Visualisation des différentes configurations du robot pour suivre une trajectoire

Ce code permet, après definition des variables, des longueurs des bras du robot, de la position desiree de l'effecteur, du nombre de solution souhaités affichées, et des butées de chaque articulation, de générer l'espace de travail du robot, puis de le visualiser dans les configurations possibles pour atteindre le point désiré par résolution de la cinmeatique inverse par la méthode variationnelle.

Definition des variables

```
% 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
a4 = 2; % Longueur du quatrieme bras
a5 = 2; % Longueur du cinquieme bras


theta1_min = -90;
theta1_max = 90;
theta2_min = -95;
theta2_max = 95;
theta3_min = -90;
theta3_max = 90;
theta4_min = -90;
theta4_max = 90;
theta5_min = -90;
theta5_max = 90;

% Stocker les valeurs dans la matrice butee
butee = [theta1_min, theta1_max; theta2_min, theta2_max; theta3_min, theta3_max;
theta4_min, theta4_max; theta5_min, theta5_max];
% Enregistrement de la matrice dans un fichier .mat
save('butee.mat', 'butee');
% Convertir les limites des intervalles de degrés en radians
butee_rad = deg2rad(butee);

num_configurations = 11;
```

Definition de la trajectoire

```
% Définir la trajectoire souhaitée (par exemple, un cercle)
num_points = 10;
radius = 11;
theta_traj = linspace(0, pi/3, num_points);
```

```
x_traj = radius * cos(theta_traj);
y_traj = radius * sin(theta_traj);
% Combiner les coordonnées x et y en une seule matrice de points de trajectoire
trajectory_points = [x_traj', y_traj'];
% Afficher les points de la trajectoire sous forme de tableau
disp('Points de la trajectoire :');
```

```
Points de la trajectoire :
```

```
fprintf('-------------------------------------------------\n');
```

```
-------------------------------------------------
```

```
fprintf('| Point  |      X     |      Y     |\n');
```

```
| Point  |      X     |      Y     |
```

```
fprintf('-------------------------------------------------\n');
```

```
-------------------------------------------------
```

```
for i = 1:num_points
    fprintf('|   %2d   |   %7.2f   |   %7.2f   |\n', i, x_traj(i), y_traj(i));
end
```

```
|    1   |     11.00   |      0.00   |
|    2   |     10.93   |      1.28   |
|    3   |     10.70   |      2.54   |
|    4   |     10.34   |      3.76   |
|    5   |      9.83   |      4.94   |
|    6   |      9.19   |      6.04   |
|    7   |      8.43   |      7.07   |
|    8   |      7.55   |      8.00   |
|    9   |      6.57   |      8.82   |
|   10   |      5.50   |      9.53   |
```

```
fprintf('-------------------------------------------------\n')
```

```
-------------------------------------------------
```

```


% Afficher la trajectoire
figure;
plot(x_traj, y_traj, 'r-', 'LineWidth', 1.5, 'DisplayName', 'Trajectoire à suivre'
);
axis equal; % Pour que l'échelle des axes soit égale
xlabel('X');
ylabel('Y');
title('Trajectoire');
grid on;
hold on;
```

Affichage de l'espace de travail :

- liste pour **stocker les coordonnees atteignables,**
- definition de l'**echantillon d'angles testés et du pas**

```
% Initialiser une liste pour stocker les coordonnées atteignables
reachable_points = [];

% % Définir la grille d'angles d'articulation
theta1_range = linspace(butee_rad(1, 1), butee_rad(1, 2), 6);
