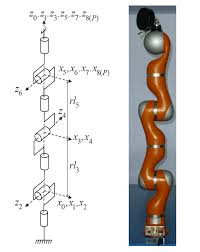
Jose Corona

Home Work 4

**Implement function for calculating the robot Jacobian**





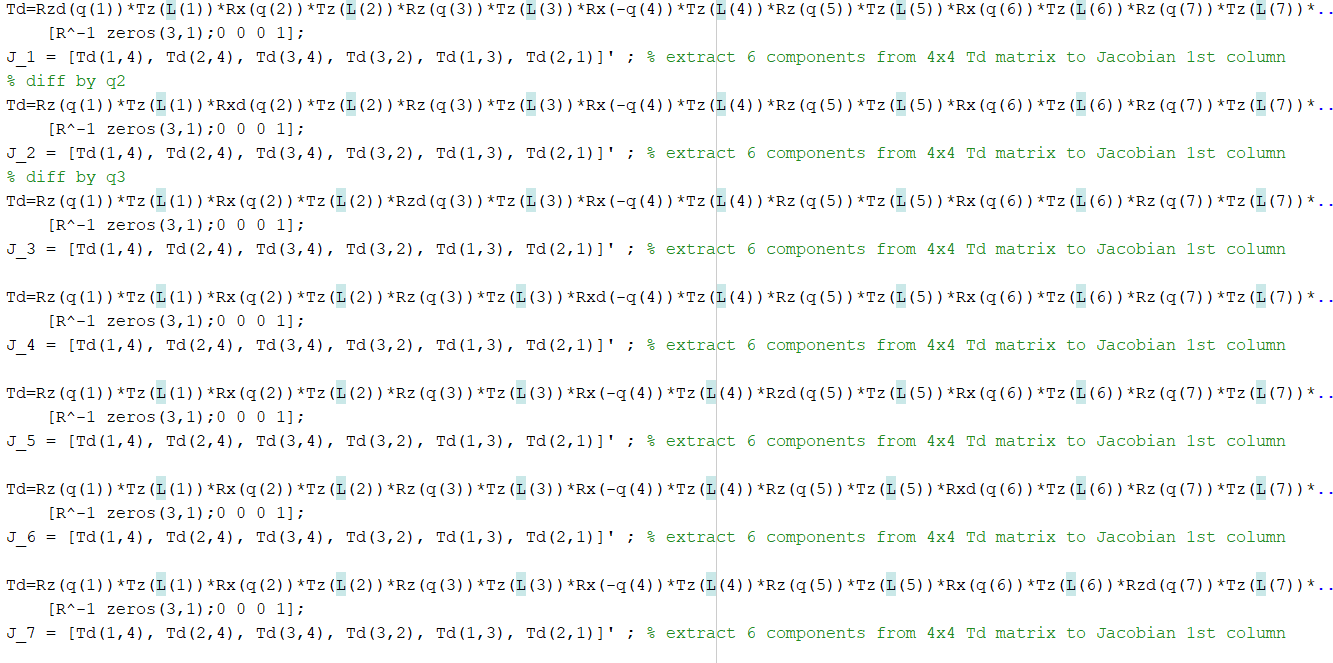
FK: Implemented using rotation in z, x and translation in z matrices.

T=(Rz(q(1))\*Tz(L(1))\*Rx(q(2))\*Tz(L(2))\*Rz(q(3))\*Tz(L(3))

\*Rx(-q(4))\*Tz(L(4))\*Rz(q(5))\*Tz(L(5))\*Rx(q(6))\*Tz(L(6))

\*Rz(q(7))\*Tz(L(7)))

Jacobian: Numerical method





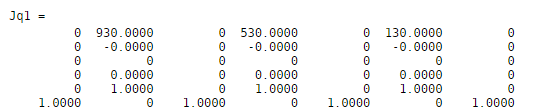
Test of the Jacobian function

%Jacobian test

L\_test1 = [340 200 200 200 200 126 4];

q\_test1 = [pi/2 0 0 0 0 0 0];

