# Métaheuristiques pour la planification de trajectoire des bras manipulateurs redondants : application à l'assistance au geste chirurgical en craniotomie

Riad Menasri

# THESE

Pour l'obtention du grade de

## DOCTEUR DE L'UNIVERSITE PARIS-EST

ECOLE DOCTORALE MATHEMATIQUES ET STIC (MSTIC, E.D. 532)

Présentée par


# Riad Menasri


# Métaheuristiques pour la planification de trajectoire des bras manipulateurs redondants. Application à l'assistance au geste chirurgical en craniotomie.

Soutenue le 1er décembre 2015

**JURY**

| | | | |
|---|---|---|---|
| Arab Ali Cherif | Professeur | Université Paris 8 | Président |
| Jin-Kao Hao | Professeur | Université d'Angers | Rapporteur |
| Véronique Perdereau | Professeur | Université Paris 6 | Rapporteur |
| Philippe Decq | Professeur | ENSAM | Examinateur |
| Maxime Gautier | Professeur | Université de Nantes | Examinateur |
| Hamouche Oulhadj | Maître de Conférences | Université Paris-Est Créteil | Examinateur |
| Boubaker Daachi | Professeur | Université Paris 8 | Co-directeur de thèse |
| Patrick Siarry | Professeur | Université Paris-Est Créteil | Directeur de thèse |

# Remerciements

Mes premiers remerciements à Dieu, tout puissant, clément et miséricordieux de m'avoir donné le courage et la persévérance pour accomplir ce travail.

Je tiens à exprimer mes plus vifs remerciements au Prof. Patrick Siarry et au Prof. Boubaker Daachi qui m'ont donné une chance pour réaliser ce travail.

Un grand merci à tous ceux qui ont participé à l'encadrement, Patrick Siarry , Boubaker Daachi, Hamouche Oulhadj et Amir Nakib, de m'avoir accordé une grande liberté au cours de ces trois années.

Merci au Prof. Philippe Decq, qui m'a aidé sur la manipulation du robot Neuromate et aussi à Amal Benslimane qui a joué un rôle important dans la réalisation de l'application décrite dans ce travail.

Mes remerciements vont également à mes rapporteurs Monsieur Jin-Kao Hao et Madame Véronique Perdereau. Je les remercie pour le soin avec lequel ils ont lu ce manuscrit, ainsi que pour la qualité de leurs critiques. Je remercie Monsieur Arab Ali Cherif et Monsieur Maxime Gautier d'avoir accepté de participer à ce jury de thèse.

Je tiens à adresser des remerciements plus décontractés, mais néanmoins chaleureux et sincères, aux autres doctorants, anciens doctorants et stagiaires du laboratoire LISSI. Un remerciement spécial à Thibaud Rohmer pour son aide.

Enfin, je dédie cette thèse à mes parents, qui m'ont toujours encouragé et soutenu pendant toutes mes années d'études. A ma femme, pour son soutien, ses encouragements dans les moments de doute. Merci d'avoir toujours cru en moi et de m'avoir donné les moyens d'accomplir ce travail. A mon frère et mes sœurs et aux anges de la famille Anis et Sirine.

# Résumé

Le problème de planification de trajectoire des bras manipulateurs redondants est largement étudié dans la littérature. Sa résolution nécessite la prise en compte d'un certain nombre de contraintes, qui sont :

— le calcul des différentes configurations par lesquelles le robot doit passer ;
— l'obtention de courbes lisses (vitesses, accélérations, *jerks*).

La prise en compte de ces deux contraintes dans la démarche de résolution peut se faire de deux manières différentes. La première consiste à supposer au préalable que les différentes courbes suivent des trajectoires lisses (utilisation de fonctions polynomiales ou trigonométriques). La résolution aura pour objectif de calculer les paramètres de chacune des courbes. La deuxième technique consiste à traiter les deux contraintes séparément. Ainsi, on calcule les différentes configurations, puis on procède à une interpolation. Outre ces deux contraintes, on doit aussi résoudre le problème de redondance du robot et la manière de l'exploiter. La première partie de cette thèse est ainsi consacrée à l'étude de cette problématique. La démarche de résolution proposée repose entièrement sur des algorithmes d'optimisation. Les deux contraintes citées précédemment étant traitées séparément, il devient aisé de prendre en compte davantage de critères dans le problème d'optimisation. Ainsi, de nouvelles formulations sont proposées. Ces dernières font appel aux techniques d'optimisation hiérarchique, afin de faciliter le traitement de la redondance, qui est exploitée pour l'évitement d'obstacles et les singularités du robot. Vu la complexité de ces formulations, nous avons préconisé une démarche de résolution approchée, qui fait appel aux métaheuristiques d'optimisation, en particulier les algorithmes génétiques. La validation de la démarche proposée est faite sur le modèle du robot Neuromate.

La réalisation de la procédure de craniotomie avec le robot Neuromate peut être divisée en deux parties : la première consiste à positionner le robot sur la tête du patient ce qui correspond à la première partie de ce travail (décrite plus haut) où on traite un cas général du problème de planification de trajectoire et la deuxième est la réalisation de l'ouverture, ce qui fait l'objet de la deuxième partie de cette thèse. L'objectif est de réaliser une petite ouverture au niveau du crâne humain afin que le chirurgien puisse glisser des instruments pour traiter des maladies affectant le cerveau. Réalisée par le chirurgien lui-même, sans assistance robotique, cette opération est très délicate, du fait du manque de précision et de l'allongement du temps d'intervention. Les risques d'aggravation sont particulièrement élevés si la zone d'intervention est proche des veines ou située dans des régions qui ont une fonction importante (région motrice ou région du langage, par exemple). La démarche classique de la craniotomie assistée fait appel à la co-manipulation, qui contraint le chirurgien à participer à l'action, et donc à fournir des efforts. Dans ce travail, une autre démarche est proposée, basée sur l'intégration au robot Neuromate d'un système d'usinage à grande vitesse. Des tests ont été réalisés sur des plaques en polyamide dont les caractéristiques mécaniques sont proches de celles du crâne.

**Mots clés :** *planification de trajectoires, évitement d'obstacles, lissage de courbes, optimisation à deux niveaux, métaheuristiques, algorithmes génétiques, craniotomie, robot Neuromate.*

# Abstract

The problem of trajectory planning is largely studied in the literature. In order to solve this problem, we need to take into account two important constraints, which are :

— the computation of the different configurations in which the robot must pass ;

— the smoothness of the resulting curves (velocities, accelerations, *jerks*).

Taking both constraints into consideration can be done in two different ways. The first one is to suppose that all the curves are smooth (using polynomial or trigonometric functions) and then, the aim of the resolution is to find the coefficient of each of them. The second way is to deal with the constraints separately. Thus, we compute in first the different configurations in which the robot must pass. After that, we compute the whole curve by interpolation. Adding to the constraints mentioned before, we have to solve the problem of the redundancy of the robot. The first part of this thesis is then devoted to the study of this problem. The proposed solving technique is entirely based on optimization algorithms. The two constraints cited above are treated separately, which allows to take more criteria into account. Thus, new formulations are proposed. They are based on the hierarchical optimization problem, which facilitates handling of the redundancy which is used for the obstacle and the singularities avoidance. Because of the high complexity of the proposed formulations, we chose to use metaheuristics to resolve them, especially the genetic algorithms. We validated the proposed technique on the model of the Neuromate robot.

The realization of the procedure of craniotomy with the Neuromate can be divided into two steps : the first one is to take the robot from any arbitrary position to the head of the patient (which corresponds to the first part of this work described before). The second step is to perform a very small hole which is the subject of this second part of this thesis. The procedure of craniotomy is used to perform a very small hole in the human skull in order to allow the surgeon to introduce medical instruments, to take care of some illness that affects brain. Achieved by the surgeon himself, without any robotics aid, this operation is very delicate, because of its lack of precision and increase of processing time. The risks are particularly high if the area of intervention is near the veins. The classical solving technique is based on the co-manipulation principle, which means that the surgeon participates in the action and then, provides an effort.

In this work, another solving technique is proposed. It is based on the integration of a machining system with the Neuromate robot. The tests are achieved on plates made of polyamide of which the mechanical characteristics are close to those of the human skull.

**Keywords :** *trajectory planning, obstacle avoidance, smoothing of curves, bilevel optimization, metaheuristic, genetic algorithm, craniotomy, Neuromate robot.*

# Table des matières


**Introduction générale** **1**

**1 Planification de trajectoire des bras manipulateurs** **6**

1.1 Introduction . . . 6
1.2 Aperçu sur les bras manipulateurs . . . 7
1.2.1 La redondance . . . 8
1.2.2 Les singularités . . . 10
1.3 Sur la planification de trajectoire . . . 12
1.3.1 Planification dans l'espace articulaire . . . 15
1.3.2 Planification dans l'espace cartésien . . . 15
1.4 Résolution du problème de planification . . . 16
1.5 Classification des méthodes de planification . . . 17
1.5.1 Classification basée environnement . . . 18
1.5.1.1 Méthodes globales . . . 18
1.5.1.2 Méthodes locales . . . 18
1.5.1.3 Méthodes hybrides . . . 18
1.5.2 Classification basée méthode de résolution . . . 19
1.5.2.1 Méthodes basées sur l'optimisation . . . 19
1.5.2.2 Méthodes basées sur la commande . . . 19
1.5.2.3 Méthodes basées sur la recherche dans un graphe . . . 19
1.6 Planification de trajectoire sous contraintes . . . 20
1.6.1 Le lissage de courbes . . . 20
1.6.2 L'évitement d'obstacles . . . 21
1.7 Conclusion . . . 22

**2 Les métaheuristiques appliquées à la robotique** **24**

2.1 Introduction . . . 24
2.2 Introduction à l'optimisation : aspects théoriques . . . 24
2.2.1 Formulation d'un problème d'optimisation . . . 24
2.2.2 Gestion des contraintes . . . 25
2.2.3 Optimisation difficile . . . 26
2.3 Méthodes de résolution approchée . . . 27
2.3.1 Heuristiques . . . 27
2.3.2 Métaheuristiques . . . 27
2.3.3 Métaheuristiques de voisinage . . . 28
2.3.3.1 Algorithme du recuit simulé . . . 28
2.3.3.2 Algorithme de recherche tabou . . . 29
2.3.3.3 Autres méthodes . . . 30

2.3.4 Métaheuristiques à population . . . 31
2.3.4.1 Algorithmes évolutionnaires . . . 31
2.3.4.2 Algorithme à évolution différentielle . . . 34
2.3.4.3 Les colonies de fourmis . . . 36
2.3.4.4 Optimisation par essaim particulaire . . . 36
2.3.4.5 Les colonies d'abeilles . . . 39
2.3.4.6 Autres méthodes . . . 41
2.4 Application des métaheuristiques en robotique . . . 41
2.5 Conclusion . . . 42

**3 Algorithmes de planification de trajectoires libres et en présence d'obstacles 44**
3.1 Introduction . . . 44
3.2 Évitement d'obstacles . . . 44
3.2.1 Modélisation des obstacles . . . 44
3.2.2 Technique d'évitement d'obstacles . . . 45
3.3 Problèmes d'optimisation à deux niveaux . . . 47
3.4 Formulation du problème de planification de trajectoires . . . 48
3.4.1 Fonctions objectifs . . . 49
3.4.2 Contraintes . . . 50
3.5 Résolution : algorithme bi-génétique . . . 51
3.6 Résultats de simulation sur le modèle du robot Neuromate . . . 53
3.7 Conclusion . . . 60

**4 Planification par interpolation via des métaheuristiques et des courbes polynomiales 61**
4.1 Introduction . . . 61
4.2 Description du problème traité . . . 61
4.3 Formulation proposée . . . 63
4.3.1 Critères . . . 63
4.3.2 Contraintes . . . 64
4.4 Résolution . . . 65
4.4.1 Le Lagrangien augmenté . . . 66
4.4.2 Opérateurs de l'algorithme génétique . . . 67
4.4.3 Codage de la solution . . . 68
4.5 Résultats de simulation . . . 69
4.6 Conclusion . . . 80

**5 Craniotomie robotisée avec le système Neuromate 81**
5.1 Introduction . . . 81
5.2 Anatomie et propriétés du crâne humain . . . 81
5.2.1 Anatomie du crâne . . . 81
5.2.2 Propriétés du crâne . . . 83
5.3 Aperçu de la procédure de craniotomie . . . 84
5.3.1 Craniotomie conventionnelle . . . 84
5.3.2 Craniotomie stéréotaxique . . . 85
5.3.3 Craniotomie robotisée . . . 86
5.4 Matériels utilisés . . . 87
5.4.1 Le système Neuromate . . . 87

5.4.2 Moteur et mèches . . . . . 88
5.4.3 Caméra numérique . . . . . 89
5.4.4 Plaques en polyamide . . . . . 89
5.5 Démarche de résolution . . . . . 89
5.6 Résultats . . . . . 92
5.6.1 Analyse des résultats obtenus . . . . . 94
5.6.2 Amélioration de la démarche de résolution . . . . . 98
5.6.3 Interface graphique . . . . . 99
5.7 Conclusion . . . . . 100

**Conclusion et perspectives** **102**

**6 Annexe** **105**
6.1 Le système Neuromate . . . . . 105
6.1.1 Modèle géométrique . . . . . 105
6.1.2 Modèle cinématique . . . . . 108
6.2 Modèle d'usinage . . . . . 109

**Références bibliographiques** **117**

# Introduction générale

Un des problèmes largement traités dans le domaine de la robotique est celui de la planification de trajectoire. En effet, les robots sont intégrés dans de nombreux domaines de l'ingénierie et leur utilisation est devenue indispensable. Ceci tient à leur grande précision et à leur maniabilité. Un des domaines les plus concernés par cette expansion est le domaine de la chirurgie. Dans ce cas d'application, on parle généralement de co-manipulation qui implique le chirurgien dans l'action, car il est en contact direct avec le robot. Une définition de base du problème de planification de trajectoires est de trouver l'ensemble des configurations ou positions par lesquelles le robot doit passer afin d'accomplir une tâche prédéfinie. Traiter ce problème dépend en premier lieu du robot utilisé. En effet, chaque type de robot possède sa propre géométrie, et donc son propre modèle cinématique et dynamique. Dans ce travail, nous nous sommes intéressés à la planification de trajectoires des bras manipulateurs, plus particulièrement ceux qui sont redondants. La première partie de cette thèse est consacrée à l'étude de cette problématique.

Dans le cas de bras manipulateurs, on distingue deux espaces de planifications : l'espace cartésien (appelé aussi espace de tâches) et l'espace articulaire. L'espace cartésien caractérise le mouvement de l'organe terminal du robot (l'effecteur) et l'espace articulaire est dédié au changement de valeur des différentes articulations du robot. Le choix d'un espace de travail dépend de la formulation du problème. Il est plus intéressant et surtout plus parlant pour l'utilisateur de travailler uniquement dans l'espace cartésien, dans lequel il est plus facile de décrire la tâche à accomplir et de suivre le déplacement de l'effecteur. Cependant, les différentes commandes du robot sont mieux représentées dans l'espace articulaire. La redondance est également traitée dans cet espace. Cette dernière exprime le fait qu'une position de l'effecteur peut être générée avec une infinité de configurations des variables articulaires. Donc, la première contrainte est le choix de l'espace de planification. Une deuxième contrainte est le lissage des courbes résultantes. En effet, pour des considérations physiques, et dans le but de préserver la structure mécanique du robot en lui évitant des mouvements brusques, il est impératif d'assurer des courbes lisses en termes de positions, vitesses, accélérations et *jerks*. D'autres contraintes s'ajoutant à ce problème sont l'évitement des différents obstacles, qui peuvent se trouver dans l'espace de travail du robot, ainsi que les singularités du robot.

Dans la littérature, énormément de techniques ont été développées dans ce contexte. On peut les diviser essentiellement en trois catégories :

— techniques se basant sur la synthèse de lois de commande ;
— techniques se basant sur la recherche en graphe ;
— techniques se basant sur des algorithmes d'optimisation.

Dans la première catégorie, le problème de planification est transformé en un problème de commande (boucle de commande). L'avantage avec cette technique est d'assurer la prise en compte du couple et le lissage simultané des courbes. Le défaut majeur de ce type de stratégie est son incapacité à prendre en compte toutes les contraintes. En effet, des simplifications sont prises en considération afin de faciliter la synthèse.

Dans la deuxième catégorie, on dispose de deux processus. Le premier vise à faire une capture de l'espace de travail du robot sous forme de graphe. Le deuxième consiste à trouver le meilleur chemin pour aller d'une configuration initiale à une configuration finale. Cette démarche peut être qualifiée d'abstraite parce qu'il y a une formulation très claire du problème. En outre, il serait difficile de prendre en compte toutes les contraintes.

Dans la troisième catégorie, tout le problème de planification est réduit à un problème d'optimisation. Sachant que toutes les contraintes du problème de planification peuvent être formalisées mathématiquement, cette démarche de résolution comporte un avantage majeur qui réside dans le fait que l'on peut prendre en compte autant de contraintes que l'on veut. La seule limite est l'obtention d'une formulation correcte, et aussi la capacité des algorithmes de résolution à trouver la meilleure solution. Avec cette démarche, on peut procéder de deux manières différentes. La première consiste à trouver l'ensemble des configurations (points) par lesquels le robot doit passer, puis à procéder à une interpolation pour traiter le problème du lissage. La deuxième solution est de donner au préalable des formes lisses aux différentes courbes, et le problème de la planification revient à calculer les paramètres caractérisant ces courbes. Procéder de la première manière permet de prendre en compte davantage de critères et de contraintes. Ayant opté pour ce choix, il reste à trouver la bonne formulation. Contrairement à la formulation classique de ce problème, on a préféré exploiter une formulation à base de problèmes d'optimisation à deux niveaux. Ceci présente l'avantage de prendre en compte plus de critères et aussi d'utiliser les deux espaces de planification en même temps.

Les problèmes d'optimisation à deux niveaux font partie des problèmes d'optimisation hiérarchique où des décisions sont prises de manière successive. Chaque niveau possède ses propres variables de décision (sa propre fonction objectif et ses contraintes) et il y a une corrélation (des variables communes) entre les niveaux. Ainsi, le premier niveau tente d'optimiser sa fonction en prenant en compte les décisions prises au deuxième niveau, le même scénario se reproduit avec le deuxième niveau. La figure 1 schématise la formulation utilisée. Les problèmes d'optimisation à deux niveaux appartiennent à la classe NP-complet, dont la résolution est réputée très difficile. C'est pourquoi nous avons eu recours aux métaheuristiques, qui sont des méthodes approchées permettant d'obtenir une solution avec un temps de convergence raisonnable. En raison de leur efficacité et de leur simplicité de programmation, nous avons choisi les algorithmes évolutionnaires, en particulier les algorithmes génétiques, que nous avons adaptés à notre problème. Ainsi, chaque niveau possède son propre algorithme génétique et les deux communiquent entre eux. Le résultat de cette démarche est l'ensemble des positions (configurations) par lesquelles le robot doit passer, afin d'atteindre la position finale tout en évitant les obstacles et les singularités. Ces positions sont un ensemble de points en une seule dimension. L'étape suivante consiste à procéder à une interpolation. Contrairement aux cas classiques d'interpolation où les données initiales sont représentées en deux dimensions, dans notre cas, elles sont sur une seule dimension. Une deuxième formulation a été proposée dans le but d'assurer le lissage des courbes de positions, vitesses, accélérations et jerks. Une résolution à base de métaheuristiques est aussi adoptée en raison de la complexité de la formulation et du nombre élevé de contraintes. Pour cette première partie, les tests de validation ont été réalisés sur le modèle du robot Neuromate.

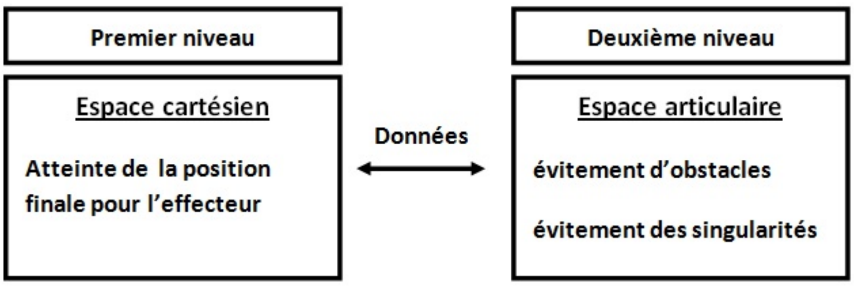



FIGURE 1 – Formulation du problème de planification.

La réalisation de la procédure de craniotomie avec le robot Neuromate peut être divisée en deux parties : la première consiste à positionner le robot sur la tête du patient ce qui correspond à la première partie de ce travail (décrite plus haut) où on traite un cas général du problème de planification de trajectoire et la deuxième est la réalisation de l'ouverture, ce qui fait l'objet de la deuxième partie de cette thèse. Cette application concerne l'assistance au geste chirurgical en craniotomie. L'opération consiste à réaliser une ouverture au niveau du crâne humain afin que le chirurgien puisse y glisser des instruments pour retirer une tumeur, arrêter une hémorragie ou traiter un abcès. Sans assistance robotique, cette intervention peut être délicate du fait du manque de précision et aussi de l'allongement du temps de l'intervention. Le robot Neuromate est un robot de guidage, conçu pour porter assistance au chirurgien en lui permettant d'avoir plus de précision au cours des interventions. Cependant, il n'a pas été conçu pour développer des efforts importants, alors qu'ils sont considérables dans une opération de craniotomie. Pour pallier ce problème, la démarche classique consiste à faire appel à la co-manipulation, qui implique le chirurgien dans l'action en intégrant ses efforts dans la boucle de commande. Pour pouvoir procéder de la sorte, il est nécessaire d'avoir accès au contrôleur du robot. Afin de contourner ce problème, nous proposons une démarche différente, basée sur l'intégration d'un système d'usinage à grande vitesse avec le robot Neuromate. Le modèle d'usinage considéré permet de quantifier les différents efforts développés. Ainsi, on peut aisément évaluer la faisabilité du robot. Plusieurs paramètres sont à vérifier lors de cette opération, tels le blocage de l'outil utilisé pour le forage, les coupons dégagés, etc. De ce fait, plusieurs vitesses de rotation de l'outil ont été testées pour trouver les bons paramètres. Un problème d'optimisation a été finalement proposé, afin de trouver les paramètres optimaux de l'usinage. Une interface graphique regroupant tous les paramètres, et qui facilitent les calculs, est aussi proposée. Les différents tests ont été réalisés sur des plaques en polyamide dont les caractéristiques mécaniques sont proches de celles du crâne.

Ce manuscrit est composé de cinq chapitres suivis d'une conclusion générale et des perspectives.

Dans le chapitre 1, on présente un état de l'art très général sur le problème de planification de trajectoire des bras manipulateurs redondants. On commence d'abord par caractériser un robot manipulateur comparativement aux autres catégories de robots. On cite les différentes techniques utilisées pour le traitement de la redondance. Puis, on définit le problème de planification, la démarche générale de résolution et une classification des techniques de résolution de ce genre de problème. Enfin, on développe quelques points importants qui sont pris en compte dans les problèmes de planification, à savoir le lissage des courbes et l'évitement d'obstacles.

Le chapitre 2 est dédié à une présentation globale des métaheuristiques. On commence par la définition des problèmes d'optimisation et leurs formulations. Ensuite, on enchaîne avec les stratégies les plus courantes dans la gestion des contraintes, et on montre le besoin de faire appel aux heuristiques et aux métaheuristiques pour résoudre des problèmes d'optimisation qualifiés de difficiles. On détaille ensuite le principe de base des métaheuristiques les plus utilisées dans la littérature en procédant à leur classification. A la fin du chapitre, on donne un aperçu sur l'utilisation des métaheuristiques pour la résolution des problèmes en robotique, en particulier les problèmes concernant la planification de trajectoires.

Dans le chapitre 3, une nouvelle technique de résolution du problème de planification de trajectoire à base de métaheuristiques est proposée. Elle a pour objectif de trouver l'ensemble des configurations par lesquelles le robot doit passer tout en évitant les singularités et les différents obstacles qui peuvent se trouver dans son espace de travail. De ce fait, une nouvelle stratégie d'évitement d'obstacles est aussi proposée. La formulation proposée repose sur l'optimisation à deux niveaux, ce qui permet de mieux gérer la redondance du manipulateur. Les résultats de simulations ont été réalisés sur le modèle du robot Neuromate.

Dans le chapitre 4, on s'intéresse à un aspect particulier du problème de planification, qui est l'obtention de courbes lisses. Ce chapitre complète le chapitre 3. En effet, on prend comme point de départ le résultat obtenu avec la technique présentée dans le chapitre 3. L'objectif est de compléter ce résultat en traitant le problème du lissage des courbes. Une formulation pour ce problème est proposée, sous la forme d'un problème d'optimisation sous contrainte et une métaheuristique à base d'algorithme génétique est utilisée pour la résolution.

Le chapitre 5 est consacré à une application réalisée avec le robot Neuromate. Elle consiste en l'opération de craniotomie. Rappelons que cette dernière a pour but de réaliser une petite ouverture au niveau du crâne humain. On commence d'abord par donner un aperçu des propriétés du crâne humain et des différentes techniques utilisées pour la réalisation de cette opération. Puis, on décrit le matériel mis à notre disposition. On présente ensuite la démarche de résolution et les résultats obtenus. A la fin du chapitre, on propose une formulation visant à améliorer la démarche de résolution précédemment décrite.

Le manuscrit se termine par une conclusion générale et deux annexes.

Chapitre Un

# Planification de trajectoire des bras manipulateurs

## 1.1 Introduction

Actuellement, les robots ont intégré énormément de domaines dans l'ingénierie (les chaînes de fabrication, le milieu médical, ...) et leur utilisation est devenue presque indispensable. Ceci est dû aux avancées majeures que connaît le domaine de la robotique et par conséquent, les robots d'aujourd'hui possèdent une grande capacité de réaliser des tâches très complexes en peu de temps et avec une grande précision, ce qui les rend très utiles. Le problème de planification de trajectoire est toujours associé à l'architecture et à la conception de ces robots. Ainsi, chaque robot dispose de son propre planificateur de trajectoire selon sa conception et selon différentes tâches qui lui seront attribuées. Le domaine de la médecine, plus particulièrement la chirurgie, est très touché par l'intégration des robots dans de nombreuses procédures. Dans notre étude, nous nous intéressons au robot **Neuromate**. Ce robot est parmi les premiers robots utilisés en chirurgie. On travaillera dans un premier temps sur le modèle de ce robot sur lequel on testera les algorithmes de planification qui seront développés et une application sur ce robot sera présentée dans le chapitre 5. Le **Neuromate** est un bras manipulateur redondant à cinq axes. De ce fait, nous présentons dans ce chapitre un aperçu sur cette catégorie de robots d'une manière générale ainsi que les différents concepts et techniques de planification de trajectoire les concernant.

Le terme planification peut prendre plusieurs formes selon le contexte, on peut trouver :

- **la planification de mouvement** : a pour but de calculer un chemin sans collision en prenant en compte la géométrie et la cinématique ;
- **la planification sensorielle** : vise à trouver les informations nécessaires pour la tâche en cours au bon moment, à sélectionner le bon capteur pour la réception de données et à définir une méthodologie de récupération ;
- **la planification pour la navigation** : son objectif est d'organiser un ensemble de primitives de localisation et de mouvements asservis sur des capteurs en vue d'atteindre

un but ou d'explorer un environnement ;

- **la planification pour la manipulation** : consiste à élaborer une stratégie avec des primitives de prise, de manipulation, de reconnaissance d'objets et d'assemblage en utilisant des retours sensoriels ;
- **la planification pour la communication** : vise à organiser les requêtes et le retour d'interaction (avec l'homme et en cas de collaboration ou de coordination multi-robots) pour l'activité en cours ;
- **la planification de tâches** : a pour objectif d'organiser l'ensemble des activités du robot dans le temps et leur attribuer des ressources, compte tenu des évolutions prévisibles dans l'environnement.

Dans notre cas, on s'intéresse plus particulièrement à la planification de mouvement, ce qui est appelé en général la planification de trajectoire. Le problème de planification de trajectoire peut être trouvé dans plusieurs domaines : la logistique, la robotique, les animations graphiques, la bio-informatique et aussi le déplacement des robots humanoïdes.

Ce chapitre est organisé comme suit : la section 1.2 est dédiée à une description générale des bras manipulateurs ainsi que leurs caractéristiques. Dans la section 1.3, nous présentons le problème de planification des bras manipulateurs et les concepts de base. La démarche générale de résolution d'un problème de planification est présentée dans la section 1.4. Une classification des techniques de planification est illustrée dans la section 1.5. Dans la section 1.6, les principales contraintes associées au problème de planification sont présentées. Nous concluons le chapitre dans la section 1.7.

## 1.2 Aperçu sur les bras manipulateurs

Dans cette section, nous allons procéder à quelques définitions de base dans le but de définir et caractériser un bras manipulateur relativement à d'autres catégories de robots.

Un solide est dit parfait si la distance entre deux quelconques de ses points est constante. On considère que tous les objets sont des solides parfaits (éliminant les objets souples et déformables). Un système mécanique est défini comme un ensemble de solides parfaits non libres, donc liés entre eux par des liaisons intérieures. Le système est alors soit libre, soit lié à un référentiel par des liaisons extérieures.

Le mouvement d'un solide peut être défini par la donnée à tout instant de sa position spatiale, et le mouvement d'un système mécanique par la donnée à tout instant du *N-uplet* de ses coordonnées généralisées.

On appelle coordonnées généralisées d'un système mécanique les $N$ paramètres indépendants $q_n$ qui déterminent complètement la configuration du système. Le nombre de degrés de liberté d'un système est défini comme le nombre $S$ de mouvements indépendants possibles.

Un système mécanique est dit holonome si les équations qui traduisent les liaisons intérieures entre ses composants ou éventuellement ses liaisons extérieures ne comportent pas les dérivées des points du système. Pour un système holonome, on a $S$=$N$ ; pour un système non holonome, $S$=$N$-$K$, où $K$ est le nombre de liaisons non holonomes.

Donc, on peut définir un robot comme un système mécanique général doté de certaines fonctionnalités particulières (déplacement autonome, etc). Avec ces définitions, on appelle généralement un robot manipulateur à $N$ degrés de liberté, un système mécanique à $N$ coordonnées généralisées composé de plusieurs solides en liaison holonome et lié à un référentiel par une liaison constante. Identiquement, on appelle un robot mobile, un système mécanique composé d'un solide assujetti à l'environnement par des liaisons extérieures généralement non holonomes. Dans les cas des manipulateurs, on distingue les manipulateurs redondants et non redondants.

Un robot manipulateur est donc constitué d'une structure articulée dont la base est fixe et dont l'extrémité terminale (effecteur du robot) est généralement munie d'un outil de préhension permettant de réaliser des tâches. Le problème général de planification de trajectoires revient donc à déterminer le mouvement d'un système mécanique à $N$ coordonnées généralisées. Sa complexité dépend du nombre et de la nature des liaisons extérieures ainsi que son environnement de travail.

### 1.2.1 La redondance

On dit qu'un manipulateur est redondant lorsque le nombre de degrés de liberté nécessaires pour spécifier la configuration du robot est strictement supérieur au nombre de degrés de liberté conférés à l'effecteur. La propriété de redondance est une propriété de l'architecture du robot (donc permanente en quelque sorte). Une autre manière de définir le concept de la redondance peut être faite en passant par la relation qui lie les vitesses articulaires (aux actionneurs) $\dot{\theta}$ à la vitesse de l'effecteur (de l'outil) $\dot{x}$. Pour un manipulateur en chaîne ouverte, on peut montrer que cette relation peut toujours se mettre sous la forme :

$$\dot{x} = J(\theta).\dot{\theta} \tag{1.2.1}$$

où $J$ est la matrice jacobienne du robot. Dans le cas où le manipulateur est redondant, cette matrice n'est pas carrée.

Comme nous allons le montrer dans la section 1.3, dans le cas des manipulateurs, la planification de trajectoires peut être faite soit dans l'espace articulaire ou bien dans l'espace cartésien. En terme de planification, la redondance des manipulateurs exprime le fait que pour une seule position de l'effecteur dans l'espace cartésien, une infinité de configurations dans l'espace articulaire lui correspondent. La majorité des bras manipulateurs industriels sont redondants ce qui rend cette propriété très intéressante. En effet, avec cette caractéristique, le robot peut réaliser des tâches très complexes et en plus il peut s'adapter à son environnement de travail. En général, la redondance est exploitée pour l'évitement d'obstacles et les singularités du robot. La redondance du robot est toujours associée au problème d'inversion du modèle dont le but est de trouver les vitesses articulaires à partir de la vitesse de l'effecteur. Plusieurs techniques ont été développées, les plus utilisées sont :

1. **Méthodes basées sur la jacobienne** : ce sont des méthodes analytiques qui se basent sur la minimisation d'une norme spécifique. Dans cette catégorie, on peut trouver :
   — L'utilisation de la pseudo inverse de la matrice jacobienne : la forme générale de la solution est $\dot{q} = J^{\#}.\dot{X}$ où $J^{\#}$ est la pseudo inverse de la matrice $J$ qu'on peut toujours calculer. La solution $\dot{q}$ minimise la norme $\frac{1}{2}.\dot{q}^t.\dot{q}$ ; Si la matrice $J$ est à plein rang, alors l'expression de $J^{\#}$ est :

   $$J^{\#} = J^t.(J.J^t)^{-1}$$

   Dans le cas contraire, $J^{\#}$ est calculée numériquement en utilisant la décomposition en valeurs singulières (SVD).
   — Pseudo inverse pondérée : même principe que la pseudo inverse en ajoutant une matrice de pondération. Dans ce cas, la solution $\dot{q}$ minimise la norme : $\frac{1}{2}.\dot{q}^t.W.\dot{q}$ avec $W$ est la matrice de pondération. Cette matrice peut être choisie proportionnellement aux intervalles de variation des variables articulaires.
2. **Méthodes basées sur l'exploitation de l'espace nul de la Jacobienne** : dans cette catégorie, des mouvements internes sont générés sans affecter le mouvement de l'effecteur. L'expression générale de la solution est la suivante :

$$\dot{q} = J^{\#}.\dot{X} + (I - J^{\#}.J).\dot{q_0}$$

Où $I$ est la matrice identité et $\dot{q_0}$ est une solution qu'il faut soigneusement choisir. Dans l'expression précédente, on a deux termes : le premier correspond à la solution présentée précédemment (la pseudo inverse) et le deuxième correspond à une projection dans l'espace nul de la matrice jacobienne (mouvements internes).

3. **Méthodes basées sur l'ajout de tâches supplémentaires** : dans ce cas, des tâches supplémentaires (contraintes) sont ajoutées à la matrice jacobienne dans le but de la rendre de rang complet et donc inversible. Ainsi, il sera possible de l'inverser. L'avantage avec cette stratégie est qu'on a une meilleure forme de la solution. Cependant des singularités peuvent se reproduire dans les calculs.

Récemment, dans certains travaux et afin de contourner le problème du calcul de modèle inverse, les chercheurs ont développé d'autres techniques de planification dans lesquelles la trajectoire est calculée dans l'espace articulaire puis testée dans l'espace cartésien. Ainsi, on n'aura pas besoin d'inverser la jacobienne et on n'utilisera que le modèle géométrique direct [Marcos et al., 2012].

### 1.2.2 Les singularités

Dans le cas des bras manipulateurs ou des robots parallèles, des singularités peuvent se produire. On peut alors définir le concept de configuration singulière comme suit. Un sous-ensemble de degrés de liberté du robot étant fixé par la définition d'une famille de configurations, le sous ensemble des degrés de liberté restants présente une redondance, c'est-à-dire que le nombre de degrés de liberté conférés à l'outil est inférieur au nombre de degrés de liberté restant libres.

En utilisant le modèle cinématique du robot (équation 1.3.1), les configurations singulières sont définies par :

$$det(J(\theta)) = 0 \tag{1.2.2}$$

où $det(J(\theta))$ représente le déterminant de la matrice jacobienne. Pour les configurations solutions de l'équation 1.2.2, on peut montrer que les lignes de la matrice $J$ sont linéairement dépendantes. Autrement dit, il existe des vitesses particulières de l'effecteur qui nécessitent des vitesses infinies au niveau articulaire. Cette situation constitue une perte de commandabilité du manipulateur puisque certaines manœuvres deviennent impossibles. Par conséquent, le comportement du système ne peut pas être maitrisé. En outre pour les vitesses qui sont accessibles, il existe un nombre infini de combinaisons des vitesses articulaires qui donnent lieu à la vitesse prescrite à l'effecteur. Dans un cas comme dans l'autre, on aboutit à une situation problématique. Les points dits de singularité (et leur voisinage) sont donc des points à éviter lors des manœuvres du robot.

Beaucoup de travaux se sont focalisés sur les études des singularités [Huberti, 2010; Chang & Khatib, 1995], la détermination de l'espace de singularité et par conséquent l'espace libre de ces configurations. Par conséquent, avoir connaissance de ces positions de singularité et les intégrer dans le processus de planification nous permet d'éviter tout comportement imprévu et indésirable pour le robot. A titre d'exemple, pour le robot **IR1400**, on distingue trois types de singularités :

1. La singularité bras tendu lorsque l'on veut atteindre la frontière de l'espace de travail en position.
2. La singularité de l'épaule (figure 1.1).
3. La singularité de poignet (figure 1.2).

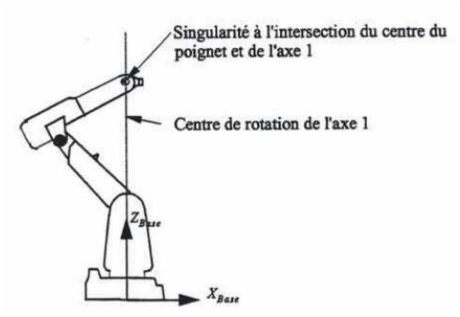



FIGURE 1.1 – Singularité de l'épaule.

La singularité d'épaule survient dans les configurations où le centre du poignet (intersection des axes 4, 5 et 6) se trouve directement au-dessus de l'axe 1 (voir figure 1.1).

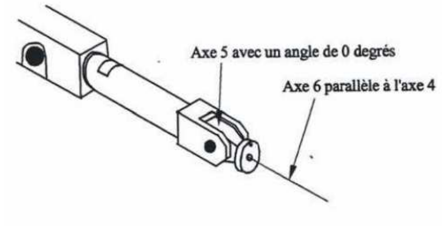



FIGURE 1.2 – Singularité de poignet.

Les singularités dites de poignet sont les configurations où l'axe 4 devient colinéaire avec l'axe 6, c'est-à-dire que l'angle de l'axe 5 est égal à 0 deg. (figure 1.2). La même étude a été faite sur le robot **PUMA** et a abouti aux même types de singularités [Chang & Khatib, 1995].

Lors d'un mouvement dans l'espace articulaire, le robot n'éprouve aucun problème à traverser des points singuliers. En effet, la génération de trajectoire étant définie sur la base de vitesses articulaires uniquement il n'y a pas de risque de solliciter des vitesses de l'effecteur qui pourraient nécessiter des vitesses infinies aux actionneurs.

