SSY156 Modelling and control of mechatronic systems

Lab Assignment 1

Daniel Söderqvist (danisode)

March 14, 2023

P.S all the functions and scripts are attached in the zipped file "Lab_Assignment_1. Where all my own functions like DH to HT, DH to Jacobian as well as the main script is under "Mina Funktioner".

Exercise 1

The first thing in order to complete this task is to find the Denavit-Hartenberg (DH) matrix. This matrix consist of all the frames preferably in joint positions that we want to consider. To construct the DH table i used the following pictures given in the task PM as reference and then went my way from frame 0 to the end effector, see figure (1): With this picture we could start constructing the DH table

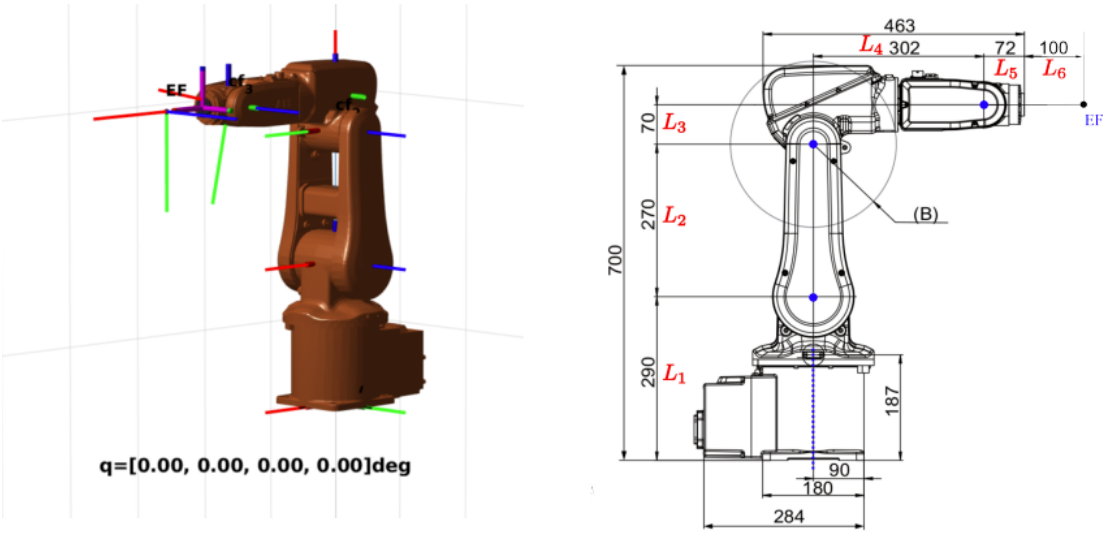

Figure 1: Models describing the frame-positions, joint-positions and the overall parameters of the robot.

for the distal method way of solving the following exercises. The finished DH table are shown in the table down below:

θ	d	a	α
$q1$	$L1$	0	$-\frac{\pi}{2}$
$q2 - \frac{\pi}{2}$	0	$L2$	0
$q3 + al$	0	$L7$	0
$q4 + \frac{\pi}{2} - al$	0	$L8$	0

Where parameters like $L7$, $L8$ and al is calculated using trigonometry from joint 2 til the end-effector. $L7 = \sqrt{L3^2 + L4^2}$, $L8 = L5 + L6$ and $al = \arctan(L4/L3)$. These parameters are used to finish the asked function "Model/abbIRB4_params". In the figure down below we can see all the coordinate frames for all joints put out that is represented in the DH table, see figure (2):