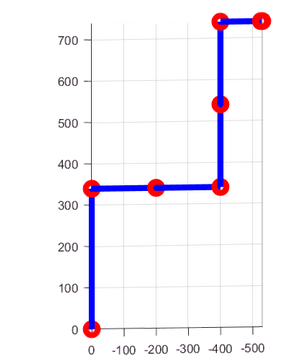
Jq1=Jacobian(q\_test1,L\_test1)



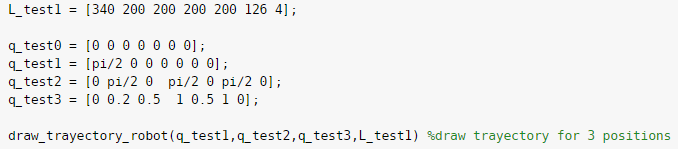
**Implement simple robot motion visualization**

Draw the robot

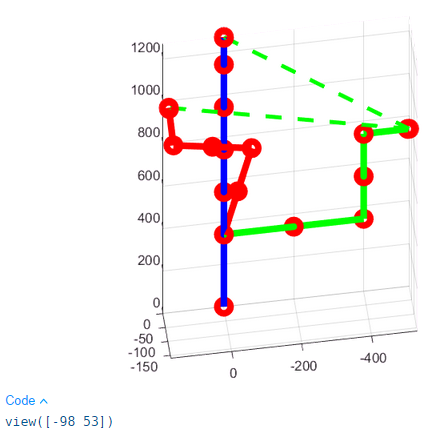




Draw the robot trajectory position

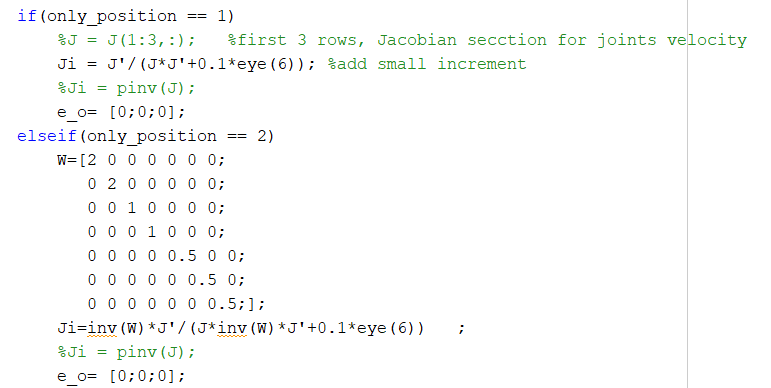


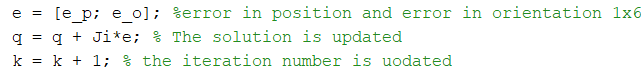
The first position is the robot arm of color blue, the second position is the robot of color green and the last position is the robot of color red. The trajectory is the line of color green.



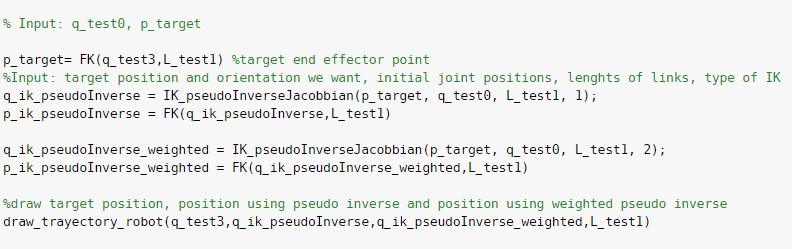
**3. Implement IK function for the robot based on**

1. Pseudoinverse and Weighted pseudoinverse: I implement both methods in the function. In the weighted pseudoinverse I increase the weighted of the first and second joints so they take priority.