Lorsque l'on veut générer un mouvement dans l'espace de travail (trajectoire linéaire ou circulaire), le robot qui traverse des points singuliers peut exciter des vitesses infinies aux actionneurs. Ce problème est évité par une procédure de sauvegarde. Le robot traverse le voisinage de points singuliers à vitesse réduite et l'exécution est contrôlée en permettant par exemple une légère erreur de l'orientation de l'effecteur à proximité des points singuliers. Une autre technique qui permet de s'éloigner des configurations singulières est de maximiser la manipulabilité du robot, cette technique sera présentée dans le chapitre 3.

## 1.3 Sur la planification de trajectoire

Afin de mieux cerner la problématique de planification de trajectoire, nous procédons dans cette section à la définition générale du problème de planification ainsi que son principe de base. La planification de trajectoire a pour but final de trouver une solution optimale sans collision et donc une trajectoire afin que le robot puisse accomplir une tâche particulière. Le problème général de la planification de trajectoires est très complexe à résoudre tel qu'il est. En effet, il est composé de plusieurs contraintes (nature du robot, environnement d'opération, ..). En conséquence, une modélisation souvent géométrique est nécessaire. La clé de sa résolution réside donc dans notre aptitude à raisonner sur ce type de modèles (représentations), à les manipuler et à les transformer afin finalement de trouver une version à partir de laquelle on puisse aisément faire apparaître une solution. Le problème de planification de trajectoires est fortement lié à l'environnement de travail du robot. En effet, en plus de la nature du robot, la complexité de ce problème augmente avec la complexité de son environnement de travail.

Dans le cadre du déplacement du robot dans un environnement connu, ses mouvements peuvent se résumer en une série de tâches du type **aller au but**. Cependant, la connaissance de l'environnement ne facilite pas pour autant au robot la connaissance du chemin à emprunter.

En conséquence, le problème de planification est souvent décomposé en plusieurs étapes comme on peut le voir sur la figure 1.3. Dans cette figure, la première étape consiste à planifier un chemin réalisant l'itinéraire désiré. Cette étape, dans la majorité des cas, consiste à prendre des décisions en fonction de la tâche à réaliser et de l'environnement de travail du robot.

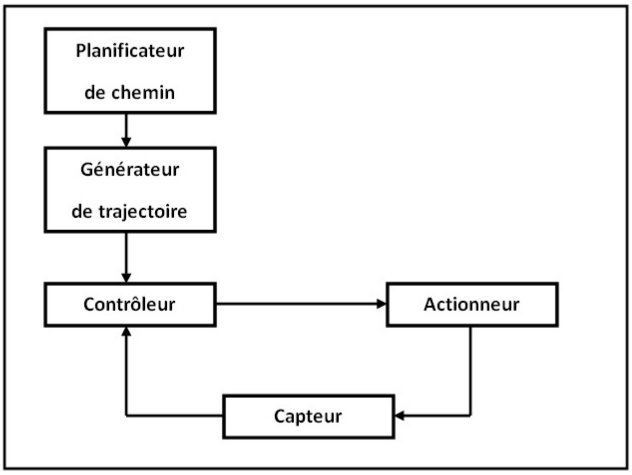



FIGURE 1.3 – Planification et génération de trajectoires.

La deuxième étape consiste à générer une trajectoire permettant de déplacer le robot le long de ce chemin. Les données de la planification sont exprimées sous la forme d'une trajectoire définie par une suite de repères (points de passage) correspondant aux situations successives de l'effecteur.

La dernière étape consiste à suivre cette trajectoire à partir de capteurs et d'actionneurs. Cette étape se résume souvent à une boucle d'asservissement permettant de réguler la trajectoire du robot par rapport à la trajectoire idéale.

Dans les problèmes de planification de trajectoires, on associe souvent le terme de fonction de tâche. Cette notion a été introduite par Claude Samson, Bernard Espiau et Patrick le Borgne en 1991 dans leur ouvrage *Robot Control* [Samson et al., 1991]. Le but d'une fonction de tâche est de traduire une tâche robotique donnée en une fonction mathématique. Cette mise en forme mathématique réalisée directement dans l'espace du capteur, permet alors d'élaborer une loi de commande et de la rendre exploitable pour la commande du robot. Le principe de ce formalisme consiste à réaliser une tâche robotique par la régulation (à zéro) d'une fonction d'erreur $e(q,t)$ de vecteur de configurations $q$ et de variable temporelle $t$ (figure 1.4).

Afin de mieux expliquer la notion de fonction de tâche, considérons un manipulateur et supposons que l'on veuille déplacer son effecteur (organe terminal) $P$ suivant une trajectoire

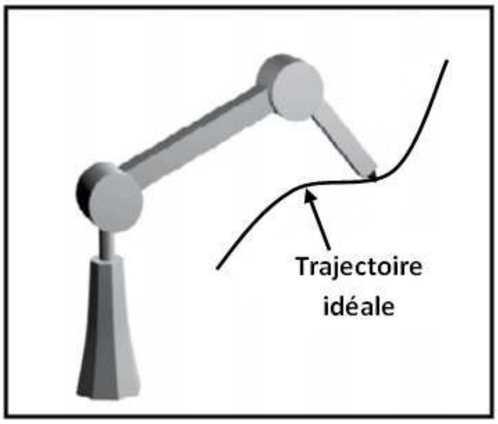



FIGURE 1.4 – Suivi d'une trajectoire.

donnée $e_d(t)$ (figure 1.4).

Si on décrit par $r$ la situation de l'effecteur $P$, alors une fonction de tâche correspondant à notre intention peut s'écrire :

$$e(q,t) = q - e_d(t) \tag{1.3.1}$$

La régulation à zéro de $e(q,t)$ permet bien de déplacer $P$ suivant la trajectoire désirée. Cependant, dans le cas des manipulateurs, on distingue deux espaces de travail : l'espace cartésien et l'espace articulaire. Donc, d'autres types de tâches robotiques décrites dans différents espaces peuvent être spécifiées suivant ce formalisme :

— dans l'espace articulaire, une fonction de tâche permettant une régulation en position peut s'écrire :

$$e(q,t) = q - q_d(t)$$

Où $q$ représente la position articulaire et $q_d(t)$ décrit la trajectoire idéale à suivre dans l'espace articulaire.

— dans l'espace cartésien, en suivant le même principe et pour un contrôle en situation, on obtient :

$$e(q,t) = r(q) - r_d(t)$$

Où $r(q)$ est un paramétrage donné de l'attitude de l'effecteur et $r_d(t)$ représente la trajectoire idéale exprimée dans l'espace cartésien.

Avec ce formalisme de fonction de tâche, un certain nombre de propriétés en découlent par exemple la notion d'admissibilité d'une tâche. Cette notion d'admissibilité consiste à vérifier si une tâche donnée est réalisable ou pas.

Comme mentionné précédemment, dans le cas des manipulateurs, on a deux espaces de travail : l'espace articulaire et l'espace cartésien. Travailler dans un espace ou dans un autre a des avantages et des limites. Ci-dessous, un aperçu sur la méthodologie de travail dans ces deux espaces :

### 1.3.1 Planification dans l'espace articulaire

L'espace articulaire est caractérisé par les variables articulaires. Le modèle cinématique et dynamique du robot est souvent exprimé dans cet espace, ce qui facilite les calculs. La génération et le suivi de trajectoire peuvent se faire directement dans cet espace. Cela se traduit par une séquence de positions articulaires (voire de vitesses ou d'accélérations) constituant les consignes des asservissements (figure 1.5).

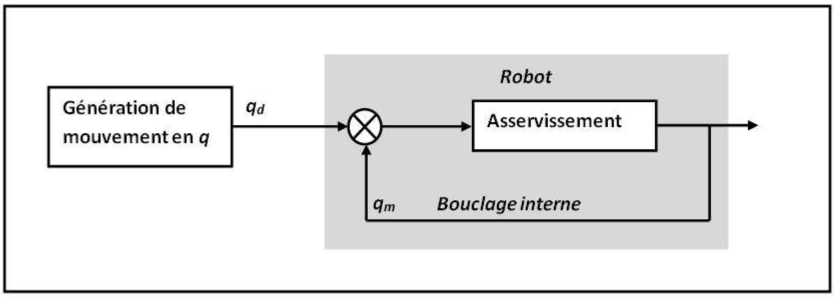



FIGURE 1.5 – Planification dans l'espace articulaire.

Les mouvements sont directement appliqués dans l'espace des actionneurs ce qui évite les problèmes de singularités dus aux calculs de modèles géométriques inverses. En contrepartie, la géométrie de déplacement du robot n'est pas contrôlée dans l'espace cartésien, ce qui peut engendrer des difficultés surtout en présence d'obstacles. Une autre limitation majeure liée à la planification dans cet espace est qu'il est très difficile de décrire une tâche robotique donnée directement dans l'espace articulaire.

### 1.3.2 Planification dans l'espace cartésien

La deuxième approche consiste à travailler dans l'espace cartésien où l'effecteur du robot évolue. Dans ce cas, il faut exprimer la trajectoire dans cet espace. Les coordonnées opérationnelles (cartésiennes) doivent alors être transformées en coordonnées articulaires par le *modèle géométrique inverse* du robot (MGI)(figure 1.6).

Il est donc facile d'exprimer une tâche robotique en fonction de la géométrie de déplacement du robot. Toutefois, dans certains cas de figure, le temps de calcul des modèles géométriques peut être très prohibitif. L'utilisation de modèle entraîne alors des biais qui peuvent conduire

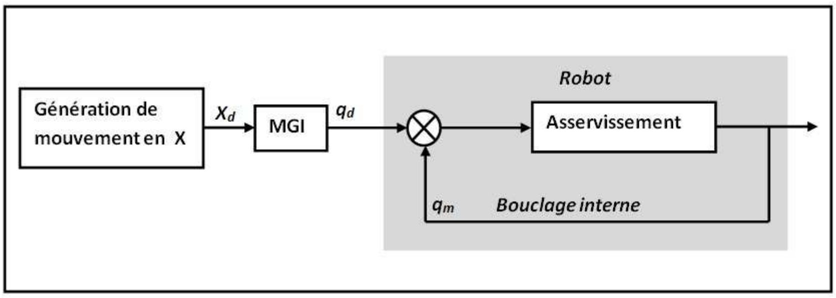



FIGURE 1.6 – Planification dans l'espace cartésien.

à des singularités de représentation.

Un autre espace de travail peut être défini si l'on dispose d'une autre source d'information. En effet, si on se place dans le cas où l'on a de l'information visuelle, on peut travailler dans un nouvel espace qui est plus parlant que les deux précédents. Cet espace est l'espace image dans lequel la trajectoire à suivre est traduite en un motif visuel (figure 1.7).

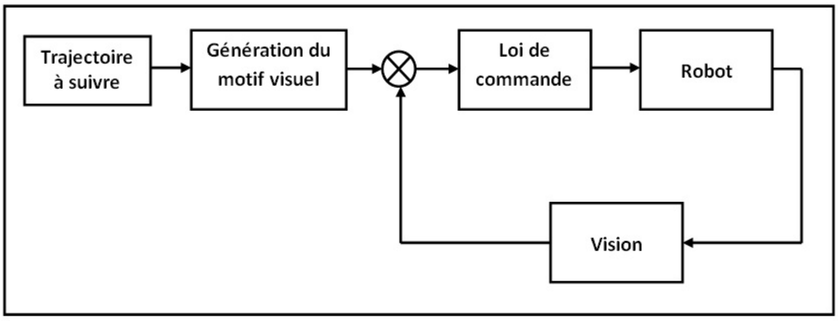



FIGURE 1.7 – Planification dans l'espace image.

## 1.4 Résolution du problème de planification

La démarche générale de la résolution d'un problème donné peut être résumée en quatre étapes fondamentales :

— saisir et comprendre les données de ce problème ;
— établir une représentation permettant la découverte d'une solution ;
— calculer une solution dans la nouvelle représentation ;
— rétablir la solution dans la représentation d'origine où l'on peut vérifier sa validité.

Dans la première étape, on essaye de recueillir toutes les informations disponibles liées au problème traité. Pour cette étape et dans notre cas, les données du problème se résument à la description de l'univers, c'est-à-dire le robot et son environnement, et la description de la tâche à effectuer. Cette dernière est décrite comme un ensemble de positions spatiales à atteindre. Par conséquent, les données de notre problème se décomposent en constantes (le modèle du robot et une partie de son environnement) et variables (positions du robot et des différents objets, configurations à atteindre).

La deuxième étape consiste à faire une représentation du travail. Cette étape est la clef du problème de planification de trajectoires : il s'agit de construire une représentation de l'univers pertinente relativement au but à atteindre, en l'occurrence la détermination de trajectoire. Dans notre cas, il paraît naturel de considérer cette représentation comme un ensemble de configurations du robot auxquelles sont associées des informations d'ordre statique et/ou dynamique. Cet ensemble ne peut être obtenu directement mais doit être dérivé de la représentation géométrique de l'univers qui constitue une donnée du problème.

La représentation élaborée doit en plus intégrer diverses contraintes de l'environnement non spécifiées explicitement dans les données initiales, comme les imprécisions du modèle fourni, les incertitudes sur la localisation des objets et les perturbations introduites par la dynamique du robot.

Le but de la troisième étape est de trouver une solution au problème. Une fois la représentation disponible, la recherche d'une trajectoire proprement dite fait appel à des techniques de planification de deux types : la recherche d'un parcours effectif dans la représentation courante (laquelle se ramène à la détermination d'une séquence de configurations de cette représentation satisfaisant les contraintes de la tâche) et la mise à jour locale de cette représentation.

La dernière étape consiste à transformer la trajectoire trouvée dans le nouvel espace en une trajectoire dans la représentation initiale. Ainsi, la trajectoire calculée doit être alors transformée en une séquence de positions spatiales respectant les contraintes cinématiques et dynamiques du robot. Les restrictions ainsi imposées par la nature des mouvements réalisables contraignent le choix final d'une trajectoire par l'éventail proposé.

## 1.5 Classification des méthodes de planification

Le problème de planification de trajectoire est largement traité dans la littérature et plusieurs classifications peuvent être établies [Yu & Muller, 1996; Pasquier, 1989]. Essentiellement, on peut avoir deux classifications : la première est relative à l'environnement de travail du robot c'est-à-dire les informations prises lors du calcul de la trajectoire. La deuxième classification concerne la technique utilisée pour la résolution du problème.

## 1.5.1 Classification basée environnement

Pour ce qui concerne la prise en compte des informations issues de l'espace de travail du robot, on peut soit prendre en compte tout l'espace de travail ce qu'on appelle les méthodes globales ou bien diviser l'espace de travail en plusieurs parties et travailler sur ces parties d'une manière successive, ce qui constitue les méthodes locales.

### 1.5.1.1 Méthodes globales

Dans le cas des méthodes globales, la technique de planification utilisée exploite toutes les informations provenant de l'espace de travail du robot lors du processus de planification. Ainsi, on a une connaissance complète de l'environnement de travail [Daachi et al., 2012]. L'avantage avec ce genre de techniques est qu'on ne laisse pas la place à l'imprévu vu qu'on a une connaissance totale sur l'espace de travail. L'inconvénient majeur est que ces méthodes sont très coûteuses en termes de temps de résolution à cause de la grande quantité d'informations utilisées pendant le calcul de trajectoire. Il sera encore plus difficile d'intégrer d'autres contraintes au problème comme l'ajout de nouveaux obstacles. Les méthodes qualifiées de globales trouvent leur intérêt dans un environnement statique pour des tâches répétitives.

### 1.5.1.2 Méthodes locales

Le principe des méthodes de planification dites locales est de décomposer l'espace de travail du robot en petites parties. En effet, dans ce genre de méthode, on ne considère pas toutes les informations relatives à l'espace de travail mais partie par partie. L'avantage avec une telle stratégie est que le temps de calcul est réduit considérablement. On laisse aussi place à l'imprévu. En effet, si au cours du déplacement du robot, des changements se produisent dans l'espace de travail (de nouveaux obstacles ), des informations supplémentaires vont être transmises au robot et donc prises en compte lors du calcul des déplacements suivants. Donc, on peut dire que les méthodes de planification qualifiées de locales possèdent une certaine adaptation vis à vis des changements qui peuvent se produire [Marcos et al., 2009].

### 1.5.1.3 Méthodes hybrides

Effectuer un bilan des avantages et des inconvénients des différentes méthodes existantes conduit fort logiquement à l'utilisation de méthodes hybrides, combinant des stratégies à la fois globales et locales, dans cet ordre [Pasquier, 1989].

### 1.5.2 Classification basée méthode de résolution

Une autre classification des méthodes de planification de trajectoires peut être faite selon la technique de résolution. En effet, selon la modélisation faite pour le problème de planification, une technique particulière est utilisée. Généralement, on peut les diviser en trois catégories : les méthodes basées sur l'optimisation, les méthodes basées sur la commande et les méthodes basées sur une recherche dans un graphe. Pour toutes ces catégories, on peut aussi faire la distinction de méthodes locales ou globales.

#### 1.5.2.1 Méthodes basées sur l'optimisation

Dans ce genre de méthodes, le problème de planification est réduit en un problème d'optimisation sous contraintes. Certains travaux proposent une formulation en mono-objectif [Garg & Kumar, 2002; Chettibi et al., 2004; Lin, 2004], d'autres en multi-objectifs [Marcos et al., 2012; E. J. Solteiro et al., 204]. Les critères les plus utilisés sont :

— la minimisation de l'erreur de position (nécessaire pour atteindre la configuration finale) ;
— la minimisation du temps de trajet ;
— la minimisation de l'énergie consommée ;
— la minimisation des vitesses, accélérations et *jerks*.

En termes de résolution, vu la complexité de ce genre de problème, on trouve généralement l'utilisation de métaheuristiques. Ainsi, on trouve l'utilisation de l'algorithme par essaim particulaire [Chakraborti et al., 2011], des algorithmes évolutionnaires [Marcos et al., 2009, 2012].

#### 1.5.2.2 Méthodes basées sur la commande

La deuxième catégorie de méthodes est celle qui se base sur la commande. On peut qualifier cette catégorie comme étant une catégorie classique de résolution. Dans ce cas, tout le problème de planification est formulé en un problème de contrôle/commande. Ainsi, pour la résolution, une commande particulière est utilisée. Beaucoup de travaux ont été développés dans ce sens [Pourazady & Ho, 1991; Le Boudec et al., 2006; Daachi et al., 2012; Madani et al., 2013]. On peut trouver un simple contrôle en vitesse [Pourazady & Ho, 1991], une commande adaptative est présentée dans [Le Boudec et al., 2006] et une autre qui se base sur les réseaux de neurones est développée dans [Daachi et al., 2012].

#### 1.5.2.3 Méthodes basées sur la recherche dans un graphe

La dernière catégorie de méthodes est celle qui repose sur la recherche dans un graphe. Dans ce genre de méthodes, on a deux étapes :

— la capture de la connectivité de l'espace de travail (construction d'un graphe) ;

— la recherche d'une solution dans le graphe construit.

La première étape a pour but de capturer la connectivité de l'espace de travail et donc construire un graphe qui sera utilisé par la suite pour trouver le chemin optimal. Beaucoup de techniques ont été développées :

1. Le graphe de visibilité (figure 1.8(b)) : dans cette technique, à partir de chaque sommet de chaque obstacle, on construit toutes les arêtes possibles. Les configurations initiale et finale sont aussi prises en compte par ce procédé. Ainsi, pour se déplacer de la configuration initiale vers la configuration finale, il suffit de trouver le chemin le plus court sur l'ensemble des arêtes construites.
2. Le diagramme de Voronoi (figure 1.8(a)) : dans le cas du diagramme de Voroni, le graphe est composé d'un ensemble d'arêtes et d'arcs. Chacun d'entre eux est construit de manière à respecter l'équidistance entre les obstacles. De ce fait, l'évitement d'obstacles est totalement traité. Pour se déplacer d'une configuration à une autre, il suffit de se mettre sur le graphe construit.
3. La décomposition cellulaire : le principe de la décomposition cellulaire est décrit dans la figure 1.8(c). A partir de l'espace libre, on procède à des décompositions successives en petites cellules qui sont numérotées. Pour la construction du graphe, on ne prend que les cellules qui ne coïncident pas avec les obstacles. Ainsi, le graphe résultant ne capture que l'espace libre. Après cela, on procède à une recherche dans le graphe afin de trouver le chemin optimal.
4. Les réseaux probabilistes et leurs variantes (figure 1.8(d)) : dans le cas des réseaux probabilistes, pour chaque nœud du graphe, on fait un tirage aléatoire. Si la position tirée se trouve dans l'espace libre, on l'ajoute au graphe, sinon on fait un autre tirage. Après cela, on fait une recherche et un lissage de la trajectoire finale.

Pour plus de détails sur ces différentes techniques, voir [Sahbani, 2003; O'Dúnlaing et al., 1987; Kuffner & LaValle, 2000; Sanchez & Latombe, 2002].

## 1.6 Planification de trajectoire sous contraintes

### 1.6.1 Le lissage de courbes

Un point important pris en compte lors de la résolution d'un problème de planification est la nature ou la forme des courbes résultantes. En effet, ce point joue un rôle crucial dans la solution développée car il touche directement à la structure mécanique du robot en question. Par conséquent, dans le but de préserver cette structure (le robot et ses actionneurs), on a intérêt à assurer des courbes lisses au niveau des positions, vitesses, accélérations. Si on prend en plus la dynamique du robot, on doit assurer ainsi une courbe lisse pour le couple généré. Pour la prise en compte du lissage des courbes dans la résolution du problème de planification, on peut procéder de deux manières différentes :

— la première façon consiste à trouver l'ensemble des points de passage du robot et puis à interpoler entre eux pour assurer le lissage ;
— la deuxième manière est de supposer au préalable que l'effecteur et les différentes articulations suivent des trajectoires particulières et par la suite, déterminer les coefficients les caractérisant.

Dans la littérature et afin d'assurer des courbes lisses, on trouve en général l'utilisation des splines [Gasparetto & Zanotto, 2010; Liu et al., 2013], des B-splines [Gasparetto & Zanotto, 2007], l'interpolation par morceaux [Tian & Collins, 2004] ou d'autres techniques qui se basent sur des fonctions trigonométriques.

### 1.6.2 L'évitement d'obstacles

Un autre point souvent associé au problème de planification de trajectoire est l'évitement d'obstacle. En effet, étant donné que la majorité des robots opèrent dans des environnements encombrés d'obstacles, il est nécessaire de développer des processus permettant ainsi de les éviter. En prenant en compte cet aspect lors de la planification, on a deux contraintes supplémentaires :

— la modélisation de ces obstacles ;
— la manière de les éviter.

Pour la première contrainte, on trouve souvent l'utilisation de surfaces lisses ou hypersurfaces (sphères, ...). Ainsi, on entoure l'obstacle initialement avec une forme particulière par l'hypersurface de rayon minimal. Cette procédure facilite la mesure de distance entre le robot et la nouvelle forme de l'obstacle. Pour la deuxième contrainte, la majorité des techniques développées se base sur une mesure de distance entre le robot et l'obstacle. Cependant, certaines méthodes comme dans la méthode du champ de potentiel [Khatib, 1985] utilisent un autre principe. En effet, le principe de cette méthode est de faire une sorte de distribution de charges dans l'espace du travail. Ainsi, un champ attractif est attribué à la position finale à atteindre et un champ répulsif est attribué à chaque obstacle ce qui permettra d'éviter les collisions.

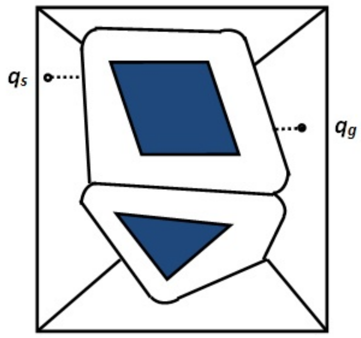
## 1.7 Conclusion

Dans ce chapitre, on a présenté un aperçu général sur les problèmes de planification de trajectoires pour les bras manipulateurs. On s'est limité à cette catégorie de robot parce que dans notre étude, on utilisera le modèle du robot **Neuromate** qui est un bras manipulateur redondant à cinq axes. On a commencé ce chapitre par l'introduction et la caractérisation de cette catégorie de robots relativement à d'autres robots. Ensuite, on a illustré les principales propriétés de ces robots, à savoir la redondance et l'étude des singularités, qui sont des points importants à prendre en compte lors du processus de planification.

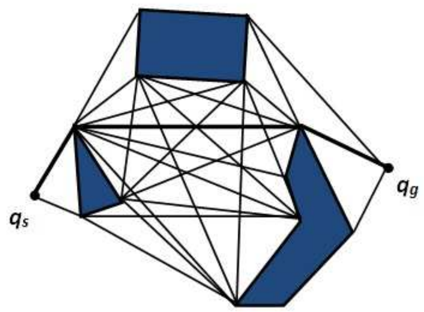
Après cela, on a abordé le problème de planification de trajectoires où on a donné une définition de base de ce problème avec une illustration basée sur le principe de la fonction de tâche. On a montré que dans le cas des manipulateurs, la planification peut se faire essentiellement dans deux espaces différents qui sont l'espace articulaire et l'espace cartésien. Une fois que le problème est défini, on a détaillé la démarche générale de résolution des problèmes de planification. On a aussi présenté une classification des différentes méthodes de planification. Dans ce cas, on peut avoir deux classifications : la première est relative à la prise en compte des informations issues de l'environnement de travail du robot et la deuxième concerne la technique utilisée dans la résolution du problème.

A la fin de ce chapitre, on a abordé les contraintes principales qui sont prises en compte dans les problèmes de planification de trajectoires. La première concerne le lissage de courbes résultantes ce qui a un impact direct sur la structure mécanique du robot et la deuxième concerne l'environnement de travail du robot. On a montré qu'il est impératif d'intégrer une technique d'évitement d'obstacles dans le processus de planification.

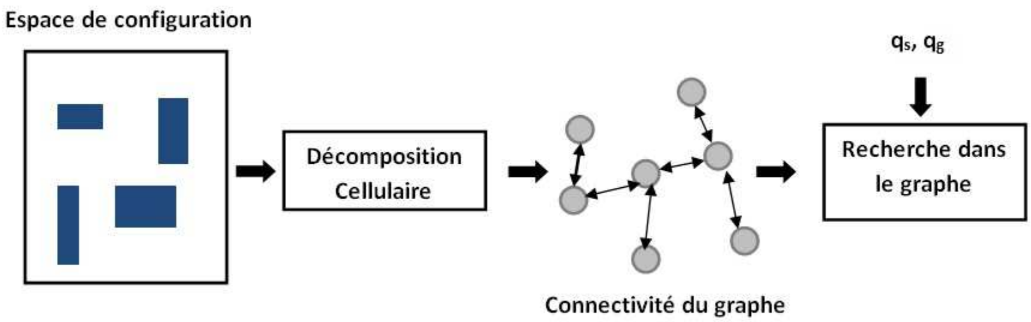
Dans la suite de la thèse, on va prendre en compte tous les points et contraintes de planifications cités dans ce chapitre. En matière de résolution, on a opté pour une technique de résolution locale basée sur l'utilisation des métaheuristiques qui seront présentées dans le chapitre qui suit.



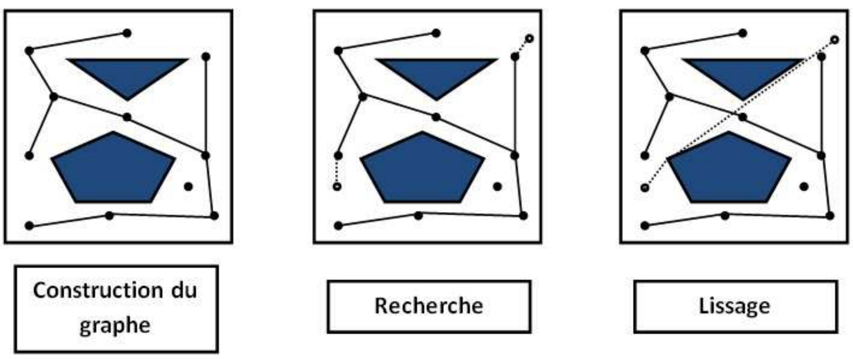

(a) Diagramme de Voronoi.




(b) Graphe de visibilité.




(c) Décompostion cellulaire.




(d) Réseaux probabilistes.

FIGURE 1.8 – Quelques techniques de construction de graphes.

Chapitre Deux

# Les métaheuristiques appliquées à la robotique

## 2.1 Introduction

Les problèmes d'optimisation et la recherche d'un état optimal touchent pratiquement tous les domaines de la vie de nos jours. En effet, dans de nombreux problèmes scientifiques, sociaux, économiques, on trouve des paramètres qui peuvent être ajustés afin de produire un résultat plus satisfaisant. Par conséquent, une formulation et une stratégie d'optimisation sont toujours associées à ces problèmes. La résolution des problèmes d'optimisation est un sujet central dans la recherche opérationnelle. Des techniques ont été conçues pour résoudre ces problèmes, notamment ceux qualifiés de *difficile*, en déterminant des solutions qui ne sont pas forcément optimales, mais qui s'en approchent. Ces méthodes, appelées heuristiques et métaheuristiques, s'inspirent généralement de phénomènes physiques, biologiques, socio-psychologiques, et peuvent faire appel au hasard.

Ce chapitre est structuré comme suit : dans la section 2.2, on présente les aspects théoriques de base de l'optimisation (formulation d'un problème d'optimisation et gestion des contraintes) et on décrit le cadre de l'optimisation difficile ; dans La section 2.3, on présente un état de l'art très global sur les méthodes de résolution approchée, on détaille le principe du fonctionnement des métaheuristiques les plus utilisées comme les algorithmes évolutionnaires, l'optimisation par essaim particulaire, les colonies de fourmis, l'algorithme du recuit simulé, etc ; dans la section 2.4, on donne un aperçu sur l'utilisation des métaheuristiques pour résoudre des problèmes liés à la robotique et plus particulièrement les problèmes de planification de trajectoires. Enfin, nous concluons ce chapitre dans la section 2.5.

## 2.2 Introduction à l'optimisation : aspects théoriques

### 2.2.1 Formulation d'un problème d'optimisation

D'une manière générale, un problème d'optimisation est défini par une fonction objectif $f$ qu'on cherche à minimiser ou à maximiser, un ensemble de variables appelées variables de décision ou de recherche et un ensemble de contraintes d'égalité (ou d'inégalité) que les variables doivent satisfaire. L'ensemble des solutions possibles du problème forme l'espace de recherche $E$ dont chaque dimension correspond à une variable. Un problème d'optimisation peut être statique ou dynamique (i.e. la fonction objectif change avec le temps), mono-objectif ou multi-objectif et avec ou sans contraintes.

Il existe plusieurs méthodes exactes (ou déterministes) permettant de résoudre certains types de problèmes d'optimisation et d'obtenir la solution optimale, en un temps raisonnable. Ces méthodes nécessitent que la fonction objectif présente un certain nombre de propriétés telles la convexité, la continuité ou la dérivabilité. On peut citer, parmi les méthodes connues, les méthodes de programmation linéaire [Schrijver, 1998], quadratique [Nocedal & Wright, 1998], la méthode du simplex [Nelder & Mead, 1965], la méthode de Newton [Nocedal & Wright, 1998] ou encore la méthode du gradient [Avriel, 1976].

### 2.2.2 Gestion des contraintes

Dans l'ingénierie, énormément de problèmes d'optimisation possèdent des contraintes, elles peuvent être des contraintes d'égalité ou d'inégalité, linéaires ou non. Une meilleure gestion de ces contraintes durant le processus d'optimisation améliore l'efficacité de l'algorithme d'optimisation en question. Plusieurs techniques (ou stratégies) ont été développées dans ce sens, on peut citer :

1. La stratégie de rejet : c'est une approche très simple, le principe est qu'au fil des itérations, on ne garde que les solutions qui satisfont les contraintes définies. Autrement dit, on ne garde que les solutions faisables. Ce type de stratégie est concevable si l'on a un nombre très réduit de contraintes. L'exploitation des informations issues des solutions infaisables peut s'avérer intéressant pour guider la recherche, ce qui montre une limitation dans cette stratégie.
2. La stratégie de pénalisation : dans cette stratégie, les solutions infaisables sont prises en considération durant la recherche en les affectant d'une pénalité. La fonction objectif à minimiser est donc composée de la fonction initiale (sans contraintes) et d'une fonction qui pénalise les solutions infaisables. C'est la stratégie la plus utilisée dans la littérature.
3. La stratégie de réparation : cette stratégie est une heuristique qui transforme les solutions infaisables en solutions faisables.
4. La stratégie de préservation : le principe de cette stratégie est qu'une représentation spécifique et des opérateurs vont assurer la génération de solutions faisables.

Il existe d'autres techniques comme la stratégie par décodage [Talbi, 2009] et le Lagrangien augmenté qui sera présenté en détail dans le chapitre 4.

### 2.2.3 Optimisation difficile

Les méthodes de résolution exacte ne sont pas applicables à toutes les problématiques. En effet, certains problèmes sont trop complexes à résoudre par ces méthodes, ceci est dû à l'existence de discontinuité, la non-dérivabilité, la présence de bruit ou encore la fonction objectif peut ne pas être définie précisément. Ajoutons à cela, le temps de calcul qui peut être excessif avec ces méthodes. Dans ces cas de figure, le problème d'optimisation est dit *difficile*, car aucune méthode exacte n'est capable de le résoudre en un temps raisonnable. Il est alors nécessaire d'avoir recours à des heuristiques de résolution dites méthodes *approchées*, qui fournissent un résultat sans garantir l'optimalité.

Les problèmes d'optimisation qualifiés de difficiles peuvent se diviser en deux types de problèmes : les problèmes à variables discrètes et les problèmes à variables continues :

— Un problème d'optimisation à variables discrètes consiste à trouver, dans un ensemble discret, une solution réalisable. Dans ce genre de problème, le but est de trouver la meilleure combinaison des variables de recherche. Le problème majeur réside dans le fait que le nombre de solutions réalisables (nombre de combinaisons possibles) est généralement très élevé, donc il est très difficile de trouver la meilleure d'entre elles en un temps raisonnable. Un exemple classique de ce type de problème est le problème du voyageur de commerce. L'utilisation d'algorithmes d'optimisation stochastique, tels que les métaheuristiques, permet de trouver une solution approchée en un temps raisonnable.

— Dans le deuxième type de problème, les variables de recherche sont continues. Ainsi, chaque variable prend sa valeur dans un intervalle qui peut être borné ou pas. Les problèmes d'identification paramétrique consistent à chercher à minimiser l'erreur entre le modèle d'un système et des observations expérimentales. Ce type de problèmes est moins formalisé que le précédent, mais un certain nombre de difficultés sont bien connues, comme l'existence de nombreuses variables présentant des corrélations non identifiées, la présence de bruit ou plus généralement une fonction objectif accessible par simulation uniquement. La grande majorité des métaheuristiques existantes étant à l'origine conçues pour résoudre des problèmes à variables discrètes [Dréo et al., 2003], les chercheurs ont été amenés à les adapter au cas continu. Il est à noter qu'il existe des problèmes à variables mixtes (le problème présente à la fois des variables discrètes et continues).

## 2.3 Méthodes de résolution approchée

### 2.3.1 Heuristiques

Étant donnée la complexité accrue de certains problèmes d'optimisation, l'utilisation des méthodes exactes n'est pas toujours possible. Dans certains cas, d'autres contraintes peuvent s'ajouter comme le temps de calcul souvent important, ou bien la difficulté, voire l'impossibilité dans certains cas, d'une définition séparée du problème. Afin de remédier à toutes ces contraintes, on utilise des méthodes approchées, appelées *heuristiques*. Ce terme dérive du grec ancien *heuriskêin* et qui signifie : trouver. Il englobe tout ce qui sert à la découverte et à l'exploitation. Il est à souligner qu'une heuristique peut être déterministe ou stochastique.

Une heuristique est un algorithme qui a l'aptitude de fournir rapidement (en un temps polynomial) une solution approchée et réalisable, mais pas nécessairement optimale, pour un problème d'optimisation difficile. Cette méthode approximative est à distinguer d'un algorithme exact qui donne une solution optimale pour un problème donné.

Il y a une multitude d'heuristiques qui ont été déjà proposées dans la littérature. On peut citer des heuristiques très simples comme les algorithmes gloutons [DeVore & Temlyakov, 1996] ou les approches par amélioration itérative [Basili & Turner, 1975]. Le principe des méthodes gloutonnes est de faire une succession de choix optimaux localement, jusqu'à ce que l'on ne puisse plus améliorer la solution, et ce, sans retour en arrière possible.

### 2.3.2 Métaheuristiques

Des heuristiques plus sophistiquées, adaptables à un grand nombre de problèmes différents et sans pour autant faire des changements majeurs dans les algorithmes, ont été développées et ont donné naissance à une nouvelle famille d'algorithmes d'optimisation stochastiques : les *métaheuristiques*. Ce terme a été inventé par Fred Glover en 1986, lors de la conception de la recherche tabou [Glover, 1986].

Ces métaheuristiques forment une famille d'algorithmes d'optimisation dans le but de résoudre des problèmes d'optimisation difficiles, pour lesquels les méthodes classiques ne sont pas efficaces. Elles sont utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, d'où le qualificatif *meta*. Elles ont la capacité d'optimiser un problème à partir d'un nombre minimal d'informations. Cependant, il y a aucune garantie que la solution trouvée soit optimale. D'un point de vue recherche opérationnelle, ce constat n'est pas forcément un désavantage, puisque l'on préfère toujours une approximation de l'optimum global trouvée rapidement à une valeur exacte trouvée dans un temps rédhibitoire.

Il existe un grand nombre de métaheuristiques différentes, allant d'une simple recherche locale à des algorithmes complexes de recherche globale. La majorité des métaheuristiques se basent sur l'utilisation de processus aléatoires et ainsi s'efforcent de récolter plus d'informations et de faire face à des problèmes comme l'explosion combinatoire. Les métaheuristiques peuvent être considérées comme des algorithmes stochastiques itératifs, qui manipulent une ou plusieurs solutions à la recherche de l'optimum. De ce fait, on peut diviser les métaheuristiques en deux grandes familles : celles qui manipulent une seule solution et celles qui manipulent un ensemble (population) de solutions. Un critère d'arrêt est utilisé afin de stopper l'algorithme comme un nombre maximal d'itérations ou bien une précision fixée au préalable. Ces méthodes tirent leur intérêt de leur capacité à éviter les optima locaux, soit en acceptant des dégradations de la fonction objectif au cours du traitement, soit en utilisant une population de points comme méthode de recherche.

Les métaheuristiques sont souvent inspirées de processus naturels qui relèvent de la physique (algorithme du recuit simulé), de la biologie de l'évolution (les algorithmes génétiques) ou encore de l'éthologie (les algorithmes de colonies de fourmis ou l'optimisation par essaim particulaire). Il y a de nombreuses méthodes existant dans la littérature, certaines, parmi les plus courantes, seront présentées ci-dessous.

