ROBÓTICA 2021-I

LABORATORIO: ANÁLISIS Y SIMULACIÓN DE UN ROBOT INDUSTRIAL

PRESENTADO POR:

JOSE JAIRO GUTIERREZ MORENO jojgutierrezmo@unal.edu.co
JOHN SEBASTIAN PANCHE ESTUPIÑÁN jspanchee@unal.edu.co
HENRY OMAR MORENO JIMENEZ homorenoj@unal.edu.co

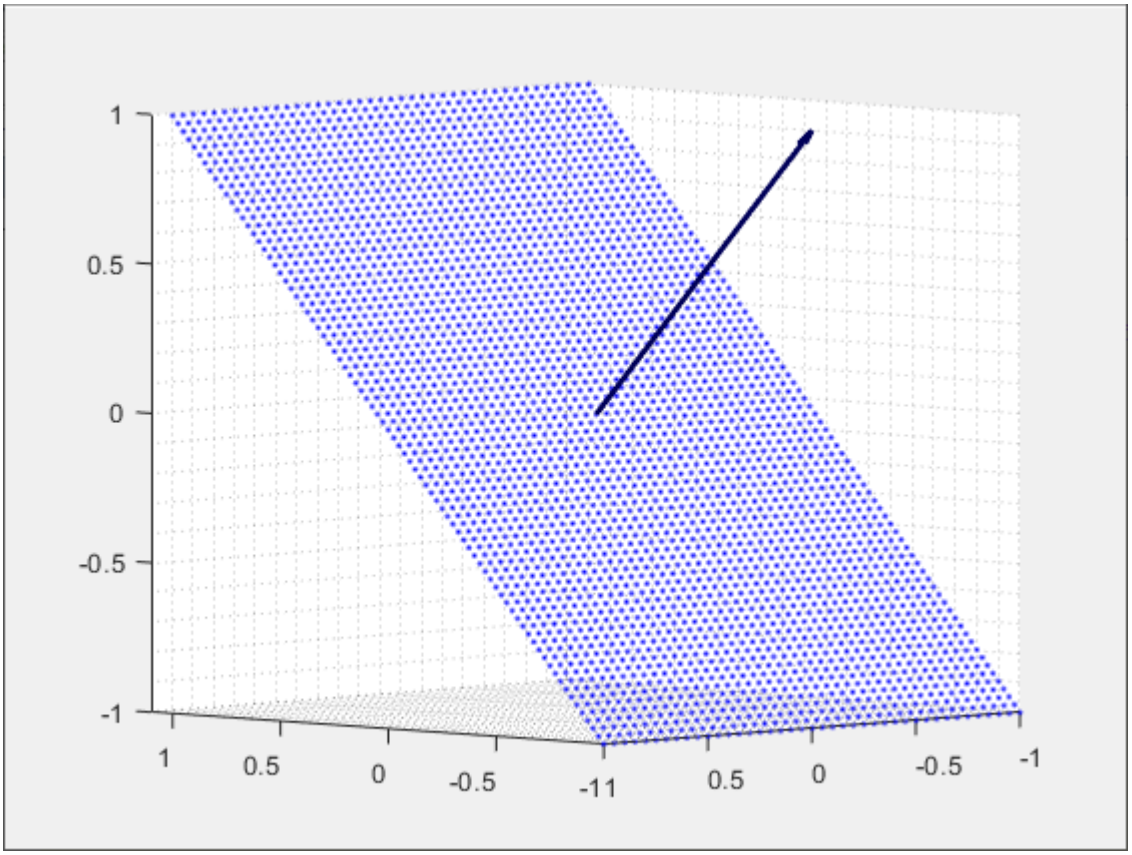
PARTE 1.

ROBOT Y RUTA

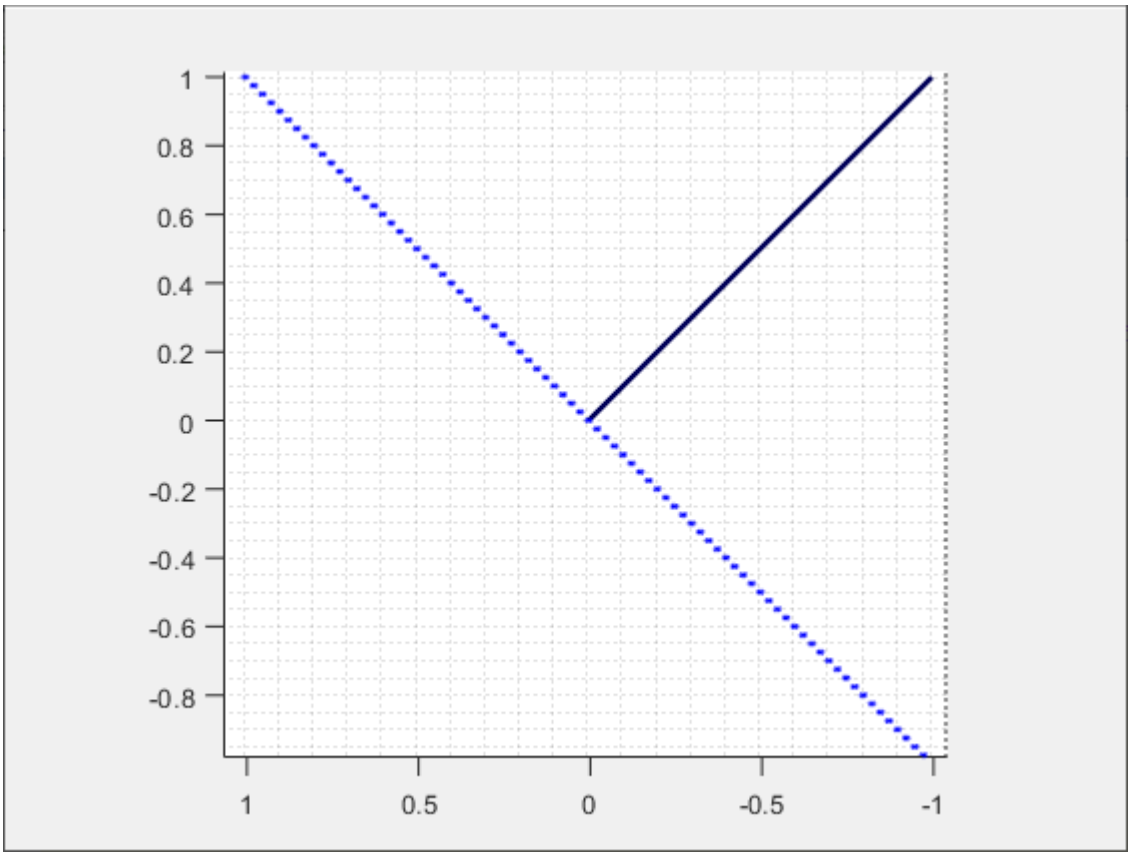
ASIGNACIÓN DEL ROBOT

ROBOT	**UNIVERSAL ROBOTS UR5 6-AXIS ROBOT**

Plano de trabajo normal al vector [-1, 0, 1]

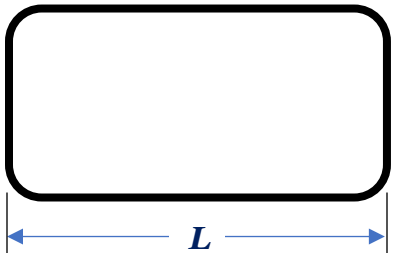


Para obtener una primera impresión de las condiciones en que se realizará la trayectoria, se grafican un conjunto de puntos en el plano de trabajo por medio de MATLAB junto con el vector indicado [-1, 0, 1]. A continuación se observa la vista lateral en el plano X-Z:



LOCALIZACIÓN DE RUTA

Para la selección de la ruta, se nos brindan dos posibles opciones en la guía de laboratorio, sin embargo dado que al final se espera realizar la integración con la trayectoria sobre la ruta elegida en este punto preliminar no se cuentan con criterios para la selección de la ruta, por esta razón y por decisión del equipo de trabajo se elige la opción del rectángulo de esquinas redondeadas (con un ángulo no especificado).

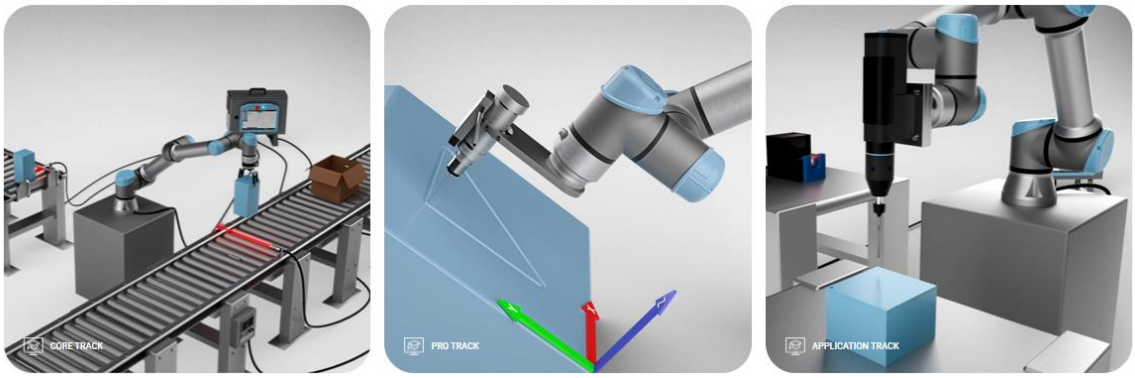


Respecto a la localización de la ruta en el espacio diestro de trabajo, primero se espera realizar la figura de la ruta sobre el plano y con centro en el origen, para posterior a esto desplazarla en uno de los ejes de modo que se ubique de frente al robot. La localización completa se detallará en la sección correspondiente del laboratorio.

CARACTERÍSTICAS Y MODELO DEL ROBOT

Acerca de...

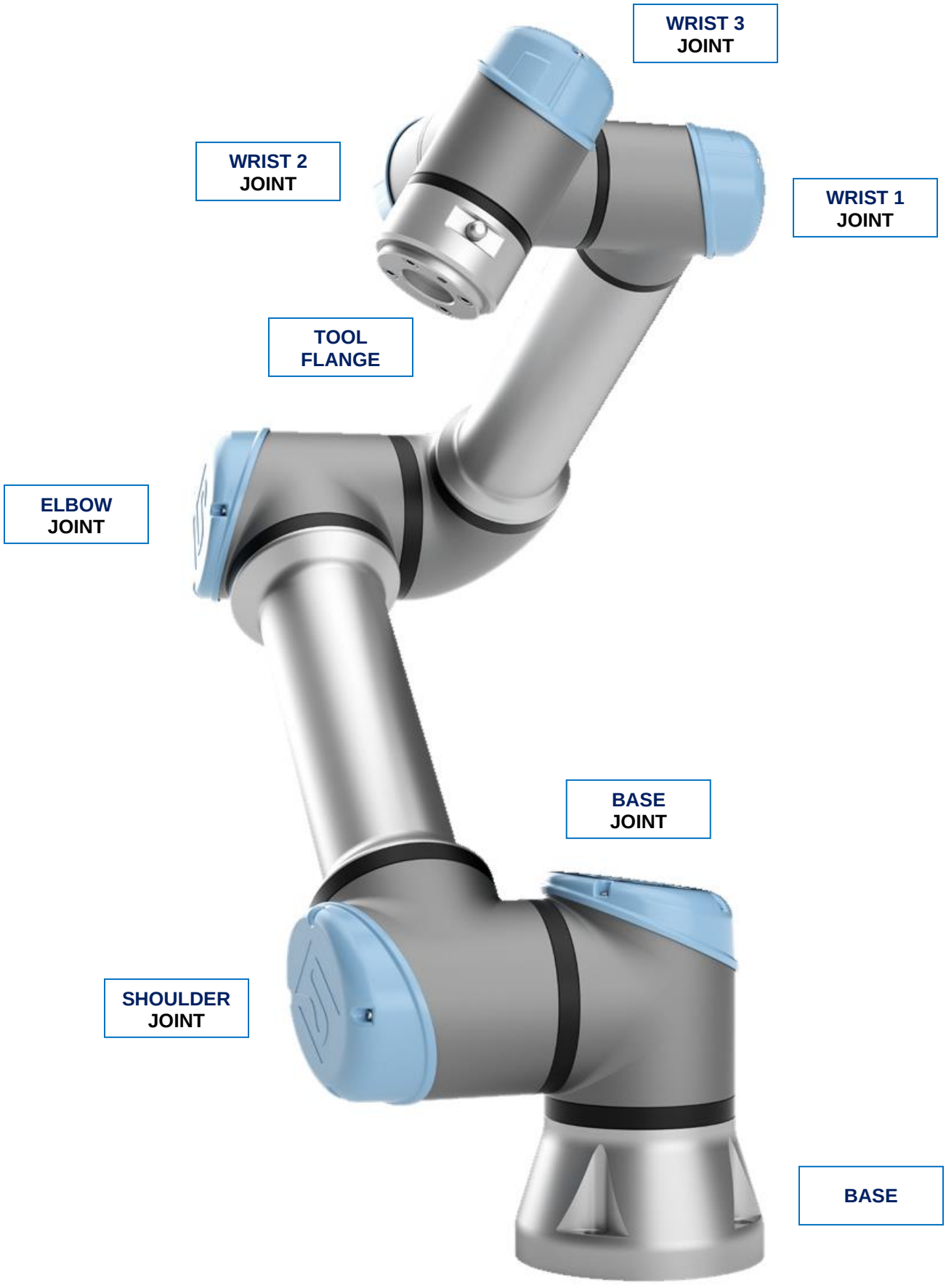
UNIVERSAL ROBOTS. Es una empresa dedicada al diseño y construcción de robots colaborativos (cobots) que se caracterizan por ser ligeros y flexibles. Fue fundada en el año 2005 por los ingenieros Esben Østergaard, Kasper Støy y Kristian Kassow de la Universidad de Dinamarca y gracias a la inversión de Syddansk Innovation con el principal objetivo de hacer la robótica industrial más accesible para las pequeñas y medianas empresas, dado que en su mayoría el mercado de la robótica estaba dominado por robots pesados y costosos. Para el año 2008 salió al mercado el primer robot colaborativo industrial **UR5**, un brazo robótico de propósito general para tareas de media carga y bastante ligero, implementando programación 3D en una interfaz de usuario intuitiva para una fácil configuración y operación. En 2015 la empresa fue adquirida por Teradyne Inc. por 285 millones de dólares. Actualmente, la compañía tiene 29 oficinas con presencia en 20 países y más de 740 empleados, perfilándose como una de las empresas de mayor crecimiento en el sector durante los últimos años.



https://academy.universal-robots.com/

Robot UR5e

El **UR5e** de Universal Robots es un robot colaborativo industrial ligero de tamaño medio ideal para automatizar aplicaciones de baja carga hasta 5 kg y un radio de alcance de 850 mm. Los propósitos generales del desarrollo de este robot son la versatilidad y la adaptabilidad. El UR5e está diseñado para integrarse perfectamente en una amplia gama de aplicaciones, fácil de programar y rápido de instalar.


WRIST 3 JOINT
WRIST 2 JOINT
WRIST 1 JOINT
TOOL FLANGE
ELBOW JOINT
BASE JOINT
SHOULDER JOINT
BASE

UR5e

Specification

Payload	5 kg (11 lbs)
Reach	850 mm (33.5 in)
Degrees of freedom	6 rotating joints
Programming	12 inch touchscreen with polyscope graphical user interface

Performance

Power, Consumption, Maximum Average	570 W
Power, Consumption, Typical with moderate settings (approximate)	200 W
Safety	17 configurable safety functions
Certifications	EN ISO 13849-1, PLd Category 3, and EN ISO 10218-1

Force Sensing, Tool Flange	Force, x-y-z	Torque, x-y-z
Range	50.0 N	10.0 Nm
Precision	3.5 N	0.2 Nm
Accuracy	4.0 N	0.3 Nm

Movement

Pose Repeatability per ISO 9283	± 0.03 mm

Axis movement	Working range	Maximum speed
Base	± 360°	± 180°/s
Shoulder	± 360°	± 180°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 180°/s
Wrist 2	± 360°	± 180°/s
Wrist 3	± 360°	± 180°/s

Typical TCP speed	1 m/s (39.4 in/s)

Features

IP classification	IP54
ISO 14644-1 Class Cleanroom	5
Noise	Less than 65 dB(A)
Robot mounting	Any orientation
I/O ports	
Digital in	2
Digital out	2
Analog in	2
Tool I/O Power Supply Voltage	12/24 V
Tool I/O Power Supply	1.5 A (Dual pin) 1 A (Single pin)

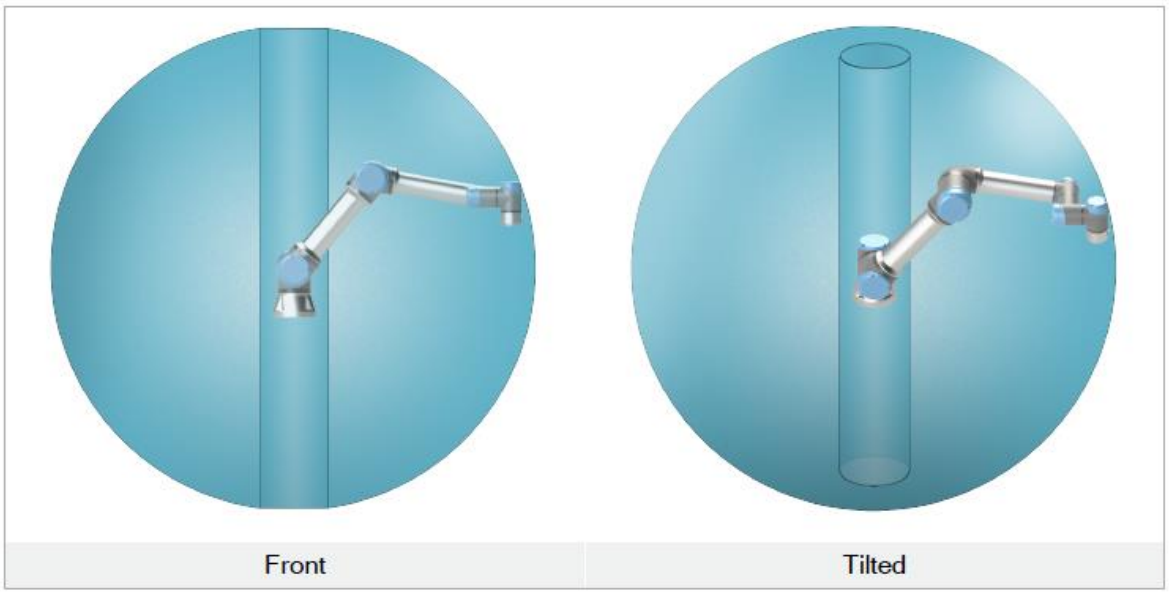
Physical

Footprint	Ø 149 mm
Materials	Aluminium, Plastic, Steel
Tool (end-effector) connector type	M8 | M8 8-pin
Cable length robot arm	6 m (236 in) cable included. 12 m (472 in) and high-flex options available.
Weight including cable	20.6 kg (45.4 lbs)
Operating temperature range	0-50°C
Humidity	90%RH (non-condensing)

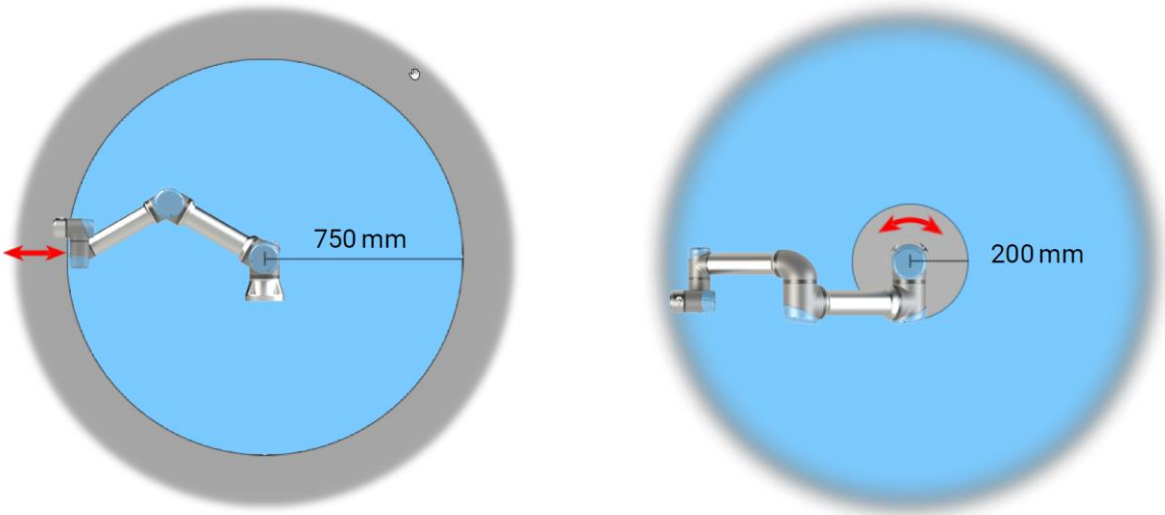
La programación del UR5 se efectúa por medio de la **PolyScope Robot User Interface**. Esta interfaz corresponde a una estructura jerárquica de pantallas que permite tomar diferentes acciones para el robot, desde su instalación y configuración, hasta su programación. El uso de este software es bastante intuitivo y está diseñado para que cualquier persona con un mínimo conocimiento de programación pueda interactuar con el robot, esto considerando el público para el cual está dirigida esta serie de Universal Robots.

