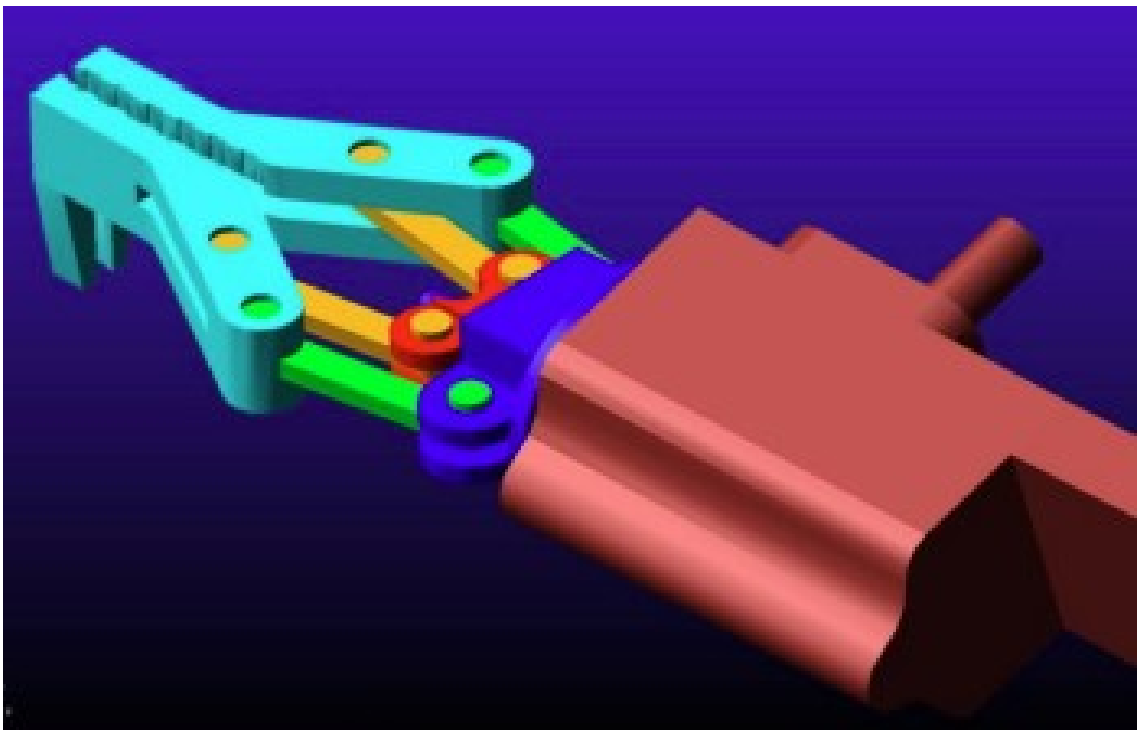
Automatización y robótica

Práctica 2: Modelado y simulación



VadymFormanyuk
vf13@alu.ua.es

4 de mayo de 2023

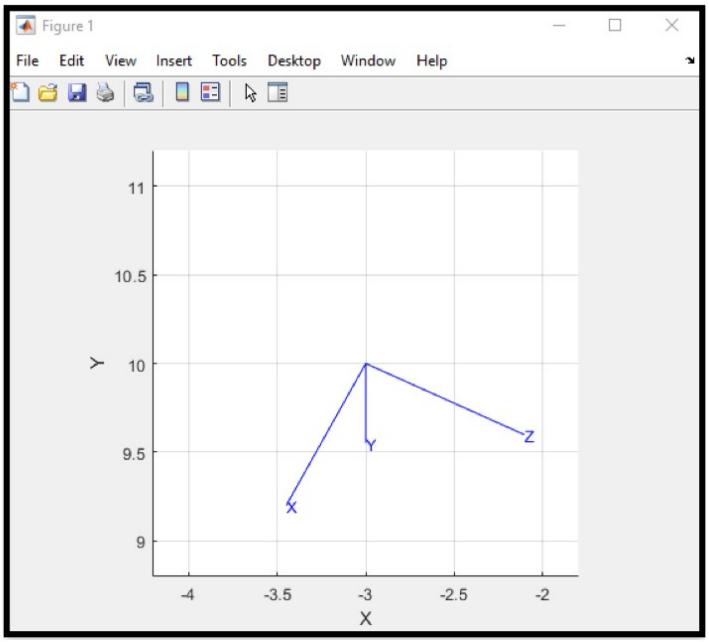
Índice

1. Ejercicio 1

Mediante las funciones de las herramientas matemáticas, obtener la matriz de transformación y graficar el resultado que representa las siguientes transformaciones sobre un sistema $OXYZ$ fijo de referencia: traslación de $(-3, 10, 10)$; giro de $-90°$ sobre el eje $O'U$ del sistema trasladado y giro de $90°$ sobre el eje $O'V'$ del sistema girado.

```
>> tr = transl(-3,10,10)*trotx(-90)*troty(90);
>> trplot(tr);
```

```
>> tr = transl(-3,10,10)*trotx(-90)*troty(90);
>> tr

tr =

   -0.4481         0    0.8940   -3.0000
   -0.7992   -0.4481   -0.4006   10.0000
    0.4006   -0.8940    0.2008   10.0000
         0         0         0    1.0000
```

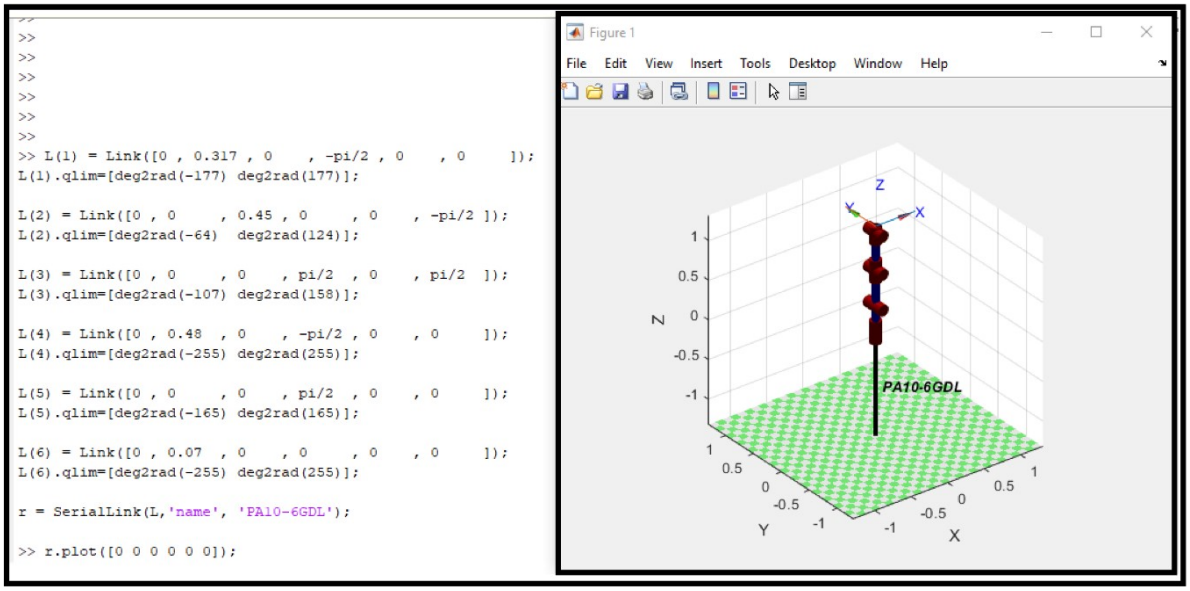


2. Ejercicio 2

Modelado del robot PA10 de 6GDL a partir de la siguiente tabla de sus parámetros DH estándar y los límites articulares. Para introducir los límites articulares y el offset de la articulación, mira en la web siguiente o teclea el comando "help SerialLink" http://www.petercorke.com/RTB/r9/html/Link.html.

