



# ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIER - MEKNÈS

PROJET DE FIN D'ÉLÉMENT

---

## Rapport Robotique Industrielle

---

**Élèves :**

Abdelouahab ZENZER  
Oussama SOUABI

**Encadrant :**

Abdelhak NAFI

**Jury :**

Abdelhak NAFI

Année Universitaire 2024-2025

## REMERCIEMENT

Nous tenons d'abord à exprimer notre profonde gratitude à notre cher professeur

**M. NAFI Abdelhak** pour son soutien inestimable et son encadrement. Votre expertise, vos conseils avisés et votre disponibilité constante ont été essentiels pour la réalisation de ce travail. Votre passion pour l'enseignement et votre dévouement envers vos étudiants sont une véritable source d'inspiration.

**Merci infiniment.**

## Introduction

La robotique industrielle, selon les normes de l'ISO, désigne un système automatisé, polyvalent et reprogrammable, contrôlé de manière précise sur trois axes ou plus. Ses principaux avantages résident dans sa vitesse d'exécution, sa précision et sa capacité à maintenir cette précision dans le temps. On envisageait il y a quelques années un développement significatif des robots parallèles pour une utilisation comme manipulateurs industriels.

Le projet actuel consiste à utiliser un logiciel appelé RokiSim pour commander un robot dans le déplacement de trois cubes. Ce logiciel permet de simuler les comportements et les configurations spécifiques du robot en créant un modèle géométrique direct, ce qui permet de calculer les positions précises de l'organe terminal en fonction des positions de ses articulations. En résumé, le modèle géométrique direct est un outil crucial pour déterminer comment un bras manipulateur se configure et fonctionne.

### Objectif :

Ce projet a pour but de calculer hors-ligne les positions articulaires permettant de saisir, avec le Pince (IRB1600\_Pince.tool) du robot (ABB\_IRB4600\_205\_45).

## I. Modèle géométrique directe :

### 1. Tableau DH :

Bras à bras	$R_i$ à $R_{i+1}$	Liaison $L_i(Z_i)$	rot $(Z_i, \theta_1)$	Trans $(Z_i, d_1)$	Trans $(X_{i+1}, a_1)$	rot $(X_{i+1}, a_1)$
0 à 1	$R_1$ à $R_2$	$L_1(Z_1)$	$\theta_1$	495	175	$-\frac{\pi}{2}$
1 à 2	$R_2$ à $R_3$	$L_2(Z_2)$	$\theta_2 - \frac{\pi}{2}$	0	900	0
2 à 3	$R_3$ à $R_4$	$L_3(Z_3)$	$\theta_3$	0	175	$-\frac{\pi}{2}$
3 à 4	$R_4$ à $R_5$	$L_4(Z_4)$	$\theta_4$	960	0	$\frac{\pi}{2}$
4 à 5	$R_5$ à $R_6$	$L_5(Z_5)$	$\theta_5$	0	0	$-\frac{\pi}{2}$
5 à 6	$R_6$ à $R_7$	$L_6(Z_6)$	$\theta_6$	135+233.37	0	0

## 2. Calcul des paramètres

Pour trouver les valeurs des **a, b, c, d, e, f** on utilise le logiciel Rokisim pour différentes positions comme le montre le tableau suivant :

$\Theta 1$	$\Theta 2$	$\Theta 3$	$\Theta 4$	$\Theta 5$	$\Theta 6$	
0	0	0	0	0	0	$X=f_x(a,b,c,d,e,f)=1270$ $Z=f_z(a,b,c,d,e,f)=1570$
0	90	-90	0	0	0	$X=f_x(a,b,c,d,e,f)=2170$ $Z=f_z(a,b,c,d,e,f)=670$
0	0	-90	0	0	0	$X=f_x(a,b,c,d,e,f)=0$ $Z=f_z(a,b,c,d,e,f)=2490$
0	0	-90	0	90	0	$X=f_x(a,b,c,d,e,f)=135$ $Z=f_z(a,b,c,d,e,f)=2355$

$$\begin{aligned}
 c+e+f &= 1270 \\
 a+d+b &= 1570 \\
 c+d+e+f &= 2170 \\
 a+d &= 670 \\
 a+d+e+f &= 2490 \\
 c-d &= 0 \quad c-d+f \\
 &= 135 \quad a+b+e \\
 &= 2355
 \end{aligned}$$

$$\text{O } a=495, b=900, c=d=175, e=960, f=135$$

Par la suite on utilise le logiciel de simulation Matlab pour afin de réaliser la tâche