theta2_range = linspace(butee_rad(2, 1), butee_rad(2, 2), 6);
theta3_range = linspace(butee_rad(3, 1), butee_rad(3, 2), 6);
theta4_range = linspace(butee_rad(4, 1), butee_rad(4, 2), 6);
theta5_range = linspace(butee_rad(5, 1), butee_rad(5, 2), 6);
% % Nombre de pas pour chaque articulation
% step_angle = 15; % Espacement entre chaque angle en degrés
```

- **calcul des coordonnees atteignables** pour chaque combinaison d'angles possibles de l'echantillon, par **cinematique directe**

```
% Calculer les coordonnées atteignables pour chaque combinaison d'angles
for i = 1:length(theta1_range)
    for j = 1:length(theta2_range)
        for k = 1:length(theta3_range)
            for l = 1:length(theta4_range)
                for m = 1:length(theta5_range)
                    % Calculer les coordonnées de l'effecteur
                    x = a1 * cos(theta1_range(i)) + a2 * cos(theta1_range(i) +
theta2_range(j)) + a3 * cos(theta1_range(i) + theta2_range(j) + theta3_range(k))
+ a4 * cos(theta1_range(i) + theta2_range(j) + theta3_range(k) + theta4_range(l))
+ a5 * cos(theta1_range(i) + theta2_range(j) + theta3_range(k) + theta4_range(l) +
theta5_range(m));
                    y = a1 * sin(theta1_range(i)) + a2 * sin(theta1_range(i) +
theta2_range(j)) + a3 * sin(theta1_range(i) + theta2_range(j) + theta3_range(k))
+ a4 * sin(theta1_range(i) + theta2_range(j) + theta3_range(k) + theta4_range(l))
+ a5 * sin(theta1_range(i) + theta2_range(j) + theta3_range(k) + theta4_range(l) +
theta5_range(m));
                     % Ajouter les coordonnées à la liste
                    reachable_points = [reachable_points; x, y];
                end
            end
        end
    end
end

points_atteign = [reachable_points(:, 1), reachable_points(:, 2)];
```

- affichage des **contours de l'espace de travail**

```
% % Calculer les contours convexes du nuage de points avec une méthode plus précise
K = convhull(points_atteign);

% % Tracer les contours de l'espace de travail
plot(points_atteign(K, 1), points_atteign(K, 2), 'b', 'LineWidth', .5,
'DisplayName', 'Contours de l''espace de travail');
fill(points_atteign(K, 1), points_atteign(K, 2), 'b', 'FaceAlpha', 0.1,
'EdgeColor', 'none', 'DisplayName', 'Espace de travail');

% Mettre à jour le graphique
title('Position du robot' );
xlabel('X');
ylabel('Y');
axis equal; % Assure que les proportions X/Y sont égales
grid on;
legend show;
legend("Position", [0.58731,0.15829,0.38571,0.11905])

hold on;
```

Resolution IK après vérification de l'appartenance du point cible à l'espace de travail (revoir la verif)

Cette partie procède tout d'abord à une verification de l'appartenance du point désiré de l'effecteur à l'esapce de travail du robot. Si cela est vérifé on lance alors le calcul de cinématique inverse par la méthode variationnelle.

Puis on affiche les solutions , d'abord en raidan et ensuite en degrès.

```
% Initialiser une matrice pour stocker les résultats finaux
final_results = [];

% Paramètres de l'algorithme d'ajustement du rayon
max_iterations = 10;
tolerance = 1;
adjusted_radius = zeros(1, num_points); % Vecteur pour stocker les rayons actualisés

% Itérer sur chaque point de la trajectoire
for i = 1:num_points
    % Initialiser le rayon avec le rayon d'origine
    current_radius = radius;
    % Initialiser le nombre d'itérations
    iterations = 0;