### 2.3.3 Métaheuristiques de voisinage

#### 2.3.3.1 Algorithme du recuit simulé

Le recuit simulé est une méthode empirique inspirée d'un processus utilisé en métallurgie. Son principe est que pour atteindre les états de basse énergie d'un solide, on chauffe celui-ci jusqu'à des températures élevées, après cela, on le laisse refroidir lentement.

Cet algorithme a été proposé par [Kirkpatrick et al., 1983], sa description classique le présente comme un algorithme probabiliste, où un point (une seule solution) évolue dans l'espace de recherche. L'algorithme du recuit simulé s'appuie sur l'algorithme de Metropolis [Metropolis et al., 1953], qui permet de décrire l'évolution d'un système en thermodynamique. Cette procédure permet de sortir des minima locaux à hautes températures et de conserver les états les plus probables à basses températures. L'algorithme 1 résume le principe du recuit simulé.

Les inconvénients de l'algorithme du recuit simulé résident en particulier dans :

**Algorithme 1** Algorithme du recuit simulé

**Déterminer** une configuration aléatoire $S$
**Choix** des mécanismes de perturbation d'une configuration
**Initialiser** la température $T$
**tant que** le critère d'arrêt n'est pas satisfait **faire**
    **tant que** l'équilibre n'est pas atteint **faire**
        **Tirer** une nouvelle configuration $S'$
        **Appliquer** la règle de Metropolis
        **si** $f(S') < f(S)$ **alors**
            $S_{min} = S'$
            $f_{min} = f(S')$
        **fin si**
    **fin tant que**
    **Décroître** la température
**fin tant que**

1. La difficulté de réglage de la température initiale (éviter de partir d'une température trop haute, comme d'une température trop basse).
2. La difficulté de trouver une loi adéquate de décroissance de la température au fil des itérations.
3. Le temps de convergence qui peut devenir très long quand la température initiale est trop haute (on part de très loin) ou que le processus de décroissance de la température est très lent.

Cependant, des études ont été faites pour remédier à ces inconvénients [Courat et al., 1993]. La méthode du recuit simulé a l'avantage d'être souple vis-à-vis des évolutions du problème et facile à implémenter.

#### 2.3.3.2 Algorithme de recherche tabou

L'algorithme de recherche tabou est une métaheuristique itérative de recherche locale introduite par Fred Glover [Glover, 1986]. L'idée de recherche tabou est de partir d'une position donnée et d'explorer le voisinage en choisissant le voisin qui minimise la fonction objectif. En rencontrant un minimum local (pas de diminution de la fonction objectif dans le voisinage), on décide de poursuivre la recherche des solutions en acceptant des déplacements qui n'améliorent pas la solution courante, et ce, en utilisant un principe de mémoire qui évite les retours en arrière (mouvements cycliques). Ce mécanisme d'augmentation de la fonction objectif, par acceptation de solutions dégradantes, permet de sortir des minima locaux. Comme l'algorithme du recuit simulé, la recherche tabou fonctionne avec une seule solution courante, qui est actualisée au cours des itérations successives. La différence avec cet algorithme réside dans la tenue et la mise à jour d'une liste de mouvements interdits (ou de solutions interdites), appelée liste tabou, afin de réduire le risque de retour à une configuration déjà visitée. Cette liste tabou évolue au fil des itérations, qui alternent des phases d'exploration (appelée aussi diversification) avec des phases d'exploitation (appelée aussi intensification). L'algorithme 2 représente le principe de la recherche tabou.

**Algorithme 2** Algorithme de recherche tabou

**Déterminer** une configuration aléatoire $s$
**Initialiser** une liste tabou vide
**tant que** le critère d'arrêt n'est pas satisfait **faire**
    **Perturbation** de $s$ suivant $N$ mouvements non tabous
    **Évaluation** des $N$ voisins
    **Sélection** du meilleur voisin $t$
    **Actualisation** de la meilleure position connue $s^*$
    **Insertion** du mouvement $t \rightarrow s$ dans la liste tabou
    s=t
**fin tant que**

Dans sa forme de base, l'algorithme de recherche tabou présente l'avantage de comporter moins de paramètres que l'algorithme du recuit simulé. Cependant, l'algorithme n'étant pas toujours performant, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle [Glover & Laguna, 1997].

#### 2.3.3.3 Autres méthodes

D'autres méthodes qui manipulent une seule solution existent dans la littérature, on peut citer :

1. **La méthode de lissage** : le principe de cette méthode est de transformer l'espace de recherche initial en une surface lisse à plusieurs reprises, ceci a l'avantage d'éliminer les optimums locaux et de se rapprocher au fur et à mesure de l'optimum global.
2. **La méthode du bruit** : l'idée de cette technique est de rajouter du bruit à la fonction objectif. Ceci permet d'avoir de la diversification dans la recherche et ainsi d'explorer l'espace de recherche. La norme du bruit ajouté est réduite au cours des itérations.
3. **Le GRASP** : la technique du GRASP a été introduite en 1989, elle se base sur l'algorithme glouton pour résoudre des problèmes d'optimisation combinatoires. Dans cette méthode, on a deux étapes principales : la première consiste à créer une solution aléatoire avec l'algorithme glouton. La deuxième étape consiste à effectuer une recherche locale à partir de la solution déjà construite.

Pour plus de détails sur ces différentes techniques, se référer à [Talbi, 2009].

### 2.3.4 Métaheuristiques à population

#### 2.3.4.1 Algorithmes évolutionnaires

Les algorithmes évolutionnaires (AEs) sont des techniques de recherche inspirées par l'évolution biologique des espèces. Ils ont été élaborés au cours des années 1950 [Fraser, 1957]. Ces algorithmes s'inspirent de l'évolution des êtres vivants pour résoudre des problèmes d'optimisation. Leur principale idée est que les individus qui ont hérité des caractères bien adaptés à leur milieu ont tendance à vivre assez longtemps pour se reproduire, tandis que les plus faibles ont tendance à disparaître.

Au cours des années 1970, avec l'apparition des calculateurs de forte puissance, de nombreuses approches de modélisation de l'évolution ont été réalisées, on peut citer :

- **Les stratégies d'évolution** [Beyer, 2001] qui ont été conçues pour résoudre des problèmes d'optimisation à variables continues. Elles sont basées sur la modélisation des paramètres stratégiques qui contrôlent la variation dans l'évolution.
- **La programmation évolutionnaire** [Fogel et al., 1966], qui vise à faire évoluer des structures d'automates à états finis par des successions de croisements et de mutations.
- **Les algorithmes génétiques** [Holland, 1975], qui ont été conçus pour résoudre des problèmes d'optimisation à variables discrètes par le biais d'une modélisation génétique.
- **La programmation génétique** [Koza, 1989], basée sur l'algorithme génétique, mais où les individus sont des programmes informatiques, représentés en utilisant une structure d'arbre.
- **L'évolution différentielle** [Storn & Price, 1997], qui a été conçue dans le but de résoudre des problèmes d'optimisation à variables continues. Sa stratégie consiste à biaiser un opérateur de mutation, appliqué à un individu, en fonction des différences calculées avec d'autres individus sélectionnés aléatoirement.

Les approches évolutionnaires s'appuient sur un modèle commun présenté dans l'algorithme 3. Les individus soumis à l'évolution sont des solutions possibles au problème posé. L'ensemble de ces individus constitue une population (d'où l'appartenance aux métaheuristiques manipulant un ensemble de solutions). Cette population évolue au cours des itérations successives, appelées générations. A chaque génération, des opérateurs sont appliqués aux individus, afin de créer la population de la génération suivante. Chacun de ces opérateurs utilise un ou plusieurs individus, appelés parents, pour créer de nouveaux individus, appelés enfants. A la fin de chaque génération, une sélection d'enfants créés durant la génération remplace un sous-ensemble d'individus de la population.

**Algorithme 3** Algorithme évolutionnaire générique

**Initialisation** de la population de $\mu$ individus
**Évaluation** des $\mu$ individus
**tant que** le critère d'arrêt n'est pas satisfait **faire**
  **Sélection** de $\rho$ individus en vue de la phase de reproduction
  **Croisement** des $\rho$ individus sélectionnés
  **Mutation** des $\lambda$ enfants obtenus
  **Évaluation** des $\lambda$ enfants obtenus
  **Sélection** pour le remplacement
**fin tant que**

Un algorithme évolutionnaire possède trois opérateurs principaux :

1. **Opérateur de sélection**, son rôle est de favoriser la propagation des meilleures solutions dans la population, tout en maintenant une certaine diversité génétique au sein de celle-ci. Ainsi, cet opérateur permet de choisir les parents qui vont participer à la création des enfants, plusieurs techniques peuvent être trouvées, on peut citer :
   — *sélection par roulette* : son principe est d'assigner à chaque individu une probabilité proportionnelle à sa *fitness* vis-à-vis de la fonction objectif. Après cela, on fait tourner la roulette et des parents sont sélectionnés. Ainsi, les parents possédant une meilleure *fitness* ont plus de chances d'être sélectionnés.
   — *sélection par tournoi* : avec cette technique, un ensemble de parents sont sélectionnés aléatoirement (le nombre de parents sélectionnés correspond à la taille du tournoi). Après cela, le meilleur individu est sélectionné.
2. **Opérateur de croisement**, mis en œuvre lors de la création des enfants. Son but est d'échanger les gènes des parents sélectionnés pour créer des enfants. Cet opérateur permet d'intensifier la recherche dans l'espace de recherche, on lui associe aussi une *probabilité de croisement* qui représente le taux de la population qui va être touché par cet opérateur. Plusieurs techniques de croisement ont été développées, on peut citer :
   — *croisement en un seul point* : l'idée est de générer aléatoirement une position de coupure pour deux parents et après cela, on échange les informations afin de créer les enfants. On peut généraliser cette technique pour avoir du croisement en deux points (illustré dans la figure 2.1) ou multi-points [Talbi, 2009].
   — *croisement intermédiaire* : pour cette technique, chaque *gène* de l'enfant créé est calculé par une somme pondérée de deux *gènes* des deux parents sélectionnés.

3. **Opérateur de mutation**, contrairement à l'opérateur de croisement, cet opérateur agit sur chaque parent individuellement. Son principe est de tirer aléatoirement une composante de l'individu parent et de la remplacer par une valeur aléatoire. Le fait d'ajouter de l'information aléatoire dans la création des enfants permet de maintenir de la diversification dans la recherche. Plusieurs techniques de mutation peuvent être utilisées, on peut citer :
   — *mutation gaussienne* : dans ce cas, la valeur ajoutée au *gène* sélectionné est issue d'une loi de distribution gaussienne. En représentation binaire, il suffit d'inverser la valeur du *gène* sélectionné (figure 2.2).
   — *mutation polynomiale* : pour cette technique, une distribution polynomiale est utilisée pour la création des enfants.

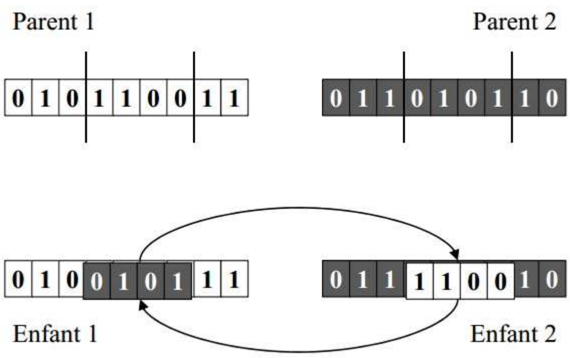



FIGURE 2.1 – Exemple d'opérateur de croisement en représentation binaire.

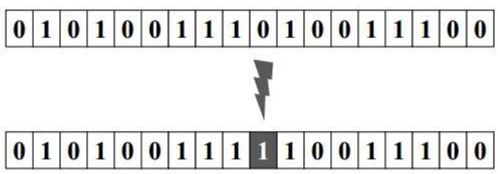



FIGURE 2.2 – Exemple d'opérateur de mutation en représentation binaire.

Le principe d'un algorithme évolutionnaire est illustré dans la figure 2.3.

En plus des trois opérateurs précédemment cités, un quatrième est utilisé dans le but de maintenir constante la taille de la population au cours des générations, ce qui est appelé *stratégie de remplacement*. Plusieurs techniques existent [Talbi, 2009], comme remplacer tous les parents par les enfants créés ou bien garder les meilleurs parents et remplacer les mauvais par les enfants créés.

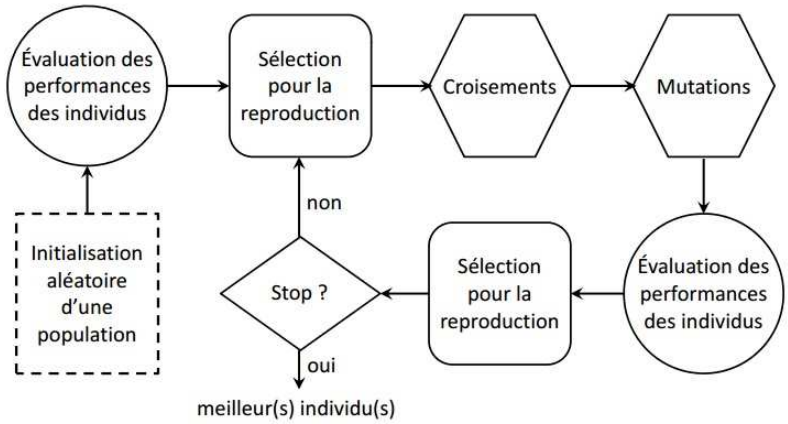



FIGURE 2.3 – Principe d'un algorithme évolutionnaire.

#### 2.3.4.2 Algorithme à évolution différentielle

L'algorithme à évolution différentielle est une métaheuristique stochastique d'optimisation, il a été inspiré par les algorithmes génétiques et des stratégies évolutionnaires combinées avec une technique géométrique de recherche. Dans le cas des algorithmes génétiques, la structure des individus change au cours des générations via les opérateurs de croisement et de mutation, alors que les stratégies évolutionnaires réalisent l'auto-adaptation par une manipulation géométrique des individus. Ces idées ont été mises en œuvre grâce à une opération de mutation de vecteurs, proposée en 1995 par K. Price et R. Storn [El. Dor, 2012]. A l'origine, la méthode de l'évolution différentielle a été conçue pour les problèmes d'optimisation continus et sans contraintes, ses extensions actuelles permettent de traiter les problèmes à variables mixtes et gèrent les contraintes non linéaires.

Dans la méthode à évolution différentielle, la population initiale est générée par un tirage aléatoire uniforme sur l'ensemble des valeurs possibles de chaque variable. les bornes inférieure et supérieure de chaque variable sont spécifiées par l'utilisateur. Suite à l'initialisation, l'algorithme effectue une série de transformations sur les individus, dans un processus appelé évolution.

La population contient $N$ individus. Chaque individu $x_{i,G}$ est un vecteur de dimension $D$, où $G$ désigne la génération.

$$x_{i,G} = (x_{1i,G}, x_{2i,G}, x_{3i,G},...,x_{Di,G}) \qquad avec \quad i = 1,2,3,...,N$$

La version standard de cet algorithme utilise trois opérateurs (mutation, croisement et sélection) comme les algorithmes génétiques. A chaque génération, l'algorithme applique successivement ces trois opérations sur chaque vecteur pour produire un vecteur d'essai :

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, u_{3i,G+1}, ..., u_{Di,G+1}) \quad avec \quad i = 1,2,3,...,N$$

Une opération de sélection permet de choisir les individus à conserver pour la nouvelle génération $G+1$.

**La mutation** : pour l'opérateur de mutation, pour chaque vecteur $x_{i,G}$, on génère un vecteur mutant $v_{i,G+1}$ qui peut être créé avec l'une des stratégies de mutation suivantes :

— **Rand** :

$$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G})$$

— **Best** :

$$v_{i,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G})$$

— **Current to best** :

$$v_{i,G+1} = x_{i,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{best,G} - x_{i,G})$$

Le indices $r1, r2, r3$ sont des entiers aléatoires et tous différents. $x_{best,G}$ est le meilleur individu à la génération $G$. $F \in [0,2]$ est une valeur constante qui contrôle l'amplification de la variation différentielle. $(x_{r_i,G} - x_{r_j,G})$.

**Le croisement** : après l'opération de mutation, l'opération de croisement binaire produit le vecteur d'essai final $u_{i,G+1}$ selon le vecteur $x_{i,G}$ et le vecteur $v_{i,G+1}$. Le nouveau vecteur $u_{i,G+1}$ est donné par la formule suivante :

$$u_{ji,G+1} = \begin{cases} v_{1i,G+1} \;\; si \;\; (rand(j) \leq CR) \quad ou \;\; j = rnbr(i) \\ x_{ji,G} \;\; si \quad (rand(j) > CR) \quad et \quad j \neq rnbr(i) \end{cases}$$
$$pour \;\; tout \;\; j \in \{1,2,3,...,D\}$$

où $rand(j)$ est la $j^{me}$ valeur fournie par un générateur de nombres aléatoires dans l'intervalle $[0,1]$. $CR$ est le coefficient de croisement qui appartient à l'intervalle $[0,1]$. $rnbr(i)$ est un indice choisi au hasard dans l'ensemble $\{1,2,...,N\}$.

**La sélection** : afin de décider quel vecteur, parmi $u_{i,G+1}$ ou $x_{i,G}$, doit être choisi pour la génération $G+1$, on doit comparer leur *fitness* par rapport à la fonction objectif. De cette façon, on garde le vecteur ayant une petite valeur (si on a un problème de minimisation). Ainsi, le vecteur $x_{i,G+1}$ est choisi comme suit :

$$x_{i,G+1} = \begin{cases} u_{i,G+1} \;\; si \;\; f(u_{i,G+1}) < f(x_{i,G}) \\ \hfill x_{i,G} \;\; sinon \end{cases}$$

Avec un bon réglage des différents paramètres de l'algorithme (taille de la population, facteur de mutation, etc), on augmente d'une manière significative son efficacité. L'auto-adaptation de ces paramètres paraît donc intéressante pour l'amélioration de l'algorithme [El. Dor, 2012].

#### 2.3.4.3 Les colonies de fourmis

Les algorithmes de colonies de fourmis sont inspirés du comportement des fourmis. Ils constituent une famille de métaheuristiques à base de population pour résoudre des problèmes complexes. Ceci est possible car les fourmis ont la capacité de communiquer entre elles indirectement, par le dépôt dans l'environnement de substances chimiques, appelées *phéromones*. Ce type de communication indirecte est appelé *stigmergie*. En anglais, le terme consacré à la principale classe d'algorithmes est *Ant Colony Optimization* (ACO).

La principale illustration de ce phénomène est donnée par la figure 2.4 où un obstacle est placé sur le trajet des fourmis. Après une étape d'exploration, les fourmis finissent par trouver le plus court chemin entre le nid et la source de nourriture [Goss et al., 1989]. Les fourmis qui sont retournées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont emprunté le chemin le plus court. Ainsi, la quantité de phéromones déposées par unité de temps sur ce trajet est plus importante que sur les autres. Par ailleurs, une fourmi est d'autant plus attirée à un certain endroit que le taux de phéromones y est important. Par conséquent, le plus court chemin a une probabilité plus importante d'être emprunté par les fourmis que les autres chemins et sera donc, à terme, emprunté par toutes les fourmis.

Le premier algorithme d'optimisation s'inspirant de cette analogie a été proposé par Colorni, Dorigo et Maniezzo [Dorigo et al., 1996] pour résoudre le problème du voyageur de commerce. L'algorithme 4 illustre l'approche par colonies de fourmis proposée par les auteurs. Pour des détails sur ce problème, voir [El. Dor, 2012].

#### 2.3.4.4 Optimisation par essaim particulaire

L'optimisation par essaim particulaire (OEP), ou en anglais *Particle Swarm Optimization* (PSO), est un algorithme qui utilise une population de solutions candidates dans le but de

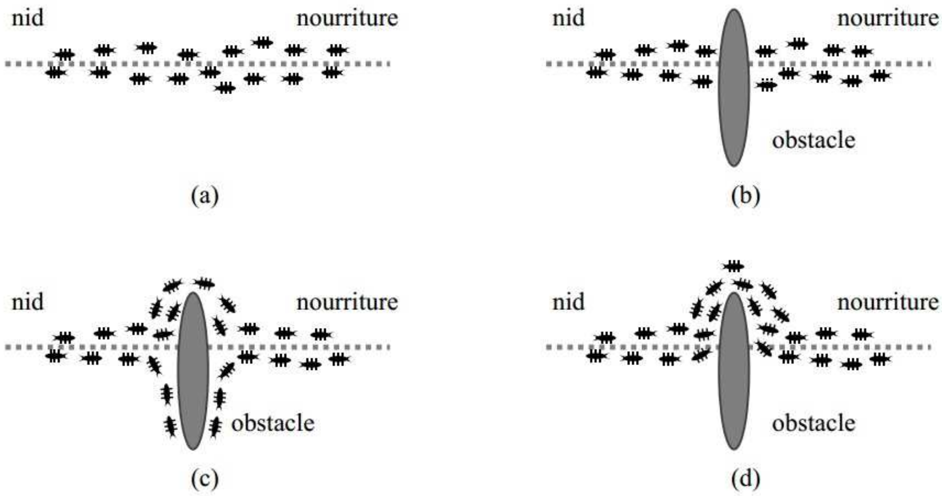



FIGURE 2.4 – Illustration de la capacité des fourmis à chercher de la nourriture en minimisant leur parcours. (a) Recherche sans obstacle, (b) Apparition d'un obstacle, (c) Recherche du chemin optimal, (d) Chemin optimal trouvé.

développer une solution optimale au problème. Il a été proposé par Russel Eberhart et James Kennedy en 1995 [Kennedy & Eberhart, 1995]. A l'origine, il s'inspire du monde du vivant, plus précisément du comportement social des animaux évoluant au sein d'un essaim, tels que les bancs de poissons et les vols groupés d'oiseaux. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une intelligence limitée, et ne dispose que d'une connaissance locale de sa situation dans l'essaim.

L'information locale et la mémoire de chaque individu sont utilisées pour décider de son déplacement. Des règles simples, telles que *rester proche des autres individus*, *aller dans la même direction* ou *garder la même vitesse*, sont suffisantes pour maintenir la cohésion de l'essaim, et ainsi permettre la mise en œuvre de comportements collectifs complexes et adaptatifs.

L'essaim correspond à une population d'agents simples, appelés particules. Chaque particule est considérée comme une solution probable du problème, elle est caractérisée par une position (le vecteur solution) et une vitesse. De plus, chaque particule possède une mémoire lui permettant ainsi de mémoriser sa meilleure performance (en position et en valeur) et la meilleure performance atteinte par les particules voisines (informatrices) : chaque particule dispose en effet d'un groupe d'informatrices, historiquement appelé son voisinage.

Dans le processus de recherche de la solution optimale, chaque particule est influencée par

**Algorithme 4** Algorithme de colonies de fourmis pour le problème du voyageur de commerce

$t \leftarrow 1$
**tant que** le critère d'arrêt n'est pas satisfait **faire**
  **pour** $k = 1$ à m **faire**
    **Choisir** une ville au hasard
    **pour** chaque ville non visitée $i$ **faire**
      **Choisir** une ville $j$ dans la liste $j_i^k$ des villes restantes
    **fin pour**
    **Déposer** une quantité de phéromones $\Delta\tau_{ij}^k(t)$ sur le trajet $T^k(t)$
  **fin pour**
  **Évaporer** les phéromones
  $t \leftarrow t + 1$
**fin tant que**

les trois composantes suivantes :

- *Une composante d'inertie* : la particule tend à suivre sa direction courante de déplacement.
- *Une composante cognitive* : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée.
- *Une composante sociale* : la particule tend à se fier à l'expérience de ses congénères et ainsi à se diriger vers le meilleur site déjà atteint par ses voisins.

La stratégie de déplacement d'une particule est illustrée dans la figure 2.5.

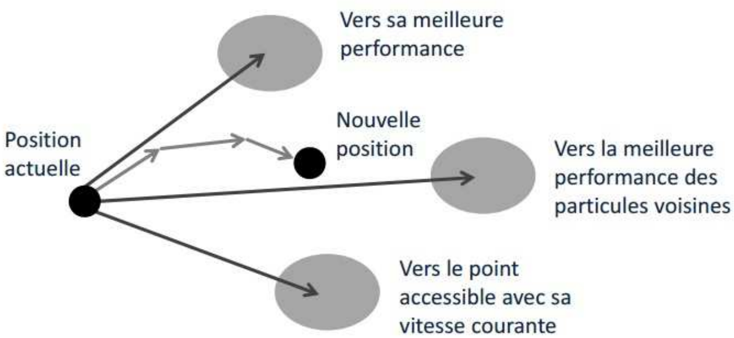



FIGURE 2.5 – Stratégie de déplacement d'une particule.

Afin d'illustrer le principe de cet algorithme, on considère un espace de recherche de dimension $D$, la particule $i$ de l'essaim est modélisée par son vecteur de position $\overrightarrow{x}_i = (x_{i1}, x_{i2},..., x_{iD})$ et par son vecteur vitesse $\overrightarrow{v}_i = (v_{i1}, v_{i2},..., v_{iD})$. La qualité de sa position est déterminée par sa valeur retournée vis-à-vis de la fonction objectif. Cette particule sauvegarde en mémoire la meilleure position par laquelle elle est déjà passée, que l'on note $\overrightarrow{P}best_i = (pbest_{i1}, ..., pbest_{iD})$. La meilleure position atteinte par les particules de l'essaim est notée $\overrightarrow{G}best = (gbest_1, ..., gbest_D)$. On raisonne par rapport à la version globale du PSO, où toutes les particules sont considérées comme voisines de la particule $i$, d'où la notation $\overrightarrow{G}best$ (*Global best*).

Au départ, les particules de l'essaim sont initialisées de manière aléatoire dans l'espace de recherche. Ensuite, à chaque itération, chaque particule se déplace, en combinant linéairement trois composantes. A l'itération $t+1$, le vecteur vitesse et le vecteur position sont calculés à partir des équations 2.3.1 et 2.3.2 respectivement :

$$v_{ij}^{t+1} = wv_{ij}^{t} + c1r1_{ij}^{t}[pbest_{ij}^{t} - x_{ij}^{t}] + c2r2_{ij}^{t}[gbest_{j}^{t} - x_{ij}^{t}], j \in \{1, 2,..., D\} \tag{2.3.1}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1}, j \in \{1, 2, ..., D\} \tag{2.3.2}$$

où $w$ est une constante, appelée coefficient d'inertie ; $c1$ et $c2$ sont deux constantes, appelées coefficients d'accélération ; $r1$ et $r2$ sont deux nombres aléatoires tirés uniformément dans [0, 1] à chaque itération $t$ et pour chaque dimension $j$. Les trois composantes mentionnées avant (i.e. d'inertie, cognitive et sociale) sont représentées dans l'équation 2.3.1 par les termes suivants :

1. $wv_{ij}^{t}$ correspond à la composante d'inertie du déplacement, où le paramètre $w$ influence la direction du déplacement sur le déplacement futur.
2. $c1r1_{ij}^{t}[pbest_{ij}^{t} - x_{ij}^{t}]$ correspond à la composante cognitive du déplacement, où le paramètre $c1$ contrôle le comportement cognitif de la particule.
3. $c2r2_{ij}^{t}[gbest_{j}^{t} - x_{ij}^{t}]$ correspond à la composante sociale, où le paramètre $c2$ contrôle l'aptitude sociale de la particule.

Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées et les deux vecteurs $\overrightarrow{P}best_i$ et $\overrightarrow{G}best$ sont mis à jour. L'algorithme 5 résume le principe de cet algorithme, où $N$ est le nombre de particules de l'essaim.

Plusieurs variantes de cet algorithme ont été proposées dans la littérature [El. Dor, 2012].

#### 2.3.4.5 Les colonies d'abeilles

Le comportement des abeilles dans la nature, ainsi que leurs coopérations afin de trouver de la nourriture en indiquant la source la plus proche par des danses particulières et leurs façons de se reproduire ont inspiré des chercheurs pour développer des algorithmes d'optimisation. De ce fait, plusieurs algorithmes se basant sur ces comportements ont vu le jour, comme l'algorithme d'optimisation par colonie d'abeilles virtuelle (VBA) [Talbi, 2009]. Cet algorithme a été proposé

**Algorithme 5** Algorithme d'optimisation par essaim particulaire

1: **Initialiser** aléatoirement $N$ particules : position et vitesse
2: **Évaluer** les positions des particules
3: **Pour** chaque particule $i$, $\overrightarrow{P}best_i = \overrightarrow{x_i}$
4: **Calculer** $\overrightarrow{G}best$
5: **tant que** le critère d'arrêt n'est pas satisfait **faire**
6: **Déplacer** les particules
7: **Évaluer** les positions des particules
8: **Mettre à jour** $\overrightarrow{P}best_i$ et $\overrightarrow{G}best$
9: **fin tant que**

par Xin-She Yang en 2005 pour la résolution des problèmes numériques d'optimisation, il peut optimiser des fonctions continues et des problèmes discrets. L'arrangement de l'algorithme VBA commence par un groupe d'abeilles virtuelles, chacune d'entre elles se déplace aléatoirement dans l'espace de recherche. Dans la plupart des cas, cet espace peut être simplement un espace de recherche 1-D ou 2-D. Les étapes principales de l'algorithme VBA sont :

— Création de la population d'abeilles (abeilles virtuelles). Chaque abeille est associée à un vecteur de solution avec plusieurs paramètres à optimiser.
— Codage des fonctions d'optimisation (fonctions objectifs) et conservation de nourriture virtuelle.
— Définition d'un critère dans le but de communiquer la direction et la distance en prenant en compte l'aptitude physique des abeilles (danse des abeilles).
— Mise à jour de la population des individus dans de nouvelles positions pour la recherche de la nourriture virtuelle, en utilisant la danse virtuelle pour définir la distance et la direction.
— Après une certaine période d'évolution, la meilleure évaluation correspond à la source la plus visitée par les abeilles.
— Décodage des résultats pour l'obtention de la solution du problème.

La position modifiée de chaque abeille peut être calculée en utilisant les équations 2.3.3 et 2.3.4.

$$x_k^{i+1} = x_k^i.(1-\beta) + x_{best}.\beta + \alpha.(rand(i) - 0.5) \tag{2.3.3}$$

$$y_k^{i+1} = y_k^i.(1-\beta) + y_{best}.\beta + \alpha.(rand(i) - 0.5) \tag{2.3.4}$$

Où $\alpha$ et $\beta$ sont des constantes positives appelées amplitude aléatoire et vitesse de convergence respectivement, $x_{best}$ et $y_{best}$ sont les meilleurs paramètres à la $i^{eme}$ itération et $rand(i)$ est un nombre aléatoire dans l'intervalle [0,1].

La $i^{eme}$ abeille dans la colonie est représentée par un vecteur de dimension $k$ tel que $x^k = (x^1, x^2, x^3, ..., x^k)$ et de $y^k = (y^1, y^2, y^3, ..., y^k)$. La position actuelle (la recherche de point dans l'espace de recherche) peut être modifiée par :

$$S_k^{i+1} = S_k^i + S_{best} + S_{rand}$$

où : $S_k^{i+1} = (x_k^{i+1}, y_k^{i+1})$, $S_k^i = (x_k^i, y_k^i)$ et $S_{best} = (x_{best}, y_{best})$.

D'autres algorithmes inspirés du comportement des abeilles ont été développés comme :

- **Algorithme d'optimisation par colonie d'abeilles BCO** : qui est introduit par Lucie et Teodorovic en 2001, afin de trouver la solution optimale pour un problème d'optimisation combinatoire difficile donné comme le problème du voyageur de commerce.
- **Algorithme d'optimisation par la danse des abeilles DBO** : il a été développé par Laga et Nouioua en 2009 pour la résolution du problème de la T-coloration des graphes. Cet algorithme est inspiré du comportement des abeilles lors de la recherche de la nourriture.
- **Algorithme d'optimisation par mariage des abeilles** : qui est apparu en 2001. Cet algorithme est inspiré du processus biologique de reproduction des abeilles.

#### 2.3.4.6 Autres méthodes

D'autres méthodes qu'on qualifie de métaheuristiques manipulant un ensemble de solutions peuvent être mentionnées, par exemple l'algorithme d'optimisation par le système immunitaire artificiel. Cet algorithme est inspiré du principe de fonctionnement du système immunitaire naturel des vertébrés. Il exploite les caractéristiques du système immunitaire pour ce qui est de l'apprentissage et de la mémorisation comme moyens de résolution du problème [Talbi, 2009].

## 2.4 Application des métaheuristiques en robotique

Comme mentionné précédemment, l'application des métaheuristiques est devenue très usuelle dans de nombreux domaines de l'ingénierie. Ceci est dû à la complexité des problèmes d'optimisation rencontrés, le nombre assez important de contraintes à satisfaire et le facteur temps de calcul joue un rôle important. Le domaine de la robotique est également concerné par l'application des métaheuristiques. En effet, on les trouve pertinentes pour résoudre de nombreux problèmes de robotique, comme :

— Les problèmes de planification de trajectoire [Maria D. G. et al., 2010; Marcos et al., 2012].
— Les problèmes de commande [Chyan & Ponnambalam, 2012].
— Les problèmes de coopération entre robots [Venegas & Marcial R., 2009].

D'autres problèmes peuvent aussi être cités comme les problèmes de vision en robotique. En ce qui concerne les problèmes de planification, on trouve beaucoup plus l'utilisation de métaheuristiques à base de population comme l'algorithme d'optimisation par essaim particulaire [Huang et al., 2008] et les algorithmes évolutionnaires [Maria D. G. et al., 2010; Yun & Xi, 1996]. Le choix de la métaheuristique dépend de la formulation du problème et du nombre de variables de décision (ce qui définit la complexité de l'espace de recherche). À titre d'exemple, dans [Maria D. G. et al., 2010], les auteurs ont utilisé un algorithme génétique afin de résoudre le problème de planification pour des bras manipulateurs hyper-redondants. Dans ce cas, chaque individu est composé des éléments de la matrice jacobienne du robot et de l'erreur de position pour l'effecteur et à chaque itération, l'algorithme fournit une nouvelle position dans l'espace articulaire, en minimisant la fonction objectif.

L'utilisation répandue des algorithmes génétiques montre la complexité accrue de ce genre de problèmes. Cette dernière est liée à la nature du robot utilisé (bras manipulateurs ou robots mobiles, plusieurs robots), la nature de l'environnement dans lequel le robot opère (environnement encombré d'obstacles, incertitudes sur l'environnement, etc) et bien évidemment la tâche à accomplir par le robot ou l'ensemble des robots (minimisation des trajets parcourus, des ressources disponibles, etc). D'autre part, l'application des algorithmes évolutionnaires montre leur grande efficacité dans la résolution de problèmes complexes relativement à d'autres métaheuristiques.

## 2.5 Conclusion

Dans ce chapitre, on a donné un aperçu général sur les différents concepts liés à l'optimisation et les différentes techniques de résolution. On a commencé par la définition d'un problème d'optimisation, sa formulation et nous avons cité les principales techniques pour la gestion des contraintes. Ensuite, on a discuté des problèmes d'optimisation qualifiés de difficiles, pour lesquels les méthodes exactes ne permettent pas de trouver des solutions, ce qui justifie l'utilisation des métaheuristiques. Nous avons détaillé le principe de base des métaheuristiques les plus utilisées. Nous avons présenté les premières versions de ces algorithmes. Depuis, beaucoup d'améliorations ont été apportées à ces algorithmes afin de résoudre d'autres classes de problèmes ou bien simplement de rendre l'algorithme plus efficace et plus rapide.

A la fin de ce chapitre, nous avons présenté un aperçu de l'utilisation de métaheuristiques pour résoudre des problèmes liés à la robotique. En effet, au cours de ces dernières années, les métaheuristiques sont très utilisées en robotique et plus particulièrement pour résoudre les problèmes de planification de trajectoires. Dans la suite de notre étude, nous allons nous intéresser aux algorithmes évolutionnaires et plus particulièrement aux algorithmes génétiques, que nous allons adapter et utiliser comme moyen de résolution.

# Algorithmes de planification de trajectoires libres et en présence d'obstacles

## 3.1 Introduction

Dans ce chapitre, nous présentons une nouvelle méthode de planification de trajectoires basée sur l'optimisation à deux niveaux. Le principe de cette approche est de trouver les points (configurations) par lesquels le robot doit passer. Ces points sont le résultat d'un problème d'optimisation dans lequel on prend en considération un certain nombre de contraintes. Dans notre cas, on ne s'intéresse qu'aux bras manipulateurs redondants. On exploite la redondance pour l'évitement d'obstacles et les singularités pour le robot. Ainsi, une nouvelle technique d'évitement d'obstacles sera aussi présentée dans ce chapitre. Pour la résolution, on utilise une métaheuristique basée sur les algorithmes génétiques.

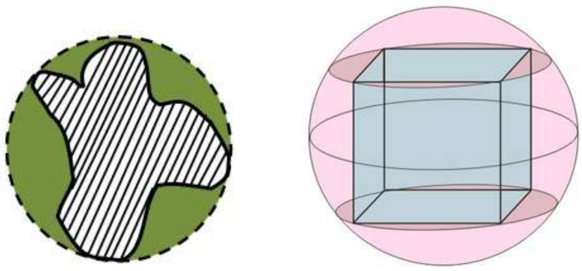
Ce chapitre est organisé comme suit : dans la section 3.2, on présente la technique utilisée pour l'évitement d'obstacles. La section 3.3 est dédiée à l'introduction des problèmes d'optimisation à deux niveaux d'une manière générale. Dans la section 3.4, on détaille la formulation proposée et l'algorithme de résolution est décrit dans la section 3.5. Des résultats de simulation sont présentés dans la section 3.6 et nous concluons ce chapitre dans la section 3.7.

## 3.2 Évitement d'obstacles

Dans cette section, nous présentons une nouvelle technique pour l'évitement d'obstacles. Cette technique est basique mais elle nous permet d'avoir un caractère adaptatif dans la formulation proposée du problème de planification.

### 3.2.1 Modélisation des obstacles

Actuellement, tous les robots industriels opèrent dans des environnements encombrés de tous types de formes d'obstacles. Ainsi, une modélisation de ces derniers est nécessaire. Le cas idéal est de les prendre avec leurs formes réelles mais ceci génère beaucoup de calculs. Pour simplifier ces derniers et réduire la complexité des algorithmes dédiés à l'évitement d'obstacles, ils sont modélisés dans la littérature par des hypersurfaces (cercles, sphères, ellipses, etc). Ces techniques de représentation présentent l'avantage de faciliter le calcul de la distance entre le robot et les obstacles. En revanche, l'inconvénient majeur avec ces nouvelles formes d'obstacles réside dans la réduction de l'espace de travail libre pour le robot. Dans notre cas, on a opté pour cette solution parce qu'elle est la mieux adaptée à la technique d'évitement d'obstacles qui sera présentée par la suite. Ainsi, en partant de n'importe quelle forme originale de l'obstacle, on l'enveloppe avec l'hypersurface de rayon minimal. Un exemple de cette représentation est montré dans la figure 3.1 :



(a) Représentation en 2D. (b) Représentation en 3D.