**Fonction de modèle géométrique direct :**

```

function TG=ModelGD(teta)

table=[teta(1)          495          175 -pi/2 ;
        teta(2)-pi/2    0          900  0   ;
        teta(3)         0          175 -pi/2 ;
        teta(4)         960         0   pi/2 ;
        teta(5)         0          0   -pi/2 ;
        teta(6)-pi      135+233.37    0   0   ];

TG=eye(4,4);
for i=1:length(table(:,1))
    TE=Rz(table(i,1))*Tz(table(i,2))*Tx(table(i,3))*Rx(table(i,4));
    TG=TG*TE;
end
end

```

Avec les fonctions suivantes :

```

1  function Matrice = Tx(a)
2
3  Matrice=[ 1 0 0 a
4            0 1 0 0
5            0 0 1 0
6            0 0 0 1 ];
7  end

```

```

1  function Matrice = Tz(d)
2
3  Matrice=[ 1 0 0 0
4            0 1 0 0
5            0 0 1 d
6            0 0 0 1 ];
7  end

```

```

1  function Matrice = Rz(teta)
2
3  c=cos(teta);
4  s=sin(teta);
5
6  Matrice=[ c -s 0 0
7            s  c 0 0
8            0  0 1 0
9            0  0 0 1 ];
10 end

```

```

1  function Matrice = Rx(teta)
2
3  c=cos(teta);
4  s=sin(teta);
5
6  Matrice=[ 1 0 0 0
7            0 c -s 0
8            0 s  c 0
9            0 0 0 1 ];
10 end

```

## 1. Test du modèle :

La fonction suivante nous permet de vérifier les fonctions précédentes :

Pour chaque valeur de **thêta** la fonction donne **la position** et **l'orientation (angles d'Euler)** dans Matlab puis on compare avec les résultats du logiciel Rokisim.

```

function Resultat = TEST_MGD ( t )|
theta = t * (pi/180);
M = ModelGD(theta);
Position = (M(1:3,4))';
Orientation = AngleEuler_ZYX(M);
Resultat = [ Position ; Orientation];
end

```

Avec les fonctions utilisées sont définie comme suit :

```

function AngleEuler = AngleEuler_ZYX(MR)
R=MR;
alpha = atan2(R(3,2),R(3,3));
beta = atan2(-R(3,1),sqrt((R(1,1)^2+R(2,1)^2)));
gama = atan2(R(2,1),R(1,1));
if beta == pi/2
    gama = 0;
    alpha = atan2(R(1,2),R(2,2));
elseif beta == -pi/2
    gama = 0;
    alpha = -atan2(R(1,2),R(2,2));
end
AngleEuler = [gama,beta,alpha]*(180/pi);
end
|

```

On vérifie par des angles quelconques de  $\theta$  qu'on entre ainsi au Rokisim et il nous donne exactement les mêmes résultats de ceux de Matlab :

## 2. Résultats

- Pour la configuration : **90, 45, 0, 0, 0, 45**

```
>> teta = [ 90 45 0 0 0 45]; n = TEST_MGD(teta)
```

n =

1.0e+03 \*

0.0000	1.8744	0.3158
-0.1447	0.0300	0.1447

### Pilotage Cartésien

**Position** (réf. outil p.r. à réf. at

0.000 mm, 1874.439 mm, 315.840 mm

**Orientation** (réf. outil p.r. à réf. atelier)

Angles d't '36°, 30.000°, 144.736°

Quaternionic 5355, 0.85355, 0.35355

**Configura** ( $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ )

(1\*) -> 90.00°, 45.00°, 0.00°, 0.00°, 0.02°, 45.00° -

Référentiel outil

Translation le

Rotation auto

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

○ Pour la configuration : 0, 90, -90, 0, 0, 0

```
>> teta = [ 0 90 -90 0 0 0]; n = TEST_MGD(teta)
```

n =

1.0e+03 \*

2.4034	0.0000	0.6700
0.1084	0.0900	0.1084

### Pilotage Cartésien

**Position** (réf. outil p.r. à réf. at

2403.370 mm, 0.000 mm, 670.000 mm

**Orientation** (réf. outil p.r. à réf. atelier)

Angles d't 0.000°, 90.000°, 0.000°

Quaternionic 0000, 0.70711, 0.00000

**Configura** ( $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ )

(3\*) -> 0.00°, 90.00°, -90.00°, 0.00°, 0.00°, 0.00° -

Référentiel outil

Translation le

Rotation auto

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

○ Pour la configuration : 0, 45, 0, 45, 0, 0

```
>> teta = [ 0 45 0 45 0 0]; n = TEST_MGD(teta)
```

