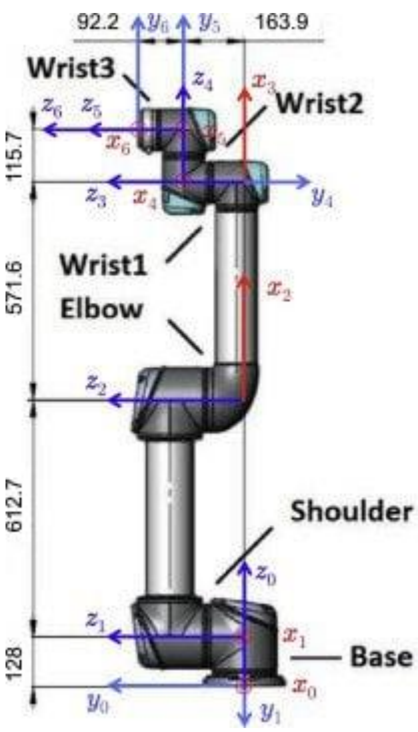
Jose Corona

# Final Exam

1.Implement IK function for theUR10robot which takes only Cartesian position (x, y, z) as the input based on deferent approaches:




| $i$ | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | $-\pi/2$ | $d_1 = 128$ | $q_1$ |
| 2 | $a_2 = 612.7$ | 0 | 0 | $q_2$ |
| 3 | $a_3 = 571.6$ | 0 | 0 | $q_3$ |
| 4 | 0 | $-\pi/2$ | $d_4 = 163.9$ | $q_4$ |
| 5 | 0 | $\pi/2$ | $d_5 = 115.7$ | $q_5$ |
| 6 | 0 | 0 | $d_6 = 92.2$ | $q_6$ |

FK

```
T=(Rz(q(1))*Tz(L(1))*Rx(q(2))*Tz(L(2))*Rx(-q(3))*Tz(L(3))*Rx(q(4))*Rz(q(5))*Tz(L(4))*Rx(q(6)));
```

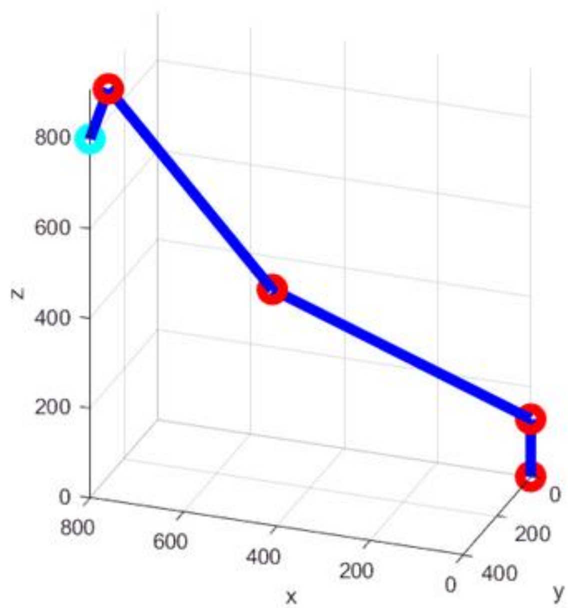
### a. Weighted pseudoinverse:

I choose in the Weighted matrix, a higher weight for the first two motors.

```
W=[2 0 0 0 0 0;
   0 2 0 0 0 0;
   0 0 2 0 0 0;
   0 0 0 1 0 0;
   0 0 0 0 1 0;
   0 0 0 0 0 1;];
Ji = inv(W^0.1)*pinv(J); %pseudo inverse weighted
e_o= [0;0;0];

e = [e_p; e_o]; %error in position and error in orientation 1x6
q = q + Ji*e; % The solution is updated
k = k + 1; % the iteration number is updated
```

The target point was (800,400,800)

```
pc_target =
   800
   400
   800
q_ik_pseudoInverse_weighted =
   1.0e+03

    0.0020
    0.0010
    0.0004
    0.0022
    0.0009
    9.1386
pc_ik_pseudoInverse_weighted =
  800.0000
  400.0000
  800.0000
```

