



# MECANIQUE POUR LA ROBOTIQUE

TP sur Matlab



THOUKAM THOTCHUM YVES  
NONO TALLA FORTUNE ALVAREZ

June 5, 2024

# Contents

<b>Contents</b>	<b>1</b>
<b>1 TP I : Initiation à la Robotique</b>	<b>3</b>
1.1 Objectifs . . . . .	3
1.2 Volume Horaire . . . . .	3
1.3 Introduction . . . . .	3
1.4 Exercice 1 : Matrices . . . . .	3
1.5 Exercice 2 . . . . .	4
1.6 Exercice 3 . . . . .	4
1.7 Exercice 4 : Graphique 2D . . . . .	6
<b>2 TP II : Robotique</b>	<b>11</b>
2.1 Volume Horaire . . . . .	11
2.2 Introduction . . . . .	11
2.3 Robot manipulateur RR . . . . .	11
2.4 Tableau DH . . . . .	11
2.4.1 Programme MATLAB . . . . .	12
2.4.2 Résultats . . . . .	13
2.5 Robot manipulateur RPP . . . . .	14
2.5.1 Tableau DH . . . . .	14
2.5.2 Programme MATLAB . . . . .	14
2.5.3 Résultats . . . . .	16
2.6 Robot manipulateur PRR . . . . .	16
2.6.1 Tableau DH . . . . .	16
2.6.2 Programme MATLAB . . . . .	16
2.6.3 Résultats . . . . .	18
<b>3 TP III : Mécanique pour la Robotique</b>	<b>19</b>
3.1 Objectifs . . . . .	19
3.2 Volume Horaire . . . . .	19
3.3 Introduction . . . . .	19
3.4 Robot PRR . . . . .	19
3.4.1 Tableau DH . . . . .	19
3.5 Robot RRP . . . . .	20
3.5.1 Tableau DH . . . . .	20
3.6 Robot RRPR . . . . .	21
3.6.1 Tableau DH . . . . .	21
3.7 Programme MATLAB . . . . .	21

<b>4 TP IV : Modèle Géométrique Inverse</b>	<b>24</b>
4.1 Exercice 1 . . . . .	24
4.1.1 Robot manipulateur RP . . . . .	24
4.1.2 Programme MATLAB pour le modèle géométrique inverse des robots RP . . . . .	25
4.1.3 Robot manipulateur RR . . . . .	26
4.1.4 Programme MATLAB pour le modèle géométrique inverse des robots RR . . . . .	28
4.2 Exercice 2 . . . . .	29
4.2.1 Programme MATLAB utilisant Newton-Raphson pour un robot RR . . . . .	31
4.3 Exercice 3: Robot manipulateur RRR . . . . .	31
4.3.1 Expression de $q_1, q_2, q_3$ en fonction de $\sigma_x, \sigma_y, \sigma_z, a_1, a_2, d_1, d_4$ . . . . .	31
4.3.2 Programme MATLAB pour calculer $q_1, q_2, q_3$ . . . . .	33
<b>5 TP V Robotique</b>	<b>34</b>
5.1 Robot A . . . . .	34
5.1.1 Tableau DH . . . . .	34
5.2 Robot B . . . . .	34
5.2.1 Tableau DH . . . . .	35
5.3 Robot C . . . . .	35
5.3.1 Tableau DH . . . . .	35
5.4 Robot D . . . . .	36
5.4.1 Tableau DH . . . . .	36
5.4.2 Programme MATLAB . . . . .	37

# 1 TP I : Initiation à la Robotique

## 1.1 Objectifs

À la fin de ce TP, nous, étudiants, aurons acquis les connaissances nécessaires pour :

- Se familiariser avec l'interface de l'environnement Matlab.
- Maîtriser les commandes de base en informatique.
- Maîtriser les commandes de gestion des variables.
- Saisir et manipuler des variables de différents types.
- Dessiner des courbes en maîtrisant les options d'affichage.
- Importer et visualiser des données.

## 1.2 Volume Horaire

4 heures

## 1.3 Introduction

Matlab est un environnement de calcul scientifique développé à l'origine pour les calculs matriciels, d'où l'abréviation MATrix Laboratory (laboratoire de matrices). L'outil permet de résoudre des problèmes à travers des algorithmes, des simulations, des diagrammes, etc. Toutes les variables de MATLAB sont des tableaux multidimensionnels, quel que soit le type de données. Une matrice est un tableau à deux dimensions fréquemment utilisé en algèbre linéaire.

## 1.4 Exercice 1 : Matrices

Considérons la matrice suivante :

$$M = \begin{pmatrix} 2 & 1 & -1 \\ 3 & 2 & 2 \\ 5 & 4 & 3 \end{pmatrix}$$

1.1. Matrice M

```
1 M = [2 1 -1; 3 2 2; 5 4 3];
```

1.2. Affichage de la 1ère colonne

```
1 M(:,1)
```

1.3. Affichage de la 3ème ligne

```
1 M(3,:)
```

1.4. Affichage des éléments (2,2) et (2,3)

```
1 M(2,2)
2 M(2,3)
```

1.5. Remplacement de la valeur de l'élément (1,3) par 4

```
1 M(1,3) = 4;
```

1.6. Transposée de M

```
1 M_transpose = M.';


```

1.7. Déterminant de M

```
1 det_M = det(M);
```

1.8. Inverse de M

```
1 inv_M = inv(M);
```

1.9. Trace et rang de M

```
1 trace_M = trace(M);
2 rang_M = rank(M);
```

1.10. Résultats des commandes `eye(4)`, `ones(5,4)`, `zeros(3)`

$$eye(4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad ones(5,4) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad zeros(3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

## 1.5 Exercice 2

Programme qui permet à l'utilisateur d'entrer un nombre et d'afficher un message si le nombre est pair ou impair.

```
1 number = input('Entrez un nombre : ');
2 if mod(number, 2) == 0
3     disp('Le nombre est pair');
4 else
5     disp('Le nombre est impair');
6 end
```

## 1.6 Exercice 3

3.1. Taper les commandes suivantes :

```

1 t = 0:0.001:1;
2 y = sin(2*pi*50*t) + 2*sin(2*pi*120*t);
3
4 figure(1)
5 plot(sin(2*pi*50*t))
6
7 figure(2)
8 plot(y)

```

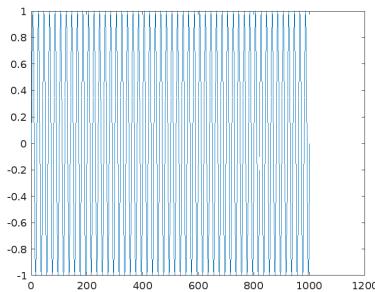


Figure 1:  $\sin(2\pi \cdot 50 \cdot t)$

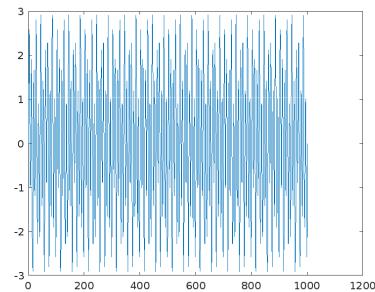


Figure 2: Graphique de y

### 3.2. Comparer les deux courbes

```

1 yn = y + 0.5*randn(size(t));
2
3 figure(3)
4 plot(t(1:50), yn(1:50))
5
6 figure(4)
7 plot(yn)

```

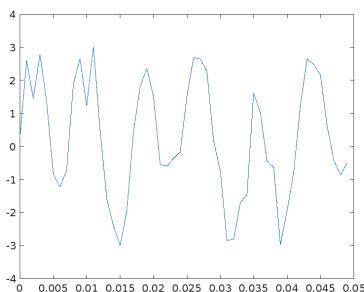


Figure 3:  $\sin(2\pi \cdot 50 \cdot t)$

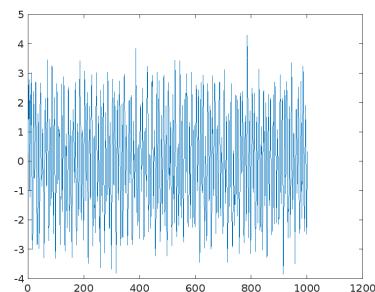


Figure 4: Graphique de y

**Figure 3 :** Ne montre pas les tendances à long terme ou le comportement global.

**Figure 4 :** Permet de voir les tendances générales et les comportements à long terme de toute la série de données.

3.3. La commande `randn()` génère des nombres aléatoires suivant une distribution normale (moyenne = 0, variance = 1).

## 1.7 Exercice 4 : Graphique 2D

Soit les deux fonctions :

$$x1 = \sin(2\pi t)$$

$$x2 = \cos(2\pi t)$$

4.1. Tracer le graphe du vecteur  $x1$  en fonction du temps avec un pas de 0.01, en utilisant la fonction `plot`.

```

1 t = 0:0.01:1;
2 x1 = sin(2*pi*t);
3
4 figure
5 plot(t, x1)

```

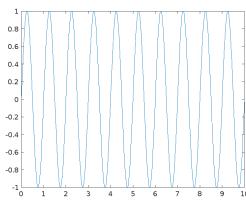


Figure 5: Graphique de  $x1 = \sin(2\pi t)$

La différence entre les fonctions `plot(x)` et `plot(t,x)` est que la première utilise les indices du vecteur comme valeurs de l'axe des x, tandis que la seconde utilise explicitement le vecteur  $t$  pour les valeurs de l'axe des x.