Transformación	Θ	d	a	α	Límite q(º)	Offset
0 → 1 $^0A_1$	q_1	0.317	0	-pi/2	[-177,177]	0
1 → 2 $^1A_2$	q_2	0	0.45	0	[-64,124]	-pi/2
2 → 3 $^2A_3$	q_3	0	0	pi/2	[-107,158]	pi/2
3 → 4 $^3A_4$	q_4	0.48	0	-pi/2	[-255,255]	0
4 → 5 $^4A_5$	q_5	0	0	pi/2	[-165,165]	0
5 → 6 $^5A_6$	q_6	0.07	0	0	[-255,255]	0

```
>> L(1) = Link([0 , 0.317 , 0 , -pi/2 , 0 , 0 ]);
L(1).qlim=[deg2rad(-177) deg2rad(177)];

L(2) = Link([0 , 0 , 0.45 , 0 , 0 , -pi/2 ]);
L(2).qlim=[deg2rad(-64) deg2rad(124)];

L(3) = Link([0 , 0 , 0 , pi/2 , 0 , pi/2 ]);
L(3).qlim=[deg2rad(-107) deg2rad(158)];

L(4) = Link([0 , 0.48 , 0 , -pi/2 , 0 , 0 ]);
L(4).qlim=[deg2rad(-255) deg2rad(255)];

L(5) = Link([0 , 0 , 0 , pi/2 , 0 , 0 ]);
L(5).qlim=[deg2rad(-165) deg2rad(165)];

L(6) = Link([0 , 0.07 , 0 , 0 , 0 , 0 ]);
L(6).qlim=[deg2rad(-255) deg2rad(255)];

r = SerialLink(L,'name', 'PA10-6GDL');
>> r.plot([0 0 0 0 0 0]);
```



3. Ejercicio 3

Definir las siguientes posiciones articulares para el PA10 (las posiciones se indican en grados, pero en Matlab hay que introducirlas en radianes), calcular la cinemática directa (matriz T) para cada uno de ellos y realizar un plot en esa posición.

Posición de home: q_h = [0, 0, 0, 0, 0, 0].
Posición de escape: q_e = [0, 30, 90, 0, 60, 0].
Posición de seguridad: q_s = [0, 45, 90, 0, -45, 0].
Posición q_1 = [0, 45, 45, 0, 90, 0].
Posición q_2 = [20, 90, 45, -22.5, 60, 0].

Nota: Se han cambiado los grados de las posiciones a radiantes (por ejemplo, 20º son 0,3491 radiantes)

Nota: Para calcular la cinemática directa, se utiliza el método "fkine"del objeto SerialLink en MATLAB. Este método toma como entrada un vector de ángulos articulares y devuelve una matriz de transformación homogénea que describe la posición y orientación del extremo del robot.

```
>> L(1) = Link([0 , 0.317 , 0 , -pi/2 , 0 , 0 ]);
L(1).qlim=[deg2rad(-177) deg2rad(177)];

L(2) = Link([0 , 0 , 0.45 , 0 , 0 , -pi/2 ]);
L(2).qlim=[deg2rad(-64) deg2rad(124)];

L(3) = Link([0 , 0 , 0 , pi/2 , 0 , pi/2 ]);
L(3).qlim=[deg2rad(-107) deg2rad(158)];

L(4) = Link([0 , 0.48 , 0 , -pi/2 , 0 , 0 ]);
L(4).qlim=[deg2rad(-255) deg2rad(255)];

L(5) = Link([0 , 0 , 0 , pi/2 , 0 , 0 ]);
L(5).qlim=[deg2rad(-165) deg2rad(165)];

L(6) = Link([0 , 0.07 , 0 , 0 , 0 , 0 ]);
L(6).qlim=[deg2rad(-255) deg2rad(255)];

r = SerialLink(L,'name', 'PA10-6GDL');
>>
>> figure(1);
q_h = [0 0 0 0 0 0];
q_h_T = r.fkine(q_h); %cinematica directa
r.plot(q_h);
>> figure(2);
q_e = [0 0.5236 1.571 0 1.0472 0];
q_e_T = r.fkine(q_e); %cinematica directa
r.plot(q_e);
>> figure(3);
q_s = [0 0.7854 1.571 0 -0.7854 0];
q_s_T = r.fkine(q_s); %cinematica directa
r.plot(q_s);
>> figure(4);
q_1 = [0 0.7854 0.7854 0 1.571 0];
q_1_T = r.fkine(q_1); %cinematica directa
r.plot(q_1);
>> figure(5);
q_2 = [0.3491 1.571 0.7854 -3.927 1.047 0];
q_2_T = r.fkine(q_2); %cinematica directa
r.plot(q_2);
>>
```

