

Project 2: Design of a parallel Schönflies motion generator

Abdellah KOUTIT
Arthur ASTIER

Mars 2023

Présentation du Problème:

Le deuxième sujet nous a beaucoup attiré car les robots de pick and place font partie intégrante du paysage industriel mondial depuis les années 1930, avec l'invention par Bill Taylor de la première grue robotisée.

S'en sont suivis des modèles de plus en plus développés, avec une architecture sérielle comme le robot SCARA (Selective Compliance Assembly Robot Arm) étant capable d'exercer un mouvement de Schönflies. Ce mouvement réside dans la capacité du robot à se déplacer linéairement dans l'espace et avec une seule rotation possible autour d'un axe fixe au cours du temps.

Il est donc idéal pour une opération de pick and place qui fait transiter un objet entre deux plans parallèles avec possiblement une orientation différente entre le premier et le second plan.

Cependant, dans certains cas la robotique sérielle n'est pas la plus performante. En effet, pour des applications rapides avec des objets légers on va préférer une architecture parallèle, qui permet d'avoir un déplacement des masses mieux réparti. Dans notre cas, le mouvement de Schönflies nous impose un mécanisme à 4 degrés de liberté PPPR. Ainsi, en regardant le livre de Kong Gosselin aux pages 155-156 on peut voir que des mécanismes parallèles avec quatre jambes identiques nous sont proposés. Nous avons choisi de partir sur un modèle composé uniquement de liaisons rotoïdes, dont les axes de rotation ne sont pas inclinés.

Le modèle choisi est le modèle (b) $R''R''R''R'R$.

1 Problème de conception :

1.1 Hypothèses:

En première approche, on souhaite simplifier un peu le problème. Nous allons donc faire un certain nombre d'hypothèses, réduisant ainsi la quantité de paramètres.

- Nous faisons l'hypothèse que les deux bras de chaque jambe seront de même section et de même matériau (d le diamètre des bras) .
- les paramètres (ρ et d) du matériau ne seront pas à optimiser ce qui est normal (les contraintes d'optimisation n'en dépendent pas)
- la position et la direction de la trajectoire ne sera pas à optimiser et puisque le système représente une symétrie par rapport à l'axe z donc on a choisit que la trajectoire sera dans le plan ($x = 0$)

1.2 Variables d'optimisation

$x=[x(1) \ x(2) \ x(3) \ x(4)]$

- $x(1) = l_1$ (longueur du premier bras)
- $x(2) = l_2$ (longueur du deuxième bras)
- $x(3) = a$ (coté du carré (plateforme mobile))
- $x(4) = b$ (coté du carré (Base))

1.3 Fonction objectif **obj.m**:

On souhaite minimiser la masse du robot donc la fonction à optimiser sera :

$$obj(x) = \rho \times \pi \times d^2 \times (l_1 + l_2 + a + b)$$

1.4 Contraintes `nonlcon.m`:

- $C_{eq}(l) = err$ représentent les contraintes géométrique et traduisent l'existence du modèle géométrique inverse le long de la trajectoire.
- $C(2 * l) = -1/cond(B) + 0.1$ représentent les contraintes de dextérité sérielles le long de la trajectoire.
- $C(2 * l - 1) = -abs(det(A)) + \epsilon$ représentent les contraintes de non-singularité de la jacobienne parallèle.

On a essayé de faire l'optimisation dans un premier temps avec la contrainte $cond(A) < max_{cond}$, mais le programme d'optimisation n'arrivait pas à converger même si en augmentant max_{cond} , jusqu'à la valeur 10^{12} du coup on a essayé d'optimiser en évitant les singularités de A sans avoir une très bonne dextérité.