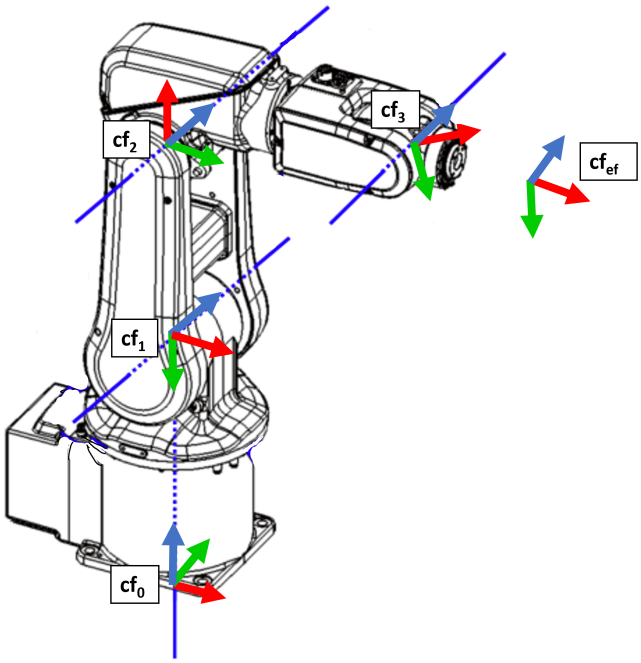

Figure 2: Figure showing all the coordinate frames for all the joints on the robot.

When the DH matrix and all parameters are defined we wanted to construct homogeneous transformations (HT) out of all the joints. I constructed my own function for converting the DH matrix to a HT matrix. Where for each row in the DH matrix the following template for converting between DH and HT with the distal method was used:

$$HT_{n-1}^{n} = \begin{bmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & a\cos(\theta) \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Where n is the number of the row or frame in the DH matrix. By multiplying the HT matrices from 0 to 1, 1 to 2 etc we get the HT matrix HT_0^2. We can do this for

all the frames going from to wherever we decide. And the function (see attached code: "DHtoHT_distal.m") can handle that just by defining the DH and from to to inputs. With the completed function we just calculated the asked HT matrices and tried the function in the grader which gave us thumbs up. I will not print the resulting HT-matrices here because they are graded and okay so if you want to see them run the attached *main.m* script and they are printed to the command window. With this function, the DH and the defined parameters we can finish the asked function "Model/getAbsoluteHT_abbIRB4". See attached files.

Exercise 2

For this task we are asked to compute the Jacobian for the end-effector wrt base frame or $Link_0$. In order to do this we needed to construct another function that calculates the Jacobian for the asked from to to frame. (See attached code: "HTtoJ.m"). We started by using the previous constructed function for converting from DH to HT in the Jacobian function. Therefore it as well as the DHtoHT function takes the DH and from to to as inputs. To calculate the Jacobian i followed the given instructions from the lecture notes given down below, see figure (3):

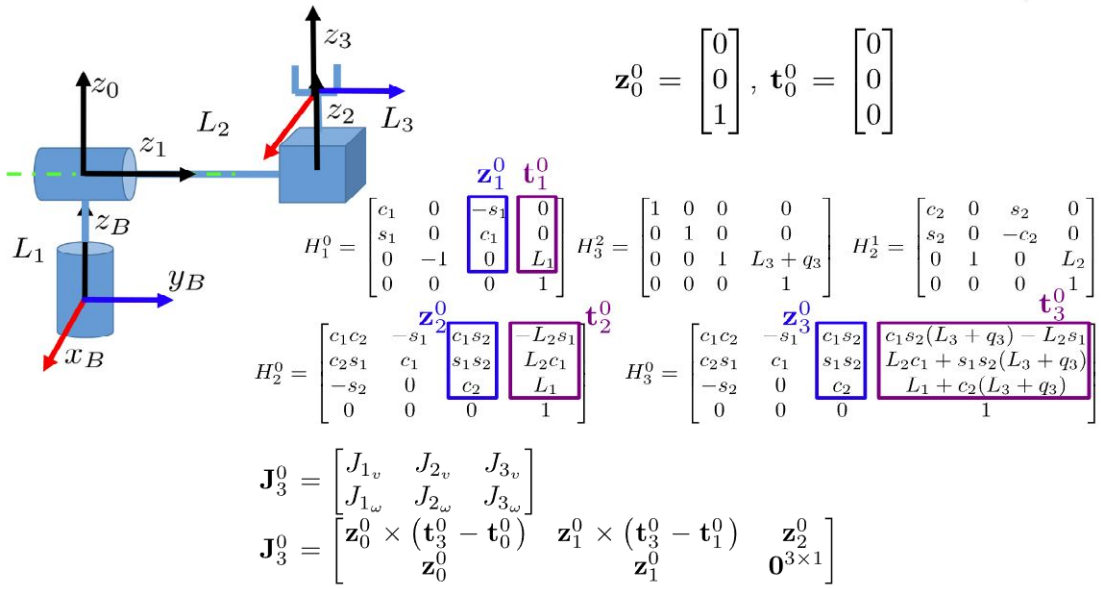

Figure 3: Showing and example and the way of computing the Jacobian for a given system. Taken from the lecture notes.

