# Devoir : Cinématique Inverse

1/ En cours on a trouver les parametres de Denavit-Hartenberg :

| articulation | theta_i | di | ai | alpha_i |
|---|---|---|---|---|
| 1 | q1 | 0 | 0 | -pi/2 |
| 2 | q2-pi/2 | 0 | 260 | pi |
| 3 | q3 | 0 | 20 | -pi/2 |
| 4 | q4 | -290 | 0 | pi/2 |
| 5 | q5 | 0 | 0 | -pi/2 |
| 6 | q6 | -70 | 0 | pi |

Dans le tableau j'écris tous en radiant car sympy prend les angles en radiant.

Pour trouver les equations j'ai utiliser python, la librairie sympy pour faire des calcules avec des valeurs symboliques et j'ai utiliser des partie du code que l'o a fait en TD et les formules que vous donnée dans l'énoncé du DM.

Dans un premier temps j'ai importé la biblio sympy, definit les valeurs symboliques, puis j'ai calculer les matrice de A01 a A56

```python
import numpy as np
from sympy import symbols,cos,sin,Matrix,simplify, nsimplify, pi, trigsimp

q1,q2,q3,q4,q5,q6 = symbols('q1,q2,q3,q4,q5,q6')
print(q1,q2,q3,q4,q5,q6)

def transo_homogene(theta, d, a ,alpha):
    C_A = cos(alpha)
    C_T = cos(theta)
    S_A = sin(alpha)
    S_T = sin(theta)

    T = Matrix([
        [C_T, -S_T*C_A, S_T*S_A, a*C_T],
        [S_T, C_T*C_A, -C_T*S_A, a*S_T],
        [0, S_A, C_A, d],
        [0, 0, 0, 1]
    ])

    return T

pi2 = pi/2

# thetai = [q1,q2-90,q3,q4,q5,q6]
thetai = [q1,q2-pi2,q3,q4,q5,q6]
di=[0,0,0,-290,0,-70]
ai = [0,260,20,0,0,0]
#alphai = [-90,180,-90,90,-90,180]
alphai = [-pi2,pi,-pi2,pi2,-pi2, pi]
all_A_matrix = []
for i in range(6):
    all_A_matrix.append(transo_homogene(thetai[i], di[i],ai[i],alphai[i]))


A01,A12,A23,A34,A45,A56 = all_A_matrix
```

on obtient les matrices suivantes :

A01:
Matrix([[cos(q1), 0, -sin(q1), 0], [sin(q1), 0, cos(q1), 0], [0, -1, 0, 0], [0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| cos(q1) | 0 | -sin(q1) | 0 |
| sin(q1) | 0 | cos(q1) | 0 |
| 0 | -1 | 0 | 0 |
| 0 | 0 | 0 | 1 |

A12:
Matrix([[sin(q2), -cos(q2), 0, 260*sin(q2)], [-cos(q2), -sin(q2), 0, -260*cos(q2)], [0, 0, -1, 0], [0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| sin(q2) | -cos(q2) | 0 | 260*sin(q2) |
| -cos(q2) | -sin(q2) | 0 | -260*cos(q2) |
| 0 | 0 | -1 | 0 |
| 0 | 0 | 0 | 1 |

A23:
Matrix([[cos(q3), 0, -sin(q3), 20*cos(q3)],[sin(q3), 0, cos(q3), 20*sin(q3)], [0, -1, 0, 0], [0, 0, 0,1]])

| | | | |
|---|---|---|---|
| cos(q3) | 0 | -sin(q3) | 20*cos(q3) |
| sin(q3) | 0 | cos(q3) | 20*sin(q3) |
| 0 | -1 | 0 | 0 |
| 0 | 0 | 0 | 1 |

A34:
Matrix([[cos(q4), 0, sin(q4), 0],[sin(q4), 0, -cos(q4), 0],[0, 1, 0, -290], [0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| cos(q4) | 0 | sin(q4) | 0 |
| sin(q4) | 0 | -cos(q4) | 0 |
| 0 | 1 | 0 | -290 |
| 0 | 0 | 0 | 1 |

A45:
Matrix([[cos(q5), 0, -sin(q5), 0],[sin(q5), 0, cos(q5), 0], [0, -1, 0, 0], [0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| cos(q5) | 0 | -sin(q5) | 0 |
| sin(q5) | 0 | cos(q5) | 0 |
| 0 | -1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |

A56:
Matrix([[cos(q6), sin(q6), 0, 0], [sin(q6), -cos(q6), 0, 0], [0, 0, -1, -70], [0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| cos(q6) | sin(q6) | 0 | 0 |
| sin(q6) | -cos(q6) | 0 | 0 |
| 0 | 0 | -1 | -70 |
| 0 | 0 | 0 | 1 |