FIGURE 3.1 – Modélisation des obstacles.

Ainsi, la figure 3.1(a) montre le cas 2D où l'on englobe l'obstacle par un cercle de rayon minimal et la figure 3.1(b) représente le cas 3D où un cube est englobé par une sphère.

### 3.2.2 Technique d'évitement d'obstacles

Dans la littérature, on peut trouver énormément de techniques d'évitement d'obstacles [Pourazady & Ho, 1991; Le Boudec et al., 2006; Daachi et al., 2012; Perdereau et al., 2002]. Chacune d'entre elles se distingue par la forme choisie du robot, des obstacles et la mesure de distance entre les deux. Dans notre cas, afin de mesurer la distance entre le robot et les différents obstacles, on définit des points de contrôle. Pour définir ces points, on prend le centre de chaque articulation et l'intersection de la normale passant par le centre de chaque obstacle et la droite formée par les deux centres de deux articulations successives, comme on peut le voir sur la figure 3.2.
Pour chaque obstacle et par rapport à chaque bras du robot, la mesure de distance s'effectue sur le plan défini par le centre de l'obstacle et les deux centres de deux articulations successives. Ainsi, sur la même figure, le plan est formé par les points $B$, $C$ et $C1$. Pour illustrer cette technique, on prend un robot planaire à 3 degrés de liberté et un obstacle sous la forme d'un disque, comme illustré sur la figure 3.3 :

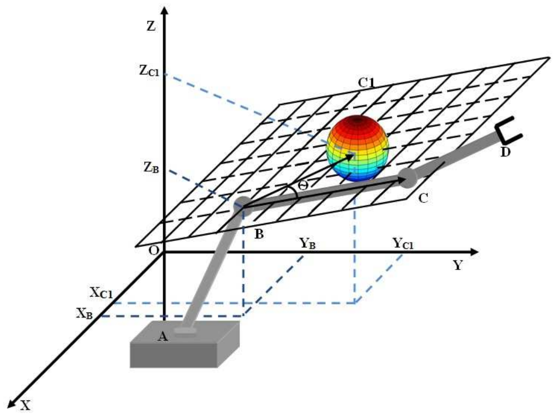



FIGURE 3.2 – Plan utilisé pour la mesure des distances.

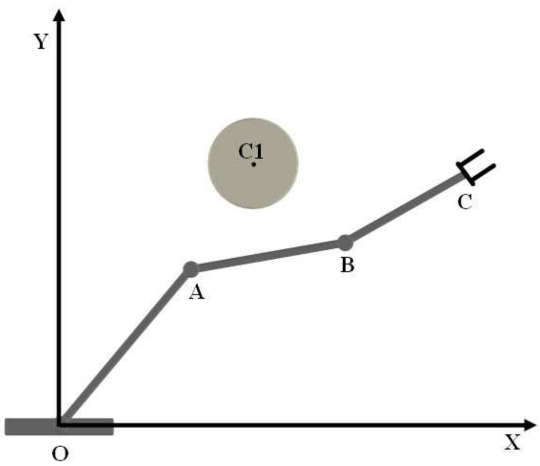
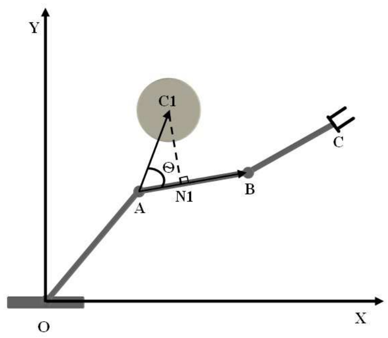



FIGURE 3.3 – Robot planaire avec un obstacle.

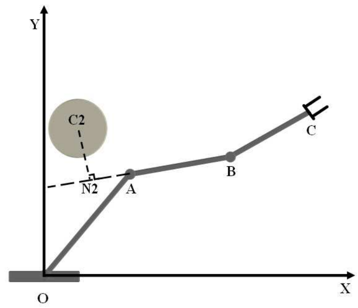
Ce robot possède 3 bras : $OA$, $AB$ et $BC$. On illustre la technique uniquement pour le bras $AB$ mais, au niveau de l'implémentation, la méthode est appliquée pour chaque bras et par rapport à chaque obstacle. Avec une telle configuration pour le robot et l'obstacle, on peut se retrouver avec 3 cas différents, comme on peut le voir sur la figure 3.4. Le premier cas est représenté dans la figure 3.4(a). Dans ce cas, la normale passant par le centre de l'obstacle (le point $C1$) appartient à la droite formée par les deux points $A$ et $B$. Ainsi, on doit vérifier que la distance $|N1C1|$ est supérieure au rayon de l'obstacle. Les deux autres cas sont représentés respectivement dans les figures 3.4(b) et 3.4(c). Dans ces cas, en traçant la normale, le point d'intersection n'appartient pas à la droite $AB$. Ainsi, on n'est pas obligé de vérifier si la distance $|N1C1|$ est supérieure ou non au rayon de l'obstacle. Afin de vérifier que le point $N1$ appartient ou non à la droite $AB$, on procède de deux manières différentes au calcul du produit scalaire des deux vecteurs $\overrightarrow{AB}$ et $\overrightarrow{AC1}$. Pour que le point $N1$ appartienne à la droite $AB$ (figure 3.4(a)), il faut avoir :

$$|\overrightarrow{AC1}| \cdot \cos(\theta) \leq |\overrightarrow{AB}|$$
$$\text{avec } -\pi/2 \leq \theta \leq \pi/2$$

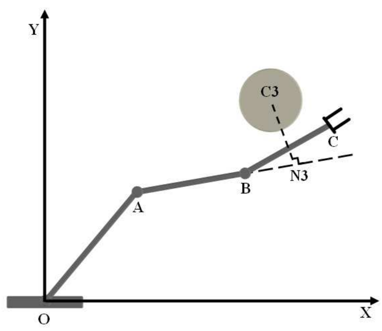
$|\overrightarrow{AC1}|$ et $|\overrightarrow{AB}|$ représentent les normes des deux vecteurs $\overrightarrow{AB}$ et $\overrightarrow{AC1}$ respectivement. La conséquence d'une telle technique est que le nombre de points de contrôle devient variable. En effet, il dépend de la position de chaque obstacle par rapport à chaque bras. Ainsi, à chaque itération, pour chaque bras, on peut avoir deux ou trois points de contrôle.




(a) 1 cas.




(b) 2 cas.




(c) 3 cas.

FIGURE 3.4 – Exemple d'illustration de la détection d'obstacles.

## 3.3 Problèmes d'optimisation à deux niveaux

Les problèmes d'optimisation à deux niveaux font partie des problèmes d'optimisation hiérarchique où l'on inclut un processus de décision. Dans ce genre de problème, un sous-ensemble des variables de décision est géré par le premier niveau et le reste des variables est géré par le deuxième niveau. Ainsi, le deuxième niveau tente d'optimiser sa propre fonction objectif en prenant en compte certaines décisions prises par le premier niveau. En retour, le premier niveau optimise sa fonction objectif en fonction des décisions prises par le deuxième niveau. Par conséquent, ce genre de problème est caractérisé par l'existence de deux problèmes d'optimisation. Un problème d'optimisation à deux niveaux peut être formulé comme suit :

$$\begin{aligned}
&\min_x F(x,y)\\
&\qquad \text{s.t.}\\
&\qquad\quad G(x,y) \leq 0\\
&\qquad\quad \min_y\ f(x,y)\\
&\qquad\qquad \text{s.t.}\\
&\qquad\qquad\quad g(x,y) \leq 0
\end{aligned}$$

$x \in R^{n1}$ et $y \in R^{n2}$ sont les variables contrôlées par le premier et le deuxième niveau respectivement. $F$, $f : R^n \to R$, $n = n1 + n2$. $G(x,y)$ et $g(x,y)$ représentent les contraintes d'inégalité.

## 3.4 Formulation du problème de planification de trajectoires

Dans cette section, nous proposons une formulation pour le problème de planification de trajectoires basée sur les problèmes d'optimisation à deux niveaux. L'idée est que, dans le premier niveau, on cherche de nouvelles positions pour l'effecteur (organe terminal du robot) tout en restant loin des différents obstacles. Une fois la position trouvée, le deuxième niveau se charge de trouver la meilleure configuration (en termes de variables articulaires) en exploitant la redondance du robot. Ainsi, la formulation est la suivante :

$$\min_X F(X,\theta) = \alpha \cdot F1(X) + \beta \cdot (X - X_c) + \gamma \cdot F2(\theta) \tag{3.4.1}$$

Sous les contraintes :

$$G1(X) \leq 0,\ G2(X) \leq 0$$

$$\min_\theta L(\theta) = \delta \cdot F3(\theta) + \zeta \cdot F4(\theta) \tag{3.4.2}$$

Sous les contraintes :

$$g1(\theta) \leq 0,\ g2(\theta) \leq 0$$

$$H(X,\theta) = 0$$

Avec $\alpha+\beta-\gamma = 1$ et $\zeta-\delta = 1$. $\alpha \in [0,1]$, $\beta \in [0,1]$, $\gamma \in [0,1]$, $\delta \in [0,1]$, $\zeta \in [0,1]$, $\alpha > \beta$ et $\alpha > \gamma$. $\alpha$ est le paramètre de pondération le plus important. On doit s'assurer qu'il est plus grand que

$\beta$ et $\gamma$. Avec un tel choix, on donne plus d'importance à la fonction $F1(X)$, ce qui nous permet d'atteindre la position finale. Les différents critères et paramètres sont décrits ci-dessous.

### 3.4.1 Fonctions objectifs

Dans la formulation proposée, on a deux fonctions objectifs, représentées par les équations 3.4.1 et 3.4.2. L'équation 3.4.1 représente la fonction objectif du premier niveau, elle est composée de trois critères. Comme déjà indiqué, le but de ce niveau est d'atteindre la position finale, tout en gardant les points de contrôle loin des différents obstacles. Pour ce niveau, les points de contrôle sont uniquement les centres de chaque articulation. Le premier critère est décrit par la fonction $F1(X)$, il représente l'erreur de position entre les positions courante et finale pour l'effecteur :

$$F1(X) = \sqrt{(x - xf)^2 + (y - yf)^2} \tag{3.4.3}$$

$X = (x,y)$ représente la position courante de l'effecteur et $(xf,yf)$ est la position finale à atteindre.

Le deuxième terme $(X - Xc)$ de l'équation 3.4.1 nous permet de prendre en considération l'erreur trouvée par les deux niveaux pour la même position de l'effecteur. La valeur de $X_c$ est trouvée par le deuxième niveau en satisfaisant toutes les contraintes. Ainsi, le terme $X - X_c$ nous donne une idée sur le degré de satisfaction des contraintes. Par conséquent, en ajoutant ce terme dans la fonction objectif, on cherche à minimiser au fil des générations le degré d'insatisfaction des contraintes causées par le deuxième niveau.
Le dernier terme de l'équation 3.4.1 est représenté par la fonction $F2(\theta)$. Cette fonction a pour objectif de garder les points de contrôle (centres des articulations) loin des différents obstacles. Son expression est :

$$F2(\theta) = \sum_{i=1}^{N} \sum_{j=1}^{M} d(\theta)\{PC_i,C_j\} \tag{3.4.4}$$

$d(\theta)\{PC_i,C_j\}$ est la distance euclidienne entre le point de contrôle $PC_i$ et le centre de chaque articulation $C_j$, $N$ est le nombre total de points de contrôle et $M$ est le nombre d'obstacles. Comme déjà précisé précédemment, pour le premier niveau, les points de contrôle sont uniquement les centres de chaque articulation. Sans l'utilisation de ce critère, le premier niveau tentera de minimiser sa fonction objectif sans prendre en compte l'aspect collision avec les obstacles. Ainsi, des collisions peuvent se reproduire et il sera plus difficile pour le deuxième niveau de trouver une solution admissible satisfaisant toutes les contraintes. La valeur de $\theta$ est retournée

par le deuxième niveau.
L'équation 3.4.2 représente la fonction objectif du deuxième niveau, elle est composée de deux critères. Le but de ce niveau est de trouver la meilleure configuration pour chacune des positions transmises par le premier niveau. Le premier critère est représenté par la fonction $F3(\theta)$ qui est utilisée pour maximiser la manipulabilité du robot, son expression est :

$$F3(\theta) = |\det\{J(\theta) \cdot J^T(\theta)\}|^{1/2} \tag{3.4.5}$$

$J$ est la jacobienne du robot et $J^T$ est sa transposée. La matrice jacobienne est obtenue par une dérivation du modèle géométrique direct du robot. Ainsi la matrice jacobienne nous permet d'avoir la relation des vitesses entre l'espace des tâches et l'espace articulaire, comme le montre l'équation ci-dessous :

$$\dot{X} = J(\theta) \cdot \dot{\theta} \tag{3.4.6}$$

$\dot{X}$ et $\dot{\theta}$ sont les vitesses respectives dans l'espace des tâches et l'espace articulaire.. Dans notre cas, la matrice jacobienne n'est pas carrée (à cause de la redondance). Cependant, avec la formulation proposée, on n'est pas obligé de l'inverser. La manipulabilité est considérée comme l'un des indices de performance du robot [Kucuk & Bingul, 2006]. En maximisant cette variable, on applique moins d'efforts pour déplacer le robot. Ainsi, on minimise le couple appliqué indirectement. D'un autre côté, la manipulabilité nous permet d'avoir une mesure de rapprochement du robot avec ses configurations singulières [Kucuk & Bingul, 2006; Tsai & Chiou, 1990]. Par conséquent, travailler avec un maximum de manipulabilité permet de s'éloigner de ces configurations.

Le deuxième critère est représenté par la fonction $F4(\theta)$. Cette fonction est utilisée pour minimiser la variation des variables articulaires et ainsi avoir de petits déplacements. Son expression est :

$$F4(\theta) = 1/2 \cdot \sum_{i=1}^{K} q_i^2(\theta) \tag{3.4.7}$$

Les $q_i$ représentent les variables articulaires et $K$ leur nombre total.

### 3.4.2 Contraintes

Comme pour les fonctions objectifs, dans la formulation proposée, on a des contraintes pour chacun des deux niveaux. En ce qui concerne le premier niveau, elles sont représentées par $G1(X)$ et $G2(X)$. La contrainte $G1(X)$ est utilisée pour s'assurer que l'effecteur n'entre pas en collision avec les différents obstacles. La contrainte $G2(X)$ sert à minimiser le déplacement de l'effecteur, son expression est :

$$G2(X) = |X(i+1) - X(i)| \leq R$$

$X(i+1)$ est la position trouvée pour l'effecteur à l'étape $(i+1)$ et $X(i)$ est sa position à l'étape $(i)$. Le choix de la valeur de $R$ est très important parce que c'est cette dernière qui délimite l'espace de recherche à chaque étape pour le premier niveau. Ainsi, à chaque itération, on cherche une nouvelle position pour l'effecteur à l'intérieur d'un parallélépipède.

Pour le deuxième niveau, les contraintes sont représentées par $g1(\theta)$, $g2(\theta)$ et $H(X,\theta)$. La contrainte $g1(\theta)$ représente les limites inférieures et supérieures des différentes variables articulaires. La contrainte $g2(\theta)$ est utilisée pour l'évitement d'obstacle. Dans ce cas, on utilise la technique décrite dans la section 3.2. La contrainte d'égalité $H(X,\theta)$ représente le modèle géométrique direct du robot que l'on peut calculer avec la convention de Denavit Hartenberg [Shukla et al., 2013]. Ce modèle représente la relation entre les coordonnées de l'effecteur et les variables articulaires :

$$X = F(\theta) \text{ et } H(X,\theta) = X - F(\theta)$$

## 3.5 Résolution : algorithme bi-génétique

Les problèmes d'optimisation à deux niveaux sont des modèles appropriés pour une large gamme de processus industriels. Cependant, leur utilisation est très limitée. Une des raisons de ce fait est le manque d'algorithmes efficaces pour la résolution de ce genre de formulation. En effet, les problèmes d'optimisation à deux niveaux sont classifiés dans la catégorie des problèmes NP-complets [Calvete et al., 2008]. Dans la littérature, on peut trouver beaucoup de travaux concernant ce genre de problème et ainsi beaucoup de techniques ont pu être développées [Jiang et al., 2013; Oduguwa & Roy, 2002; Hejazi et al., 2002]. On peut diviser ces techniques en deux grandes catégories. La première se base sur la transformation du problème à deux niveaux en un problème à un seul niveau, en utilisant la méthode de Karush-Kuhn-Tucker [Jiang et al., 2013; Lv et al., 2007] et ensuite sur la résolution de la nouvelle formulation. La deuxième catégorie tente de résoudre le problème directement sans aucune transformation [Oduguwa & Roy, 2002; Li & Wang, 2006]. Pour chacune des deux catégories, on peut trouver énormément d'algorithmes et chacun d'eux traite une forme particulière de ce problème (en fonction de la fonction objectif, la nature des contraintes, etc). Ainsi, on peut trouver l'utilisation de l'algorithme d'optimisation par essaim particulaire [Jiang et al., 2013], une méthode utilisant des fonctions de pénalités [Wan et al., 2011], les algorithmes évolutionnaires [Oduguwa & Roy, 2002; Hejazi et al., 2002], etc. Dans notre cas, on a opté pour l'utilisation d'une métaheuristique basée sur les algorithmes génétiques. Beaucoup de travaux ont pu prouver leur efficacité par rapport à d'autres algorithmes d'optimisation. On utilise un algorithme génétique pour chaque niveau, comme on peut le voir sur la figure 3.5 :

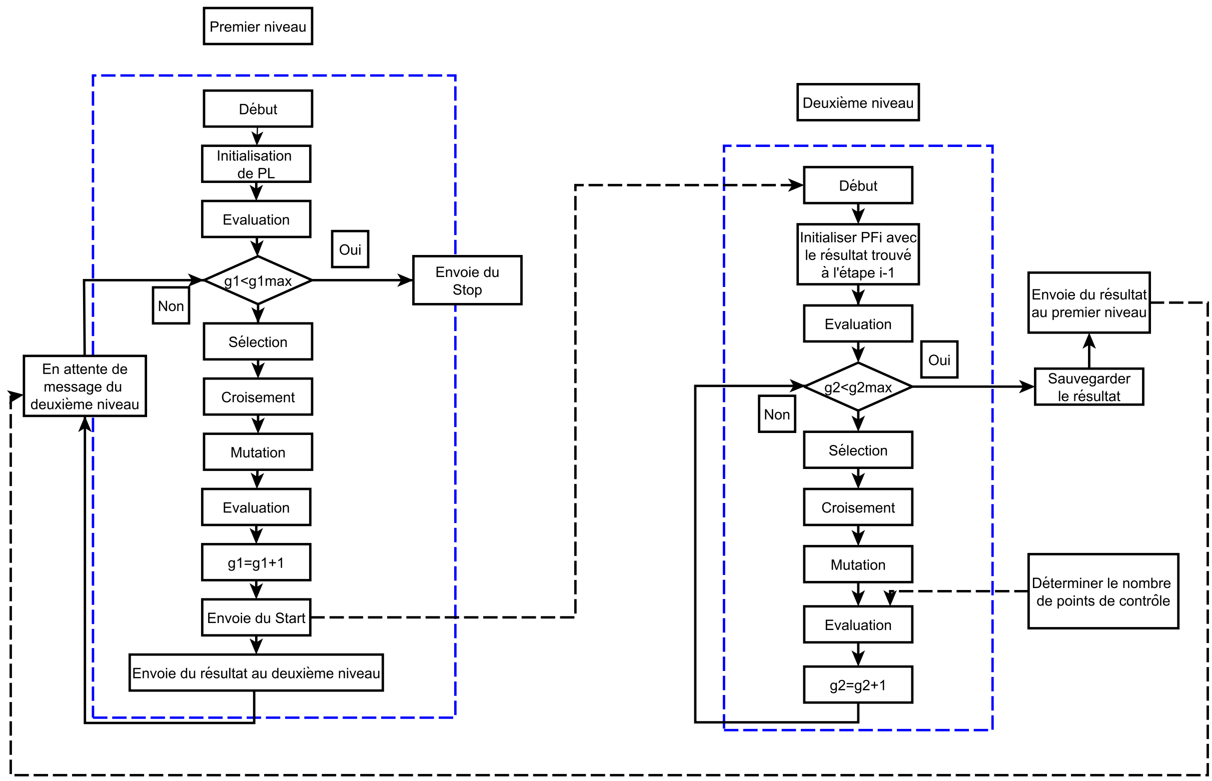



FIGURE 3.5 – Représentation de l'algorithme bi-génétique.

Comme on peut le voir sur la figure 3.5, on a deux parties : celle qui représente le premier niveau (à gauche) et celle du deuxième niveau (à droite).

Dans la formulation présentée dans la section 3.4, les variables de décision (à optimiser) pour le premier niveau sont les coordonnées de l'effecteur dans l'espace cartésien. Pour le deuxième niveau, les variables de décision sont les variables articulaires. Au début de l'exécution, toute la population du premier niveau est initialisée au même point, qui représente la configuration initiale du robot. Après évaluation de la population, on applique les opérateurs de l'algorithme génétique, à savoir, la sélection, le croisement et la mutation. A chaque génération, le premier niveau envoie son résultat pour le deuxième niveau, ce qui lui permet de déclencher son exécution. Après cela, il se met en état d'attente du résultat produit par le deuxième niveau avant de continuer son exécution.

Le deuxième niveau, après réception des données du premier niveau, commence son exécution. A chaque exécution du deuxième niveau, on initialise la population avec le résultat trouvé dans la précédente étape. Procéder de cette façon est important pour la fonction $F4(\theta)$. En effet, cette fonction a pour objectif de minimiser les variations des variables articulaires entre la précédente et la nouvelle solution à trouver, ce qui permet aussi d'accélérer l'exécution du code. Une fois la population initialisée, on applique les opérateurs de l'algorithme génétique durant $g2max$ générations, qui correspondent au maximum de générations pour le deuxième niveau. A chaque $g2max$, le deuxième niveau suspend son exécution, sauvegarde le résultat trouvé et l'envoie au premier niveau. Après, il se met en attente de données provenant du premier niveau. Le temps d'exécution et la complexité du deuxième niveau dépendent du nombre et de la position des différents obstacles. En effet, si on a un nombre élevé d'obstacles, le nombre de contraintes augmente et ainsi, l'exécution prend plus de temps.
Au maximum de générations $g1max$, le premier niveau envoie un message "STOP" au deuxième niveau et ainsi est marquée la fin de l'exécution.

## 3.6 Résultats de simulation sur le modèle du robot Neuromate

Pour tester la méthode proposée, on utilise le modèle du robot Neuromate. Ce robot est utilisé dans les interventions chirurgicales au niveau de la tête, comme dans le traitement de la maladie de l'hydrocéphalie. Ainsi, planifier une trajectoire pour ce robot consiste à le ramener de n'importe quelle configuration vers celle où l'effecteur sera placé à un endroit précis sur la tête du patient. Plus de détails sur l'utilisation de ce robot peuvent être trouvés dans [Renishaw]. La figure 3.6 montre une représentation de ce robot :

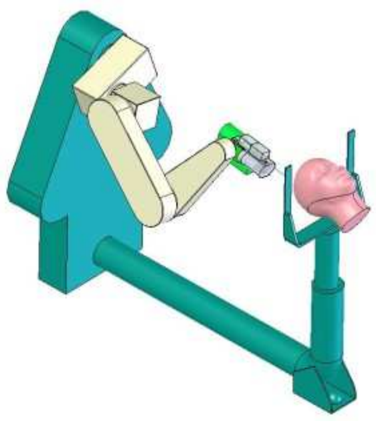


FIGURE 3.6 – Représentation du robot **Neuromate**.

Le robot possède cinq degrés de liberté. Toutes les articulations sont rotatives. Le modèle géométrique du robot est calculé par le biais de la convention de Denavit-Hartenberg [Shukla et al., 2013] et le modèle cinématique est obtenu par une simple dérivation de ce dernier. Les détails de ces calculs sont donnés en annexe. Les limites articulaires pour chaque articulation sont résumées dans le tableau suivant :

| Joints | Min(rad) | Max(rad) |
|---|---|---|
| $\theta_1$ | $-\pi$ | $\pi$ |
| $\theta_2$ | $-\pi$ | $\pi$ |
| $\theta_3$ | $-\pi/2$ | $\pi/2$ |
| $\theta_4$ | $-\pi/2$ | $\pi/2$ |
| $\theta_5$ | $-\pi/2$ | $\pi/2$ |

TABLEAU 3.1 – Limites articulaires.

La position initiale de l'effecteur est $[1{,}1727cm, -6{,}3085cm, -0{,}75cm]$ et la position finale à atteindre est $[5{,}5cm, 1{,}5cm, 0{,}8cm]$. Pour l'évitement d'obstacles, on en considère 15, tous avec le même rayon, qui est de $0{,}4cm$, leurs positions dans l'espace de travail du robot sont indiquées dans le tableau suivant :

| Positions | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ | $S_9$ | $S_{10}$ | $S_{11}$ | $S_{12}$ | $S_{13}$ | $S_{14}$ | $S_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $X(cm)$ | 1,0 | 0,5 | 1,5 | 2,0 | 2,0 | 2,0 | 3,0 | 2,5 | 3,5 | 4,5 | 4,5 | 4,5 | 4,0 | 3,0 | 3,5 |
| $Y(cm)$ | −4,0 | −4,0 | −4,0 | −3,0 | −1,0 | −2,0 | 0,0 | 0,0 | 0,0 | 1,5 | 2,0 | 2,5 | 2,0 | 2,0 | 2,0 |
| $Z(cm)$ | −0,5 | −0,5 | −0,5 | −1,2 | −1,2 | −1,2 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | 1,0 | 0,0 | 0,0 | 0,0 |

TABLEAU 3.2 – Position des obstacles dans l'espace de travail du robot.

Le chromosome du premier niveau est composé de trois gènes qui représentent les coordonnées de l'espace cartésien en 3 dimensions. Pour le deuxième niveau, le chromosome est composé de cinq gènes représentant les variables articulaires, comme on peut le voir sur la figure 3.7 :

| X | Y | Z |
|---|---|---|

(a) Premier niveau.

| Θ1 | Θ2 | Θ3 | Θ4 | Θ5 |
|---|---|---|---|---|

(b) Deuxième niveau.

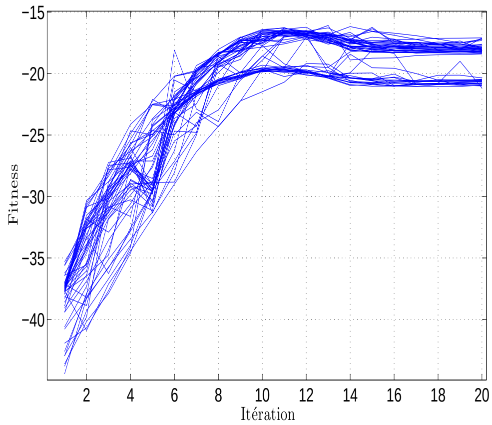
FIGURE 3.7 – Les chromosomes de l'algorithme.

Pour le premier test réalisé, les paramètres de pondération sont choisis de sorte à donner plus d'importance au premier critère (erreur de position) représenté par l'équation 3.4.3. Ainsi, on prend $\alpha = 0{,}4$, $\beta = 0{,}3$ et $\gamma = -0{,}3$. Pour le deuxième niveau, on donne plus d'importance aux variations des variables articulaires (équation 3.4.7). Ainsi, on aura $\delta = -0{,}4$ et $\zeta = 0{,}6$. Le but de ce choix est de valider l'algorithme et la formulation proposée et, par la même occasion, de tester les paramètres choisis pour les deux algorithmes génétiques qui sont représentés dans le tableau suivant :

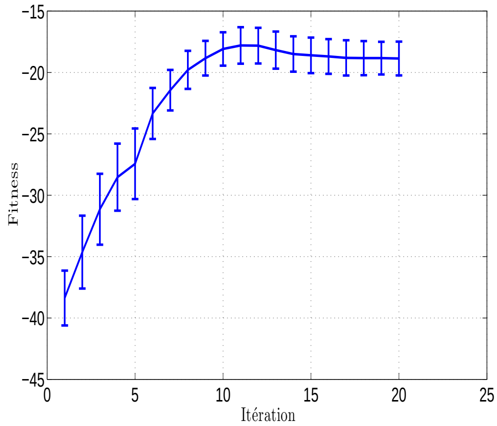
| Paramètres | Premier niveau | Deuxième niveau |
|---|---|---|
| Taille de la population | 60 | 70 |
| Nombre de générations | 20 | 60 |
| Taux de mutation | 0,2 | 0,3 |
| Probabilité de croisement | 0,6 | 0,6 |
| Taille du tournoi | 4 | 4 |

TABLEAU 3.3 – Paramètres de l'algorithme.

Dans la tableau 3.3, seul le nombre de générations et la taille de la population ont été réglés durant les tests. Pour les autres paramètres, on prend les valeurs par défaut utilisées dans la littérature. Comme l'algorithme utilise des opérateurs probabilistes, on est obligé de le relancer plusieurs fois afin de valider les résultats des tests. Dans notre cas, on le relance 50 fois. Les résultats de simulation sont représentés dans les figures ci-dessous. Les deux figures 3.8(a) et 3.8(b) présentent l'évolution du meilleur individu pour le premier niveau. Dans la figure 3.8, on peut voir l'influence des opérateurs de l'algorithme génétique. En effet, on peut remarquer la diversification due à l'opérateur de mutation et l'intensification due à l'opérateur de croisement. Ainsi, on peut valider le choix pour les paramètres concernant l'algorithme (tableau 3.3).



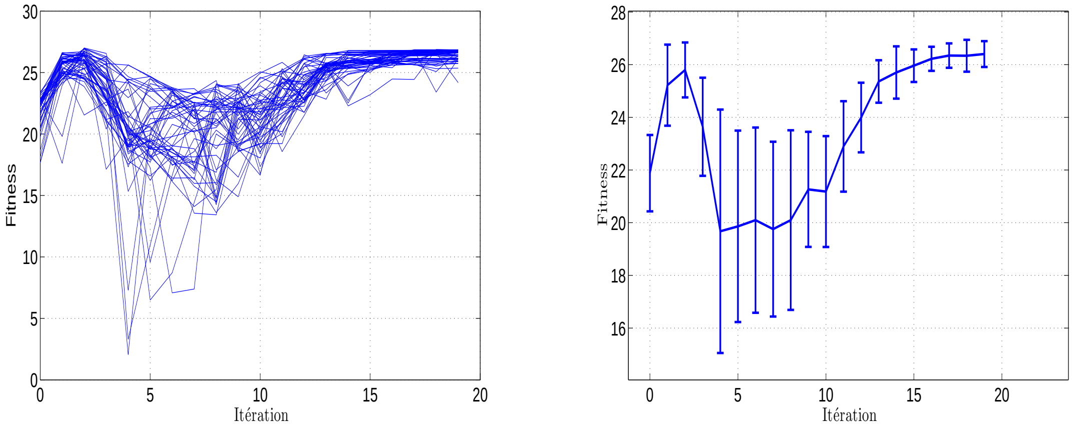

(a) Toutes les fonctions de *fitness*.




(b) Moyenne et écart-type.

FIGURE 3.8 – Évolution du meilleur individu : premier niveau.

Dans la figure 3.8(b), on peut remarquer que la valeur de la fonction objectif augmente et après se stabilise à la fin (à la convergence). On peut aussi noter que l'écart-type est petit au début et à la convergence, alors qu'il est plus important autre part. Comme on initialise toute la population au même point au départ, théoriquement, on devait avoir un écart-type nul. Les résultats montrent un petit écart-type parce que la valeur du second membre dans la fonction objectif $(X - X_c)$ n'est pas nulle. A la convergence, on a aussi un petit écart-type, c'est dû aux valeurs transmises par le deuxième niveau et par le fait qu'on peut atteindre la position finale par plusieurs chemins. Les deux figures qui suivent montrent le même type de résultat au deuxième niveau :



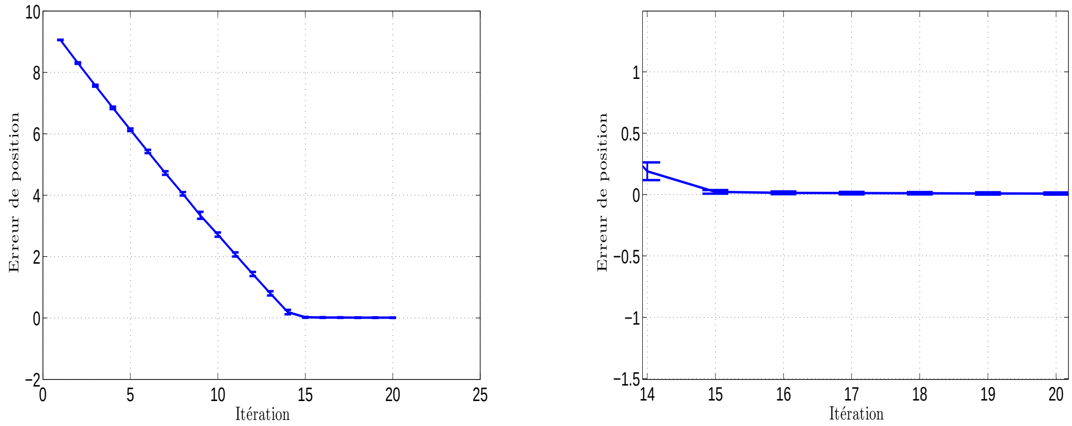

(a) Toutes les fonctions de *fitness*.　　(b) Moyenne et écart-type.

FIGURE 3.9 – Évolution du meilleur individu : deuxième niveau.

Comme on peut le voir sur la figure 3.9, on a les mêmes conclusions que celles présentées pour le premier niveau, en ce qui concerne le choix des paramètres de l'algorithme génétique, ce qui confirme le bon choix de ces derniers.

L'allure de la courbe présentée dans la figure 3.9 dépend de la disposition choisie pour les différents obstacles. Dans ce niveau, on maximise la manipulabilité du robot et on minimise la variation des variables articulaires. Théoriquement, on devrait avoir une courbe qui augmente tout le temps. Cependant, comme on peut le voir sur la figure 3.9, on a une partie de la courbe qui décroît. Ceci est dû à la position choisie pour certains obstacles (figure 3.12). En effet, à cette étape, les obstacles sont très proches du robot. Ainsi, le nombre de contraintes augmente et l'espace libre du robot diminue.

Dans le premier niveau, le plus important des critères est l'erreur de position pour l'effecteur (ce qui nous permet d'atteindre la position finale : fonction $F1(X)$). Ainsi, il est important de montrer l'évolution de cette erreur, ce qui est fait dans les figures suivantes :




(a) Moyenne et écart-type. (b) Une partie de la courbe.

FIGURE 3.10 – Erreur de position pour l'effecteur.

Comme on peut le constater sur la figure 3.10, l'erreur de position de l'effecteur décroît d'une manière progressive. Ainsi, l'effecteur atteint correctement la position finale. On peut aussi remarquer qu'après les 15 générations de l'exécution, l'erreur de position est très petite. Afin de conclure sur la nécessité ou non d'utiliser plus de générations, on zoome une partie de la courbe entre la 14 ème génération et la dernière génération. Le résultat est présenté dans la figure 3.10(b). Avec cette figure, on peut conclure que l'erreur de position est inférieure à 1 % ce qui est acceptable dans notre cas. Ce résultat permet aussi de valider le nombre de générations choisi pour le premier niveau.

La figure suivante présente le temps CPU nécessaire pour atteindre la position finale :

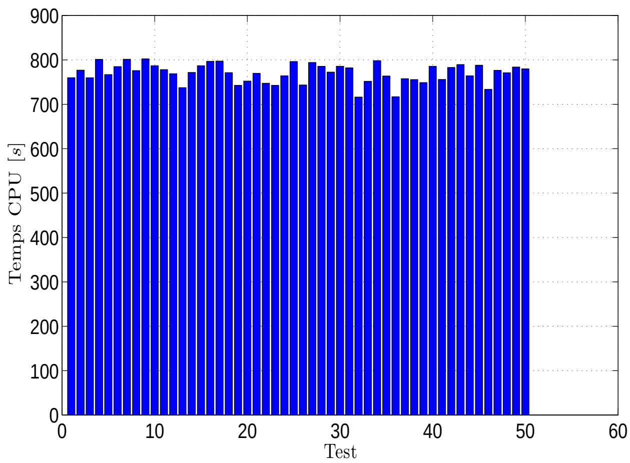
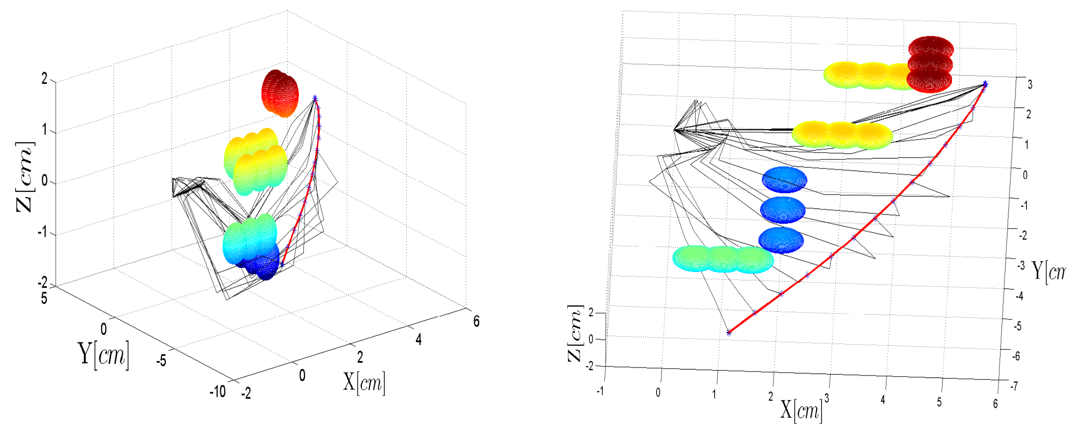



FIGURE 3.11 – Temps CPU.

