Optimization in Robotique
Rapport de projet

Adrien Callico Nicolas Saunier
Université de Nantes
Master Informatique parcours Optimisation en Recherche Opérationnelle

novembre 2022

Table des matières

1 Introduction

Le dossier rendu comporte :

— les fichiers sources des différentes parties du projet

— les modèles et sorties du solveur IBEX (ibex files)

— les sorties graphiques (affichage du robot après calibration, path following, différents chemin du path planning)

— les librairies fournies (robot, etc)

Il faut de préférence exécuter main.py (contenant tout le déroulé du projet) dans un IDE adapté, car les figures ne s'affichent pas si on l'exécute depuis un terminal (via python3 main.py par exemple)

2 Calibration

La calibration se fait en deux étapes : on prend des mesures pour des angles allant de 0 à 90 pour q_1 et 135 à 180 pour q_2, et par la suite de 0 à 45 pour q_1 et 90 à 180 pour q_2.

Les fonctions cinématiques du robot 5R sont les suivantes :

$$f_1(x,q) = -l_2^2 + (-a_1 + x_1 - l_1 \times \cos(q_1))^2 + (x_2 - l_1 \times \sin(q_1) - a_2)^2$$

$$f_2(x,q) = -l_4^2 + (-a_3 + x_1 - l_3 \times \cos(q_1))^2 + (x_2 - l_3 \times \sin(q_1) - a_4)^2$$

On s'assure de la bonne calibration du robot en essayant d'atteindre deux points, $(5,-20)$ puis $(5,-20$, depuis les 4 solutions calculées avec IBEX (cf modèles assessing_calib.mbx et assessing_calib2.mbx). L'affichage en console atteste de la bonne précision de la calibration.

3 Path Following

On souhaite suivre le centre de centre $(0,-20)$ et de rayon 5. On le discrétise en 100 points via la formule indiquée dans le sujet en utilisant linspace de la librairie numpy.

On choisit le point de départ $(5,-20)$. On utilise IBEX pour calculer les commandes correspondant à ce point, et on en obtient 4, car chacun des deux bras peut être dans deux positions différentes.

On constate que chacune des solutions de départ conduit à des trajectoires différentes : si pour les solutions 1 et 4, le robot suit "correctement" le cercle (et ce pour les deux architectures), il rencontre une singularité pour la solution 2, et pour la solution 3, il trace un cercle déformé pour l'architecture nominale, et un légèrement décalé pour l'architecture calibrée. (cf les figures dans le dossier /out/pathfollowing)

4 Path planning

On souhaite maintenant se déplacer du point $(0,-15)$ au point $(0,15)$ en évitant des obstacles et sans passer par une singularité.

Les obstacles sont des cercles de centre (x_{1c}, x_{2c}) et de rayon r_c. On ajoute donc les contraintes suivantes, pour chaque cercle c :

$$(x_1 - x_{1c})^2 + (x_2 - x_{2c})^2 > r\text{š}$$

Maintenant, pour éviter les singularités, nous avons essayé d'ajouter les contraintes sur les déterminants des jacobiennes (énoncées dans le cours) :

— $\det \partial_q f(x,q) \neq 0$

— $\det \partial_x f(x,q) \neq 0$

Nous obtenons :

$$\partial_q f(x,q) = \begin{pmatrix} 2 \times (-a_1 \times l1 \times \sin(q_1) + l_1 \times a_2 * \cos(q_1) + l_1 \times x_1 \times \sin(q_1) - l_1 \times x_2 \times \cos(q_1)) & 0 \\ 0 & 2 \times (-a_3 \times l_2 \times \sin(q_2) + l_2 \times a_4 \times \cos(q_2) + l_2 \times x_1 \times \sin(q_2) - l_2 \times x_2 \times \cos(q_2)) \end{pmatrix}$$

$$\partial_x f(x,q) = \begin{pmatrix} 2 \times (x_1 - a_1 - 2 \times l_1 \times \cos(q_1)) & 2 \times (x_2 - a_2 - 2 \times l_1 \times \sin(q_1)) \\ 2 \times (x_1 - a_3 - 2 \times l_2 \times \cos(q_2)) & 2 \times (x_2 - a_4 - 2 \times l_2 \times \sin(q_2)) \end{pmatrix}$$

Nous avons rajouté les contraintes $(\det \partial_q f(x,q))^2 > 0$ et $(\det \partial_x f(x,q))^2 > 0$ au modèle IBEX, mais ça n'empêchait pas le robot de rencontrer des singularités (nous ne savons pas si c'est dû à une erreur de calcul, ou si ce n'est tout simplement pas suffisant de construire le pavage ainsi pour éviter les singularités). De plus, même si on forçait une petite largeur de boîte (-E0.6), certains centres de boîtes des chemins calculés étaient à l'intérieur d'un obstacle.

Pour les singularités, nous avons observé qu'elles se produisaient notamment lorsque le robot passait d'un mode à l'autre. Nous avons alors décidé de filtrer les chemins calculés pour chaque mode de la façon suivante :

— Pour chaque boîte, on regarde si les moyennes des bornes inférieures et supérieures conduisent à une singularité (via circles_intersections du fichier robot.py), en rejetant le chemin si c'est le cas,

— on fait aussi la vérification précédente, mais avec les coordonnées calculées de proche en proche (comme dans le path following) au lieu des coordonnées "théoriques" du centre de chaque boîte.

— on vérifie si le centre de chaque boîte est bien atteignable pour le mode en cours. Pour ce faire, on calcule la distance entre les coordonnées théoriques du point pour ce mode et les moyennes des commandes de la boîte (via circles_intersections) et le centre de la boite (en faisant les moyennes des bornes inférieures et supérieures pour chaque coordonnée) ; si elle est trop importante, c'est que cette boîte "ne correspond pas" à ce mode. Ainsi, on vérifie en fait si le chemin peut se faire sans jamais changer de mode.

Nous obtenons au final 2 chemins valides (1 pour chaque mode), au sens que le robot ne rencontre aucune singularité ni ne passe par un obstacle. Cependant, nous n'arrivons pas trop à expliquer le "détour" du deuxième chemin.