By calculating all the data within the function that we want like first the HT matrices and then take the position array, the last column in the rotational matrix and then put them together use the command cross to cross multiply and then we got a Jacobian matrix for the asked frame. Since we want the Jacobian from base to the end-effector all values will be filled in the Jacobian matrix with the size 6x4 where 4 is the amount of frames or joints. In case we want for example the Jacobian from base wrt another joint or frame we use that frame as reference and calculate the matrix to that frame and then fill the rest of the matrix with zeros so it has the right dimensions in our case 6x4. As i described in previous question the finished Jacobian matrix will not be printed here but it was okay in the grader and if you want to see the results run as mentioned before the attached script "main.m" and the answer will be printed to the command window. With the finished Jacobian function we can finish the asked script "Model/J_EF_abbIRB4" and "Model/DifFK_abbIRB4". The second of the two simply calculates the Jacobian and multiply it with the $\dot{q}$ array which gives a 6x1 matrix instead describing the linear and angular velocities of in our case the end-effector with respect to the robot base.

Exercise 3

In this task we are supposed to validate the results from previous exercises in a plot of the robot where by clicking we can go to the next frame. I finished the asked functions and script with some changes in the paths so on and then received the following results when plotting, see figure (4) and (5) down below:

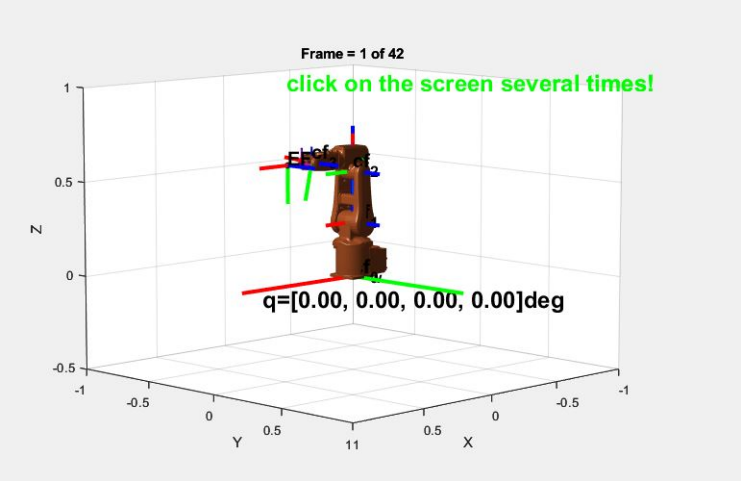

Figure 4: Plot of the starting position of the robot where, q = [0, 0, 0, 0]deg.

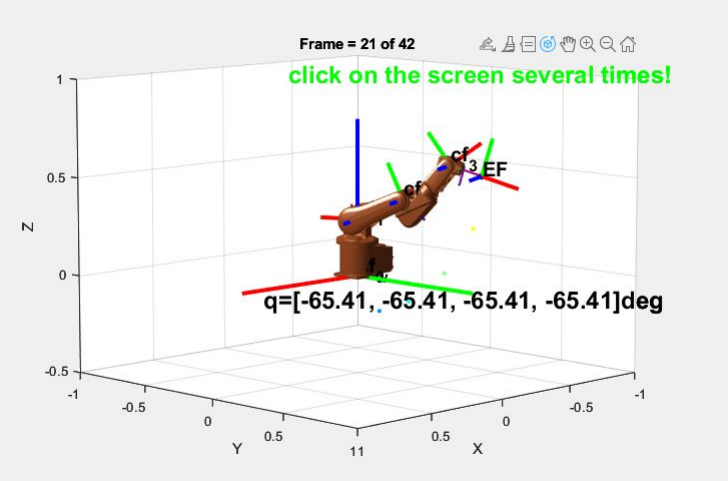

Figure 5: Plot of the frame number 21 of the robot where, q = [-65.41, -65.41, -65.41, -65.41]deg.

