**Résolution IK Variationnelle pour atteindre un point selon de nombreuses configurations**

Le code ci-dessous permet par la méthode de la pseudo inverse de la Jacobienne de résoudre les équations de cinématique inverse afin de savoir quel angle doit adopter le robot à chaque articulation pour que son effecteur atteigne la position désirée. Il ne s’arrête pas à la première convergence il continue selon le nombre de configurations que l’on souhaite. Toutefois en amont, il vérifie que le point atteignable soit bien dans l’espace de travail (bien que cela ne soit pas la seule condition requise). Une fois qu’il a les solutions des angles ce code permet de visualiser le robot de telle manière à ce qu’il aille atteindre le point cible. L’espace de travail est également visualisable via un nuage de points. Puis, il trace la surface tridimensionnelle des coordonnées de l'effecteur en fonction des deux premiers angles d'articulation et il définit aussi une fonction pour visualiser les coordonnées de l'effecteur dans le plan XY en fonction des angles d'articulation

**%PARAMETRISATION**

% 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 position désirée de l'effecteur

x\_desired = 1;

y\_desired = 6;

**%ESPACE DE TRAVAIL**

% Initialiser une liste pour stocker les coordonnées atteignables

reachable\_points = [];

% Définir la grille d'angles d'articulation

theta1\_range = linspace(-pi, pi, 10);

theta2\_range = linspace(-pi, pi, 10);

theta3\_range = linspace(-pi, pi, 10);

theta4\_range = linspace(-pi, pi, 10);

% 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)

% 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) + theta4\_range(l));

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) + theta4\_range(l));

% Ajouter les coordonnées à la liste

reachable\_points = [reachable\_points; x, y];

end

end

end

end

min\_x = min(reachable\_points(:, 1));

max\_x = max(reachable\_points(:, 1));

min\_y = min(reachable\_points(:, 2));

max\_y = max(reachable\_points(:, 2));

% Vérifier si le point désiré est dans l'espace de travail

if x\_desired < min\_x || x\_desired > max\_x || y\_desired < min\_y || y\_desired > max\_y

disp('Le point désiré est en dehors de l''espace de travail du robot.');

return; % Sortir du programme

else

disp('Le point désiré est dans l''espace de travail du robot. Continuer le calcul des angles d''articulation.');

% Ajoutez votre code pour continuer le calcul des angles d'articulation ici

end

Le point désiré est dans l'espace de travail du robot. Continuer le calcul des angles d'articulation.

% Initialiser une figure

figure;

hold on;

% Tracer le nuage de points représentant l'espace de travail atteignable

scatter(reachable\_points(:, 1), reachable\_points(:, 2), 1, 'filled');

% Tracer le point cible en rouge

plot(x\_desired, y\_desired, 'ro', 'MarkerSize', 10, 'DisplayName', 'Point désiré');

**%RECHERCHE DES SOLUTIONS**

% Initialiser une liste pour stocker les configurations atteignables

all\_solutions = [];

% Définir une fonction pour la cinématique inverse

**inverse\_kinematics = @(x\_desired, y\_desired) find\_inverse\_kinematics(a1, a2, a3, x\_desired, y\_desired);**

% Nombre de configurations à rechercher

num\_configurations = 40; % Modifier selon vos besoins

% Recherche des configurations

for i = 1:num\_configurations

% Utiliser la fonction de cinématique inverse pour trouver une solution

[theta1, theta2, theta3, theta4] = inverse\_kinematics(x\_desired, y\_desired);

**%VERIFICATION**

% Calculer les coordonnées de l'effecteur pour cette configuration

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

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

% Stocker la solution et les coordonnées de l'effecteur

all\_solutions = [all\_solutions; [theta1, theta2, theta3, theta4, x\_effecteur, y\_effecteur]];

end

Convergence atteinte.

