ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA MAÎTRISE
EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M. Ing.

PAR
Francis BOURBONNAIS

UTILISATION OPTIMALE DE L'ESPACE DE TRAVAIL DES ROBOTS PARALLÈLES EN AFFRONTANT CERTAINS TYPES DE SINGULARITÉS

MONTRÉAL, LE 13 AOÛT 2012

PRÉSENTATION DU JURY

CETTE MÉMOIRE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Pascal Bigras, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Ilian Bonev, codirecteur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Vincent Duchaine, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Maarouf Saad, président du jury
Département de génie électrique à l'École de technologie supérieure

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 16 JUILLET 2012

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

UTILISATION OPTIMALE DE L'ESPACE DE TRAVAIL DES ROBOTS PARALLÈLES EN AFFRONTANT CERTAINS TYPES DE SINGULARITÉS

Francis BOURBONNAIS

RÉSUMÉ

Les robots parallèles prennent de plus en plus d'importance dans l'industrie. On en retrouve des milliers à travers le monde, dont la majorité sont utilisés dans des opérations de transfert rapide. Ils sont plus rapides que les robots sériels, car leurs moteurs sont généralement fixés à la base, allégeant ainsi la partie mobile du manipulateur. La croissance de la popularité de ces robots sur le marché est cependant limitée en raison de leur espace de travail réduit. En effet, l'approche actuellement privilégiée est d'éviter toute singularité, ce qui résulte en un espace de travail très limité.

L'objectif de ce projet est d'augmenter l'espace de travail d'un robot parallèle à cinq barres à l'aide d'une planification de trajectoire qui, d'une part, profite du passage des singularités de type 1 tout en évitant les singularités de type 2, et d'autre part, optimise la trajectoire pour minimiser le temps du parcours. L'algorithme de planification de trajectoire détermine, à partir d'une position (x, y) voulue de l'effecteur terminal, la meilleure des quatre configurations possibles pour l'ensemble des deux bras du robot. Le processus d'optimisation du temps de la trajectoire est réalisé en considérant les contraintes de vitesse et d'accélération maximales ainsi que le couple maximal des moteurs en fonction du modèle dynamique. Un algorithme de commande pour maximiser la réponse dynamique du robot a aussi été conçu. Le modèle dynamique identifié est utilisé pour atteindre les critères de performances visés tout en assurant la précision du suivi. Les algorithmes ont été validés sur un prototype à deux degrés de liberté développé au laboratoire CoRo de l'École de technologie supérieure.

Les résultats obtenus démontrent que l'espace de travail du robot est considérablement augmenté par rapport à un robot industriel d'une taille comparable qui ne change pas de configurations. De plus, le changement de configuration permet d'améliorer le temps de parcours d'une trajectoire.

Ce projet de recherche a donc permis de développer une nouvelle approche démontrant qu'il est possible de travailler de concert avec les singularités de type 1 dans le but de maximiser les performances et l'espace de travail de certains robots parallèles.

Mots-Clés : robots parallèles, singularités, planification de trajectoires, optimisation, rapidité

OPTIMAL USE OF THE WORKSPACE OF PARALLEL ROBOTS BY CROSSING A CERTAIN TYPE OF SINGULARITIES

Francis BOURBONNAIS

ABSTRACT

The popularity of parallel robot has been increasing significantly in industry. There are thousands of them around the world, most frequently used for pick and place. They are faster than serials robots because their actuators are generally fixed to their base reducing the weight of the moving part. However, the growth of the popularity of these robots on the market is slowed down because of their reduced workspace. Indeed, the privileged approach is to avoid all singularities which results in a considerable reduction of the workspace.

The main objective of this project is to increase the workspace of a five-bar parallel robot using path planning. An algorithm for path planning has been developed to take advantage of type 1 singularities in order to avoid type 2 singularities and also to optimize the trajectory by minimizing the cycle time. A control algorithm has also been designed to maximize the dynamic response of the robot. Those algorithms were validated on a two degrees-of-freedom prototype developed in the CoRo laboratory of the École de technologie supérieure. The path planning algorithm determines the best of the four possible configurations for the two arms of the robot, for a desired Cartesian position (x, y) of the end-effector. The optimization process of the cycle time considers several constraints: the maximum velocity, the maximum acceleration, the maximum jerk and also the maximum torque based on the dynamic model of the robot. The dynamic model is also used in the control algorithm to achieve the specified performance criterion while ensuring the accuracy of the robot.

The results show that the workspace of the robot is significantly increased in comparison to a robot of the same size that does not change configurations. Moreover, changing configurations also improves the cycle time. The type 2 singularities are avoided which ensures the rigidity of the robot throughout the workspace and allows it to do fast operations of pick and place where the path between the points is optimized.

This research project has developed a new approach demonstrating that it is possible to work with singularities in order to maximize the dynamic performance and the workspace of this type of robot.

Keywords: parallel robots, singularities, path planning, optimisation, speed

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

A*	Algorithme A étoile
ASRT	Algorithme stochastique de recherche de trajectoires
GUI	Interface utilisateur (« Graphical User Interface »)
MAP	Méthode de programmation approximée
MGD	Modélisation géométrique directe
MGI	Modélisation géométrique inverse
MVC	Courbe de vitesse maximale
PMP	Principe du maximum de Pontriaguine
PNL	Programmation non linéaire
SCA	Splines cubiques aléatoires
WM	Mode de fonctionnement (« working mode »)
AM	Mode d'assemblage (« assembly mode »)

LISTE DES SYMBOLES ET UNITÉS DE MESURE

A_1, A_2	Articulations passives intermédiaires du robot
C	Effecteur terminal du robot
c_{ij}	$\cos\left(q_i + q_j\right)$
cm_{ji}	Coordonnées de la position du centre de masse du lien i par rapport à l'articulation i selon l'axe j
d	Distance entre les bases du robot en mm
$\mathbf{F}\left(\mathbf{q}, \dot{\mathbf{q}}\right)$	Vecteur de force
g	Constante de gravité
$\mathbf{G}(\mathbf{q})$	Vecteur de gravité
I_{ji}	Moment d'inertie principal par rapport au centre de masse du lien i selon l'axe j
$\mathbf{J}_g$	Jacobienne de la contrainte
l_1, l_3	Longueur des liens proximaux en mm
l_2, l_4	Longueur des liens distaux en mm
$\mathbf{M}(\mathbf{q})$	Matrice de masse
m_i	Masse du lien i
N	Nombre de degrés de liberté du robot
O_1, O_2	Articulations actives du robot
q_1, q_3	Positions articulaires des articulations actives
q_2, q_4	Positions articulaires des articulations passives intermédiaires
$\dot{q}_i$	Vitesse articulaire de l'articulation i
$\ddot{q}_i$	Accélération articulaire de l'articulation i
$\dddot{q}_i$	Saccade (« jerk ») articulaire de l'articulation i
s_{ij}	$\sin\left(q_i + q_j\right)$
τ_i	Couple de l'articulation i en Nm

INTRODUCTION

Les robots parallèles prennent de plus en plus d'importance dans l'industrie. On en retrouve des milliers à travers le monde, dont la majorité est utilisée dans des opérations de transfert rapide (« pick and place »). Ils sont plus rapides que les robots sériels, car leurs moteurs sont généralement fixés à la base, ce qui allège la partie mobile (Figure 1). La croissance de ces robots sur le marché est cependant ralentie à cause de leur complexité. De plus, l'approche privilégiée présentement est d'éviter toute singularité ce qui résulte en un espace de travail très réduit par rapport à l'espace des robots sériels. Dans ce mémoire, une approche innovatrice est présentée pour traverser les singularités de type 1 d'un manipulateur parallèle, nommé DexTAR, de façon à maximiser son espace de travail.

Figure 1 Robot sériel (à gauche) et robot parallèle (à droite)
Tirée de Mitsubishi (2012, http://www.mitsubishi-automation.fr/)

Dans le chapitre 1, la problématique générale est présentée en détail. Les différents types de singularités sont abordés ainsi que les notions d'espace de travail. Les différents types de mouvements et les types de modélisations sont aussi expliqués. Ce chapitre est en fait une explication des concepts qui sont traités en détail dans le reste de ce mémoire.

Le chapitre 2 porte sur la revue bibliographique. Deux principaux thèmes sont touchés soit : *i*) les différents moyens pour augmenter l'espace de travail des robots parallèles et *ii*) les méthodes de planification de trajectoire.

Le chapitre 3 contient les éléments qui touchent la modélisation du robot DexTAR (Figure 2). D'abord, l'architecture du robot est présentée. Ensuite, les calculs concernant la modélisation géométrique directe et inverse sont démontrés. Finalement, les calculs pour la modélisation dynamique sont présentés.

Figure 2 Robot DexTAR

Dans le chapitre 4, l'espace de travail du robot est revu en détail. Ce chapitre permet de bien comprendre la différence entre l'espace de travail cartésien et l'espace articulaire. Les calculs pour visualiser les singularités sont présentés et les calculs pour déterminer les interférences mécaniques le sont aussi. Ces calculs servent à introduire l'espace de travail réel du robot.

Dans le chapitre 5, la mise en œuvre du robot est expliquée en détail. Les différentes composantes du système sont détaillées soit : *i*) la mécanique du robot, *ii*) les moteurs, *iii*) le contrôleur et finalement *iv*) l'interface utilisateur.

Le chapitre 6 est sans doute le plus important. Il traite de la planification de trajectoire, un des principaux sujets de ce mémoire. Tout d'abord, le chapitre présente un retour sur les

méthodes proposées dans la revue bibliographique. Ensuite, la stratégie pour générer une trajectoire optimale est dévoilée. Pour ce faire, il est expliqué qu'il faut générer l'environnement. Il faut ensuite déterminer les chemins possibles puis générer les trajectoires pour trouver celle répondant le plus favorablement aux critères d'optimisation. L'algorithme stochastique de recherche de trajectoire qui est utilisé dans une des méthodes retenues est finalement expliqué.

Finalement, le chapitre 7 porte sur les expérimentations et les résultats. En premier lieu, les résultats concernant la planification de trajectoire sont présentés. Plusieurs trajectoires ont été sélectionnées pour démontrer différents aspects traités dans ce mémoire. En second lieu, les résultats concernant l'espace de travail réel du robot sont expliqués et comparés avec un robot industriel disponible sur le marché. Enfin, les performances du contrôleur sont dévoilées.

CHAPITRE 1

PROBLÉMATIQUE GÉNÉRALE

1.1 Introduction

Ce chapitre a pour but d'expliquer brièvement les concepts qui seront développés en détail tout au long de ce mémoire. Il s'agit d'une description sommaire qui permet de comprendre la problématique générale des robots parallèles et de la planification de trajectoires. Les différents types de singularités sont expliqués ainsi que la notion d'espace de travail. Suivront une brève description des types de mouvements que le robot peut effectuer et une description des deux types de modélisation.

1.2 Singularités

Les singularités sont des configurations pour lesquelles le comportement du robot change soudainement. (Zlatanov, Fenton et Benhabib, 1998) ont proposé une classification de tous les types de singularités. Nous allons plutôt utiliser la classification simple proposée dans (Gosselin et Angeles, 1990) :

- singularités de type 1 ou singularités sérielles ;
- singularités de type 2 ou singularités parallèles.

1.2.1 Singularités de type 1

Les singularités de type 1 sont des configurations pour lesquelles l'effecteur terminal du robot perd un ou plusieurs degrés de liberté. Elles correspondent aux frontières de l'espace de travail, c'est-à-dire l'endroit où un des bras atteint sa limite intérieure ou extérieure. La figure 3 montre le bras gauche d'un robot parallèle à cinq barres (comme celui de la figure 1) en singularité de type 1.

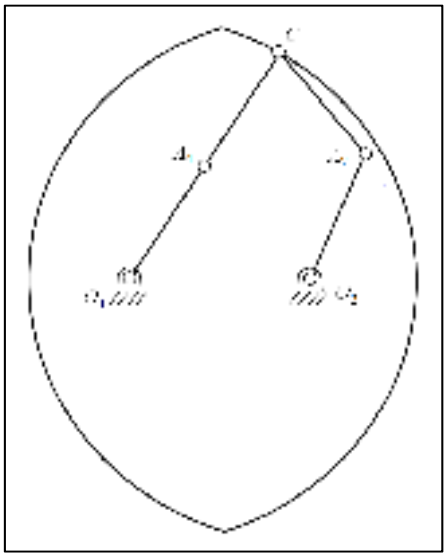


Figure 3 Un exemple de singularité de type 1

1.2.2 Singularités de type 2

Les singularités de type 2 sont des configurations plus complexes où le robot perd sa raideur et gagne un ou plusieurs degrés de liberté. L'effecteur terminal devient alors mobile même si les actionneurs sont bloqués. Les actionneurs, quant à eux, ne peuvent résister à une force ou à un moment appliqué à l'effecteur terminal et le robot n'est plus contrôlable.

Les singularités de type 2 se trouvent à l'intérieur de l'espace de travail, le séparant ainsi en plusieurs zones. Le passage d'une zone à une autre peut se faire en traversant une singularité de type 2. La figure 4 montre deux configurations d'un robot en singularités de type 2. À gauche, les liens A_1C et A_2C sont superposés tandis qu'à droite, ils sont alignés. Dans le cas de la superposition, l'ensemble A_1CA_2 n'est pas contraint et peut tourner librement autour du point A_1. Dans le cas de l'alignement, le robot est incapable de résister à une force normale à la droite A_1A_2. De plus, le robot ne peut pas être dégagé de ses configurations singulières de type 2 par les actionneurs et le mécanisme est complètement bloqué. Pour dégager le robot, une force externe (telle que la gravité) doit donc être utilisée.

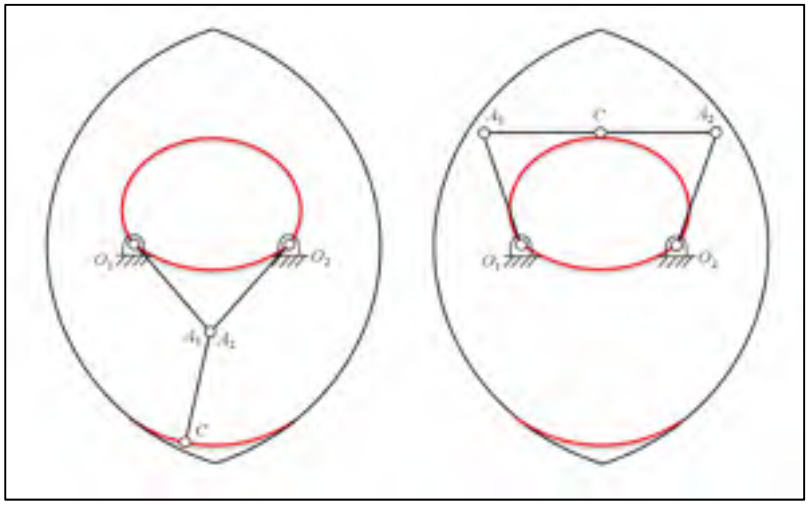


Figure 4 Singularités de type 2

À cause de la nature des singularités de type 2, on limite généralement les déplacements de l'effecteur dans une des zones exemptes de ces singularités. L'espace de travail est alors considérablement réduit. Si on cherche à traverser ces singularités, le robot ne pourra pas exécuter un mouvement continu et précis. Pour des opérations de « pick-and-place », la trajectoire n'a pas besoin d'être précise et il est théoriquement possible de passer les singularités de type 2. Cependant, le contrôle du robot est momentanément perdu au passage de la singularité, ce qui peut ponctuellement entraîner un comportement imprécis. Il est aussi impossible de prendre ou de déposer un objet se trouvant proche d'une courbe de singularité de type 2.

1.3 Espace de travail

L'optimisation de l'espace de travail d'un robot parallèle est un enjeu important. La méthode souvent privilégiée est d'éviter des mouvements où le robot passe par une singularité de type 2 en réduisant l'espace de travail. Ainsi, le contrôle du robot est assuré dans tout l'espace réduit et le risque d'endommager le mécanisme est minimisé. Cependant, cet espace doit être encore réduit pour éviter les interférences mécaniques qui peuvent être relativement complexes pour les robots parallèles. L'espace de travail réel est donc considérablement

réduit comparé à celui d'un robot sériel. Afin d'utiliser l'espace de travail d'un robot parallèle à son plein potentiel, de façon à ce qu'il soit comparable à celui d'un robot sériel, il est possible d'éviter la réduction de l'espace en utilisant des méthodes qui permettent au robot de passer d'une zone à une autre de l'espace grâce au passage par des singularités de type 1.

1.4 Types de mouvements

De façon générale, il existe principalement trois types de mouvement pour accomplir une tâche industrielle :

- Mouvements suivant un chemin prédéfini. Ce sont des mouvements où l'effecteur terminal du robot doit suivre un chemin soumis à des contraintes (par exemple, le soudage en continu ou le découpage). La trajectoire sera déterminée en tenant compte des obstacles à éviter et de la tâche à exécuter. Le robot est constamment en interaction avec l'environnement.

- Mouvements point-à-point prédéfinis (« pick and place »). Dans des opérations de transfert rapide, ce type de mouvement consiste à prendre un objet à un endroit prédéfini et à le déplacer à un autre endroit prédéfini sans se soucier du chemin si ce n'est que pour éviter des obstacles prédéfinis qui pourraient être dans l'espace de travail.

- Mouvements couplés à un système de vision. Ces mouvements sont utilisés lorsque les pièces arrivent sur un convoyeur de façon aléatoire. Un système de vision repère les objets et indique leur position au robot. Celui-ci doit saisir l'objet et le déposer à un autre endroit (souvent prédéfini).

Les deux premiers types de mouvements sont prédéfinis. Le chemin peut donc être calculé au préalable à l'aide d'un algorithme hors ligne. Ceci est très intéressant, car le temps de calcul n'est pas limité de sorte que des méthodes d'optimisation, souvent gourmandes en calculs,

mais très puissantes, peuvent être utilisées. La troisième méthode doit utiliser des algorithmes qui ne donneront pas nécessairement des solutions optimales, faute de temps de calcul suffisamment long, mais donneront des solutions acceptables en temps réel.

Généralement, les trajectoires comportant moins de contraintes offrent plus de possibilités. La planification est alors un plus grand défi, car il y davantage place à l'optimisation. En déterminant le chemin optimal entre deux points, il est possible d'optimiser, selon plusieurs critères comme le temps de parcours, la sollicitation des actionneurs et la fluidité du mouvement. Pour un chemin prédéfini, il est possible d'optimiser le temps et la fluidité, mais les contraintes sont généralement fixées par la tâche, c'est-à-dire, par exemple, que la vitesse est contrainte par le temps nécessaire pour l'application d'un produit.

1.5 Types de modélisations

Lors du processus d'optimisation de la trajectoire, les contraintes liées aux performances maximales du robot doivent être considérées. Tout dépendant de la complexité de la tâche ou du niveau de performances requis, il existe deux types de modélisations :

- modélisation géométrique;
- modélisation dynamique.

La modélisation cinématique se limite à contraindre la trajectoire à ne pas dépasser l'accélération maximale et la vitesse maximale. La vitesse maximale peut être calculée à partir des spécifications des actionneurs et du ratio de transmission alors que l'accélération maximale peut être approximée par le ratio du couple maximal des actionneurs par rapport à l'inertie maximale des articulations. Ces approximations engendrent souvent des bornes maximales inférieures à la réalité. Généralement, on utilise la modélisation cinématique dans des tâches où il faut éviter des obstacles. Pour ce faire, un algorithme pour trouver le chemin le plus court peut être couplé à un algorithme qui détermine le temps minimum de la

trajectoire. Le problème de planification de trajectoire est alors beaucoup plus simple à traiter puisqu'il s'exprime sous forme linéaire.

La modélisation dynamique incorpore au modèle les caractéristiques inertielles comme la masse, le moment d'inertie et le centre de gravité des composantes en mouvement. Les forces générées par les mouvements sont alors transformées en couples qui dépendent de la position, de la vitesse et de l'accélération du robot. Il s'agit donc d'une modélisation beaucoup plus réaliste du comportement du robot qui permet d'obtenir des performances supérieures. On peut utiliser un algorithme pour trouver le chemin le plus court ou utiliser un algorithme qui trouvera le chemin optimal en considérant l'énergie ou la sollicitation des actionneurs par exemple. Un autre algorithme optimisera la trajectoire selon plusieurs critères possibles. Le problème de planification de trajectoire avec un modèle dynamique est plus complexe puisqu'il faut tout d'abord bien définir ce modèle et le valider pour ensuite traiter le problème d'optimisation de trajectoire, qui sera généralement non-linéaire.

1.6 Conclusion

Ce chapitre a introduit les éléments clés qui sont traités dans les chapitres suivants. Il s'agit d'une brève introduction pour présenter les enjeux lors de la conception des robots parallèles et lors de la planification de trajectoires. Ces concepts sont élaborés plus en détail dans ce mémoire en présentant les calculs qui s'y rattachent.

CHAPITRE 2

REVUE DE BIBLIOGRAPHIE

1.7 Augmentation de l'espace de travail des robots parallèles

Lors de la conception d'un robot parallèle, l'étude de son espace de travail, c'est-à-dire l'espace atteignable par l'effecteur du robot, et son optimisation sont des facteurs importants pour accroître ses performances de déplacement. Un des principaux inconvénients d'un robot parallèle par rapport à un robot sériel est son enveloppe de travail réduite. C'est pourquoi une étude complète de l'architecture et de l'utilisation de diverses stratégies pour augmenter l'espace de travail est de mise. Il existe plusieurs approches pour augmenter l'espace de travail :

- optimisation du manipulateur;
- franchissement des singularités de type 2;
- redondance cinématique du manipulateur;
- accouplement d'articulations;
- changement de mode de fonctionnement.

Cette section présente ces approches et en fait un bilan. Chaque approche est résumée et ses avantages ainsi que ses inconvénients sont expliqués.

1.7.1 Optimisation du manipulateur

La conception d'un manipulateur passe par une étude de celui-ci et son optimisation pour satisfaire les critères de performances désirés. On cherche alors à déterminer les paramètres du robot pour que celui-ci satisfasse divers critères. Plusieurs méthodes utilisent l'espace de travail et les propriétés cinématiques comme critère de conception. Arsenault et Boudreau (Arsenault et Boudreau, 2004) ont aussi considéré d'autres critères. Leur but était d'obtenir

un mécanisme optimal pour une tâche définie en considérant les singularités combinées avec l'optimisation de l'espace de travail et la maximisation de la dextérité[1] du mécanisme. Dû à la complexité du problème, ils ont utilisé un algorithme génétique dans le processus d'optimisation. D'autres auteurs (Liu, Wang et Pritschow, 2006) ont réalisé des processus d'optimisation complexe dans le but d'optimiser un robot à cinq barres en considérant comme critères la vitesse de déplacement moyenne, la charge à déplacer et la raideur.

La conception d'un robot parallèle est beaucoup moins intuitive que celle d'un robot sériel : le nombre de paramètres à considérer est beaucoup plus grand et les singularités ainsi que leur évitement sont plus complexes. Les critères de performance désirés sont souvent contradictoires. La majorité des travaux qui cherchent uniquement à optimiser le manipulateur le font de sorte qu'aucune singularité ne soit franchie. Cela a pour conséquence que l'espace opérationnel est inférieur à l'espace de travail géométrique.

1.7.2 Franchissement de singularités de type 2

En franchissant une singularité de type 2, le robot change de mode d'assemblage[2]. Ainsi, l'espace de travail peut être augmenté en comparaison avec une stratégie qui ne fait pas ces passages de singularité. Par contre, le robot perd de la raideur lors de l'approche d'une singularité de type 2. Il devient alors difficile à commander. Il perd aussi de la dextérité. Ainsi, il ne pourra pas suivre une trajectoire de façon précise. Il ne pourra pas non plus s'opposer adéquatement à certaine force extérieure. Yasuda et al. (2000) se sont intéressés à utiliser une force extérieure comme l'inertie ou le poids pour traverser ce type de singularités. Par exemple, on peut faire accélérer le robot à l'approche d'une singularité de type 2 de façon à ce que le robot puisse passer la singularité grâce au mouvement inertiel. Durant le passage, la commande des actionneurs est interrompue puis réactivée. Cette technique demeure cependant complexe, incertaine et difficile à réaliser.

---

[1] Dextérité : habilité à exécuter un petit déplacement de l'effecteur dans une position spécifique.
[2] Mode d'assemblage : une des solutions possibles au modèle géométrique directe.

1.7.3 Redondance cinématique

La redondance cinématique consiste à ajouter un degré de liberté à une chaine cinématique d'un manipulateur non redondant. Ce degré de liberté peut être obtenu en remplaçant une articulation passive par une articulation active. Une articulation active est une articulation motorisée tandis qu'une articulation passive ne l'est pas. Un nouveau lien entre la base du robot et l'effecteur peut aussi être ajouté pour gagner un degré de liberté. La redondance cinématique a plusieurs avantages : elle ne compromet pas la mobilité du robot, elle permet d'augmenter sa dextérité, elle permet de répartir une partie des forces sur les actionneurs supplémentaires pour réduire la sollicitation de l'ensemble des actionneurs et finalement, et elle permet d'éviter les singularités de type 2 dans l'espace de travail. Ce dernier avantage, qui touche directement l'optimisation de l'espace de travail, a été expérimenté par Wang et Gosselin (Wang et Gosselin, 2004). Ils ont démontré que l'ajout d'un actionneur permet de réduire considérablement les singularités à l'intérieur de l'espace de travail, permettant par conséquent de l'augmenter.

L'utilisation d'un actionneur redondant paraît une bonne solution dans le but d'augmenter l'espace de travail. Cependant, il faut considérer le coût supplémentaire associé à l'ajout d'un actionneur. De surcroît, l'ajout de poids dans la partie mobile du robot entraîne une réduction des performances dynamiques. Bien que la redondance cinématique permette d'augmenter l'espace de travail, elle introduit une complexité supplémentaire ainsi qu'une augmentation des coûts et possiblement une réduction des performances.

1.7.4 Accouplement d'articulations

L'accouplement d'articulations (« joint-coupling ») consiste à commander plusieurs articulations avec un seul actionneur sans perdre de mobilité. Cela nécessite de synchroniser les déplacements des articulations actives et passives pour en faire un ensemble d'articulations actives, ce qui a pour effet de modifier les conditions de singularités de type 2 en intégrant un coefficient de liaison (Chen, Angeles et Li, 2003). Une méthode

d'optimisation non linéaire peut être utilisée pour déterminer ce coefficient. Ainsi, un mouvement pourra être réalisé sans traverser de singularités de type 2. L'espace de travail est alors augmenté.

Cette stratégie demeure complexe à réaliser. Elle demande une analyse poussée de l'architecture et elle engendre un coût supplémentaire. De plus, les singularités pourront être évitées dans certains cas, mais l'espace de travail demeure divisé en plusieurs zones.

1.7.5 Changement de mode de fonctionnement

Les modes de fonctionnement correspondent aux solutions du modèle géométrique inverse (MGI). Un robot parallèle a généralement plusieurs solutions à son MGI donc plusieurs configurations possibles pour atteindre la même pose avec son effecteur. La figure 5 montre les quatre modes de fonctionnement pour un robot à cinq barres (*RRRRR*). Pour chacun des bras, il peut y avoir deux configurations possibles. Ces configurations diffèrent par le signe de l'angle de l'articulation passive. Le mode de fonctionnement est déterminé par l'ensemble des deux bras et il est défini sur la figure 5.

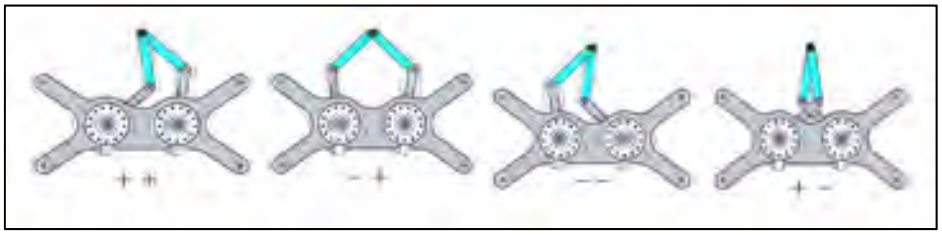


Figure 5 Modes de fonctionnement

Les parcours des singularités de type 2 sont différents pour chacun des modes de fonctionnement (Bonev et Gosselin, 2001). Ainsi, en changeant de mode de fonctionnement, le robot peut atteindre une zone de l'espace de travail qui était inaccessible. L'espace de travail du robot peut ainsi être augmenté. Par exemple, aux figures 3 et 6, deux architectures robotiques différentes sont présentées. Les dimensions de ces deux manipulateurs sont

cependant du même ordre de grandeur (en extension, leurs bras ont la même longueur). À gauche, le robot peut atteindre tout l'espace coloré en changeant de mode de fonctionnement et en considérant les interférences mécaniques. À droite, la zone grise représente l'espace de travail atteignable par le robot sans changer de mode de fonctionnement et en considérant les interférences mécaniques (Figielski, Bonev et Bigras, 2007). L'espace de travail est donc considérablement augmenté grâce aux changements de modes de fonctionnement.

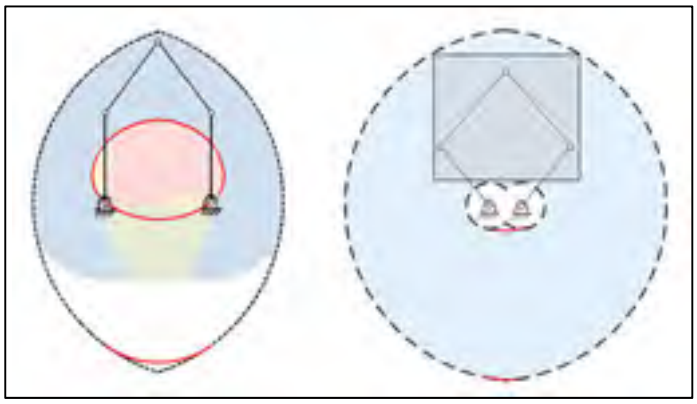


Figure 6 Comparaisons des espaces de travail

Comme le montre la figure 7, pour changer de monde de fonctionnement, le robot doit passer par une singularité de type 1. À cette configuration, un des bras du robot est aligné ou replié sur lui-même. Le passage d'une singularité de type 1 se fait naturellement dans l'espace articulaire tandis qu'au contraire, le passage des singularités de type 2 (indiqué par les courbes rouges sur la figure 7) demande une stratégie particulière puisque les articulations sont bloquées à l'instant du passage.

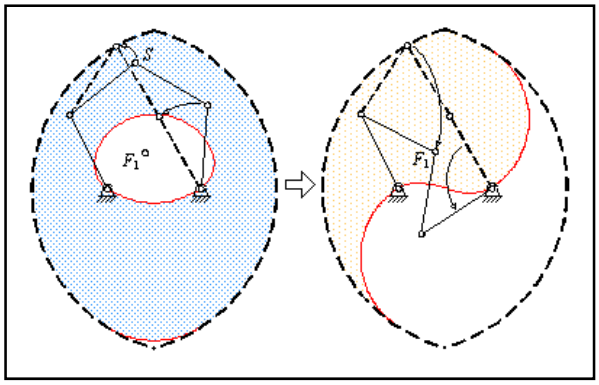


Figure 7 Passage d'une singularité de type 1

Le fait d'avoir plusieurs configurations possibles pour une même position cartésienne de l'effecteur a aussi d'autres avantages. Le mode de fonctionnement optimal pour une position donnée (de l'effecteur) peut être déterminé selon différents critères tels que le temps de déplacement, l'énergie nécessaire ou encore la raideur du robot (Bonev, 2003). Le robot pourra également éviter de passer près d'une singularité de type 2 ou encore utiliser le changement de mode de fonctionnement pour effectuer un trajet plus court. Cependant, l'algorithme de planification de trajectoire devra considérer beaucoup de paramètres pour trouver la configuration optimale.

1.7.6 Conclusion

Les stratégies pour augmenter l'espace de travail présentées dans cette section ont chacune leurs avantages et leurs inconvénients. L'optimisation du manipulateur est certes essentielle, mais il est possible de tirer un gain d'espace de travail encore plus important en couplant cette méthode avec d'autres. Le franchissement de singularité de type 2 est complexe et ne permet pas de commander précisément ni de conserver la raideur du robot dans tout l'espace. La redondance cinématique est intéressante, mais elle ampute les performances du robot et elle nécessite un coût supplémentaire. De même, l'accouplement d'articulations est complexe et couteux.

Finalement, l'idée de changer de mode de fonctionnement est une méthode novatrice. Elle permet un accroissement significatif de l'espace de travail sans engendrer un coût supplémentaire. Cette méthode permet aussi d'optimiser les performances du robot en tirant parti des modes de fonctionnement pour réaliser des déplacements plus rapides ou en considérant d'autres critères. Par contre, la planification de trajectoire est plus laborieuse, mais elle demeure relativement simple si elle est réalisée dans l'espace articulaire.

1.8 Planification de trajectoire

L'utilisation des robots dans un contexte industriel a pour but d'accroître la productivité de tâches répétitives. Cependant, pour exploiter les performances du robot au maximum et ainsi avoir un temps de cycle optimal, les trajectoires doivent être choisies judicieusement. Un mouvement non optimal répété pendant plusieurs milliers de cycles a des répercussions importantes sur l'efficacité du système. Les mouvements doivent aussi être de qualité pour éviter de trop solliciter inutilement les actionneurs. Cela aura pour effet d'augmenter leur durée de vie et d'abaisser les coûts de maintenance.

Le problème de planification de trajectoire est largement traité dans la littérature. Certaines méthodes ont été retenues pour leur efficacité ou pour leur simplicité de mise en œuvre. Ces méthodes seront décrites dans la prochaine section.

1.8.1 Stratégies pour trouver le chemin le plus court

Dijkstra

Cet algorithme permet de se rendre du point initial au point final en connaissant l'environnement. Il a été introduit par Dijkstra (Dijkstra, 1959). L'environnement est discrétisé en pixels de dimension donnée appelés des nœuds. Ces nœuds sont connectés par des liens ayant un coût défini. Le coût peut varier en fonction de la longueur entre les nœuds ou d'autres critères.

Pour lancer l'algorithme, il faut une carte de l'environnement. Chaque point ou pixel qui détermine la résolution de la discrétisation correspond à un nœud. Lorsqu'il y a un obstacle, il n'y a pas de nœud.

L'algorithme se déroule en deux parties. Tout d'abord, à partir du nœud de départ, toutes les possibilités de chemin vers le nœud final sont calculées. Ensuite, à partir de la liste des chemins possibles, il faut trouver celui qui a le coût le plus petit. Cet algorithme a été utilisé par Vigneron (Vigneron, Kallel et Maaref, 2006) pour calculer la trajectoire optimale pour un robot circulaire dans un environnement encombré. Il est couplé notamment avec un graphe de connectivité pour construire le réseau routier (l'environnement). Il a aussi été utilisé pour un robot mobile par Kenzai (Kenzai, 2005).

Algorithme génétique

Les algorithmes génétiques permettent de trouver une solution quasi optimale à divers problèmes d'optimisation telle que la recherche du chemin le plus court (Farritor et Dubowsky, 2002). Ils appartiennent à la famille des algorithmes évolutionnistes et utilisent la théorie de la sélection naturelle développée par Darwin. Prenons par exemple un nombre de points qui constituent la trajectoire entre un point initial et un point final. Chacun de ces points constituera un chromosome, ou un individu, et sera formé de gêne. L'ensemble des trajectoires possibles constituera la population. La population correspond à des solutions potentielles au problème.

Les chromosomes subiront plusieurs opérations pour former une nouvelle génération d'individus. Les opérations sont la sélection, la recombinaison et la mutation. La sélection évalue chaque individu et conserve ceux qui donnent les meilleurs résultats. La recombinaison prend deux individus sélectionnés et échange des gênes pour former des nouveaux individus. Finalement, la mutation permet d'éviter la convergence de l'algorithme à un minimum local. Il s'agit de l'échange d'un gène d'un chromosome à un autre de façon aléatoire. Ce processus terminera lorsque la solution sera jugée suffisamment optimisée.

L'optimisation ne fournit pas nécessairement la solution optimale, mais augmente les chances de s'y rapprocher. Il s'agit d'une méthode générale qui s'applique sur des fonctions qui ne sont pas nécessairement lisses. Cette méthode a été utilisée dans la recherche du chemin le plus court par de nombreux auteurs, par exemple par Farritor et Dubowsky (Farritor et Dubowsky, 2002). Cependant, les taux de mutation et de recombinaison doivent être bien ajustés et ne seront pas nécessairement les mêmes pour chacune des trajectoires. Ainsi, on peut coupler l'algorithme génétique avec un réseau de neurones pour réévaluer les paramètres, ce qui allonge le temps d'exécution et complexifie la mise en œuvre. De plus, il n'est pas garanti que les mouvements soient lisses, ce qui peut engendrer un suivi de trajectoire difficile.

Champs de potentiel

La méthode des champs de potentiel est très repandue dans la littérature. Cette méthode est analogue au magnétisme qui fait qu'une particule peut être attirée ou repoussée d'un élément selon sa polarité. Basé sur ce principe, un robot peut être contrôlé par des champs de potentiel artificiels. Le point final possède un champ de potentiel attractif alors que les obstacles ont un champ répulsif. De cette façon, le robot part de sa position initiale et il est attiré naturellement par la position finale. Il sera naturellement dévié s'il rencontre des obstacles sur son parcours. Cette méthode a été introduite par Khatib (Khatib, 1986) et a été reprise par plusieurs auteurs tels que Feder (Feder et Slotine, 1997) et Lee (Lee et al., 1997).

Cette approche s'avère très performante pour déterminer la trajectoire de robots mobiles ou de manipulateurs pour une application en temps réel. Cependant, elle compose avec un inconvénient important : les minimums locaux. Ces minimums locaux sont dus à la quantité d'obstacles, leur disposition dans l'espace et leur force de répulsion. La particule qui représente l'effecteur du robot peut se bloquer ou osciller dans un minimum local. Les minimums locaux peuvent être contrés par l'application de mouvements aléatoires (Barraquand et Latombe, 1991) ou encore par l'application d'une force supplémentaire qui dépend des forces d'attraction ou de répulsion (Vadakkepat, Tan et Ming-Liang, 2000).

Certains auteurs comme Feder et Soltine (Feder et Slotine, 1997) ont repris les mêmes notions, mais en appliquant les formules de la mécanique des fluides pour supprimer les minimums locaux comme l'eau d'une rivière qui ne s'arrête pas devant un rocher, mais qui finit par le contourner.

La méthode des champs de potentiel a certes beaucoup d'avantages par sa facilité de mise en œuvre et sa simplicité conceptuelle. Lorsque des robots parallèles sont considérés, les champs de potentiels peuvent être ajustés pour que le robot évite certains obstacles comme les singularités de type 2. Cependant, cette approche demeure complexe lorsque des algorithmes pour éviter des minimums locaux sont ajoutés. De plus, les calculs de champs répulsifs pour des cercles, des ellipses et d'autres formes triviales sont simples, mais le fait de négocier avec des obstacles de formes irrégulières complexifie considérablement le calcul de ces champs.

L'algorithme A*

L'algorithme A* a pour but de trouver le chemin le plus court entre deux points en évitant des obstacles. Il s'agit d'un algorithme qui ne trouve pas nécessairement la solution optimale, mais qui est en mesure de donner une solution rapidement. Pour y arriver, une carte discrétisée de l'environnement ou un graphe de connectivité doit être fourni à l'algorithme. L'algorithme tente à chaque itération de se rapprocher du point final, c'est-à-dire qu'il ne considère pas les solutions qui s'éloignent de la solution. En ce sens, il est plus performant que Dijkstra. Il mémorise tous les points parcourus au cas où le chemin direct est impossible à atteindre à cause d'obstacles.

Pour déterminer la meilleure solution, un coût doit être déterminé à chaque point. Le coût peut prendre différentes formes selon le critère à optimiser. Cet algorithme est très efficace et permet toujours de trouver une solution, si elle existe, peu importe le nombre d'obstacles.

Par sa simplicité de mise en œuvre, l'algorithme A* est très intéressant. Il nécessite peu de calculs et converge rapidement vers une solution. Cet algorithme peut trouver un chemin faisable de la position de départ vers la cible. Toute la documentation sur sa mise en œuvre est disponible en ligne (Schwartz, 2006).