    % Itération jusqu'à ce que le point soit à l'intérieur de l'espace de travail
ou que le nombre maximal d'itérations soit atteint
    while iterations < max_iterations
        % Calculer les coordonnées du point de la trajectoire avec le rayon actuel
        x_desired = current_radius * cos(theta_traj(i));
```

```
        y_desired = current_radius * sin(theta_traj(i));

        % Vérifier si le point désiré se trouve dans l'espace de travail
        if point_in_workspace(x_desired, y_desired, points_atteign, K)
            % Si le point est à l'intérieur de l'espace de travail, sortir de la
boucle
            break;
        else
            % Si le point est à l'extérieur de l'espace de travail, ajuster le rayon
            current_radius = current_radius - tolerance;
            iterations = iterations + 1;
        end
    end

    % Enregistrer le rayon actualisé dans le vecteur
    adjusted_radius(i) = current_radius;

    % Si le nombre maximal d'itérations est atteint sans trouver une solution
satisfaisante, afficher un avertissement
    if iterations == max_iterations
        warning('Impossible d''ajuster le rayon pour que le point de la trajectoire
%d soit à l''intérieur de l''espace de travail.', i);
    end

    % Appeler la fonction de cinématique inverse pour trouver les solutions pour ce
point
    num_configurations = 10; % Vous pouvez ajuster le nombre de configurations
souhaitées ici
    all_solutions = find_inverse_kinematics(a1, a2, a3, a4, a5, x_desired,
y_desired, num_configurations);

    % Ajouter une colonne pour le numéro du point de trajectoire
    point_numbers = repmat(i, size(all_solutions, 1), 1);
    all_solutions_with_point_numbers = [point_numbers, all_solutions];

    % Ajouter les résultats à la matrice finale
    final_results = [final_results; all_solutions_with_point_numbers];
end

% Afficher le rayon actualisé
fprintf('Rayon actualisé pour la trajectoire :\n');
```

```
Rayon actualisé pour la trajectoire :
```

```
fprintf('%s\n', num2str(adjusted_radius));
```

```
10  10  10  10  10  10  10  10  10  10
```

```

% Afficher le tableau final
```

```
fprintf('------------------------------------------------------------------------
--\n');
```

```
------------------------------------------------------------------------
```

```
fprintf('| Point  | Theta1   | Theta2   | Theta3   | Theta4   | Theta5   |
X_calculé | Y_calculé |\n');
```

```
| Point  | Theta1   | Theta2   | Theta3   | Theta4   | Theta5   | X_calculé | Y_calculé |
```

```
fprintf('------------------------------------------------------------------------
--\n');
```

```
------------------------------------------------------------------------
```

```
for row = 1:size(final_results, 1)
    fprintf('|   %2d   | %8.2f | %8.2f | %8.2f | %8.2f | %8.2f | %9.2f | %9.2f
|\n', final_results(row, :));
end
```

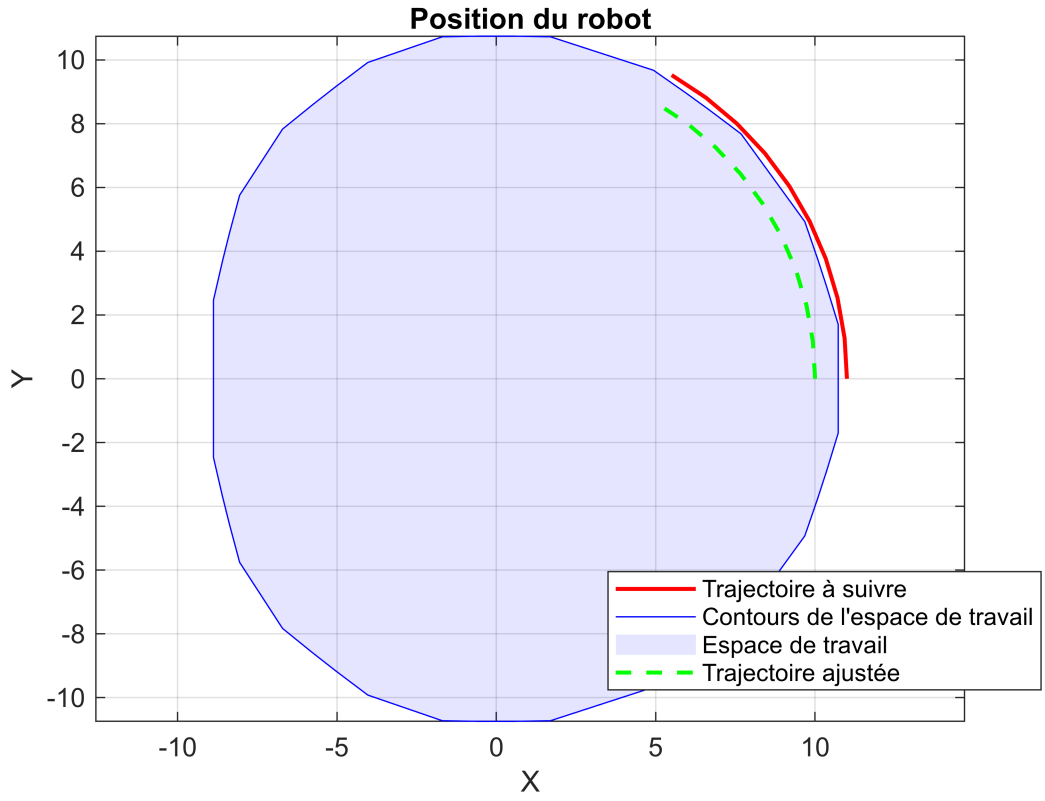
```
|    1   |     6.14 |     6.10 |     0.95 |     5.20 |     0.95 |     10.00 |     -0.00 |
|    1   |     5.49 |     0.75 |     0.34 |     5.95 |     0.61 |     10.00 |     -0.00 |
|    1   |     5.50 |     1.11 |     5.86 |     6.24 |     0.63 |     10.00 |     -0.00 |
|    1   |     5.94 |     0.25 |     0.53 |     5.28 |     1.19 |     10.00 |     -0.00 |
|    1   |     5.73 |     0.79 |     6.05 |     5.83 |     1.11 |     10.00 |     -0.00 |