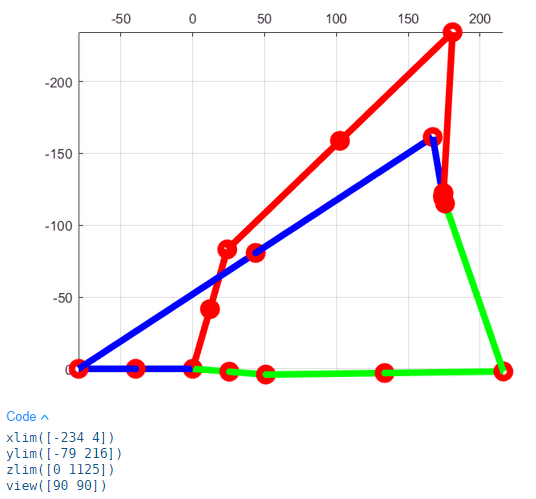


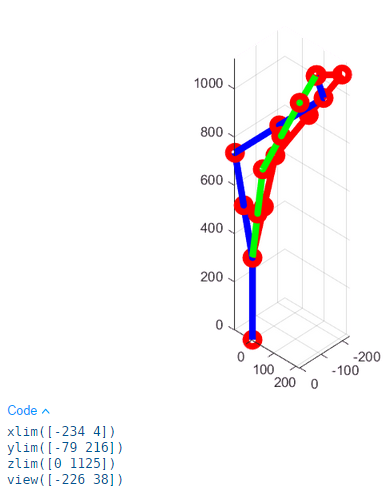
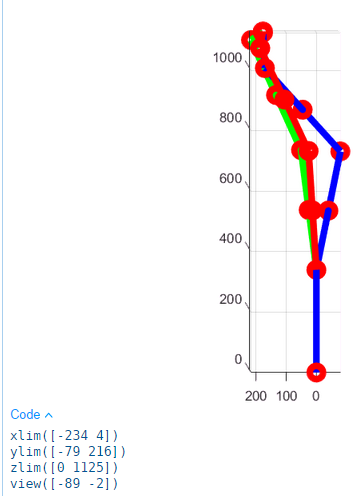
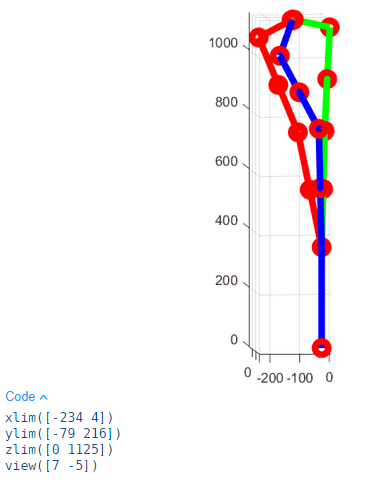


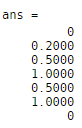
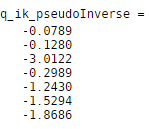
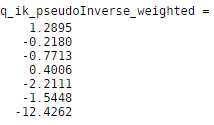
Then I visualize the arm position.



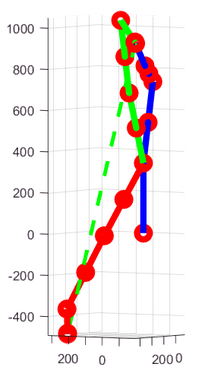
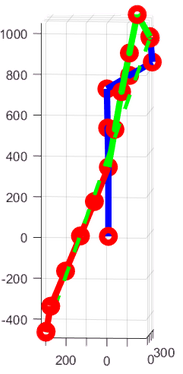
In the next figure, the arm of color blue is the target joint position, in green is the pseudo inverse solution and in color red is the solution using the weighted pseudo inverse Jacobian.



Comparing the solution, we could see that in the weighted version move a little bit more the first joint. In other position the weighted version diverges, so we have to take into account singularities in the workspace.

b. Damped Least Squares :

**Link Git hub:**

<https://github.com/Jose-R-Corona/AR-HomeTask4>