2 Modèle géométrique inverse:

[Model.m](#) (click here to see code)

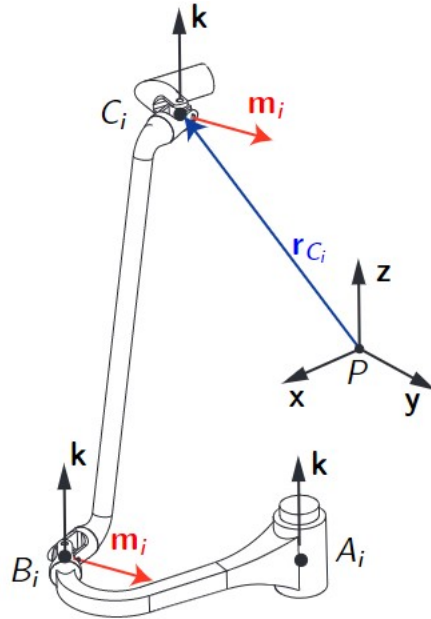


Figure 1: RUU limb

Pour déterminer le modèle géométrique inverse du mécanisme étudié (trouver le vecteur $[\phi_1 \phi_2 \phi_3 \phi_4]$) il suffit de déterminer les coordonnées du point B_i pour chaque jambe.

formulation mathématique:

soit $i = 1, 2, 3, 4$:

$$B_i \in \mathcal{C}(A_i, l_1) \cap \mathcal{S}(C_i, l_2)$$

Solution numérique:

Par une approche symbolique sur Matlab on a essayé de trouver une solution à ce problème d'inversion. ([Solu.m](#))

```
1 syms x_b y_b x_a x_c y_a y_c z_c l1 l2;  
2  
3 Cercle = (x_b-x_a)^2+(y_b-y_a)^2 == l1^2;  
4 Sphere = (x_b-x_c)^2+(y_b-y_c)^2+(z_c)^2 == l2^2;  
5  
6 eqn = [Cercle , Sphere];  
7 S= solve(eqn,[x_b,y_b]);  
8  
9 Sx=simplify(S.x_b(1));  
10 Sy=simplify(S.y_b(1));
```

On a implémenté le résultat trouvé dans la fonction $[B, err] = \text{trouver}_B(i, P, a, b, l1, l2)$ qui donne les coordonnées du point B s'il existe une inversion, sinon elle retourne une $err = 1$.

soit $x4$ le vecteur d'optimisation de la contrainte géométrique seulement.