n =

1.0e+03 \*

1.8744	0.0000	0.3158
0.1253	0.0300	0.1447

### Pilotage Cartésien

**Position** (réf. outil p.r. à réf. at

1874.439 mm, 0.000 mm, 315.840 mm

**Orientation** (réf. outil p.r. à réf. atelier)

Angles d't 125.264°, 30.000°, 144

Quaternionic 5355, 0.85355, 0.14645

**Configura** ( $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ )

(1\*) -> 0.00°, 45.00°, 0.00°, 45.00°, 0.02°, 0.00° -

Référentiel outil

Translation le

Rotation auto

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

XYZ

## II. Modèle géométrique inverse :

### 1. Algorithme de la matrice jacobienne :



**Algorithme 2-2 : Méthode de calcul des coordonnées articulaires en un point à partir d'un point initial connu (fonction reach.m)**

1. Données
  - $p_f$   $R_f$  : position et orientation désirées
  - $\theta$  : coordonnées articulaire initiales
  - n : nombre maximum d'itérations
  - e : erreur initiale (choisie grande pour amorcer le calcul)
  - e0 : critère de convergence requise
  - $\alpha$  : amortissement du pas
2. Calculer la position p et l'orientation R de la fin de l'effecteur à l'état initiale par le MGD
3. Calculer  $dp = \alpha (p_f - p)$  et
 
$$dq = \text{vect}(R_f R^T) \text{ Poser } dP \equiv \begin{bmatrix} dq \\ dp \end{bmatrix}$$
4. Calculer la matrice Jacobienne  $J(\theta)$
5. Calculer  $d\theta = J^+ dP$
6. Calculer la nouvelle configuration  $\theta = \theta + d\theta$
7. Calculer la nouvelle position p et l'orientation R de la fin de l'effecteur pour la nouvelle  $\theta$  en utilisant le MGD
8. Calculer la nouvelle erreur  $dp = p_f - p$  et  $dq = \text{vect}(R_f R^T)$
9. Calculer l'erreur  $e = \text{norm} \left( \begin{bmatrix} dq \\ dp \end{bmatrix} \right)$  et décrémente n
10. Si  $n > 0$  et  $e > e0$  recommencer le calcul à partir de l'étape 3.  
Sinon fin de calcul et retour des nouvelles valeurs de  $\theta$

## 2. Validation de l'algorithme :

Tester l'algorithme de la matrice jacobienne revient tester la fonction **REACH** qui calcule les cordonnés articulaire d'un point donné, on calcul les cordonnés d'un point avec le modèle géométrique direct puis on utilise la fonction **REACH**, il faut trouver les même cordonnés.

```

1  function tet = TEST_MGI( t,ti )
2
3  -   teta = t * (pi/180);
4  -   M = ModelGD(teta);
5  -   a = M(1:3,4);
6  -   b = M(1:3,1:3);
7  -   tetai = ti * (pi/180) ;
8  -   tet = reach(a, b, tetai') * (180/pi);
9
10 - end

```

La fonction utilisées REACH est la suivante



```

1  function t = reach(pf,qf,t)
2  % pf position final (desir?e) 3x1
3  % qf rotation final (desir?e) 3x3
4  % t matrice colonne 6x1 contenant les teta I de la position actuelle
5  %-----
6  n=50; % Nombre maximum d'it?ration
7  e=1; % Erreur initiale tres grande (1 metre)
8  a=0.75; % Amortissement sur le pas
9  MGD=ModelGD(t); % Position Cart?sienne actuelle
10 p=MGD(1:3,4); %position actuelle
11 q=MGD(1:3,1:3); % rotation actuelle
12 while(n>0 && e>0.0001) % Tant que l'erreur est plus de 0.1mm (et n>0)
13 dp=a*(pf-p); % D?placement Cartesien requis
14 dq= vect(qf* q'); % rotation requise
15 dT=[dq;dp]; % torseur de vitesse rotation et vitesse de deplacement (dT = J . dt)
16 J=Jacobienne(t); % ?valuation de la Jacobienne
17 dt=pinv(J)*dT; % Calcul du d?placement thetaapprox.
18 t=t+dt; % Calcul la nouvelle position t
19 MGD=ModelGD(t); % Position Cart?sienne actuelle
20 p=MGD(1:3,4); %Calcul de la nouvelle position p
21 q=MGD(1:3,1:3); % Calcul de la nouvelle rotation q
22 DeltaPos=pf-p;
23 DeltaRot=vect(qf *q');
24 Critere=[DeltaPos;DeltaRot]; % Matrice colonne 6x1
25 e=norm(Critere); % Erreur entre p et pf d?sir?e et entre q et qf desiree
26 n=n-1; % Une iteration en moins dedisp.
27 end
28 if n==0 % Affiche un message lorsque>0.1 mm
29 disp('Reach: Erreur ');
30 e
31 end
32 end

