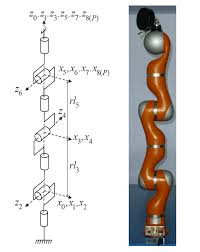
Jose Corona

Home Work 4

**Implement function for calculating the robot Jacobian**





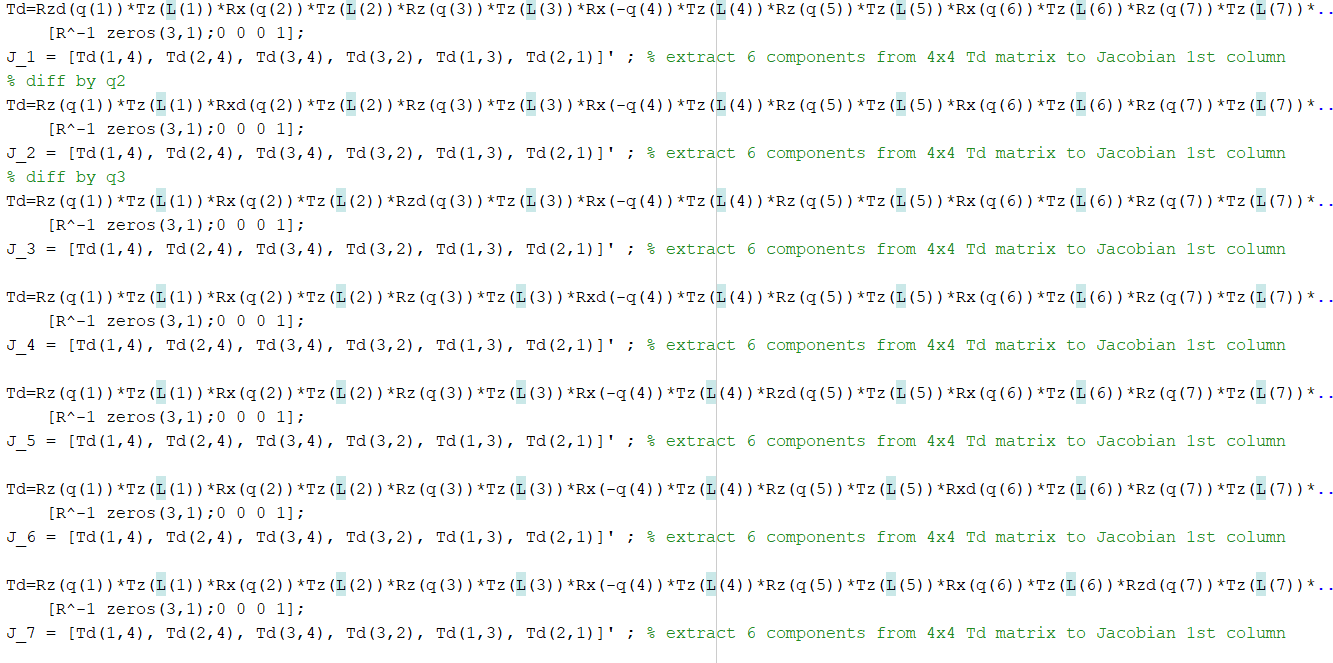
FK: Implemented using rotation in z, x and translation in z matrices.

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





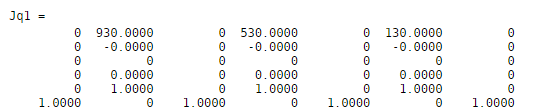
Test of the Jacobian function

%Jacobian test

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

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

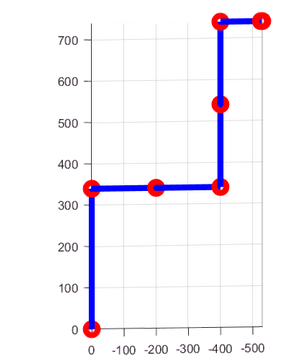
Jq1=Jacobian(q\_test1,L\_test1)