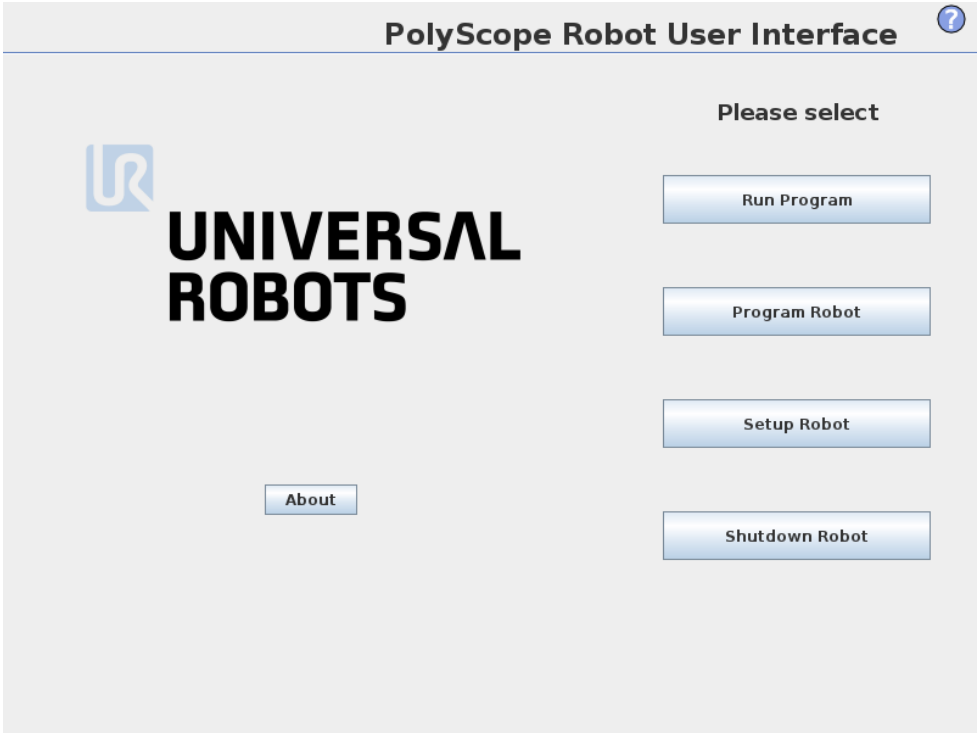
La guía de usuario adjunta en el repositorio contiene un manual completo de instrucciones paso a paso para realizar cada una de estas tareas.

El **espacio de trabajo del robot** ocupa 850 mm desde la junta de la base. Al elegir el lugar de instalación del robot, es importante tener en cuenta el volumen cilíndrico justo encima y debajo de la base del robot. Evite acercar la herramienta a este volumen cilíndrico, ya que esto provocaría que las juntas del robot se movieran rápido cuando la herramienta lo hiciera despacio, lo que causa que el robot trabaje de forma ineficiente y dificulta la realización de la evaluación de riesgos.



ADVERTENCIA
Cuando el robot se extiende, el efecto de articulación de rodilla puede causar fuerzas elevadas en dirección radial (alejándose de la base) a velocidades bajas. De forma similar, el brazo de apalancamiento corto, cuando la herramienta/efector final está cerca de la base y se mueve alrededor de la base, puede causar fuerzas elevadas a velocidades bajas. Los peligros de enganche pueden evitarse eliminando obstáculos en estas zonas, colocando el robot de otra forma o utilizando una combinación de planos de seguridad y límites de eje para eliminar el peligro impidiendo que el robot se mueva hacia esta región de su espacio de trabajo.





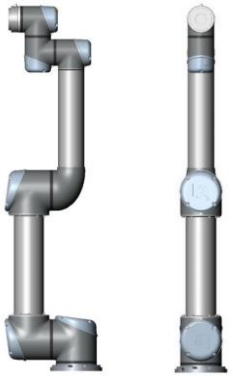
PARTE 2.

MODELOS

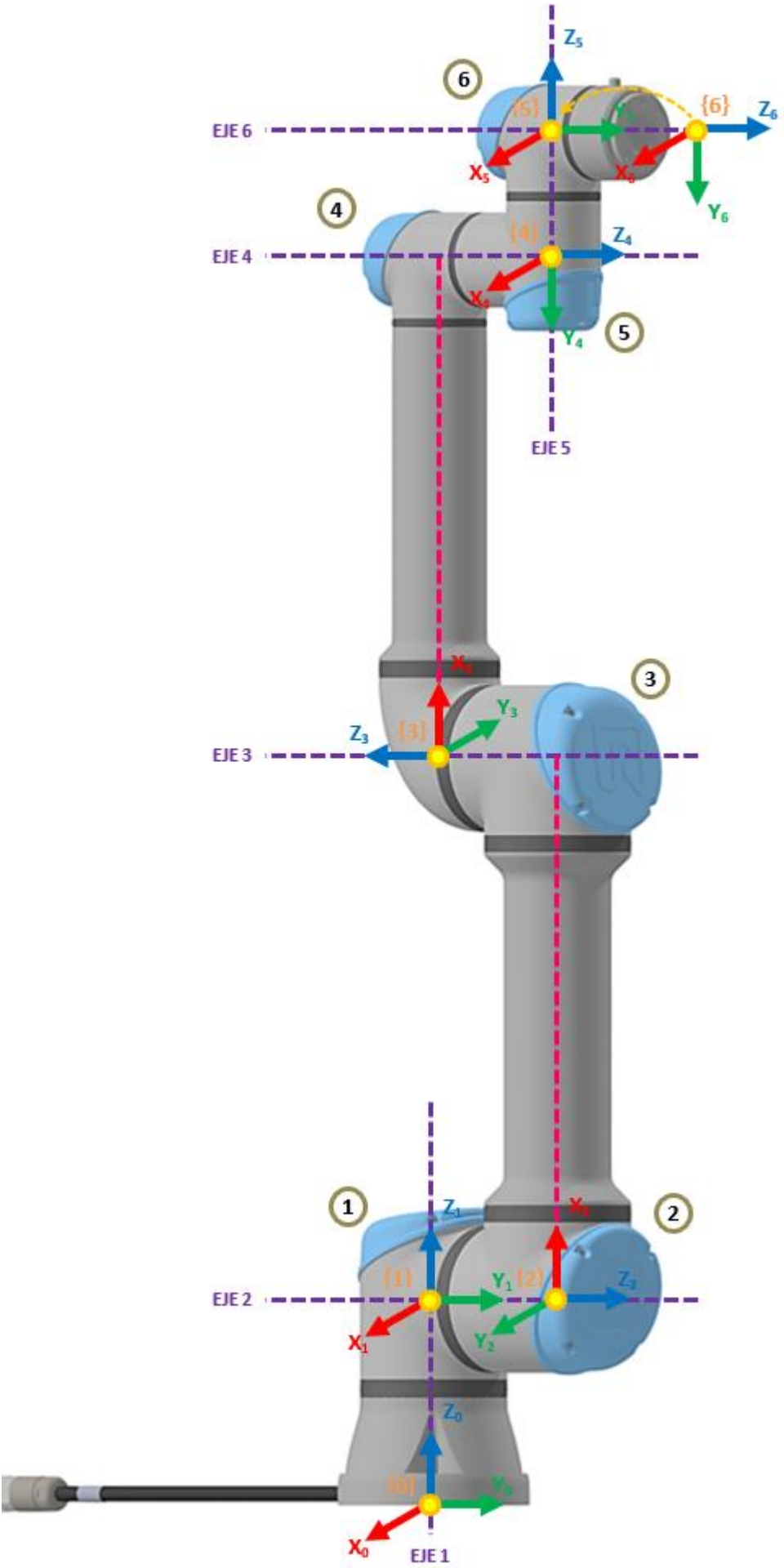
INICIANDO

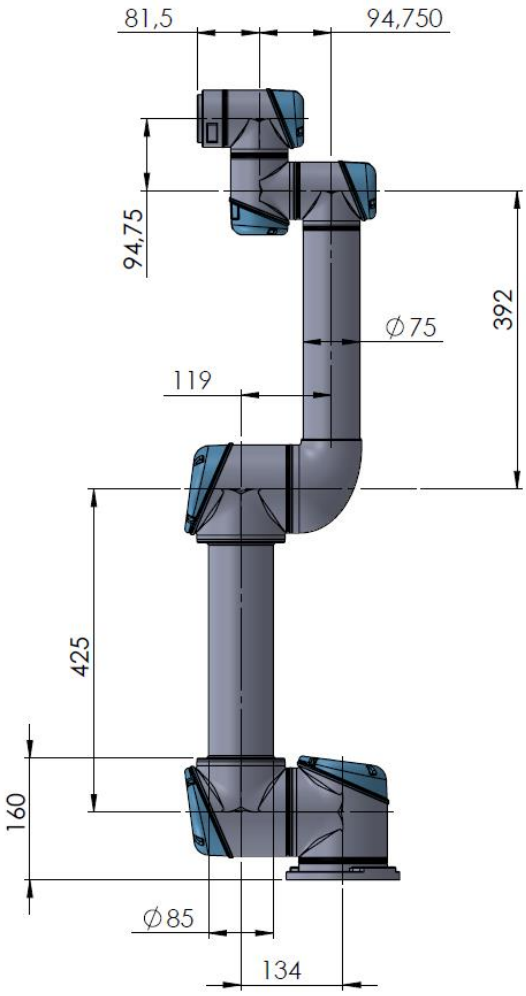
1) Realice el análisis geométrico del robot asignado a través de DH modificado y utilice las funciones de RVC (*Toolbox de Peter Corke*) para construir un modelo en alambres.

Para realizar el análisis geométrico del robot UR5 de 6 GDL, partimos de recopilar la información necesaria del fabricante para el desarrollo del modelo. Desde la página de **Universal Robots** se dispone de la guía de usuario (*Universal Robots e-Series User Manual – UR5e*) en donde se especifica la posición de HOME del robot para el respectivo análisis, además se obtiene el plano esquemático con las dimensiones de cada uno de los eslabones (en mm) y el modelo CAD para esta referencia.



Posición de HOME





$$T_{tool} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 81{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

i	a_{i-1}	α_{i-1}	d_i	θ_i	offset
1	0	0	89.2	q_1	0
2	0	-π/2	134	q_2	-π/2
3	425	π	119	q_3	0
4	392	π	94,75	q_4	π/2
5	0	π/2	94,75	q_5	0
6	0	-π/2	0	q_6	0

Universal Robots UR5e

```
clear
clf
clc

%////////////////////////////////////////////////////////////////////////
% Se determinan las MTH generales expresando las variables de forma simbólica
syms q1 q2 q3 q4 q5 q6
%      Link('revolute/prismatic','a','alpha','d/theta','offset','qlim','modified/standard')
L(1) = Link('revolute', 'a',   0, 'alpha',     0, 'd',  89.2, 'offset',     0, 'modified',
'qlim', [-2*pi/3 2*pi/3]); %'theta'=q1
L(2) = Link('revolute', 'a',   0, 'alpha', -pi/2, 'd',   134, 'offset', -pi/2, 'modified',
'qlim', [-2*pi/3 2*pi/3]); %'theta'=q2
L(3) = Link('revolute', 'a', 425, 'alpha',    pi, 'd',   119, 'offset',     0, 'modified',
'qlim', [-2*pi/3 2*pi/3]); %'theta'=q3
L(4) = Link('revolute', 'a', 392, 'alpha',    pi, 'd', 94.75, 'offset',  pi/2, 'modified',
'qlim', [-2*pi/3 2*pi/3]); %'theta'=q4
L(5) = Link('revolute', 'a',   0, 'alpha',  pi/2, 'd', 94.75, 'offset',     0, 'modified',
'qlim', [-2*pi/3 2*pi/3]); %'theta'=q5
L(6) = Link('revolute', 'a',   0, 'alpha', -pi/2, 'd',     0, 'offset',     0, 'modified',
'qlim', [-2*pi/3 2*pi/3]); %'theta'=q6

plot_opt = {'workspace',[-1 1 -1 1 -0.5
3]*750,'scale',0.4,'nowrist','tilesize',0.5,'tile1color',[0.756, 0.945,
0.933],'jaxes','view',[130 30],'noa'};
axis equal
% Construccion del robot/manipulador
Robot = SerialLink(L,'name','Universal Robot UR5e','plotopt',plot_opt);
Robot.tool = [1  0  0    0;...
              0  1  0    0;...
              0  0  1 81.5;...
              0  0  0    1];
Robot
MTH_0T1 = roundA(L(1).A(q1),1)
MTH_1T2 = roundA(L(2).A(q2),1)
MTH_2T3 = roundA(L(3).A(q3),1)
MTH_3T4 = roundA(L(4).A(q4),1)
MTH_4T5 = roundA(L(5).A(q5),1)
MTH_5T6 = roundA(L(6).A(q6),1)
TCP = Robot.fkine([q1 q2 q3 q4 q5 q6]); % Solución problema geométrico directo
TCP = roundA(TCP,1)

%////////////////////////////////////////////////////////////////////////
q = [0 0 0 0 0 0];
% Robot.plot(q) %Grafica en plano para poder publicar desde MATLAB.
Robot.teach(q)
hold on

MTH0T1 = L(1).A(0);
MTH1T2 = L(2).A(0);
MTH2T3 = L(3).A(0);
MTH3T4 = L(4).A(0);
```

```
MTH4T5 = L(5).A(0);
MTH5T6 = L(6).A(0);
TCP = Robot.fkine(q); % Solución problema geométrico directo
trplot(eye(4),'length',100,'thick',3,'rgb') % Grafica el origen
trplot(MTH0T1,'length',100,'thick',3,'rgb')
trplot(MTH0T1*MTH1T2,'length',100,'thick',3,'rgb')
trplot(MTH0T1*MTH1T2*MTH2T3,'length',100,'thick',3,'rgb')
trplot(MTH0T1*MTH1T2*MTH2T3*MTH3T4,'length',100,'thick',3,'rgb')
trplot(MTH0T1*MTH1T2*MTH2T3*MTH3T4*MTH4T5,'length',100,'thick',3,'rgb')
trplot(MTH0T1*MTH1T2*MTH2T3*MTH3T4*MTH4T5*MTH5T6,'length',100,'thick',3,'rgb')
trplot(TCP,'length',100,'rgb')

%//////////////////////////////////////////////////////////////////////////
```

```
Robot =

Universal Robot UR5e (6 axis, RRRRRR, modDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|       89.2|          0|          0|          0|
|  2|         q2|        134|          0|     -1.571|     -1.571|
|  3|         q3|        119|        425|      3.142|          0|
|  4|         q4|      94.75|        392|      3.142|      1.571|
|  5|         q5|      94.75|          0|      1.571|          0|
|  6|         q6|          0|          0|     -1.571|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1           0           0           0
          0         0  1  0  0           0           1           0           0
       9.81         0  0  1  0           0           0           1        81.5
                    0  0  0  1           0           0           0           1

MTH_0T1 =

[ cos(q1), -sin(q1), 0,     0]
[ sin(q1),  cos(q1), 0,     0]
[       0,        0, 1, 446/5]
[       0,        0, 0,     1]


MTH_1T2 =

[ sin(q2),  cos(q2), 0,   0]
[       0,        0, 1, 134]
[ cos(q2), -sin(q2), 0,   0]
[       0,        0, 0,   1]


MTH_2T3 =

[  cos(q3), -sin(q3),  0,  425]
```

```
[ -sin(q3), -cos(q3),  0,    0]
[        0,        0, -1, -119]
[        0,        0,  0,    1]


MTH_3T4 =

[ -sin(q4), -cos(q4),  0,    392]
[ -cos(q4),  sin(q4),  0,      0]
[        0,        0, -1, -474/5]
[        0,        0,  0,      1]


MTH_4T5 =

[ cos(q5), -sin(q5),  0,      0]
[       0,        0, -1, -474/5]
[ sin(q5),  cos(q5),  0,      0]
[       0,        0,  0,      1]


MTH_5T6 =

[  cos(q6), -sin(q6), 0, 0]
[        0,        0, 1, 0]
[ -sin(q6), -cos(q6), 0, 0]
[        0,        0, 0, 1]


TCP =

[ (cos(q1 + q2 - q3)*cos(q4)*cos(q5)*cos(q6))/2 - cos(q6)*sin(q1)*sin(q5) + (cos(q1 - q2 +
q3)*cos(q4)*cos(q5)*cos(q6))/2 - (sin(q1 + q2 - q3)*cos(q5)*cos(q6)*sin(q4))/2 + (sin(q1 -
q2 + q3)*cos(q5)*cos(q6)*sin(q4))/2 - cos(q1)*cos(q2)*cos(q3)*sin(q4)*sin(q6) +
cos(q1)*cos(q2)*cos(q4)*sin(q3)*sin(q6) - cos(q1)*cos(q3)*cos(q4)*sin(q2)*sin(q6) -
cos(q1)*sin(q2)*sin(q3)*sin(q4)*sin(q6), sin(q1)*sin(q5)*sin(q6) - (cos(q1 + q2 -
q3)*cos(q4)*cos(q5)*sin(q6))/2 - (cos(q1 - q2 + q3)*cos(q4)*cos(q5)*sin(q6))/2 + (sin(q1 +
q2 - q3)*cos(q5)*sin(q4)*sin(q6))/2 - (sin(q1 - q2 + q3)*cos(q5)*sin(q4)*sin(q6))/2 -
cos(q1)*cos(q2)*cos(q3)*cos(q6)*sin(q4) + cos(q1)*cos(q2)*cos(q4)*cos(q6)*sin(q3) -
cos(q1)*cos(q3)*cos(q4)*cos(q6)*sin(q2) - cos(q1)*cos(q6)*sin(q2)*sin(q3)*sin(q4),
cos(q1)*cos(q3)*sin(q2)*sin(q4)*sin(q5) - cos(q1)*cos(q2)*cos(q3)*cos(q4)*sin(q5) -
cos(q1)*cos(q2)*sin(q3)*sin(q4)*sin(q5) - cos(q5)*sin(q1) -
cos(q1)*cos(q4)*sin(q2)*sin(q3)*sin(q5), 425*cos(q1)*sin(q2) - (549*sin(q1))/5 -
(163*cos(q5)*sin(q1))/2 - 392*cos(q1)*cos(q2)*sin(q3) + 392*cos(q1)*cos(q3)*sin(q2) +
(474*cos(q1)*cos(q2)*cos(q3)*sin(q4))/5 - (474*cos(q1)*cos(q2)*cos(q4)*sin(q3))/5 +
(474*cos(q1)*cos(q3)*cos(q4)*sin(q2))/5 + (474*cos(q1)*sin(q2)*sin(q3)*sin(q4))/5 -
(163*cos(q1)*cos(q2)*cos(q3)*cos(q4)*sin(q5))/2 -
(163*cos(q1)*cos(q2)*sin(q3)*sin(q4)*sin(q5))/2 +
(163*cos(q1)*cos(q3)*sin(q2)*sin(q4)*sin(q5))/2 -
(163*cos(q1)*cos(q4)*sin(q2)*sin(q3)*sin(q5))/2]
[ cos(q1)*cos(q6)*sin(q5) + (cos(q1 + q2 - q3)*cos(q5)*cos(q6)*sin(q4))/2 - (cos(q1 - q2 +
q3)*cos(q5)*cos(q6)*sin(q4))/2 + (sin(q1 + q2 - q3)*cos(q4)*cos(q5)*cos(q6))/2 + (sin(q1 -
q2 + q3)*cos(q4)*cos(q5)*cos(q6))/2 - cos(q2)*cos(q3)*sin(q1)*sin(q4)*sin(q6) +
cos(q2)*cos(q4)*sin(q1)*sin(q3)*sin(q6) - cos(q3)*cos(q4)*sin(q1)*sin(q2)*sin(q6) -
```

```
sin(q1)*sin(q2)*sin(q3)*sin(q4)*sin(q6), (cos(q1 - q2 + q3)*cos(q5)*sin(q4)*sin(q6))/2 -
(cos(q1 + q2 - q3)*cos(q5)*sin(q4)*sin(q6))/2 - cos(q1)*sin(q5)*sin(q6) - (sin(q1 + q2 -
q3)*cos(q4)*cos(q5)*sin(q6))/2 - (sin(q1 - q2 + q3)*cos(q4)*cos(q5)*sin(q6))/2 -
cos(q2)*cos(q3)*cos(q6)*sin(q1)*sin(q4) + cos(q2)*cos(q4)*cos(q6)*sin(q1)*sin(q3) -
cos(q3)*cos(q4)*cos(q6)*sin(q1)*sin(q2) - cos(q6)*sin(q1)*sin(q2)*sin(q3)*sin(q4),
cos(q1)*cos(q5) - cos(q2)*cos(q3)*cos(q4)*sin(q1)*sin(q5) -
cos(q2)*sin(q1)*sin(q3)*sin(q4)*sin(q5) + cos(q3)*sin(q1)*sin(q2)*sin(q4)*sin(q5) -
cos(q4)*sin(q1)*sin(q2)*sin(q3)*sin(q5), (549*cos(q1))/5 + (163*cos(q1)*cos(q5))/2 +
425*sin(q1)*sin(q2) - 392*cos(q2)*sin(q1)*sin(q3) + 392*cos(q3)*sin(q1)*sin(q2) +
(474*cos(q2)*cos(q3)*sin(q1)*sin(q4))/5 - (474*cos(q2)*cos(q4)*sin(q1)*sin(q3))/5 +
(474*cos(q3)*cos(q4)*sin(q1)*sin(q2))/5 + (474*sin(q1)*sin(q2)*sin(q3)*sin(q4))/5 -
(163*cos(q2)*cos(q3)*cos(q4)*sin(q1)*sin(q5))/2 -
(163*cos(q2)*sin(q1)*sin(q3)*sin(q4)*sin(q5))/2 +
(163*cos(q3)*sin(q1)*sin(q2)*sin(q4)*sin(q5))/2 -
(163*cos(q4)*sin(q1)*sin(q2)*sin(q3)*sin(q5))/2]
[
- cos(q2 - q3 + q4)*sin(q6) - sin(q2 - q3 + q4)*cos(q5)*cos(q6),
sin(q2 - q3 + q4)*cos(q5)*sin(q6) - cos(q2 - q3 + q4)*cos(q6),
cos(q2 - q3 + q4 - q5)/2 - cos(q2 - q3 + q4 + q5)/2,
425*cos(q2) + 392*cos(q2)*cos(q3) + 392*sin(q2)*sin(q3) + (474*cos(q2)*cos(q3)*cos(q4))/5 +
(474*cos(q2)*sin(q3)*sin(q4))/5 - (474*cos(q3)*sin(q2)*sin(q4))/5 +
(474*cos(q4)*sin(q2)*sin(q3))/5 + (163*cos(q2)*cos(q3)*sin(q4)*sin(q5))/2 -
(163*cos(q2)*cos(q4)*sin(q3)*sin(q5))/2 + (163*cos(q3)*cos(q4)*sin(q2)*sin(q5))/2 +
(163*sin(q2)*sin(q3)*sin(q4)*sin(q5))/2 + 446/5]
[
0,
0,
0,
1]
```