4.2. Pour ajouter un titre et des étiquettes aux axes, on tape les commandes suivantes :

```

1 title('Graphique_de_x1_en_fonction_du_temps')
2 xlabel('Temps_(s)')
3 ylabel('Amplitude')

```

4.3. En utilisant `hold on`, on obtient le graphe ci-dessous :

```

1 >> plot(x1, t)
2 >> hold on
3 >> plot(x2, t)

```

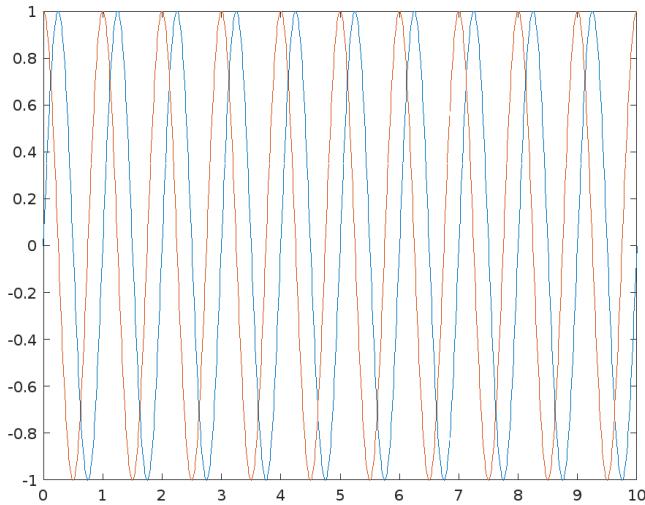


Figure 6: Graphiques avec hold on : x1 et x2

La commande **hold on** permet de superposer plusieurs graphes sur une même figure sans effacer le graphique existant.

4.4. En utilisant **subplot** comme défini, on obtient le graphe ci-dessous :

```
1 >> subplot(2,1,1); plot(t, x1);
2 >> subplot(2,1,2); plot(t, x2);
```

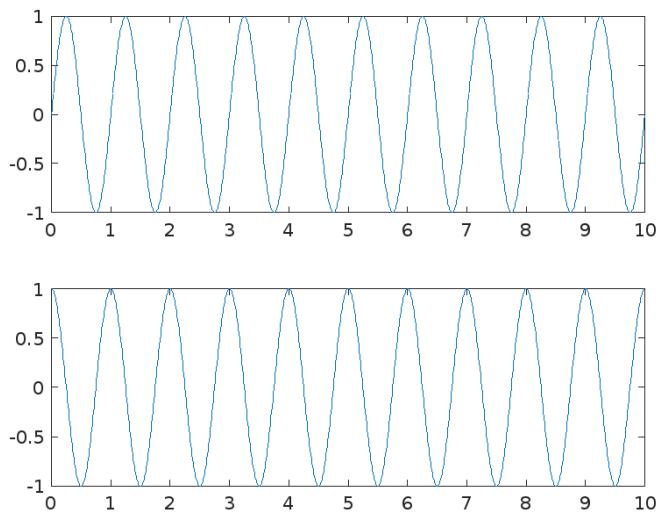


Figure 7: Graphiques en sous-plots : x1 et x2

La fonction `subplot` est utilisée pour diviser une figure en plusieurs sous-graphes, permettant ainsi de tracer plusieurs graphiques distincts dans une même fenêtre de figure.

4.5. Entrons les commandes suivantes:

```
1 >> plot(t, x, 'r--o')
2 >> plot(x, y1, 'r:+')
```

Résultat :

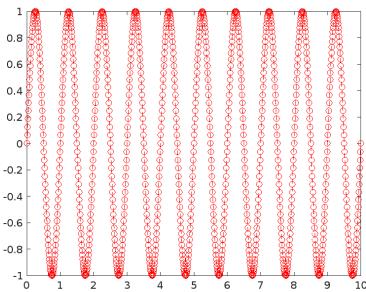


Figure 8: `plot(t, x, 'r--o')`

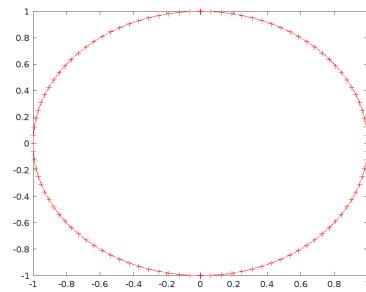


Figure 9: `plot(x, y1, 'r:+')`

4.6. Entrons les commandes suivantes :

```
1 >> x = 0:pi/100:2*pi;
2 >> y1 = sin(x);
3 >> y2 = sin(x-0.25);
4 >> plot(x, y1, x, y2)
5 >> legend('sin(x)', 'sin(x-0.25)')
```

Résultat :

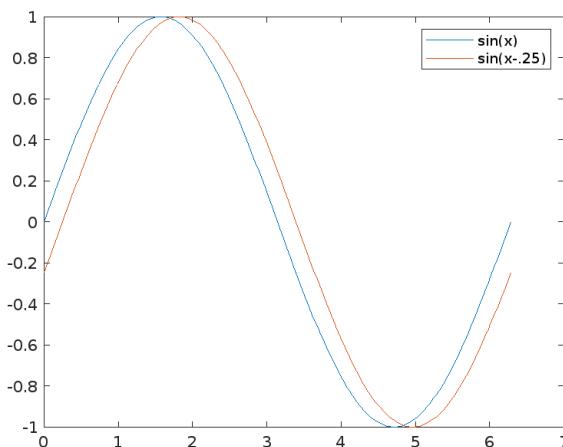


Figure 10: Ajout d'une légende aux graphiques

4.7. Entrons les commandes suivantes :

```
1 >> x = 1:10;
2 >> y = sin(x);
3 >> stem(x, y);
4 >> plot(x, y);
5 >> bar(x, y);
```

Résultat :

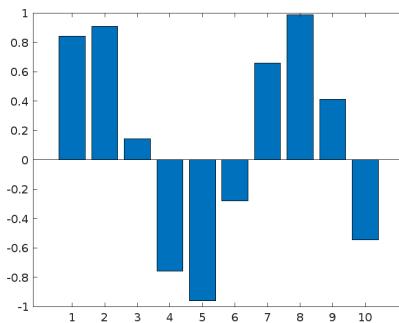


Figure 11: Histogramme

La commande **stem(x, y)** produit un diagramme en tiges où chaque point de données est représenté par une tige verticale. La commande **plot(x, y)** crée un graphique linéaire en connectant les points de données avec des lignes droites. Enfin, la commande **bar(x, y)** génère un graphique en barres, où chaque barre représente la valeur d'un point de données.

4.8. Entrons les commandes suivantes :

```
1 >> x = -pi:pi/20:pi;
2 >> plot(x, exp(-x.^2));
3 >> hold on
4 >> bar(x, exp(-x.^2), 'g');
5 >> hold off
6 >> stem(x, exp(-x.^2), 'r');
```

Résultat :

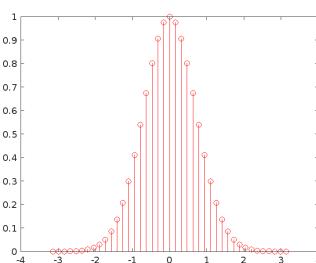


Figure 12: Figure 12

#### 4.9. Simulink : Refaire le diagramme.

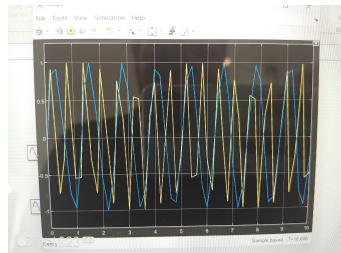


Figure 13: Premier résultat obtenu

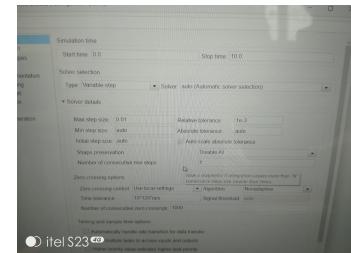


Figure 14: Paramétrage du max step size

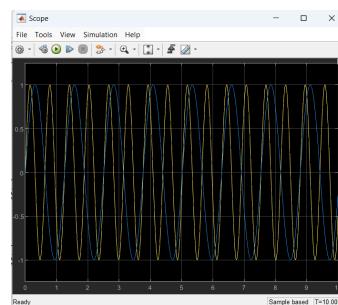


Figure 15: Résultat final

## 2 TP II : Robotique

### 2.1 Volume Horaire

4 heures

### 2.2 Introduction

Ce rapport présente les résultats des calculs du modèle géométrique direct pour trois types de robots : RR, RPP et PRR (les résultats fournis ici ont été calculés sur Matlab à l'aide d'un programme qui sera joint à ce document).