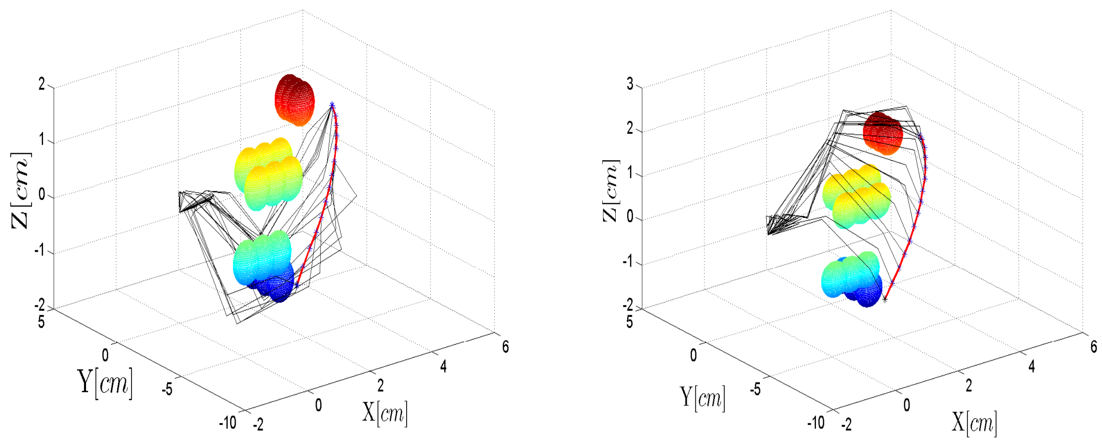
Comme on peut le voir sur la figure 3.11, pour tous les tests réalisés, le temps CPU nécessaire est compris entre 700 et 800 secondes. On ne peut pas conclure d'une manière objective sur ce résultat parce qu'il dépend fortement de la configuration choisie (nombre d'obstacles et leurs positions, la nature du robot). Avoir un nombre important d'obstacles fait augmenter fortement le nombre de contraintes dans la formulation proposée. Ainsi, on aura besoin de plus de temps pour minimiser les fonctions objectifs en satisfaisant toutes les contraintes. La figure suivante présente toutes les configurations obtenues pour le robot :




(a) Vue 1. (b) Vue 2.

FIGURE 3.12 – Configurations successives du robot.

Dans cette figure, on peut voir que le robot atteint sa position finale et évite tous les obstacles qui se trouvent dans son espace de travail. Les points avec le symbole $\star$ représentent les positions successives de l'effecteur. La ligne avec la couleur bleue représente la trajectoire suivie par l'effecteur. La figure suivante présente deux solutions différentes trouvées par l'algorithme :




(a) Solution 1. (b) Solution 2.

FIGURE 3.13 – Deux solutions trouvées par l'algorithme.

Comme on peut le voir, l'algorithme peut trouver deux chemins différents pour atteindre la position finale, du fait des opérateurs probabilistes utilisés par l'algorithme génétique. Ceci explique les résultats présentés précédemment (figure 3.9 et figure 3.10). La figure suivante présente la variation des variables articulaires pour un test choisi arbitrairement :

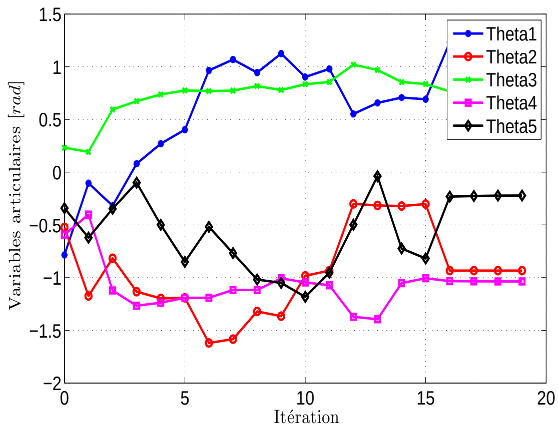
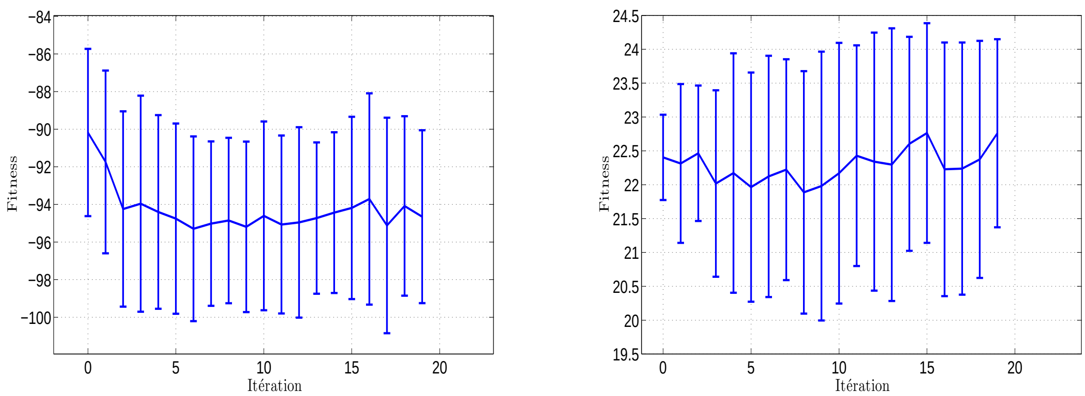



FIGURE 3.14 – Variation des variables articulaires.

Dans cette figure, on peut constater qu'il n'y a pas d'oscillations significatives dans la variation des variables articulaires. Ainsi, on évite au robot des résonances mécaniques et des mouvements brusques. On peut aussi remarquer que les variations sont très petites pour les variables articulaires. Le deuxième cas étudié a pour objectif de montrer l'importance du choix des paramètres de pondération pour les fonctions objectifs. Par exemple, on prend $\alpha = 0{,}1$, $\beta = 0{,}2$ et $\gamma = -0{,}7$ pour la fonction objectif du premier niveau et les mêmes valeurs pour le deuxième niveau. Dans ce cas et pour le premier niveau, on donne plus d'importance pour la fonction $F2(\theta)$ représentée par l'équation 3.4.4. Le résultat est présenté dans la figure suivante :




(a) Premier niveau.

(b) Deuxième niveau.

FIGURE 3.15 – Évolution du meilleur individu.

Comme on peut le voir dans les deux figures 3.15(a) et 3.15(b), les deux fonctions objectifs ne convergent pas à la fin de l'exécution. Ainsi, l'effecteur dans ce cas n'atteindra jamais la position finale. La raison est qu'on donne plus d'importance pour la fonction $F2(\theta)$. Ainsi, le meilleur individu est celui qui possède la plus grande valeur relativement à cette fonction et sa valeur pour la fonction $F1(X)$ ne joue pas un rôle important. Donc, on peut conclure que le choix des paramètres de pondération est très important. Comme déjà mentionné dans la section 3.4, on doit prendre $\alpha$ supérieur à $\beta$ et à $\gamma$ pour assurer la convergence.

## 3.7 Conclusion

Dans ce chapitre, on a présenté une nouvelle technique de planification de trajectoires pour les bras manipulateurs redondants. La formulation proposée est basée sur les problèmes d'optimisation à deux niveaux et une métaheuristique à base d'algorithmes génétiques a été utilisée pour la résolution. Dans la formulation proposée, on a pris en compte tous les aspects liés au problème de planification (évitement d'obstacles, singularité du robot). Le résultat de la technique proposée est un ensemble de configurations par lesquelles le robot doit passer. Un avantage majeur de cette méthode est qu'il n'est pas nécessaire de calculer le modèle cinématique inverse. Dans ce genre d'étude, il est difficile de procéder à une comparaison avec d'autres méthodes parce que chaque travail est caractérisé par le type de robot utilisé, la formulation et la technique de résolution. Cette partie a fait l'objet de deux publications [Menasri et al., 2015, 2013].

Chapitre Quatre

# Planification par interpolation via des métaheuristiques et des courbes polynomiales

## 4.1 Introduction

Dans ce chapitre, nous proposons une méthode d'interpolation pour une classe particulière de problèmes de planification de trajectoires. Cette méthode vise en premier lieu à compléter celle décrite dans le chapitre précédent. En effet, le résultat obtenu avec la méthode décrite dans le chapitre 3 est un ensemble de configurations par lesquelles le robot doit passer. Cependant, avec ce type de résultat, on n'a aucune idée du temps nécessaire pour passer d'une configuration à une autre, du temps total du mouvement et de l'allure des courbes finales. D'où l'objet de ce chapitre qui vise à compléter ce genre de résultat. On propose une formulation pour ce problème sous forme d'un problème d'optimisation sous contraintes. Une métaheuristique à base d'algorithme génétique associé au lagrangien augmenté est utilisée pour la résolution.

Ce chapitre est organisé comme suit : la section 4.2 donne une description détaillée du problème traité dans ce chapitre. La formulation proposée est décrite dans la section 4.3. L'algorithme de résolution est présenté dans la section 4.4. Les résultats de simulation sont présentés dans la section 4.5 et nous concluons le chapitre dans la section 4.6.

## 4.2 Description du problème traité

Comme mentionné précédemment, dans ce chapitre on propose une méthode d'interpolation pour une classe particulière de problèmes de planification de trajectoires. Dans la littérature, de nombreuses techniques ont été développées dans ce contexte [Gasparetto & Zanotto, 2007; Tian & Collins, 2004; Solteiro E. J. et al., 2007]. Le point commun entre toutes ces techniques est que les données initiales doivent être en deux dimensions (position en fonction du temps). Cependant, le résultat fourni par la méthode présentée dans le chapitre 3 est un ensemble de configurations et donc de données en une seule dimension. Ainsi, on ne peut pas appliquer les méthodes classiques d'interpolation directement. Afin de trouver une solution à ce problème, on peut procéder selon deux approches. La première consiste à affecter à chaque position une valeur sur l'axe du temps, et ensuite appliquer les méthodes classiques d'interpolation. La deuxième approche est de résoudre le problème directement. Dans notre cas, on privilégie la deuxième approche parce que tous les critères que l'on va définir sont corrélés entre eux. Par conséquent, on ne peut pas les traiter séparément. Dans les problèmes d'interpolation, généralement on utilise soit des fonctions trigonométriques [Su & Zou, 2012], soit des fonctions polynomiales [Gasparetto & Zanotto, 2007, 2010; Shukla et al., 2013]. L'avantage avec les fonctions trigonométriques est qu'elles sont infiniment dérivables. Par conséquent, le lissage des courbes obtenues est assuré. L'inconvénient est qu'on ne peut pas procéder à des changements sur la nature des données. En effet, un des problèmes majeurs dans les problèmes d'interpolation est la nature des données initiales. Si les nœuds sont trop près l'un de l'autre, l'interpolation devient plus difficile. Dans ce cas, l'utilisation des fonctions polynomiales est plus intéressante puisqu'on peut facilement modifier les données. Dans notre cas, on a opté pour ce choix. Afin d'avoir de bons résultats d'interpolation en robotique, on doit assurer des courbes lisses à la fois au niveau des positions, vitesses et accélérations. Pour garantir ce type de résultat, on utilise des fonctions polynomiales d'ordre 4 comme on peut le voir sur la figure 4.1 :

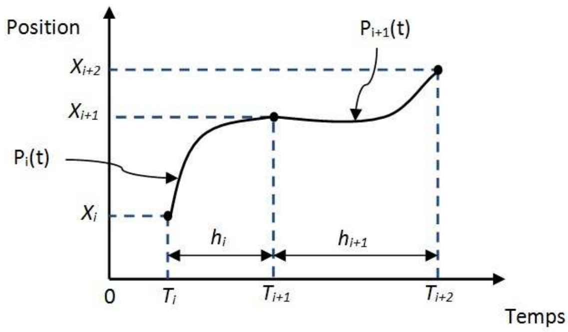



FIGURE 4.1 – Exemple d'utilisation des fonctions polynomiales d'ordre 4.

$P_i(t)$ et $P_{i+1}(t)$ sont des fonctions polynomiales d'ordre 4.

$$P_i(t) = a_0^i + a_1^i \cdot t + a_2^i \cdot t^2 + a_3^i \cdot t^3 + a_4^i \cdot t^4$$
$$P_{i+1}(t) = a_0^{i+1} + a_1^{i+1} \cdot t + a_2^{i+1} \cdot t^2 + a_3^{i+1} \cdot t^3 + a_4^{i+1} \cdot t^4$$

Dans la figure précédente, initialement, on a uniquement les valeurs de $X_i$, $X_{i+1}$ et $X_{i+2}$. Notre objectif est de compléter la trajectoire par interpolation. Cette opération est réalisée en calculant les coefficients de chacune des fonctions $P_i$ et $P_{i+1}$ et en déterminant en même temps les meilleures valeurs des variables de temps $T_i$,$T_{i+1}$ et $T_{i+2}$.

## 4.3 Formulation proposée

Dans cette section, nous proposons une formulation pour notre problème sous forme d'un problème d'optimisation sous contraintes. Beaucoup de critères peuvent être définis dans ce contexte. Cependant, le cas traité étant un cas très général, on ne peut pas utiliser des critères reliés à la cinématique ou à la dynamique du robot. Comme on utilise une fonction polynomiale entre chaque paire de nœuds successifs, on est obligé d'assurer la continuité au niveau de chacun de ces points, ce qui est réalisé dans notre cas par la définition de contraintes. La formulation sera expliquée relativement au cas d'étude présenté illustré par la figure 4.1.

### 4.3.1 Critères

Dans la formulation proposée, on utilise quatre critères. Le premier a pour objectif d'optimiser le temps total du mouvement, de la position initiale à la position finale. Vu que les données initiales sont sur une seule dimension, on cherche les meilleures valeurs sur l'axe du temps, et le fait d'aller plus vite nous permet de minimiser la consommation d'énergie. Son expression est la suivante :

$$F_1(\vec{a}) = \sum_{i=1}^{N-1} h_i^2(\vec{a}) \tag{4.3.1}$$

où

$$h_i = T_{i+1} - T_i$$

$N$ représente le nombre total de nœuds. $h_i$ simule le temps de mouvement entre chaque paire de nœuds successifs (figure 4.1). Le vecteur $\vec{a}$ représente le vecteur des variables de décision (paramètres à trouver), il est composé des coefficients des différentes fonctions polynomiales et des paramètres temps.

Afin de minimiser la longueur des courbes, le critère suivant est utilisé :

$$F_2(\vec{a}) = \sum_{i=1}^{N-1} \dot{q}_i^2(\vec{a}) \tag{4.3.2}$$

$\dot{q}_i$ représente la vitesse angulaire. En effet, pour une fonction $y = g(x)$, la longueur de la courbe est définie par l'équation Eq.4.3.3 et, par conséquent, l'expression simplifiée Eq.4.3.4 est adoptée pour minimiser la longueur de la courbe :

$$\int \left[1 + \left(\frac{dg}{dx}\right)^2\right] dx \tag{4.3.3}$$

$$\int \left(\frac{dg}{dx}\right)^2 dx = \int \dot{g}^2 dx \tag{4.3.4}$$

Relativement à la figure 4.1, $\dot{q}_i$ est la première dérivée de la fonction $P_i$.

Le troisième critère est utilisé pour minimiser les ondulations dans les déplacements du robot, il est exprimé comme suit :

$$F_3(\vec{a}) = \sum_{i=1}^{N-1} \ddot{q}_i^2(\vec{a}) \tag{4.3.5}$$

$\ddot{q}_i$ représente l'accélération angulaire. Avec la représentation de la figure 4.1, $\ddot{q}_i$ est la deuxième dérivée de la fonction $P_i$.

Le dernier critère est utilisé pour minimiser le *jerk* ce qui permet de préserver la structure mécanique du robot en réduisant la résonance des actionneurs. Son expression est :

$$F_4(\vec{a}) = \sum_{i=1}^{N-1} \dddot{q}_i^2(\vec{a}) \tag{4.3.6}$$

$\dddot{q}_i$ représente le *jerk* angulaire. Avec la représentation de la figure 4.1, $\dddot{q}_i$ est la troisième dérivée de la fonction $P_i$.

### 4.3.2 Contraintes

Dans la formulation proposée, les contraintes sont utilisées pour assurer la continuité aux différents nœuds. En effet, comme on utilise une fonction polynomiale entre chaque paire de nœuds, on doit s'assurer que la totalité de la trajectoire ne présente pas de points de discontinuité. Ainsi, relativement à la représentation faite dans la figure 4.1, on définit des contraintes de continuité au niveau des positions, vitesses, accélérations et *jerks*.

$$P_i^{\vec{a}}(T_{i+1}) = P_{i+1}^{\vec{a}}(T_{i+1}) \tag{4.3.7}$$

$$\dot{q}_i^{\vec{a}}(T_{i+1}) = \dot{q}_{i+1}^{\vec{a}}(T_{i+1}) \tag{4.3.8}$$

$$\ddot{q}_i^{\vec{a}}(T_{i+1}) = \ddot{q}_{i+1}^{\vec{a}}(T_{i+1}) \tag{4.3.9}$$

$$\dddot{q}_i^{\vec{a}}(T_{i+1}) = \dddot{q}_{i+1}^{\vec{a}}(T_{i+1}) \tag{4.3.10}$$

$P$, $\dot{q}$, $\ddot{q}$, $\dddot{q}$ représentent les courbes de position, vitesse, accélération et *jerk* respectivement. Pour s'assurer que la courbe représentant la position passe au plus près des différents nœuds, on définit une contrainte supplémentaire pour chaque nœud. Par exemple, pour le nœud avec la valeur $X_i$, on aura :

$$|P_i^{\vec{a}}(T_i) - X_i| \leq R \tag{4.3.11}$$

$R$ est un paramètre dont la valeur doit être choisie très faible afin de passer au plus près des différents nœuds. En plus de ces différentes contraintes, on a aussi les conditions initiales et finales à respecter. Toutes les contraintes précédemment définies sont dupliquées au niveau de chaque nœud.

La fonction objectif globale à minimiser est composée des critères 4.3.1, 4.3.2, 4.3.5, 4.3.6. On divise chacun d'entre eux par sa valeur maximale dans le but de normaliser la fonction objectif. Ainsi, avec ces critères et les différentes contraintes, la formulation est la suivante :

$$\min\ F(\vec{a}) = \alpha \cdot F_1(\vec{a})/T_{max}^2 + \beta \cdot F_2(\vec{a})/V_{max}^2 + \gamma \cdot F_3(\vec{a})/A_{max}^2 + \delta \cdot F_4(\vec{a})/J_{max}^2 \tag{4.3.12}$$

$$\text{Sous les contraintes} \begin{cases} Conditions\ de \text{ continuité} \\ Conditions\ initiales\ et\ finales \end{cases}$$

Où $\alpha$, $\beta$, $\gamma$, $\delta$ sont des paramètres de pondération tels que $\alpha + \beta + \gamma + \delta = 1$. $T_{max}, V_{max}, A_{max}$ et $J_{max}$ représentent le maximum autorisé pour le temps de mouvement, la vitesse , l'accélération et le *jerk* respectivement. Seul $T_{max}$ est choisi arbitrairement pour satisfaire certaines exigences (une limitation dans le temps total du mouvement). Les autres paramètres doivent être choisis relativement aux paramètres physiques du robot.

## 4.4 Résolution

Dans la formulation proposée, on a beaucoup de contraintes à satisfaire, ce qui produit une complexité de résolution importante, qui croît en fonction du nombre de nœuds initiaux. En effet, en augmentant le nombre de nœuds on fait augmenter automatiquement la dimension de l'espace de recherche et le nombre de contraintes. Ainsi, trouver une solution satisfaisante tout en respectant toutes les contraintes est un problème assez difficile. Avec la formulation proposée, le lissage des courbes résultantes est complètement géré par les contraintes. C'est pourquoi on est obligé de satisfaire l'ensemble des contraintes. Pour résoudre ce problème, on combine la méthode du lagrangien augmenté avec un algorithme génétique. L'utilisation du lagrangien augmenté permet d'avoir une meilleure gestion des contraintes et ensuite l'algorithme génétique permet de déterminer la meilleure solution.

### 4.4.1 Le Lagrangien augmenté

Pour gérer les différentes contraintes dans la formulation proposée, on a opté pour la méthode du lagrangien augmenté [Lewis & Torczon, 2002]. Dans les problèmes comportant un nombre assez important de contraintes, ce n'est pas facile pour n'importe quel algorithme de manœuvrer dans les régions admissibles. Cependant, la méthode du Lagrangien augmenté admet des solutions non faisables, mais elle converge progressivement vers des solutions faisables et optimales au fil des itérations, ce qui justifie notre choix [Shukla et al., 2013]. Pour un problème d'optimisation de type :

$$min_x f(x) \tag{4.4.1}$$

sous les contraintes :

$$c_i(x) \leq 0, i = 1...m \tag{4.4.2}$$

$$ceq_i(x) = 0, i = m+1...mt \tag{4.4.3}$$

Le vecteur $c(x)$ représente les contraintes non linéaires d'inégalité, et $ceq(x)$ les contraintes non linéaires d'égalité, $m$ est le nombre total de contraintes inégalité, et $mt$ le nombre total de contraintes. Le problème d'optimisation sous contraintes est transformé en un problème sans contraintes avec la fonction du lagrangien augmenté, comme suit :

$$F(x,\lambda,s,\rho) = f(x) - \sum_{i=1}^{m} \lambda_i.s_i.log(s_i - c_i(x)) + \sum_{i=m+1}^{mt} \lambda_i.ceq_i(x) + \rho/2.\sum_{i=m+1}^{mt} ceq_i(x)^2 \tag{4.4.4}$$

où les composantes $\lambda_i$ du vecteur $\lambda$ sont positives, connues comme étant les estimations des multiplicateurs de Lagrange. Les éléments $s_i$ du vecteur $s$ sont des changements positifs. $\rho$ est le paramètre de pénalité (positif). Ces paramètres sont actualisés à chaque génération. Ainsi pour l'itération $k+1$, les paramètres sont actualisés comme suit :

$$\rho^{k+1} = 10.\rho^k \tag{4.4.5}$$

A chaque itération, le paramètre de pénalité est multiplié par un facteur 10. En général, le choix du facteur dépend du problème à résoudre.

$$\lambda_i^{k+1} = \lambda_i^k + \rho^k.ceq_i^k(x) \tag{4.4.6}$$

Comme on peut le voir dans l'équation 4.4.4, le vecteur $\lambda$, qui représente les estimations des multiplicateurs de Lagrange, est composé de deux parties : la première partie concerne les contraintes d'égalité et la deuxième est dédiée aux contraintes d'inégalité. L'équation 4.4.6 montre l'actualisation pour la partie concernant les contraintes d'égalité. Comme on peut le voir, chaque contrainte d'égalité a son propre coefficient $\lambda_i$ qui est actualisé à chaque itération, en prenant en compte la valeur de la contrainte en question dans la précédente itération.

$$\lambda_i^{k+1} = \lambda_i^k . s_i^k / [s_i^k - c_i^k(x)] \tag{4.4.7}$$

L'équation 4.4.7 montre l'actualisation de la deuxième partie du vecteur $\lambda$, celle concernant les contraintes d'inégalité.

$$s_i^{k+1} = 1/\rho . \lambda_i^{k+1} \tag{4.4.8}$$

Pour calculer les nouvelles valeurs du vecteur $s$, il est nécessaire de mettre à jour le vecteur $\lambda$ et le facteur de pénalité.

Un autre point important dans l'utilisation de la méthode du Lagrangien augmenté est qu'on n'est pas obligé d'avoir une valeur très élevée pour le facteur de pénalité pour assurer la convergence. Il est aussi important de noter qu'il y a plusieurs variantes de cette méthode. Une description complète de cette méthode peut être trouvée dans [Lewis & Torczon, 2002; Conn et al., 1992, 1991].

### 4.4.2 Opérateurs de l'algorithme génétique

En ce qui concerne les opérateurs de l'algorithme, on prend ceux qui sont usuellement utilisés dans la littérature. Cependant, on a modifié l'opérateur de mutation, afin d'accélérer la convergence de l'algorithme. Les différents opérateurs sont cités ci-dessous :

**Opérateur de sélection** : pour cet opérateur, on choisit la sélection par tournoi.

**Opérateur de croisement** : pour cet opérateur, on utilise le croisement à deux points.

**Opérateur de mutation** : en ce qui concerne l'opération de mutation, on utilise la mutation gaussienne, mais avec une adaptation. Cette adaptation vise à accélérer la convergence. Cet opérateur agit sur un seul individu pour créer la progéniture. Chaque individu est codé sous forme de chromosome, ce dernier est composé d'un certain nombre de gènes. L'opérateur de mutation agit sur quelques gènes qui sont sélectionnés selon la probabilité de mutation. La forme générale du calcul de la progéniture à partir des parents est la suivante :

$$x^{'} = x + M$$

M est une variable aléatoire, $x$ représente le parent et $x^{'}$ est la progéniture créée. Pour la valeur de $M$, on prend une distribution gaussienne $M = N(0,\sigma)$, $N(0,\sigma)$ suit une loi normale de moyenne 0 et de variance $\sigma$. L'opérateur de mutation vise à avoir de la diversification dans la recherche, ce qui permet d'explorer tout l'espace de recherche. Cependant, au fil des itérations, il est plus intéressant de réduire l'effet de cet opérateur pour converger plus rapidement. C'est pourquoi on définit un facteur qui sera actualisé à chaque itération comme suit :

$$\text{scale} = \text{scale-scale.génération}_{actuelle}/(\text{maximum de générations})$$

Après cela, on calcule la progéniture comme suit :

$$x' = x + scale.M$$

Le schéma global de l'algorithme utilisé est illustré par la figure 4.2.

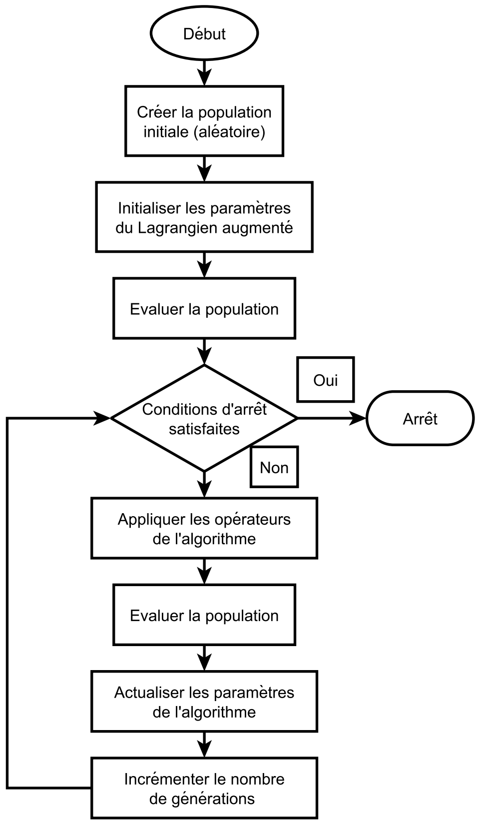



FIGURE 4.2 – Schéma global de l'algorithme

### 4.4.3 Codage de la solution

La dimension de l'espace de recherche dépend du nombre prédéfini de points de contrôle (les nœuds). En effet, la taille du chromosome est directement reliée au nombre de nœuds. Par exemple, pour le cas représenté par la figure 4.1, le codage du chromosome est représenté sur la figure 4.3.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a_0^i$ | $a_1^i$ | $a_2^i$ | $a_3^i$ | $a_4^i$ | $a_0^{i+1}$ | $a_1^{i+1}$ | $a_2^{i+1}$ | $a_3^{i+1}$ | $a_4^{i+1}$ | $T_i$ | $T_{i+1}$ | $T_{i+2}$ |

FIGURE 4.3 – Codage de la solution.

avec $a^i \in R$, $T_i \in R_+^*$.
Ainsi, le chromosome est composé des coefficients de toutes les fonctions polynomiales utilisées ainsi que des valeurs sur l'axe des temps. Pour $N$ nœuds, la taille du chromosome est $6.N - 5$.

## 4.5 Résultats de simulation

Pour tester la formulation proposée, on prend quatre points de contrôle (positions angulaires) avec les valeurs : $-80°$, $-10°$, $70°$, $150°$. Ainsi, le chromosome représentant la solution est composé de 19 gènes. Pour l'algorithme génétique utilisé, on fixe la taille de la population, le taux de mutation, la probabilité de croisement et la taille du tournoi. Le nombre de générations est ajusté pendant les tests. Le tableau suivant résume les choix opérés pour l'ensemble de ces paramètres :

| Paramètres | Valeurs |
|---|---|
| Taille de la population | 50 |
| Nombre de générations | 20 |
| Taux de mutation | 1 |
| Probabilité de croisement | 0,4 |
| Taille du tournoi | 4,0 |

TABLEAU 4.1 – Paramètres de l'algorithme génétique.

| Paramètres | Valeurs |
|---|---|
| $T_{max}$ | $20\ s$ |
| $V_{max}$ | $40\ deg/s$ |
| $A_{max}$ | $40\ deg/s^2$ |
| $J_{max}$ | $25\ deg/s^3$ |

TABLEAU 4.2 – Paramètres choisis pour la formulation.

Dans le tableau 4.2, on indique les valeurs des paramètres utilisés dans la formulation proposée (équation 4.3.12). On utilise le maximum de générations pour arrêter l'exécution de l'algorithme. La population initiale est fixée aléatoirement. Les valeurs sur l'axe du temps et les différents coefficients sont trouvés simultanément. On prend $R = 5.10^{-10}$, et les conditions initiales et finales sont nulles pour la vitesse et l'accélération.

Le premier cas traité a pour objectif de montrer le résultat produit par l'algorithme pour un choix arbitraire des paramètres de pondération de la fonction objectif. Ainsi, on prend : $\alpha = 0{,}3, \beta = 0{,}25, \gamma = 0{,}25, \delta = 0{,}2$. Par ce choix, on donne pratiquement la même pondération pour tous les critères. Le premier résultat est montré dans les figures 4.4, 4.5, 4.6 :

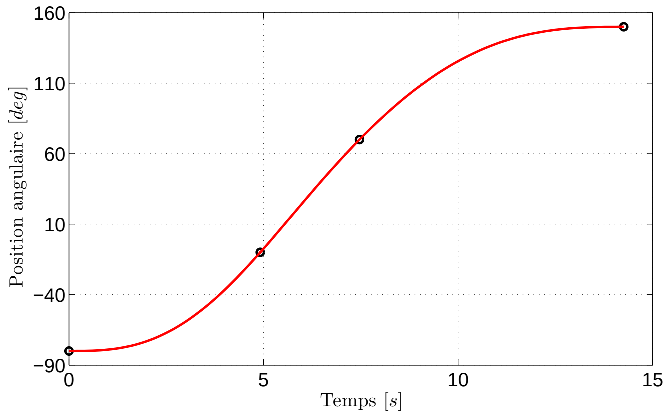



FIGURE 4.4 – Position angulaire.

Au niveau de la position, comme on peut le voir sur la figure 4.4, on obtient un bon résultat en termes de lissage de la trajectoire et la courbe résultante passe par tous les points prédéfinis qui sont représentés par des petits cercles noirs. Aussi, on peut remarquer que le temps total du mouvement est inférieur à 15 secondes (inférieur à $T_{max}$).

La figure 4.5 présente le résultat obtenu au niveau de la vitesse :

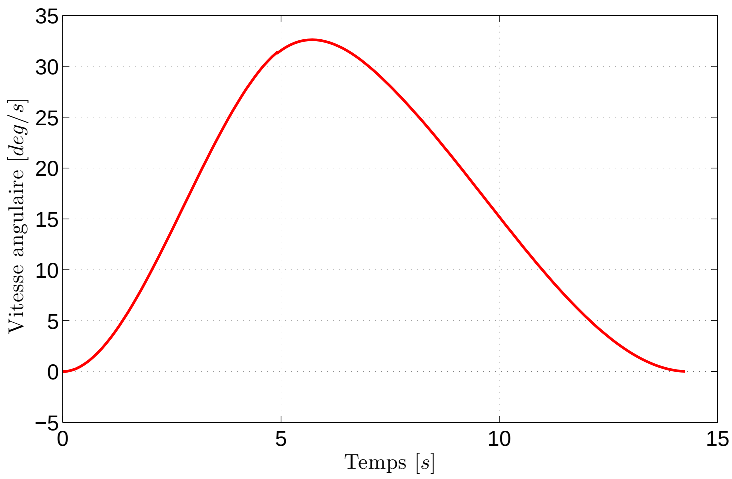



FIGURE 4.5 – Vitesse angulaire.

A ce niveau, on obtient aussi une courbe lisse. Les conditions initiales et finales sont également respectées. Sur la figure 4.5, on peut remarquer que le maximum de vitesse est proche de 35

$deg/s$ (inférieur à $V_{max}$).

La figure 4.6 montre le résultat au niveau de l'accélération :

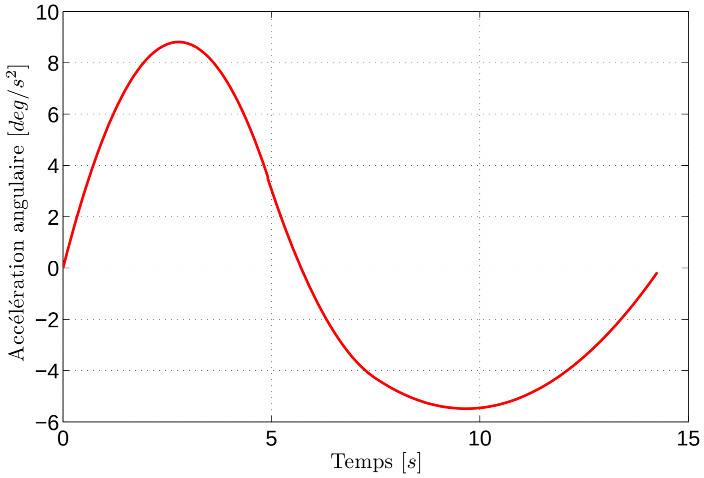



FIGURE 4.6 – Accélération angulaire.

Dans le cas de l'accélération, on obtient aussi un résultat tout à fait satisfaisant. En effet, on a une courbe lisse qui respecte les conditions initiales et finales. On peut aussi remarquer que le maximum d'accélération est proche de 10 $deg/s^2$ qui est inférieur à $A_{max}$ (défini dans le tableau 4.2).

Pour les résultats présentés précédemment, l'algorithme a été exécuté une seule fois. Donc, à ce stade, on ne peut pas conclure sur les résultats obtenus. L'algorithme étant stochastique, il est en effet nécessaire de le réexécuter plusieurs fois afin de mettre en évidence son efficacité et sa capacité à reproduire les résultats. Ainsi, on exécute l'algorithme 20 fois. Le premier résultat est illustré par la figure 4.7, il représente la variation de la fonction objectif :

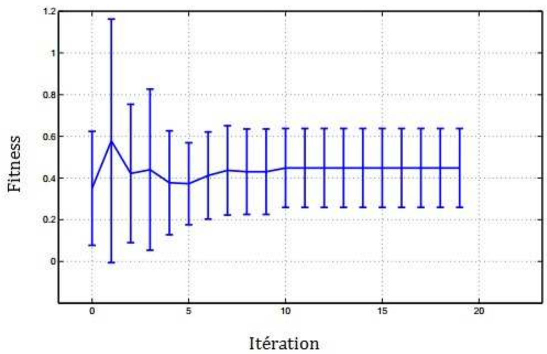



FIGURE 4.7 – Moyenne et écart-type de la fonction objectif.

Comme on peut le constater sur la figure 4.7, l'algorithme converge vers la même région. Aux dernières générations, on obtient une moyenne de 0,4484 et un écart-type de 0,3784. On peut noter que l'écart-type est grand. Ceci est dû au réglage des paramètres de l'algorithme génétique et probablement à la présence de minima locaux. Donc, il est nécessaire d'ajuster la valeur des paramètres afin d'améliorer ce résultat en réduisant l'écart-type.

Les figures suivantes illustrent les résultats concernant les maxima de vitesse, accélération, *jerk* et le temps :

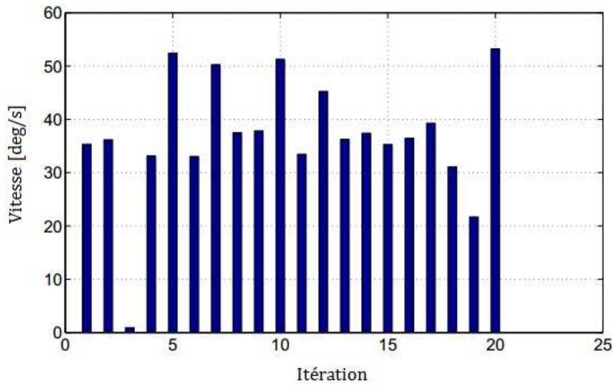



FIGURE 4.8 – Maximum de vitesses.

Comme on peut le voir dans la figure 4.8, on obtient un résultat satisfaisant tout en respectant la valeur prédéfinie pour le maximum de vitesse (dans notre cas 40 $deg/s$) dans la majorité des tests. Cependant, les paramètres de l'algorithme doivent être réajustés. La figure 4.9 présente le résultat concernant le maximum et le minimum des accélérations :

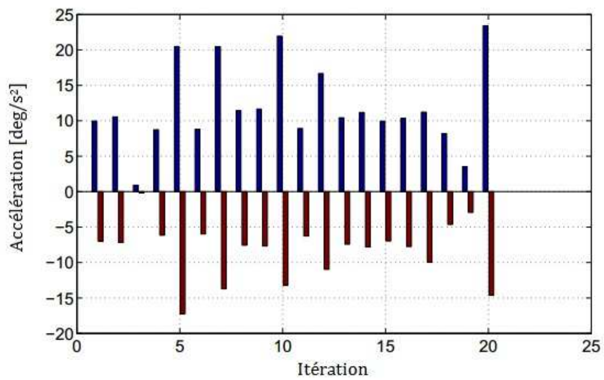



FIGURE 4.9 – Maximum et minimum des accélérations.

Comme le met en évidence la figure 4.9, au niveau des accélérations, tous les tests sont satisfaisants et la valeur prédéfinie pour le maximum d'accélération $A_{max}$ (40 $deg/s^2$) est respectée. La figure 4.10 représente le résultat obtenu pour le maximum de *jerk*. Sur cette figure, il y a un seul test où la valeur prédéfinie pour le maximum de *jerk* n'est pas respectée. Donc, on peut conclure que le résultat est en général satisfaisant à ce niveau. La figure 4.11 présente le résultat pour le temps total du mouvement.

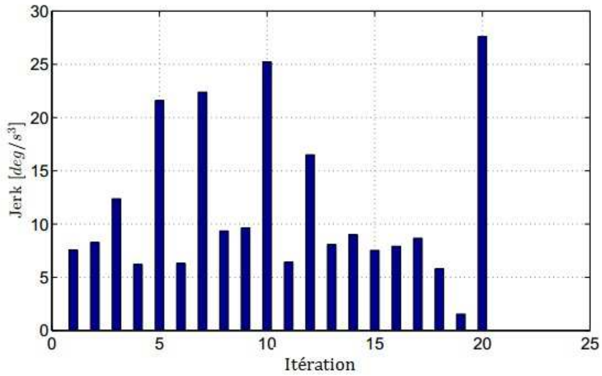



FIGURE 4.10 – Maximum de *jerk*.

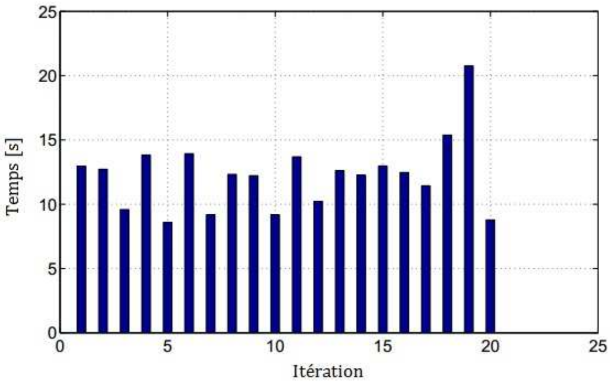



FIGURE 4.11 – Maximum du temps de mouvement.

