

# Laboratorio No. 2

Diana Carolina Cruz Palacios, Daniel Fernando Díaz Coy, Michael Alexander Rodriguez Urbina

Robotica 2021-1

## Enlaces del Laboratorio

Github: [https://github.com/mirodriguezur/Laboratorio2\\_RobotUR5](https://github.com/mirodriguezur/Laboratorio2_RobotUR5)

Wiki: <http://lab2analisisur5.wikidot.com/>

Youtube: <https://youtu.be/JB11XijlGhc>

## 1 Objetivos

- 1.1 Hacer una introducción al manejo de la herramienta MATLAB® para la simulación de robots.
- 1.2 Simular el comportamiento del robot industrial (asignado a su grupo) teniendo en cuenta sus características y haciendo uso de los toolboxes RVCTools de Peter Corke y Robotics System Toolbox (RST) de MATLAB®
- 1.3 Construir los modelos cinemáticos directo, inverso y diferencial de un robot Industrial
- 1.4 Generar rutas y trayectorias y simular el seguimiento de las trayectorias. Implementar una GUI en MATLAB® para demostrar el funcionamiento de los modelos

## 2. Requisitos

Toolboxes RVCTools de Peter Corke y Robotics System Toolbox instalados.

## 3. Asignacion del Robot

Tabla 1. Asignacion de Grupos

Equipo de:	Robot	Plano - Vector
David Rivas	FANUC M-16iB 6-axis robot	[1, 0, 1]
Laura Garzón	ABB IRB 1600 6-axis robot	[0, 1, 1]
Sebastian Alfonso	FANUC LR Mate 200iB 6-axis robot	[-1, 0, 1]
Nicolas Pinilla	ABB IRB 120 6-axis robot	[0, -1, 1]
Michael Rodriguez	Universal Robots UR5 6-axis robot	[1, 0, 1]
Alejandro Hernández	Yaskawa Motoman MH5 6-axis robot	[0, 1, 1]
Natalia Tello	ABB IRB 120 6-axis robot	[-1, 0, 1]
Luis Bohórquez	ABB IRB 1600 6-axis robot	[0, -1, 1]
Luis Chávez	FANUC LR Mate 200iB 6-axis robot	[1, 0, 1]
Alexander Caicedo	FANUC M-16iB 6-axis robot	[0, 1, 1]
Sebastian Panche	Universal Robots UR5 6-axis robot	[-1, 0, 1]
Jorge Avellaneda	Yaskawa Motoman MH5 6-axis robot	[0, -1, 1]

Nuestro grupo de trabajo tiene asignado el robot **UR5** de Universal Robotics

## 4. Localización de ruta

**4.1 Plano - Vector, significa que la ruta que el robot va a simular debe estar en un plano normal al vector indicado en la tabla. en nuestro caso usaremos el vector [1,0,1]**

**4.2 La ruta debe tener una de las dos formas mostradas en la figura. Cada grupo selecciona la que deseen.**

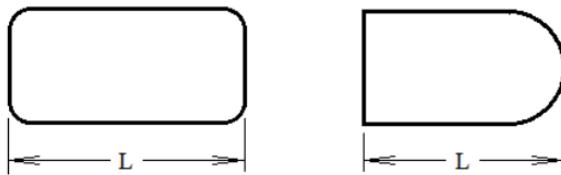


Figura 1. Opciones de selección de ruta

**4.3 La dimensión L debe ser aproximadamente 0,4 del alcance horizontal máximo del robot.**

**4.4 El grupo decide la posición de la ruta en el espacio cartesiano de manera que quede dentro del espacio diestro del robot.**

#### 4.5 Resultado

Se escogio el tipo de trayectoria de la derecha de la figura 1, la dimension elegida es  $L = 0.4m$  lo que permite ubicarla en el espacio de trabajo del robot.

Como comprobacion de los puntos escojidos hacemos una ecuacion general del plano en el espacio teniendo en cuenta el vector normal, este surge de la definicion del plano mediante la operacion

$\vec{P} \cdot \vec{N} = 0$ . El Vector P puede definirse como un vector perteneciente al plano desplazado una posicion

$(x_0, y_0, z_0)$ , de modo que  $\vec{P} = [(x - x_0), (y - y_0), (z - z_0)]$ . Haciendo el producto punto del vector

$\vec{P} \cdot \vec{N} = [(x - x_0), (y - y_0), (z - z_0)] \cdot [1, 0, 1] = 0$  entonces  $(x - x_0) + (z - z_0) = 0 \rightarrow x + z = (x_0 + z_0)$  y asi obtenemos la ecuacion del plano desplazado en el espacio que contiene nuestra trayectoria.

Con el fin de optimizar el resultado se busco que la division de puntos equidistantes (mencionado en el aparte de Integración) fuera similar tanto en las trayectorias lineales como en la curva, teniendo como resultado que la cantidad de puntos necesarios que cumplen el requisito es  $64 \cdot n$

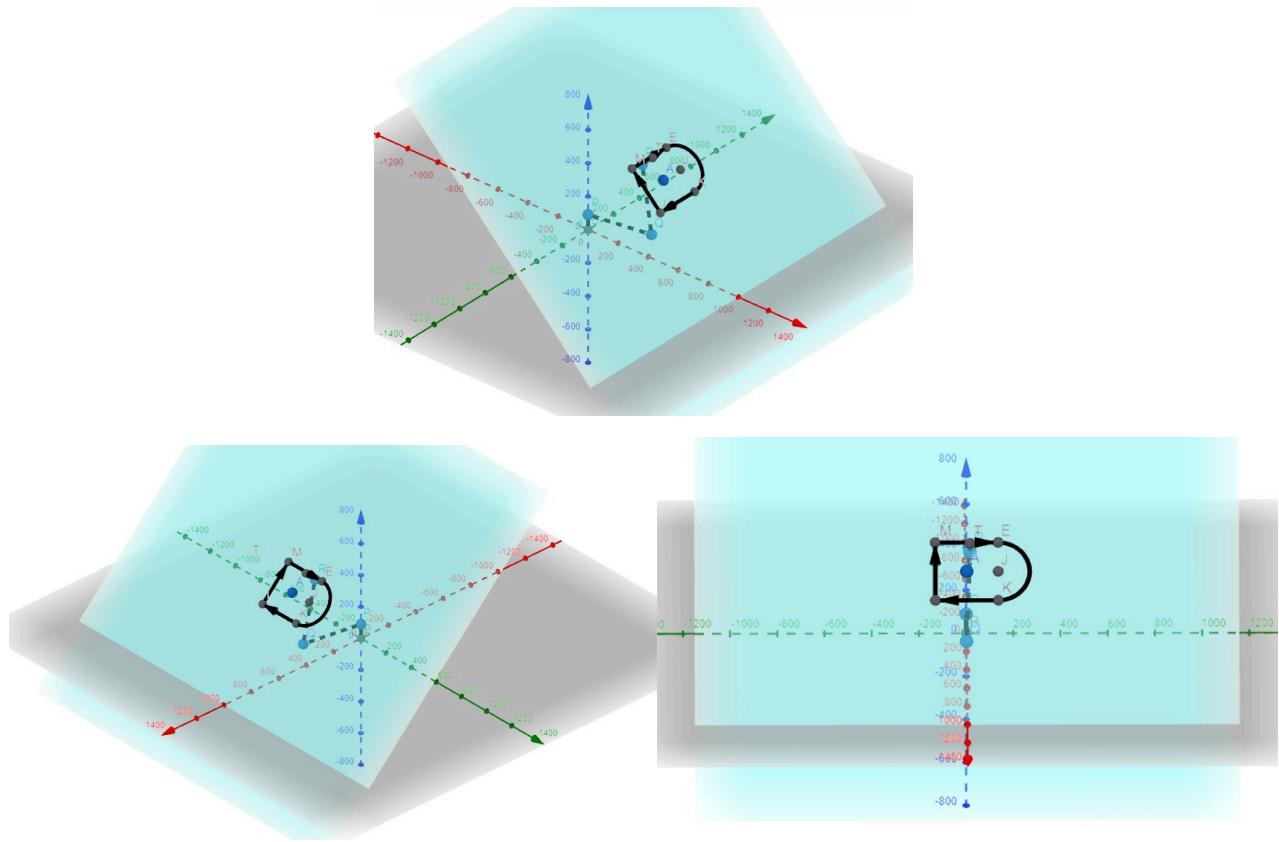


Figura 2. Ubicación de la ruta elegida

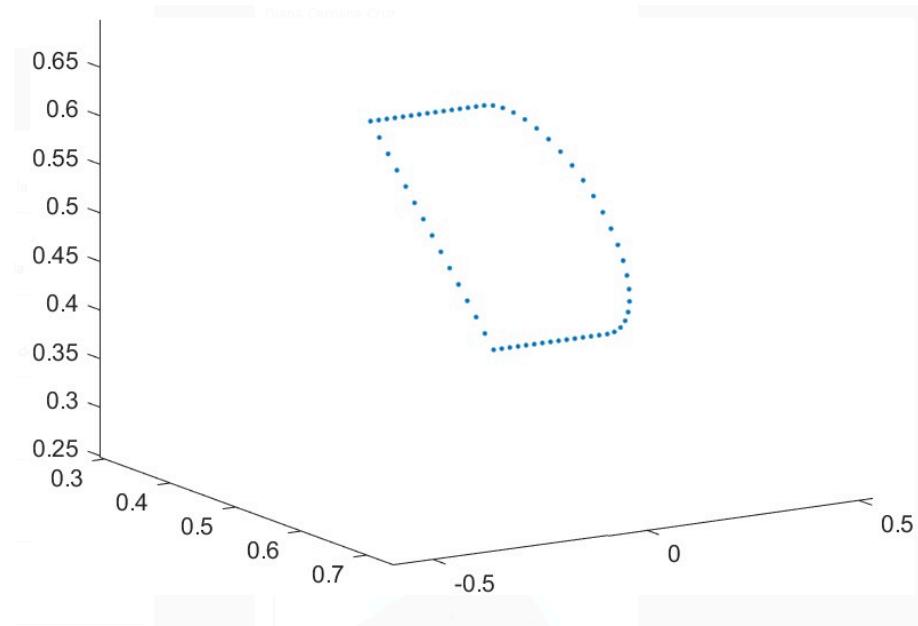


Figura3. Primera ubicación espacial de puntos

```
% parametro t
t=0.019048;

K = [0.59428, 0.13333, 0.40572];
```

```

M = [0.40572, -0.13333, 0.59428];
E = [0.40572, 0.13333, 0.59428];
I = [0.59428, -0.13333, 0.40572];

J = [0.5, 0.13333, 0.5];

%recta entre 2 puntos KI = I-K
punto=[0.59428, 0.13333, 0.40572]';
for i=1:14
    punto (:,i+1)=[K(1),(K(2)-i*t),K(3)];
end

%Recta ENTRE 2 PUNTOS ME IM=M-I
%punto(:,16)= [0.59428, -0.13333, 0.40572]';

for i=1:14
    punto (:,i+15)=[(I(1)-i*t/sqrt(2)),(I(2)),(I(3)+i*t/sqrt(2))];
end

%Recta ENTRE 2 PUNTOS ME ME=E-M
%punto(:,31)= [0.40572, -0.13333, 0.59428]';

for i=1:14
    punto (:,i+29)=[M(1),(M(2)+i*t),M(3)];
end

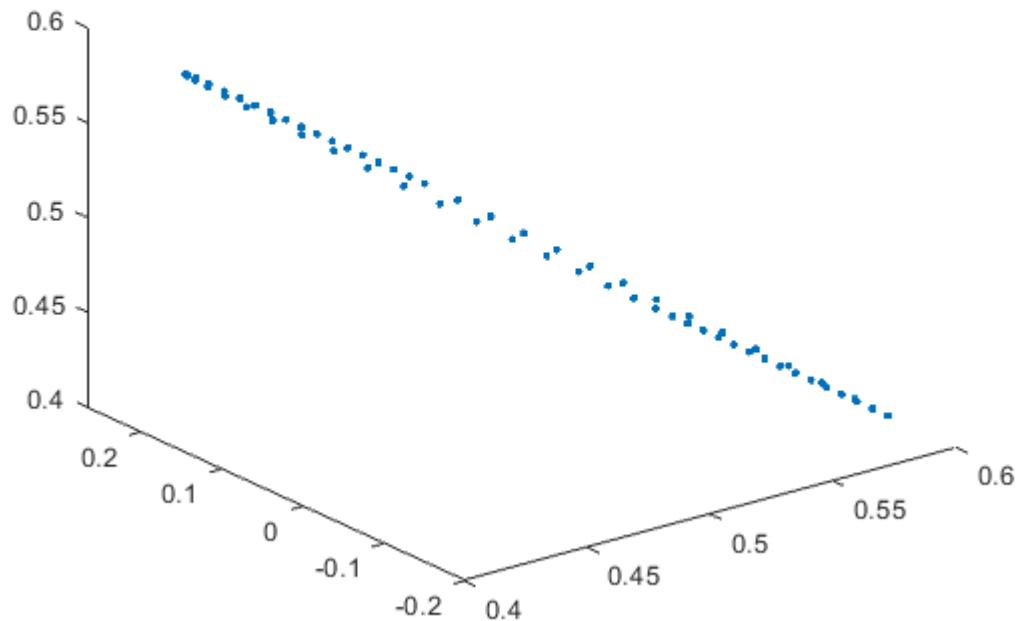
%Semicirculo centro radio JK, EMPIEZA EN E, TERMINA EN K

j=-90;
for i=1:22

    punto (:,i+43)= [J(1)+(J(1)-E(1))*sind(t+j),J(2)+J(2)*cosd(t+j), J(3)+(J(3)-E(3))*sind(t+j)]
    j=j+8.2;
end

p = plot3(punto(1,:),punto(2,:),punto(3,:),'.');


```



## 5. Características y modelo del robot

### 5.1 Imágenes del robot.

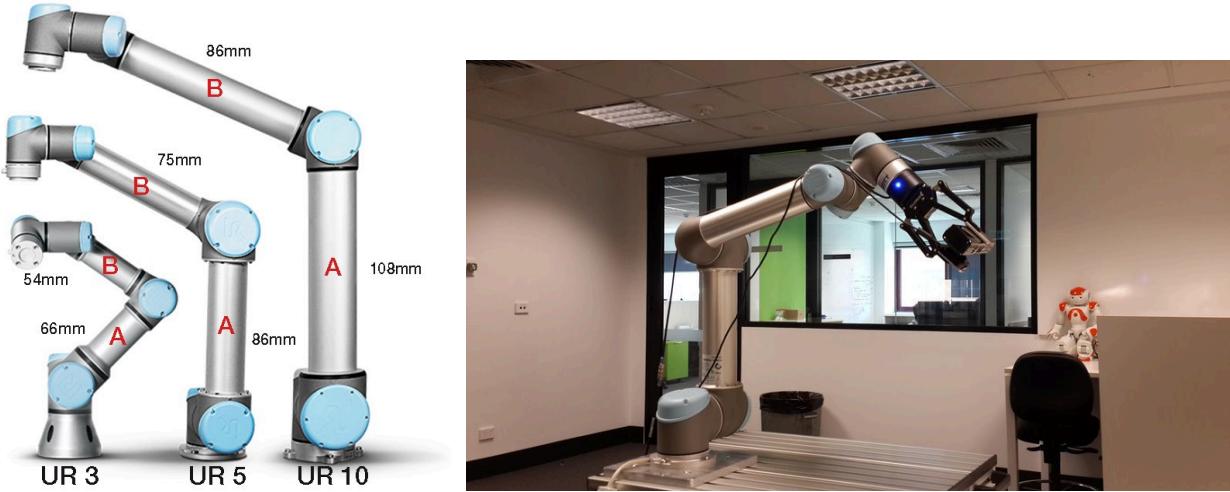


Figura 4. Ilustraciones del robot

### 5.2 Capacidad de carga.

El robot tiene una capacidad de carga de 5kg/11lb

### 5.3 Alcance vertical y horizontal.

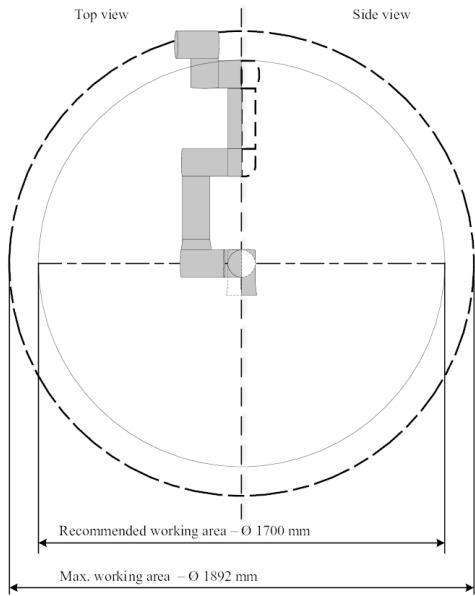


Figura 5. Extension

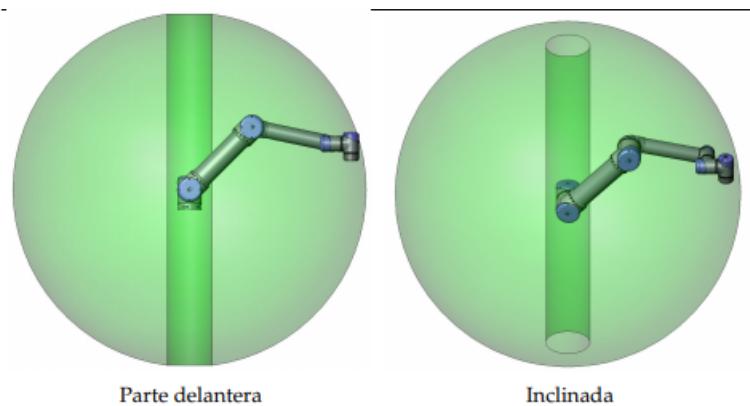
Fuente: [https://www.researchgate.net/figure/Collaborative-robot-UR5-working-area-top-side-view\\_fig1\\_347021253](https://www.researchgate.net/figure/Collaborative-robot-UR5-working-area-top-side-view_fig1_347021253)

Al ser un robot sin limitaciones aparentes en la rotación de sus articulaciones (-360° a 360°) el espacio alcanzable es una esfera

#### 5.4 Repetibilidad.

Segun la hoja técnica del robot, este tiene una repetibilidad de +/- 0.1 mm

#### 5.5 Gráfica(s) de espacio alcanzable.



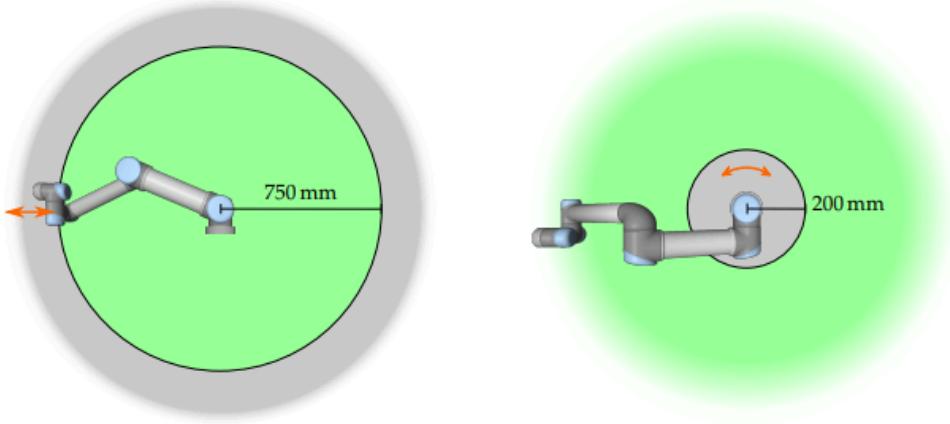


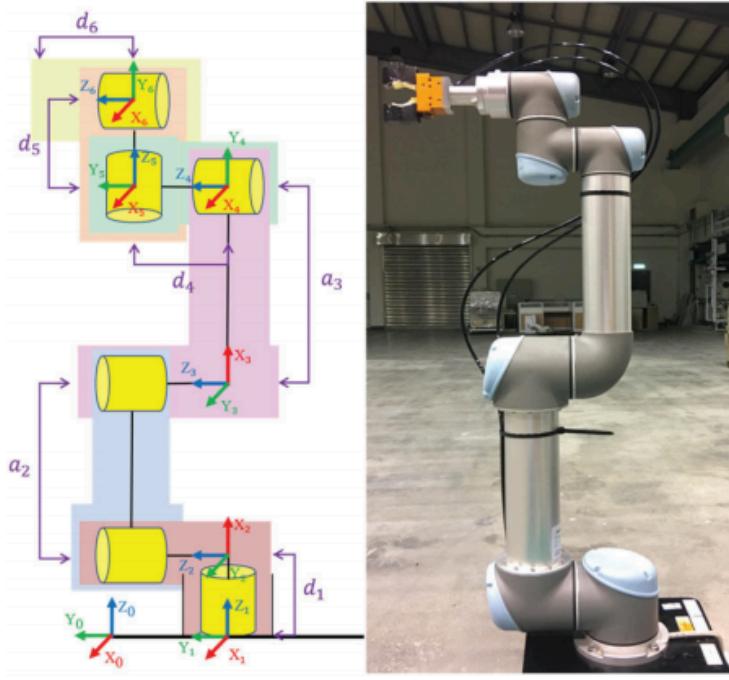
Figura 5.1: En determinadas zonas del espacio de trabajo debe prestarse atención a los peligros de enganche, debido a las propiedades físicas del brazo robótico. Se define una zona para movimientos radiales cuando la junta de la muñeca 1 está a una distancia de al menos 750 mm de la base del robot. La otra zona está a 200 mm de la base del robot, cuando el movimiento se produce en dirección tangencial.