### 2.3 Robot manipulateur RR

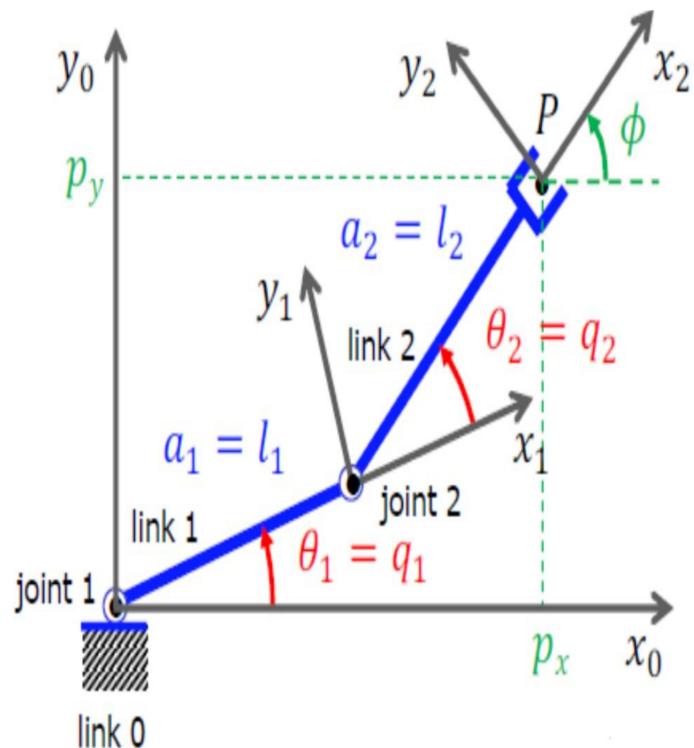


Figure 16: Robot RR

### 2.4 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$q1$	0	$l1$	0
2	$q2$	0	$l2$	0

### 2.4.1 Programme MATLAB

```

1 close all
2 clear all
3 function robotRR_DH()
4 % Demande l'utilisateur de saisir les valeurs des paramtres
5 l1 = input('Entrez la longueur du premier bras (l1):');
6 l2 = input('Entrez la longueur du deuxime bras (l2):');
7 q1 = input('Entrez l''angle de la premiere articulation (q1) en degrs:');
8 q2 = input('Entrez l''angle de la deuxime articulation (q2) en degrs:');
9
10 % Conversion des angles en radians
11 q1 = deg2rad(q1);
12 q2 = deg2rad(q2);
13
14 % Paramtres DH
15 DH_params = [
16     q1, 0, l1, 0; % [theta, d, a, alpha] pour le lien 1
17     q2, 0, l2, 0 % [theta, d, a, alpha] pour le lien 2
18 ];
19
20 % Calcul de la matrice de transformation homogne totale
21 T = eye(4);
22 for i = 1:size(DH_params, 1)
23     T_i = DH_transform(DH_params(i, :));
24     T = T * T_i;
25 end
26
27
28 % Extraire la position et orientation finale
29 x = T(1, 4);
30 y = T(2, 4);
31 z = T(3, 4);
32 R = T(1:3, 1:3);
33
34 % Affichage des rsultats
35 fprintf('Position de l''effecteur:\n');
36 fprintf('x: %.2f\n', x);
37 fprintf('y: %.2f\n', y);
38 fprintf('z: %.2f\n', z);
39
40 fprintf('\nOrientation de l''effecteur (matrice de rotation):\n');
41 disp(R);
42 fprintf('\nalpha:0\n')
43 fprintf('beta:0\n')
44 fprintf('gamma: %.2f\n', rad2deg(q1+q2))
45
46 end
47 function T = DH_transform(params)
48 % Calcule la matrice de transformation homogne pour les paramtres DH
49 theta = params(1);
50 d = params(2);
51 a = params(3);
52 alpha = params(4);
53
54 T = [cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha), a*cos(theta);
55      sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta);
56      0, sin(alpha), cos(alpha), d;
57      0, 0, 0, 1];
58 end
59 robotRR_DH()

```

### 2.4.2 Résultats

Les résultats pour  $q1 = q2 = 30^\circ$  et  $l1 = l2 = 2m$  sont :

Position de l'effecteur:

x: 2.73

y: 2.73

z: 0.00

Orientation de l'effecteur (matrice de rotation):

0.5000	-0.8660	0
0.8660	0.5000	0
0	0	1.0000

alpha:0

beta:0

gamma: 60.00

## 2.5 Robot manipulateur RPP

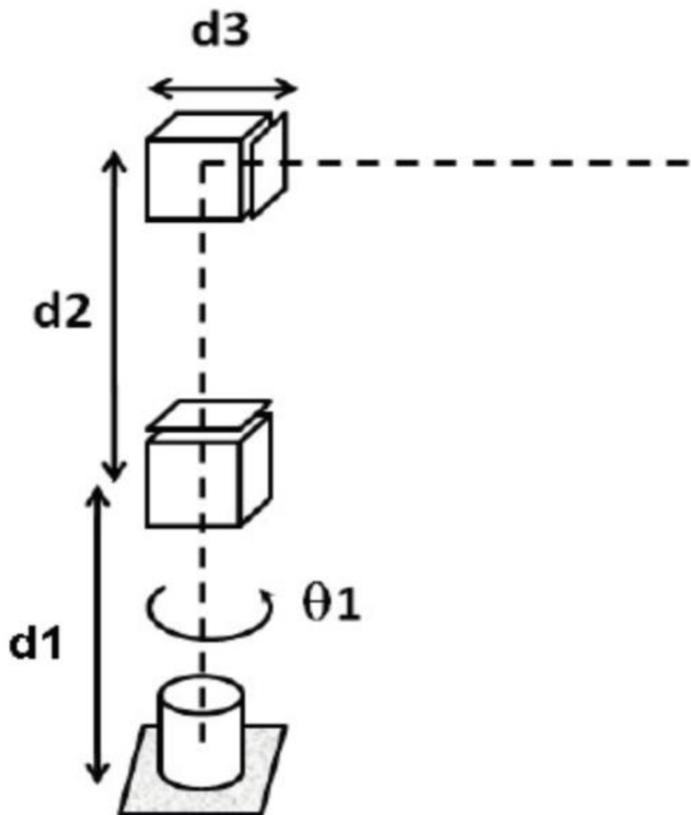


Figure 17: Robot RPP

### 2.5.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$q1$	$l1$	0	0
2	0	$d2$	0	$-\frac{\pi}{2}$
3	0	$d3$	0	0

### 2.5.2 Programme MATLAB

```

1 function robotRPP()
2 % Demande l'utilisateur de saisir les valeurs des parametres
3 l1= input('Entrez la longueur du premier bras (l1):');
4 q1 = input('Entrez l\'angle de la rotation (q1) en degrs:');
5 d2 = input('Entrez la translation du premier prisme (d2):');
6 d3 = input('Entrez la translation du deuxieme prisme (d3):');
7

```

```

8 % Conversion de l'angle en radians
9 q1 = deg2rad(q1);
10
11 % Paramtres DH
12 DH_params = [
13     q1, l1, 0, 0; % [theta, d, a, alpha] pour la rotation
14     0, d2, 0, -pi/2; % [theta, d, a, alpha] pour la premire translation
15     0, d3, 0, 0; % [theta, d, a, alpha] pour la deuxime translation
16 ];
17
18 % Calcul de la matrice de transformation homogne totale
19 T = eye(4);
20 for i = 1:size(DH_params, 1)
21     T_i = DH_transform(DH_params(i, :));
22     T = T * T_i;
23 end
24
25 % Extraire la position et orientation finale
26 x = T(1, 4);
27 y = T(2, 4);
28 z = T(3, 4);
29 R = T(1:3, 1:3);
30
31 % Affichage des rsultats
32 fprintf('Position de l''effecteur:\n');
33 fprintf('x: %.2f\n', x);
34 fprintf('y: %.2f\n', y);
35 fprintf('z: %.2f\n', z);
36
37 fprintf('\nOrientation de l''effecteur (matrice de rotation):\n');
38 disp(R);
39 alpha=0;
40 beta=0;
41 gamma=rad2deg(q1);
42
43 fprintf('alpha: %.2f\n', alpha);
44 fprintf('beta: %.2f\n', beta);
45 fprintf('gamma: %.2f\n', gamma);
46
47 end
48
49 function T = DH_transform(params)
50 % Calcule la matrice de transformation homogne pour les paramtres DH
51 theta = params(1);
52 d = params(2);
53 a = params(3);
54 alpha = params(4);
55
56 T = [cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha), a*cos(theta);
57       sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta);
58       0, sin(alpha), cos(alpha), d;
59       0, 0, 0, 1];
60 end
61
62 % Fonction DH_transform identique celle utilise pour le robot RR
63 % Vrification pour des valeurs spcifiques
64 fprintf('Vrification pour q1=30 degrs, d2=2m, d3=2m, l1=2m\n');
65 robotRPP();

```

### 2.5.3 Résultats

Les résultats pour  $q1 = 30^\circ$ ,  $d2 = 2m$ ,  $d3 = 2m$  et  $l1 = 2m$  sont :

x: -1.00

y: 1.73

z: 4.00

Orientation de l'effecteur (matrice de rotation):

