



جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL



المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

Rapport du projet Robotique Industrielle **ABB IRB4600_205_45**



Réalisé Par :

Sadik Omar

Inrhaoun Hamza

Encadré par :

Mr. Nafi Abdelhak

Remerciements :

Nous souhaitons exprimer nos sincères remerciements à notre encadrant, dont la précieuse contribution a été constante tout au long du projet de robotique industrielle.

Son encadrement attentif, ses conseils éclairés et son expertise ont joué un rôle crucial dans la réussite de ce projet. Sa disponibilité permanente et sa capacité à partager ses connaissances ont enrichi notre expérience, nous guidant efficacement à travers les défis techniques rencontrés.

Nous sommes reconnaissants pour le soutien continu de notre encadrant, qui a été un mentor dévoué, partageant avec nous sa passion pour le domaine de la robotique industrielle. Sa contribution a été essentielle pour acquérir des compétences techniques et une compréhension approfondie des enjeux liés à la conception d'un robot industriel.

Ce projet a représenté une opportunité exceptionnelle d'apprentissage, et nous tenons à exprimer notre gratitude envers notre encadrant pour avoir été un guide exemplaire tout au long de ce parcours. Son engagement a largement contribué à notre développement professionnel et académique.

Nous sommes reconnaissants pour les précieuses leçons que nous avons tirées de cette expérience et avons hâte d'appliquer ces compétences et connaissances dans notre vie professionnelle future. Encore une fois, un sincère merci à notre encadrant pour sa contribution inestimable et son soutien continu.

Table des matières

Remerciements :	2
Introduction :	4
Chapitre 1 : Présentation du projet.....	5
1. Description du projet :	5
2. Modèle géométrique direct :	5
3. Les équations du modèles géométrique direct et sa validation numérique :	6
4. La position de référence du Robot :	8
5. Les équations de la matrice Jacobéenne :	8
6. La validation numérique de la matrice jacobéenne :	11
7. Méthode de commande cinématique :	11
8. Évaluation de la précision du trajet et de la dextérité :	14
9. Graphique des positions articulaires pour la trajectoire :	14
Chapitre 2 : Simulation sur RokiSim	16
1. Fonction simulation_trj.m :	16
2. Simulation des trajectoires :	16
3. Simulation sur le logiciel Rokisim :	18
4. Visualisation des résultats de la simulation sur Rokisim :	18
Conclusion :	19

Introduction :

La robotique est un domaine pluridisciplinaire englobant l'étude, la conception et la fabrication de robots, ou simplement de machines automatiques. Cette discipline combine des compétences techniques et des connaissances scientifiques en électronique, informatique et mécanique. Le terme "robotique" dérive du mot "robot", introduit par Isaac Asimov dans une de ses nouvelles intitulée "Runaround". Toutefois, c'est à travers la publication en mai 1941 de son récit de science-fiction "Menteur !" dans Astounding Science-Fiction que le public en a pris connaissance pour la première fois.

Un robot est un dispositif qui intègre la mécanique, l'électronique et l'informatique. Doté d'une structure corporelle comprenant un ou plusieurs membres, il est contrôlé par un ordinateur qui agit comme son cerveau. Le terme "robot" trouve son origine dans une pièce de théâtre tchèque où il désignait un ouvrier artificiel destiné au "travail forcé", appelé "robota" en tchèque.

Les robots sont conçus pour exécuter des tâches dangereuses, pénibles voire simples pour l'Homme. Leur avantage réside dans leur capacité à effectuer ces tâches avec une précision accrue et de manière autonome. Depuis les années 70, les robots sont devenus mobiles, équipés d'ordinateurs embarqués, de caméras et, surtout, capables de raisonnement.

À ce jour, le Japon occupe une position prééminente dans le domaine de la robotique industrielle. Le pays se distingue notamment par ses avancées dans les robots androïdes, conçus à l'image de l'être humain.

Un bras robotisé est un dispositif programmable dont le fonctionnement et le comportement imitent ceux d'un bras humain. Les différentes parties de ce robot sont articulées et reliées entre elles pour permettre des mouvements de rotation et de translation.

À son extrémité se trouve la main robotique, prenant la forme d'une pince, d'une ventouse ou d'une griffe selon les tâches à accomplir. Ces tâches peuvent inclure la fixation, le déplacement de marchandises, le prélèvement d'objets (picking) ou l'assemblage de pièces.