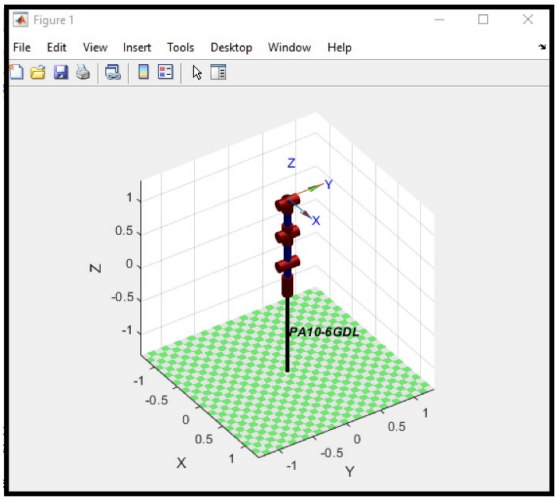


Figure 1
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

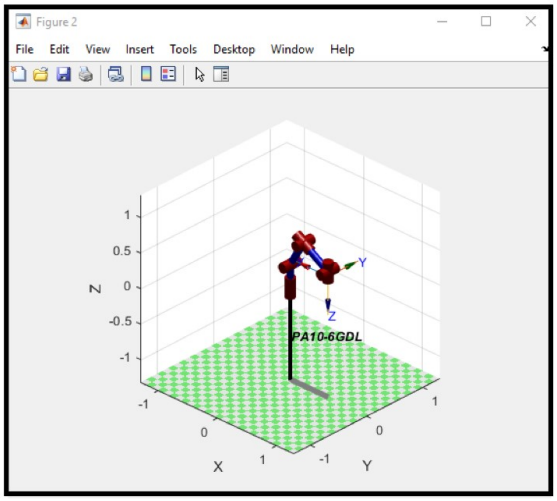


Figure 2
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

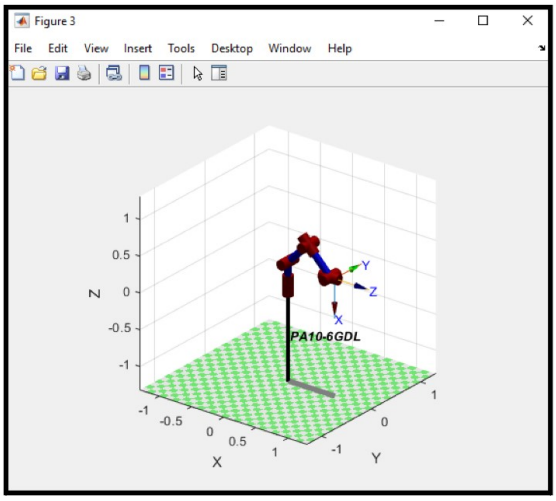


Figure 3
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

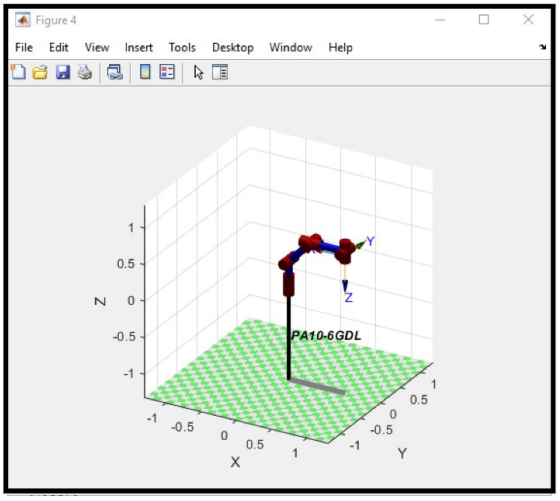


Figure 4
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

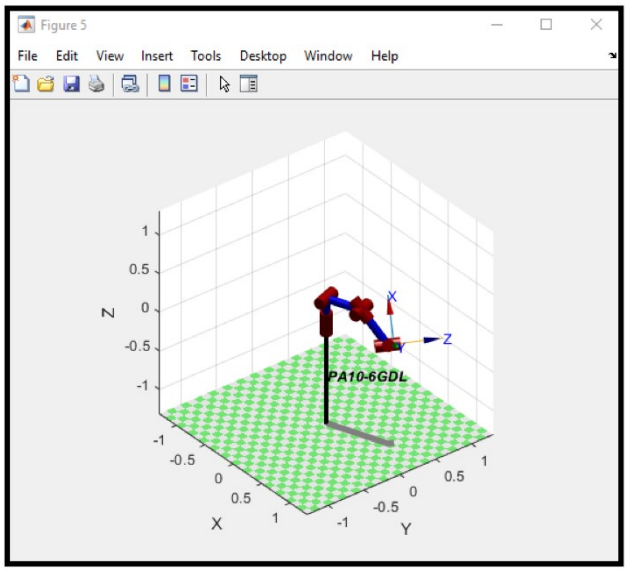


Figure 5
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

4. Ejercicio 4

Realizar la resolución de la cinemática inversa para el resto de posiciones del PA10 (qe, qs, q1, q2) siguiendo el mismo procedimiento que en el ejemplo mostrado utilizando las funciones *ikine6s* **e** *ikunc***. Para más información de los métodos, se puede acceder mediante el comando "help ikine6s" y "help ikunc" en Matlab.**

Nota: El método 'ikine6s' resuelve la cinemática inversa sin restricciones adicionales, mientras que 'ikunc' es un método iterativo que utiliza la función de Jacobiano inverso para encontrar una solución que cumpla con las restricciones especificadas.

```
>> L(1) = Link([0 , 0.317 , 0 , -pi/2 , 0 , 0 ]);
L(1).qlim=[deg2rad(-177) deg2rad(177)];

L(2) = Link([0 , 0 , 0.45 , 0 , 0 , -pi/2 ]);
L(2).qlim=[deg2rad(-64) deg2rad(124)];

L(3) = Link([0 , 0 , 0 , pi/2 , 0 , pi/2 ]);
L(3).qlim=[deg2rad(-107) deg2rad(158)];

L(4) = Link([0 , 0.48 , 0 , -pi/2 , 0 , 0 ]);
L(4).qlim=[deg2rad(-255) deg2rad(255)];

L(5) = Link([0 , 0 , 0 , pi/2 , 0 , 0 ]);
L(5).qlim=[deg2rad(-165) deg2rad(165)];

L(6) = Link([0 , 0.07 , 0 , 0 , 0 , 0 ]);
L(6).qlim=[deg2rad(-255) deg2rad(255)];

r = SerialLink(L,'name', 'PA10-6GDL');
>> figure(1);
q_h = [0 0 0 0 0 0];
q_h_T = r.fkine(q_h); %cinematica directa
q_h_T_inv = r.ikine6s(q_h_T); %cinematica inversa
r.plot(q_h_T_inv);
>> figure(2);
q_e = [0 0.5236 1.571 0 1.0472 0];
q_e_T = r.fkine(q_e); %cinematica directa
q_e_T_inv = r.ikine6s(q_e_T); %cinematica inversa
r.plot(q_e_T_inv);
>> figure(3);
q_s = [0 0.7854 1.571 0 -0.7854 0];
q_s_T = r.fkine(q_s); %cinematica directa
q_s_T_inv = r.ikine6s(q_s_T); %cinematica inversa
r.plot(q_s_T_inv);
>> figure(4);
q_1 = [0 0.7854 0.7854 0 1.571 0];
q_1_T = r.fkine(q_1); %cinematica directa
q_1_T_inv = r.ikine6s(q_1_T); %cinematica inversa
r.plot(q_1_T_inv);
>> figure(5);
q_2 = [0.3491 1.571 0.7854 -3.927 1.047 0];
q_2_T = r.fkine(q_2); %cinematica directa
q_2_T_inv = r.ikine6s(q_2_T); %cinematica inversa
r.plot(q_2_T_inv);
```