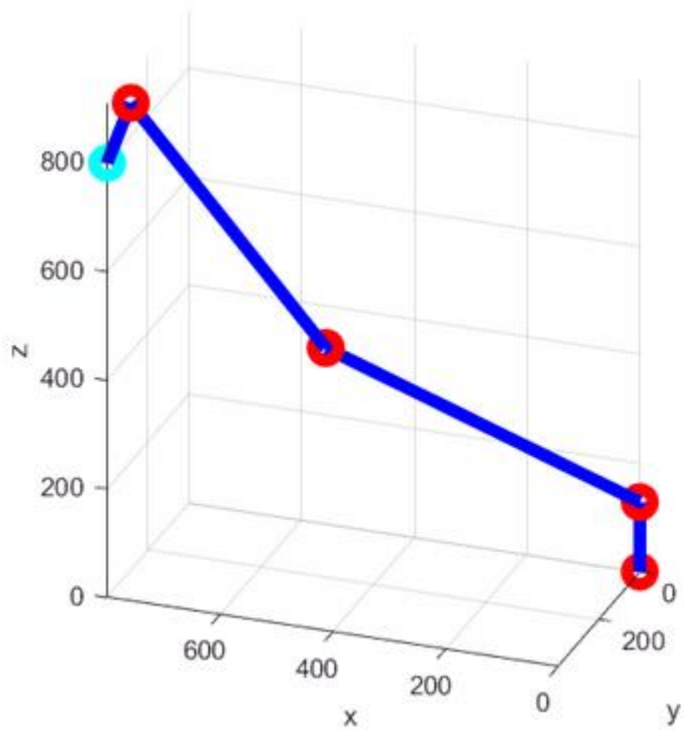



## b. Damped Least Squares

```
T = FK(q,L0);%double(subs(FowardK, [q_symbs], [q'] ));
J = Jacobian(q,L0); %double(subs(Jacobian, [q_symbs], [q'] ));

p = T(1:3,4);

e_p = pd - p;  %is the error in position, destination position -

e_o= [0;0;0];
u_2 = 0.1 ;
Ji = J'/(J*J'+u_2*eye(6)); %Damped Least Squares

e = [e_p; e_o]; %error in position and error in orientation 1x6
q = q + Ji*e; % The solution is updated
k = k + 1; % the iteration number is uodated
```

The target point was (800,400,800)

```
pc_target =
   800
   400
   800
q_ik_Damped_Least_Squares =
     2.0344
     1.0364
     0.4266
     2.1807
    -0.1828
   -54.2640
pc_ik_Damped_Least_Squares =
   800.0000
   400.0000
   800.0000
```




c. Null-space method with objective functions H(q) which maximize the distance from joint limits

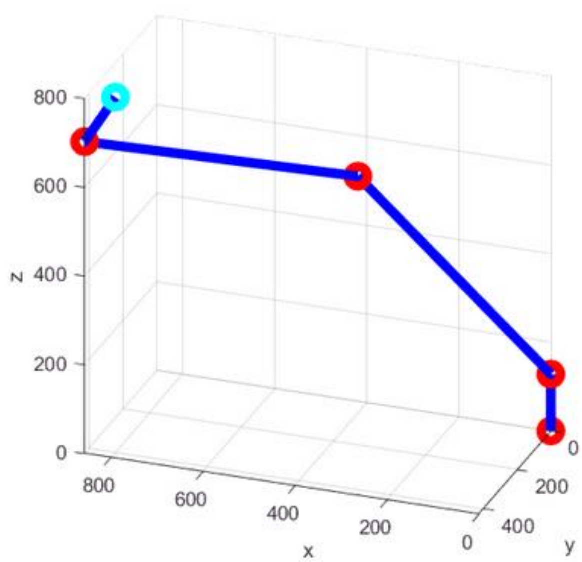
I implemented Null space in other file. I calculated the jacobian for each joint, not all the joint at the same time to get a differentiation for each join for dq0 . Since the determinat of everything is just one value, was always cero if I use the jacaobian of all joints.

```
for i=1:6
    onlinjoinq=[0 0 0 0 0 0];
    onlinjoinq(i)=q_last(i);

    J_last=Jacobian(onlinjoinq,L0);
    H_last=real(sqrt(det(J_last*(J_last'))));

    onlinjoinq=[0 0 0 0 0 0];
    onlinjoinq(i)=q_next(i);
    J_next=Jacobian(onlinjoinq,L0);
    H_next=real(sqrt(det(J_next*(J_next'))));

    Numerical_dev= 0.1*(H_next-H_last);
    d_q0(i)=Numerical_dev;
end
```

```
d_q0=d_q0';
delta_r=e;

delta_q= pinv(J)*delta_r + (eye(6)-pinv(J)*J)*d_q0; % The solution is updated

q = q + delta_q;

k = k + 1; % the iteration number is uodated
```