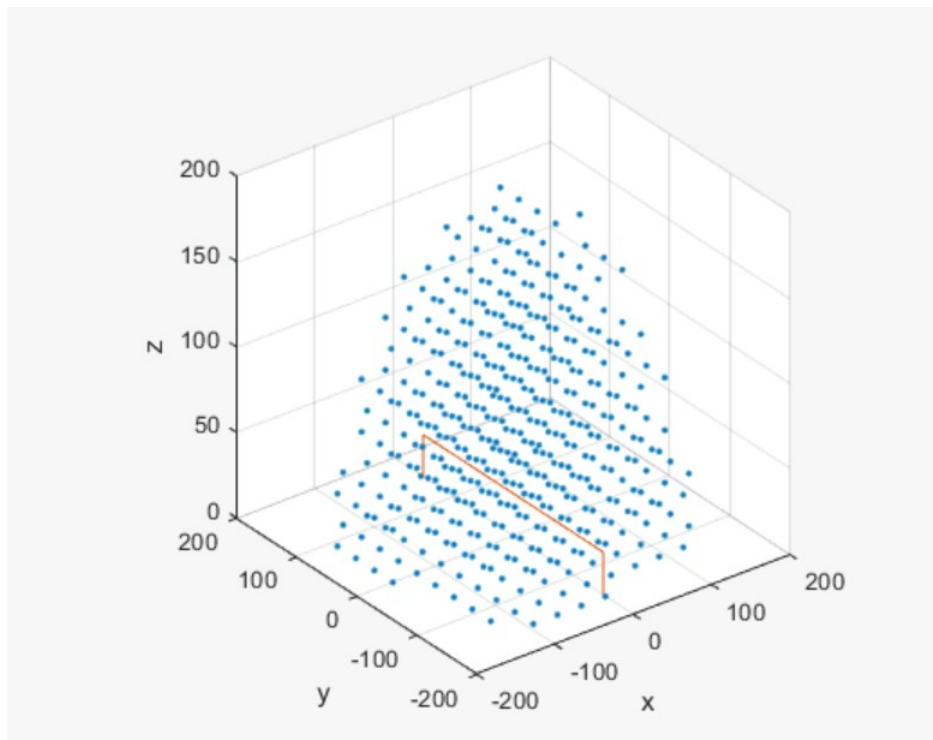


Figure 2: Espace atteignable par le robot correspondant a $x4$

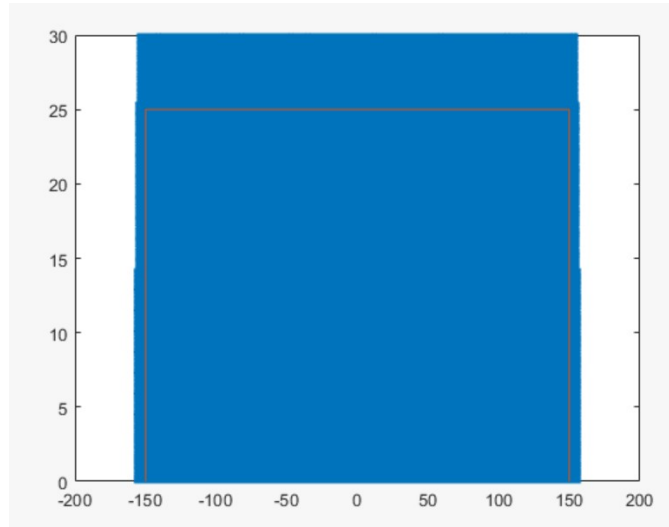


Figure 3: Espace atteignable suivant le plan de la trajectoire par le robot

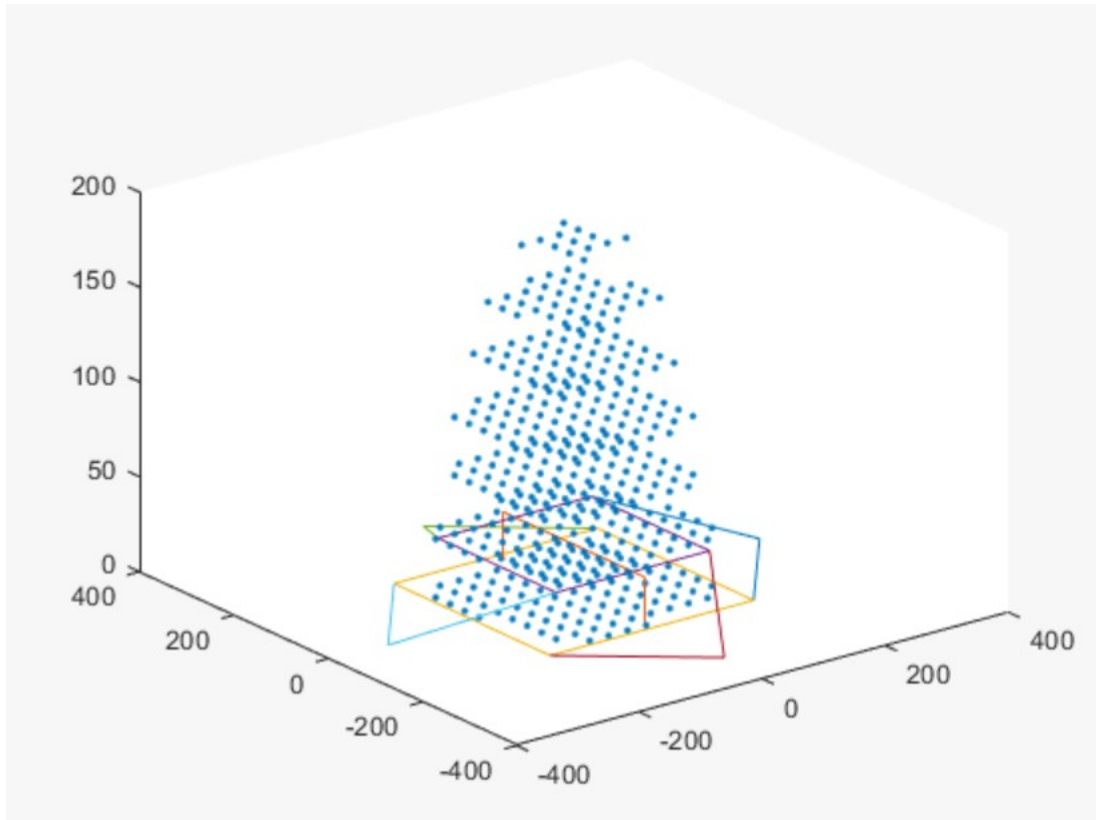


Figure 4: Espace atteignable par le robot correspondant a x_4

Remarque: en premier temps on a intégré le code ([Solu.m](#)) dans la fonction [trouver_B.m](#) ça marchait mais après pour la partie optimisation il fallait attendre la résolution symbolique pour chaque itération c'est pour cela qu'on