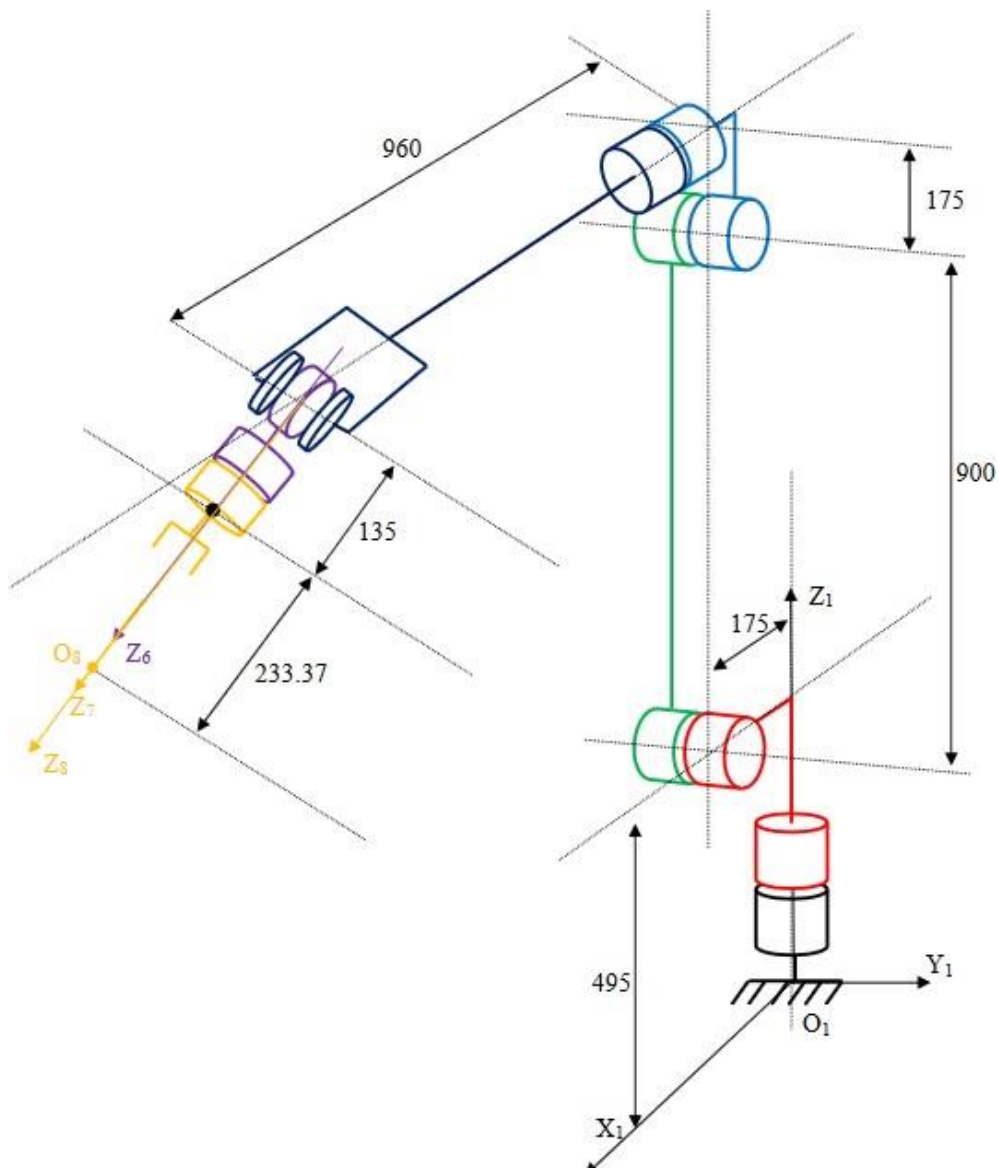
Chapitre 1 : Présentation du projet

1. Description du projet :

Dans cette étude, notre objectif est de calculer préalablement les positions articulaires nécessaires pour saisir des blocs à l'aide de la pince (outil IRB1600_Pince.tool) du robot (ABB_IRB4600_205_45), situés dans son espace de travail.

2. Modèle géométrique direct :

Schéma cinématique spatiale du robot ABB_IRB4600_205_45 :



Nous allons construire le tableau D-H (Denavit-Hartenberg) :

Bras à Bras	Ri à Ri+1	Liaison Li(Zi)	Rot(Θ_i , Zi)	Trans (di, Zi)	Trans(ai, Xi+1)	Rot(α_i , Xi+1)
			Θ_i	di	ai	α_i
0 à 1	R1 à R2	L1(Z1)	Θ_1	495	175	$-\frac{\pi}{2}$
1 à 2	R2 à R3	L2(Z2)	Θ_2	0	900	0
2 à 3	R3 à R4	L3(Z3)	Θ_3	0	175	$-\frac{\pi}{2}$
3 à 4	R4 à R5	L4(Z4)	Θ_4	960	0	$\frac{\pi}{2}$
4 à 5	R5 à R6	L5(Z5)	Θ_5	0	0	$-\frac{\pi}{2}$
5 à 6	R6 à R7	L6(Z6)	Θ_6	135+233.37	0	0

3. Les équations du modèles géométrique direct et sa validation numérique :

Matrice de rotation Rot (Θ_i , Zi) :

```
Rz.m × +
/MATLAB Drive/Rz.m
1 function matrice = Rz(teta)
2   c=cos(teta);
3   s=sin(teta);
4   matrice=[c -s 0 0; s c 0 0; 0 0 1 0; 0 0 0 1];
```

Matrice de translation Trans (di, Zi) :

```
Tz.m × Rz.m × +
/MATLAB Drive/Tz.m
1 function matrice = Tz(d)
2
3   matrice=[1 0 0 0; 0 1 0 0; 0 0 1 d; 0 0 0 1];
```

Matrice de translation Trans (ai, Xi+1) :

```
Tz.m × Rz.m × Tx.m × +
/MATLAB Drive/Tx.m
1 function matrice = Tx(a)
2
3   matrice=[1 0 0 a; 0 1 0 0; 0 0 1 0; 0 0 0 1];
```

Matrice de rotation Rot (α_i , X_{i+1}) :

```
Tz.m x  Rz.m x  Tx.m x  Rx.m x  +
/MATLAB Drive/Rx.m
1 function matrice = Rx(teta)
2   c=cos(teta);
3   s=sin(teta);
4   matrice=[1 0 0 0; 0 c -s 0; 0 s c 0; 0 0 0 1];
```