Dans le cas du temps total du mouvement, pour la majorité des tests, la valeur prédéfinie de $T_{max}$ est respectée. En effet, comme on peut le remarquer, il y a un seul test où la valeur n'est pas respectée. Un autre point important à vérifier est le temps CPU consommé par l'algorithme pour fournir un résultat, ce qui est illustré par la figure 4.12. Pour la majorité des tests, le temps CPU nécessaire est inférieur à $6s$, ce qui est acceptable dans notre cas. Ce point montre l'efficacité de l'algorithme génétique.

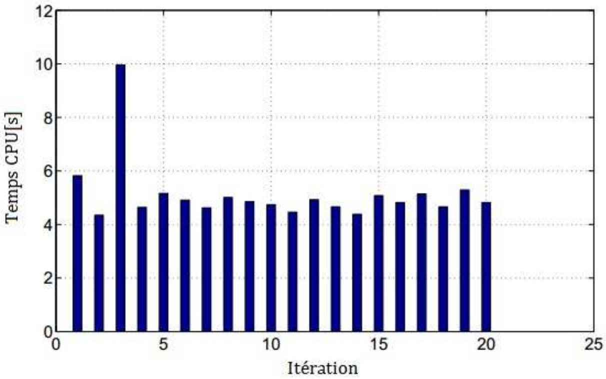



FIGURE 4.12 – Temps CPU.

A ce stade, on peut conclure que le choix des paramètres de pondération pour la fonction objectif, où on donne la même importance pour tous les critères, et des paramètres de l'algorithme génétique est acceptable. En effet, nous l'avons montré dans les figures précédentes, ce choix nous a permis d'obtenir des résultats satisfaisants, en termes de lissage des courbes de position, vitesse et accélération. En exécutant l'algorithme plusieurs fois, on a montré qu'il converge vers la même région, dans la majorité des cas, les conditions aux limites sont respectées (maximum de vitesse, accélération, ...). Un autre point important est que le temps CPU nécessaire est acceptable dans notre cas.

Comme mentionné précédemment, le premier cas traité est réalisé pour un choix arbitraire des paramètres de pondération. Cependant, il est nécessaire d'évaluer le comportement de l'algorithme quand ces paramètres changent. Le but de ce deuxième cas d'étude est d'explorer ce point. Ainsi, on définit 20 tests dans lesquels on change les paramètres de pondération en gardant les mêmes valeurs pour les paramètres de l'algorithme génétique. Pour chaque test, on exécute l'algorithme 20 fois et on calcule la moyenne et l'écart-type pour certaines variables. La figure suivante montre l'ensemble des tests réalisés :

| α=0.1 | | | | α=0.3 | | | α=0.5 | | α=0.7 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| β=0.1 | β=0.3 | β=0.5 | β=0.7 | β=0.1 | β=0.3 | β=0.5 | β=0.1 | β=0.3 | β=0.1 | | |
| | | | Test 10 | | | Test 16 | | Test 19 | Test 20 | δ=0.1 | γ=0.1 |
| | | Test 8 | | | Test 14 | | Test 17 | | | δ=0.3 | |
| | Test 5 | | | Test 11 | | | | | | δ=0.5 | |
| Test 1 | | | | | | | | | | δ=0.7 | |
| | | Test 9 | | | Test 15 | | Test 18 | | | δ=0.1 | γ=0.3 |
| | Test 6 | | | Test 12 | | | | | | δ=0.3 | |
| Test 2 | | | | | | | | | | δ=0.5 | |
| | Test 7 | | | Test 13 | | | | | | δ=0.1 | γ=0.5 |
| Test 3 | | | | | | | | | | δ=0.3 | |
| Test 4 | | | | | | | | | | δ=0.1 | γ=0.7 |

FIGURE 4.13 – Les tests réalisés.

Comme on peut le voir sur la figure 4.13, chaque test correspond à un choix des paramètres de pondération. Par exemple, le test 15 correspond au choix ($\alpha = 0{,}3$, $\beta = 0{,}3$, $\gamma = 0{,}3$, $\delta = 0{,}1$). Pour chacun de ces tests, on garde toujours $\alpha + \beta + \gamma + \delta = 1$. Les figures suivantes illustrent les résultats obtenus :

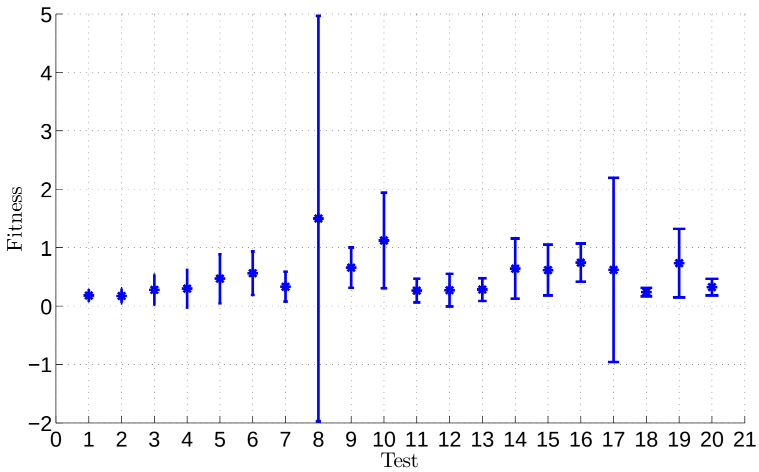
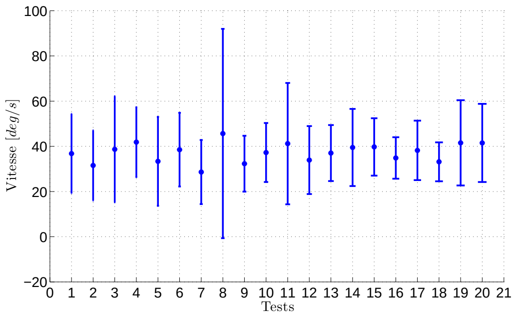


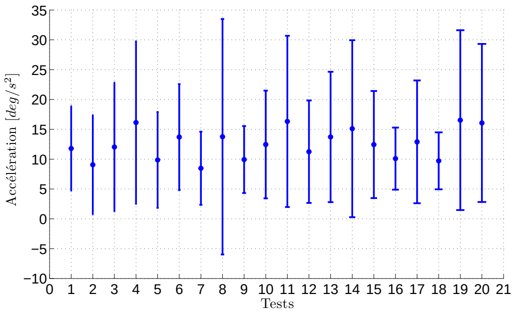

FIGURE 4.14 – Moyenne et écart-type de la fonction objectif à la convergence.

La figure 4.14 représente la moyenne et l'écart-type pour chacun des tests réalisés (résultats obtenus à la convergence), on peut conclure qu'on obtient des bons et des mauvais résultats en termes de convergence. En effet, dans le test 8 ($\alpha = 0{,}1$, $\beta = 0{,}5$, $\gamma = 0{,}1$, $\delta = 0{,}3$), l'écart-type est très grand comparativement à ceux obtenus avec les autres tests. Avec le test 18 ($\alpha = 0{,}5$,

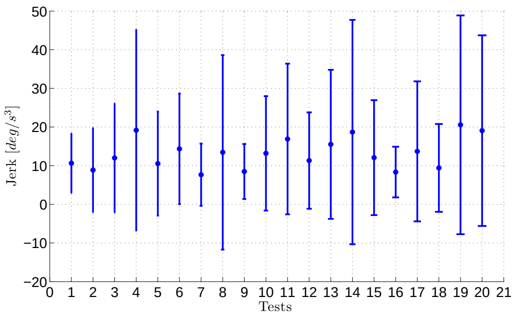
$\beta = 0{,}1$, $\gamma = 0{,}3$, $\delta = 0{,}1$), on obtient le meilleur résultat. En effet, on obtient le minimum des valeurs pour la moyenne et l'écart-type. Ainsi, toute la population converge vers le même minimum. Les figures qui suivent montrent les résultats pour la vitesse, l'accélération, le *jerk* et le temps de mouvement total nécessaire :



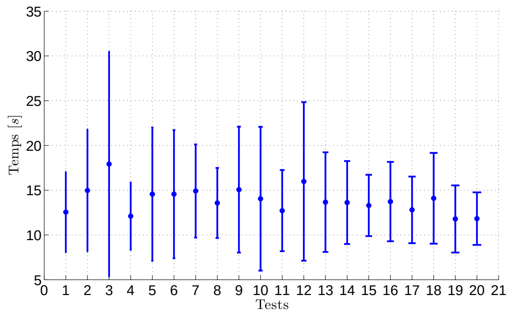

(a) Maximum de vitesse.




(b) Maximum d'accélération.




(c) Maximum de *jerk*.




(d) Maximum de temps.

FIGURE 4.15 – Résultats pour la vitesse, accélération, *jerk* et temps.

La figure 4.15 représente la moyenne et l'écart-type pour les maxima de vitesse, accélération, *jerk* et de temps total nécessaire pour le mouvement. Le but est de pouvoir conclure sur le respect des valeurs prédéfinies pour chacune de ces quantités ($V_{max}$, $A_{max}$, $J_{max}$, $T_{max}$). En termes de vitesse (figure 4.15(a)), le test 8, qui produit le plus grand écart-type, est aussi le test le moins bon. Le test 18 reste le meilleur des tests, parce qu'une fois de plus il produit le plus petit écart-type.

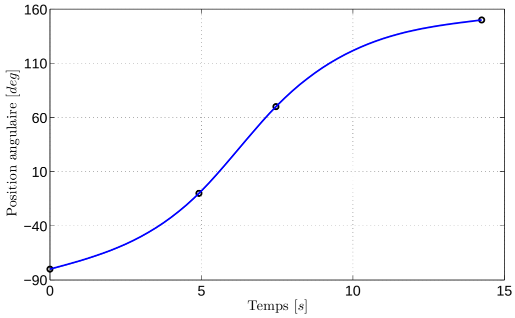
En termes de maximum d'accélération (figure 4.15(b)), on n'a pas de problème avec le respect des valeurs prédéfinies. En effet, dans tous les tests réalisés, cette valeur est respectée. En conclusion, le test 18 est le test le plus performant car il produit le plus petit écart-type.

En ce qui concerne le maximum de *jerk* (figure 4.15(c)), avec certains choix des paramètres, les résultats obtenus ne respectent pas le maximum prédéfini $J_{max}$. Pour les autres tests, comme les tests 7, 9, 16, 18,.., on obtient un bon résultat.

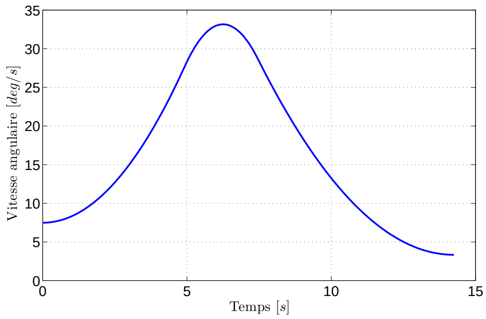
Le dernier résultat concerne le temps total du mouvement (figure 4.15(d)). Pour ce paramètre, la majorité des résultats obtenus sont acceptables. Le test 3 est le moins bon et le test 20 est le plus performant.

Avec les résultats présentés précédemment, on ne peut pas analyser l'impact de chaque paramètre de pondération indépendamment des autres parce qu'ils sont interdépendants. On peut aussi noter que, pour ce choix des paramètres de l'algorithme génétique, certaines configurations conduisent à des résultats performants (test 7, test 18). Ces résultats sont des compromis entre les différents critères définis tout en respectant les conditions initiales et finales ainsi que les limitations pour la vitesse, accélération, le *jerk* et le temps de mouvement.

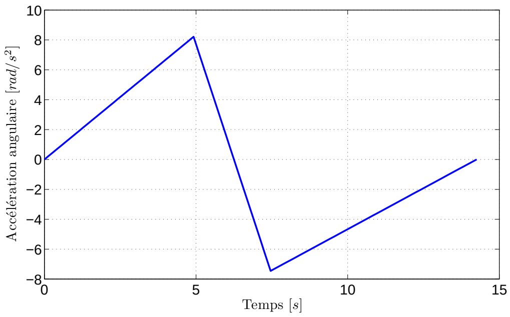
A notre connaissance, on ne trouve pas de méthode d'interpolation dans la littérature qui accepte des données initiales seulement en une seule dimension. Ainsi, pour comparer nos résultats à ceux des autres méthodes, il est nécessaire d'exécuter d'abord notre algorithme afin d'affecter à chaque position une valeur sur l'axe des temps. Ensuite, il est aisé d'extraire les coordonnées à deux dimensions des nœuds de contrôle (nœuds initiaux), qui serviront de paramètres d'entrée aux méthodes d'interpolation concurrentes. Par exemple, avec les nœuds initiaux à deux dimensions $(0,-80°)$, $(4{,}92,-10°)$, $(7{,}46,70°)$, $(14{,}25,150°)$, extraits des résultats de la première étude (figures 4.4, 4.5, 4.6), on a appliqué la méthode des splines cubiques pour effectuer l'interpolation entre les nœuds successifs. Les résultats obtenus sont illustrés par la figure 4.16. Comme on peut le voir sur cette figure, la continuité aux différents nœuds est assurée pour les courbes de position, vitesse et accélération. Cependant, on peut remarquer quelques limitations dans l'utilisation des splines cubiques. En effet, pour la position, la courbure est plus importante avec le résultat produit par notre méthode (figure 4.4) qu'avec celui obtenu par les splines (figure 4.16(a)). Ceci est l'effet du degré des fonctions polynomiales choisies. Pour les vitesses (figure 4.16(b)), on peut constater que les conditions initiales et finales n'ont pas été respectées par les splines. Au niveau des accélérations (figure 4.16(c)), le lissage de la courbe n'a pas été non plus assuré par les splines. Ceci est dû au fait que les splines cubiques utilisent des polynômes de degré 3. Cependant, les conditions initiales et finales ont bien été respectées à ce niveau.




(a) Position angulaire.




(b) Vitesse angulaire.




(c) Accélération angulaire.

FIGURE 4.16 – Résultats avec les splines cubiques.

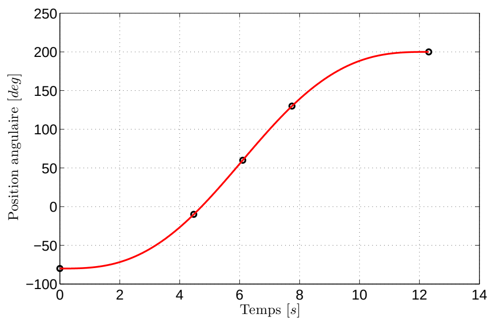
On peut conclure que l'utilisation des splines cubiques est très limitée pour assurer des courbes lisses et leur problème majeur est l'intégration d'autres contraintes (comme l'évitement d'obstacles). Le tableau 4.3 montre quelques caractéristiques d'autres méthodes de planification de trajectoires (basées sur des opérations d'interpolation) :

| Méthodes | Type de données | Modifier les données | Conditions limites | Lissage |
|---|---|---|---|---|
| Méthode proposée | une dimension | possible | assurées | assuré |
| Spline cubique | deux dimensions | possible | aux accélérations | position,vitesse |
| Méthode 3 | deux dimensions | impossible | aux accélérations | assuré |
| Méthode 4 | deux dimensions | impossible | assurées | assuré |

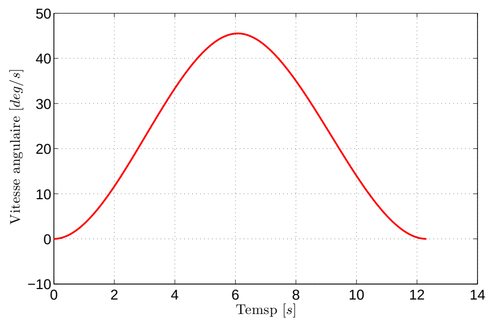
TABLEAU 4.3 – Caractéristiques de quelques méthodes.

La méthode 3 correspond à la technique proposée dans [Su & Zou, 2012], la méthode 4 à celle présentée dans [Gasparetto & Zanotto, 2007]. Comme on peut le constater dans le tableau 4.3, la différence principale entre la méthode proposée et les autres techniques citées est dans les données initiales. En effet, comme déjà mentionné, dans notre cas, les données initiales sont sur une seule dimension. Ainsi, on a l'avantage de pouvoir minimiser le temps total du mouvement. Avec notre méthode et avec les splines, on peut facilement modifier les données, par exemple pour éloigner les nœuds les uns des autres et faciliter ainsi l'interpolation. Ceci est possible parce que les deux méthodes se basent sur l'utilisation de polynômes. Par contre, avec les méthodes proposées dans [Su & Zou, 2012] (basée sur des fonctions trigonométriques) et [Gasparetto & Zanotto, 2007] (basée sur les B-splines), on ne peut pas modifier les données. Avec notre méthode et celle proposée dans [Gasparetto & Zanotto, 2007], on peut facilement ajouter d'autres conditions à la formulation, ce qui leur donne une meilleure flexibilité. En ce qui concerne le lissage des courbes résultantes, il y a seulement la méthode des splines cubiques qui n'assure pas le lissage au niveau des accélérations.

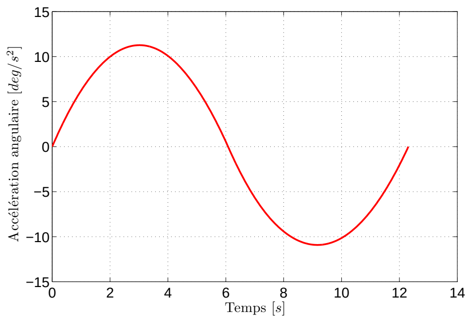
A partir des résultats présentés dans le tableau 4.3, on peut dire que la méthode proposée possède beaucoup d'avantages par rapport aux méthodes concurrentes de la littérature. Il est important de noter que la formulation proposée n'est pas limitée à 4 nœuds. En effet, avec plus de 4 nœuds, on doit juste dupliquer les contraintes de continuité à chacun de ces nœuds et ajouter une fonction polynomiale pour chaque nœud ajouté. Par exemple, pour 5 nœuds ($-80°$, $-10°$, $70°$, $150°$, $200°$) et les mêmes paramètres de pondération ($\alpha = 0{,}3, \beta = 0{,}25, \gamma = 0{,}25$, $\delta = 0{,}2$), on obtient les résultats suivants :




(a) Position angulaire.




(b) Vitesse angulaire.




(c) Accélération angulaire.

FIGURE 4.17 – Résultats pour 5 nœuds.

Comme on peut le constater sur la figure 4.17, on obtient aussi de bons résultats avec un lissage des courbes à tous les niveaux.

## 4.6 Conclusion

Dans ce chapitre, on a présenté une méthode qui traite un aspect important des problèmes de planification de trajectoires qui est l'interpolation. La méthode proposée complète ainsi les résultats décrits au chapitre précédent. Nous avons proposé une formulation pour ce problème sous forme d'un problème d'optimisation sous contraintes. Pour le traitement de ces dernières, on a tiré profit de la méthode du Lagrangien augmenté, qui est très efficace pour les problèmes d'optimisation possédant un nombre important de contraintes. La formulation est basée sur l'utilisation de fonctions polynomiales d'ordre 4 qui permettent d'assurer des courbes lisses à la fois en position, vitesse et accélération. La formulation proposée est très flexible. En effet, on peut intégrer très facilement de nouvelles contraintes comme l'évitement d'obstacles. La complexité dans la formulation proposée tient au fait que l'on cherche à trouver simultanément les coefficients des différents polynômes et leurs projections sur l'axe du temps. Cette complexité croît avec le nombre de points de contrôle utilisés. Les résultats de simulation ont montré sans ambiguïté l'efficacité de la méthode proposée. En effet, comme on a pu le vérifier, elle produit des résultats satisfaisants avec un nombre réduit de générations. On a aussi illustré l'effet des paramètres de pondération sur les résultats et on a indiqué comment les choisir pour assurer la convergence et reproduire ainsi les résultats. Cette partie a fait l'objet de deux publications [Menasri et al., 2014a,b].

Les chapitres 3 et 4 ont pour objectif de développer une méthode de planification de trajectoire tout en assurant le lissage des courbes résultantes. Cette technique peut être appliquée dans la réalisation de la procédure de craniotomie. En effet, cette procédure peut être réalisée en deux étapes. La première consiste à ramener le robot et à le positionner sur la tête du patient et la deuxième est la réalisation de l'ouverture, ce qui fait l'objet du chapitre 5.

Chapitre Cinq

# Craniotomie robotisée avec le système Neuromate

## 5.1 Introduction

Ce chapitre est consacré à l'opération de craniotomie réalisée avec le système Neuromate. Elle a pour objectif de faire une ouverture au niveau du crâne humain et représente la première étape dans le traitement de plusieurs maladies affectant le cerveau. Actuellement, est dans la plupart des cas, elle est réalisée à main levée (par le chirurgien en utilisant une perceuse). Cette façon de faire engendre deux problèmes majeurs : la durée importante de l'opération et le manque de précision. Nous proposons dans ce chapitre la réalisation de cette opération en utilisant un robot Neuromate et un moteur à grande vitesse. La procédure consiste à appliquer un modèle d'usinage à grande vitesse en utilisant des plaques en polyamide dont les caractéristiques mécaniques sont proches de celles du crâne humain. Cette partie a été réalisée au sein du laboratoire IBM (Institut de Biomécanique Humaine) de l'ENSAM, sous la direction du professeur Philippe Decq et la collaboration de Amal Benslimane, dans le cadre de son projet de master [Benslimane, 2015].

Ce chapitre est organisé comme suit. Dans la section 5.2, on fait un exposé très général de l'anatomie du crâne humain et ses propriétés. Dans la section 5.3, on présente l'évolution des différentes techniques utilisées pour la réalisation de la craniotomie. Une description du matériel utilisé est présentée dans la section 5.4. La démarche de résolution suivie est décrite dans la section 5.5, et les résultats obtenus dans la section 5.6. Nous concluons ce chapitre dans la section 5.7.

## 5.2 Anatomie et propriétés du crâne humain

### 5.2.1 Anatomie du crâne

Le crâne humain est la partie supérieure du squelette, il est essentiellement destiné à protéger le cerveau. Il repose sur le rachis cervical par l'intermédiaire de l'atlas ou première vertèbre cervicale, maintient en antérieur le massif facial. L'ensemble composé par la tête et le crâne représente environ un huitième du poids du corps et est la partie la plus solide. Le crâne est constitué de plusieurs os qui peuvent être plats ou convexes. Ces derniers sont reliés entre eux par des dentelures profondes appelées sutures.

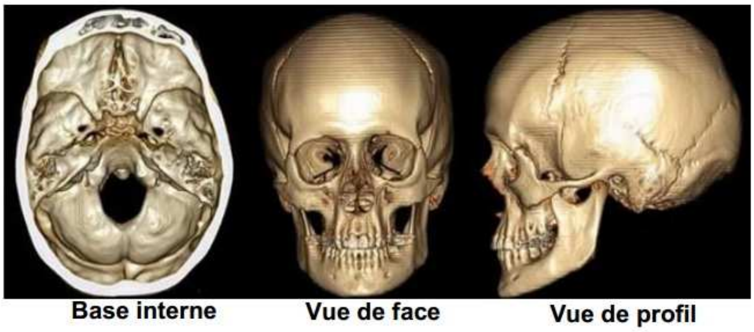



FIGURE 5.1 – Le crâne humain.

Le crâne est composé de deux parties : le neurocrâne (boite crânienne) et le viscérocrâne (massif facial) comme on peut le voir sur la figure 5.2.

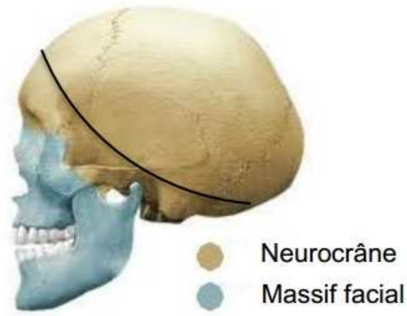



FIGURE 5.2 – Neurocrâne et massif facial.

La boîte crânienne comprend aussi deux parties :

— **La base** : appelée aussi le plancher, elle est composée de trois fosses crâniennes :

1. la fosse crânienne antérieure ;
2. la fosse crânienne moyenne ;
3. la fosse crânienne supérieure.

— **La voûte** : elle représente la partie supérieure de la boîte crânienne, elle est formée de plaques osseuses, soudées entre elles par des sutures interdigitées extrêmement solides. C'est la partie qui nous intéresse, parce que c'est à ce niveau que la craniotomie sera réalisée.

Au-dessous de l'os du crâne se trouve la dure-mère : une membrane dure et rigide qui protège le cerveau et la moelle épinière (figure 5.3). Elle est fibreuse et adhère à l'os. Lors de la procédure de craniotomie, cette couche ne doit absolument pas être transpercée, de risque que des débris d'os infiltrent le cerveau et causent des dommages à celui-ci.

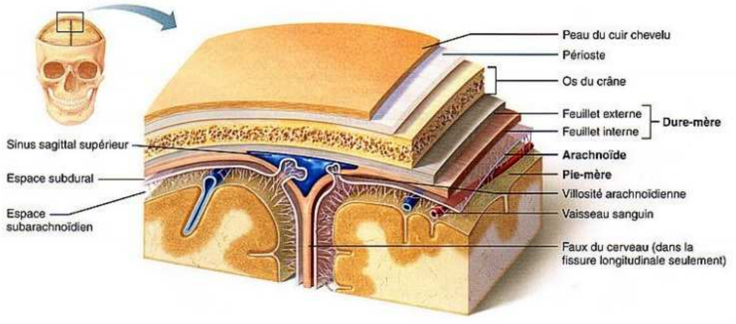



FIGURE 5.3 – Coupe du crâne.

L'os de la voute crânienne est composé de trois couches comme on peut le voir sur la figure 5.4 : deux couches d'os compact avec à l'intérieur une couche d'os spongieux nommé diploé. L'épaisseur de ce dernier varie selon la partie du crâne et aussi d'une personne à l'autre avec une moyenne de $8mm$. L'os compact garde la même épaisseur.

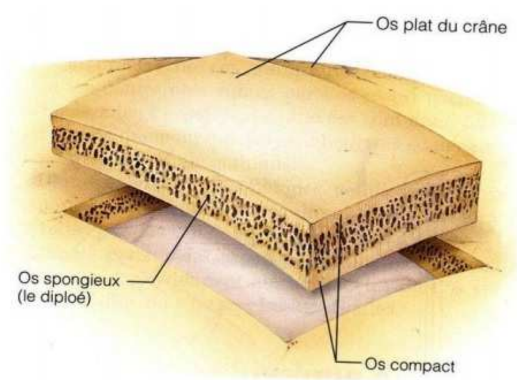



FIGURE 5.4 – Coupe de l'os plat du crâne.

### 5.2.2 Propriétés du crâne

L'os du crâne possède énormément de propriétés, on peut citer essentiellement :

- **Densité** : le tissu osseux est un matériau de densité variable. La porosité de l'os cortical varie entre 5% et10%. L'os spongieux, quant à lui, possède une porosité beaucoup plus élevée, elle varie entre 50% et 95%.
- **Propriété mécanique** : la différence de densité entre les deux types d'os entraîne une variation dans leurs propriétés mécaniques. Le tableau 5.1 récapitule l'ensemble des ces propriétés.

— **Isotropie** : en ce qui concerne les propriétés mécaniques des différents points du crâne, une légère variation a été remarquée selon les directions radiale et transversale, ce qui permet de considérer l'os cortical en particulier comme étant transversalement isotrope.

— **Le coefficient de frottement** : le coefficient de frottement caractérise le contact métal/os. Ce point a fait l'objet de plusieurs travaux de recherche [Tetsuya et al., 1995]. En translation avec l'os et en considérant que le métal ne dépasse pas une vitesse de 3 $mm/s$, le coefficient de frottement os/métal est alors égal en moyenne à 0.5.

— **Effet de la température** : l'os est un mauvais conducteur thermique, la chaleur induite par le perçage ne peut pas être dissipée facilement ce qui peut causer des problèmes significatifs à l'os. Si la température de l'os dépasse $47degrs$ durant une minute, une nécrose thermique est inévitable. Pendant l'opération de perçage de l'os, sa température dépasse largement le seuil toléré, mais en augmentant les vitesses de coupes, on diminue efficacement le temps de forage. De plus l'irrigation peut diminuer significativement la température, et favoriser également le dégagement des copeaux.

## 5.3 Aperçu de la procédure de craniotomie

### 5.3.1 Craniotomie conventionnelle

La craniotomie est une procédure qui a pour objectif la réalisation d'une petite ouverture au niveau du crâne. Cette procédure a connu beaucoup de progrès. Les premières techniques remontent à l'âge de la pierre polie (9000 ans av. J.C), ce qui est qualifié de craniotomie conventionnelle qui est faite à main levée. Elle a été décrite depuis le XVIème siècle par Ambroise Pare, en particulier. Cette intervention est entrée dans l'ère moderne avec l'apparition de la neurochirurgie. Elle peut être décrite en plusieurs étapes successives [Zhang et al., 2001] :

— la mise sous anesthésie générale du patient ;

— la fixation de la tête de façon rigide par le biais d'un dispositif (une têtière) de telle sorte qu'elle soit maintenue immobile tout au long de la procédure (figure 5.5(a)) ;

— la réalisation d'une incision de la peau derrière la racine des cheveux. Cette dernière est ensuite détachée de l'os et rabattue afin d'exposer la voûte du crâne ;

— la réalisation d'une ouverture au niveau de la dure-mère à l'aide de ciseaux chirurgicaux.

A la fin de l'intervention neurochirurgicale, la dure-mère est fermée avec des sutures, les débris osseux sont regroupés, collés de façon à refermer l'orifice créé, ensuite les muscles et la peau sont suturées ensemble.

Les interventions auxquelles nous nous intéressons nécessitent seulement la réalisation d'ori-

| Partie de l'os | Propriétés | Moyenne |
|---|---|---|
| **Propriétés communes** | Age (ans) | 57 |
| | Densité ($Kg/m^3$) | 1446 |
| | Épaisseur du crâne (mm) | 7 |
| | Contrainte à la $1^{re}$ rupture (MPa) | 71,5 |
| | Contrainte à la $2^{me}$ rupture (MPa) | 110,6 |
| **Os compact** | Limite élastique (MPa) | 41,8 |
| | Module de Young (GPa) | 8,75 |
| | Module tangent (GPa) | 4,62 |
| | Module de cisaillement (GPa) | 3,47 |
| | Module de compression (GPa) | 7,12 |
| | Coefficient de poisson | 0,25 |
| | Déformation à la limite élastique (%) | 0,63 |
| | Déformation maximale (%) | 1,61 |
| | **Résistance au cisaillement (MPa)** | 90 |
| **Os spongieux** | Épaisseur du diploé (mm) | 2,89 |
| | Limite élastique (MPa) | 13,6 |
| | Module de Young (GPa) | 4,66 |
| | Module tangent (GPa) | 0,17 |
| | Module de cisaillement (GPa) | 1,85 |
| | Module de compression (GPa) | 3,47 |
| | Coefficient de poisson | 0,25 |
| | Déformation à la limite élastique (%) | 0,38 |
| | Déformation maximale (%) | 0,95 |
| | **Résistance au cisaillement (MPa)** | 3 |

TABLEAU 5.1 – Tableau récapitulatif des propriétés mécaniques de l'os du crâne

fices de 2 à $18mm$. Ceci peut être accompli à l'aide d'un trépan (figure 5.5(b)).

### 5.3.2 Craniotomie stéréotaxique

Avec les méthodes conventionnelles, il est clair qu'il y a un manque de précision dans la détermination du point de l'incision. Avec l'apparition des scanners et l'utilisation des images IRM, d'autres outils ont été développés afin d'avoir une meilleure précision, d'où l'utilisation des cardes stéréotaxiques (figure 5.6). Cette technique, nommée stéréotaxie, s'appuie sur l'utilisation d'un système fixé de façon rigide au crâne. Ce dernier permet de définir un espace normé doté d'une référence origine. Une fois la tête équipée avec ce cadre, elle est imagée permettant ainsi de repérer l'ensemble de l'espace tête et son contenu. Ainsi, on peut obtenir la position de la cible et les trajectoires nécessaires pour l'atteindre [Pandey & Panda, 2013].


(a) Préparation du patient. (b) Exposition de la dure-mère.

Figure 5.5 – Étapes de réalisation d'une craniotomie conventionnelle.

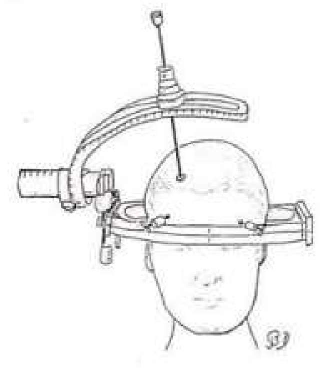


Figure 5.6 – Cadre stéréotaxique.

### 5.3.3 Craniotomie robotisée

Avec l'utilisation des cadres stéréotaxiques et l'imagerie, on arrive à déterminer avec beaucoup de précision la cible où l'incision doit être faite. En effet, avec le robot **Neuromate**, qui est un robot de guidage, on a une grande précision. Cependant, une fois la cible déterminée, la procédure en elle-même est réalisée à main levée par le chirurgien avec une simple perceuse. Ceci peut induire de l'imprécision et conduire à un allongement du temps de la procédure. De ce fait, des chercheurs se sont intéressés à cette procédure en essayant de la rendre complètement automatique à l'aide des robots.

Réaliser une craniotomie robotisée dépend beaucoup plus du robot lui même. Dans la littérature et avec le système **Neuromate**, on trouve une seule réalisation [Liu et al., 2007]. L'idée principale est d'utiliser le principe de la co-manipulation. Par conséquent, le chirurgien fournit des efforts qui sont intégrés dans la boucle de commande du robot.

## 5.4 Matériels utilisés

### 5.4.1 Le système Neuromate

De conception française, le robot **Neuromate** est un robot utilisé en neurochirurgie. Il a été mis au point par la société Renishaw Mayfield. Le premier modèle a vu le jour en 1989 (figure 5.7). Ce robot est conçu pour assister le chirurgien lors des opérations au niveau du cerveau, ceci avec plus de sécurité, précision, répétabilité et fiabilité, sans pour autant modifier le protocole opératoire classique du neurochirurgien. Ainsi, ce robot joue la fonction de guidage.

FIGURE 5.7 – Le robot Neuromate.

Le système **Neuromate** est un système stéréotaxique robotisé. Il est composé d'un bras robotisé auquel est associé un logiciel informatique de neuro-navigation. Il peut être guidé par l'image et piloté par ordinateur. Le robot est un bras manipulateur à cinq axes permettant ainsi le positionnement spatial et l'orientation d'un porte-outil se trouvant à l'extrémité de l'effecteur du robot. Le chirurgien peut ensuite insérer ses instruments classiques de stéréotaxie et procéder à l'intervention. L'outil fixé à l'extrémité de l'effecteur du robot peut être déplacé par le robot avec un joystick selon une trajectoire préalablement définie en fonction des impératifs de l'intervention. Une modélisation géométrique et cinématique du robot est présentée en annexe.

La position et l'orientation du porte-outil sont manipulées par le chirurgien en utilisant des images pré-opératoires du crâne du patient. Cette opération est réalisée à l'aide du logiciel de planification de trajectoire **Voxim**, qui est un logiciel d'imagerie interactif et de planification installé sur un PC lié au robot. Avec ce logiciel, on peut traiter les images du crâne prises préalablement et les présenter en 3D. Avec de différentes vues du crâne, on peut facilement planifier la trajectoire voulue.

Après la détermination des trajectoires et des cibles, une mise en correspondance entre le robot et le PC est réalisée. Cette dernière opération permet de fixer les différents repères. Ainsi, le robot disposera des informations nécessaires concernant la position de la tête du patient. Le robot lui-même est aussi équipé de son propre PC lui permettant de gérer les différents modules et la communication avec le PC de navigation. Il est aussi équipé d'un contrôleur afin d'exécuter la trajectoire planifiée.

Le robot est ensuite actionné par le chirurgien par le biais d'un joystick lui permettant de se positionner avec beaucoup de précision en tenant compte de la trajectoire définie. Le robot peut fonctionner selon deux modes : le mode endoscopique et le mode non endoscopique. Avec le mode endoscopique, le robot effectue des mouvements pilotés par le chirurgien et limités par des frontières de sécurité préalablement définies sur **Voxim** à des vitesses lentes. En mode endoscopique, l'effecteur du robot suit une trajectoire linéaire à une vitesse d'avance $V_a$ d'environ $3.5mm/s$. En dehors du mode endoscopique, le robot ne bouge pas sauf s'il en reçoit l'ordre.

Dans l'axe de l'outil, le robot peut fournir un effort maximal de $50N$. Cependant, pour avoir une meilleure précision, les efforts doivent se limiter à $10N$. La précision maximale du robot est inférieure à $1mm$ pour la position est inférieure à $0.2degrs$ pour l'orientation.

### 5.4.2 Moteur et mèches

Afin de réaliser l'opération, un petit moteur sera fixé au porte outil permettant ainsi de fournir le couple de rotation nécessaire pour faire tourner la mèche (figure 5.8).

(a) Moteur Microspeed. (b) Foret hélicoïdal à deux lèvres.

FIGURE 5.8 – Moteur et mèche utilisée.

Le moteur possède les caractéristiques suivantes :

— plage des vitesses de rotation ($Vr$) : 10000 à $80000tr/min$
— puissance : $P = 115w$
— poids : $m = 80g$
— couple maximal : $M_{max} = 26N.mm$

La mèche a les propriétés suivantes :

— matériaux : alliage de cobalt (phynox)
— diamètre de l'outil : $D = 2mm$
— angle d'hélice : $\delta = 30\deg$
— angle de pointe : $2Kr = 80\deg$

### 5.4.3 Caméra numérique

Afin de suivre le déroulement de la procédure et mieux analyser les résultats, une caméra numérique est utilisée (figure 5.9). Cette dernière a une résolution de $256 * 256$ pixels, avec un débit allant jusqu'à 4000 fps (image par seconde).

FIGURE 5.9 – Caméra utilisée.

### 5.4.4 Plaques en polyamide

Dans le but de tester la démarche de résolution proposée, des plaques de polyamide sont utilisées (figure 5.10). Le polyamide est un polymère contenant des fonctions amides, il est généralement à structures semi-cristallines, et présente un bon compromis entre caractéristiques mécaniques et chimiques. Ses propriétés mécaniques sont proches de celles du crâne humain avec une résistance de cisaillement de l'ordre de $100Mpa$. Sa température de fusion, qui varie selon la concentration en amide, est plutôt faible d'environ $170degrs$ et l'atteinte de cette dernière conduit à sa liquéfaction. Les plaques ont une dimension de $115 * 80 * 8mm$.