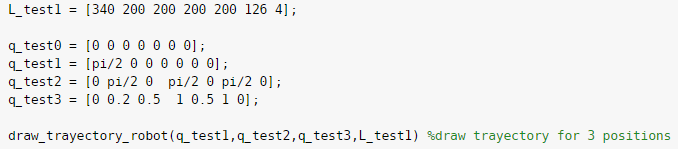
**Implement simple robot motion visualization**

Draw the robot

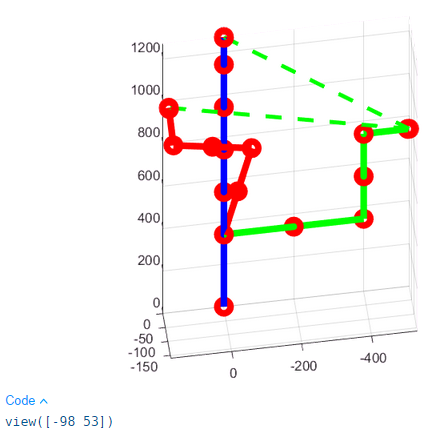




Draw the robot trajectory position, 3D visualiation

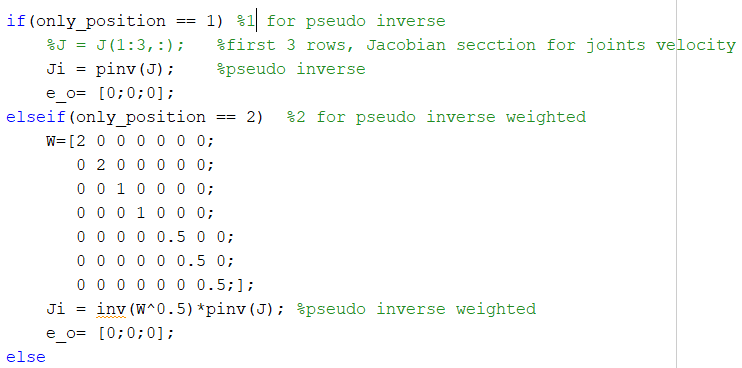


The first position is the robot arm of color blue, the second position is the robot of color green and the last position is the robot of color red. The trajectory is the line of color green.

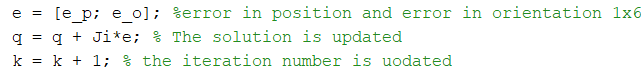


**3. Implement IK function for the robot based on**

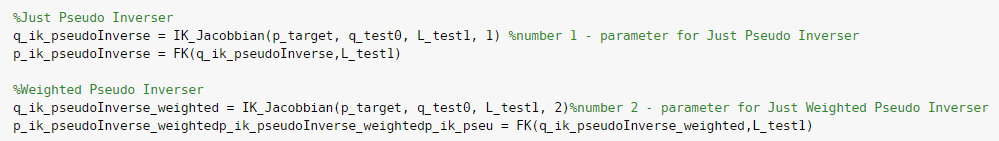
1. Weighted pseudoinverse: I implement just pseudoinverse and Weighted pseudoinverse methods in one function. In the weighted pseudoinverse I increase the weighted of the first and second joints so they take priority.