|    1   |     0.68 |     5.25 |     0.32 |     5.85 |     0.86 |     10.00 |      0.00 |
|    1   |     5.52 |     1.12 |     5.70 |     0.22 |     0.47 |     10.00 |      0.00 |
|    1   |     6.10 |     6.09 |     0.94 |     5.38 |     0.93 |     10.00 |      0.00 |
|    1   |     5.68 |     0.82 |     6.04 |     5.95 |     1.04 |     10.00 |      0.00 |
|    1   |     5.52 |     0.80 |     0.15 |     5.95 |     0.78 |     10.00 |     -0.00 |
|    2   |     6.26 |     0.58 |     6.17 |     5.85 |     5.61 |      9.93 |      1.16 |
|    2   |     5.70 |     0.95 |     6.26 |     5.63 |     0.88 |      9.93 |      1.16 |
|    2   |     6.19 |     0.69 |     6.18 |     5.71 |     5.80 |      9.93 |      1.16 |
|    2   |     5.75 |     0.90 |     6.13 |     5.77 |     0.99 |      9.93 |      1.16 |
|    2   |     6.08 |     0.89 |     5.29 |     0.03 |     0.70 |      9.93 |      1.16 |
|    2   |     0.11 |     5.92 |     1.00 |     5.16 |     0.91 |      9.93 |      1.16 |
|    2   |     0.12 |     5.87 |     0.85 |     5.40 |     1.06 |      9.93 |      1.16 |
|    2   |     5.76 |     0.93 |     5.49 |     0.91 |     6.14 |      9.93 |      1.16 |
|    2   |     5.80 |     0.76 |     0.12 |     5.52 |     1.05 |      9.93 |      1.16 |
|    2   |     5.75 |     1.09 |     5.58 |     0.04 |     0.65 |      9.93 |      1.16 |
|    3   |     5.90 |     0.56 |     0.64 |     5.38 |     0.77 |      9.73 |      2.31 |
|    3   |     6.13 |     0.37 |     0.63 |     5.08 |     0.98 |      9.73 |      2.31 |
|    3   |     6.19 |     0.70 |     5.92 |     5.54 |     1.15 |      9.73 |      2.31 |
|    3   |     5.95 |     0.16 |     0.76 |     0.01 |     0.07 |      9.73 |      2.31 |
|    3   |     6.04 |     0.50 |     0.42 |     5.30 |     1.10 |      9.73 |      2.31 |
|    3   |     5.88 |     0.92 |     5.67 |     0.25 |     0.70 |      9.73 |      2.31 |
|    3   |     0.18 |     0.36 |     6.10 |     5.24 |     1.17 |      9.73 |      2.31 |
|    3   |     5.88 |     0.92 |     5.78 |     0.02 |     0.85 |      9.73 |      2.31 |
|    3   |     6.08 |     0.71 |     6.11 |     5.54 |     1.19 |      9.73 |      2.31 |
|    3   |     6.12 |     0.91 |     6.08 |     5.85 |     5.81 |      9.73 |      2.31 |
|    4   |     6.06 |     0.43 |     0.67 |     5.49 |     0.84 |      9.40 |      3.42 |
|    4   |     6.26 |     0.73 |     0.28 |     5.24 |     0.05 |      9.40 |      3.42 |
|    4   |     6.24 |     0.36 |     0.91 |     5.05 |     0.39 |      9.40 |      3.42 |
|    4   |     1.03 |     5.61 |     0.29 |     5.58 |     6.13 |      9.40 |      3.42 |
|    4   |     5.96 |     0.61 |     0.46 |     5.64 |     0.84 |      9.40 |      3.42 |
|    4   |     6.03 |     0.76 |     5.89 |     0.12 |     0.87 |      9.40 |      3.42 |
|    4   |     5.88 |     0.94 |     5.75 |     0.66 |     0.12 |      9.40 |      3.42 |
|    4   |     0.40 |     0.29 |     0.15 |     5.41 |     5.98 |      9.40 |      3.42 |
|    4   |     5.89 |     0.72 |     6.24 |     0.22 |     0.53 |      9.40 |      3.42 |
|    4   |     6.13 |     0.72 |     5.90 |     6.10 |     1.10 |      9.40 |      3.42 |
|    5   |     0.22 |     0.66 |     5.41 |     0.02 |     0.98 |      8.94 |      4.49 |
```

5	6.08	0.46	0.25	0.24	0.38	8.94	4.49
5	0.32	0.66	5.40	6.05	0.93	8.94	4.49
5	0.84	0.07	5.58	0.17	5.65	8.94	4.49