Ensuite on calcule A04 et A46

```
A04 = simplify(A01 @ A12 @ A23 @ A34)
A46 = simplify(A45 @ A56)
```

on passe le resultat dans simplify afin de « supprimer » ce qui peut l'être comme les cos(q1)**2 + sin(q1)**2 =1

A04 Matrix([
[-sin(q1)*sin(q4) + sin(q2 - q3)*cos(q1)*cos(q4), -cos(q1)*cos(q2 - q3), sin(q1)*cos(q4) + sin(q4)*sin(q2 - q3)*cos(q1), 10*(26*sin(q2) + 2*sin(q2 - q3) + 29*cos(q2 - q3))*cos(q1)],
[sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1), -sin(q1)*cos(q2 - q3), sin(q1)*sin(q4)*sin(q2 - q3) - cos(q1)*cos(q4), 10*(26*sin(q2) + 2*sin(q2 - q3) + 29*cos(q2 - q3))*sin(q1)],
[cos(q4)*cos(q2 - q3), sin(q2 - q3), sin(q4)*cos(q2 - q3), -290*sin(q2 - q3) + 260*cos(q2) + 20*cos(q2 - q3)],
[0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| -sin(q1)*sin(q4) + sin(q2 - q3)*cos(q1)*cos(q4) | -cos(q1)*cos(q2 - q3) | sin(q1)*cos(q4) + sin(q4)*sin(q2 - q3)*cos(q1) | 10*(26*sin(q2) + 2*sin(q2 - q3) + 29*cos(q2 - q3))*cos(q1)] |
| sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1) | -sin(q1)*cos(q2 - q3) | sin(q1)*sin(q4)*sin(q2 - q3) - cos(q1)*cos(q4) | 10*(26*sin(q2) + 2*sin(q2 - q3) + 29*cos(q2 - q3))*sin(q1) |
| cos(q4)*cos(q2 - q3) | sin(q2 - q3) | sin(q4)*cos(q2 - q3) | -290*sin(q2 - q3) + 260*cos(q2) + 20*cos(q2 - q3) |
| 0 | 0 | 0 | 1 |

A46 Matrix([[cos(q5)*cos(q6), sin(q6)*cos(q5), sin(q5), 70*sin(q5)],
[sin(q5)*cos(q6), sin(q5)*sin(q6), -cos(q5), -70*cos(q5)],
[-sin(q6), cos(q6), 0, 0],
[0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| cos(q5)*cos(q6) | sin(q6)*cos(q5) | sin(q5) | 70*sin(q5) |
| sin(q5)*cos(q6) | sin(q5)*sin(q6) | -cos(q5) | -70*cos(q5) |
| -sin(q6) | cos(q6) | 0 | 0 |
| 0 | 0 | 0 | 1 |

Ensuite faut calculer l'inverse de la matrice A46

```
inv_A46 = simplify(A46.inv())
```

inv A46 Matrix([[cos(q5)*cos(q6), sin(q5)*cos(q6), -sin(q6), 0], [sin(q6)*cos(q5), sin(q5)*sin(q6), cos(q6), 0], [sin(q5), -cos(q5), 0, -70], [0, 0, 0, 1]])

| | | | |
|---|---|---|---|
| cos(q5)*cos(q6) | sin(q5)*cos(q6) | -sin(q6) | 0 |
| sin(q6)*cos(q5) | sin(q5)*sin(q6) | cos(q6) | 0 |
| sin(q5) | -cos(q5) | 0 | -70 |
| 0 | 0 | 0 | 1 |

On sait que :

$$ {}^{0}T_{6}\left({}^{4}A_{6}\right)^{-1} = {}^{0}A_{4} $$

Donc :

$({}^{0}T_{6}\ ({}^{4}A_{6})^{-1}\ ) - {}^{0}A_{4} = 0$

en python :

```
T06invA46 =  simplify(T06 @ inv_A46)

eqs = T06invA46-A04
```

Matrix([[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]])

On pose comme équation :

eq1=(sin(q5)*cos(q4)*cos(q2 - q3) - sin(q2 - q3)*cos(q5))*sin(q5) + (-sin(q4)*sin(q6)*cos(q2 - q3) + sin(q5)*sin(q2 - q3)*cos(q6) + cos(q4)*cos(q5)*cos(q6)*cos(q2 - q3))*cos(q5)*cos(q6) + (sin(q4)*cos(q6)*cos(q2 - q3) + sin(q5)*sin(q6)*sin(q2 - q3) + sin(q6)*cos(q4)*cos(q5)*cos(q2 – q3))*sin(q6)*cos(q5) - cos(q4)*cos(q2 - q3)

eq2=-(sin(q5)*cos(q4)*cos(q2 - q3) - sin(q2 - q3)*cos(q5))*cos(q5) + (-sin(q4)*sin(q6)*cos(q2 - q3) + sin(q5)*sin(q2 - q3)*cos(q6) + cos(q4)*cos(q5)*cos(q6)*cos(q2 - q3))*sin(q5)*cos(q6) + (sin(q4)*cos(q6)*cos(q2 - q3) + sin(q5)*sin(q6)*sin(q2 - q3) + sin(q6)*cos(q4)*cos(q5)*cos(q2 – q3))*sin(q5)*sin(q6) - sin(q2 - q3)

eq3=((sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1))*sin(q5) + sin(q1)*cos(q5)*cos(q2 - q3))*sin(q5) + (-(sin(q1)*sin(q4)*sin(q2 - q3) - cos(q1)*cos(q4))*sin(q6) + (sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1))*cos(q5)*cos(q6) - sin(q1)*sin(q5)*cos(q6)*cos(q2 - q3))*cos(q5)*cos(q6) + ((sin(q1)*sin(q4)*sin(q2 - q3) - cos(q1)*cos(q4))*cos(q6) + (sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1))*sin(q6)*cos(q5) - sin(q1)*sin(q5)*sin(q6)*cos(q2 – q3))*sin(q6)*cos(q5) - sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1)

en python :

```
eq1 = str(T06invA46[2,0]) + "+" + str(A04[2,0])
eq2 = str(T06invA46[2,1]) + "+" + str(A04[2,1])
eq3 = str(T06invA46[1,0]) + "+" + str(A04[1,0])
```

je fait les calcules en chaine de caractere car sinon il return 0 car il simplifie l’équation

Les 3 equations du dessus pourrait être plus simplifier si on remplace T06 par les valeur numérique est pas les valeurs symbolique

Comme on connaît une des combinaison possible de q1 q2 q3 les équations seront plus « simple » a résoudre avec fsolve, la fonction ressemblera au screen si dessous, a priorie.

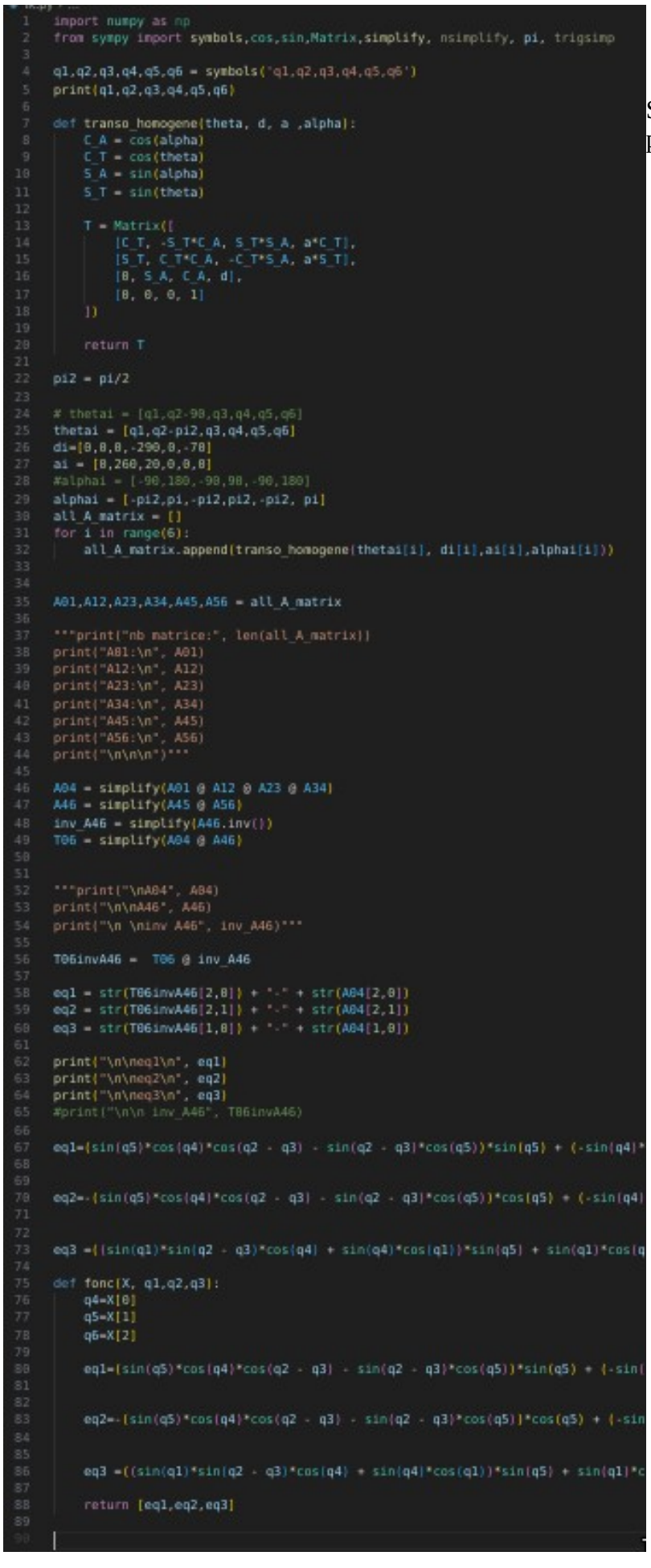
```
def fonc(X, q1,q2,q3):
    q4=X[0]
    q5=X[1]
    q6=X[2]

    eq1=(sin(q5)*cos(q4)*cos(q2 - q3)


    eq2=-(sin(q5)*cos(q4)*cos(q2 - q3


    eq3 =((sin(q1)*sin(q2 - q3)*cos(q4

    return [eq1,eq2,eq3]
```



```python
import numpy as np
from sympy import symbols,cos,sin,Matrix,simplify, nsimplify, pi, trigsimp

q1,q2,q3,q4,q5,q6 = symbols('q1,q2,q3,q4,q5,q6')
print(q1,q2,q3,q4,q5,q6)

def transo_homogene(theta, d, a ,alpha):
    C_A = cos(alpha)
    C_T = cos(theta)
    S_A = sin(alpha)
    S_T = sin(theta)

    T = Matrix([
        [C_T, -S_T*C_A, S_T*S_A, a*C_T],
        [S_T, C_T*C_A, -C_T*S_A, a*S_T],
        [0, S_A, C_A, d],
        [0, 0, 0, 1]
    ])

    return T

pi2 = pi/2

# thetai = [q1,q2-90,q3,q4,q5,q6]
thetai = [q1,q2-pi2,q3,q4,q5,q6]
di=[0,0,0,-290,0,-70]
ai = [0,260,20,0,0,0]
#alphai = [-90,180,-90,90,-90,180]
alphai = [-pi2,pi,-pi2,pi2,-pi2, pi]
all_A_matrix = []
for i in range(6):
    all_A_matrix.append(transo_homogene(thetai[i], di[i],ai[i],alphai[i]))


A01,A12,A23,A34,A45,A56 = all_A_matrix

"""print("nb matrice:", len(all_A_matrix))
print("A01:\n", A01)
print("A12:\n", A12)
print("A23:\n", A23)
print("A34:\n", A34)
print("A45:\n", A45)
print("A56:\n", A56)
print("\n\n\n")"""

A04 = simplify(A01 @ A12 @ A23 @ A34)
A46 = simplify(A45 @ A56)
inv_A46 = simplify(A46.inv())
T06 = simplify(A04 @ A46)


"""print("\nA04", A04)
print("\n\nA46", A46)
print("\n \ninv A46", inv_A46)"""

T06invA46 =  T06 @ inv_A46

eq1 = str(T06invA46[2,0]) + "-" + str(A04[2,0])
eq2 = str(T06invA46[2,1]) + "-" + str(A04[2,1])
eq3 = str(T06invA46[1,0]) + "-" + str(A04[1,0])

print("\n\neq1\n", eq1)
print("\n\neq2\n", eq2)
print("\n\neq3\n", eq3)
#print("\n\n inv_A46", T06invA46)

eq1=(sin(q5)*cos(q4)*cos(q2 - q3) - sin(q2 - q3)*cos(q5))*sin(q5) + (-sin(q4)*

eq2=-(sin(q5)*cos(q4)*cos(q2 - q3) - sin(q2 - q3)*cos(q5))*cos(q5) + (-sin(q4)

eq3 =((sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1))*sin(q5) + sin(q1)*cos(q

def fonc(X, q1,q2,q3):
    q4=X[0]
    q5=X[1]
    q6=X[2]

    eq1=(sin(q5)*cos(q4)*cos(q2 - q3) - sin(q2 - q3)*cos(q5))*sin(q5) + (-sin(


    eq2=-(sin(q5)*cos(q4)*cos(q2 - q3) - sin(q2 - q3)*cos(q5))*cos(q5) + (-sin


    eq3 =((sin(q1)*sin(q2 - q3)*cos(q4) + sin(q4)*cos(q1))*sin(q5) + sin(q1)*c

    return [eq1,eq2,eq3]

```

Screen de mon code pour faire le DM