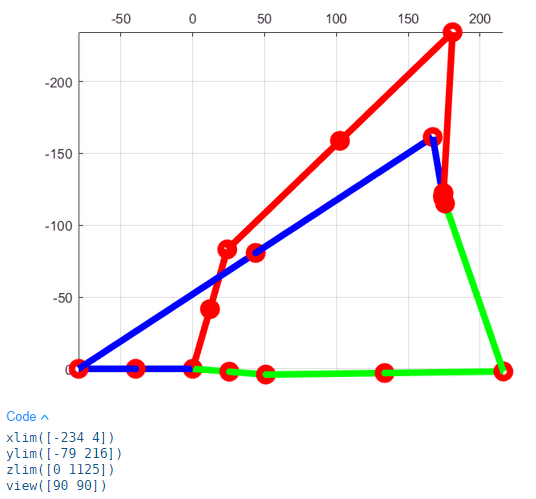
And calculated de joint position

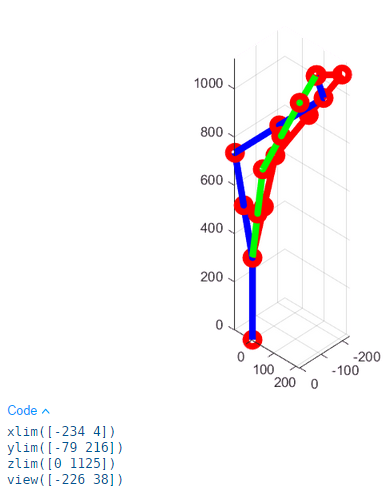
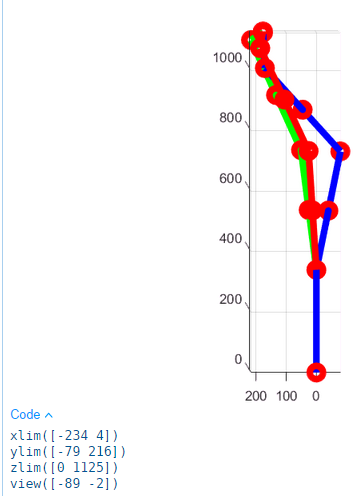
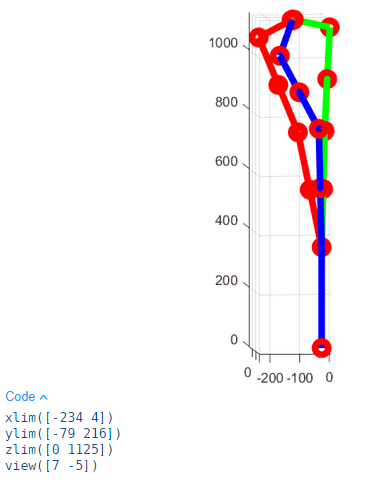


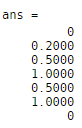
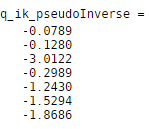
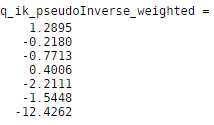
Then I visualize the arm position.



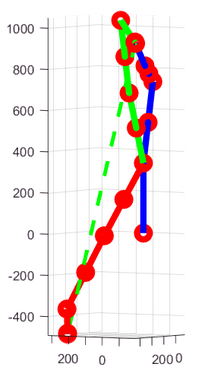
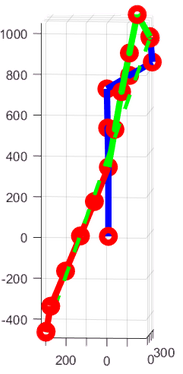
In the next figures, are presented different views of one target position of the arm. The arm of color blue is the target joint position, in green is the pseudo inverse solution and in color red is the solution using the weighted pseudo inverse Jacobian.