**%RESULTAT NUM**

% Afficher toutes les configurations trouvées

disp('Configurations possibles du robot pour la position désirée :');

Configurations possibles du robot pour la position désirée

disp(' Theta1 Theta2 Theta3 Theta4 X\_calculated Y\_calculated');

Theta1 Theta2 Theta3 Theta4 X\_calculated Y\_calculated

disp(all\_solutions);

1.9065 5.9488 4.7179 6.8657 1.0000 6.0000

7.6450 0.5566 -3.1480 1.8670 1.0000 5.9674

1.3642 0.5526 8.3611 -3.3570 1.0000 5.9703

2.0454 5.6931 5.7835 -0.2863 1.0000 6.0000

1.4999 5.6610 1.6196 5.9142 1.0000 6.0000

1.8989 5.9623 8.2291 3.3458 1.0000 6.0000

7.6455 0.5558 11.6019 -0.3162 1.0000 5.9678

1.9697 5.8348 6.5483 11.3990 1.0000 6.0000

7.6450 0.5567 9.7360 -4.7338 1.0000 5.9673

0.8754 0.3870 3.9231 3.3071 1.0000 6.0000

7.6424 0.5610 11.2043 6.3625 1.0000 5.9645

1.6895 6.3201 1.5479 3.5617 1.0000 6.0000

7.6443 0.5579 14.7996 -9.7982 1.0000 5.9661

2.1760 5.4202 3.6802 2.1041 1.0000 6.0000

-4.2633 12.0249 6.4225 11.5991 1.0000 6.0000

2.0228 5.7362 12.1883 -0.4453 1.0000 6.0000

1.3601 0.5596 3.0048 8.2791 1.0000 5.9650

1.4079 0.4830 6.5769 4.7254 1.0000 5.9944

7.6451 0.5565 9.9294 1.3560 1.0000 5.9673

1.6183 6.4367 -8.7281 13.7996 1.0000 6.0000

2.1765 5.0232 4.7076 2.0728 1.0000 6.0000

1.7251 5.3686 1.1275 0.0227 1.0000 6.0000

2.0393 5.7048 4.3302 1.1568 1.0000 6.0000

2.1227 5.0345 1.3606 -0.7256 1.0000 6.0000

1.5692 6.5158 0.4017 4.6499 1.0000 6.0000

1.3638 0.5533 10.3736 0.9134 1.0000 5.9699

7.6448 0.5570 17.7833 -12.7814 1.0000 5.9668

1.4833 5.6846 -2.3050 3.5586 1.0000 6.0000

2.0303 5.7221 4.0482 1.4237 1.0000 6.0000

1.5502 6.5460 8.5053 -3.4599 1.0000 6.0000

7.6458 0.5552 9.5421 -4.5393 1.0000 5.9683

7.6428 0.5606 8.7093 -3.7090 1.0000 5.9643

1.9786 5.8186 1.5652 10.1113 1.0000 6.0000

2.2425 5.0538 1.2358 -1.0129 1.0000 6.0000

2.0197 5.7420 1.8896 -2.7178 1.0000 6.0000

1.3629 0.5548 7.5849 -2.5819 1.0000 5.9686

7.6442 -5.7251 7.1136 -2.1123 1.0000 5.9660

7.0996 0.4947 0.4621 0.4041 1.0000 6.0000

2.0886 5.6083 3.2667 2.3098 1.0000 6.0000

13.9243 -5.7196 22.5067 -4.9419 1.0000 5.9614

**% VISUALISATION DU ROBOT**

% Définir les couleurs pour chaque type de segment

color\_segment\_OA = 'b'; % Segment OA

color\_segment\_AB = 'r'; % Segment AB

color\_segment\_BC = 'g'; % Segment BC

% Visualisation des configurations du robot

for i = 1:size(all\_solutions, 1)

% Extraire les angles d'articulation pour cette configuration

theta1 = all\_solutions(i, 1);

theta2 = all\_solutions(i, 2);

theta3 = all\_solutions(i, 3);

theta4 = all\_solutions(i, 4);

% Calculer les positions des articulations

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

A = [a1 \* cos(theta1), a1 \* sin(theta1)]; % Position de l'articulation A

B = A + [a2 \* cos(theta1 + theta2), a2 \* sin(theta1 + theta2)]; % Position de l'articulation B

