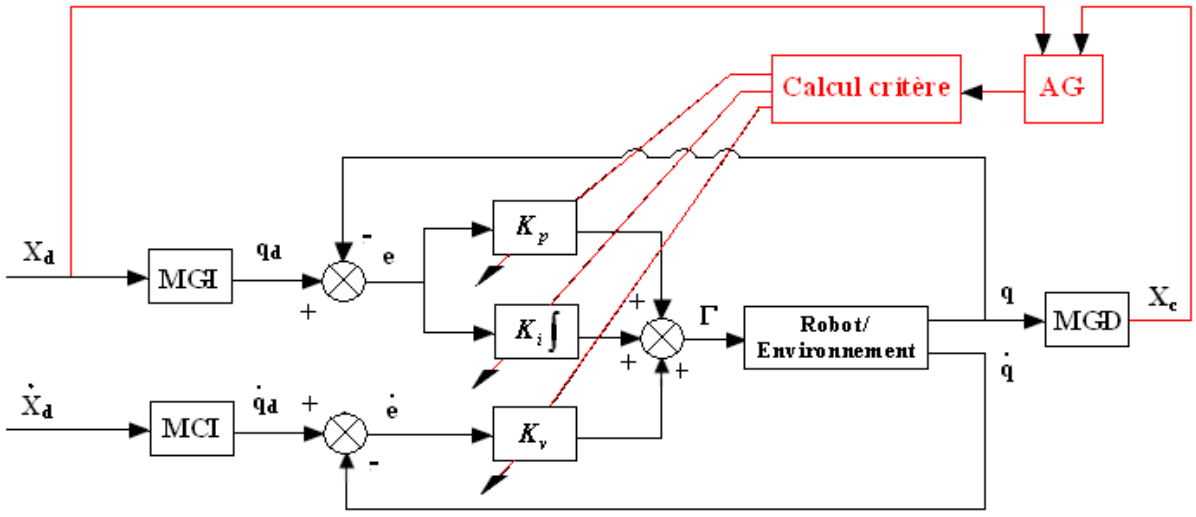

Figure 5.4 : *Application de l'AG sur la LCP*

L'algorithme génétique cherche à trouver les trois valeurs optimales **K_p', K_v'** et **K_i'** correspondant à une erreur quadratique moyenne minimale.

3.1. Genèse de la population initiale

Dans une phase d'initialisation, un ensemble d'individus est tiré aléatoirement à travers la fonction **Rand** de Matlab. On forme alors une population initiale $\mathbf{X_1}$ de taille $\mathbf{N_p}$. Le génotype de chaque individu est un vecteur de trois entiers qui représentent respectivement les matrices de gain proportionnel, dérivé et intégral. La figure 5.5 nous montre un exemple de population initiale de taille $\mathbf{N_p = 20}$:

```
X1 =

      150000         250          15
      125751         285           8
      181429         225          15
      134999         220           8
      161605         248           9
      183083         259          11
      191720         229          13
      175373         239          11
      107586         206          11
      177917         294           7
      156883         247           6
      133713         217          13
      131122         253           7
      160199         227          12
      168922         275          10
      108383         223          15
      115238         283          11
      199614         208          10
      110666         297           6
      177492         282          14
```

Figure 5.5 : *Population initiale X_1 de taille N_p=20*

Remarque

La génération de la population initiale s'est faite en imposant une certaine limite à notre espace de configuration, ainsi chaque trois entiers formant un individu ont une limite supérieure *max* et une limite inférieure *min*, un individu est représenté comme suit :

$$X_{1i} = [(K_{pi}; K_{pf})\ (K_{vi}; K_{vf})\ (K_{ii}; K_{if})] \text{ avec } i \in [1, N_p]$$

3.2. Codage du problème

L'espace de recherche $\mathbf{X_1}$ est dans notre cas décimal qu'on va convertir en un espace de recherche binaire $\mathbf{P_1}$ $\mathbf{(2^N)}$. Un point de $\mathbf{P_1}$ est alors décrit par un vecteur de **N** bits. Pour coder un élément de la configuration $\mathbf{X_1}$ on a procédé à un codage multi – paramétré, nous utilisons par exemple des vecteurs de **24 bits** : les 8 premiers bits représentent $\mathbf{K_p}$, les 8 autres $\mathbf{K_v}$ et les 8 restants $\mathbf{K_i}$ c.à.d. qu'on a concaténé les trois chaines binaires correspondantes à nos trois matrices gain en une seule suite binaire formant un individu.

Afin de coder nos variables en binaire, nous discrétisons l'espace de recherche. Ainsi si on code un gène sur 24 bits cela implique une discrétisation des intervalles en $\mathbf{g_{1max} = 2^{24}-1 = 1\ 677\ 7215}$ valeurs discrètes.

A chaque variable $\mathbf{X_{1i}}$ on associe un entier $\mathbf{g_{1i}}$, la formule de codage est la suivante :

$$g_{1i} = \frac{X_{1i} - X_{1i\min}}{X_{1i\max} - X_{1i\min}} . g_{1\max} \qquad (5.3)$$

Chaque entier $\mathbf{g_{1i}}$ obtenue sera alors converti en valeur binaire $\mathbf{P_{1i}}$ via la fonction **dec2bin** de MATLAB, de cette manière, nous discrétisons l'espace $\mathbf{X_1}$ en suites binaires. Un individu quelconque représente alors une configuration. Par exemple l'individu $\mathbf{P_{1i}}$ **= 01000001; 11011000 ; 01001100** représente la configuration $\mathbf{X_{1i}}$ **= [125751 ; 285 ; 8].**

La figure 5.6 nous montre le codage binaire de notre population $\mathbf{X_1}$:

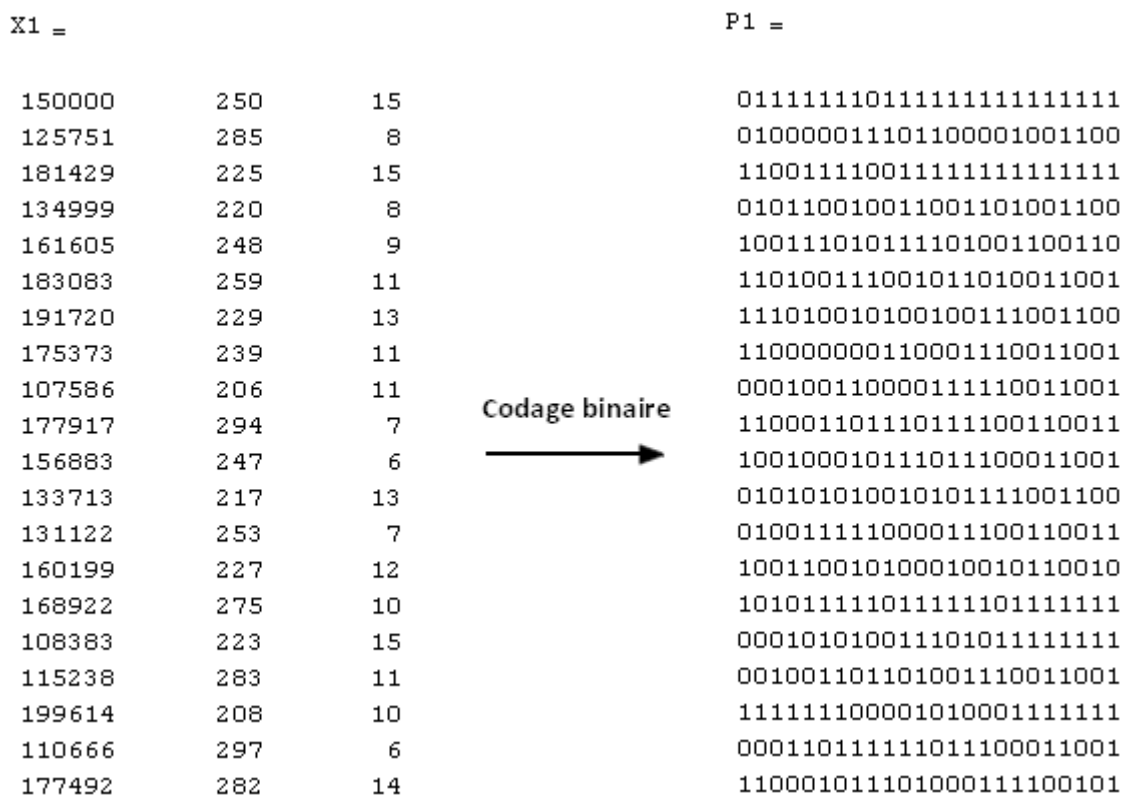

Figure 5.6 : *Codage binaire à 24 bits de la population X_1*

3.3. Résolution du problème

Le but de notre algorithme génétique est de trouver l'optimum d'une certaine fonction $\mathbf{F_1}$, qui est la fonction d'évaluation, sur l'espace de recherche $\mathbf{P_1}$. La fonction d'évaluation dans notre loi de commande en position est l'erreur quadratique moyenne.

L'algorithme génétique itère alors sur les 4 étapes suivantes jusqu' à ce qu'un optimum satisfaisant soit atteint :

3.3.1. Evaluation

L'erreur quadratique moyenne entre la trajectoire désirée et la trajectoire résultante est calculée pour chaque configuration, on aura alors un vecteur de $\mathbf{N_p}$ valeurs. La fonction $\mathbf{F_1}$ est donc établie de manière à classer toute la population du moins bon jusqu'au meilleur.