Graphe de connectivité – Algorithme SIVIA

Pour un environnement donné, il est possible de construire un graphe de connectivité qui permet de simplifier l'espace de recherche en un graphique de points et d'arcs. L'environnement correspond ici à un pavé parsemé d'obstacles. Ce pavé est séparé en sous-pavés qui ne contiennent pas d'obstacles. La résolution minimale du sous-pavé est déterminée avant les calculs. La figure 8 montre un exemple de sous-pavés. Une fois ce sous-pavé obtenu, il est possible de constituer un graphe de connectivité. Le graphe présenté à la figure 9 représente le lien entre les sous-pavés. La coordonnée du centre du sous-pavage représente son repère. Ensuite, les cases qui sont collées à celle-ci sont déterminées et leurs distances avec le repère du sous-pavé sont calculées. Lorsqu'il y a contact entre les cases, la distance entre le centre de celle-ci est placée dans une matrice d'adjacence. L'algorithme Sivia (Jaulin et al., 2001) est utilisé pour calculer les sous-pavés.

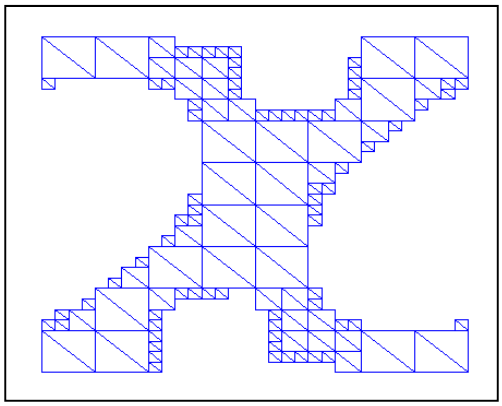


Figure 8 Pavés de l'environnement

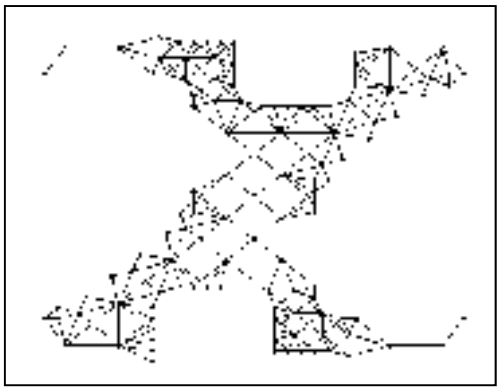


Figure 9 Graphe de connectivité

L'avantage d'un graphe de connectivité versus une carte discrétisée est le nombre d'éléments à traiter dans la recherche du chemin le plus court. Avec A* et encore plus avec Dijikstra, le nombre d'éléments (de nœuds) qui constituent l'environnement a un impact direct sur le temps de calcul. Le nombre de nœuds est inférieur pour un graphe de connectivité versus une carte discrétisée par pixel.

Conclusion

Dans cette section, plusieurs algorithmes de recherche du chemin le plus court ont été présentés. Tous ces algorithmes sont efficaces, mais certains se caractérisent par leur simplicité de mise en œuvre.

Premièrement, la méthode des champs de potentiel est complexe lorsqu'il y a des obstacles de formes irrégulières. De plus, le fait d'avoir des obstacles irréguliers augmente les possibilités de minimums locaux. Il est donc absolument nécessaire d'incorporer un autre algorithme pour éviter les minimums locaux.

Deuxièmement, la mise en œuvre d'un algorithme génétique est complexe. S'il faut en plus le coupler à un réseau de neurones pour ajuster les paramètres de mutation, la complexité en est qu'augmentée.

Finalement, les algorithmes Dijikstra et A* sont similaires. Les deux utilisent une carte discrétisée de l'environnement pour déterminer le chemin le plus court. Par contre, l'algorithme A* est un peu plus complexe à mettre en œuvre, mais beaucoup plus performant. Dans les deux cas, il est essentiel d'utiliser un graphe de connectivité comme carte de l'environnement pour maximiser le temps de convergence de l'algorithme vers une solution.

1.8.2 Planification de trajectoire d'un chemin imposé

La planification de trajectoire d'un chemin imposé se fait nécessairement après le calcul du chemin. Il faut donc préalablement déterminer le chemin. Pour ce faire, les méthodes de la section précédente peuvent être utilisées. Par la suite, les algorithmes présentés dans cette section pourront être utilisés pour générer la trajectoire.

Programmation non linéaire (PNL)

La programmation non linéaire optimise le temps du parcours. Elle a été introduite par Lin, Chang et Luh (Lin, Chang et Luh, 1983) et reprise plus tard par Cao, Doods et Irwin (Cao, Doods et Irwin, 1994) ainsi que par Tse et Wang (Tse et Wang, 1998). Le principe est simple : à partir d'une trajectoire définie par un point initial, un point final et quelques points intermédiaires, l'algorithme cherche à optimiser le temps entre les points en satisfaisant les contraintes.

La quantité des points intermédiaires a un impact direct sur le temps de convergence de l'algorithme. Cette méthode utilise les fonctions spline de degré 3. Pour un ensemble de points formant la trajectoire, elle calcule des segments de fonctions spline de degré 3 entre les points. L'idée est d'optimiser le temps pour exécuter un segment sous contraintes pour ainsi optimiser le temps total de la trajectoire. Les contraintes ici sont la vitesse et l'accélération maximales des articulations. L'algorithme présenté par Lin, Chang et Luh (Lin, Chang et Luh, 1983) pour la résolution est complexe, mais Gasparetto et Zanotto (Gasparetto

et Zanotto, 2008) le simplifient en proposant l'utilisation d'une méthode de minimisation non linéaire multivariable.

À cet algorithme de minimisation du temps, certains auteurs ajouteront d'autres critères tels que la minimisation de l'énergie et de la saccade (Gasparetto et Zanotto, 2008). Cette méthode a aussi été appliquée à des robots parallèles (Hu, Zhang et Zhan, 2008). Cet algorithme est simple à mettre en œuvre et il permet d'obtenir un résultat rapidement. Cependant, si les contraintes ne sont pas bien déterminées, il ne convergera pas vers une solution optimale. De plus, l'algorithme ne considère que les contraintes cinématiques. Il pourrait cependant être adapté pour inclure le modèle dynamique qui serait discrétisé tout au long de la trajectoire.

Courbe de vitesse maximale (MVC)

Lorsqu'on s'intéresse au mouvement le long d'un chemin prédéfini, plusieurs auteurs se sont intéressés à la technique de plan de phase ou de « courbe de vitesse maximale » (MVC). Cette méthode n'utilise pas de fonctions d'optimisation à proprement dite, mais une étude particulière de la trajectoire pour déterminer la vitesse maximale à chaque point sur le chemin dans l'espace cartésien. La planification de trajectoire consiste alors à optimiser un seul paramètre qui est l'abscisse curviligne. Cela résulte en une courbe de vitesse maximale qui ne doit jamais être dépassée pour assurer le maintien des contraintes. Avec cette approche, le modèle dynamique du robot est considéré. L'idée a tout d'abord été proposée par Bobrow (Bobrow, Dubowsky et Gibson, 1985) puis par Pfeiffer (Pfeiffer et Johanni, 1987) pour ensuite être reprise par de nombreux auteurs (Zlajpah, 1996), (Shiller et Lu, 1992) et (Costantinescu et Croft, 2000). Ensuite, cette méthode a été adaptée aux robots parallèles (Abdellatif et Heimann, 2005), (Pietsch et al., 2003).

La première étape de l'algorithme consiste à déterminer la courbe de limitation définie par les contraintes. Pour y parvenir, il faut discrétiser la trajectoire. À ce moment, le temps de calcul dépend directement de la résolution de la discrétisation. Par la suite, il faut construire

la courbe de limite de vitesse. Les diverses stratégies pour construire cette courbe sont complexes. La courbe, dans certains cas, sera discontinue et conduira à des variations brusques dans les courbes de vitesse et d'accélération articulaire. Des algorithmes permettant d'éviter ces variations ont cependant été proposés (Pietsch et al., 2003).

Conclusion

L'algorithme de programmation non linéaire est simple à implanter. Il nécessite un nombre raisonnable de calculs et permet d'obtenir une solution rapidement. Son inconvénient majeur est qu'il ne considère pas le modèle dynamique. L'algorithme de la courbe de vitesse maximale est pour sa part beaucoup plus complexe. Il peut demander beaucoup de temps de calcul pour déterminer la trajectoire. De plus, il calcule la trajectoire uniquement dans l'espace cartésien.

1.8.3 Planification de trajectoire d'un chemin point-à-point

La planification de trajectoire d'un chemin point-à-point a la particularité que le chemin entre le point initial et final n'a pas d'importance. Dans cette section, des techniques pour calculer un chemin point-à-point seront présentées en considérant ou non les obstacles.

Techniques pour une ligne droite dans l'espace articulaire

La méthode la plus simple pour calculer une trajectoire d'un robot à N degrés de liberté consiste en une ligne droite dans l'espace articulaire. Ce type de planification de trajectoire est soumis à un seul type de contraintes cinématiques, soit l'accélération et la vitesse maximale. Il existe plusieurs méthodes pour calculer ce type de trajectoire dans l'espace articulaire. Par leur simplicité de calcul, ces méthodes peuvent être calculées en temps réel. Ils ne tiennent cependant pas compte du modèle dynamique et ils ne peuvent pas prendre en compte des points intermédiaires (Dombre et Khalil, 2007).

Par exemple, il est possible de calculer la trajectoire à l'aide d'un modèle polynomial de 5e ordre. Ce modèle assure une trajectoire lisse qui est continue au niveau de sa position, sa vitesse et son accélération. Le temps de la trajectoire *T* est alors à minimiser sans dépasser les contraintes cinématiques. Il faut calculer le temps minimum pour chaque articulation et ensuite utiliser le plus grand temps minimum pour calculer la trajectoire de chaque articulation. Une autre possibilité est d'utiliser un modèle de vitesse trapézoïdale. La trajectoire générée par ce modèle sera cependant discontinue au niveau de l'accélération (de type bang-bang). Le dernier modèle proposé par Dombre et Khalil (2007) est un modèle de vitesse trapézoïdale lissée. Celui-ci est calculé pratiquement de la même façon que le modèle précédent, mais il permet d'obtenir une accélération continue qui aura un profil polynomial de second degré. Cela réduit donc les variations brusques du contrôleur lors du suivi de la trajectoire.

Les fondements de ces méthodes polynomiales simples sont largement traités dans la littérature (Craig, 1989). Ils ne permettent cependant pas de considérer les contraintes dynamiques du manipulateur et les obstacles.

Principe du maximum de Pontriaguine

Le principe du maximum de Pontriaguine (PMP) est utilisé dans les problèmes de commande optimale pour changer l'état d'un système dynamique en respectant des contraintes. Transposé à la planification de trajectoire, ce principe permet de planifier la trajectoire libre point-à-point en considérant les contraintes cinématiques et dynamiques qui sont prises en compte directement dans les équations. Le modèle dynamique doit être représenté en modèle d'état qui est interprété comme un système d'équations différentielles non linéaires de premier ordre.

Cette méthode théorique devient complexe lors de la prise en compte des contraintes. De plus, le PMP ne traite pas directement les obstacles bien que certains auteurs aient proposé des alternatives (Galicki, 1998). La convergence n'est pas assurée s'il n'y a pas de procédure

d'initialisation précise. Le PMP est aussi utilisé par Chen, Chien et Desrochers (1992) pour planifier une trajectoire définie.

Programmation non linéaire et discrétisation du modèle dynamique

Cette méthode utilise une modélisation discrète de la dynamique du robot pour ensuite rechercher la trajectoire optimale par une méthode d'optimisation non linéaire. La méthode proposée a été basée initialement sur une technique semblable à la programmation non linéaire pour un chemin imposé. Flash et Potts (1988) ont aussi proposé un planificateur de trajectoire dans l'espace des articulations pour un mouvement point à point en utilisant une méthode de programmation linéaire. Le modèle dynamique n'est cependant pas pris en compte.

Tan et Potts (1988) proposent quant à eux une méthode utilisant la programmation non linéaire et le modèle dynamique discret du robot formulé par Neuman et Tourassis (1985). Dans cette méthode, un intervalle de temps qui correspond au temps nécessaire pour que le robot passe de la configuration initiale à la configuration finale est utilisé. Cet intervalle est séparé en sous-intervalles correspondant à des nœuds où les coordonnées articulaires seront déterminées durant le processus d'optimisation.

Cette méthode est intéressante, car elle fournit une solution presque optimale. Cependant, le temps de calcul peut être laborieux puisqu'il est directement lié au nombre de nœuds de la trajectoire qui, pour sa part, détermine la précision de l'optimisation. Il faut alors trouver un compromis entre la précision et la rapidité de calcul. Finalement, les méthodes de programmation non linéaires peuvent fournir des résultats correspondant à un minimum local et peuvent même ne pas converger dans certaines situations.

Champs de potentiel

La méthode des champs de potentiel qui a été introduite dans la recherche du chemin le plus court peut être adaptée à la planification de trajectoire point-à-point. Certains auteurs (Mohri, Yang et Yamamoto, 1995) et (Galicki, 1992) y ont aussi intégré les équations du modèle dynamique. Ces équations sont exprimées par rapport à un paramètre s qui correspond à la position normalisée sur la trajectoire (s = [0,1]). La trajectoire initiale sera déterminée en considérant uniquement les obstacles. Ensuite, en utilisant les expressions de la dynamique, la trajectoire sera déformée tout en s'assurant que les obstacles sont évités.

Cette méthode minimise seulement le temps de la trajectoire. Elle a les mêmes inconvénients que lorsqu'elle est utilisée pour la recherche du chemin le plus court, soit que l'algorithme peut converger vers un minimum local. Il faut alors un algorithme particulier pour éviter les minimums locaux. De plus, pour des obstacles de formes irrégulières, la mise en équations de champs potentiels est plus complexe et la probabilité de converger vers un minimum local accentuée.

Splines cubiques aléatoires (SCA)

Cette méthode récente, proposée par Chettibi (Chettibi et Lehtihet, 2002) permet d'optimiser une trajectoire en considérant différents critères comme un temps minimal ou une pondération entre le temps minimal et l'effort quadratique moyen. De façon générale, un grand nombre de trajectoires aléatoires doit être généré par le biais de méthode stochastique puis la fonction objective doit être calculée. Cette fonction objective est minimisée en conservant toujours la trajectoire qui a un coût inférieur à la meilleure trajectoire sauvegardée. La vitesse, l'accélération et la saccade peuvent être contraintes. L'algorithme peut aussi considérer les obstacles dans l'espace en utilisant une carte discrétisée de l'environnement. Cette méthode utilise en plus le modèle dynamique du robot pour contraindre le couple des actionneurs. L'idée de l'algorithme est de dissocier le temps total T

de la fonction objective. Ainsi, la fonction objective est représentée par une courbe où le minimum correspond à la valeur optimale pour la trajectoire.

Cette méthode évolue donc selon deux boucles. La première boucle détermine de façon croissante le nombre de points sur la trajectoire et elle calcule l'intervalle de recherche pour chaque point. Ensuite, une autre boucle génère aléatoirement des trajectoires jusqu'à ce que la fonction objective ne soit pas améliorée pour un certain nombre d'itérations. À l'intérieur de cette boucle, la fonction objective est calculée pour vérifier si la trajectoire actuelle est meilleure que la meilleure trajectoire sauvegardée.

Chettibi présente plusieurs articles sur la méthode qu'il a conçue en collaboration avec Lehtihet. Cette méthode peut être couplée avec un algorithme stochastique pour générer les trajectoires à tester (Chettibi et al., 2006). Un algorithme génétique peut aussi être utilisé pour tester les différentes possibilités (Chettibi, 2006). L'algorithme peut également être adapté aux robots parallèles (Chettibi et al., 2005) et aux robots industriels qui doivent éviter des obstacles (Chettibi et al., 2004). Il peut aussi être appliqué aux robots mobiles (Haddad et al., 2007). Finalement, pour un espace de travail encombré d'obstacles, un algorithme qui calcule le chemin le plus court entre les points est proposé dans (Hentout et al., 2010). Cet algorithme est exécuté avant celui de minimisation de la fonction objective pour donner une forme initiale au chemin. Cette méthode est également présentée dans le livre de Dombre et Khalil (Dombre et Khalil, 2007).

La méthode des splines cubiques aléatoires utilise le modèle dynamique du robot sous sa forme standard. Ainsi, elle peut traiter tout type d'architecture ayant un grand nombre de degrés de liberté sans ajouter de complexité au problème. Si l'algorithme de génération de trajectoire qui est utilisé en prétraitement est efficace, l'algorithme permet de traiter toutes les possibilités de trajectoire sans converger vers un minimum local. La méthode traite aussi directement les contraintes cinématiques et la gestion des obstacles ne constitue pas une complexité supplémentaire. Les paramètres de l'algorithme doivent cependant être bien

définis pour assurer un bon fonctionnement, mais ceux-ci restent les mêmes, peu importe la forme de la trajectoire.

Adaptation pour les méthodes à chemin imposé

En général, les algorithmes présentés dans la section de planification de trajectoire d'un chemin imposé pourraient être repris pour la planification de trajectoire de chemin point-à-point. Il suffit d'ajouter une boucle supérieure pour faire varier les points de la trajectoire et ainsi tester un grand nombre de trajectoires afin de trouver celle qui est optimale. Cependant, les méthodes précédentes demandent, en soi, un effort de calcul important. Donc, en y ajoutant une autre boucle, la complexité temporelle pourrait être exagérément augmentée.

1.8.4 Conclusion

La planification de trajectoire d'un chemin point-à-point est étudiée depuis le milieu des années 1980. Il s'agit d'un sujet en constante évolution dont les nouvelles méthodes s'adaptent sans trop de modifications à plusieurs variantes du problème.

Pour la planification de trajectoire d'un chemin point-à-point sans obstacles et sans considérer le modèle dynamique, la méthode la plus simple est évidemment de tracer une ligne droite dans l'espace articulaire et de calculer la trajectoire de sorte que les contraintes cinématiques soient respectées. Cette méthode est largement utilisée pour tous les types de robots (sériels ou parallèles), car elle ne nécessite pas beaucoup de calculs. Cette méthode s'avère cependant inefficace s'il y a des obstacles puisque certaines trajectoires seraient irréalisables. Le robot est aussi sous-utilisé puisque les contraintes dynamiques ne sont pas considérées.

Les autres méthodes utilisant la programmation non linéaire, le principe du maximum de Pontriaguine ou les champs potentiels ont tous les mêmes inconvénients. Tout d'abord, la complexité du problème est accentuée selon le nombre de degrés de liberté du robot. La mise

en équation est valable uniquement pour l'architecture étudiée. L'ajout de contraintes et le traitement des obstacles ajoutent aussi une complexité qui nécessite souvent la reformulation complète du problème et un réajustement des paramètres du programme pour la résolution numérique. La majorité de ces méthodes peuvent converger vers des minimums locaux et elles ne fournissent pas nécessairement des solutions optimales.

La méthode des splines cubiques aléatoire est une méthode novatrice qui a été introduite par Chettibi (2002). Il s'agit d'une méthode efficace et facile à mettre en œuvre. Elle est moins susceptible de converger vers des minimums locaux lorsque l'algorithme qui génère les trajectoires est stochastique. Cette méthode est flexible et s'adapte facilement à des architectures complexes.

CHAPITRE 2

MODÉLISATION DU ROBOT

2.1 Introduction

La modélisation du robot est une mise en équation qui permet de connaître son comportement géométrique ou dynamique. La nomenclature du robot et les dimensions des liens sont d'abord définies. Ensuite, la modélisation géométrique directe et inverse est élaborée pour faire le lien entre les positions articulaires et cartésiennes. Les modes d'assemblage et de fonctionnement sont expliqués et les calculs pour les déterminer sont présentés. Finalement, la dernière section porte sur la modélisation dynamique du manipulateur.

2.2 Architecture du robot

Telle qu'illustrée par la figure 10, l'architecture étudiée consiste en un robot parallèle de type « mécanisme à cinq barres ». Les articulations O_1 et O_2 constituent la base et sont actionnées (motorisées). Les articulations intermédiaires A_1 et A_2 sont passives et l'articulation passive C correspond aussi à l'effecteur terminal. Les liens 1 et 3 ont la même longueur et sont appelés les liens proximaux. Les liens 2 et 4 ont aussi la même longueur et sont appelés les liens distaux. On définit un bras comme l'ensemble d'un lien proximal et d'un lien distal. Ainsi, le bras 1 est constitué des liens 1 et 2 tandis que le bras 2 est constitué des liens 3 et 4. Les actionneurs sont à la même position d'origine selon les axes y et z du repère de la base et ils sont séparés d'une distance d selon l'axe x. Les positions articulaires des actionneurs sont donc définies par rapport à l'axe x. Les positions articulaires des articulations passives sont quant à elles par rapport aux extrémités des membres proximaux. Les dimensions des liens sont définies dans le tableau 1.

Tableau 1 Dimensions des liens

Liens	Dimension
proximaux : $l_1 = l_3$	230 mm
distaux : $l_2 = l_4$	230 mm
d	275 mm

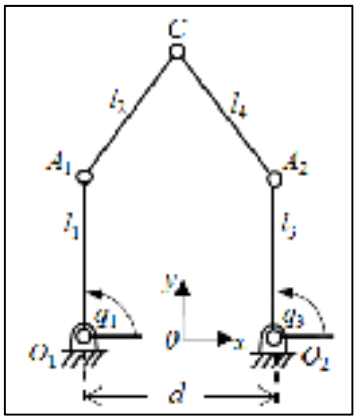


Figure 10 Nomenclature du robot

Parce que le robot de la figure 10 ne peut que bouger dans un plan, les positions cartésiennes des articulations sont définies par des vecteurs à deux dimensions :

$$\mathbf{r}_{OO_1} = \begin{bmatrix} x_{OO_1} \\ y_{OO_1} \end{bmatrix}, \mathbf{r}_{OO_2} = \begin{bmatrix} x_{OO_2} \\ y_{OO_2} \end{bmatrix}, \mathbf{r}_{OA_1} = \begin{bmatrix} x_{OA_1} \\ y_{OA_1} \end{bmatrix}, \mathbf{r}_{OA_2} = \begin{bmatrix} x_{OA_2} \\ y_{OA_2} \end{bmatrix} \text{ et } \mathbf{r}_{OC} = \begin{bmatrix} x_{OC} \\ y_{OC} \end{bmatrix} \qquad (2.1)$$

2.3 Modélisation géométrique

La modélisation géométrique directe (MGD) a pour but de calculer la position de l'effecteur en fonction de la position des articulations motorisées. Elle peut être calculée à l'aide de la convention de Denavit-Hartenberg ou de celle de Craig (Denavit-Hartenberg modifiée) ou

encore par des méthodes géométriques. Les deux premières approches sont utilisées majoritairement pour les robots sériels puisque le modèle est alors de façon générale une fonction de toutes les articulations du robot. Dans le cas du robot étudié, ces méthodes s'avèrent plus complexes puisque le modèle doit être fonction uniquement des articulations fixées à la base du robot, ce qui nécessite la résolution d'équations de contraintes. Une approche géométrique sera donc privilégiée. Cependant, la méthode de Craig sera utilisée lors de la modélisation dynamique, qui, de toute façon, est d'une plus grande complexité parce qu'elle nécessite une étude du mouvement de tous les membres du système.

Le modèle géométrique aura deux solutions qui correspondent aux deux modes d'assemblage. Ces deux solutions sont détaillées dans la section suivante.

2.3.1 Solutions géométriques

Par géométrie, les positions des articulations passives sont facilement calculables en fonction des positions des articulations actives (figure 11) :

$$\mathbf{r}_{OA_1} = \begin{bmatrix} x_{OO_1} + l_1 \cos q_1 \\ y_{OO_1} + l_1 \sin q_1 \end{bmatrix} \tag{2.2}$$

$$\mathbf{r}_{OA_2} = \begin{bmatrix} x_{OO_2} + l_3 \cos q_3 \\ y_{OO_2} + l_3 \sin q_3 \end{bmatrix} \tag{2.3}$$

Pour distinguer les deux solutions possibles, la variable s est ensuite définie de la façon suivante :

$$s = \sqrt{(x_{OA_1} - x_{OA_2})^2 + (y_{OA_1} - y_{OA_2})^2} \tag{2.4}$$

de sorte que si $s > l_2 + l_4$, il n'y a pas de solution. Autrement, si on définit

$$\mathbf{v} = \frac{\mathbf{r}_{OA_1} - \mathbf{r}_{OA_2}}{2} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{2.5}$$

$$\mathbf{h} = \begin{bmatrix} v_x \\ -v_y \end{bmatrix} \frac{2}{s} \sqrt{l_2^2 - \left(\frac{s}{2}\right)^2} \quad . \tag{2.6}$$

il y a deux solutions possibles (figure 11) :

$$\mathbf{r}_{OC} = \mathbf{r}_{OA_2} + \mathbf{v} \pm \mathbf{h} \tag{2.7}$$

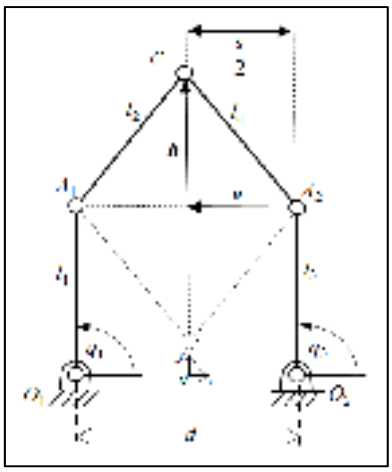


Figure 11 Variables de la solution algébrique de la MGD

2.3.2 Modes d'assemblage

Le modèle géométrique a toujours deux solutions possibles qui correspondent aux modes d'assemblage (voir les figures 12 et 13). Cependant, le robot doit toujours être dans le même mode d'assemblage lors de son fonctionnement. Le mode d'assemblage du DexTAR a été choisi positif. On peut alors en tout temps connaître la position de l'effecteur terminal en fonction de la position des moteurs en considérant que le mode d'assemblage positif. Dans la planification de trajectoire, il sera impossible de changer de mode d'assemblage puisque cela implique de passer par une singularité de type 2, ce qui n'est pas préconisé dans ce mémoire.

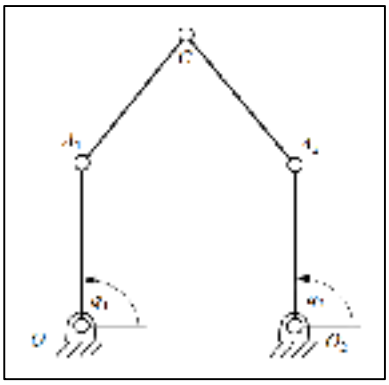


Figure 12 Mode d'assemblage positif

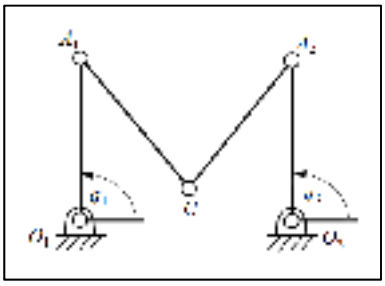


Figure 13 Mode d'assemblage négatif

2.3.3 Déterminer le mode d'assemblage

Pour calculer le mode d'assemblage, il faut connaître la position de l'effecteur terminal C et celle des articulations passives A_1 et A_2, calculées aux équations (2.2) à (2.7). Il faut ensuite calculer :

$$\mathbf{r}_{CA_i} = \begin{bmatrix} x_{CA_i} \\ y_{CA_i} \end{bmatrix} = \mathbf{r}_{OC} - \mathbf{r}_{OA_i} \text{ où } i = 1,2 \tag{2.8}$$

Le mode d'assemblage est positif si :

$$x_{CA_1} y_{CA_2} - x_{CA_2} y_{CA_1} \geq 0 \tag{2.9}$$

2.4 Modélisation géométrique inverse

La modélisation géométrique inverse (MGI) a pour but de trouver les coordonnées articulaires pour une coordonnée cartésienne de l'effecteur terminal donné. Elle est également obtenue par des méthodes géométriques. Ce modèle inverse peut être calculé indépendamment pour chaque bras. Un bras est un ensemble de membres qui relient la base à l'effecteur tel que défini dans la section 2.2. Un bras est constitué d'une articulation active et d'une articulation passive. Deux solutions sont possibles par articulation active pour un total de quatre configurations pour le robot. Ces quatre possibilités sont appelées les modes de fonctionnement du robot.

2.4.1 Équations

Chacun des bras est donc traité indépendamment. La figure 14 montre les deux solutions possibles pour un des bras.

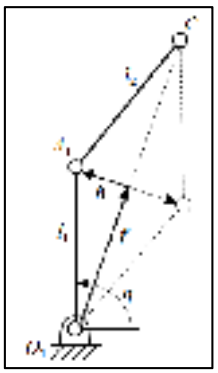


Figure 14 Variables de la solution algébrique de la MGI

La position de l'effecteur et des bases étant alors connue, la distance entre O_i et C est calculée comme suit :

$$\sqrt{\mathbf{r}_{O_iC}^T \mathbf{r}_{O_iC}} = \sqrt{\left(\mathbf{r}_{OO_i} - \mathbf{r}_{OC}\right)^T \left(\mathbf{r}_{OO_i} - \mathbf{r}_{OC}\right)} \text{ où } i = 1,2 \qquad (2.10)$$

Il n'y a pas de solution si la distance entre la base et l'effecteur est plus grande que la somme de la longueur des liens, car le mécanisme serait alors désassemblé:

$$\sqrt{\mathbf{r}_{O_iC}^T \mathbf{r}_{O_iC}} > l_1 + l_2 \text{ ou } \mathbf{r}_{O_2C}^T \mathbf{r}_{O_2C} > l_3 + l_4 \tag{2.11}$$

Puis, considérons le vecteur unitaire le long de O_iC définit par:

$$\mathbf{u}_{O_iC} = \frac{\mathbf{r}_{OO_i} - \mathbf{r}_{OC}}{\sqrt{\mathbf{r}_{O_iC}^T \mathbf{r}_{O_iC}}} \tag{2.12}$$

En utilisant les propriétés des triangles, on peut alors déterminer la position du point A_i à l'aide des équations suivantes :

$$\mathbf{f}^T\mathbf{f} = \frac{l_i^2 + l_j^2 + \mathbf{r}_{O_iC}^T \mathbf{r}_{O_iC}}{2\sqrt{\mathbf{r}_{O_iC}^T \mathbf{r}_{O_iC}}} \tag{2.13}$$

$$\mathbf{h}^T\mathbf{h} = \sqrt{l_i^2 - \mathbf{f}^T\mathbf{f}} \tag{2.14}$$

$$\mathbf{r}_{OA_i} = \mathbf{r}_{OO_i} + \mathbf{f}^T\mathbf{f}\mathbf{u}_{O_iC} \pm \mathbf{h}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\mathbf{u}_{O_iC} \tag{2.15}$$

En connaissant la position cartésienne de A_i et de la base, l'angle formé par ce vecteur par rapport au référentiel est alors donné par:

$$q_i = \operatorname{atan2}\left(y_{OA_i} - y_{OO_i}\ , x_{OA_i} - x_{OO_i}\right) \tag{2.16}$$

2.4.2 Position articulaire des articulations passives

Pour connaître la position articulaire des articulations passives, il suffit de formuler les équations suivantes connaissant C et A_i :

$$q_2 = \text{atan2}\left(y_{OC} - y_{OA_1}, x_{OC} - x_{OA_1}\right) - q_1 \tag{2.17}$$

$$q_4 = \text{atan2}\left(y_{OC} - y_{OA_2}, x_{OC} - x_{OA_2}\right) - q_3 \tag{2.18}$$

2.4.3 Mode de fonctionnement (WM)

Les modes de fonctionnement correspondent aux quatre solutions du modèle géométrique inverse. À chaque position cartésienne, il peut y avoir jusqu'à quatre solutions, mais elles ne sont pas toutes réalisables puisqu'il faut aussi considérer le mode d'assemblage de ces positions et les interférences mécaniques. Le signe du mode de fonctionnement est déterminé par l'angle entre en le lien distal et le lien proximal. Celui-ci sera positif si l'angle est positif et vice-versa. La figure 15 montre de nouveau les quatre modes de fonctionnement pour une position cartésienne de l'effecteur.

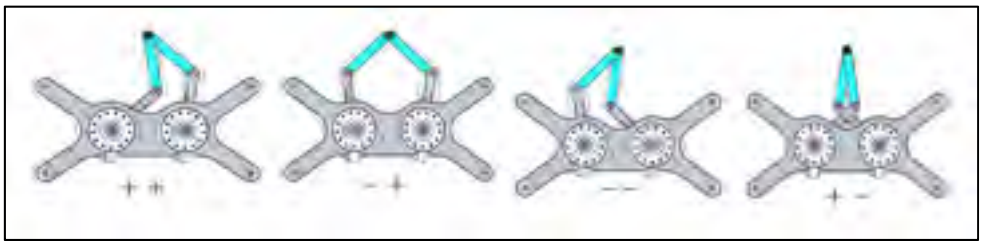


Figure 15 Les quatre modes de fonctionnement

2.4.4 Déterminer le mode de fonctionnement

Pour calculer le mode de fonctionnement, la position de l'effecteur C doit être connue et celles des articulations passives A_1 et A_2 doivent être calculées (équations (2.2) et (2.3)).

Les vecteurs des liens distaux et proximaux sont ensuite calculés:

$$\mathbf{r}_{CA_i} = \mathbf{r}_{OC} - \mathbf{r}_{OA_i} \quad \text{où} \quad i = 1\ , 2 \tag{2.19}$$

$$\mathbf{r}_{O_iA_i} = \mathbf{r}_{OO_i} - \mathbf{r}_{OA_i} \quad \text{où} \quad i = 1, 2 \tag{2.20}$$

Il faut finalement calculer le facteur w_i qui déterminera le mode de fonctionnement :

$$w_i = -x_{O_iA_i} y_{CA_i} + y_{O_iA_i} x_{CA_i} \quad \text{où} \quad i = 1, 2 \tag{2.21}$$

Le mode de fonctionnement sera positif si : $w_i > 0$

2.5 Modèle dynamique

La modélisation dynamique inverse permet de calculer les couples que devront fournir les moteurs pour suivre une trajectoire articulaire donnée. Pour ce calcul, le modèle dynamique inverse a besoin des positions, des vitesses et des accélérations des moteurs à chaque instant de la trajectoire. Il s'agit d'un puissant outil d'analyse, car il permet de simuler des trajectoires et d'analyser le comportement du robot. De plus, le modèle dynamique est utilisé pour l'optimisation de trajectoire et pour la commande. Cette section sera séparée en plusieurs parties. Tout d'abord, les paramètres de liaison et les paramètres dynamiques seront définis. La contrainte de fermeture de la chaine cinématique sera ensuite modélisée. Le

modèle dynamique complet sera présenté et il sera réduit. Le modèle réduit sera utilisé pour la planification de trajectoire. Finalement, une section portera sur la validation des modèles et l'acquisition des paramètres.

2.5.1 Définition des paramètres de liaisons

Le robot est constitué de deux chaines cinématiques qui seront d'abord considérées comme ouvertes, puis qui seront fermées à l'aide d'une contrainte de fermeture. Chacune des chaînes représente donc un bras du robot. La figure 16 représente les paramètres de liaisons. Les dimensions du robot sont les mêmes que définies précédemment à l'exception des élévations en *z* qui avaient été négligées.

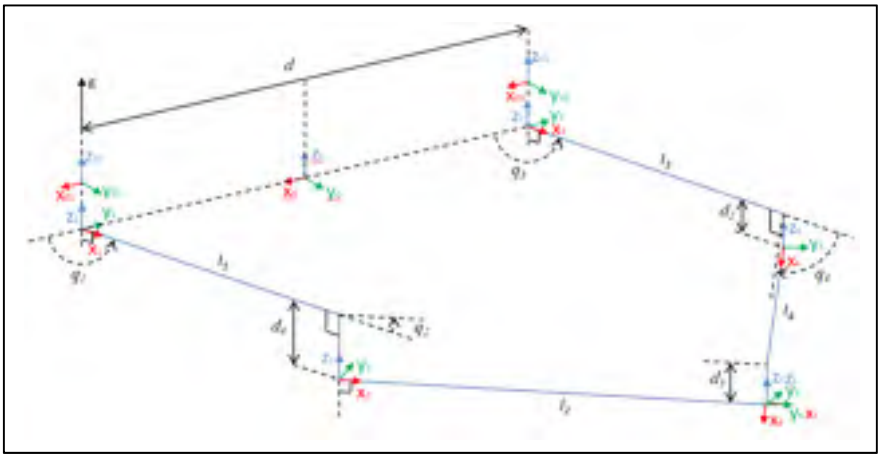


Figure 16 Représentation schématique des paramètres de liaisons

2.5.2 Définition des paramètres dynamiques

La modélisation dynamique est basée sur la méthode de Lagrange. Cette méthode nécessite les paramètres de liaison de la section précédente et les spécifications dynamiques comme le centre de masse de chaque lien par rapport à l'articulation, sa masse et sa matrice d'inerties par rapport à son centre de masse.

De façon générale, six paramètres inertiels sont nécessaires pour modéliser les mouvements angulaires de chaque partie mobile d'un robot évoluant dans un espace 3D. Cependant, toutes les parties mobiles du robot DexTAR ne peuvent bouger que dans le plan perpendiculaire à l'axe *z*. Pour cette raison, seuls les paramètres inertiels dans cet axe de rotation sont nécessaires pour la modélisation. Ainsi, le tableau 2 ne montre que les paramètres inertiels selon l'axe z et les coordonnées du centre de masse en *x* et en *y*, même si le logiciel de CAO peut calculer tous les autres paramètres. Les valeurs de chacun des paramètres seront données plus bas. Dans ce mémoire, la bibliothèque de modélisation robotique sous Maple élaborée par le laboratoire de calcul avancé de l'École de technologie supérieure (Bigras, 2003) a été utilisée pour effectuer les calculs de la modélisation dynamique. Les détails de cette méthode seront explicités plus bas.

Tableau 2 Spécifications dynamiques

membres	$\boldsymbol{Cm_{xi}}$	$\boldsymbol{Cm_{yi}}$	$\boldsymbol{m_i}$	$\boldsymbol{I_{zi}}$
1	Cm_{x1}	Cm_{y1}	m_1	I_{z1}
2	Cm_{x2}	Cm_{y2}	m_2	I_{z2}
3	Cm_{x3}	Cm_{y3}	m_3	I_{z3}
4	Cm_{x4}	Cm_{y4}	m_4	I_{z4}

Les paramètres sont définis comme suit :

- cm_{xi} = Coordonnées de la position du centre de masse du lien *i* par rapport à l'articulation *i* selon l'axe $\hat{X}_i$
- cm_{yi} = Coordonnées de la position du centre de masse du lien *i* par rapport à l'articulation *i* selon l'axe $\hat{Y}_i$
- m_i = masse du lien *i*

- I_{zi} = Moment d'inertie principal par rapport au centre de masse du lien *i* selon l'axe $\hat{Z}_{cm_i}$

2.5.3 Modélisation de la contrainte

La contrainte correspond à la fermeture de la chaîne cinématique définie par les deux bras. Elle peut être obtenue géométriquement. Il faut alors définir les équations en *x*, *y* et *z* pour lesquelles la chaîne cinématique sera fermée, c'est-à-dire que le vecteur $\mathbf{g_c(q)}$ suivant sera égal à **0**:

$$\mathbf{g}_{\mathrm{c}}(\mathbf{q}) = \begin{bmatrix} g_{c1}(\mathbf{q}) \\ g_{c2}(\mathbf{q}) \\ g_{c3}(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} l_2 c_{12} + l_1 c_1 - d_p - l_4 c_{34} - l_3 c_3 \\ l_2 s_{12} + l_1 s_1 - l_4 s_{34} - l_3 s_3 \\ -d_5 - d_2 + d_4 \end{bmatrix} = \mathbf{0} \tag{2.22}$$

2.5.4 Jacobienne de la contrainte

La jacobienne est une forme multidimensionnelle de la dérivée. Elle peut être utilisée pour représenter les formes différentielles de la contrainte suivantes

$$\mathbf{v} = \mathbf{J}_g(\mathbf{q})\dot{\mathbf{q}} \tag{2.23}$$

$$\mathbf{a} = \dot{\mathbf{J}}_g(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{q}} + \mathbf{J}_g(\mathbf{q})\ddot{\mathbf{q}} \tag{2.24}$$

Où **v** et **a** représentent la vitesse et l'accélération cartésienne tandis que $\mathbf{J_g}$ représente la matrice jacobienne $\partial \mathbf{g}_{\mathrm{c}} / \partial \mathbf{q}$.