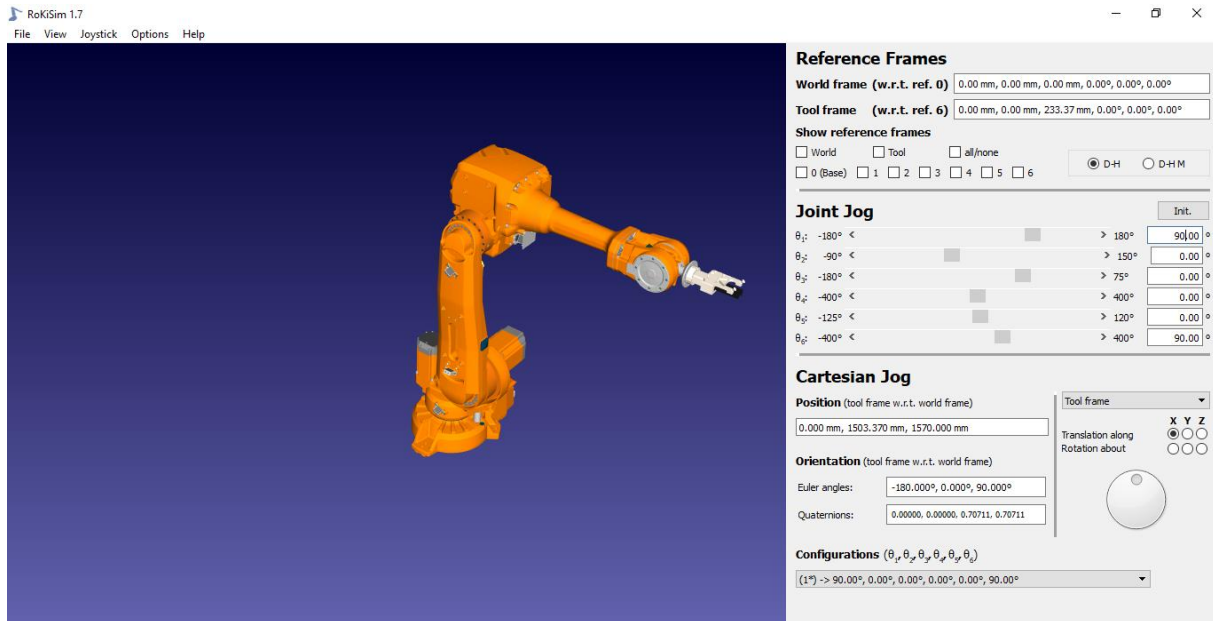
Calcul de la matrice de transformation par la méthode du modèle géométrique direct :

```
modellgd.m x  +
/MATLAB Drive/modellgd.m
1 function TG = modellgd(~)
2   teta=[1.2;1.5;0;1.4;0;1];
3
4   table=[ teta(1) 495 175 -pi/2;
5           teta(2)-pi/2 0 900 0;
6           teta(3) 0 175 -pi/2;teta(4) 960 0 pi/2;
7           teta(5) 0 0 -pi/2;
8           teta(6)-pi 368.37 0 0];
9   TG=eye(4,4);
10  for i=1:length(table(:,1))
11     TE=Rz(table(i,1))*Tz(table(i,2))*Tx(table(i,3))*Rx(table(i,4));
12     TG=TG*TE;
13  end
14
```

Pour teta = [1.2;1.5;0;1.4;0;1]; on trouve les résultats suivants :

```
Command Window
New to MATLAB? See resources for Getting Started.
modellgd
ans =
1.0e+03 *
-0.0004    0.0009    0.0000    0.4860
0.0009    0.0004    0.0001    1.2501
0.0001    0.0000   -0.0010   -0.7540
0         0         0         0.0010
```

4. La position de référence du Robot :



5. Les équations de la matrice Jacobéenne :

```

modellgd.m x  jacobienne.m x  jacobienne.m x  Tx.m x  Tz.m x  Rz.m x  Rx.m x  +
/MATLAB Drive/jacobienne.m

1 function Ja=jacobienne (teta)
2 teta=[1.2;1.5;0;1.4;0;1];
3 table=[teta(1) 495 175 -pi/2;
4         teta(2)-pi/2 0 900 0;
5         teta(3) 0 175 -pi/2;
6         teta(4) 960 0 pi/2;
7         teta(5) 0 0 -pi/2;
8         teta(6)-pi 368.37 0 0];
9
10 T1=Rz(table(1,1))*Tz(table(1,2))*Tx (table(1,3))*Rx (table(1,4) );
11 T2=Rz(table(2,1))*Tz(table (2,2))*Tx (table (2,3))*Rx(table(2,4) );
12 T3=Rz(table(3, 1))*Tz(table(3,2))*Tx (table (3,3))*Rx (table (3,4));
13 T4=Rz(table(4,1))*Tz(table(4,2))*Tx(table(4,3))*Rx(table(4,4));
14 T5=Rz(table(5,1))*Tz(table(5,2))*Tx (table (5,3))*Rx(table(5,4));
15 T6=Rz(table(6,1))*Tz(table(6,2))*Tx(table(6,3))*Rx(table(6,4));
16
17 R1=T1(1:3,1:3) ;
18 R2=T2 (1:3,1:3) ;
19 R3=T3 (1:3, 1:3) ;
20 R4=T4 (1:3, 1:3) ;
21 R5=T5 (1:3,1:3) ;
22 R6=T6 (1:3,1:3) ;

```



```

23 R11=R1;
24 R12=R11*R2;
25 R13=R12*R3;
26 R14=R13*R4;
27 R15=R14*R5;
28 R16=R15*R6;
29 k=[0 0 1]';
30
31 e1=k;
32 e2=R11*k;
33 e3=R12*k;
34 e4=R13*k;
35 e5=R14*k;
36 e6=R15*k;
37 P6=T6(1:3,4);
38 P5=T5 (1:3,4);
39 P4=T4 (1:3,4);
40 P3=T3 (1:3,4);
41 P2=T2 (1:3,4);
42 P1=T1 (1:3,4);
43 r6p=P6;
44

```

```

45 r5p=P5+R5*r6p;
46 r4p=P4+R4*r5p;
47 r3p=P3+R3*r4p;
48 r2p=P2+R2*r3p;
49 r1p=P1+R1*r2p;
50
51 r6=R15*r6p;
52 r5=R14*r5p; r4=R13*r4p;
53 r3=R12*r3p; r2=R11*r2p;
54 r1=r1p;
55
56 Ja(:,1)=[e1; cross(e1, r1)];
57 Ja(:,2)= [e2; cross(e2, r2)];
58 Ja(:,3)=[e3; cross(e3, r3)];
59 Ja(:,4)=[e4; cross(e4, r4)];
60 Ja(:,5)= [e5; cross(e5, r5)];
61 Ja(:,6)= [e6; cross(e6,r6)];