Figura 6. Espacio Alcanzable del robot UR5

Fuente: <https://www.universal-robots.com/media/1803022/5ework.png?width=704&height=731>

## 5.6 Tabla de parámetros DH.

Aunque mas adelante obtenemos esta tabla, se pueden encontrar en varios artículos que muestran la operación de este robot como los autores obtienen la tabla de parámetros DH de acuerdo a su posición de Home, en este caso ellos colocan el robot en posición vertical



**Figure 3.** UR robot positions and local reference coordinates of each joint.

**Table I.** UR robot D-H parameters.

$i$	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	89.159	0	$\pi/2$
2	$\theta_2$	0	425	0
3	$\theta_3$	0	392.25	0
4	$\theta_4$	109.15	0	$\pi/2$
5	$\theta_5$	94.65	0	$-\pi/2$
6	$\theta_6$	82.3	0	0

Figura 7. Tabla de parametros DH obtenida en el articulo "An image vision and automatic calibration system for universal robots"

Fuente: <https://journals.sagepub.com/doi/pdf/10.1177/1461348419874925>

### 5.7 ¿Qué software utiliza el fabricante para diseño de celdas o programación?.

Usa un programa propietario llamado PolyScope, requiere de un controlador propio de la marca alimentado por 24V, la programación se realiza por bloques de código dentro de la interfaz

El robot se comunica a la interfaz PC por medio del protocolo MODBUS ([Modbus - Wikipedia, la enciclopedia libre](#))

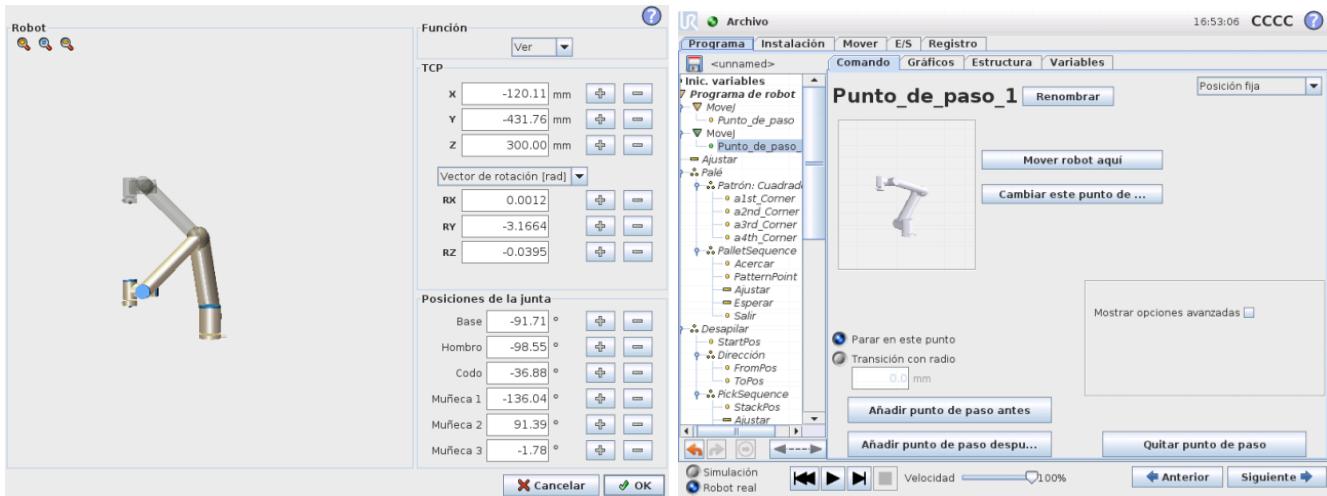


Figura 8. Ejemplos de ventanas de comandos de PolyScope

## 5.8 Enlaces de Interés

Especificaciones técnicas básicas: [https://www.universal-robots.com/media/50588/ur5\\_en.pdf](https://www.universal-robots.com/media/50588/ur5_en.pdf)

Manual de usuario: [https://www.cfzcobots.com/wp-content/uploads/2017/03/ur5\\_user\\_manual\\_es\\_global.pdf](https://www.cfzcobots.com/wp-content/uploads/2017/03/ur5_user_manual_es_global.pdf)

Ejemplo de Código de operacion: <https://www.zacobria.com/universal-robots-zacobria-forum-hints-tips-how-to-script-client-server-example/>

## 6. Iniciando el Análisis

Haciendo uso de MATLAB® y los toolboxes **construya un modelo del robot o utilice uno disponible en los toolboxes y que corresponda al robot asignado.**

Para el inicio del análisis y que el archivo mlx pueda ser ejecutado en cualquier computador , se carga inicialmente el Toolbox de peter corke RVC y se eliminan las variables que esten presentes antes de esta inicialización

```
addpath 'C:\Program Files\MATLAB\R2020b\toolbox\rvctools'
startup_rvc
```

```
Robotics, Vision & Control: (c) Peter Corke 1992-2011 http://www.petercorke.com
- Robotics Toolbox for Matlab (release 9.10)
- pHIWARE (release 1.1): pHIWARE is Copyrighted by Bryan Moutrie (2013-2021) (c)
** Release <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved here.</p>
</body></html>
now available
```

```
Run rtbdemo to explore the toolbox
- Machine Vision Toolbox for Matlab (release 3.4)
```

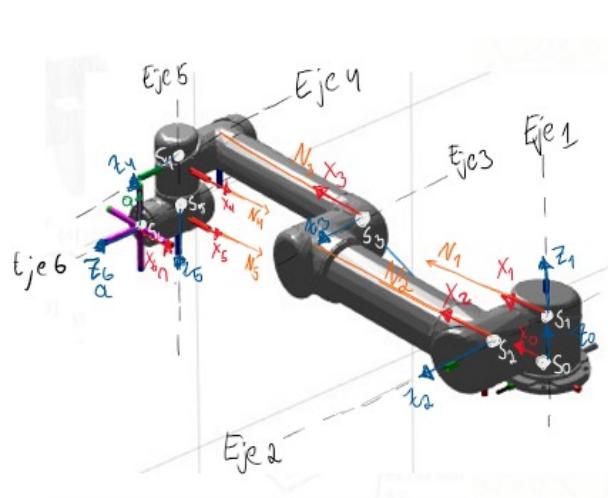
```
clc;
clear all;
```

```
close all;
```

## 6.1 Realice el análisis geométrico del robot asignado a través de DH modificado y utilice las funciones de RVC para construir un modelo en alambres.

### 6.1.1 Obtención de los parametros DH

Los datos obtenidos de las longitudes de los eslabones y su ubicacion en el espacio debido a que las articulaciones del robot no comparten un plano de trabajo debido a su construccion. Debido a esto se buscaron las dimensiones estandar del robot pero la pagina del fabricante no comparte esta informacion. Por ello se extrajeron las dimensiones del modelo URDF que usa el toolbox de matlab, esto nos permite obtener una excelente precision en la obtencion de los datos.



$i$	$\alpha_{i-1}$	$d_{i-1}$	$\theta_i$	offset
1	$\alpha_0 = \frac{x_0 z_1}{x_0}$ 0	$d_0 = \frac{y_0 z_1}{x_0}$ 0	$\theta_1 = \frac{x_0 x_1}{z_1}$ $q_1$	0
2	$\alpha_1 = \frac{z_1 z_2}{x_1}$ $-\pi/2$	$d_1 = \frac{x_1 z_2}{z_1}$ 0	$\theta_2 = \frac{x_1 x_2}{z_2}$ $q_2$	0
3	$\alpha_2 = \frac{z_2 z_3}{x_2}$ 0	$d_2 = \frac{x_2 z_3}{z_2}$ $L_3$	$\theta_3 = \frac{x_2 x_3}{z_3}$ $q_3$	0
4	$\alpha_3 = \frac{z_3 z_4}{x_3}$ 0	$d_3 = \frac{x_3 z_4}{z_3}$ $L_5$	$\theta_4 = \frac{x_3 x_4}{z_4}$ $q_4$	$-\pi$
5	$\alpha_4 = \frac{z_4 z_5}{x_4}$ $\pi/2$	$d_4 = \frac{x_4 z_5}{z_4}$ 0	$\theta_5 = \frac{x_4 x_5}{z_5}$ $q_5$	0
6	$\alpha_5 = \frac{z_5 z_6}{x_5}$ $-\pi/2$	$d_5 = \frac{x_5 z_6}{z_5}$ 0	$\theta_6 = \frac{x_5 x_6}{z_6}$ $q_6$	0

Figura 9. Obtención manual de parametros DHmod del robot UR5

Se establecio que la posición de home usada para nuestro laboratorio sera la misma posición de home del toolbox de Matlab RST para facilitar las comprobaciones.

### 6.1.2 Construccion del robot serial usando RVC

Con los parametros obtenidos anteriormente se genera el robot

```
% Li indica las longitudes extraidas del modelo URDF del robot, todas ellas
% son usadas en el análisis
L1 = 0.089159; L2 = 0.13585; L3 = 0.425; L4 = 0.1197;
L5 = 0.39225; L6 = 0.093; L7= 0.09465; L8= 0.0823;

L9 = 0; %L9 Es la distancia del Tool, pero para comparar resultados facilmente se deja como cero

L(1) = Link('revolute','alpha',0,'a',0,'d',L1,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha',-pi/2,'a',0,'d',L2,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha',0,'a',L3,'d', -L4,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha',0,'a',L5,'d',L6,'offset',-pi,'modified','qlim',[-2*pi 2*pi]);
L(5) = Link('revolute','alpha',pi/2,'a',0,'d',L7,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(6) = Link('revolute','alpha',-pi/2,'a',0,'d',L8,'offset',0,'modified','qlim',[-2*pi 2*pi]);

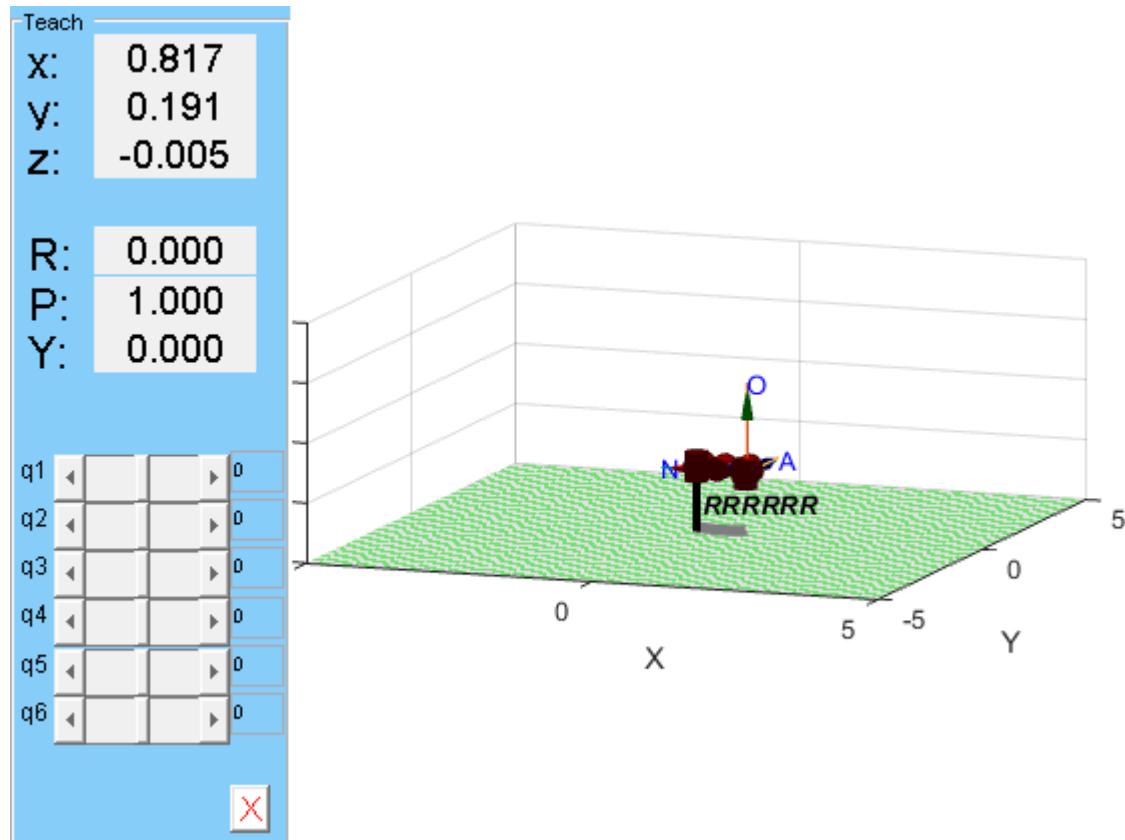
UR5_RVC = SerialLink(L, 'name', 'RRRRRR')
```

```
UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)
```

j	theta	d	a	alpha	offset
1	q1	0.08916	0	0	0
2	q2	0.1358	0	-1.571	0
3	q3	-0.1197	0.425	0	0
4	q4	0.093	0.3922	0	-3.142
5	q5	0.09465	0	1.571	0
6	q6	0.0823	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 0
                  0 0 0 1     0 0 0 1
```

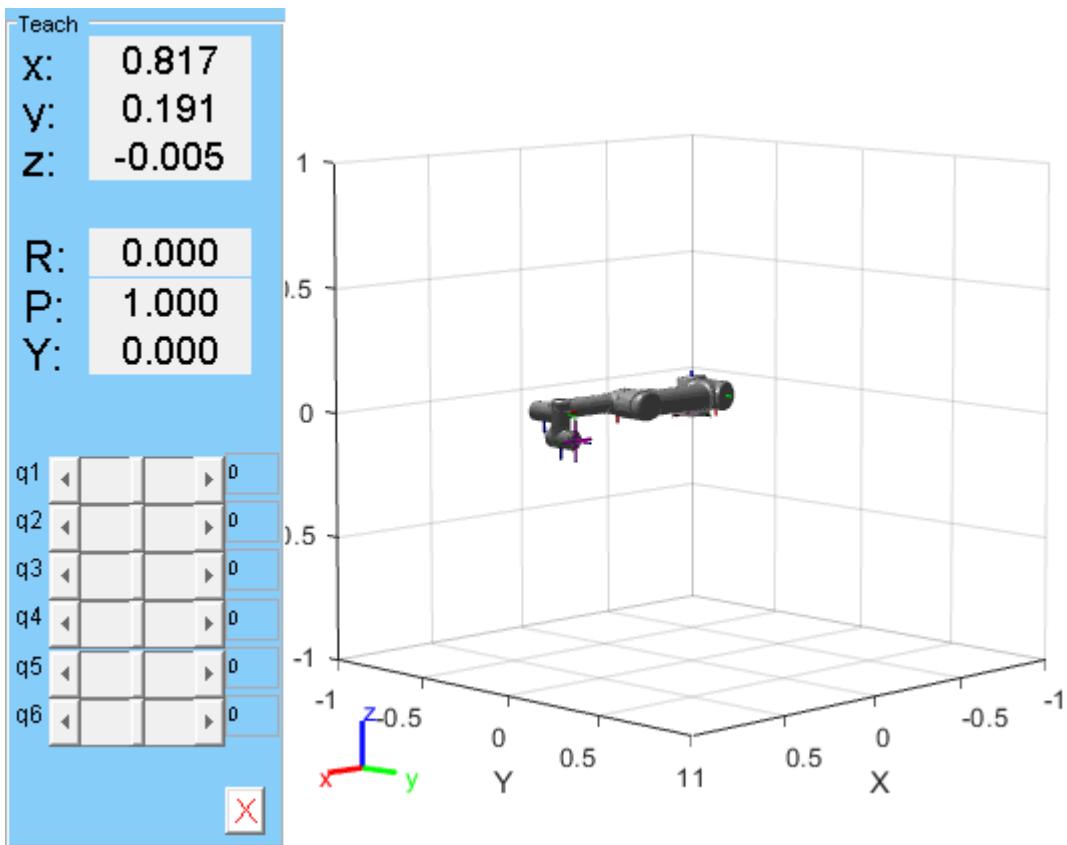
```
UR5_RVC.plot([0 0 0 0 0 0], 'workspace', [-5 5 -5 5 -1 3], 'noa', 'view', [20 10]);
UR5_RVC.teach;
```



## 6.2 Considerando el robot asignado, construya el modelo del robot utilizando RST.

Debido a que el toolbox de Matlab posee el robot UR5 en el toolbox como un robot de ejemplo, se utiliza el robot directamente sin necesidad de construirlo

```
UR5_RST = loadrobot("universalUR5");
show(UR5_RST);
```



### 6.3 Compare los dos métodos.

El metodo del toolbox RVC depende de la configuracion que el usuario defina en su robot como home, y los parametros deben ingresarse manualmente sin importar el tipo de robot ya que el toolbox no posee robots preestablecidos. Por otro lado el toolbox de Matlab, RST, como posee el robot en sus archivos, y este robot tiene una configuracion de home definida desde el toolbox es mas simple generar el modelo de robot y operar con el. Ademas de esto, el toolbox RST al usar modelos 3D cargados desde una interfaz de ROS es visualmente mas realista que el modelo de alambres que se genera con el toolbox RVC

De manera similar, las funciones y metodos de ambos toolboxes son distintas y se operan de maneras diferentes.

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

El fabricante en su manual de usuario y en su manual de servicio tecnico no provee puntos de calibracion. Existe una normativa ISO (ISO 9283) que provee un metodo de calibracion para robots industriales. Fuente: [ISO9283 Performance Testing - RoboDK Documentation](#)

## 7 Modelo geométrico directo

### 7.1 Halle el modelo geométrico directo de su robot asignado usando MTH.

A continuación se halla el modelo geométrico directo de forma simbólica extrayendo las matrices de transformación homogénea del toolbox de Peter Corke.

```
% Se borran las variables anteriores y se cierran las instancias de
% funciones creadas
clc; clear all; close all;

% Se crea de manera simbolica las variables q1 y L1 que indican ángulo de
% articulacion y longitudes respectivamente
syms q1 q2 q3 q4 q5 q6 L1 L2 L3 L4 L5 L6 L7 L8 L9

% Se definen los eslabones del robot serial
L(1) = Link('revolute','alpha',0,'a',0,'d',L1,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha',sym(-pi/2),'a',0,'d',L2,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha',0,'a',L3,'d', -L4,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha',0,'a',L5,'d',L6,'offset',sym(-pi),'modified','qlim',[-2*pi 2*pi]);
L(5) = Link('revolute','alpha',sym(pi/2),'a',0,'d',L7,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(6) = Link('revolute','alpha',sym(-pi/2),'a',0,'d',L8,'offset',0,'modified','qlim',[-2*pi 2*pi]);

% Se crea el elemento tipo "SerialLink" del toolbox RVC
UR5_RVC = SerialLink(L, 'name', 'RRRRRR')
```

UR5\_RVC =  
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

j	theta	d	a	alpha	offset
1	q1	L1	0	0	0
2	q2	L2	0	-pi/2	0
3	q3	-L4	L3	0	0
4	q4	L6	L5	0	-pi
5	q5	L7	0	pi/2	0
6	q6	L8	0	-pi/2	0

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

% Se define la MTH de la herramienta  
UR5\_RVC.tool = [ 1 0 0 0; 0 1 0 0; 0 0 1 L9; 0 0 0 1]

UR5\_RVC =  
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

j	theta	d	a	alpha	offset
1	q1	L1	0	0	0
2	q2	L2	0	-pi/2	0
3	q3	-L4	L3	0	0
4	q4	L6	L5	0	-pi
5	q5	L7	0	pi/2	0
6	q6	L8	0	-pi/2	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   L9
          0   0   0   1      0   0   0   1

```

%Calculo de T\_0\_1  
 $T_{0\_1} = L(1).A(q1)$

$$T_{0\_1} = \begin{pmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

%Calculo de T\_1\_2  
 $T_{1\_2} = L(2).A(q2)$

$$T_{1\_2} = \begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ -\sin(q_2) & -\cos(q_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

%Calculo de T\_2\_3  
 $T_{2\_3} = L(3).A(q3)$

$$T_{2\_3} = \begin{pmatrix} \cos(q_3) & -\sin(q_3) & 0 & L_3 \\ \sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & -L_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

%Calculo de T\_3\_4  
 $T_{3\_4} = L(4).A(q4)$

$$T_{3\_4} = \begin{pmatrix} -\cos(q_4) & \sin(q_4) & 0 & L_5 \\ -\sin(q_4) & -\cos(q_4) & 0 & 0 \\ 0 & 0 & 1 & L_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

%Calculo de T\_4\_5  
 $T_{4\_5} = L(5).A(q5)$

T\_4\_5 =

$$\begin{pmatrix} \cos(q_5) & -\sin(q_5) & 0 & 0 \\ 0 & 0 & -1 & -L_7 \\ \sin(q_5) & \cos(q_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

%Calculo de T\_5\_6

T\_5\_6 = L(6).A(q6)

T\_5\_6 =

$$\begin{pmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0 \\ 0 & 0 & 1 & L_8 \\ -\sin(q_6) & -\cos(q_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

%Calculo de MTH general de manera simbólica

T\_0\_Tool = simplify(T\_0\_1 \* T\_1\_2 \* T\_2\_3 \* T\_3\_4 \* T\_4\_5 \* T\_5\_6 \* UR5\_RVC.tool)

T\_0\_Tool =

$$\begin{pmatrix} \sigma_1 \cos(q_1) \sin(q_6) - \cos(q_6) \sigma_4 & \sin(q_6) \sigma_4 + \sigma_1 \cos(q_1) \cos(q_6) & \sigma_5 \cos(q_1) \sin(q_5) - \cos(q_5) \sin(q_1) & L_4 \\ \cos(q_6) \sigma_3 + \sigma_1 \sin(q_1) \sin(q_6) & \sigma_1 \cos(q_6) \sin(q_1) - \sin(q_6) \sigma_3 & & \sigma_2 \\ \sigma_5 \sin(q_6) + \sigma_1 \cos(q_5) \cos(q_6) & \sigma_5 \cos(q_6) - \sigma_1 \cos(q_5) \sin(q_6) & & -\sigma_1 \sin(q_5) \\ 0 & 0 & & 0 \end{pmatrix}$$

where

$$\sigma_1 = \sin(q_2 + q_3 + q_4)$$

$$\sigma_2 = \cos(q_1) \cos(q_5) + \sigma_5 \sin(q_1) \sin(q_5)$$

$$\sigma_3 = \cos(q_1) \sin(q_5) - \sigma_5 \cos(q_5) \sin(q_1)$$

$$\sigma_4 = \sin(q_1) \sin(q_5) + \sigma_5 \cos(q_1) \cos(q_5)$$

$$\sigma_5 = \cos(q_2 + q_3 + q_4)$$

Tambien se halla la cinemática directa de forma manual planteando directamente la siguiente matriz:

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

% Matrices de transformación homogénea

$$\begin{aligned} T_{0\_1m} = & [\cos(q_1) \ -\sin(q_1) \ 0 \ 0; \\ & \sin(q_1)*\cos(\theta) \ \cos(q_1)*\cos(\theta) \ -\sin(\theta) \ -\sin(\theta)*L1; \\ & \sin(q_1)*\sin(\theta) \ \cos(q_1)*\sin(\theta) \ \cos(\theta) \ \cos(\theta)*L1; \\ & 0 \ 0 \ 0 \ 1] \end{aligned}$$

$T_{0\_1m} =$

$$\begin{pmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$T_{1\_2m} = [\cos(q_2) \ -\sin(q_2) \ 0 \ 0;$

$$\begin{aligned} & \sin(q_2)*\cos(sym(-pi/2)) \ \cos(q_2)*\cos(sym(-pi/2)) \ -\sin(sym(-pi/2)) \ -\sin(sym(-pi/2))*L2; \\ & \sin(q_2)*\sin(sym(-pi/2)) \ \cos(q_2)*\sin(sym(-pi/2)) \ \cos(sym(-pi/2)) \ \cos(sym(-pi/2))*L2; \\ & 0 \ 0 \ 0 \ 1] \end{aligned}$$

$T_{1\_2m} =$

$$\begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ -\sin(q_2) & -\cos(q_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$T_{2\_3m} = [\cos(q_3) \ -\sin(q_3) \ 0 \ L3;$

$$\begin{aligned} & \sin(q_3)*\cos(\theta) \ \cos(q_3)*\cos(\theta) \ -\sin(\theta) \ -\sin(\theta)*-L4; \\ & \sin(q_3)*\sin(\theta) \ \cos(q_3)*\sin(\theta) \ \cos(\theta) \ \cos(\theta)*-L4; \\ & 0 \ 0 \ 0 \ 1] \end{aligned}$$

$T_{2\_3m} =$

$$\begin{pmatrix} \cos(q_3) & -\sin(q_3) & 0 & L_3 \\ \sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & -L_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$T_{3\_4m} = [\cos(q4+sym(-pi)) \ -\sin(q4+sym(-pi)) \ 0 \ L5;$

$$\begin{aligned} & \sin(q4+sym(-pi))*\cos(\theta) \ \cos(q4+sym(-pi))*\cos(\theta) \ -\sin(\theta) \ -\sin(\theta)*L6; \\ & \sin(q4+sym(-pi))*\sin(\theta) \ \cos(q4+sym(-pi))*\sin(\theta) \ \cos(\theta) \ \cos(\theta)*L6; \\ & 0 \ 0 \ 0 \ 1] \end{aligned}$$

```
T_3_4m =
```

$$\begin{pmatrix} -\cos(q_4) & \sin(q_4) & 0 & L_5 \\ -\sin(q_4) & -\cos(q_4) & 0 & 0 \\ 0 & 0 & 1 & L_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
T_4_5m = [cos(q5) -sin(q5) 0 0;
```

$$\begin{aligned} & \sin(q5)*\cos(\text{sym}(\pi/2)) \cos(q5)*\cos(\text{sym}(\pi/2)) -\sin(\text{sym}(\pi/2)) -\sin(\text{sym}(\pi/2))*L7; \\ & \sin(q5)*\sin(\text{sym}(\pi/2)) \cos(q5)*\sin(\text{sym}(\pi/2)) \cos(\text{sym}(\pi/2)) \cos(\text{sym}(\pi/2))*L7; \\ & 0 0 0 1] \end{aligned}$$

```
T_4_5m =
```

$$\begin{pmatrix} \cos(q5) & -\sin(q5) & 0 & 0 \\ 0 & 0 & -1 & -L7 \\ \sin(q5) & \cos(q5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
T_5_6m = [cos(q6) -sin(q6) 0 0;
```

$$\begin{aligned} & \sin(q6)*\cos(\text{sym}(-\pi/2)) \cos(q6)*\cos(\text{sym}(-\pi/2)) -\sin(\text{sym}(-\pi/2)) -\sin(\text{sym}(-\pi/2))*L8; \\ & \sin(q6)*\sin(\text{sym}(-\pi/2)) \cos(q6)*\sin(\text{sym}(-\pi/2)) \cos(\text{sym}(-\pi/2)) \cos(\text{sym}(-\pi/2))*L8; \\ & 0 0 0 1] \end{aligned}$$

```
T_5_6m =
```

$$\begin{pmatrix} \cos(q6) & -\sin(q6) & 0 & 0 \\ 0 & 0 & 1 & L8 \\ -\sin(q6) & -\cos(q6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
T_6_toolm = [1 0 0 0;
```

$$0 1 0 0;$$

$$0 0 1 L9;$$

$$0 0 0 1]$$

```
T_6_toolm =
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L9 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
%MTH total del robot UR5
```

```
T_0_toolm = simplify(T_0_1m*T_1_2m*T_2_3m*T_3_4m*T_4_5m*T_5_6m*T_6_toolm)
```

```
T_0_toolm =
```

$$\begin{pmatrix} \sigma_1 \cos(q_1) \sin(q_6) - \cos(q_6) \sigma_4 & \sin(q_6) \sigma_4 + \sigma_1 \cos(q_1) \cos(q_6) & \sigma_5 \cos(q_1) \sin(q_5) - \cos(q_5) \sin(q_1) & L_4 \\ \cos(q_6) \sigma_3 + \sigma_1 \sin(q_1) \sin(q_6) & \sigma_1 \cos(q_6) \sin(q_1) - \sin(q_6) \sigma_3 & \sigma_2 \\ \sigma_5 \sin(q_6) + \sigma_1 \cos(q_5) \cos(q_6) & \sigma_5 \cos(q_6) - \sigma_1 \cos(q_5) \sin(q_6) & -\sigma_1 \sin(q_5) \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = \sin(q_2 + q_3 + q_4)$$

$$\sigma_2 = \cos(q_1) \cos(q_5) + \sigma_5 \sin(q_1) \sin(q_5)$$

$$\sigma_3 = \cos(q_1) \sin(q_5) - \sigma_5 \cos(q_5) \sin(q_1)$$

$$\sigma_4 = \sin(q_1) \sin(q_5) + \sigma_5 \cos(q_1) \cos(q_5)$$

$$\sigma_5 = \cos(q_2 + q_3 + q_4)$$

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

```
L1 = 0.089159; L2 = 0.13585; L3 = 0.425; L4 = 0.1197;
L5 = 0.39225; L6 = 0.093; L7= 0.09465; L8= 0.0823;
L9 = 0; %L9 Es la distancia del Tool

L(1) = Link('revolute','alpha',0,'a',0,'d',L1,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha',-pi/2,'a',0,'d',L2,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha',0,'a',L3,'d', -L4,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha',0,'a',L5,'d',L6,'offset',-pi,'modified','qlim',[-2*pi 2*pi]);
L(5) = Link('revolute','alpha',pi/2,'a',0,'d',L7,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(6) = Link('revolute','alpha',-pi/2,'a',0,'d',L8,'offset',0,'modified','qlim',[-2*pi 2*pi]);

UR5_RVC = SerialLink(L, 'name', 'RRRRRR')
```

```
UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)
```

j	theta	d	a	alpha	offset
1	q1	0.08916	0	0	0
2	q2	0.1358	0	-1.571	0
3	q3	-0.1197	0.425	0	0
4	q4	0.093	0.3922	0	-3.142
5	q5	0.09465	0	1.571	0
6	q6	0.0823	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  0
                  0  0  0  1           0  0  0  1
```

```
UR5_RVC.tool = [ 1 0 0 0; 0 1 0 0; 0 0 1 L9; 0 0 0 1]
```

```
UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)
```

j	theta	d	a	alpha	offset
1	q1	0.08916	0	0	0
2	q2	0.1358	0	-1.571	0
3	q3	-0.1197	0.425	0	0
4	q4	0.093	0.3922	0	-3.142
5	q5	0.09465	0	1.571	0
6	q6	0.0823	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  0
                  0  0  0  1           0  0  0  1
```

Una vez creado el robot se procede a realizar los calculos de cinematica directa con los valores siguientes:

q
[0.5, 0.2, 0.4, 0.5, 0, 1.5]
[-π/2, 0.3, 0, π/2, 0.4, 1.2]
[0, 1, -0.5 2 1 0.5]
[-1, -0.3, -π/5, 0.4 0.2 1]

Tabla 2. Valores Articulares del Laboratorio

### % Caso 1

```
q=[0.5 0.2 0.4 0.5 0 1.5]
```

```
q = 1x6
 0.5000    0.2000    0.4000    0.5000      0    1.5000
```

```
T_0_1 = L(1).A(q(1)); T_1_2 = L(2).A(q(2)); T_2_3 = L(3).A(q(3));
T_3_4 = L(4).A(q(4)); T_4_5 = L(5).A(q(5)); T_5_6 = L(6).A(q(6));
```

```
T_0_Tool = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6 * UR5_RVC.tool
```

```
T_0_Tool = 4x4
0.7520    0.4524   -0.4794    0.4838
0.4108    0.2471   0.8776    0.4825
0.5155   -0.8569   0.0000   -0.2597
0          0         0       1.0000
```

```
% Se extrae el vector de posicion obtenido
vPosicion=T_0_Tool(1:3,4)
```

```
vPosicion = 3x1
0.4838
0.4825
-0.2597
```

```
% Se Extrae la matriz de rotacion de la matriz obtenida de Transformacion
mRotacion = T_0_Tool(1:3,1:3)
```

```
mRotacion = 3x3
0.7520    0.4524   -0.4794
0.4108    0.2471   0.8776
0.5155   -0.8569   0.0000
```

```
% se obtienen los ángulos roll, pitch, yaw segun la definicion original
roll = atan2(-mRotacion(2,3),mRotacion(3,3))*(180/pi)
```

```
roll = -90
```

```
sr = sin(degtorad(roll));
cr = cos(degtorad(roll));
pitch= atan2(mRotacion(1,3),cr*mRotacion(3,3)-sr*mRotacion(2,3))*(180/pi)
```

```
pitch = -28.6479
```

```
yaw = atan2(-mRotacion(1,2),mRotacion(1,1))*(180/pi)
```

```
yaw = -31.0310
```

```
% Se genera el vector fila de coordenadas generalizadas
cGeneral1 = [vPosicion', roll, pitch, yaw]
```

```
cGeneral1 = 1x6
0.4838    0.4825   -0.2597   -90.0000   -28.6479   -31.0310
```

```
% Caso 2
```

```
q=[-pi/2 0.3 0 pi/2 0.4 1.2]
```

```
q = 1x6
-1.5708    0.3000        0    1.5708    0.4000    1.2000
```

```
T_0_1 = L(1).A(q(1)); T_1_2 = L(2).A(q(2)); T_2_3 = L(3).A(q(3));
T_3_4 = L(4).A(q(4)); T_4_5 = L(5).A(q(5)); T_5_6 = L(6).A(q(6));
```

```
T_0_Tool = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6 * UR5_RVC.tool
```

```
T_0_Tool = 4x4
0.1411   -0.3630    0.9211    0.1850
-0.9890   -0.0925    0.1151   -0.6809
  0.0434   -0.9272   -0.3720   -0.1550
      0         0         0     1.0000
```

```
% Se extrae el vector de posicion obtenido
vPosicion=T_0_Tool(1:3,4)
```

```
vPosicion = 3x1
  0.1850
  -0.6809
  -0.1550
```

```
% Se Extrae la matriz de rotacion de la matriz obtenida de Transformacion
mRotacion = T_0_Tool(1:3,1:3)
```

```
mRotacion = 3x3
  0.1411   -0.3630    0.9211
  -0.9890   -0.0925    0.1151
   0.0434   -0.9272   -0.3720
```

```
% se obtienen los ángulos roll, pitch, yaw segun la definicion original
roll = atan2(-mRotacion(2,3),mRotacion(3,3))*(180/pi)
```

```
roll = -162.8113
```

```
sr = sin(degtorad(roll));
cr = cos(degtorad(roll));
pitch= atan2(mRotacion(1,3),cr*mRotacion(3,3)-sr*mRotacion(2,3))*(180/pi)
```

```
pitch = 67.0817
```

```
yaw = atan2(-mRotacion(1,2),mRotacion(1,1))*(180/pi)
```

```
yaw = 68.7549
```

```
% Se genera el vector fila de coordenadas generalizadas
cGeneral2 = [vPosicion', roll, pitch, yaw]
```

```
cGeneral2 = 1x6
  0.1850   -0.6809   -0.1550  -162.8113   67.0817   68.7549
```

```
% Caso 3
```

```
q=[0 1 -0.5 2 1 0.5]
```

```
q = 1x6
  0    1.0000   -0.5000    2.0000    1.0000    0.5000
```

```
T_0_1 = L(1).A(q(1)); T_1_2 = L(2).A(q(2)); T_2_3 = L(3).A(q(3));
T_3_4 = L(4).A(q(4)); T_4_5 = L(5).A(q(5)); T_5_6 = L(6).A(q(6));
```

```
T_0_Tool = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6 * UR5_RVC.tool
```

```
T_0_Tool = 4x4
0.6668    0.3177   -0.6741    0.4617
0.7385   -0.4034    0.5403    0.1536
-0.1003   -0.8581   -0.5036   -0.4221
0           0           0       1.0000
```

```
% Se extrae el vector de posicion obtenido
vPosicion=T_0_Tool(1:3,4)
```

```
vPosicion = 3x1
0.4617
0.1536
-0.4221
```

```
% Se Extrae la matriz de rotacion de la matriz obtenida de Transformacion
mRotacion = T_0_Tool(1:3,1:3)
```

```
mRotacion = 3x3
0.6668    0.3177   -0.6741
0.7385   -0.4034    0.5403
-0.1003   -0.8581   -0.5036
```

```
% se obtienen los ángulos roll, pitch, yaw segun la definicion original
roll = atan2(-mRotacion(2,3),mRotacion(3,3))*(180/pi)
```

```
roll = -132.9862
```

```
sr = sin(degtorad(roll));
cr = cos(degtorad(roll));
pitch= atan2(mRotacion(1,3),cr*mRotacion(3,3)-sr*mRotacion(2,3))*(180/pi)
```

```
pitch = -42.3873
```

```
yaw = atan2(-mRotacion(1,2),mRotacion(1,1))*(180/pi)
```

```
yaw = -25.4748
```

```
% Se genera el vector fila de coordenadas generalizadas
cGeneral3 = [vPosicion', roll, pitch, yaw]
```

```
cGeneral3 = 1x6
0.4617    0.1536   -0.4221 -132.9862  -42.3873  -25.4748
```

```
% Caso 4
```

```
q=[-1 -0.3 -pi/5 0.4 0.2 1]
```

```
q = 1x6
-1.0000   -0.3000   -0.6283    0.4000    0.2000    1.0000
```

```
T_0_1 = L(1).A(q(1)); T_1_2 = L(2).A(q(2)); T_2_3 = L(3).A(q(3));
T_3_4 = L(4).A(q(4)); T_4_5 = L(5).A(q(5)); T_5_6 = L(6).A(q(6));
```

```
T_0_Tool = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6 * UR5_RVC.tool
```

```
T_0_Tool = 4x4
-0.3860    0.0970    0.9174    0.5395
 0.7998   -0.4605    0.3852   -0.4889
 0.4598    0.8824    0.1001    0.4553
      0         0         0     1.0000
```

```
% Se extrae el vector de posicion obtenido
vPosicion=T_0_Tool(1:3,4)
```

```
vPosicion = 3x1
 0.5395
-0.4889
 0.4553
```

```
% Se Extrae la matriz de rotacion de la matriz obtenida de Transformacion
mRotacion = T_0_Tool(1:3,1:3)
```

```
mRotacion = 3x3
-0.3860    0.0970    0.9174
 0.7998   -0.4605    0.3852
 0.4598    0.8824    0.1001
```

```
% se obtienen los ángulos roll, pitch, yaw segun la definicion original
roll = atan2(-mRotacion(2,3),mRotacion(3,3))*(180/pi)
```

```
roll = -75.4249
```

```
sr = sin(degtorad(roll));
cr = cos(degtorad(roll));
pitch= atan2(mRotacion(1,3),cr*mRotacion(3,3)-sr*mRotacion(2,3))*(180/pi)
```

```
pitch = 66.5494
```

```
yaw = atan2(-mRotacion(1,2),mRotacion(1,1))*(180/pi)
```

```
yaw = -165.8915
```

```
% Se genera el vector fila de coordenadas generalizadas
cGeneral4 = [vPosicion', roll, pitch, yaw]
```

```
cGeneral4 = 1x6
 0.5395   -0.4889    0.4553   -75.4249   66.5494  -165.8915
```

### 7.3 Haga uso de las funciones de cinemática directa de ambos toolboxes y compruebe los resultados anteriores.

Haciendo uso de los valores definidos en la tabla 2 se obtienen los resultados de cinematica directa

```
%% Haciendo uso del Toolbox de Peter Corke
```

```
% Caso 1
```

```
q = [0.5 0.2 0.4 0.5 0 1.5];
```

```
T_RVC1 = UR5_RVC.fkine(q) % Cinemática directa del robot en el caso 1
```

```
T_RVC1 = 4x4
0.7520  0.4524 -0.4794  0.4838
0.4108  0.2471  0.8776  0.4825
0.5155 -0.8569  0.0000 -0.2597
0         0         0       1.0000
```

```
gen_RVC1=[transl(T_RVC1)', tr2rpy(T_RVC1, 'deg')] % Coordenadas generalizadas de Posicion y orientación
```

```
gen_RVC1 = 1x6
0.4838  0.4825 -0.2597 -90.0000 -28.6479 -31.0310
```

## % Caso 2

```
q = [-pi/2 0.3 0 pi/2 0.4 1.2];
```

```
T_RVC2 = UR5_RVC.fkine(q) % Cinemática directa del robot en el caso 2
```

```
T_RVC2 = 4x4
0.1411 -0.3630  0.9211  0.1850
-0.9890 -0.0925  0.1151 -0.6809
0.0434 -0.9272 -0.3720 -0.1550
0         0         0       1.0000
```

```
gen_RVC2=[transl(T_RVC2)', tr2rpy(T_RVC2, 'deg')] % Coordenadas generalizadas de Posicion y orientación
```

```
gen_RVC2 = 1x6
0.1850 -0.6809 -0.1550 -162.8113  67.0817  68.7549
```

## % Caso 3

```
q = [0 1 -0.5 2 1 0.5];
```

```
T_RVC3 = UR5_RVC.fkine(q) % Cinemática directa del robot en el caso 3
```

```
T_RVC3 = 4x4
0.6668  0.3177 -0.6741  0.4617
0.7385 -0.4034  0.5403  0.1536
-0.1003 -0.8581 -0.5036 -0.4221
0         0         0       1.0000
```

```
gen_RVC3=[transl(T_RVC3)', tr2rpy(T_RVC3, 'deg')] % Coordenadas generalizadas de Posicion y orientación
```

```
gen_RVC3 = 1x6
0.4617  0.1536 -0.4221 -132.9862 -42.3873 -25.4748
```

## % Caso 4

```
q = [-1 -0.3 -pi/5 0.4 0.2 1];
```

```
T_RVC4 = UR5_RVC.fkine(q) % Cinemática directa del robot en el caso 4
```

```
T_RVC4 = 4x4
-0.3860  0.0970  0.9174  0.5395
0.7998 -0.4605  0.3852 -0.4889
0.4598  0.8824  0.1001  0.4553
```

```
0          0          0    1.0000
```

```
gen_RVC4=[transl(T_RVC4)', tr2rpy(T_RVC4, 'deg')] % Coordenadas generalizadas de Posicion y orientacion
```

```
gen_RVC4 = 1x6
0.5395   -0.4889    0.4553   -75.4249   66.5494  -165.8915
```

```
% Usando el Toolbox de Matlab
```

```
% Se crea el robot usando RST
UR5_RST = loadrobot("universalUR5","Gravity", [0 0 -9.81]);
% Para facilitar el uso de este robot, se define como dato de entrada un
% vector fila que posee las configuraciones de las articulaciones
UR5_RST.DataFormat = 'row';
```

```
% Caso 1
```

```
q = [0.5 0.2 0.4 0.5 0 1.5];
T_RST1=getTransform(UR5_RST,q, 'tool0')
```

```
T_RST1 = 4x4
0.7520   0.4524   -0.4794   0.4838
0.4108   0.2471   0.8776   0.4825
0.5155   -0.8569   0.0000   -0.2597
0          0          0    1.0000
```

```
gen_RST1=[transl(T_RST1)', tr2rpy(T_RST1, 'deg')]
```

```
gen_RST1 = 1x6
0.4838   0.4825   -0.2597   -90.0000   -28.6479   -31.0310
```

```
% Caso 2
```

```
q = [-pi/2 0.3 0 pi/2 0.4 1.2];
T_RST2=getTransform(UR5_RST,q, 'tool0')
```

```
T_RST2 = 4x4
0.1411   -0.3630   0.9211   0.1850
-0.9890   -0.0925   0.1151   -0.6809
0.0434   -0.9272   -0.3720   -0.1550
0          0          0    1.0000
```