Published with MATLAB® R2020a

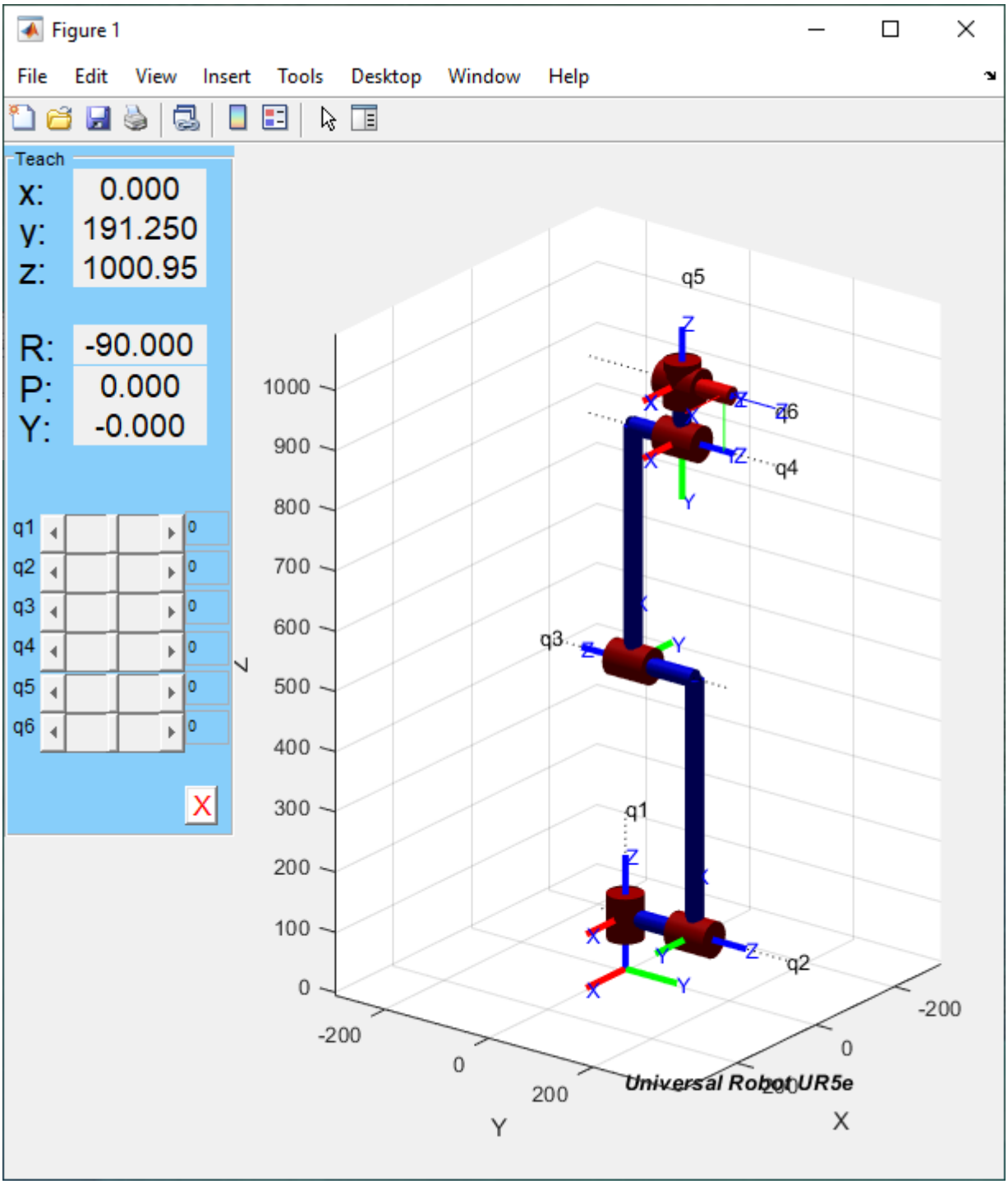


Figure 1
File Edit View Insert Tools Desktop Window Help
Teach
x: 0.000
y: 191.250
z: 1000.95
R: -90.000
P: 0.000
Y: -0.000
q1 0
q2 0
q3 0
q4 0
q5 0
q6 0
X
q5
q6
q4
q3
q1
q2
Universal Robot UR5e
Z
1000
900
800
700
600
500
400
300
200
100
0
-200
0
200
Y
-200
0
200
X

2) Considerando el robot asignado, construya el modelo del robot utilizando RST (*Robotics Systems Toolbox* de MATLAB®).

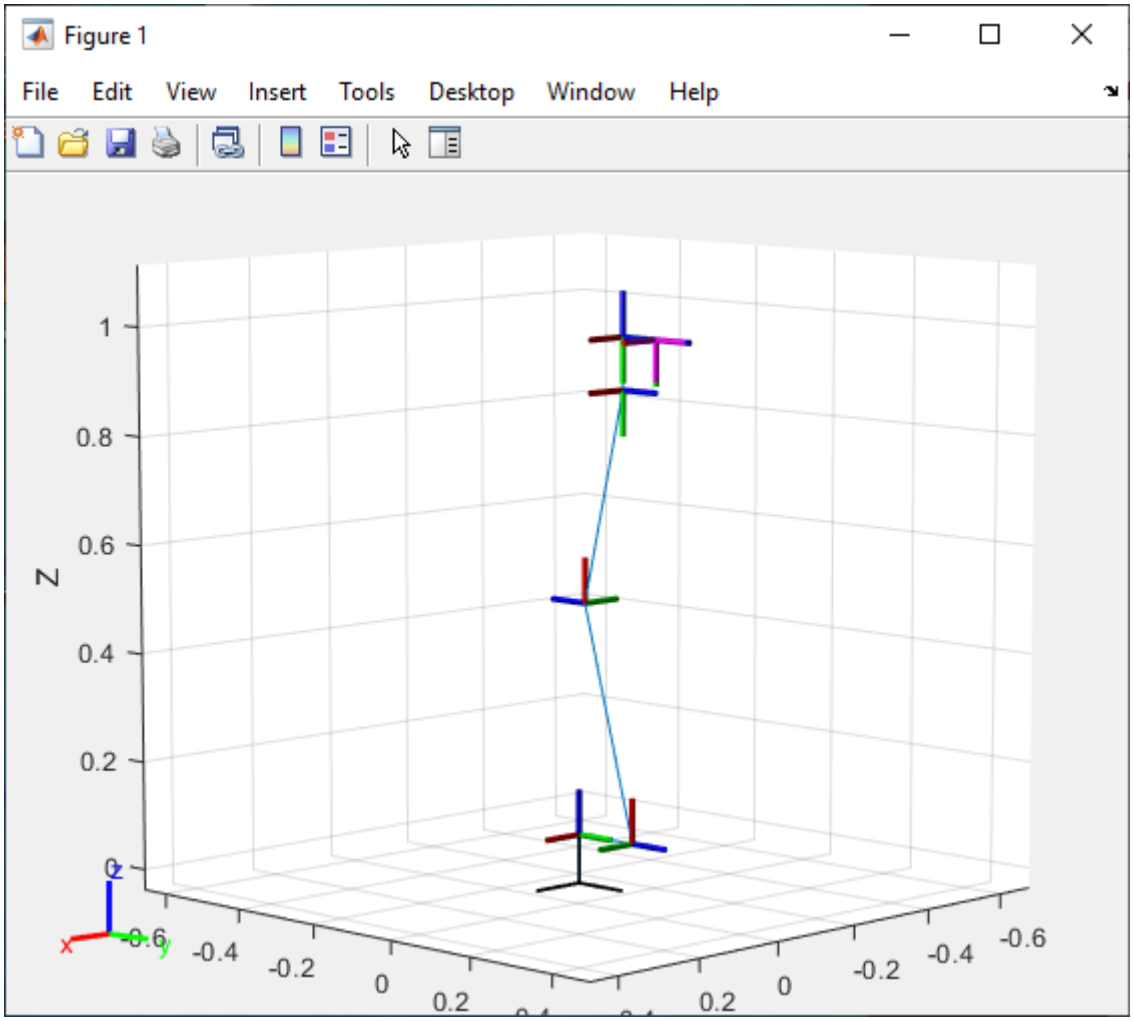
Universal Robots UR5e

```
clear
clf
clc

%//////////////////////////////////////////////////////////////////////////

robot = rigidBodyTree;

body1 = rigidBody('body1');
jnt1 = rigidBodyJoint('jnt1','revolute');
jnt1.HomePosition = 0;
tform = trvec2tform([0,0,89.2/1000])*axang2tform([1 0 0 0]); % User defined
setFixedTransform(jnt1,tform);
body1.Joint = jnt1;
addBody(robot,body1,'base')

body2 = rigidBody('body2');
jnt2 = rigidBodyJoint('jnt2','revolute');
jnt2.HomePosition = -pi/2; % User defined
tform2 = trvec2tform([0,134/1000,0])*axang2tform([1 0 0 -pi/2]); % User defined
setFixedTransform(jnt2,tform2);
body2.Joint = jnt2;
addBody(robot,body2,'body1'); % Add body2 to body1

body3 = rigidBody('body3');
jnt3 = rigidBodyJoint('jnt3','revolute');
jnt3.HomePosition = 0; % User defined
tform3 = trvec2tform([425/1000,0,-119/1000])*axang2tform([1 0 0 pi]); %User defined
setFixedTransform(jnt3,tform3);
body3.Joint = jnt3;
addBody(robot,body3,'body2'); % Add body3 to body2

body4 = rigidBody('body4');
jnt4 = rigidBodyJoint('jnt4','revolute');
jnt4.HomePosition = pi/2; % User defined
tform4 = trvec2tform([392/1000,0,-94.75/1000])*axang2tform([1 0 0 pi]); %User defined
setFixedTransform(jnt4,tform4);
body4.Joint = jnt4;
addBody(robot,body4,'body3'); % Add body4 to body3

body5 = rigidBody('body5');
jnt5 = rigidBodyJoint('jnt5','revolute');
jnt5.HomePosition = 0; % User defined
tform5 = trvec2tform([0,-94.75/1000,0])*axang2tform([1 0 0 pi/2]); %User defined
setFixedTransform(jnt5,tform5);
body5.Joint = jnt5;
```

```
addBody(robot,body5,'body4'); % Add body5 to body4

body6 = rigidBody('body6');
jnt6 = rigidBodyJoint('jnt6','revolute');
jnt6.HomePosition = 0; % User defined
tform6 = trvec2tform([0,0,0])*axang2tform([1 0 0 -pi/2]); %User defined
setFixedTransform(jnt6,tform6);
body6.Joint = jnt6;
addBody(robot,body6,'body5'); % Add body6 to body5

bodyEndEffector = rigidBody('endeffector');
tform7 = trvec2tform([0, 0, 81.5/1000]); % User defined
setFixedTransform(bodyEndEffector.Joint,tform7);
addBody(robot,bodyEndEffector,'body6');

showdetails(robot)
show(robot)

%universalUR5 = loadrobot("universalUR5")
%config = homeConfiguration(universalUR5)
%config(2).JointPosition = -pi/2;
%config(4).JointPosition = -pi/2;
%show(universalUR5,config,'Visuals','on')
```

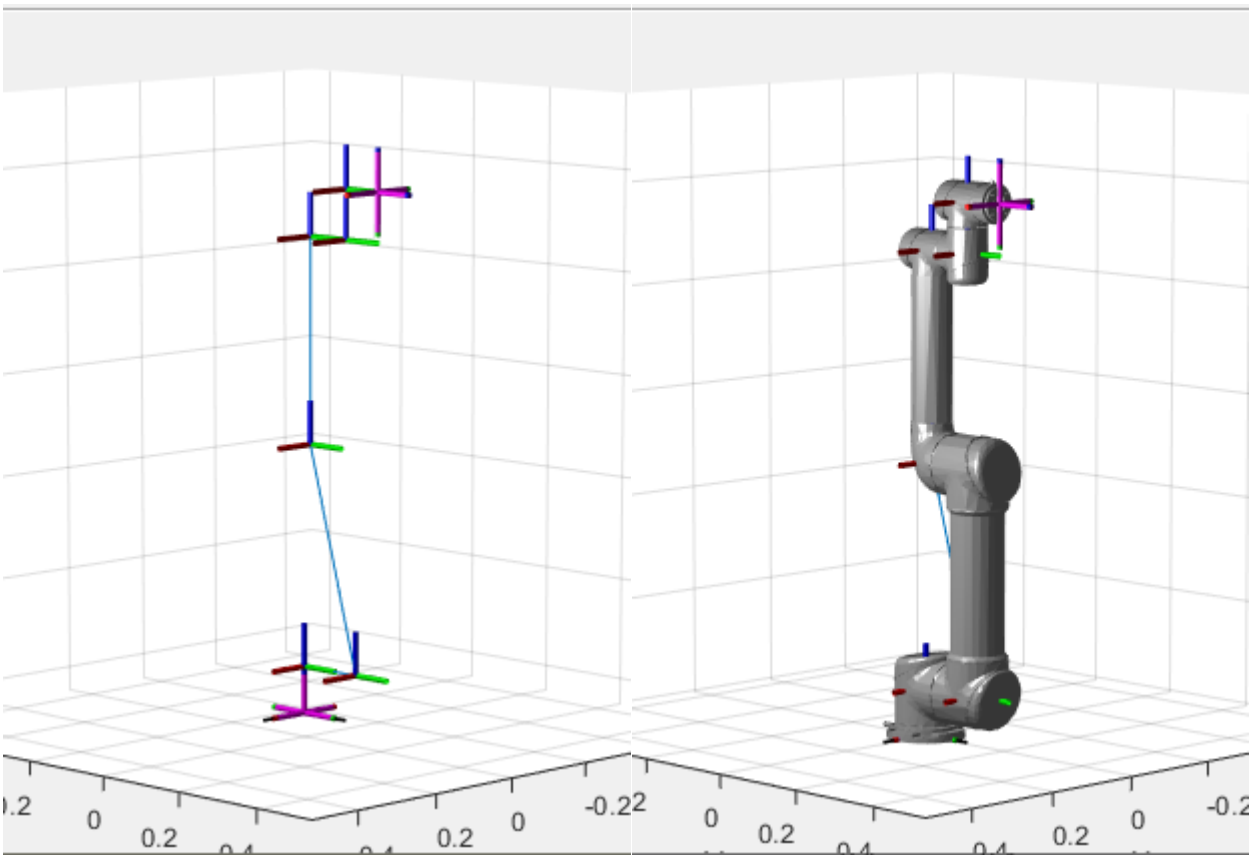
```
--------------------
Robot: (7 bodies)

 Idx          Body Name             Joint Name             Joint Type          Parent
Name(Idx)   Children Name(s)
 ---          ---------             ----------             ----------          ------------
----   ----------------
   1             body1                   jnt1               revolute
base(0)   body2(2)
   2             body2                   jnt2               revolute
body1(1)   body3(3)
   3             body3                   jnt3               revolute
body2(2)   body4(4)
   4             body4                   jnt4               revolute
body3(3)   body5(5)
   5             body5                   jnt5               revolute
body4(4)   body6(6)
   6             body6                   jnt6               revolute
body5(5)   endeffector(7)
   7         endeffector        endeffector_jnt                  fixed
body6(6)
--------------------

ans =

  Axes (Primary) with properties:

            XLim: [-1.5000 1.5000]
            YLim: [-1.5000 1.5000]
          XScale: 'linear'
```

```
          YScale: 'linear'
   GridLineStyle: '-'
        Position: [0.1300 0.1100 0.7750 0.8150]
           Units: 'normalized'

  Use GET to show all properties
```



Published with MATLAB® R2020a

La *Robotics Systems Toolbox* de MATLAB® incluye una amplia biblioteca de robots predeterminados para su uso, entre los cuáles podemos encontrar el universalUR5 ya programado. A modo de comparación, se presenta a continuación el modelo con la posición y orientación de los marcos de coordenadas predeterminados en la RST, en el cuál puede observarse la diferencia con respecto a nuestro modelo desarrollado paso a paso.



Es posible observar que los sistemas de coordenadas difieren en ambos modelos, en tanto que para el modelo predeterminado la ubicación de los orígenes para los marcos {4} y {5} es diferente a nuestro modelo (no coincide con las intersecciones de los ejes) y el sentido en que se encuentran orientados los ejes X, Y, y Z es el mismo en todos los sistemas del {0} al {6}, por tanto se puede concluir que la convención con la cual se analizó el mecanismo no corresponde al DH modificado que hemos venido desarrollando a lo largo de la asignatura.

3) Compare los dos métodos.

Debido al trabajo realizado a lo largo de la asignatura, estamos bastante familiarizados con el uso de la RVC (*Toolbox de Peter Corke*). Para este método, basta con ingresar los parámetros obtenidos para la convención de Denavit-Hartenberg modificada para la creación de cada uno de los `Link` que conforman el robot, a diferencia del método de RST (*Robotics Systems Toolbox* de MATLAB®) en el cual se crean cada uno de los cuerpos rígidos que corresponden a los eslabones y las articulaciones con que interactúan, asociando cada par articulación/eslabón de forma independiente, y vinculando la transformación de cada sistema.

Aunque a primera vista RVC es más fácil a la hora de construir la geometría del robot, RST es mucho más flexible, dado que permite ingresar las transformaciones

directamente por medio de la MTH, o construirla por partes con los vectores de traslación o las rotaciones alrededor de un vector, por tanto no se limita a la convención de Denavit-Hartenberg estándar y modificada como sí es el caso de RVC. Para el caso particular de este ejemplo, no fue un trabajo adicional ingresar las transformaciones para RST a partir de los parámetros ya obtenidos, logrando que ambos modelos coincidan totalmente en las posiciones y orientaciones de los sistemas de coordenadas.

4) Con la hoja técnica del robot, el fabricante provee puntos de calibración. Con la ayuda de la cinemática directa verifique dichos puntos.

Para el caso del robot asignado, la guía de usuario no provee mayor información respecto a puntos de calibración dado que la misma se realiza por medio de un software diseñado exclusivamente para ese propósito, motivo por el cuál procedemos a realizar la validación del punto del TCP en la configuración de HOME con la cinemática directa por ambos métodos (RVC y RST), confirmando que coinciden exactamente como es lo esperado.

UR5e_RVC

```
%////////////////////////////////////////////////////////////////////////
q = [0 0 0 0 0 0]
TCP = Robot.fkine(q) % Solución problema geométrico directo para el *q* dado
[ROT,POS]=tr2rt(TCP);
POS = POS' %Posición en coordenadas cartesianas
ROT = tr2rpy(ROT,'deg') %Orientación en ángulos fijos.
%////////////////////////////////////////////////////////////////////////
```

```
TCP =

   1.0e+03 *

    0.0010         0    0.0000    0.0000
   -0.0000    0.0000    0.0010    0.1912
   -0.0000   -0.0010    0.0000    1.0010
         0         0         0    0.0010


POS =

   1.0e+03 *

    0.0000    0.1912    1.0010


ROT =

  -90.0000    0.0000         0
```

UR5e_RST

```
TCP=getTransform(robot,homeConfiguration(robot),'endeffector')
POS=tform2trvec(TCP)*1000
ROT=rad2deg(tform2eul(TCP,'XYZ'))
```

```
TCP =

    1.0000         0    0.0000    0.0000
   -0.0000    0.0000    1.0000    0.1912
   -0.0000   -1.0000    0.0000    1.0010
         0         0         0    1.0000


POS =

   1.0e+03 *

    0.0000    0.1912    1.0010


ROT =

  -90.0000    0.0000         0
```

MODELO GEOMÉTRICO DIRECTO

1) Halle el modelo geométrico directo de su robot asignado usando MTH.