3.3.2. Sélection

La méthode de sélection prise dans notre travail est la « **Roulette wheel selection** » on peut la résumer comme suit :

- Calculer le vecteur fitness relative : $\mathbf{F_r = F_1/sum\ (F_1)}$;
- Etablir un axe gradué **E** et normalisé entre 0 et 1, il associe à chaque individu un segment dont la longueur est proportionnelle à sa fitness relative $\mathbf{F_r}$;
- On tire un nombre aléatoire de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné, et on reproduit l'individu correspondant.

Les meilleurs individus (dont l'EQM est la plus petite) ayant plus de chances d'être sélectionnés que les autres. Une nouvelle population ayant de nouveaux individus, appelés descendants, sont produits à partir de cette étape.

Remarque

1. Un individu peut être éventuellement sélectionné plusieurs fois.
2. Le cas idéal d'application de cette méthode est celui où la population est de taille infinie car sur des populations de petite taille, il est difficile d'obtenir exactement l'espérance mathématique de sélection à cause du faible nombre de tirages

3.3.3. Reproduction

La reproduction est effectuée en utilisant les opérateurs génétiques de croisement et mutation :

Croisement : La méthode de croisement utilisée dans notre travail est la " **N-point slicing crossover** " avec **N=3** et $\mathbf{P_c = 0.5}$ qui est la probabilité de croisement. Le principe de cette méthode est simple :

- On Mélange aléatoirement les individus de la population binaire $\mathbf{P_1}$, on forme ainsi une population $\mathbf{P_{1m}}$ mélangée;
- On tire un nombre aléatoire **rand** de valeur comprise entre 0 et 1, si **rand** $\leq \mathbf{P_c}$ alors on sélectionne une paire d'individus de la population $\mathbf{P_{1m}}$ et on divise chaque suite binaire formant cette paire en trois segments, on n'a qu'à faire par la suite une substitution (un swap) de ces segments, chaque individu est alors croisé avec son voisin.

La figure 5.7 représente le croisement de notre population binaire $\mathbf{P_1}$:

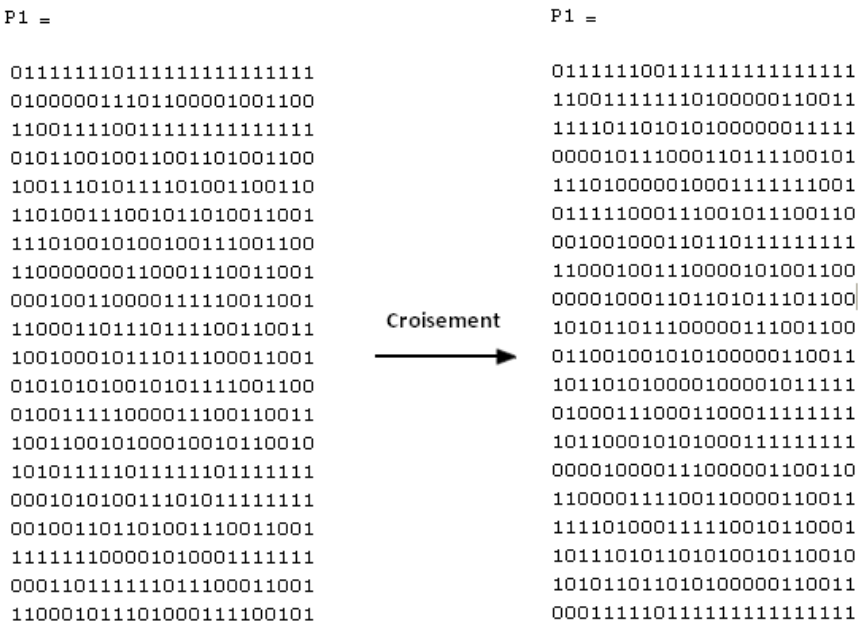

Figure 5.7 : *Croisement à 0.5 % de la population binaire P_1*

Mutation : On a utilisé une mutation par bit où on a associé une probabilité $\mathbf{P_m}$ à chaque bit qui est égale à $\mathbf{1/N_b}$ où $\mathbf{N_b}$ est le nombre de bits composant un chromosome. Cette méthode consiste à :

- Sélectionner chaque individu de la population déjà croisée $\mathbf{P_{1c}}$ et parcourir sa suite binaire bit par bit. Pour chaque bit on génère un nombre aléatoire **rand** de valeur comprise entre 0 et 1 et on le compare avec la probabilité de mutation $\mathbf{P_m}$, si **rand** $\leq \mathbf{P_m}$ alors on inverse ce bit sinon on passe au bit suivant.

La figure 5.8 représente la mutation de notre population binaire $\mathbf{P_1}$:

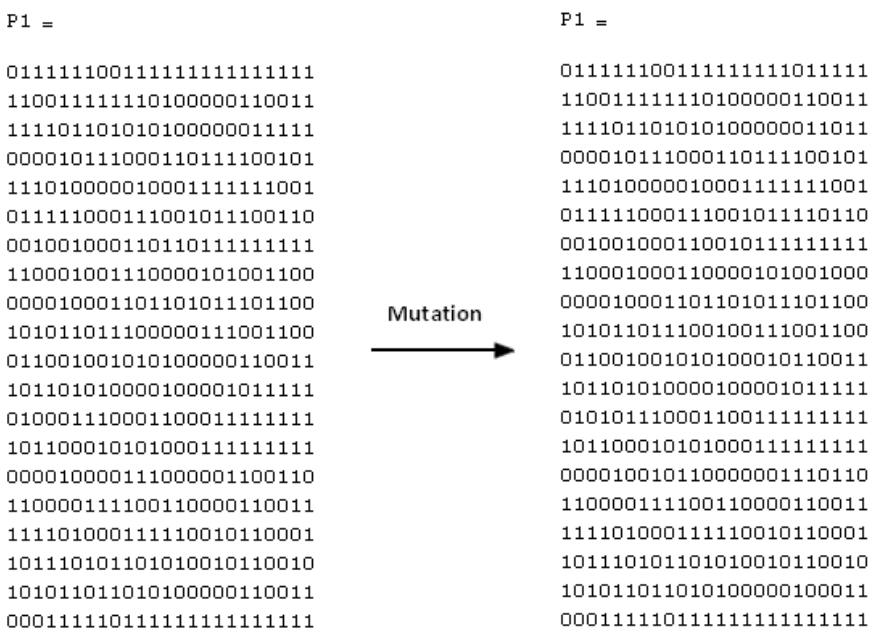

Figure 5.8 : *Mutation à $1/N_b$ de la population binaire P_1*

3.3.4. Remplacement

Une nouvelle population est engendrée en remplaçant certains des individus de la vieille population par des jeunes venant d'être crées.

On recommence ce cycle jusqu'à qu'un critère d'arrêt soit satisfait. Dans notre cas, le critère d'arrêt est une valeur fitness seuil fixée au début, si cette dernière

n'est pas atteinte alors, et à chaque itération, on recherchera la meilleure solution présente dans la population jusqu'à aboutir à un nombre de générations déjà fixé au préalable (fin du cycle).

Remarques

1. L'opération du codage est réalisée après avoir évalué la population $\mathbf{X_1}$ c.à.d. juste avant d'appliquer les opérateurs génétiques de croisement et mutation ;
2. Une opération de décodage est réalisée après l'étape de reproduction pour qu'on puisse évaluer à nouveau notre nouvelle population. Chaque individu binaire $\mathbf{P_{1i}}$ sera alors converti en un entier $\mathbf{g_{1i}}$ via la fonction **bin2dec**. Par la suite en applique la formule suivante :

$$X_{1i} = X_{1i\min} + \left(X_{1i\max} - X_{1i\min}\right).\frac{g_{1i}}{g_{1\max}} \tag{5.4}$$

La résolution du problème consiste à exécuter l'algorithme génétique décrit précédemment et montrer en pseudo code ci-dessous :

```
AG ()
    Début
        t = 0;
        Initialiser (Population)
        Evaluer (Population)
        While (condition d'arrêt)
        t = t + 1;
        / * Créer les couples * /
        Couples = Sélection (Population);
        / * Créer les descendants * /
        Descendants = Reproduction (Couples);
        / * Substituer la population actuelle par les meilleurs descendants * /
        Population = Remplacement (Descendants);
        /* Evaluer la nouvelle population */
        Evaluer (Population);
    Fin
```

Le tableau suivant représente les paramètres de notre algorithme génétique utilisé lors de la simulation :

Taille de la population $\mathbf{N_p}$	**30**
Nombre d'itérations $\mathbf{K_{max}}$	**50**
Nombre de variables **N**	**3**
Type de codage utilisé	**Binaire**
Nombre de bits représentant chaque variable $\mathbf{N_b}$	**[8 8 8]**
Nombre de bits représentant chaque chromosome $\mathbf{Nn_b}$	**24**
Méthode de sélection	**Roulette wheel selection**
Méthode de croisement	**N-point slicing crossover**
Probabilité de croisement $\mathbf{P_c}$	**0.5**
Méthode de mutation	**Bit mutation**
Probabilité de mutation $\mathbf{P_m}$	**$1/ Nn_b = 0.041$**
Espace des configurations de chaque variables	**K_p € [200000 300000]** **K_v € [100 300]** **K_i € [5 50]**

Tableau 5.1 : *Paramètres de l'algorithme génétique*

Une fois l'algorithme lancé l'évolution de la population converge vers l'individu qui permet de parvenir à un meilleur suivi de trajectoire. La configuration de cet individu est alors injectée directement dans le PID et on calcule la trajectoire issue par cette structure de commande.

Note

L'AG a été programmé en MATLAB 7.5 et exécuté sur un PC Portable Acer ASPIRE 7220 ayant les caractéristiques suivantes : Processeur AMD SEMPRON 3600+ (2Ghz), 2 Go de RAM.