# The target point was (600,900,400)

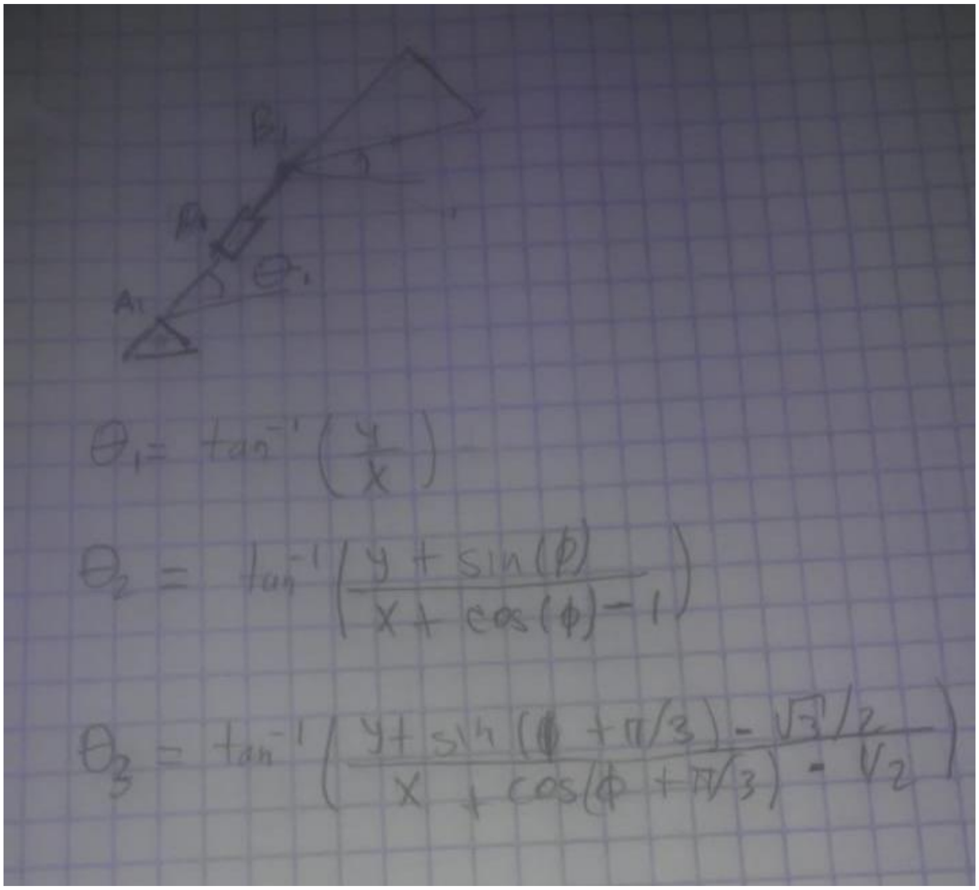
```
pc_target =
   600
   900
   400
q_ik_Null_Space =
     2.5536
     1.2134
    -0.4269
    -2.2114
     0.8337
  -237.3795
pc_ik_Null_Space =
   600.0000
   900.0000
   400.0000
```



2.Implement functions for the3-RPR parallel

**Inverse Kinematics**

Knowing the orientation for each leg




we could find the displacement in the joint. As in presentation of lecture 9

$$
\begin{aligned}
\rho_1^2 &= x^2 + y^2 \ , \\
\rho_2^2 &= (x + l_2 \cos\Phi - c_2)^2 + (y + l_2 \sin\Phi)^2 \ , \\
\rho_3^2 &= (x + l_3 \cos(\Phi + \theta) - c_3)^2 + (y + l_3 \sin(\Phi + \theta) - d_3)^2
\end{aligned}
$$

Asumming:

L1=L2=L3=1. Equilateral triangle . Then $\theta$=pi/3 , And the positon of the base (ground) for each leg will be:

A1=[0,0]   A2=[1,0]  A3=[0.5, $\sqrt{3}/2$]