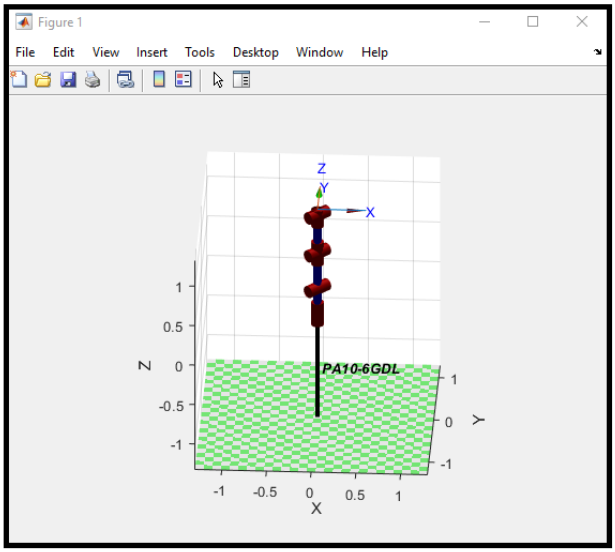


Figure 1
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

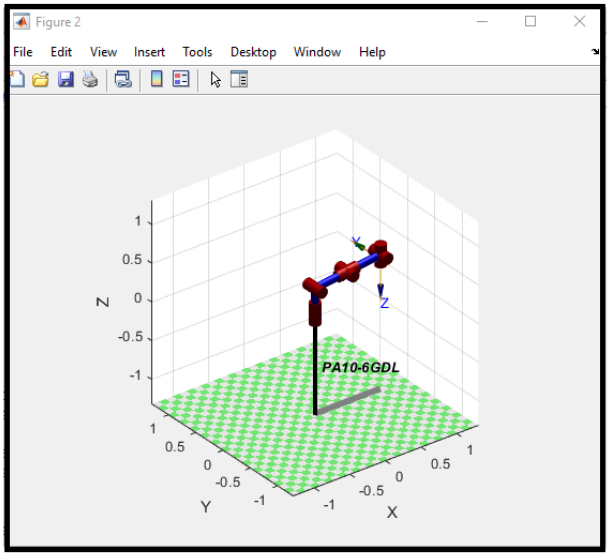


Figure 2
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

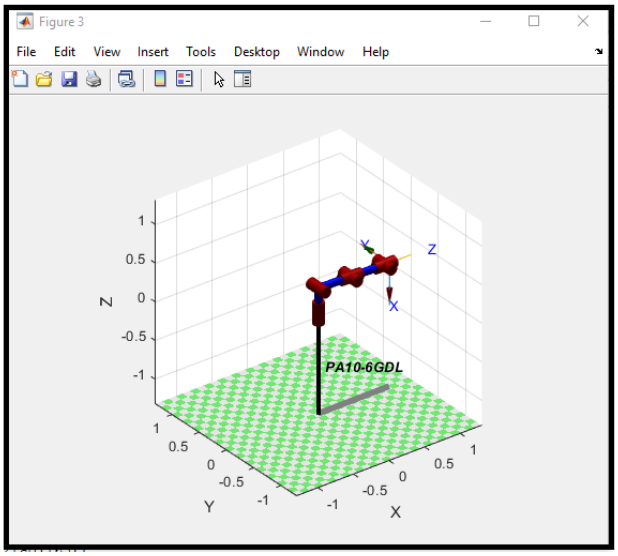


Figure 3
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

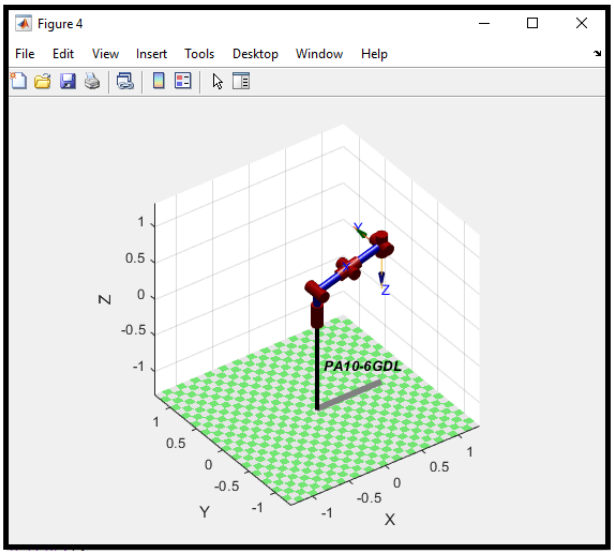


Figure 4
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

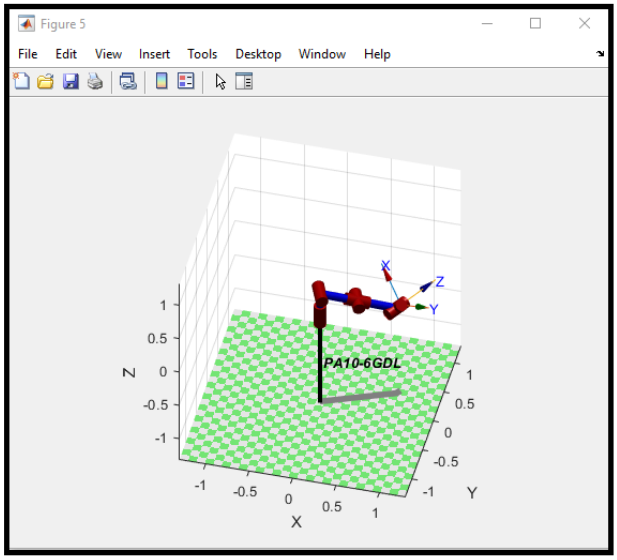


Figure 5
File Edit View Insert Tools Desktop Window Help
PA10-6GDL

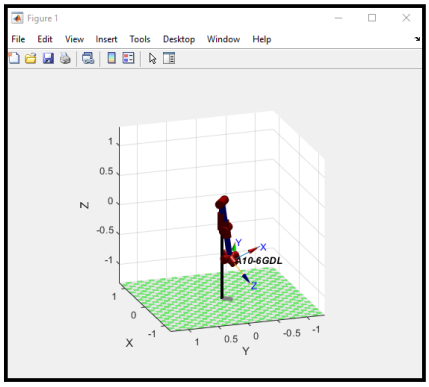
5. Ejercicio 5

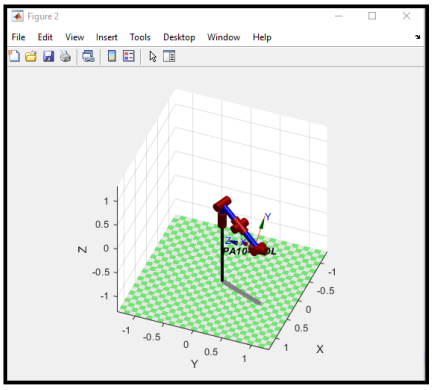
Evalúa al robot PA10 y al robot planar en otras posiciones al límite de su espacio de trabajo o donde existan alineaciones de ejes (puedes emplear la función *rand* para probar diferentes posiciones). Para el robot planar, sólo ten en cuenta las dos primeras filas y la última de la matriz Jacobiana, ya que el resultado no es una matriz cuadrada, y sólo es necesario evaluar el espacio cartesiano plano y uno de los vectores de orientación del robot en el plano (el robot planar sólo puede posicionarse y orientarse en el plano).

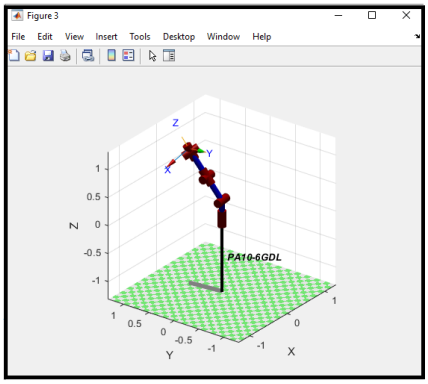
Nota: En este ejemplo, se definió el robot planar con dos eslabones de longitud 1 unidad cada uno y se evaluó en tres posiciones al límite de su espacio de trabajo y donde existen alineaciones de ejes. Luego se calculó la matriz Jacobiana para cada posición y se evaluó el espacio cartesiano plano y un vector de orientación en el plano. Como se mencionó en el enunciado, se tuvieron en cuenta solo las dos primeras filas y la última de la matriz Jacobiana, ya que el robot planar solo puede posicionarse y orientarse en el plano.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%ROBOT PA10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L(1) = Link([0 , 0.317 , 0 , -pi/2 , 0 , 0 ]);
L(1).qlim=[deg2rad(-177) deg2rad(177)];

L(2) = Link([0 , 0 , 0.45 , 0 , 0 , -pi/2 ]);
L(2).qlim=[deg2rad(-64) deg2rad(124)];

L(3) = Link([0 , 0 , 0 , pi/2 , 0 , pi/2 ]);
L(3).qlim=[deg2rad(-107) deg2rad(158)];

L(4) = Link([0 , 0.48 , 0 , -pi/2 , 0 , 0 ]);
L(4).qlim=[deg2rad(-255) deg2rad(255)];

L(5) = Link([0 , 0 , 0 , pi/2 , 0 , 0 ]);
L(5).qlim=[deg2rad(-165) deg2rad(165)];

L(6) = Link([0 , 0.07 , 0 , 0 , 0 , 0 ]);
L(6).qlim=[deg2rad(-255) deg2rad(255)];

r = SerialLink(L,'name', 'PA10-6GDL');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%OTRAS POSICIONES DE PA10 AL L MITE DE SU ESPACIO DE TRABAJO %%%%%%%%
figure(1);
PA_rand_pos_1 = [deg2rad(randi([-177, 177])), deg2rad(randi([-64, 124])), deg2rad(randi([-107, 158])),
    deg2rad(randi([-255, 255])), deg2rad(randi([-165, 165])), deg2rad(randi([-255, 255]))];
PA_rand_pos_1_T = r.fkine(PA_rand_pos_1); %cinematica directa
PA_rand_pos_1_T_inv = r.ikine6s(PA_rand_pos_1_T); %cinematica inversa
r.plot(PA_rand_pos_1_T_inv);

figure(2);
PA_rand_pos_2 = [deg2rad(randi([-177, 177])), deg2rad(randi([-64, 124])), deg2rad(randi([-107, 158])),
    deg2rad(randi([-255, 255])), deg2rad(randi([-165, 165])), deg2rad(randi([-255, 255]))];
PA_rand_pos_2_T = r.fkine(PA_rand_pos_2); %cinematica directa
PA_rand_pos_2_T_inv = r.ikine6s(PA_rand_pos_2_T); %cinematica inversa
r.plot(PA_rand_pos_2_T_inv);

figure(3);
PA_rand_pos_3 = [deg2rad(randi([-177, 177])), deg2rad(randi([-64, 124])), deg2rad(randi([-107, 158])),
    deg2rad(randi([-255, 255])), deg2rad(randi([-165, 165])), deg2rad(randi([-255, 255]))];
PA_rand_pos_3_T = r.fkine(PA_rand_pos_3); %cinematica directa
PA_rand_pos_3_T_inv = r.ikine6s(PA_rand_pos_3_T); %cinematica inversa
r.plot(PA_rand_pos_3_T_inv);
```