a fait la résolution ailleurs et on a pris que les solutions.

En donnant les coordonnées du point P et l'orientation de l'effecteur, on définit les positions des points C_i cela est codé au niveau de la fonction $C_i = \text{trouver}_{C_i}(P, i, a)$, les points A_i sont fixes et générés par la fonction $A_i = \text{trouver}_{A_i}(i, b)$

validation du MGI:

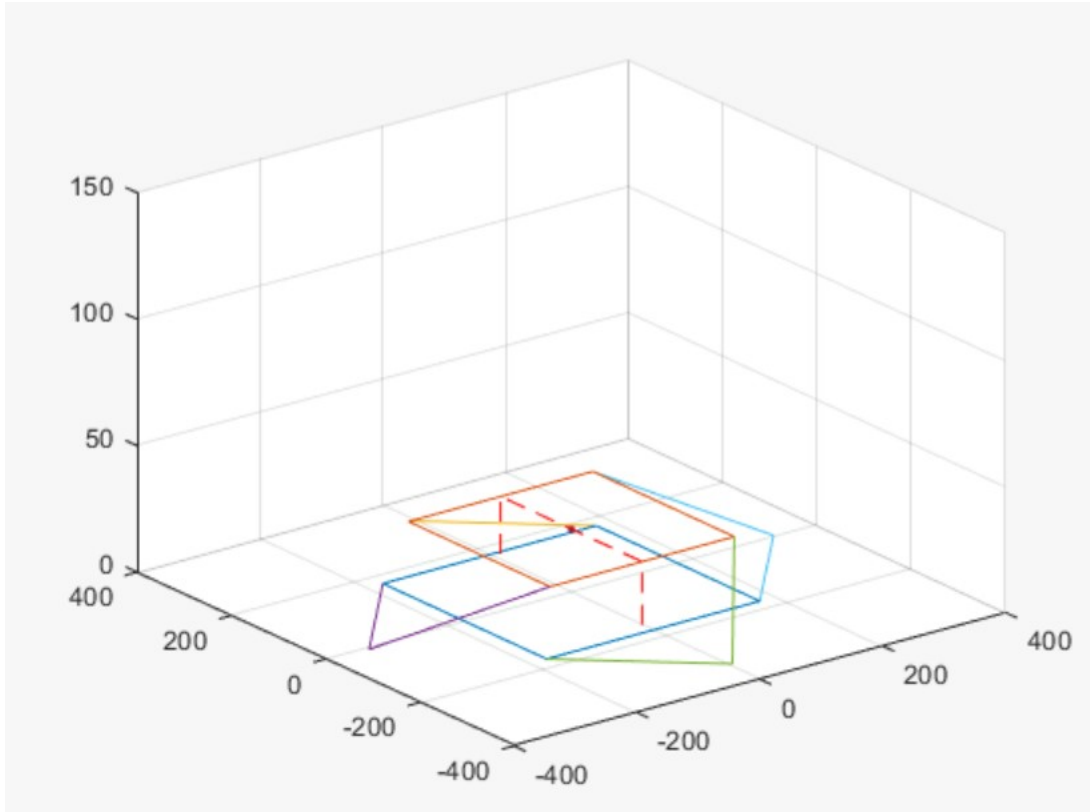


Figure 5: MGI pour $P = [0 \ 0 \ 0 \ \pi/2]$

Notre [MGI.m](#) donne qu'une seule solution parmi les 4^2 solutions, On s'intéresse dans notre problème seulement à l'existence de ces solutions !!

