TR Théorie du contrôle - Sujet 2 : Schumacher

Paul Fraenkel

Contents

Part I
Commande en vitesses

1 Présentation du modèle

En première approximation, on considère que le robot est commandé en vitesse horizontale u et en vitesse de lacet Ψ, et en faisant cela on suppose donc qu'il y a roulement sans glissement au niveau du contact avec le sol. On modélise la dynamique du robot par les équations suivantes :

$$\begin{cases} \dot{x} &= ucos(\psi) \\ \dot{y} &= usin(\psi) \\ \dot{\psi} &= r \end{cases}$$

2 Modèle sans collisions

But : se familiariser avec le problème et les outils d'optimisation. Objectif : partir d'une position $(x_0, y_0, \psi_0, u_0, r_0)$ et aller à une position $(x_f, y_f, \psi_f, u_f, r_f)$ en minimisant le temps de déplacement T. Contraintes :

$$\begin{cases} -u_{max} \leq u \leq u_{max} \\ -r_{max} \leq r \leq r_{max} \end{cases}$$

Le problème est non-linéaire mais lisse, on peut donc utiliser un solveur numérique de points intérieurs (justifier pourquoi c'est un des plus adaptés)

2.1 Méthode d'optimisation pour un système régit par des équations différentielles

Il y a deux courants principaux pour l'optimisation d'équations différentielles : "optimize then discretize" et "discretize then optimize". On utilise ici la dernière : On discrétise l'équation différentielle selon un schéma défini (Euler explicite, Euler implicite, trapèzes, etc...), puis on utilise les égalités qui ne découlent comme contraintes de notre problème d'optimisation.

On note l'état $X = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$ et la commande $U = \begin{bmatrix} u \\ r \end{bmatrix}$

Le problème à discrétiser est le suivant :

$$\begin{aligned} \min_{X,U} \quad & T \\ & \dot{X} = B(X) \cdot U \\ \text{s.t.} \quad & X_0 = \bar{X_0} \\ & X_f = \bar{X_f} \end{aligned} \tag{1}$$

Cas du schéma d'Euler avec la méthode des trapèzes sur N+1 points :

$$\begin{aligned} \min_{X,U} \quad & T \\ & X_{i+1} = X_i + \frac{\delta t}{2}(B(X_i) \cdot U_i + B(X_{i+1}) \cdot U_{i+1}) \\ \text{s.t.} \quad & X_0 = \bar{X_0} \\ & X_N = \bar{X_N} \end{aligned} \tag{2}$$

Avec $\delta t = \frac{T}{N-1}$ et $B(X) = B(\psi) = \begin{bmatrix} cos(\psi) & 0 \\ sin(\psi) & 0 \\ 0 & 1 \end{bmatrix}$

2.2 Recherche analytique de la forme des solutions optimales

On va utiliser ici le principe du maximum de Pontryagin pour essayer d'en apprendre le plus possible sur la forme des solutions du problème 1.

2.2.1 Définition des grandeurs

Le principe est défini de la manière suivante :
On veut minimiser la fonction objectif

$$J(u) = h(x(tf), tf) + \int_{t_0}^{t_f} g(x(t), u(t))dt$$

pour le système dynamique $\dot{x} = a(x(t), u(t), t)$. On considère le Hamiltonien général

$$\mathcal{H}(x(t)) = g(x(t), u(t), t) + p^T(t)[a(x(t), u(t), t)]$$

dans le cas du problème en temps optimal, on a : $g = 1$ et $h = 0$ de manière à avoir $J(u) = tf$.
Le Hamiltonien à considérer pour notre problème est donc :

$$\mathcal{H}(x(t), u(t), p(t)) = 1 + p^T[B(x(t)) \cdot u(t)] \tag{3}$$

2.2.2 Conditions nécessaires d'optimalité de Pontryagin

La solution $(x^\star, u^\star, p^\star)$ doit vérifier les conditions nécessaires d'optimalité suivantes :

- $$\dot{x}^\star = \frac{\partial \mathcal{H}}{\partial p}(x^\star(t), u^\star(t), p^\star(t)) \tag{4}$$