```

Pour teta = [1.2;1.5;0;1.4;0;1]; on trouve les résultats suivant :

```
>> jacobienne
ans =
1.0e+03 *
    0    -0.0009   -0.0009    0.0000    0.0002    0.0000
    0    0.0004    0.0004    0.0001    0.0010    0.0001
    0.0010    0.0000    0.0000   -0.0010    0.0001   -0.0010
   -1.2501   -0.4526   -0.4757    0.0000   -0.3610     0
    0.4860   -1.1641   -1.2235     0     0.0733     0
    0   -1.1663   -0.2685     0   -0.0044    0.0000
```

On a utilisé cette algorithme depuis le cours :

Algorithme 1 : calcul de la matrice jacobienne

1. Paramètres D-H du robot
2. Construire les matrices de transformation homogène T_i ; (position et rotation du repère F_{i+1} par rapport F_i) ; $i = 1 : 6$ tel que $T_i = \begin{bmatrix} R_i & p_i \\ 0 & 1 \end{bmatrix}$
3. Construire les matrices de rotation R_{1i} ; $i = 1 : 6$ tel que $R_{11} = R_1$ et $R_{1i} = R_1 R_2 \dots R_i$
4. Exprimer les vecteurs k du repère F_i dans le repère de base F_1 :
 $e_1 = k = [0 \ 0 \ 1]^T$
et $e_i = R_{1i} k \quad i = 2 : 6$;
5. Construire les vecteurs positions r_{ip}
(Position de la fin de l'effecteur exprimée dans le repère F_i) ; tel que :
 $r_{6p} = p_6$
 $r_{5p} = p_5 + R_5 r_{6p}$
 $r_{ip} = p_i + R_i r_{(i+1)p} \quad i = 4, 3, 2, 1$
6. Exprimer les vecteurs positions r_{ip} dans le repère de base F_1 ; tel que :
 $r_1 = r_{1p}$
 $r_2 = R_{11} r_{2p}$
 $r_i = R_{1(i-1)} r_{ip} \quad i = 3, 4, 5, 6$
7. Construire la matrice Jacobienne J (6x6) à partir des 6 matrices colonnes J_i (6x1) ; tel que :
 $J_i = \begin{bmatrix} e_i \\ e_i \times r_i \end{bmatrix}$
 $J = [J_1 \quad \dots \quad J_6]$

6. La validation numérique de la matrice jacobéenne :

```
>> script

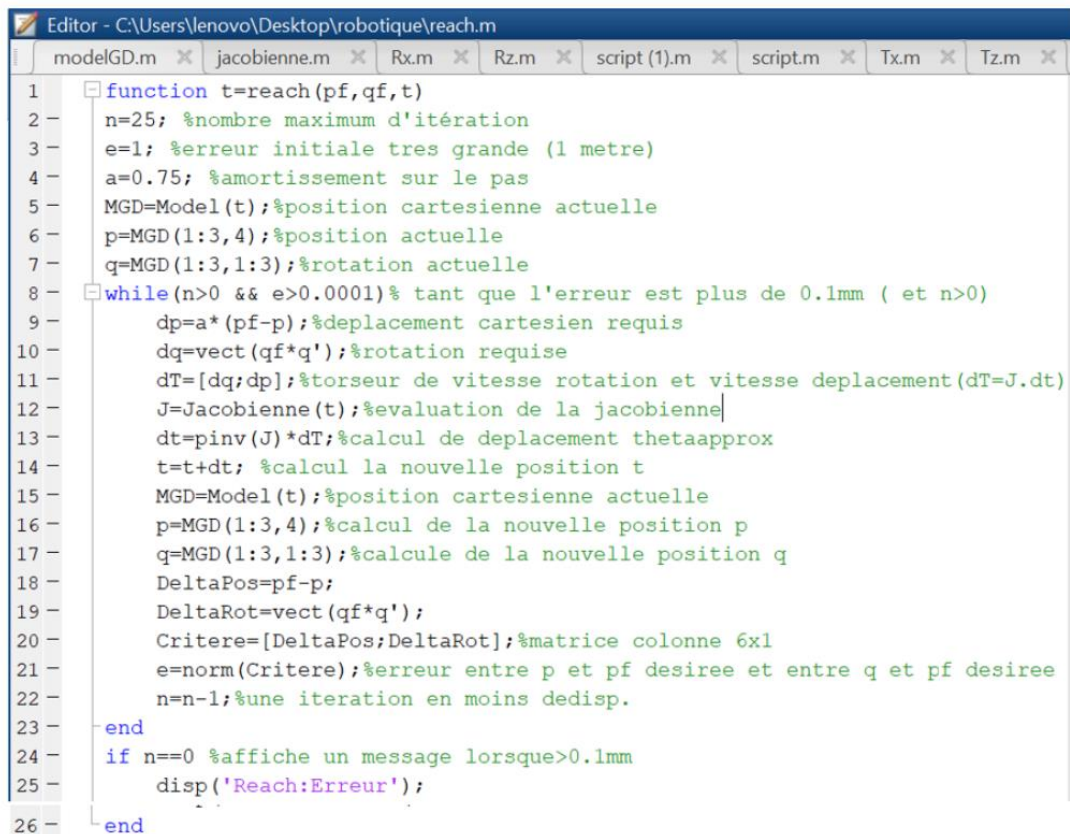
Ja =

    1.0e+03 *

         0   -0.0010   -0.0010         0   -0.0010         0
         0    0.0000    0.0000    0.0010    0.0000    0.0010
    0.0010    0.0000    0.0000   -0.0000    0.0000   -0.0000
   -1.5034   -0.0000   -0.0000         0   -0.0000         0
    0.0000    1.0750    0.1750         0   -0.0000         0
         0   -1.3284   -1.3284         0   -0.3684         0
```