5	6.25	0.68	5.91	6.16	1.09	8.94	4.49
5	0.28	0.60	5.33	1.05	5.36	8.94	4.49
5	0.60	6.18	0.66	5.46	5.69	8.94	4.49
5	5.96	0.67	0.38	6.16	0.43	8.94	4.49
5	0.99	5.55	0.82	5.34	6.17	8.94	4.49
5	0.86	6.23	6.11	5.69	6.02	8.94	4.49
6	6.05	1.03	6.03	6.26	0.62	8.35	5.50
6	0.29	0.79	5.06	0.77	0.15	8.35	5.50
6	6.13	0.54	0.46	6.16	0.46	8.35	5.50
6	0.12	0.04	0.64	0.04	0.40	8.35	5.50
6	0.04	0.74	5.56	0.65	0.48	8.35	5.50
6	6.28	0.58	0.60	5.28	0.80	8.35	5.50
6	0.31	0.77	5.54	5.92	0.99	8.35	5.50
6	0.16	0.68	0.24	5.97	5.42	8.35	5.50
6	0.00	0.74	0.05	5.60	1.10	8.35	5.50
6	0.49	0.52	6.23	5.69	5.75	8.35	5.50
7	0.15	0.69	0.44	5.97	5.49	7.66	6.43
7	0.43	0.10	1.11	5.09	0.20	7.66	6.43
7	1.20	5.21	1.08	5.80	6.11	7.66	6.43
7	1.11	5.71	0.56	6.04	5.32	7.66	6.43
7	1.11	5.89	5.95	5.93	1.22	7.66	6.43
7	0.25	0.07	1.13	5.60	0.13	7.66	6.43
7	0.86	5.46	0.71	0.51	5.91	7.66	6.43
7	1.05	6.26	5.12	0.95	5.93	7.66	6.43
7	0.87	0.32	5.71	6.27	5.59	7.66	6.43
7	0.04	0.64	0.27	5.78	0.94	7.66	6.43
8	1.32	5.92	5.97	5.63	1.04	6.86	7.27
8	1.36	5.97	5.28	0.53	0.30	6.86	7.27
8	0.51	0.51	0.37	5.86	5.41	6.86	7.27
8	0.96	0.45	5.06	0.38	0.08	6.86	7.27
8	1.59	5.20	0.25	0.33	5.56	6.86	7.27
8	0.68	0.08	0.88	5.48	5.68	6.86	7.27
8	0.38	0.43	0.03	5.88	1.22	6.86	7.27
8	0.35	0.35	0.88	5.15	0.65	6.86	7.27
8	0.60	6.17	0.15	0.25	0.83	6.86	7.27
8	0.63	0.38	0.58	5.06	0.07	6.86	7.27
9	0.50	0.83	5.49	0.07	0.90	5.97	8.02
9	1.48	5.10	0.99	5.98	6.19	5.97	8.02
9	0.38	0.87	5.72	6.24	0.95	5.97	8.02
9	1.21	0.08	5.50	6.01	0.99	5.97	8.02
9	0.72	0.05	0.77	6.14	5.22	5.97	8.02
9	0.58	0.96	5.49	5.94	0.68	5.97	8.02
9	0.67	0.18	0.85	4.96	0.81	5.97	8.02
9	0.48	6.24	0.86	6.28	0.05	5.97	8.02
9	0.51	0.78	6.20	6.24	5.32	5.97	8.02
9	0.48	0.83	5.75	0.71	5.21	5.97	8.02
10	1.34	0.06	5.65	5.75	0.99	5.00	8.66
10	1.12	5.51	0.91	0.13	0.02	5.00	8.66
10	1.59	5.50	6.22	6.15	1.04	5.00	8.66
10	0.74	0.19	0.67	6.24	5.19	5.00	8.66
10	0.82	0.11	0.63	0.01	5.12	5.00	8.66
10	1.01	0.17	0.40	5.85	5.33	5.00	8.66
10	0.98	0.63	5.54	5.75	0.82	5.00	8.66
10	1.03	5.96	0.64	0.44	5.08	5.00	8.66
10	1.54	5.82	0.04	5.34	1.07	5.00	8.66
10	1.59	5.39	0.73	5.30	0.78	5.00	8.66

```
fprintf('-------------------------------------------------------------------------
--\n');
```

```
% Afficher la trajectoire avec le rayon ajusté
x_traj_adjusted = adjusted_radius .* cos(theta_traj);
y_traj_adjusted = adjusted_radius .* sin(theta_traj);

% Tracer la nouvelle trajectoire avec le rayon ajusté sur la même figure
hold on;
plot(x_traj_adjusted, y_traj_adjusted, 'g--', 'LineWidth', 1.5, 'DisplayName',
'Trajectoire ajustée');
```




```
% Mettre à jour le graphique
legend show;
```

```
function in_workspace = point_in_workspace(x, y, points_atteign, K)
    % Vérifie si le point (x, y) est dans l'espace de travail du robot
    in_workspace = inpolygon(x, y, points_atteign(K, 1), points_atteign(K, 2));
end
```

```
function [all_solutions] = find_inverse_kinematics(a1, a2, a3, a4, a5, x_desired,
y_desired, num_configurations)
```

```
    % Définir les angles d'articulation
    syms theta1 theta2 theta3 theta4 theta5;

    % Définir une tolérance et un nombre maximum d'itérations
    tolerance = 0.0000001;
    max_iterations = 100;

    % Initialiser une liste pour stocker les configurations atteignables
    all_solutions = [];
    point_number = [];
    % Recherche des configurations
    while size(all_solutions, 1) < num_configurations