```
45
46 %%%%%%%%%%%%%%ROBOT PLANAR%%%%%%%%%
47 L(1) = Link([0, 0, 0, 0, 0]);
48 L(1).qlim = [deg2rad(-180), deg2rad(180)];
49
50 L(2) = Link([0, 0, 1, 0, 0]);
51 L(2).qlim = [deg2rad(-180), deg2rad(180)];
52
53 planar_robot = SerialLink([L(1) L(2)], 'name', 'Robot planar');
54 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55
56 % Evaluar el robot en tres posiciones limite
57 q1 = [0, pi/2];
58 q2 = [-pi/2, pi/3];
59 q3 = [pi/2, -pi/2];
60
61 % Calcular la matriz Jacobiana en cada posicion
62 J1 = planar_robot.jacob0(q1);
63 J2 = planar_robot.jacob0(q2);
64 J3 = planar_robot.jacob0(q3);
65
66 % Evaluar el espacio cartesiano plano y un vector de orientacion en el plano para cada
       posici n
67 pos1 = planar_robot.fkine(q1).t;
68 pos2 = planar_robot.fkine(q2).t;
69 pos3 = planar_robot.fkine(q3).t;
70
71 ori1 = J1([1 2 6], 1);
72 ori2 = J2([1 2 6], 1);
73 ori3 = J3([1 2 6], 1);
74
75 % Imprimir los resultados
76 disp("Posiciones l mite del robot planar:");
77 disp("q1 = ");
78 disp(q1);
79 disp("q2 = ");
80 disp(q2);
81 disp("q3 = ");
82 disp(q3);
83
84 disp("Posiciones cartesianas y vectores de orientaci n en el plano:");
85 disp("Posici n 1:");
86 disp(pos1);
87 disp(ori1);
88 disp("Posici n 2:");
89 disp(pos2);
90 disp(ori2);
91 disp("Posici n 3:");
92 disp(pos3);
93 disp(ori3);
```








```
Posiciones límite del robot planar:
q1 =
         0    1.5708

q2 =
   -1.5708    1.0472

q3 =
    1.5708   -1.5708

Posiciones cartesianas y vectores de orientación en el plano:
Posición 1:
    0.0000
    1.0000
         0

   -1.0000
    0.0000
    1.0000

Posición 2:
    0.8660
   -0.5000
         0

    0.5000
    0.8660
    1.0000

Posición 3:
     1
     0
     0

     0
     1
     1
```

6. Ejercicio 6

Calcula los pares articulares del resto de posiciones del robot PA10(qs, q1 y q2) utilizando el comando robot.rne(q0, v0, a0).

Nota: por defecto, la RT calcula la dinámica inversa de un robot empleando un método rápido llamado fastRNE. Si el comando robot.rne no funcionara correctamente (da un error que cierra Matlab), será necesario cambiarlo mediante la opción robot.fast al método slowRNE. Para ello se pone el valor de robot.fast a 0 (robot.fast=0).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = DynamicParams(loadPA10Params());
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

q_s = [0 0.7854 1.571 0 -0.7854 0];
q_1 = [0 0.7854 0.7854 0 1.571 0];
q_2 = [0.3491 1.571 0.7854 -3.927 1.047 0];

qs_rne = r.rne(q_s, [0 0 0 0 0 0], [0 0 0 0 0 0]);
q1_rne = r.rne(q_1, [0 0 0 0 0 0], [0 0 0 0 0 0]);
q2_rne = r.rne(q_2, [0 0 0 0 0 0], [0 0 0 0 0 0]);

disp("Pares articulares para qs: ");
disp(qs_rne);

disp("Pares articulares para q1: ");
disp(q1_rne);

disp("Pares articulares para q2: ");
disp(q2_rne);
```

```
Pares articulares para qs:
   -0.0000  -62.3115  -19.2512    0.0000    0.6263         0

Pares articulares para q1:
    0.0000  -71.1772  -28.1169    0.0000   -0.0001         0

Pares articulares para q2:
   -0.0000  -83.9400  -19.3848   -0.2711   -0.5401         0
```

7. Ejercicio 7

Calcula los resultados dinámicos (par articular, par de gravedad, par de coriolis, par de inercia) para distintas posiciones con el valor de la gravedad en la Luna ($g = 1,62$ m/s^2). Justifica los resultados.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = DynamicParams(loadPA10Params());
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

q_s = [0 0.7854 1.571 0 -0.7854 0];
q_1 = [0 0.7854 0.7854 0 1.571 0];
q_2 = [0.3491 1.571 0.7854 -3.927 1.047 0];

r.gravity = [0 0 1.62];

%%%% PAR ARTICULAR %%%%
qd = [0 0 0 0 0 0]; % Velocidades articulares
qdd = [0 0 0 0 0 0]; % Aceleraciones articulares

tau_s = r.rne(q_s, qd, qdd);
tau_1 = r.rne(q_1, qd, qdd);
tau_2 = r.rne(q_2, qd, qdd);

%%%% PAR DE GRAVEDAD %%%%
g_s = r.gravload(q_s);
g_1 = r.gravload(q_1);
g_2 = r.gravload(q_2);

%%%% PAR DE CORIOLIS %%%%
qd_c = [0.1 0.1 0.1 0.1 0.1 0.1]; % Velocidades articulares no cero
C_s = r.coriolis(q_s, qd_c) * qd_c';
C_1 = r.coriolis(q_1, qd_c) * qd_c';
C_2 = r.coriolis(q_2, qd_c) * qd_c';

%%%% PAR DE INERCIA %%%%
I_s = r.inertia(q_s);
I_1 = r.inertia(q_1);
I_2 = r.inertia(q_2);

disp('Par articular para q_s:');
disp(tau_s);

disp('Par articular para q_1:');
disp(tau_1);

disp('Par articular para q_2:');
disp(tau_2);

disp('Par de gravedad para q_s:');
disp(g_s);

disp('Par de gravedad para q_1:');
disp(g_1);

disp('Par de gravedad para q_2:');
disp(g_2);

disp('Par de Coriolis para q_s:');
disp(C_s);

disp('Par de Coriolis para q_1:');
disp(C_1);

disp('Par de Coriolis para q_2:');
disp(C_2);

disp('Par de inercia para q_s:');
```

```
63 disp(I_s);
64
65 disp('Par de inercia para q_1:');
66 disp(I_1);
67
68 disp('Par de inercia para q_2:');
69 disp(I_2);
```

```
Par articular para q_s:
   -0.0000  -10.2900   -3.1791    0.0000    0.1034         0

Par articular para q_1:
    0.0000  -11.7540   -4.6432    0.0000   -0.0000         0

Par articular para q_2:
    0.0000  -13.8616   -3.2012   -0.0448   -0.0892         0
```

```
Par de gravedad para q_s:
   -0.0000  -10.2900   -3.1791    0.0000    0.1034         0

Par de gravedad para q_1:
    0.0000  -11.7540   -4.6432    0.0000   -0.0000         0

Par de gravedad para q_2:
    0.0000  -13.8616   -3.2012   -0.0448   -0.0892         0
```

```
Par de Coriolis para q_s:
   -0.0163
   -0.0428
    0.0241
   -0.0005
    0.0007
   -0.0000

Par de Coriolis para q_1:
    0.0421
   -0.0433
    0.0105
   -0.0011
   -0.0017
    0.0000

Par de Coriolis para q_2:
   -0.0599
   -0.0117
    0.0252
   -0.0000
   -0.0013
   -0.0000
```

```
Par de inercia para q_s:
    3.3202    0.0009    0.0013    0.0128   -0.0000   -0.0000
    0.0009    3.9861    1.1732   -0.0000   -0.0295    0.0000
    0.0013    1.1732    1.1938   -0.0000   -0.0092    0.0000
    0.0128   -0.0000   -0.0000    0.0191    0.0000    0.0004
   -0.0000   -0.0295   -0.0092    0.0000    0.0125    0.0000
   -0.0000    0.0000    0.0000    0.0004    0.0000    0.0006

Par de inercia para q_1:
    4.5375   -0.0004   -0.0000   -0.0510    0.0000   -0.0006
   -0.0004    5.9352    2.1694   -0.0000    0.0328    0.0000
   -0.0000    2.1694    1.2371   -0.0000    0.0125    0.0000
   -0.0510   -0.0000   -0.0000    0.0238    0.0000   -0.0000
    0.0000    0.0328    0.0125    0.0000    0.0125    0.0000
   -0.0006    0.0000    0.0000   -0.0000    0.0000    0.0006