0.8660	-0.0000	-0.5000
0.5000	0.0000	0.8660
0	-1.0000	0.0000

alpha: 0.00

beta: 0.00

gamma: 30.00

## 2.6 Robot manipulateur PRR

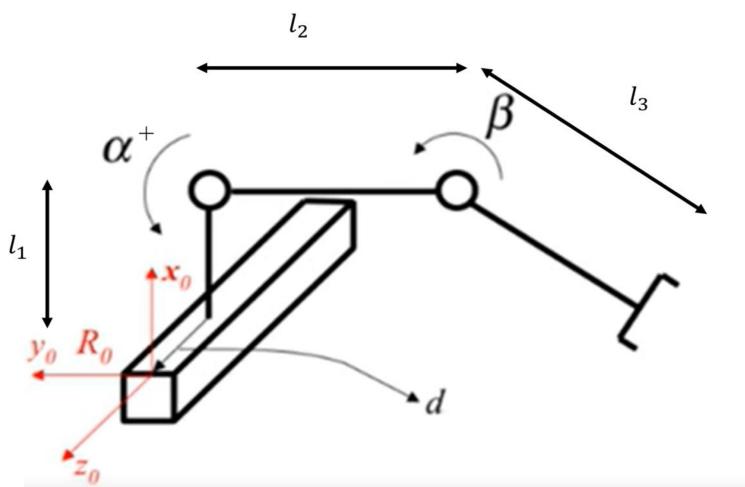


Figure 18: Robot PRR

### 2.6.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	0	$d1$	0	0
2	$q2$	0	$l2$	0
3	$q3$	0	$l3$	0

### 2.6.2 Programme MATLAB

```

1 function robotPRR()
2 % Demande l'utilisateur de saisir les valeurs des paramtres
3 d1 = input('Entrez la translation du prisme (d1):');
4 l1 = input('Entrez la longueur du premier bras (l1):');
5 l2 = input('Entrez la longueur du deuxime bras (l2):');
6 l3= input('Entrez la longueur du troisieme bras (l3):');
7 q2 = input('Entrez l''angle de la premiere rotation (q2) en degres:');
8 q3 = input('Entrez l''angle de la deuxime rotation (q3) en degres:');
9 % Conversion des angles en radians
10 q2 = deg2rad(q2);
11 q3 = deg2rad(q3);
12 % Paramtres DH
13 DH_params = [
14     0, d1, l1, 0; % [theta, d, a, alpha] pour la translation
15     q2, 0, l2, 0; % [theta, d, a, alpha] pour la premire rotation
16     q3, 0, l3, 0 % [theta, d, a, alpha] pour la deuxime rotation
17 ];
18 % Calcul de la matrice de transformation homogne totale
19 T = eye(4);
20 for i = 1:size(DH_params, 1)
21     T_i = DH_transform(DH_params(i, :));
22     T = T * T_i;
23 end
24 % Extraire la position et orientation finale
25 x = T(1, 4);
26 y = T(2, 4);
27 z = T(3, 4);
28 R = T(1:3, 1:3);
29 % Affichage des rsultats
30 fprintf('Position de l''effecteur:\n');
31 fprintf('x: %.2f\n', x);
32 fprintf('y: %.2f\n', y);
33 fprintf('z: %.2f\n', z);
34 fprintf('\nOrientation de l''effecteur (matrice de rotation):\n');
35 disp(R);
36 alpha=0;
37 beta=0;
38 gamma=rad2deg(q3)+rad2deg(q2);
39 fprintf('alpha: %.2f\n', alpha)
40 fprintf('beta: %.2f\n', beta)
41 fprintf('gamma: %.2f\n', gamma)
42 end
43 function T = DH_transform(params)
44 % Calcule la matrice de transformation homogne pour les paramtres DH
45 theta = params(1);
46 d = params(2);
47 a = params(3);
48 alpha = params(4);
49 T = [cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha), a*cos(theta);
50       sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta);
51       0, sin(alpha), cos(alpha), d;
52       0, 0, 0, 1];
53 end
54 % Fonction DH_transform identique celle utilise pour le robot RR
55 % Vrification pour des valeurs spcifiques
56 fprintf('Vrification pour d1=2m, q2=30 degres, l1=l2=l3=2m, q3=30 degres\n');
57 robotPRR();

```

### 2.6.3 Résultats

Les résultats pour  $d1 = 2m$ ,  $q2 = 30^\circ$ ,  $l1 = 2m$ ,  $l2 = 2m$ ,  $q3 = 30^\circ$ ,  $l3 = 2m$  sont :

Position de l'effecteur:

x: 4.73

y: 2.73

z: 2.00

Orientation de l'effecteur (matrice de rotation):

0.5000	-0.8660	0
0.8660	0.5000	0
0	0	1.0000

alpha: 0.00

beta: 0.00

gamma: 60.00

### 3 TP III : Mécanique pour la Robotique

#### 3.1 Objectifs

À la fin de ce TP, nous, étudiants, aurons acquis les connaissances nécessaires pour :

- Comprendre le modèle géométrique direct des robots.
- Savoir utiliser les paramètres DH (Denavit-Hartenberg).
- Programmer le modèle géométrique direct des robots sur matlab.

#### 3.2 Volume Horaire

4 heures

#### 3.3 Introduction

Ce rapport présente les résultats des calculs du modèle géométrique direct pour trois types de robots : PRR, RRP et RRPR.

#### 3.4 Robot PRR

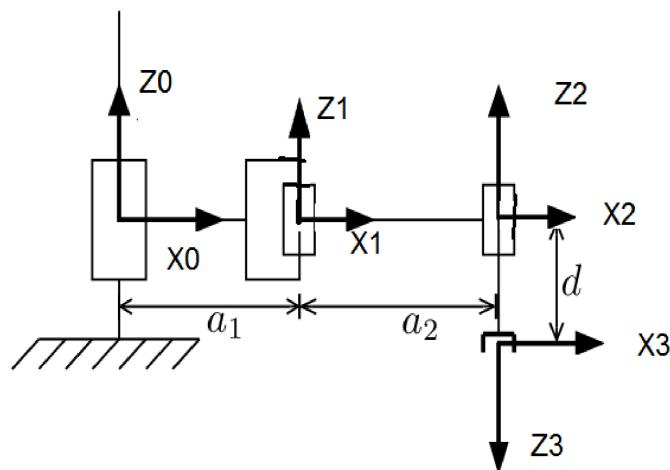


Figure 19: Robot PRR

##### 3.4.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	0	$q1$	$a1$	0
2	$\theta_2$	0	$a2$	0
3	$\theta_3$	$-d$	0	$\pi$

### 3.5 Robot RRP

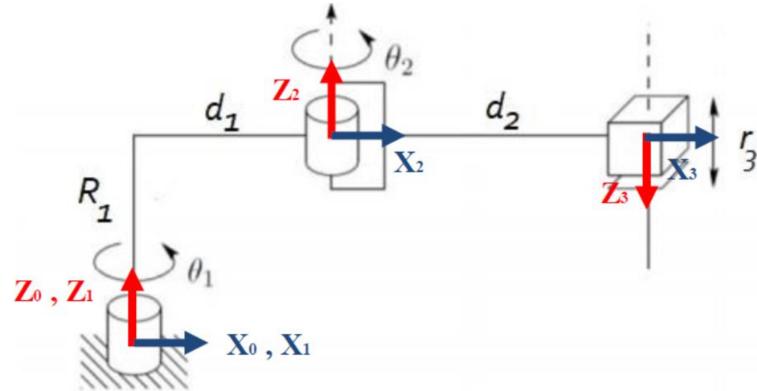


Figure 20: Robot RRP

#### 3.5.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	0	0
2	$\theta_2$	$R_1$	$d1$	0
3	0	$r_3$	$d2$	$\pi$

### 3.6 Robot RRPR

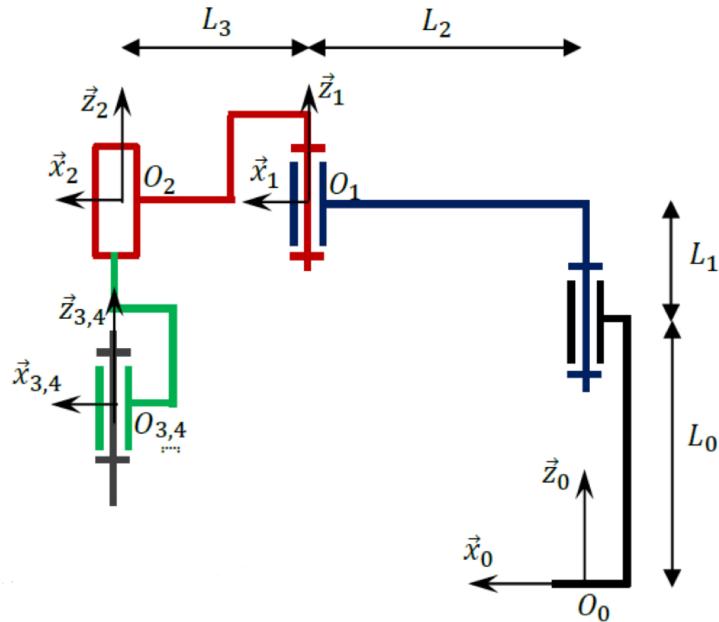


Figure 21: Robot RRPR

#### 3.6.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	$l1 + l0$	$l2$	0
2	$\theta_2$	0	$l3$	0
3	0	$q3$	0	0
4	$\theta_4$	0	0	0

### 3.7 Programme MATLAB

```

1 function main
2 % Demander l'utilisateur le type de robot
3 robot_type = input('Entrez le type de robot (PRR, RRP, RRPR)', 's');
4
5 % Determiner le nombre d'articulations partir du type de robot
6 n = length(robot_type);
7
8 % Initialiser la matrice des paramtres DH
9 DH_params = zeros(n, 4);
10
11 % Boucle pour demander l'utilisateur de saisir les paramtres DH pour chaque
12 % articulation
13 for i = 1:n
14     fprintf('Entrez les paramtres DH pour la %d articulation\n', i);
15     theta = input('Theta (en degrs)');
16     d = input('d');
17
18 end