Les résultats obtenus sont présentés aux figures 5.9 et 5.10:

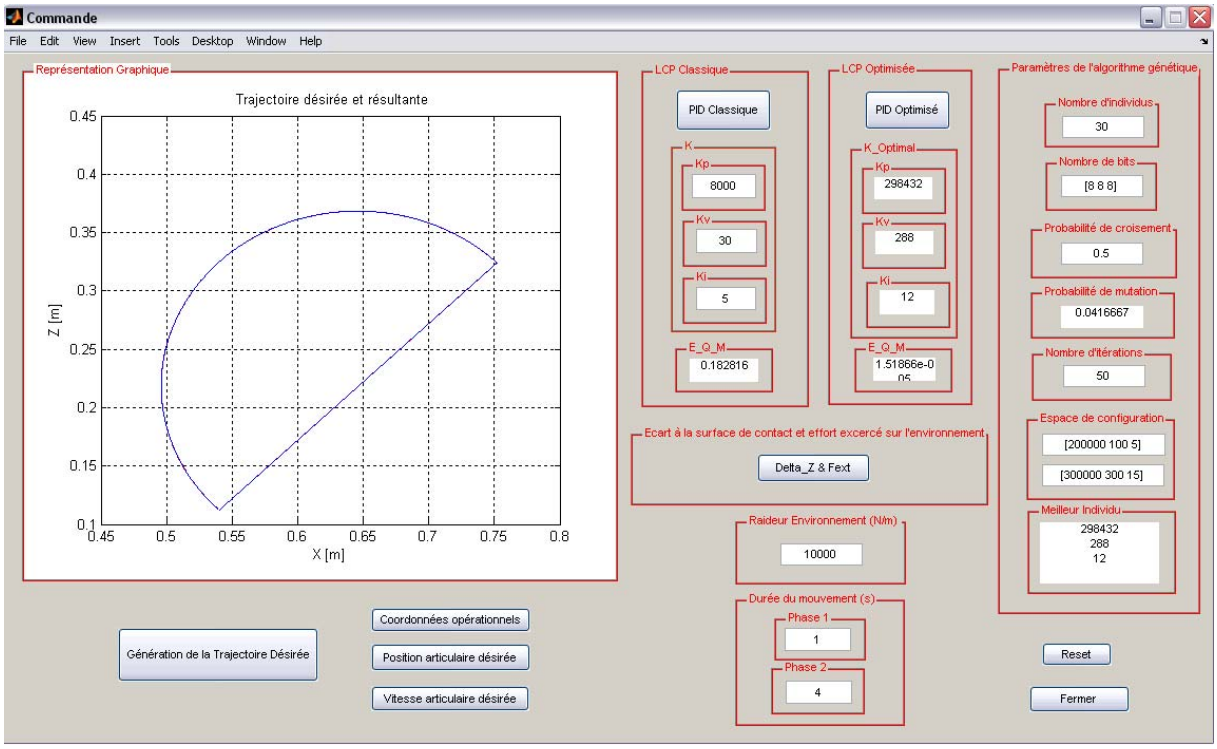

Figure 5.9 : *Poursuite de la trajectoire avec une correction PID optimisée par un AG*

On remarque après l'introduction de l'algorithme génétique que le robot suit parfaitement sa consigne et que la précision de notre système a nettement augmenté vu que l'erreur quadratique moyenne entre la trajectoire désirée et la trajectoire résultante est très faible (EQM=1.51 10^{-5}) (figure 5.9).

Par ailleurs l'effort exercé sur l'environnement a également diminué mais le système reste comme même instable car on a de fortes oscillations ce qui traduit un maintien du contact avec vibration de l'effecteur (figure 5.10), cette instabilité est induite d'un coté par la tentative du robot à donner une précision supérieure (réduire le dépassement initial montré sur la figure 5.2) et d'un autre coté par l'absence d'un contrôleur d'effort vu que notre système est commandé seulement en position.

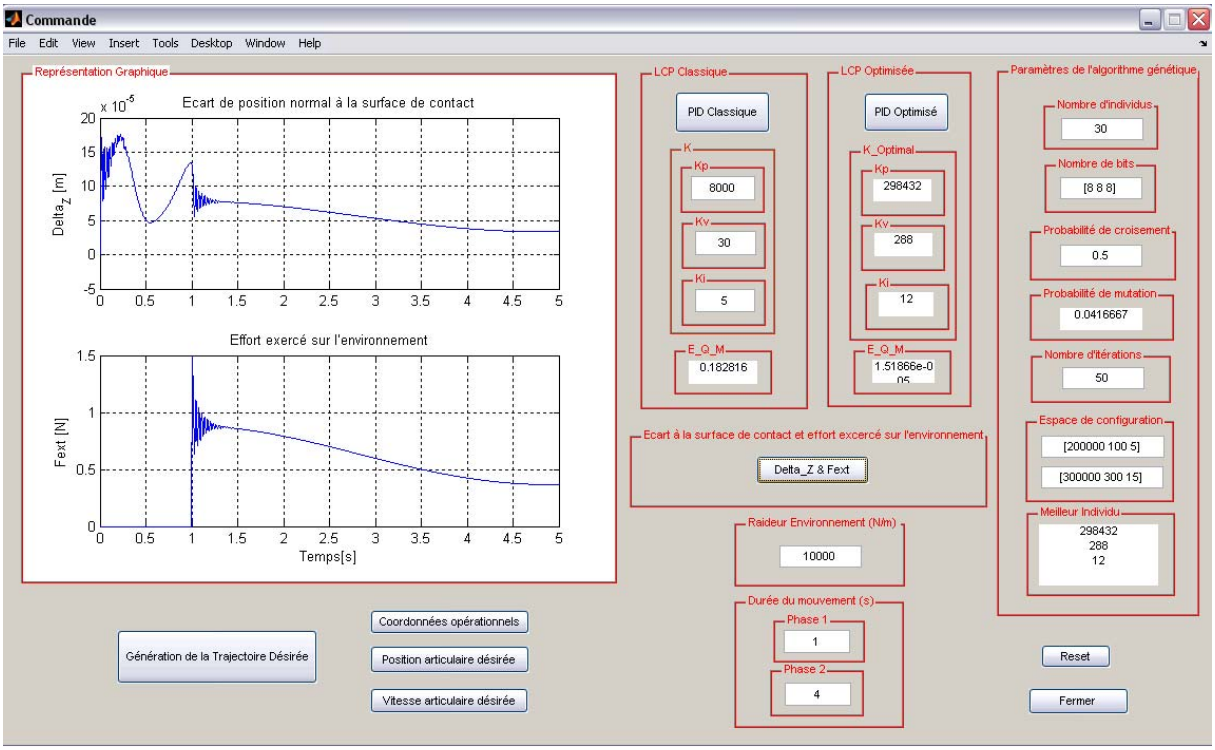

Figure 5.10 : *Ecart normal à la surface de contact et effort exercé sur l'environnement après introduction de l'AG*

Dans le paragraphe suivant on va introduire notre loi de commande en effort qui sera optimisée elle-même par un algorithme génétique.

4. Loi de commande en effort

L'asservissement de force opère dans l'espace cartésien. Pour atteindre la consigne en force $\mathbf{F_d}$ sans erreur statique, certain chercheurs considèrent qu'une simple loi de commande intégrale au niveau du contrôleur est peut être une solution bien adaptée [VOL 92].

Dans notre travail, nous avons choisi d'établir notre contrôleur d'effort à partir d'un contrôleur à action intégral optimisé par un algorithme génétique.

Le schéma de cette commande est présenté à la figure 5.11 :

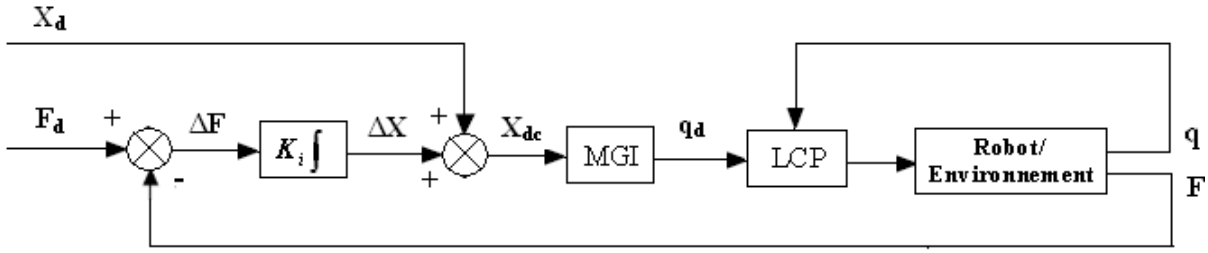

Figure 5.11 : *Loi de commande en effort par correction à action intégral*

Avec :

$$\Delta X = K_e^{-1}.K_i.\int_{t_0}^{t} \Delta F.\partial\tau \qquad (5.5)$$

Où :

K_e représente la raideur de l'environnement;

K_i représente la matrice de gain intégral;

ΔX représente l'incrément de position issu du contrôleur d'effort;

X_d représente la position cartésienne désirée;

X_{dc} représente la consigne de position cartésienne corrigée ;

ΔF représente l'erreur en effort;

F_d représente l'effort désiré;

MGI représente le modèle géométrique inverse ;

q_d représente la consigne de position articulaire ;

q représente la position articulaire mesurée ;

F représente l'effort mesuré;

LCP représente la loi de commande en position.

5. Application de l'algorithme génétique (AG) sur la loi de commande en effort (LCF)

La variable à optimiser n'est que la matrice de gain intégral $\mathbf{K_i}$. L'algorithme génétique cherche à trouver la valeur optimale $\mathbf{K_i'}$ correspondante à un dépassement et à un temps de réponse en effort minimaux.