La jacobienne de la contrainte et sa dérivée seront utilisées pour formuler le modèle dynamique contraint. La jacobienne consiste en une matrice $m \times n$ où *m* est le nombre de fonctions et *n* est le nombre de variables. Chaque colonne correspond à une dérivée partielle

des variables. Dans le cas de la contrainte, la jacobienne est calculée avec les équations de la contrainte et les positions articulaires des articulations. Comme $g_{c3}(\mathbf{q}) = 0$ est indépendante du vecteur $\mathbf{q}$, cette équation est éliminée. Voici donc la matrice jacobienne $\mathbf{J}_g$ de la contrainte :

$$\mathbf{J}_g(\mathbf{q}) = \begin{bmatrix} \frac{\partial g_{c1}}{\partial q_1} & \frac{\partial g_{c1}}{\partial q_2} & \frac{\partial g_{c1}}{\partial q_3} & \frac{\partial g_{c1}}{\partial q_4} \\ \frac{\partial g_{c2}}{\partial q_1} & \frac{\partial g_{c2}}{\partial q_2} & \frac{\partial g_{c2}}{\partial q_3} & \frac{\partial g_{c2}}{\partial q_4} \end{bmatrix} \tag{2.25}$$

$$\mathbf{J}_g(\mathbf{q}) = \begin{bmatrix} -l_2 s_{12} - l_1 s_1 & -l_2 s_{12} & l_4 s_{34} + l_3 s_3 & l_4 s_{34} \\ l_2 c_{12} + l_1 c_1 & l_2 c_{12} & -l_4 c_{34} - l_3 c_3 & -l_4 c_{34} \end{bmatrix} \tag{2.26}$$

2.5.5 Énergie cinétique des bras

L'énergie cinétique est l'énergie du mouvement d'un corps. Pour chacun des liens d'un robot, cette énergie est exprimée par l'équation suivante :

$$k_i = \frac{1}{2} m_i \mathbf{v}_{C_i}^T \mathbf{v}_{C_i} + \frac{1}{2} {}^{i}\boldsymbol{\omega}_i^{T\,C_i}\mathbf{I}_i \, {}^{i}\boldsymbol{\omega}_i \tag{2.27}$$

où m_i est la masse du corps *i*, $\boldsymbol{v}_{C_i}$ est la vitesse linéaire de son centre de masse, ${}^{i}\boldsymbol{\omega}_i$ est la vitesse angulaire de son centre de masse et ${}^{C_i}\mathbf{I}_i$ est sa matrice d'inertie, qui par un choix judicieux du repère associé au corps, peut être exprimée sous la forme principale suivante:

$${}^{C_i}\mathbf{I}_i = \begin{bmatrix} I_{x1} & 0 & 0 \\ 0 & I_{y1} & 0 \\ 0 & 0 & I_{z1} \end{bmatrix} \tag{2.28}$$

L'énergie cinétique totale d'un bras du manipulateur est exprimée par la somme de l'énergie cinétique de chaque lien :

$$k = \sum_{i=1}^{n} k_i \tag{2.29}$$

Les vitesses linéaires et angulaires des centres de masse sont des fonctions de $\mathbf{q}$ et $\dot{\mathbf{q}}$ qui peuvent être obtenues à l'aide du modèle géométrique différentiel du robot (Craig, 1989). L'énergie cinétique du manipulateur peut donc être exprimée par un scalaire en fonction des positions et vitesse des articulations: $k(\mathbf{q}, \dot{\mathbf{q}})$.

2.5.6 Énergie potentielle des bras

L'énergie potentielle correspond à l'énergie emmagasinée par un corps et qui peut être transformée en énergie cinétique. Elle dépend uniquement de la position du système. L'énergie potentielle d'un membre est exprimée par l'équation suivante :

$$u_i = -m_i {}^0\mathbf{g}^T {}^0\boldsymbol{c}_i \tag{2.30}$$

où ${}^0\mathbf{g}$ et ${}^0\mathbf{c}_i$ sont respectivement le vecteur de gravité et la position du centre de masse du corps i exprimés par rapport à la base du robot. Selon la figure 16, le vecteur de gravité est défini par:

$${}^0\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{2.31}$$

La masse des membres est définie dans le tableau 2 tout comme le vecteur qui localise le centre de masse de chacun des corps. L'énergie potentielle totale d'un bras du manipulateur

est donc exprimée comme étant la somme des énergies potentielles des membres qui le compose :

$$u = \sum_{i=1}^{n} u_i \tag{2.32}$$

Le vecteur qui localise le centre de masse est exprimé en fonction de **q**. L'énergie potentielle du manipulateur peut donc être exprimée par un scalaire en fonction de la position des articulations : u(**q**).

2.5.7 Modèle dynamique sans contrainte

La formulation dynamique par la méthode de Lagrange (Murray, Li et Sastry, 1994) permet de dériver les équations du mouvement à partir d'une fonction scalaire appelée le Lagrangien. Pour un système mécanique, le lagrangien est défini comme la différence entre l'énergie cinétique et l'énergie potentielle :

$$\mathrm{L}\left(\mathbf{q}, \dot{\mathbf{q}}\right) = k\left(\mathbf{q}, \dot{\mathbf{q}}\right) - u(\mathbf{q}) \tag{2.33}$$

L'équation du mouvement est alors donnée par :

$$\tau = \frac{d}{dt}\frac{d\mathrm{L}}{d\dot{\mathbf{q}}} - \frac{d\mathrm{L}}{d\mathbf{q}} = \frac{d}{dt}\frac{dk}{d\dot{\mathbf{q}}} - \frac{dk}{d\mathbf{q}} + \frac{du}{d\mathbf{q}} \tag{2.34}$$

La dynamique de chacun des bras du robot DexTAR peut alors s'exprimer sous une forme générale qui est abondamment utilisée pour l'asservissement des robots. Cette forme est obtenue pour les deux bras du DexTAR par les équations suivantes :

$$\mathbf{M}_1\left(\mathbf{q}_{12}\right)\ddot{\mathbf{q}}_{12} + \mathbf{F}_1\left(\mathbf{q}_{12}, \dot{\mathbf{q}}_{12}\right) = \mathbf{\Gamma}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\tau_1 \text{ où } \mathbf{q}_{12} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{2.35}$$

où

$$\mathbf{M}_1(\mathbf{q}_{12}) = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \text{ où } \mathbf{q}_{12} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{2.36}$$

$$\mathbf{M}_{11} = m_1 C_{x1}{}^2 + m_2 C_{x2}{}^2 + m_1 C_{y1}{}^2 + m_2 l_1{}^2 + 2 l_1 m_2 \left(C_{x2} c_2 - C_{y2} s_2 \right) + I_{z1} + I_{z2}$$

$$\mathbf{M}_{12} = m_2 C_{x2}{}^2 + m_2 C_{y2}{}^2 + l_1 m_2 \left(C_{x2} c_2 - C_{y2} s_2 \right) + I_{z2}$$

$$\mathbf{M}_{21} = \mathbf{M}_{12}$$

$$\mathbf{M}_{22} = m_2 C_{x2}{}^2 + m_2 C_{y2}{}^2 + I_{z2}$$

$$\mathbf{F}_1(\mathbf{q}_{12}, \dot{\mathbf{q}}_{12}) = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \end{bmatrix} \text{ où } \mathbf{q}_{12} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{2.37}$$

$$\mathbf{F}_{11} = -l_1 m_2 \dot{q}_2 \left(C_{y2} c_2 + C_{x2} s_2 \right) \left(2\dot{q}_1 + \dot{q}_2 \right)$$

$$\mathbf{F}_{12} = l_1 m_2 \dot{q}_1{}^2 \left(C_{y2} c_2 + C_{x2} s_2 \right)$$

et

$$\mathbf{M}_3(\mathbf{q}_{34}) \ddot{\mathbf{q}}_{34} + \mathbf{F}_3(\mathbf{q}_{34}, \dot{\mathbf{q}}_{34}) = \mathbf{\Gamma}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tau_3 \text{ où } \mathbf{q}_{34} = \begin{bmatrix} q_3 \\ q_4 \end{bmatrix} \tag{2.38}$$

où

$$\mathbf{M}_3(\mathbf{q}_{34}) = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \text{ où } \mathbf{q}_{34} = \begin{bmatrix} q_3 \\ q_4 \end{bmatrix} \tag{2.39}$$

$$\mathbf{M}_{11} = m_3 C_{x3}{}^2 + m_4 C_{x4}{}^2 + m_3 C_{y3}{}^2 + m_4 l_3{}^2 + 4 l_3 m_4 \left(C_{x4} c_4 - C_{y4} s_4 \right) + I_{z3} + I_{z4}$$

$$\mathbf{M}_{12} = m_4 C_{x4}{}^2 + m_4 C_{y4}{}^2 + l_3 m_4 \left(C_{x4} c_4 - C_{y4} s_4 \right) + I_{z4}$$

$$\mathbf{M}_{21} = \mathbf{M}_{12}$$

$$\mathbf{M}_{22} = m_4 C_{x4}{}^2 + m_4 C_{y4}{}^2 + I_{z4}$$

$$\mathbf{F}_3\left(\mathbf{q}_{34},\dot{\mathbf{q}}_{34}\right)=\begin{bmatrix}\mathbf{F}_{13} & \mathbf{F}_{14}\end{bmatrix} \text{ où } \mathbf{q}_{34}=\begin{bmatrix}q_3\\ q_4\end{bmatrix} \tag{2.40}$$

$$\mathbf{F}_{13}=-l_3m_4\dot{q}_4\left(C_{y4}c_4+C_{x4}s_4\right)\left(2\dot{q}_3+\dot{q}_4\right)$$

$$\mathbf{F}_{14}=l_3m_4\dot{q}_3^{\,2}\left(C_{y4}c_4+C_{x4}s_4\right)$$

Les modèles des bras peuvent ensuite être combinés de la façon suivante:

$$\begin{bmatrix}\mathbf{M}_1\left(\mathbf{q}_{12}\right) & 0\\ 0 & \mathbf{M}_3\left(\mathbf{q}_{34}\right)\end{bmatrix}\begin{bmatrix}\ddot{\mathbf{q}}_{12}\\ \ddot{\mathbf{q}}_{34}\end{bmatrix}+\begin{bmatrix}\mathbf{F}_1\left(\mathbf{q}_{12},\dot{\mathbf{q}}_{12}\right)\\ \mathbf{F}_3\left(\mathbf{q}_{34},\dot{\mathbf{q}}_{34}\right)\end{bmatrix}=\begin{bmatrix}\mathbf{\Gamma}_1\\ \mathbf{\Gamma}_3\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_3\end{bmatrix} \tag{2.41}$$

D'où

$$\mathbf{M}\left(\mathbf{q}\right)\ddot{\mathbf{q}}+\mathbf{F}\left(\mathbf{q},\dot{\mathbf{q}}\right)=\mathbf{\Gamma}=\mathbf{B}\boldsymbol{\tau} \text{ où } \mathbf{q}=\begin{bmatrix}\mathbf{q}_{12}\\ \mathbf{q}_{34}\end{bmatrix} \tag{2.42}$$

2.5.8 Modèle dynamique avec la contrainte

Le modèle dynamique donné par l'équation (2.42) ne tient pas compte de la contrainte. Il s'agit donc de deux bras libres qui évoluent de façon indépendante. En considérant la contrainte sous sa forme différentielle obtenue à la section 3.5.4, le modèle devient (Murray, Li et Sastry, 1994) :

$$\mathbf{M}\left(\mathbf{q}\right)\ddot{\mathbf{q}}+\mathbf{F}\left(\mathbf{q},\dot{\mathbf{q}}\right)+\mathbf{J}_g^T(\mathbf{q})\boldsymbol{\lambda}=\mathbf{B}\boldsymbol{\tau} \tag{2.43}$$

$$\dot{\mathbf{J}}_g\left(\mathbf{q}\right)\dot{\mathbf{q}}+\mathbf{J}_g\left(\mathbf{q}\right)\ddot{\mathbf{q}}=\mathbf{0}$$

$$\mathbf{g}_c\left(\mathbf{q}(t_0)\right)=\mathbf{0}$$

où $\boldsymbol{\lambda}$ est le vecteur des multiplicateurs de Lagrange qui représente le vecteur des forces de contrainte et $\mathbf{J}_g$ est la matrice jacobienne de la contrainte.

2.5.9 Modèle dynamique réduit

La réduction du modèle dynamique est utilisée pour exprimer l'effort des moteurs en fonction du mouvement des articulations motorisées sans faire intervenir les forces de contrainte comme dans l'équation (2.43). Ainsi, le modèle dynamique réduit aura comme variables les positions, vitesses et accélérations des articulations actives q_1 et q_3. Ce modèle sera utilisé pour la planification de trajectoire ainsi que pour l'asservissement. Les équations (2.17) et (2.18) permettent de calculer les positions de q_2 et q_4. pour obtenir ce modèle, le changement de variables suivantes est d'abord considéré:

$$\mathbf{q}=\begin{bmatrix}\mathbf{q}_{13}\\ \mathbf{q}_{24}\end{bmatrix}=\begin{bmatrix}q_1\\ q_3\\ q_2\\ q_4\end{bmatrix},\ \boldsymbol{\beta}=\begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\\ 0 & 0\end{bmatrix},\ \mathbf{J}_g=\frac{\partial \mathbf{g}_c}{\partial \mathbf{q}},\ \boldsymbol{\tau}=\begin{bmatrix}\tau_1\\ \tau_3\end{bmatrix} \tag{2.44}$$

L'équation de la contrainte (2.22) demeure la même, mais la composante en *z* qui est égale à 0 est supprimée, c'est-à-dire $g_{c3}(\mathbf{q})$. Le modèle dynamique contraint à la même forme que celui calculé précédemment, mais en considérant la nouvelle définition :

$$\mathbf{M}\ddot{\mathbf{q}}+\mathbf{F}\left(\dot{\mathbf{q}},\mathbf{q}\right)+\mathbf{J}_g{}^T\boldsymbol{\lambda}=\boldsymbol{\beta}\boldsymbol{\tau} \tag{2.45}$$

La contrainte de vitesse peut alors être récrite sous la forme suivante:

$$\mathbf{J}_g\dot{\mathbf{q}} = 0 \Rightarrow \begin{bmatrix} \mathbf{J}_{g1} & \mathbf{J}_{g2} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{13} \\ \dot{\mathbf{q}}_{24} \end{bmatrix} \tag{2.46}$$

Les vitesses articulaires des articulations passives peuvent alors être exprimées en fonction des vitesses des articulations motorisées :

$$\dot{\mathbf{q}}_{24} = \begin{bmatrix} \dot{q}_2 \\ \dot{q}_4 \end{bmatrix} = -\mathbf{J}_{g2}^{-1}\mathbf{J}_{g1}\dot{\mathbf{q}}_{13} \tag{2.47}$$

Le changement de variable suivant est alors utilisé pour réduire le modèle dynamique :

$$\dot{\bar{\mathbf{q}}} = \mathbf{J}_T\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{I}_2 & 0 \\ \mathbf{J}_g & \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{13} \\ \dot{\mathbf{q}}_{24} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_2 & 0 \\ \mathbf{J}_{g1} & \mathbf{J}_{g2} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{13} \\ \dot{\mathbf{q}}_{24} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}_{13} \\ \mathbf{J}_g\dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}_{13} \\ 0 \end{bmatrix} \tag{2.48}$$

Et

$$\dot{\mathbf{q}} = \mathbf{J}_T^{-1}\dot{\bar{\mathbf{q}}} = \begin{bmatrix} \mathbf{I}_2 & 0 \\ -\mathbf{J}_{g2}^{-1}\mathbf{J}_{g1} & \mathbf{J}_{g2}^{-1} \end{bmatrix} \dot{\bar{\mathbf{q}}} \tag{2.49}$$

En effet, cette définition des vitesses est choisie pour faire en sorte que la première partie du vecteur est une vitesse libre de contraintes alors que la deuxième partie est complètement contrainte. En appliquant le changement de variable défini par (2.48)–(2.49) au modèle (2.45), le modèle dynamique prend la forme suivante :

$$\bar{\mathbf{M}}\dot{\bar{\mathbf{q}}} + \bar{\mathbf{F}} + \bar{\mathbf{J}}_g^{-T}\boldsymbol{\lambda} = \bar{\boldsymbol{\beta}}\boldsymbol{\tau} \tag{2.50}$$

où

$$\bar{\mathbf{M}} = \mathbf{J}_T^{-T}\mathbf{M}\mathbf{J}_T^{-1} \tag{2.51}$$

$$\bar{\mathbf{F}} = \mathbf{J}^{-T}\left[\mathbf{F} - \mathbf{M}\mathbf{J}_T^{-1}\dot{\mathbf{J}}_T\dot{\mathbf{q}}\right] \tag{2.52}$$

$$\bar{\mathbf{J}}_g = \mathbf{J}_g\mathbf{J}_T^{-1} \tag{2.53}$$

$$\bar{\boldsymbol{\beta}} = \mathbf{J}_T^{-T}\boldsymbol{\beta} \tag{2.54}$$

Pour correspondre au partitionnement de la définition de vitesse donné par (2.48), le modèle peut être réécrit comme suit:

$$\begin{bmatrix}\bar{\mathbf{M}}_{11} & \bar{\mathbf{M}}_{12}\\ \bar{\mathbf{M}}_{21} & \bar{\mathbf{M}}_{22}\end{bmatrix}\begin{bmatrix}\ddot{\bar{\mathbf{q}}}\\ 0\end{bmatrix}+\begin{bmatrix}\bar{\mathbf{F}}_1\\ \bar{\mathbf{F}}_2\end{bmatrix}+\begin{bmatrix}0\\ \bar{\mathbf{J}}_{g2}{}^{T}\end{bmatrix}\boldsymbol{\lambda}=\begin{bmatrix}\bar{\boldsymbol{\beta}}_1\\ \bar{\boldsymbol{\beta}}_2\end{bmatrix}\boldsymbol{\tau} \tag{2.55}$$

Étant donné que les forces de contraintes ne sont pas utilisées par les méthodologies présentées dans ce mémoire, seule la première partie de l'équation est retenue :

$$\bar{\mathbf{M}}_{11}\ddot{\bar{\mathbf{q}}} + \bar{\mathbf{F}}_1 = \bar{\boldsymbol{\beta}}_1\boldsymbol{\tau} \tag{2.56}$$

où, des équations (2.50)–(2.54), on peut déduire que

$$\bar{\mathbf{M}}_{11} = \begin{bmatrix}\mathbf{I}_2 & -\mathbf{J}_{g1}^{T}\mathbf{J}_{g2}^{-T}\end{bmatrix}\mathbf{M}\begin{bmatrix}\mathbf{I}_2\\ -\mathbf{J}_{g2}^{-1}\mathbf{J}_{g1}\end{bmatrix} \tag{2.57}$$

$$\bar{\mathbf{F}}_1 = \begin{bmatrix}\mathbf{I}_2 & -\mathbf{J}_{g1}^{T}\mathbf{J}_{g2}^{-T}\end{bmatrix}(\mathbf{F} - \mathbf{M}\mathbf{J}_T^{-1}\dot{\mathbf{J}}_T\dot{\mathbf{q}}) \tag{2.58}$$

$$\bar{\boldsymbol{\beta}}_1 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \tag{2.59}$$

2.5.10 Validation des modèles dynamiques

Dans les sections précédentes, le modèle dynamique contraint a été obtenu par la méthode de Lagrange. La dynamique du robot peut aussi être calculée à l'aide de la bibliothèque SimMechanics du logiciel Simulink/Matlab. Le modèle SimMechanics se présente sous une forme schématique facilement éditable. Des blocs peuvent être édités et interconnectés de façon à définir virtuellement l'architecture du mécanisme. Les articulations actives et passives sont représentées par des blocs tout comme les membres. Des capteurs sont placés pour pouvoir analyser les résultats. Chaque membre possède les propriétés du robot qui ont été définies dans les figures 1 et 2. Ce modèle sert principalement à valider les modèles dynamiques dont les calculs ont été présentés précédemment. Ainsi, un contrôleur PID a été construit dans Simulink et différentes trajectoires ont été exécutées pour analyser le comportement des différents modèles. Les résultats obtenus sont très peu différents. Ces différences peuvent s'expliquer par les différences de représentation du modèle et des contraintes qui engendrent nécessairement de petites différences lors de l'intégration numériques.

2.5.11 Acquisition des paramètres à partir de SolidWorks

Les paramètres dynamiques d'un corps peuvent être facilement extraits d'un logiciel CAO. Pour ce mémoire, le logiciel SolidWorks a été utilisé. Pour le calcul de ces paramètres, un système de coordonnées est défini et le logiciel calculera automatiquement les données à partir de celui-ci. Comme le fichier CAO du robot DexTAR était disponible, cette fonction a été utilisée pour obtenir les valeurs numériques du tableau 2. Le système de coordonnées devait alors être le point d'attache avec le corps précédent tel que représenté à la figure 16. Les figures 17, 18, 19 présentent les corps tel quel modélisés dans le logiciel SolidWorks et les tableaux 3, 4 et 5 présentent les paramètres dynamiques propres à chaque corps.

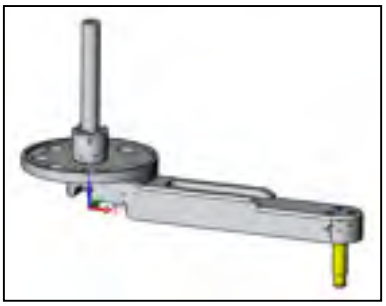


Figure 17 Liens proximaux 1 et 3

Tableau 3 Liens proximaux 1 et 3

Distance entre les points d'attache (en m)	
$a_{O1} = a_{O2}$	0.230
d_2	0.0142
d_4	0.0302
Masse du membre (en kg)	
$m_1 = m_3$	0.838
Position du centre de gravité par rapport au système de coordonnées (en m)	
$Cm_{x1} = Cm_{x3}$	0.09163
$Cm_{y1} = Cm_{y3}$	0
$Cm_{z1} = Cm_{z3}$	0.0262
Moment d'inertie au centre de gravité et aligné avec le système de coordonnées (en kg m2)	
$I_{x1} = I_{x3}$	0.00082
$I_{y1} = I_{y3}$	0.00731
$I_{z1} = I_{z3}$	0.00694

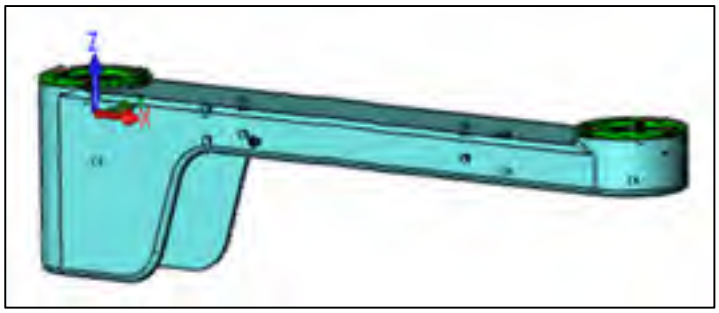


Figure 18 Lien distal 2

Tableau 4 Lien distal 2

Distance entre les points d'attache (en m)	
a_1	0.230
d_5	0.016
Masse du membre (en kg)	
m_2	0.43309048
Position du centre de gravité par rapport au système de coordonnées (en m)	
Cm_{x2}	0.09860
Cm_{y2}	0
Cm_{z2}	-0.00096
Moment d'inertie au centre de gravité et aligné avec le système de coordonnées (en kg m2)	
I_{x2}	0.00011058
I_{y2}	0.00306398
I_{z2}	0.00306266

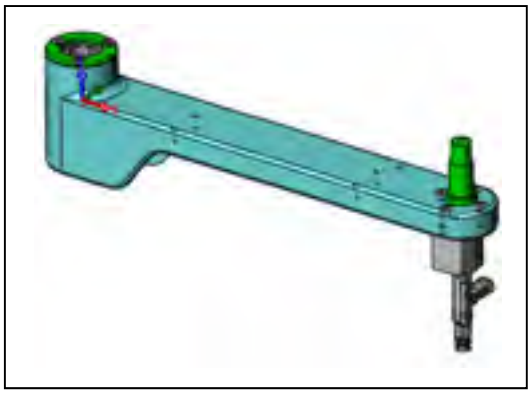


Figure 19 Lien distal 4

Tableau 5 Lien distal

Distance entre les points d'attache (en m)	
a_3	0.230
Masse du membre (en kg)	
m_4	0.51444
Position du centre de gravité par rapport au système de coordonnées (en m)	
Cm_{x4}	0.12294
Cm_{y4}	0.00011
Cm_{z4}	0.00202
Moment d'inertie au centre de gravité et aligné avec le système de coordonnées (en kg m^2)	
I_{x4}	0.00015
I_{y4}	0.00427
I_{z4}	0.00423

2.6 Conclusion

La modélisation du robot est une étape cruciale pour connaître son comportement géométrique et dynamique. La modélisation géométrique est utilisée lors de l'analyse de l'espace de travail, du calcul des singularités, du calcul des interférences mécaniques et inévitablement lors de la planification de trajectoire. La modélisation dynamique est utilisée pour la simulation. Elle permet de connaître le comportement du robot lors de l'exécution de trajectoire et ainsi valider que le prototype ne soit pas endommagé. La modélisation dynamique permet aussi de planifier des trajectoires optimales en s'assurant que le couple maximal des moteurs ne sera pas dépassé.

CHAPITRE 4

ESPACE DE TRAVAIL

4.1 Introduction

L'espace de travail caractérise l'espace atteignable du robot. Ce chapitre présente les calculs qui permettent de définir l'espace de travail. Tout d'abord, il y aura la distinction entre l'espace de travail cartésien et l'espace de travail articulaire. Ensuite, les calculs pour l'ensemble des singularités et des interférences mécaniques seront présentés. Finalement, il y aura une comparaison entre l'espace de travail réel et théorique du robot.

4.2 Notions de l'espace de travail

4.2.1 Espace cartésien

L'espace cartésien est l'espace de la tâche relative à un référentiel. C'est l'espace physique du robot où on définit des coordonnées x et y de l'effecteur. Les positions à atteindre par le robot sont définies dans l'espace cartésien. Elles sont ensuite transformées en coordonnées articulaires grâce au modèle géométrique inverse. Les trajectoires sont généralement définies dans l'espace cartésien puis transformées dans l'espace articulaire pour le contrôle. L'espace cartésien théorique du robot DexTAR est représenté à la figure 20 pour le mode d'assemblage positif.

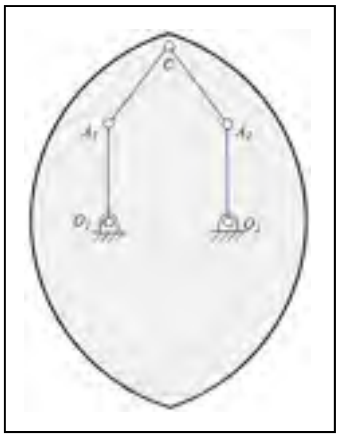


Figure 20 Espace cartésien

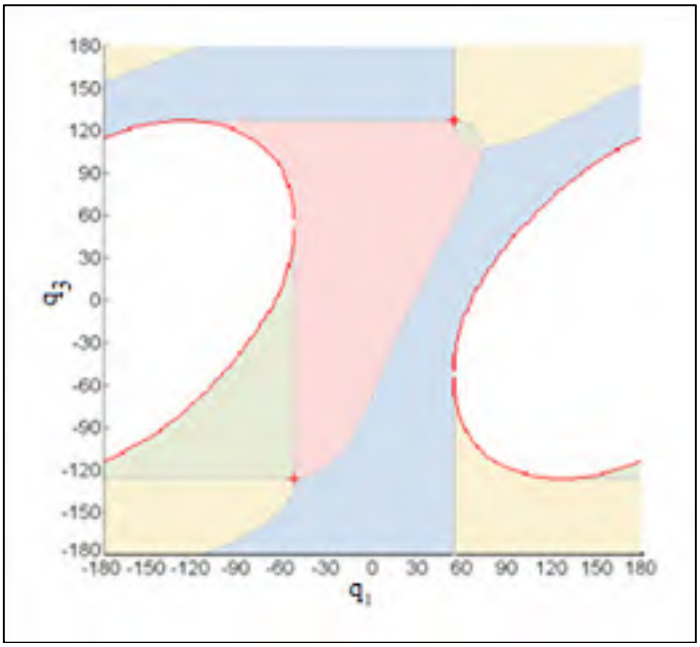


Figure 21 Espace articulaire

4.2.2 Espace articulaire

L'espace articulaire consiste en l'espace des actionneurs. Pour un robot à deux degrés de liberté, cet espace est facile à illustrer dans un graphique à deux dimensions. Plus il y a d'articulations motorisées (degrés de liberté), plus il est difficile de représenter la position du

robot, car chaque articulation motorisée correspond à une dimension. Pour un robot parallèle, la planification de trajectoire dans l'espace articulaire est peu utilisée puisque les déplacements simples dans cet espace (par exemple les lignes droites) sont rarement réalisables à cause des interférences mécaniques et des zones inaccessibles.

L'espace articulaire des actionneurs du robot DexTAR est représenté sous forme graphique à la figure 21. En abscisse, l'angle de q_1 (articulation motorisée 1) est représenté tandis qu'en ordonnée, on retrouve la position de q_3 (articulation motorisée 2). Dans l'espace articulaire, les différents modes de fonctionnement sont représentés par quatre couleurs (bleu, jaune, rose et vert).

Les zones blanches sont inaccessibles puisque, dans ces zones, la chaine cinématique du robot devrait être ouverte pour atteindre ces zones. À ces zones inaccessibles blanches, s'ajouteront d'autres zones qui caractérisent les interférences mécaniques du robot.

Finalement, le courbes rouges correspondent aux configurations singulières de type 2 où les deux liens distaux sont alignés alors que les deux croix rouges correspondent aux configurations singulières de type 2 où les deux liens distaux sont superposés.

4.3 Singularités de type 1

Les singularités de type 1 ont été expliquées au chapitre 1 dans la problématique générale. En résumé, ces configurations sont les frontières de l'espace de travail. C'est l'endroit où le robot atteint sa limite intérieure ou extérieure. Pour un robot à cinq barres comme le DexTAR, ceci correspond à l'endroit où un bras est étendu (externe) ou replié sur lui-même (interne). Les singularités de type 1 externes sont des cercles autour de chacun des axes motorisés dont le rayon est la somme des liens du bras. Pour le DexTAR, les singularités de type 1 internes se trouvent à la base de chacun des moteurs puisque les liens proximaux et distaux ont la même longueur (Figure 22). Les courbes en gris de la figure 21 représentent les singularités de type 1 dans l'espace articulaire.

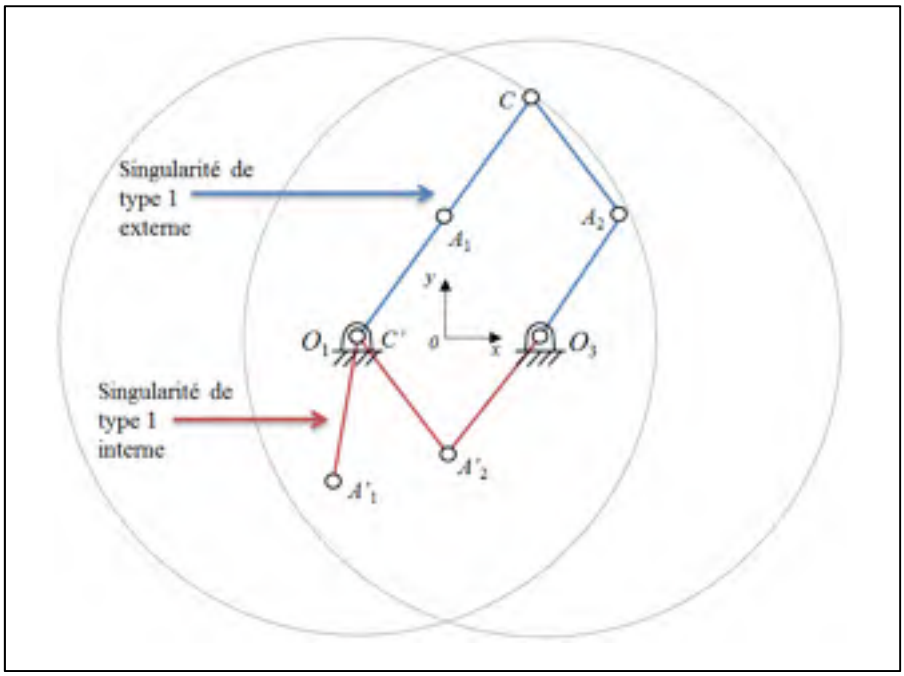


Figure 22 Singularité de type 1

4.3.1 Calculs dans l'espace cartésien et articulaire

Il est possible de calculer l'ensemble des positions où le robot est en singularités de type 1 pour l'espace cartésien et articulaire. Il faut trouver l'ensemble des coordonnées où un des bras est étendu :

1. Un cercle autour de l'axe 1 de longueur l_1+l_2 et un autre autour de l'axe 3 de longueur l_3+l_4 sont tracés. Ces cercle sont calculés à partir d'un ensemble de n points répartis uniformément de $-\pi$ à π qui représentent les positions articulaires de l'axe. Les points du cercle correspondent à des positions cartésiennes.

2. Pour connaître les positions articulaires de l'autre axe, il faut calculer le modèle géométrique inverse de celui-ci à chaque point du cercle pour les deux modes de

fonctionnement. Il en résultera un ensemble de coordonnées articulaire qui constitue des singularités de type 1.

3. Les singularités de type 1 ainsi calculées sont pour tous les modes d'assemblage. Il faut donc filtrer les données en conservant uniquement le mode d'assemblage positif.

Ensuite, il faut calculer l'ensemble des coordonnées où un des bras du robot est replié sur lui-même. Les positions cartésiennes de ces singularités correspondent aux positions des bases O_1 et O_2 puisque les membres distaux et proximaux du DexTAR sont de la même longueur. Puis, les positions articulaires doivent être calculées. En observant le mécanisme de la figure 22, on constate que la position articulaire du bras qui n'est pas en singularité de type 1 ne change pas et peut admettre deux valeurs. L'articulation passive correspondant à ce bras est à 0 en x. Donc les deux positions articulaires de ce bras sont déterminées ainsi:

$$c_i = -\frac{x_{OO_i}}{l_i} \text{ où } i = 1,3 \tag{4.1}$$

$$s_{i1} = \sqrt{1 - c_i^2} \tag{4.2}$$

$$s_{i2} = -\sqrt{1 - c_i^2} \tag{4.3}$$

Ainsi,

$$q_{i1} = \operatorname{atan2}\left(s_{i1}, c_i\right) \tag{4.4}$$

$$q_{i2} = \operatorname{atan2}\left(s_{i2}, c_i\right) \tag{4.5}$$

où $c_i = \cos q_i$ et $s_{i1} = \sin q_i$ pour la première valeur et $s_{i2} = \sin q_i$ pour la deuxième valeur.

Ces deux positions articulaires correspondent à la position de l'articulation du bras qui n'est pas en singularité de type 1. Le bras qui est en singularité de type 1 peut tourner librement autour de sa base. Il faut calculer toutes les positions articulaires de ce bras qui sont dans le mode d'assemblage positif. Il y aura au total quatre ensembles de coordonnées. La procédure pour calculer ces ensembles est la suivante :

1. L'ensemble q_{nj} est défini comme un ensemble de n points répartis uniformément de $-\pi$ à π où $j = 1$ si $i = 3$, $j = 3$ si $i = 1$.

2. La cinématique directe est calculée pour chacun des ensembles :

$$[x_{ni1}\ y_{ni1}] = MGD(q_{i1}, q_{nj}) \tag{4.6}$$

$$[x_{ni2}\ y_{ni2}] = MGD(q_{i2}, q_{nj}) \tag{4.7}$$

   où MGD désigne le modèle géométrique direct donné aux équations (2.2) à (2.7).

3. Il faut ensuite vérifier si la position cartésienne calculée correspond à la position de la base. Si c'est le cas, les valeurs articulaires font partie de l'ensemble où le robot est en singularité de type 1.

4.3.2 Déterminer si le robot est en singularité de type 1

Comme expliqué dans la section précédente, le robot est en singularité de type 1 dans deux cas : *i*) si la position cartésienne de l'effecteur C est égale à la position de l'une des deux bases O_1 ou O_2 ou *ii*) si la distance entre l'effecteur C et une des bases du robot est égale à la somme des liens qui constitue le bras.

4.3.3 Représentation des singularités de type 1

Grâce à la procédure de la section 4.3.1, les singularités de type 1 peuvent être calculées aussi bien dans l'espace articulaire que cartésien. Elles sont représentées en noir sur la figure 20 dans l'espace cartésien et en gris sur la figure 21 dans l'espace articulaire. Ces courbes grises illustrent aussi les frontières de chacun des modes de fonctionnement. Dans l'espace articulaire, le changement de mode de fonctionnement se fait facilement en passant par une singularité de type 1.

4.4 Singularités de type 2

Les singularités de type 2 correspondent à des positions où le robot perd sa raideur en gagnant un degré de liberté. L'effecteur du robot est alors mobile tandis que les actionneurs sont bloqués. En ce qui concerne les robots à cinq barres comme le DexTAR, il y a une singularité de type 2 lorsque les liens distaux sont alignés (figure 23). La courbe formée par l'ensemble de ces singularités de type 2 dans l'espace cartésien est appelée un sextique. Lorsque A_1 et A_2 coïncident (figure 24), il s'agit aussi d'une singularité de type 2 et cela résulte en une infinité de positions cartésiennes pour une même position articulaire. Les singularités correspondantes forment deux cercles qui s'intersectent en O_1 et O_2 dans l'espace cartésien.

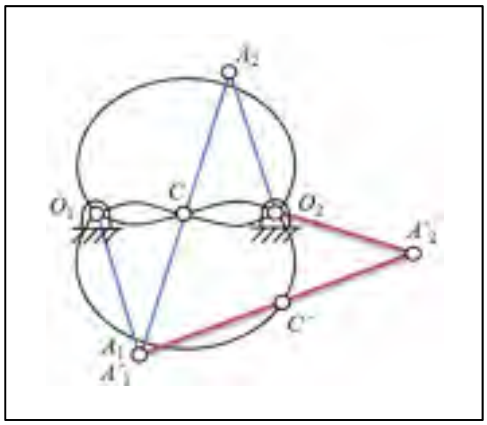


Figure 23 Singularités de type 2 formant un sextique

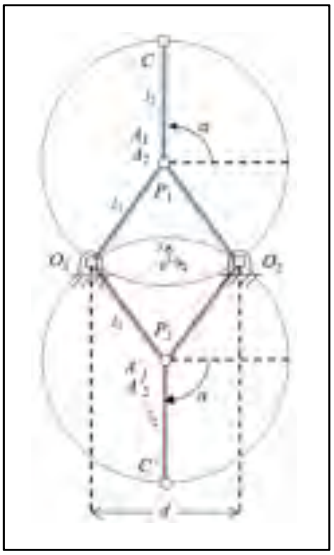


Figure 24 Singularités de type 2 formant deux cercles

4.4.1 Calculs dans l'espace cartésien et articulaire

L'ensemble des positions cartésiennes qui correspond à l'endroit où $A_1 = A_2$ est calculé facilement à partir des équations (4.8) et (4.9). Les deux ensembles de coordonnées articulaires lorsque $A_1 = A_2$ sont $\begin{bmatrix} q_{11} & q_{31} \end{bmatrix}$ et $\begin{bmatrix} q_{12} & q_{32} \end{bmatrix}$. Aux positions correspondantes dans l'espace cartésien, l'effecteur peut tourner librement autour de ce point et former un cercle (Figure 24). Les calculs pour déterminer un ensemble de points sur ces cercles sont simples :

Les points cartésiens où $A_1 = A_2$ sont donnés par :

$$x_{OP_1} = 0 \text{ et } y_{OP_1} = \sqrt{l_1^2 - \frac{d^2}{4}} \tag{4.8}$$

$$x_{OP_2} = 0 \text{ et } y_{OP_2} = -\sqrt{l_1^2 - \frac{d^2}{4}} \tag{4.9}$$

Soit α_n un ensemble de n échantillons répartis uniformément entre $-\pi$ et π en correspondance avec la définition de α de la figure 24. Deux ensembles de coordonnées cartésiennes qui correspondent à des positions de singularité de type 2, soit $[x_{1n}\ y_{1n}]$ *et* $[x_{2n}\ y_{2n}]$, peuvent alors être calculés de la façon suivante :

$$x_{1n} = l_2 \cos\alpha_n + x_{OP_1} \tag{4.10}$$

$$y_{1n} = l_2 \sin\alpha_n + y_{OP_1} \tag{4.11}$$

$$x_{2n} = l_2 \cos\alpha_n + x_{OP_2} \tag{4.12}$$

$$y_{2n} = l_2 \sin\alpha_n + y_{OP_2} \tag{4.13}$$

Pour séparer les segments de cercle pour chacun des modes de fonctionnement, une analyse géométrique doit être réalisée. La figure 25 montre que la position sur le cercle qui correspond au changement de mode de fonctionnement peut être utilisée, c'est-à-dire lorsqu'un des membres est étendu, pour déterminer les segments de cercles qui constituent chacun des modes de fonctionnement. Pour ce faire, l'angle suivant est défini :

$$\beta = \sin^{-1}\left(\frac{d}{l_1 + l_2}\right) \tag{4.14}$$

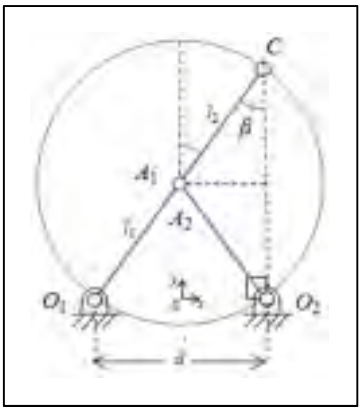


Figure 25 Position du robot lors d'un changement de mode de fonctionnement

Les deux cercles sont séparés tel qu'illustré sur la figure 26.