C = B + [a3 \* cos(theta1 + theta2 + theta3 + theta4), a3 \* sin(theta1 + theta2 + theta3 + theta4)]; % Position de l'articulation C (effecteur)

% Tracer le robot pour cette configuration

plot([O(1), A(1)], [O(2), A(2)], 'LineWidth', 2, 'Color', color\_segment\_OA, 'DisplayName', ['Segment OA - Solution ', num2str(i)]);

plot([A(1), B(1)], [A(2), B(2)], 'LineWidth', 2, 'Color', color\_segment\_AB, 'DisplayName', ['Segment AB - Solution ', num2str(i)]);

plot([B(1), C(1)], [B(2), C(2)], 'LineWidth', 2, 'Color', color\_segment\_BC, 'DisplayName', ['Segment BC - Solution ', num2str(i)]);

scatter([O(1), A(1), B(1), C(1)], [O(2), A(2), B(2), C(2)], 20, 'filled', 'DisplayName', ['Articulations - Solution ', num2str(i)]);

end

% Calculer les limites des axes

xlim([min(reachable\_points(:, 1)), max(reachable\_points(:, 1))]);

ylim([min(reachable\_points(:, 2)), max(reachable\_points(:, 2))]);

% Configurer le titre et les étiquettes des axes

title('Espace de travail et configurations du robot');

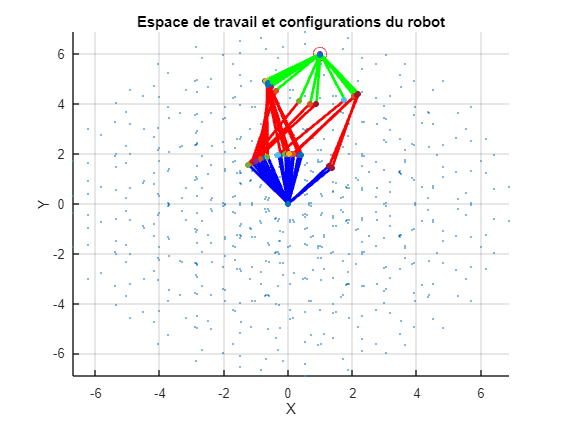
xlabel('X');

ylabel('Y');

% Affichage de la grille

grid on;

hold off;



**% Fonction pour la cinématique inverse**

function **[theta1, theta2, theta3, theta4] = find\_inverse\_kinematics(a1, a2, a3, x\_desired, y\_desired)**

% Définir les angles d'articulation

syms theta1 theta2 theta3 theta4;

% Définir une tolérance et un nombre maximum d'itérations

tolerance = 0.00001;

max\_iterations = 100;

% Initialiser les angles d'articulation

theta1 = rand \* 2 \* pi; % Initialiser à des valeurs aléatoires

theta2 = rand \* 2 \* pi;

theta3 = rand \* 2 \* pi;

theta4 = 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) + 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

end

**BONUS**

*% Calculez les coordonnées de l'effecteur final pour chaque combinaison d'angles*

*X\_coordinates = zeros(length(theta1\_range), length(theta2\_range));*

*Y\_coordinates = zeros(length(theta1\_range), length(theta2\_range));*

*for i = 1:length(theta1\_range)*

*for j = 1:length(theta2\_range)*

*% Calculez les coordonnées de l'effecteur final en fonction des angles articulaires*

*X\_coordinates(i, j) = a1 \* cos(theta1\_range(i)) + a2 \* cos(theta1\_range(i) + theta2\_range(j)) + a3 \* cos(theta1\_range(i) + theta2\_range(j) + theta3 + theta4);*