```

```

16     a = input('a:');
17     alpha = input('Alpha(en_degrs):');
18
19     % Convertir les angles de degrs en radians
20     theta = deg2rad(theta);
21     alpha = deg2rad(alpha);
22
23     % Stocker les paramtres DH dans la matrice
24     DH_params(i, :) = [theta, d, a, alpha];
25 end
26
27 % Calculer la matrice de transformation totale
28 T = direct_kinematics(DH_params);
29
30 % Afficher la matrice de transformation totale
31 disp('La matrice de transformation totale est :');
32 disp(T);
33
34 % Extraire les positions et orientations de la matrice de transformation
35 position = T(1:3, 4); % Coordonnes x, y, z
36 R = T(1:3, 1:3); % Matrice de rotation
37
38 % Afficher les positions et orientations
39 fprintf('Position de l''effecteur :\n');
40 fprintf('x=%f\n', position(1));
41 fprintf('y=%f\n', position(2));
42 fprintf('z=%f\n', position(3));
43
44 % Calculer gamma en fonction du type de robot
45 switch robot_type
46     case 'PRR'
47         gamma = rad2deg(DH_params(2, 1) + DH_params(3, 1));
48     case 'RRP'
49         gamma = rad2deg(DH_params(1, 1) + DH_params(2, 1));
50     case 'RRPR'
51         gamma = rad2deg(DH_params(1, 1) + DH_params(2, 1) + DH_params(4, 1));
52     otherwise
53         error('Type de robot non pris en charge');
54 end
55
56 % Afficher les angles d'Euler (alpha, beta, gamma)
57 fprintf('Orientation de l''effecteur (en degrs):\n');
58 fprintf('alpha=%f\n', 0.00);
59 fprintf('beta=%f\n', 0.00);
60 fprintf('gamma=%f\n', gamma);
61 end
62
63 function T = direct_kinematics(DH_params)
64     % Fonction pour calculer le modle gomtrique direct partir des paramtres DH
65     %
66     % Entre :
67     % DH_params - Une matrice nx4 o chaque ligne correspond aux paramtres DH [theta,
68     % d, a, alpha]
69     %
70     % Sortie :
71     % T - La matrice de transformation homogne totale de la base l'effecteur
72
73     % Nombre d'articulations
74     n = size(DH_params, 1);
75
76     % Initialisation de la matrice de transformation totale comme matrice identit
T = eye(4);

```

```

77
78 % Boucle sur chaque articulation pour calculer la matrice de transformation
79 for i = 1:n
80     theta = DH_params(i, 1);
81     d = DH_params(i, 2);
82     a = DH_params(i, 3);
83     alpha = DH_params(i, 4);
84
85 % Matrice de transformation pour l'articulation i
86 A_i = [
87     cos(theta), -sin(theta) * cos(alpha), sin(theta) * sin(alpha), a * cos(
88         theta);
89     sin(theta), cos(theta) * cos(alpha), -cos(theta) * sin(alpha), a * sin(
90         theta);
91     0, sin(alpha), cos(alpha), d;
92     0, 0, 0, 1;
93 ];
94
95 % Mise jour de la matrice de transformation totale
96 T = T * A_i;
97 end
98
99 % Appeler la fonction principale
main()

```

Ce programme en MATLAB calcule et affiche la position et l'orientation de l'effecteur d'un robot en utilisant les paramètres de Denavit-Hartenberg (DH). L'utilisateur spécifie le type de robot parmi **PRR**, **RRP** et **RRPR**, ce qui détermine le nombre d'articulations en fonction du nombre de lettres du type de robot. Ensuite, pour chaque articulation, l'utilisateur entre les paramètres DH :  $\theta$ ,  $d$ ,  $a$  et  $\alpha$ , où  $\theta$  et  $\alpha$  sont convertis de degrés en radians. La fonction **direct\_kinematics** est utilisée pour calculer la matrice de transformation totale à partir des paramètres DH, en multipliant successivement les matrices de transformation individuelles pour chaque articulation. La position de l'effecteur est extraite de cette matrice et affichée. L'orientation de l'effecteur est calculée en fonction du type de robot, avec les angles d'Euler  $\alpha$  et  $\beta$  fixés à zéro et  $\gamma$  calculé selon une règle spécifique à chaque type de robot. Enfin, les résultats, y compris la position et l'orientation de l'effecteur, sont affichés.

**ATTENTION:**  $\alpha$ ,  $\beta$  et  $\gamma$  ici sont calculer manuellement en fonction de chaque type de robot, donc ce programme ne fonctionne que pour ces robots spécifique.

## 4 TP IV : Modèle Géométrique Inverse

### 4.1 Exercice 1

#### 4.1.1 Robot manipulateur RP

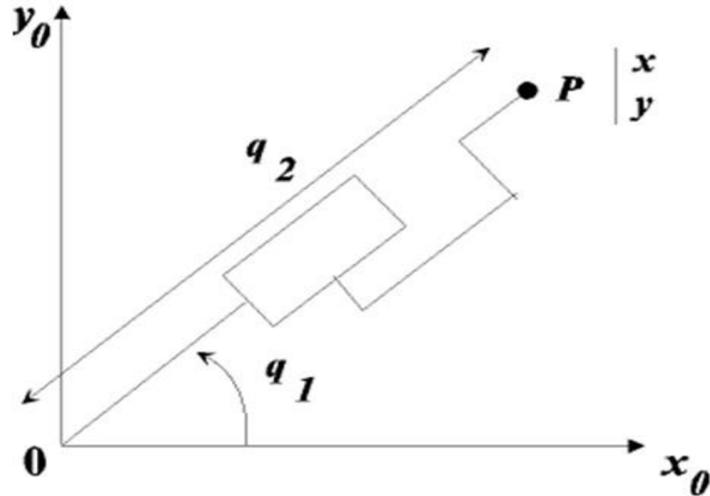


Figure 22: Robot RP

Le modèle géométrique du robot est défini par la relation suivante :

$$x = q_2 \cos(q_1)$$

$$y = q_2 \sin(q_1)$$

Pour déterminer  $q_2$ , nous utilisons les deux équations données :

$$x = q_2 \cos(q_1)$$

$$y = q_2 \sin(q_1)$$

En élevant les deux équations au carré et en les additionnant, nous obtenons :

$$x^2 = q_2^2 \cos^2(q_1)$$

$$y^2 = q_2^2 \sin^2(q_1)$$

En additionnant ces deux équations, nous avons :

$$x^2 + y^2 = q_2^2 \cos^2(q_1) + q_2^2 \sin^2(q_1)$$

$$x^2 + y^2 = q_2^2(\cos^2(q_1) + \sin^2(q_1))$$

Pour déterminer  $q_1$ , nous utilisons les relations trigonométriques de base.

Nous avons :

$$\tan(q_1) = \frac{y}{x}$$

Donc,  $q_1$  peut être trouvé en utilisant la fonction arctangente :

$$q_1 = \arctan\left(\frac{y}{x}\right)$$

## Résultats

En résumé, les valeurs de  $q_1$  et  $q_2$  sont données par :

$$q_2 = \sqrt{x^2 + y^2}$$

$$q_1 = \arctan\left(\frac{y}{x}\right)$$

### 4.1.2 Programme MATLAB pour le modèle géométrique inverse des robots RP

```

1 function inverse_kinematics_RR
2 % Entre des paramtres de l'utilisateur
3 x = input('Entrez la coordonne x de l''effecteur : ');
4 y = input('Entrez la coordonne y de l''effecteur : ');
5
6 % Calcul des angles des articulations
7 q2 = sqrt(x^2 + y^2); % Deux solutions: -sqrt(x^2 + y^2) ou +sqrt(x^2 + y^2)
8 q1 = atan2(y, x);
9
10 % Conversion en degrs
11 q1 = rad2deg(q1);
12 q2 = rad2deg(q2);
13
14 % Affichage des rsultats
15 fprintf('Les angles des articulations sont :\n');
16 fprintf('q1 : %.2f degrs\n', q1);
17 fprintf('q2 : %.2f degrs\n', q2);
18 end
19
20 % Vrification avec des valeurs d'exemple
21 inverse_kinematics_RR;

```

On constate ici 2 configuration possible soit: pour  $q_2$  positif et pour  $q_2$  négatif.