- $$\dot{p}^\star = -\frac{\partial \mathcal{H}}{\partial x}(x^\star(t), u^\star(t), p^\star(t)) \tag{5}$$

- Pour chaque composante de la commande u_i :

$$\begin{cases} u_i^\star(t) & = u_{i,max} \quad \text{si} \quad {p^\star}^T(t) b_i(x^\star(t)) < 0 \\ u_i^\star(t) & = u_{i,min} \quad \text{si} \quad {p^\star}^T(t) b_i(x^\star(t)) > 0 \\ u_i^\star(t) & \text{indéterminé sinon} \end{cases} \tag{6}$$

  Avec $b_i(x(t))$ la colonne i de $B(x(t))$

- $$\left[\frac{\partial h}{\partial x}(x^\star(t_f), t_f) - p^s tar(t_f)\right]^T \delta x_f + \left[\mathcal{H}(x^\star(t_f), u^\star t_f, p^\star(t_f), t_f) + \frac{\partial h}{\partial t}\right] \delta t_f = 0 \tag{7}$$

Exploitation des conditions La condition 4 donne : $\dot{x}^\star(t) = B(x(t))u(t)$, la dynamique est bien respectée.
La condition 5 donne : $\dot{p}^\star(t) = {p^\star}^T(t) \frac{\partial B}{\partial \psi}(x^\star(t), u^\star(t))$. On en déduit

$$\begin{cases} p_1^\star(t) & = p_1^\star(0) = p_1 \\ p_2^\star(t) & = p_2^\star(0) = p_2 \\ p_3^\star(t) & = -u_1(t) \begin{bmatrix} p1 \\ p2 \end{bmatrix} \cdot \begin{bmatrix} -sin(\psi(t)) \\ cos(\psi(t)) \end{bmatrix} \end{cases} \tag{8}$$

Exploitons la condition 6 :
Si on se trouve sur un arc singulier pour la vitesse : ${p^\star}^T(t) b_1(x^\star(t)) = 0$, ce qui se réécrit $\begin{bmatrix} p1 \\ p2 \end{bmatrix} \cdot \begin{bmatrix} cos(\psi(t)) \\ sin(\psi(t)) \end{bmatrix} = 0$ La trajectoire est rectiligne, perpendiculaire au vecteur $\begin{bmatrix} p1 \\ p2 \end{bmatrix}$. Si on se trouve sur un arc singulier pour la rotation : ${p^\star}^T(t) b_2(x^\star(t)) = 0$, donc $p_3^\star(t) = 0$.
D'après le dernier résultat de la condition 5, $p_3^\star(t) = -u_1(t) \begin{bmatrix} p1 \\ p2 \end{bmatrix} \cdot \begin{bmatrix} -sin(\psi(t)) \\ cos(\psi(t)) \end{bmatrix} = 0$

$$\Leftrightarrow \begin{cases} u_1(t) = 0 \\ \text{ou} \\ \begin{bmatrix} p1 \\ p2 \end{bmatrix} \cdot \begin{bmatrix} -sin(\psi(t)) \\ cos(\psi(t)) \end{bmatrix} = 0 \quad \text{La trajectoire est parallèle au vecteur } \begin{bmatrix} p1 \\ p2 \end{bmatrix} \end{cases}$$

Comme le vecteur $\begin{bmatrix} p1 \\ p2 \end{bmatrix}$ apparaît plusieurs fois, nommons le v.

Interprétation Jusqu'ici, on sait que le robot ne tourne pas s'il se déplace dans la direction du vecteur v. Cela vient remettre en cause le cas où on se trouverait sur un arc singulier en vitesse qui prédit aussi que la trajectoire est rectiligne mais perpendiculaire au vecteur v. On en déduit que les arcs singuliers pour la vitesse sont instables. La vitesse atteint donc tout le temps ses extrema, sauf en des points singuliers où le robot est perpendiculaire au vecteur v.

