Rapport de TP : Résolution du Modèle Géométrique Inverse par Optimisation Numérique

Étudiant : ABOURICH Ayoub
Filière : Génie Électrique (Option ARII)

3 février 2026

Résumé

Ce rapport présente la résolution du problème inverse de cinématique pour des robots manipulateurs plans. L'approche traditionnelle analytique étant parfois complexe ou inadaptée aux robots redondants, nous proposons une méthode numérique basée sur la minimisation d'une fonction coût sous contraintes. L'étude porte sur un robot à 2 degrés de liberté (RR) puis est généralisée à un robot redondant à 3 degrés de liberté (RRR), mettant en évidence l'efficacité de l'algorithme de descente de gradient (via `fmincon` sous MATLAB) pour converger vers une solution optimale.

Table des matières

1 Introduction et Contexte

En robotique, le **Modèle Géométrique Inverse (MGI)** consiste à déterminer les coordonnées articulaires $\mathbf{q}$ nécessaires pour placer l'organe terminal du robot à une position et une orientation désirées dans l'espace cartésien.

Pour un robot sériel simple, une solution analytique (fermeture géométrique) est souvent possible. Cependant, pour des robots complexes ou **redondants** (nombre de DDL $>$ dimension de la tâche), l'inversion analytique devient ardue. L'approche par optimisation numérique permet de contourner cette difficulté en transformant le problème d'inversion en un problème de minimisation d'erreur.

L'objectif de ce TP est d'implémenter cette méthode pour atteindre la cible cartésienne $P_{cible} = (1.2, 0.5)$ m.

2 Partie 1 : Étude du Robot RR (2 DDL)

2.1 Modélisation du Robot

Le robot est constitué de deux corps rigides reliés par des liaisons pivots.
— Longueurs des segments : $L_1 = 1.0$ m, $L_2 = 0.8$ m.
— Variables articulaires : $\mathbf{q} = [q_1, q_2]^T$.

2.1.1 Question 1 : Modèle Géométrique Direct (MGD)

Le MGD permet de calculer la position (x, y) de l'effecteur en fonction des angles :

$$\begin{bmatrix} x \\ y \end{bmatrix} = f(\mathbf{q}) = \begin{bmatrix} L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) \\ L_1 \sin(q_1) + L_2 \sin(q_1 + q_2) \end{bmatrix} \tag{1}$$

2.1.2 Question 2 : Espace de Travail

L'espace de travail représente l'ensemble des points atteignables par l'effecteur. Nous l'avons généré par une méthode de Monte-Carlo (tirage aléatoire de $N = 3000$ configurations).

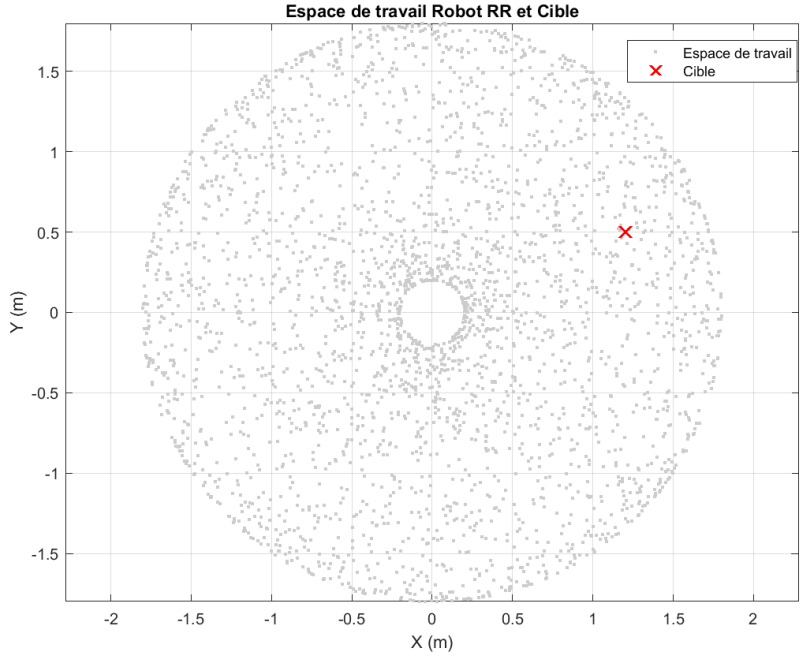


FIGURE 1 – Nuage de points de l'espace de travail du robot RR. La croix rouge indique la cible.