## 5.5 Démarche de résolution

Comme déjà mentionné précédemment, le système **Neuromate** est un système de guidage conçu pour assister le chirurgien lors des interventions au niveau du cerveau. On rappelle que

FIGURE 5.10 – plaque en polyamide.

le robot ne peut pas fournir de grands efforts. Cependant, l'opération de craniotomie est une intervention qui nécessite une certaine quantité d'efforts qui doit être fournie par le robot. L'utilisation du principe de la co-manipulation est une solution pour remédier à cette contrainte. Dans notre cas, on n'a pas accès au contrôleur principal du robot. Par conséquent, on ne pourra pas procéder avec la co-manipulation. Une autre solution est d'utiliser un modèle d'usinage permettant de quantifier les différents efforts, puis de vérifier la faisabilité de l'intervention.

L'usinage est un procédé de fabrication qui consiste à enlever de la matière d'une pièce à l'aide d'une machine-outil, et ce par la conjonction de deux mouvements : un mouvement de coupe et un mouvement d'avance. La qualité et la précision de l'usinage sont définies par la conformité de la pièce aux conditions de coupe :

— la vitesse de coupe ou vitesse de rotation $V_r$ ;
— la vitesse d'avance $V_a$ ;
— la profondeur de passe et l'avance $f$ ;
— la lubrification ;
— la géométrie et le matériau de l'outil ;
— le matériau des pièces à usiner ;
— la puissance de la machine et la nature de l'opération d'usinage (perçage, fraisage, tournage, etc).

Dans notre cas, il s'agit d'un perçage de l'os du crâne humain (la pièce) avec le système **Neuromate**, le moteur et les mèches. On dispose donc d'un mouvement de rotation de la mèche fourni par le moteur (mouvement de coupe) et un mouvement de translation linéaire fourni par le bras robotisé en mode endoscopique (mouvement d'avance). Les paramètres réglables sont les conditions de coupe, plus exactement la vitesse de coupe $V_r$, l'avance $f$ et la lubrification. Du coté robot, on a aussi la vitesse d'avance $V_a$ qui peut être modifiée.

Pour analyser et évaluer l'effet de ces différents paramètres, une modélisation du procédé d'usinage est nécessaire. Dans la littérature, plusieurs modélisations sont présentées [Elhachimi et al., 1999; Miguel et al., 2015] et un état de l'art sur la problématique de forage des os est présenté dans [Pandey & Panda, 2013]. Ces modèles varient selon l'opération à réaliser, la nature de l'outil, sa géométrie et aussi la prise en compte ou non de l'effet thermique. On peut distinguer deux types majeurs d'usinage : l'usinage conventionnel et l'usinage à grande vitesse (UGV). L'UGV peut lui-même être divisé en deux sous-parties :

— l'usinage à coupe orthogonale où l'arrête de coupe est perpendiculaire à la surface à usiner ;
— l'usinage à coupe oblique où l'arrête de coupe forme un angle différent de 0 (angle d'inclinaison) avec le plan perpendiculaire à la surface à usiner.

Dans notre cas, on utilise un perçage oblique à grande vitesse, car :

— la température n'est pas prise en considération dans le modèle étudié, et le perçage se fait à sec, donc les vitesses doivent être assez grandes pour minimiser le temps de l'opération au maximum ;
— la géométrie de l'outil dont nous disposons et l'opération à effectuer favorisent un angle d'inclinaison non nul ;
— avec un usinage à grande vitesse, on obtient une meilleure qualité comparativement à l'usinage conventionnel.

Dans notre cas, on a opté pour le modèle proposé dans [Elhachimi et al., 1999]. Ce modèle de perçage oblique à grande vitesse est dédié au perçage par forets hélicoïdale conventionnel, et est basé sur l'analyse de la continuité de l'effort d'avance $f_l$ et le couple de coupe $M_l$ à partir de la répartition de la force le long des arêtes de coupe. Ce modèle est fondé sur le développement du modèle de la zone de cisaillement établi par Oxley [Oxley, 2003] pour la coupe des métaux dans les deux parties de coupe du foret : les deux lèvres et le bord du ciseau. Cependant, dans notre cas , on va négliger les efforts causés par le bord du ciseau vu que ce dernier a des dimensions négligeables. Le modèle prend en considération :

— la géométrie de l'outil (figure 5.11) ;
— la contrainte au cisaillement du matériau à usiner $K_s$ ;
— le coefficient de frottement entre l'outil et le matériau $\lambda_n$ ;
— l'avance $f$ qui représente en millimètres la pénétration de l'outil en effectuant un seul tour.

Ainsi, on peut quantifier avec ce modèle la vitesse d'avance et le couple de coupe nécessaire. Les détails de ces calculs sont représentés en annexe.

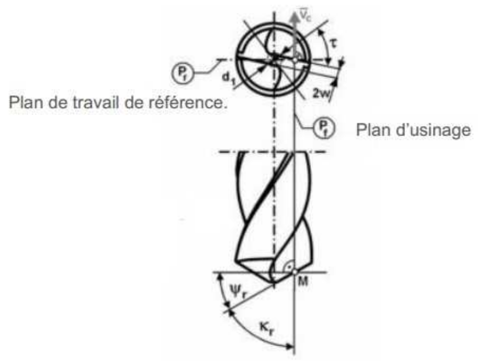
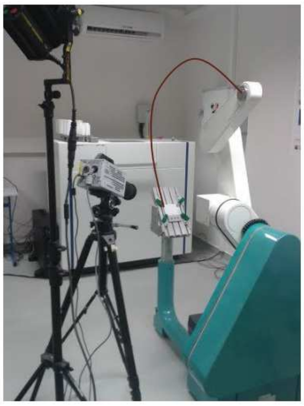


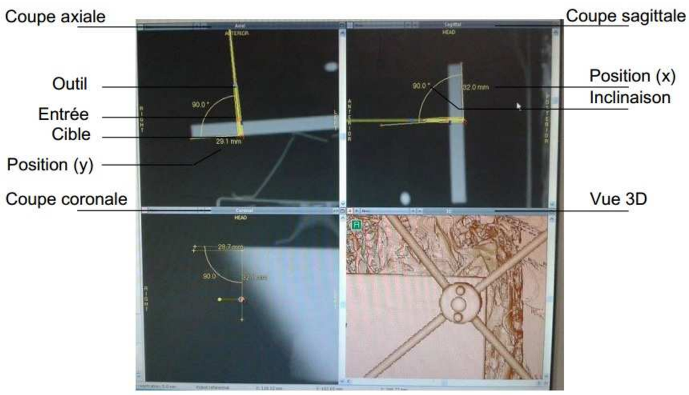

FIGURE 5.11 – Géométrie d'un foret hélicoïdal à deux lèvres.

## 5.6 Résultats

Après l'analyse et le calcul des différents paramètres, les tests sont réalisés en fixant la vitesse d'avance du bras à $3.5mm/s$ et en faisant varier la vitesse de rotation. La configuration du système utilisé est représenté dans la figure 5.12(a).



(a) Configuration du système utilisé.




(b) Capture d'écran sur **Voxim**.

FIGURE 5.12 – Système de test utilisé et une vue du logiciel de planification.

Dans la figure 5.12(b), on peut voir une capture d'écran du logiciel de planification utilisé avec les différentes informations. Afin de tester ce modèle de perçage, nous procédons à la réalisation de 48 trous sur la plaque en polyamide avec sept vitesses différentes. Pour chaque vitesse, on distingue deux types de trous :

— des perçages perpendiculaires à la surface de la plaque ;

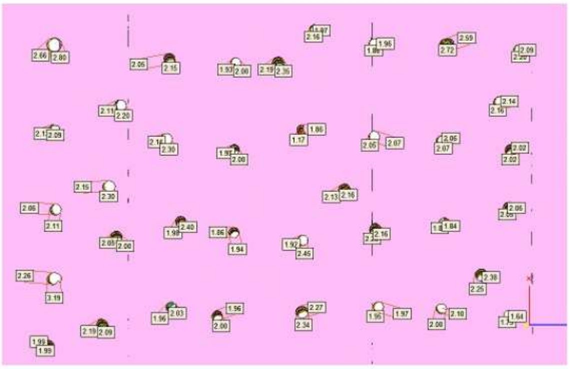
— des perçages où l'axe de l'outil forme un angle de $75degrs$ avec la surface de la plaque.

Pour conclure sur les différents trous réalisés, on analyse les différentes vidéos prises en mesurant :

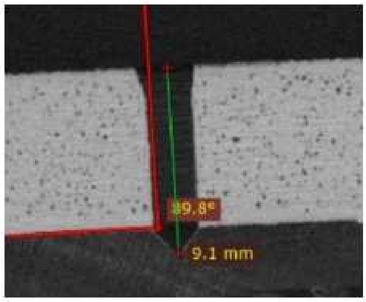
— le diamètre et la position spatiale des trous avec le logiciel **Matic** (figure 5.13(a)) ;

— la profondeur des trous en utilisant le logiciel **RadiAnt** (figure 5.13(b)).

Pour avoir une meilleure précision dans l'analyse des résultats, les mesures ont été faites sur quatre plans de coupe avec le logiciel **Matic**. A partir des mesures et des positions prédéterminées sur le logiciel **Voxim**, les erreurs de position spatiale et angulaire sont calculées dans chaque plan de coupe. La figure 5.14 représente la plaque après perçage.



(a) Mesure des diamètres sur **Matic**.




(b) Mesure de la profondeur sur **RadiAnt**.

FIGURE 5.13 – Logiciels utilisés pour les mesures.

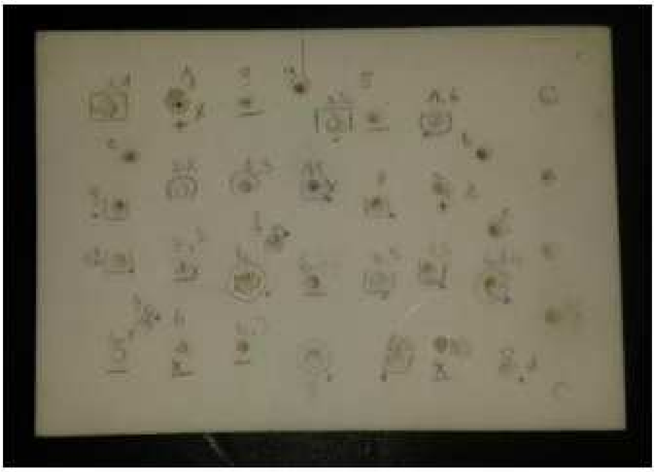


FIGURE 5.14 – Plaque après le perçage.

L'ensemble des résultats de mesure et des calculs effectués, sont représentés dans le tableau 5.2. Dans ce cas, les calculs ont été faits pour une vitesse de rotation de 10000 $tr/min$.

Dans le tableau 5.2, la partie paramètres du trou contient les différentes caractéristiques de chaque trou : son numéro, l'ordre de sa réalisation, la vitesse de rotation, son inclinaison et les deux efforts de coupe engendrés. L'inclinaison est notée **0** si l'axe de l'outil est perpendiculaire à la surface à usiner, et notée **1** s'il forme un angle de $75 degrs$ avec la surface. La colonne profondeur représente la distance entre la surface de la plaque et le fond du trou. Les parties

| **Paramètres du trou** | | | | | | | **Coupe à 0,5 mm** | | | **Coupe à 5,75** | | | **Coupe à 7,5 mm** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Numéro du trou** | **Ordre de réalisation** | **Vitesse de rotation** | **Efforts d'avance** | **Effort de coupe** | **Inclinaison** | **Profondeur** | **D** | **ex** | **ey** | **D** | **ex** | **ey** | **D** | **ex** | **ey** | **Erreur angulaire** | **Copeaux résiduels** |
| 1 | 3 | 10000 | | | 0 | 6,5 | 2,66 | 0,95 | 0,62 | 1,70 | 1,16 | 0,61 | nd | nd | nd | 2,33 | |
| 2 | 8 | 10000 | | | 0 | 7,4 | 2,55 | 0,15 | 1,39 | 1,84 | 0,19 | 1,55 | nd | nd | nd | 0,42 | x |
| 3 | 17 | 10000 | | | 0 | 7,7 | 2,45 | 0,24 | 0,91 | 2,05 | 0,09 | 1,05 | 0,65 | 0,04 | 1,10 | 3,64 | |
| 4 | 20 | 10000 | | | 0 | 9 | 2,54 | 0,90 | 0,95 | 1,91 | 0,68 | 1,40 | 2,01 | 0,72 | 1,45 | 2,39 | |
| 5 | 9 | 10000 | | | 0 | 8,9 | 2,12 | 0,34 | 1,82 | 1,96 | 0,52 | 2,16 | 1,86 | 0,35 | 2,40 | 1,93 | |
| 6 | 16 | 10000 | | | 0 | 9,4 | 2,57 | 1,88 | 1,12 | 1,97 | 1,91 | 1,12 | 1,93 | 1,84 | 1,05 | 0,33 | |
| 7 | 21 | 10000 | 2,48 | 0,88 | 0 | 8,3 | 1,79 | 0,51 | 1,85 | 2,00 | 0,68 | 2,17 | 1,39 | 0,77 | 2,08 | 1,89 | x |
| 8 | 19 | 10000 | | | 1 | 9,4 | 2,60 | 1,25 | 0,88 | 1,95 | 1,39 | 1,00 | 2,06 | 1,47 | 1,21 | 0,13 | |
| 9 | 23 | 10000 | | | 1 | 7,4 | 2,49 | 3,71 | 0,78 | 1,99 | 3,75 | 0,59 | nd | nd | nd | 1,75 | |
| 10 | 22 | 10000 | | | 1 | 8,6 | 1,85 | 1,81 | 0,71 | 1,80 | 2,14 | 0,47 | 1,54 | 2,30 | 0,85 | 4,61 | x |
| 11 | 18 | 10000 | | | 1 | 9,5 | 2,69 | 1,35 | 0,81 | 1,93 | 1,48 | 0,98 | 1,96 | 1,34 | 0,88 | 0,09 | |
| 12 | 6 | 10000 | | | 1 | 7 | 2,73 | 0,69 | 1,04 | 1,87 | 0,96 | 0,91 | nd | nd | nd | 1,37 | |
| 13 | 2 | 10000 | | | 1 | 7,2 | 2,84 | 0,79 | 0,58 | 1,93 | 0,81 | 1,01 | nd | nd | nd | 1,17 | |
| 14 | 10 | 10000 | | | 1 | 8,9 | 2,68 | 1,14 | 3,63 | 2,31 | 0,91 | 3,89 | 2,00 | 1,14 | 4,03 | 0,90 | |
| 15 | 24 | 10000 | | | 0 | 10,2 | 2,61 | 2,08 | 0,56 | 2,19 | 2,16 | 0,93 | 1,93 | 1,84 | 1,05 | 0,33 | |

TABLEAU 5.2 – Résultats de calcul pour une vitesse de rotation de 10000 $tr/min$

*coupe à xx mm* présentent, à chaque coupe de la plaque, le diamètre du trou et l'erreur de position par rapport aux deux axes $X$ et $Y$. Dans ce même tableau, on trouve aussi l'erreur angulaire et l'état de la surface réalisée.

Les mêmes calculs ont été réalisés avec d'autres vitesses de rotation (tableau 5.3 et 5.4).

### 5.6.1 Analyse des résultats obtenus

A partir des vidéos prises et des différents calculs réalisés, on peut diviser les résultats en trois grandes catégories selon la vitesse de rotation :

1. **Catégorie 1** : perçages réalisés avec une vitesse de 10000$tr/min$. Dans ce cas, on a remarqué des blocages fréquents de l'outil. Ces derniers sont dus essentiellement à l'incapacité de dégagement des copeaux à cette vitesse, ce qui a provoqué une élévation considérable de la température et par la suite la fusion du polyamide. Celui-ci se solidifie ensuite sur l'outil qui se bloque (figure 5.15). On remarque aussi que la moyenne des

| **Paramètres du trou** | | | | | | | **Coupe à 0,5 mm** | | | **Coupe à 5,75** | | | **Coupe à 7,5 mm** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Numéro du trou** | **Ordre de réalisation** | **Vitesse de rotation** | **Efforts d'avance** | **Effort de coupe** | **Inclinaison** | **Profondeur** | **D** | **ex** | **ey** | **D** | **ex** | **ey** | **D** | **ex** | **ey** | **Erreur angulaire** | **Copeaux résiduels** |
| 16 | 5 | 20000 | | | 0 | 6,4 | 2,65 | 0,65 | 1,17 | 2,07 | 0,39 | 1,19 | nd | nd | nd | 2,92 | |
| 17 | 11 | 20000 | | | 0 | 8,4 | 2,49 | 0,15 | 1,07 | 2,06 | 0,32 | 1,30 | 1,84 | 0,40 | 1,44 | 1,83 | |
| 18 | 38 | 20000 | 1,24 | 0,44 | 0 | 9,1 | 2,33 | 0,10 | 0,90 | 2,09 | 0,08 | 1,02 | 2,12 | 0,28 | 0,92 | 3,10 | |
| 19 | 32 | 20000 | | | 1 | 7,8 | 2,95 | 0,60 | 1,45 | 2,30 | 0,85 | 1,55 | 0,53 | 1,20 | 1,69 | 1,22 | |
| 20 | 4 | 20000 | | | 1 | 7,7 | 2,68 | 0,78 | 1,91 | 2,08 | 0,51 | 1,87 | 0,85 | 0,57 | 1,75 | 3,95 | |
| 21 | 44 | 20000 | | | 1 | 0,6 | 2,21 | 0,62 | 0,16 | 2,16 | 0,95 | 0,67 | 2,05 | 0,95 | 0,86 | 1,45 | |
| 22 | 30 | 25000 | | | 0 | 8,7 | 2,38 | 0,00 | 1,84 | 2,15 | 0,00 | 2,00 | 1,98 | 0,01 | 2,00 | 0,03 | |
| 23 | 27 | 25000 | | | 0 | 9,9 | 2,58 | 0,73 | 1,49 | 2,16 | 0,69 | 1,53 | 2,13 | 0,77 | 1,54 | 0,48 | |
| 24 | 37 | 25000 | 0,99 | 0,35 | 0 | 9,8 | 2,28 | 0,53 | 0,24 | 2,03 | 0,77 | 0,01 | 2,02 | 0,48 | 0,05 | 0,36 | x |
| 25 | 31 | 25000 | | | 1 | 8,6 | 2,74 | 0,86 | 2,00 | 2,09 | 0,85 | 1,75 | 1,87 | 1,11 | 1,85 | 1,45 | x |
| 26 | 29 | 25000 | | | 1 | 9,0 | 2,58 | 1,53 | 0,99 | 2,12 | 1,48 | 1,00 | 2,00 | 1,39 | 1,11 | 1,89 | |
| 27 | 43 | 25000 | | | 1 | 9,2 | 2,41 | 0,45 | 0,52 | 2,15 | 0,59 | 0,50 | 2,17 | 0,58 | 0,61 | 0,01 | x |
| 28 | 28 | 30000 | | | 0 | 9,3 | 2,86 | 0,96 | 1,15 | 2,23 | 1,13 | 1,45 | 2,20 | 1,08 | 1,32 | 1,79 | x |
| 29 | 36 | 30000 | | | 0 | 9,2 | 2,36 | 0,93 | 0,85 | 2,06 | 0,88 | 0,63 | 2,01 | 1,09 | 0,52 | 1,32 | |
| 30 | 46 | 30000 | 0,83 | 0,29 | 0 | 9,6 | 2,25 | 0,61 | 0,04 | 2,08 | 0,84 | 0,23 | 2,05 | 1,08 | 0,31 | 3,81 | x |
| 31 | 34 | 30000 | | | 1 | 8,7 | 2,25 | 0,70 | 0,69 | 2,01 | 1,17 | 0,59 | 1,87 | 1,56 | 0,48 | 5,70 | |
| 32 | 42 | 30000 | | | 1 | 10 | 2,52 | 1,86 | 0,19 | 2,01 | 2,13 | 0,42 | 2,12 | 1,98 | 0,33 | 0,13 | |
| 33 | 48 | 30000 | | | 1 | 9,9 | 2,61 | 1,38 | 0,41 | 2,14 | 1,37 | 0,51 | 2,08 | 1,58 | 0,13 | 0,52 | x |

Tableau 5.3 – Résultats de calcul pour les vitesses de rotation 20000, 25000 et 30000 $tr/min$.

diamètres des trous réalisés est inférieure à $2mm$. L'état de la surface interne et externe est généralement bon, lisse à l'intérieur, avec des copeaux résiduels négligeables.

2. **Catégorie 2** : perçages réalisés à $20000, 25000, 30000tr/min$. Dans ces cas, il n'y a eu aucun blocage de l'outil. Les diamètres sont légèrement supérieurs à $2mm$ avec une moyenne de $2.16mm$. L'état de la surface est plutôt bon, mais l'augmentation de la vitesse produit de légers copeaux résiduels sur la surface (figure 5.16(a)). Un détail à ne pas négliger est l'ordre de réalisation des trous. En effet, au fur et à mesure, l'outil se chauffe et affecte l'état de la surface de la plaque.

3. **Catégorie 3** : perçages réalisés à $35000, 45000, 80000tr/min$. Dans ce cas, il n'y a pas de blocage et les diamètres sont beaucoup plus grands, avec une moyenne de $2.21mm$. L'état de la surface est mauvais pour cette catégorie, et la présence de copeaux résiduels

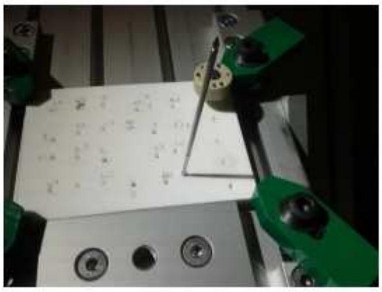
| **Paramètres du trou** | | | | | | | **Coupe à 0,5 mm** | | | **Coupe à 5,75** | | | **Coupe à 7,5 mm** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Numéro du trou** | **Ordre de réalisation** | **Vitesse de rotation** | **Efforts d'avance** | **Effort de coupe** | **Inclinaison** | **Profondeur** | **D** | **ex** | **ey** | **D** | **ex** | **ey** | **D** | **ex** | **ey** | **Erreur angulaire** | **Copeaux résiduels** |
| 34 | 26 | 35000 | | | 0 | 9,7 | 2,17 | 0,42 | 1,11 | 2,07 | 0,58 | 1,19 | 2,04 | 0,54 | 1,40 | 1,72 | x |
| 35 | 33 | 35000 | | | 0 | 10 | 2,28 | 0,43 | 0,50 | 2,13 | 0,46 | 0,64 | 2,16 | 0,44 | 0,61 | 0,05 | x |
| 36 | 45 | 35000 | 0,71 | 0,25 | 0 | 9,5 | 2,10 | 0,91 | 0,05 | 2,07 | 1,04 | 0,13 | 2,15 | 1,04 | 0,19 | 1,06 | x |
| 37 | 41 | 35000 | | | 1 | 8,8 | 2,48 | 1,11 | 0,39 | 2,16 | 1,02 | 0,23 | 1,93 | 1,33 | 0,23 | 0,70 | x |
| 38 | 47 | 35000 | | | 1 | 9,1 | 2,21 | 1,38 | 0,48 | 2,18 | 1,57 | 0,58 | 2,14 | 1,92 | 0,63 | 3,15 | x |
| 39 | 35 | 35000 | | | 1 | 9,5 | 2,47 | 1,23 | 0,32 | 2,00 | 1,50 | 0,27 | 1,93 | 1,60 | 0,60 | 1,84 | |
| 40 | 12 | 45000 | | | 0 | $>11$ | 2,77 | 0,83 | 2,34 | 2,66 | 1,59 | 2,12 | 2,67 | 1,49 | 2,16 | 8,21 | x |
| 41 | 13 | 45000 | | | 0 | 8,3 | 2,95 | 0,28 | 0,83 | 2,22 | 0,08 | 1,31 | 2,05 | 0,06 | 1,07 | 2,25 | x |
| 42 | 15 | 45000 | | | 1 | 8,3 | 2,64 | 0,17 | 1,20 | 2,18 | 0,46 | 1,77 | 1,88 | 0,76 | 1,71 | 1,58 | x |
| 43 | 14 | 45000 | 0,55 | 0,20 | 1 | 8,4 | 3,01 | 0,27 | 0,79 | 2,26 | 0,00 | 0,65 | 1,67 | 0,17 | 0,80 | 4,07 | |
| 44 | 25 | 45000 | | | 1 | 9,6 | 3,18 | 1,02 | 1,59 | 1,98 | 0,87 | 1,67 | 2,13 | 0,87 | 1,87 | 2,86 | x |
| 45 | 1 | 80000 | | | 0 | 7,3 | 2,43 | 0,42 | 1,53 | 2,19 | 0,44 | 1,51 | nd | nd | nd | 0,15 | x |
| 46 | 39 | 80000 | 0,31 | 0,11 | 0 | 9,5 | 2,39 | 1,16 | 0,20 | 2,15 | 1,29 | 0,30 | 2,22 | 1,56 | 0,26 | 3,31 | x |
| 47 | 7 | 80000 | | | 1 | 7,9 | 2,43 | 1,05 | 0,97 | 2,27 | 1,25 | 1,26 | nd | nd | nd | 0,69 | x |
| 48 | 40 | 80000 | | | 1 | 9,2 | 2,37 | 0,86 | 0,31 | 2,26 | 1,15 | 0,21 | 2,50 | 1,43 | 0,26 | 3,42 | x |

TABLEAU 5.4 – Résultats de calcul pour les vitesses de rotation 35000, 45000 et 80000 $tr/min$.

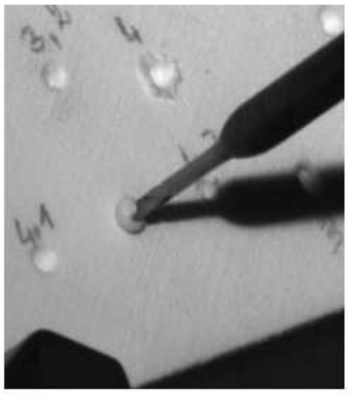
sur la surface est plus importante (figure 5.16(b)).

A partir des différents résultats obtenus, on peut conclure que les vitesses $10000, 35000$, 45000 et $80000tr/min$ sont à éviter vu qu'elles provoquent des blocages de l'outil ou des vibrations avec un mauvais état de la surface.

Pour ce qui est de l'erreur de position, elle a une moyenne de $1.02mm$. Elle est plus élevée pour les petites vitesses de rotation. Ceci peut être expliqué par l'importance des efforts, ce qui provoque une diminution de la précision. L'erreur est inférieure à $1mm$ pour les vitesses de rotation plus élevées, ce qui est conforme à la précision maximale du robot. A partir des différents résultats, la vitesse de rotation $20000tr/min$ est la plus adéquate parce qu'elle permet de donner de bons résultats vis-à-vis de l'état de la surface finale, du diamètre du trou qui est assez convenable, de l'erreur de position qui est de $0.91mm$, et de l'absence de vibration et de blocage.

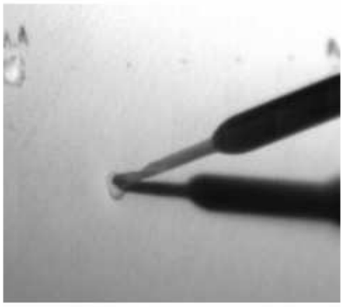


(a) Outil bloqué **Matic**.



(b) Solidification des copeaux autour de l'outil.

FIGURE 5.15 – Exemple de blocage de l'outil.



(a) Formation de copeaux autour de l'outil.

(b) Présence de copeaux sur la surface.

FIGURE 5.16 – Exemples de présence de copeaux.

Pour ce qui est de l'erreur angulaire, celle-ci est en moyenne beaucoup plus grande que la précision angulaire du robot, car elle peut atteindre $8degrs$. Cette différence est due à la précision des scannes réalisés et à l'erreur humaine.

La profondeur des trous réalisés est un paramètre très important, vu que le perçage doit être débouchant mais en même temps respecter la dure-mère. Avec le logiciel **RadiAnt**, on a pu mesurer la distance entre la surface et le fond du trou. Cependant, vu que les scannes utilisés pour la planification visuelle de la trajectoire ne sont pas d'une très bonne qualité, il est difficile d'avoir des valeurs exactes de la profondeur planifiée. Donc, seule la distance entre l'entrée et la cible de l'outil est fournie par **Voxim**. Ainsi, on ne peut pas procéder à une comparaison. Toutefois, on a pu déterminer la profondeur nécessaire à planifier pour que l'outil dépasse juste assez la surface et avoir ainsi un trou débouchant. Pour un trou perpendiculaire, elle est de

$9.73mm$, et pour les trous inclinés de $9.4mm$.

### 5.6.2 Amélioration de la démarche de résolution

Comme on a pu le constater, en explorant une gamme de vitesses de rotation de l'outil et en fixant la vitesse d'avance du bras, on a réussi à trouver la vitesse utile de rotation permettant de réaliser correctement l'opération de craniotomie. Afin d'améliorer la démarche de résolution, on propose dans cette section une formulation sous forme d'un problème d'optimisation, permettant de trouver les paramètres optimaux. Cette formulation est élémentaire et elle est valable uniquement en se mettant dans les mêmes conditions du cas étudié dans ce chapitre. Pour ce qui de l'opération de craniotomie, beaucoup de critères peuvent être définis. Intuitivement, l'idéal est d'aller plus vite afin de minimiser au maximum le temps de l'opération, c'est pourquoi, il sera judicieux de maximiser l'avance $f$ (qui représente en millimètres la pénétration de l'outil en effectuant un seul tour). Cette variable dépend à la fois de la vitesse de rotation de l'outil $V_r$ et de la vitesse d'avance du bras robotisé $V_a$. Cette dépendance est décrite dans l'équation 5.6.1.

$$f(V_a,V_r) = V_a/2.V_r \tag{5.6.1}$$

Ainsi, la fonction objectif à maximiser correspond à la fonction $f(V_a,V_r)$, avec comme variables de recherche $V_a$ et $V_r$. Les contraintes associées à cette fonction objectif ont pour but principal de respecter les limites physiques du système utilisé. Par conséquent, pour le bras robotisé, on doit respecter la plage des vitesses autorisées comme montré dans l'équation 5.6.2.

$$V_{a_{min}} \leq V_a \leq V_{a_{max}} \tag{5.6.2}$$

Avec $V_{a_{min}}$ et $V_{a_{max}}$ qui représentent respectivement le minimum et le maximum autorisés pour la vitesse d'avance. On a la même limitation pour la vitesse de rotation comme on peut le voir dans l'équation 5.6.3.

$$V_{r_{min}} \leq V_r \leq V_{r_{max}} \tag{5.6.3}$$

Avec $V_{r_{min}}$ et $V_{r_{max}}$ qui représentent respectivement le minimum et le maximum autorisés pour la vitesse de rotation. Avec le modèle d'usinage utilisé précédemment, on peut avoir le couple et l'effort de coupe nécessaires pour effectuer l'opération. L'expression de l'effort est décrite par l'équation 5.6.4.

$$Fl = A.Ks.f \tag{5.6.4}$$

Avec $A$ qui est un coefficient de proportionnalité propre au modèle d'usinage (voir annexe). $Ks$ est la contrainte de cisaillement qui dépend du matériau et $f$ est l'avance de l'outil. Connaissant l'effort de coupe nécessaire, on doit s'assurer que ce dernier peut être fourni par le bras robotisé en vérifiant qu'il est inférieur au maximum de l'effort disponible (équation 5.6.5), où $F_{max}$ est le maximum de l'effort.

$$Fl \leq F_{max} \tag{5.6.5}$$

Avec le même modèle, on peut quantifier le couple nécessaire selon l'équation 5.6.6.

$$Ml = B.Ks.f \tag{5.6.6}$$

Une fois le couple nécessaire obtenu, on peut calculer la puissance utile à fournir par le moteur pour produire ce couple, son expression est donnée dans l'équation 5.6.7.

$$P = Ml.\pi.V_r/30 \tag{5.6.7}$$

Aussi, on doit s'assurer que cette puissance est inférieure à la puissance maximale qui peut être fournie par le moteur (équation 5.6.8).

$$P \leq P_{max} \tag{5.6.8}$$

La formulation proposée est la suivante :

$$\max\ f(V_a,V_r) = V_a/2.V_r \tag{5.6.9}$$

$$\text{Sous les contraintes} \begin{cases} V_{a_{min}} \leq V_a \leq V_{a_{max}} \\ V_{a_{min}} \leq V_a \leq V_{a_{max}} \\ A.Ks.f \leq F_{max} \\ Ml.\pi.V_r/30 \leq P_{max} \end{cases}$$

### 5.6.3 Interface graphique

Afin de compléter la démarche de résolution proposée et faciliter le calcul des paramètres, on propose dans cette section une interface graphique réalisée sous **Matlab**. La figure 5.17 donne un aperçu général de cette interface.

La figure 5.17 synthétise les données d'entrée et de sortie de la technique de résolution proposée. Comme on peut le voir sur la figure, on a plusieurs sous-fenêtres qui peuvent être décrites comme suit :

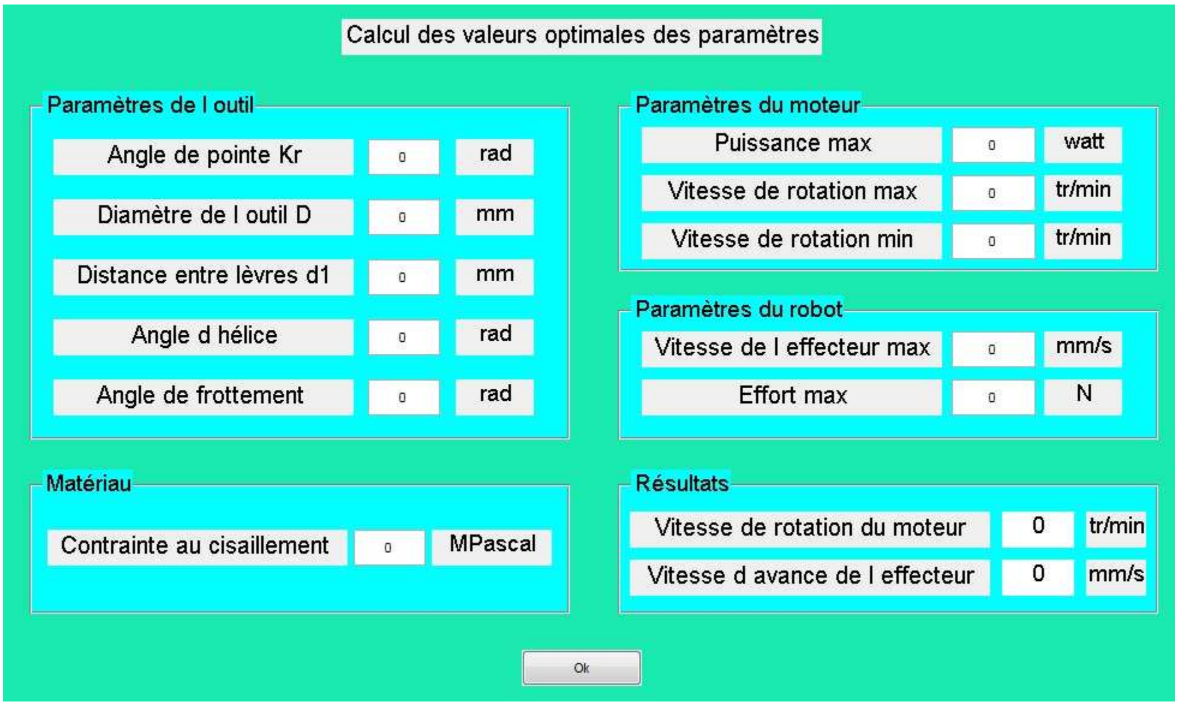



FIGURE 5.17 – Interface graphique.

- **Fenêtre 01** : la première fenêtre caractérise l'outil ou la mèche utilisée. Ces paramètres sont nécessaires pour quantifier l'effort et le couple nécessaire (voir annexe ) ;
- **Fenêtre 02** : cette deuxième fenêtre est utilisée pour prendre en compte les caractéristiques du matériau sur lequel l'opération va être effectuée. Dans notre cas, on prend juste la contrainte de cisaillement qui représente une sorte de résistante face à l'avance de l'outil ;
- **Fenêtre 03** : cette fenêtre est dédiée au moteur utilisé, qui permet de fournir le couple nécessaire pour faire tourner l'outil. Ainsi, pour pouvoir utiliser la formulation proposée dans la section précédente, on aura besoin de la puissance du moteur et des vitesses minimale et maximale ;
- **Fenêtre 04** : cette fenêtre est consacrée au bras robotisé. Elle permet la saisie de la vitesse maximale de l'effecteur permise ainsi que l'effort maximal que peut supporter le robot ;
- **Fenêtre 05** : cette dernière fenêtre sert à l'affichage des résultats obtenus. Dans notre cas, la vitesse d'avance de l'effecteur ainsi que la vitesse de rotation du moteur.

## 5.7 Conclusion

Dans ce chapitre, nous avons présenté la démarche préconisée pour réaliser l'opération de craniotomie en utilisant le système **Neuromate**. Rappelons que ce dernier est un robot de guidage, qui n'est pas conçu pour fournir des efforts. Malgré cet inconvénient, l'utilisation d'un modèle d'usinage à grande vitesse nous a permis de réaliser correctement cette opération. La démarche présentée tient compte du matériel disponible. En effet, vu qu'on n'avait pas accès au contrôleur du robot, on ne pouvait pas agir sur la boucle commande, mais on pouvait définir la consigne en termes de vitesse de l'effecteur. Tenant compte du type de mèche utilisé, le modèle d'usinage proposé permet de calculer l'effort de coupe et le couple nécessaire. Ces données sont essentielles pour statuer sur la capacité du matériel à réaliser l'opération.

nous avons montré à travers les résultats obtenus que ce modèle est valide et l'opération peut être réalisée correctement. En fixant la vitesse d'avance suite aux observations, et en explorant la gamme des vitesses de rotation, on a réussi à trouver la vitesse utile pour laquelle il n'y avait pas de blocage de l'outil, et aussi avoir une bonne précision dans les résultats.

En dernier point , la formulation proposée permet de trouver les paramètres optimaux par rapport au critère défini. Cette formulation est élémentaire, mais elle peut être améliorée en intégrant l'aspect thermique dans le modèle proposé.

# Conclusion générale et perspectives

Cette thèse est composée de deux grandes parties. La première concerne le problème de planification de trajectoire des bras manipulateurs redondants, la deuxième la problématique de craniotomie. Après avoir exposé un état de l'art sur le problème de planification, nous avons décrit en détail notre démarche de résolution, basée sur des algorithmes d'optimisation, qui nous permet de prendre en compte davantage de critères (contraintes) pour traiter au mieux ce problème. Nous avons divisé le problème de planification en deux sous-problèmes. Le premier a pour but de trouver l'ensemble de configurations (positions) par lesquelles le robot doit passer et le deuxième à interpoler entre ces positions afin de construire entièrement les courbes de déplacement (vitesses, accélérations et jerks). Ces deux points ont fait l'objet des deux chapitres 3 et 4, respectivement.

La technique de résolution des problèmes de planification de trajectoire proposée est exposée en détail dans le chapitre 3. Les différentes contraintes prises en compte dans ce traitement sont :

— atteinte de la position finale pour l'effecteur du robot ;
— évitement des différents obstacles qui peuvent se trouver dans l'espace de travail du robot ;
— évitement des singularités pour le robot.