```

```

1  function Ja = Jacobienne(teta)
2
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    135+233.37 0    0  ] ;
9
10 -   %-----
11 -   T1=TranElem(table,1);
12 -   T2=TranElem(table,2);
13 -   T3=TranElem(table,3);
14 -   T4=TranElem(table,4);
15 -   T5=TranElem(table,5);
16 -   T6=TranElem(table,6);
17
18 -   R1=T1(1:3,1:3);
19 -   R2=T2(1:3,1:3);
20 -   R3=T3(1:3,1:3);
21 -   R4=T4(1:3,1:3);
22 -   R5=T5(1:3,1:3);
23 -   R6=T6(1:3,1:3);
24
25 -   R12=R1;
26 -   R13=R12*R2;
27 -   R14=R13*R3;
28 -   R15=R14*R4;
29 -   R16=R15*R5;
30 -   k=[0 0 1]';

```

```

37  %-----
38 -   P6=T6(1:3,4);
39 -   P5=T5(1:3,4);
40 -   P4=T4(1:3,4);
41 -   P3=T3(1:3,4);
42 -   P2=T2(1:3,4);
43 -   P1=T1(1:3,4);
44  %-----
45 -   r6p=P6;
46 -   r5p=P5+R5*r6p;
47 -   r4p=P4+R4*r5p;
48 -   r3p=P3+R3*r4p;
49 -   r2p=P2+R2*r3p;
50 -   r1p=P1+R1*r2p;
51  %-----
52 -   r1=r1p;
53 -   r2=R12*r2p;
54 -   r3=R13*r3p;
55 -   r4=R14*r4p;
56 -   r5=R15*r5p;
57 -   r6=R16*r6p;
58  %-----
59 -   Ja(:,1)=[e1;cross(e1,r1)];
60 -   Ja(:,2)=[e2;cross(e2,r2)];
61 -   Ja(:,3)=[e3;cross(e3,r3)];
62 -   Ja(:,4)=[e4;cross(e4,r4)];
63 -   Ja(:,5)=[e5;cross(e5,r5)];
64 -   Ja(:,6)=[e6;cross(e6,r6)];
65  %-----
66 -   end

```

```

1  function mat=vect(A)
2
3  -   mat=.5*[A(3,2)-A(2,3)
4             A(1,3)-A(3,1)
5             A(2,1)-A(1,2)];
6
7  -   end

```

```

1  function T = TranElem( MGD, i )
2
3  -   T=Rz(MGD(i,1))*Tz(MGD(i,2))*Tx(MGD(i,3))*Rx(MGD(i,4));
4
5  -   end

```

### 3. Résultats obtenus :

```
>> tetai = [ 0 0 0 0 0 0 ];
>> teta = [ 5 7 10 8 10 5];
>> t = TEST_MGI(teta,tetai)
```

```
t =
```

```
5.0000
7.0000
10.0000
8.0000
10.0000
5.0000
```

```
>> Jacobienne
```

```
ans =
```

```
0 -0.8011 -0.0531 -0.6722 -0.6722 -0.6722
0 0.0000 -0.9978 -0.5441 -0.5441 -0.5441
1.0000 0.5985 0.0396 0.5021 0.5021 0.5021
724.3329 432.5617 -635.6531 0 0.6847 0
-893.8303 -1.5708 -0.1041 0.0000 -1.3179 0
0 579.0297 -853.5080 0.0000 -0.5115 0
```

#### 4. Commande cinématique

On utilise pour ce fait une fonction trajectoire qui a comme entrée l'angle initiale, les différents points choisis pour réaliser la tâche voulue, le pas et la durée, et il a comme sortie un fichier txt qui stock les différents cordonne articulaires et l'état de la pince.