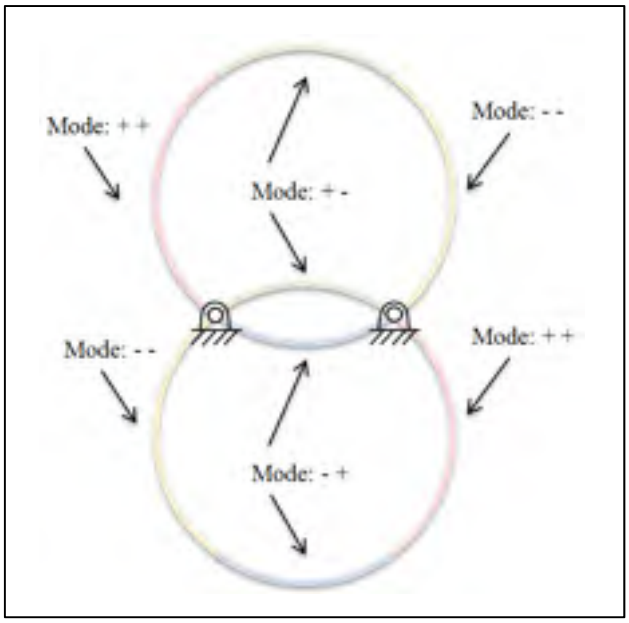


Figure 26 Singularité de type 2 associés aux modes de fonctionnement

Pour chacun des modes de fonctionnement, la plage d'angles α_n (définie à la figure 24) sur chacun des cercles qui correspondent à des positions de singularités de type 2 peut alors être définie comme suit :

- mode de fonctionnement + + ;
  - cercle supérieur : $\alpha_n = \left[\frac{\pi}{2} + \beta \ , \quad \frac{3\pi}{2} - \beta \right]$;
  - cercle inférieur : $\alpha_n = \left[\frac{\pi}{2} + \beta + \pi \ , \quad \frac{3\pi}{2} - \beta + \pi \right]$.

- mode de fonctionnement + − ;
  - cercle supérieur : $\alpha_n = \left[\frac{\pi}{2} - \beta, \quad \frac{\pi}{2} + \beta \right]$;
  - cercle inférieur : $\alpha_n = \left[\frac{\pi}{2} - \beta \ , \quad \frac{\pi}{2} + \beta \right]$.

- mode de fonctionnement − + ;
  - cercle supérieur : $\alpha_n = \left[-\frac{\pi}{2} - \beta \ , \quad -\frac{\pi}{2} + \beta \right]$;
  - cercle inférieur : $\alpha_n = \left[-\frac{\pi}{2} - \beta \ , \quad -\frac{\pi}{2} + \beta \right]$.

- mode de fonctionnement − −.
  - cercle supérieur : $\alpha_n = \left[-\frac{\pi}{2} + \beta \ , \quad \frac{\pi}{2} - \beta \right]$;
  - cercle inférieur : $\alpha_n = \left[-\frac{\pi}{2} + \pi - \beta, \quad -\frac{\pi}{2} + \pi + \beta \right]$.

Les ensembles de α définis précédemment permettent de calculer les singularités de type 2 pour chacun des modes de fonctionnement. Pour ce faire, il faut utiliser les équations (4.10) à (4.13).

Les courbes définies par l'effecteur lorsque les deux liens distaux sont alignés sont des sextiques. Pour déterminer l'ensemble des coordonnées qui répondent à ce critère, il faut

faire tourner un des moteurs sur toute sa plage. Il faut donc définir un ensemble de *n* valeurs de q_1 répartis uniformément de $-\pi$ à π. Ensuite, la marche à suivre est la suivante :

1. Pour chacune des valeurs de l'ensemble de q_1, il faut calculer la position de A_1 associé et sa distance par rapport à la base du membre 2. Pour connaître A_1, il faut utiliser l'équation (2.2). La distance entre la base O_2 et l'articulation A_1 est donnée par :

$$\mathbf{r}_{O_2A_1}^T \mathbf{r}_{O_2A_1} = \sqrt{\left(\mathbf{r}_{OO_2} - \mathbf{r}_{OA_1}\right)^T \left(\mathbf{r}_{OO_2} - \mathbf{r}_{OA_1}\right)} \tag{4.15}$$

2. Si $\mathbf{r}_{O_2A_1}^T \mathbf{r}_{O_2A_1} \geq \left|2l_2 - l_1\right|$, c'est que A_1, C et A_2 peuvent être colinéaires. Comme l'indique la figure 27, la position d'A_1 peut être atteinte de deux façons par le robot en conservant A_1, C et A_2 colinéaires. Les deux positions possibles de l'effecteur et les positions articulaires des actionneurs qui y correspondent (Figure 27) sont calculées :

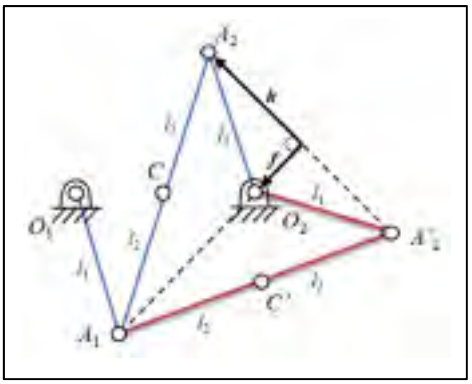


Figure 27 Singularité de type 2 lorsque A_1, C et A_2 sont colinéaires

a) Tout d'abord, il faut calculer la position de A_2 et A_2':

Le vecteur $\mathbf{u}_f$ est calculé pour connaître l'orientation du vecteur:

$$\mathbf{u}_f = \frac{\mathbf{r}_{OA_1} - \mathbf{r}_{OO_2}}{\mathbf{r}^T_{O_2A_1}\mathbf{r}_{O_2A_1}} \tag{4.16}$$

La longueur des vecteurs **f** et **h** est aussi calculée pour déterminer la distance entre A_2 et O_2 par rapport à l'axe O_2A_1:

$$\mathbf{f}^T\mathbf{f} = \frac{l_1^2 + (2l_2)^2 + \mathbf{r}^T_{O_2A_1}\mathbf{r}_{O_2A_1}{}^2}{2\mathbf{r}^T_{O_2A_1}\mathbf{r}_{O_2A_1}} \tag{4.17}$$

$$\mathbf{h}^T\mathbf{h} = \sqrt{l_1^2 - \mathbf{f}^T\mathbf{f}^2} \tag{4.18}$$

Finalement, la position des points A_2 et A_2' se calcule à l'aide de l'équation suivante :

$$\mathbf{r}_{OA_2'} = \mathbf{r}_{OO_2} + \mathbf{f}^T\mathbf{f}\mathbf{u}_f - \mathbf{h}^T\mathbf{h}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\mathbf{u}_f \tag{4.19}$$

$$\mathbf{r}_{OA_2} = \mathbf{r}_{OO_2} + \mathbf{f}^T\mathbf{f}\mathbf{u}_f + \mathbf{h}^T\mathbf{h}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\mathbf{u}_f \tag{4.20}$$

b) Il faut ensuite calculer la position de C et C' :

$$\mathbf{r}_{OC} = \frac{\mathbf{r}_{OA_1} + \mathbf{r}_{OA_2}}{2} \tag{4.21}$$

$$\mathbf{r}_{OC'} = \frac{\mathbf{r}_{OA_1} + \mathbf{r}_{OA_2'}}{2} \tag{4.22}$$

c) Les positions de l'effecteur *C* constituent des positions où le robot est en singularité de type 2. Le mode de fonctionnement est facile à déterminer puisque la position des articulations A_1 et A_2 est connue. Les positions cartésiennes en singularité de type 2 peuvent alors être classées par mode de fonctionnement.

Finalement, les positions articulaires correspondantes sont calculées:

$$q_2 = \text{atan}\,2\left(y_{OA_2} - y_{OO_2}, x_{OA_2} - x_{OO_2}\right) \tag{4.23}$$

$$q_2' = \text{atan}\,2\left(y_{OA_2'} - y_{OO_2}, x_{OA_2'} - x_{OO_2}\right) \tag{4.24}$$

Les deux positions articulaires $[q_1, q_2]$ et $[q_1, q_2']$ constituent une singularité de type 2.

4.4.2 Déterminer si le robot est en singularité de type 2

Pour déterminer si le robot est en singularité de type 2, il faut d'abord calculer la position cartésienne des articulations passives A_1 et A_2. Par la suite, le robot est en singularité de type 2 si les positions cartésiennes d'A_1 et A_2 sont égales ou si la distance entre A_1 et A_2 est égale à la somme des longueurs des liens distaux ($l_2 + l_4$).

4.4.3 Représentation des singularités de type 2

Dans la figure 21, les croix rouges représentent les singularités de type 2 lorsque A_1 et A_2 coïncident. Les autres courbes rouges représentent les singularités de type 2 lorsque les liens distaux sont colinéaires et définissent aussi la limite où le robot demeure assemblé. Les traces rouges de la figure 28 représentent les singularités de type 2 dans l'espace cartésien, pour chaque mode de fonctionnement.

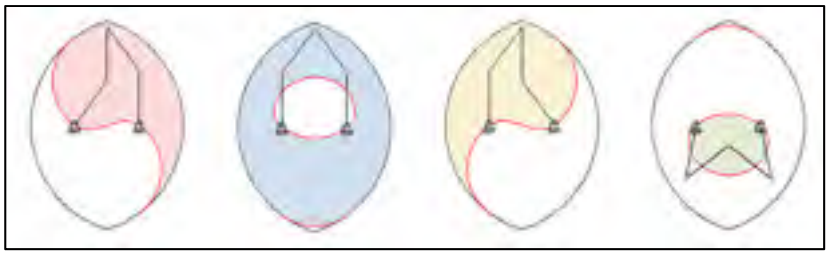


Figure 28 Lieux de singularité de type 2 par mode de fonctionnement

4.5 Interférences mécaniques

Les interférences mécaniques viennent ajouter une complexité supplémentaire dans le calcul de l'espace de travail réel du robot. En fait, bien que la conception du robot ait été réalisée de sorte que son espace de travail soit optimisé, il en réside des zones que le robot ne peut atteindre. Ces zones inaccessibles sont le résultat de trois principales contraintes:

- Distance minimale entre les liens proximaux
- Angle minimal entre les liens distaux
- Angle maximal entre les liens distaux

4.5.1 Contact entre les liens proximaux

Le premier type d'interférences mécaniques à considérer est la distance minimale entre les liens proximaux. Ces membres peuvent entrer en contact puisqu'ils sont sur le même plan *xy*. Ainsi, l'axe central de ces bras doit être séparé d'au moins 40 mm, soit la largeur des bras (figure 29).

Figure 29 Contact entre les liens proximaux

Pour ce faire, le problème est divisé en deux parties. Tout d'abord, il faut déterminer si l'axe central des membres proximaux se croise (figures 30 et 31). À partir des positions des articulations, les équations paramétriques de l'axe central des membres sont calculées selon les équations suivantes :

$$a_i = y_{OO_i} - y_{OA_i} \tag{4.25}$$

$$b_i = x_{OA_i} - x_{OO_i} \tag{4.26}$$

$$c_i = x_{OO_i}\, y_{OA_i} - x_{OA_i}\, y_{OO_i} \tag{4.27}$$

où i = 1, 2

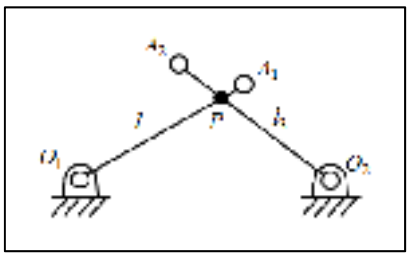


Figure 30 Contact entre les liens proximaux

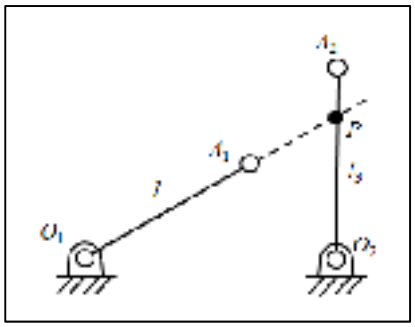


Figure 31 Faux contact entre les liens proximaux

Le point d'intersection entre ses droites est calculé :

$$x_{OP} = \frac{\left(\frac{b_2 c_1}{b_1} - c_2\right)}{\left(a_2 - \frac{b_2 a_1}{b_1}\right)} \tag{4.28}$$

$$y_{OP} = -\frac{a_1 x_{OP} + c_1}{b_1} \tag{4.29}$$

Le point calculé est le point d'intersection entre les droites, mais il n'est pas nécessairement situé sur les liens (figure 31). Il faut alors calculer la distance entre chaque articulation et le point d'intersection. Le point est situé sur les membres et il s'agit d'une interférence mécanique si la condition suivante est respectée :

$$\mathbf{r}_{PO_1}^T \mathbf{r}_{PO_1} > l_1,\ \mathbf{r}_{PA_1}^T \mathbf{r}_{PA_1} > l_1,\ \mathbf{r}_{PO_2}^T \mathbf{r}_{PO_2} > l_3 \text{ et } \mathbf{r}_{PA_2}^T \mathbf{r}_{PA_2} > l_3 \tag{4.30}$$

Si les axes centraux des liens proximaux ne se touchent pas, il faut ensuite calculer la distance la plus petite entre l'articulation passive d'un membre et l'axe central de l'autre membre. En fait, si les liens ne se coupent pas, le point sur un lien qui est le plus près de l'autre est toujours l'articulation passive et la distance entre les liens ne doit pas être

inférieure à la distance minimale. La distance entre un point et une droite est la plus petite lorsque l'axe formé par ce point et un autre point projeté sur la droite est perpendiculaire à la droite (figures 32 et 33).

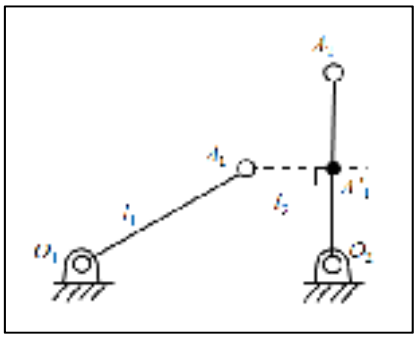


Figure 32 Distance minimale entre les liens proximaux

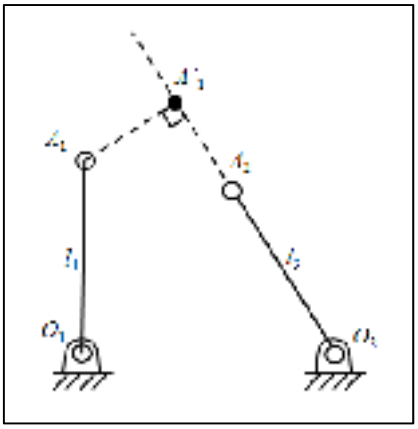


Figure 33 Cas où le point projeté n'est pas sur le membre

Le point projeté A_j' sur la droite A_jO_i qui est le plus près du point A_j est calculé de la façon suivante:

$$u = \frac{\left(x_{OA_j} - x_{OO_i}\right)\left(x_{OA_i} - x_{OO_i}\right) + \left(y_{OA_j} - y_{OO_i}\right)\left(y_{OA_i} - y_{OO_i}\right)}{l_k^2} \tag{4.31}$$

$$x_{OA'_j} = x_{OO_i} + u(x_{OA_i} - x_{OO_i}) \tag{4.32}$$

$$y_{OA'_j} = y_{OO_i} + u\left(y_{OA_i} - y_{OO_i}\right) \tag{4.33}$$

où $i = 1, 2, j = 1, 2, i \neq j, k = 1$ si $i = 1$ et $k = 3$ si $i = 2$

À la figure 33, le point projeté n'est pas situé sur le lien. Dans ce cas, il n'y a pas d'interférence mécanique. Il faut donc s'assurer que le point projeté est bien sur le lien. La distance entre ce point et les articulations du membre est calculée et les conditions suivantes doivent être respectées:

$$\mathbf{r}^T_{A'_jO_i}\mathbf{r}_{A'_jO_i} > l_k \text{ et } \mathbf{r}^T_{A'_jA_i}\mathbf{r}_{A'_jA_i} > l_k \tag{4.34}$$

où $i = 1,2, j = 1, 2, i \neq j, k = 1$ si $i = 1$ et $k = 3$ si $i = 2$.

La distance entre celui-ci et le point A_j et A'_j est calculée. Il s'agit de la plus courte distance les liens et elle doit être supérieure à la distance minimale de 40 mm pour éviter des collisions.

4.5.2 Angle minimal et maximal entre les liens distaux

L'angle entre les liens distaux peut aussi constituer une interférence mécanique. Tout d'abord, comme il est démontré à la figure 34, le tuyau d'air pour la ventouse vient empêcher le mécanisme de fermer plus. De plus, lorsque le robot approche des singularités de type 2 (A_1 et A_2 sont presque coïncidant), le comportement de celui-ci devient imprévisible. Il faut donc éviter de s'approcher trop près de cette singularité.

Figure 34 Passage du tuyau d'air

Les équations paramétriques des droites formées par les membres distaux sont calculées :

$$a_i = y_{OC} - y_{OA_i} \tag{4.35}$$

$$b_i = x_{OA_i} - x_{OC} \tag{4.36}$$

$$c_i = x_{OC}\, y_{OA_i} - x_{OA_i} y_{OC} \tag{4.37}$$

où $i = 1, 2$

L'angle α entre les droites est représenté à la figure 35 et il est calculé par l'équation suivante :

$$\alpha = \operatorname{atan2}\left(a_1 b_2 - a_2 b_1, a_1 a_2 + b_1 b_2\right) \tag{4.38}$$

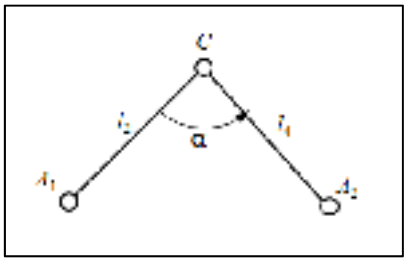


Figure 35 Angle entre les liens distaux

Si l'angle entre les liens distaux est supérieur ou inférieur aux valeurs qui seront définies expérimentalement, le robot est en interférence mécanique.

4.5.3 Stratégie pour calculer les interférences mécaniques

Pour calculer l'ensemble des interférences mécaniques, il faut procéder par discrétisation des deux espaces de travail. Pour l'espace cartésien, une résolution de un millimètre sera choisie. La zone est définie comme celle à l'intérieur des singularités externes du robot soit l'espace atteignable théorique. Le balayage est fait de sorte que toutes les positions soient testées. À chacune de ces positions, tous les modes de fonctionnement sont vérifiés, car les interférences mécaniques diffèrent d'un mode à l'autre. Pour l'espace articulaire, une résolution d'un degré est choisie. L'espace articulaire sera balayé de $-180°$ à $180°$ pour q_1 et la même chose pour q_3. Ainsi, tout l'espace articulaire sera couvert. La cinématique directe est calculée dans le mode d'assemblage positif à chacune des positions. Si elle est valide, c'est-à-dire que le robot n'est pas déconnecté, les calculs sont faits pour déterminer si le robot est en interférence mécanique. Ainsi, une fois les deux balayages exécutés, l'espace réel du robot sera défini.

4.6 Espaces de travail réel et théorique

Les interférences mécaniques amputent grandement l'espace de travail du robot. Heureusement, le changement de mode de fonctionnement est possible pour atteindre une zone qui est inatteignable dans un autre mode de fonctionnement. Une zone derrière le robot devient cependant inatteignable. Les interférences mécaniques ont été estimées dans cette section, elles seront validées dans le chapitre 7 en fonction des résultats obtenus par les tests sur le prototype. Voici des figures qui représentent l'espace de travail de chaque mode de fonctionnement une fois les interférences mécaniques considérées. À gauche, l'espace théorique est illustré et, à droite, l'espace réel.

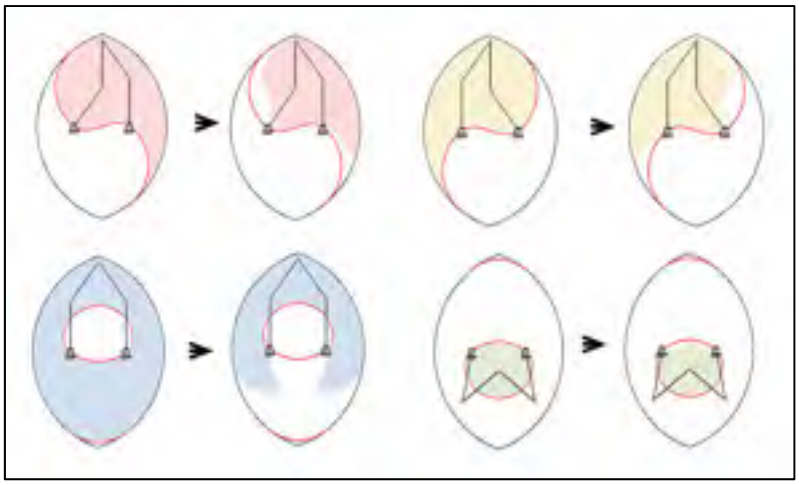


Figure 36 Réduction de l'espace cartésien due aux interférences mécaniques, pour chacun des quatre modes de fonctionnement

Dans l'espace articulaire (Figure 37), les interférences mécaniques sont représentées par des zones noires, alors que les zones blanches ne sont pas atteignables, même en théorie.

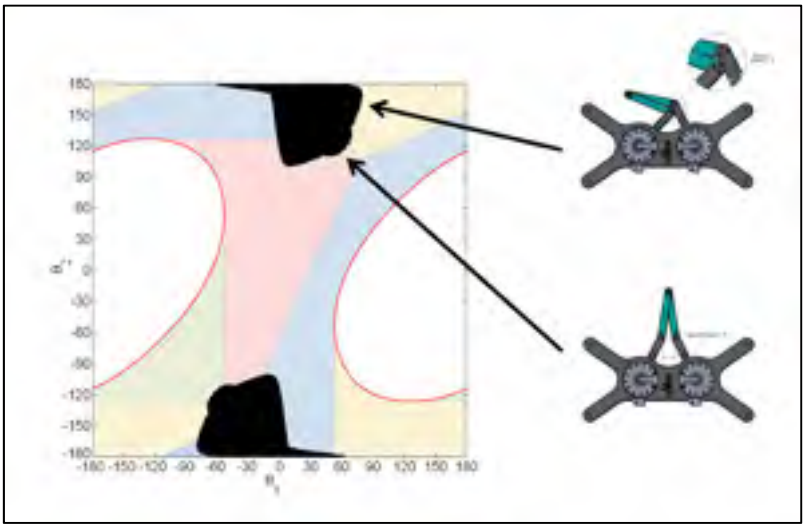


Figure 37 Interférences mécaniques (en noir) dans l'espace articulaire

4.7 Conclusion

Ce chapitre a permis de définir l'espace de travail du robot. Cette étape est très importante, car elle permet de déterminer les positions où le robot est en singularité. De plus, en calculant les interférences mécaniques, l'espace de travail réel du robot est défini. Ces calculs sont nécessaires pour la planification de trajectoire afin d'éviter de générer des trajectoires impossibles à réaliser. De plus, les calculs permettent d'avoir un visuel de l'espace de travail qui est très utile lors de l'analyse du comportement du robot.

CHAPITRE 5

MISE EN ŒUVRE DU ROBOT DEXTAR

5.1 Introduction

Ce chapitre explique en détail la mise en œuvre du robot DexTAR. Tout d'abord, la composition du système est expliquée. Les détails mécaniques de l'architecture sont omis, mais ils sont bien expliqués dans l'article de Campos (Campos et al., 2010). Ensuite, les méthodes utilisées pour l'identification des paramètres sont présentées. Ce dernier point est brièvement expliqué pour montrer les techniques utilisées, mais certains détails ne sont pas présentés puisqu'il s'agit de travaux exécutés en parallèle à ce mémoire. Les composantes matérielles et logicielles du contrôleur sont expliquées ainsi que des modules complémentaires qui y ont été ajoutés pour assurer la fiabilité et la sécurité du robot.

5.2 Composition du système

Le système est composé de quatre parties : *i*) la mécanique du robot; *ii*) les moteurs combinés aux variateurs; *iii*) le contrôleur du robot qui comprend le système d'acquisition des données, l'ordinateur temps réel et le module d'asservissement qui s'appuie sur une identification du système et *iv*) l'interface utilisateur, développée pour générer et transférer facilement les trajectoires au contrôleur du robot. La figure 38 présente la composition du système de façon schématisée.

Figure 38 Composition du système

5.2.1 La mécanique du robot

La conception mécanique du robot a été réalisée au préalable. Il faut mentionner que le robot a été spécialement conçu pour permettre le changement de mode de fonctionnement. Ainsi, les articulations actives et les articulations intermédiaires peuvent effectuer un tour complet sans interférences. Les bras du robot n'interfèrent pas non plus avec sa base. Une ventouse est installée à l'effecteur terminal et un actionneur pneumatique miniature a été choisi pour le déplacement vertical. L'effecteur permet donc de prendre et de déposer des billes sur une plaque trouée couvrant tout l'espace de travail. Des roulements à billes à angle de haute précision ont été sélectionnés pour les articulations passives. Ceux-ci permettent un

assemblage qui réduit les jeux mécaniques. Pour acheminer l'air à l'effecteur, des tuyaux d'air traversent l'intérieur de l'architecture de chaque bras. Le système pneumatique est conçu de sorte que le robot peut tourner dans tous les sens sans restriction. Pour ce faire, les adaptateurs pneumatiques sélectionnés sont munis de roulements à billes.

5.2.2 Les moteurs combinés aux variateurs

Les moteurs utilisés sont de type « hollow shaft » pour permettre le passage des tuyaux d'air comprimé. Les moteurs utilisés sont sans engrenage (« direct-drive ») de façon à ce qu'il n'y ait aucun jeu mécanique (« backslash »). Les encodeurs utilisés sont absolus de type sinusoïdal pour assurer une très grande précision. Les variateurs sont utilisés dans le mode de contrôle de courant (qui est approximativement proportionnel au couple pour ce type de moteur). Un module à l'intérieure du variateur simule un encodeur incrémental qui est utilisé pour asservir le robot. Le signal de commande (la consigne de couple des moteurs) est quant à lui transféré du module d'asservissement aux variateurs via un signal analogique.

5.2.3 Le contrôleur

Le système d'acquisition des données

La carte électronique utilisée pour l'acquisition des signaux des moteurs du robot est le modèle Q8 de la compagnie Quanser. Cette carte possède suffisamment d'entrées et de sorties discrètes et analogiques. Parmi les signaux nécessaires, il y a le signal analogique de la consigne de couple tel que décrit précédemment pour chacun des moteurs ainsi que les entrées discrètes pour le compte des impulsions des encodeurs. Il y a aussi des sorties discrètes qui sont utilisées pour activer les variateurs et commander le cylindre pneumatique et la ventouse.

L'ordinateur temps réel

La carte d'acquisition est intégrée dans un ordinateur indépendant de l'interface utilisateur. Cet ordinateur comprend aussi le contrôleur fonctionnant en temps réel grâce au système d'exploitation « xPC Target ». Celui-ci permet de télécharger un programme de contrôleur conçu sur un autre ordinateur fonctionnant sous Windows. Le logiciel d'édition du contrôleur est Simulink de la compagnie MathWorks. Un lien Ethernet fait le pont entre l'ordinateur dédié et l'ordinateur Windows qui contient aussi l'interface utilisateur. Ce lien permet notamment d'envoyer les trajectoires désirées, les signaux pour démarrer la trajectoire et il permet de transférer les données résultantes du suivi d'une trajectoire du contrôleur vers l'interface utilisateur.

L'identification des paramètres

Le contrôleur du robot s'appuie sur le modèle dynamique du système. Pour que ce modèle corresponde au système réel, une identification des paramètres a été réalisée. Pour y parvenir, le problème a été divisé en deux parties : l'identification du frottement et des paramètres moteurs et l'identification des paramètres dynamiques du robot. La première partie permet de connaitre le comportement de chaque moteur. La seconde partie permet d'obtenir un modèle dynamique du robot indépendant des moteurs. Les deux parties sont ensuite fusionnées pour composer un modèle dynamique complet.

L'identification du frottement et des paramètres des moteurs est une étape importante dans la réalisation du contrôleur. L'identification est basée sur la méthode de Rohrig (Rohrig et Jochheim, 2001). La méthode procède à l'identification de l'inertie du moteur, du gain, du frottement visqueux et du frottement de Coulomb ondulatoire qui est modélisé par une série de Fourier. Le modèle représente les paramètres du moteur en fonction de sa position, de sa vitesse et de son accélération.

L'identification des paramètres dynamique du robot est basée sur la méthode de Khalil (Khalil et Dombre, 2002). Cette étape est nécessaire pour prédire les couples appliqués aux moteurs lors des déplacements et ainsi les introduire dans la boucle de contrôle pour atteindre les critères de performances visés tout en assurant la précision du suivi. Cette méthode consiste à déterminer les équations dynamiques du système et à regrouper les paramètres inconnus de sorte qu'il soit possible de les identifier en déplaçant le robot. La trajectoire utilisée pour l'identification doit solliciter le plus possible le robot pour que les paramètres soient le plus près de la réalité possible. Lors du suivi de la trajectoire, la position, la vitesse, l'accélération et les couples sont acquis et les paramètres sont ensuite calculés à l'aide de la méthode des moindres carrés.

L'identification des paramètres constitue une étape complexe. Cette étape a été réalisée en parallèle à ce mémoire. Les détails de l'identification ne sont donc pas présentés.

L'asservissement

Le contrôleur utilisé pour l'asservissement du robot est un PID avec une loi de commande linéarisante et une compensation du frottement (Craig, 1989). La loi de commande linéarisante comprend le modèle dynamique du robot identifié précédemment. Elle nécessite à l'entrée le résultat de la loi de commande du PID, la position et la vitesse actuelle des actionneurs ainsi que l'accélération désirée. Le compensateur du frottement ajoute le couple de frottement visqueux estimé directement à la consigne de couple envoyée aux actionneurs en fonction de la vitesse désirée. Le schéma de commande est présenté à la figure 39.

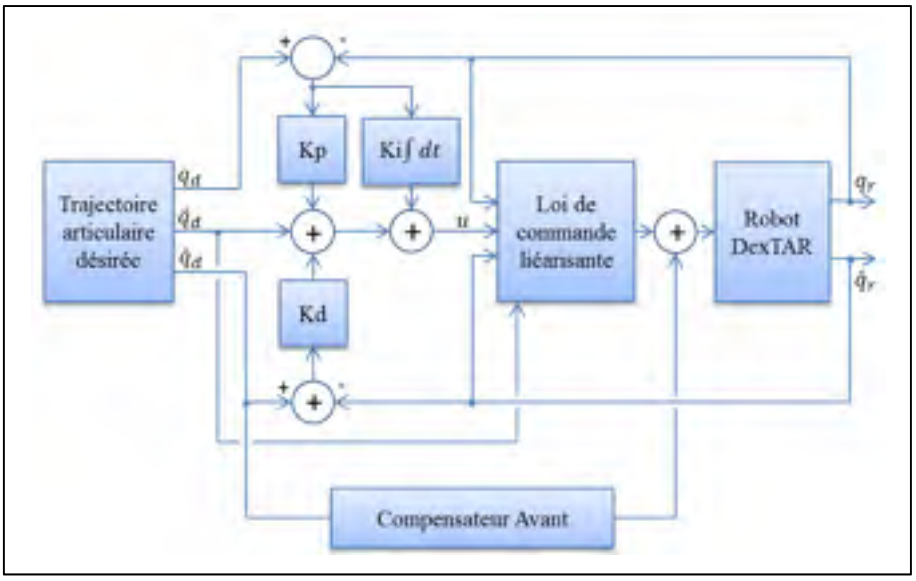


Figure 39 Contrôleur utilisé

De façon générale, la loi de commande linéarisante est donnée par :

$$\tau = \mathbf{M}(\mathbf{q})u + \mathbf{F}(\dot{\mathbf{q}}, \mathbf{q}) \tag{5.1}$$

Où $\mathbf{M}$ est la matrice de masse du modèle réduit, $\mathbf{F}$ est le vecteur des forces centrifuges et de Coriolis et τ est le vecteur des couples des moteurs.

La loi de commande du PID est alors la suivante :

$$\mathbf{u} = \ddot{\mathbf{q}}_d + \mathbf{K}_p\tilde{\mathbf{q}} + \mathbf{K}_i\int_0^t \tilde{\mathbf{q}}dt + \mathbf{K}_d\dot{\tilde{\mathbf{q}}} \tag{5.2}$$

où $\mathbf{K}_p$ est la matrice des gains proportionnels, $\mathbf{K}_i$ est la matrice des gains intégrales, $\mathbf{K}_d$ est la matrice des gains dérivée, $\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q}_r$ est l'erreur de suivi de la trajectoire, tandis que *d* et *r* sont des indices qui désignent respectivement la trajectoire désirée et réelle. En supposant que le modèle dynamique identifié ainsi que le frottement correspondent parfaitement au

système réel, l'application des lois de commande permet d'exprimer l'erreur de suivi de la trajectoire sous la forme suivante :

$$\dddot{\tilde{\mathbf{q}}} + \mathbf{K}_d \ddot{\tilde{\mathbf{q}}} + \mathbf{K}_p \dot{\tilde{\mathbf{q}}} + \mathbf{K}_i \tilde{\mathbf{q}} = \mathbf{0} \tag{5.3}$$

Puis, la dynamique de l'erreur peut être exprimée sous la forme découplée suivante lorsque les matrices de gains sont diagonales :

$$\dddot{\tilde{\mathbf{q}}}_i + k_{di} \ddot{\tilde{\mathbf{q}}} + k_{pi} \dot{\tilde{\mathbf{q}}} + k_{ii} \tilde{\mathbf{q}} = 0 \text{ où } i = 1,2 \tag{5.4}$$

Les gains de chaque articulation *i* sont alors calculés par imposition des pôles (Murray, Li et Sastry, 1994) de l'équation d'erreur (5.4). Dans ce mémoire, nous avons choisi de calculer les gains de façon à imposer un seul pôle de multiplicité trois. Les gains sont donc donnés par :

$$k_{pi} = 3\lambda^2 \tag{5.5}$$

$$k_{ii} = \lambda^3 \tag{5.6}$$

$$k_{di} = 3\lambda \tag{5.7}$$

où λ est le pôle de multiplicité trois qui est déterminé par expérimentation de façon à réduire le plus possible l'erreur de suivi sans pour autant rendre le système vibratoire. Ainsi, λ a été fixé à 60.

Modules complémentaires

Au fil des expérimentations, des modules complémentaires ont été élaborés pour assurer le bon fonctionnement du robot. Tout d'abord, une sortie discrète de la carte d'acquisition est

utilisée pour activer les variateurs. Ainsi, le robot ne peut pas bouger tant que le contrôleur n'est pas en marche. Une protection a également été ajoutée pour couper l'alimentation lorsque l'erreur de positionnement devient trop grande. Ce problème pourrait survenir lors d'un bris d'un des câbles d'encodeur par exemple.

Ensuite, comme les moteurs du robot peuvent admettre des déplacements angulaires supérieurs à 360°, un module qui ajuste la trajectoire en fonction de la position réelle du robot a été ajouté. Avant d'exécuter une trajectoire, la position réelle du robot, soit celle cumulée dans le contrôleur, est comparée avec celle de la trajectoire. Si la position du robot est la même que la position initiale de la trajectoire, mais que la position des moteurs dans le contrôleur est à l'extérieur de l'intervalle -180^{o} à 180^{o}, un décalage est ajouté à la trajectoire pour que les positions concordent.

Un autre problème a été observé en lien avec la capacité des actionneurs de se déplacer de plus de 360°. Dans l'ordinateur « xPC », un programme gère les comptes des impulsions des encodeurs venant de la carte d'acquisition. Lorsqu'un moteur exécute plusieurs rotations complètes, le compteur de la carte Q8 peut atteindre sa limite maximale et reprendre le compte à partir de zéro. Cette situation occasionne de graves répercussions sur l'asservissement du robot puisque l'erreur de positionnement devient, en apparence, très élevée à cet instant précis. Pour contrer ce problème, un module ajoute un décalage approprié au compteur lorsque celui-ci dépasse sa valeur maximale.

De façon à ce que le robot exécute une trajectoire à tout moment quand le contrôleur est activé, un autre module a été réalisé pour indexer le tableau de la trajectoire selon la période d'échantillonnage lorsqu'une commande de départ est envoyée. Ainsi, lors de l'exécution de la trajectoire, un compteur fait évoluer un pointeur qui envoie les données de la trajectoire au contrôleur. Lorsque la trajectoire est terminée, la dernière position de celle-ci est mémorisée et le robot conserve cette position jusqu'à ce qu'une autre commande de départ soit envoyée.

Pour permettre d'exécuter des trajectoires en boucle lors de démonstrations, un module détecte la fin du suivi de la trajectoire de façon à envoyer une commande de départ et ainsi d'exécuter la trajectoire en boucle. Le module de boucles est complètement codé à l'intérieur du contrôleur. Il est activé lorsqu'une variable booléenne provenant de l'interface usager l'indique.

Le contrôleur permet l'acquisition des résultats suite à l'exécution d'une trajectoire. Cette acquisition nécessitait cependant l'arrêt complet du contrôleur. Pour pallier cet inconvénient, un module a été développé pour enregistrer les résultats d'une trajectoire lors de son exécution de façon à pouvoir les charger sans arrêter le contrôleur. Le temps d'enregistrement est cependant limité à 10 secondes.

5.2.4 L'interface utilisateur

Pour utiliser le robot DexTAR, une application a été développée sous Matlab à l'aide du GUI (Graphical User Interface). Cette application a été créée dans l'optique de rendre conviviale l'interaction avec le robot. L'application permet d'articuler graphiquement le robot à l'aide de la souris, autant dans l'espace cartésien qu'articulaire, de façon à faciliter la génération de trajectoires. L'interface permet ensuite de simuler la poursuite de ces trajectoires et finalement, de faire faire le suivi en temps réel par le robot. Les paramètres du robot comme les dimensions et les interférences mécaniques sont facilement modifiables de sorte que l'espace de travail est recalculé automatiquement. Des modules complémentaires ont été développés permettant d'ajouter des obstacles dans l'espace cartésien ou de générer des trajectoires correspondant à des formes géométriques. Pour obtenir plus de détails sur le fonctionnement de l'application, un guide d'utilisation est disponible (Bourbonnais, 2012). Un aperçu de l'interface principale est également montré à la figure 40.