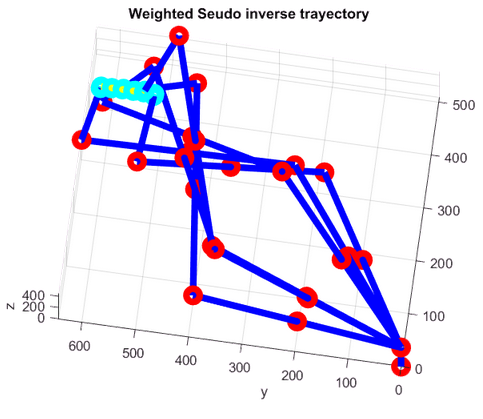
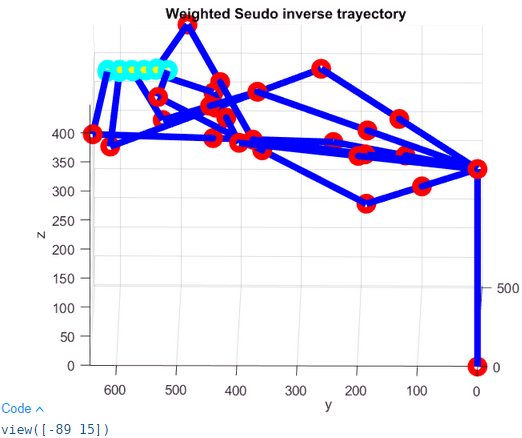
  

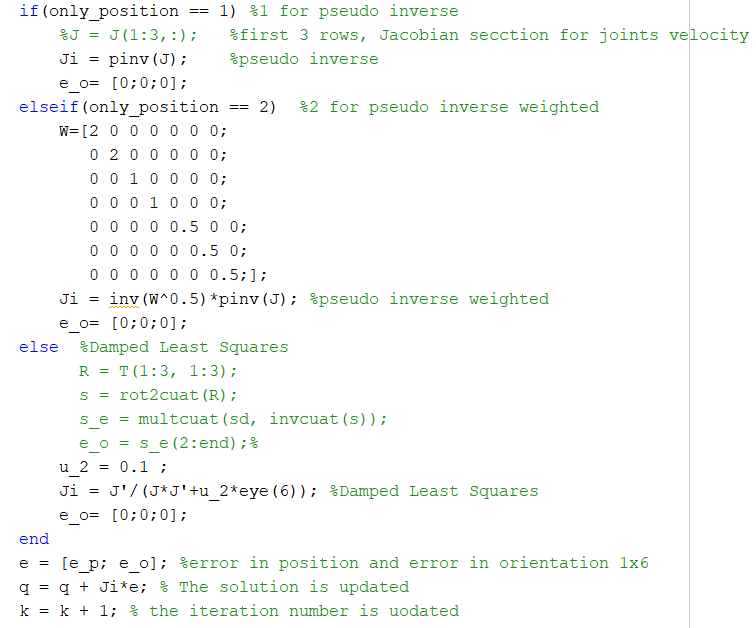
I try other positions, and in some of them the weighted and pseudoinverse version diverges, so we have to take into account singularities in the workspace.

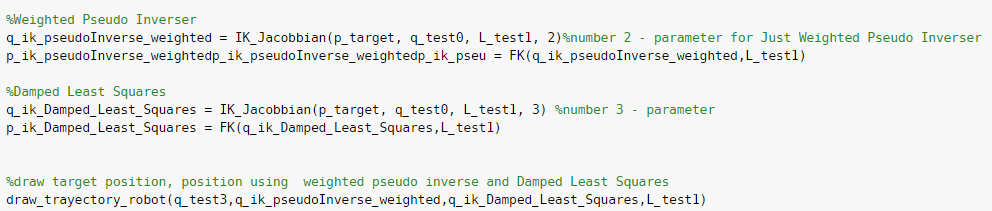
In the next figure we could the robot fallowing a trayectoy, where the final effector is in color cyan, the trayectory line is color yellow and the green dotted line follows the final effector trayectory. The trayectory to follows was a stright line in Y from 500 to 620, X=400 and Z=400