```
gen_RST2=[transl(T_RST2)', tr2rpy(T_RST2, 'deg')]
```

```
gen_RST2 = 1x6
0.1850   -0.6809   -0.1550   -162.8113   67.0817   68.7549
```

```
% Caso 3
```

```
q = [0 1 -0.5 2 1 0.5];
T_RST3=getTransform(UR5_RST,q, 'tool0')
```

```
T_RST3 = 4x4
0.6668   0.3177   -0.6741   0.4617
0.7385   -0.4034   0.5403   0.1536
-0.1003   -0.8581   -0.5036   -0.4221
```

```
0      0      0    1.0000
```

```
gen_RST3=[transl(T_RST3)', tr2rpy(T_RST3, 'deg')]
```

```
gen_RST3 = 1x6
0.4617    0.1536   -0.4221  -132.9862  -42.3873  -25.4748
```

#### % Caso 4

```
q = [-1 -0.3 -pi/5 0.4 0.2 1];
T_RST4=getTransform(UR5_RST,q, 'tool0')
```

```
T_RST4 = 4x4
-0.3860    0.0970    0.9174    0.5395
0.7998   -0.4605    0.3852   -0.4889
0.4598    0.8824    0.1001    0.4553
0         0         0       1.0000
```

```
gen_RST4=[transl(T_RST4)', tr2rpy(T_RST4, 'deg')]
```

```
gen_RST4 = 1x6
0.5395   -0.4889    0.4553   -75.4249   66.5494  -165.8915
```

Los resultados se muestran en las siguientes tablas:

Usando MTH calculados.							
	q	x	y	z	roll	pitch	yaw
1	[0.5, 0.2, 0.4, 0.5, 0, 1.5]	0,4838	0,4825	-0,2597	-90	-28,65	-31
2	[-pi/2, 0.3, 0, pi/2, 0.4, 1.2]	0,185	-0,681	-0,155	-162,8	67,1	68,75
3	[0, 1, -0.5, 2, 1, 0.5]	0,4617	0,1536	-0,4222	-132,98	-42,39	-25,48
4	[-1, -0.3, -pi/5, 0.4, 0.2, 1]	0,5395	-0,489	0,455	-75,425	66,55	-165,9

Tabla 3. Valores en coordenadas generalizadas del efecto final usando MTH

	Usando Toolbox de Peter Corke (RVC)						
	q	x	y	z	roll	pitch	yaw
1	[0.5, 0.2, 0.4, 0.5, 0, 1.5]	0,4838	0,4825	-0,2597	-90	-28,65	-31
2	[-pi/2, 0.3, 0, pi/2, 0.4, 1.2]	0,185	-0,681	-0,155	-162,8	67,1	68,75
3	[0, 1, -0.5, 2, 1, 0.5]	0,4617	0,1536	-0,4222	-132,98	-42,39	-25,48
4	[-1, -0.3, -pi/5, 0.4, 0.2, 1]	0,5395	-0,489	0,455	-75,425	66,55	-165,9

Tabla 4. Valores en coordenadas generalizadas del efecto final usando RVC

Usando Toolbox de Matlab (RST)							
	<b>q</b>	<b>x</b>	<b>y</b>	<b>z</b>	<b>roll</b>	<b>pitch</b>	<b>yaw</b>
1	[0.5, 0.2, 0.4, 0.5, 0, 1.5]	0,4838	0,4825	-0,2597	-90	-28,65	-31
2	[-pi/2, 0.3, 0, pi/2, 0.4, 1.2]	0,185	-0,681	-0,155	-162,8	67,1	68,75
3	[0, 1, -0.5, 2, 1, 0.5]	0,4617	0,1536	-0,4222	-132,98	-42,39	-25,48
4	[-1, -0.3, -pi/5, 0.4, 0.2, 1]	0,5395	-0,489	0,455	-75,425	66,55	-165,9

Tabla 5. Valores en coordenadas generalizadas del efecto final usando RST

#### 7.4 Compare los métodos

Se Observa que la realizacion manual implica el conocimiento de la aplicación de la matriz general de transformacion, y las ecuaciones para la obtencion de la posicion y los angulos de orientacion.

En el caso del Toolbox RVC para calcular la cinemática directa se usa la funcion **fkine()** y en el toolbox de Matlab, RST, se utiliza la funcion **getTransform()**.

En los 3 casos, se da el mismo resultado.

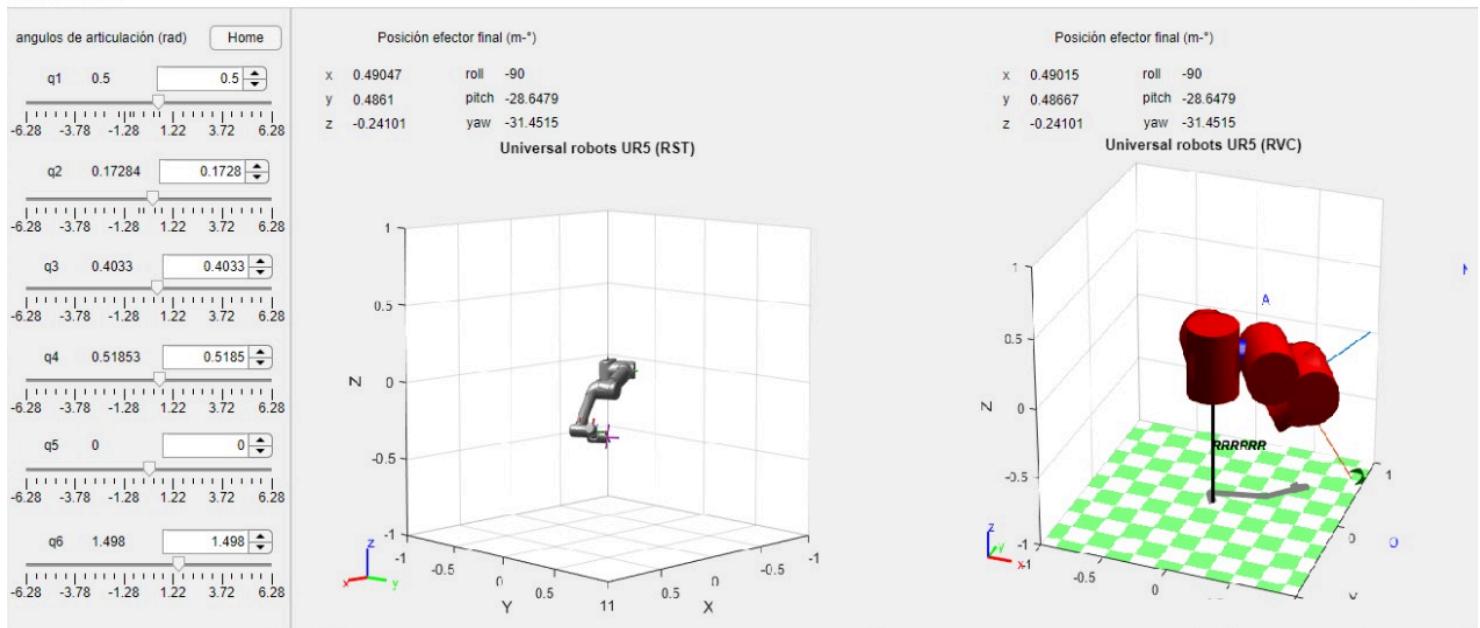
#### 7.5 Elija uno de los métodos anteriormente usados y desarrolle una GUI que permita mover cada articulación mediante controles tipo Slider, visualizar el robot y la posición del efecto final.

Para la GUI, se usaron los dos métodos para comparar los resultados entre ambos.

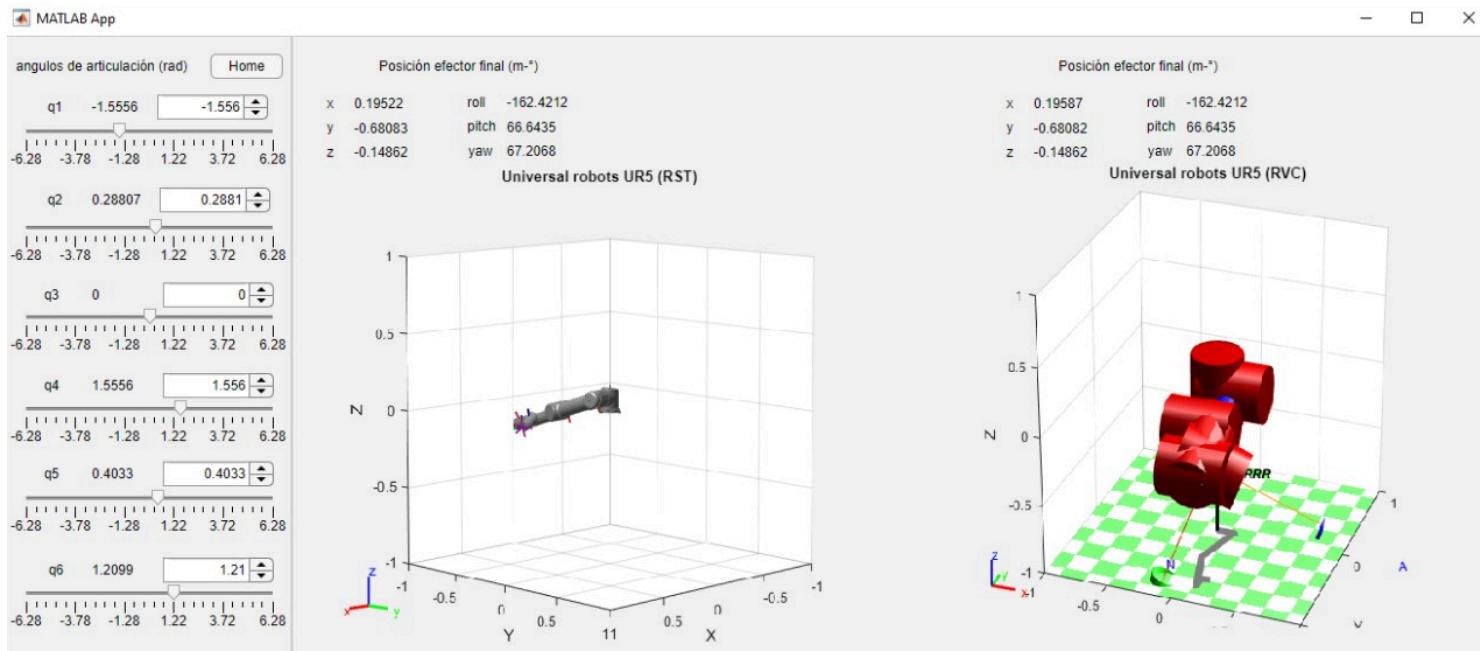
#### 7.6 En el informe incluya capturas de pantalla verificando las posiciones EF y las articulaciones.

Caso 1 :

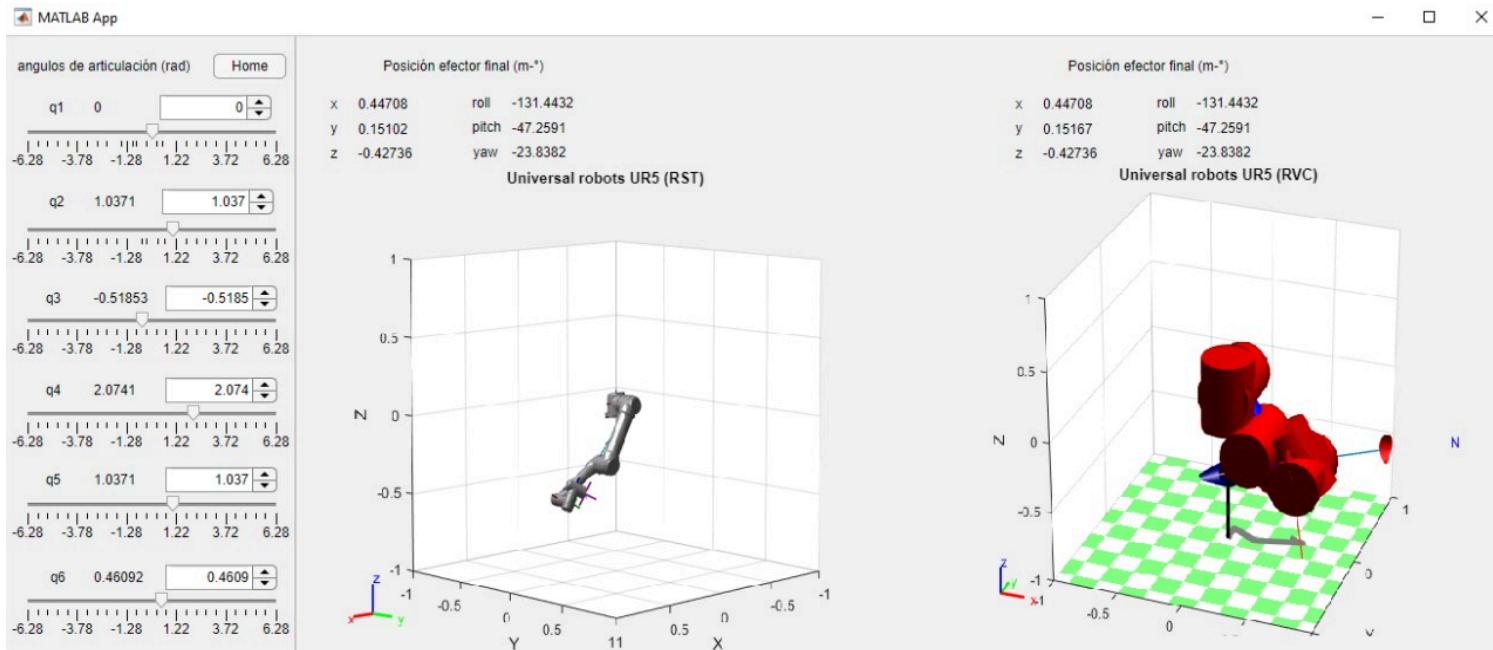
MATLAB App



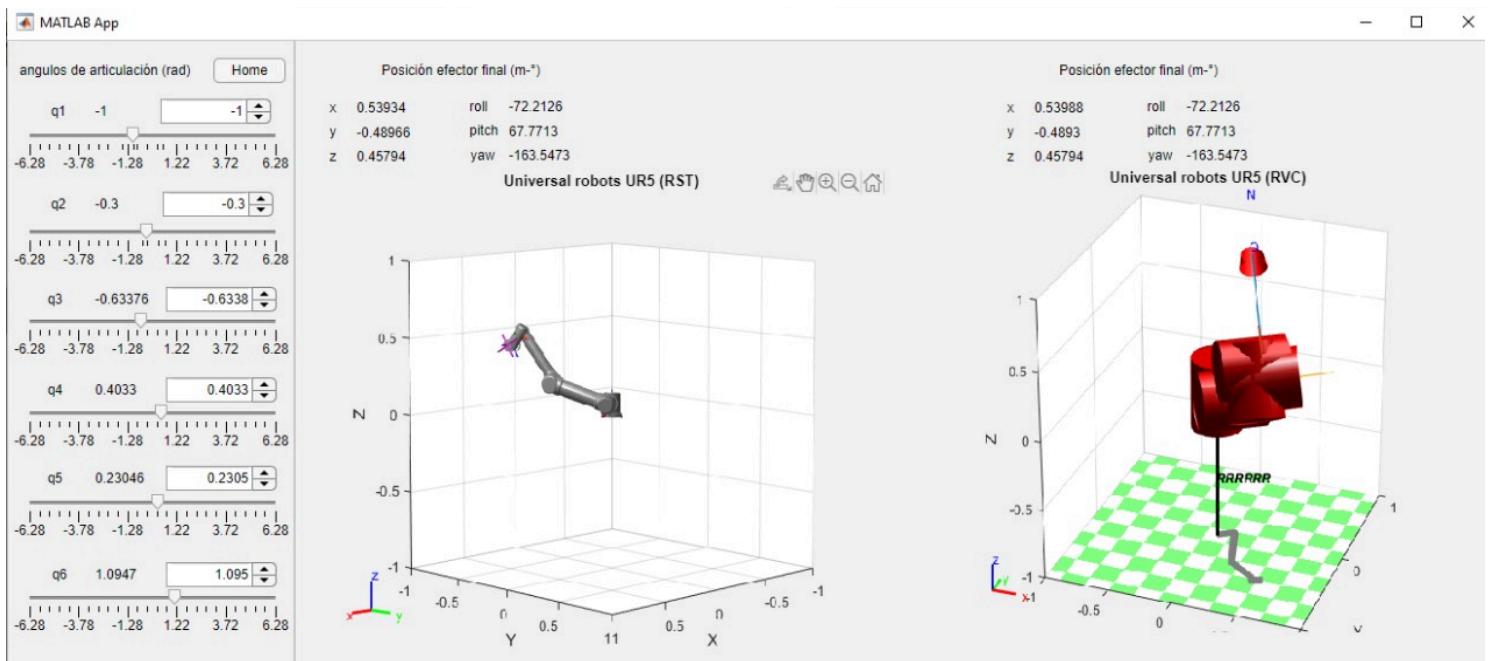
## Caso 2:



## Case 3:



#### Caso 4:



## 8 Modelo Geométrico inverso

### 8.1 Determine el modelo geométrico inverso del robot asignado haciendo uso de la metodología explicada en clase

Para el desarrollo de esta solución inversa se tiene como base la transformación  ${}^0T_6 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \cdot {}^5T_6$  sin tener en cuenta la MTH de la herramienta, ya que no se ha definido ninguna en especial y facilita la comparación con los toolboxes RVC y RST. Además de esto se restringen los ángulos posibles al rango  $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \in [0, 2\pi]$ .

Debido a la configuración del robót donde las articulaciones principales q1, q2, q3 no comparten un mismo plano se crea un "plano virtual" que emularia la posición de las articulaciones 1, 2 y 3 en un único plano, haciendo que se redefina la distancia 4 como se muestra en la figura:

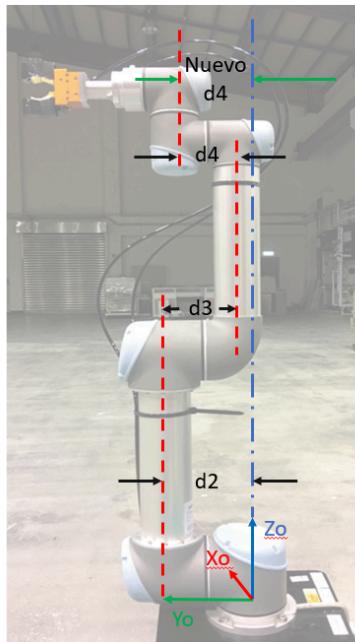


Figura n. Redefinicion de la distancia  $d_4$

Pero este cambio implica directamente reposicionar los marcos de referencia de manera distinta a como se tiene en el DH usado en la cinemática directa a ubicarlas sobre este "plano virtual", esto facilita mucho los calculos ya que hace un modelo mas simple del robot aunque el resultado no varie.

Para ello se reposiciona el robot en un modelo de alambres y se busca que al menos un eje de las tramas apunte en una dirección comun, por ello se realiza entonces el siguiente re-modelado del robot:

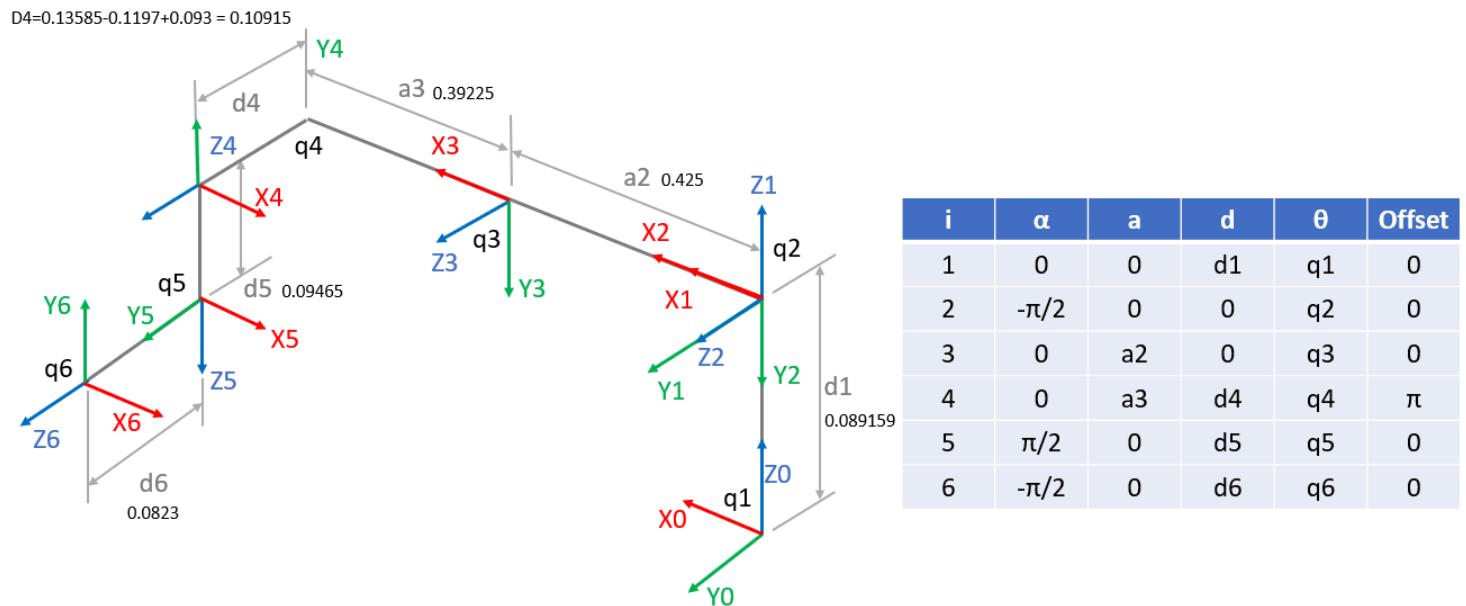


Figura n+1. Reorganización de parámetros DHmod

Una vez hecho esto y teniendo en cuenta que todos los  $X_i$  apuntan en la misma dirección se reconstruye el robot usando el toolbox RVC