Then c2=1 ,   c3 = 0.5  and  d3=$\sqrt{3}/2$

We could get

```
function q=IK_RPR(p,phi)
x = p(1);
y=p(2);

q1=sqrt(x^2+y^2);
q2=sqrt((x+cos(phi)-1)^2 + (y+sin(phi))^2);
q3=sqrt((x+cos(phi+pi/3)-1/2)^2 + (y+sin(phi+pi/3)-sqrt(3)/2)^2);

q=[q1;q2;q3];
end
```

**For the Forward Kinematics**:

$$\begin{aligned}
\rho_1^2 &= x^2 + y^2 \ , \\
\rho_2^2 &= x^2 + y^2 + Rx + Sy + Q \ , \\
\rho_3^2 &= x^2 + y^2 + Ux + Vy + W \ .
\end{aligned}$$

$$\begin{aligned}
\rho_1^2 &= x^2 + y^2 \ , \\
\rho_2^2 - \rho_1^2 &= Rx + Sy + Q \ , \\
\rho_3^2 - \rho_1^2 &= Ux + Vy + W \ .
\end{aligned}$$

## Direct kinematics of 3-RPR robot

Solution for position

$$x = -(SA_1 - VA_2)/(RV - SU)$$
$$y = (RA_1 - UA_2)/(RV - SU)$$

$$A_1 = \rho_3^2 - \rho_1^2 - W \qquad A_2 = \rho_2^2 - \rho_1^2 - Q$$

Solution for orientation $\quad \rho_1^2 = x^2 + y^2$

$$(SA_1 - VA_2)^2 + (RA_1 - UA_2)^2 - \rho_1^2(RV - SU)^2 = 0$$

$$T = \tan(\frac{\Phi}{2}) \quad , \quad \cos(\Phi) = \frac{1 - T^2}{1 + T^2} \quad , \quad \sin(\Phi) = \frac{2T}{1 + T^2}$$

Polynomial equation for orientation

$$C_0 + C_1T + C_2T^2 + C_3T^3 + C_4T^4 + C_5T^5 + C_6T^6 = 0$$

Each real solution of this equation gives a value of Φ, which in turn gives a pair *x*, *y*.

From the polynomial equation we could obtain

$$(c_3(\rho_1^2 - \rho_2^2 + 4c_2^2 - 4c_3c_2) + c_2(\rho_3^2 - \rho_1^2))t^3 + d_3(8c_3c_2 - 4c_2^2 + \rho_2^2 - \rho_1^2)t^2$$
$$+(c_3(\rho_1^2 - \rho_2^2) + \rho_3^2c_2 - 4d_3^2c_2 - \rho_1^2c_2)t + d_3(\rho_2^2 - \rho_1^2) = 0$$

## Jacobian

Let $\vartheta$ be the rotation angle of the platform around *C*
Let us define **Ω** as (0, 0, $\dot{\vartheta}$) and **ni** as the unit vectors of the legs. The velocity **VB** of the point *B* is

$$\mathbf{V_{B_i}} = \mathbf{V} + \mathbf{B_iC} \times \boldsymbol{\Omega} \qquad \mathbf{V_{B_i}} = \dot{\rho}_i \mathbf{n_i}$$

Equating the dot product by **ni** of the right terms of these equations leads to

$$\dot{\rho}_i = \mathbf{n_i}.\mathbf{V} + (\mathbf{CB_i} \times \mathbf{n_i}).\boldsymbol{\Omega}$$

constraint equations indicating that the robot motion are planar

$$\mathbf{V.z} = 0 \qquad \boldsymbol{\Omega}.\mathbf{y} = 0 \qquad \boldsymbol{\Omega}.\mathbf{x} = 0$$

The full velocity equations may therefore be written as:

$$\begin{pmatrix} \dot{\rho}_1 \\ \dot{\rho}_2 \\ \dot{\rho}_3 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathrm{J}^{-1}\mathbf{W} = \begin{pmatrix} \mathbf{n_1} & \mathbf{CB_1} \times \mathbf{n_1} \\ \mathbf{n_2} & \mathbf{CB_2} \times \mathbf{n_2} \\ \mathbf{n_3} & \mathbf{CB_3} \times \mathbf{n_3} \\ \mathbf{z} & \mathbf{0} \\ \mathbf{0} & \mathbf{x} \\ \mathbf{0} & \mathbf{y} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix}$$

which establish a full inverse kinematic jacobian

Git Hub

https://github.com/Jose-R-Corona/FinalExam-AR