7. Méthode de commande cinématique :

Il est nécessaire de commencer par définir la fonction "reach", qui permet de déterminer les coordonnées articulaires d'un point cible à partir d'un point de départ connu.



```
Editor - C:\Users\lenovo\Desktop\robotique\reach.m
modelGD.m x jacobienne.m x Rx.m x Rz.m x script(1).m x script.m x Tx.m x Tz.m x

1  function t=reach(pf,qf,t)
2  n=25; %nombre maximum d'itération
3  e=1; %erreur initiale tres grande (1 metre)
4  a=0.75; %amortissement sur le pas
5  MGD=Model(t);%position cartesienne actuelle
6  p=MGD(1:3,4);%position actuelle
7  q=MGD(1:3,1:3);%rotation actuelle
8  while(n>0 && e>0.0001)% tant que l'erreur est plus de 0.1mm ( et n>0)
9      dp=a*(pf-p);%deplacement cartesien requis
10     dq=vect(qf*q');%rotation requise
11     dT=[dq;dp];%torseur de vitesse rotation et vitesse deplacement(dT=J.dt)
12     J=Jacobienne(t);%evaluation de la jacobienne
13     dt=pinv(J)*dT;%calcul de deplacement thetaapprox
14     t=t+dt; %calcul la nouvelle position t
15     MGD=Model(t);%position cartesienne actuelle
16     p=MGD(1:3,4);%calcul de la nouvelle position p
17     q=MGD(1:3,1:3);%calcul de la nouvelle position q
18     DeltaPos=pf-p;
19     DeltaRot=vect(qf*q');
20     Critere=[DeltaPos;DeltaRot];%matrice colonne 6x1
21     e=norm(Critere);%erreur entre p et pf desiree et entre q et pf desiree
22     n=n-1;%une iteration en moins dedisp.
23 end
24 if n==0 %affiche un message lorsque>0.1mm
25     disp('Reach:Erreur');
26 end
```

La fonction vect :

```
/MATLAB Drive/mat_vect.m
1 function mat_vect(A)
2     mat=.5*[A(3,2) -A(2,3)
3             A(1,3) -A(3,1)
4             A(2,1) -A(1,2)];
5 end
```

La fonction trajectoire :

```
/MATLAB Drive/trajectoire.m
1 clear all
2 clc
3
4 P1=[0;0;0];
5 P2=[-200;1400;500];
6 P3=[0;1500;500];
7 P4=[200;1400;500];
8
9 pas=0.04; % Pas de simulation (seconde)
10 T = 5; % Durée pour parcourir le segment (seconde)
11 t = [0:pas:T]; % Discrétisation du temps
12 s = (t/T)-sin(2*pi*t/T)/(2*pi); % Discrétisation de l'espace
13 figure(1)
14 plot(t,s,'rs')
15 xlabel('temps(s)');
16 ylabel('Discrétisation de l'espace ');
17 grid on
18
19 P11 = P1*ones(size(s)) + (P2-P1)*s; % Discrétisation du segment
20 P22 = P2*ones(size(s)) + (P3-P2)*s;
21 P33 = P3*ones(size(s)) + (P4-P3)*s;
22 P44 = P4*ones(size(s)) + (P4-P3)*s;
23 P55 = P4*ones(size(s)) + (P3-P4)*s;
24 P66 = P3*ones(size(s)) + (P4-P3)*s;
25 P77 = P4*ones(size(s)) + (P1-P4)*s;
26 figure(2)
27 plot3(P11(1,:),P11(2,:),P11(3:,:), 'r*')
28 hold on;
29 plot3(P22(1,:),P22(2,:),P22(3:,:), 'b')
30 hold on;
31 plot3(P33(1,:),P33(2,:),P33(3:,:), 'k')
32 hold on;
33 plot3(P44(1,:),P44(2,:),P44(3:,:), 'g')
34 hold on;
35 plot3(P55(1,:),P55(2,:),P55(3:,:), 'y')
36 hold on;
37 plot3(P66(1,:),P66(2,:),P66(3:,:), 'm+')
38 hold on;
39 plot3(P77(1,:),P77(2,:),P77(3:,:), 'c*')
40 hold on;
41 axis([-200,500,0,1500,-500,700]);
42 xlabel('x');
43 ylabel('y ');
44 zlabel('z');
45 grid on
46
47 % Vitesse de la fin d'effecteur aus points du segment [P1,P2]
48 v1(1)=0;
49 v1(1)=0;
50 v2(1)=0;
51 v3(1)=0;
52 v4(1)=0;
53 v5(1)=0;
```

```

54     v6(1)=0;
55     v7(1)=0;
56
57     for i=2:length(s)-1
58         v1(i)=norm(P11(:,i+1)-P11(:,i))/pas;
59         v2(i)=norm(P22(:,i+1)-P22(:,i))/pas;
60         v3(i)=norm(P33(:,i+1)-P33(:,i))/pas;
61         v4(i)=norm(P44(:,i+1)-P44(:,i))/pas;
62         v5(i)=norm(P55(:,i+1)-P55(:,i))/pas;
63         v6(i)=norm(P66(:,i+1)-P66(:,i))/pas;
64         v7(i)=norm(P77(:,i+1)-P77(:,i))/pas;
65         i=i+1;
66     end
67
68     v1(length(v1)+1)=0;
69     v2(length(v2)+1)=0;
70     v3(length(v3)+1)=0;
71     v4(length(v4)+1)=0;
72     v5(length(v5)+1)=0;
73     v6(length(v6)+1)=0;
74     v7(length(v7)+1)=0;
75     v5(length(v5)+1)=0;
76     v6(length(v6)+1)=0;
77     v7(length(v7)+1)=0;
78
79     figure(3)
80     plot(t,v1,'r*')
81     hold on;
82     plot(t,v2,'b')
83
84     hold on;
85     plot(t,v3,'k')
86     hold on;
87     plot(t,v4,'g')
88     hold on;
89     plot(t,v5,'yo')
90     hold on;
91     plot(t,v6,'m*')
92     hold on;
93     plot(t,v7,'c')
94     hold on;
95     xlabel('temps(s)');
96     ylabel('vitesse (mm/s)');
97     grid on
98
99     % acceleration de la fin d'effecteur aux points du segment [P1,P2]
100    % -----
101    a1(1) = 0;
102    a2(1) = 0;
103    a3(1) = 0;
104    a4(1) = 0;
105    a5(1) = 0;
106    a6(1) = 0;
107    a7(1) = 0;
108
109    for i=2:length(s)-1
110        a1(i)=(v1(i+1)-v1(i))/pas;
111        a2(i)=(v2(i+1)-v2(i))/pas;