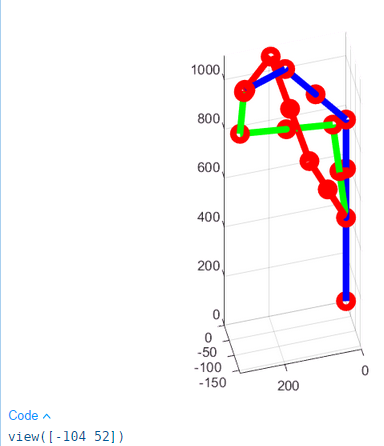
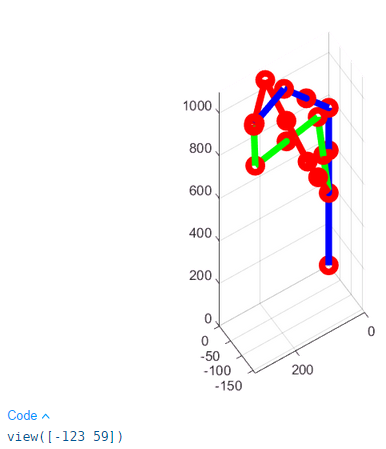
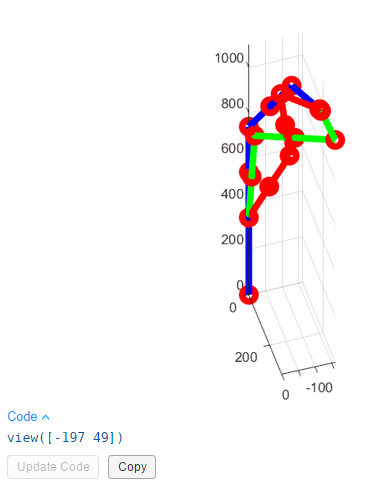
b. Damped Least Squares : I implemented this method in the same function.



And visualize a targe position

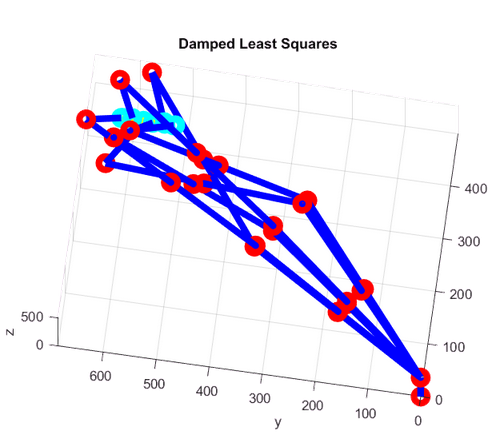
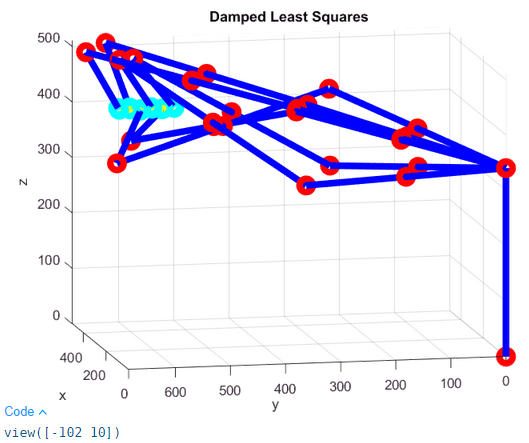


In the next figures, are presented different views of one target position of the arm. The arm of color blue is the target joint position, in green is the weighted pseudo inverse Jacobian and in color red is the solution using the Damped Least Squares.

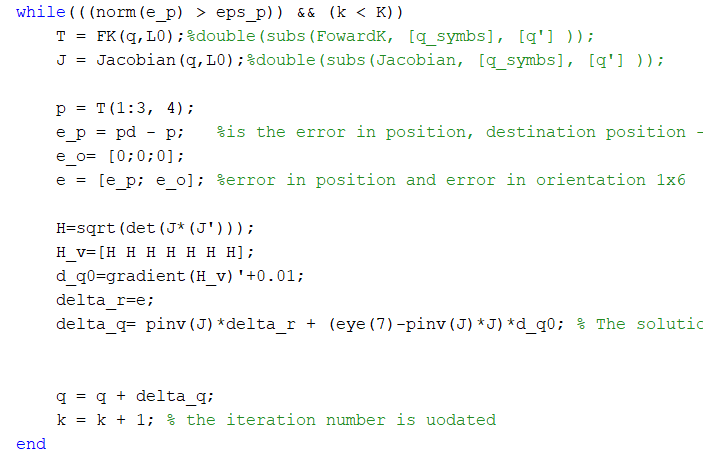
I try other positions, and in some of diverges, so for Jacobian based methods we have to take into account singularities in the workspace.

