# Práctica 2: Modelado y simulación de sistemas robóticos

## Automatización y Robótica

Francisco Javier Pérez Martínez

11 de mayo de 2022

# Índice

# 1. Descripción

Para el presente documento, correspondiente a la práctica 2 de la asignatura, se trabajará con la librería de Robotics Toolbox de Peter Corke para Matlab en el que modelaremos y simularemos robots manipuladores y móviles a partir de sus características así como las posibilidades de simulación gráfica que nos ofrece Matlab.

# 2. 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 -90º sobre el eje O'U del sistema trasladado y giro de 90º sobre el eje O'V' del sistema girado.

## 2.1. Código Matlab

```
%% Ejercicio 1

% Rotación de -90º sobre U y de 90º sobre V --> postmultiplicar
tr = transl(-3,10,10) * trotx(-90) * troty(90);

%redondea y muestra la matriz
disp(round(tr));

% Graficar el resultado de la transformación
trplot(tr);
```

## 2.2. Resultados

```
>> ejercicio1
     0     0     1    -3
    -1     0     0    10
     0    -1     0    10
     0     0     0     1
```

Figura 1: Matriz de transformación

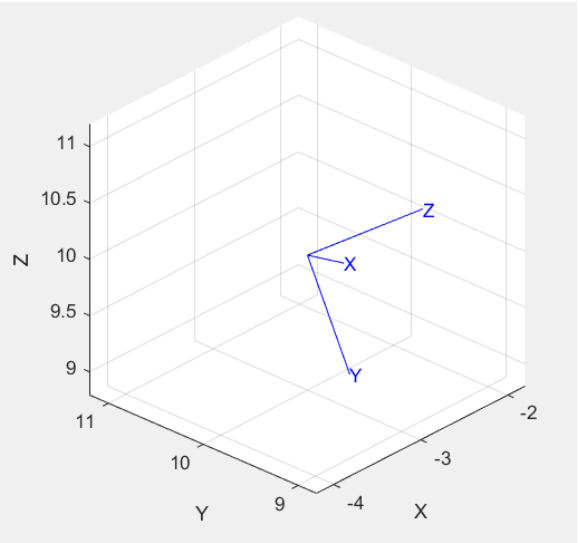


Figura 2: Gráfico de la transformación

# 3. 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.

## 3.1. Código Matlab

```
%% Ejercicio 2

% Objeto Link ([theta, d, a, alpha, límite, offset])
L1 = Link ([0 0.317 0 -pi/2 0]);
L2 = Link ([0 0 0.45 0 0 -pi/2]);
L3 = Link ([0 0 0 pi/2 0 pi/2]);
L4 = Link ([0 0.48 0 -pi/2 0]);
L5 = Link ([0 0 0 pi/2 0]);
L6 = Link ([0 0.07 0 0 0]);

% Introducir límites articulares
L1.qlim = [-177, 177];
L2.qlim = [-64, 124];
L3.qlim = [-107, 158];
L4.qlim = [-255, 255];
L5.qlim = [-165, 165];
L6.qlim = [-255, 255];

L = [L1 L2 L3 L4 L5 L6];

robot = SerialLink(L, 'name', 'PA10_6GDL')
robot.n
robot.links
```

## 3.2. Resultados

```
>> ejercicio2

robot =

PA10_6GDL:: 6 axis, RRRRRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|      0.317|          0|    -1.5708|          0|
|  2|         q2|          0|       0.45|          0|    -1.5708|
|  3|         q3|          0|          0|     1.5708|     1.5708|
|  4|         q4|       0.48|          0|    -1.5708|          0|
|  5|         q5|          0|          0|     1.5708|          0|
|  6|         q6|       0.07|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+


ans =

     6
```

Figura 3: Tabla parámetros DH y número de GDL del robot

```
ans =
Revolute(std):  theta=q1   d=0.317      a=0          alpha=-1.571      offset=0
Revolute(std):  theta=q2   d=0          a=0.45       alpha=0           offset=-1.571
Revolute(std):  theta=q3   d=0          a=0          alpha=1.571       offset=1.571
Revolute(std):  theta=q4   d=0.48       a=0          alpha=-1.571      offset=0
Revolute(std):  theta=q5   d=0          a=0          alpha=1.571       offset=0
Revolute(std):  theta=q6   d=0.07       a=0          alpha=0           offset=0
```

Figura 4: Robot.links para mostrar los paráemtros DH de cada articulación

# 4. 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: qh = [0, 0, 0, 0, 0, 0].
- Posición de escape: qe = [0, 30, 90, 0, 60, 0].
- Posición de seguridad: qs = [0, 45, 90, 0, -45, 0].
- Posición q1 = [0, 45, 45, 0, 90, 0].
- Posición q2 = [20, 90, 45, -22.5, 60, 0].

## 4.1. Código Matlab

```
%% Ejercicio 3

% Cargar robot 6GDL
ejercicio2

% Posición de home
qh = [0 0 0 0 0 0]
Tqh = robot.fkine(qh)
robot.plot(qh)

% Posición de escape
qe = [0 deg2rad(30) deg2rad(90) 0 deg2rad(60) 0]
Tqe = robot.fkine(qe)
robot.plot(qe)

% Posición de seguridad
qs = [0 deg2rad(45) deg2rad(90) 0 deg2rad(-45) 0]
Tqs = robot.fkine(qs)
robot.plot(qs)

% Posición q1
q1 = [0 deg2rad(45) deg2rad(45) 0 deg2rad(90) 0]
Tq1 = robot.fkine(q1)
robot.plot(q1)

% Posición q2
q2 = [deg2rad(20) deg2rad(90) deg2rad(45) deg2rad(-22.5) deg2rad(60) 0]
Tq2 = robot.fkine(q2)
robot.plot(q2)
```