We can see that this is the exact same as the given figures in the assignment PM which validates the results. We can also see that when going through all the frames the q's are the same which also validates that all of the coordinate frames are aligned.

Exercise 4

In this task we are supposed to finish the script "02_ManipulabilityEllipsoid/manipulability_abb_irb4". In this script there are several "Todos", these todos starts by defining world to base frames which is the same in our case and gives us the identity matrix. Then we collect the rotational matrix from this HT. The script is followed by extracting the base to end-effector frame and world to end-effector frame which in our case is the same again. These are extracted by firstly using the function "[HT_0, HT_W] = getAbsoluteHT_abbIRB4(Q,L, H0_W)" and then extracting from this. Then from these extracting the position array and rotational matrix as well. We continue by extracting the linear and angular velocities given from "Xp4_0=DifFK_abbIRB4(Q, Qp, L)" where the 3 first rows is the linear velocities and the last 3 rows is the angular. To get the right dimensions a 0 is added the the both. Then we used the previously defined "jef_0=J_EF_abbIRB4(Q, L)" to extract the Jacobian. Of this Jacobian we caluclated the manipulability index by using the following formula: $\sqrt{(det(J * J^\top))}$. We followed up with calculating the time derivative of R_0^4. Then the skew matrix with the given formula: $S([wx; wy; wz]) = dR/dtR^\top$, then from this extracting the angular velocities in the given positions above ([wx;wy;wz]) from the skew matrix. See equation down below for a clearer view:

$$S = \begin{bmatrix} 0 & -wz & wy \\ wz & 0 & -wx \\ -wy & wx & 0 \end{bmatrix} \tag{2}$$

Lastly we want to map these angular velocities to the world coordinate frame which is simply done by multiplying the rotational matrix with the newly extracted angular velocities. Then add a 0 for the sake of dimensions.

Exercise 5

With the finished functions from the previous tasks we can now simulate the system in Simulink. We changed the simulation time to 20 seconds instead and started

the simulation. The results from the simulation are shown down below in figure (6) and (7):