        % Initialiser les angles d'articulation en fonction des butées
        % Générer des angles d'articulation aléatoires entre 0 et 2*pi radians
        theta1 = rand * 2*pi;
        theta2 = rand * 2*pi;
        theta3 = rand * 2*pi;
        theta4 = rand * 2*pi;
        theta5 = rand * 2*pi;

        % Boucle d'itération pour trouver les angles d'articulation
        for i = 1:max_iterations
            % Calculer les équations directes de cinématique directe
            x = a1*cos(theta1) + a3*cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - a3*sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + a2*cos(theta1)*cos(theta2) - a2*sin(theta1)*sin(theta2)
+ a5*cos(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) - sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)))) +
a4*cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2))
- sin(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1)))
- a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))) + sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)))) - a4*sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)));
            y = a1*sin(theta1) + a3*cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + a3*sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) + a2*cos(theta1)*sin(theta2) + a2*cos(theta2)*sin(theta1)
+ a5*cos(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))) + sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)))) +
a4*cos(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1))
```

```
+ sin(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2)))
+ a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) - sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)))) + a4*sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)));

            % 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) - a3*cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - a3*sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) - a2*sin(theta1)*sin(theta2) - a2*cos(theta1)*cos(theta2)
- a4*sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2))
- sin(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1)))
- a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) - sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))));
            dx_dtheta2 = -a3*cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - a3*sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + a2*cos(theta1)*cos(theta2) - a2*sin(theta1)*sin(theta2)
- a4*sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2))
- sin(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1)))
- a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) - sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))));
            dx_dtheta3 = a3*sin(theta3)*(sin(theta1)*sin(theta2) -
cos(theta1)*cos(theta2)) + a4*sin(theta4)*(sin(theta1)*sin(theta2) -
cos(theta1)*cos(theta2)) + a5*sin(theta5)*(sin(theta1)*sin(theta2) -
cos(theta1)*cos(theta2));
            dx_dtheta4 = -a4*sin(theta4)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - a5*sin(theta5)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2));
            dx_dtheta5 =
-a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1))) -
sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1))
+ sin(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2)))