3 Matrices jacobienne:

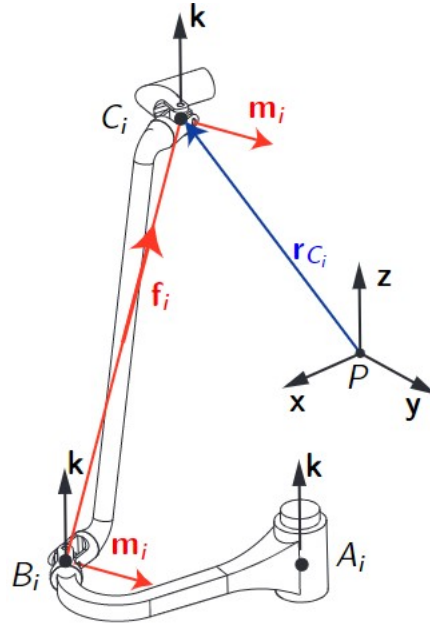


Figure 6: RUU limb

les deux fonctions $\text{Jacob}_A(P, b, a, l1, l2)$ et $\text{Jacob}_B(P, b, a, l1, l2)$ calcule les matrices jacobiennes réduites parallèle et sérielle pour une configuration du Robot.

$$A = \begin{bmatrix} (r_{C1} \times f_1)^T & f_1^T \\ (r_{C2} \times f_2)^T & f_2^T \\ (r_{C3} \times f_3)^T & f_3^T \\ (r_{C4} \times f_4)^T & f_4^T \\ (k \times m_1)^T & O_3^T \\ (k \times m_2)^T & O_3^T \end{bmatrix}$$

$$B = \begin{bmatrix} (\overrightarrow{A_1 C_1} \times f_1)^T k & 0 & 0 & 0 & 0 & 0 \\ 0 & (\overrightarrow{A_2 C_2} \times f_2)^T k & 0 & 0 & 0 & 0 \\ 0 & 0 & (\overrightarrow{A_3 C_3} \times f_3)^T k & 0 & 0 & 0 \\ 0 & 0 & 0 & (\overrightarrow{A_4 C_4} \times f_4)^T k & 0 & 0 \\ 0 & 0 & 0 & 0 & (\overrightarrow{A_5 C_5} \times f_1)^T k & 0 \\ 0 & 0 & 0 & 0 & 0 & (\overrightarrow{A_6 C_6} \times f_1)^T k \end{bmatrix}$$

4 Résultats:

Pour l'aluminium ($\rho=2800 \text{ kg/m}^3$, $d = 0,02/ m$) on a trouvé:

4.1 Optimisation en appliquant les contraintes géométriques:

```
1 %Aluminium
2 x4 =
3
4 240.7638 234.7732 300.0884 347.1476
5 % l1 l2 a b
6
7 >> m=obj(x4)
8 m =
9
10 3.9506
```

run code [animation.m](#) to see [animation.mp4](#).

4.2 Optimisation en appliquant les contraintes géométriques sans considérer les singularités de la matrice A

```
1 %Aluminium
2 x7 =
3
4 330.4928 331.0376 473.7821 472.4131
5 % l1 l2 a b
6
7 >> m=obj(x7)
8 m =
9
10 5.6569
```

4.3 Optimisation en appliquant toutes les contraintes géométriques et cinématiques

```
1 %Aluminium
2 x =
3
4 427.9689 406.2424 664.9646 568.9070
5 % l1 l2 a b
6
7 >> m=obj(x)
8 m =
9
10 7.2767
```