```
% Dimensiones físicas
L1 = 0.089159; L2 = 0.13585; L3 = 0.425; L4 = 0.1197;
L5 = 0.39225; L6 = 0.093; L7= 0.09465; L8= 0.0823;
L9 = 0; %L9 Es la distancia del Tool

% Ubicación de las dimensiones físicas
d1=L1; d4=L2-L4+L6; d5=L7; d6=L8; a2=L3; a3=L5;

L(1) = Link('revolute','alpha',0,'a',0,'d',d1,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha',-pi/2,'a',0,'d',0,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha',0,'a',a2,'d', 0,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha',0,'a',a3,'d',d4,'offset',pi,'modified','qlim',[-2*pi 2*pi]);
L(5) = Link('revolute','alpha',pi/2,'a',0,'d',d5,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(6) = Link('revolute','alpha',-pi/2,'a',0,'d',d6,'offset',0,'modified','qlim',[-2*pi 2*pi]);

UR5_RVC = SerialLink(L, 'name', 'RRRRRR')
```

```
UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

+-----+-----+-----+-----+-----+
| j | theta | d | a | alpha | offset |
+-----+-----+-----+-----+-----+
| 1| q1| 0.08916| 0| 0| 0|
| 2| q2| 0| 0| -1.571| 0|
| 3| q3| 0| 0.425| 0| 0|
| 4| q4| 0.1091| 0.3922| 0| 3.142|
| 5| q5| 0.09465| 0| 1.571| 0|
| 6| q6| 0.0823| 0| -1.571| 0|
+-----+-----+-----+-----+-----+
grav = 0 base = 1 0 0 0 tool = 1 0 0 0
      0 0 1 0 0 1 0 0 1
      9.81 0 0 1 0 0 1 0 0 1
```

```
%UR5_RVC.tool = [ 0 -1 0 0; 1 0 0 0; 0 0 1 L9; 0 0 0 1]
UR5_RVC.tool = [ 1 0 0 0; 0 1 0 0; 0 0 1 L9; 0 0 0 1]
```

```
UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

+-----+-----+-----+-----+-----+
| j | theta | d | a | alpha | offset |
+-----+-----+-----+-----+-----+
| 1| q1| 0.08916| 0| 0| 0|
| 2| q2| 0| 0| -1.571| 0|
| 3| q3| 0| 0.425| 0| 0|
| 4| q4| 0.1091| 0.3922| 0| 3.142|
```

```

| 5|      q5|    0.09465|      0|    1.571|      0|
| 6|      q6|    0.0823|      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  0
                  0  0  0  1        0  0  0  1

```

```
% Se revisa que las MTH correspondan en ambos modelos
```

```
MTH_home=UR5_RVC.fkine([0 0 0 0 0 0])
```

```
MTH_home = 4x4
-1.0000  -0.0000      0    0.8173
 0.0000  -0.0000    1.0000  0.1915
-0.0000    1.0000    0.0000 -0.0055
 0        0        0    1.0000
```

```
% Para la solucion numérica en coordenadas generalizadas [x y z r p y]
```

```
Gprueba=[0.42 0.35 0.57 15 30 45]
```

```
Gprueba = 1x6
 0.4200    0.3500    0.5700   15.0000   30.0000   45.0000
```

```
%Gprueba=[0.2 0.15 0.4 15 30 45]
```

```
% Se crea la matriz de transformacion OT6 del punto
```

```
Tprueba=rt2tr(rpy2r(Gprueba(4),Gprueba(5),Gprueba(6),'deg'), [Gprueba(1) Gprueba(2) Gprueba(3)])
```

```
Tprueba = 4x4
 0.6124  -0.6124    0.5000    0.4200
 0.7745    0.5915  -0.2241    0.3500
-0.1585    0.5245    0.8365    0.5700
 0        0        0    1.0000
```

```
Pprueba=(transl(Tprueba))'
```

```
Pprueba = 1x3
 0.4200    0.3500    0.5700
```

### 8.1.1 Encontrar $\theta_1$

Para encontrar  $\theta_1$  se determina primero la posicion del marco de coordenadas {5}, este se toma en cuenta ya que la posicion y la orientacion del marco {6} es conocida y la distancia entre estos marcos, denominada  $d_6$ , tambien, esto implica que puede definirse la posicion  ${}^0P_5$  marco {5} en el espacio

$${}^0P_5 = {}^0P_6 - d_6 \cdot {}^0Z_6 \rightarrow {}^0T_6 \cdot [0, 0, -d_6, 1]'$$

De aqui obtenemos los puntos  $x_5, y_5, z_5$  del marco 5, ahora, se ubica la vista del robot que permita ver el plano XY asi el valor de z del marco  $\{5\}$  puede descartarse. Se puede ver que existe un ángulo desde el eje  $X_0$  al punto  ${}^0P_5$  que denominamos  $\varphi_1 = \text{atan}2({}^0P_{5y}, {}^0P_{5x})$ , pero ademas de este vemos un ángulo creado por el cambio de posicion debido a  $d_4$  y a la distancia general  $|{}^0P_5|$ , por lo que se puede obtener un

$$\varphi_2 = \text{asin}\left(\frac{d_4}{\sqrt{({}^0P_{5x})^2 + ({}^0P_{5y})^2}}\right), \text{ Por lo que } \theta_1 = \varphi_1 - \varphi_2$$

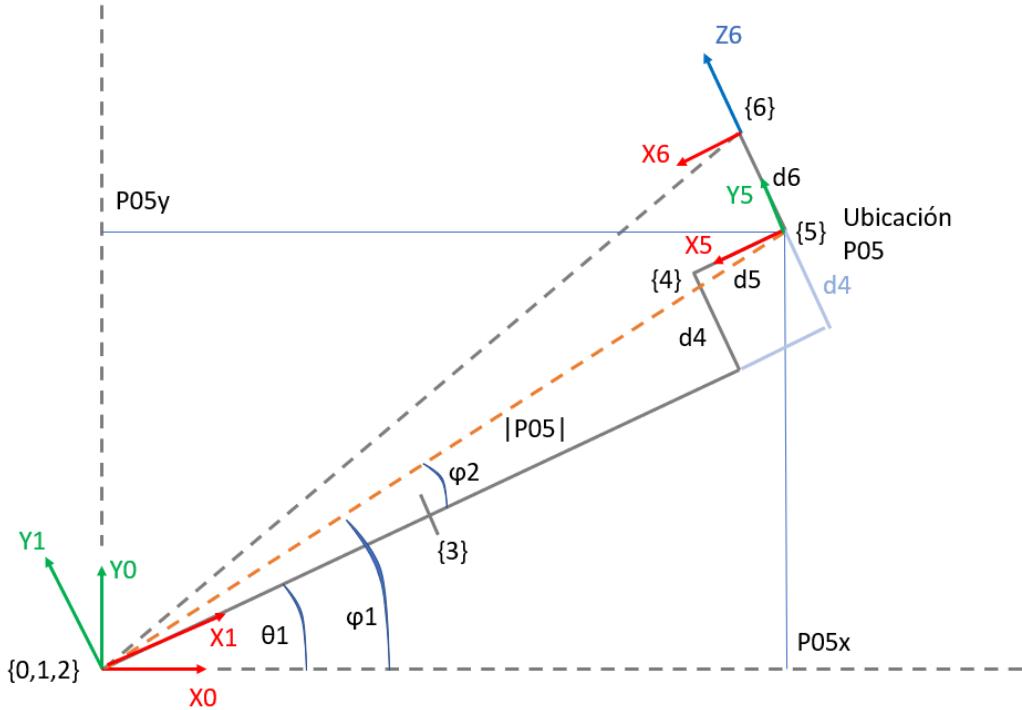


Figura n+1. Obtención del ángulo  $\theta_1$

Se pueden notar dos cosas, si el desplazamiento  $d_4$  no siguiera la orientacion que esta en la figura anterior sino cambiara al lado opuesto, el ángulo  $\theta_1$  cambiaria apenas un poco, entonces existirian una única solucion, pero al tener esta posibilidad con  $d_4$  (lo cual es posible si alteramos la composición física del robot) existen 2 posibles soluciones

```
% Se extrae la posicion de acuerdo a la formula anterior
p05= (Tprueba*[0;0;-d6;1])'
```

```
p05 = 1x4
    0.3789    0.3684    0.5012    1.0000
```

```
% Dos opciones de ángulo para q1,
% q_12 existe si y solo si se altera fisicamente el robót
q_11=atan2(p05(2),p05(1))-asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_11 = 0.5634
```

```
q_12=atan2(p05(2),p05(1))+asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

$$q_{12} = 0.9795$$

### 8.1.2 Encontrar $\theta_5$

Teniendo en cuenta que ya se tiene el punto en el marco  $\{5\}$  y el punto del efecto final con su orientación se puede obtener el ángulo  $\theta_5$ , para ello se mantiene la vista de plano XY y se gira un poco el robót para hacer visible el ángulo

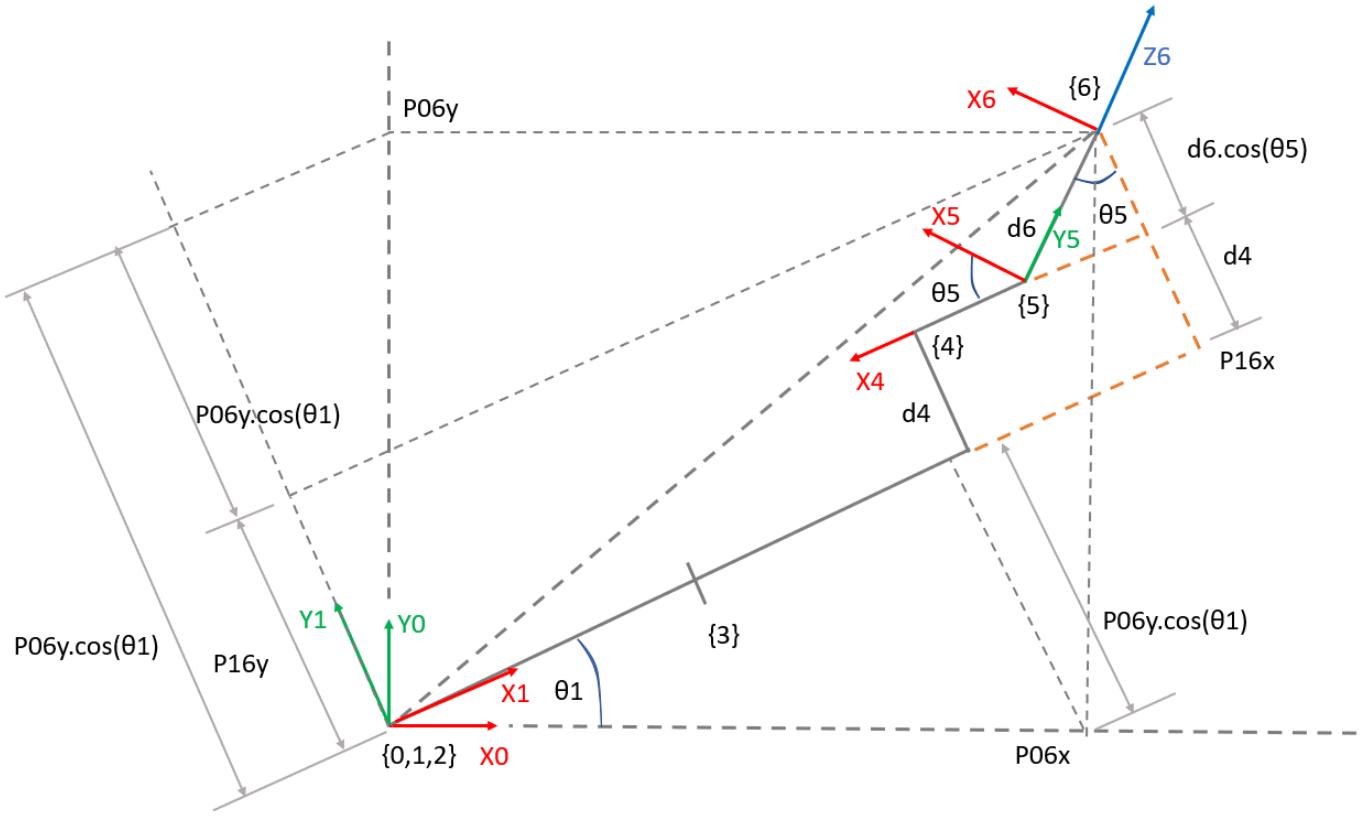


Figura n+2. Obtención del ángulo  $\theta_5$

Se nota entonces que es posible trazar un radio hacia la posición del marco  $\{6\}$ , se observa en la figura anterior que  ${}^1P_6(y) = d_4 + d_6 \cdot \cos(\theta_5)$ . pero se debe tener en cuenta que el marco  $\{1\}$  se encuentra rotado con respecto al marco  $\{0\}$ , por lo que es necesaria tener en cuenta esta rotación alrededor del eje  $Z_0$ . por tanto:

$${}^0P_6 = ({}^0R_1 \cdot {}^1P_6) \rightarrow {}^1P_6 = ({}^0R_1)^T \cdot {}^1P_6$$

$$\begin{bmatrix} {}^1P_{6x} \\ {}^1P_{6y} \\ {}^1P_{6z} \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^0P_{6x} \\ {}^0P_{6y} \\ {}^0P_{6z} \end{bmatrix} \rightarrow \begin{bmatrix} {}^0P_{6x} \\ {}^0P_{6y} \\ {}^0P_{6z} \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^1P_{6x} \\ {}^1P_{6y} \\ {}^1P_{6z} \end{bmatrix}$$

$${}^1P_{6y} = -{}^0P_{6x} \cdot \sin(\theta_1) + {}^0P_{6y} \cdot \cos(\theta_1)$$

Combinando la ecuacion resultante de  ${}^1P_{6y}$  con la obtenida al principio se tiene:

$$-d_4 - d_6 \cdot \cos(\theta_5) = {}^1P_{6y} = -{}^0P_{6x} \cdot (\sin(\theta_1)) + {}^0P_{6y} \cdot \cos(\theta_1)$$

$$\cos(\theta_5) = \frac{-{}^0P_{6x} \cdot \sin(\theta_1) + {}^0P_{6y} \cdot \cos(\theta_1) - d_4}{d_6}$$

$$\theta_5 = \pm \arccos\left(\frac{-{}^0P_{6x} \cdot \sin(\theta_1) + {}^0P_{6y} \cdot \cos(\theta_1) - d_4}{d_6}\right)$$

Nuevamente como en el caso anterior existen 2 soluciones que podrian verse como cuando la muñeca del robot se encuentra en una parte superior o en una parte inferior, la combinacion de las articulaciones 2, 3, 4 puede proveer este efecto que puede ser "corregido" por el ángulo  $\theta_6$ . Se debe notar tambien que el valor interno de la funcion acos no debe ser superior a 1, por lo cual existe una primera restriccion a la solucion:

$$-d_4 - d_6 \cdot \cos(\theta_5) = {}^1P_{6y} \rightarrow \theta_5 = \arccos\left(\frac{-{}^1P_{6y} - d_4}{d_6}\right)$$

$$|{}^1P_{6y} - d_4| \leq |d_6|$$

```
% se tiene que Pprueba es el mismo punto definido como P_06
q_1=q_11
```

q\_1 = 0.5634

```
q_51=acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

q\_51 = 2.0449

```
q_52=-acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

q\_52 = -2.0449

### 8.1.3 Encontrar $\theta_6$

Para entender la orientacion  $\theta_6$  es necesario ver los marcos de referencia del robot y la orientacion de sus ejes XYZ, lo primero que se nota es que el eje  $Z_6$  es paralelo y colineal a  $Y_5$  lo que indica la dependencia de  $\theta_5$ , ademas, se nota que el eje  $Y_1$  siempre es paralelo a los ejes  $Z_2, Z_3, Z_4$

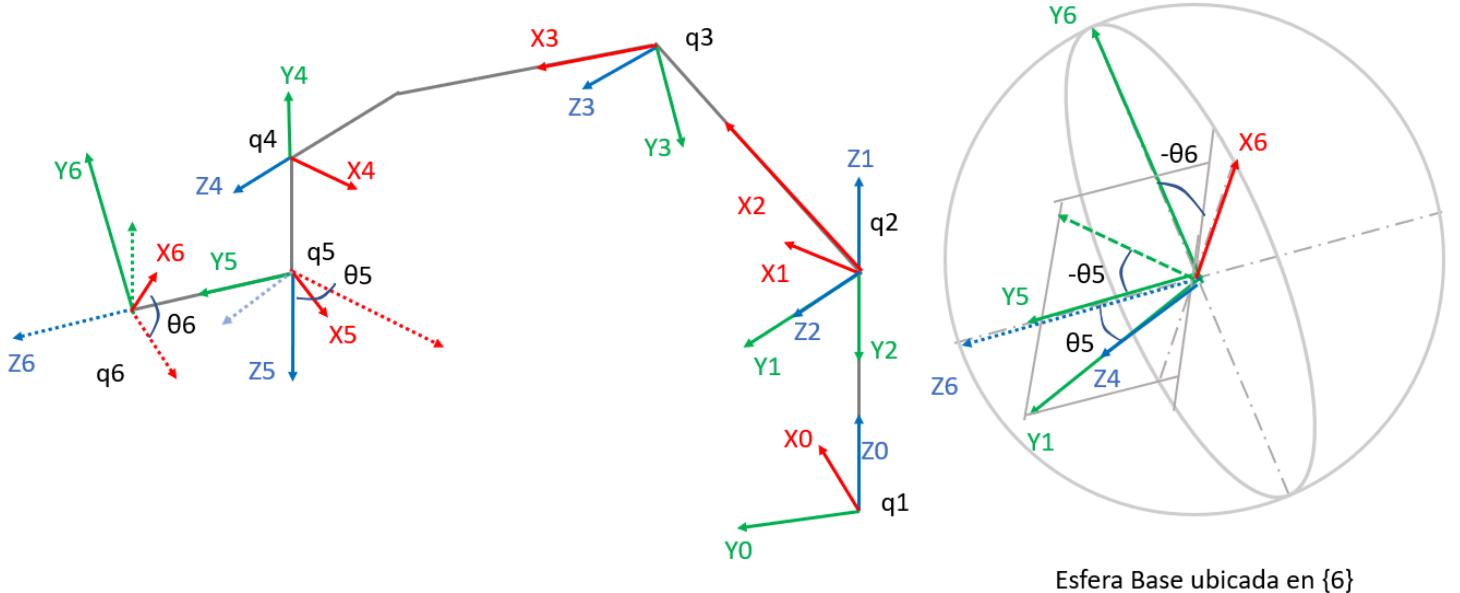


Figura n+3. Obtencion del ángulo  $\theta_6$

Al girar el ángulo  $\theta_5$  se nota que el eje  $Z_6$  gira respecto al ángulo  $Z_4$  (y por consiguiente al eje  $Y_1$ ) cambiando por completo la orientacion del marco  $\{6\}$ , el cual puede girar un ángulo  $\theta_6$  alrededor del eje  $Z_6$ . Este tipo de giro tambien puede verse en la conversion rectangular a polar esferica donde  $-\theta_6$  marca la orientacion (azimuth) y  $\theta_5$  marca la elevacion (zenith) respecto al eje  $Y_6$ . como se esta tomando la orientacion desde el marco  $\{6\}$  se tiene entonces que:

$${}^6\hat{Y}_1 = \begin{bmatrix} \sin(\theta_5)\cos(-\theta_6) \\ \sin(\theta_5)\sin(-\theta_6) \\ \cos(\theta_5) \end{bmatrix}$$

Aunque la formula facilita una conversion para obtener  $\theta_6$  se busca que este ángulo no dependa directamente del marco  $\{1\}$  sino del origen general del sistema,  $\{0\}$ , por lo tanto se retoma la idea de la rotación vista al hallar  $\theta_5$  para hallar el equivalente entre  ${}^1Y_6$  y  ${}^1Y_6$

$${}^1P_{6y} = {}^0P_{6x} \cdot (-\sin(\theta_1)) + {}^0P_{6y} \cdot \cos(\theta_1) \rightarrow {}^1Y_{6y} = {}^0X_6 \cdot (-\sin(\theta_1)) + {}^0Y_6 \cdot \cos(\theta_1)$$

$${}^6Y_1 = \begin{bmatrix} {}^6X_{0x} \cdot (-\sin(\theta_1)) + {}^6Y_{0x} \cdot \cos(\theta_1) \\ {}^6X_{0y} \cdot (-\sin(\theta_1)) + {}^6Y_{0y} \cdot \cos(\theta_1) \\ {}^6X_{0z} \cdot (-\sin(\theta_1)) + {}^6Y_{0z} \cdot \cos(\theta_1) \end{bmatrix} = \begin{bmatrix} \sin(\theta_5)\cos(-\theta_6) \\ \sin(\theta_5)\sin(-\theta_6) \\ \cos(\theta_5) \end{bmatrix}$$

Como  $\theta_1$  y  $\theta_5$  son conocidos por los pasos anteriores, se pueden computar las dos primeras filas (y sus equivalentes), por lo cual se tiene:

$$\begin{aligned} -{}^6X_{0x} \cdot \sin(\theta_1) + {}^6Y_{0x} \cdot \cos(\theta_1) &= \sin(\theta_5)\cos(-\theta_6) & \cos(\theta_6) &= \frac{-{}^6X_{0x} \cdot \sin(\theta_1) + {}^6Y_{0x} \cdot \cos(\theta_1)}{\sin(\theta_5)} \\ -{}^6X_{0y} \cdot \sin(\theta_1) + {}^6Y_{0y} \cdot \cos(\theta_1) &= \sin(\theta_5)\sin(-\theta_6) & \sin(\theta_5) &= \frac{{}^6X_{0y} \cdot \sin(\theta_1) - {}^6Y_{0y} \cdot \cos(\theta_1)}{\sin(\theta_5)} \end{aligned}$$

$$\theta_6 = \text{atan2}\left(\frac{-{}^6X_{0y} \cdot \sin(\theta_1) + {}^6Y_{0y} \cdot \cos(\theta_1)}{\sin(\theta_5)}, \frac{{}^6X_{0x} \cdot \sin(\theta_1) - {}^6Y_{0x} \cdot \cos(\theta_1)}{\sin(\theta_5)}\right)$$