El modelo geométrico se encontró en la sección anterior por medio de la RVC (*Toolbox de Peter Corke*), obteniendo las matrices de transformación para cada marco de referencia asignado. Para encontrar la MTH del efector final respecto a la base o MTH_0T6 (equivalente a la `TCP` obtenida por RVC) se debe seguir la ecuación de lazo, multiplicando cada una de las matrices:

$$\text{MTH\_0T1} = \begin{bmatrix} \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 \\ \sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 89.2 \end{bmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

$$\text{MTH\_1T2} = \begin{bmatrix} \begin{bmatrix} \sin(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 1 \\ \cos(q_2) & -\sin(q_2) & 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 134 \\ 0 \end{bmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

$$\text{MTH\_2T3} = \begin{bmatrix} \begin{bmatrix} \cos(q_3) & -\sin(q_3) & 0 \\ -\sin(q_3) & -\cos(q_3) & 0 \\ 0 & 0 & -1 \end{bmatrix} & \begin{bmatrix} 425 \\ 0 \\ -119 \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

$$\text{MTH\_3T4} = \begin{bmatrix} \begin{bmatrix} -\sin(q_4) & -\cos(q_4) & 0 \\ -\cos(q_4) & \sin(q_4) & 0 \\ 0 & 0 & -1 \end{bmatrix} & \begin{bmatrix} 392 \\ 0 \\ -94.8 \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

$$\text{MTH\_4T5} = \begin{bmatrix} \begin{bmatrix} \cos(q_5) & -\sin(q_5) & 0 \\ 0 & 0 & -1 \\ \sin(q_5) & \cos(q_5) & 0 \end{bmatrix} & \begin{bmatrix} 0 \\ -94.8 \\ 0 \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

$$\text{MTH\_5T6} = \begin{bmatrix} \begin{bmatrix} \cos(q_6) & -\sin(q_6) & 0 \\ 0 & 0 & 1 \\ -\sin(q_6) & -\cos(q_6) & 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

2) Haciendo uso del modelo cinemático directo obtenga los valores de posición y de orientación en coordenadas generalizadas del efector final de su robot asignado para los siguientes valores articulares:

#	*q*						X	Y	Z	ROLL	PITCH	YAW
1	0.5	0.2	0.4	0.5	0.0	1.5	-61.3638	184.4050	980.4325	-90.0000	-28.6479	103.1324
2	-π/2	0.3	0.0	π/2	0.4	1.2	184.8165	-341.3372	872.0295	17.1887	67.0817	68.7549
3	0.0	1.0	-0.5	2.0	1.0	0.5	779.6286	153.7846	233.7715	-118.6483	51.9991	-116.6190
4	-1.0	-0.3	-π/5	0.4	0.2	1.0	187.5457	58.8766	947.7684	-78.5740	48.1238	89.8949

3) Haga uso de las funciones de cinemática directa de ambos toolboxes y compruebe los resultados anteriores.

A continuación, empleamos ambos métodos (RVC y RST) para evaluar cada uno de los vectores de configuración ***q*** dados por medio de la cinemática directa y así poder comparar la posición en coordenadas cartesianas y la orientación en ángulos RPY.

UR5e_RVC

```
%////////////////////////////////////////////////////////////////////////
q = [0.5 0.2 0.4 0.5 0 1.5];
TCP = Robot.fkine(q); % Solución problema geométrico directo para el *q* dado
[ROT1,POS1]=tr2rt(TCP);
POS1 = POS1' %Posición en coordenadas cartesianas
ROT1 = tr2rpy(ROT1,'deg') %Orientación en ángulos fijos.
%////////////////////////////////////////////////////////////////////////
```

```
q = [-pi/2 0.3 0 pi/2 0.4 1.2];
TCP = Robot.fkine(q); % Solución problema geométrico directo para el *q* dado
[ROT2,POS2]=tr2rt(TCP);
POS2 = POS2' %Posición en coordenadas cartesianas
ROT2 = tr2rpy(ROT2,'deg') %Orientación en ángulos fijos.
%//////////////////////////////////////////////////////////////////////////
q = [0 1 -0.5 2 1 0.5];
TCP = Robot.fkine(q); % Solución problema geométrico directo para el *q* dado
[ROT3,POS3]=tr2rt(TCP);
POS3 = POS3' %Posición en coordenadas cartesianas
ROT3 = tr2rpy(ROT3,'deg') %Orientación en ángulos fijos.
%//////////////////////////////////////////////////////////////////////////
q = [-1 -0.3 -pi/5 0.4 0.2 1];
TCP = Robot.fkine(q); % Solución problema geométrico directo para el *q* dado
[ROT4,POS4]=tr2rt(TCP);
POS4 = POS4' %Posición en coordenadas cartesianas
ROT4 = tr2rpy(ROT4,'deg') %Orientación en ángulos fijos.
%//////////////////////////////////////////////////////////////////////////
```

```
POS1 =

  -61.3638  184.4050  980.4325


ROT1 =

  -90.0000  -28.6479  103.1324


POS2 =

  184.8165 -341.3372  872.0295


ROT2 =

   17.1887   67.0817   68.7549


POS3 =

  779.6286  153.7846  233.7715


ROT3 =

 -118.6483   51.9991 -116.6190


POS4 =

  187.5457   58.8766  947.7684
```

```
ROT4 =

  -78.5740   48.1238   89.8949
```

UR5e_RST

```
%CINEMÁTICA DIRECTA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q=homeConfiguration(robot);
config1 = [0.5 0.2 0.4 0.5 0 1.5];
for i=1:6
    q(i).JointPosition = q(i).JointPosition + config1(i);
end
TCP=getTransform(robot,q,'endeffector');
POS1=tform2trvec(TCP)*1000
ROT1=rad2deg(tform2eul(TCP,'XYZ'))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q=homeConfiguration(robot);
config2 = [-pi/2 0.3 0 pi/2 0.4 1.2];
for i=1:6
    q(i).JointPosition = q(i).JointPosition + config2(i);
end
TCP=getTransform(robot,q,'endeffector');
POS2=tform2trvec(TCP)*1000
ROT2=rad2deg(tform2eul(TCP,'XYZ'))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q=homeConfiguration(robot);
config3 = [0 1 -0.5 2 1 0.5];
for i=1:6
    q(i).JointPosition = q(i).JointPosition + config3(i);
end
TCP=getTransform(robot,q,'endeffector');
POS3=tform2trvec(TCP)*1000
ROT3=rad2deg(tform2eul(TCP,'XYZ'))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q=homeConfiguration(robot);
config4 = [-1 -0.3 -pi/5 0.4 0.2 1];
for i=1:6
    q(i).JointPosition = q(i).JointPosition + config4(i);
end
TCP=getTransform(robot,q,'endeffector');
POS4=tform2trvec(TCP)*1000
ROT4=rad2deg(tform2eul(TCP,'XYZ'))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
POS1 =

  -61.3638  184.4050  980.4325


ROT1 =
```

```
  -90.0000  -28.6479  103.1324

POS2 =

  184.8165 -341.3372  872.0295

ROT2 =

   17.1887   67.0817   68.7549

POS3 =

  779.6286  153.7846  233.7715

ROT3 =

 -118.6483   51.9991 -116.6190

POS4 =

  187.5457   58.8766  947.7684

ROT4 =

  -78.5740   48.1238   89.8949
```

4) Compare los métodos.

Los dos métodos para el cálculo numérico de la cinemática directa están determinados por el uso de una única función para cada caso: `fkine()` (en RVC, *Toolbox de Peter Corke*) y `getTransform()` (en RST, *Robotics Systems Toolbox* de MATLAB®). En ambas funciones se requieren como parámetros básicos un robot (tipo `SerialLink` en RVC y tipo `rigidBodyTree` en RST) y un vector ***q*** de configuraciones de articulación. Sin embargo, para obtener una posición en coordenadas cartesianas y una orientación en ángulos fijos, el método de RVC resulta ser más práctico, esto dado que la cantidad de funciones y métodos disponibles para esta *Toolbox* es más amplia que en RST, para la cuál se requieren algunas líneas de código adicionales para realizar las transformaciones de coordenadas necesarias.

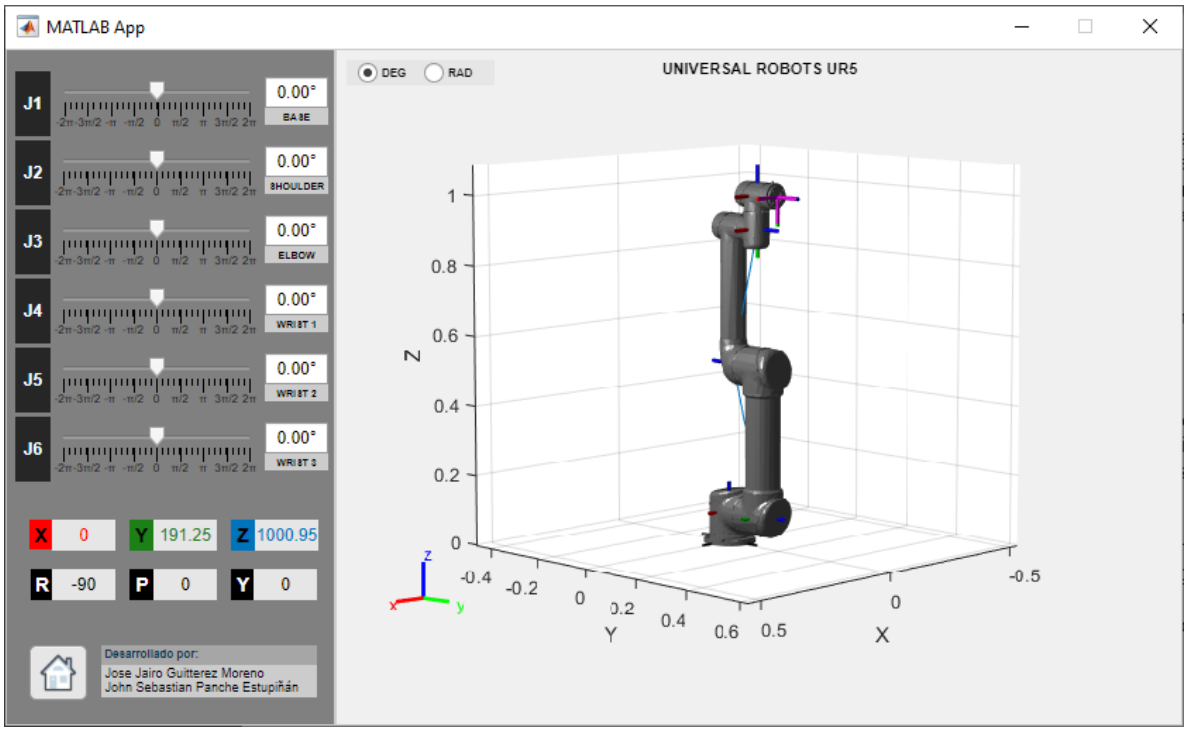
En cuanto a los resultados aplicando la cinemática directa, todas las coordenadas de posición y orientación obtenidas por ambos métodos coinciden, esto se debe a que ambos mecanismos se construyeron basados en el diagrama del análisis geométrico y la ubicación de los sistemas coordenados por Denavit-Hartenberg modificado obtenido

en un numeral anterior, por tanto las articulaciones rotacionales de ambos robots girarán en el mismo sentido y las coordenadas del efector final serán por tanto iguales.

5) Elija uno de los métodos anteriormente usados y desarrolle una GUI que permita mover cada articulación mediante controles tipo *Slider*, visualizar el robot y la posición del efector final.

Dado que a lo largo del semestre hemos trabajado con las funciones de RVC (*Toolbox de Peter Corke*), y el cuál ya dispone de una interfaz gráfica de usuario predeterminada (`Robot.teach(q)`) optamos por realizar la GUI utilizando la RST (*Robotics Systems Toolbox* de MATLAB®), para lo cual se utilizó el código del robot construido paso a paso en la sección anterior y se agregó además el aspecto visual asociando los archivos de malla (*.stl*) para cada uno de los respectivos *body*, obteniendo así gráficos más realistas del modelo del robot asignado.

Para este caso, se empleó la herramienta *AppDesigner* de MATLAB®, un entorno de desarrollo interactivo para el diseño y programación de una aplicación que integra el editor de MATLAB® y un conjunto de diversos componentes interactivos de la IU (**I**nterfaz de **U**suario) como *buttons*, *sliders*, *drop-down lists*, *trees*, entre muchos otros.



La GUI diseñada por el equipo consta de dos paneles principales: un panel izquierdo con los controles interactivos e ingreso de datos por parte del usuario (además de la

visualización de las coordenadas generalizadas) y un panel derecho donde se presenta el gráfico del robot correspondiente a la configuración seleccionada.

En el panel izquierdo se pueden observar tres secciones:

- 1ra sección, en donde se encuentran los respectivos *sliders* asociados a cada una de las articulaciones (J1, J2, J3, J4, J5, J6) y los campos de entrada de datos (o *fields*) de modo que el usuario pueda ingresar valores numéricos por medio del teclado bien sea en grados o en radiantes y dentro de los límites de cada articulación para el cálculo de las coordenadas del efector final.

- 2da sección, correspondiente a la visualización y lectura por medio de *labels* de las coordenadas generalizadas, presentando al usuario la posición del efector final en coordenadas cartesianas **X**, **Y**, **Z** y la orientación en ángulos fijos **R**, **P**, **Y** expresados en grados.

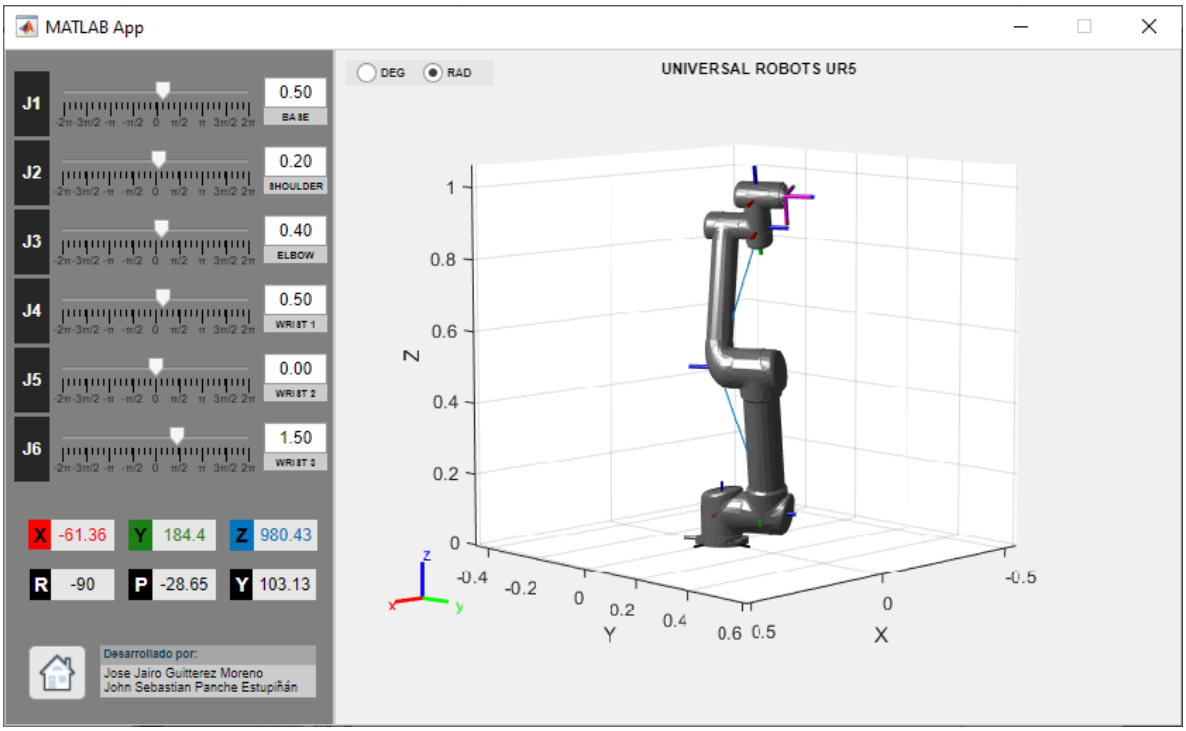
- Y 3ra sección, donde se encuentra un *button* que permite enviar la configuración del robot directamente a HOME, y un pequeño campo de texto con los nombres de los integrantes del equipo de desarrollo de la GUI.

En el panel derecho, se presenta una captura de la gráfica obtenida para la configuración programada en el panel izquierdo, además de contar con un grupo de dos *radio button* que permite cambiar la base de los valores de las articulaciones (tanto de entrada como de salida) entre grados (DEG) y radianes (RAD). El funcionamiento detallado de la GUI y el código programado serán explicados con mayor detalle por medio del vídeo presentado junto con este informe de laboratorio.

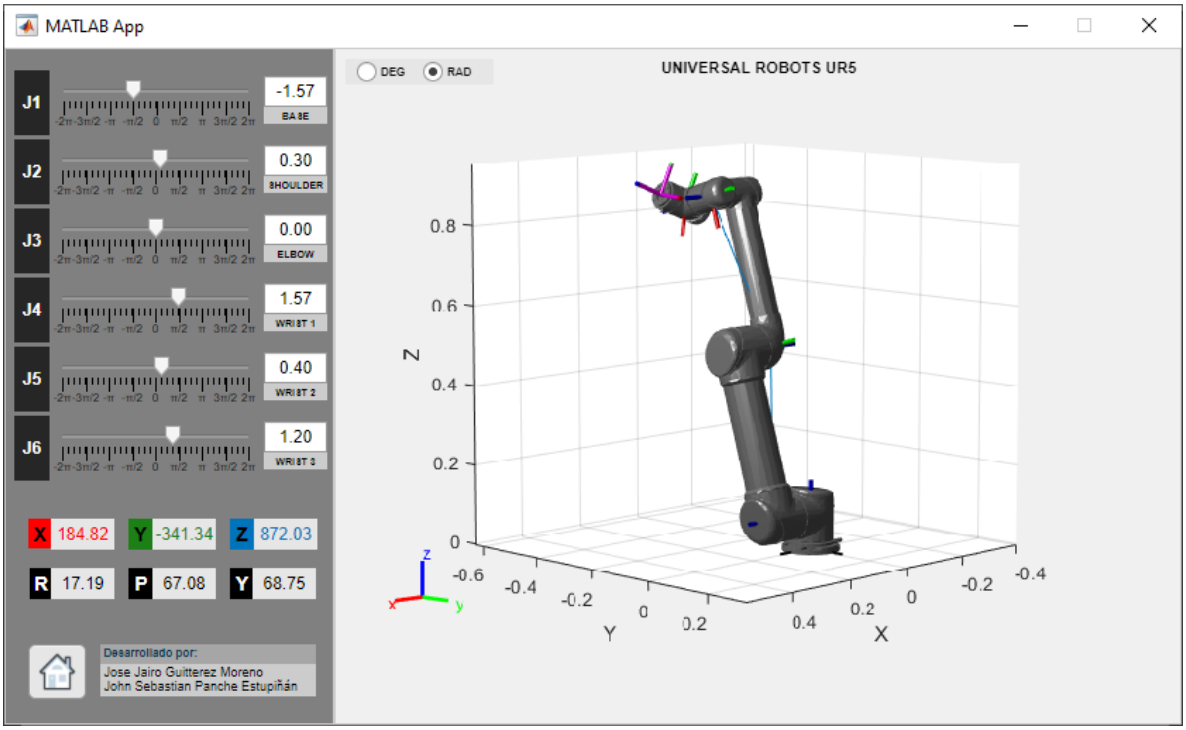
6) En el informe incluya capturas de pantalla verificando las posiciones EF y las articulaciones.

Dado que la GUI se diseñó y programó a partir del robot construido con RST (cuyos resultados ya comparamos con RVC en un numeral anterior), se validan nuevamente cada uno de los resultados ingresando los valores en radianes dentro de cada uno de los campos de las articulaciones (J1, J2, J3, J4, J5, J6) y comparando las coordenadas generalizadas consignadas en la tabla con los valores de los campos **X**, **Y**, **Z** y **R**, **P**, **Y** obtenidos a partir de nuestra aplicación.