##### 1. Les points :

Les points choisis (la 4 variable représente l'état de la fin d'effecteur : 0 ouvert et 1 sinon) :

```
[ -200 1400 520 0;
-200 1400 500 0;
-200 1400 530 1;
000 1500 530 1;
000 1500 520 1;
000 1500 500 0;
000 1500 530 1;
200 1400 530 1;
200 1400 520 1;
200 1400 500 0;
200 1400 510 1;
-200 1400 510 1;
-200 1400 500 1;
-200 1400 540 0;
-200 1400 550 1;
000 1500 550 1;
```

```

000 1500 500 1;
000 1500 530 0;
-200 1400 530 0;
-200 1400 520 0;
-200 1400 530 1;
200 1400 530 1;
200 1400 500 1;
200 1400 520 0;
000 1503 1570 0; ]

```

## 2. La fonction trajectoire :

```

1 - function e = Trajectoire(tetai,points,pas,duree)
2
3 -   trj = trajectoire2points(tetai*(pi/180),points(1,:), pas, duree);
4 -   for i=2:length(points(:,1))
5 -       trj = [ trj ; trajectoire2points(trj(126*(i-1),1:6)*(pi/180),points(i,:), pas, duree)];
6 -   end
7 -   e = enregistre_trj( trj,pas,duree, 'fichier_trj' );
8 - end

```

Avec les fonctions utilisées sont définie comme suit :

```

1 function trj=trajectoire2points(teta,posef,PAS,duree)
2 % teta : coordonnées articulaires du 1er point du segment
3 % posef : vect 1x4 comporte la position et l'état de la fin d'effecteur d'axes au 2ieme point du segment
4 MGD = ModelGD(teta);
5 p=MGD(1:3,4); % Approche du 1er point position p
6 Qd=[-1 0 0; 0 0 1; 0 1 0]; % Rotation d'axe de la fin d'effecteur d'axes au 2ieme point du segment
7 P1 = p; % 1er Point du segment
8 P2 = (posef(1,1:3))'; % 2e Point du segment
9 pas=PAS; % Pas de simulation (seconde)
10 T = duree; % Durée du segment (seconde)
11 iteration=(T/pas)+1; % Nombre d'itérations
12 t = [0:pas:T]; % Discrétisation du temps
13 s = (t/T)-sin(2*pi*t/T)/(2*pi); % Discrétisation de l'espace
14 P = P1*ones(size(s)) + (P2-P1)*s; % Discrétisation du segment
15 teta = teta';
16 for i=1:iteration % Calcul des positions
17 teta = (reach(P(:,i),Qd,teta));
18 trj(i,:)=[(teta * (180/pi))',posef(4)];
19 end
20
21 end

```

```

1 function etat = enregistre_trj( trj,pas,duree,nom_fichier )
2
3 etat=0;
4 [a,z]=size(trj); % Grandeur de trj
5 fid = fopen(nom_fichier,'w'); % Ouverture du fichier .trj
6
7 fprintf(fid,'pas          : ');
8 fprintf(fid,'%10.5f \n',pas); % Pas
9
10 fprintf(fid,'duree          : ');
11 fprintf(fid,'%10.5f \n',duree); % Pas
12
13 fprintf(fid,'nombre de ligne : ');
14 fprintf(fid,'%10.0f \n',a); % Nombre de ligne
15
16 fprintf(fid,'\n');
17 fprintf(fid,'teta1          teta2          teta3          teta4          teta5          teta6          etat\n');
18
19 for i=1:a % Angles du manipulateur
20     fprintf(fid,'%5.9f      %5.9f      %5.9f      %5.9f      %5.9f      %5.9f      %5.9f \n',trj(i,:));
21 end
22 fclose(fid); % Fermeture du fichier
23 etat=1; % Fichier créer
24
25 end