#### 4.1.3 Robot manipulateur RR

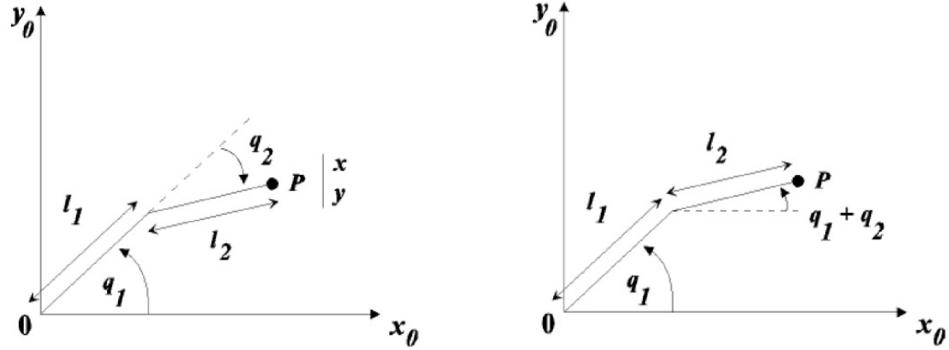


Figure 23: Robot RR

Le modèle géométrique du robot est défini par la relation suivante :

$$x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$

$$y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$

Pour trouver  $q_2$ , nous pouvons suivre les étapes suivantes :

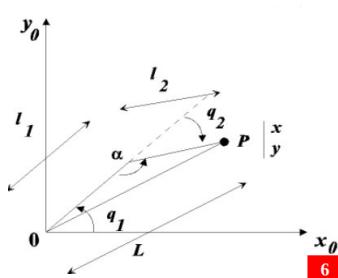


Figure 24: analyse geometrique

On a ainsi d'après la figure précédente

$$L^2 = x^2 + y^2$$

on a également

$$L^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos \alpha$$

avec

$$\alpha = \pi + q_2$$

et

$$\cos(\pi + a) = -\cos(a)$$

donc

$$x^2 + y^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos \alpha$$

$$\cos(q2) = \frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1 l_2}$$

d'où

$$q2 = \arccos\left(\frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1 l_2}\right)$$

Pour trouver  $q1$ , nous pouvons suivre les étapes suivantes :

### **Isoler les termes impliquant $q1$**

Nous réécrivons les équations en utilisant les identités trigonométriques.  
Nous savons que :

$$\cos(q1 + q2) = \cos(q1) \cos(q2) - \sin(q1) \sin(q2)$$

$$\sin(q1 + q2) = \sin(q1) \cos(q2) + \cos(q1) \sin(q2)$$

En substituant ces identités dans les équations initiales, nous obtenons :

$$x = l_1 \cos(q1) + l_2 (\cos(q1) \cos(q2) - \sin(q1) \sin(q2))$$

$$y = l_1 \sin(q1) + l_2 (\sin(q1) \cos(q2) + \cos(q1) \sin(q2))$$

Regroupons les termes en  $\cos(q1)$  et  $\sin(q1)$  :

$$x = (l_1 + l_2 \cos(q2)) \cos(q1) - l_2 \sin(q2) \sin(q1)$$

$$y = (l_1 + l_2 \cos(q2)) \sin(q1) + l_2 \sin(q2) \cos(q1)$$

### **Utiliser l'identité trigonométrique pour résoudre $q1$**

Nous pouvons isoler  $\tan(q1)$  en divisant les deux équations obtenues :

$$\frac{y}{x} = \frac{(l_1 + l_2 \cos(q2)) \sin(q1) + l_2 \sin(q2) \cos(q1)}{(l_1 + l_2 \cos(q2)) \cos(q1) - l_2 \sin(q2) \sin(q1)}$$

En utilisant l'identité trigonométrique, nous pouvons simplifier cette équation pour trouver  $q1$  :

$$\frac{y}{x} = \frac{\sin(q1) + \frac{l_2 \sin(q2)}{l_1 + l_2 \cos(q2)} \cos(q1)}{\cos(q1) - \frac{l_2 \sin(q2)}{l_1 + l_2 \cos(q2)} \sin(q1)}$$

Simplifions l'équation pour obtenir :

$$\frac{y}{x} = \frac{A + k \cos(q1)}{\cos(q1) - k \sin(q1)}$$

où  $k = \frac{l_2 \sin(q2)}{l_1 + l_2 \cos(q2)}$ .

Posons  $A = \tan(q1)$ . Nous avons donc :

$$\frac{y}{x} = \frac{A + k}{1 - kA}$$

En isolant  $A$ , nous obtenons :

$$A = \frac{\frac{y}{x} - k}{1 + k \frac{y}{x}}$$

Ainsi, nous avons :

$$\tan(q1) = \frac{\frac{y}{x} - k}{1 + k \frac{y}{x}}$$

Et finalement :

$$q1 = \arctan\left(\frac{\frac{y}{x} - k}{1 + k \frac{y}{x}}\right)$$

### Résultat final

Le résultat final pour  $q1$  et  $q2$  sont :

$$q2 = \arccos\left(\frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1l_2}\right)$$

$$q1 = \arctan\left(\frac{\frac{y}{x} - \frac{l_2 \sin(q2)}{l_1 + l_2 \cos(q2)}}{1 + \frac{l_2 \sin(q2)}{l_1 + l_2 \cos(q2)} \frac{y}{x}}\right)$$

#### 4.1.4 Programme MATLAB pour le modèle géométrique inverse des robots RR

```

1 function inverse_kinematics
2 % Demande de saisie des paramètres par l'utilisateur
3 x = input('Entrez la coordonnée x:');
4 y = input('Entrez la coordonnée y:');
5 l1 = input('Entrez la longueur du premier segment l1:');
6 l2 = input('Entrez la longueur du deuxième segment l2:');
7 % Calcul de q2
8 q2 = acos((x^2 + y^2 - l1^2 - l2^2) / (2 * l1 * l2));
9 % Calcul de q1
10 q1 = atan2(y * (l1 + l2 * cos(q2)) - x * l2 * sin(q2), x * (l1 + l2 * cos(q2)) +
    y * l2 * sin(q2));
11 % Conversion des angles en degrés
12 q1 = rad2deg(q1);
13 q2 = rad2deg(q2);
14 % Affichage des résultats
15 fprintf('L'angle q1 est: %.2f degrs\n', q1);
16 fprintf('L'angle q2 est: %.2f degrs\n', q2);
17 end
18 % Appel de la fonction
19 inverse_kinematics;

```

## 4.2 Exercice 2

La méthode de Newton-Raphson est une technique itérative utilisée pour trouver les racines d'équations non linéaires. Dans cet exercice, nous appliquons cette méthode pour résoudre le problème du modèle géométrique inverse d'un robot manipulateur planaire à deux degrés de liberté (2-DOF).

Le modèle géométrique direct d'un robot manipulateur à deux degrés de liberté est donné par les équations suivantes :

$$x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$

$$y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$

Où :

- $l_1$  et  $l_2$  sont les longueurs des segments du robot,
- $q_1$  et  $q_2$  sont les angles des articulations,
- $x$  et  $y$  sont les coordonnées de la position de l'extrémité du robot.

Le problème du modèle géométrique inverse consiste à déterminer les angles des articulations  $q_1$  et  $q_2$  à partir des coordonnées  $x$  et  $y$ .

### Méthode de Newton-Raphson

La méthode de Newton-Raphson est utilisée pour résoudre des systèmes d'équations non linéaires. Pour appliquer cette méthode, nous devons d'abord définir un vecteur d'erreur :

$$\mathbf{e}(\mathbf{q}) = \begin{pmatrix} x - (l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)) \\ y - (l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)) \end{pmatrix}$$

Nous voulons trouver les valeurs de  $q_1$  et  $q_2$  qui minimisent ce vecteur d'erreur. La méthode de Newton-Raphson est itérative et utilise la jacobienne du vecteur d'erreur :

$$\mathbf{J}(\mathbf{q}) = \begin{pmatrix} -(l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{pmatrix}$$

L'itération de Newton-Raphson est donnée par :

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \mathbf{J}^{-1}(\mathbf{q}_k) \mathbf{e}(\mathbf{q}_k)$$

### Programme MATLAB

Le programme MATLAB suivant implémente la méthode de Newton-Raphson pour résoudre le modèle géométrique inverse :

```

1 function inverse_kinematics_NR
2 % Demande de saisie des paramètres par l'utilisateur
3 x = input('Entrez la coordonnee x:');
4 y = input('Entrez la coordonnee y:');
5 l1 = input('Entrez la longueur du premier segment l1:');
6 l2 = input('Entrez la longueur du deuxième segment l2:');
7
8 % Vrifier les valeurs initiales
9 if x == 0 && y == 0
10    error('Les coordonnées x et y ne peuvent pas toutes être nulles');
11 end
12 % Estimation initiale des angles
13 q1 = atan2(y, x);
14 q2 = acos((x^2 + y^2 - l1^2 - l2^2) / (2 * l1 * l2));
15 if isnan(q2)
16    error('Les paramètres fournis ne permettent pas de calculer des angles rels');
17 end
18 % Newton-Raphson
19 tol = 1e-6;
20 max_iter = 100;
21 iter = 0;
22 while iter < max_iter
23    iter = iter + 1;
24    % Calcul des fonctions
25    f1 = l1 * cos(q1) + l2 * cos(q1 + q2) - x;
26    f2 = l1 * sin(q1) + l2 * sin(q1 + q2) - y;
27    % Jacobienne
28    J = [-l1 * sin(q1) - l2 * sin(q1 + q2), -l2 * sin(q1 + q2);
29          l1 * cos(q1) + l2 * cos(q1 + q2), l2 * cos(q1 + q2)];
29    % Vrifier si la Jacobienne est singulire
30    if abs(det(J)) < tol
31       error('La matrice Jacobienne est singulire ou presque singulire');
32    end
33    % Calcul de l'inverse de la Jacobienne
34    delta_q = -J \ [f1; f2];
35    % Mise jour des angles
36    q1 = q1 + delta_q(1);
37    q2 = q2 + delta_q(2);
38    % Condition d'arrêt
39    if norm([f1; f2]) < tol
40       break;
41    end
42 end
43 if iter == max_iter
44    error('La méthode de Newton-Raphson n''a pas converg');
45 end
46 % Conversion des angles en degrés
47 q1 = rad2deg(q1);
48 q2 = rad2deg(q2);
49 % Affichage des résultats
50 fprintf('L'angle q1 est: %.2f degrs\n', q1);
51 fprintf('L'angle q2 est: %.2f degrs\n', q2);
52 end
53 % Appel de la fonction
54 inverse_kinematics_NR;

```

## Conclusion

La méthode de Newton-Raphson est un outil puissant pour résoudre le problème du modèle géométrique inverse des robots manipulateurs. En

utilisant cette méthode, nous pouvons déterminer les angles des articulations nécessaires pour atteindre une position donnée.