La figure 5.12 représente le schéma d'asservissement de la loi de commande en effort optimisée par un algorithme génétique :

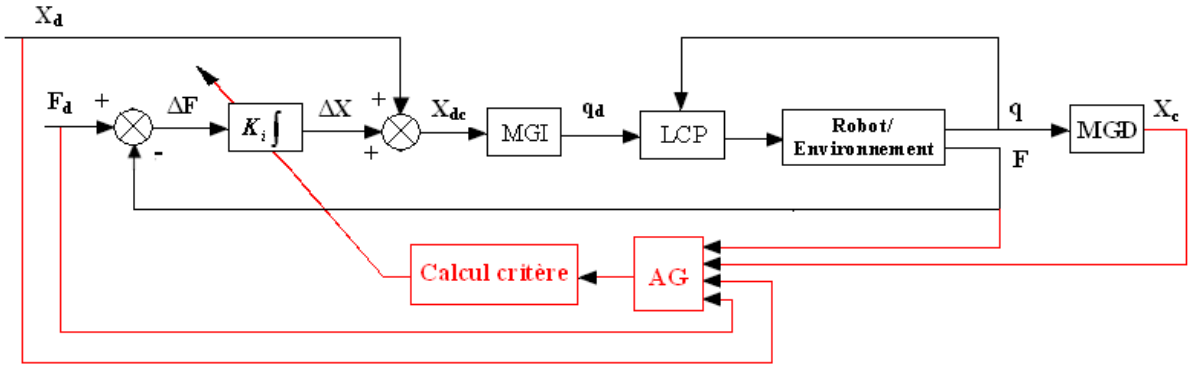

Figure 5.12 : *Application de l'AG sur la LCF*

5.1. Résolution du problème

Dans une phase d'initialisation, On forme une population initiale $\mathbf{X_2}$ de taille $\mathbf{N_p}$. Le génotype de chaque individu est un entier qui représente la matrice de gain intégral.

L'espace de recherche $\mathbf{X_2}$ sera converti en un espace de recherche binaire $\mathbf{P_2}$ $\mathbf{(2^N)}$. Pour coder un élément de la configuration $\mathbf{X_2}$, nous utilisons par exemple un vecteur de **8 bits** qui représentent le $\mathbf{K_i}$. De cette manière, nous discrétisons l'espace des configurations $\mathbf{[K_{ii} ; K_{if}]}$ en suites binaires.

Comme pour la LCP, notre algorithme génétique cherche à trouver l'optimum de la fonction d'évaluation, sur l'espace de recherche $\mathbf{P_2}$.

La fonction d'évaluation dans notre loi de commande en effort est une pondération du dépassement ainsi que le temps de réponse en effort de notre système. Notre choix s'explique par le fait que la performance (stabilité et rapidité) de notre LCF peut être caractérisée par ces deux grandeurs.

On recommence les 4 étapes mentionnées ci dessous jusqu' à atteindre un optimum satisfaisant :

5.1.1. Evaluation

Le dépassement maximal entre l'effort calculé et l'effort désiré ainsi que le temps de réponse en effort sont calculés pour chaque configuration, par la suite on fait une pondération de ces deux grandeurs. La fonction $\mathbf{F_2}$ est donc établie de manière à classer toute la population du moins bon jusqu'au meilleur.

5.1.2. Sélection

La méthode de sélection prise est la même c.à.d. la « **Roulette wheel selection** » Les meilleurs individus (dont la valeur corrélée est la plus petite) forment ainsi une nouvelle population.

5.1.3. Reproduction

La méthode de croisement utilisée est la même c.à.d. " **N-point slicing crossover** " avec **N=1** et $\mathbf{P_c = 0.5}$ comme probabilité de croisement. Une probabilité de mutation $\mathbf{P_m}$ de valeur $\mathbf{1/N_b}$ est associée à chaque bit.

5.1.4. Remplacement

Des jeunes venant d'être créés viennent pour remplacer certains des individus de la vieille population, une nouvelle population vient d'être créer. On recommence ce cycle jusqu'à qu'un critère d'arrêt soit satisfait.

Après l'introduction de la loi de commande en effort, le schéma global de notre commande en effort externe devient comme suit :

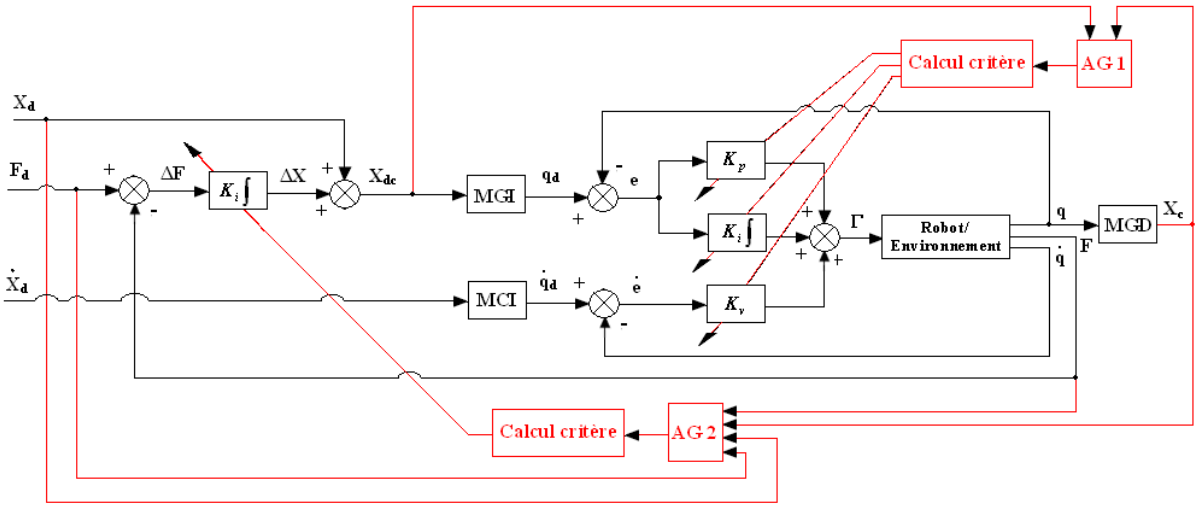

Figure 5.13 : *Application de l'AG sur la commande en effort externe*

Une fois l'effort désiré fixer on lance notre algorithme, l'évolution de la population converge vers l'individu permettant d'avoir un dépassement et un temps de réponse en effort minimaux. On injecte directement la configuration de cet individu dans le contrôleur d'effort et on calcule la trajectoire ainsi que l'effort exercé sur l'environnement grâce à notre structure en effort externe.

Les résultats obtenus sont présentés aux figures 5.14 et 5.15 :

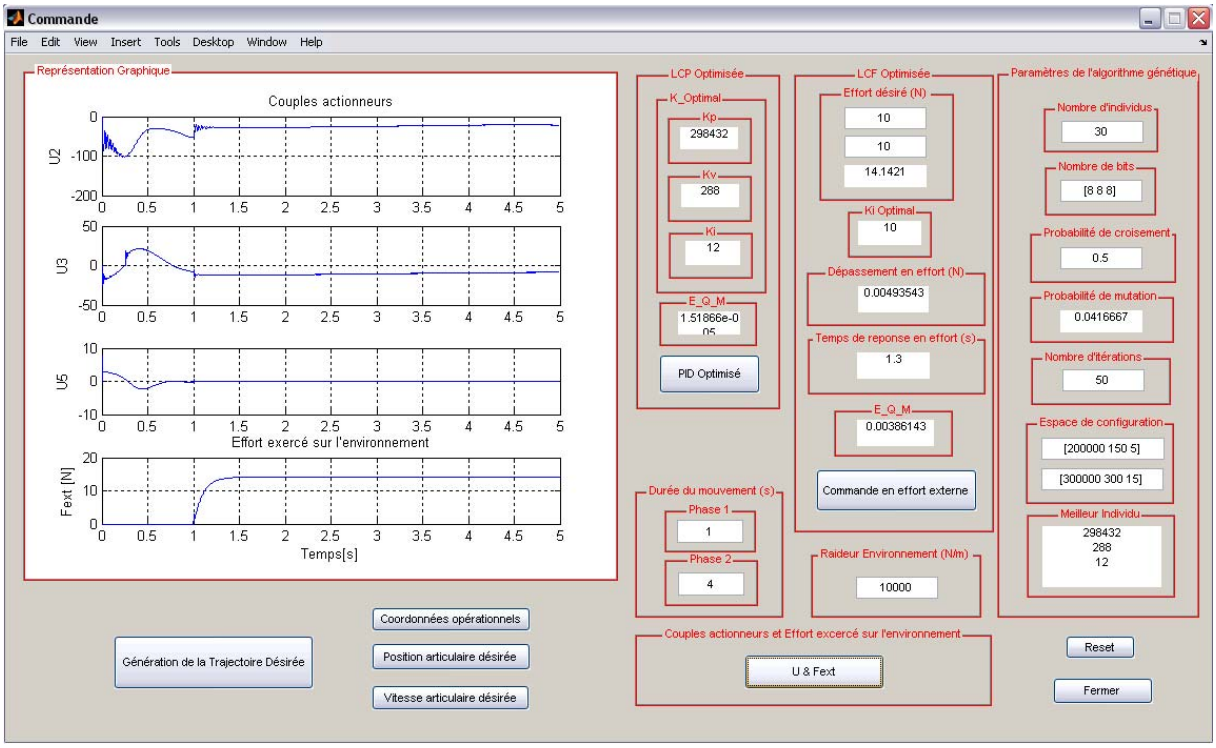

Figure 5.14 : *Couples actionneurs et effort résultant issues de la commande en effort externe après introduction de l'AG*

Après avoir observé ces résultats on peut remarquer que le régime transitoire en effort (l'impact) est doux : le dépassement maximal est très faible (0.0049N), les oscillations sont éliminées et les couples actionneurs sont lisses de plus, la valeur de l'effort exercé sur l'environnement se stabilise rapidement (temps de réponse rapide : 0.3s) à la valeur désirée et n'oscille pas autour (figure 5.14), ceci traduit un maintien du contact avec non vibration de l'effecteur, le mouvement est non vibratoire et demeure stable. Notre système est correctement amorti.