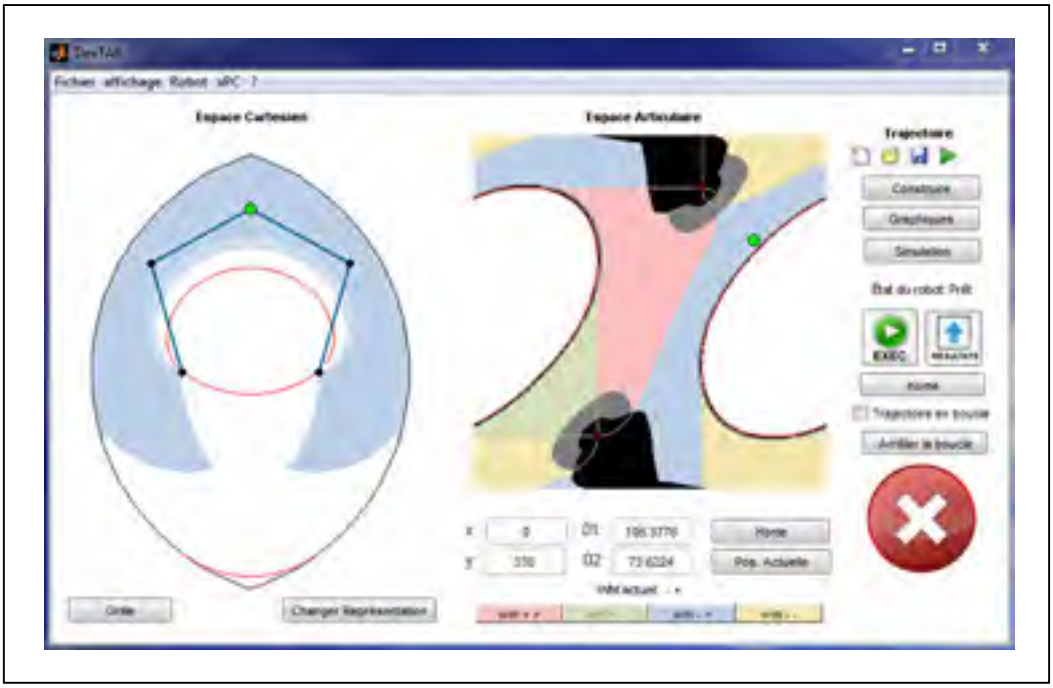


Figure 40 Interface utilisateur principale

5.3 Conclusion

Ce chapitre a présenté tous les éléments qui ont permis la mise en œuvre du robot DexTAR. En particulier, le contrôleur et les différents modules ainsi que l'interface utilisateur ont été développés au cours de ce mémoire ou en parallèle, et ce, spécifiquement pour le robot DexTAR.

CHAPITRE 6

PLANIFICATION DE TRAJECTOIRE

6.1 Introduction

La planification de trajectoire est une étape importante dans l'optimisation des déplacements des robots. Ce chapitre présente un bref retour sur les méthodes décrites dans la revue bibliographique (section 2.2) afin d'en faire ressortir les plus pertinentes. Ensuite, une section présentera la stratégie utilisée pour générer une trajectoire optimale. Cette stratégie nécessite trois étapes qui seront détaillées par la suite : la génération de l'environnement, le calcul des chemins possibles et la génération des trajectoires possibles. Finalement, un algorithme stochastique de recherche de trajectoire est présenté.

6.2 Retour sur les méthodes proposées

Les méthodes proposées dans la revue bibliographique ont chacune des avantages et des inconvénients. Les méthodes qui ont été retenues l'ont été pour leur simplicité de mise en œuvre ou pour leur efficacité. Dans tous les cas, la planification de trajectoire du chemin optimale se fait en deux étapes :

1. La première étape consiste à trouver un chemin sans obstacles entre la position initiale et les différentes positions finales possibles. Un chemin correspond à un ensemble de points qui ne dépend pas du temps contrairement à une trajectoire qui correspond au chemin en fonction du temps.

2. La deuxième étape consiste à générer les trajectoires optimales entre les positions et à retenir celle qui obtient les meilleures performances selon une fonction objective déterminée.

Pour réaliser la première étape qui consiste à trouver le chemin sans obstacles entre une position initiale et finale, l'algorithme A* qui calcule le chemin le plus court sans obstacle est privilégié. Pour appliquer cet algorithme, il faut à priori obtenir une carte discrétisée de l'environnement qui indique les zones inaccessibles et les interférences mécaniques. L'algorithme A* cherche alors la position finale en réajustant sa course si le chemin direct est impossible à cause des obstacles. Il est très performant, car il n'analyse pas toutes les possibilités, mais uniquement celles qui ont les meilleures chances de fournir un résultat optimal. Il est aussi facile à comprendre et à mettre en œuvre. Cependant, le temps de calcul du chemin A* dépend de la dimension de la carte de l'environnement. Pour le robot DexTAR, une carte discrétisée de 1^{o} est utilisée et comporte 129 600 nœuds à traiter. Dans certains cas, l'algorithme peut avoir de la difficulté à converger à cause du trop grand nombre de nœuds. Ainsi, la carte de l'environnement doit être transformée en graphe de connectivité, ce qui réduit considérablement le nombre de nœuds et décroît le temps de calcul.

La seconde étape consiste à calculer les trajectoires optimisées à l'aide des chemins les plus courts calculés précédemment et un algorithme de planification de trajectoire. L'algorithme retenu est celui des splines cubiques aléatoires de (Chettibi et Lehtihet, 2002). Cet algorithme intègre le modèle dynamique du robot dans son processus d'optimisation selon la formulation présentée au chapitre 3. De plus, le processus stochastique de recherche de trajectoire évite les minimums locaux. L'algorithme ne nécessite pas nécessairement de chemin calculé préalablement, mais si l'algorithme ne converge pas après quelques itérations, des points de ce chemin peuvent être fournis afin d'aider l'algorithme. Ces points sont modifiés au besoin si l'algorithme ne les considère pas optimaux.

Finalement, les deux autres méthodes qui sont présentées servent principalement à comparer la méthode privilégiée. Ces méthodes n'intègrent pas le modèle dynamique. Elles sont utilisées généralement dans la planification de trajectoire, car elles sont simples à mettre en œuvre. La première méthode retenue s'appuie sur une ligne droite dans l'espace articulaire. En effet, il est souvent possible de tracer une ligne droite entre la position initiale et une des positions finales possibles. L'algorithme du chemin le plus court n'est alors pas nécessaire,

car cette méthode privilégie la ligne droite la plus courte. La seconde méthode consiste en la programmation non linéaire d'un chemin fixe. Cet algorithme nécessite un nombre défini de points sur le chemin le plus court. Le temps de calcul de l'algorithme dépend directement du nombre de points choisis. Cette méthode peut aussi être utilisée pour générer un chemin défini entre deux points avec des points de passage. Par exemple, cette approche peut être utilisée pour définir des formes dans l'espace cartésien ou des chemins prédéterminés par une tâche. La méthode des splines cubiques aléatoires peut aussi être adaptée pour planifier des chemins fixes en y effectuant quelques modifications au niveau de l'algorithme stochastique de recherche de trajectoire.

Bien que certaines méthodes proposent l'optimisation de la trajectoire via une fonction objective qui inclut plusieurs critères, le seul critère utilisé est le temps minimal de la trajectoire pour faciliter la comparaison entre les méthodes.

6.3 Stratégie pour générer une trajectoire optimale

Pour générer une trajectoire optimale, il faut d'abord avoir des données sur l'environnement où le robot se déplacera : soit l'espace articulaire. La première étape pour générer une trajectoire optimale consiste donc à discrétiser l'espace articulaire et ensuite générer le graphe de connectivité. Ces étapes sont réalisées en prétraitement et les données recueillies sont utilisées tant que l'environnement ne change pas.

Ensuite, il faut connaître tous les chemins possibles. Pour y arriver, il faut définir une position initiale et finale dans l'espace cartésien et les transformer dans l'espace articulaire. Le robot peut atteindre ces positions finales en empruntant plusieurs chemins. Il faut donc déterminer les chemins possibles et finalement les calculer en utilisant un algorithme de recherche du chemin le plus court.

Puis, toutes les trajectoires possibles sont générées. Pour ce faire, une des méthodes de planification de trajectoire est choisie : *i*) la méthode des splines cubiques aléatoires (chemin

optimal) ; *ii*) la méthode des splines cubiques aléatoires (chemin fixe) ; *iii*) la méthode s'appuyant sur une ligne droite dans l'espace articulaire ; ou encore *iv*) la programmation non linéaire.

Finalement, une fois toutes les trajectoires possibles générées, il faut déterminer la trajectoire optimale selon le critère établi.

L'ensemble des étapes de la stratégie pour générer une trajectoire optimale peut donc être définie de façon schématique par la figure 41.

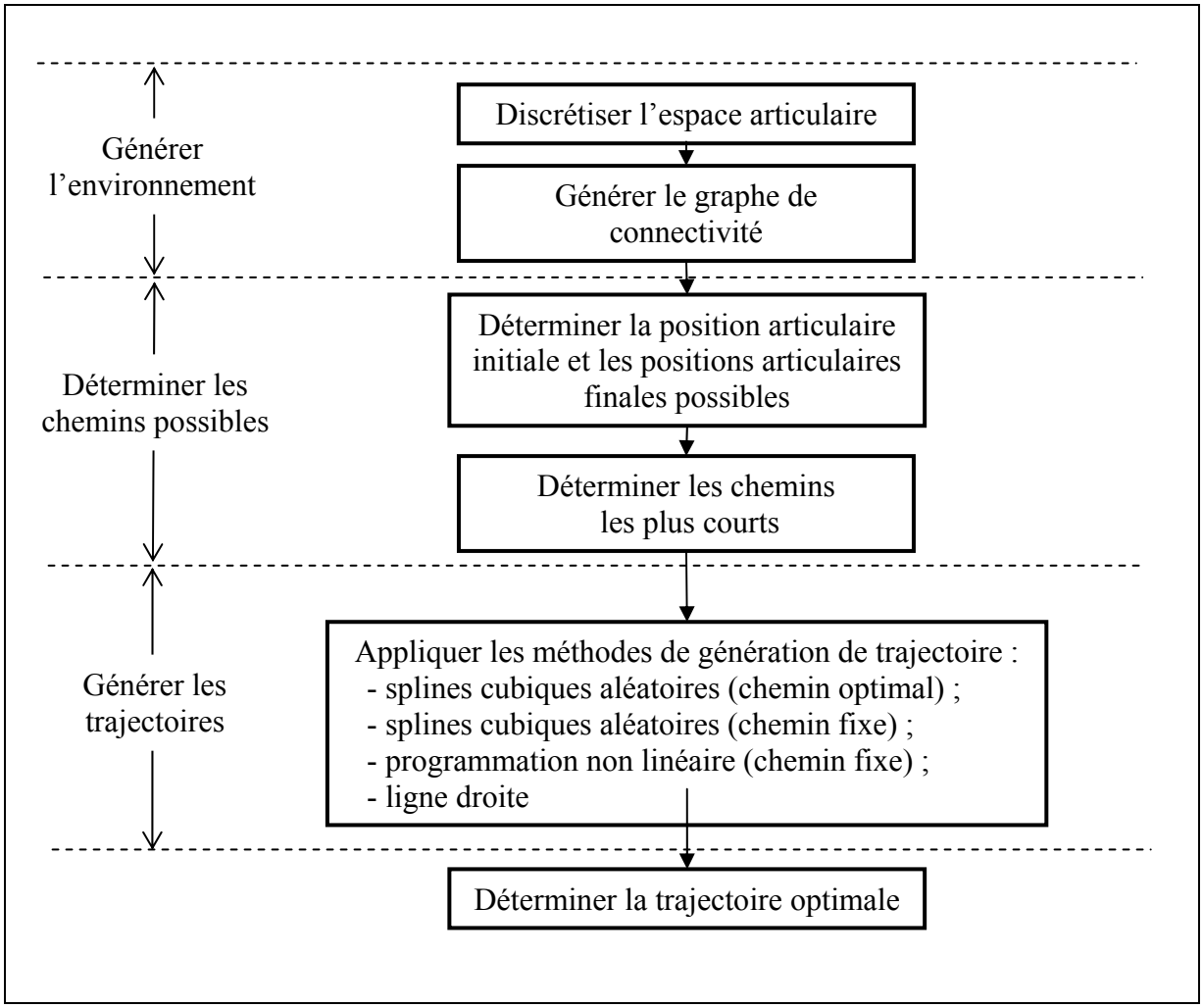


Figure 41 Stratégie pour générer une trajectoire optimale

6.4 Générer l'environnement

6.4.1 Discrétiser l'espace articulaire

La discrétisation de l'espace articulaire permet de construire la carte de l'environnement, soit l'espace atteignable du robot. La discrétisation consiste à balayer tout l'espace articulaire avec une résolution qui a été fixée à 1°. La carte est donc une matrice booléenne de 360×360 dont chaque coordonnée constitue une position articulaire et la valeur booléenne indique une position inatteignable (1) ou une position atteignable (0). Le processus de génération de cette matrice est expliqué à la figure 42. Une fois le processus de balayage terminé, la carte est prête à être utilisée pour calculer le graphe de connectivité.

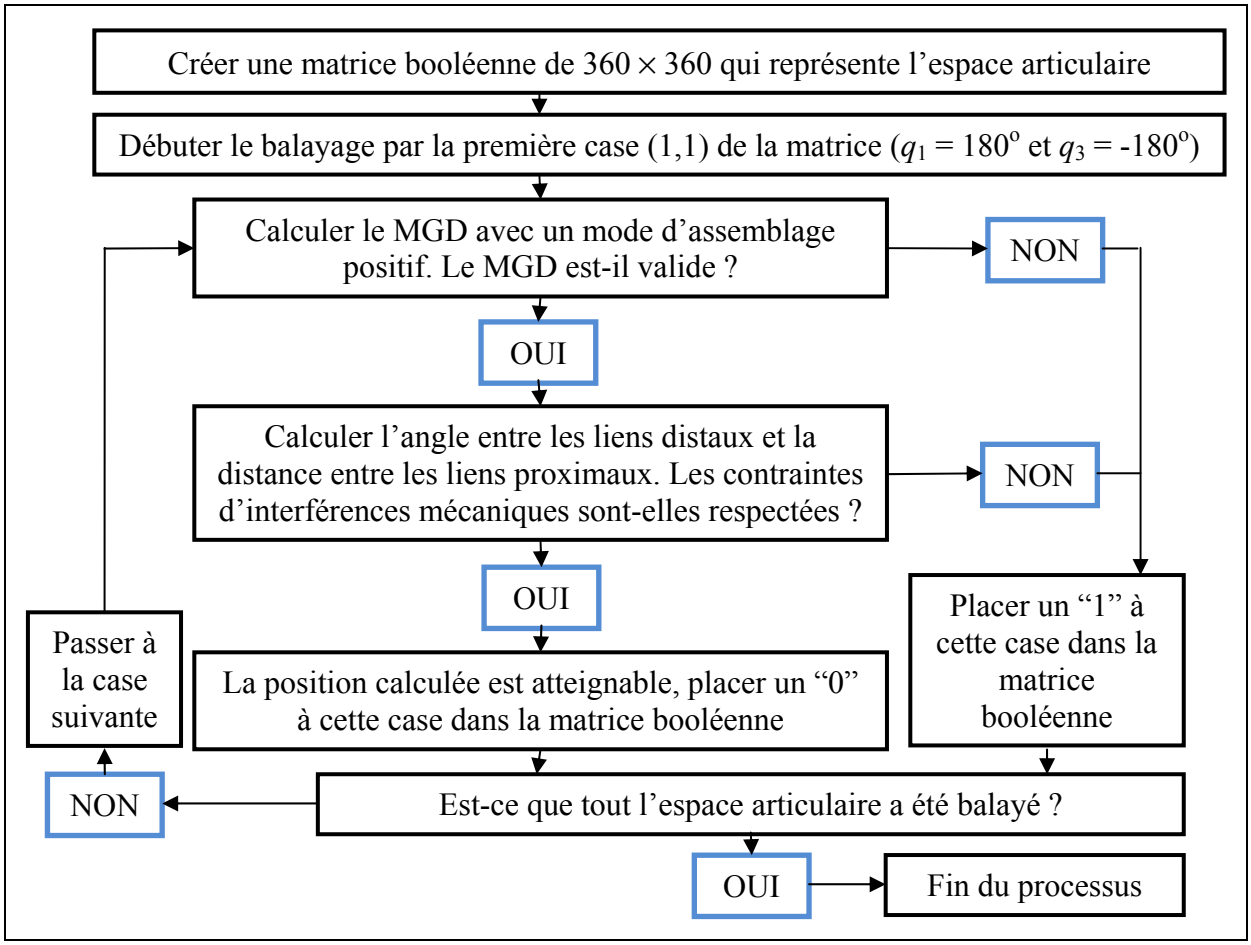


Figure 42 Génération de la carte de l'environnement

6.4.2 Créer le graphe de connectivité

L'algorithme Sivia (Jaulin et al., 2001) gère le processus de constitution du pavé qui correspond à une région de l'environnement exempt d'obstacle. Le graphe de connectivité définit la carte de l'environnement. Il comprend les limites de chaque pavé (par exemple les limites des articulations motorisées du DexTAR : $q_1$min, $q_1$max, $q_3$min et $q_3$max) et sa position ainsi que la distance par rapport aux autres pavés adjacents. Un algorithme doit générer au préalable les sous-pavés (voir figure 43).

Une fois les pavés créés, il faut calculer le graphe de connectivité. Il est constitué de deux tableaux : un qui comprend tous les pavés de l'environnement (la liste finale) et un autre qui comprend la matrice d'adjacence, soit les relations entre les pavés. Cette matrice est de dimensions $m \times m$, où m correspond au nombre de pavés. Pour constituer la matrice d'adjacence, il faut comparer chaque élément de la liste finale. À chaque élément, il faut trouver à l'aide des valeurs limites ceux qui se touchent. Une fois qu'un contact est trouvé, il faut calculer la distance centre-à-centre entre les deux pavés. Cette distance est placée dans la matrice d'adjacences à l'intersection de la ligne correspondante au premier sous-pavés et de la colonne du second sous-pavés et vice-versa. Si les pavés ne se touchent pas, la distance entre les pavés dans le tableau est fixée à l'infini.

Dans le cas d'une résolution minimale de un degré, la mémoire nécessaire pour la matrice d'adjacence dépasse la mémoire disponible de Matlab. L'algorithme a donc été considérablement modifié pour contrer ce problème, mais le principe demeure le même. En particulier, la matrice d'adjacence est modifiée : Elle devient un tableau $m \times n \times k$, où m correspond au nombre de sous-pavés, n correspond au nombre maximal de contacts entre les sous-pavés et $k = 2$. La première dimension de k correspond au numéro du pavé en contact et la deuxième dimension à la distance entre les deux sous-pavés.

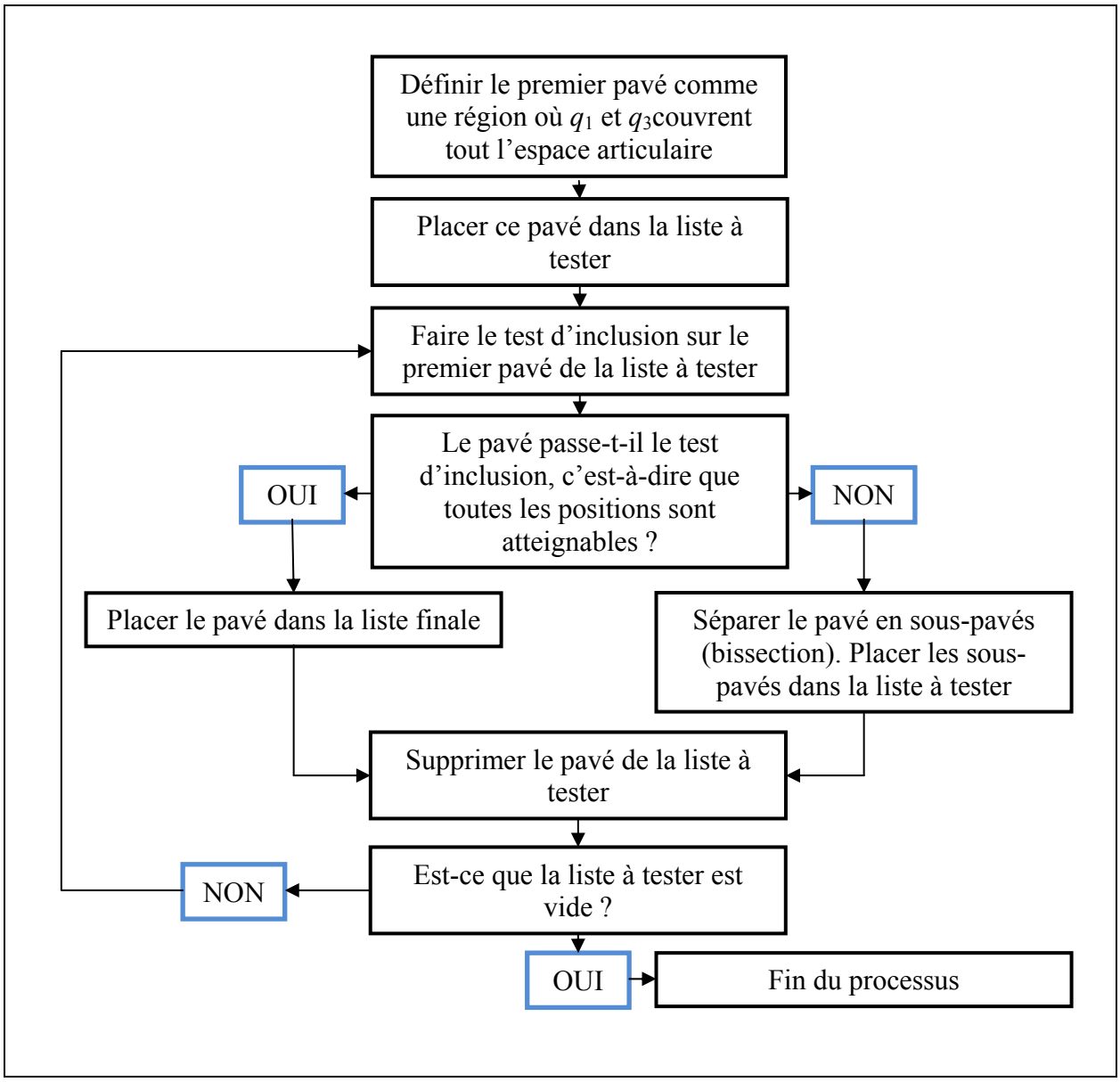


Figure 43 Génération des pavés

6.5 Déterminer les chemins possibles

L'astuce pour augmenter l'espace de travail du robot DexTAR consiste à changer de mode de fonctionnement pour atteindre des régions qui seraient inaccessibles autrement. Le chemin doit être planifié dans l'espace articulaire puisque chaque mode de fonctionnement a un espace cartésien différent.

Dans l'espace articulaire, le chemin ne peut pas toujours être une ligne droite à cause de la présence d'interférences mécaniques et la présence de zone où il n'y a pas de solution à la cinématique inverse. Par conséquent, il est nécessaire d'utiliser un algorithme qui calculera un chemin dans un environnement encombré pour éviter ces zones.

6.5.1 Déterminer les positions initiales et finales

La position initiale est unique, car le robot est physiquement dans cette position avant d'exécuter la trajectoire. Le mode de fonctionnement initial étant connu, il y aura une seule solution au modèle géométrique inverse donc une seule position articulaire. Au contraire, la position cartésienne finale correspond à plusieurs positions articulaires à cause du mode de fonctionnement final qui peut être différent du mode initial. Le processus d'optimisation détermine donc le mode de fonctionnement final optimal. Ainsi, il peut y avoir jusqu'à quatre positions articulaires finales possibles. Les positions finales retenues doivent être atteignables dans le mode d'assemblage positif et elles ne doivent pas être en interférence mécanique.

6.5.2 Déterminer les chemins les plus courts

Aux considérations expliquées dans les paragraphes précédents, il faut aussi ajouter le fait que chaque moteur peut faire plus d'un tour dans les deux directions. Le robot peut donc dépasser l'espace articulaire dont l'échelle est de -180^{o} à 180^{o} pour les deux axes (voir figure 44). Ainsi, pour une position initiale désirée (point vert), le robot aura non seulement jusqu'à quatre configurations possibles dans l'espace articulaire, mais il aura neuf façons de les

atteindre pour un total de 36 positions finales possibles (points bleus). L'algorithme de planification de trajectoire doit considérer toutes ces possibilités pour trouver une solution optimale. La trajectoire en vert illustrée à la figure 44 montre un exemple où le point final de la trajectoire optimale est à l'extérieur de la plage [−180°, 180°].

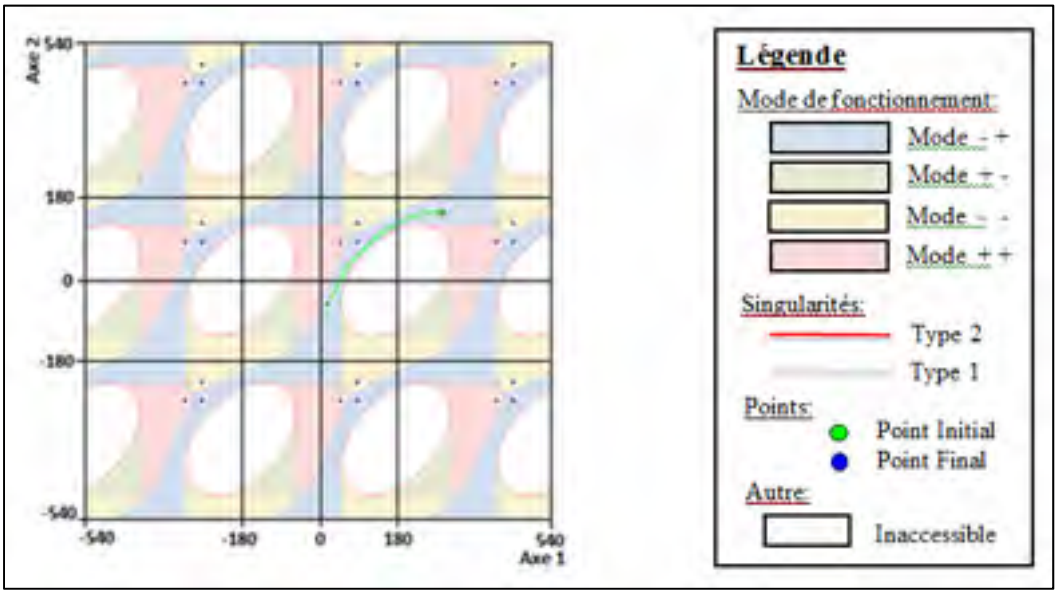


Figure 44 Espace articulaire

La recherche du chemin le plus court est utilisée pour calculer les chemins possibles entre la position initiale et les positions finales. Pour ce faire, l'algorithme A* et le graphe de connectivité sont utilisés. Lors du processus A*, l'algorithme attribue un coût à chaque déplacement. Un déplacement correspond au passage d'un sous-pavé à un autre et le coût est la distance entre le centre de ceux-ci. L'algorithme évolue avec deux liste appelées liste ouverte et liste fermée. La liste ouverte contient les nœuds étudiés. Si l'algorithme traite actuellement un nœud, il est placé dans cette liste. La liste fermée contient les nœuds qui pourraient être considérés comme une partie de la solution durant le processus de recherche. Les nœuds de la liste fermée passent préalablement par la liste ouverte pour être analysés.

Un nœud est constitué des éléments suivants :

1. Valeurs de q_1 et q_3 (les articulations motorisées) du parent, c'est-à-dire le nœud parcouru précédemment qui a mené au nœud actuel;
2. Le coût G (la distance entre le point de départ et le nœud actuel);
3. Le coût H (la distance entre le nœud actuel et le point de destination);
4. Le coût F (somme du coût G et du coût H);
5. Valeurs de q_1 et q_3 actuelles.

La figure 45 présente le processus de création des listes ouverte et fermée.

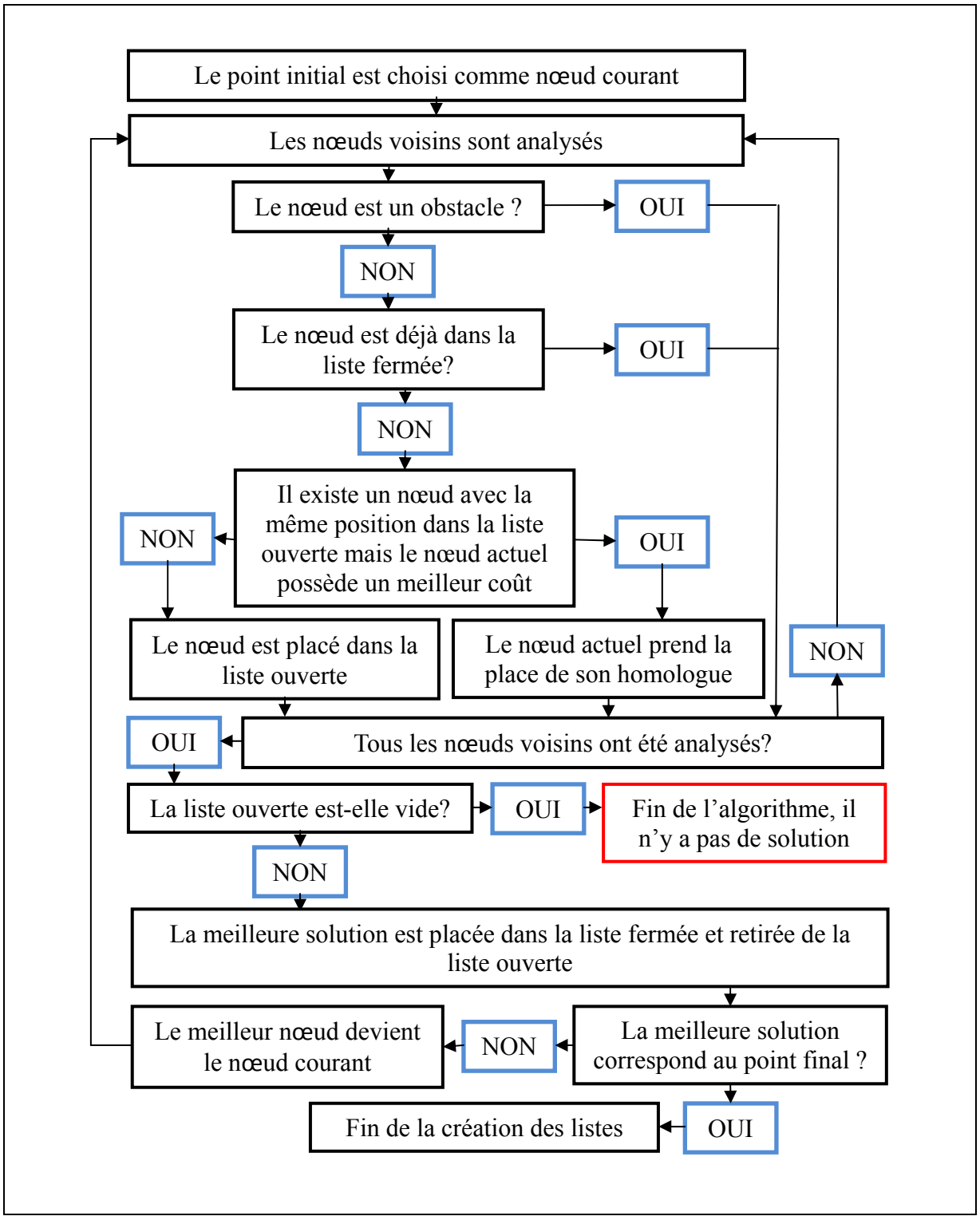


Figure 45 Processus de la création des listes

Une fois les listes constituées, la liste ouverte n'est plus utilisée tandis que la liste fermée sert à retrouver le chemin le plus court. Pour ce faire, le nœud correspondant à la position finale est d'abord sélectionné. Le nœud parent à ce dernier est retrouvé dans la liste finale et constitue le second point du chemin. Ce nœud passe ensuite au nœud à l'étude. Ces parents sont cherchés dans la liste finale et correspondent à un autre point du chemin. Ce processus est répété tant que le nœud parent ne correspond pas au point initial. Les points trouvés constituent le chemin le plus court. Comme les points aux extrémités sont arrondis à la résolution de la carte, ils sont en dernier lieu remplacés par les positions réelles. De plus, le nombre de points du chemin est réduit (entre 6 et 10 points) pour faciliter le calcul de la trajectoire et les traitements qui suivent.

6.6 Générer les trajectoires

6.6.1 Splines cubiques aléatoires (chemin optimal)

La méthode des splines cubiques aléatoires est très performante pour trouver une trajectoire point-à-point optimale (Chettibi et al., 2006). Il s'agit d'un processus itératif où les trajectoires testées sont générées à l'aide d'un algorithme stochastique. Les chances de tomber dans un minimum local sont donc réduites. La fonction objective est présentée à l'équation (6.1). Elle est un compromis entre le temps T pour réaliser la trajectoire et une moyenne quadratique de l'effort des actionneurs. Afin de comparer les trajectoires obtenues de cette méthode avec celles obtenues des autres méthodes, la fonction objective minimisera uniquement le temps de la trajectoire ($u = 1$).

$$F = \mu T + \frac{(1-u)}{2} \int_0^T \sum_{j=1}^{N} \left(\frac{\tau_j(t)}{\tau_j^{max}} \right)^2 dt \tag{6.1}$$

où $0 \leq u \leq 1$.

La trajectoire doit être telle que le couple, l'accélération et la vitesse maximale des moteurs ne doivent jamais être dépassés. Elle ne peut pas non plus entrer en contact avec les obstacles (interférences mécaniques et zones inaccessibles). On peut donc définir les contraintes ainsi :

- couples maximaux : $\left|\tau_j(t)\right| \le \tau_j^{\max}$ pour $j = 1, 2\ldots, N$ (6.2)
- accélération maximale : $\left|\ddot{q}_j(t)\right| \le \ddot{q}_j^{max}$ pour $j = 1, 2\ldots, N$ (6.3)
- vitesse maximale : $\left|\dot{q}_j(t)\right| \le \dot{q}_j^{max}$ pour $j = 1, 2\ldots, N$ (6.4)
- présence d'obstacles: $E(q_1, q_2) = 0$ (6.5)

où **E** correspond à une matrice de l'espace discrétisée (voir section 6.4.1)

Le problème consiste alors à extraire le temps de parcours T des équations d'optimisation pour calculer facilement le coût de la fonction objective. L'équation (3.59), qui exprime la dynamique du DexTAR, peut toujours être réécrite sous la forme suivante:

$$\tau_i(t) = \sum_{i=1}^{N} M_{ji}\left(\mathbf{q}(t)\right)\ddot{\mathbf{q}}_i(t) + Q_j\left(\mathbf{q}(t), \dot{\mathbf{q}}(t)\right) + G_j(\mathbf{q}(t)),\ j = 1,..,N \tag{6.6}$$

où **M** est la matrice des masses, **Q** est le vecteur des forces centrifuges et de Coriolis et **G** est le vecteur des forces de gravité. Dans le cas du robot à l'étude, **G** est nul puisque le robot bouge sur le plan *xy* alors que la force de gravité est dans la direction orthogonale à ce plan. Cela simplifie légèrement les calculs. En normalisant les variables de l'équation du mouvement relativement au couple maximal des moteurs :

$$\psi_j(t) = \frac{\tau_j(t)}{\tau_j^{max}},\ \bar{M}_{ji} = \frac{M_{ji}}{\tau_j^{max}},\ \bar{Q}_i = \frac{Q_j}{\tau_j^{max}} \text{ et } \bar{G}_j = \frac{G_j}{\tau_j^{max}}, \tag{6.7}$$

L'équation du mouvement peut alors être réécrite de la façon suivante :

$$\psi_j(t) = \sum_{i=1}^{N} \bar{M}_{ji}(\mathbf{q}(t))\ddot{q}_i(t) + \bar{Q}_j(\mathbf{q}(t), \dot{\mathbf{q}}(t)) + \bar{G}_j(\mathbf{q}(t)) \tag{6.8}$$

Grâce à ce modèle dynamique normalisé, les contraintes dynamiques seront incorporées dans la génération de la trajectoire sous forme de courbes splines, mais pour y arriver, une échelle de temps normalisée est définie de la façon suivante :

$$t = xT \Rightarrow dt = Tdx \text{ avec } x \in [0,1] \tag{6.9}$$

La vitesse et l'accélération sont alors réécrites en fonction de l'échelle de temps normalisée. La relation entre la vitesse et l'accélération de la trajectoire selon le temps et celles selon l'échelle normalisée sont les suivantes :

$$\dot{\mathbf{q}}(t) = \frac{1}{T}\mathbf{q}' \text{ et } \ddot{\mathbf{q}}(t) = \frac{1}{T^2}\mathbf{q}'' \tag{6.10}$$

où $\mathbf{q}' = \frac{d\mathbf{q}(x)}{dx}$, $\mathbf{q}'' = \frac{d^2\mathbf{q}(x)}{dx^2}$.

En substituant (6.10) dans (6.8), l'équation du mouvement devient:

$$\psi_j(x) = \frac{1}{T^2}H_j + \bar{G}_j \tag{6.11}$$

où

$$H_j = \sum_{i=1}^{N} \bar{M}_{ji} q''_i(x) + \bar{Q}_j$$

La fonction objective est alors réécrite de façon à tenir compte des couples normalisés

$$F = \mu T + \frac{1-u}{2}\int_{t_0}^{t_f}\sum_{j=1}^{N}\psi_j^{\,2}(t)\,dt \tag{6.12}$$

En remplaçant (6.11) dans (6.12), la fonction objective prend la forme suivante:

$$F = S_1 T + \frac{S_2}{T^3} + \frac{S_3}{T} \tag{6.13}$$

$$S_1 = \mu + \frac{1-u}{2}\int_0^1 \bar{G}_j^2\,dx \tag{6.14}$$

$$S_2 = \frac{(1-u)}{2}\int_0^1\sum_{j=1}^{N} H_j^{\,2}\,dx \tag{6.15}$$

$$S_3 = (1-u)\int_0^1\sum_{j=1}^{N} H_j \bar{G}_j\,dx \tag{6.16}$$

Les coefficients S_i sont alors indépendant des temps de parcours T. Ces coefficients sont donc calculés à priori et le temps minimum est calculé en trouvant le minimum de la fonction objective. La fonction objective par rapport à T a généralement l'allure suivante :

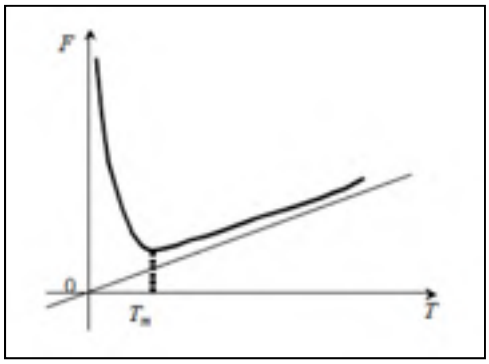


Figure 46 Fonction objective en fonction de T

La solution du problème de minimisation de la fonction objective (6.12) peut facilement être déterminée sous sa forme analytique :

$$T_m = \left(\frac{6S_2}{\sqrt{S_3^2 + 12S_1S_2} - S_3} \right)^{\frac{1}{2}} = \left(\frac{S_3 + \sqrt{S_3^2 + 12S_1S_2}}{2S_1} \right)^{\frac{1}{2}} \tag{6.17}$$

Les deux expressions de l'équation (6.17) sont équivalentes. On utilisera une ou l'autre dépendant du signe de S_3.