```

```

111         a3(i)=(v3(i+1)-v3(i))/pas;
112         a4(i)=(v4(i+1)-v4(i))/pas;
113         a5(i)=(v5(i+1)-v5(i))/pas;
114         a6(i)=(v6(i+1)-v6(i))/pas;
115         a7(i)=(v7(i+1)-v7(i))/pas;
116         i=i+1;
117     end
118
119     a1(length(a1)+1)=0;
120     a2(length(a2)+1)=0;
121     a3(length(a3)+1)=0;
122     a4(length(a4)+1)=0;
123     a5(length(a5)+1)=0;
124     a6(length(a6)+1)=0;
125     a7(length(a7)+1)=0;
126     a5(length(a5)+1) = 0;
127     a6(length(a6)+1) = 0;
128     a7(length(a7)+1) = 0;
129     figure(4)
130     plot(t, a1, 'r*')
131     hold on;
132     plot(t, a2, 'b')
133     hold on;
134     plot(t, a3, 'k')
135     hold on;
136     plot(t, a4, 'g')
137     hold on;
138     plot(t, a5, 'yo')
139     hold on;
140     plot(t, a6, 'm*')
141     hold on;
142     plot(t, a7, 'c')
143     hold on;
144     xlabel('temps(s)');
145     ylabel('accélération (mm/s^2)');
146     grid on
147

```

8. Évaluation de la précision du trajet et de la dextérité :

La précision de la trajectoire et la dextérité sont mesurées à l'aide de la fonction "reach", où l'on définit une variable e représentant l'erreur, soit la différence entre la position actuelle et la position souhaitée. Cette fonction inclut une boucle qui se poursuit jusqu'à ce que l'erreur atteigne un seuil spécifique, ici fixé à 0.0001, qui sert de critère de convergence pour garantir la précision du trajet.

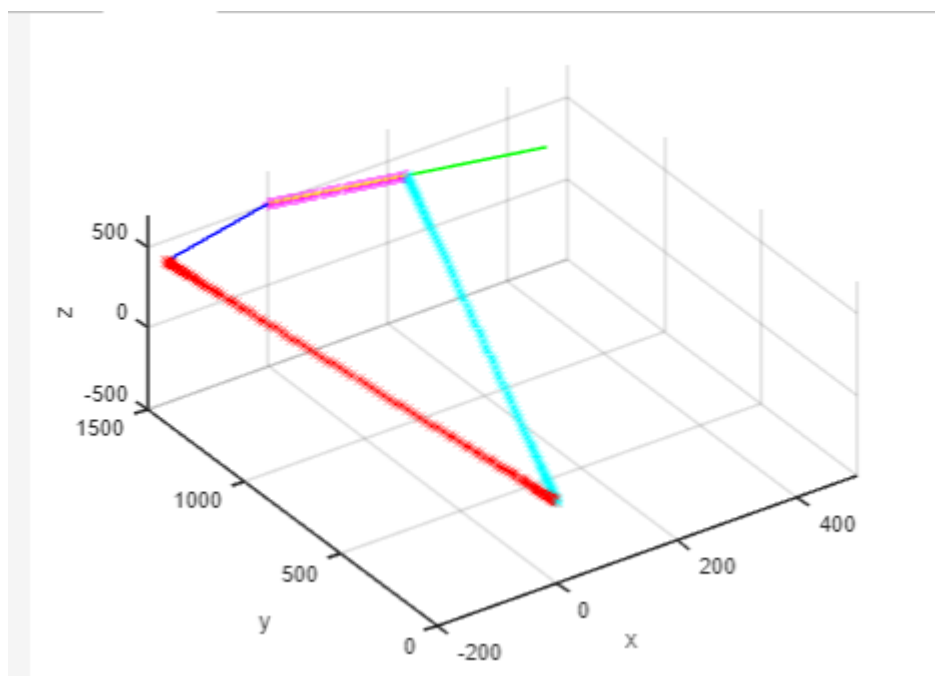
9. Graphique des positions articulaires pour la trajectoire :

Ce graphique est généré par la fonction "trajectoire", qui trace tous les points parcourus par l'extrémité de l'effecteur au cours de son déplacement, incluant sa vitesse et son accélération. Les actions réalisées sont les suivantes :

- Prendre le bloc bleu et le placer sur le bloc jaune ;
- Prendre les deux blocs et les placer sur le bloc rouge ;
- Déplacer les trois blocs à l'emplacement "bleu1T" ;
- Déplacer le bloc bleu à "jaune1T" ;
- Déplacer le bloc jaune à "rouge1T" ;

Retourner à la position de départ.

Les points :