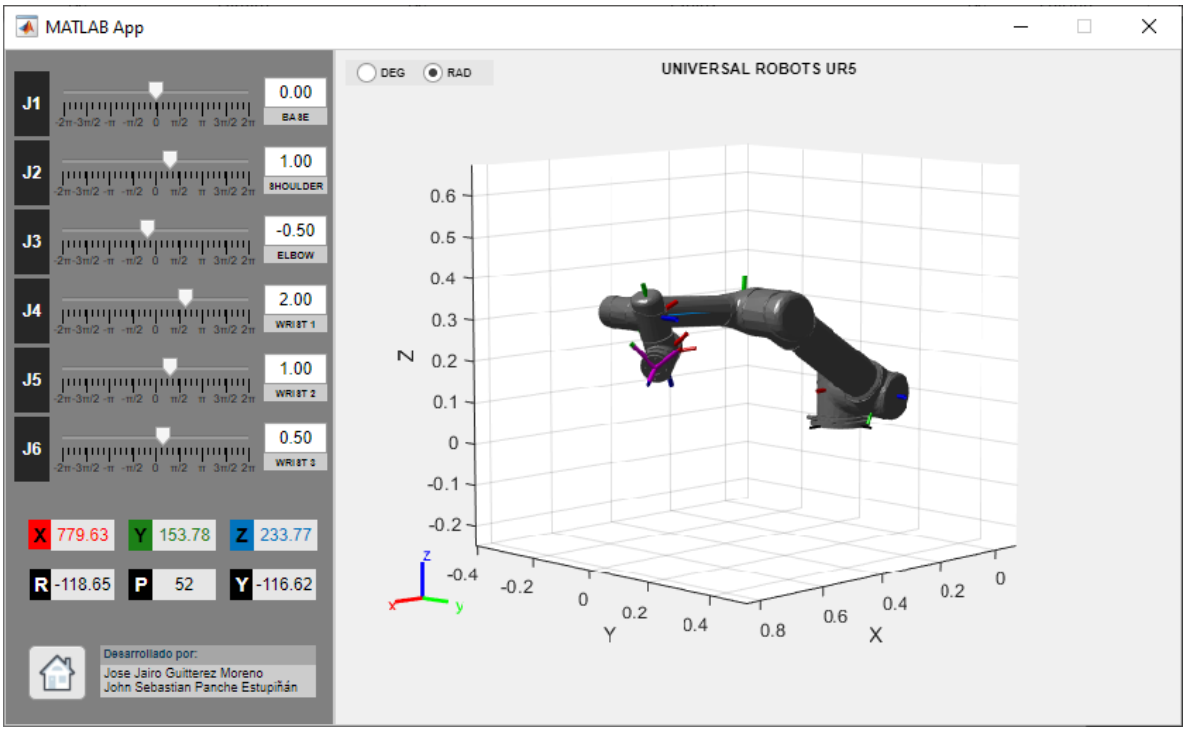
#	*Q*						X	Y	Z	ROLL	PITCH	YAW
1	0.5	0.2	0.4	0.5	0.0	1.5	-61.3638	184.4050	980.4325	-90.0000	-28.6479	103.1324



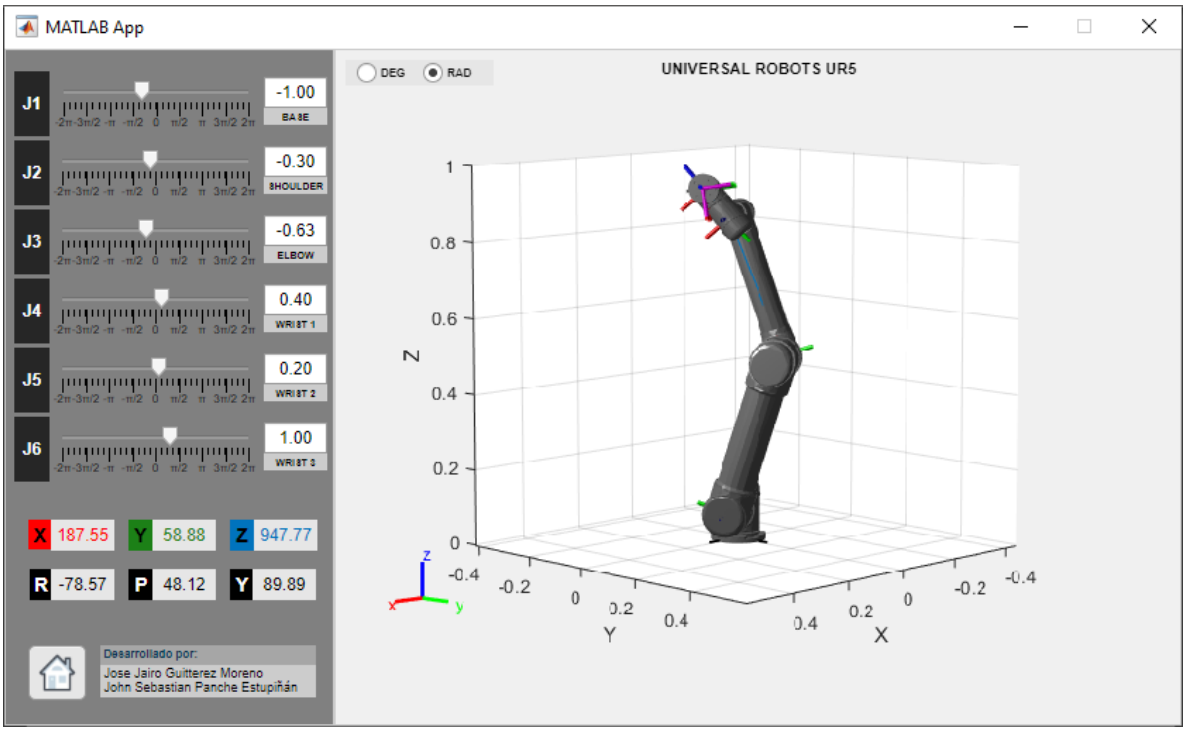
#	Q						X	Y	Z	ROLL	PITCH	YAW
2	-π/2	0.3	0.0	π/2	0.4	1.2	184.8165	-341.3372	872.0295	17.1887	67.0817	68.7549



#	Q						X	Y	Z	ROLL	PITCH	YAW
3	0.0	1.0	-0.5	2.0	1.0	0.5	779.6286	153.7846	233.7715	-118.6483	51.9991	-116.6190



#	Q						X	Y	Z	ROLL	PITCH	YAW
4	-1.0	-0.3	-π/5	0.4	0.2	1.0	187.5457	58.8766	947.7684	-78.5740	48.1238	89.8949



MODELO GEOMÉTRICO INVERSO

1) Determine el modelo geométrico inverso del robot asignado haciendo uso de la metodología explicada en clase.

2) En las soluciones de la inversa incluya consideraciones respecto a multiplicidad de soluciones.

3) Haga uso de las funciones del RVC para hallar la cinemática inversa de su robot asignado y compruebe los resultados del punto anterior. Ya que existen varias funciones en el Toolbox explique: ¿Cuál es la diferencia entre estas funciones? ¿Cuál debe usar para su robot y por qué? (Revise la documentación del *Toolbox*).

Partiendo de la guía de la RVC encontramos los métodos asociados con la función SerialLink relacionados con la cinemática inversa:

ikine6s	inverse kinematics for 6-axis spherical wrist revolute robot
ikine	inverse kinematics using iterative numerical method
ikunc	inverse kinematics using optimisation
ikcon	inverse kinematics using optimisation with joint limits
ikine_sym	analytic inverse kinematics obtained symbolically

Cada una de las funciones emplea diferentes métodos de resolución para encontrar la cinemática inversa para una determinada pose del efector final (*x, y, z, roll, pitch, yaw*), por tanto, dependiendo de la geometría del robot, algunos de estos métodos permiten obtener mejores resultados que otros.

SerialLink.ikine. Numerical inverse kinematics.
SerialLink.ikine6s. Analytical inverse kinematics.
SerialLink.ikinem. Numerical inverse kinematics by minimization.
SerialLink.ikunc. Numerical inverse manipulator without joint limits.
SerialLink.ikcon. Numerical inverse kinematics with joint limits.
SerialLink.ikine_sym. Symbolic inverse kinematics.

De los métodos disponibles, sólo dos de ellos permiten encontrar una solución analítica al problema de cinemática inversa, sin embargo el *ikine6s* se limita a robots manipuladores con muñeca esférica (intersección de tres de los ejes en un mismo punto) el cuál no es el caso del robot asignado UR5 ni tampoco es aplicable para convenciones diferentes a los parámetros de Denavit-Hartenberg estándar; y la función *ikine_sym* permite obtener una solución simbólica con ecuaciones comparables a la respuesta obtenida por la metodología convencional aplicada en clase, sin embargo, la misma aún presenta muchos errores y se encuentra en desuso dado que requiere funciones obsoletas de la *Symbolic Math Toolbox* de MATLAB®, razón por la cuál no fue posible utilizarla para obtener la solución general de la cinemática inversa por RVC.

Las demás funciones presentadas, permiten obtener una solución numérica de la cinemática inversa partiendo de una postura (*x, y, z, roll, pitch, yaw*) conocida empleando diferentes métodos de iteración y aproximación, y serán exploradas un poco más adelante.

4) Haga uso del RST para hallar la cinemática inversa de su robot asignado y compruebe los resultados anteriores.

Partiendo de la guía de la RST, la función *analyticalInverseKinematics* permite obtener una solución analítica al problema de cinemática inversa, sin embargo y de modo similar a lo ocurrido con RVC, la función presentaba muchos errores y era muy poco precisa al grado en que para la última versión de la *Toolbox* esta función ya no está disponible. De modo análogo con RVC, se tiene una función que permite obtener la solución numérica de la cinemática inversa partiendo de una postura (*x, y, z, roll, pitch, yaw*) conocida empleando diferentes métodos de iteración.

5) Compare los métodos.

Dado que no fue posible obtener una solución analítica por ninguno de los dos métodos, se realiza una comparación entre las soluciones numéricas obtenidas en el punto siguiente.

6) Calcule la configuración del robot para las siguientes posturas de la herramienta. Proponga 4 posturas $(x, y, z, roll, pitch, yaw)^T$ que estén dentro del espacio de trabajo y determine la configuración del manipulador y complete la tabla:

Con el fin de demostrar que el cálculo del vector de configuración ***q*** se realizó por medio de cinemática inversa, se seleccionan 4 posturas (*x, y, z, roll, pitch, yaw*) con valores cerrados, de modo que no pueden obtenerse con facilidad variando las articulaciones con los *sliders* de forma manual en la GUI.

#	X	Y	Z	ROLL	PITCH	YAW	q_1	q_2	q_3	q_4	q_5	q_6
1	500.00	500.00	500.00	-90°	0°	0°	0.5278	0.6336	-1.0047	-1.6383	-0.5278	-0.0000
2	100.00	-200.00	300.00	90°	90°	45°	-2.0566	2.1309	2.6423	0.5114	0.4858	-2.3562
3	-300.00	400.00	-500.00	-30°	-45°	-90°	1.8991	3.1473	1.1838	0.3504	0.9820	1.2284
4	123.00	456.00	789.00	-90°	-45°	-30°	0.8915	1.0338	0.8847	-0.1491	-0.1061	-0.5236

UR5e_RST

```
%CINEMÁTICA INVERSA
ik = inverseKinematics('RigidBodyTree',robot);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C1 = [500 500 500 -90 0 0]
TRVEC=[C1(1,1:3)/1000 1]';
ROTM=eul2tform(deg2rad(C1(1,4:6)),'XYZ');
TCP = [ROTM(:,1:3) TRVEC];
[configSol,solInfo] = ik('endeffector',TCP,[1 1 1 1 1 1],homeConfiguration(robot));
qhome=homeConfiguration(robot);
q=zeros(1,6);
for i=1:6
    q(i)=configSol(i).JointPosition-qhome(i).JointPosition;
end
q
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C2 = [100 -200 300 90 90 45]
TRVEC=[C2(1,1:3)/1000 1]';
ROTM=eul2tform(deg2rad(C2(1,4:6)),'XYZ');
TCP = [ROTM(:,1:3) TRVEC];
[configSol,solInfo] = ik('endeffector',TCP,[1 1 1 1 1 1],homeConfiguration(robot));
qhome=homeConfiguration(robot);
q=zeros(1,6);
for i=1:6
    q(i)=configSol(i).JointPosition-qhome(i).JointPosition;
end
q
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C3 = [-300 400 -500 -30 -45 -90]
TRVEC=[C3(1,1:3)/1000 1]';
ROTM=eul2tform(deg2rad(C3(1,4:6)),'XYZ');
```

```
TCP = [ROTM(:,1:3) TRVEC];
[configSol,solInfo] = ik('endeffector',TCP,[1 1 1 1 1 1],homeConfiguration(robot));
qhome=homeConfiguration(robot);
q=zeros(1,6);
for i=1:6
    q(i)=configSol(i).JointPosition-qhome(i).JointPosition;
end
q
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C4 = [123 456 789 -90 -45 -30]
TRVEC=[C4(1,1:3)/1000 1]';
ROTM=eul2tform(deg2rad(C4(1,4:6)),'XYZ');
TCP = [ROTM(:,1:3) TRVEC];
[configSol,solInfo] = ik('endeffector',TCP,[1 1 1 1 1 1],homeConfiguration(robot));
qhome=homeConfiguration(robot);
q=zeros(1,6);
for i=1:6
    q(i)=configSol(i).JointPosition-qhome(i).JointPosition;
end
q
```

```
C1 =

   500   500   500   -90     0     0


q =

    0.5278    0.6336   -1.0047   -1.6383   -0.5278   -0.0000


C2 =

   100  -200   300    90    90    45


q =

   -2.0566    2.1309    2.6423    0.5114    0.4858   -2.3562


C3 =

  -300   400  -500   -30   -45   -90


q =

    1.8991    3.1473    1.1838    0.3504    0.9820    1.2284


C4 =
```

```
   123   456   789   -90   -45   -30

q =

    0.8915    1.0338    0.8847   -0.1491   -0.1061   -0.5236
```

Ahora, con el fin de comparar la cinemática inversa por ambos métodos, realizamos un bloque de código para obtener la inversa numérica en RVC por medio de la función `ikinem`. Con el fin de obtener los mismos vectores ***q*** (considerando la multiplicidad de soluciones y los diferentes métodos que emplean las funciones en cada *Toolbox*) se agregó parte de las soluciones obtenidas por RST como punto de partida para la iteración de la función `ikinem`, obteniendo los siguientes valores:

#	X	Y	Z	ROLL	PITCH	YAW	q_1	q_2	q_3	q_4	q_5	q_6
1	500.00	500.00	500.00	-90°	0°	0°	0.5244	0.6435	-0.9884	-1.6306	-0.4682	-0.0001
2	100.00	-200.00	300.00	90°	90°	45°	-2.0629	2.1279	2.6315	0.4385	0.4639	-2.3012
3	-300.00	400.00	-500.00	-30°	-45°	-90°	1.8974	3.1440	1.1687	0.2981	0.9699	1.2520
4	123.00	456.00	789.00	-90°	-45°	-30°	0.8909	1.0333	0.8850	-0.1123	-0.0682	-0.5572

UR5e_RVC

```
%CINEMÁTICA INVERSA
options = struct('MaxFunEvals', 1000);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C1 = [500 500 500 -90 0 0]
TRVEC=[C1(1,1:3) 1]';
ROTM=rpy2tr(deg2rad(C1(1,4:6)));
TCP = [ROTM(:,1:3) TRVEC];
qC1 = Robot.ikinem(TCP,[0.5 0.6 -1.0 -1.6 -0.5 0.0])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C2 = [100 -200 300 90 90 45]
TRVEC=[C2(1,1:3) 1]';
ROTM=rpy2tr(deg2rad(C2(1,4:6)));
TCP = [ROTM(:,1:3) TRVEC];
qC2 = Robot.ikinem(TCP,[-2.0 2.1 2.6 0.5 0.4 -2.3])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C3 = [-300 400 -500 -30 -45 -90]
TRVEC=[C3(1,1:3) 1]';
ROTM=rpy2tr(deg2rad(C3(1,4:6)));
TCP = [ROTM(:,1:3) TRVEC];
qC3 = Robot.ikinem(TCP,[1.8 3.1 1.1 0.3 0.9 1.2])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C4 = [123 456 789 -90 -45 -30]
TRVEC=[C4(1,1:3) 1]';
ROTM=rpy2tr(deg2rad(C4(1,4:6)));
TCP = [ROTM(:,1:3) TRVEC];
```

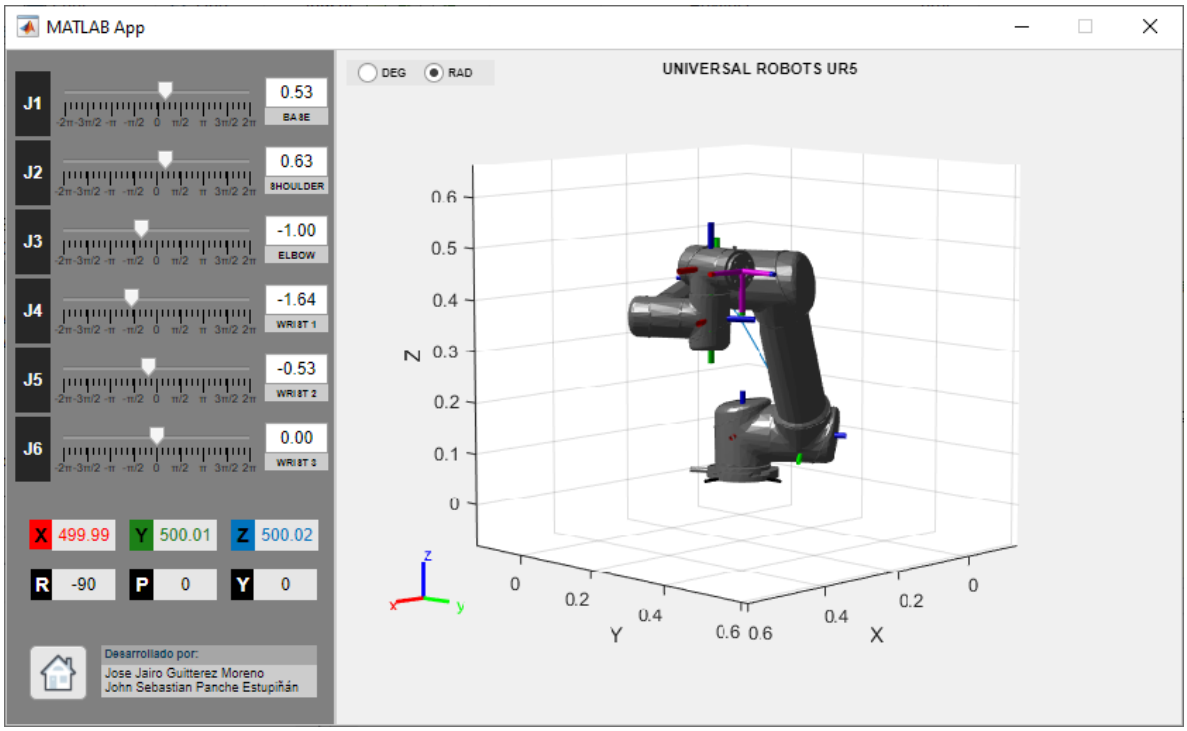
```
qC4 = Robot.ikinem(TCP,[0.8 1.0 0.8 -0.1 -0.1 -0.5])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
C1 =

   500   500   500   -90     0     0

qC1 =

    0.5244    0.6435   -0.9884   -1.6306   -0.4682   -0.0001

C2 =

   100  -200   300    90    90    45

qC2 =

   -2.0629    2.1279    2.6315    0.4385    0.4639   -2.3012

C3 =

  -300   400  -500   -30   -45   -90

qC3 =

    1.8974    3.1440    1.1687    0.2981    0.9699    1.2520

C4 =

   123   456   789   -90   -45   -30

qC4 =

    0.8909    1.0333    0.8850   -0.1123   -0.0682   -0.5572
```

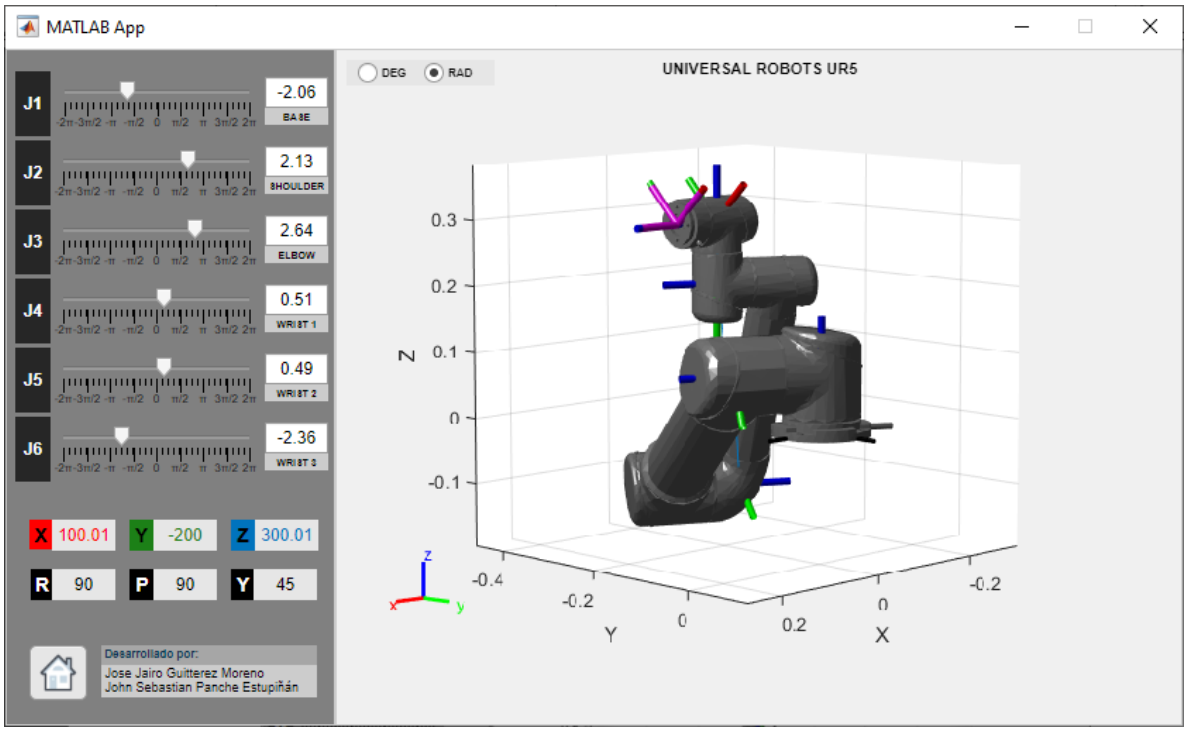
7) Haciendo uso de la GUI verifique que con la configuración calculada se obtiene la postura indicada de la herramienta.