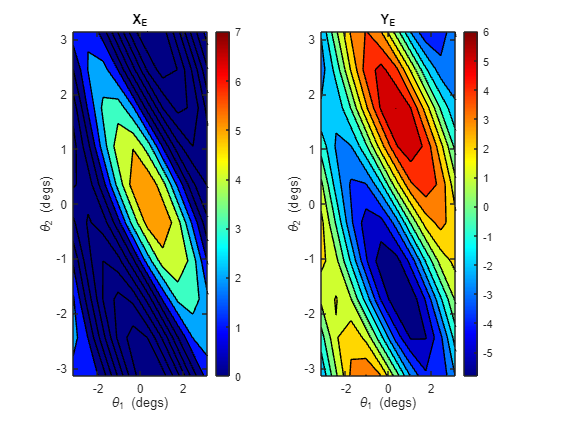
*Y\_coordinates(i, j) = a1 \* sin(theta1\_range(i)) + a2 \* sin(theta1\_range(i) + theta2\_range(j)) + a3 \* sin(theta1\_range(i) + theta2\_range(j) + theta3 + theta4);*

*end*

*end*

*% Visualisez les coordonnées de l'effecteur final*

*plot\_XY\_given\_theta\_2dof(theta1\_range, theta2\_range, X\_coordinates, Y\_coordinates, (a1 + a2 + a3));*



*% Créer une nouvelle figure pour la visualisation tridimensionnelle*

*figure;*

*% Tracer la surface tridimensionnelle des coordonnées X en fonction de θ1 et θ2*

*subplot(1, 2, 1);*

*surf(theta1\_mesh, theta2\_mesh, X\_coordinates);*

*xlabel('\theta\_1');*

*ylabel('\theta\_2');*

*zlabel('X');*

*title('Coordonnées X en fonction de \theta\_1 et \theta\_2');*

*% Tracer la surface tridimensionnelle des coordonnées Y en fonction de θ1 et θ2*

*subplot(1, 2, 2);*

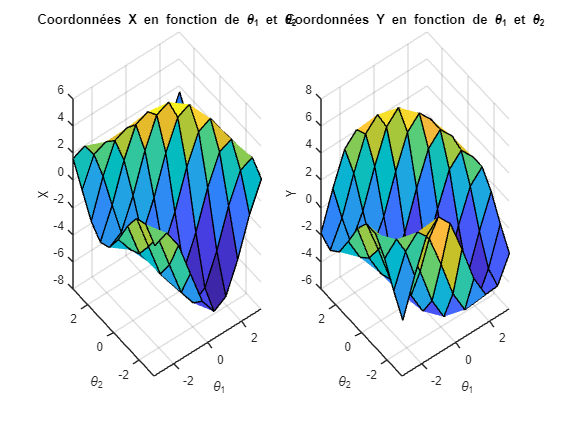
*surf(theta1\_mesh, theta2\_mesh, Y\_coordinates);*

*xlabel('\theta\_1');*

*ylabel('\theta\_2');*

*zlabel('Y');*

*title('Coordonnées Y en fonction de \theta\_1 et \theta\_2');*



function **plot\_XY\_given\_theta\_2dof**(theta\_1\_mat\_degs,theta\_2\_mat\_degs,...

X\_mat,Y\_mat,a\_cmax)

xlab\_str = '\theta\_1 (degs)';

ylab\_str = '\theta\_2 (degs)';

figure;

hax(1) = subplot(1,2,1);

contourf(theta\_1\_mat\_degs, theta\_2\_mat\_degs, X\_mat);

clim(hax(1), [0 a\_cmax]);

colormap(gca,'jet'); colorbar

xlabel(xlab\_str, 'Interpreter', 'tex');

ylabel(ylab\_str, 'Interpreter', 'tex');

title(hax(1), 'X\_E', 'Interpreter', 'tex')

hax(2) = subplot(1,2,2);

contourf(theta\_1\_mat\_degs, theta\_2\_mat\_degs, Y\_mat);

clim(hax(1), [0 a\_cmax]);

colormap(gca,'jet'); colorbar

xlabel(xlab\_str, 'Interpreter', 'tex');

ylabel(ylab\_str, 'Interpreter', 'tex');

title(hax(2), 'Y\_E', 'Interpreter', 'tex')

end