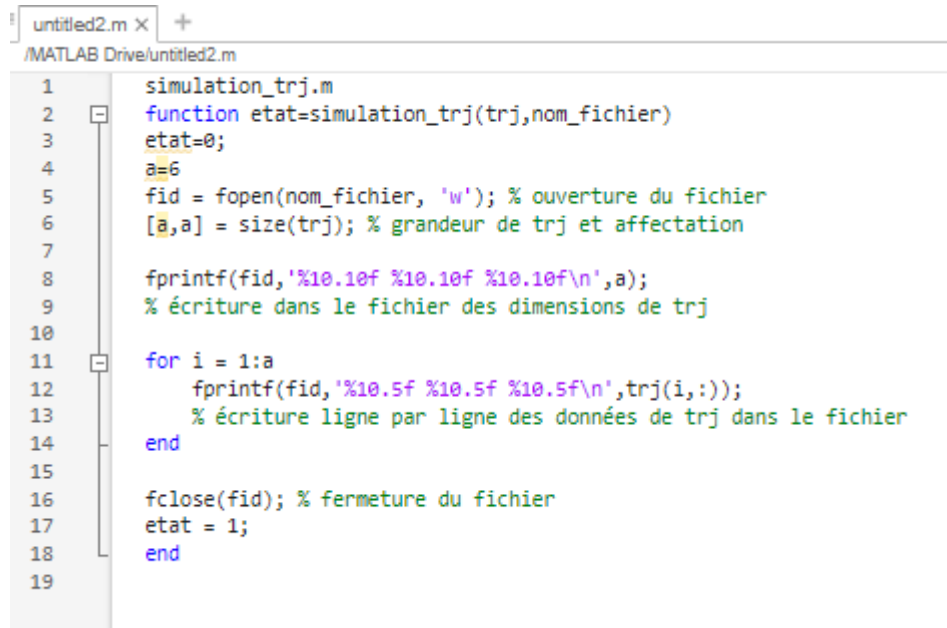


Chapitre 2 : Simulation sur RokiSim

Cette section traite de la phase finale de notre projet, où nous tenterons de simuler notre robot et de présenter les résultats obtenus.

1. Fonction `simulation_trj.m` :

Cette fonction nous permet de générer un fichier texte contenant les coordonnées articulaires.



```
1 simulation_trj.m
2 function etat=simulation_trj(trj,nom_fichier)
3 etat=0;
4 a=6;
5 fid = fopen(nom_fichier, 'w'); % ouverture du fichier
6 [a,a] = size(trj); % grandeur de trj et affectation
7
8 fprintf(fid,'%10.10f %10.10f %10.10f\n',a);
9 % écriture dans le fichier des dimensions de trj
10
11 for i = 1:a
12     fprintf(fid,'%10.5f %10.5f %10.5f\n',trj(i,:));
13     % écriture ligne par ligne des données de trj dans le fichier
14 end
15
16 fclose(fid); % fermeture du fichier
17 etat = 1;
18 end
19
```

2. Simulation des trajectoires :

Pour obtenir la représentation 3D du manipulateur de notre robot en fonction des positions articulaires décrites dans le fichier `position.tjr`, ainsi que les 3 blocs à saisir avec les pinces du manipulateur, il est nécessaire de modifier les coordonnées pour les angles θ et l'état de la pince à chaque étape. Cette approche permet de trouver la nouvelle trajectoire et de l'ajouter aux précédentes, jusqu'à atteindre le dernier bloc.

Programme de commande avec l'état de la pince :


```

1 simulation\trj
2 clear all
3 clc
4
5 % tetav = [pi/2 0 0 pi/2]';
6 % etat0=[90 0 0 90]';
7
8 Tinit=[0 1503.37 1570];
9 Tbleu=[-200 -1500 -500];
10 Tjaune=[-200 1400 -500];
11 tetav=[90 200 300 -90]';
12 trouge=[2001.4 -10000 -500];
13
14 P1=trajectoire2points(Tinit,Tbleu,etat,tetav);
15 P2=trajectoire2points(tetav,Tinit,etat,S); couleur(2);
16 [(180/pi)*tetav']
17

```

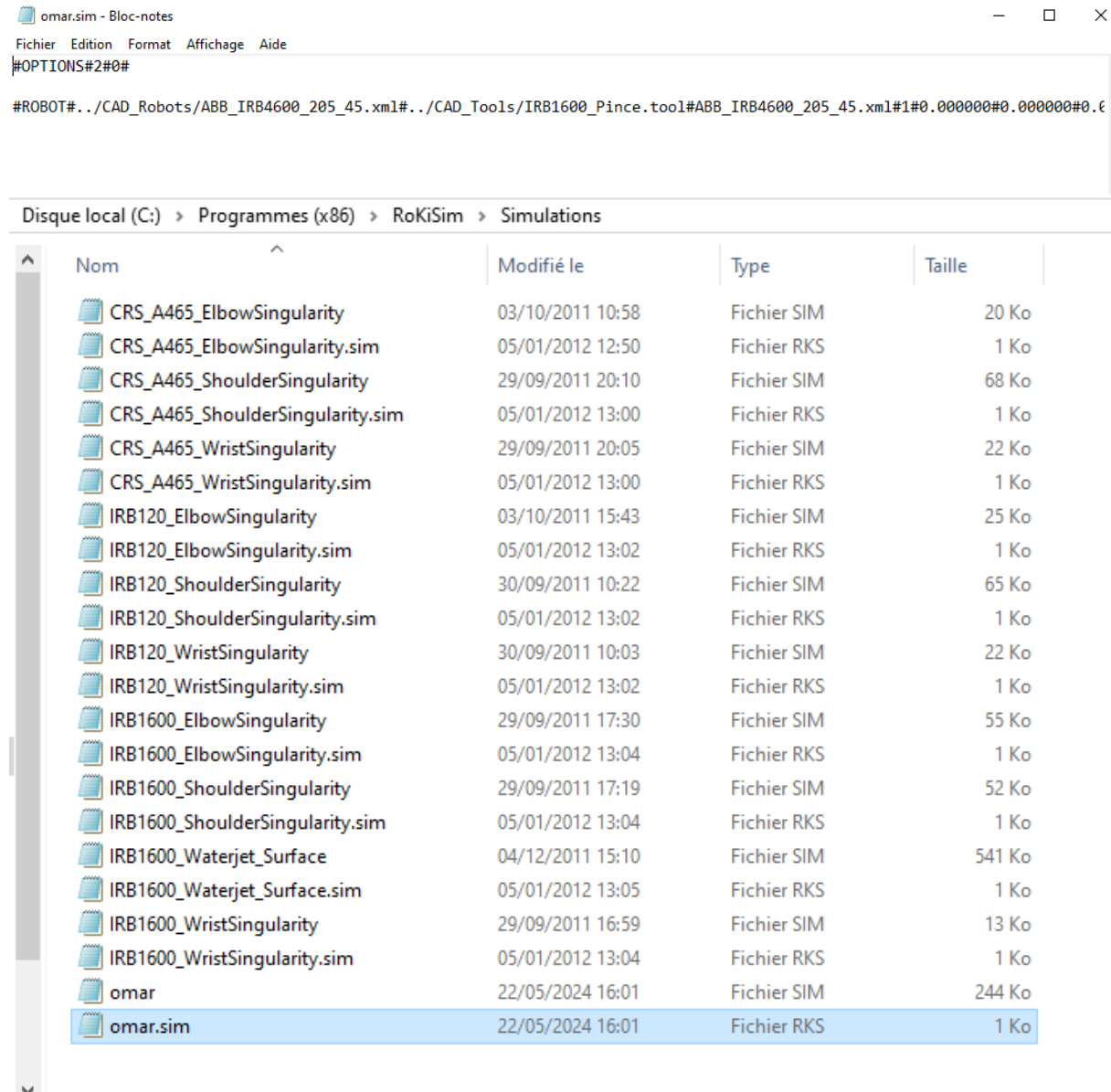
Et par conséquent, nous obtenons un fichier contenant plusieurs lignes, chacune représentant une coordonnée articulaire avec un angle différent :