Par de inercia para q_2:
    4.9453   -0.0102   -0.0093    0.0136   -0.0116    0.0000
   -0.0102    5.8196    2.0967    0.0284    0.0268    0.0004
   -0.0093    2.0967    1.2073    0.0159    0.0020    0.0004
    0.0136    0.0284    0.0159    0.0215    0.0000    0.0003
   -0.0116    0.0268    0.0020    0.0000    0.0125    0.0000
    0.0000    0.0004    0.0004    0.0003    0.0000    0.0006
```

8. Ejercicio 8

¿Cómo afecta añadir una carga de este tipo a la componente gravitacional e inercial? ¿Y si la separamos también 0.3 m en el eje X? ¿Añadir una carga afectará sólo a la componente gravitacional? Justifica las respuestas haciendo uso del robot PA10.

Añadir una carga al extremo del robot PA10 afectará tanto a la componente gravitacional como a la inercial. Veamos cómo esto sucede:

1. **Componente gravitacional**: Al agregar una carga, aumenta la masa del sistema y, por lo tanto, la fuerza gravitacional que actúa sobre el sistema. Si la carga se encuentra en el extremo del efector final y se separa en el eje Z, afectará principalmente a la componente gravitacional. Al separarla en el eje X, también se modificará la distribución de la fuerza gravitacional.

2. **Componente inercial**: La inercia del sistema está relacionada con la masa y la distribución de la masa en el sistema. Al añadir una carga, la inercia del sistema cambiará, lo que afectará el esfuerzo necesario para acelerar y desacelerar el robot durante los movimientos.

Para analizar cómo afecta la carga en el robot PA10, primero añadimos la carga al extremo del robot en el eje Z y luego en el eje X. Asumamos una carga de 2.2 kg.

Después de agregar la carga en cada caso, se pueden recalcular las componentes gravitacionales e inerciales siguiendo los pasos que se describian anteriormente en el ejercicio 7 para las posiciones q_s, q_1 y q_2. Al comparar estos resultados con los resultados sin carga, se podrán ver cómo la carga afecta a las componentes gravitacionales e inerciales.

Añadir una carga no solo afectará la componente gravitacional, también afectará la inercia del sistema, como se mencionó anteriormente. La inercia se ve influenciada por la distribución de la masa en el sistema, por lo que cualquier cambio en la masa y su ubicación afectará la inercia del robot.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = DynamicParams(loadPA10Params());
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

q_s = [0 0.7854 1.571 0 -0.7854 0];
q_1 = [0 0.7854 0.7854 0 1.571 0];
q_2 = [0.3491 1.571 0.7854 -3.927 1.047 0];

r.payload(2.2, [0.3, 0, 0]);
r.gravity = [0 0 1.62];

%%%% PAR ARTICULAR %%%%
qd = [0 0 0 0 0 0]; % Velocidades articulares
qdd = [0 0 0 0 0 0]; % Aceleraciones articulares

tau_s = r.rne(q_s, qd, qdd);
tau_1 = r.rne(q_1, qd, qdd);
tau_2 = r.rne(q_2, qd, qdd);

%%%% PAR DE GRAVEDAD %%%%
g_s = r.gravload(q_s);
g_1 = r.gravload(q_1);
g_2 = r.gravload(q_2);

%%%% PAR DE CORIOLIS %%%%
qd_c = [0.1 0.1 0.1 0.1 0.1 0.1]; % Velocidades articulares no cero
C_s = r.coriolis(q_s, qd_c) * qd_c';
C_1 = r.coriolis(q_1, qd_c) * qd_c';
C_2 = r.coriolis(q_2, qd_c) * qd_c';

%%%% PAR DE INERCIA %%%%
I_s = r.inertia(q_s);
I_1 = r.inertia(q_1);
I_2 = r.inertia(q_2);

```

```
disp('Par articular para q_s:');
disp(tau_s);

disp('Par articular para q_1:');
disp(tau_1);

disp('Par articular para q_2:');
disp(tau_2);

disp('Par de gravedad para q_s:');
disp(g_s);

disp('Par de gravedad para q_1:');
disp(g_1);

disp('Par de gravedad para q_2:');
disp(g_2);

disp('Par de Coriolis para q_s:');
disp(C_s);

disp('Par de Coriolis para q_1:');
disp(C_1);

disp('Par de Coriolis para q_2:');
disp(C_2);

disp('Par de inercia para q_s:');
disp(I_s);

disp('Par de inercia para q_1:');
disp(I_1);

disp('Par de inercia para q_2:');
disp(I_2);

```

```
Par articular para q_s:
    0.0000  -12.4396   -4.7359    0.0000   -0.8212    0.0000

Par articular para q_1:
    0.0000  -12.1717   -4.4680    0.0000    1.0694    0.0000

Par articular para q_2:
    0.0000  -15.6726   -4.1738    0.6229    0.6239    0.5345

Par de gravedad para q_s:
    0.0000  -12.4396   -4.7359    0.0000   -0.8212    0.0000

Par de gravedad para q_1:
    0.0000  -12.1717   -4.4680    0.0000    1.0694    0.0000

Par de gravedad para q_2:
    0.0000  -15.6726   -4.1738    0.6229    0.6239    0.5345
```

```
Par de Coriolis para q_s:
   -0.0464
   -0.0904
    0.0326
   -0.0014
    0.0089
   -0.0113

Par de Coriolis para q_1:
    0.0206
   -0.0681
    0.0027
   -0.0106
    0.0225
   -0.0104

Par de Coriolis para q_2:
   -0.0735
    0.0097
    0.0309
    0.0026
   -0.0005
    0.0021
```

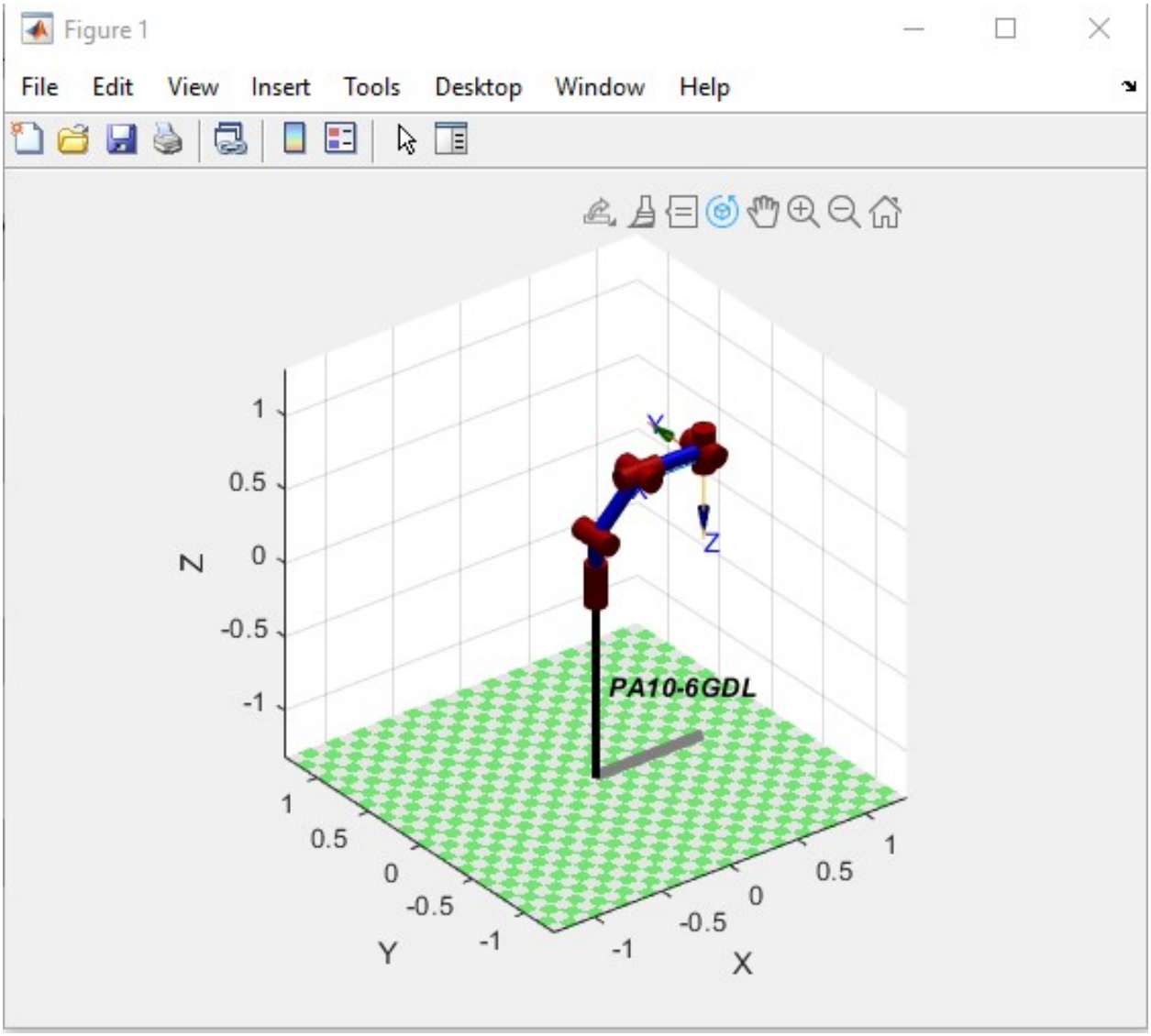
```
Par de inercia para q_s:
    4.7279    0.0009    0.0013    0.0672    0.0000    0.6121
    0.0009    5.6204    2.6031    0.0000    0.7177    0.0000
    0.0013    2.6031    2.6522    0.0000    0.7665    0.0000
    0.0672    0.0000    0.0000    0.0199    0.0000    0.0144
    0.0000    0.7177    0.7665    0.0000    0.3704    0.0000
    0.6121    0.0000    0.0000    0.0144    0.0000    0.1986

Par de inercia para q_1:
    4.4145   -0.0004   -0.0000    0.2264    0.0000    0.3282
   -0.0004    5.7252    1.9425   -0.0000   -0.3178   -0.0000
   -0.0000    1.9425    1.2262   -0.0000    0.0535   -0.0000
    0.2264   -0.0000   -0.0000    0.1836    0.0000    0.1782
    0.0000   -0.3178    0.0535    0.0000    0.3704    0.0000
    0.3282   -0.0000   -0.0000    0.1782    0.0000    0.1986

Par de inercia para q_2:
    6.1789    0.0205    0.0213   -0.6228    0.0077   -0.5064
    0.0205    6.7792    2.5533   -0.3624   -0.3272   -0.3139
    0.0213    2.5533    1.3937   -0.1893   -0.1540   -0.1654
   -0.6228   -0.3624   -0.1893    0.3453    0.0000    0.2536
    0.0077   -0.3272   -0.1540    0.0000    0.3704    0.0000
   -0.5064   -0.3139   -0.1654    0.2536    0.0000    0.1986
```