Une fois le temps minimal calculé, il faut s'assurer que les contraintes sont respectées. Si ce n'est pas le cas, les contraintes de vitesse et d'accélération seront remplacées par une seule contrainte définie par :

$$T \geq T^* \tag{6.18}$$

Où

$$T^* = max(T_v, T_a) \tag{6.19}$$

Pour obtenir T_v, la contrainte de vitesse est d'abord normalisée de la façon suivante:

$$\left|\dot{q}_j(t)\right| \leq \dot{q}_j^{max} \; \forall t \grave{o} [0,T] \Rightarrow \frac{1}{T}\left|q'_j(x)\right| \leq \dot{q}_j^{max} \Rightarrow T \geq \frac{\left|q'_j(x)\right|}{\dot{q}_j^{max}} \; pour \; j = 1,2,\ldots,N \tag{6.20}$$

de sorte que le temps maximum associé à la contrainte de vitesse devient

$$T_v = \max_{j=1,2,\ldots,N}\left[\max_{[0,1]}\left(\frac{\left|q'_j(x)\right|}{\dot{q}_j^{max}}\right)\right] \tag{6.21}$$

De même, pour obtenir T_a, la contrainte d'accélération peut être normalisée de la façon suivante

$$\left|\ddot{q}_j(t)\right| \le \ddot{q}_j^{max} \;\forall\, t \grave{o} [0,T] \Rightarrow \frac{1}{T^2}\left|q'_j(x)\right| \le \ddot{q}_j^{max} \Rightarrow T \ge \left(\frac{\left|q''_j(x)\right|}{\ddot{q}_j^{max}}\right)^{\frac{1}{2}} \text{ pour } j = 1,2,\ldots,N \tag{6.22}$$

de sorte que

$$T_a = \max_{j=1,2,\ldots,N}\left[\max_{[0,1]}\left(\frac{\left|q''_j(x)\right|}{\ddot{q}_j^{max}}\right)^{\frac{1}{2}}\right] \tag{6.23}$$

Puis, la trajectoire est échantillonnée uniformément pour évaluer le couple maximal. La contrainte suivante doit également être respectée à tout moment :

$$\left|\tau_j(t)\right| \le \tau_j^{max} \Rightarrow \left|\frac{1}{T^2}H_j + \bar{G}_j\right| \le 1 \;\forall\, x \in [0,1] \tag{6.24}$$

Pour ce faire, il faut évaluer les bornes minimale (T_L) et maximale (T_R) à chaque échantillon. Ces valeurs sont déterminées selon le tableau 6. Il faut ensuite conserver la plus grande valeur de T_L et la plus petite valeur de T_R de tous les échantillons. Le temps de parcours T doit donc se trouver dans l'intervalle :

$$T \grave{o} [T_L, T_R] \tag{6.25}$$

Tableau 6 Évaluation de T_L et T_R à un moment $x \grave{o} [0,1]$ (Dombre et Khalil, 2007)

<table>
<tr><td colspan="3">$H_j(x) \geq 0$</td><td colspan="3">$H_j(x) < 0$</td></tr>
<tr><td>$a_j(x) < 0$</td><td colspan="2">$a_j(x) \geq 0$</td><td>$b_j(x) < 0$</td><td colspan="2">$b_j(x) \geq 0$</td></tr>
<tr><td rowspan="2">$\varnothing$</td><td>$b_j(x) < 0$</td><td>$b_j(x) \geq 0$</td><td rowspan="2">$\varnothing$</td><td>$a_j(x) < 0$</td><td>$a_j(x) \geq 0$</td></tr>
<tr><td>$T_{Lj} = \sqrt{\frac{H_j(x)}{a_j}}$
$T_{Rj} = \sqrt{\frac{H_j(x)}{-b_j}}$</td><td>$T_{Lj} = \sqrt{\frac{H_j(x)}{a_j}}$
$T_{Rj} = +\infty$</td><td>$T_{Lj} = \sqrt{\frac{H_j(x)}{a_j}}$
$T_{Rj} = \sqrt{\frac{H_j(x)}{-b_j}}$</td><td>$T_{Lj} = \sqrt{\frac{H_j(x)}{-b_j}}$
$T_{Rj} = +\infty$</td></tr>
</table>

où $a_j(x) = 1 - \bar{G}_j(x)$ et $b_j(x) = 1 + \bar{G}_j(x)$

L'algorithme exécute deux boucles imbriquées. La boucle principale sert à déterminer le nombre de points libres qui seront utilisés pour générer la trajectoire. Elle détermine aussi le moment où l'algorithme s'arrête. À la première itération, il y a un point libre ainsi que le point initial et le point final. Une fois que la boucle interne se termine, un point entre les points déjà générés est réinjecté et ainsi de suite. Les détails de l'algorithme stochastique de recherche de trajectoires (ASRT) qui génère ses points sont donnés à la section 6.7. La boucle principale se termine lorsque la meilleure fonction objective ne change que très peu d'une itération à l'autre. Malheureusement, il est possible que l'algorithme ne converge pas pour une itération donnée. C'est le cas si un obstacle dans la trajectoire est trop gros et que la région de recherche des points de la trajectoire n'est pas suffisamment grande pour contourner l'obstacle. Dans (Hentout et al., 2010), il est recommandé d'utiliser un point approximé par une autre méthode. Dans ce mémoire, l'algorithme A* est utilisé pour choisir ces points.

La boucle interne consiste à générer aléatoirement des trajectoires selon le nombre de points. L'algorithme pour générer ces points est présenté plus bas à la section 6.7. Ensuite, pour chaque trajectoire, les contraintes sont vérifiées et la fonction objective est calculée. Si la fonction objective est meilleure que la précédente, elle la remplace et la trajectoire devient la trajectoire optimale. Le processus itératif se poursuit jusqu'à ce qu'un certain nombre d'échec préétabli soit atteint. Le nombre d'échecs est le nombre de tentatives qui n'ont pas amélioré la fonction objective. Une fois ce nombre atteint, l'algorithme retourne à la boucle principale. La figure 47 résume l'algorithme.

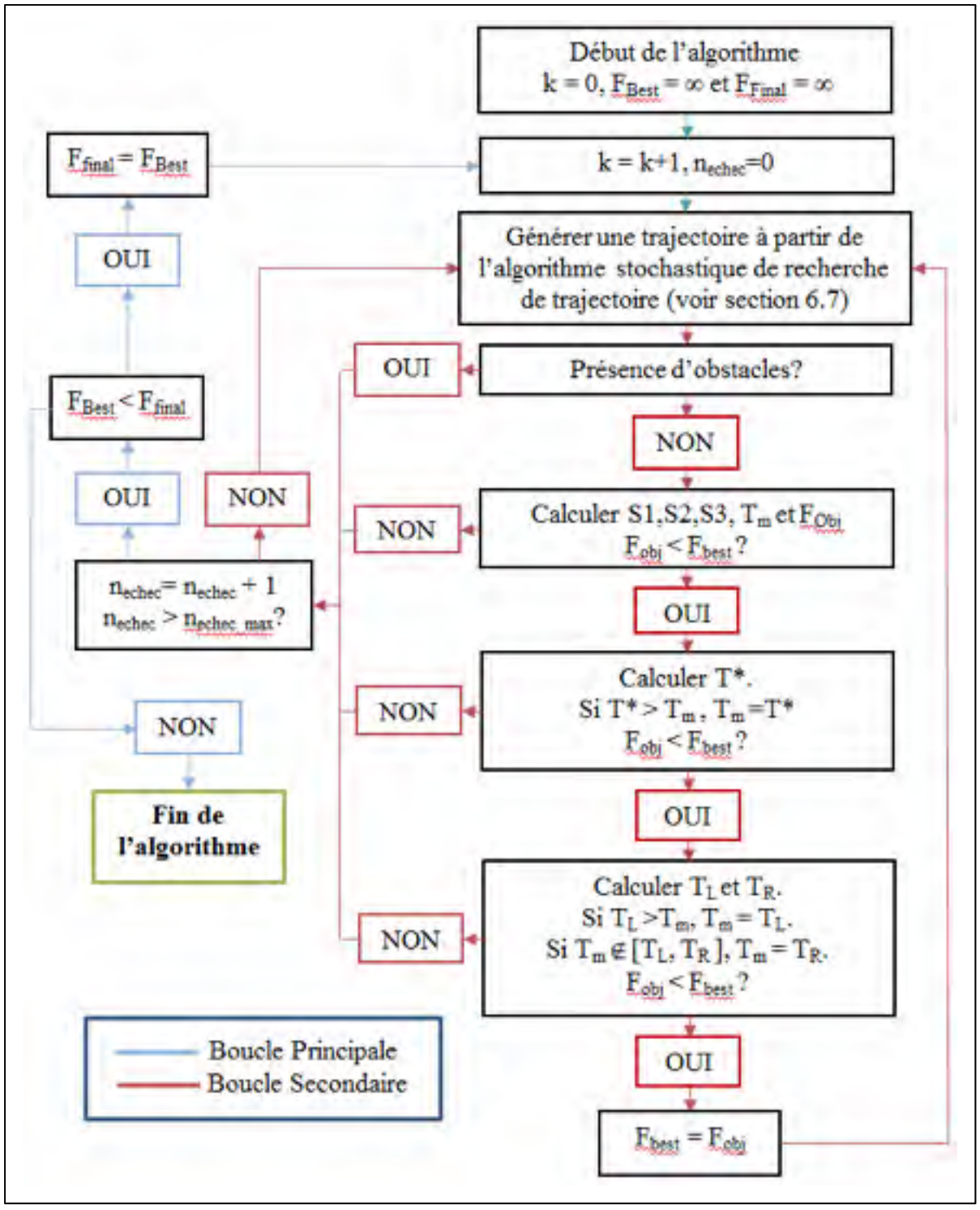


Figure 47 Algorithme des splines cubiques aléatoires

6.6.2 Splines cubiques aléatoires (chemin fixe)

L'algorithme des splines cubiques aléatoires peut aussi être utilisé pour calculer des trajectoires avec un chemin fixe. Les calculs sont les mêmes que la section précédente et le principe de l'algorithme est le même que la figure 47. Le chemin ne sera pas optimisé et l'algorithme utilise le chemin calculé préalablement à l'aide de l'algorithme A*. La différence est au niveau de l'algorithme stochastique de recherche de trajectoires (ASRT) de la section 6.7. Les points générés aléatoirement par cet algorithme sont des points qui se retrouvent sur le chemin au lieu d'être des points dans l'espace articulaire. Le fonctionnement demeure le même, c'est-à-dire que l'algorithme démarre avec trois points soit la position initiale, la position finale et un point placé aléatoirement sur le chemin. À la prochaine itération, un point est ajouté entre chaque point trouvé et le processus est recommencé.

6.6.3 Méthode traditionnelle (ligne droite dans l'espace articulaire)

Les méthodes traditionnelles de génération de trajectoire tracent une ligne droite dans l'espace articulaire. Il existe trois principaux modèles pour calculer les profils de position, vitesse et accélération : le modèle polynomial de cinquième ordre, le modèle de vitesse trapézoïdale et le modèle de vitesse trapézoïdale lissée. Le modèle de vitesse trapézoïdale sera présenté dans cette section; mais les calculs détaillés sont disponibles dans la référence suivante : (Dombre et Khalil, 2007). Ce modèle est caractérisé par une phase d'accélération, une phase de vitesse constante et une phase de décélération. Les caractéristiques principales exploitées lors de la génération de ces trajectoires sont : une accélération de type bang-bang; un temps de trajectoire minimisé; des vitesses et des accélérations qui respectent des contraintes.

La vitesse maximale permise est calculée à partir des spécifications du moteur et du ratio de l'engrenage. L'accélération maximale peut être approximée par un ratio entre le torque maximal du moteur et l'inertie maximale (approximée à partir de la matrice de masse (**M**) du

modèle dynamique). Cette approximation est cependant toujours une surestimation, ce qui rend la solution conservatrice.

Pour chacune des articulations, il faut être assuré que les limites d'accélération et de vitesse sont respectées. Le temps minimal de la trajectoire est calculé en considérant des temps d'accélération et de vitesse identiques pour tous les actionneurs. Ces deux constantes temporelles peuvent alors être facilement ajustées de façon à minimiser le temps de la trajectoire tout en respectant les contraintes de vitesse et d'accélération (Dombre et Khalil, 2007).

Les profils de vitesse et d'accélération sont calculés en dérivant les équations du profil de position. La figure 48 présente un exemple de profil obtenu avec cette méthode. Ce profil peut être amélioré en utilisant la méthode de vitesse trapézoïdale lissée (Dombre et Khalil, 2007). Il s'agit du même problème, mais l'accélération n'est pas constante.

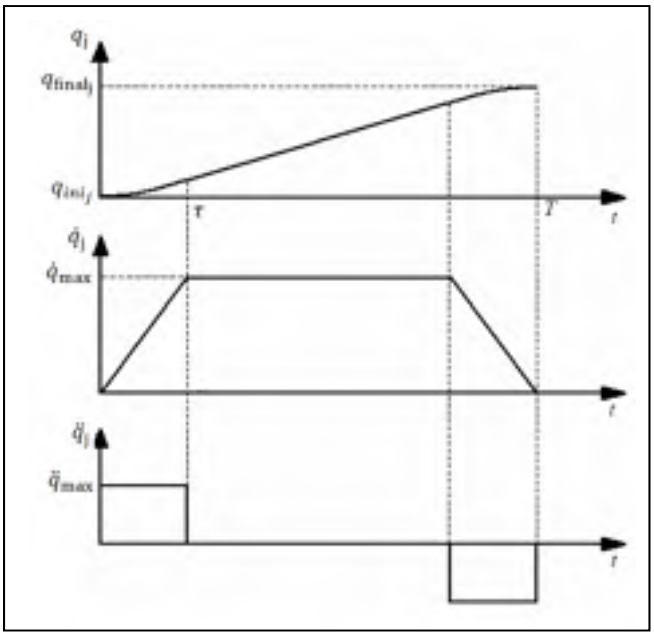


Figure 48 Profils de position, vitesse et accélération

6.6.4 Programmation non linéaire (chemin fixe)

La programmation non linéaire permet de calculer les profils de position, vitesse et d'accélération pour chaque articulation d'un chemin fixe (Lin, Chang et Luh, 1983). Le chemin est calculé préalablement à l'aide de l'algorithme A*. L'algorithme fait varier l'intervalle de temps *h* entre les points pour optimiser une fonction objective. Comme il a été mentionné plus tôt, la fonction objective choisie dans ce mémoire est le temps de la trajectoire.

À chaque itération, l'accélération entre les points est calculée. Pour ce faire, une trajectoire de type courbe spline de degré 3 (Lin, Chang et Luh, 1983) est générée. Les conditions aux limites de cette trajectoire sont une vitesse et une accélération initiale nulle et une continuité de la position, de la vitesse et de l'accélération entre chaque segment de la trajectoire. L'optimisation est soumise aux contraintes de vitesse, d'accélération et de saccade qui ne doivent pas dépasser les valeurs maximales prescrites.

À partir des positions initiales et des contraintes, les intervalles *h* initiaux sont calculés. À chaque itération de l'algorithme, la fonction objective est calculée. Les contraintes sont aussi évaluées pour s'assurer qu'elles ne soient pas violées. Le processus itératif se poursuit avec de nouveaux intervalles *h* déterminés par une procédure d'optimisation non linéaire jusqu'à ce que la fonction objective ait atteint son minimum. Ensuite, les temps *h* entre les points servent à calculer les profils de position, de vitesse et d'accélération. Pour plus de détails sur cette approche, consulter la référence suivante : (Lin, Chang et Luh, 1983).

L'algorithme est exécuté à l'aide de la fonction *fmincon* de Matlab. Cette fonction sert à trouver un minimum d'une fonction objective soumise à plusieurs contraintes. La fonction détermine les nouveaux intervalles de temps et détermine quand le minimum de la fonction objective est atteint.

6.7 Algorithme stochastique de recherche de trajectoires

L'algorithme stochastique de recherche de trajectoires (ASRT) permet de trouver une trajectoire à l'aide d'un processus récursif. Ce processus est appelé plusieurs fois par l'algorithme SCA illustré par la figure 47. La première fois que l'ASRT est appelé, la trajectoire à générer est constituée des positions initiale et finale imposées et d'une position intermédiaire choisie par l'algorithme. Cette trajectoire est désignée par l'indice k=1 pour indiquer qu'il s'agit du premier niveau de la récursion. Chaque niveau de récursion peut être appelé plusieurs fois. Cet algorithme se charge de donner des points, c'est-à-dire des positions articulaires qui pourraient former la trajectoire, mais il ne considère pas les obstacles. Ceux-ci sont traités dans l'algorithme SCA.

Au premier niveau de récursion, un point est choisi de façon aléatoire (selon une distribution uniforme) dans une région de recherche définie autour du point initial et final. Un facteur d'expansion permet d'obtenir une région semblable à celle qui est illustrée sur la figure 49.

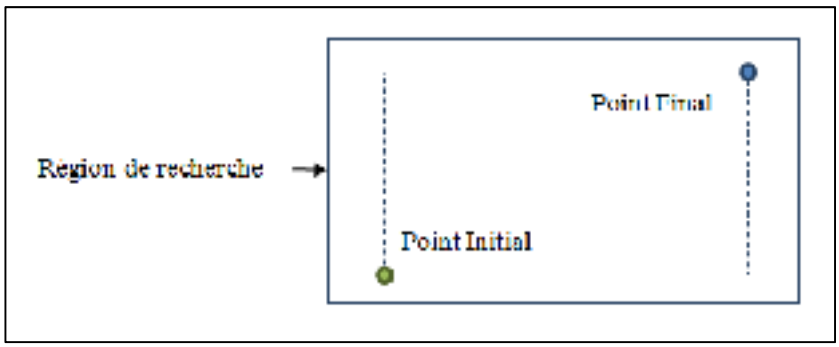


Figure 49 Région de recherche initiale

Pour les autres niveaux de récursion (k = 2, 3, 4, …), la trajectoire de l'étape (k-1) est utilisée comme point de départ. À cette trajectoire initiale sont ajoutés de nouveaux points. Chaque nouveau point s'insère entre deux points de la trajectoire initiale. Tous les points de la trajectoire sont alors régénérés de façon aléatoire à partir d'une région différente selon qu'il s'agisse de nouveaux points ou de points de la trajectoire initiale. Les points de la trajectoire

initiale sont utilisés comme point de base puisqu'ils ont été optimisés et donnent une forme à la trajectoire. Ils peuvent cependant être modifiés au cours du présent niveau de récursion pour s'adapter aux nouveaux points calculés avant et après.

La région de recherche est déterminée selon deux conditions:

- S'il s'agit d'un nouveau point, la région de recherche est déterminée de la même façon que pour la première étape de récursion. Le point avant et après celui recherché sont alors considérés pour définir les limites de la région.

- S'il s'agit d'un point de la trajectoire initiale, la région est calculée comme si c'était un nouveau point, mais la région est contractée d'un facteur déterminé tel qu'illustré à la figure 50.

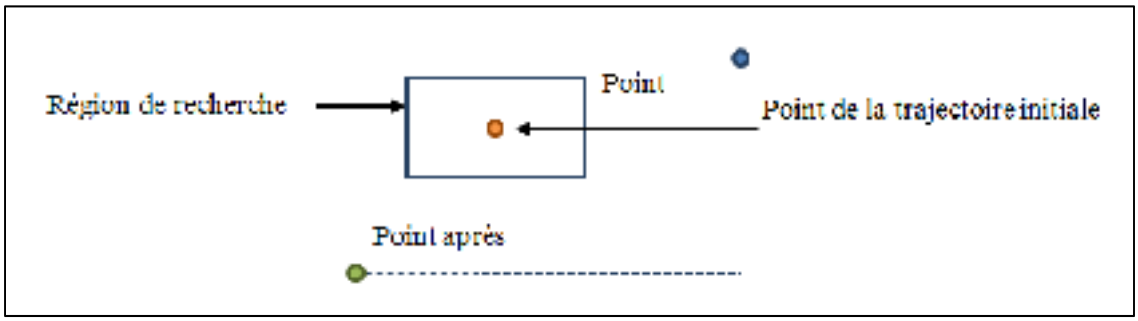


Figure 50 Région de recherche de la seconde itération

6.8 Conclusion

Ce chapitre a démontré comment calculer des trajectoires optimales dans l'espace articulaire pour le robot DexTAR. La planification de trajectoire considère les changements de mode de fonctionnement et les obstacles dus aux interférences mécaniques. Deux autres méthodes ont été présentées afin de pouvoir les comparer à la méthode retenue, soit celle des splines cubiques aléatoires. Les calculs et opérations nécessaires à l'utilisation de l'algorithme des

splines cubiques aléatoires comme la génération de l'environnement sous forme de graphe de connectivité et la recherche du chemin le plus cours ont aussi été détaillés tout comme la méthode pour former les splines et l'algorithme stochastique pour la génération de trajectoires aléatoires. Les résultats sont présentés dans le prochain chapitre.

CHAPITRE 7

EXPÉRIMENTATIONS ET RÉSULTATS

7.1 Introduction

Ce chapitre présente les résultats obtenus des différentes méthodes de planification étudiées dans ce mémoire. D'une part, des trajectoires ont été générées en utilisant les méthodes expliquées au chapitre 6. Ces trajectoires ont été choisies pour faciliter la comparaison et pour montrer les limites de celle-ci. Au total, cinq trajectoires ont été sélectionnées. Les critères justifiant leur sélection sont énoncés dans les sous-sections consacrées à la définition de ces trajectoires. La seconde section porte sur les résultats obtenus sur le prototype. L'espace réel du robot après expérimentations est dévoilé et il est comparé avec le robot RP5-AH de Mitsubishi (figure 1). Ensuite, les performances du contrôleur avec l'intégration du modèle dynamique sont détaillées.

7.2 Présentation des résultats (planification de trajectoire)

7.2.1 Contraintes

Les trajectoires ont été générées en considérant les performances maximales du robot. Tout d'abord, la vitesse maximale du robot a été calculée à partir des spécifications des moteurs. Comme il n'y a pas d'engrenage, la vitesse maximale spécifiée par le fabricant des moteurs a directement été utilisée. Le couple maximal a également été fixé à partir des spécifications du fabricant des moteurs. Par contre, l'accélération maximale ne peut pas être déduite seulement à partir des paramètres du fabricant, car elle dépend de la méthode de planification utilisée. Par exemple, lorsque la méthode considère le modèle dynamique, l'accélération maximale a été fixée arbitrairement grande puisque le couple aux moteurs est alors contraint et que l'accélération le devient alors forcément par l'entremise de la dynamique. Pour les autres méthodes, l'accélération maximale a été approximée par un ratio entre le couple maximal du

moteur et l'inertie maximale (calculée à partir de la matrice **M** du modèle dynamique) (Dombre et Khalil, 2007). La saccade maximale a été fixée suffisamment grande pour ne pas trop contraindre la trajectoire puisqu'il n'est pas utilisé dans toutes les méthodes. Le tableau 7 résume les contraintes utilisées. La figure 51 se veut un rappel des couleurs utilisées pour les représentations de l'espace articulaire.

Tableau 7 Contraintes de la planification de trajectoire

Contrainte	Valeur
Vitesse maximale	52 rad/s (500 RPM)
Accélération maximale (sans modèle dynamique)	220 rad/s^2
Accélération maximale (avec modèle dynamique)	500 rad/s^2
Saccade maximale	60 000 rad/s^3
Couple maximal	20 Nm

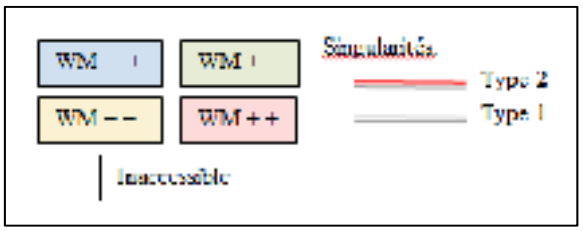


Figure 51 Rappel de la légende des couleurs utilisées dans l'espace articulaire

7.2.2 Trajectoire de test #1

La trajectoire présentée dans cette section a comme positions cartésiennes initiales et finales les valeurs données dans le tableau 8. Les figures 52 et 53 comparent la trajectoire obtenue par la méthode des splines cubiques aléatoires (SCA) avec celles obtenues par les 3 autres méthodes présentées au chapitre 6. Le tableau 9 présente les couleurs des courbes de chacune des méthodes. Le tableau 10 présente également un comparatif de la performance temporelle

des trajectoires obtenues. Les zones noires dans l'espace cartésien sont des zones inatteignables. Ces zones seront modifiées dans la prochaine section des résultats suite aux expérimentations effectuées sur le prototype. Cette trajectoire a été sélectionnée puisqu'elle démontre bien que le temps de parcours est réduit lorsque le chemin planifié est plus lisse dans l'espace cartésien. En effet, en évitant des déviations dans le déplacement cartésien, le temps de parcours est inférieur. Ces résultats montrent l'atout d'utiliser la méthode des SCA couplée à un algorithme stochastique pour rechercher des trajectoires possibles. Cette trajectoire est toutefois réalisable par toutes les méthodes. La moins efficace est évidemment la ligne droite suivie de la programmation non linéaire (PNL) d'un chemin fixe. Cela est dû au fait que ces méthodes n'intègrent pas le modèle dynamique. La méthode des SCA avec un chemin fixe améliore considérablement le temps de parcours, soit une augmentation de 24.83% comparé à la PNL. Selon le tableau 10, le gain entre la méthode des SCA avec un chemin libre ou fixe est seulement de 2.29%, mais démontre malgré tout la valeur ajoutée d'utiliser un chemin qui n'est pas nécessairement le plus court ou une ligne droite.

Tableau 8 Position de la trajectoire 1

Position initiale		**Position finale**	
x	−20	**x**	280
y	40	**y**	−70

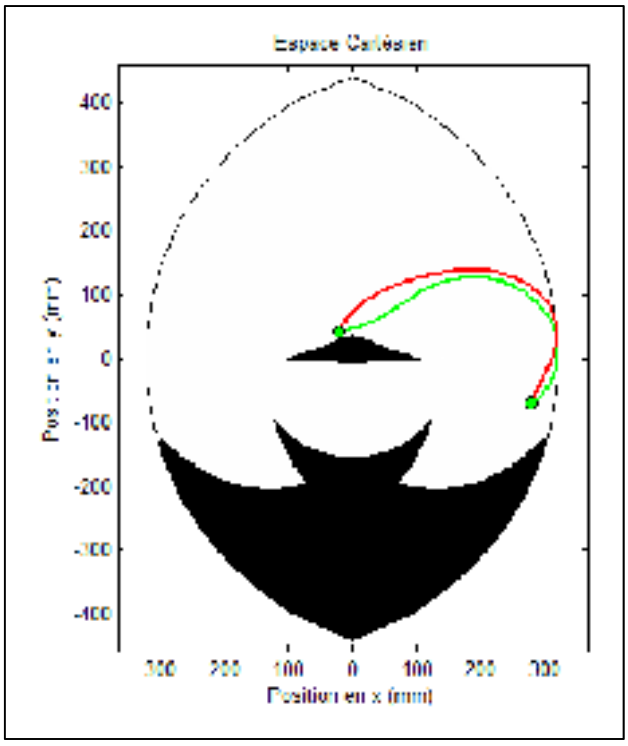


Figure 52 Espace cartésien de la trajectoire 1

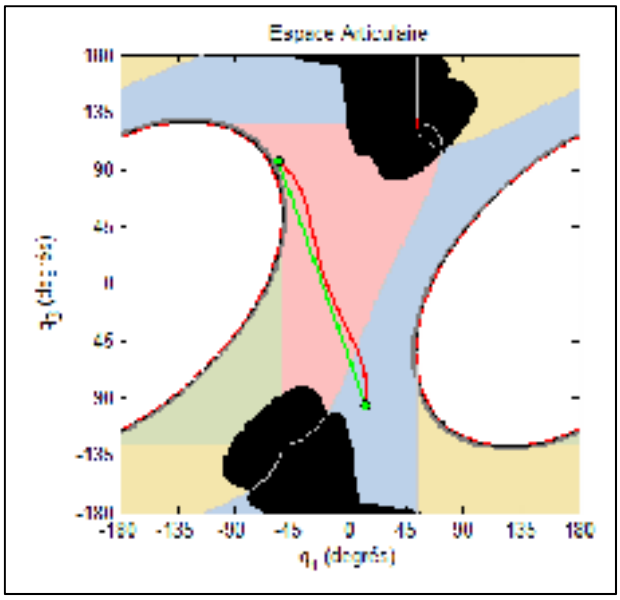


Figure 53 Espace articulaire de la trajectoire 1

Tableau 9 Légende des courbes

	SCA
	SCA (chemin fixe), PNL, Ligne Droite

Tableau 10 Résultats de la trajectoire 1

Méthode	Temps (s)	Gain 1[3]	Gain 2[4]	Gain 3[5]
Splines cubiques aléatoires	0.213	29.47%	26.55%	2.29%
Splines cubiques aléatoires (chemin fixe)	0.218	27.81%	24.83%	
Programmation non-linéaire	0.290	3.97%		
Ligne droite dans l'espace articulaire	0.302			

7.2.3 Trajectoire de test #2

La seconde trajectoire a comme positions cartésiennes initiales et finales les valeurs données dans le tableau 11. Les figures 54 et 55 comparent la trajectoire obtenue par la méthode des SCA avec celles obtenues par les 3 autres méthodes présentées au chapitre 6. Le tableau 12 présente les couleurs des courbes de chacune des méthodes. Le tableau 27 présente également un comparatif de la performance temporelle des trajectoires obtenues. La trajectoire présentée dans cette section est plus classique : elle est souvent utilisée à des fins de comparaisons (Nabat et al., 2008). Il s'agit d'une trajectoire de 300 mm (environ 12"). Le changement de mode de fonctionnement n'est pas nécessaire, mais il est utilisé pour atteindre la position finale en traçant une ligne droite dans l'espace articulaire. Le temps optimal est une fois de plus atteint avec la méthode des SCA. Le chemin aléatoire choisi suit davantage

[3] Gain 1 = Gain de temps par rapport à une trajectoire en ligne droite
[4] Gain 2 = Gain de temps par rapport à la programmation non-linéaire
[5] Gain 3 = Gain de temps par rapport à la spline cubique aléatoire (chemin fixe)

la singularité de type 2 de sorte qu'il y a un gain important comparé à la méthode des SCA avec un chemin fixe (24.24%). Le chemin le plus court calculé s'éloigne de la singularité de type 2, car l'algorithme A* utilisé avec le graphe de connectivité a calculé un chemin plus court en empruntant les plus gros sous-pavés qui suivent la zone inaccessible au lieu de suivre de près cette zone. Il s'agit d'une limitation au graphe de connectivité lorsque le chemin est très près des obstacles. De plus, en comparant les deux méthodes qui n'utilisent pas le modèle dynamique, soit la PNL et la ligne droite dans l'espace articulaire, le gain en temps d'utiliser le chemin le plus court par rapport à la ligne droite est considérable, soit de 12.59%.

Tableau 11 Positions de la trajectoire 2

Position initiale		**Position finale**	
x	−150	**x**	150
y	225	**y**	225

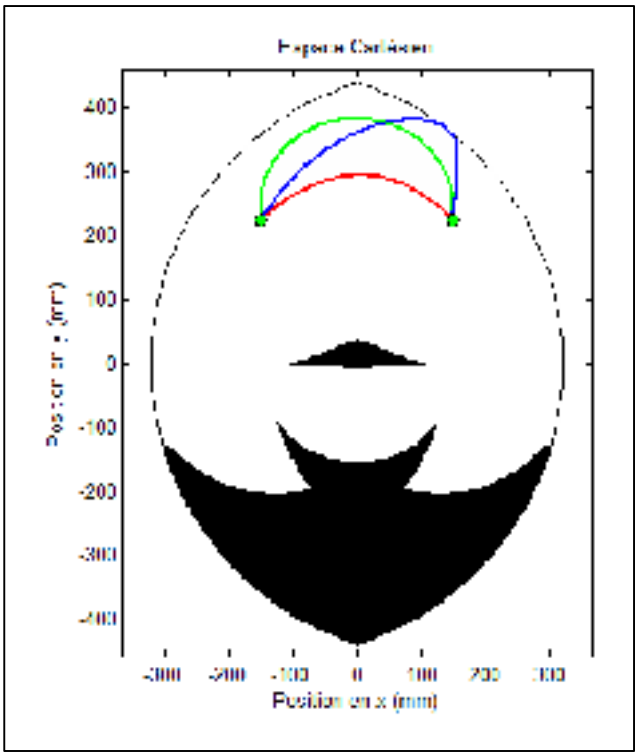


Figure 54 Espace cartésien de la trajectoire 2

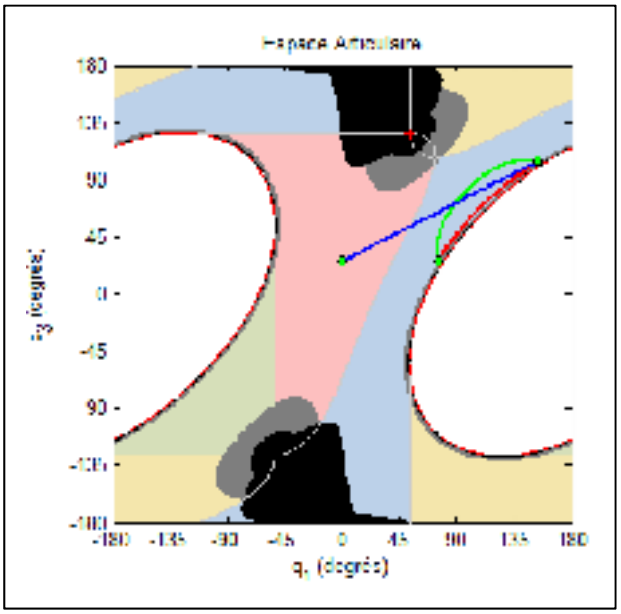


Figure 55 Espace articulaire de la trajectoire 2

Tableau 12 Légende des courbes de la trajectoire 2

	SCA
	SCA (chemin fixe), PNL
	Ligne Droite

Tableau 13 Résultats de la trajectoire 2

Méthode	Temps (s)	Gain 1[6]	Gain 2[7]	Gain 3[8]
Splines cubiques aléatoires	0.176	34.81%	25.42%	24.14%
Splines cubiques aléatoires (chemin fixe)	0.232	14.07%	1.69%	
Programmation non-linéaire	0.236	12.59%		
Ligne droite dans l'espace articulaire	0.270			

7.2.4 Trajectoire de test #3

La troisième trajectoire a comme positions cartésiennes initiales et finales les valeurs données dans le tableau14. Les figures 56 et 57 comparent la trajectoire obtenue par la méthode des SCA avec celles obtenues par les 3 autres méthodes présentées au chapitre 6. Le tableau 15 présente les couleurs des courbes de chacune des méthodes. Le tableau 16 présente également un comparatif de la performance temporelle des trajectoires obtenues.

Cette trajectoire est intéressante puisqu'elle montre plusieurs avantages à utiliser les méthodes de planification de trajectoire étudiées dans ce mémoire. Tout d'abord, la

[6] Gain 1 = Gain de temps par rapport à une trajectoire en ligne droite
[7] Gain 2 = Gain de temps par rapport à la Programmation non linéaire
[8] Gain 3 = Gain de temps par rapport à la Spline cubique aléatoire (Chemin fixe)

trajectoire ne peut être réalisée en traçant une ligne droite dans l'espace articulaire. Ainsi, l'intérêt d'utiliser un algorithme de recherche du chemin le plus court prend tout son sens. De plus, le fait de changer de mode de fonctionnement peut optimiser beaucoup le temps de déplacement. Cette trajectoire aurait pu être réalisée sans changer de mode de fonctionnement, mais le gain en temps de 27.09% justifie le changement de mode de fonctionnement, autant dans le but d'accroître l'espace de travail que dans celui d'optimiser les déplacements. Finalement, cette trajectoire montre aussi les limites des SCA avec un chemin libre. Le processus de recherche de la trajectoire n'a pu trouver celle qui longeait les interférences mécaniques à cause des changements brusques dans celle-ci. Il faut noter que la méthode des SCA génère la plupart du temps des trajectoires avec 1, 3 ou 5 points libres. À chaque itération, le nombre de points libre est augmenté. Ainsi, dans ce cas, l'algorithme a trouvé un chemin à 1 point et il n'a pas réussi à trouver un meilleur chemin avec plusieurs points. Cette limite pourrait être partiellement contournée en améliorant l'algorithme de génération de trajectoires aléatoires (l'algorithme stochastique). Cependant, face à de telles situations, il est plutôt recommandé d'utiliser l'algorithme des SCA avec un chemin fixe.

Tableau 14 Positions de la trajectoire 3

Position initiale		**Position finale**	
x	−308	**x**	308
y	−55	**y**	−55

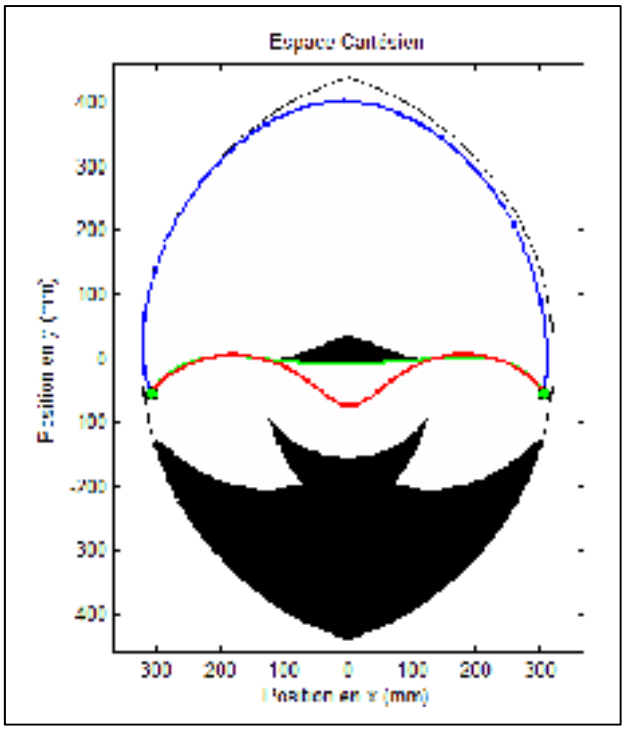


Figure 56 Espace cartésien de la trajectoire 3

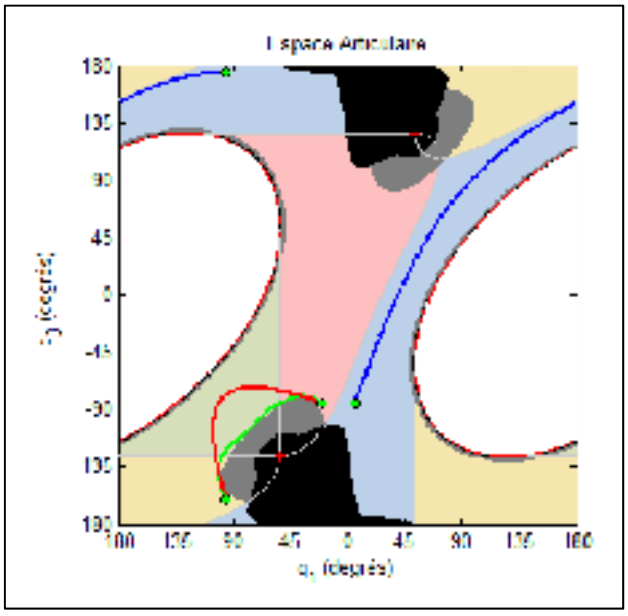


Figure 57 Espace articulaire de la trajectoire 3

Tableau 15 Légende des courbes de la trajectoire 3

	SCA
	SCA (chemin fixe), PNL
	SCA (WM − +)

Tableau 16 Résultats de la trajectoire 3

Méthode	Temps (s)	Gain 2[9]	Gain 3[10]
Splines cubiques aléatoires	0.270	10.89%	
Splines cubiques aléatoires (chemin fixe)	0.218	28.05%	27.09%
Programmation non-linéaire	0.303		
Splines cubiques aléatoires (wm − +)	0.299		

7.2.5 Trajectoires avec obstacles

La quatrième trajectoire a comme positions cartésiennes initiales et finales les valeurs données dans le tableau 17. Elle comporte aussi des obstacles dans l'espace cartésien. Les figures 58 et 59 comparent la trajectoire obtenue par la méthode des SCA avec celles obtenues par la méthode NL. Le tableau 18 présente les couleurs des courbes de chacune des méthodes. Le tableau 19 présente également un comparatif de la performance temporelle des trajectoires obtenues.

Cette trajectoire a été choisie pour montrer que les méthodes étudiées peuvent aussi servir à planifier des trajectoires avec des obstacles dans l'espace cartésien. Ces obstacles sont

[9] Gain 2 = Gain de temps par rapport à la Programmation Non-Linéaire
[10] Gain 3 = Gain de temps par rapport à la Spline cubique aléatoire (wm − +)

d'abord transformés dans l'espace articulaire. La carte de l'environnement encombré est ensuite fusionnée avec la carte déjà calculée et le graphe de connectivité est recalculé en considérant les nouveaux obstacles. Les méthodes de planification de trajectoire avec un chemin fixe sont utilisées. Les résultats montrent que le temps de parcours est amélioré de 9.49% en utilisant les SCA avec un chemin fixe versus la PNL. Le chemin pour les deux méthodes est le même, mais l'évolution temporelle est différente. La méthode des lignes droites dans ce cas-ci n'est pas possible puisqu'il n'y a pas de chemin en ligne droite réalisable dans l'espace articulaire. Les SCA avec un chemin libre n'est pas conseillé dans ce type de trajectoire puisque le chemin est difficile à trouver. Le temps de calcul est beaucoup plus long et l'algorithme peut trouver des points qui ne sont optimaux, mais qui sont conservés, car la recherche des points n'a pas mené à de meilleurs points avant d'atteindre le nombre d'échecs maximal. Donc, plus il y a d'obstacles dans une trajectoire et que le chemin doit comporter plusieurs points pour les éviter, plus l'algorithme des SCA avec un chemin libre s'avère inefficace. L'utilisation d'une méthode avec chemin fixe est alors fortement conseillée.