## 4.2. Resultados

```
qh =

     0     0     0     0     0     0


Tqh =
         1         0         0         0
         0         1         0         0
         0         0         1     1.317
         0         0         0         1

qe =

         0    0.5236    1.5708         0    1.0472         0

Tqe =
        -1         0         0    0.6407
         0         1         0         0
         0         0        -1    0.3967
         0         0         0         1
```

Figura 5: Posiciones articulares qh y qe

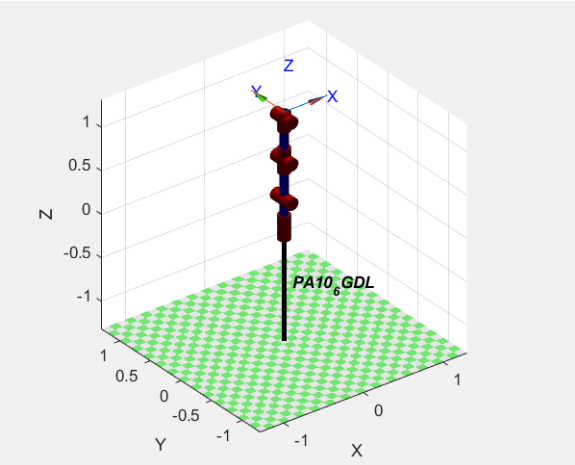


Figura 6: Plot qh

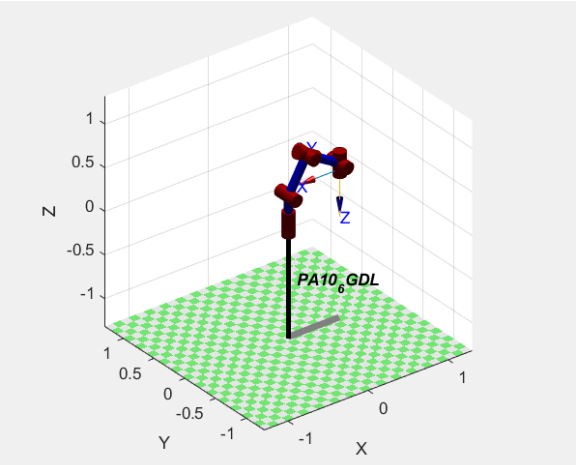


Figura 7: Plot qe

```
qs =

         0    0.7854    1.5708         0   -0.7854         0


Tqs =
         0         0         1    0.7276
         0         1         0         0
        -1         0         0    0.2958
         0         0         0         1

q1 =

         0    0.7854    0.7854         0    1.5708         0


Tq1 =
        -1         0         0    0.7982
         0         1         0         0
         0         0        -1    0.5652
         0         0         0         1
```

Figura 8: Posiciones articulares qs y q1

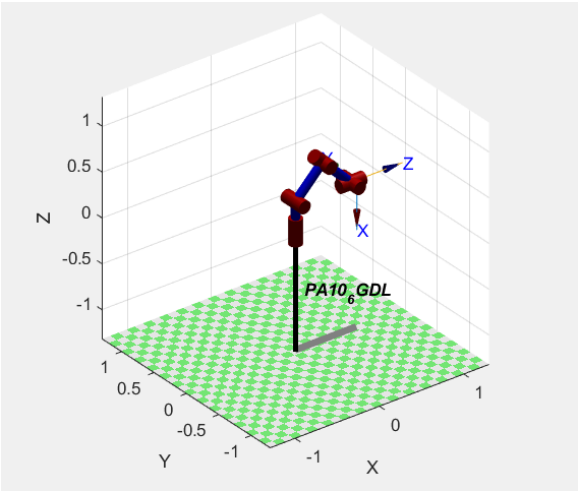


Figura 9: Plot qs

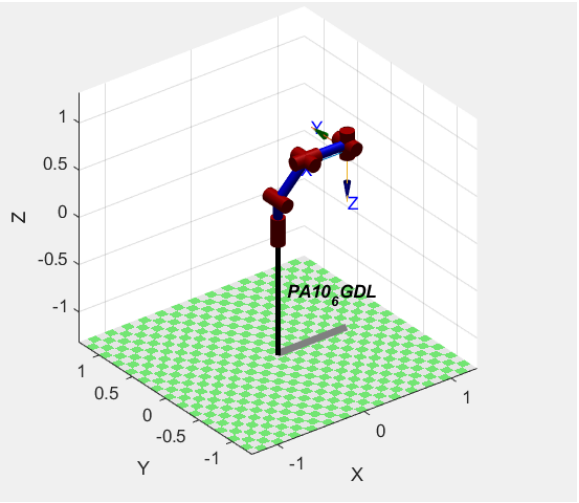


Figura 10: Plot q1

```
q2 =

    0.3491    1.5708    0.7854   -0.3927    1.0472         0


Tq2 =
   -0.8169   -0.5703   -0.0861    0.7358
   -0.5010    0.7756   -0.3840    0.2431
    0.2857   -0.2706   -0.9193   -0.08676
         0         0         0         1
```