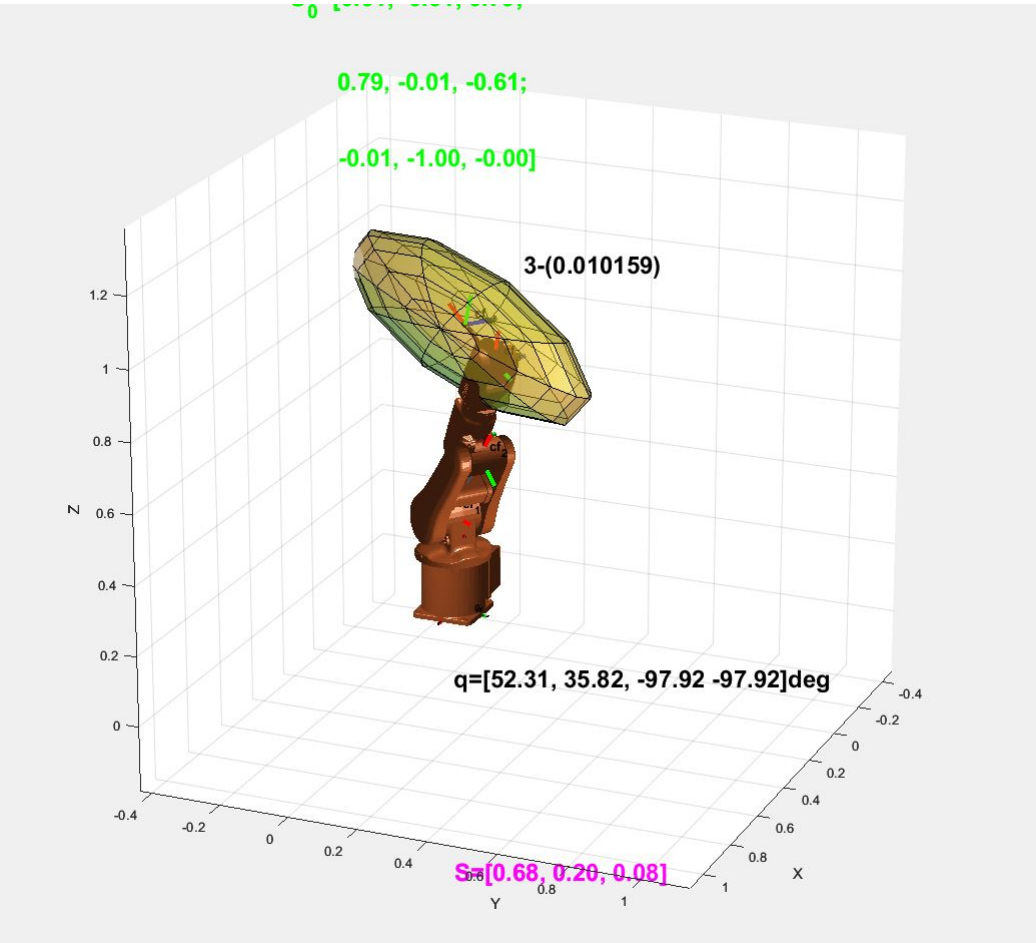

Figure 6: Plot of the manipulability ellipsoid when the robot is in the joint position, q = [52.31, 35.82, -97.92, -97.92]deg.

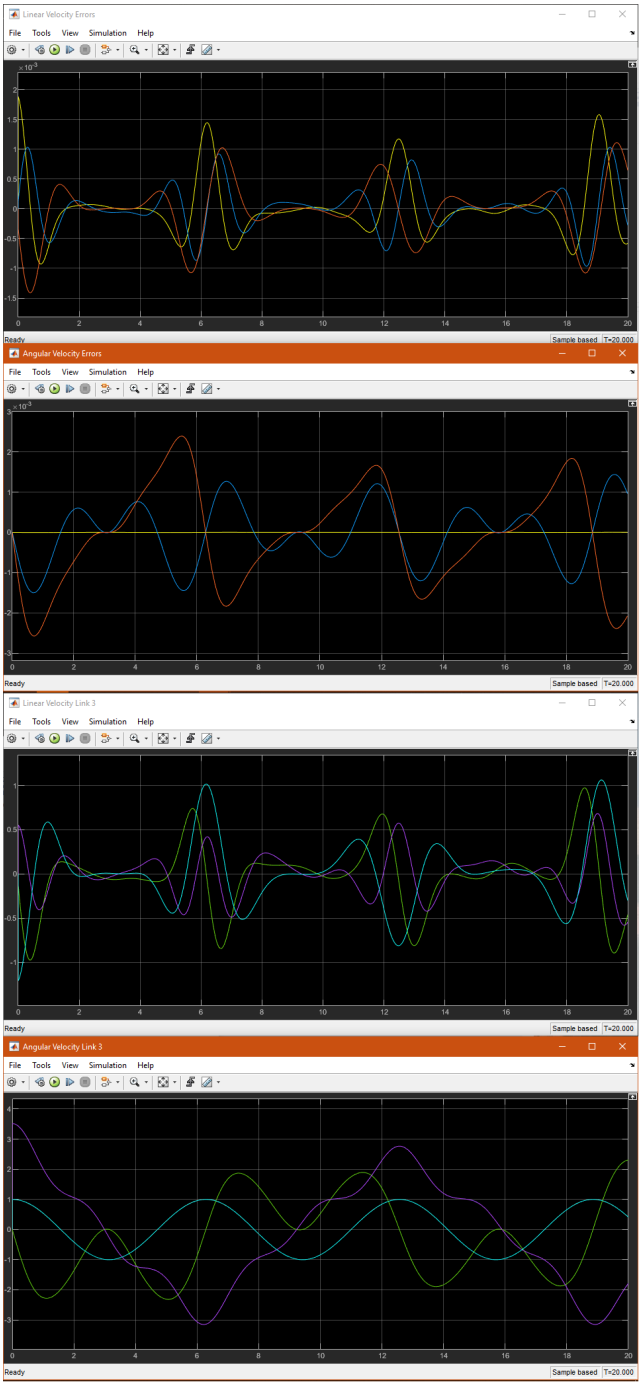

Figure 7: Plot of the linear and angular velocities along with the errors of each respective when simulating over 20 seconds.