De aqui se nota algo importante, si  $\sin(\theta_5) = 0$  la solucion es indeterminada, en este caso se asume que el eje 6 esta alineado con los ejes 2,3 o 4, en este caso se considera la articulacion 6 como redindante y  $\theta_6$  puede asumir cualquier valor. Si el denominador de las ecuaciones es igual en ambos casos a cero la ecuacion es indeterminada, en este caso  $\sin(\theta_5)$  debe ser igual a cero obligatoriamente, cuando esto ocurre se deben revisar ambos terminos de la ecuación

```
% Para probar un caso, se asume que la muñeca esta apuntando hacia abajo
q_5=q_51
```

```
q_5 = 2.0449
```

```
% Se define las rotaciones de los ejes respecto al marco 6
% usando la matriz inversa de T_prueba
T60=Tprueba^-1
```

```
T60 = 4x4
0.6124  0.7745 -0.1585 -0.4379
-0.6124  0.5915  0.5245 -0.2488
0.5000 -0.2241  0.8365 -0.6084
0         0         0      1.0000
```

```
% Se extrae de la MTH las posiciones de los vectores unitarios mencionados
```

```
X60x=T60(1,1)
```

```
X60x = 0.6124
```

```
Y60x=T60(1,2)
```

```
Y60x = 0.7745
```

```
X60y=T60(2,1)
```

```
X60y = -0.6124
```

```
Y60y=T60(2,2)
```

```
Y60y = 0.5915
```

```
% Se opera con los datos obtenidos
```

$$q_6 = \text{atan}2((X60y * \sin(q_1)) - Y60y * \cos(q_1)) / \sin(q_5), (-X60x * \sin(q_1)) + Y60x * \cos(q_1)) / \sin(q_5))$$

$$q_6 = -1.1936$$

### 8.1.4 Encontrar $\theta_3$

A partir de este punto es posible encontrar  $\theta_3$  teniendo como base la MTH  ${}^1T_4$ , debido a que se conocen los marcos  ${}^0T_1$ ,  ${}^4T_5$  y  ${}^5T_6$  en base a los ángulos ya obtenidos  $\theta_1$ ,  $\theta_5$  y  $\theta_6$ . Para determinar el angulo  $\theta_3$  se toma el punto  ${}^1P_4$  y se determina su distancia respecto al marco  $\{1\}$ , para hacer una vizualización efectiva se hace una vista respecto al plano XZ del marco  $\{1\}$

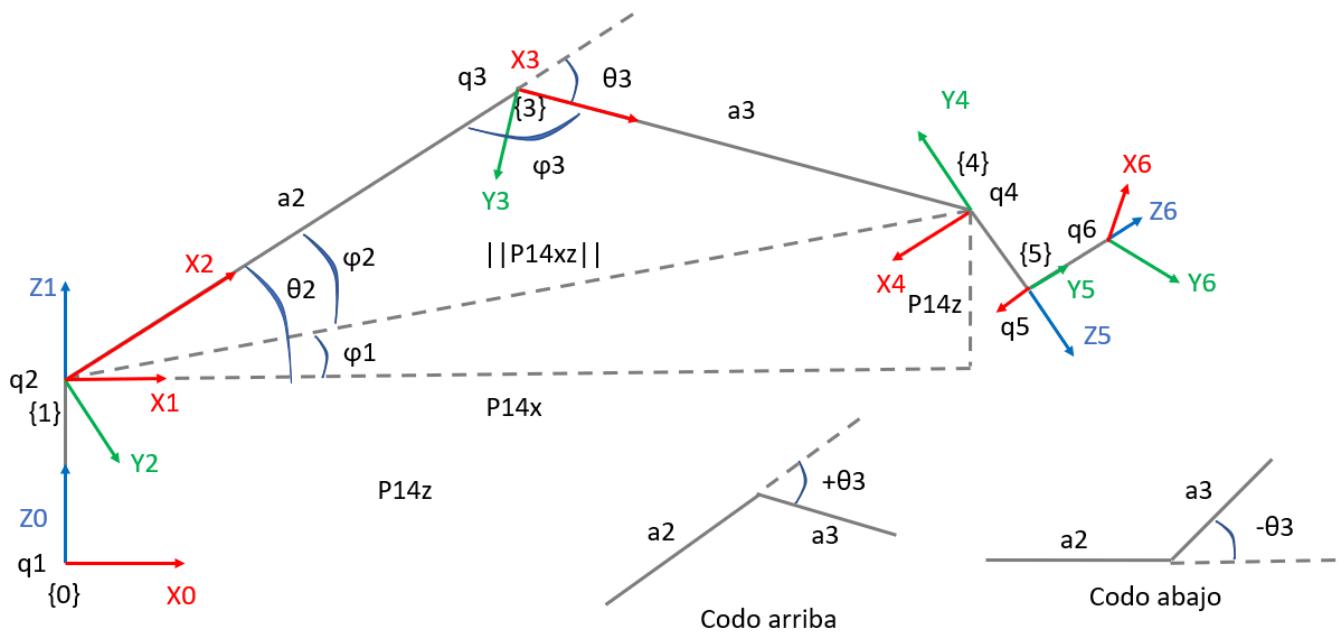


Figura n+4. Obtención del ángulo  $\theta_3$

Para la figura, sabiendo que los sistemas de referencia  $\{1\}$  y  $\{4\}$  estan fijos(o ya se han obtenido),  $\theta_3$  y  $\varphi_3$  depende únicamente de las distancias  $a_2$  y  $a_3$ , se obtiene facilmente el ángulo  $\varphi_3$  usando la ley de los cosenos

$$\cos(\varphi_3) = \frac{(a_2)^2 + (a_3)^2 - ||{}^1P_4||^2}{2 \cdot a_2 \cdot a_3}$$

Se puede establecer la relacion entre  $\theta_3$  y  $\varphi_3$ , de manera que:

$$\theta_3 = \pi - \varphi_3 \rightarrow \cos(\theta_3) = \cos(\pi - \varphi_3) = -\cos(\varphi_3)$$

Asi entonces se puede obtener:

$$\cos(\theta_3) = -\frac{a_3^2 + a_4^2 - ||{}^1P_4||^2}{2 \cdot a_3 \cdot a_4} \rightarrow \theta_3 = \pm \arccos\left(\frac{||{}^1P_4||^2 - a_3^2 - a_4^2}{2 \cdot a_3 \cdot a_4}\right)$$

Para que exista la solucion se condiciona que, para exista el triangulo,  $||{}^1P_4||$  debe ser lo suficientemente largo para que dos de los lados del triangulo sean mas largas que el lado restante por lo que  $\theta_3$  existe

si  $\|{}^1P_4\| \in [|a_2 - a_3|, |a_2 + a_3|]$ , además nota que existen 2 soluciones para  $\theta_3$  que corresponden a las configuraciones codo arriba y codo abajo.

```
% Asumimos que ya poseemos los ángulos anteriormente hallados

% Se hallan las MTH 0-1, 4-5, 5-6, que contienen q1, q5 y q6
T_01 = L(1).A(q_1); T_45 = L(5).A(q_5); T_56 = L(6).A(q_6);

% se obtiene la MTH 1->4
T_14=T_01^-1*Tprueba*(T_45*T_56)^-1;

% Se extrae el vector de posición de 4 respecto a 1 y se extrae su
% distancia en el plano xz
P_14=transl(T_14)

P_14 =
0.4281
0.1091
0.4442
```

```
P14xz=sqrt(P_14(1)^2+P_14(3)^2)
```

```
P14xz = 0.6169
```

```
q_31=acos((P14xz^2-a2^2-a3^2)/(2*a2*a3))
```

```
q_31 = 1.4321
```

```
q_32=-acos((P14xz^2-a2^2-a3^2)/(2*a2*a3))
```

```
q_32 = -1.4321
```

### 8.1.5 Encontrar $\theta_2$

Retomando la figura anterior que se uso para obtener  $\theta_3$  se puede encontrar  $\theta_2$  como la relacion entre  $\varphi_1$  y  $\varphi_2$ , tal que  $\theta_3 = \varphi_1 - \varphi_2$ . cada uno de estos se puede obtener usando atan2 y la ley de los senos

$$\begin{aligned} \varphi_1 &= \text{atan2}(-{}^1P_{4z}, -{}^1P_{4x}) \\ \frac{\sin(\varphi_2)}{a_3} &= \frac{\sin(\varphi_3)}{\|{}^1P_4\|} \rightarrow \varphi_2 = \text{asin}\left(\frac{a_3 \cdot \sin(\theta_3)}{\|{}^1P_4\|}\right) \\ \theta_3 &= \varphi_1 - \varphi_2 = \text{atan2}(-{}^2P_{4z}, -{}^2P_{4x}) - \text{asin}\left(\frac{a_3 \cdot \sin(\theta_3)}{\|{}^1P_4\|}\right) \end{aligned}$$

```
% para esta solución tomamos q3 negativo, lo que implica que el robot se
% ubica "codo arriba"
```

```
q_3=q_32;
```

```
q_2=-atan2(P_14(3),P_14(1))+asin(-a3*sin(q_3)/P14xz)
```

```
q_2 = -0.1227
```

### 8.1.6 Encontrar $\theta_4$

Finalmente, el último ángulo  $\theta_4$  puede obtenerse usando el cambio de posicion de los ejes XYZ vistos en la MTH  ${}^3T_4$

$${}^{i-1}{}_iT = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos(\alpha_{i-1}) & \cos \theta_i \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin \theta_i \sin(\alpha_{i-1}) & \cos \theta_i \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura n+5. Matriz de transformación homogenea para DH modificado

Como  $\alpha_{i-1} = 0$  se puede extraer el vector  ${}^3\hat{X}_4 = [\cos(\theta_4), \sin(\theta_4), 0]^T$  por lo tanto,  $\theta_4$  se puede obtener como:

$$\theta_4 = \text{atan2}({}^3\hat{X}_{4y}, {}^3\hat{X}_{4x})$$

hay que notar algo importante, existe un offset de  $\pi$  en esta articulacion, lo que implica que se puede sumar o restar este offset al ángulo obtenido, pero tambien se puede cambiar de signo a los terminos del arcotangente, dejando asi:

$$\theta_4 = \text{atan2}(-{}^3\hat{X}_{4y}, -{}^3\hat{X}_{4x})$$

```
% Se obtienen las matrices de transformacion restantes
T_12 = L(2).A(q_2); T_23 = L(3).A(q_3);
```

```
T_34=(T_12*T_23)^-1*T_14
```

```
T_34 = 4x4
-0.9456  0.3255 -0.0000  0.3922
-0.3255 -0.9456 -0.0000 -0.0000
-0.0000 -0.0000  1.0000  0.1091
  0        0        0    1.0000
```

```
X34x=T_34(1,1)
```

```
X34x = -0.9456
```

```
X34y=T_34(2,1)
```

```
X34y = -0.3255
```

```
q_4=atan2(-X34y,-X34x)
```

```
q_4 = 0.3315
```

Por lo tanto, el vector de solucion obtenido es equivalente a

```
qInvSol = [q_1, q_2, q_3, q_4, q_5, q_6]
```

```
qInvSol = 1x6
0.5634 -0.1227 -1.4321 0.3315 2.0449 -1.1936
```

Comparandolo con la solucion numerica del RVC se tiene

```
Solucion = UR5_RVC.ikunc(Tprueba,[0 0 0 0 0 0])
```

```
Solucion = 1x6
0.5634 -0.1227 -1.4321 0.3315 2.0449 -1.1936
```

Ahora usando esta metodologia de solucion, la aplicamos para los puntos de prueba restantes con sus coordenadas generalizadas.

```
%% Caso 1
disp('Caso 1')
```

```
Caso 1
```

```
Gprueba=[0.2 0.15 0.4 15 30 45]
```

```
Gprueba = 1x6
0.2000 0.1500 0.4000 15.0000 30.0000 45.0000
```

```
Tprueba=rt2tr(rpy2r(Gprueba(4),Gprueba(5),Gprueba(6),'deg'), [Gprueba(1) Gprueba(2) Gprueba(3)]);
Pprueba=(transl(Tprueba))';
```

```
% --- obtencion de Q1
% Se extrae la posicion de acuerdo a la formula anterior
p05= (Tprueba*[0;0;-d6;1])';
% Dos opciones de ángulo para q1,
q_11=atan2(p05(2),p05(1))-asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_11 = 0.3238
```

```
q_12=atan2(p05(2),p05(1))+asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_12 = 1.3056
```

```
% --- obtencion de Q5
% se toma como referencia q11
q_1=q_11
```

```
q_1 = 0.3238
```

```
% dos opciones dependientes del q_1 anterior
q_51=acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

```
q_51 = 1.9515
```

```
q_52=-acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

```
q_52 = -1.9515
```

```
% --- obtencion de Q6  
% se toma como referencia q_51  
q_5=q_51
```

```
q_5 = 1.9515
```

```
T60=Tprueba^-1;  
X60x=T60(1,1); Y60x=T60(1,2); X60y=T60(2,1); Y60y=T60(2,2);  
q_6=atan2((X60y*sin(q_1)-Y60y*cos(q_1))/sin(q_5),(-X60x*sin(q_1)+Y60x*cos(q_1))/sin(q_5))
```

```
q_6 = -0.9508
```

```
% --- obtencion de Q3
```

```
T_01 = L(1).A(q_1); T_45 = L(5).A(q_5); T_56 = L(6).A(q_6);  
T_14=T_01^-1*Tprueba*(T_45*T_56)^-1;
```

```
P_14=transl(T_14);  
P14xz=sqrt(P_14(1)^2+P_14(3)^2);
```

```
q_31=acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));  
q_32=-acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));
```

```
% --- obtencion de Q2  
% se toma como referencia q32  
q_3=q_32;
```

```
q_2=-atan2(P_14(3),P_14(1))+asin(-a3*sin(q_3)/P14xz)
```

```
q_2 = -0.0845
```

```
% --- obtencion de Q4
```

```
T_12 = L(2).A(q_2); T_23 = L(3).A(q_3);  
T_34=(T_12*T_23)^-1*T_14;
```

```
X34x=T_34(1,1); X34y=T_34(2,1);
```

```
q_4=atan2(-X34y,-X34x)
```

```
q_4 = 1.3377
```

```
qInvSol1 = [q_1, q_2, q_3, q_4, q_5, q_6]
```

```
qInvSol1 = 1×6  
0.3238 -0.0845 -2.3753 1.3377 1.9515 -0.9508
```

```
% Solucion numerica optimizada  
SolRVC1 = UR5_RVC.ikunc(Tprueba,[0 0 0 0 0 0])
```

```
SolRVC1 = 1x6  
0.3238 -0.0845 -2.3753 1.3377 1.9515 -0.9508
```

```
%%% Caso 2  
disp('Caso 2')
```

```
Caso 2
```

```
Gprueba=[-0.3 0.2 -0.2 75 -20 0]
```

```
Gprueba = 1x6  
-0.3000 0.2000 -0.2000 75.0000 -20.0000 0
```

```
Tprueba=rt2tr(rpy2r(Gprueba(4),Gprueba(5),Gprueba(6),'deg'), [Gprueba(1) Gprueba(2) Gprueba(3)]  
Pprueba=(transl(Tprueba));
```

```
% --- obtencion de Q1  
% Se extrae la posicion de acuerdo a la formula anterior  
p05= (Tprueba*[0;0;-d6;1])';  
% Dos opciones de ángulo para q1,  
q_11=atan2(p05(2),p05(1))-asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_11 = 2.0647
```

```
q_12=atan2(p05(2),p05(1))+asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_12 = 2.6373
```

```
% --- obtencion de Q5  
% se toma como referencia q11  
q_1=q_11
```

```
q_1 = 2.0647
```

```
% dos opciones dependientes del q_1 anterior  
q_51=acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

```
q_51 = 0.7504
```

```
q_52=-acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

```
q_52 = -0.7504
```

```
% --- obtencion de Q6  
% se toma como referencia q_52, debido a inconsistencia en el numero  
q_5=q_52
```

```
q_5 = -0.7504
```

```

T60=Tprueba^-1;
X60x=T60(1,1); Y60x=T60(1,2); X60y=T60(2,1); Y60y=T60(2,2);

q_6=atan2((X60y*sin(q_1)-Y60y*cos(q_1))/sin(q_5),(-X60x*sin(q_1)+Y60x*cos(q_1))/sin(q_5))

q_6 = -0.1809

```

```

% --- obtencion de Q3

T_01 = L(1).A(q_1); T_45 = L(5).A(q_5); T_56 = L(6).A(q_6);
T_14=T_01^-1*Tprueba*(T_45*T_56)^-1;

P_14=transl(T_14);
P14xz=sqrt(P_14(1)^2+P_14(3)^2);

q_31=acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));
q_32=-acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));

% --- obtencion de Q2
% se toma como referencia q31, debido a inconsistencia en la solucion
q_3=q_31;

q_2=-atan2(P_14(3),P_14(1))+asin(-a3*sin(q_3)/P14xz)

q_2 = -0.4146

```

```

% --- obtencion de Q4

T_12 = L(2).A(q_2); T_23 = L(3).A(q_3);
T_34=(T_12*T_23)^-1*T_14;

X34x=T_34(1,1); X34y=T_34(2,1);

q_4=atan2(-X34y,-X34x)

q_4 = -1.1666

```

```
qInvSol2 = [q_1, q_2, q_3, q_4, q_5, q_6]
```

```
qInvSol2 = 1x6
2.0647 -0.4146 1.9458 -1.1666 -0.7504 -0.1809
```

```
% Solucion numerica optimizada
SolRVC2 = UR5_RVC.ikunc(Tprueba,[0 0 0 0 0 0])
```

```
SolRVC2 = 1x6
2.0647 -0.4146 1.9458 -1.1666 -0.7504 -0.1809
```

```
%%% Caso 3
disp('Caso 3')
```

Caso 3

```
Gprueba=[-0.35 0.3 0.1 -10 35 50]
```

```
Gprueba = 1x6  
-0.3500    0.3000    0.1000   -10.0000   35.0000   50.0000
```

```
Tprueba=rt2tr(rpy2r(Gprueba(4),Gprueba(5),Gprueba(6),'deg'), [Gprueba(1) Gprueba(2) Gprueba(3)]  
Pprueba=(transl(Tprueba))';
```

```
% --- obtencion de Q1  
% Se extrae la posicion de acuerdo a la formula anterior  
p05= (Tprueba*[0;0;-d6;1])';  
% Dos opciones de ángulo para q1,  
q_11=atan2(p05(2),p05(1))-asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_11 = 2.2895
```

```
q_12=atan2(p05(2),p05(1))+asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

```
q_12 = 2.7380
```

```
% --- obtencion de Q5  
% se toma como referencia q11  
q_1=q_11
```

```
q_1 = 2.2895
```

```
% dos opciones dependientes del q_1 anterior  
q_51=acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

```
q_51 = 2.1239
```

```
q_52=-acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

```
q_52 = -2.1239
```

```
% --- obtencion de Q6  
% se toma como referencia q_51  
q_5=q_51
```

```
q_5 = 2.1239
```

```
T60=Tprueba^-1;  
X60x=T60(1,1); Y60x=T60(1,2); X60y=T60(2,1); Y60y=T60(2,2);
```

```
q_6=atan2((X60y*sin(q_1)-Y60y*cos(q_1))/sin(q_5),(-X60x*sin(q_1)+Y60x*cos(q_1))/sin(q_5))
```

```
q_6 = -3.1354
```

```
% --- obtencion de Q3
```

```
T_01 = L(1).A(q_1); T_45 = L(5).A(q_5); T_56 = L(6).A(q_6);  
T_14=T_01^-1*Tprueba*(T_45*T_56)^-1;
```

```

P_14=transl(T_14);
P14xz=sqrt(P_14(1)^2+P_14(3)^2);

q_31=acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));
q_32=-acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));

% --- obtencion de Q2
% se toma como referencia q32
q_3=q_32;

q_2=-atan2(P_14(3),P_14(1))+asin(-a3*sin(q_3)/P14xz)

```

q\_2 = 1.2083

```

% --- obtencion de Q4

T_12 = L(2).A(q_2); T_23 = L(3).A(q_3);
T_34=(T_12*T_23)^-1*T_14;

X34x=T_34(1,1); X34y=T_34(2,1);

q_4=atan2(-X34y,-X34x)

```

q\_4 = -0.9758

qInvSol3 = [q\_1, q\_2, q\_3, q\_4, q\_5, q\_6]

qInvSol3 = 1×6  
2.2895 1.2083 -2.1269 -0.9758 2.1239 -3.1354

```

% Solucion numerica optimizada
SolRVC3 = UR5_RVC.ikunc(Tprueba,[0 0 0 0 0 0])

```

SolRVC3 = 1×6  
2.2895 1.2083 -2.1269 -0.9758 2.1239 -3.1354

```

%%% Caso 4
disp('Caso 4')

```

Caso 4

Gprueba=[0.3 -0.4 0 90 -45 -45]

Gprueba = 1×6  
0.3000 -0.4000 0 90.0000 -45.0000 -45.0000

Tprueba=rt2tr(rpy2r(Gprueba(4),Gprueba(5),Gprueba(6),'deg'), [Gprueba(1) Gprueba(2) Gprueba(3)]  
Pprueba=(transl(Tprueba))';

```

% --- obtencion de Q1
% Se extrae la posicion de acuerdo a la formula anterior
p05= (Tprueba*[0;0;-d6;1])';

```