Figura 11: Posición articular q2

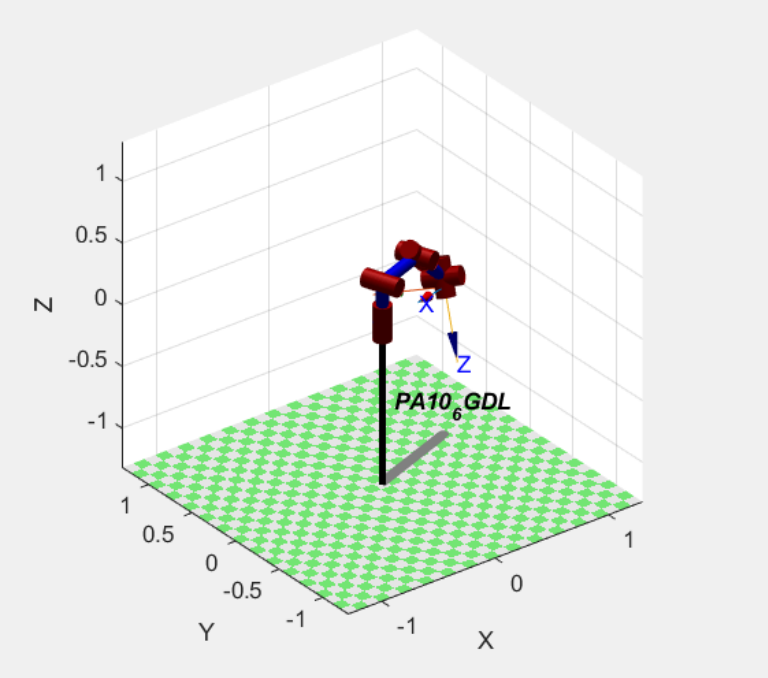


Figura 12: Plot q2

# 5. 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.

## 5.1. Código Matlab

```
%% Ejercicio 4

% Cargar robot 6GDL y posiciones articulares

% Calcular matriz T para la posición articular qe
T1 = robot.fkine(qe)

% A partir de la matriz T, calcular qinversa
qinversa1 = robot.ikine6s(T1)
robot.plot(qinversa1)
Tinversa1 = robot.fkine(qinversa1)

% Calcular matriz T para la posición articular qs
T2 = robot.fkine(qs)

% A partir de la matriz T, calcular qinversa
qinversa2 = robot.ikine6s(T2)
robot.plot(qinversa2)
Tinversa2 = robot.fkine(qinversa2)

% Calcular matriz T para la posición articular q1
T3 = robot.fkine(q1)

% A partir de la matriz T, calcular qinversa
qinversa3 = robot.ikine6s(T3)
robot.plot(qinversa3)
Tinversa3 = robot.fkine(qinversa3)

% Calcular matriz T para la posición articular q2
T4 = robot.fkine(q2)

% A partir de la matriz T, calcular qinversa
qinversa4 = robot.ikine6s(T4)
robot.plot(qinversa4)
Tinversa4 = robot.fkine(qinversa4)
```

## 5.2. Resultados

```
T1 =
        -1         0         0    0.6407
         0         1         0         0
         0         0        -1    0.3967
         0         0         0         1

qinversa1 =

   -3.1416   -1.4470   -0.0000    0.0000   -1.6946   -3.1416


Tinversa1 =
        -1         0         0    0.9229
         0         1         0         0
         0         0        -1    0.3618
         0         0         0         1
```

Figura 13: qe inversa

```
T2 =
         0         0         1    0.7276
         0         1         0         0
        -1         0         0    0.2958
         0         0         0         1

qinversa2 =

   -3.1416   -1.5999    0.0000   -3.1416   -0.0291    0.0000


Tinversa2 =
         0         0         1    0.9996
         0         1         0         0
        -1         0         0    0.2899
         0         0         0         1
```

Figura 14: qs inversa

```
T3 =
        -1         0         0    0.7982
         0         1         0         0
         0         0        -1    0.5652
         0         0         0         1

qinversa3 =

   -3.1416   -1.2693   -0.0000    0.0000   -1.8723   -3.1416


Tinversa3 =
        -1         0         0    0.8881
         0         1         0         0
         0         0        -1    0.5231
         0         0         0         1
```

Figura 15: qe inversa

```
T4 =
   -0.8169   -0.5703   -0.0861    0.7358
   -0.5010    0.7756   -0.3840    0.2431
    0.2857   -0.2706   -0.9193  -0.08676
         0         0         0         1

qinversa4 =

   -2.8225   -2.0511   -0.0000   -0.3557   -1.3228    2.9973


Tinversa4 =
   -0.8169   -0.5703   -0.0861    0.7771
   -0.5010    0.7756   -0.3840    0.2319
    0.2857   -0.2706   -0.9193   -0.1771
         0         0         0         1
```

Figura 16: qs inversa

# 6. 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).

## 6.1. Código Matlab

```
% Ejercicio 5

% Cargar robot 6GDL
ejercicio2

% Robot planar
L10 = Link ([0 0 1 0]);
L11 = Link ([0 0 1 0]);
L12 = Link ([0 0 1 0]);

planar = [L10 L11 L12];
planar = SerialLink(planar, 'name', 'planar');

% ------ ROBOT PA10 ------ %
% El robot se encuentra al límite de su espacio de trabajo
q = [0 pi/2 0 0 0 0];

% vector coordenadas random para probar
qrand = [0 rand 0 0 0 rand];

% Matriz Jacobiana
J = jacob0(robot,q)

% Determinante = nulo
det(J)

robot.plot(qrand);
figure
robot.plot(q);
% ------ ROBOT PA10 ------ %

% ------ ROBOT PLANAR ------ %
qplanar = [1 0 0];
Jplanar = jacob0(planar, qplanar)
planar.plot(qplanar);
% ------ ROBOT PLANAR ------ %
```

## 6.2. Resultados

```
J =

   -0.0000    0.0000    0.0000         0    0.0000         0
    1.0000    0.0000    0.0000         0    0.0000         0
         0   -1.0000   -0.5500         0   -0.0700         0
   -0.0000         0         0    1.0000         0    1.0000
         0    1.0000    1.0000    0.0000    1.0000    0.0000
    1.0000    0.0000    0.0000    0.0000    0.0000    0.0000


DetJ =

     0


Jplanar =

   -2.5244   -1.6829   -0.8415
    1.6209    1.0806    0.5403
         0         0         0
         0         0         0
         0         0         0
    1.0000    1.0000    1.0000
```