Analyse : La densité des points est plus faible en périphérie. La cible $(1.2, 0.5)$ est clairement située à l'intérieur de l'enveloppe de travail (dont le rayon max est $L_1 + L_2 = 1.8$m), ce qui garantit l'existence d'au moins une solution.

2.2 Formulation du Problème d'Optimisation

2.2.1 Question 3 : Fonction Coût

Nous définissons une fonction objective scalaire $J(\mathbf{q})$ qui quantifie l'écart entre la position atteinte et la cible. Nous utilisons la norme Euclidienne de l'erreur :

$$J(\mathbf{q}) = \|\mathbf{X}_{cible} - \mathbf{X}_{MGD}(\mathbf{q})\|_2 = \sqrt{(x_d - x)^2 + (y_d - y)^2} \tag{2}$$

Le problème devient alors : trouver $\mathbf{q}^* = \arg\min_{\mathbf{q}} J(\mathbf{q})$.

2.2.2 Questions 4 et 5 : Résolution sous Contraintes

Nous utilisons le solveur `fmincon` de MATLAB. Contrairement à une simple descente de gradient, ce solveur permet de gérer explicitement les butées articulaires (inégalités).

— **Algorithme :** *Interior-Point* ou *SQP* (Sequential Quadratic Programming). Ces méthodes utilisent le gradient ∇J et le Hessien pour converger rapidement.

— **Contraintes (Question 5) :** Les angles sont bornés physiquement :

$$-\pi \leq q_i \leq \pi, \quad \forall i \in \{1, 2\} \tag{3}$$

2.3 Résultats et Validation

2.3.1 Question 6 : Vérification Numérique

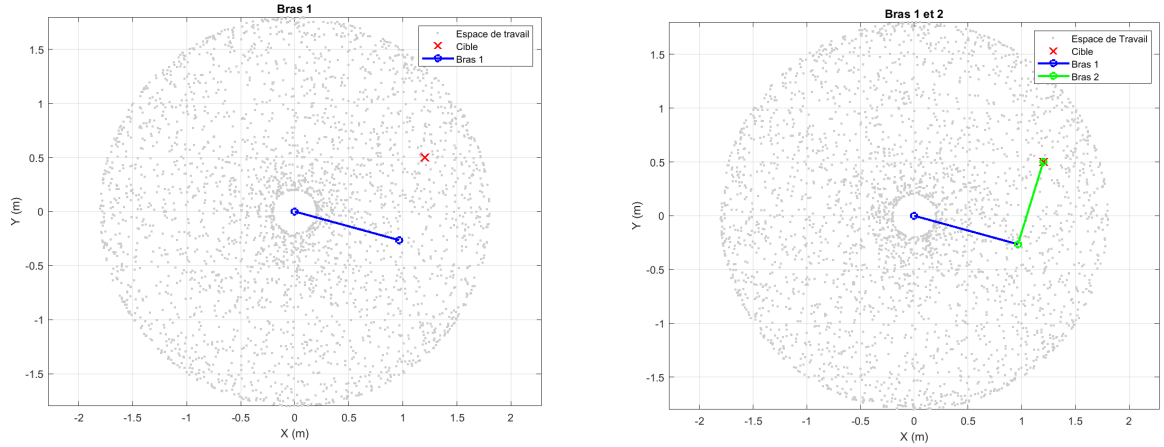
L'optimisation a convergé vers la solution suivante :

$$\mathbf{q}^* = [-15.34^\circ, 88.21^\circ]$$

En réinjectant ces valeurs dans le MGD, nous trouvons la position $(1.2000, 0.5000)$. L'erreur résiduelle est inférieure à la tolérance du solveur (10^{-6}), validant mathématiquement le résultat.

2.3.2 Question 7 : Validation Graphique

Les figures ci-dessous illustrent la configuration géométrique trouvée par l'algorithme.