9. Ejercicio 9

Realiza 3 trayectorias articulares con el robot PA10 entre diferentes puntos probando el perfil trapezoidal y polinomial. Para visualizar los valores de velocidad y aceleración puedes emplear el comando plot(qd). Realiza 3 trayectorias cartesianas con el robot PA10 cambiando los valores de la posición cartesiana del robot. Para todas las trayectorias, representa gráficamente los valores de las posiciones en los tres ejes del espacio cartesiano X Y Z a lo largo de la trayectoria y los valores de su jacobiano (determinante matriz J).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%ROBOT PA10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L(1) = Link([0 , 0.317 , 0 , -pi/2 , 0 , 0 ]);
L(1).qlim=[deg2rad(-177) deg2rad(177)];

L(2) = Link([0 , 0 , 0.45 , 0 , 0 , -pi/2 ]);
L(2).qlim=[deg2rad(-64) deg2rad(124)];

L(3) = Link([0 , 0 , 0 , pi/2 , 0 , pi/2 ]);
L(3).qlim=[deg2rad(-107) deg2rad(158)];

L(4) = Link([0 , 0.48 , 0 , -pi/2 , 0 , 0 ]);
L(4).qlim=[deg2rad(-255) deg2rad(255)];

L(5) = Link([0 , 0 , 0 , pi/2 , 0 , 0 ]);
L(5).qlim=[deg2rad(-165) deg2rad(165)];

L(6) = Link([0 , 0.07 , 0 , 0 , 0 , 0 ]);
L(6).qlim=[deg2rad(-255) deg2rad(255)];

r = SerialLink(L,'name', 'PA10-6GDL');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

q_h = [0 0 0 0 0 0];
q_e = [0 0.5236 1.571 0 1.0472 0];
q_s = [0 0.7854 1.571 0 -0.7854 0];
q_1 = [0 0.7854 0.7854 0 1.571 0];
q_2 = [0.3491 1.571 0.7854 -3.927 1.047 0];

k1 = jtraj(q_h, q_e, 50);
r.plot(k1);

k2 = jtraj(q_h, q_s, 50);
r.plot(k2);