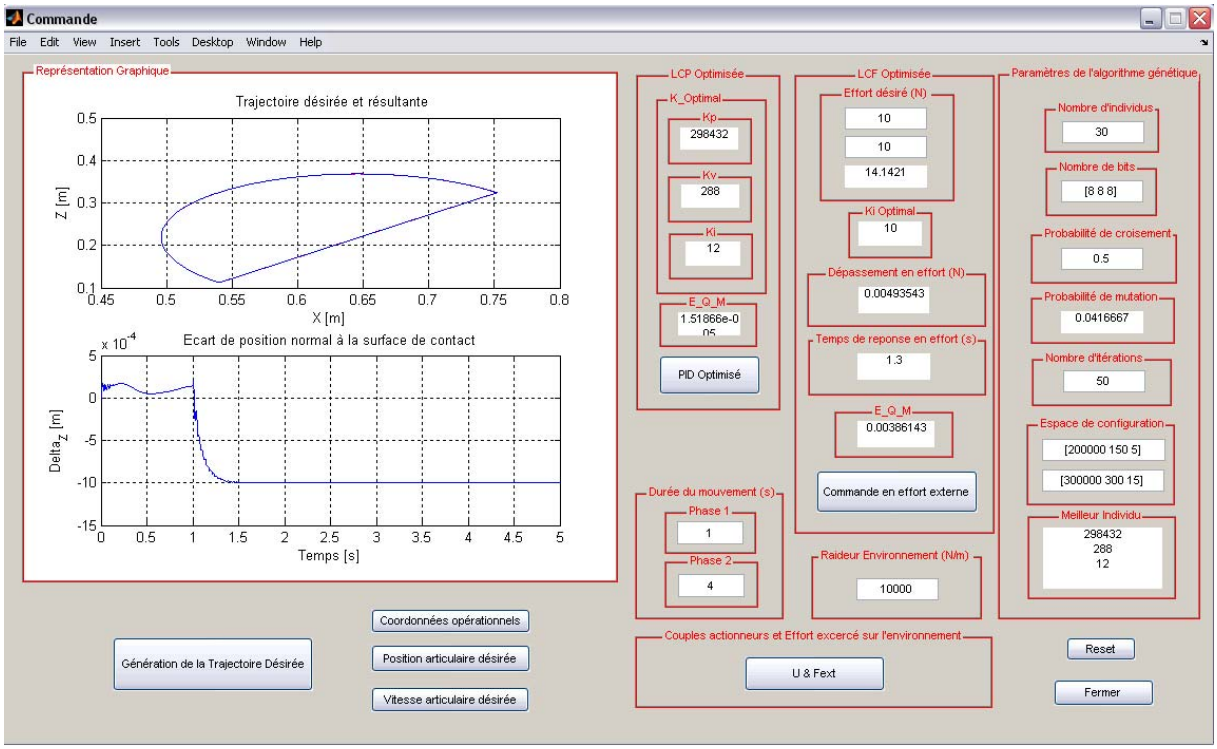

Figure 5.15 : *Poursuite et erreur de trajectoire avec une commande en effort externe optimisée par un AG*

On remarque à la figure 5.15 que le robot suit parfaitement sa consigne dans les deux phases de mouvement cependant la précision de notre système a un peu diminué vu que l'erreur quadratique moyenne s'est un peu incrémentée (EQM=0.0038), mais reste comme même faible (moins d'un millimètre), cela veut dire que si on veut garantir la stabilité dans la phase de contact on perd un peu de précision.

6. Influence des paramètres de l'algorithme génétique

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres fixés à l'avance, dont les valeurs influencent la réussite ou non d'un algorithme génétique. Ces paramètres sont : taille de la population N_p, longueur du codage de chaque individu N_b, probabilité de croisement P_c et probabilité de mutation P_m :

6.1. Taille de la population N_p

Pour commencer nous allons voir au tableau 5.2 l'influence de la taille de population sur les performances de notre système :

Taille de la population	Erreur quadratique moyenne	Dépassement maximal en effort (N)	Temps de réponse en effort (s)	Temps de calcul
$N_p = 20$	0.0045	0.0064	0. 4270	56m : 22s
$N_p = 30$	0.0038	0.0049	0.3	1h : 09m : 20s
$N_p = 60$	0.003720	0.0040	0.27	2h : 42m : 38s

Tableau 5.2 : *Influence de la taille de la population N_p*

On remarque que si $\mathbf{N_p}$ est trop grand le temps de calcul de l'algorithme peut s'avérer très important, mais les performances du système sont nettement meilleures par rapport à un $\mathbf{N_p}$ trop petit, dans ce dernier cas l'algorithme converge trop rapidement vers un mauvais chromosome. Cette importance de la taille est essentiellement due à la notion de parallélisme implicite qui implique que le nombre d'individus traité par l'algorithme est au moins proportionnel au cube du nombre d'individus. En général, la valeur de la taille de la population est comprise entre 30 et 50 individus.

6.2. Longueur du codage de chaque individu N_b

On fixe le nombre de population à 30 et on varie le nombre de bit de chaque variable,
Les résultats sont représentés au tableau 5.3 :

Nombre de bit représentant chaque variable	Erreur quadratique moyenne	Dépassement maximal en effort (N)	Temps de réponse en effort (s)	Temps de calcul
N_b = [8 8 8]	0.0038	0.0049	0.3	1h : 09m : 20s
N_b = [16 16 16]	0.0038	0.0038	0.2770	1h : 40m : 57s
N_b = [32 32 32]	0.0038	0.0031	0.26	2h : 12m : 41s

Tableau 5.3 : *Influence du nombre de bits représentant chaque variable*

On remarque bien que la précision de notre système reste la même pour les trois différentes configurations et que le dépassement et le temps de réponse ont un peu amélioré mais le temps de calcul reste quand même important, cet exemple illustre bien l'inconvénient majeur du codage binaire qui réside dans la difficulté à l'adapter dans le cas de problème d'optimisation à haute précision numérique.

6.3. Probabilité de croisement P_c

Elle dépend de la forme de la fonction sélective. Son choix est en général heuristique (tout comme pour $\mathbf{P_m}$). Plus elle est élevée, plus la population subit de changements importants. Le taux habituel est choisi entre **50%** et **100%.**

6.4. Probabilité de mutation P_m

Ce taux est généralement faible (entre 0.1% et 5%), puisqu'un taux élevé risque de conduire à une recherche trop aléatoire. Dans notre cas on a fixé $\mathbf{P_m}$ à **1/ 24 = 0.041.**

7. Conclusion

Nous nous sommes intéressés, dans ce chapitre, à l'application de l'algorithme génétique en vue de l'adapter à notre structure de commande en effort externe. Nous avons, ainsi, implémenté une première version de l'algorithme afin d'optimiser simultanément les matrices de gain de nos deux lois de commande en effort et en position, nous avons pu fournir de bons résultats préliminaires, quant à la mise en œuvre des algorithmes génétiques, que se soit pour le suivi de trajectoire ou la réponse en effort appliqué.

Par ailleurs l'algorithme génétique possède plusieurs paramètres de configuration qui ne peuvent pas être fixés arbitrairement. En effet un mauvais choix des ces paramètres a une influence néfaste sur le coût optimal. Pour faire le bon choix, nous avons étudié, à travers plusieurs essais, l'impact de ces paramètres. Ceci nous a permis d'aboutir à une version améliorée que nous avons utilisée pour une meilleure optimisation de notre structure de commande en effort externe.

Malgré nos efforts pour rendre les algorithmes génétiques plus rapides, Ils restent encore très coûteux en temps de calculs et cela peut limiter considérablement leur application en temps réel.

Conclusion générale

Nous avons présenté dans ce mémoire une nouvelle approche de commande pour traiter le problème de contrôle des robots compliants qui sont en interaction avec leur environnement.

Pour exposer notre démarche, nous avons introduit dans une première étape les différentes structures de commande force/position des robots proposées dans la littérature. Nous savons qu'une commande de cette classe possède deux boucles d'asservissement : l'une pour la correction de la position du robot et l'autre pour la correction de l'effort qu'il exerce sur son environnement.

L'analyse de ces différentes structures de commandes force/position nous a amené à choisir la structure de commande en effort externe. Cette dernière est simple à synthétiser et à mettre en œuvre. De plus, elle ne fait pas appel à une matrice de sélection encore moins aux transformations géométriques, ce qui réduit considérablement le temps de calcul.

Nous avons montré par la suite que le problème du contrôle force/position peut se ramener à la résolution d'un problème d'optimisation. Il en ressort que les algorithmes évolutionnaires sont souvent de très bonnes méthodes d'optimisation. Ceci nous a conduit à proposer une approche de commande basée sur l'utilisation d'un algorithme évolutionnaire de type génétique (AG). Nous avons ainsi décrit l'outil AG et son principe d'utilisation.