Figura 17: Salida ej5

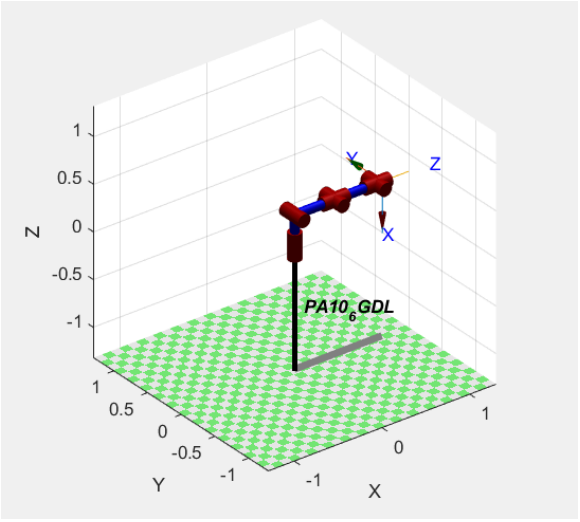


Figura 18: Figura robot 6GDL

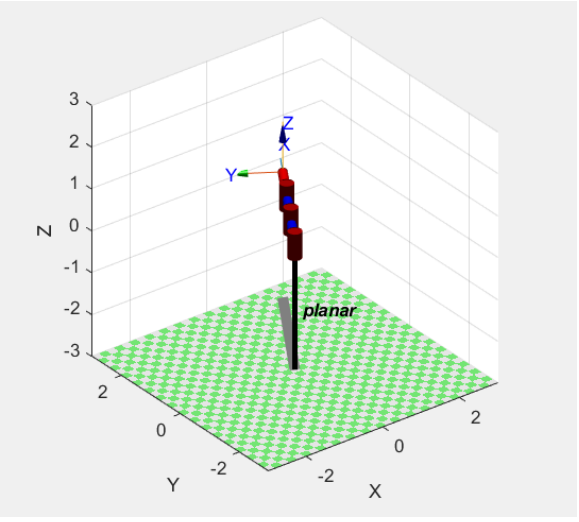


Figura 19: Figura robot planar

# 7. 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).

## 7.1. Código Matlab

```
% Ejercicio 6

PA10 = DynamicParams(loadPA10Params());

qs = [0 deg2rad(45) deg2rad(90) 0 deg2rad(-45) 0];
q1 = [0 deg2rad(45) deg2rad(45) 0 deg2rad(90) 0];
q2 = [deg2rad(20) deg2rad(90) deg2rad(45) deg2rad(-22.5) deg2rad(60) 0];

tau = PA10.rne(qs, [0 0 0 0 0 0], [0 0 0 0 0 0])
tau = PA10.rne(q1, [0 0 0 0 0 0], [0 0 0 0 0 0])
tau = PA10.rne(q2, [0 0 0 0 0 0], [0 0 0 0 0 0])
```

## 7.2. Resultados

```
>> ejercicio6

tau =

   -0.0000  -62.3155  -19.2553    0.0000    0.6263         0


tau =

   -0.0000  -71.1769  -28.1168    0.0000    0.0000         0


tau =

   -0.0000  -84.5689  -20.0145    0.1468   -0.1789         0
```

Figura 20: Salida partes articulares qs, q1 y q2

# 8. 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/s2). Justifica los resultados.

## 8.1. Código Matlab

```
% Ejercicio 7

PA10 = DynamicParams(loadPA10Params())
qs = [0 deg2rad(45) deg2rad(90) 0 deg2rad(-45) 0];
q1 = [0 deg2rad(45) deg2rad(45) 0 deg2rad(90) 0];
q2 = [deg2rad(20) deg2rad(90) deg2rad(45) deg2rad(-22.5) deg2rad(60) 0];

% Par debido a la gravedad G
% componente gravitacional de la dinámica del robot
PA10.gravity = [0 0 1.62]
G0 = PA10.gravload(qs)
G1 = PA10.gravload(q1)
G2 = PA10.gravload(q2)

% Par debido a la inercia M
% componente inercial de la dinámica del robot
M0 = PA10.itorque(qs, [1 0 0 0 0 0])
M1 = PA10.itorque(q1, [0 1 0 0 0 0])
M2 = PA10.itorque(q2, [0 0 1 0 0 0])

% Matriz de Coriolis C
C0 = PA10.coriolis(qs, [pi/2 0 pi 0 0 0])
C1 = PA10.coriolis(q1, [1 0 pi/2 0 0 0])
C2 = PA10.coriolis(q2, [1 0 pi 0 0 0])
```

## 8.2. Resultados

```
G0 =

    0.0000  -10.2906   -3.1798    0.0000    0.1034         0


G1 =

   -0.0000  -11.7540   -4.6431    0.0000    0.0000         0


G2 =

    0.0000  -13.9655   -3.3051    0.0242   -0.0296         0


M0 =

    3.3207    0.0009    0.0013    0.0129   -0.0000    0.0000


M1 =

   -0.0004    5.9352    2.1694   -0.0000    0.0328    0.0000


M2 =

    0.0078    2.1248    1.2067   -0.0086   -0.0026   -0.0002
```