k3 = jtraj(q_h, q_1, 50);
r.plot(k3);
```

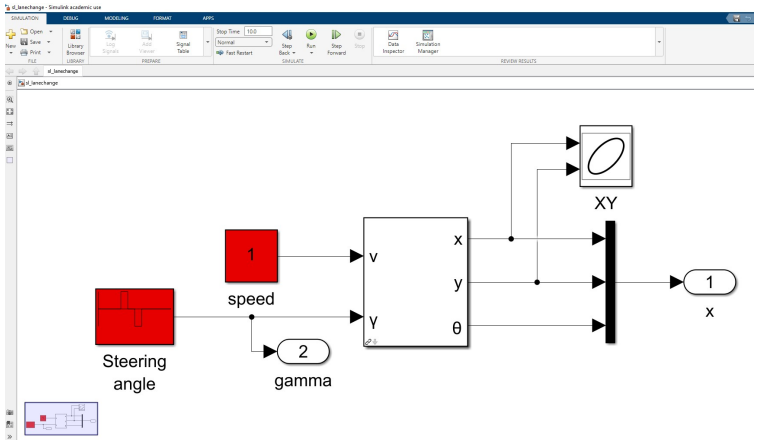
Nota: La figura se mostrará una imagen en movimiento (desde la posición inicial que es totalmente erecta(qh), hasta la posición final qe,qs,q1).

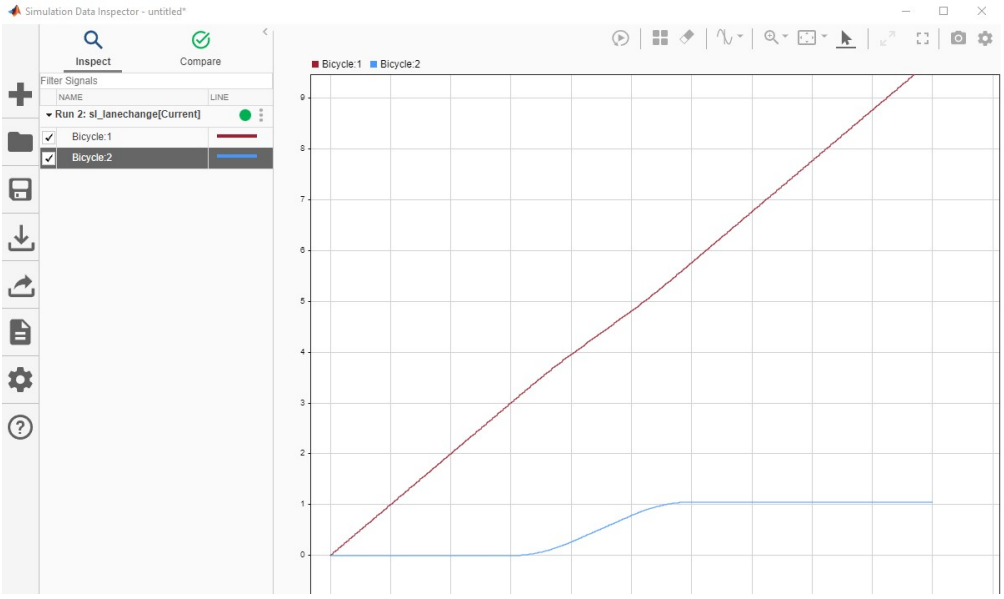


10. Ejercicio 10

Inserta el comando sl_lanechange en la línea de comandos de Matlab para abrir el archivo Simulink. Ejecuta dicho archivo y visualiza la entrada de dirección (Steering angle), así como el valor del ángulo theta. Cambia los valores máximos/mínimos de la dicha entrada y visualiza los cambios en el visor XY. ¿Qué es lo que representa esta gráfica XY? Cambia los parámetros del bloque Bicycle y visualiza los cambios en la posición del vehículo.

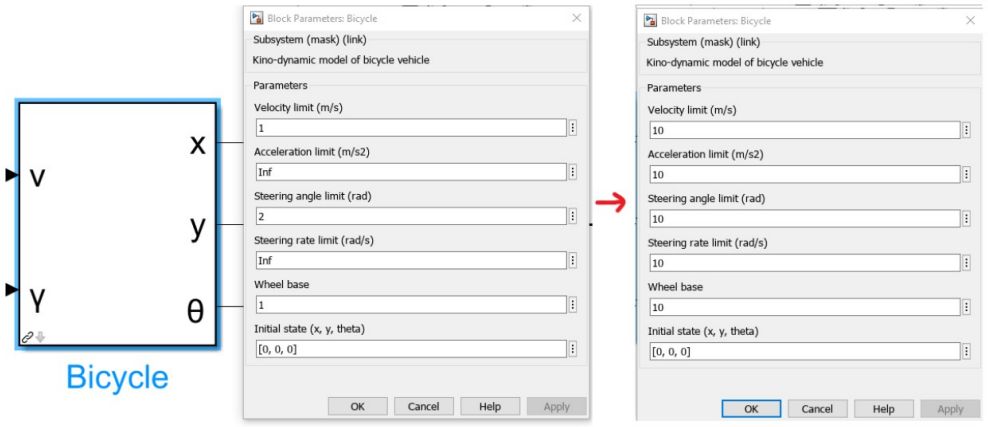
Nota: Para este ejercicio he instalado el addón 'Simulink'.

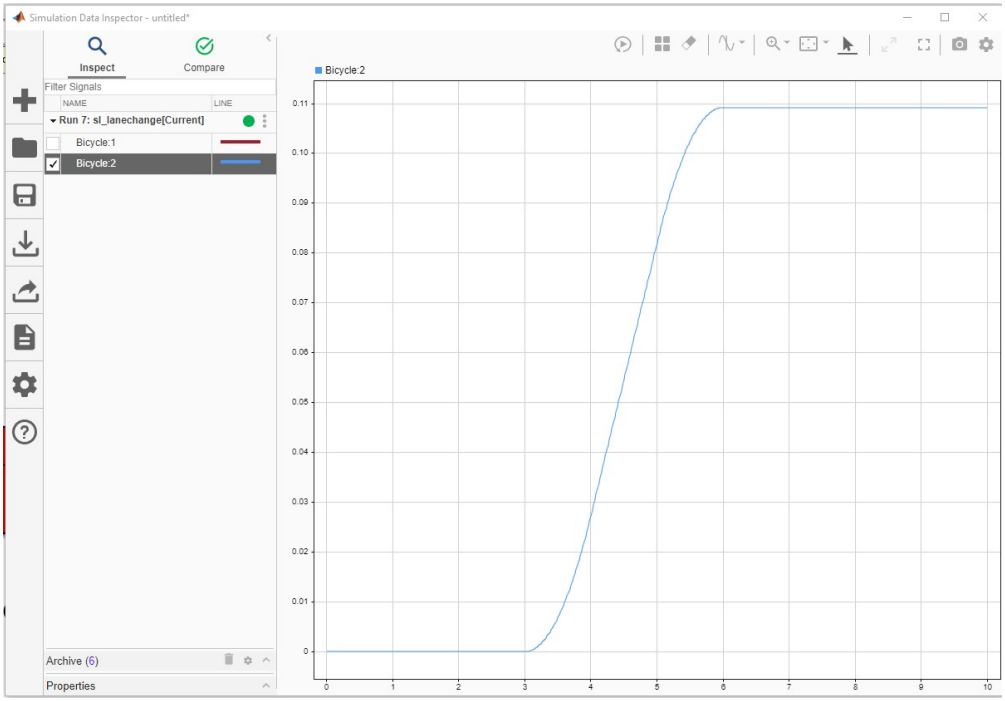




La gráfica XY en el Simulation Data Inspector de Simulink representa la relación entre dos variables de los datos de simulación en un diagrama de dispersión. En lugar de graficar una variable en función del tiempo (como en una gráfica típica de señales temporales), una gráfica XY muestra cómo una variable depende de otra variable. En este ejercicio se analiza la relación entre la posición en el eje X (posición longitudinal) y la posición en el eje Y (posición lateral) de las bicicletas. En este caso, la gráfica XY mostraría la trayectoria de cada bicicleta en el espacio bidimensional. Esto te permite analizar la geometría de las trayectorias, cómo se cruzan o se acercan, y cómo se comportan durante un cambio de carril u otras maniobras.

Hago los cambios en 'Bicycle':

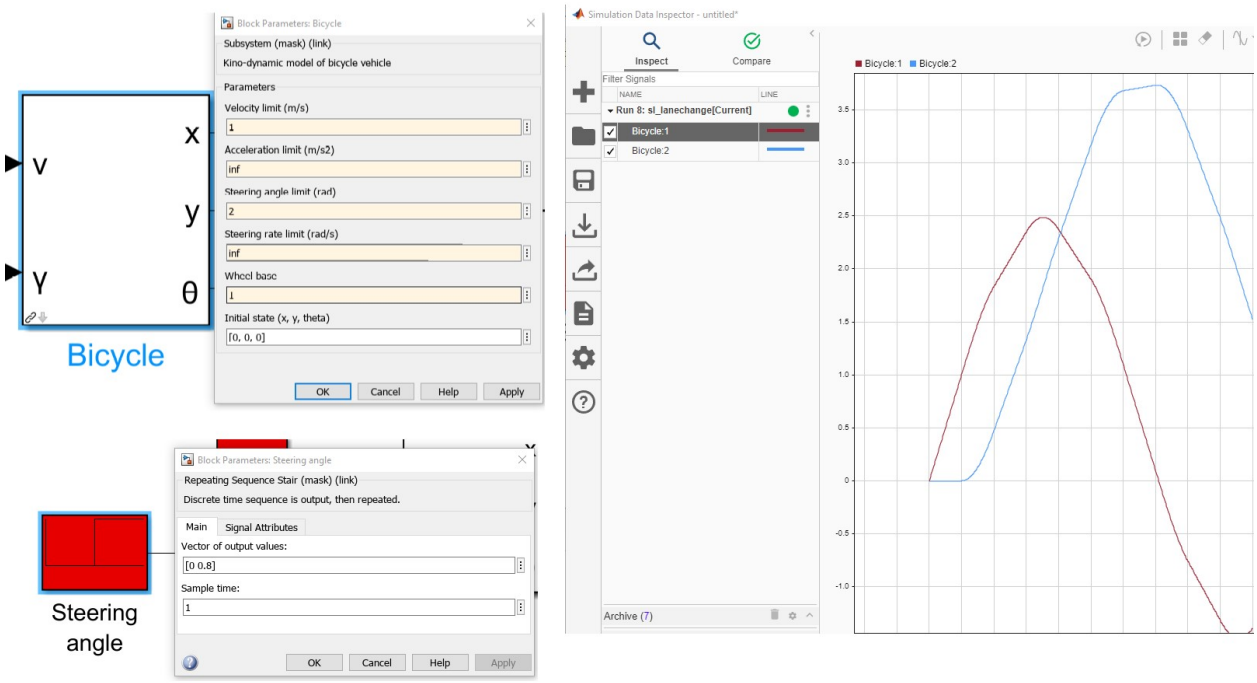




11. Ejercicio 11

Sobre el archivo Simulink introduce otras entradas en la dirección del vehículo y visualiza los cambios en la trayectoria. ¿Qué tipo de entrada y qué valor se debe introducir al vehículo para que la trayectoria XY sea una circunferencia en un tiempo de 10 seg? Nota: Para este ejercicio he instalado el addón 'Simulink'.

Aplico cambios al steeringwheel:



Circunferencia en un tiempo de 10 seg: