UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELLING AND CONTROL OF MANIPULATORS

Second Assignment

Manipulator Geometry and Direct Kinematics

Author:

Triki Karim

Student ID:

s5528602

Professors:

Giovanni Indiveri
Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo

December 20,2022

Contents

1 Assignment description **3**
1.1 Exercise 1 3

2 Exercise 1 **4**
2.1 Q1.1 4
2.2 Q1.2 8
2.3 Q1.3 9
2.4 Q1.4 11
2.5 Q1.5 12

Mathematical expression	Definition	MATLAB expression
$< w >$	World Coordinate Frame	w
${}^{a}_{b}R$	Rotation matrix of frame $< b >$ with respect to frame $< a >$	aRb
${}^{a}_{b}T$	Transformation matrix of frame $< b >$ with respect to frame $< a >$	aTb

Table 1: Nomenclature Table

1 Assignment description

The second assignment of Modelling and Control of Manipulators focuses on manipulators geometry and direct kinematics.

The second assignment is **mandatory** and consists of one exercise. You are asked to:

- Download the .zip file called MOCOM-LAB2 from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling the predefined files called "*main.m*", "*BuildTree.m*", "*GetDirectGeometry.m*", "*DirectGeometry.m*", "*GetTransformationWrtBase.m*", "*GetBasicVectorWrtBase.m*" and "*GetFrameWrtFrame.m*".
- Write a report motivating your answers, following the predefind format on this document.

1.1 Exercise 1

Given the following CAD model of an industrial 7 dof manipulator:

Q1.1 Define all the model matrices, by filling the structures in the *BuildTree()* function. Be careful to define the z-axis coinciding with the joint rotation axis, and such that the positive rotation is the same as showed in the CAD model you received. Draw on the CAD model the reference frames for each link and insert it into the report.

Q1.2 Implement a function called *DirectGeometry()* which can calculate how the matrix attached to a joint will rotate if the joint rotates. Then, develop a function called GetDirectGeometry() which returns all the model matrices given the following joint configuration **q** $= [1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3]$.

Q1.3 Calculate all the transformation matrices between any two links, between a link and the base and the corresponding distance vectors, filling respectively: *GetFrameWrtFrame()*, *GetTransformationWrtBase()*, *GetBasicVectgorWrtBase()*

Q1.4 Given the following starting and ending configuration:

- $\mathbf{q}_i = [1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3]$ and $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$
- $\mathbf{q}_i = [1.3, 0, 1.3, 1.7, 1.3, 0.8, 1.3]$ and $\mathbf{q}_f = [2, 0, 1.3, 1.7, 1.3, 0.8, 1.3]$
- $\mathbf{q}_i = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$

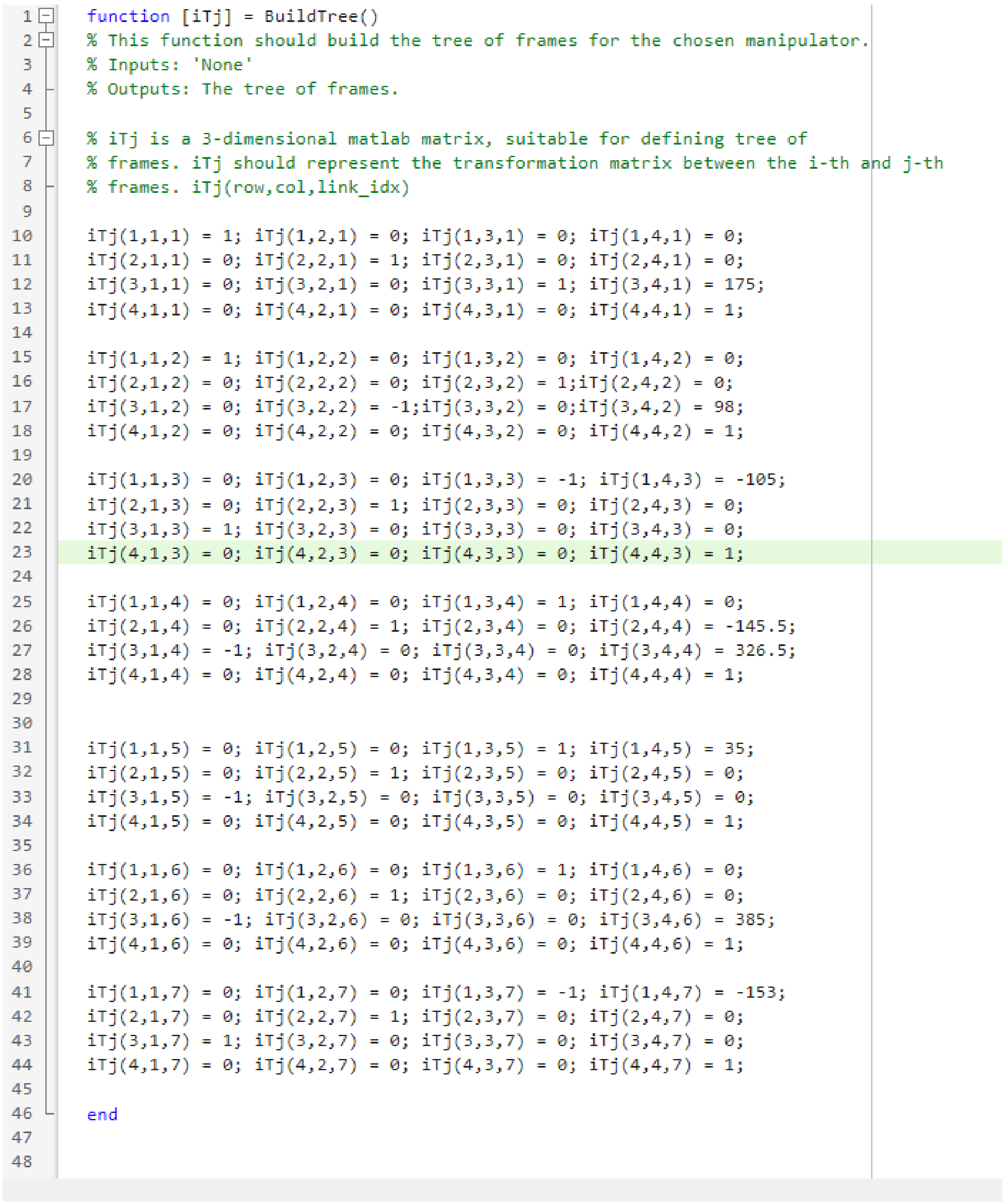
Plot the intermediate link positions in between the two configurations (you can use the plot3() or line() functions) and comment the results obtained.

Q1.5 Test your algorithm by changing one joint position at the time and plot the results obtained for at least 3 configurations.

2 Exercise 1

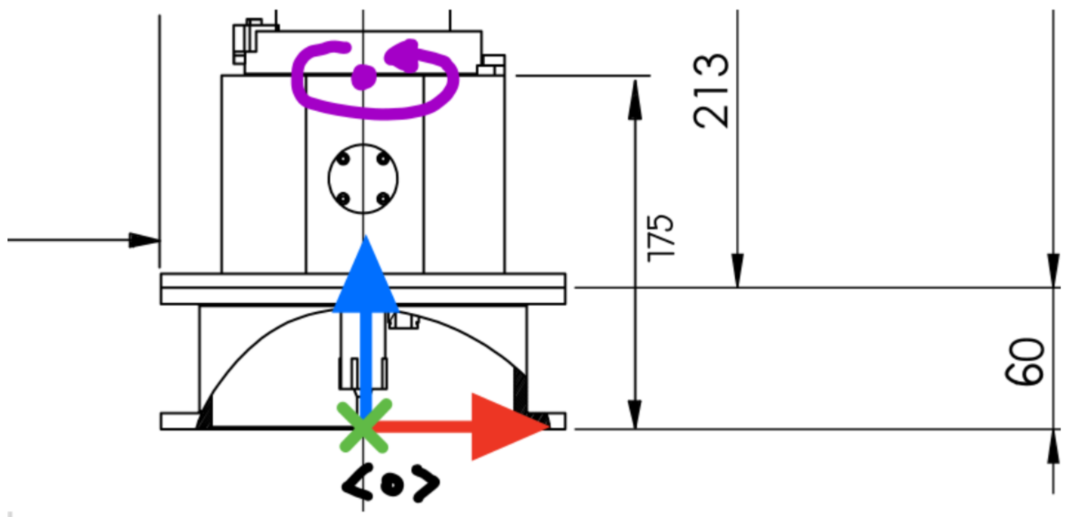
2.1 Q1.1

In this question we used the right hand rule in order to know the direction of the z-axis , according to this point we can chose direction of Y-axis and X-axis after that we calculate the difference between distances in the second page we find the frames and here those are the matrices code in the next pages I have explain the code how it works



```
function [iTj] = BuildTree()
% This function should build the tree of frames for the chosen manipulator.
% Inputs: 'None'
% Outputs: The tree of frames.

% iTj is a 3-dimensional matlab matrix, suitable for defining tree of
% frames. iTj should represent the transformation matrix between the i-th and j-th
% frames. iTj(row,col,link_idx)

iTj(1,1,1) = 1; iTj(1,2,1) = 0; iTj(1,3,1) = 0; iTj(1,4,1) = 0;
iTj(2,1,1) = 0; iTj(2,2,1) = 1; iTj(2,3,1) = 0; iTj(2,4,1) = 0;
iTj(3,1,1) = 0; iTj(3,2,1) = 0; iTj(3,3,1) = 1; iTj(3,4,1) = 175;
iTj(4,1,1) = 0; iTj(4,2,1) = 0; iTj(4,3,1) = 0; iTj(4,4,1) = 1;

iTj(1,1,2) = 1; iTj(1,2,2) = 0; iTj(1,3,2) = 0; iTj(1,4,2) = 0;
iTj(2,1,2) = 0; iTj(2,2,2) = 0; iTj(2,3,2) = 1;iTj(2,4,2) = 0;
iTj(3,1,2) = 0; iTj(3,2,2) = -1;iTj(3,3,2) = 0;iTj(3,4,2) = 98;
iTj(4,1,2) = 0; iTj(4,2,2) = 0; iTj(4,3,2) = 0; iTj(4,4,2) = 1;

iTj(1,1,3) = 0; iTj(1,2,3) = 0; iTj(1,3,3) = -1; iTj(1,4,3) = -105;
iTj(2,1,3) = 0; iTj(2,2,3) = 1; iTj(2,3,3) = 0; iTj(2,4,3) = 0;
iTj(3,1,3) = 1; iTj(3,2,3) = 0; iTj(3,3,3) = 0; iTj(3,4,3) = 0;
iTj(4,1,3) = 0; iTj(4,2,3) = 0; iTj(4,3,3) = 0; iTj(4,4,3) = 1;

iTj(1,1,4) = 0; iTj(1,2,4) = 0; iTj(1,3,4) = 1; iTj(1,4,4) = 0;
iTj(2,1,4) = 0; iTj(2,2,4) = 1; iTj(2,3,4) = 0; iTj(2,4,4) = -145.5;
iTj(3,1,4) = -1; iTj(3,2,4) = 0; iTj(3,3,4) = 0; iTj(3,4,4) = 326.5;
iTj(4,1,4) = 0; iTj(4,2,4) = 0; iTj(4,3,4) = 0; iTj(4,4,4) = 1;


iTj(1,1,5) = 0; iTj(1,2,5) = 0; iTj(1,3,5) = 1; iTj(1,4,5) = 35;
iTj(2,1,5) = 0; iTj(2,2,5) = 1; iTj(2,3,5) = 0; iTj(2,4,5) = 0;
iTj(3,1,5) = -1; iTj(3,2,5) = 0; iTj(3,3,5) = 0; iTj(3,4,5) = 0;
iTj(4,1,5) = 0; iTj(4,2,5) = 0; iTj(4,3,5) = 0; iTj(4,4,5) = 1;

iTj(1,1,6) = 0; iTj(1,2,6) = 0; iTj(1,3,6) = 1; iTj(1,4,6) = 0;
iTj(2,1,6) = 0; iTj(2,2,6) = 1; iTj(2,3,6) = 0; iTj(2,4,6) = 0;
iTj(3,1,6) = -1; iTj(3,2,6) = 0; iTj(3,3,6) = 0; iTj(3,4,6) = 385;
iTj(4,1,6) = 0; iTj(4,2,6) = 0; iTj(4,3,6) = 0; iTj(4,4,6) = 1;

iTj(1,1,7) = 0; iTj(1,2,7) = 0; iTj(1,3,7) = -1; iTj(1,4,7) = -153;
iTj(2,1,7) = 0; iTj(2,2,7) = 1; iTj(2,3,7) = 0; iTj(2,4,7) = 0;
iTj(3,1,7) = 1; iTj(3,2,7) = 0; iTj(3,3,7) = 0; iTj(3,4,7) = 0;
iTj(4,1,7) = 0; iTj(4,2,7) = 0; iTj(4,3,7) = 0; iTj(4,4,7) = 1;

end
```

Build Tree Code Matlab



```
iTj(1,1,1) = 1; iTj(1,2,1) = 0; iTj(1,3,1) = 0; iTj(1,4,1) = 0;
iTj(2,1,1) = 0; iTj(2,2,1) = 1; iTj(2,3,1) = 0; iTj(2,4,1) = 0;
iTj(3,1,1) = 0; iTj(3,2,1) = 0; iTj(3,3,1) = 1; iTj(3,4,1) = 175;
iTj(4,1,1) = 0; iTj(4,2,1) = 0; iTj(4,3,1) = 0; iTj(4,4,1) = 1;

iTj(1,1,2) = 1; iTj(1,2,2) = 0; iTj(1,3,2) = 0; iTj(1,4,2) = 0;
iTj(2,1,2) = 0; iTj(2,2,2) = 0; iTj(2,3,2) = 1;iTj(2,4,2) = 0;
iTj(3,1,2) = 0; iTj(3,2,2) = -1;iTj(3,3,2) = 0;iTj(3,4,2) = 98;
iTj(4,1,2) = 0; iTj(4,2,2) = 0; iTj(4,3,2) = 0; iTj(4,4,2) = 1;
```

iTj(1,1,1) = 1 $\longrightarrow$ <x> in the 1st frame

iTj(2,2,1) = 1 $\longrightarrow$ <y> in the 1st frame

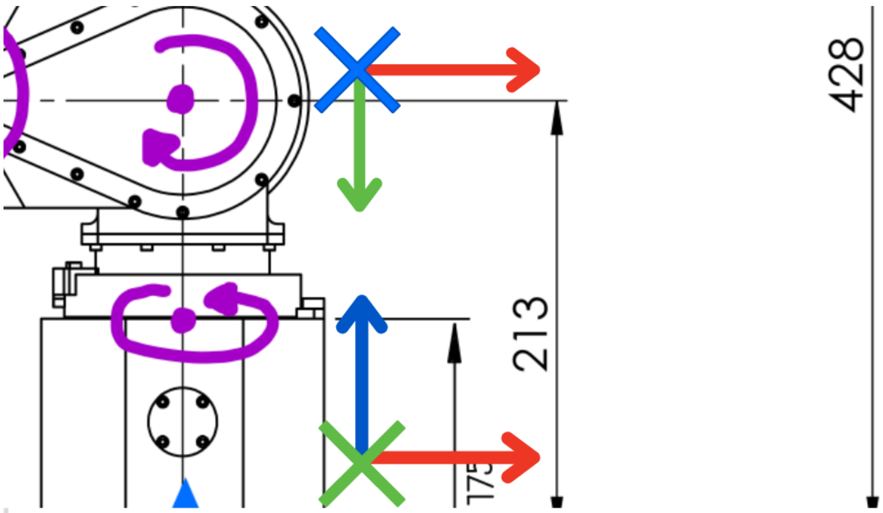
iTj(3,3,1) = 1 $\longrightarrow$ <z> in the 1st frame

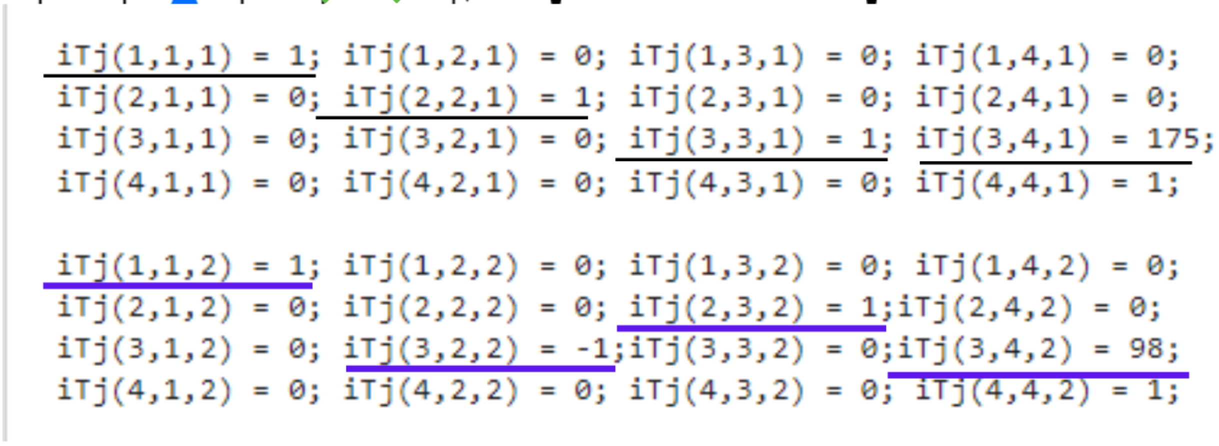
Rotation& move to the 2nd frame

iTj(1,1,2) = 1 $\longrightarrow$ <x> in the 2nd frame

iTj(3,2,2) = -1 $\longrightarrow$ <y> in the 2nd frame

iTj(2,3,2) = 1 $\longrightarrow$ <z> in the 2nd frame





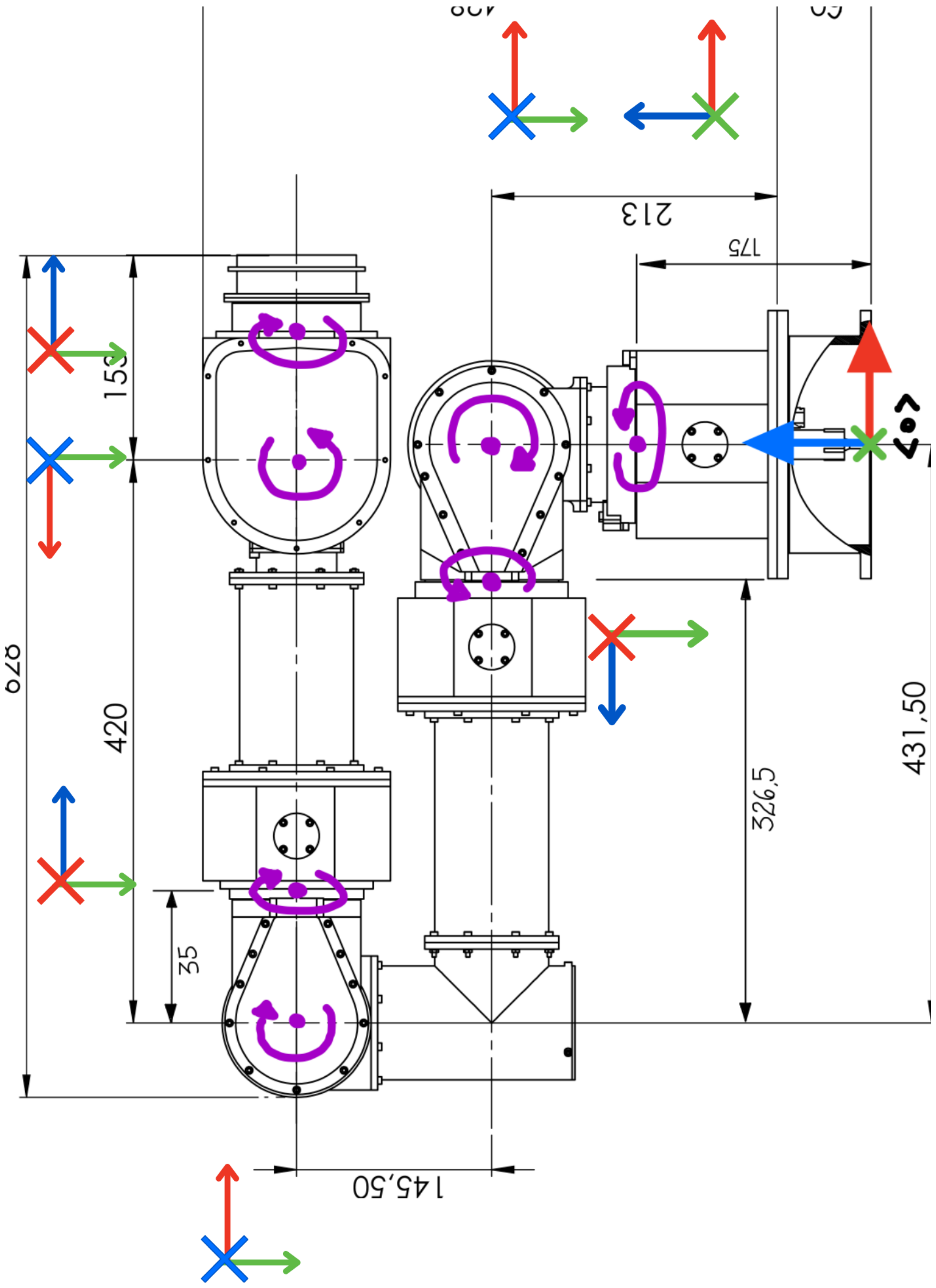
```
iTj(1,1,1) = 1; iTj(1,2,1) = 0; iTj(1,3,1) = 0; iTj(1,4,1) = 0;
iTj(2,1,1) = 0; iTj(2,2,1) = 1; iTj(2,3,1) = 0; iTj(2,4,1) = 0;
iTj(3,1,1) = 0; iTj(3,2,1) = 0; iTj(3,3,1) = 1; iTj(3,4,1) = 175;
iTj(4,1,1) = 0; iTj(4,2,1) = 0; iTj(4,3,1) = 0; iTj(4,4,1) = 1;

iTj(1,1,2) = 1; iTj(1,2,2) = 0; iTj(1,3,2) = 0; iTj(1,4,2) = 0;
iTj(2,1,2) = 0; iTj(2,2,2) = 0; iTj(2,3,2) = 1;iTj(2,4,2) = 0;
iTj(3,1,2) = 0; iTj(3,2,2) = -1;iTj(3,3,2) = 0;iTj(3,4,2) = 98;
iTj(4,1,2) = 0; iTj(4,2,2) = 0; iTj(4,3,2) = 0; iTj(4,4,2) = 1;
```

Why iTj(3,4,2) = 98 ? :

(213+60)-175=98

Why iTj(3,2,2) = -1 ? :

Applying the right-hand rule, frame Y takes the place of frame Z in the opposite direction



213
175
153
420
870
35
145,50
326,5
431,50

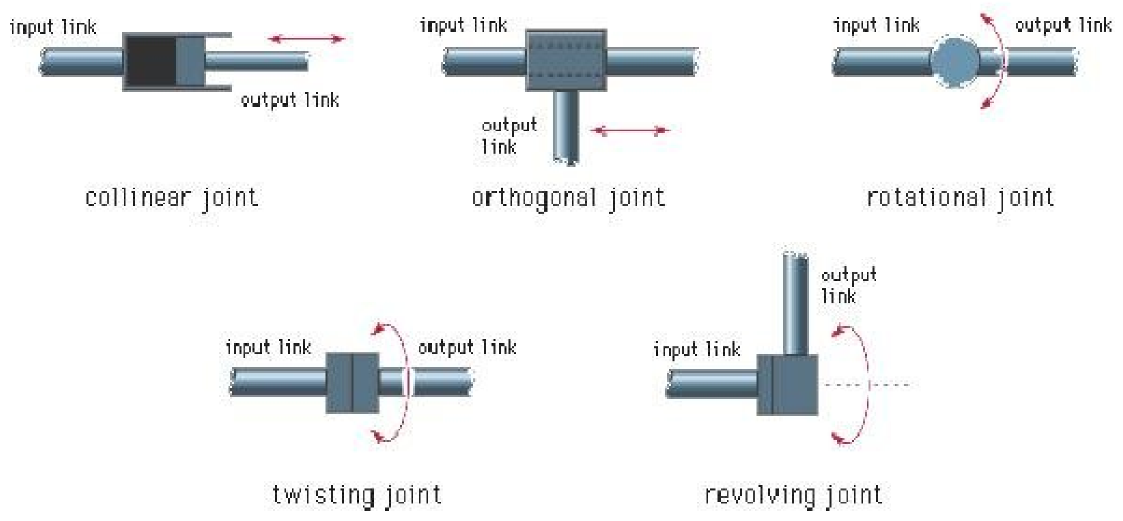
2.2 Q1.2

During the solution, an important question was spinning in my mind, which is what is the joint for the robot ? what are its types ? and how we can define it in the system ?

so after long search I got a conclusion that joint are the main point in the real robot without them robot can do its own tasks .

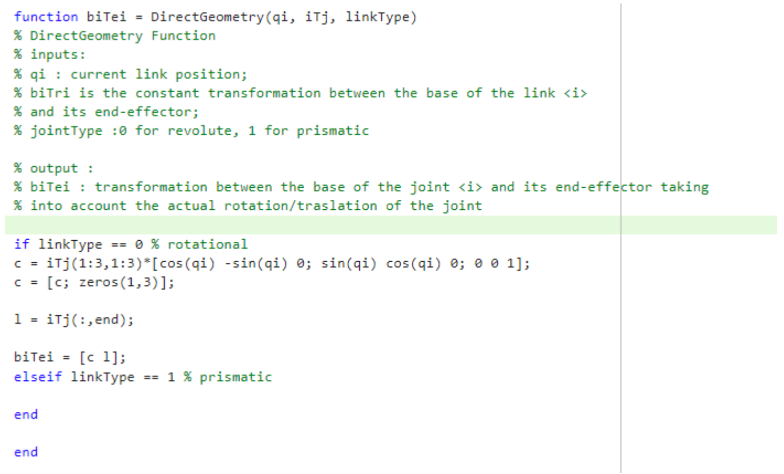
In order to make a lot of robots that can make more than one task and different tasks we should have another types of joint like ROTATIONAL Joint "as in the example" Linear Joint and this one give to the robot linear motion according to the input and the output without forgetting that joint can do sliding and translation movement and I think this makes Linear Joint Special.

In the Other Part we have Twisting Joint , Orthogonal Joint , Revolving Joint but in this question we need to focus more about the rotational joint



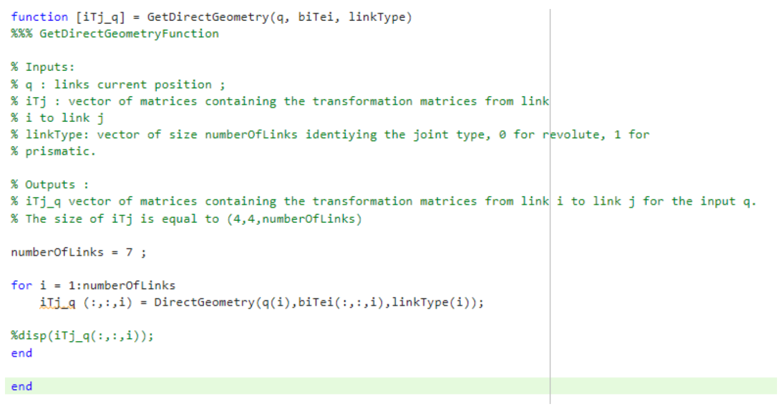
types of joints

Rotational Joint allows to 3D bodies to rotate relative to each other about a fixed axis and permits no other relative motion



```
function biTei = DirectGeometry(qi, iTj, linkType)
% DirectGeometry Function
% inputs:
% qi : current link position;
% biTri is the constant transformation between the base of the link <i>
% and its end-effector;
% jointType :0 for revolute, 1 for prismatic

% output :
% biTei : transformation between the base of the joint <i> and its end-effector taking
% into account the actual rotation/traslation of the joint

if linkType == 0 % rotational
c = iTj(1:3,1:3)*[cos(qi) -sin(qi) 0; sin(qi) cos(qi) 0; 0 0 1];
c = [c; zeros(1,3)];

l = iTj(:,end);

biTei = [c l];
elseif linkType == 1 % prismatic

end

end
```

Direct Geometry code



```
function [iTj_q] = GetDirectGeometry(q, biTei, linkType)
%%% GetDirectGeometryFunction

% Inputs:
% q : links current position ;
% iTj : vector of matrices containing the transformation matrices from link
% i to link j
% linkType: vector of size numberOfLinks identiying the joint type, 0 for revolute, 1 for
% prismatic.

% Outputs :
% iTj_q vector of matrices containing the transformation matrices from link i to link j for the input q.
% The size of iTj is equal to (4,4,numberOfLinks)

numberOfLinks = 7 ;

for i = 1:numberOfLinks
    iTj_q (:,:,i) = DirectGeometry(q(i),biTei(:,:,i),linkType(i));

%disp(iTj_q(:,:,i));
end

end
```

Get Direct Geometry code

In this code we write the rotational matrices ${}^{i}Tjq = \begin{bmatrix} cosqi & -sinqi & 0 \\ sinqi & cosqi & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ${}^{i}Tjq = \begin{bmatrix} cosqi & -sinqi & 0 \\ sinqi & cosqi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

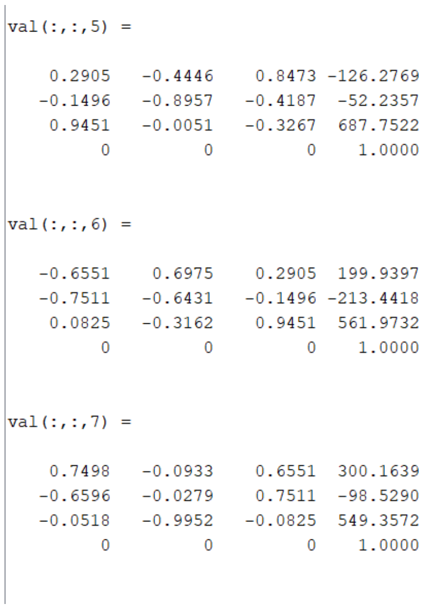
we mention the rotational matrices because we want to calculate how the matrix attached to a joint will rotate and we will get results in the workspace

2.3 Q1.3

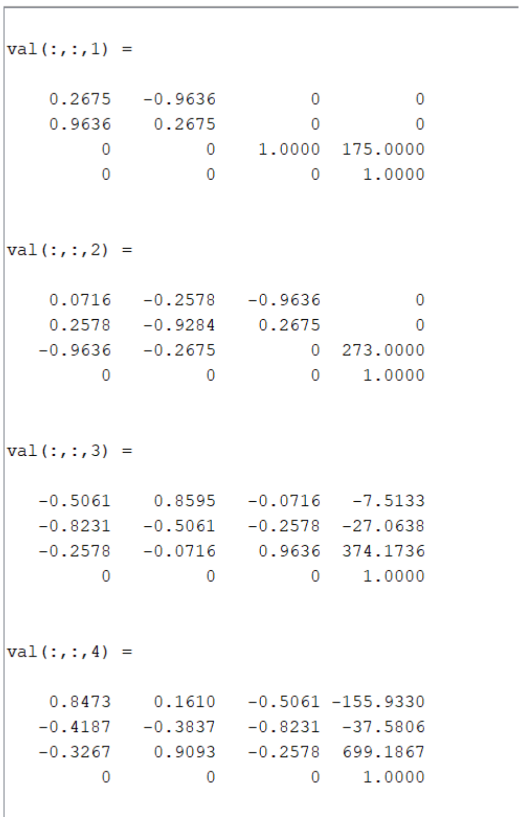
in this question we want to calculate the transformation matrices between any two links , link and a base and corresponding distance vectors, All of that we will make it by making the code in MATLAB : *GetFrame()*, *GetTransformationWrtBase(), GetBasicVectgorWrtBase()*

- *GetFrame()*: the mission of this function is to compute the transformation matrices between 2 links or 2 frames and It give us *iTj* matrix ${}^{i}Tj(:,:,1) = \begin{bmatrix} 0.1768 & 0.3266 & -0.9284 & -142.0519 \\ 0.9820 & 0.0051 & 0.1888 & 288.0812 \\ 0.0664 & -0.9451 & -0.3199 & 165.2037 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- *GetTransformationWrtBase()* this function give us the output of array *bTi* from link 1 to link 7



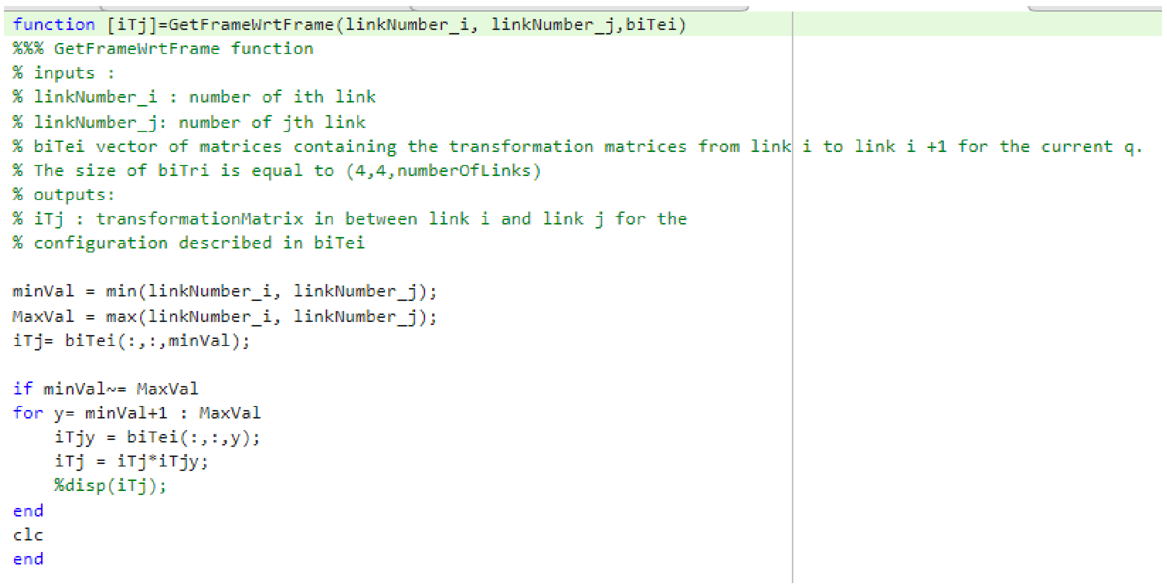
```
val(:,:,5) =

    0.2905   -0.4446    0.8473 -126.2769
   -0.1496   -0.8957   -0.4187  -52.2357
    0.9451   -0.0051   -0.3267  687.7522
         0         0         0    1.0000


val(:,:,6) =

   -0.6551    0.6975    0.2905  199.9397
   -0.7511   -0.6431   -0.1496 -213.4418
    0.0825   -0.3162    0.9451  561.9732
         0         0         0    1.0000


val(:,:,7) =

    0.7498   -0.0933    0.6551  300.1639
   -0.6596   -0.0279    0.7511  -98.5290
   -0.0518   -0.9952   -0.0825  549.3572
         0         0         0    1.0000
```



```
val(:,:,1) =

    0.2675   -0.9636         0         0
    0.9636    0.2675         0         0
         0         0    1.0000  175.0000
         0         0         0    1.0000


val(:,:,2) =

    0.0716   -0.2578   -0.9636         0
    0.2578   -0.9284    0.2675         0
   -0.9636   -0.2675         0  273.0000
         0         0         0    1.0000


val(:,:,3) =

   -0.5061    0.8595   -0.0716   -7.5133
   -0.8231   -0.5061   -0.2578  -27.0638
   -0.2578   -0.0716    0.9636  374.1736
         0         0         0    1.0000


val(:,:,4) =

    0.8473    0.1610   -0.5061 -155.9330
   -0.4187   -0.3837   -0.8231  -37.5806
   -0.3267    0.9093   -0.2578  699.1867
         0         0         0    1.0000
```

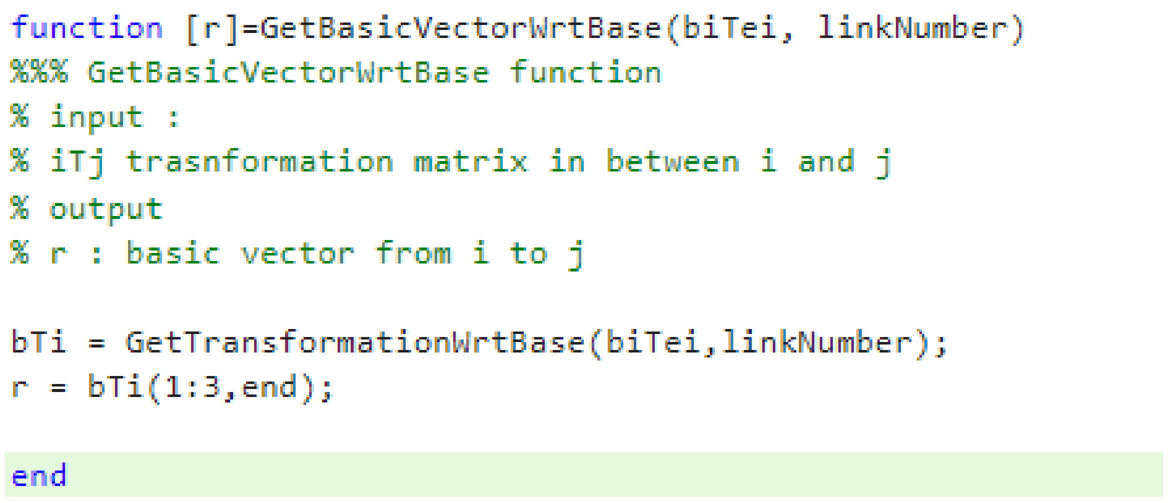
- *GetBasicVectgorWrtBase()*

  this function give us the output of distance vecotr *bri*



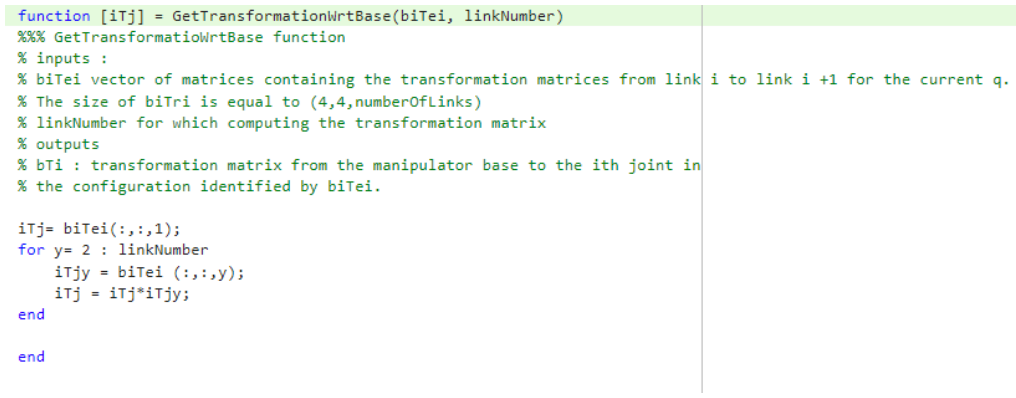
```
function [iTj]=GetFrameWrtFrame(linkNumber_i, linkNumber_j,biTei)
%%% GetFrameWrtFrame function
% inputs :
% linkNumber_i : number of ith link
% linkNumber_j: number of jth link
% biTei vector of matrices containing the transformation matrices from link i to link i +1 for the current q.
% The size of biTri is equal to (4,4,numberOfLinks)
% outputs:
% iTj : transformationMatrix in between link i and link j for the
% configuration described in biTei

minVal = min(linkNumber_i, linkNumber_j);
MaxVal = max(linkNumber_i, linkNumber_j);
iTj= biTei(:,:,minVal);

if minVal~= MaxVal
for y= minVal+1 : MaxVal
    iTjy = biTei(:,:,y);
    iTj = iTj*iTjy;
    %disp(iTj);
end
clc
end
```

GetFrameWrtFrame Code



```
function [r]=GetBasicVectorWrtBase(biTei, linkNumber)
%%% GetBasicVectorWrtBase function
% input :
% iTj trasnformation matrix in between i and j
% output
% r : basic vector from i to j

bTi = GetTransformationWrtBase(biTei,linkNumber);
r = bTi(1:3,end);

end
```

GetBasicVectorWrtBase Code



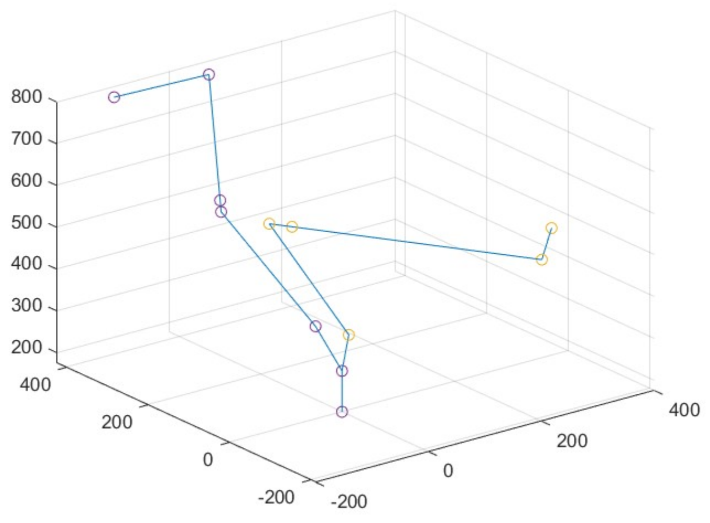
```
function [iTj] = GetTransformationWrtBase(biTei, linkNumber)
%%% GetTransformatioWrtBase function
% inputs :
% biTei vector of matrices containing the transformation matrices from link i to link i +1 for the current q.
% The size of biTri is equal to (4,4,numberOfLinks)
% linkNumber for which computing the transformation matrix
% outputs
% bTi : transformation matrix from the manipulator base to the ith joint in
% the configuration identified by biTei.

iTj= biTei(:,:,1);
for y= 2 : linkNumber
    iTjy = biTei (:,:,y);
    iTj = iTj*iTjy;
end

end
```

GetTransformationWrtBase Code

2.4 Q1.4

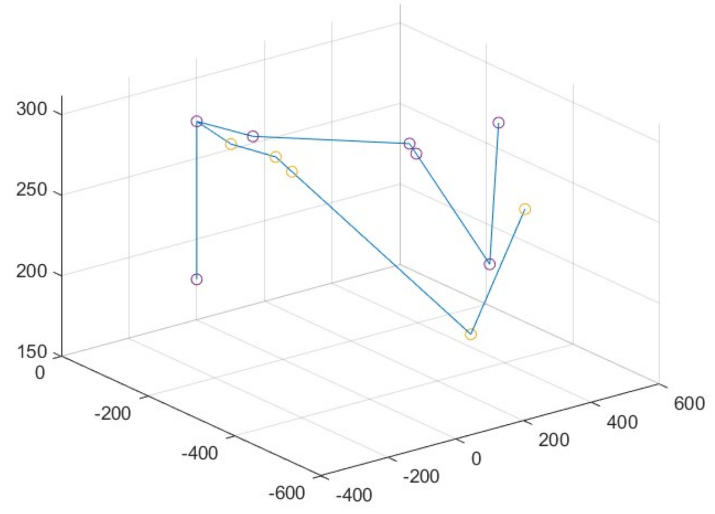
in this part, We Want to see the results of Starting and Ending Of configuration by our codes

- $\mathbf{q}_i = [1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3]$ and $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$



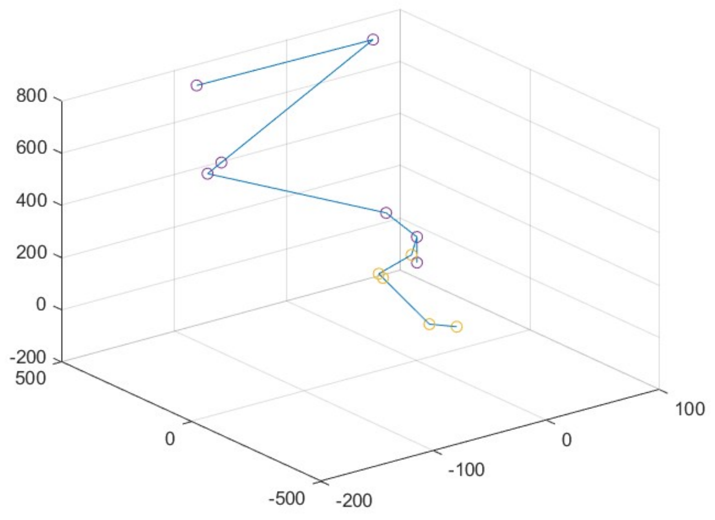
1st Figure

- $\mathbf{q}_i = [1.3, 0, 1.3, 1.7, 1.3, 0.8, 1.3]$ and $\mathbf{q}_f = [2, 0, 1.3, 1.7, 1.3, 0.8, 1.3]$



2nd Figure

- $\mathbf{q}_i = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$



3rd Figure

in each one we can see the number of links (lines) and number of steps (circles).

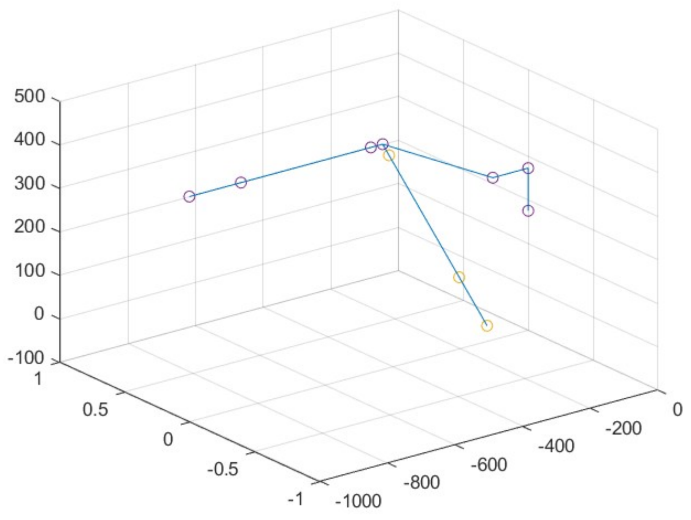
in the other side, we give to the MATLAB The **qi** and **qf** and by using the previous function that we made them in last questions they give us those figures

2.5 Q1.5

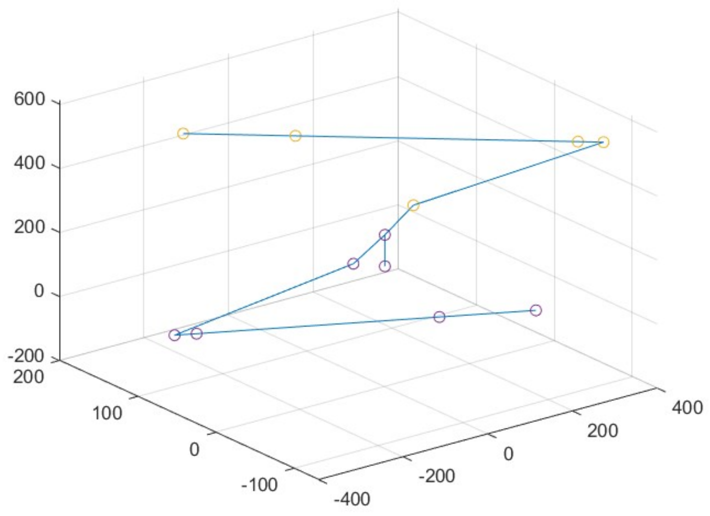
Based on the previous exercise, I solved this question, in addition to that we had the freedom to choose the values of the matrix, so I made these matrices

- $\mathbf{q}_5 = [0, 0, 0, 1, 0, 0, 0]$ and $\mathbf{q}_5 f = [0, 0, 0, \pi, 0, 0, 0]$
- $\mathbf{q}_6 = [75, 1001, 69, 1, 0, 0, 0]$ and $\mathbf{q}_6 f = [75, 1111, 69, 1, 0, 0, 0]$
- $\mathbf{q}_7 = [7, 10, 11, 35, 8, 16, 15]$ and $\mathbf{q}_7 f = [7, 10, 11, 35, 8, 16, 31]$

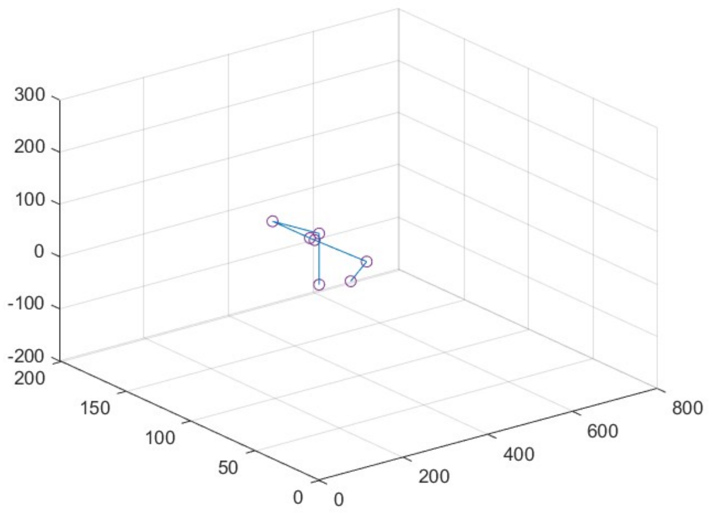
and those are the figures :



q5 and q5f figure



q6 and q6f figure



q7 and q7f figure

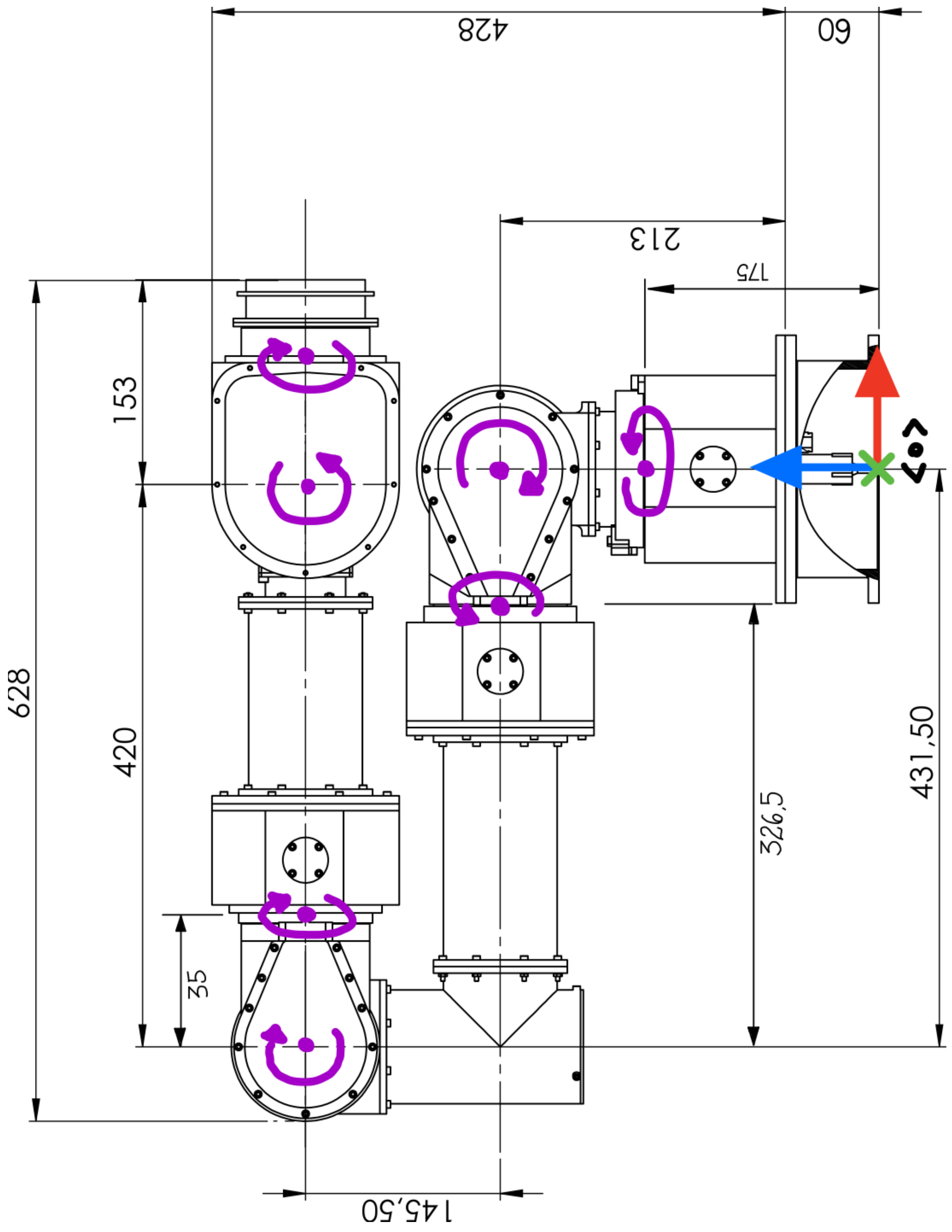


Figure 1: CAD model of the robot