position - Bloc-notes

Fichier	Edition	Format	Affichage	Aide			
90.0000	-0.0000	0.0000	-0.0000	-0.0000	90.0000	0	
90.0000	-0.0000	0.0000	11.3228	-0.0000	78.6772	0	
90.0000	-0.0001	0.0003	8.6418	-0.0002	81.3582	0	
90.0001	-0.0002	0.0009	8.9558	-0.0007	81.0442	0	
90.0003	-0.0005	0.0022	8.9758	-0.0017	81.0242	0	
90.0005	-0.0010	0.0044	8.9758	-0.0034	81.0242	0	
90.0009	-0.0017	0.0075	8.9759	-0.0059	81.0241	0	
90.0015	-0.0027	0.0119	8.9821	-0.0093	81.0179	0	
90.0022	-0.0041	0.0178	8.9823	-0.0139	81.0177	0	
90.0031	-0.0058	0.0254	8.9825	-0.0198	81.0175	0	
90.0042	-0.0080	0.0348	8.9843	-0.0271	81.0157	0	
90.0056	-0.0106	0.0462	8.9846	-0.0361	81.0154	0	
90.0073	-0.0137	0.0600	8.9849	-0.0468	81.0151	0	
90.0093	-0.0174	0.0762	8.9853	-0.0595	81.0147	0	
90.0116	-0.0218	0.0951	8.9858	-0.0742	81.0142	0	
90.0142	-0.0267	0.1168	8.9864	-0.0912	81.0137	0	
90.0173	-0.0324	0.1416	8.9874	-0.1106	81.0126	0	
90.0207	-0.0388	0.1697	8.9881	-0.1325	81.0119	0	
90.0245	-0.0459	0.2011	8.9889	-0.1571	81.0111	0	
90.0288	-0.0539	0.2362	8.9898	-0.1845	81.0103	0	
90.0336	-0.0627	0.2751	8.9908	-0.2150	81.0093	0	
90.0388	-0.0725	0.3179	8.9919	-0.2485	81.0082	0	
90.0446	-0.0831	0.3649	8.9931	-0.2853	81.0071	0	
90.0509	-0.0947	0.4162	8.9944	-0.3255	81.0058	0	
90.0577	-0.1073	0.4720	8.9958	-0.3693	81.0044	0	
90.0652	-0.1209	0.5324	8.9973	-0.4167	81.0029	0	
90.0732	-0.1355	0.5977	8.9991	-0.4679	81.0012	0	
90.0818	-0.1512	0.6679	9.0009	-0.5231	80.9995	0	
90.0911	-0.1680	0.7431	9.0028	-0.5823	80.9977	0	
90.1011	-0.1859	0.8237	9.0048	-0.6457	80.9958	0	
90.1117	-0.2050	0.9096	9.0070	-0.7134	80.9937	0	
90.1230	-0.2252	1.0010	9.0093	-0.7855	80.9915	0	
90.1350	-0.2465	1.0980	9.0118	-0.8622	80.9892	0	
90.1478	-0.2691	1.2009	9.0144	-0.9434	80.9868	0	
90.1613	-0.2928	1.3095	9.0171	-1.0295	80.9843	0	
90.1756	-0.3177	1.4242	9.0200	-1.1203	80.9817	0	

3. Simulation sur le logiciel Rokisim :

Pour effectuer la simulation sur Rokisim, nous avons d'abord dû copier les deux fichiers nommés respectivement `Robot.sim` (qui contient les coordonnées articulaires) et Robot.sim.rks, (qui contient le nom du robot utilisé et l'outil) dans le dossier de simulation des fichiers de Rokisim.



Pour visualiser la simulation, on met d'abord les positions de référence du robot ; on charge la simulation et on visualise.

4. Visualisation des résultats de la simulation sur Rokisim :

Pour voir la simulation, vous pouvez cliquer à ce lien :

https://www.canva.com/design/DAGF-KI-OXA/pgP0x5def8cdqoBF-mitVA/watch?utm_content=DAGF-KI-OXA&utm_campaign=designshare&utm_medium=link&utm_source=editor

Conclusion :

Ce projet sur la robotique industrielle a été une excellente opportunité pour nous, futurs ingénieurs, de découvrir et de nous familiariser avec de nouveaux logiciels, tels que Rokisim dans notre cas. Il nous a permis de mettre en pratique nos connaissances théoriques et tout ce que nous avons appris en cours.

Bien que la tâche que nous avons réalisée soit relativement simple par rapport à ce qui existe dans le monde industriel, cela ne nous empêche pas de continuer à développer nos compétences dans ce domaine. Au contraire, ce projet nous encourage à explorer davantage les aspects cachés et les défis de la robotique industrielle.