Figura 21: Salida ej7

```
C0 =

   -3.8677    0.7172   -1.9298   -0.0258   -0.0000   -0.0009
   -0.7132   -3.9881   -3.9881    0.0047   -0.0043    0.0005
    1.9338    0.0000    0.0000    0.0272   -0.0681    0.0005
   -0.0545   -0.0047   -0.0272   -0.0000   -0.0017    0.0006
    0.0000    0.0681    0.0681    0.0017         0    0.0005
   -0.0009   -0.0005   -0.0005   -0.0006   -0.0005         0


C1 =

    0.0801    2.0264    0.0538   -0.0183    0.0510   -0.0000
   -2.0235   -1.4007   -1.4007    0.0325    0.0800         0
   -0.0510    0.0000   -0.0000    0.0122    0.0481    0.0000
   -0.0192   -0.0325   -0.0122    0.0000   -0.0109   -0.0005
   -0.0510   -0.0481   -0.0481    0.0109         0         0
   -0.0000   -0.0000   -0.0000    0.0005   -0.0000         0


C2 =

   -4.6360   -1.4259   -1.4539   -0.0139    0.0539    0.0002
    1.4477   -2.7822   -2.7822    0.0117    0.1683   -0.0001
    1.4757    0.0000    0.0000   -0.0095    0.0836   -0.0001
    0.0123    0.0095    0.0095   -0.0000   -0.0188   -0.0007
   -0.0563   -0.0836   -0.0836    0.0188         0    0.0001
    0.0002    0.0001    0.0001    0.0007   -0.0001         0
```

Figura 22: Salida ej7

# 9. 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.

## 9.1. Código Matlab

```
% Ejercicio 8

PA10 = DynamicParams(loadPA10Params());

qs = [0 deg2rad(45) deg2rad(90) 0 deg2rad(-45) 0];
q1 = [0 deg2rad(45) deg2rad(45) 0 deg2rad(90) 0];
q2 = [deg2rad(20) deg2rad(90) deg2rad(45) deg2rad(-22.5) deg2rad(60) 0];

% añadir carga
PA10.payload(2.5, [0 0 0.1])

grav1 = PA10.gravload(qs)
grav2 = PA10.gravload(q1)
grav3 = PA10.gravload(q2)

in1 = PA10.itorque(qs, [1 0 0 0 0 0])
in2 = PA10.itorque(q1, [1 0 0 0 0 0])
in3 = PA10.itorque(q2, [1 0 0 0 0 0])

% separación 0.3m en el eje X
PA10.payload(2.5, [0.3, 0, 0.1])

grav1 = PA10.gravload(qs)
grav2 = PA10.gravload(q1)
grav3 = PA10.gravload(q2)

in1_1 = PA10.itorque(qs, [1 0 0 0 0 0])
in2_2 = PA10.itorque(q1, [1 0 0 0 0 0])
in3_3 = PA10.itorque(q2, [1 0 0 0 0 0])

% Al añadir una carga afectará a la componente gravitacional e inercial.
% Como se puede apreciar al ejectuar el programa los parámetros de ambas
% componentes varian.
```

## 9.2. Resultados

```
>> ejercicio8

grav1 =

         0  -75.6123  -28.0259   -0.0000   -3.3164         0


grav2 =

         0  -82.5309  -34.9445    0.0000   -0.0000         0


grav3 =

   -0.0000  -94.9612  -24.0058   -0.7772    0.9476         0


in1 =

    4.5481    0.0009    0.0013   -0.2248    0.0000    0.0000


in2 =

    5.4613   -0.0004   -0.0000    0.2698    0.0000   -0.0006


in3 =

    5.8204   -0.0601   -0.0593    0.1832   -0.0739   -0.0005
```

Figura 23: Salida ej8

```
grav1 =

    0.0000  -75.6123  -28.0259    0.0000   -3.3164    0.0000


grav2 =

         0  -75.1734  -27.5870    0.0000    7.3575    0.0000


grav3 =

   -0.0000  -88.0524  -17.0970   -1.7726    7.7114   -1.9909


in1_1 =

    4.5481    0.0009    0.0013    0.2141    0.0000    0.6207


in2_2 =

    4.4890   -0.0004   -0.0000    0.1423   -0.0000    0.3731


in3_3 =

    4.9821   -0.1069   -0.1060    0.2335    0.0615    0.3032
```

Figura 24: Salida ej8

# 10. 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).

## 10.1. Código Matlab

```
% Ejercicio 9

PA10 = DynamicParams(loadPA10Params());
qh = [0 0 0 0 0 0];
q1 = [0 deg2rad(45) deg2rad(45) 0 deg2rad(90) 0];
q2 = [deg2rad(20) deg2rad(90) deg2rad(45) deg2rad(-22.5) deg2rad(60) 0];
qs = [0 deg2rad(45) deg2rad(90) 0 deg2rad(-45) 0];

[q, qd, qdd] = jtraj(qh, q1, 50);
PA10.plot(q);
plot(qd);
pause(2);

[q, qd, qdd] = jtraj(qh, q2, 50);
PA10.plot(q);
plot(qd);
pause(2);

[q, qd, qdd] = jtraj(qh, qs, 50);
PA10.plot(q);
plot(qd);
```