Tableau 17 Positions de la trajectoire 4

Position initiale		**Position finale**	
x	−180	**x**	180
y	300	**y**	300

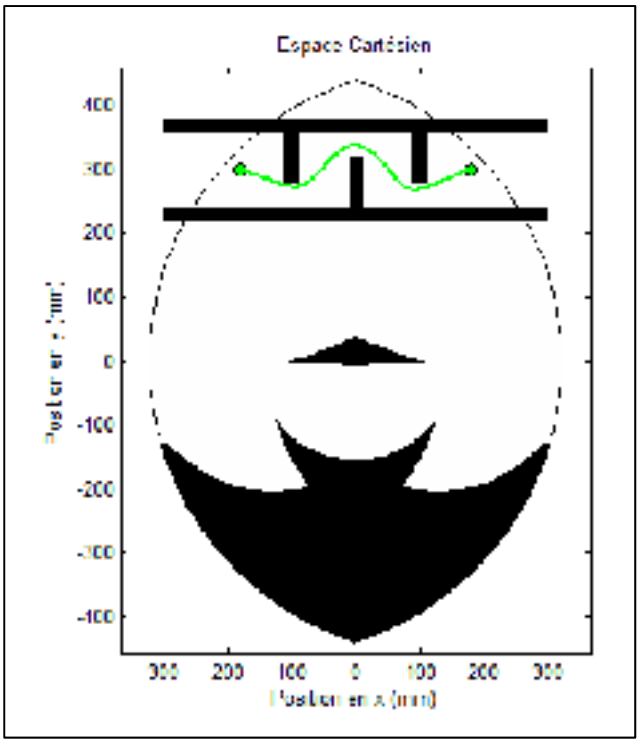


Figure 58 Espace cartésien de la trajectoire 4

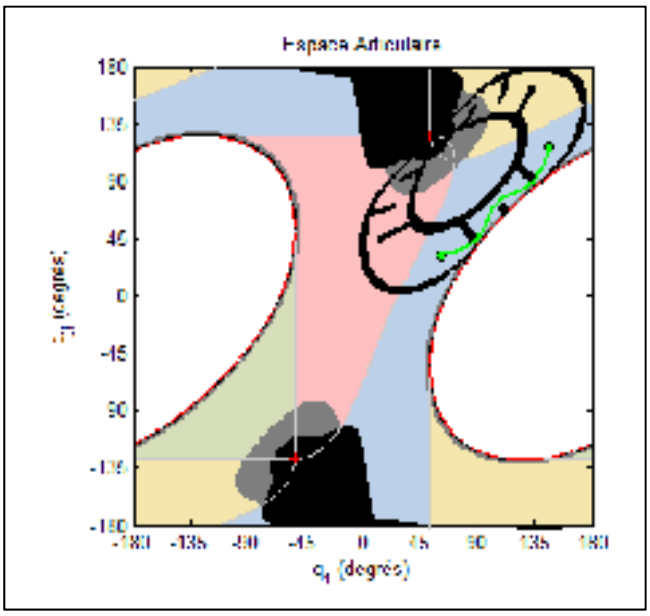


Figure 59 Espace articulaire de la trajectoire 4

Tableau 18 Légende des courbes de la trajectoire 4

	SCA (chemin fixe), PNL

Tableau 19 Résultats d'une trajectoire avec obstacles dans l'espace cartésien

Méthode	Temps (s)	Gain 2[11]
Splines cubiques aléatoires (chemin fixe)	0.286	9.49%
Programmation non-linéaire	0.316	

7.2.6 Trajectoires correspondant à des formes géométriques

La dernière trajectoire, illustrée par les figures 60 et 61, montre comment la PNL peut être utilisée pour tracer des trajectoires dans l'espace cartésien. Le tableau 20 présente les couleurs des courbes de chacune des méthodes. Dans les autres cas présentés, il s'agissait seulement de partir d'une position cartésienne initiale pour se rendre à une position finale. Cette extension aux méthodes de planification point-a-point est intéressante, car le robot peut être utilisé pour d'autres fonctions. La PNL a été légèrement modifiée pour que le temps entre les points de la trajectoire cartésienne soit le même. Le temps optimal calculé est de 0.911 s. Afin que la trajectoire cartésienne soit continue, le changement de mode de fonctionnement est cependant impossible. Malgré cette restriction, le robot peut exécuter un cercle de 150 mm comme montré à la figure 61. Comme le chemin est fixe, la méthode des lignes droites ne peut être utilisée sauf s'il y a une pause entre chaque point du cercle. Les SCA avec un chemin libre ne sont d'aucun intérêt puisque le chemin désiré est un chemin fixe tandis que les SCA avec chemin fixe ne donnent pas un résultat cohérent, car l'algorithme détermine le nombre de points sur le chemin pour obtenir un temps optimal et que le chemin résultant ne suit pas du tout le cercle.

[11] Gain 2 = Gain de temps par rapport à la Programmation Non-Linéaire

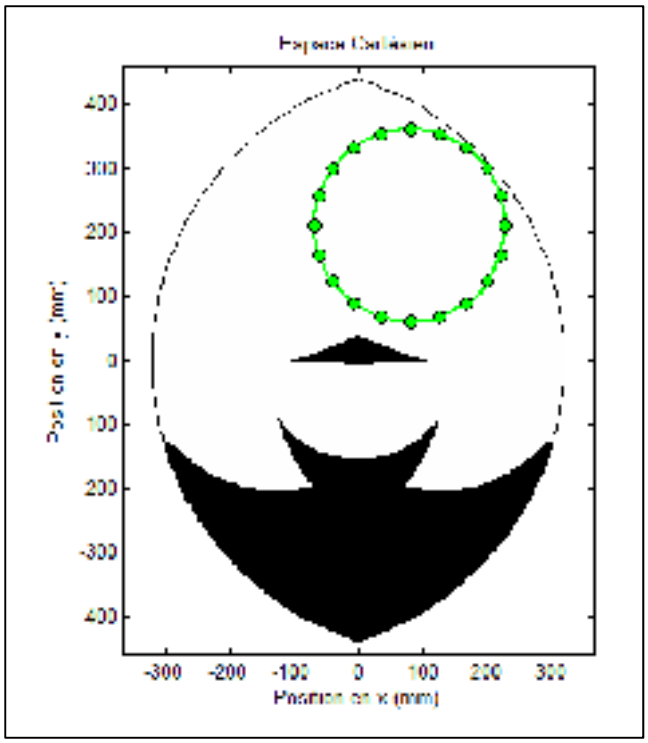


Figure 60 Espace cartésien de la trajectoire 5

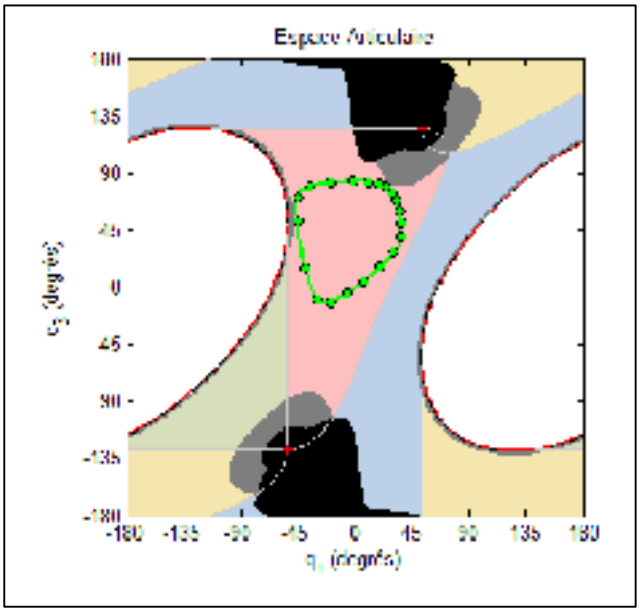


Figure 61 Espace articulaire de la trajectoire 5

Tableau 20 Légende des courbes de la trajectoire 5

	PNL

7.2.1 Retour sur toutes les trajectoires

Les trajectoires présentées dans cette section montrent les avantages et les inconvénients de chacune des méthodes de planification de trajectoires. Dans la plupart des cas, la méthode SCA avec un chemin libre trouve le chemin optimal. Cependant, dans certains cas, comme lorsque la trajectoire comporte un chemin avec des changements brusques, la méthode des SCA avec un chemin fixe s'avère plus efficace. La PNL fournit généralement des résultats inférieurs puisque cette méthode n'intègre pas le modèle dynamique, mais elle demeure utile pour tracer des formes géométriques dans l'espace cartésien. La méthode des lignes droites dans l'espace articulaire n'a aucun avantage outre sa simplicité de mise en œuvre et sa rapidité de calcul. Les résultats obtenus par cette méthode, lorsqu'il y en a, sont inférieurs aux autres méthodes plus avancées proposées dans ce mémoire. D'autre part, la trajectoire 3 démontre que le changement de mode de fonctionnement ne fait pas qu'augmenter l'espace de travail atteignable du robot, mais qu'il peut aussi servir à optimiser le temps de déplacement entre deux points. Finalement, la méthode des SCA avec un chemin libre pourrait être améliorée en modifiant l'algorithme de recherche des points. En utilisant une méthode plus avancée, l'algorithme pourrait converger plus rapidement vers des résultats optimaux.

7.3 Présentation des résultats (prototype)

7.3.1 Espace de travail réel du robot

L'espace de travail réel du robot est déterminé suite à des expérimentations. Pour ce faire, une trajectoire test a été choisie. La trajectoire est composée de points qui vont d'une limite à l'autre de l'espace de travail. Deux modes de fonctionnement sont testés : *i*) le mode $-+$

(bleu) et *ii*) le mode + + (rose). Les figures 63 et 65 montrent un exemple de trajectoire test pour chaque mode de fonctionnement en considérant les interférences mécaniques du tableau 21. La méthode utilisée est celle des splines cubiques aléatoires.

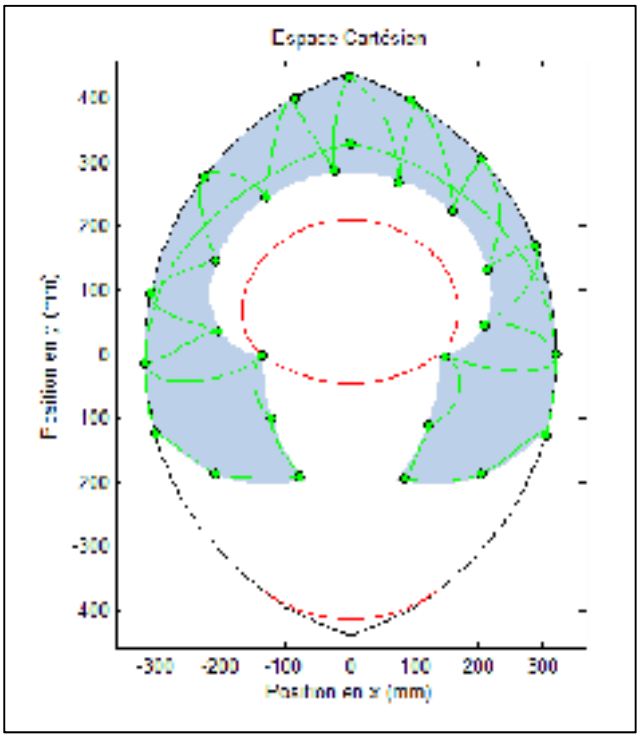


Figure 62 Trajectoire test dans l'espace cartésien (wm − +)

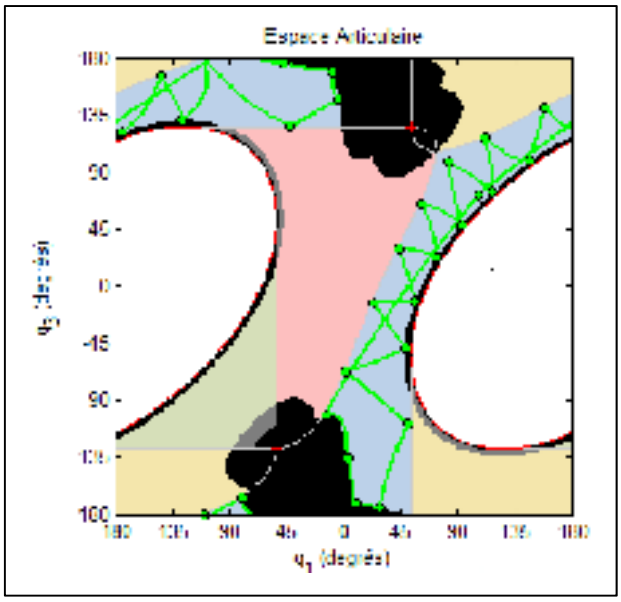


Figure 63 Trajectoire test dans l'espace articulaire (wm − +)

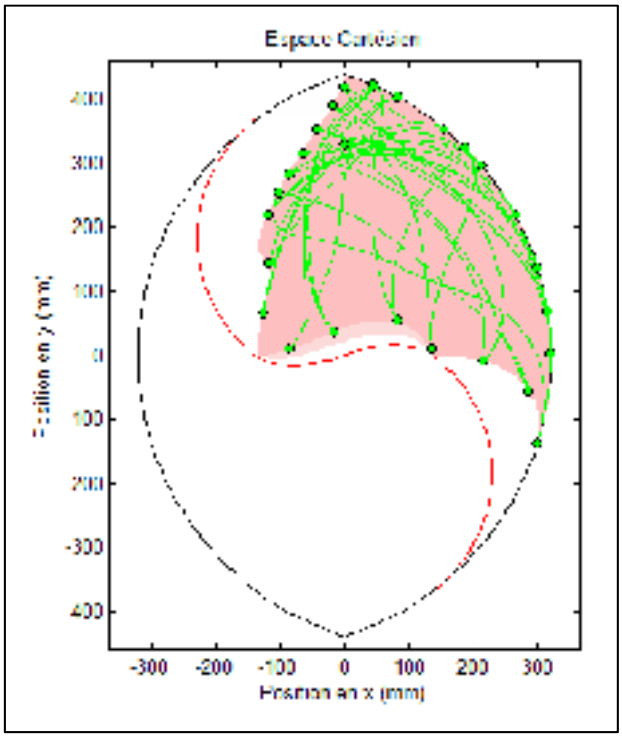


Figure 64 Trajectoire test dans l'espace cartésien (wm + +)

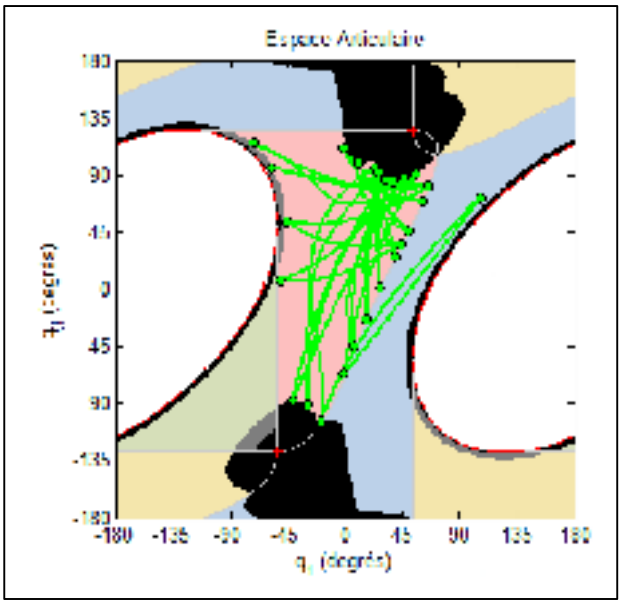


Figure 65 Trajectoire test dans l'espace articulaire (wm + +)

Tableau 21 Interférences mécaniques de trajectoires la trajectoire test présentée

Interférence mécanique	Valeur
Angle minimal entre les liens distaux	35°
Angle maximal entre les liens distaux	145°
Distance minimale entre les liens proximaux	40 mm

Une fois la trajectoire de test choisie, la technique pour déterminer l'espace de travail réel consiste à exécuter cette trajectoire en augmentant graduellement le couple jusqu'à ce qu'un comportement instable soit décelé ou lorsque la contrainte de couple maximale est atteinte (20 Nm). Lorsqu'une des deux conditions est atteinte, l'ensemble des interférences mécaniques est réduit pour agrandir l'espace de travail. Le tableau 22 montre l'évolution des interférences mécaniques dans les différents tests. Le cinquième test couvre tout l'espace de travail. Le tableau 23 montre les contraintes maximales utilisées.

Tableau 22 Interférences mécaniques utilisées dans les expérimentations

Interférence mécanique	Test 1	Test 2	Test 3	Test 4	Test 5
Angle minimal entre les liens distaux	35°	30°	25°	20°	18°
Angle maximal entre les liens distaux	145°	150°	155°	160°	162°
Distance minimale entre les liens proximaux	40 mm	40 mm	40 mm	40 mm	40 mm

Tableau 23 Contraintes réelles du robot

Contrainte	Valeur
Vitesse maximale	52 rad/s (500 RPM)
Accélération maximale (sans modèle dynamique)	240 rad/s^2
Accélération maximale (avec modèle dynamique)	1000 rad/s^2
Saccade maximale	60000 rad/s^3
Couple maximal	22 Nm

Les expérimentations permettre de conclure que l'ensemble des interférences mécaniques du « Test 1 » est les seuls où le robot a un comportement stable jusqu'au couple maximal. Les autres tests ne se sont pas avérés concluants puisqu'à l'approche des singularités de type 2, les moteurs vibrent et le couple maximal était supérieur à 1 Nm. Le tableau 24 expose les interférences mécaniques maximales. L'espace de travail total est représenté à la figure 66 (en blanc).

Tableau 24 Interférences mécaniques réelles

Interférence mécanique	Valeur
Angle minimal entre les liens distaux	35°
Angle maximal entre les liens distaux	145°
Distance minimale entre les liens proximaux	40 mm

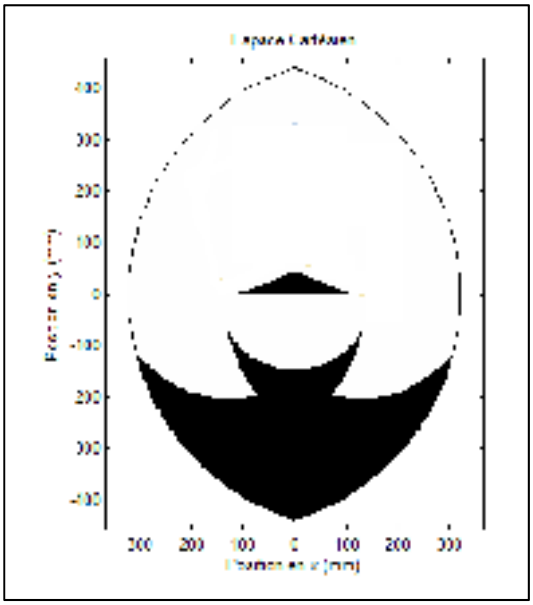


Figure 66 Espace de travail réel du DexTAR

Comme l'espace de travail réel possède un trou lorsque les contraintes maximales sont utilisées, un autre ensemble de contraintes maximales a été déterminé qui permet d'atteindre tout l'espace de travail. L'espace de travail est couvert au complet lorsque l'angle minimal entre les liens distaux est de 18° et l'angle maximal est de 162°. Tout comme pour trouver l'espace réel du robot, des trajectoires sont testées dans cette zone pour trouver le couple maximal pour lequel le robot devient instable. Ces contraintes sont présentées au tableau 25 et elles sont appelées « contraintes restreintes ».

Tableau 25 Contraintes restreintes du robot

Contrainte	Valeur
Vitesse maximale	52 rad/s (500 RPM)
Accélération maximale (sans modèle dynamique)	10 rad/s^2
Accélération maximale (avec modèle dynamique)	10 rad/s^2
Saccade maximale	60000 rad/s^3
Couple maximal	1 Nm

En utilisant ces contraintes, le robot peut se déplacer dans le trou et ainsi couvrir tout l'espace de travail. L'algorithme de planification de trajectoire utilise automatiquement les contraintes restreintes si la trajectoire a un point à l'intérieur de cette zone. S'il n'y a pas de points dans cette zone, l'algorithme de planification de trajectoire fait en sorte que le robot va l'éviter.

La figure 67 montre l'espace de travail total réel du robot pour chaque mode de fonctionnement. Les zones aux couleurs plus pâles correspondent aux zones qui doivent être atteintes avec les contraintes restreintes. Finalement, la figure 67 montre l'espace de travail total réel comprenant tous les modes de fonctionnement dans l'espace cartésien et l'espace articulaire. La couleur noire correspond à des zones inatteignables tandis que la couleur grise à des zones atteignables avec les contraintes restreintes.

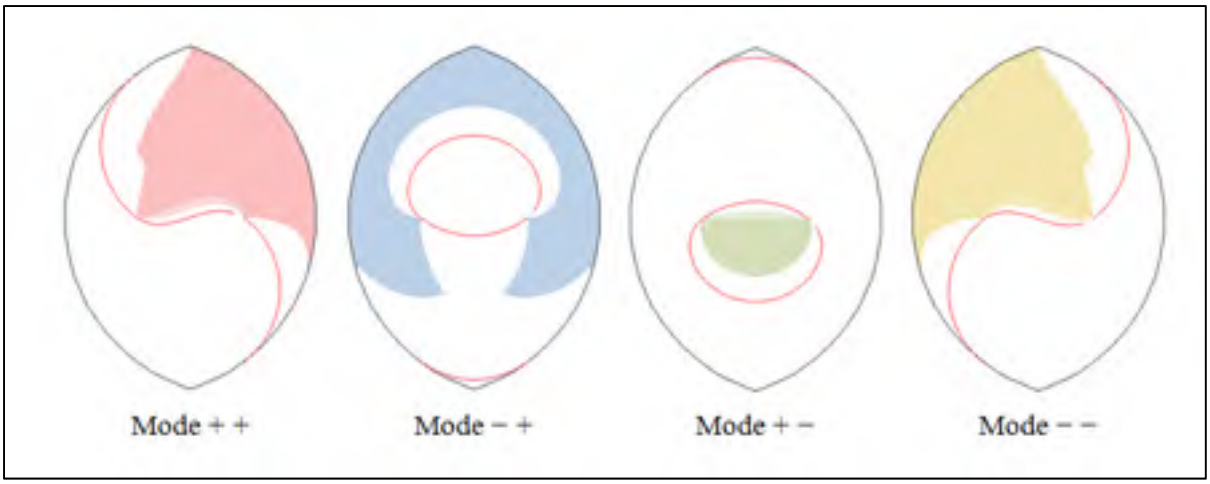


Figure 67 Espace de travail réel par mode de fonctionnement

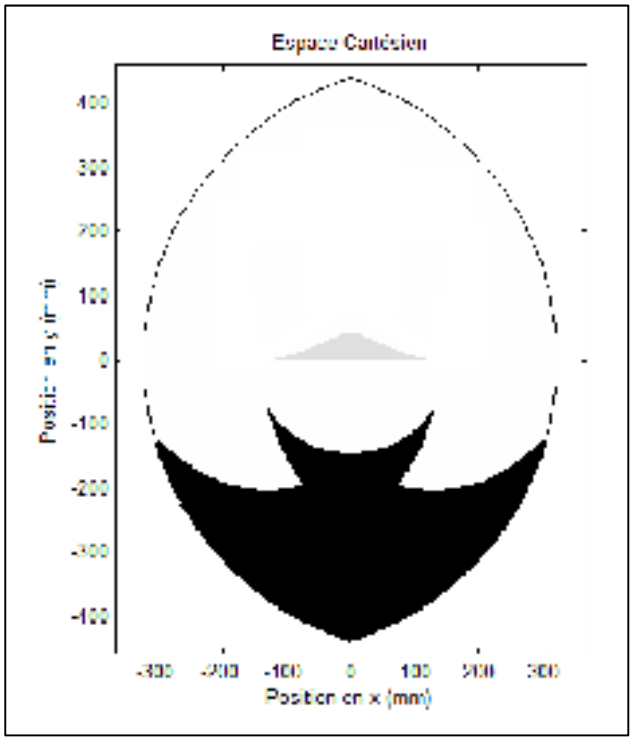


Figure 68 Espace de travail cartésien réel total du robot

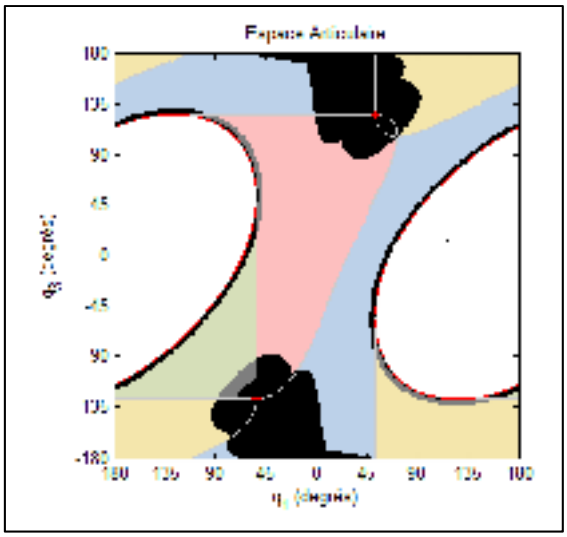


Figure 69 Espace de travail articulaire réel total du robot

7.3.2 Comparaison de l'espace de travail avec le robot Mitsubishi

L'objectif de ce mémoire est principalement de démontrer que l'espace de travail d'un robot parallèle peut être augmenté en changeant de mode de fonctionnement. Pour y arriver, un robot industriel est choisi pour comparer la grandeur de son espace de travail avec celle du DexTAR. Ce robot, le RP-5AH, est fabriqué par la compagnie Mitsubishi. Les liens proximaux ont une longueur de 200 mm tandis que les liens distaux ont une longueur de 260 mm. La distance entre les moteurs est de 170 mm. Les dimensions sont représentées à la figure 70. Le tableau1 de la page 34 présente pour sa part les dimensions du robot DexTAR : 230 mm pour les liens distaux et proximaux et 275 mm de distance entre les moteurs. Les bras proximaux du robot Mitsubishi sont légèrement plus courts, soit de 30 mm et les liens distaux sont plus long de 30 mm. La distance entre les moteurs est supérieure pour le robot DexTAR, mais l'espace de travail théorique (sans les singularités et les interférences mécaniques) est sensiblement la même pour les deux robots puisque la somme d'un membre composé d'un lien proximal et d'un lien distal est la même pour les deux robots. Ainsi, la limite extérieure des deux robots est pratiquement la même. Cependant, le robot de Mitsubishi ne peut pas changer de mode de fonctionnement donc son espace de travail est limité à un seul côté de sa base qui correspond au mode de fonctionnement − +.

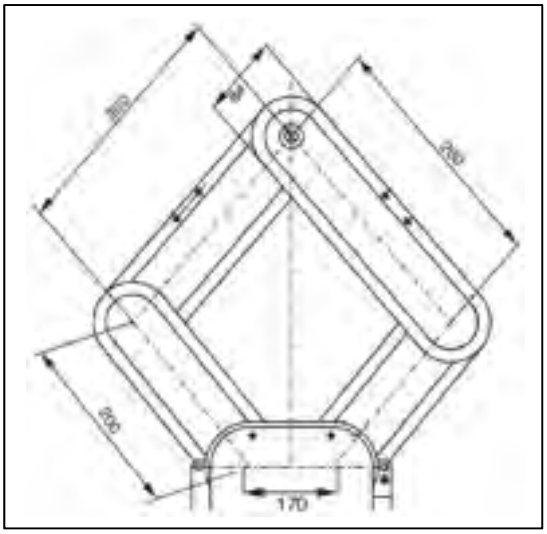


Figure 70 Dimensions du robot RP-5AH du Mitsubishi Electric
Tirée de Mitsubishi (2012, http://www.mitsubishi-automation.fr/)

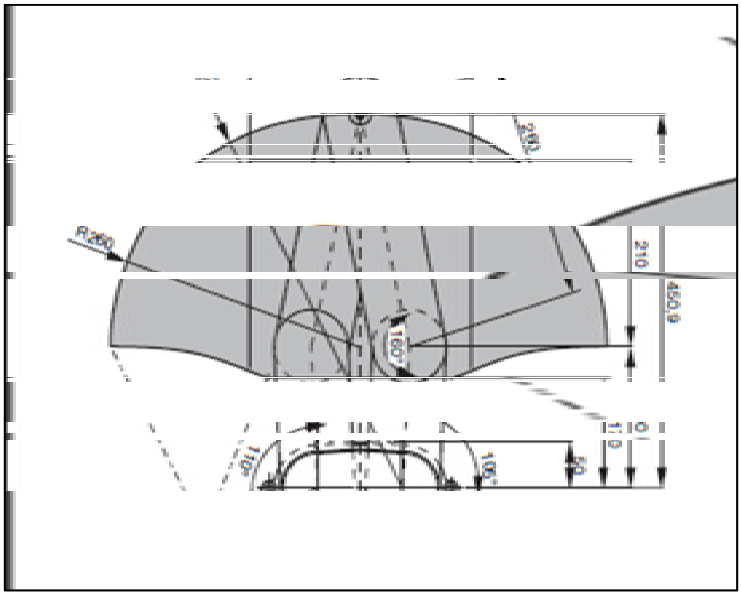


Figure 71 Espace de travail du robot RP-5AH
Tirée de Mitsubishi (2012, http://www.mitsubishi-automation.fr/)

La figure 71 présente l'espace de travail du RP-5AH du Mitsubishi Electric. Le fabricant indique dans les spécifications que le robot peut couvrir une zone rectangulaire de 207 mm × 210 mm soit 43 470 mm^2. En réalité, tout comme le robot DexTAR, l'espace de travail du robot n'est pas rectangulaire. Donc, afin de comparer l'espace totale du robot Mitsubishi avec celui du DexTAR, l'espace totale du robot Mitsubishi est calculée. La figure 72 présente l'espace totale du robot. Un rectangle de 380 mm × 640 mm est tracé autour de celui-ci. La surface couverte par le rectangle est de 243 200 mm^2. L'image est traitée dans Matlab pour connaître le ratio couvert par l'espace de travail. Pour se faire, chaque pixel est analysé pour savoir s'il est noir ou non. L'image comporte 39 485 pixels et l'espace de travail est composée de 25 857 pixels pour un ratio de couverture de 65.49%. Ainsi, l'espace de travail réel du robot RP-5AH est approximativement de 159 260 mm^2.

Dans le même ordre d'idée, les calculs sont exécutés pour le robot DexTAR. L'espace total couvert par le robot est représenté à la figure 73. La surface totale du rectangle est de 630 000 mm^2 et le ratio de couverture est de 46.37%. L'espace de travail total du robot est donc approximativement de 292 110 mm^2. L'espace de travail du robot DexTAR est considérablement augmenté en changeant de mode de fonctionnement. Ainsi, à titre de comparaison, l'espace de travail du robot DexTAR est 45.48% plus grand que l'espace de travail du robot Mitsubishi.

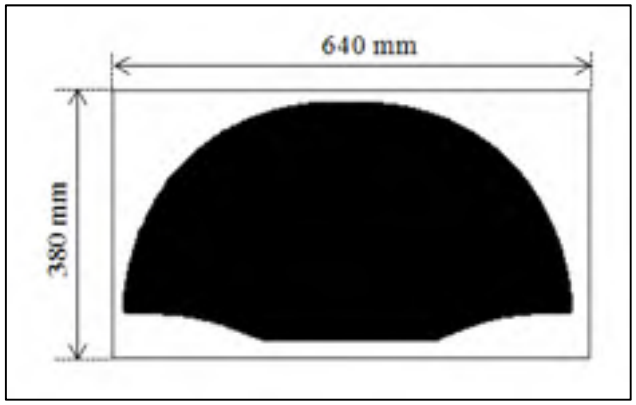


Figure 72 Espace totale du robot RP-5AH

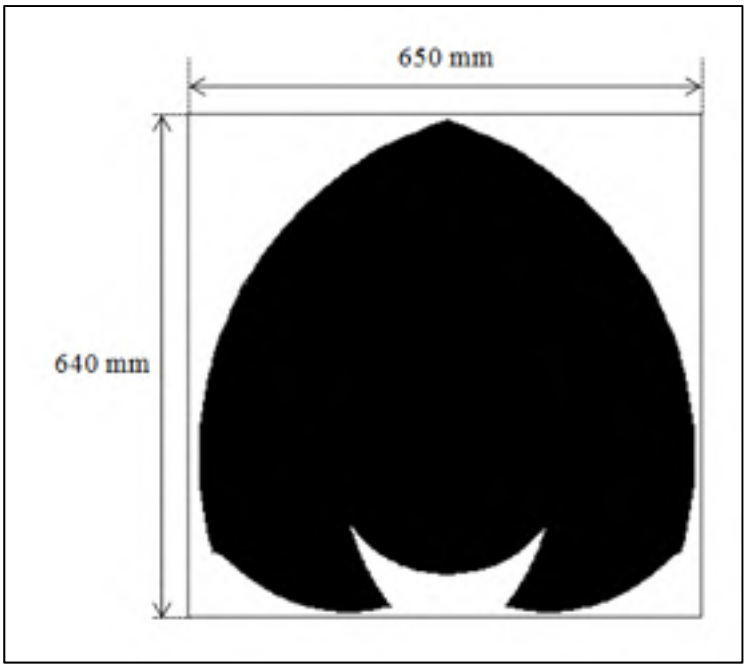


Figure 73 Espace total du robot DexTAR

7.3.3 Performances du contrôleur

Le modèle dynamique utilisé dans le contrôleur permet d'atteindre les critères de performances visés tout en assurant la précision du suivi. Cette section présente les résultats de suivi d'une trajectoire en utilisant le contrôleur décrit à section 5.4. La trajectoire sélectionnée est présentée à la figure 75 et les points cartésiens ainsi que les modes de fonctionnement sont indiqués dans le tableau 26. Les critères de sélection de cette trajectoire sont les suivants :

- la trajectoire couvre au maximum l'espace de travail;
- les déplacements sont les plus longs possible;
- la trajectoire passe par tous les modes de fonctionnement;
- la trajectoire a des points dans les zones restreintes;
- les contraintes maximales sont utilisées (voir tableau 23 et 25).

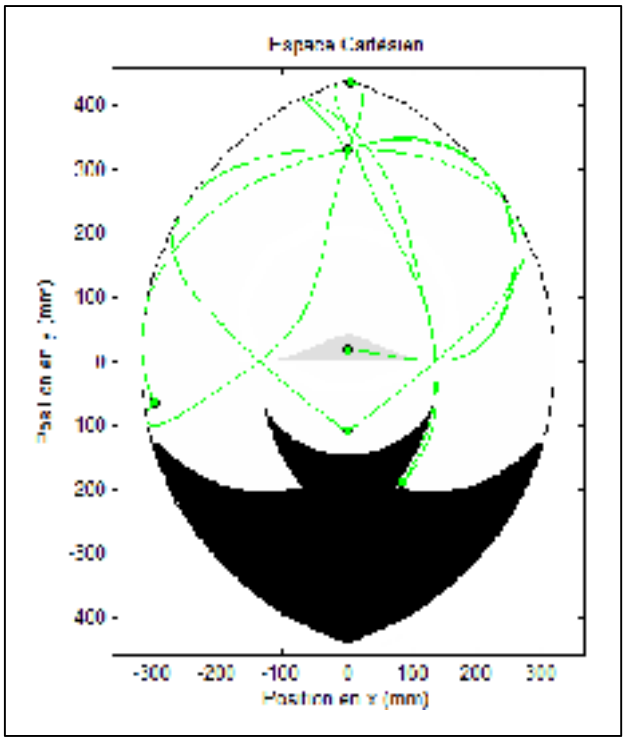


Figure 74 Trajectoire test dans l'espace cartésien

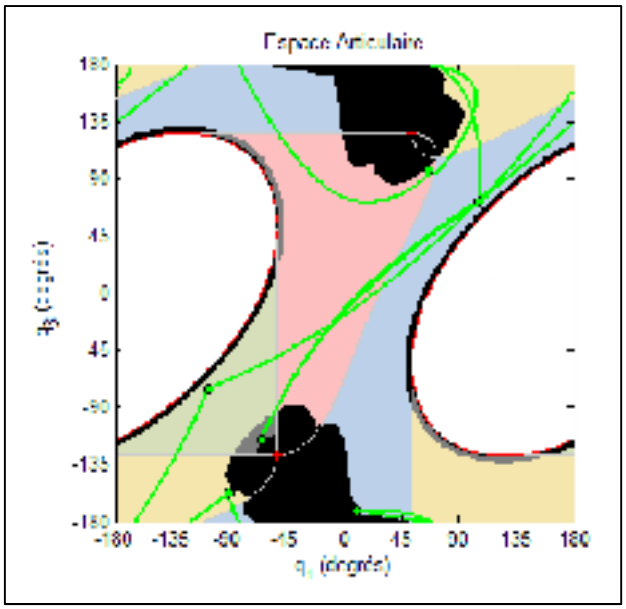


Figure 75 Trajectoire test

Tableau 26 Points de la trajectoire sélectionnée

#	Opération	X	Y	t(s)	WM
1	Point	0.000	330.000	---	- +
2	Point	85.000	-190.000	---	- +
3	Pause			0.2	
4	Point	5.000	135.000		+ +
5	Pause			0.2	
6	Point	300.000	65.000		
7	Pause			0.2	
8	Point	0.000	330.000	---	- +
9	Pause			0.2	
10	Point	0.000	-110.000	---	+ -
11	Pause			0.2	
12	Point	0.000	330.000	---	- +
13	Pause			0.2	
14	Point	0.000	18.000	---	+ -
15	Pause			0.2	
16	Point	0.000	330.000	---	- +

Une fois la trajectoire exécutée par le robot, les résultats sont acquis. Les figures 76 et 77 présentent les résultats du suivi de la trajectoire de positions des moteurs 1 et 2. L'erreur de positionnement est présentée à la figure 78 pour le moteur 1 et à la figure 79 pour le moteur 2. Celle-ci ne dépasse pas 0.7° pour les deux moteurs. Pour éviter d'alourdir la présentation des résultats, les courbes de suivi de vitesse et d'accélération ne sont pas présentées. Cependant les figures 80 et 81 présentent les erreurs de suivi de vitesse pour chaque moteur. L'erreur de suivi de vitesse ne dépasse pas 50 degrés/s pour les deux moteurs et l'amplitude maximale est de 1300 degrés/s.