+ a5*cos(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
```

```
sin(theta1)*sin(theta2))) - sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)))));

            dy_dtheta1 = a1*cos(theta1) + a3*cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + a3*sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) + a2*cos(theta2)*sin(theta1) + a2*cos(theta1)*sin(theta2)
+ a4*cos(theta4)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2))
+ a5*cos(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))) + sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)))) +
a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) - sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))));
            dy_dtheta2 = a3*cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + a3*sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - a2*sin(theta1)*sin(theta2) - a2*cos(theta1)*cos(theta2)
- a4*sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2))
- sin(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1)))
- a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) - sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2))));
            dy_dtheta3 = - a3*sin(theta3)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + a4*sin(theta4)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) + a5*sin(theta5)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1));
            dy_dtheta4 = - a4*sin(theta4)*(cos(theta1)*sin(theta2)
+ cos(theta2)*sin(theta1)) - a5*sin(theta5)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1));
            dy_dtheta5 =
a5*cos(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)) + sin(theta3)*(cos(theta1)*cos(theta2)
- sin(theta1)*sin(theta2))) -
sin(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2))
- sin(theta3)*(cos(theta1)*sin(theta2) + cos(theta2)*sin(theta1)))
- a5*sin(theta5)*(cos(theta4)*(cos(theta3)*(cos(theta1)*cos(theta2) -
sin(theta1)*sin(theta2)) - sin(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1))) + sin(theta4)*(cos(theta3)*(cos(theta1)*sin(theta2) +
cos(theta2)*sin(theta1)))));

            % Construire la Jacobienne
            J = [dx_dtheta1, dx_dtheta2, dx_dtheta3, dx_dtheta4, dx_dtheta5;
             dy_dtheta1, dy_dtheta2, dy_dtheta3, dy_dtheta4, dy_dtheta5];
```

```
            % Vérifier la convergence
            if norm(error) < tolerance
                %disp('Convergence atteinte.');
                % Stocker la solution
                all_solutions = [all_solutions; [theta1, theta2, theta3, theta4,
theta5, x, y]];

                break;
            end

            % Calculer la variation des angles d'articulation
            delta_theta = pinv(J) * error;

            % Mise a jour des les angles d'articulation
            theta1 = theta1 + delta_theta(1);
            theta2 = theta2 + delta_theta(2);
            theta3 = theta3 + delta_theta(3);
            theta4 = theta4 + delta_theta(4);
            theta5 = theta5 + delta_theta(5);

            % Maintien des angles entre 0 et 2*pi
            theta1 = mod(theta1, 2*pi);
            theta2 = mod(theta2, 2*pi);
            theta3 = mod(theta3, 2*pi);
            theta4 = mod(theta4, 2*pi);
            theta5 = mod(theta5, 2*pi);
            % Incrémenter le numéro du point de la trajectoire
            point_number = point_number + 1;
        end

    end
end
```