#### 4.2.1 Programme MATLAB utilisant Newton-Raphson pour un robot RR

#### 4.3 Exercice 3: Robot manipulateur RRR

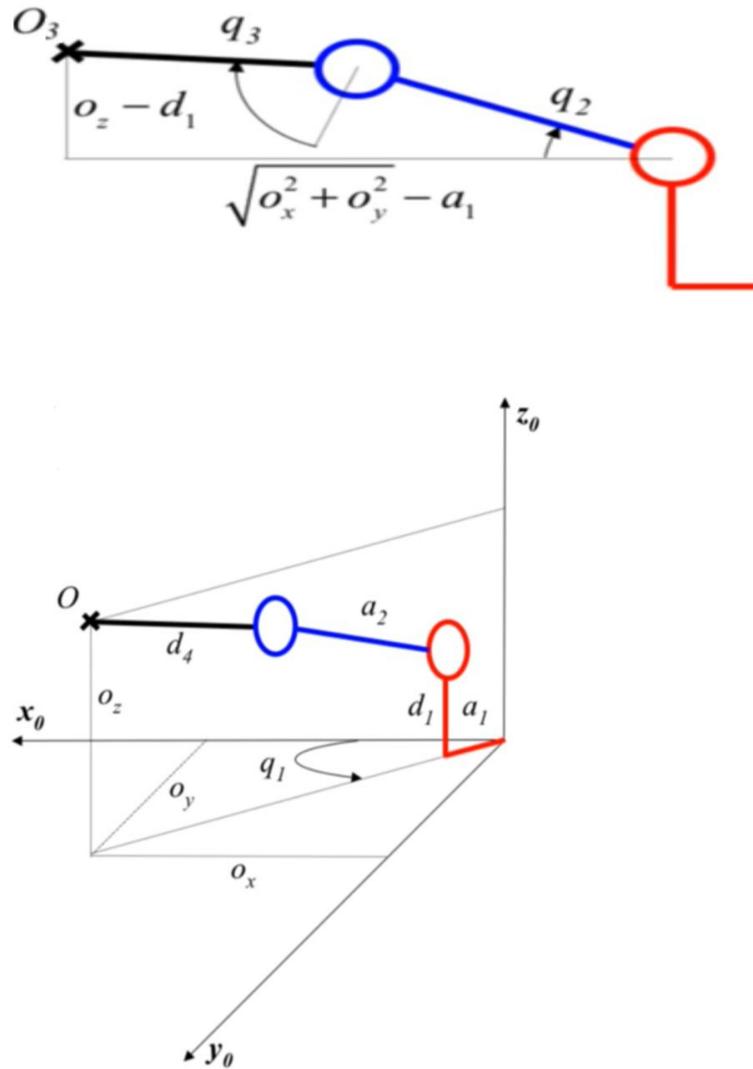


Figure 25: Robot RRR

##### 4.3.1 Expression de $q_1, q_2, q_3$ en fonction de $\sigma_x, \sigma_y, \sigma_z, a_1, a_2, d_1, d_4$ .

Le modèle géométrique du robot est défini par la relation suivante :

$$\begin{aligned}x &= a_2 \cos(q_1) + d_4 \sin(q_3 - q_1) \\y &= a_2 \sin(q_2) + d_4 \cos(q_3 - q_2)\end{aligned}$$

## Relation entre $x^2$ et $y^2$

En élévant au carré et en additionnant les équations, on obtient :

$$x^2 + y^2 = a_2^2 + d_4^2 + 2a_2d_4 [\sin(q_3 - q_2 + q_1)]$$

Ce qui peut être simplifié pour trouver  $\sin q_3$  :

$$\sin q_3 = \frac{x^2 + y^2 - a_2^2 - d_4^2}{2a_2d_4}$$

Et donc :

$$q_3 = \sin^{-1} \left( \frac{x^2 + y^2 - a_2^2 - d_4^2}{2a_2d_4} \right)$$

## Déterminant et cosinus de $q_2$

Pour résoudre  $\cos q_2$ , on utilise le déterminant des matrices associées :

$$\det s = \begin{vmatrix} a_2 + d_4 \sin q_3 & -d_4 \cos q_3 \\ d_4 \cos q_3 & a_2 + d_4 \sin q_3 \end{vmatrix} = (a_2 + d_4 \sin q_3)^2 + d_4^2 \cos^2 q_3$$

Puis, en résolvant pour  $\cos q_2$  :

$$\cos q_2 = \frac{x(a_2 + d_4 \sin q_3) + d_4 \cos q_3 y}{(a_2 + d_4 \sin q_3)^2 + d_4^2 \cos^2 q_3}$$

Donc :

$$q_2 = \cos^{-1} \left( \frac{x(a_2 + d_4 \sin q_3) + d_4 \cos q_3 y}{(a_2 + d_4 \sin q_3)^2 + d_4^2 \cos^2 q_3} \right)$$

## Calcul des angles en fonction des coordonnées et des paramètres

En utilisant les coordonnées de l'origine et les paramètres :

$$\begin{cases} x = \sqrt{\sigma_x^2 + \sigma_y^2} - a_1 \\ y = \sigma_y - d_1 \end{cases}$$

On peut définir :

$$\tan q_1 = \frac{\sigma_y}{\sigma_x}$$

Donc :

$$q_1 = \tan^{-1} \left( \frac{\sigma_y}{\sigma_x} \right)$$

## Synthèse des équations

- Angle  $q_1$  :

$$q_1 = \tan^{-1} \left( \frac{\sigma_y}{\sigma_x} \right)$$

- Angle  $q_3$  :

$$q_3 = \sin^{-1} \left( \frac{x^2 + y^2 - a_2^2 - d_4^2}{2a_2d_4} \right)$$

- Angle  $q_2$  :

$$q_2 = \cos^{-1} \left( \frac{x(a_2 + d_4 \sin q_3) + d_4 \cos q_3 y}{(a_2 + d_4 \sin q_3)^2 + d_4^2 \cos^2 q_3} \right)$$

### 4.3.2 Programme MATLAB pour calculer $q_1, q_2, q_3$

```

1 % Demande des paramtres l'utilisateur
2 a1 = input('Entrez la valeur de a1:');
3 a2 = input('Entrez la valeur de a2:');
4 d1 = input('Entrez la valeur de d1:');
5 d4 = input('Entrez la valeur de d4:');
6 Ox = input('Entrez la coordonnee Ox:');
7 Oy = input('Entrez la coordonnee Oy');

8 % Calcul des coordonnes x et y
9 x = sqrt(Ox^2 + Oy^2) - a1;
10 y = Oy - d1;

11 % Calcul de q1
12 q1 = atan2(Oy, Ox);

13 % Calcul de q3
14 q3 = asin((x^2 + y^2 - a2^2 - d4^2) / (2 * a2 * d4));

15 % Calcul de q2
16 numerator = x * (a2 + d4 * sin(q3)) + d4 * cos(q3) * y;
17 denominator = (a2 + d4 * sin(q3))^2 + d4^2 * cos(q3)^2;
18 cos_q2 = numerator / denominator;
19 q2 = acos(cos_q2);

20 % Affichage des rsultats
21 fprintf('Les angles calculs sont:\n');
22 fprintf('q1=%f radians (%f degrs)\n', q1, rad2deg(q1));
23 fprintf('q2=%f radians (%f degrs)\n', q2, rad2deg(q2));
24 fprintf('q3=%f radians (%f degrs)\n', q3, rad2deg(q3));

```

## 5 TP V Robotique

### 5.1 Robot A

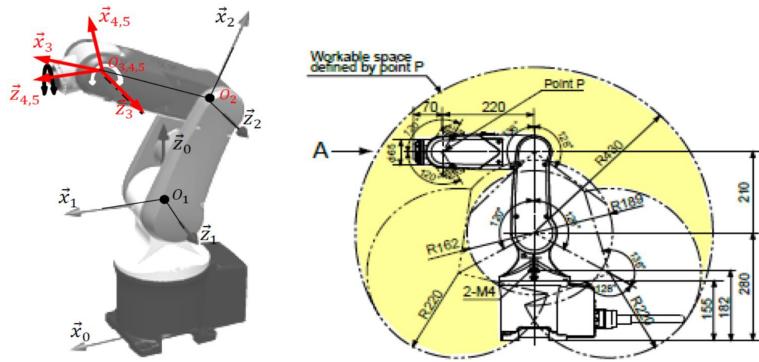


Figure 26: Robot A

#### 5.1.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	280	0	$\frac{\pi}{2}$
2	$\theta_2$	0	210	0
3	$\theta_3$	0	220	0
4	$\theta_4$	0	0	$-\frac{\pi}{2}$
5	$\theta_5$	0	0	0

