**Resolution IK par Pseudo Inverse pour le suivi d’une trajectoire**

Ce code fait comme le précédent sauf qu’il travaille ici avec des trajectoire et cree une animation du robot qui suit cette trajectoire

% Définir les symboles pour les angles d'articulation

syms theta1 theta2 theta3 theta4;

% Définir les longueurs des bras

a1 = 2; % Longueur du premier bras

a2 = 3; % Longueur du deuxième bras

a3 = 2; % Longueur du troisième bras

% Définir la trajectoire souhaitée (par exemple, un cercle)

num\_points = 30;

radius = 6.6;

theta\_traj = linspace(0, 2\*pi, num\_points);

x\_traj = radius \* cos(theta\_traj);

y\_traj = radius \* sin(theta\_traj);

%x\_traj = linspace(-10, 10, num\_points); % Variation linéaire le long de l'axe des x

%y\_traj = 3 + 3 \* sin(2\*pi\*x\_traj/10); % Variation sinusoidale le long de l'axe des y

**%CINEMATIQUE INVERSE**

% Initialiser une liste pour stocker les angles d'articulation le long de la trajectoire

trajectory\_angles = zeros(num\_points, 4);

% Boucle pour calculer les angles d'articulation le long de la trajectoire

for i = 1:num\_points

x\_desired = x\_traj(i);

y\_desired = y\_traj(i);

% Initialiser les angles d'articulation à des valeurs de départ

theta1 = 0;

theta2 = 0;

theta3 = 0;

theta4 = 0;

% Réaliser la cinématique inverse pour atteindre le point de la trajectoire actuel

% Initialiser une tolérance et un nombre maximum d'itérations

tolerance = 0.00001;

max\_iterations = 100;

% Boucle d'itération pour trouver les angles d'articulation

for j = 1:max\_iterations

% Calculer les équations directes de cinématique directe

*x = a1 \* cos(theta1) + a2 \* cos(theta1 + theta2) + a3 \* cos(theta1 + theta2 + theta3 + theta4);*

*y = a1 \* sin(theta1) + a2 \* sin(theta1 + theta2) + a3 \* sin(theta1 + theta2 + theta3 + theta4);*

% Calculer l'erreur de position

error\_x = x\_desired - x;

error\_y = y\_desired - y;

error = [error\_x; error\_y];

% Calculer les dérivées partielles

*dx\_dtheta1 = -a1 \* sin(theta1) - a2 \* sin(theta1 + theta2) - a3 \* sin(theta1 + theta2 + theta3 + theta4);*

*dx\_dtheta2 = -a2 \* sin(theta1 + theta2) - a3 \* sin(theta1 + theta2 + theta3 + theta4);*

*dx\_dtheta3 = -a3 \* sin(theta1 + theta2 + theta3 + theta4);*

*dx\_dtheta4 = -a3 \* sin(theta1 + theta2 + theta3 + theta4);*

*dy\_dtheta1 = a1 \* cos(theta1) + a2 \* cos(theta1 + theta2) + a3 \* cos(theta1 + theta2 + theta3 + theta4);*

*dy\_dtheta2 = a2 \* cos(theta1 + theta2) + a3 \* cos(theta1 + theta2 + theta3 + theta4);*

*dy\_dtheta3 = a3 \* cos(theta1 + theta2 + theta3 + theta4);*

*dy\_dtheta4 = -a3 \* sin(theta1 + theta2 + theta3 + theta4);*

*% Construire la Jacobienne*

*J = [dx\_dtheta1, dx\_dtheta2, dx\_dtheta3, dx\_dtheta4;*

*dy\_dtheta1, dy\_dtheta2, dy\_dtheta3, dy\_dtheta4];*

% Vérifier la convergence

if norm(error) < tolerance

disp('Convergence atteinte.');

break;

end

% Calculer la variation des angles d'articulation

delta\_theta = pinv(J) \* error;

% Mettre à jour les angles d'articulation

theta1 = theta1 + delta\_theta(1);

theta2 = theta2 + delta\_theta(2);

theta3 = theta3 + delta\_theta(3);

theta4 = theta4 + delta\_theta(4);

end

% Stocker les angles d'articulation dans la liste

trajectory\_angles(i, :) = [theta1, theta2, theta3, theta4];

end

Convergence atteinte.