We can see from the figures that the results are the same as the results in the assignment PM. This validates that the functions are correct and the simulation is working as it should be. We can also see that as expected the errors are very small.

For validation purposes we tried the function to switch to a specific joint position and try to imitate the figures given in the assignment PM. We can see by comparing that they are the exact same which validates the functions and the simulations. We tried the positions $q = [0, 0, 0, 0]$, $q = [0, \frac{\pi}{2}, 0, 0]$ and $q = [0, -\frac{\pi}{2}, 0, 0]$. As can be seen in the figures down below (8) they are the same as the ones given in the assignment PM.

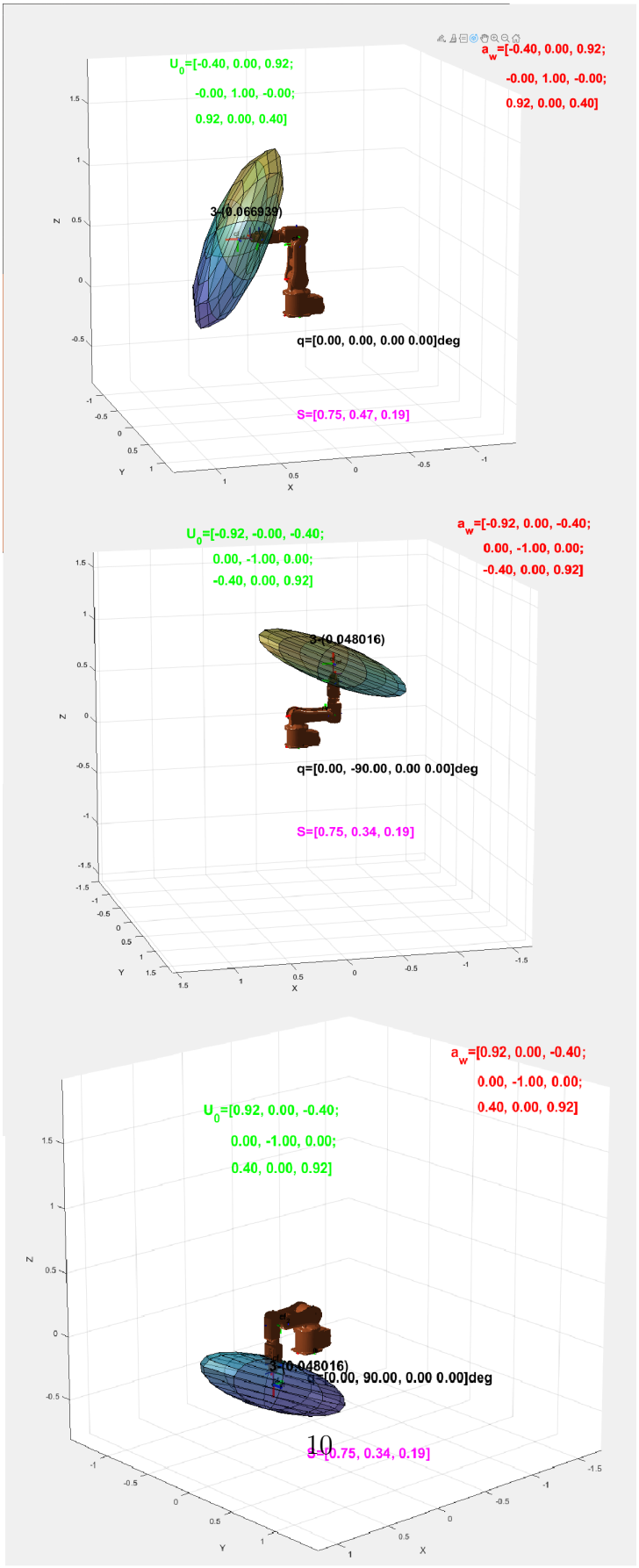

Figure 8: Plot of the manipulability ellipsoid when the robot is in different joint positions.