```
% Dos opciones de ángulo para q1,  
q_11=atan2(p05(2),p05(1))-asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

q\_11 = -0.9843

```
q_12=atan2(p05(2),p05(1))+asin(d4/(sqrt(p05(1)^2+p05(2)^2)))
```

q\_12 = -0.5397

```
% --- obtencion de Q5  
% se toma como referencia q11  
q_1=q_11
```

q\_1 = -0.9843

```
% dos opciones dependientes del q_1 anterior  
q_51=acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

q\_51 = 2.9427

```
q_52=-acos((-Pprueba(1)*sin(q_1)+Pprueba(2)*cos(q_1)-d4)/d6)
```

q\_52 = -2.9427

```
% --- obtencion de Q6  
% se toma como referencia q_52 debido a inconsistencias en la solución  
q_5=q_52
```

q\_5 = -2.9427

```
T60=Tprueba^-1;  
X60x=T60(1,1); Y60x=T60(1,2); X60y=T60(2,1); Y60y=T60(2,2);
```

```
q_6=atan2((X60y*sin(q_1)-Y60y*cos(q_1))/sin(q_5),(-X60x*sin(q_1)+Y60x*cos(q_1))/sin(q_5))
```

q\_6 = 2.3562

```
% --- obtencion de Q3
```

```
T_01 = L(1).A(q_1); T_45 = L(5).A(q_5); T_56 = L(6).A(q_6);  
T_14=T_01^-1*Tprueba*(T_45*T_56)^-1;
```

```
P_14=transl(T_14);  
P14xz=sqrt(P_14(1)^2+P_14(3)^2);
```

```
q_31=acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));  
q_32=-acos((P14xz^2-a2^2-a3^2)/(2*a2*a3));
```

```
% --- obtencion de Q2  
% se toma como referencia q31 debido a inconsistencias en la solucion  
q_3=q_31;
```

```
q_2=-atan2(P_14(3),P_14(1))+asin(-a3*sin(q_3)/P14xz)
```

```
q_2 = -0.4745
```

```
% --- obtencion de Q4
```

```
T_12 = L(2).A(q_2); T_23 = L(3).A(q_3);  
T_34=(T_12*T_23)^-1*T_14;
```

```
X34x=T_34(1,1); X34y=T_34(2,1);
```

```
q_4=atan2(-X34y,-X34x)
```

```
q_4 = 1.8415
```

```
qInvSol4 = [q_1, q_2, q_3, q_4, q_5, q_6]
```

```
qInvSol4 = 1x6  
-0.9843 -0.4745 1.7746 1.8415 -2.9427 2.3562
```

```
% Solucion numerica optimizada
```

```
SolRVC4 = UR5_RVC.ikunc(Tprueba,[0 0 0 0 0 0])
```

```
SolRVC4 = 1x6  
-0.9843 -0.4745 1.7746 1.8415 -2.9427 2.3562
```

## 8.2 En las soluciones de la inversa incluya consideraciones respecto a multiplicidad de soluciones.

Existen, segun la solucion obtenida, 8 posibles soluciones para una MTH bajo los criterios mencionados, 2 de  $\theta_1$ , 2 de  $\theta_5$ , una de  $\theta_6$ , 2 de  $\theta_3$ , una de  $\theta_2$  y una de  $\theta_4$ , las cuales se mencionan en el código como q\_11,q\_12 como las 2 posibles soluciones de q\_1

se debe tener en cuenta que los pasos son secuenciales, asi que uno a uno los ángulos dependen de sus anteriores soluciones y elegir otra opcion implica recalcular nuevamente los ángulos siguientes.

## 8.3 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:

### 8.3.1 ¿Cuál es la diferencia entre estas funciones?

El toolbox RVC cuenta con 7 funciones para el uso de la cinematica inversa: ikcon, ikine, ikine3, ikine6s, ikine\_sym, ikinem,ikunc.

**ikcon** usa cinematica inversa con un rango limitado de movimiento en las articulaciones

**ikine3, ikine6s, ikine** usa cinematica inversa usando métodos numéricos, bajo la condicion que el robot serial usa una muñeca esferica, en el caso ikine3 hace la cinematica inversa para un robot de posicion de 3 grados de libertad sin esta muñeca esferica, el caso ikine6s esta optimizado analiticamente para robots de 6 grados de libertad con muñeca esferica

**ikine\_sym** devuelve la solucion simbolica de la solucion como un vector de Nx1 donde cada elemento muestra la solucion por cada articulación del robot serial

**ikinem** hace una solucion numerica por minimizacion en la configuracion, depende intrinsecamente de una configuracion inicial definida en un vector q y usa el metodo de minimizacion Levenberg\_Marquadt

**ikunc** hace una solucion numerica similar a la anterior, pero usa algoritmos de optimizacion, depende del toolbox de optimizacion de matlab y retorna ademas el error del procedimiento

### 8.3.2 ¿Cuál debe usar para su robot y por qué?

Se usa la función ikunc debido a que al probar las funciones posibles, esta fue la que mejor dió resultados al compararlo entre ellas. No se usa la función ikine ya que estas tiende a hacer muchas iteraciones y su aproximación es mala.

A continuación se lista un código con el que se realizaron las pruebas:

```
L1 = 0.089159; L2 = 0.13585; L3 = 0.425; L4 = 0.1197;
L5 = 0.39225; L6 = 0.093; L7= 0.09465; L8= 0.0823;

L9 = 0; %L9 Es la distancia del Tool

L(1) = Link('revolute','alpha',0,'a',0,'d',L1,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha',-pi/2,'a',0,'d',L2,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha',0,'a',L3,'d', -L4,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha',0,'a',L5,'d',L6,'offset',-pi,'modified','qlim',[-2*pi 2*pi]);
L(5) = Link('revolute','alpha',pi/2,'a',0,'d',L7,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(6) = Link('revolute','alpha',-pi/2,'a',0,'d',L8,'offset',0,'modified','qlim',[-2*pi 2*pi]);

UR5_RVC = SerialLink(L, 'name', 'RRRRRR')
```

UR5\_RVC =

RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

j	theta	d	a	alpha	offset
1	q1	0.08916	0	0	0
2	q2	0.1358	0	-1.571	0
3	q3	-0.1197	0.425	0	0
4	q4	0.093	0.3922	0	-3.142
5	q5	0.09465	0	1.571	0
6	q6	0.0823	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 0
                  0 0 0 1      0 0 0 1
```

```
%UR5_RVC.tool = [ 0 -1 0 0; 1 0 0 0; 0 0 1 L9; 0 0 0 1]
UR5_RVC.tool = [ 1 0 0 0; 0 1 0 0; 0 0 1 L9; 0 0 0 1]
```

```

UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

+-----+-----+-----+-----+
| j | theta | d | a | alpha | offset |
+-----+-----+-----+-----+
| 1 | q1 | 0.08916 | 0 | 0 | 0 |
| 2 | q2 | 0.1358 | 0 | -1.571 | 0 |
| 3 | q3 | -0.1197 | 0.425 | 0 | 0 |
| 4 | q4 | 0.093 | 0.3922 | 0 | -3.142 |
| 5 | q5 | 0.09465 | 0 | 1.571 | 0 |
| 6 | q6 | 0.0823 | 0 | -1.571 | 0 |
+-----+-----+-----+-----+
grav = 0 base = 1 0 0 0 tool = 1 0 0 0
      0 0 1 0 0 1 0 0
      9.81 0 0 1 0 0 0 1
      0 0 0 1 0 0 1 0

```

% Para realizar la prueba se crea un vector de prueba q que contienen una  
% configuracion aleatoria

```
q_pred=[pi/6 pi/6 pi/6 pi/6 pi/6 pi/6]
```

```
q_pred = 1x6
0.5236 0.5236 0.5236 0.5236 0.5236 0.5236
```

% Se obtienen las matrices de transformacion

```
T_0_1 = L(1).A(q_pred(1)); T_1_2 = L(2).A(q_pred(2)); T_2_3 = L(3).A(q_pred(3));
T_3_4 = L(4).A(q_pred(4)); T_4_5 = L(5).A(q_pred(5)); T_5_6 = L(6).A(q_pred(6));
T_0_Tool = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6 * UR5_RVC.tool
```

```
T_0_Tool = 4x4
0.2165 0.8750 -0.4330 0.3164
0.6250 0.2165 0.7500 0.3910
0.7500 -0.4330 -0.5000 -0.5042
0 0 0 1.0000
```

% Al ser un metodo numérico requiere un punto inicial de prueba, el cual se  
% define manualmente, la eficiencia de computo dependera de que tan cercano  
% este el vector a la solución

```
q_prueba=[0.1 0.1 0.1 0.1 0.1 0.1]
```

```
q_prueba = 1x6
0.1000 0.1000 0.1000 0.1000 0.1000 0.1000
```

% Se ejecutan las 4 funciones posibles del toolbox RVC para comparar los  
% resultados

% Aclarar que Pi/6=0.5236...

```
A=UR5_RVC.ikcon(T_0_Tool, q_prueba) % sirve nivel intermedio
```

```
A = 1x6  
0.5237 0.5239 0.5229 0.5239 0.5236 0.5236
```

```
B=UR5_RVC.ikinem(T_0_Tool, q_prueba) % sirve, aunque deficientemente
```

```
B = 1x6  
0.6561 0.8423 0.1717 1.0339 -1.5864 -0.3707
```

```
C=UR5_RVC.ikunc(T_0_Tool, q_prueba) % sirve, nivel avanzado
```

```
C = 1x6  
0.5236 0.5236 0.5237 0.5236 0.5236 0.5236
```

```
D=UR5_RVC.ikine(T_0_Tool, q_prueba) % el peor resultado
```

```
Warning: Initial joint configuration results in a (near-)singular configuration, this may slow convergence  
Warning: solution diverging at step 46, try reducing alpha  
Warning: solution diverging at step 79, try reducing alpha  
Warning: solution diverging at step 266, try reducing alpha  
Warning: solution diverging at step 283, try reducing alpha  
Warning: solution diverging at step 304, try reducing alpha  
Warning: solution diverging at step 306, try reducing alpha  
Warning: solution diverging at step 364, try reducing alpha  
Warning: solution diverging at step 381, try reducing alpha  
Warning: solution diverging at step 522, try reducing alpha  
Warning: solution diverging at step 579, try reducing alpha  
Warning: solution diverging at step 588, try reducing alpha  
Warning: solution diverging at step 706, try reducing alpha  
Warning: solution diverging at step 713, try reducing alpha  
Warning: solution diverging at step 717, try reducing alpha  
Warning: solution diverging at step 793, try reducing alpha  
Warning: solution diverging at step 855, try reducing alpha  
Warning: solution diverging at step 906, try reducing alpha  
Warning: ikine: iteration limit 1000 exceeded (row 1), final err 1.887970
```

```
D = 1x6  
0.0917 6.4404 -0.0020 -8.2314 -0.0027 -0.1091
```

Concluida la prueba, se demuestra la eficiencia de **ikunc**.

```
% Punto de prueba 1
```

```
G_prueba=[0.2 0.15 0.4 15 30 45]
```

```
G_prueba = 1x6  
0.2000 0.1500 0.4000 15.0000 30.0000 45.0000
```

```
% Se crea la matriz de transformacion a partir del vector de  
% coordenadas generalizadas usando funciones del toolbox RVC
```

```
T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prue
```

```
T_prueba = 4x4  
0.6124 -0.6124 0.5000 0.2000  
0.7745 0.5915 -0.2241 0.1500  
-0.1585 0.5245 0.8365 0.4000
```

```
0          0          0    1.0000
```

```
Solucion1=UR5_RVC.ikunc(T_prueba,q_prueba)
```

```
Solucion1 = 1x6
0.3238   -0.0845   -2.3753    1.3377    1.9515   -0.9508
```

```
% Punto de prueba 2
```

```
G_prueba=[ -0.3 0.2 -0.2 75 -20 0]
```

```
G_prueba = 1x6
-0.3000    0.2000   -0.2000    75.0000   -20.0000        0
```

```
% Se crea la matriz de transformacion a partir del vector de
% coordenadas generalizadas usando funciones del toolbox RVC
```

```
T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prue
```

```
T_prueba = 4x4
0.9397        0   -0.3420   -0.3000
-0.3304    0.2588   -0.9077    0.2000
0.0885    0.9659    0.2432   -0.2000
0          0        0    1.0000
```

```
Solucion2=UR5_RVC.ikunc(T_prueba,q_prueba)
```

```
Solucion2 = 1x6
2.0647   -0.4146    1.9458   -1.1666   -0.7504   -0.1809
```

```
% Punto de prueba 3
```

```
G_prueba=[ -0.35 0.3 0.1 -10 35 50]
```

```
G_prueba = 1x6
-0.3500    0.3000    0.1000   -10.0000    35.0000   50.0000
```

```
% Se crea la matriz de transformacion a partir del vector de
% coordenadas generalizadas usando funciones del toolbox RVC
```

```
T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prue
```

```
T_prueba = 4x4
0.5265   -0.6275    0.5736   -0.3500
0.6904    0.7093    0.1422    0.3000
-0.4961    0.3211    0.8067    0.1000
0          0        0    1.0000
```

```
Solucion3=UR5_RVC.ikunc(T_prueba,q_prueba)
```

```
Solucion3 = 1x6
2.2895    1.2083   -2.1269   -0.9758    2.1239   -3.1354
```

```
% Punto de prueba 4
```

```
G_prueba=[0.3 -0.4 0 90 -45 -45]
```

```
G_prueba = 1x6
0.3000    -0.4000         0    90.0000   -45.0000   -45.0000
```

```
% Se crea la matriz de transformacion a partir del vector de
% coordenadas generalizadas usando funciones del toolbox RVC
```

```
T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prue
```

```
T_prueba = 4x4
0.5000    0.5000   -0.7071    0.3000
-0.5000   -0.5000   -0.7071   -0.4000
-0.7071    0.7071    0.0000      0
     0        0        0       1.0000
```

```
Solucion4=UR5_RVC.ikunc(T_prueba,q_prueba)
```

```
Solucion4 = 1x6
-0.9843   -0.4745    1.7746    1.8415   -2.9427    2.3562
```

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

```
UR5_RST = loadrobot("universalUR5","Gravity", [0 0 -9.81]);
UR5_RST.DataFormat = 'row';
ik = inverseKinematics('RigidBodyTree',UR5_RST)
```

```
ik =
inverseKinematics with properties:

RigidBodyTree: [1x1 rigidBodyTree]
SolverAlgorithm: 'BFGSGradientProjection'
SolverParameters: [1x1 struct]
```

**%Caso 1**

```
G_prueba=[0.2 0.15 0.4 15 30 45]
```

```
G_prueba = 1x6
0.2000    0.1500    0.4000   15.0000   30.0000   45.0000
```

```
T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prue
puntoIncial = [0.1 0.1 0.1 0.1 0.1 0.1];
weights = [0.25 0.25 0.25 1 1 1];
[configSoln1,solnInfo] = ik('tool0',T_prueba,weights,puntoIncial)
```

```
configSoln1 = 1x6
0.3238   -0.0845   -2.3753    1.3377    1.9515   -0.9508
solnInfo = struct with fields:
    Iterations: 30
    NumRandomRestarts: 0
    PoseErrorNorm: 7.1548e-09
    ExitFlag: 1
    Status: 'success'
```

**%Caso 2**

```
G_prueba=[-0.3 0.2 -0.2 75 -20 0]
```

```

G_prueba = 1×6
-0.3000    0.2000   -0.2000    75.0000   -20.0000         0

T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prueba(3)]
puntoIncional = [0.1 0.1 0.1 0.1 0.1 0.1];
weights = [0.25 0.25 0.25 1 1 1];
[configSoln2,solnInfo] = ik('tool0',T_prueba,weights,puntoIncional)

configSoln2 = 1×6
2.0647   -0.4146    1.9458   -1.1666   -0.7504   -0.1809
solnInfo = struct with fields:
    Iterations: 32
    NumRandomRestarts: 0
    PoseErrorNorm: 1.3239e-08
    ExitFlag: 1
    Status: 'success'

```

### %Caso 3

```
G_prueba=[-0.35 0.3 0.1 -10 35 50]
```

```

G_prueba = 1×6
-0.3500    0.3000    0.1000   -10.0000    35.0000    50.0000

T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prueba(3)])
puntoIncional = [0.1 0.1 0.1 0.1 0.1 0.1];
weights = [0.25 0.25 0.25 1 1 1];
[configSoln3,solnInfo] = ik('tool0',T_prueba,weights,puntoIncional)

configSoln3 = 1×6
2.2895    1.2083   -2.1269   -0.9758   -4.1592   -3.1354
solnInfo = struct with fields:
    Iterations: 94
    NumRandomRestarts: 2
    PoseErrorNorm: 9.2979e-09
    ExitFlag: 1
    Status: 'success'

```

### %Caso 4

```
G_prueba=[0.3 -0.4 0 90 -45 -45]
```

```

G_prueba = 1×6
0.3000   -0.4000         0    90.0000   -45.0000   -45.0000

T_prueba=rt2tr(rpy2r(G_prueba(4),G_prueba(5),G_prueba(6),'deg'), [G_prueba(1) G_prueba(2) G_prueba(3)])
puntoIncional = [0.1 0.1 0.1 0.1 0.1 0.1];
weights = [0.25 0.25 0.25 1 1 1];
[configSoln4,solnInfo] = ik('tool0',T_prueba,weights,puntoIncional)

configSoln4 = 1×6
-0.9843   -0.4745    1.7746    1.8415   -2.9427    2.3562
solnInfo = struct with fields:
    Iterations: 32
    NumRandomRestarts: 0
    PoseErrorNorm: 2.2715e-09
    ExitFlag: 1
    Status: 'success'

```

## 8.5 Compare los métodos

Se Observa que la realizacion manual implica un proceso analítico de bastante cuidado para lograr obtener la solución, siendo susceptible a presentar errores.

En el caso del Toolbox RVC para calcular la cinemática inversa se usa la funcion *ikunc()* y en el toolbox de Matlab, RST, se utiliza la función *ik()*.

**8.6 Calcule la configuración del robot para las siguientes posturas de la herramienta. Proponga 4 posturas (x, y, z, roll, pitch, yaw)' que esten dentro del espacio de trabajo y determine la configuración del manipulador y complete la tabla.**

Usando Toolbox de Peter Corke (RVC)												
	x	y	z	roll	pitch	yaw	q1	q2	q3	q4	q5	q7
1	0,2	0,15	0,4	15	30	45	0,3238	-0,085	-2,375	1,3377	1,9515	-0,951
2	-0,3	0,2	-0,2	75	-20	0	2,0647	-0,415	1,9458	-1,167	-0,75	-0,181
3	-0,35	0,3	0,1	-10	35	50	2,2895	1,2083	-2,127	-0,976	2,1239	-3,135
4	0,3	-0,4	0	90	-45	-45	-0,984	-0,475	1,7746	1,8415	-2,943	2,3562

Tabla 6. Valores de la configuración del robot usando RVC

Usando Toolbox de Matlab (RST)												
	x	y	z	roll	pitch	yaw	q1	q2	q3	q4	q5	q7
1	0,2	0,15	0,4	15	30	45	0,3238	-0,085	-2,375	1,3377	1,9515	-0,951
2	-0,3	0,2	-0,2	75	-20	0	2,0647	-0,415	1,9458	-1,167	-0,75	-0,181
3	-0,35	0,3	0,1	-10	35	50	2,2895	1,2083	-2,127	-0,976	-4,159	-3,135
4	0,3	-0,4	0	90	-45	-45	-0,984	-0,475	1,7746	1,8415	-2,943	2,3562

Tabla 7. Valores de la configuración del robot usando RST

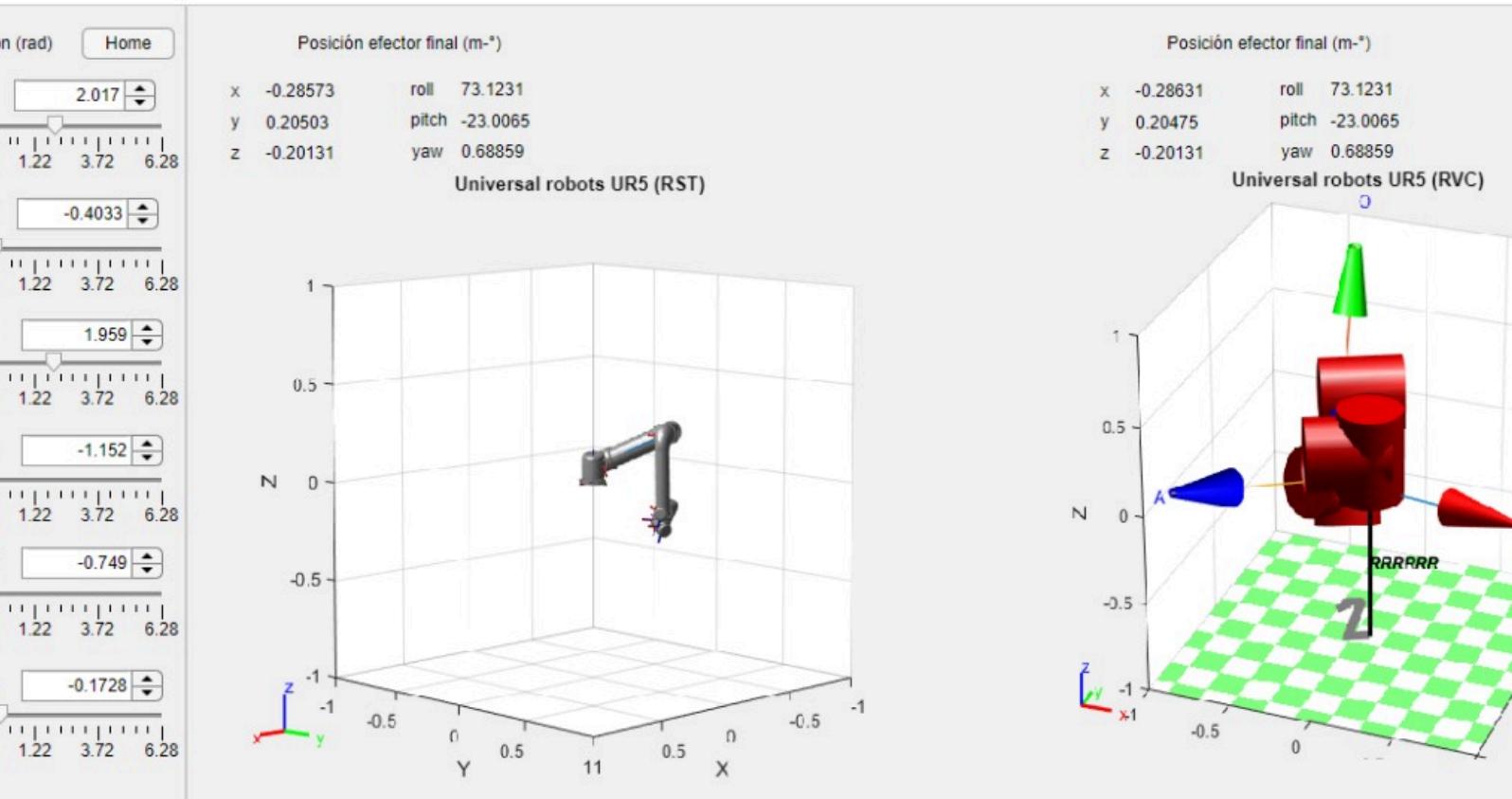
Como se observa solo la articulación q5 del vector 3 cambia entre los dos toolbox debido a la multiplicidad de soluciones que se pueden presentar.