Dado a que los vectores de ***q*** obtenidos por ambos métodos son bastante similares (con ligeras diferencias de precisión en los decimales), realizamos la validación de las configuraciones en la GUI con las soluciones obtenidas por el método de RST:

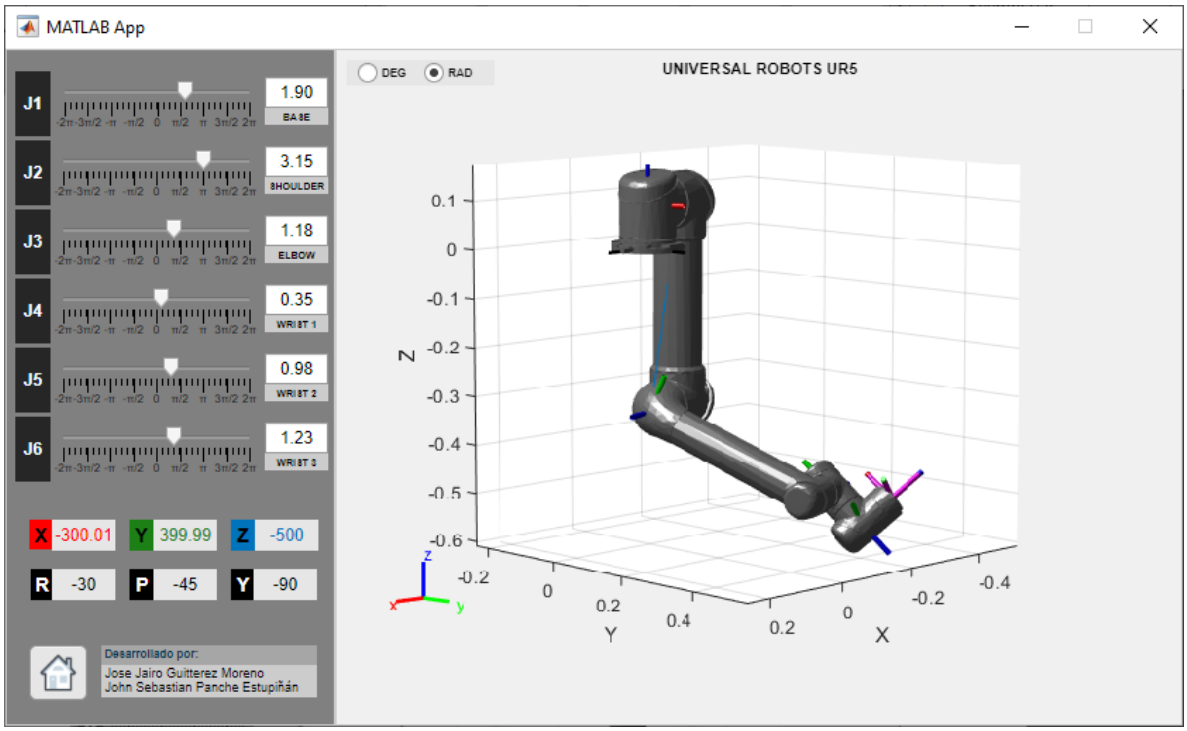
#	X	Y	Z	ROLL	PITCH	YAW	q_1	q_2	q_3	q_4	q_5	q_6
1	500.00	500.00	500.00	-90°	0°	0°	0.5278	0.6336	-1.0047	-1.6383	-0.5278	-0.0000



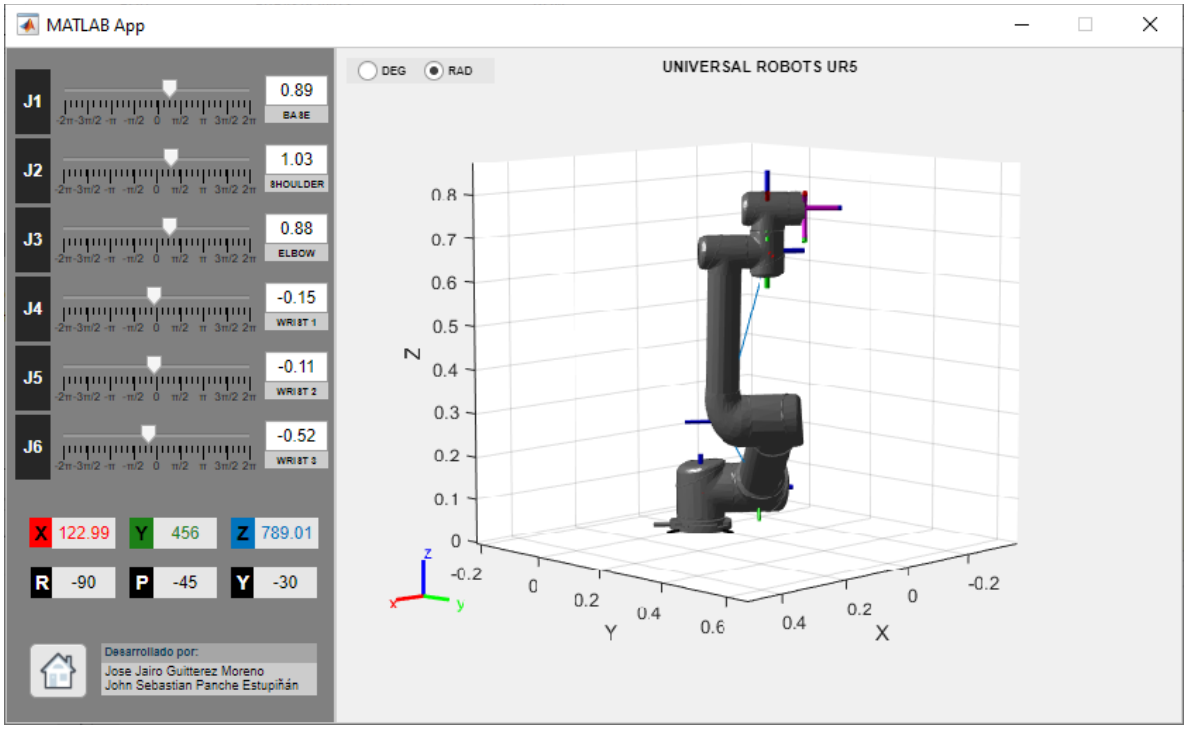
#	X	Y	Z	ROLL	PITCH	YAW	q_1	q_2	q_3	q_4	q_5	q_6
2	100.00	-200.00	300.00	90°	90°	45°	-2.0566	2.1309	2.6423	0.5114	0.4858	-2.3562



#	X	Y	Z	ROLL	PITCH	YAW	q_1	q_2	q_3	q_4	q_5	q_6
3	-300.00	400.00	-500.00	-30°	-45°	-90°	1.8991	3.1473	1.1838	0.3504	0.9820	1.2284



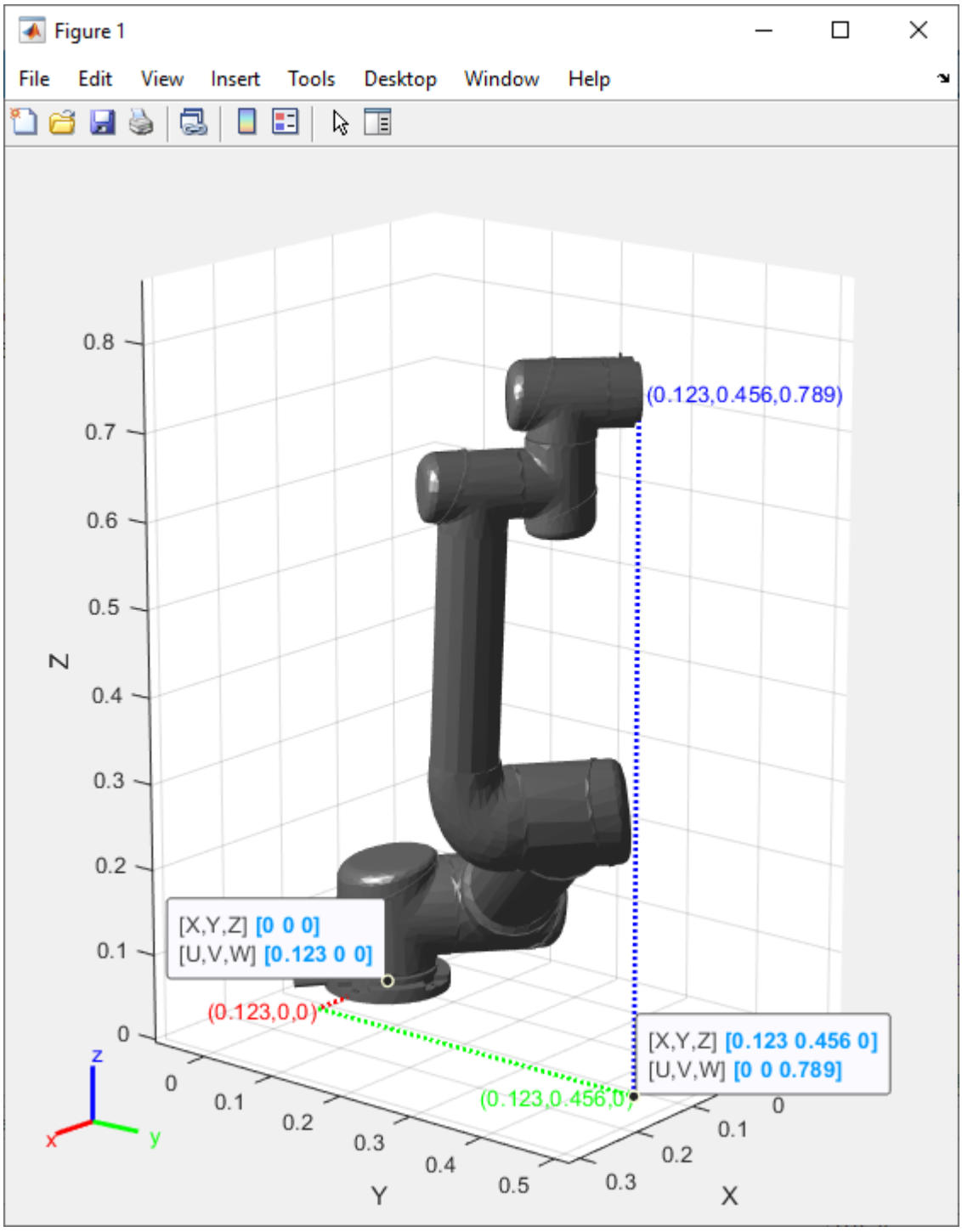
#	X	Y	Z	ROLL	PITCH	YAW	q_1	q_2	q_3	q_4	q_5	q_6
4	123.00	456.00	789.00	-90°	-45°	-30°	0.8915	1.0338	0.8847	-0.1491	-0.1061	-0.5236



8) Para uno de los cuatro casos haga capturas de pantalla resaltando con cotas los valores x y z, e incluya en el informe los resultados.

En el numeral anterior se incluyen las capturas desde la GUI para los 4 casos, y en donde se pueden destacar que la posición (*x, y, z*) observada para el sistema NOA en cada una de las gráficas coincide con los valores de los campos **X**, **Y**, **Z**, esto teniendo en consideración que la escala que se maneja para RST es de 1/1000 por limitaciones de la toolbox.

Con el fin de resaltar con cotas los valores de posición conforme se requiere en el enunciado, realizamos un bloque de código adicional con el fin de tomar el último *q* calculado para la posición (123.00, 456.00, 789.00) en el numeral anterior y graficar cada una de las cotas como vectores en 'X', 'Y' y 'Z' partiendo del origen hasta la respectiva posición del TCP. A continuación se presenta la captura de la gráfica resultante y el bloque de código utilizado (por el método de RST) para corroborar la transparencia en los valores de las cotas graficadas.



UR5e_RST

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C4 = [123 456 789 -90 -45 -30]
TRVEC=[C4(1,1:3)/1000 1]';
ROTM=eul2tform(deg2rad(C4(1,4:6)),'XYZ');
```

```
TCP = [ROTM(:,1:3) TRVEC];
[configSol,solInfo] = ik('endeffector',TCP,[1 1 1 1 1 1],homeConfiguration(robot));
qhome=homeConfiguration(robot);
q=zeros(1,6);
for i=1:6
    q(i)=configSol(i).JointPosition-qhome(i).JointPosition;
end
q
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
config=q;
qtest=homeConfiguration(robot);
for i=1:6
    qtest(i).JointPosition = qtest(i).JointPosition + config(i);
end
TCP=getTransform(robot,qtest,'endeffector');
POS=tform2trvec(TCP);
show(robot,qtest,'Frames','off')
axis tight;
hold on;
quiver3(0,0,0,POS(1),0,0,':r','LineWidth',2,'AutoScale','off','ShowArrowHead','off')
quiver3(POS(1),0,0,0,POS(2),0,':g','LineWidth',2,'AutoScale','off','ShowArrowHead','off')
quiver3(POS(1),POS(2),0,0,0,POS(3),':b','LineWidth',2,'AutoScale','off','ShowArrowHead','of
f')
text(POS(1),0,0," ("+POS(1)+",0,0)",'Color','r','HorizontalAlignment','right')
text(POS(1),POS(2),0,"("+POS(1)+","+POS(2)+",0)",'Color','g','HorizontalAlignment','right')
text(POS(1),POS(2),POS(3),"
("+POS(1)+","+POS(2)+","+POS(3)+")",'Color','b','HorizontalAlignment','left')
```

MODELO DIFERENCIAL DE PRIMER ORDEN

Considerando que para que el robot ejecute los movimientos, se desea establecer una relación entre las velocidades del efector final y las articulaciones.

1) Con los valores numéricos de longitudes y desplazamientos de los eslabones del robot asignado obtenga el Jacobiano (como una matriz de valores numéricos, NO simbólico) en función de los ángulos de articulación.

Dado que el Jacobiano dependerá de los ángulos de articulación en la configuración $\boldsymbol{q}$, se desarrolla el código por el método de RVC para obtener tanto el Jacobiano geométrico como el Jacobiano analítico en función de $\boldsymbol{q} = [q_1, q_2, q_3, q_4, q_5, q_6]$.

Además, considerando que según el enunciado se requiere como respuesta una matriz de valores numéricos, y que el siguiente numeral requiere calcular el Jacobiano analítico para la 1er postura del punto anterior, se opta por realizar el cálculo del Jacobiano de velocidad por los tres métodos presentados en el bloque de código de RVC para la postura $\boldsymbol{q}$ = [0.5278 0.6336 -1.0047 -1.6383 -0.5278 -0.0000].

UR5e_RVC

```
%MODELO DIFERENCIAL DE PRIMER ORDEN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Jacobiano geométrico/convencional/base (6xn)
MTH_0T1 = MTH_0T1;
MTH_0T2 = MTH_0T1*MTH_1T2;
MTH_0T3 = MTH_0T1*MTH_1T2*MTH_2T3;
MTH_0T4 = MTH_0T1*MTH_1T2*MTH_2T3*MTH_3T4;
MTH_0T5 = MTH_0T1*MTH_1T2*MTH_2T3*MTH_3T4*MTH_4T5;
MTH_0T6 = MTH_0T1*MTH_1T2*MTH_2T3*MTH_3T4*MTH_4T5*MTH_5T6;

%Cálculo de los Z respecto a 0
Z01=MTH_0T1(1:3,3);
Z02=MTH_0T2(1:3,3);
Z03=MTH_0T3(1:3,3);
Z04=MTH_0T4(1:3,3);
Z05=MTH_0T5(1:3,3);
Z06=MTH_0T6(1:3,3);

%Cálculo de los P
P17=TCP(1:3,4)-MTH_0T1(1:3,4);
P27=TCP(1:3,4)-MTH_0T2(1:3,4);
P37=TCP(1:3,4)-MTH_0T3(1:3,4);
P47=TCP(1:3,4)-MTH_0T4(1:3,4);
P57=TCP(1:3,4)-MTH_0T5(1:3,4);
P67=TCP(1:3,4)-MTH_0T6(1:3,4);

%Cálculo de las J
Jg1=[cross(Z01,P17); Z01]; %revolute
Jg2=[cross(Z02,P27); Z02]; %revolute
Jg3=[cross(Z03,P37); Z03]; %revolute
Jg4=[cross(Z04,P47); Z04]; %revolute
Jg5=[cross(Z05,P57); Z05]; %revolute
Jg6=[cross(Z06,P67); Z06]; %revolute
Jg=simplify([Jg1 Jg2 Jg3 Jg4 Jg5 Jg6])

%//////////////////////////////////////////////////////////////////////////

%Jacobiano analítico de posición en coordenadas cartesianas (3xn)
posX = TCP(1,4);
posY = TCP(2,4);
posZ = TCP(3,4);
J11 = diff(posX,q1); J12 = diff(posX,q2); J13 = diff(posX,q3); J14 = diff(posX,q4); J15 =
diff(posX,q5);  J16 = diff(posX,q6);
J21 = diff(posY,q1); J22 = diff(posY,q2); J23 = diff(posY,q3); J24 = diff(posY,q4); J25 =
diff(posY,q5);  J26 = diff(posY,q6);
J31 = diff(posZ,q1); J32 = diff(posZ,q2); J33 = diff(posZ,q3); J34 = diff(posZ,q4); J35 =
diff(posZ,q5);  J36 = diff(posZ,q6);
Jpos = simplify([J11 J12 J13 J14 J15 J16;
                 J21 J22 J23 J24 J25 J26;
                 J31 J32 J33 J34 J35 J36]);
```

```
%Jacobiano analítico de orientación por derivada de la matriz de rotación
syms q1_dot q2_dot q3_dot q4_dot q5_dot q6_dot q1 q2 q3 q4 q5 q6
R_p1 = diff(TCP(1:3,1:3),q1)*q1_dot;
R_p2 = diff(TCP(1:3,1:3),q2)*q2_dot;
R_p3 = diff(TCP(1:3,1:3),q3)*q3_dot;
R_p4 = diff(TCP(1:3,1:3),q4)*q4_dot;
R_p5 = diff(TCP(1:3,1:3),q5)*q5_dot;
R_p6 = diff(TCP(1:3,1:3),q6)*q6_dot;

R_p = simplify(R_p1 + R_p2 + R_p3 + R_p4 + R_p5 + R_p6);
skew = simplify (R_p * transpose(TCP(1:3,1:3)));
%Se obtiene el vector w
w=[skew(3,2) skew(1,3) skew(2,1)];
w=expand(w);
w_temp=children(w);

Jrot=[0 w_temp{1}(1)/q2_dot w_temp{1}(2)/q3_dot w_temp{1}(3)/q4_dot
(w_temp{1}(5)+w_temp{1}(6)+w_temp{1}(7)+w_temp{1}(8))/q5_dot
(w_temp{1}(4)+w_temp{1}(9)+w_temp{1}(10)+w_temp{1}(11)+w_temp{1}(12))/q6_dot;
      0 w_temp{2}(1)/q2_dot w_temp{2}(2)/q3_dot w_temp{2}(3)/q4_dot
(w_temp{2}(5)+w_temp{2}(6)+w_temp{2}(7)+w_temp{2}(8))/q5_dot
(w_temp{2}(4)+w_temp{2}(9)+w_temp{2}(10)+w_temp{2}(11)+w_temp{2}(12))/q6_dot;
      w_temp{3}(1)/q1_dot 0 0 0
(w_temp{3}(2)+w_temp{3}(3)+w_temp{3}(4)+w_temp{3}(5))/q5_dot
(w_temp{3}(6)+w_temp{3}(7)+w_temp{3}(8)+w_temp{3}(9))/q6_dot;];

Ja = simplify([Jpos;Jrot])

%////////////////////////////////////////////////////////////////////////
q = [0.5278 0.6336 -1.0047 -1.6383 -0.5278 0.0000]
J = Robot.jacob0(q)
%Se evalúan (subs) y simplifican (vpa) los J para el q dado
Jg_q = vpa(subs(Jg,[q1,q2,q3,q4,q5,q6],[q]),6)
Ja_q = vpa(subs(Ja,[q1,q2,q3,q4,q5,q6],[q]),6)
%////////////////////////////////////////////////////////////////////////
```