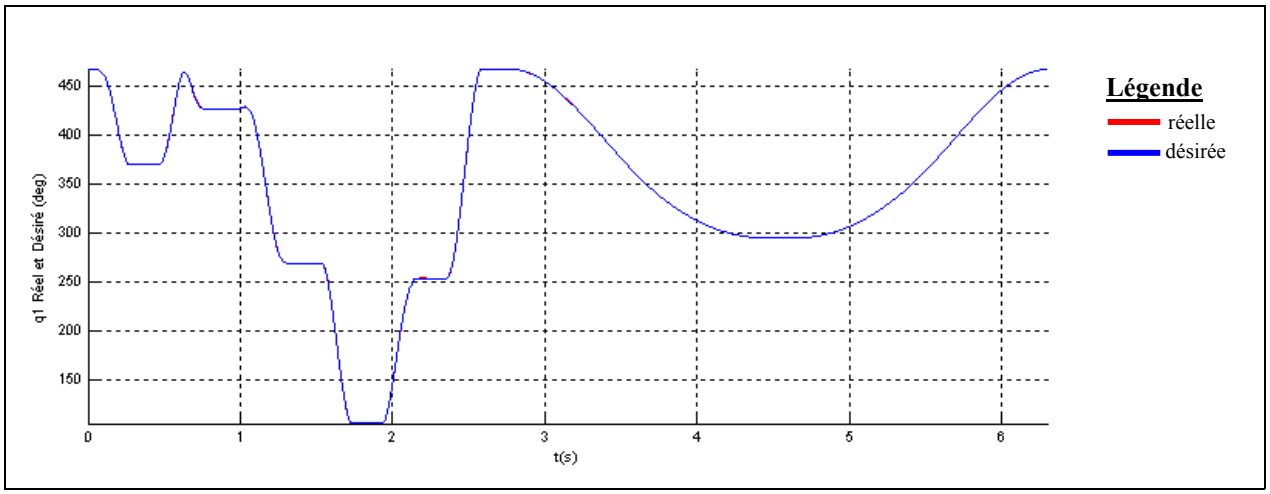


Figure 76 Suivi de position du moteur 1

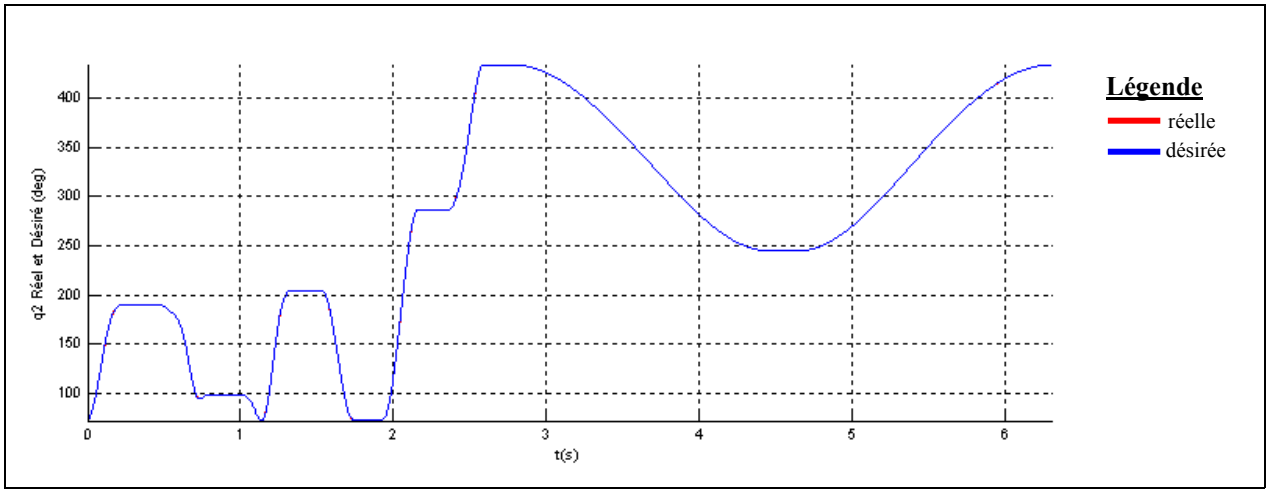


Figure 77 Suivi de position du moteur 2

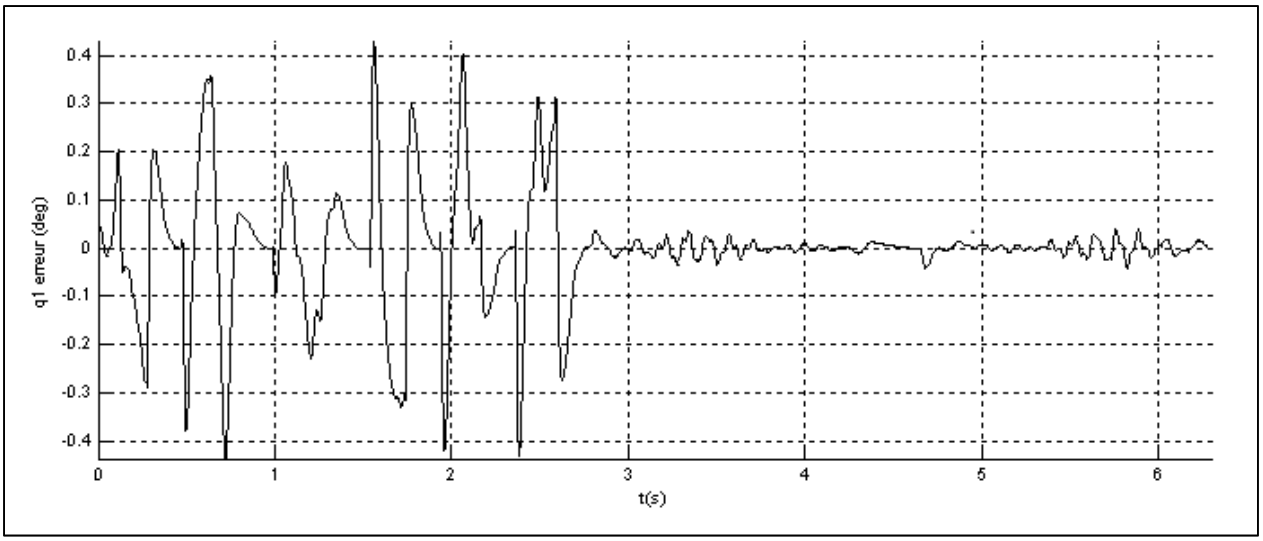


Figure 78 Erreur de positionnement du moteur 1

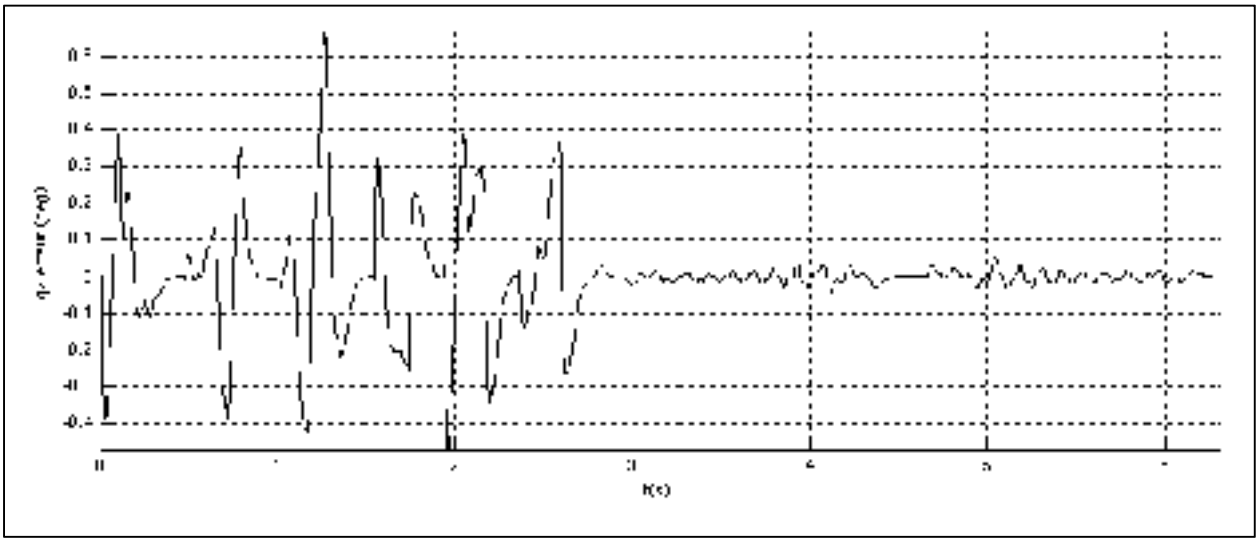


Figure 79 Erreur de positionnement du moteur 2

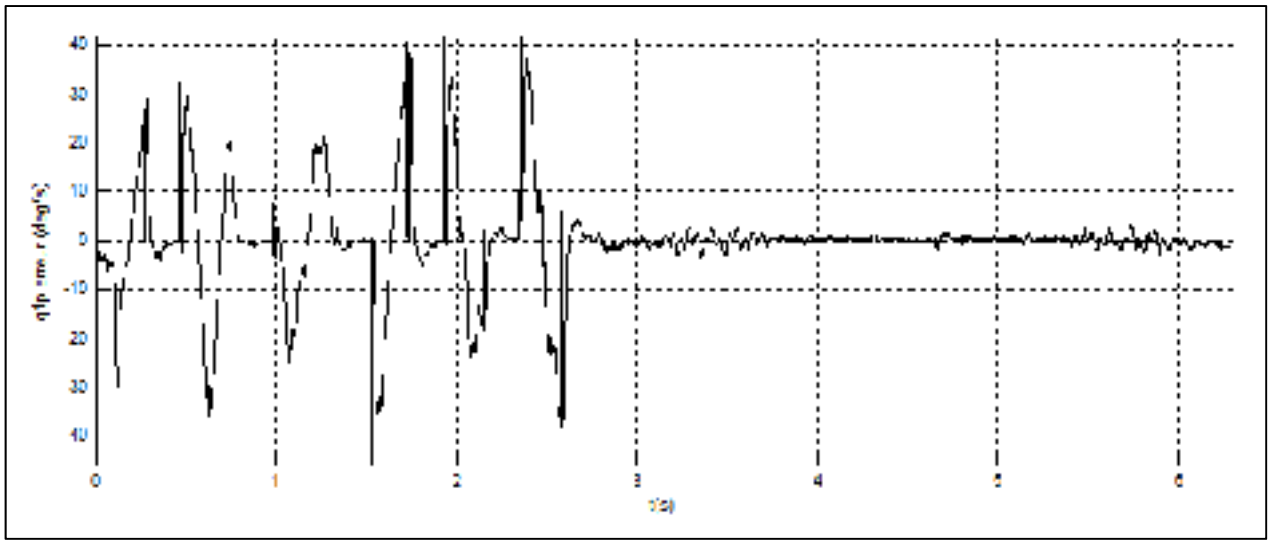


Figure 80 Erreur de suivi de vitesse du moteur 1

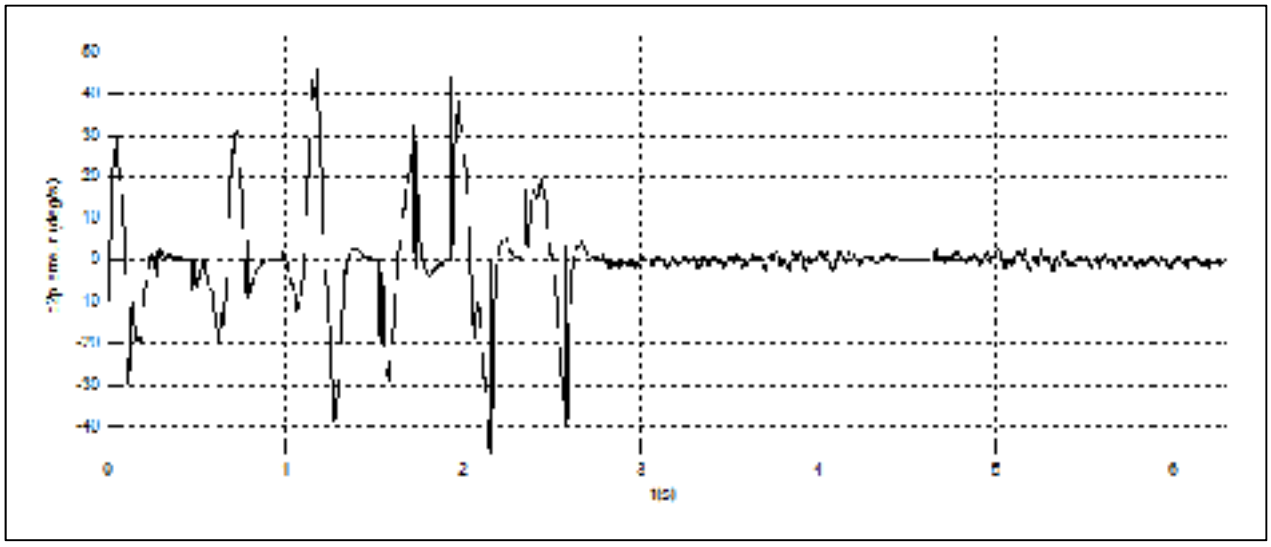


Figure 81 Erreur de suivi de vitesse du moteur 2

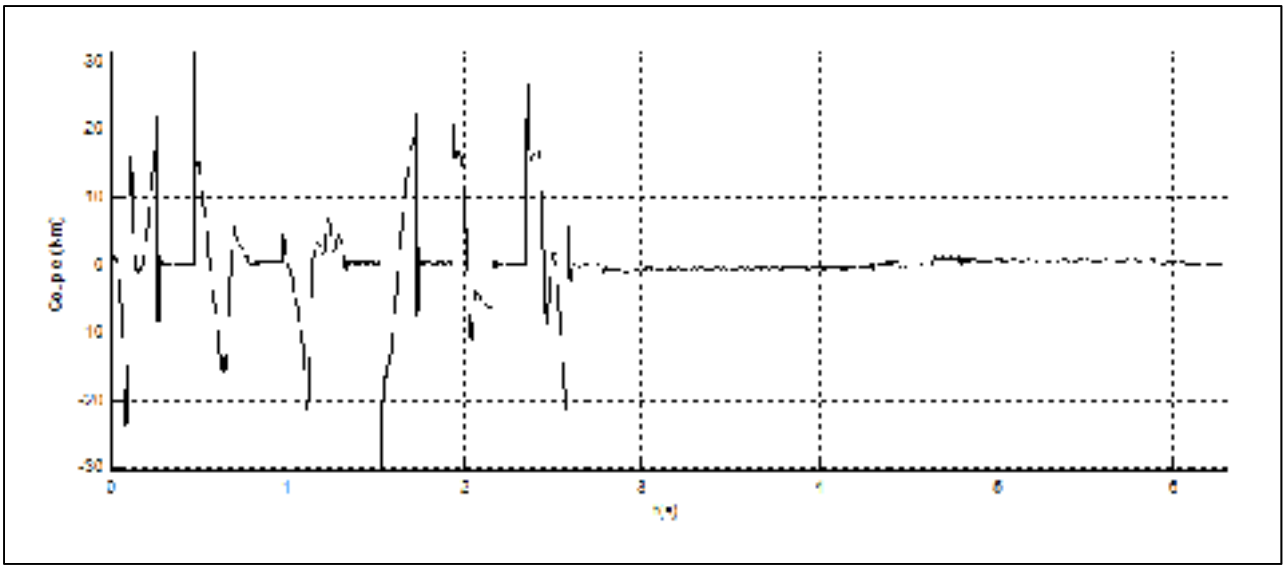


Figure 82 Couple appliqué au moteur 1

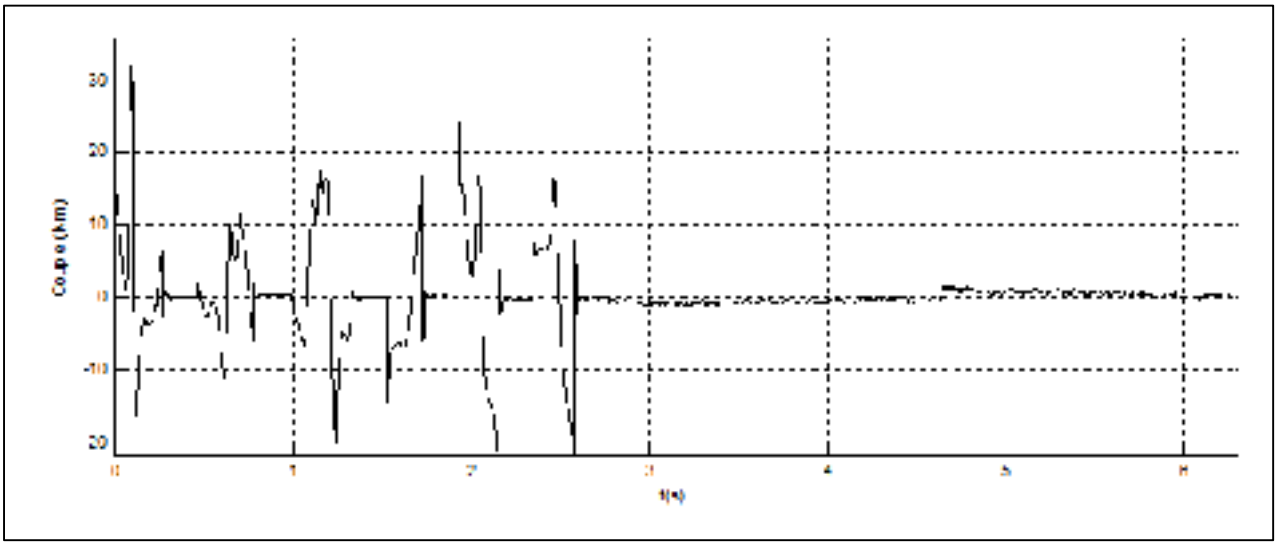


Figure 83 Couple appliqué au moteur 2

À titre d'information, les figures 82 et 83 présentent le couple appliqué aux moteurs durant la trajectoire. Les couples maximaux obtenus dépassent la limite maximale fixée dans la planification de trajectoire possiblement à cause que celle-ci de tient pas compte des incertitudes du modèle et du contrôleur. Cependant, les couples ont été fixé de façon conservatrice puisque le couple de pointe maximal des moteurs est de 60.6 Nm. Avec le

modèle dynamique mis en œuvre dans le contrôleur, le suivi de la trajectoire est assuré. Le but de ce mémoire n'est pas de comparer ou d'évaluer les performances du contrôleur. Cependant, au début des expérimentations, un simple PID était utilisé. L'inertie moyenne du robot pour chaque moteur avait été évaluée et les gains avaient été calculés en conséquence. Les performances avec ce contrôleur étaient modestes. L'erreur de suivi de trajectoire pouvait atteindre 6 degrés. Ainsi, le fait d'utiliser le modèle dynamique a permis d'améliorer significativement les performances du robot. En effet, le suivi de la trajectoire de test a engendré une erreur maximale de 0.7°.

7.4 Conclusion

Ce chapitre a présenté les résultats obtenus des divers sujets traités dans ce mémoire. Tout d'abord, plusieurs trajectoires ont été générées à l'aide des méthodes de planification de trajectoire présentées au chapitre 6. Ces trajectoires ont permis de montrer que la méthode des splines cubiques aléatoires avec un chemin libre était la plus efficace, mais qu'elle comporte aussi quelques limitations. De plus, certaines trajectoires ont montrés qu'il était possible d'utiliser les méthodes présentées pour générer des trajectoires comportant des obstacles ou pour suivre des formes géométriques. Par la suite, la technique pour déterminer l'espace de travail réel du robot a été présentée. Cette technique a permis d'évaluer l'espace de travail réel du robot à 292 110 mm^2. Comme l'objectif du mémoire est de démontrer que le changement de mode de fonctionnement peut accroître l'espace de travail, le robot DexTAR a été comparé à un robot de dimensions semblables. Les résultats sont éloquents puisque l'espace de travail est augmenté de 45.48%. Finalement, les performances du système de commande du robot ont été présentées relativement au suivi de trajectoire. Le contrôleur utilisé incorpore le modèle dynamique identifié du robot. Pour cette raison, les erreurs de suivi sont relativement faibles avec une valeur maximale de 0.7° pour une trajectoire présélectionnée.

CONCLUSION

L'objectif principal de ce mémoire était de démontrer que l'espace de travail d'un robot parallèle peut être augmenté considérablement en reconfigurant celui-ci automatiquement. Pour ce faire, deux types de singularités ont été étudiés. Les singularités de type 1 sont utilisées pour reconfigurer le robot. Le robot passe par ses singularités pour changer de mode de fonctionnement. Les singularités de type 2 doivent être évitées pour assurer la raideur du robot lors des déplacements.

Un autre objectif était de développer un algorithme de planification de trajectoire d'un robot à cinq barres qui permet le changement de mode de fonctionnement. Pour y arriver, la modélisation complète du robot a été réalisée, soit la modélisation géométrique, la modélisation géométrique inverse et la modélisation dynamique. Par la suite, une étude complète de l'espace de travail a été réalisée pour connaître les singularités et les interférences mécaniques du robot dans l'espace cartésien et dans l'espace articulaire.

L'algorithme de planification de trajectoire utilise la méthode des splines cubiques aléatoires. À priori, une carte de l'environnement discrétisée qui indique les obstacles dans l'espace articulaire doit être générée. Ensuite, l'algorithme calcule le chemin optimal en minimisant le temps de parcours et en considérant les contraintes de vitesse, d'accélération et de couple. Pour contraindre le couple, le modèle dynamique du robot est intégré au processus de planification.

Les différentes trajectoires générées ont été testées sur un vrai prototype. Les lois de commande ont été mise en oeuvre ainsi qu'une application qui permet de transférer et d'exécuter facilement les trajectoires.

Les résultats sont éloquents. La planification de trajectoire d'un chemin libre en utilisant la méthode des splines cubiques aléatoires peut améliorer de plus de 29% le temps de parcours comparé à une trajectoire générée en traçant une ligne droite dans l'espace articulaire. De

plus, l'espace de travail est considérablement augmenté en changeant de mode de fonctionnement. En effet, le robot DexTAR a été comparé à un robot industriel aux dimensions semblables. L'espace de travail du DexTAR est supérieur de 45.48%. Finalement, le contrôleur qui intègre le modèle dynamique a permis d'obtenir des erreurs de positionnement dynamique maximales de 0.7°.

Durant ce mémoire, plusieurs observations ont permis de proposer quelques recommandations. D'abord, en ce qui concerne le prototype DexTAR, l'utilisateur qui souhaiterait que le robot suive des trajectoires qui sont près des singularités de type 2 devrait être vigilant. Le robot devient facilement instable à l'approche de ces régions, particulièrement lorsque la trajectoire est rapide.

Aussi, d'autres recherches pourraient être effectuées pour que le robot puisse se déplacer dans tout l'espace de travail en considérant les contraintes maximales. Présentement, le robot est stable dans tout l'espace de travail, mais il ne peut atteindre une zone au centre lorsque les contraintes sont maximales. L'algorithme de planification considère ces zones de sorte que l'utilisateur n'aura pas de mauvaises surprises en utilisant le robot. Cependant, les performances sont alors dégradées.

Dans un autre ordre d'idée, des améliorations sont possibles au niveau de la planification de trajectoire. L'algorithme stochastique pour générer les trajectoires aléatoires est basique. Le fait de l'améliorer en utilisant des méthodes comme l'analyse par intervalles permettrait sans aucun doute d'améliorer le temps de calcul des trajectoires.

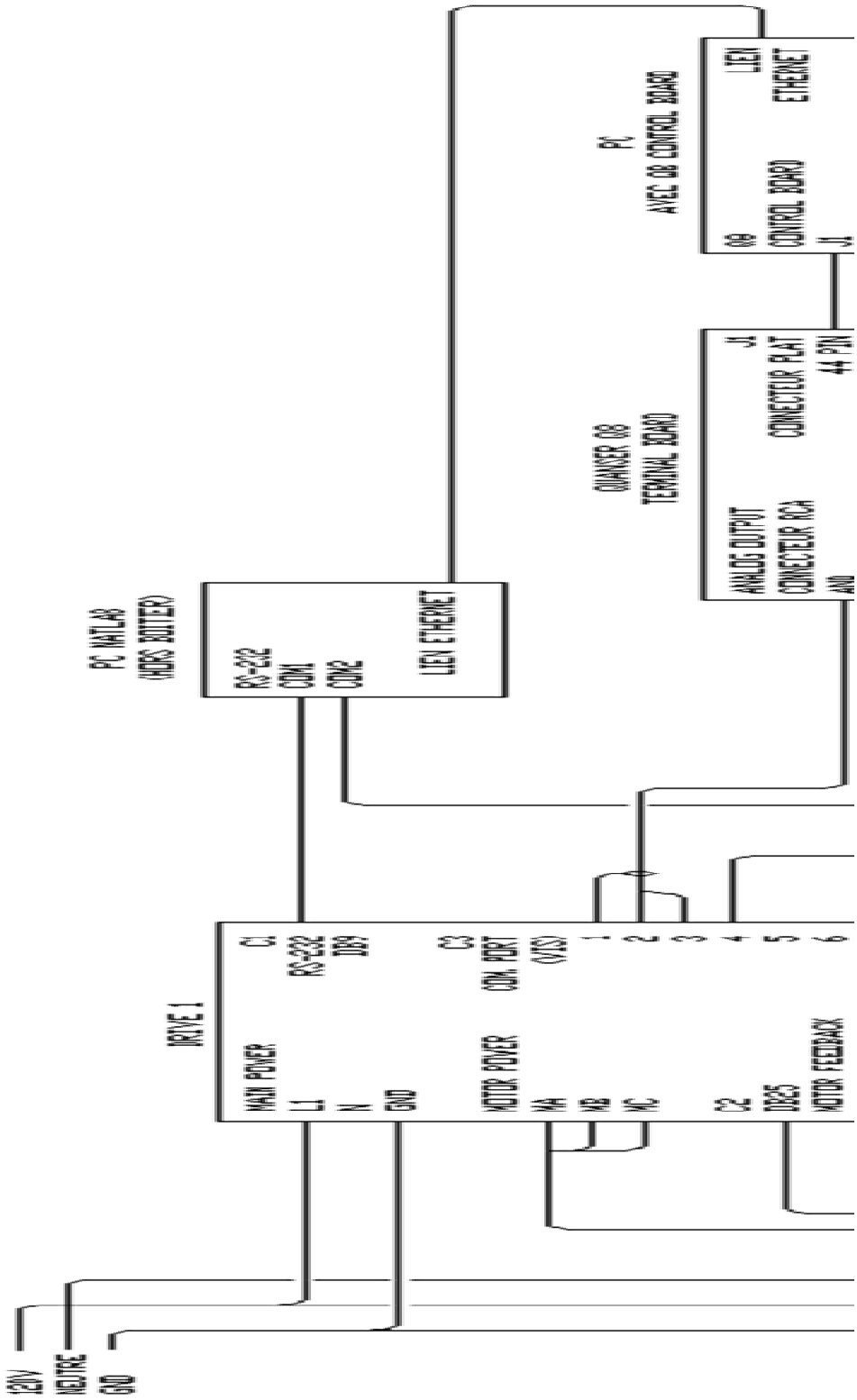
ANNEXE I BRANCHEMENTS DEXTAR

Le schéma de la page suivante montre le détail des branchements du robot.

Tableau du matériel

NOM	Fabriquant	Numéro de pièce
Drive 1	Kollmorgen	CB06550
Drive 2	Kollmorgen	CB06550
Moteur 1	Kollmorgen	D063M-12-1310
Moteur 2	Kollmorgen	D063M-12-1310
Câble Motor Power	Kollmorgen	CF-SS-S3HAE-03
Câble Motor FeedBack	Kollmorgen	CF-SS-S3HAE-03
PS 24V	Phoenix	MINI-PS-100-240AC/24DC/1.3
Relais Drive Enable	Phoenix	DEK-0V-5DC/24DC/3
Relais Axe Z	Phoenix	DEK-0V-5DC/24DC/3
Relais Succion	Phoenix	DEK-0V-5DC/24DC/3
Valve Solenoid (Axe Z)	Festo	MHE3-MS1H 3/26-QS-6
Valve Solenoid (Succion)	Festo	MHE3-MS1H 3/26-QS-6

Pour connaître les schémas de connections des connecteurs, les spécifications des câbles et l'ampérage des fusibles, il faut se référer aux manuels des différents fabricants.


+12V
NEUTRE
GND
DRIVE 1
MAIN POWER
L1
N
GND
C1
RS-232
DB9
C3
COM PORT
(VIS)
MOTOR POWER
MA
MB
MC
C2
DB25
MOTOR FEEDBACK
PC MATLAB
(MORS ROUTER)
RS-232
COM1
COM2
LIEN ETHERNET
ONDULEUR 108
TERMINAL BOARD
ANALOG OUTPUT
CONNECTEUR RCA
AN0
J1
CONNECTEUR PLAT
44 PIN
PC
AVEC QB CONTROL BOARD
J1
CONTROL BOARD
J69
LIEN
ETHERNET

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Abdellatif, Houssem, et Bodo Heimann. 2005. « Adapted time-optimal trajectory planning for parallel manipulators with full dynamic modelling ». In *International Conference on Robotics and Automation*. (Barcelona (Spain), Avril 2005), p. 411-416. Hannover (Germany).

Arsenault, Marc, et Roger Boudreau. 2004. « The Synthesis of Three Degree of Freedom Planar Parallel Mechanisms with Revolute Joints (3 RRR) for an Optimal Singularity Free Workspace ». *Journal of Robotic Systems,* vol. 21, n° 5, p. 259-274.

Barraquand, Jérôme, et Jean-Claude Latombe. 1991. « Robot motion planning: A distributed representation approach ». *The International journal of robotics research,* vol. 10, n° 6, p. 628.

Bigras, Pascal. 2003. « Maple: Bibliothèque de modélisation robotique ». En ligne. 23 p. < https://cours.etsmtl.ca/sys827/.%5CDocuments%5CLogiciels%5Crobot.zip >. Consulté le 29 juin 2011.

Bobrow, James E., Steven Dubowsky et John S. Gibson. 1985. « Time-optimal control of robotic manipulators along specified paths ». *The International journal of robotics research,* vol. 4, n° 3, p. 3-17.

Bonev, Ilian A. 2003. « Geometric analysis of parallel mechanisms ». Thèse de doctorat en génie en ligne. Québec (Canada), Université Laval, 189 p. In ÉTS. Montréal: École de technologie supérieure. < http://etsmtl.ca/Professeurs/ibonev/documents/pdf/ThesisBonev.pdf >. Consulté le 9 mars 2011.

Bonev, Ilian A., et Clément M. Gosselin. 2001. « Singularity loci of planar parallel manipulators with revolute joints ». p. 291–299.

Bourbonnais, Francis. 2012. *Guide d'utilisation du robot DexTAR*. PDF. Montréal (Canada): École de technologie supérieure, 29 p.

Campos, Lucas, Francis Bourbonnais, Ilian A. Bonev et Pascal Bigras. 2010. « Development of a Five-Bar Parallel Robot With Large Workspace ». In *Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2010*. (Montréal (Canada), 15 Août 2010), p. 917-922

Cao, Bailin, Gordon I Doods et Geroge W. Irwin. 1994. « Time-optimal and smooth constrained path planning for robot manipulators ». In *Proceedings of the 1994 IEEE*

International Conference on Robotics and Automation. (San Diego (USA)) Vol. 3, p. 1853-1858. IEEE.

Chen, I-Ming, Jorge Angeles et Chuan Li. 2003. « The management of parallel-manipulator singularities using joint-coupling ». In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*. (Taipei (Taiwan), 14-19 septembre 2003) Vol. 1, p. 773-778. IEEE.

Chen, Yaobin, et Stanley Y.-P. Chien. 1992. « General structure of time-optimal control of robotic manipulators moving along prescribed paths ». *International Journal of Control,* vol. 56, n° 4, p. 767-782.

Chettibi, Taha. 2006. « Synthesis of dynamic motions for robotic manipulators with geometric path constraints ». *Mechatronics,* vol. 16, n° 9, p. 547-563.

Chettibi, Taha, Moussa Haddad, A. Labed et S. Hanchi. 2005. « Generating optimal dynamic motions for closed-chain robotic systems ». *European Journal of Mechanics-A/Solids,* vol. 24, n° 3, p. 504-518.

Chettibi, Taha, Moussa Haddad, Samir Rebai et Abd Elfath Hentout. 2006. « A Stochastic Off Line Planner of Optimal Dynamic Motions for Robotic Manipulators ». *Informatics in Control, Automation and Robotics I*, p. 73-80.

Chettibi, Taha, et HE Lehtihet. 2002. « A new approach for point to point optimal motion planning problems of robotic manipulators ». In *6th Biennial Conference on Engineering Systems Design and Analysis (ASME)*. (Istanbul (Turkey), 8-11 juillet 2002), sous la dir. de School, Mechanical Laboratory of Structures Military Polytechnic.

Chettibi, Taha, HE Lehtihet, Moussa Haddad et S. Hanchi. 2004. « Minimum cost trajectory planning for industrial robots ». *European Journal of Mechanics-A/Solids,* vol. 23, n° 4, p. 703-715.

Costantinescu, Daniela , et Elizabeth A. Croft. 2000. « Smooth and time-optimal trajectory planning for industrial manipulators along specified paths ». *Journal of Robotic Systems,* vol. 17, n° 5, p. 233-249.

Craig, John J. 1989. *Introduction to robotics: mechanics and control*, 3e édition. Coll. « Pearson Prentice Hall ». Upper Saddle River (USA): Pearson Eaducation Inc., 400 p.

Dijkstra, Edsger W. 1959. « A note on two problems in connexion with graphs ». *Numerische mathematik,* vol. 1, n° 1, p. 269-271.

Dombre, Etienne, et Wisama Khalil. 2007. *Robot Manipulators: Modeling, Performance Analysis and Control*. Chippenham, Wiltshire (UK): Wiley-ISTE, 413 p.

Farritor, Shane, et Seven Dubowsky. 2002. « Genetic planning method and its application to planetary exploration ». *Journal of dynamic systems, measurement, and control,* vol. 124, p. 698-702.

Feder, Hans Jacob S., et Jean-Jacques E. Slotine. 1997. « Real-time path planning using harmonic potentials in dynamic environments ». In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*. (Albuquerque (USA), avril 1997) Vol. 1, p. 874-881. Cambridge, USA: IEEE.

Figielski, Alexandre, Ilian A. Bonev et Pascal Bigras. 2007. « Towards development of a 2-DOF planar oparallel robot with optimal workspace use ». In *2007. ISIC. IEEE International Conference on Systems, Man and Cybernetics*. (Montreal (Canada), 7-10 Octobre 2007), p. 1562-1566. IEEE.

Flash, Tamar, et Renfrey B. Potts. 1988. « Discrete trajectory planning ». *International Journal of Robotics Research,* vol. 7, p. 48-57.

Galicki, Miroslaw. 1992. « Optimal planning of a collision-free trajectory of redundant manipulators ». *The International journal of robotics research,* vol. 11, n° 6, p. 549.

Galicki, Miroslaw. 1998. « The planning of robotic optimal motions in the presence of obstacles ». *The International journal of robotics research,* vol. 17, n° 3, p. 248-259.

Gasparetto, Alessandro, et Vanni Zanotto. 2008. « A technique for time-jerk optimal planning of robot trajectories ». *Robotics and Computer-Integrated Manufacturing,* vol. 24, n° 3, p. 415-426.

Gosselin, Clément M., et Jorge Angeles. 1990. « Singularity analysis of closed-loop kinematic chains ». *IEEE Transactions on Robotics and Automation,* vol. 6, n° 3, p. 281-290.

Haddad, Moussa, Taha Chettibi, S. Hanchi et HE Lehtihet. 2007. « A random-profile approach for trajectory planning of wheeled mobile robots ». *European Journal of Mechanics-A/Solids,* vol. 26, n° 3, p. 519-540.

Hentout, Abd Elfath, HE Lehtihet, Taha Chettibi et B. Bouzouia. 2010. « Roadmap-Based Collision-Free Trajectory Planning for Manipulator Robots ». *The Journal of Modelling and Simulation of Systems (JMSS),* vol. 1, n° 1, p. 40-49.

Hu, Junfeng, Xianmin Zhang et Jinqing Zhan. 2008. « Trajectory planning of a novel 2-DOF high-speed planar parallel manipulator ». *Intelligent Robotics and Applications*, p. 199-207.

Jaulin, Luc, Michel Kieffer, Olivier Didrit et eric Walter. 2001. *Applied interval analysis*. Gateshead, Tyne and Wear (UK): Springer London.

Kenzai, Abderrahmane. 2005. « Planification de trajectoires de robots mobiles via des méthodes ensemblistes ». Mémoire DEA. Angers, Université d'Angers, 47 p.

Khalil, Wisama, et Etienne Dombre. 2002. *Modeling, Identification & Control of Robots*. New York (USA): Taylor & Francis Books Inc., 480 p.

Khatib, Oussama. 1986. « Real-time obstacle avoidance for manipulators and mobile robots ». *The International journal of robotics research,* vol. 5, n° 1, p. 90-98.

Lee, Sukhan, Hadi Moradi, George Kardaras et Sung-Kwon Kim. 1997. « Geometric collision detection and potential field based time delay planning for dual arm systems ». In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*. (Albuquerque (USA), avril 1997) Vol. 1, p. 882-887. IEEE.

Lin, Chun-Shin, PO-Rong Chang et J.Y.S. Luh. 1983. « Formulation and optimization of cubic polynomial joint trajectories for industrial robots ». *IEEE Transactions on Automatic Control,* vol. 28, n° 12, p. 1066-1074.

Liu, Xin-Ju, Jinsong Wang et G. Pritschow. 2006. « Performance atlases and optimum design of planar 5R symmetrical parallel mechanisms ». *Mechanism and machine theory,* vol. 41, n° 2, p. 119-144.

Mohri, Akira, Xiang Dong Yang et Motoji Yamamoto. 1995. « Collision free trajectory planning for manipulator using potential function ». In *1995 IEEE International Conference on Robotics and Automation* (Nagoya (Japon), 21-27 mai 1995) Vol. 3, p. 3069-3074 IEEE.

Murray, Richard M., Zexiang Li et S. Shankar Sastry. 1994. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton (USA): CRC Press.

Nabat, Vincent, Sébastien Krut, Olivier Company, Philippe Poignet et François Pierrot. 2008. « On the Design of a Fast Parallel Robot Based on Its Dynamic Model Experimental Robotics ». In, sous la dir. de Khatib, Oussama, Vijay Kumar et Daniela Rus. Vol. 39, p. 409-419. Coll. « Springer Tracts in Advanced Robotics »: Springer Berlin / Heidelberg. < http://dx.doi.org/10.1007/978-3-540-77457-0_38 >.

Neuman, Charles P. 1985. « Discrete dynamic robot models ». *IEEE transactions on systems, man, and cybernetics,* vol. 15, n° 2, p. 193-204.

Pfeiffer, Friedrich, et Rainer Johanni. 1987. « A concept for manipulator trajectory planning ». *IEEE Journal of Robotics and Automation,* vol. 3, n° 2, p. 115-123.

Pietsch, Ingo T., Oliver T. Becker, Mathias Krefft et Jürgen Hesselbach. 2003. « Time-optimal trajectory planning for adaptive control of plane parallel robots ». In *The Fourth International Conference on Control and Automation*. (Montréal (Qc), 10-12 juin 2003), p. 639-643. IEEE.

Rohrig, Christof, et Andreas Jochheim. 2001. « Identification and compensation of force ripple in linear permanent magnet motors ». In *Proceedings of the 2001 American Control Conference*. (Arlington (VA), 25-27 juin 2001) Vol. 3, p. 2161-2166

Schwartz, Pierre 2006. « Recherche de chemin par l'algorithme A* ». In *Developpez.com*. En ligne. < http://khayyam.developpez.com/articles/algo/astar/ >. Consulté le 10 Décembre 2010.

Shiller, Zvi, et Hsueh-Hen Lu. 1992. « Computation of path constrained time optimal motions with dynamic singularities ». *Journal of dynamic systems, measurement, and control,* vol. 114, p. 34-40.

Tan, H., et RB Potts. 1988. « Minimum time trajectory planner for the discrete dynamic robot model with dynamic constraints ». *IEEE Journal of Robotics and Automation,* vol. 4, n^{o} 2, p. 174-185.

Tse, Kai-Ming, et Chi-Hsu Wang. 1998. « Evolutionary optimization of cubic polynomial joint trajectories for industrial robots ». In *1998 IEEE International Conference on Systems, Man, and Cybernetics, 1998*. (San Diego (USA), 11-14 Octobre 1998) Vol. 4, p. 3272-3276. IEEE.

Vadakkepat, Prahlad, Kay Chen Tan et Wand Ming-Liang. 2000. « Evolutionary artificial potential fields and their application in real time robot path planning ». In *Proceedings of the 2000 Congress on Evolutionary Computation*. (La Jolla (USA), 16-19 juillet 2000) Vol. 1, p. 256-263. IEEE.

Vigneron, Vincent, Ryadh Kallel et Hichem Maaref. 2006. « Un algorithme pour la planification de trajectoire basé sur le calcul ensembliste ». *Rencontres Francophones sur la Logique Floue et ses Applications (LFA 2006), Toulouse : France (2006)*.

Wang, Jing, et Clément M. Gosselin. 2004. « Kinematic analysis and design of kinematically redundant parallel mechanisms ». *Journal of Mechanical Design,* vol. 126, p. 109-118.

Yasuda, Tomohiko, Dragomir N. Nenchev, Kazuo Aida et Hisashi Tamura. 2000. « Experiments with a parallel robot with singularity-perturbed design ». In *IECON 2000. 26th Annual Conference of the IEEE Industrial Electronics Society*. (Nagoya (Japon), 22-28 octobre 2000) Vol. 1, p. 217-222. IEEE.

Zlajpah, Leon. 1996. « On time optimal path control of manipulators with bounded joint velocities and torques ». In *Proceedings., 1996 IEEE International Conference on Robotics and Automation*. (Minneapolis (USA), 22-28 avril 1996) Vol. 2, p. 1572-1577. IEEE.

Zlatanov, Dimiter, Robert G. Fenton et Beno Benhabib. 1998. « Identification and classification of the singular configurations of mechanisms ». *Mechanism and machine theory,* vol. 33, n^{o} 6, p. 743-760.