In the next figure we could the robot fallowing a trayectoy, where the final effector is in color cyan, the trayectory line is color yellow and the green dotted line follows the final effector trayectory. The trayectory to follows was a stright line in Y from 500 to 620, X=400 and Z=400

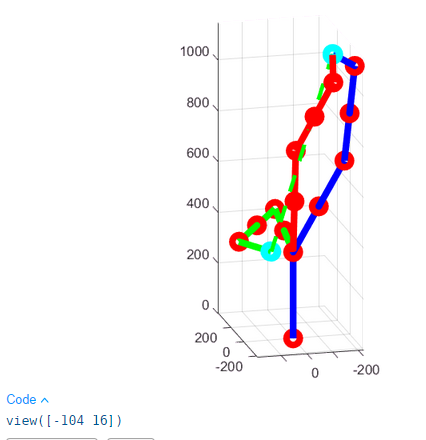
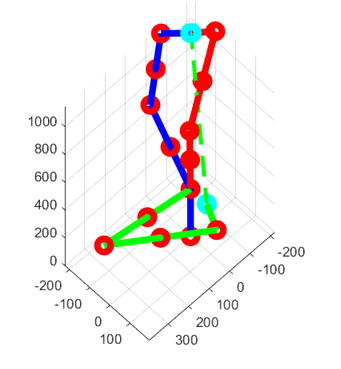
 

C. Null space method :

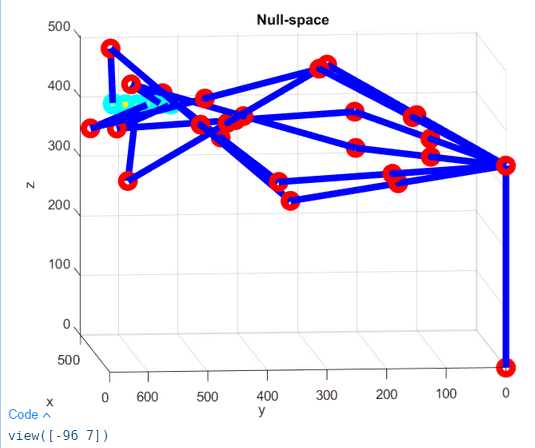
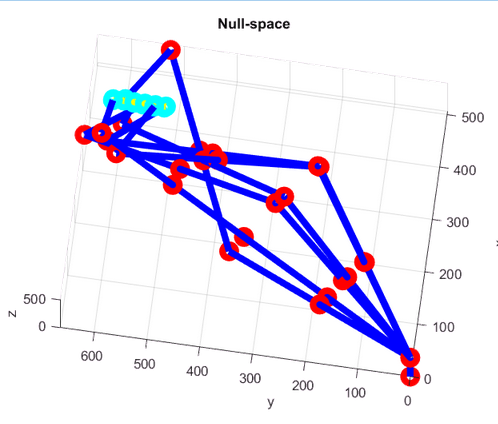
I implemented Null space in other file. Becouse of the determinant was just one value I copy to a vector fo 7x1 and applied the gradient.



In the next figures, are presented different views of one target position of the arm. The arm of color blue is the target joint position, in green is the null space solution with joint initial position zero and in color red is the solution using other initial position.



In the next figure we could the robot fallowing a trayectoy, where the final effector is in color cyan, the trayectory line is color yellow and the green dotted line follows the final effector trayectory. The trayectory to follows was a stright line in Y from 500 to 620, X=400 and Z=400



**Link Git hub:**

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