```
Jg =

[ 392*cos(q2)*sin(q1)*sin(q3) - (163*cos(q1)*cos(q5))/2 - 425*sin(q1)*sin(q2) -
(549*cos(q1))/5 - 392*cos(q3)*sin(q1)*sin(q2) - (474*cos(q2)*cos(q3)*sin(q1)*sin(q4))/5 +
(474*cos(q2)*cos(q4)*sin(q1)*sin(q3))/5 - (474*cos(q3)*cos(q4)*sin(q1)*sin(q2))/5 -
(474*sin(q1)*sin(q2)*sin(q3)*sin(q4))/5 + (163*cos(q2)*cos(q3)*cos(q4)*sin(q1)*sin(q5))/2 +
(163*cos(q2)*sin(q1)*sin(q3)*sin(q4)*sin(q5))/2 -
(163*cos(q3)*sin(q1)*sin(q2)*sin(q4)*sin(q5))/2 +
(163*cos(q4)*sin(q1)*sin(q2)*sin(q3)*sin(q5))/2, cos(q1)*(392*cos(q2 - q3) + (163*cos(q2 -
q3 + q4 - q5))/4 + (474*cos(q2 - q3 + q4))/5 + 425*cos(q2) - (163*cos(q2 - q3 + q4 +
q5))/4), -cos(q1)*(392*cos(q2 - q3) + (163*cos(q2 - q3 + q4 - q5))/4 + (474*cos(q2 - q3 +
q4))/5 - (163*cos(q2 - q3 + q4 + q5))/4), cos(q1)*((163*cos(q2 - q3 + q4 - q5))/4 +
(474*cos(q2 - q3 + q4))/5 - (163*cos(q2 - q3 + q4 + q5))/4), (163*sin(q1)*sin(q5))/2 -
(163*cos(q1)*cos(q2)*cos(q3)*cos(q4)*cos(q5))/2 -
(163*cos(q1)*cos(q2)*cos(q5)*sin(q3)*sin(q4))/2 +
(163*cos(q1)*cos(q3)*cos(q5)*sin(q2)*sin(q4))/2 -
```

```
(163*cos(q1)*cos(q4)*cos(q5)*sin(q2)*sin(q3))/2,
0]
[ 425*cos(q1)*sin(q2) - (549*sin(q1))/5 - (163*cos(q5)*sin(q1))/2 -
392*cos(q1)*cos(q2)*sin(q3) + 392*cos(q1)*cos(q3)*sin(q2) +
(474*cos(q1)*cos(q2)*cos(q3)*sin(q4))/5 - (474*cos(q1)*cos(q2)*cos(q4)*sin(q3))/5 +
(474*cos(q1)*cos(q3)*cos(q4)*sin(q2))/5 + (474*cos(q1)*sin(q2)*sin(q3)*sin(q4))/5 -
(163*cos(q1)*cos(q2)*cos(q3)*cos(q4)*sin(q5))/2 -
(163*cos(q1)*cos(q2)*sin(q3)*sin(q4)*sin(q5))/2 +
(163*cos(q1)*cos(q3)*sin(q2)*sin(q4)*sin(q5))/2 -
(163*cos(q1)*cos(q4)*sin(q2)*sin(q3)*sin(q5))/2, sin(q1)*(392*cos(q2 - q3) + (163*cos(q2 -
q3 + q4 - q5))/4 + (474*cos(q2 - q3 + q4))/5 + 425*cos(q2) - (163*cos(q2 - q3 + q4 +
q5))/4), -sin(q1)*(392*cos(q2 - q3) + (163*cos(q2 - q3 + q4 - q5))/4 + (474*cos(q2 - q3 +
q4))/5 - (163*cos(q2 - q3 + q4 + q5))/4), sin(q1)*((163*cos(q2 - q3 + q4 - q5))/4 +
(474*cos(q2 - q3 + q4))/5 - (163*cos(q2 - q3 + q4 + q5))/4),
(163*cos(q3)*cos(q5)*sin(q1)*sin(q2)*sin(q4))/2 -
(163*cos(q2)*cos(q3)*cos(q4)*cos(q5)*sin(q1))/2 -
(163*cos(q2)*cos(q5)*sin(q1)*sin(q3)*sin(q4))/2 - (163*cos(q1)*sin(q5))/2 -
(163*cos(q4)*cos(q5)*sin(q1)*sin(q2)*sin(q3))/2,
0]
[
0,           (163*sin(q2 - q3 + q4 + q5))/4 - (163*sin(q2 - q3 + q4 - q5))/4 - (474*sin(q2
- q3 + q4))/5 - 425*sin(q2) - 392*sin(q2 - q3),            392*sin(q2 - q3) + (163*sin(q2 -
q3 + q4 - q5))/4 + (474*sin(q2 - q3 + q4))/5 - (163*sin(q2 - q3 + q4 + q5))/4,
(163*sin(q2 - q3 + q4 + q5))/4 - (474*sin(q2 - q3 + q4))/5 - (163*sin(q2 - q3 + q4 -
q5))/4,
(163*sin(q2 - q3 + q4 - q5))/4 + (163*sin(q2 - q3 + q4 + q5))/4,
0]
[
0,
-sin(q1),
sin(q1),
-sin(q1),
sin(q2 - q3 + q4)*cos(q1), - cos(q5)*sin(q1) - cos(q2 - q3 + q4)*cos(q1)*sin(q5)]
[
0,
cos(q1),
-cos(q1),
cos(q1),
sin(q2 - q3 + q4)*sin(q1),   cos(q1)*cos(q5) - cos(q2 - q3 + q4)*sin(q1)*sin(q5)]
[
1,
0,
0,
0,
cos(q2 - q3 + q4),                              sin(q2 - q3 + q4)*sin(q5)]


Ja =

[ 392*cos(q2)*sin(q1)*sin(q3) - (163*cos(q1)*cos(q5))/2 - 425*sin(q1)*sin(q2) -
(549*cos(q1))/5 - 392*cos(q3)*sin(q1)*sin(q2) - (474*cos(q2)*cos(q3)*sin(q1)*sin(q4))/5 +
(474*cos(q2)*cos(q4)*sin(q1)*sin(q3))/5 - (474*cos(q3)*cos(q4)*sin(q1)*sin(q2))/5 -
(474*sin(q1)*sin(q2)*sin(q3)*sin(q4))/5 + (163*cos(q2)*cos(q3)*cos(q4)*sin(q1)*sin(q5))/2 +
(163*cos(q2)*sin(q1)*sin(q3)*sin(q4)*sin(q5))/2 -
```

```
(163*cos(q3)*sin(q1)*sin(q2)*sin(q4)*sin(q5))/2 +
(163*cos(q4)*sin(q1)*sin(q2)*sin(q3)*sin(q5))/2, (cos(q1)*(3920*cos(q2 - q3) + (815*cos(q2
- q3 + q4 - q5))/2 + 948*cos(q2 - q3 + q4) + 4250*cos(q2) - (815*cos(q2 - q3 + q4 +
q5))/2))/10, -(cos(q1)*(3920*cos(q2 - q3) + (815*cos(q2 - q3 + q4 - q5))/2 + 948*cos(q2 -
q3 + q4) - (815*cos(q2 - q3 + q4 + q5))/2))/10, (cos(q1)*(815*cos(q2 - q3 + q4 - q5) +
1896*cos(q2 - q3 + q4) - 815*cos(q2 - q3 + q4 + q5)))/20, (163*sin(q1)*sin(q5))/2 -
(163*cos(q1)*cos(q2)*cos(q3)*cos(q4)*cos(q5))/2 -
(163*cos(q1)*cos(q2)*cos(q5)*sin(q3)*sin(q4))/2 +
(163*cos(q1)*cos(q3)*cos(q5)*sin(q2)*sin(q4))/2 -
(163*cos(q1)*cos(q4)*cos(q5)*sin(q2)*sin(q3))/2,
0]
[ 425*cos(q1)*sin(q2) - (549*sin(q1))/5 - (163*cos(q5)*sin(q1))/2 -
392*cos(q1)*cos(q2)*sin(q3) + 392*cos(q1)*cos(q3)*sin(q2) +
(474*cos(q1)*cos(q2)*cos(q3)*sin(q4))/5 - (474*cos(q1)*cos(q2)*cos(q4)*sin(q3))/5 +
(474*cos(q1)*cos(q3)*cos(q4)*sin(q2))/5 + (474*cos(q1)*sin(q2)*sin(q3)*sin(q4))/5 -
(163*cos(q1)*cos(q2)*cos(q3)*cos(q4)*sin(q5))/2 -
(163*cos(q1)*cos(q2)*sin(q3)*sin(q4)*sin(q5))/2 +
(163*cos(q1)*cos(q3)*sin(q2)*sin(q4)*sin(q5))/2 -
(163*cos(q1)*cos(q4)*sin(q2)*sin(q3)*sin(q5))/2, (sin(q1)*(3920*cos(q2 - q3) + (815*cos(q2
- q3 + q4 - q5))/2 + 948*cos(q2 - q3 + q4) + 4250*cos(q2) - (815*cos(q2 - q3 + q4 +
q5))/2))/10, -(sin(q1)*(3920*cos(q2 - q3) + (815*cos(q2 - q3 + q4 - q5))/2 + 948*cos(q2 -
q3 + q4) - (815*cos(q2 - q3 + q4 + q5))/2))/10, (sin(q1)*(815*cos(q2 - q3 + q4 - q5) +
1896*cos(q2 - q3 + q4) - 815*cos(q2 - q3 + q4 + q5)))/20,
(163*cos(q3)*cos(q5)*sin(q1)*sin(q2)*sin(q4))/2 -
(163*cos(q2)*cos(q3)*cos(q4)*cos(q5)*sin(q1))/2 -
(163*cos(q2)*cos(q5)*sin(q1)*sin(q3)*sin(q4))/2 - (163*cos(q1)*sin(q5))/2 -
(163*cos(q4)*cos(q5)*sin(q1)*sin(q2)*sin(q3))/2,
0]
[
0,              (163*sin(q2 - q3 + q4 + q5))/4 - (163*sin(q2 - q3 + q4 - q5))/4 -
(474*sin(q2 - q3 + q4))/5 - 425*sin(q2) - 392*sin(q2 - q3),              392*sin(q2 - q3) +
(163*sin(q2 - q3 + q4 - q5))/4 + (474*sin(q2 - q3 + q4))/5 - (163*sin(q2 - q3 + q4 +
q5))/4,     (163*sin(q2 - q3 + q4 + q5))/4 - (474*sin(q2 - q3 + q4))/5 - (163*sin(q2 - q3 +
q4 - q5))/4,
(163*sin(q2 - q3 + q4 - q5))/4 + (163*sin(q2 - q3 + q4 + q5))/4,
0]
[
0,
-sin(q1),
sin(q1),
-sin(q1),
sin(q1 + q2 - q3 + q4)/2 - sin(q1 - q2 + q3 - q4)/2,
cos(q1)*cos(q3)*sin(q2)*sin(q4)*sin(q5) - cos(q1)*cos(q2)*cos(q3)*cos(q4)*sin(q5) -
cos(q1)*cos(q2)*sin(q3)*sin(q4)*sin(q5) - cos(q5)*sin(q1) -
cos(q1)*cos(q4)*sin(q2)*sin(q3)*sin(q5)]
[
0,
cos(q1),
-cos(q1),
cos(q1),
cos(q1 - q2 + q3 - q4)/2 - cos(q1 + q2 - q3 + q4)/2, cos(q1)*cos(q5) -
cos(q2)*cos(q3)*cos(q4)*sin(q1)*sin(q5) - cos(q2)*sin(q1)*sin(q3)*sin(q4)*sin(q5) +
cos(q3)*sin(q1)*sin(q2)*sin(q4)*sin(q5) - cos(q4)*sin(q1)*sin(q2)*sin(q3)*sin(q5)]
[
```

```
1,
0,
0,
0,
cos(q2 - q3 + q4),
cos(q2 - q3 + q4 - q5)/2 - cos(q2 - q3 + q4 + q5)/2]
```

Finalmente, se logra evidenciar que el Jacobiano calculado para el ***q*** seleccionado coincide (con diferencias poco significativas en los decimales) para todos los métodos aplicados en RVC:

```
q =

    0.5278    0.6336   -1.0047   -1.6383   -0.5278         0


J =

 -500.0147  354.9116  -59.0130   81.8562  -81.5000         0
  499.9903  206.9012  -34.4026   47.7193    0.0000         0
    0.0000 -683.7746  432.1534  -41.0462   -0.0000         0
    0.0000   -0.5036    0.5036   -0.5036   -0.0000    0.0000
   -0.0000    0.8639   -0.8639    0.8639    0.0000    1.0000
    1.0000    0.0000    0.0000   -0.0000    1.0000   -0.0000


Jg_q =

[ -500.058,   354.955,  -59.0562,   81.8993,      -81.5,   0]
[  499.965,   206.926,  -34.4277,   47.7445, 1.0516e-12,   0]
[       0,  -683.775,   432.153,  -41.0462,          0,   0]
[       0, -0.503634,  0.503634, -0.503634,          0,   0]
[       0,  0.863917, -0.863917,  0.863917,          0, 1.0]
[     1.0,         0,         0,         0,        1.0,   0]


Ja_q =

[ -500.058,   354.955,  -59.0562,   81.8993,      -81.5,            0]
[  499.965,   206.926,  -34.4277,   47.7445, 1.0516e-12,            0]
[       0,  -683.775,   432.153,  -41.0462,          0,            0]
[       0, -0.503634,  0.503634, -0.503634,          0, -2.66454e-15]
[       0,  0.863917, -0.863917,  0.863917,          0,          1.0]
[     1.0,         0,         0,         0,        1.0,            0]
```

2) Para la postura 1 de la tabla anterior, obtenga las velocidades de articulación para:

$$v_H = \begin{bmatrix} 100 \\ 200 \\ 50 \end{bmatrix} \text{mm/s} \qquad \omega_H = \begin{bmatrix} 5 \\ 10 \\ -5 \end{bmatrix} \text{rad/s}$$

Dado que en el numeral anterior ya calculamos el Jacobiano numérico para la 1er postura del punto solicitado ($\boldsymbol{q}$ = [0.5278 0.6336 -1.0047 -1.6383 -0.5278 -0.0000]), para obtener el vector de velocidades de articulación $\dot{\boldsymbol{q}}$ podemos despejar la ecuación:

$$\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}_{(6x1)} = J_0(\boldsymbol{q})_{(6xn)} \cdot \dot{\boldsymbol{q}}_{(nx1)}$$

$$J_0(\boldsymbol{q})^{-1}_{(6xn)} \cdot \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}_{(6x1)} = \dot{\boldsymbol{q}}_{(nx1)}$$

Para este caso, igual procedemos a confirmar el Jacobiano calculado y verificado previamente pero ahora utilizando la función geometricJacobian del RST, de modo que no quede duda en el cálculo de este y confirmando que los valores obtenidos coinciden nuevamente (salvo diferencias poco significativas en los decimales):

UR5e_RST

```
%MODELO DIFERENCIAL DE PRIMER ORDEN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q = homeConfiguration(robot);
configq = [0.5278 0.6336 -1.0047 -1.6383 -0.5278 -0.0000]
for i=1:6
    q(i).JointPosition = q(i).JointPosition + configq(i);
end
J0 = geometricJacobian(robot,q,'endeffector');
J0 = [J0(4:6,:)*1000;J0(1:3,:)]

v=[100 200 50]'
w=[5 10 -5]'
q_dot=inv(J0)*[v;w]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
configq =

    0.5278    0.6336   -1.0047   -1.6383   -0.5278         0


J0 =

 -500.0147  354.9116  -59.0130   81.8562  -81.5000    0.0000
  499.9903  206.9012  -34.4026   47.7193    0.0000   -0.0000
         0 -683.7746  432.1534  -41.0462   -0.0000   -0.0000
   -0.0000   -0.5036    0.5036   -0.5036   -0.0000    0.0000
   -0.0000    0.8639   -0.8639    0.8639    0.0000    1.0000
    1.0000    0.0000    0.0000   -0.0000    1.0000   -0.0000


v =
```

```
   100
   200
    50


w =

     5
    10
    -5


q_dot =

    0.5098
    2.3805
    2.9980
   -9.3104
   -5.5098
   18.5768
```

Finalmente, ya con el Jacobiano de la configuración $\boldsymbol{q}$ y los vectores $\boldsymbol{v}$ y $\boldsymbol{\omega}$, obtenemos el vector de velocidades de articulación de dimensión (nx1) que corresponde a $\dot{\boldsymbol{q}}$ = [0.5098, 2.3805 2.9980 -9.3104 -5.5098 18.5768].

INTEGRACIÓN

Ahora con la ayuda de los algoritmos desarrollados y de la GUI construida.

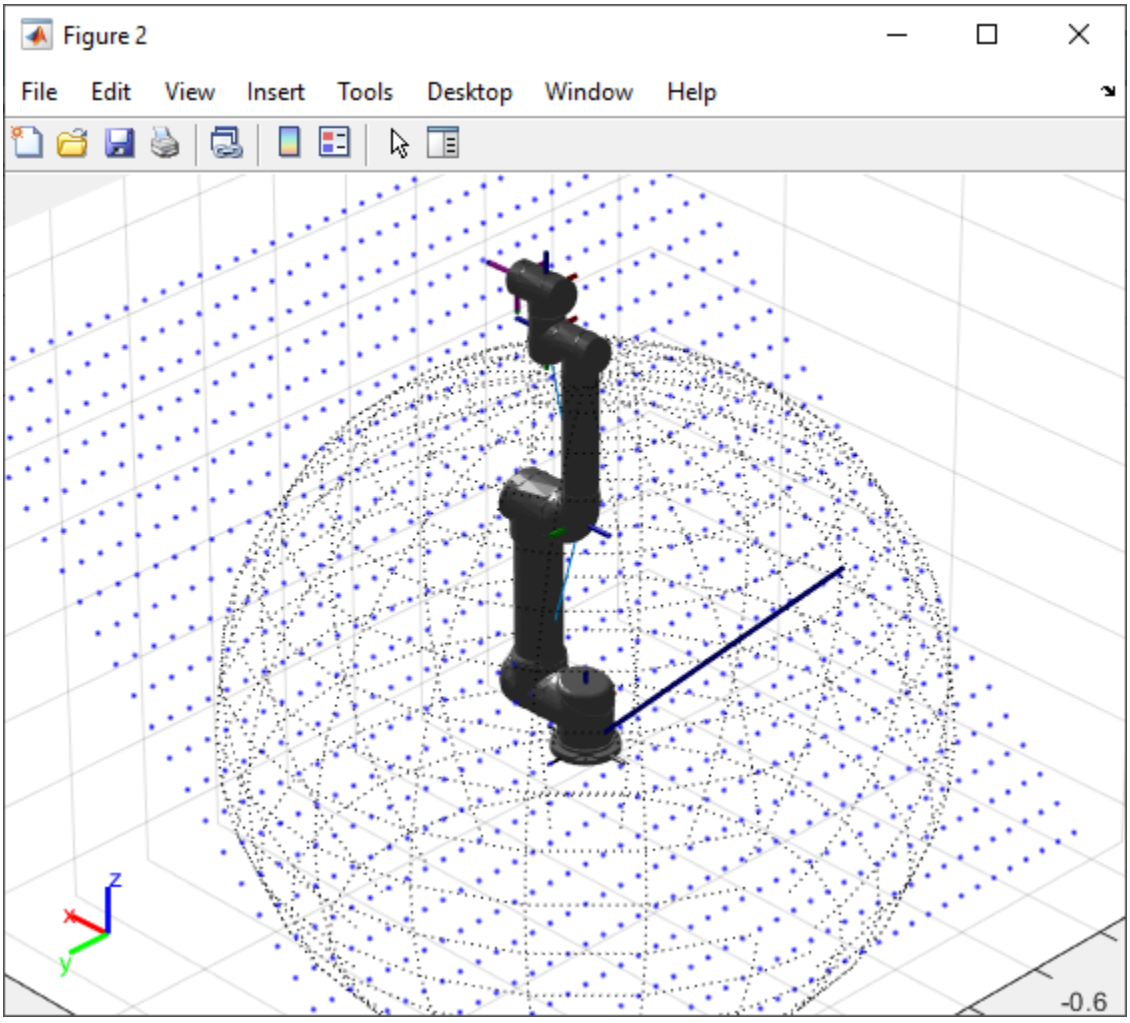
1) Ubique la ruta seleccionada con la orientación indicada dentro del espacio diestro del robot.

Partiendo del gráfico del plano asignado y el cual se obtuvo al principio del laboratorio, se adiciona el modelo del robot con base en el origen para permitirnos dimensionar el alcance del robot y ubicar así al ruta dentro del espacio diestro de trabajo. En la gráfica, la esfera representa el espacio de trabajo recomendado en la guía de usuario del UR5, el cuál limita el alcance a 750mm.

Universal Robots UR5e

```
%INTEGRACIÓN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GRAFICA DEL PLANO
figure(2)
vector = [-1 0 1];
x=0; y=0; z=0; d=0;
q=homeConfiguration(robot);
```

```
q(1).JointPosition=-pi/2;%Rotación del Robot para poner en posición del plano
show(robot,q)
hold on
quiver3(0,0,0,-1,0,1,'Color','#000058','LineWidth',2,'AutoScale','off')
axis equal
axis([-0.5 1.1 -1.0 1.0 -0.8 0.8])
view([-135 30])
r = 750/1000; %Para un espacio de trabajo de 750mm
[Xs,Ys,Zs]=sphere;
Xs=Xs*r; Ys=Ys*r;Zs=Zs*r;
surf(Xs,Ys,Zs+89.2/1000,'LineStyle',':','FaceColor','none')
for i=-1:0.05:1
    for j=-1:0.05:1
        x=i; y=j; z=1*x-0*y+d; %Ecuación del plano: -1x+0y+1z=0
        plot3(x,y,z,'b.','MarkerSize',5)
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



2) Defina un conjunto de puntos equidistantes que pertenezcan a la ruta (viapoints). Mínimo 60 puntos. El eje z de la herramienta debe mantenerse perpendicular al plano que contiene la ruta.