La première contrainte est le but final de tout problème de planification de trajectoire. Les robots, d'une manière générale, opèrent dans des environnements encombrés d'obstacles. Ainsi, l'évitement d'obstacle est indispensable dans le processus de planification. L'évitement des singularités permet de garantir que les solutions générées sont bien à l'intérieur de l'espace de travail du robot. Dans notre étude, nous avons proposé une formulation basée sur les problèmes d'optimisation à deux niveaux, qui nous a permis d'avoir une formulation claire tout en travaillant dans les deux espaces de planification du robot en même temps. Dans le cas des manipulateurs, la tâche à accomplir (par l'effecteur) est souvent décrite dans l'espace cartésien. Ainsi, le déplacement de l'effecteur doit être suivi dans cet espace. Cependant, c'est les différentes articulations qui constituent le robot qui agissent sur le déplacement de l'effecteur. Donc, il faut prendre en compte la variation au cours du temps de ces dernières. Dans notre cas et avec les problèmes d'optimisation à deux niveaux, chaque niveau est dédié à un espace et les deux communiquent entre eux. Les deux niveaux s'exécutent alternativement et de manière séquentielle jusqu'à convergence de l'algorithme. Dans le premier niveau, on vise à atteindre la position finale pour l'effecteur tout en évitant les obstacles pour les centres des différentes articulations. Le premier niveau commence son exécution pour trouver une nouvelle position de l'effecteur, qui sera ensuite transmise au deuxième niveau avant de se mettre en état d'attente. Le deuxième niveau, après avoir reçu l'ordre d'exécution du premier, commence à son tour son exécution pour tenter de trouver la meilleure configuration (en termes de variables articulaires ) en évitant les obstacles et les singularités du robot. Cette configuration est testée par rapport à la valeur transmise par le premier niveau en considérant le modèle géométrique direct du robot comme une contrainte dans la formulation. Ainsi, une nouvelle technique d'évitement d'obstacle est proposée. Cette dernière s'adapte à la position actuelle de chaque bras du robot par rapport à chaque obstacle. Une fois que le deuxième niveau finit son exécution, il transmet le résultat au premier niveau qui va reprendre son exécution avant de se mettre une nouvelle fois en état d'attente. L'algorithme s'arrête une fois que l'effecteur a atteint la position finale. Le résultat de ce traitement est un ensemble de positions (en une seule dimension), qui constituent le chemin à emprunter pour aller de la position initiale à la position finale. En termes de moyens de résolution, nous avons choisi d'utiliser les métaheuristiques à cause de la complexité de la formulation. Nous avons montré à travers les différents tests la validité de la technique proposée.

Le chapitre 4 complète la démarche décrite dans le chapitre 3. En effet, dans ce chapitre, le point de départ est un ensemble de points en une seule dimension, qu'on cherche à interpoler. Les problèmes d'interpolation ont fait l'objet de beaucoup d'études dans la littérature. Cependant, le problème des données initiales à une seule dimension n'a pas encore été traité à notre connaissance. Une formulation est proposée dans ce travail. Le but étant d'assurer des courbes lisses en termes de vitesses, accélérations et jerks. Les contraintes de ce problème sont :

— atteindre la position finale très rapidement (minimiser le temps de parcours) ;
— minimiser la longueur des courbes ;
— minimiser les ondulations dans le déplacements.

Ces différents critères sont liés à la minimisation des vitesses, accélérations et jerks. On a opté pour une interpolation à base de fonctions polynomiales, ce qui nous permet de manipuler plus facilement les données. Avec la formulation proposée, on a énormément de contraintes à satisfaire (en termes d'optimisation). C'est pourquoi nous avons utilisé une métaheuristique à base d'algorithme génétique pour la résolution. Nous avons montré à travers les différents tests comment choisir les paramètres de pondération (pour la fonction objectif) et nous avons validé la technique en confirmant les résultats escomptés.

La deuxième partie de cette thèse concerne l'opération de craniotomie avec le robot Neuromate. Rappelons que cette opération consiste à réaliser une petite ouverture au niveau du crâne humain, afin d'y introduire des instruments pour retirer une tumeur, interrompre une hémorragies ou traiter un abcès. Assistée par un robot, cette opération d'incision des os du crâne représente est une forme particulière des problèmes de planification de trajectoires (avec des retours visuels). Ici, nous nous sommes intéressés uniquement à la réalisation de l'ouverture et non à toute la planification. La démarche classique de la craniotomie assistée par un robot est basée sur le principe de co-manipulation, ce qui impose l'accès au contrôleur principal. Dans notre cas, la démarche proposée repose sur l'utilisation d'un système d'usinage à grande vitesse, dont l'unique intervention humaine nécessite essentiellement de fixer la vitesse d'avance de l'effecteur et la vitesse de rotation de l'outil. Avec ce modèle, on a réussi à quantifier les forces et les couples nécessaires pour cette opération, et par suite la vitesse de déplacement de l'effecteur et celle de rotation de l'outil d'incision des os. On a procédé à plusieurs tests avec différentes vitesses de rotation pour valider la démarche. A la fin du chapitre, après avoir exploré les différents paramètres, on a proposé une formulation permettant d'obtenir les paramètres optimaux nécessaires à l'usinage. Les tests ont été réalisés sur des plaques en polyamides.

Plusieurs pistes peuvent être explorées pour améliorer ce travail. Dans la démarche de résolution du problème de planification de trajectoire proposée, on s'est limité à la cinématique du robot. Dans le cas des robots utilisés en chirurgie, les déplacements sont très petits, et en conséquence il n'est pas nécessaire de prendre en compte la dynamique du robot. Cependant, il est facile d'intégrer le modèle dynamique dans la formulation. Dans la technique d'interpolation proposée dans le chapitre 4, l'évitement d'obstacles n'a pas été pris en compte parce qu'elle a déjà été traité au niveau du chapitre 3. Cependant, on peut rendre la technique plus générique en intégrant l'évitement d'obstacles dans la formulation. D'un point de vue pratique, il sera intéressant d'implémenter cette partie sur le système Neuromate et l'adapter à un cas réel d'utilisation. En ce qui concerne la procédure de craniotomie, les tests ont été réalisés sur des plaques en polyamides. Il serait intéressant de tester la procédure sur des cadavres avant de l'appliquer aux vivants. Une autre piste est de modifier le modèle d'usinage afin de l'adapter à différentes formes d'outils d'incision.

# Annexe

## 6.1 Le système Neuromate

Dans cette section, on présente le modèle géométrique et cinématique du robot **Neuromate**. Comme déjà mentionné, on utilisera la convention de Denavit hartenberg qui est la plus utilisée pour la modélisation des systèmes articulés.

### 6.1.1 Modèle géométrique

Le robot **Neuromate** possède une structure cinématique série classique à cinq articulations pivots, la configuration géométrique du robot est représentée dans la figure A.1.

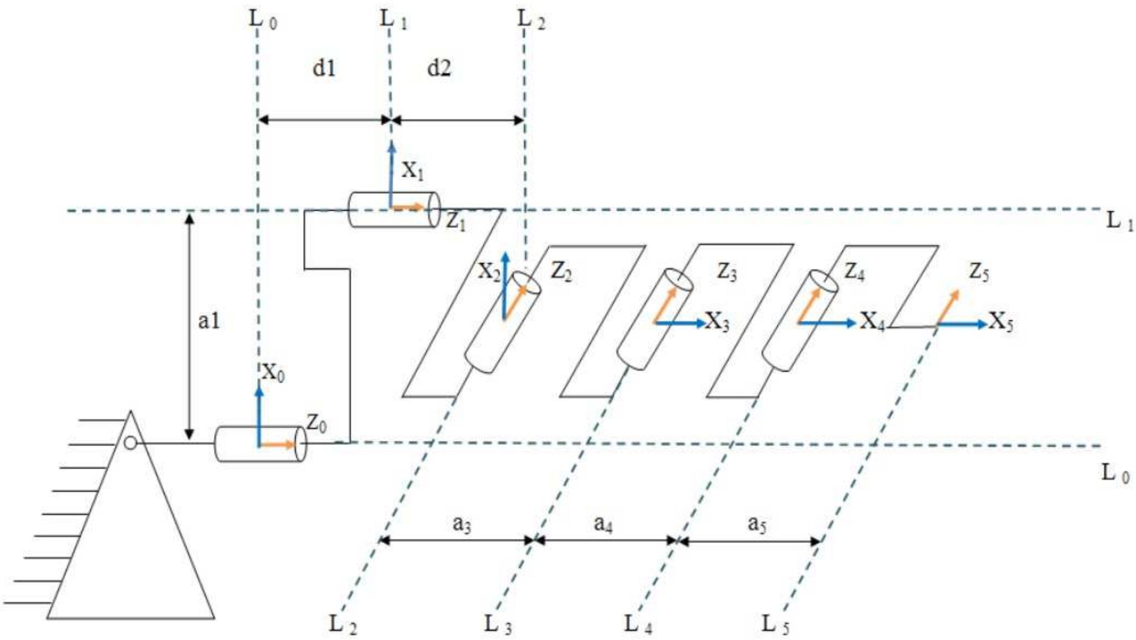



Figure A.1 – Configuration géométrique du robot.

Les limites des articulations du robot sont représentées dans le tableau suivant :

| N° articulation | Min(degrés) | Max(degrés) |
|---|---|---|
| 1 | -180 | 180 |
| 2 | -180 | 180 |
| 3 | -90 | 90 |
| 4 | -90 | 90 |
| 5 | -90 | 90 |

Tableau A.1 – limites articulaires du robot.

Le but de la convention de Denavit Hartenberg (DH) est de trouver les coordonnées de l'effecteur du robot dans un repère lié à la base du robot (repère fixe). Pour ce faire, on attribue à chaque articulation un repère relatif et avec la convention DH, on détermine les paramètres géométriques du robot qui sont mentionnés dans la tableau suivant :

| corps | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$ | $\phi_i$ |
|---|---|---|---|---|---|
| 1 | $a_1$ | 0 | $d_1$ | $q_1$ | 0 |
| 2 | 0 | $\pi/2$ | $d_2$ | $q_2$ | 0 |
| 3 | $a_3$ | 0 | 0 | $q_3$ | 0 |
| 4 | $a_4$ | 0 | 0 | $q_4$ | 0 |
| 5 | $a_5$ | 0 | 0 | $q_5$ | 0 |

Tableau A.2 – Tableau des paramètres géométriques.

Les paramètres ($a_i$,$\alpha_i$,$d_i$,$\theta_i$,$\phi_i$), appelés paramètres géométriques, tels que :

— $\phi_i$ : type de l'articulation (0 : rotoide et 1 : prismatique).

— $\alpha_i$ : angle entre $Z_{i-1}$ et $Z_i$ autour de $X_i$.

— $a_{i-1}$ : distance entre $Z_{i-1}$ et $Z_i$ le long de $X_i$.

— $\theta_i$ : angle entre $X_{i-1}$ et $X_i$ le long de $Z_{i-1}$.

— $d_i$ : distance entre $X_{i-1}$ et $X_i$ le long de $Z_{i-1}$.

En prenant $\cos(\theta_i) \equiv C_i$, $\sin(\theta_i) \equiv S_i$, $\cos(\alpha_i) \equiv C\alpha_i$ et $\sin(\alpha_i) \equiv S\alpha_i$, la forme générale de la matrice de passage entre le repère $R^{i-1}$ et le repère $R^i$ est la suivante :

$$T_i^{i-1} = \begin{pmatrix} C_i & -S_iC\alpha_i & S_iS\alpha_i & a_iC_i \\ S_i & C_iC\alpha_i & -C_iS\alpha_i & a_iS_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Les matrices de transformations sont les suivantes :

$$T_1^0 = \begin{pmatrix} C_1 & -S_1 & 0 & a_1C_1 \\ S_1 & C_1 & 0 & a_1S_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T_2^1 = \begin{pmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_3^2 = \begin{pmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T_4^3 = \begin{pmatrix} C_4 & -S_4 & 0 & a_4C_4 \\ S_4 & C_4 & 0 & a_4S_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_5^4 = \begin{pmatrix} C_5 & -S_5 & 0 & a_5C_5 \\ S_5 & C_5 & 0 & a_5S_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Le modèle géométrique direct du robot est obtenu en faisant une multiplication de toutes ces matrices. De cette manière, on obtient les coordonnées de l'effecteur dans le repère $R_0$. Ainsi, on cherche le produit $T_5^0 = T_1^0.T_2^1.T_3^2.T_4^3.T_5^4$, appelé matrice homogène globale. Pour effectuer ces calculs, on utilise le logiciel **Maple**. La valeur de la matrice $T_5^0$ (après simplification) est :

$$T_5^0 = \begin{pmatrix} C_{12}C_{345} & -C_{12}S_{345} & S_{12} & a_5C_{12}C_{345} + a_4C_{12}C_{34} + a_3C_{12}C_3 + a_1C_1 \\ -S_{12}C_{345} & -S_{12}S_{345} & -C_{12} & a_5S_{12}C_{345} + a_4S_{12}C_{34} + a_3S_{12}C_3 + a_1S_1 \\ S_{345} & C_{345} & 0 & a_5S_{345} + a_4S_{34} + a_3S_3 + d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La position de l'effecteur est donnée par :

$$\begin{pmatrix} X = a_5C_{12}C_{345} + a_4C_{12}C_{34} + a_3C_{12}C_3 + a_1C_1 \\ Y = a_5S_{12}C_{345} + a_4S_{12}C_{34} + a_3S_{12}C_3 + a_1S_1 \\ Z = a_5S_{345} + a_4S_{34} + a_3S_3 + d_1 + d_2 \end{pmatrix}$$

### 6.1.2 Modèle cinématique

Le modèle cinématique du robot décrit la relation entre la vitesse de l'effecteur $\dot{X}$ et les vitesses des articulations $\dot{q}$, il est définit par :

$$\dot{X} = J\dot{q}$$

telle que la variable J est la matrice jacobienne, son expression est la suivante :

$$J = \begin{pmatrix} \frac{\partial X}{\partial q1} & \frac{\partial X}{\partial q2} & \frac{\partial X}{\partial q3} & \frac{\partial X}{\partial q4} & \frac{\partial X}{\partial q5} \\ \frac{\partial Y}{\partial q1} & \frac{\partial Y}{\partial q2} & \frac{\partial Y}{\partial q3} & \frac{\partial Y}{\partial q4} & \frac{\partial Y}{\partial q5} \\ \frac{\partial Z}{\partial q1} & \frac{\partial Z}{\partial q2} & \frac{\partial Z}{\partial q3} & \frac{\partial Z}{\partial q4} & \frac{\partial Z}{\partial q5} \end{pmatrix}$$

Avec :

$J_{11} = \frac{\partial X}{\partial q1} = -a_5 S_{12} C_{345} - a_4 S_{12} C_{34} - a_3 C_3 S_{12} - a_1 S_1$
$J_{12} = \frac{\partial X}{\partial q2} = -a_5 S_{12} C_{345} - a_4 S_{12} C_{34} - a_3 C_3 S_{12}$
$J_{13} = \frac{\partial X}{\partial q3} = -a_5 C_{12} S_{345} - a_4 C_{12} S_{34} - a_3 S_3 C_{12}$
$J_{14} = \frac{\partial X}{\partial q4} = -a_5 C_{12} S_{345} - a_4 C_{12} S_{34}$
$J_{15} = \frac{\partial X}{\partial q5} = -a_5 C_{12} S_{345}$

$J_{21} = \frac{\partial Y}{\partial q1} = a_5 C_{12} C_{345} + a_4 C_{12} C_{34} + a_3 C_3 C_{12} + a_1 C_1$
$J_{22} = \frac{\partial Y}{\partial q2} = a_5 C_{12} C_{345} + a_4 C_{12} C_{34} + a_3 C_3 C_{12}$
$J_{23} = \frac{\partial Y}{\partial q3} = -a_5 S_{12} S_{345} - a_4 S_{12} S_{34} - a_3 S_3 S_{12}$
$J_{24} = \frac{\partial Y}{\partial q4} = -a_5 S_{12} S_{345} - a_4 S_{12} S_{34}$
$J_{25} = \frac{\partial Y}{\partial q5} = -a_5 S_{12} S_{345}$

$J_{31} = \frac{\partial Z}{\partial q1} = 0$
$J_{32} = \frac{\partial Z}{\partial q2} = 0$
$J_{33} = \frac{\partial Z}{\partial q3} = a_5 C_{345} + a_4 C_{34} - a_3 S_3$
$J_{34} = \frac{\partial Z}{\partial q4} = a_5 C_{345} + a_4 C_{34}$
$J_{35} = \frac{\partial Z}{\partial q5} = a_5 C_{345}$

## 6.2 Modèle d'usinage

Comme mentionné dans le chapitre 5, un modèle d'usinage à grande vitesse a été utilisé. Ce dernier nous permet de quantifier l'effort de coupe ainsi que le couple nécessaire. Rappelons que ce modèle est appliqué pour un certain modèle de mèches, comme celui montré dans la figure A.2.

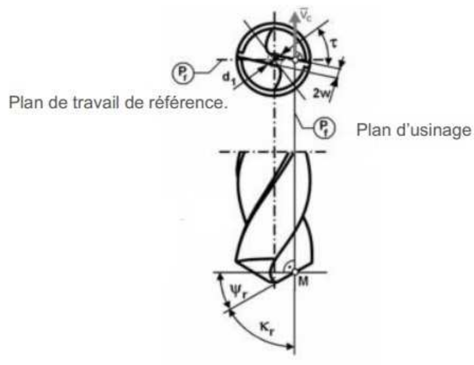



FIGURE A.2 – Géométrie d'un foret hélicoïdal à deux lèvres.

Ce modèle donne les efforts en coupe en fonction de la contrainte de cisaillement du matériau à usiner, l'avance et l'intégrale d'une fonction qui dépend de la géométrie de l'outil et du coefficient de frottement, par rapport au rayon de l'outil sur les arêtes de coupes. Les formules obtenues sont :

$$Fl = Ks.f.\int_{d1/2}^{d/2} \frac{\sin(Kr).\cos(\mu)}{\sin(\phi_n).\cos(\theta_n)}.(\sin(\lambda_n - \gamma_n - \mu).\sin(Kr) - \cos(Kr)).\frac{r}{\sqrt{r^2 - w^2}}dr \quad (6.2.1)$$

$$Ml = Ks.f.\int_{d1/2}^{d/2} \frac{\sin(Kr).\cos(\mu)}{\sin(\phi_n).\cos(\theta_n)}.\cos(\phi_n - \gamma_n - i).\frac{r}{\sqrt{r^2 - w^2}}dr \quad (6.2.2)$$

L'équation 6.2.1 représente la formule utilisée pour le calcul de l'effort de coupe et l'équation 6.2.2 celle utilisée pour le couple. La fonction intégrale n'est pas définie explicitement, ce qui rend le calcul de cette intégrale assez complexe. En effet, le équations nous permettent d'obtenir $r$ (rayon de l'outil) en fonction de $\theta$ (angle intermédiaire). Cependant, pour faire ce calcul, on aura besoin de déterminer $\theta$ en fonction de $r$.

Pour donner une estimation de la valeur de cette intégrale, on a utilisé la somme de Riemann associée à une subdivision irrégulière de l'intervalle $[\frac{d1}{2}, \frac{d}{2}]$ de la forme $r_i = r(\theta_n^i)$. Les intégrales à calculer sont :

$$\int_{d1/2}^{d/2} \frac{\sin(Kr).\cos(\mu)}{\sin(\phi_n).\cos(\theta_n)}.(\sin(\lambda_n - \gamma_n - \mu).\sin(Kr) - \cos(Kr)).\frac{r}{\sqrt{r^2 - w^2}}dr \tag{6.2.3}$$

$$\int_{d1/2}^{d/2} \frac{\sin(Kr).\cos(\mu)}{\sin(\phi_n).\cos(\theta_n)}.\cos(\phi_n - \gamma_n - i).\frac{r}{\sqrt{r^2 - w^2}}dr \tag{6.2.4}$$

Avec :

- $Kr$ :1/2 de l'angle de pointe
- $w$ :1/2 distance entre les lèvres
- $r$ : rayon de l'outil avec $r \in I = [\frac{d1}{2}, \frac{d}{2}]$
- $d$ : diamètre de l'outil
- $\mu$ : angle intermédiaire avec

$$\mu = \arctan(\tan(\arcsin(\frac{w}{r}))\cos(Kr))$$

- $\gamma_n$ : angle de coupe normale

$$\gamma_n = \frac{\tan(\frac{d}{2r}\tan(\delta_0))\cos(\arcsin(\frac{w}{r}))}{\sin(Kr) - \cos(Kr)\tan(\frac{d}{2r}\tan(\delta_0))\frac{w}{r}} - \mu$$

- $\lambda_n$ : angle de frottement avec : $\lambda_n = \arctan(0.5)$
- $d1$ : distance entre les lèvres avec $d1 = 2w$
- $\delta_0$ : angle d'hélice de l'outil
- $\phi_n$ : angle de cisaillement
- $\theta_n$ : angle intermédiaire
- $i$ : angle d'inclinaison avec $\arcsin(\frac{w}{r}\sin(Kr))$

L'angle de cisaillement $\psi_n$ et l'angle intermédiaire $\theta_n$ sont donnés par les équations 6.2.5,6.2.6 et 6.2.7.

$$h = \frac{t_1\sin(\theta_n)}{\cos(\lambda_n)\sin(\theta_n - \lambda_n + 2Kr)}(1 + \frac{Cn}{3\tan(\theta_n)}) \tag{6.2.5}$$

$$\theta_n = \psi_n + \lambda_n - 2Kr \tag{6.2.6}$$

$$\tan(\theta_n) = 1 + 2(\frac{\pi}{4} - \psi_n) - Cn \tag{6.2.7}$$

Avec :

- $h$ : longueur de l'outil : $h = 2r\pi$
- $t_1$ : épaisseur du copeau non déformé

— $C$ : constante de déformation

— $n$ : index de déformation et durcissement

Après simplification, on obtient les deux formules suivantes :

$$\psi_n = \theta_n - \lambda_n + 2Kr \tag{6.2.8}$$

$$r = \frac{d\sin(\theta_n)}{4\pi\cos(\lambda_n)\sin(\theta_n - \lambda_n + 2Kr)}\left(1 + \frac{1 + \frac{\pi}{2} - 2(\theta_n - \lambda_n + 2Kr) - \tan(\theta_n)}{3\tan(\theta_n)}\right) \tag{6.2.9}$$

Par conséquent, on obtient $r$ en fonction de $\theta_n$. A partir de la relation décrite par l'équation 6.2.9, et en utilisant un solveur Excel, on peut déterminer l'intervalle $J$ de $\theta_n$ pour lequel $r \in I = [\frac{d1}{2}, \frac{d}{2}]$. On subdivise cet intervalle en un nombre assez grand de parties pour bien cerner l'intégral. Le pas choisi est de $10^{-8}$, c'est à dire :

$$\theta_n^{i+1} = \theta_n^i + 10^{-8} \tag{6.2.10}$$

Une fois les $\theta_n^i$ déterminées, on calcule les $r_i$ :

$$r_i = r(\theta_n^i) \tag{6.2.11}$$

Ainsi, on a obtenu une subdivision suffisamment fine de l'intervalle $I$. En suite à l'aide de la formule de Riemann, on calcule les deux intégrales (équations 6.2.1 et 6.2.2). Les résultats sont :

$$Fl = 2.75Ksf$$

$$Ml = 0.97Ksf$$

# Références bibliographiques


[Avriel, 1976] M. Avriel. *Nonlinear Programming : Analysis and Methods.* Prentice-Hall, 1976.

[Basili & Turner, 1975] V. R. Basili & A. Turner. Iterative enhancement : A practical technique for software development. In *IEEE Transactions on Software Engineering*, pp. 390–396, 1975.

[Benslimane, 2015] A. Benslimane. Craniotomie robotisée : intégration d'un système de perçage à un robot de neurochirurgie. Master's thesis, ENSAM, 2015.

[Beyer, 2001] H. G. Beyer. *The theory of evolution strategies.* Springer, 2001.

[Calvete et al., 2008] H. I. Calvete, C. Galé, & P. M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1): 14–28, 2008.

[Chakraborti et al., 2011] T. Chakraborti, A.l Sengupta, A. Konar, & R. Janarthanan. Application of swarm intelligence to a two-fold optimization scheme for trajectory planning of a robot arm. In *Swarm, Evolutionary, and Memetic Computing*, vol. 7077, pp. 89–96. Springer Berlin Heidelberg, 2011.

[Chang & Khatib, 1995] K.-S. Chang & O. Khatib. Manipulator control at kinematic singularities : a dynamically consistent strategy. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, pp. 84–88, 1995.

[Chettibi et al., 2004] T. Chettibi, H. E. Lehtihet, M. Haddad, & S. Hanch. Minimum cost trajectory planning for industrial robots. *European Journal of Mechanics*, 23(4): 703–715, 2004.

[Chyan & Ponnambalam, 2012] G. S. Chyan & S. G. Ponnambalam. Obstacle avoidance control of redundant robots using variants of particle swarm optimization. *Robotics and Computer-Integrated Manufacturing*, 28(2): 147–153, 2012.

[Conn et al., 1991] A. R. Conn, N. I. M. Gould, & P. L. Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2): 545–572, 1991.

[Conn et al., 1992] A. R. Conn, N. I. M. Gould, & P. L. Toint. A globally convergent augmented lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, 66(0): 261–288, 1992.

[Courat et al., 1993] J.-P. Courat, G. Raynaud, I. Mrad, & P. Siarry. Electronic component model minimization based on log simulated annealing. *Circuits and Systems I : Fundamental Theory and Applications, IEEE Transactions on*, 41(12): 790–795, 1993.

[Daachi et al., 2012] B. Daachi, T. Madani, & A. Benallegue. Adaptive neural controller for redundant robot manipulators and collision avoidance with mobile obstacles. *Neurocomputing*, 79(0): 50–60, 2012.

[DeVore & Temlyakov, 1996] R. A. DeVore & V. N. Temlyakov. Some remarks on greedy algorithms. *Advances in Computational Mathematics*, 5(1): 173–187, 1996.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo, & A. Colorni. Ant system : optimization by a colony of cooperating agents. In *IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics*, pp. 29–41, 1996.

[Dréo et al., 2003] J. Dréo, A. Pétrowski, P. Siarry, & E. Taillard. *Métaheuristiques pour l'Optimisation Difficile*. Eyrolles, 2003.

[E. J. Solteiro et al., 204] P. E. J. Solteiro, M. J. A. Tenreiro, & P. B. de Moura Oliveira. Robot trajectory planning using multi-objective genetic algorithm optimization. In *Genetic and Evolutionary Computation*, vol. 3102, pp. 615–626. Springer Berlin Heidelberg, 204.

[El. Dor, 2012] A. El. Dor. *Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en ' electronique*. PhD thesis, Universit 'e Paris Est, 2012.

[Elhachimi et al., 1999] M. Elhachimi, S. Torbaty, & P. Joyot. Mechanical modelling of high speed drilling. 1 : predicting torque and thrust. *International Journal of Machine Tools and Manufacture*, 39(4): 553–568, 1999.

[Fogel et al., 1966] L. J. Fogel, A. J. Owens, & M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, 1966.

[Fraser, 1957] A. S. Fraser. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10: 484–491, 1957.

[Garg & Kumar, 2002] D. P. Garg & M. Kumar. Optimization techniques applied to multiple manipulators for path planning and torque minimization. *Engineering Applications of Artificial Intelligence*, 15(3–4): 241–252, 2002.

[Gasparetto & Zanotto, 2007] A. Gasparetto & V. Zanotto. A new method for smooth trajectory planning of robot manipulators. *Mechanism and Machine Theory*, 42(4): 455–471, 2007.

[Gasparetto & Zanotto, 2010] A. Gasparetto & V. Zanotto. Optimal trajectory planning for industrial robots. *Advances in Engineering Software*, 41(4): 548–556, 2010.

[Glover, 1986] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13: 533–549, 1986.

[Glover & Laguna, 1997] F. Glover & M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[Goss et al., 1989] S. Goss, S. Aron, J. L. Deneubourg, & J. M. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12): 579–581, 1989.

[Hejazi et al., 2002] S. R. Hejazi, A. Memariani, G. R. Jahanshahloo, & M. M. Sepehri. Linear bilevel programming solution by genetic algorithm. *Computers & Operations Research*, 29(13): 1913–1925, 2002.

[Holland, 1975] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[Huang et al., 2008] P. Huang, G. Liu, J. Yuan, & Y. Xu. Multi-objective optimal trajectory planning of space robot using particle swarm optimization. In *Advances in Neural Networks - ISNN 2008*, vol. 5264, pp. 171–179. Springer Berlin Heidelberg, 2008.

[Huberti, 2010] J. Huberti. *Manipulateurs parallèles, singularités et analyse statique.* PhD thesis, Ecole des Mines de Paris, 2010.

[Jiang et al., 2013] Y. Jiang, X. Li, C. Huang, & X. Wu. Application of particle swarm optimization based on chks smoothing function for solving. *Applied Mathematics and Computation*, 219(9): 4332–4339, 2013.

[Kennedy & Eberhart, 1995] J. Kennedy & R. C. Eberhart. Particle swarm optimization. In *In : Proceedings of the IEEE International Conference on Neural Networks IV,*, pp. 1942–1948, 1995.

[Khatib, 1985] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, pp. 500–505, 1985.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C. D. Gellatt, & M. P. Vecchi. Optimization by simulated annealing. *Science*, 220: 671–680, 1983.

[Koza, 1989] J. R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. *Morgan Kaufmann, Detroit, Michigan, USA*, pp. 768–774, 1989.

[Kucuk & Bingul, 2006] S. Kucuk & Z. Bingul. Comparative study of performance indices for fundamental robot manipulators. *Robotics and Autonomous Systems*, 54(7): 567–573, 2006.

[Kuffner & LaValle, 2000] J. J. Kuffner & S. M. LaValle. Rrt-connect : An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, pp. 995–1001, 2000.

[Le Boudec et al., 2006] B. Le Boudec, M. Saad, & V. Nerguizian. Modeling and adaptive control of redundant robots. *Mathematics and Computers in Simulation (MATCOM)*, 71(4): 395–403, 2006.

[Lewis & Torczon, 2002] M. R. Lewis & V. Torczon. A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4): 1075–1089, 2002.

[Li & Wang, 2006] H. Li & Y. Wang. A hybrid genetic algorithm for solving a class of nonlinear bilevel programming problems. In *Simulated Evolution and Learning*, vol. 4247, pp. 408–415. Springer Berlin Heidelberg, 2006.

[Lin, 2004] C. J Lin. Motion planning of redundant robots by perturbation method. *Mechatronics*, 14(3): 281–297, 2004.

[Liu et al., 2013] H. Liu, X. Lai, & W. Wu. Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints. *Robotics and Computer-Integrated Manufacturing*, 29(2): 309–317, 2013.

[Liu et al., 2007] J. Liu, Y. Zhang, & L. Zhen. Improving the positioning accuracy of a neurosurgical robot system. *Mechatronics, IEEE/ASME Transactions on*, 12(5): 527–533, 2007.

[Lv et al., 2007] Y. Lv, T. Hu, G. Wang, & Z. Wan. A penalty function method based on kuhn-tucker condition for solving linear bilevel programming. *Applied Mathematics and Computation*, 188(1): 808–813, 2007.

[Madani et al., 2013] T. Madani, B. Daachi, & A. Benallegue. Adaptive variable structure controller of redundant robots with mobile/fixed obstacles avoidance. *Robotics and Autonomous Systems*, 61(6): 555–564, 2013.

[Marcos et al., 2009] M. D. G Marcos, J.A. T. Machado, & T. P. Perdicoulis A. Trajectory planning of redundant manipulators using genetic algorithms. *Communications in Nonlinear Science and Numerical Simulation*, 14(7): 2858–2869, 2009.

[Marcos et al., 2012] M. D. G. Marcos, J.A. T. Machado, & T. P. Perdicoulis A. A multi-objective approach for the motion planning of redundant manipulators. *Applied Soft Computing*, 12(2): 589–599, 2012.

[Maria D. G. et al., 2010] Marcos Maria D. G., J. A. Machado T., & T. P. Perdicoulis A. An evolutionary approach for the motion planning of redundant and hyper-redundant manipulators. *Nonlinear Dynamics*, 60(1-2): 115–129, 2010.

[Menasri et al., 2013] R. Menasri, A. Nakib, H. Oulhadj, B. Daachi, P. Siarry, & G. Hains. Path planning for redundant manipulators using metaheuristic for bilevel optimization and maximum of manipulability. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pp. 145–150, December 2013.

[Menasri et al., 2014a] R. Menasri, H. Oulhadj, B Daachi, A. Nakib, & P. Siarry. Smooth trajectory planning for robot using particle swarm optimization. In *Swarm Intelligence Based Optimization*, vol. 8472, pp. 50–59. Springer, 2014.

[Menasri et al., 2014b] R. Menasri, H. Oulhadj, B. Daachi, A. Nakib, & P. Siarry. A genetic algorithm designed for robot trajectory planning. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pp. 228–233, October 2014.

[Menasri et al., 2015] R. Menasri, A. Nakib, B. Daachi, H. Oulhadj, & P. Siarry. A trajectory planning of redundant manipulators based on bilevel optimization. *Applied Mathematics and Computation*, 250: 934–947, 2015.

[Metropolis et al., 1953] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, & E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6): 1087–1092, 1953.

[Miguel et al., 2015] M. Miguel, R. Marcos, S. Carlos, G. Eugenio, & H. H. Maria. A review on recent advances in numerical modelling of bone cutting. *Journal of the Mechanical Behavior of Biomedical Materials*, 44: 179–201, 2015.

[Nelder & Mead, 1965] J. A. Nelder & R. Mead. A simplex method for function minimization. *The computer journal*, 7(4): 308–313, 1965.

[Nocedal & Wright, 1998] J. Nocedal & S. J. Wright. *Numercial Optimization*. Springer, 1998.

[Oduguwa & Roy, 2002] V. Oduguwa & R. Roy. Bi-level optimisation using genetic algorithm. In *Artificial Intelligence Systems, 2002. (ICAIS 2002). 2002 IEEE International Conference on*, pp. 322–327, 2002.

[O'Dúnlaing et al., 1987] C. O'Dúnlaing, M. Sharir, & C. Yap. Generalized voronoi diagrams for a ladder : Efficient construction of the diagram. *Algorithmica*, 2(1-4): 27–59, 1987.

[Oxley, 2003] P. L. B. Oxley. Modelling machining processes with a view to their optimization and to the adaptive control of metal cutting machine tools. *Robotics and Computer-Integrated Manufacturing*, 4(1-2): 103–119, 2003.

[Pandey & Panda, 2013] R. K. Pandey & S. S. Panda. Drilling of bone : A comprehensive review. *Journal of Clinical Orthopaedics and Trauma*, 4(1): 15–30, 2013.

[Pasquier, 1989] M. Pasquier. *Planification de trajectoires pour un robot manipulateur.* PhD thesis, Instiut National Polytechnique de Grenoble, 1989.

[Perdereau et al., 2002] V. Perdereau, C. Passi, & M. Drouin. Real-time control of redundant robotic manipulators for mobile obstacle avoidance. *Robotics and Autonomous Systems*, 41(1): 41–59, 2002.

[Pourazady & Ho, 1991] M. Pourazady & L. Ho. Collision avoidance control of redundant manipulators. *Mechanism and Machine Theory*, 26(6): 603–611, 1991.

[Renishaw] Renishaw. The Neuromate robot. `http://www.renishaw.com/`.

[Sahbani, 2003] A. Sahbani. *Planification de t âches de manipulation en robotique par des approches probabilistes.* PhD thesis, Université Paul Sabatier de Toulouse, 2003.

[Samson et al., 1991] C. Samson, M. Le Borgne, & B. Espiau. *Robot control : the task function approach.* Oxford engineering science series. Clarendon Press, 1991. ISBN 9780198538059. URL `https://books.google.fr/books?id=-DZSAAAAMAAJ`.

[Sanchez & Latombe, 2002] G. Sanchez & J. C. Latombe. On delaying collision checking in prm planning : Application to multi-robot coordination. *The International Journal of Robotics Research*, 21(1): 5–26, 2002.

[Schrijver, 1998] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998.

[Shukla et al., 2013] A. Shukla, E. Singla, P. Wahi, & B. Dasgupta. A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces. *Robotics and Autonomous Systems*, 61(2): 209–220, 2013.

[Solteiro E. J. et al., 2007] P. Solteiro E. J., O. De Moura P. B., & M. Tenreiro J. A. Manipulator trajectory planning using a moea. *Applied Soft Computing*, 7(3): 659–667, 2007.

[Storn & Price, 1997] R. Storn & K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4): 341–359, 1997.

[Su & Zou, 2012] B. Su & L. Zou. Manipulator trajectory planning based on the algebraic-trigonometric hermite blended interpolation spline. *Procedia Engineering*, 29 (0): 2093–2097, 2012.

[Talbi, 2009] E. G. Talbi. *Metaheuristics : From design to implementation.* Wiley, 2009.

[Tetsuya et al., 1995] N. Tetsuya, M. Shigeyuki, A. Toshiaki, & O. Koshiro. Mechanical properties of human cranium and effect of cranial fractures on extradural hematoma. *Transactions of the Japan Society of Mechanical Engineers Series A*, 61(591): 2386–2392, 1995.

[Tian & Collins, 2004] L. Tian & C. Collins. An effective robot trajectory planning method using a genetic algorithm. *Mechatronics*, 14(5): 455–470, 2004.

[Tsai & Chiou, 1990] M. J Tsai & Y. H Chiou. Manipulabilty of manipulators. *Mechanism and Machine Theory*, 25(5): 575–585, 1990.

[Venegas & Marcial R., 2009] H. A. M. Venegas & J. R. Marcial R. An evolutionary path planner for multiple robot arms. In *Applications of Evolutionary Computing*, vol. 5484, pp. 353–362. Springer Berlin Heidelberg, 2009.

[Wan et al., 2011] Z. Wan, G. Wang, & Y. Lv. A dual-relax penalty function approach for solving nonlinear bilevel programming with linear lower level problem. *Acta Mathematica Scientia*, 31(2): 652–660, 2011.

[Yu & Muller, 1996] J. S. Yu & P. C. Muller. An on-line cartesian space obstacle avoidance scheme for robot arms. *Mathematics and Computers in Simulation*, 41(5–6): 627–637, 1996.

[Yun & Xi, 1996] Wei-Min Yun & Yu-Geng Xi. Optimum motion planning in joint space for robots using genetic algorithms. *Robotics and Autonomous Systems*, 18(4): 373–393, 1996.

[Zhang et al., 2001] L. B. Zhang, L. J. Wang, X. Y. Liu, H. W. Zahon, X. Wang, & H. Y. Luo. Mechanical model for predicting thrust and torque in vibration drilling fibre-reinforced composite materials. *International Journal of Machine Tools and Manufacture*, 41(5): 641–657, 2001.