3 Test annexe : champ de polygones

Objectif : mettre à l'épreuve le solveur en lui demandant de traverser un espace rempli de polygones convexes.

3.1 Génération des polygones

Pour des soucis de simplicité, les polygones ont tous le même nombre de faces.

3.1.1 Approche naïve

Dans l'implémentation du fichier `speed_model_random_terrain.jl`, j'ai utilisé des triangles pour avoir des polygones convexes.

Ils sont générés en positionnant de manière uniforme un point central dans une zone rectangulaire délimitée. Les 3 sommets du triangle sont générés uniformément dans un carré autour du point central associé.

3.1.2 Algorithm de Valtr

Afin de générer des polygones convexes avec un nombre de sommets défini, on peut utiliser l'algorithme de Valtr, qui a le bon goût d'être de complexité linéaire contrairement aux algorithmes de rejet. Le principe utilisé est de générer des vecteurs aléatoires, tels que leur somme soit nulle, puis de les ordonner par orientation afin de garantir la convexité du polygone.
Additionnellement, on peut adapter leur échelle pour qu'ils aient une aire spécifiée.

3.2 Représentation des polyèdres convexes

3.2.1 Représentation par ses faces

Dans la méthode décrite dans l'article [nacofinal.pdf -> mettre la vrai référence], un polygone $P^{(i)}$ est représenté par une matrice $C^{(i)}$ et un vecteur $d^{(i)}$ de la manière suivante :

$$P^{(i)} = \{y \in \mathbb{R}^3 \mid C^{(i)}y \leq d^{(i)}\} \tag{9}$$

Les polyèdres sont représentés par l'intersection non vide d'un nombre fini de demi-espaces fermés.

Une interprétation est la suivante :
Les lignes de $C^{(i)}$ sont les normales des faces, et les coefficients de $d^{(i)}$ associés sont les distances signées respectives de l'origine à ces faces.

Exemple Soit O l'origine du repère, n la normale à la face j du polygone $P^{(i)}$ et $M^{(i,j)}$ un sommet de $P^{(i)}$ appartenant à la face j.

$$\begin{cases} C_j^{(i)} & = n^T \\ d_j^{(i)} & = OM^{(i,j)} \cdot n \end{cases}$$

Pour obtenir les normales, on considère 2 sommets (consécutifs...), a et b, dont on calcule un vecteur orthogonal :

$$n^\star = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \frac{ab}{||ab||} \tag{10}$$

Soit c un autre sommet du polygone.

$$n = \begin{cases} +n^\star & \text{si} \quad n^\star \cdot ac \leq 0 \\ -n^\star & \text{sinon} \end{cases} \tag{11}$$

3.2.2 Représentation par ses sommets

Il existe une autre représentation d'un polyèdre convexe, qui utilise cette fois-ci ses sommets. Étant donné un ensemble de points $P = \{s_1, ..., s_n\}$, un point x appartient à l'enveloppe convexe de P ssi

$$\exists\{\lambda_1, \lambda_n\} \quad \text{tq} \begin{cases} \sum_{i=1}^n \lambda_i s_i = x \\ \sum_{i=1}^n \lambda_i = 1 \\ \forall i \in [1, n] \quad \lambda_i \geq 0 \end{cases}$$

3.3 Observations

Les essais ont été réalisés avec 101 points. L'algorithme de converge généralement assez lentement, sauf dans les cas où le robot peut atteindre l'objectif en ligne droite.
Le modèle est initialisé avec une solution en ligne droite entre le point de départ et le point d'arrivée. L'optimisation privilégie en général les solutions qui en sont proches, parfois au prix de tomber sur des minima locaux infaisables.

4 Test annexe : Parking

Demandons au robot d'effectuer un créneau dans une place de longueur l et de profondeur p.
Remarque : Avec la méthode utilisée pour décrire les collisions, il n'est pas obligatoire d'utiliser des polygones. Un polygone est représenté comme l'intersection de demi-espaces délimités par des droites (les faces du polygone). Il est donc possible de prendre des espaces ouverts comme "polygone".

La place de parking est modélisée par 3