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

**8.8 Para uno de los cuatro casos haga una captura de pantalla con los valores x y z, e incluya en el informe los resultados**

### Caso 2:

Si se compara con la tabla anterior, los resultados son similares.



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

9.1 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.

```

clear
L1 = 0.089159; L2 = 0.13585; L3 = 0.425; L4 = 0.1197;
L5 = 0.39225; L6 = 0.093; L7= 0.09465; L8= 0.0823;

L9 = 0; %L9 Es la distancia del Tool

L(1) = Link('revolute','alpha',0,'a',0,'d',L1,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha',-pi/2,'a',0,'d',L2,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha',0,'a',L3,'d', -L4,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha',0,'a',L5,'d',L6,'offset',-pi,'modified','qlim',[-2*pi 2*pi]);
L(5) = Link('revolute','alpha',pi/2,'a',0,'d',L7,'offset',0,'modified','qlim',[-2*pi 2*pi]);
L(6) = Link('revolute','alpha',-pi/2,'a',0,'d',L8,'offset',0,'modified','qlim',[-2*pi 2*pi]);

UR5_RVC = SerialLink(L, 'name', 'RRRRRR')

```

```

UR5_RVC =
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

```

j	theta	d	a	alpha	offset
1	q1	0.08916	0	0	0
2	q2	0.1358	0	-1.571	0
3	q3	-0.1197	0.425	0	0
4	q4	0.093	0.3922	0	-3.142
5	q5	0.09465	0	1.571	0
6	q6	0.0823	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 0
                  0 0 0 1      0 0 0 1
```

```
%UR5_RVC.tool = [ 0 -1 0 0; 1 0 0 0; 0 0 1 L9; 0 0 0 1]
UR5_RVC.tool = [ 1 0 0 0;
                  0 1 0 0;
                  0 0 1 L9;
                  0 0 0 1]
```

UR5\_RVC =  
RRRRRR (6 axis, RRRRRR, modDH, fastRNE)

j	theta	d	a	alpha	offset
1	q1	0.08916	0	0	0
2	q2	0.1358	0	-1.571	0
3	q3	-0.1197	0.425	0	0
4	q4	0.093	0.3922	0	-3.142
5	q5	0.09465	0	1.571	0
6	q6	0.0823	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 0
                  0 0 0 1      0 0 0 1
```

% Vector de valores de prueba de Q, para obtener una matriz Jacobiana numerica

```
q=[0 0 0 0 0 0]
```

```
q = 1x6
  0     0     0     0     0     0
```

% Matrices de Transformacion Homogenea

```
A_01=L(1).A(q(1)); A_12=L(2).A(q(2)); A_23=L(3).A(q(3)); A_34=L(4).A(q(4)); A_45=L(5).A(q(5));
A_6T=UR5_RVC.tool;
```

% Matriz de Transformacion  
A\_0T=fkine(UR5\_RVC,q)

```
A_0T = 4x4
-1.0000    0.0000      0    0.8173
-0.0000   -0.0000    1.0000    0.1915
```

```

0.0000    1.0000    0.0000   -0.0055
      0          0          0       1.0000

```

Ahora se Calcula el Jacobiano de Manera Directa

```

% Extracción de las matrices de rotacion

z01 = A_01(1:3,3); %Articulación 1

A_02 = A_01*A_12;
z02 = A_02(1:3,3); %Articulación 2

A_03 = A_02*A_23;
z03 = A_03(1:3,3); %Articulación 3

A_04 = A_03*A_34;
z04 = A_04(1:3,3); %Articulación 4

A_05 = A_04*A_45;
z05 = A_05(1:3,3); %Articulación 5

A_06 = A_05*A_56;
z06 = A_06(1:3,3); %Articulación 6

% Cálculo del Vector de desplazamiento entre Articulaciones

p6t = A_0T(1:3,4) - A_06(1:3,4);
p5t = A_0T(1:3,4) - A_05(1:3,4);
p4t = A_0T(1:3,4) - A_04(1:3,4);
p3t = A_0T(1:3,4) - A_03(1:3,4);
p2t = A_0T(1:3,4) - A_02(1:3,4);
p1t = A_0T(1:3,4) - A_01(1:3,4);

% Cálculo de las Columnas del Jacobiano

% J1 =[ (skew(z01) * p1t); z01]; % Rotacional
% J2 =[ (skew(z02) * p2t); z02]; % Rotacional
% J3 =[ (skew(z03) * p3t); z03]; % Rotacional
% J4 =[ (skew(z04) * p4t); z04]; % Rotacional
% J5 =[ (skew(z05) * p5t); z05]; % Rotacional
% J6 =[ (skew(z06) * p6t); z06]; % Rotacional

J1 =[cross(z01,p1t); z01]; % Rotacional
J2 =[cross(z02,p2t); z02]; % Rotacional
J3 =[cross(z03,p3t); z03]; % Rotacional
J4 =[cross(z04,p4t); z04]; % Rotacional
J5 =[cross(z05,p5t); z05]; % Rotacional
J6 =[cross(z06,p6t); z06]; % Rotacional

Jgeo = [J1, J2, J3, J4, J5, J6]

```

```

Jgeo = 6x6
-0.1915   -0.0947   -0.0947   -0.0946   0.0823       0
  0.8173     0.0000     0.0000         0   0.0000       0
      0   -0.8173   -0.3923         0   -0.0000       0

```

```

0      0      0      0     -0.0000      0
0    1.0000   1.0000   1.0000     0.0000   1.0000
1.0000   0.0000   0.0000   0.0000    -1.0000   0.0000

```

```
Jtool = UR5_RVC.jacob0(q)
```

```
Jtool = 6x6
-0.1915  -0.0946  -0.0946  -0.0946  0.0823      0
0.8173   0.0000   0.0000  -0.0000  0.0000      0
0  -0.8173  -0.3922  0.0000  -0.0000      0
0      0      0      0  -0.0000      0
0    1.0000   1.0000   1.0000     0.0000   1.0000
1.0000   0.0000   0.0000   0.0000    -1.0000   0.0000
```

**9.2 Para la postura 1 de la Tabla 2 obtenga las velocidades de articulacion para V=[100,200,50]' mm/s y w=[5,10,-5]' rad/s**

```
q=[0.3238 -0.085 -2.375 1.3377 1.9515 -0.951]
```

```
q = 1x6
0.3238  -0.0850  -2.3750  1.3377  1.9515  -0.9510
```

% Matrices de Transformacion Homogenea

```
A_01=L(1).A(q(1)); A_12=L(2).A(q(2)); A_23=L(3).A(q(3)); A_34=L(4).A(q(4)); A_45=L(5).A(q(5));
A_6T=UR5_RVC.tool;
```

% Matriz de Transformacion

```
A_0T=fkine(UR5_RVC,q)
```

```
A_0T = 4x4
0.6125  -0.6123  0.4999  0.1999
0.7744  0.5916  -0.2242  0.1500
-0.1585  0.5244  0.8366  0.4002
0      0      0      1.0000
```

% Extraccion de las matrices de rotacion

```
z01 = A_01(1:3,3); %Articulación 1
```

```
A_02 = A_01*A_12;
z02 = A_02(1:3,3); %Articulación 2
```

```
A_03 = A_02*A_23;
z03 = A_03(1:3,3); %Articulación 3
```

```
A_04 = A_03*A_34;
z04 = A_04(1:3,3); %Articulación 4
```

```
A_05 = A_04*A_45;
z05 = A_05(1:3,3); %Articulación 5
```

```
A_06 = A_05*A_56;
z06 = A_06(1:3,3); %Articulación 6
```

```
% Cálculo del Vector de desplazamiento entre Articulaciones
```

```
p6t = A_0T(1:3,4) - A_06(1:3,4);  
p5t = A_0T(1:3,4) - A_05(1:3,4);  
p4t = A_0T(1:3,4) - A_04(1:3,4);  
p3t = A_0T(1:3,4) - A_03(1:3,4);  
p2t = A_0T(1:3,4) - A_02(1:3,4);  
p1t = A_0T(1:3,4) - A_01(1:3,4);
```

```
% Cálculo de las Columnas del Jacobiano
```

```
% J1 =[ (skew(z01) * p16); z01]; % Rotacional  
% J2 =[ (skew(z02) * p26); z02]; % Rotacional  
% J3 =[ (skew(z03) * p36); z03]; % Rotacional  
% J4 =[ (skew(z04) * p46); z04]; % Rotacional  
% J5 =[ (skew(z05) * p56); z05]; % Rotacional  
% J6 =[ (skew(z06) * p6t); z06]; % Rotacional
```

```
J1 =[cross(z01,p1t); z01]; % Rotacional  
J2 =[cross(z02,p2t); z02]; % Rotacional  
J3 =[cross(z03,p3t); z03]; % Rotacional  
J4 =[cross(z04,p4t); z04]; % Rotacional  
J5 =[cross(z05,p5t); z05]; % Rotacional  
J6 =[cross(z06,p6t); z06]; % Rotacional
```

```
Jgeo = [J1, J2, J3, J4, J5, J6]
```

```
Jgeo = 6x6  
-0.1500 0.2949 0.2607 0.0264 0.0117 0  
0.1999 0.0990 0.0875 0.0088 -0.0767 0  
0 -0.2373 0.1862 -0.1184 -0.0276 0  
0 -0.3182 -0.3182 -0.3182 0.8543 0.4999  
0 0.9480 0.9480 0.9480 0.2867 -0.2242  
1.0000 0.0000 0.0000 0.0000 -0.4336 0.8366
```

```
Jtool = UR5_RVC.jacob0(q)
```

```
Jtool = 6x6  
-0.1500 0.2949 0.2607 0.0264 0.0117 0  
0.1999 0.0990 0.0875 0.0088 -0.0767 0  
-0.0000 -0.2373 0.1862 -0.1184 -0.0276 0  
0 -0.3182 -0.3182 -0.3182 0.8543 0.4999  
0.0000 0.9480 0.9480 0.9480 0.2867 -0.2242  
1.0000 0 0 0 -0.4336 0.8366
```

```
% Para la pose de prueba 1, teniendo en cuenta las velocidades Lineales
```

```
V_Gen = [0.1 0.2 0.05 5 10 -5]' % Vector de velocidades generalizadas
```

```
V_Gen = 6x1  
0.1000  
0.2000  
0.0500  
5.0000  
10.0000  
-5.0000
```

```
%Velocidades e articulación
q_velocidades = Jgeo^(-1)*V_Gen
```

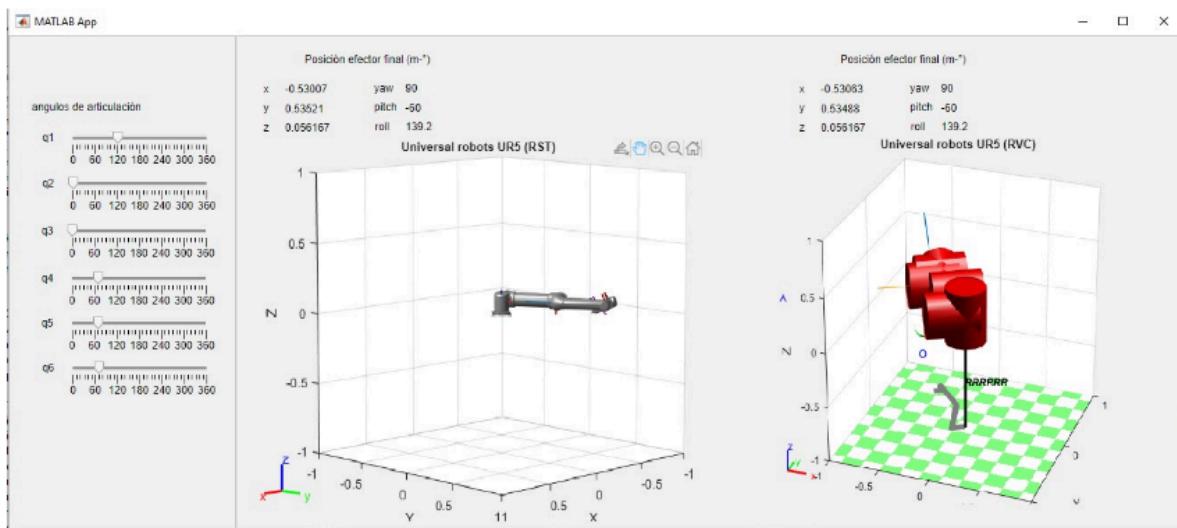
```
q_velocidades = 6x1
4.2562
-1.0097
3.0829
3.8529
11.1520
-5.2840
```

```
q_tool = 6x1
4.2562
-1.0097
3.0829
3.8529
11.1520
-5.2840
```

## INTEGRACIÓN

### GUI.

La interfaz de usuario creada usando la herramienta de App Designer de Matlab. En la imagen se puede observar la distribución de los elementos. En primer lugar cuenta con 6 controles tipo slider que permiten mover las articulaciones del Robot UR5. También es posible visualizar el modelo del robot por los dos métodos resultantes de usar las toolbox requeridas; al lado izquierdo se encuentra la representación por RST y al derecho, por RVC. Por ultimo, justo encima de la visualización por cada método, se puede encontrar la posición y orientación del efecto final.



### Iniciar el programa.

Cuando el programa se inicializa, aparecen todos los elementos, y se visualiza el robot en posición de Home para los dos métodos. Todas las articulaciones comienzan en 0° y la posición final del efecto final corresponde

a [0.81725 0.19145 -0.0055 -90 0 180]; las primeras tres corresponden a la posición XYZ en metros y las ultimas tres son la orientación en grados. En los dos métodos, la posición del efecto final es la misma con un error máximo de 0.34%.

Para poder encontrar la posición del efecto final, se utiliza la cinemática directa, usando las funciones correspondientes a cada caso y resolviendo con la configuración de home. En el caso del Toolbox RVC para calcular la cinemática directa se usa la función fkine() y en el toolbox de Matlab, RST, se utiliza la función getTransform().

### Movimiento de las articulaciones.

Existen 6 sliders que controlan cada uno de los movimientos articulares del robot, en grados. Cuando se detecta que se ha efectuado un cambio en el valor de cualquiera de los sliders, se llama una función callback, mediante la cual se toma el nuevo valor del slider, se genera una conversión de unidades de grados a radianes y se cambia el valor en ambas formas de visualización del robot, y se busca nuevamente la posición del efecto final, para finalmente mostrar este valor en los Labels creados para este fin. El valor actual de los sliders se almacena en variables globales, por lo cual pueden ser accedidas en cualquier función según sea necesario.

### Viapoints, simulación del seguimiento de las trayectorias.

Para generar la simulación, se utiliza el RST de Matlab. Primero se carga el robot y se definen parámetros como la cantidad de articulaciones en el robot, en este caso 6, los ángulos de Euler necesarios para que el eje z del efecto final quede perpendicular al plano en el que esta construida la trayectoria durante toda la simulación. Los puntos creados anteriormente, junto con esta orientación, conforman los viapoints.

Tambien se definen los tiempos correspondientes a una velocidad de herramienta

```
%% script que saca nuestros puntos
clc
clear
close

% parametro t, separacion entre cada punto
t=0.019048;

K = [0.59428, 0.13333, 0.40572]; %vertices del cuadrado
M = [0.40572, -0.13333, 0.59428];
E = [0.40572, 0.13333, 0.59428];
I = [0.59428, -0.13333, 0.40572];

J = [0.5, 0.13333, 0.5]; % Punto medio EK, centro del circulo

%segmento entre 2 puntos KI = I-K (15 puntos)
punto=[0.59428, 0.13333, 0.40572]';
for i=1:14
    punto (:,i+1)=[K(1),(K(2)-i*t),K(3)];
end

%segmento ENTRE 2 PUNTOS ME IM=M-I (14 puntos)
for i=1:14
    punto (:,i+15)=[(I(1)-i*t/sqrt(2)),(I(2)),(I(3)+i*t/sqrt(2))];
```

```

end

%segmento ENTRE 2 PUNTOS ME ME=E-M (14 puntos)
for i=1:14
    punto (:,i+29)=[M(1),(M(2)+i*t),M(3)];
end

%Semicirculo centro radio JK, EMPIEZA EN E, TERMINA EN K (22 puntos)
j=-90;
for i=1:22
    punto (:,i+43)= [J(1)+(J(1)-E(1))*sind(t+j),J(2)+J(2)*cosd(t+j), J(3)+(J(3)-E(3))*sind(t+j)]
    j=j+8.2;
end
p = plot3(punto(1,:),punto(2,:),punto(3,:),'.');

ur5 = loadrobot("universalUR5",'Gravity',[0 0 -9.81]);
ur5.DataFormat = 'row';
showdetails(ur5)

eeName = "tool0"; % Define end-effector body name

numJoints = 6; % Define the number of joints in the manipulator

eeOrientation = [pi/2, 0, pi/3]; % The Euler angles for the desired ee orientation

clear wayPoints % Clear previous waypoints and begin building wayPoint array

% create array
for i = 1:65
    waypt1 = [punto(:,i)',eeOrientation];
    wayPoints(:,i) = waypt1';
end

q0 = zeros(numJoints,1)';

%Specify the trajectory time step and approximate desired tool speed.
toolSpeed = 0.5; % m/s
distance = norm(punto(:,1)'-punto(:,2)');
initTime = 0;
timeStep = (distance/toolSpeed) - initTime; % seconds
finalTime = timeStep*64;
trajTimes = initTime:timeStep:finalTime;
timeInterval = [trajTimes(1); trajTimes(end)];

% Define a [1x6] vector of relative weights on the orientation and
% position error for the inverse kinematics solver.
weights = ones(1,6);

% Transform the first waypoint to a Homogenous Transform Matrix for initialization
initTargetPose = eul2tfm(wayPoints(4:6,1)');
initTargetPose(1:3,end) = wayPoints(1:3,1);%> wayPoints(1:3,1)

% Solve for q0 such that the manipulator begins at the first waypoint

```

```

ik = inverseKinematics('RigidBodyTree',ur5);
q0 = ik(eeName,initTargetPose,weights,q0);

%Show the initial configuration of the robot.

show(ur5,q0,'PreservePlot',false,'Frames','off');
hold on
axis([-1 1 -1 1 -0.1 1.5]);
show(ur5,q0,'PreservePlot',false,'Frames','off');
plot3(punto(1,i),punto(2,i),punto(3,i),'b.','MarkerSize',20)
drawnow;
%Visualize the task-space trajectory. Iterate through the stateTask states and interpolate base

for i=2:length(trajTimes)
    % Current time
    tNow= trajTimes(i);

    TargetPose = eul2tfm(wayPoints(4:6,i)');
    TargetPose(1:3,end) = wayPoints(1:3,i);%> wayPoints(1:3,i)

    % Solve for q
    q0(i,:) = ik(eeName,TargetPose,weights,q0(i-1,:));
    show(ur5,q0(i,:), 'PreservePlot', false, 'Frames', 'off');
    plot3(punto(1,i),punto(2,i),punto(3,i),'b.','MarkerSize',20)
    drawnow;
end

```

En las siguientes imágenes se pueden ver la velocidad y el cambio de ángulo por articulación para la trayectoria del robot, se puede ver claramente el cambio que existe cuando llega al vértice y se genera el cambio de dirección en la herramienta.

```

%% IMAGENES DE CAMBIO ARTICULACION

plot(trajTimes(:,q0(:,1)',trajTimes(:,q0(:,2)',trajTimes(:,q0(:,3)',trajTimes(:,q0(:,4)',trajTimes(:,q0(:,5)',trajTimes(:,q0(:,6)'
title('Variacion de angulo de articulacion, para la trayectoria escogida')
xlabel('tiempo')
ylabel('angulo (rad)')
legend({'q1','q2','q3','q4','q5','q6'},'Location','northeast')

%% IMAGENES DE VELOCIDAD ARTICULACION
for i=1:64
    vel(i,:) = (q0(i+1,:)-q0(i,:))/timeStep;
end

plot(trajTimes(1:64),vel(:,1)',trajTimes(1:64),vel(:,2)',trajTimes(1:64),vel(:,3)',trajTimes(1:64),vel(:,4)',trajTimes(1:64),vel(:,5)',trajTimes(1:64),vel(:,6)'
title('Velocidad en rad/s, para la trayectoria escogida')
xlabel('tiempo (s)')
ylabel('velocidad (rad/s)')
legend({'vel_q1','vel_q2','vel_q3','vel_q4','vel_q5','vel_q6'},'Location','northeast')

```