Nous avons montré la validité de l'algorithme génétique en l'appliquant parallèlement sur deux lois de commande. La première est une loi de commande en position, la deuxième une loi de commande en effort pour un bras manipulateur à six degrés de liberté effectuant une tâche de soudure et placé dans un environnement statique. L'algorithme a été défini comme étant un bloc capable de fournir une liste d'entrées pour nos deux contrôleurs et permettant d'atteindre certaines objectifs à travers deux fonctions d'adaptation.

L'algorithme génétique a montré sa capacité à adapter les différentes matrices gain de nos contrôleurs de position et d'effort, c'est-à-dire qu'il était utilisé comme un adaptateur réactif pour nos deux lois de commande.

La conception d’une structure en effort externe basée sur une approche évolutionnaire montre que les algorithmes génétiques permettent de faire une adaptation optimale des paramètres internes de cette structure de commande. Cette adaptation nous a permis de faire émerger des comportements de l'interaction du robot avec son environnement. Cette propriété est très intéressante et complémentaire aux techniques de conception classiques. Les algorithmes génétiques pourraient donc être un outil important d'aide à la conception de commandes force/position.

Bibliographie

[AN 87] C. H. AN, J. M. HOLLERBACH « *Dynamic stability issues in force control of manipulator* » Acts IEEE International Conference on Robotics and automation, Raleigh, USA, 31 mars – 3 April 1987, pp. 890-896.

[ARM 86] Armstrong B., Khatib O., BURDICK J « *The explicit dynamic model and inertial parameters of the PUMA 560 arm* » Proc. IEEE Int Conf. on robotics and automation, p. 510-518, San Francisco, April 1986.

[ARM 88] Armstrong B « *Dynamics for robot control: Friction Modelling and Ensuring Excitation During Parameter Identification* » Ph. D Thesis, Dept. Electrical Engineering, Stanford University, May 1988.

[BAC 92] T. Back « *Self-Adaptation in genetic algorithms* » Proceedings of the first European conference on Artificial Life, 1992.

[BAC 96] Bäck T « *Evolutionary Algorithms in Theory and Practice* » Oxford University Press, 1996. BU : 006.3 BAC.

[BAC 97] Bäck T., Hammel U. et Schwefel H.P « *Evolutionary Computation: Comments on the History and Current State* » IEEE Transactions on Evolutionary Computation, vol. 1, n°1, p. 3-17, April 1997.

[BE 05] Bey. A « *Commande en effort des robots manipulateurs* » Thèse de magister, ENP, Alger 2005.

[BEA 93a] Beasley D., Bull D.R. et Martin R.R « *An Overview of Genetic Algorithms: Part 1, Fundamentals* » University Computing, vol. 15, n°2, p. 58-59, 1993a.

[BEA 93b] Beasley D., Bull D.R. et Martin R.R « *An Overview of Genetic Algorithms: Part 2, Research Topics* » University Computing, vol. 15, n°4, p. 170-181, 1993b.

[BG 91] C.L. Bridges et D.E Goldberg « *An analysis of multipoint crossover* » Proceedings of the Foundation of Genetic Algorithms, 1991.

[BM 08] F. Boughias et S. Mouche « *Commande force-position d'un bras manipulateur utilisant une approche de commande intelligente : Application au suivi de trajectoire dans un milieu contraint* » Mémoire d'ingéniorat, USTHB, Alger 2008.

[CRA 89] Craig J.J « *Introduction to robotics Mechanics and Control* 2nd Edition » Addison-Wesley Publishing Company, 1989.

[DAR 59] C. Darwin « *On the Origin of Species by means of natural selection, or the Preservations of favored races in the struggle of life* » 1859.

[DASF 94] D. Delahaye, J.M Alliot, M. Schoenauer et J.L Farges « *Genetic Algorithms for partitionning airspace.Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Application* » 1994.

[DAV 91] Davis L., Ed « *Handbook of Genetic Algorithms* » New York: Van Nostrand Reinhold, 1991. BU: 511.6 HAN.

[DES 96] Dessales J « *L'ordinateur génétique* » Paris : Hermès, 1996.

[FER 08] F. Ferguene « *Contribution à la commande en effort d'un bras manipulateur* » Thèse de doctorat d'état, USTHB, Alger 2008.

[FOG 95] Fogel D.B « *Evolutionary Computation : A New Transactions* » IEEE Transactions on Evolutionary Computation , vol. 1, n°1, p. 1, avril 1997.

[FRA 94] P. FRAISSE « *Contribution à la Commande robuste Force-position des robots manipulateurs à architectures complexe : Application à un robot à deux bras* » Thèse de doctorat, Université Montpellier II, 17 février 1994.

[FSJP 93] S. Forrest, R.E. Smith, B. Jakornik et A.S. Perelson « *Using Genetic Algorithms to explore pattern recognition in the immune system* » Evolutionary computation, 1993.

[GOL 89] Goldberg D.E « *Algorithmes génétiques, exploration, optimisation et apprentissage automatique* » Paris : Ed. Addison-Wesley, 1989. BU: 006.3 GOL.

[GOL 94] David Goldberg, « *Algorithmes génétiques* » Paris : Ed. Addison Wesley Juin 1994.

[HOG 85] N. HOGAN « *Impedance control: an approach to manipulator* » Trans of ASME, J. of Dynamic Systems, Measurement, and control, Vol 107 Mars. 1985, pp. 1-24.

[HOL 75] Holland, J.H « *Adaptation in natural and artificial systems* » The University of Michigan Press, 1975.

[HSL 93] A. Homaifar, G. Shanguchuan et G.A. Liepins « *A new approach of the travelling salesman problem by genetic algorithms* » Proceedings of the fifth European conference on Genetic Algorithm, 1993.

[KAZ 86] Kazerooni H., Sheridan T., houpt P « *Robust Compliant Motion for manipulators, Parts I and II* » IEEE Journal of Robotics and automation, RA-2(2):83-105, June 1986.

[KHA 86] Khalil W., Dombre E « *Modelisation, Identification et commande des robots* » 2éme édition revue et augmentée, Hermès Automation, 1986, p.1381-1386.

[KLR 92] Kleffe, J. et Rao J.N.K « *Estimation of mean squared error of empirical best linear unbiased predictors under a random error variance linear model* » Journal of multivariate analysis, Vol43, p. 1-15. 1992.

[LAD 00] A. LO et A. LADJ « *Etude comparative de deux structures de commande hybride force-position : Application au suivi de contour* » Mémoire d'ingéniorat, USTHB, 2000.
[LIA 04] R. LIAMINI « *Commande hybride force-position d'un bras manipulateur par une approche prédictive* » Mémoire d'ingéniorat, USTHB, Alger 2000.

[MAG 06] Vincent Magnin, « *Optimisation et algorithmes génétiques* » Cours gratuits Polytech Lille/IEMN sur URL : http : // magnin.plil.net.

[MAS 81] M. T. MASON « *Compliance and force control for computer controller manipulators* » IEEE Trans. On systems, Man and Cybernetics, vol SMC-11(6) 1981, pp. 418-432.

[MEN 97] Menozzi R. et Piazzi A « *HEMT and HBT Small-Signal Model Optimization Using a Genetic Algorithm* » *EDMO'97*, Londres, p.13-18, 24-25 November 1997.

[MIC 94] Michalewicz Z « *Genetic Algorithms + Data Structures = Evolution Programs* » *Berlin*: Springer-Verlag, second edition, 1994. BU: 005.1 MIC.

[MOO 97] Moosburger R., Kostrzewa C., Fischbeck G. et Petermann K « *Shaping the Digital Optical Switch Using Evolution Strategies and BPM* » IEEE Photonics Technology Letters , vol. 9, n°11, p. 1484-1486, November 1997.

[MON, FLO 03] Francesco Mondada et Dario Floreano « *Conception évolutionniste de réseaux de neurones pour le contrôle de robots mobiles* » Publication interne : Laboratoire de microinformatique Ecole polytechnique fédérale de Lausanne, 2 juin 2003.

[NEV 73] J. L. NEVINSS, D. E. WHITNEY « *The force vector assembler concept* » Proc. 1st CISM - IFToMM Symp. On theory and Practice of Robot and Manipulators, Udine, Sept. 1973, pp. 273-288.

[NEV 77] J. L. NEVINSS et AL « *Exploratory research in industrial modular assembly* » Charles Stark Draper Lab, Report R-1111, 1977.

[PER 91] V.PERDEREAU, « *Contribution à la commande hybride force-position : application à la coopération de deux robot* » Thèse de Doctorat, Université Pierre et Marie Curie, Paris VI, Février 1991.

[PER 03] Perdereau V. and Drouin M « *A new scheme for hybrid force-position control* » Cambridge University Press, 1993, vol.11, pp. 453-464.

[PRE 97] PRELLE. Christine, « *Contribution au contrôle de la compliance d'un bras de robot à actionnement électropneumatique* » Thèse de Doctorat, Institut national des sciences appliquées de Lyon. Chapitre 1, pp. 21, 1997.

[QP 93] X. Qi et F. Palmieri « *The diversification role of the crossover in the genetic algorithm* » Proceedings of fifth European conference on Genetic Algorithm, p. 132-137, Urbana-Champaign, 1993.

[REB 85] C. REBOULET and A. ROBERT « *Hybrid control of manipulator equipped with an active compliant wrist* » Proc. 3rd Int. Symp. Of Robotics Research, Gouvieux, France pp. 237-241, Octobre 1985.

[REI 97] Reineix A., Eclercy D. et Jecko B « *FDTD/genetic algoritm coupling for antennas optimization* » Annales de Télécommunications, vol. 52, n°9-10, 1997.

[REN 95] Renders J.M « *Algorithmes génétiques et réseaux de neurones* » Paris : Hermès, 1995. BU : 066.3 REN.