```



### 3. Programme principal

```
%les positions articulaires initiales en degre :
teta1=90;
teta2=0;
teta3=0;
teta4=0;
teta5=0;
teta6=90;
%on va ecrire les angles en radian :
t1=teta1*(pi/180);
t2=teta2*(pi/180);
t3=teta3*(pi/180);
t4=teta4*(pi/180);
t5=teta5*(pi/180);
t6=teta6*(pi/180);
teta=[t1;t2;t3;t4;t5;t6];
%position initial
Ti=ModelGD(teta);
%position des bloc :
%BLEU
Tb=[-1,0,0,-200;0,0,1,1400;0,1,0,500;0,0,0,1];
%JAUNE
Tj=[-1,0,0,0;0,0,1,1500;0,1,0,500;0,0,0,1];
%ROUGE
Tr=[-1,0,0,200;0,0,1,1400;0,1,0,500;0,0,0,1];

%aller vers le bloc bleu
traj1=trajectoire2points(teta,Tb)*(180/pi);
trajectoire1=traj1'
teta=traj1(:,126) ; %les nouveaux coordonnés
%aller vers le jaune
traj2=trajectoire2points(teta,Tj)*(180/pi);
trajectoire2=traj2'
teta=traj2(:,126); %les nouveaux coordonnés
%aller vers le bloc rouge
traj3=trajectoire2points(teta,Tr)*(180/pi);
trajectoire3=traj3'
teta=traj3(:,126); %les nouveaux coordonnés
%aller vers le bloc bleu
traj4=trajectoire2points(teta,Tb)*(180/pi);
trajectoire4=traj4'
teta=traj4(:,126); %les nouveaux coordonnés
%aller vers le bloc jaune
traj5=trajectoire2points(teta,Tj)*(180/pi);
trajectoire5=traj5'
teta=traj5(:,126); %les nouveaux coordonnés
%aller vers le bloc rouge
traj6=trajectoire2points(teta,Tr)*(180/pi);
trajectoire6=traj6'
teta=traj6(:,126) ; %les nouveaux coordonnés
%revenir au position initiale
traj7=trajectoire2points(teta,Ti)*(180/pi);
trajectoire7=traj7'
teta=traj7(:,126) ; %les nouveaux coordonnés

h=rad2deg(teta)
```

```
>> clear all
>> tetai = [ 90 0 0 0 0 90 ];
>> points = [-200 1400 520 0 ; -200 1400 500 1];
>> pas = 0.04;
>> T = 5;
>> e = Trajectoire(tetai,points,pas,T)
```

e =

1

fichier\_trj - Bloc-notes

Fichier Edition Format Affichage Aide

```
pas : 0.04000
duree : 5.00000
nombre de ligne : 252
```

teta1	teta2	teta3	teta4	teta5	teta6	etat
90.00000000	-0.00000000	0.00000000	-0.00000000	-0.00000000	90.00000000	0.00000000
90.000032852	-0.000062298	0.000270472	8.801767675	-0.000210538	81.198232325	0.00000000
90.000270995	-0.000503749	0.002186072	9.150525795	-0.001704009	80.849474209	0.00000000
90.000916663	-0.001703489	0.007393125	9.152335315	-0.005763005	80.847664731	0.00000000
90.002172124	-0.004035482	0.017517235	9.152601823	-0.013655613	80.847398433	0.00000000
90.004237780	-0.007869631	0.034171053	9.153037326	-0.026640638	80.846963646	0.00000000
90.007313035	-0.013571358	0.058955825	9.153685429	-0.045969884	80.846317467	0.00000000
90.011595145	-0.021497963	0.093449868	9.154587577	-0.072880203	80.845419704	0.00000000
90.017276354	-0.031991624	0.139183200	9.155783220	-0.108574887	80.844232941	0.00000000
90.024573000	-0.045300000	0.197663000	9.157344700	-0.151045000	80.842700000	0.00000000

### **Lien Fichier DRIVE**

- Trajectoire Txt + **Fonctions** MATLAB :  
[Projet Robotique \(ZENZER abdelouahab SOUABI oussama\) - Google Drive](#)

- Simulation de la trajectoire : **Vidéo** (en perspective **ISO**)

[https://drive.google.com/file/d/1tloH\\_XdEMvVulTWre8mTpxj6zF2USINM/view?usp=drive\\_link](https://drive.google.com/file/d/1tloH_XdEMvVulTWre8mTpxj6zF2USINM/view?usp=drive_link)



## Conclusion

En conclusion, ce projet nous a offert une opportunité précieuse d'appliquer les connaissances et les compétences acquises lors du cours de robotique industrielle. Nous avons pu mettre en pratique une série de fonctions soigneusement coordonnées pour accomplir la tâche spécifique consistant à saisir et déplacer des blocs de différentes couleurs, puis les ramener à leur position initiale. De plus, nous avons eu l'occasion d'évaluer et de vérifier les résultats de nos simulations grâce à l'utilisation du logiciel Rokisim. Ce processus nous a permis de consolider notre compréhension des principes théoriques en les appliquant dans un contexte pratique, ce qui constitue une étape cruciale dans notre apprentissage de la robotique industrielle.