Dado que la ruta seleccionada corresponde a un rectángulo de esquinas redondeadas (con un ángulo no especificado), se genera un bloque de código por medio de estructuras de control que nos permite obtener una serie de puntos equidistantes sobre el plano de trabajo. Para este algoritmo, partimos de fraccionar la ruta completa en 4 segmentos de recta (`sideUp`, `sideRight`, `sideDown`, `sideLeft`) y 4 segmentos de arco (`corner1`, `corner2`, `corner3`, `corner3`), cada uno de las cuáles tiene asociada una ecuación y por tanto se requiere un cálculo diferente para la obtención de los puntos:

Universal Robots UR5e

```
%TRAYECTORIAS Y PUNTOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
step = 0.005;
lim = 0.7*750/1000;
radius = 0.07; %Radio de redondeo de las esquinas
d = 0.4; %Desplazamiento X de la ruta en el plano
viapoints=[0 0 0];

%sideUp
for i=-lim/2+radius:step:lim/2-radius
    x=lim/5+d; y=-i; z=1*x-0*y; %Ecuación del plano: -1x+0y+1z=0
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%corner1
for i=0:step/2:radius
    x=lim/5-radius+d+sqrt(radius^2-i^2); y=-lim/2+radius-i; z=1*x-0*y;
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%sideRight
for i=-lim/5+radius:step:lim/5-radius
    x=-i+d; y=-lim/2; z=1*x-0*y; %Ecuación del plano: -1x+0y+1z=0
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%corner2
for i=0:step/2:radius
    x=-lim/5+radius+d-i; y=-lim/2+radius-sqrt(radius^2-i^2); z=1*x-0*y;
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%sideDown
for i=-lim/2+radius:step:lim/2-radius
    x=-lim/5+d; y=i; z=1*x-0*y; %Ecuación del plano: -1x+0y+1z=0
```

```
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%corner3
for i=0:step/2:radius
    x=-lim/5+radius-sqrt(radius^2-i^2)+d; y=lim/2-radius+i; z=1*x-0*y;
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%sideLeft
for i=-lim/5+radius:step:lim/5-radius
    x=i+d; y=lim/2; z=1*x-0*y; %Ecuación del plano: -1x+0y+1z=0
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
%corner4
for i=0:step/2:radius
    x=lim/5-radius+i+d; y=lim/2-radius+sqrt(radius^2-i^2); z=1*x-0*y;
    plot3(x,y,z,'b.','MarkerSize',12)
    viapoints = [viapoints; [x y z]];
end
viapoints
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Además, se consideró el orden en que se recorren los segmentos de recta y arco para obtenerlos todos en un mismo sentido, de modo que al momento de crear la trayectoria esto no implique para el robot movimientos independientes efectuando cada segmento de forma individual, sino que realice toda la ruta como un movimiento continuo.

En cada una de las estructuras de control `for` y por cada iteración, se almacena el punto obtenido como un vector fila de tres elementos (*x, y, z*) en la matriz `viapoints`, de esta manera una vez ejecutado todo el bloque, el tamaño en filas de dicha matriz corresponderá al total de puntos obtenidos bajo la resolución actual (`step`).

El tamaño de la matriz de puntos obtenida con el código presentado es de aproximadamente 300 filas, razón por la cuál no se agrega directamente en el informe pero se encuentra entre los documentos adjuntos.

3) Calcule las configuraciones correspondientes a cada viapoint. Presente una gráfica de cada ángulo de articulación al recorrer la ruta.

En este punto, tenemos todas las posiciones deseadas para que la trayectoria pueda recorrer la ruta seleccionado. Ahora, dado que requerimos obtener los vectores de configuración de las articulaciones para cada postura del robot, debemos precisar cuál será la orientación de cada punto. En la guía, se especifica que el eje z de la herramienta o del efector final debe ser normal al plano, por tanto, el TCP irá orientado en la misma dirección de nuestro vector [-1, 0, 1], y esto se logra aplicando una orientación en ángulos fijos de (0° 135° 180°) para todas las posturas. Previo a aplicarlo en el código,

se validaron unas posturas con esta configuración haciendo de la GUI desarrollado en los numerales anteriores.

En un numeral anterior, se desarrolló un breve algoritmo que recibe como entrada una postura (*x, y, z, roll, pitch, yaw*) y devuelve la solución numérica de la cinemática inversa en un vector `configSol`. Dado que se requiere efectuar éste mismo procedimiento para cada una de las filas de la matriz `viapoints` se agrega ese código dentro de una estructura de control para obtener la cinemática inversa de cada uno de los puntos, calculando la siguiente con la referencia anterior, de modo que se limiten las soluciones a encontrar la más cercana a la configuración inmediatamente anterior.

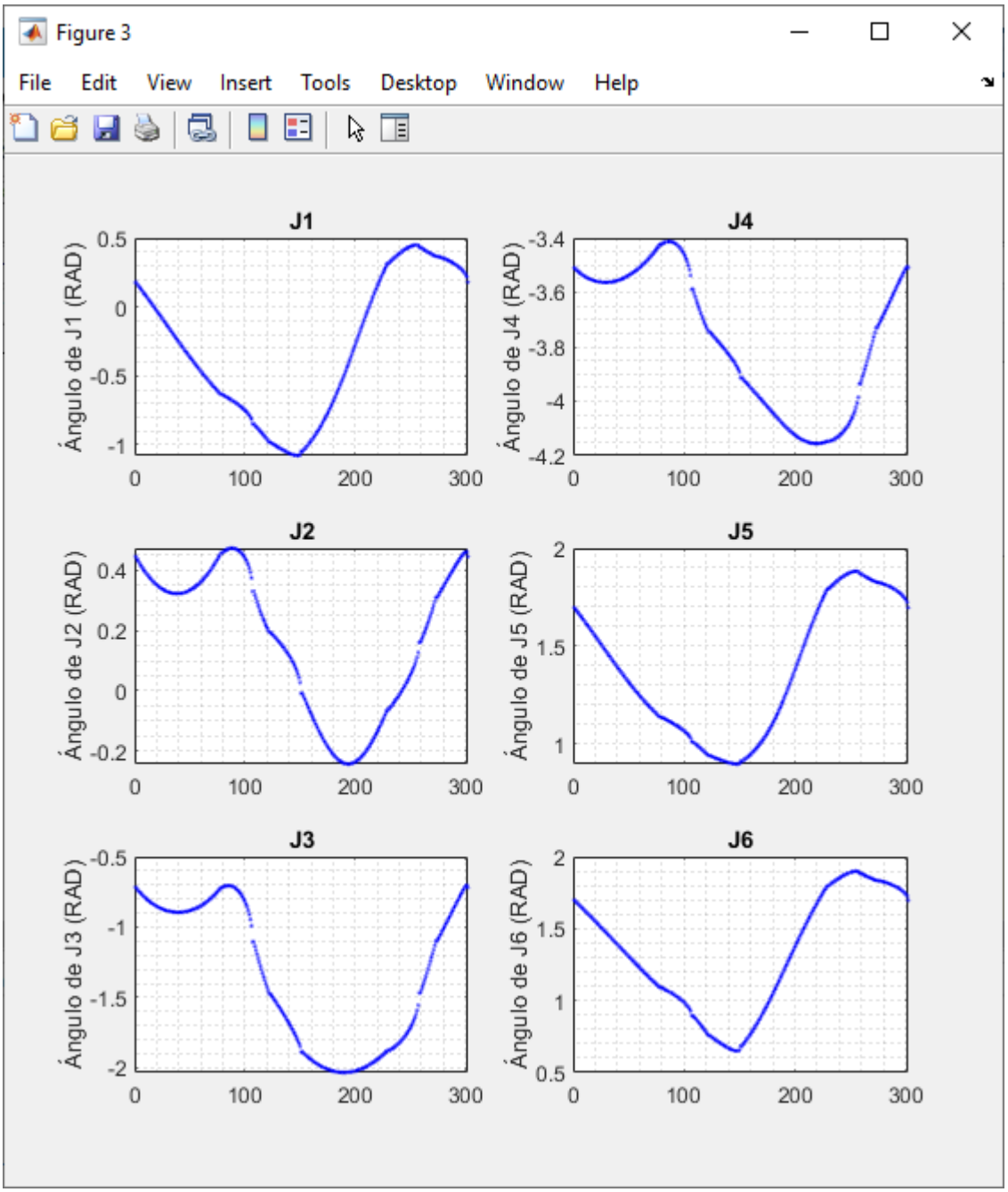
Para cada ejecución dentro de la estructura de control se realiza el mismo procedimiento que efectuamos con el `viapoint` pero ahora con una matriz denominada `qs` en la cuál se irán indexando cada una de las configuraciones obtenidas hasta lograr una matriz con la misma cantidad de filas que de puntos considerados, pero 6 columnas, donde cada una corresponde a una de las 6 articulaciones rotacionales del robot (J1, J2, J3, J4, J5, J6).

Universal Robots UR5e

```
%CINEMÁTICA INVERSA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
configPreview=homeConfiguration(robot);
qinitial = [0.2469    0.4564   -0.7207   -3.5179    1.7445    1.7471]; %Vector 'q' de
referencia para reducir multiplicidad
for i=1:6
    configPreview(i).JointPosition = configPreview(i).JointPosition + qinitial(i);
end
qs=[0 0 0 0 0 0]; %Vector 'qs' para las configuraciones de trayectoria
for i=2:size(viapoints)
    C = [viapoints(i,1) viapoints(i,2) viapoints(i,3) 0 135 180];
    TRVEC=[C(1,1:3) 1]';
    ROTM=eul2tform(deg2rad(C(1,4:6)),'XYZ');
    TCP = [ROTM(:,1:3) TRVEC];
    [configSol,solInfo] = ik('endeffector',TCP,[1 1 1 1 1 1],configPreview);
    qhome=homeConfiguration(robot);
    q=zeros(1,6);
    for j=1:6
        q(j)=configSol(j).JointPosition-qhome(j).JointPosition;
    end
    configPreview=configSol;
    qs = [qs; q];
end
qs

%GRAFICO ANGULOS DE ARTICULACIÓN
figure(3)
[m,n]=size(qs);
axis([0 m -1 1])
for i=2:m
```

```
    subplot(3,2,1)
    grid minor
    ylabel('Ángulo de J1 (RAD)')
    plot(i-1,qs(i,1),'b.','MarkerSize',3)
    hold on
    title('J1')

    subplot(3,2,3)
    grid minor
    ylabel('Ángulo de J2 (RAD)')
    plot(i-1,qs(i,2),'b.','MarkerSize',3)
    hold on
    title('J2')

    subplot(3,2,5)
    grid minor
    ylabel('Ángulo de J3 (RAD)')
    plot(i-1,qs(i,3),'b.','MarkerSize',3)
    hold on
    title('J3')

    subplot(3,2,2)
    grid minor
    ylabel('Ángulo de J4 (RAD)')
    plot(i-1,qs(i,4),'b.','MarkerSize',3)
    hold on
    title('J4')

    subplot(3,2,4)
    grid minor
    ylabel('Ángulo de J5 (RAD)')
    plot(i-1,qs(i,5),'b.','MarkerSize',3)
    hold on
    title('J5')

    subplot(3,2,6)
    grid minor
    ylabel('Ángulo de J6 (RAD)')
    plot(i-1,qs(i,6),'b.','MarkerSize',3)
    hold on
    title('J6')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Después de obtener la matriz con los vectores de configuración de cada viapoint, se realiza otra estructura para recorrer todas las filas y graficar cada uno de los puntos hasta obtener las gráficas de cada ángulo de articulación. En el gráfico obtenido (y pese a ser gráficos discretos) se evidencia la continuidad en el ángulo para cada articulación, es decir, no hay saltos bruscos ni cambios repentinos que no puedan efectuarse con facilidad por el robot, además que el ángulo en la primer postura coincide con el ángulo en la última en cada J.

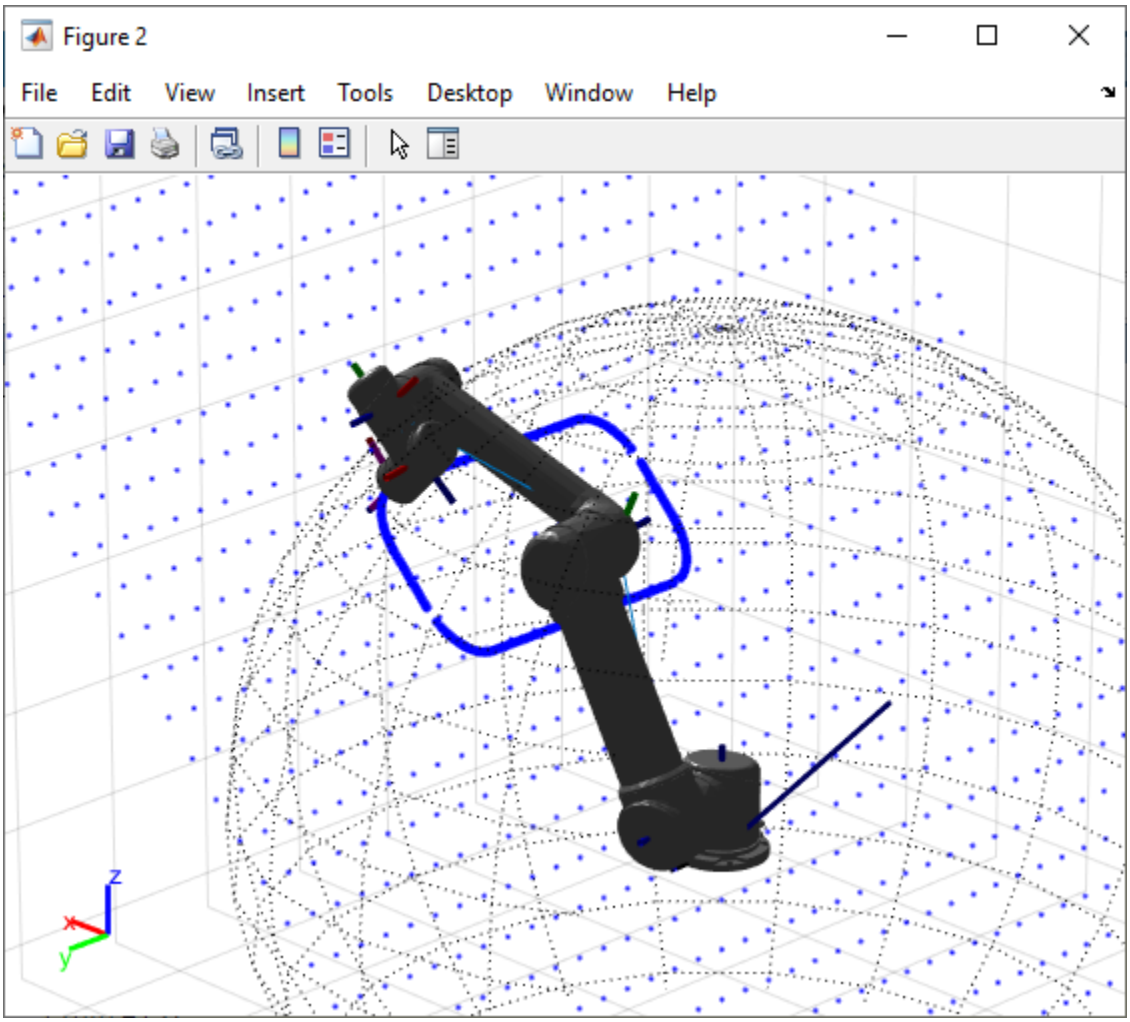


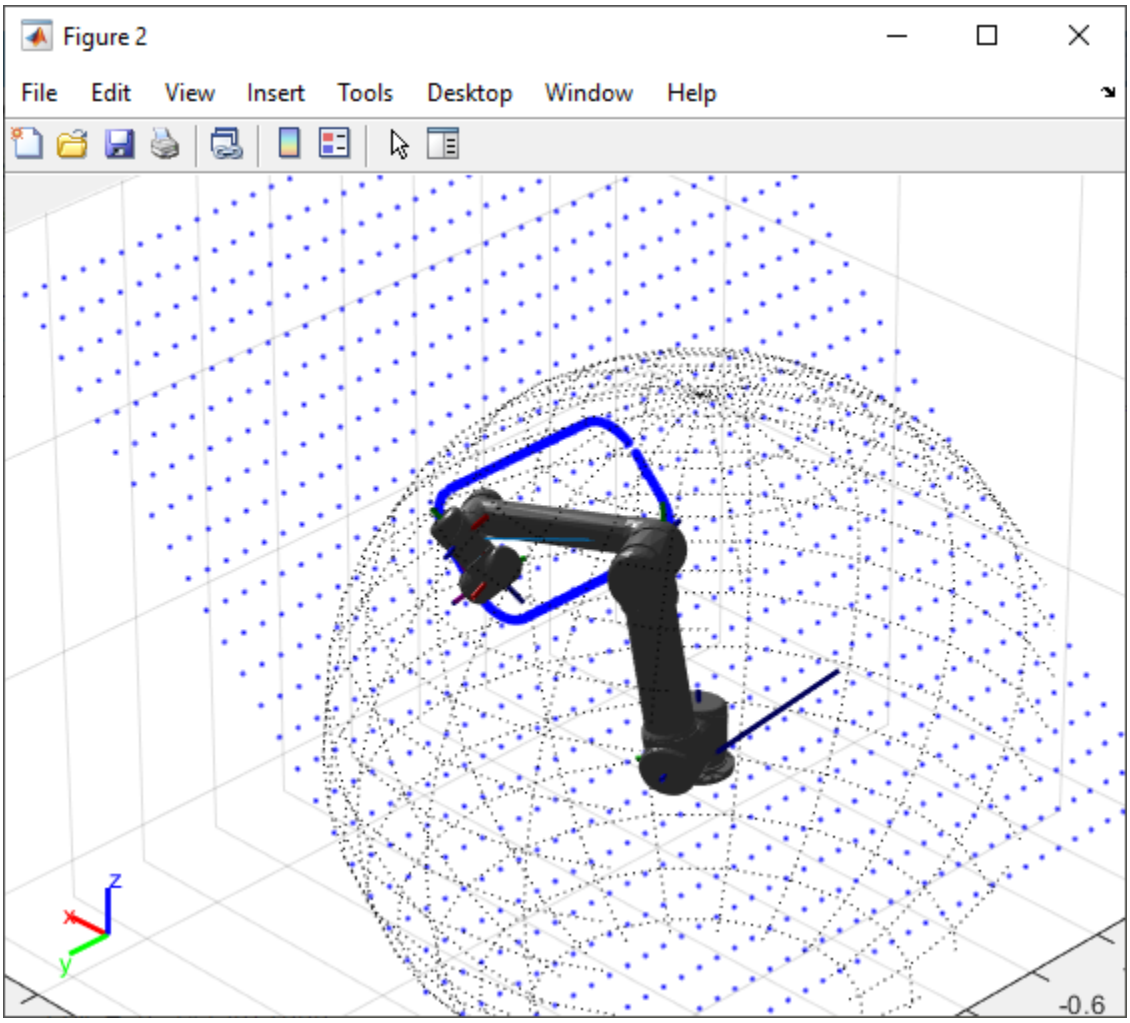
4) Calcule las velocidades en cada viapoint de manera que la herramienta recorra la ruta a una velocidad de 500 mm/s. Presente gráficas de velocidad de cada articulación.

Para graficar la animación de la trayectoria final, se realizó el siguiente bloque de código con una estructura de control para actualizar cada una a una las configuraciones de la matriz `qs` a medida que se recorren los puntos, y donde la velocidad está controlada por medio de un tiempo en `pause()` calculado a partir del paso y la velocidad dada de 500mm/s.

Universal Robots UR5e

```
%GRAFICO DE TRAYECTORIA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vel=500/1000;
tim=step/vel;
for i=2:size(qs)
    q=homeConfiguration(robot);
    config1 = qs(i,:);
    for j=1:6
        q(j).JointPosition = q(j).JointPosition + config1(j);
    end
    show(robot,q,'PreservePlot',false)
    drawnow
    pause(tim/50)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```





En las capturas se puede observar la ruta obtenida por los viapoints en el plano determinado y el recorrido que efectúa el robot. La explicación con mayor detalle de la trayectoria junto con la animación del robot recorriendo punto a punto se puede observar en el vídeo compartido junto con este informe de laboratorio.