### 5.2 Robot B

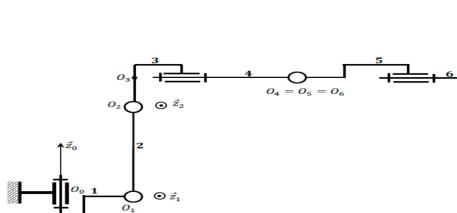


Figure 27: Robot B

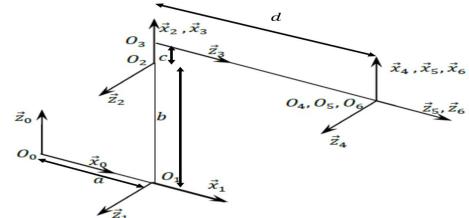


Figure 28: Robot C

### 5.2.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	$a$	$\frac{\pi}{2}$
2	0	$q2$	$b$	0
3	$\theta_3$	0	$c$	0
4	0	0	$q4$	$-\frac{\pi}{2}$
5	0	$q3$	$q5$	$\frac{\pi}{2}$
6	$\theta_6$	0	0	0

### 5.3 Robot C

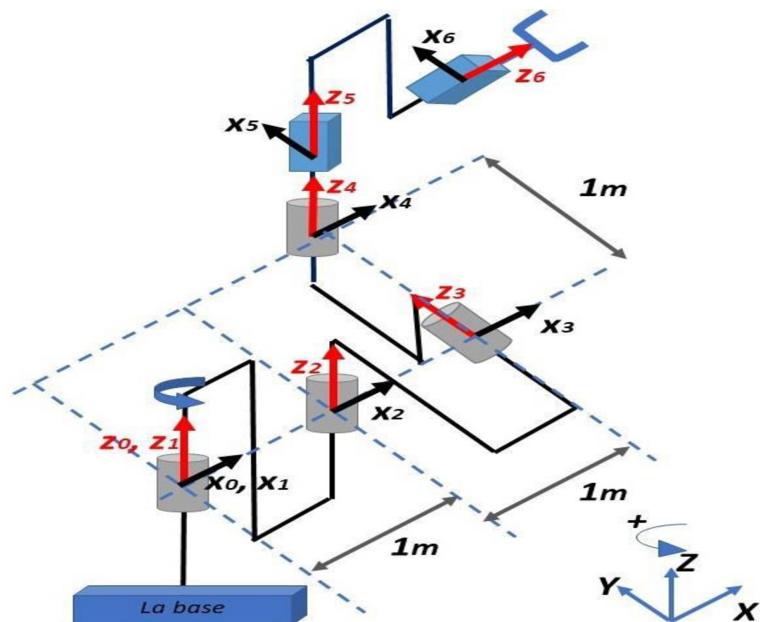


Figure 29: Robot C

### 5.3.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	0	0
2	$\theta_2$	0	1	0
3	$\theta_3$	0	1	$\frac{\pi}{2}$
4	0	1	0	$-\frac{\pi}{2}$
5	0	$q5$	0	0
6	$\theta_6$	$q6$	0	$-\frac{\pi}{2}$

## 5.4 Robot D

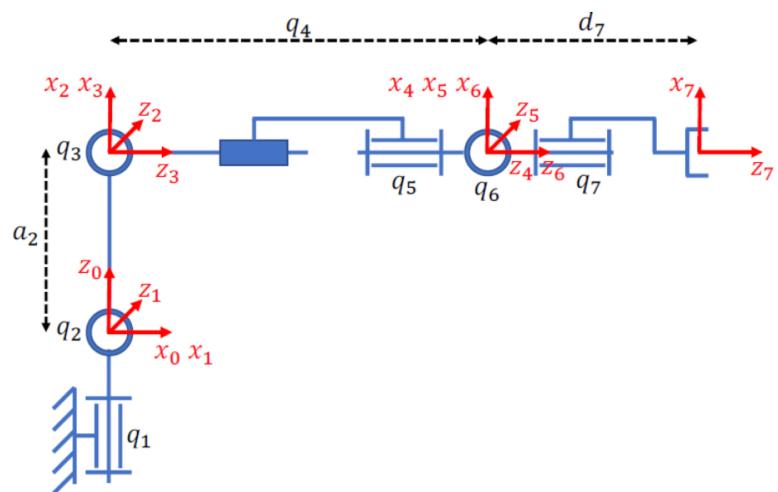


Figure 30: Robot D

### 5.4.1 Tableau DH

Link	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	0	$-\frac{\pi}{2}$
2	$\theta_2$	0	$a_2$	0
3	$\theta_3$	0	0	$-\frac{\pi}{2}$
4	0	$q_4$	0	0
5	$\theta_5$	0	0	$\frac{\pi}{2}$
6	$\theta_6$	0	0	$\frac{\pi}{2}$
7	$\theta_7$	$d_7$	0	$\frac{\pi}{2}$

### 5.4.2 Programme MATLAB

Ce programme calcule le modèle géométrique direct de n'importe quel robot, mais l'orientation du robot n'est pas toujours exacte et peut donner des valeurs erronées pour certains robots.

```

1 function main
2 % Demander l'utilisateur le nombre d'articulations
3 n = input('Entrez le nombre d''articulations : ');
4
5 % Initialiser la matrice des paramètres DH
6 DH_params = zeros(n, 4);
7
8 % Boucle pour demander l'utilisateur de saisir les paramètres DH pour chaque
9 % articulation
10 for i = 1:n
11     fprintf('Entrez les paramètres DH pour l''articulation %d\n', i);
12     theta = input('Theta (en degrés) : ');
13     d = input('d : ');
14     a = input('a : ');
15     alpha = input('Alpha (en degrés) : ');
16
17     % Convertir les angles de degrés en radians
18     theta = deg2rad(theta);
19     alpha = deg2rad(alpha);
20
21     % Stocker les paramètres DH dans la matrice
22     DH_params(i, :) = [theta, d, a, alpha];
23 end
24
25 % Calculer la matrice de transformation totale
26 T = direct_kinematics(DH_params);
27
28 % Afficher la matrice de transformation totale
29 disp('La matrice de transformation totale est : ');
30 disp(T);
31
32 % Extraire les positions et orientations de la matrice de transformation
33 position = T(1:3, 4); % Coordonnées x, y, z
34 R = T(1:3, 1:3); % Matrice de rotation
35
36 % Afficher les positions et orientations
37 fprintf('Position de l''effecteur :\n');
38 fprintf('x = %.2f\n', position(1));
39 fprintf('y = %.2f\n', position(2));
40 fprintf('z = %.2f\n', position(3));
41
42 % Calculer les angles d'Euler (alpha, beta, gamma)
43 [alpha, beta, gamma] = rotm2eul(R);
44
45 % Afficher les angles d'Euler
46 fprintf('Orientation de l''effecteur (en degrés) :\n');
47 fprintf('alpha = %.2f\n', rad2deg(alpha));
48 fprintf('beta = %.2f\n', rad2deg(beta));
49 fprintf('gamma = %.2f\n', rad2deg(gamma));
50
51 function T = direct_kinematics(DH_params)
52 % Fonction pour calculer le modèle géométrique direct partir des paramètres DH
53 %

```

```

54 % Entre :
55 % DH_params - Une matrice nx4 où chaque ligne correspond aux paramètres DH [theta,
56 % d, a, alpha]
57 %
58 % Sortie :
59 % T - La matrice de transformation homogène totale de la base l'effecteur
60
61 % Nombre d'articulations
62 n = size(DH_params, 1);
63
64 % Initialisation de la matrice de transformation totale comme matrice identité
65 T = eye(4);
66
67 % Boucle sur chaque articulation pour calculer la matrice de transformation
68 for i = 1:n
69     theta = DH_params(i, 1);
70     d = DH_params(i, 2);
71     a = DH_params(i, 3);
72     alpha = DH_params(i, 4);
73
74     % Matrice de transformation pour l'articulation i
75     A_i = [
76         cos(theta), -sin(theta) * cos(alpha), sin(theta) * sin(alpha), a * cos(
77             theta);
78         sin(theta), cos(theta) * cos(alpha), -cos(theta) * sin(alpha), a * sin(
79             theta);
80         0, sin(alpha), cos(alpha), d;
81         0, 0, 0, 1;
82     ];
83
84     % Mise à jour de la matrice de transformation totale
85     T = T * A_i;
86 end
87
88 function [alpha, beta, gamma] = rotm2eul(R)
89 % Fonction pour convertir une matrice de rotation en angles d'Euler
90 % Entre :
91 % R - Une matrice de rotation 3x3
92 %
93 % Sortie :
94 % alpha, beta, gamma - Les angles d'Euler (en radians)
95
96 sy = sqrt(R(1,1)^2 + R(2,1)^2);
97
98 if sy > 1e-6
99     alpha = atan2(R(2,1), R(1,1));
100    beta = atan2(-R(3,1), sy);
101    gamma = atan2(R(3,2), R(3,3));
102 else
103     alpha = atan2(-R(2,3), R(2,2));
104     beta = atan2(-R(3,1), sy);
105     gamma = 0;
106 end
107
108 % Appeler la fonction principale
109 main()

```