(a) Positionnement du bras 1 (b) Configuration finale (Bras 1+2)

FIGURE 2 – Visualisation de la solution cinématique pour le robot RR.

Interprétation : Le robot adopte une configuration "coude bas". Il est important de noter que pour ce robot non-redondant, il existe généralement deux solutions (coude haut / coude bas). L'algorithme a convergé vers celle-ci car elle était probablement plus proche du point d'initialisation $\mathbf{q}_0 = [0, 0]$.

3 Partie 2 : Extension au Robot Redondant RRR (Question 8)

3.1 Problématique de la Redondance

Nous ajoutons un troisième bras de longueur $L_3 = 0.6$ m. Le robot possède maintenant 3 DDL pour une tâche plane (2 DDL : x, y).

$$\text{Degré de redondance} = n - m = 3 - 2 = 1$$

Cela signifie qu'il existe une **infinité de solutions** pour atteindre la même cible. L'optimisation ne cherche plus "la" solution, mais "une" solution optimale (souvent celle qui minimise le déplacement par rapport à la position initiale).

3.2 Modélisation

— **Nouveau MGD :**

$$x = L_1 c_1 + L_2 c_{12} + L_3 c_{123}$$
$$y = L_1 s_1 + L_2 s_{12} + L_3 s_{123}$$

— **Espace de travail :** Il est plus vaste que celui du robot RR, offrant une meilleure accessibilité.

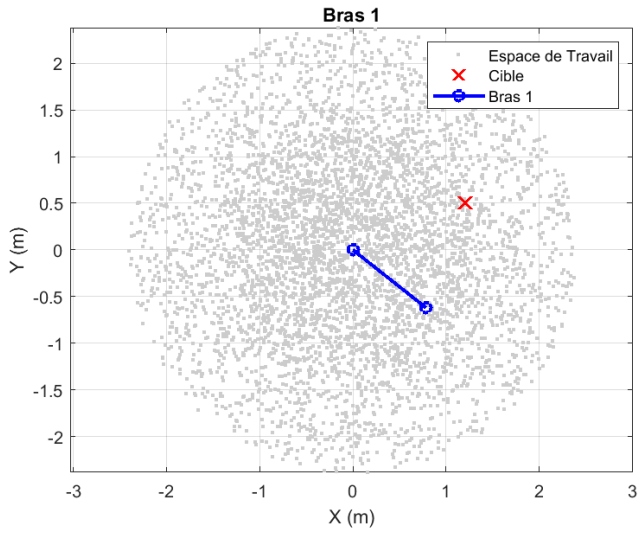


FIGURE 3 – Positionnement du premier bras (Bras 1) dans l'espace de travail.

3.3 Résultats de l'optimisation

L'algorithme a été relancé avec le vecteur d'état $\mathbf{q} \in \mathbb{R}^3$.

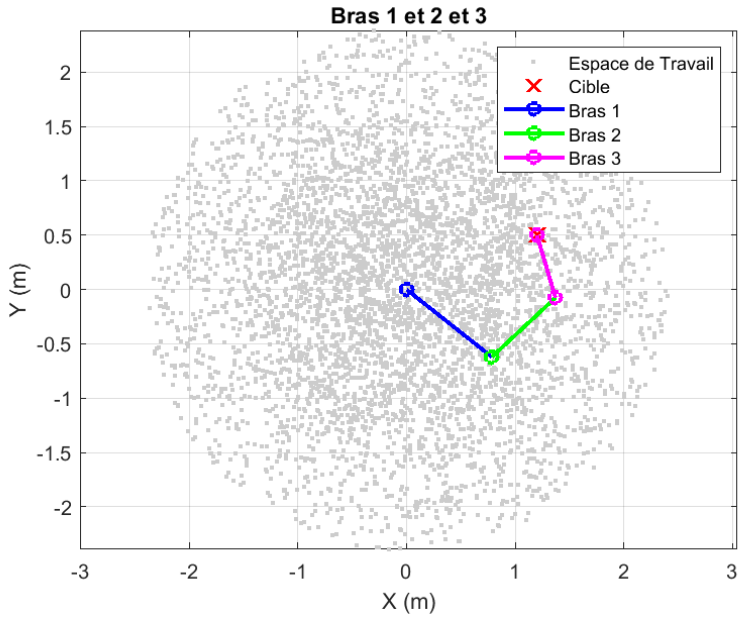


FIGURE 4 – Solution optimale pour le robot RRR.