[SAA 97] N. SAADIA « *Contribution à la commande hybride force-position des robots compliants selon une approche neuronale* » Thèse de Doctorat, Université Paris XII 1997.

[SAL 80] J. K. SALISBURY « *Active Stiffness control of a manipulator in Cartesian coordinates* » Proc. 19 th IEEE Conf. On Decision and Control, Albuquerque, Dec. 1980, p. 95-100.

[SCH 88] J. DE SCHUTTER et H. VAN BRUSSEL « *Compliant robot motion II. A control approach based on external loop* » Int. Journal of Robotics research, Vol.4 n°04, p. 18-33, August 1988.

[SIC 96] Chiaverini, S., Siciliano, B., & Villani, L « *Parallel force/position control schemes with experiments on an industrial manipulator* » In Proceedings of the IFAC 13th triennial world congress, San Francisco, pp. 25-30. 1996.

[SOR, WYL 06] Thierry Sorg et Marc von Wyl « *Applications d'algorithmes évolutionnistes à des problèmes de génie logiciel* » 30 Mai 2006.

[SYS 89] G. Syswerda « *Uniform crossover in genetic algorithms* » Proceedings of the third European Conference on Genetic Algorithm, p. 2 – 9, George Mason University, United States, 1989.
.

[TOL 03] Tollari Sabrina « *Algorithmes génétiques pour résoudre le problème du commis voyageur* » Laboratoire des sciences de l'information et des systèmes, 23 Mai 2003.

[VOL 90] Volpe R.A « *Real and Artificial forces in the control of manipulators: Theory and Experiments* » Ph. D Thesis, Dep. Physics, Carnegie Mellon university, Pittsburgh, September 1990.

[VOL 92] R.VOLPE, P.KHOSLA « *An experimental evaluation and comparison of explicit force control strategies for robotic manipulators* » In Proc of the 1992 IEEE Int.Conf. On Robotics and Automation, p. 1387-1393, Nice, France, May, 1992.

[WAL 91] Waldron, K.J. et Hunt, K.H « *Series-parallel dualities in actively coordinated mechanisms* » The Int. Journal of Robotics Research, Vol. 10, No. 2, pp. 473- 480. 1991.

[WHI 79] D. E. WHITNEY, J. L. NEVINS « *What is the remote center compliance (RCC) and what can it do* » Proc. 9th Int. Symp. On industrial Robots, Washington, mars 1979, pp. 135-147.

[WHI 85] Whitley. D « *Histirical perspective and stade of the art in robot force control* » Proc. IEEE Conf. On Robotics and Automation, St Louis, mars 1985, pp. 262-268.

[WHI 93] Whitley. D « *A Genetic Algorithm Tutorial* » Technical Report CS-93-103, Colorado State University, Department of Computer Science, 1993.

[WILL 91] Wills. C « *La sagesse des gènes, nouvelles perspectives sur l'évolution* » Flammarion, 1991.

Annexe

Le lecteur trouvera dans cette annexe, les paramètres du modèle géométrique et dynamique du PUMA 560 d'Unimation [ARM 86].

1. Paramètres du modèle géométrique et dynamique du PUMA 560

Matrice d'inertie (A)

L'unité des éléments est [Kg-m^2]

a_{11}=2.57+1.38*(cos(q(2)))^2 + 0.30*(sin(q(2) +q(3)))^2 + 0.744*cos(q(2))*sin(q(2) + q(3));
a_{12}=0.69*sin(q(2))-0.134*cos(q(2)+q(3))+0.0238*cos(q(2));
a_{13}=-0.134*cos(q(2)+q(3))-0.00397*sin(q(2)+q(3));
a_{14}= 0;
a_{15}= 0;
a_{16}= 0;
a_{22}= 6.79+0.744*sin (q(3));
a_{23}= 0.333+0.372*sin (q(3))-0.011*cos (q(3)) ;
a_{24}= 0;
a_{25}= 0;
a_{26}= 0;
a_{33}= 1.16 ;
a_{34}=-0.00125*sin (q(4))*sin (q(5));
a_{35}= 0.00125*cos (q(4))*cos (q(5));
a_{36}= 0;
a_{44}= 0.20;
a_{45}= 0;
a_{46}= 0;
a_{55}= 0.18;
a_{56}= 0;
a_{66}= 0.19;

A = [a_{11} a_{12} a_{13} a_{14} a_{15} a_{16};
a_{12} a_{22} a_{23} a_{24} a_{25} a_{26} ;
a_{13} a_{23} a_{33} a_{34} a_{35} a_{36} ;
a_{14} a_{24} a_{34} a_{44} a_{45} a_{46} ;
a_{15} a_{25} a_{35} a_{45} a_{55} a_{56} ;
a_{16} a_{26} a_{36} a_{46} a_{56} a_{66}];

Matrice des couples de Coriolis (B)

b_{112}=-2.76*sin(cos(q(2)))+0.744*cos(2*q(2)+q(3))+0.60*sin(cos(q(3)+q(2)))- 0.0213*(1-2*sin(sin(q(2)+q(3)))) ;
b_{113}=0.744*cos(q(3))*cos(q(2)+q(3))+0.6*sin(cos(q(2)+q(3)))+ 0.022*cos(q(2))*sin(q(2) + q(3)) - 0.0213*(1-2*sin(sin(q(2)+q(3)))) ;
b_{114} = -0.0025*sin(cos(q(2) + q(3)))*sin(q(4))*sin(q(5)) + 0.00086*cos(q(4))*sin(q(5)) - 0.00248*cos(q(2))*cos(q(2)+q(3))*sin(q(4))*sin(q(5)) ;
b_{115}=-0.0025*(sin(sin(q(2)+q(3)))*sin(q(5)) - sin(cos(q(2) + q(3)))*cos(q(4))*cos(q(5))) - 0.00248*cos(q(2))*(sin(q(2) + q(3))*sin(q(5)) - cos(q(2) + q(3))*cos(q(4))*cos(q(5))) + 0.00086*sin(q(4))*cos(q(5)) ;
b_{116} = 0 ;
b_{123} = 0.267*sin (q(2)+q(3)) - 0.00758*cos(q(2)+q(3)) ;
b_{124} = 0 ;
b_{125} = 0 ;
b_{126} = 0 ;
b_{134} = b_{124};
b_{135} = b_{125} ;
b_{136} = b_{126} ;
b_{145} = 0 ;
b_{146} = 0 ;
b_{156} = 0 ;
b_{212} = 0 ;
b_{213} = 0 ;
b_{214} = 0.00164*sin(q(2)+q(3)) -0.0025*cos(q(2)+q(3))*cos(q(4))*sin(q(5)) + 0.00248*sin(q(2))*cos(q(4))*sin(q(5)) + 0.0003*sin(q(2)+q(3))*(1-(2*sin(sin(q(4))))) ;
b_{215}=-0.0025*cos(q(2)+q(3))*sin(q(4))*cos(q(5))+ 0.00248*sin(q(2))*sin(q(4))*cos(q(5)) - 0.000642*cos(q(2)+q(3))*sin(q(4)) ;
b_{216} =- b_{126} ;
b_{223} = 0.022*sin (q(3)) + 0.744*cos (q(3)) ;
b_{224} = -0.00248*cos (q(3))*sin (q(4))*sin (q(5)) ;
b_{225} = -0.0025*sin (q(5)) + 0.00248*(cos (q(3))*cos (q(4))*cos ((5)) - sin (q(3))*sin (q(5))) ;
b_{226} = 0 ;
b_{234} = b_{224} ;
b_{235} = b_{225} ;
b_{236} = 0 ;
b_{245} = 0 ;
b_{246} = 0 ;
b_{256} = 0 ;
b_{312} = 0 ;
b_{313} = 0 ;

b_{314}=-0.0025*cos(q(2)+q(3))*cos(q(4))*sin(q(5))+0.00164*sin(q(2)+q(3))+ 0.0003*sin(q(2)+q(3))*(1-2*sin(sin(q(4)))) ;
b_{315} = - 0.0025*cos (q(2) +q(3))*sin (q(4))*cos (q(5))-0.000642*cos (q(2) +q(3))*sin (q(4)) ;
b_{316} = - b_{136} ;
b_{323} = 0 ;
b_{324} = 0 ;
b_{325} = -0.25*sin (q(5)) ;
b_{326} = 0 ;
b_{334} = b_{324} ;
b_{335} = b_{325} ;
b_{336} = 0 ;
b_{345} = -0.0025*sin (q(4))*cos (q(5)) ;
b_{346} = b_{246} ;
b_{356} = b_{256} ;
b_{412} = - b_{214} ;
b_{413} = - b_{314} ;
b_{414} = 0 ;
b_{415} = -0.000642*sin (q(2) +q(3))*cos (q(4)) ;
b_{416} = - b_{146} ;
b_{423} = - b_{324} ;
b_{424}= 0 ;
b_{425}=0.000642*sin (q(4)) ;
b_{426}=- b_{246} ;
b_{434}= 0 ;
b_{435}= b_{425};
b_{436}=- b_{346};
b_{445}= 0 ;
b_{446}= 0 ;
b_{456}= 0 ;
b_{512}=- b_{215} ;
b_{513}=- b_{315};
b_{514}=- b_{415} ;
b_{515}= 0 ;
b_{516}=- b_{156} ;
b_{523}=- b_{325};
b_{524}=- b_{425} ;
b_{525}= 0 ;
b_{526}=- b_{256} ;
b_{534}= b_{524} ;
b_{535}= 0 ;
b_{536}=- b_{356};
b_{545}= 0 ;

b_{546}=- b_{456} ;
b_{556}= 0 ;
b_{612}= b_{126};
b_{613}= b_{136} ;
b_{614}= b_{146};
b_{615}= b_{156};
b_{616}= 0 ;
b_{623}= 0 ;
b_{624}= b_{246} ;
b_{625}= b_{256};
b_{626}= 0 ;
b_{634}= b_{624} ;
b_{635}= b_{625} ;
b_{636}= 0 ;
b_{645}= b_{456};
b_{646}= 0 ;
b_{656}= 0 ;