% Tracer la trajectoire souhaitée

plot(x\_traj, y\_traj, 'k--', 'LineWidth', 1); % Trajectoire en pointillés

hold on;

**%RESULTAT, SIMULATION DU MOUVEMENT DU ROBOT**

% Parcourir la liste des angles d'articulation pour simuler le mouvement du robot le long de la trajectoire

for i = 1:num\_points

% Obtenir les angles d'articulation pour le point de la trajectoire

angles = trajectory\_angles(i, :);

**%RESULTAT**

% **A**fficher **les angles pour ce point de la trajectoire**

disp(['Angles pour le point ', num2str(i), ':']);

disp(['Theta1: ', num2str(rad2deg(angles(1))), ' degrés']);

disp(['Theta2: ', num2str(rad2deg(angles(2))), ' degrés']);

disp(['Theta3: ', num2str(rad2deg(angles(3))), ' degrés']);

disp(['Theta4: ', num2str(rad2deg(angles(4))), ' degrés']);

disp(' ');

**%VISUALISATION DU ROBOT**

% Positions des articulations

O = [0, 0]; % Position de l'origine

A = [a1\*cos(angles(1)), a1\*sin(angles(1))]; % Position de A

B = A + [a2\*cos(angles(1)+angles(2)), a2\*sin(angles(1)+angles(2))]; % Position de l'articulation B

C = B + [a3\*cos(angles(1)+angles(2)+angles(3)+angles(4)), a3\*sin(angles(1)+angles(2)+angles(3)+angles(4))]; % Position de l'articulation C (extrémité de l'effecteur)

% Tracer le robot dans sa configuration actuelle

plot([O(1), A(1)], [O(2), A(2)], 'b', 'LineWidth', 2); % Segment OA

hold on;

plot([A(1), B(1)], [A(2), B(2)], 'b', 'LineWidth', 2); % Segment AB

plot([B(1), C(1)], [B(2), C(2)], 'b', 'LineWidth', 2); % Segment BC

scatter([O(1), A(1), B(1), C(1)], [O(2), A(2), B(2), C(2)], 20, 'filled', 'DisplayName', 'Articulations');

% Tracer la position de l'effecteur

scatter(C(1), C(2), 50, 'filled', 'MarkerFaceColor', 'm', 'DisplayName', 'Effecteur');

% Ajuster les limites des axes pour un meilleur affichage

xlim([-10, 10]);

ylim([-10, 10]);

axis equal;

grid on;

% Pause pour ralentir l'animation (ajustez selon la vitesse souhaitée)

pause(0.1);

end

**Angles pour le point 1**:

Theta1: 0 degrés

Theta2: 0 degrés

Theta3: 0 degrés

Theta4: 0 degrés

…

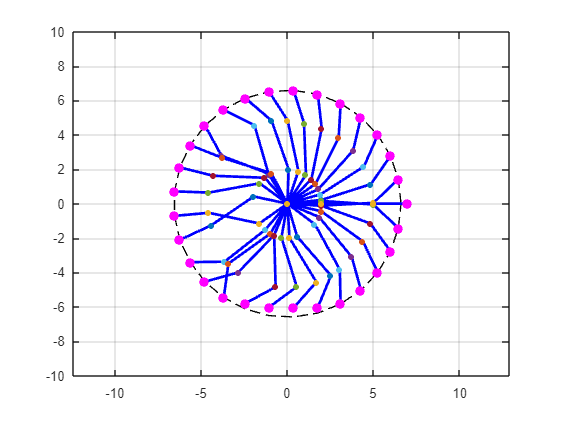
**Angles pour le point 30:**

Theta1: -8.8485e-15 degrés

Theta2: -6.3203e-15 degrés

Theta3: -2.5281e-15 degrés

Theta4: -1.1272e-31 degrés



**Améliorations possibles :**

* Intégrer directement les paramètres de DH
* Intégrer les transformations homogènes pour les calculs des coordonnées de l’effecteur
* Calcul de la jacobienne automatisé
* Utiliser autre méthode de résolution pour avec toutes les solutions possibles

**Futures itérations :**

* Utiliser autre méthode de résolution pour avec toutes les solutions possibles
* Des fois certaines trajectoire il n’arrive pas a y aller => LE DIRE ET AJUSTER CERTAINES VARIABLES DE LA TRAJ
* Intégrer volume et contrainte d’angles
* Collisions