Discussion : Le robot atteint la cible avec succès. La forme "repliée" du bras montre comment l'optimiseur a utilisé le degré de liberté supplémentaire pour satisfaire la contrainte de position tout en minimisant la "distance" dans l'espace des configurations par rapport à l'origine.

4 Conclusion

Ce travail pratique a mis en évidence la puissance des méthodes numériques pour la résolution du MGI.

- Pour le **robot RR**, la méthode retrouve efficacement l'une des solutions analytiques possibles.
- Pour le **robot RRR**, la méthode gère naturellement la redondance sans nécessiter de calculs complexes (comme la pseudo-inverse du Jacobien).

L'utilisation de `fmincon` garantit également le respect des limites articulaires, ce qui est un avantage majeur pour la commande de robots réels.

A Annexes : Codes MATLAB

Les codes ci-dessous implémentent la résolution complète décrite dans ce rapport.

A.1 Code Principal - Robot RR

```
% --- Parametres du Robot RR ---
L1 = 1.0; L2 = 0.8;
X_target = 1.2; Y_target = 0.5;

% 1. Generation Espace de travail
N = 3000;
q1_rand = (rand(N,1)*2*pi) - pi;
q2_rand = (rand(N,1)*2*pi) - pi;
X_ws = L1*cos(q1_rand) + L2*cos(q1_rand+q2_rand);
Y_ws = L1*sin(q1_rand) + L2*sin(q1_rand+q2_rand);

figure(1); clf; hold on;
plot(X_ws, Y_ws, '.', 'Color', [0.8 0.8 0.8]);
plot(X_target, Y_target, 'rx', 'MarkerSize', 10, 'LineWidth', 2);
title('Espace de travail Robot RR'); grid on; axis equal;

% 2. Definition de la Fonction Cout
costFunction = @(q) sqrt( ...
    (X_target - (L1*cos(q(1)) + L2*cos(q(1)+q(2))))^2 + ...
    (Y_target - (L1*sin(q(1)) + L2*sin(q(1)+q(2))))^2 );

% 3. Optimisation fmincon
q0 = [0, 0]; % Initialisation
lb = [-pi, -pi]; ub = [pi, pi]; % Contraintes
options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'sqp')
    ;

[q_opt, fval] = fmincon(costFunction, q0, [], [], [], [], lb, ub, [],
    options);

% 4. Affichage Resultats
x0=0; y0=0;
x1=L1*cos(q_opt(1)); y1=L1*sin(q_opt(1));
x2=x1+L2*cos(q_opt(1)+q_opt(2)); y2=y1+L2*sin(q_opt(1)+q_opt(2));

plot([x0 x1], [y0 y1], 'b-o', 'LineWidth', 2);
plot([x1 x2], [y1 y2], 'g-o', 'LineWidth', 2);
legend('Workspace', 'Cible', 'Link 1', 'Link 2');
```

A.2 Code Principal - Robot RRR

```
% --- Parametres du Robot RRR ---
L1 = 1.0; L2 = 0.8; L3 = 0.6;
X_target = 1.2; Y_target = 0.5;

% Definition fonction cout RRR
costFunction = @(q) sqrt( ...
```

```
    (X_target - (L1*cos(q(1)) + L2*cos(q(1)+q(2)) + L3*cos(q(1)+q(2)+q(3))))^2 + ...
    (Y_target - (L1*sin(q(1)) + L2*sin(q(1)+q(2)) + L3*sin(q(1)+q(2)+q(3))))^2 );

% Optimisation
q0 = [0, 0, 0];
lb = [-pi, -pi, -pi]; ub = [pi, pi, pi];
options = optimoptions('fmincon', 'Display', 'off');
q_opt = fmincon(costFunction, q0, [], [], [], [], lb, ub, [], options);

% Trace (simplifie)
figure(2); clf; hold on;
% (Code de trace similaire au cas RR avec 3 segments)
```