Prod_v = [v(1)*v(2) v(1)*v(3) v(1)*v(4) v(1)*v(5) v(1)*v(6) v(2)*v(3) v(2)*v(4) v(2)*v(5) v(2)*v(6) v(3)*v(4) v(3)*v(5) v(3)*v(6) v(4)*v(5) v(4)*v(6) v(5)*v(6)];

B = [b_{112} b_{113} b_{114} b_{115} b_{116} b_{123} b_{124} b_{125} b_{126} b_{134} b_{135} b_{136} b_{145} b_{146} b_{156} ;
b_{212} b_{213} b_{214} b_{215} b_{216} b_{223} b_{224} b_{225} b_{226} b_{234} b_{235} b_{236} b_{245} b_{246} b_{256} ;
b_{312} b_{313} b_{314} b_{315} b_{316} b_{323} b_{324} b_{325} b_{326} b_{334} b_{335} b_{336} b_{345} b_{346} b_{356} ;
b_{412} b_{413} b_{414} b_{415} b_{416} b_{423} b_{424} b_{425} b_{426} b_{434} b_{435} b_{436} b_{445} b_{446} b_{456} ;
b_{512} b_{513} b_{514} b_{515} b_{516} b_{523} b_{524} b_{525} b_{526} b_{534} b_{535} b_{536} b_{545} b_{546} b_{56} ;
b_{612} b_{613} b_{614} b_{615} b_{616} b_{623} b_{624} b_{625} b_{626} b_{634} b_{635} b_{636} b_{645} b_{646} b_{656}]*(Prod_v)^{T};

Matrice des couples centrifuges (C)

c_{11}= 0 ;
c_{12}= 0.69*cos (q(2)) +0.134*sin (q(2) +q(3))-0.0238*sin (q(2)) ;
c_{13}= 0.5*b_{123} ;
c_{14}= 0 ;
c_{15}= 0 ;
c_{16}= 0 ;
c_{21}=-0.5*b_{112} ;
c_{22}= 0 ;
c_{23}= 0.5*b_{223} ;
c_{24}= 0 ;
c_{25}= 0 ;
c_{26}= 0 ;

c_{31}=-0.5*b_{113} ;
c_{32}=-c_{23} ;
c_{33}= 0 ;
c_{34}=-0.00125*cos (q(4))*sin (q(5)) ;
c_{35}= c_{34} ;
c_{36}= 0 ;
c_{41}=-0.5*b_{114} ;
c_{42}=-0.5*b_{224} ;
c_{43}= 0.5*b_{423} ;
c_{44}= 0 ;
c_{45}= 0 ;
c_{46}= 0 ;
c_{51}=-0.5*b_{115} ;
c_{52}=-0.5*b_{225} ;
c_{53}= 0.5*b_{523} ;
c_{54}=-0.5*b_{445} ;
c_{55}= 0 ;
c_{56}= 0 ;
c_{61}= 0 ;
c_{62}= 0 ;
c_{63}= 0 ;
c_{64}= 0 ;
c_{65}= 0 ;
c_{66}= 0 ;

C = [c_{11} c_{12} c_{13} c_{14} c_{15} c_{16} ;
c_{21} c_{22} c_{23} c_{24} c_{25} c_{26} ;
c_{31} c_{32} c_{33} c_{34} c_{65} c_{36} ;
c_{41} c_{42} c_{43} c_{44} c_{45} c_{46} ;
c_{51} c_{52} c_{53} c_{54} c_{55} c_{56} ;
c_{61} c_{62} c_{63} c_{64} c_{65} c_{66}]*(v)^2;

Vecteur des couples de gravité (G)

g_1= 0 ;
g_2=-37.2*cos (q(2))-8.4*sin (q(2) +q(3)) +1.02*sin (q(2)) ;
g_3=-8.4*sin (q(2) +q(3)) +0.25*cos (q(2) +q(3)) ;
g_4= 0.028*sin (q(2) +q(3))*sin (q(4))*sin (q(5)) ;
g_5=-0.028*(cos (q(2) +q(3))*sin (q(5)) +sin (q(2) +q(3))*cos (q(4))*cos (q(5))) ;
g_6= 0 ;

G = [g_1 g_2 g_3 g_4 g_5 g_6] ^{T};

Vecteur des frottements (F)

f_1= (-8.26+3.45*v(1))*(sign (v(1)) == -1) + (8.43+4.94*v(1))*(sign (v(1)) == 1);
f_2= (-11.34+8.53*v(2))*(sign (v(2)) == -1) + (12.77+7.67*v(2))*(sign (v(2)) == 1);
f_3= (-5.57+3.02*v(3))*(sign (v(3)) == -1) + (5.93+3.27*v(3))*(sign (v(3)) == 1);
f_4=0;
f_5=0;
f_6=0;

F = [f_1 f_2 f_3 f_4 f_5 f_6]^T;

Le lecteur trouvera dans cette annexe la méthode d'intégration numérique utilisée pour le calcule de la position ainsi que la vitesse articulaire de notre manipulateur PUMA 560.

2. Méthode de RUNGE-KUTTA d'ordre 4

Considérons une équation différentielle du premier ordre :

$$\frac{dy}{dt} = f(t, y)$$

La méthode **RK4** utilise plusieurs points intermédiaires pour calculer la valeur de $\mathbf{y_{i+1}}$ à partir de la valeur $\mathbf{y_i}$.

On considère un point intermédiaire **A** d'abscisse $\mathbf{t_i+h/2}$ dont la valeur de l'ordonnée est donnée par :

$$y_{iA} = y_i + \left(\frac{dy}{dt}\right)_i \cdot \frac{h}{2} \quad \text{soit} \quad y_{iA} - y_i = + \left(\frac{dy}{dt}\right)_i \cdot \frac{h}{2} = \frac{k_1}{2}$$

Puis un point **B** d'ordonnée :

$$y_{iB} = y_i + \left(\frac{dy}{dt}\right)_{iA} \cdot \frac{h}{2} \quad \text{soit} \quad y_{iB} - y_i = + \left(\frac{dy}{dt}\right)_{iA} \cdot \frac{h}{2} = \frac{k_2}{2}$$

On calcule alors l'ordonnée d'un point **C** d'abscisse $\mathbf{t_i+h}$ à l'aide de la relation :

$$y_{iC} = y_i + \left(\frac{dy}{dt}\right)_{iB} .h \quad \text{soit} \quad y_{iC} - y_i = + \left(\frac{dy}{dt}\right)_{iB} .h = k_3$$

Soit $\left(\frac{dy}{dt}\right)_{iC}$ la valeur de $\left(\frac{dy}{dt}\right)$ au point **C**. on pose : $\left(\frac{dy}{dt}\right)_{iC} .h = k_4$

L'ordonnée définitive $\mathbf{y_{i+1}}$ du point d'abscisse $\mathbf{t_i+h}$ est donnée par la relation :

$$y_{i+1} = y_i + \frac{1}{6}\left[\left(\frac{dy}{dt}\right)_i + 2.\left(\frac{dy}{dt}\right)_{iA} + 2.\left(\frac{dy}{dt}\right)_{iB} + \left(\frac{dy}{dt}\right)_{iC}\right].h$$

La méthode de **RUNGE KUTTA** d'ordre 4 définit deux suites, **h** étant le pas de discrétisation en **t** :

- Une première qui permet de définir les valeurs de **t** : $\mathbf{t_{i+1}=t_i+1}$ **;**
- Une deuxième qui permet d'évaluer les valeurs de **y** :

$$y_{i+1} = y_i + \frac{1}{6}\left[k_1 + 2.k_2 + 2.k_3 + k_4\right]$$

Avec :

$$k_1 = h.f(t_i, y_i)$$
$$k_2 = h.f(t_i + h/2, y_i + k_1/2)$$
$$k_3 = h.f(t_i + h/2, y_i + k_2/2)$$
$$k_4 = h.f(t_i + h, y_i + k_3)$$

Résumé

Dans l'exécution de certaines tâches par un bras manipulateur, une commande force/position est nécessaire car elle permet à celui ci de prendre en compte l'interaction avec son environnement pour mieux adapter son comportement. Durant ces dernières années, afin d'améliorer les performances des manipulateurs, des recherches avancées ont permis de faire émerger de nouvelles approches de commande appliquées aux robots manipulateurs.

Nous avons présenté dans ce mémoire un cadre méthodologique pour la synthèse d'une nouvelle approche de commande de tels systèmes. Cette méthode générique s'appuie sur l'utilisation d'un algorithme évolutionnaire, de type génétique, utilisant une technique d'optimisation stochastique et qui fait partie du champ de l'Intelligence Artificielle.

L'objectif est de montrer la capacité des algorithmes évolutionnaires pour traiter le problème de la commande des systèmes compliants actifs dont les robots manipulateurs constituent un exemple représentatif.

L'algorithme génétique a pour rôle d'optimiser les paramètres internes de notre structure force/position qui est la commande en effort externe. Cette dernière comporte une loi de commande en effort à action intégrale et une loi de commande en position à partir d'un correcteur PID.

L'implantation d'une tâche de suivi de trajectoire, effectuée par un bras manipulateur PUMA 560, dans un milieu contraint est donnée en exemple pour tester et évaluer notre approche.