## 10.2. Resultados

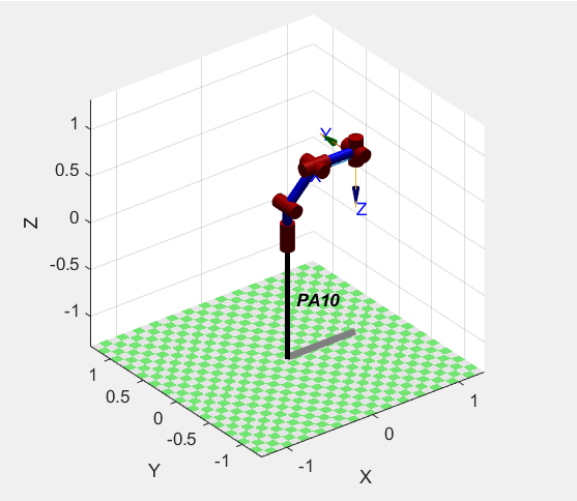


Figura 25: Figura q1

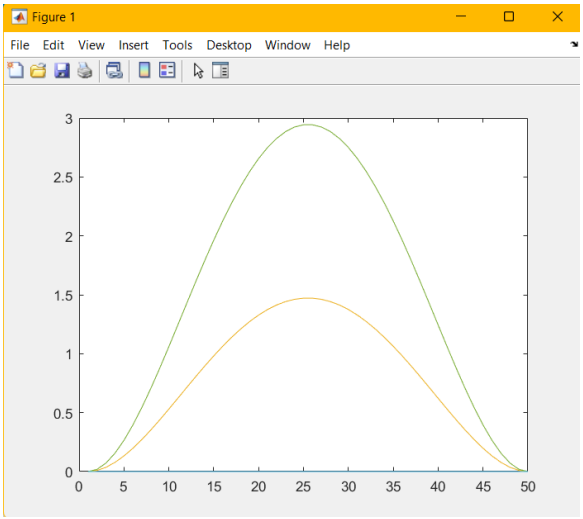


Figura 26: Gráfica q1

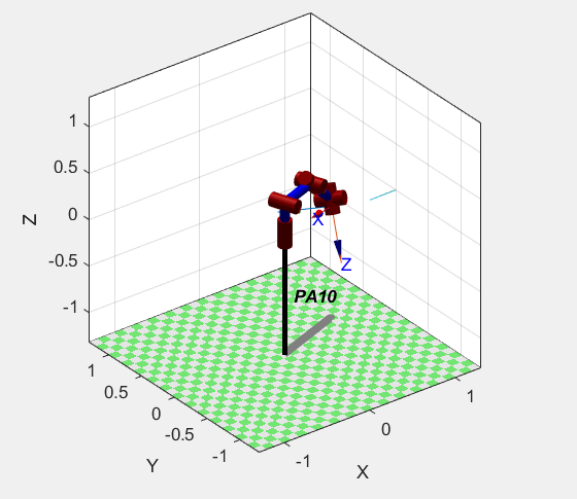


Figura 27: Figura q2

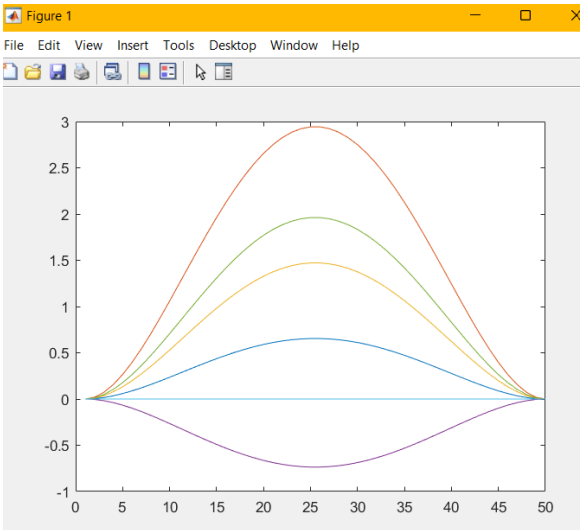


Figura 28: Gráfica q2

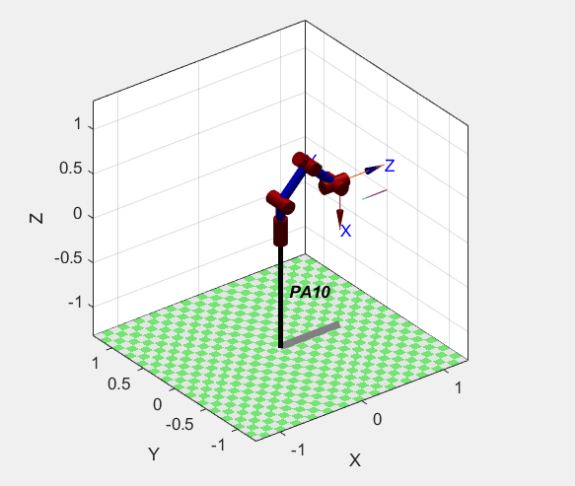


Figura 29: Figura qs

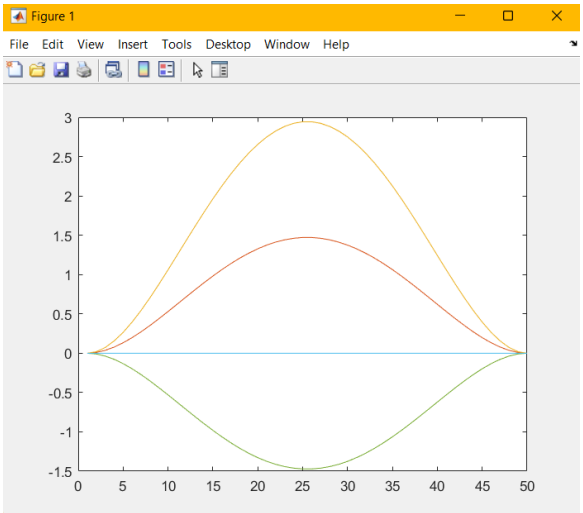


Figura 30: Gráfica qs

# 11. 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.

## 11.1. Configuración simulink

Al ejecutar el comando sl_lanechange se os abre la siguiente ventana:

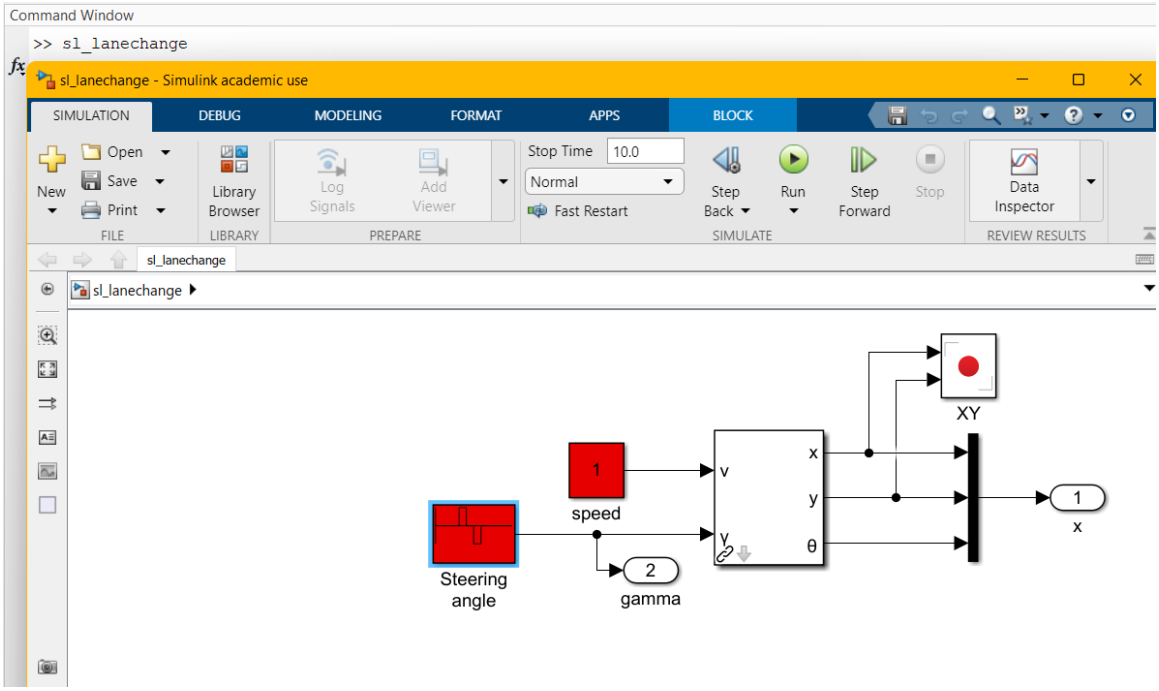


Figura 31: sl_lanechange

Parámetros del bloque steering angle por defecto:

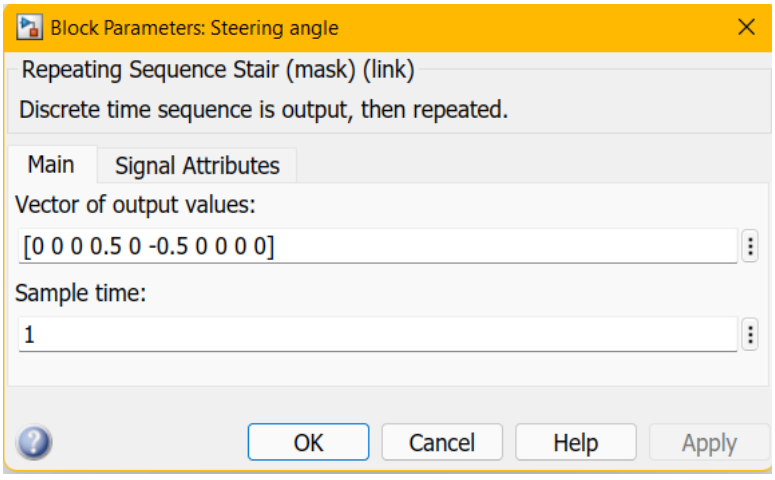


Figura 32: Parámetros del bloque Steering angle

## 11.2. Resultados

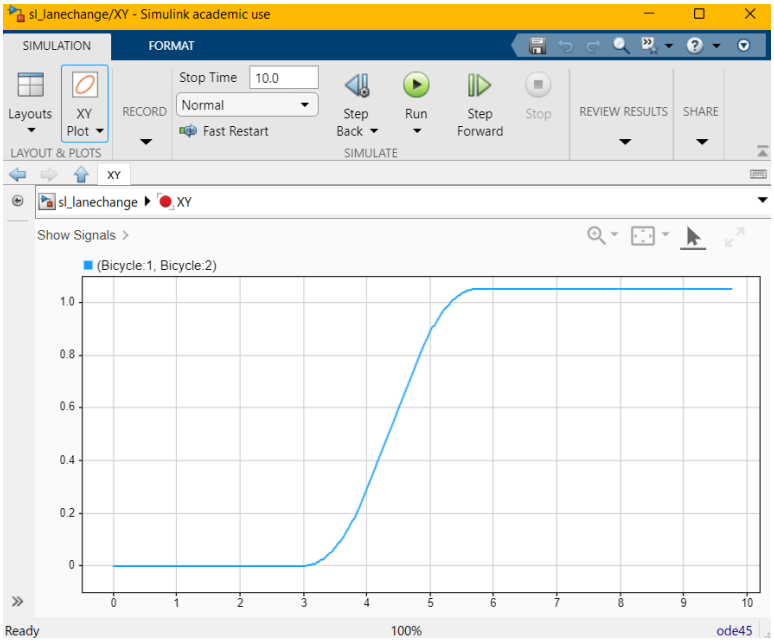


Figura 33: Bloque XY por defecto

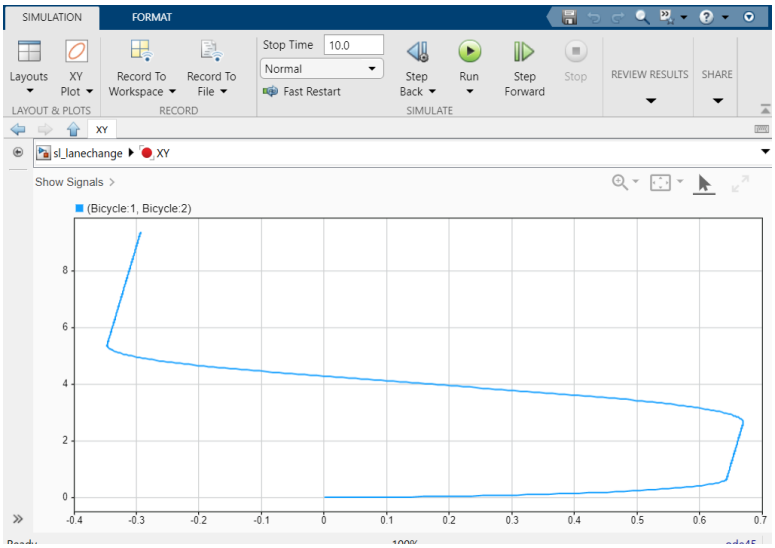


Figura 34: Bloque XY con los valores cambiados

# 12. 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?

## 12.1. Configuración simulink

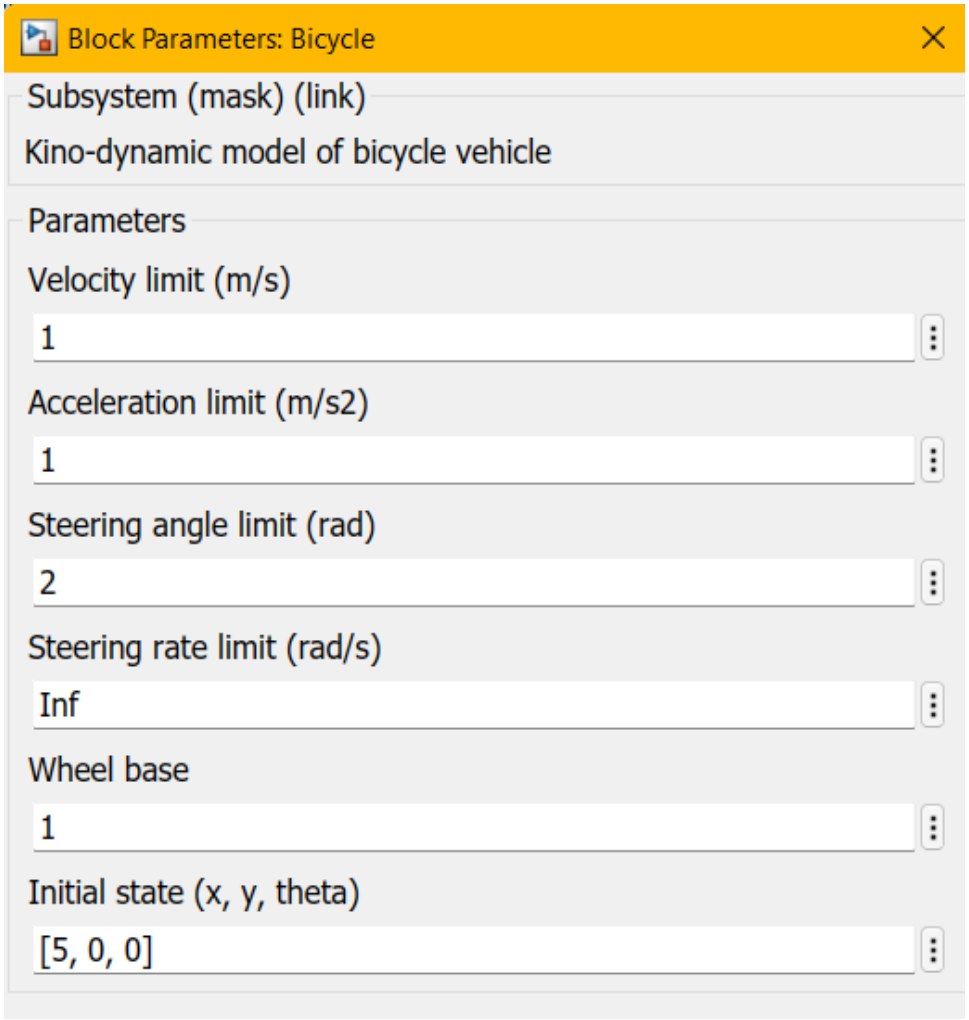


Figura 35: Parámetros del bloque bicycle, estado inicial X=5

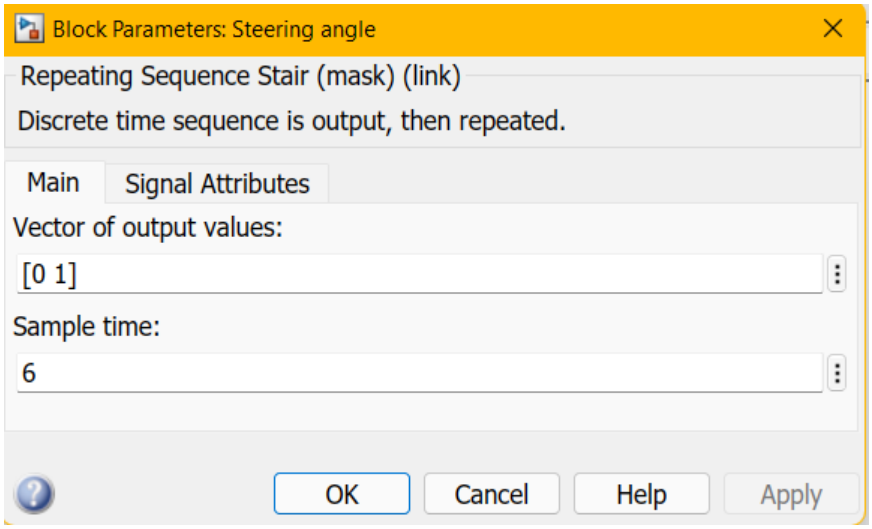


Figura 36: Vector y sample time modificados para trayectoria circular

## 12.2. Resultados

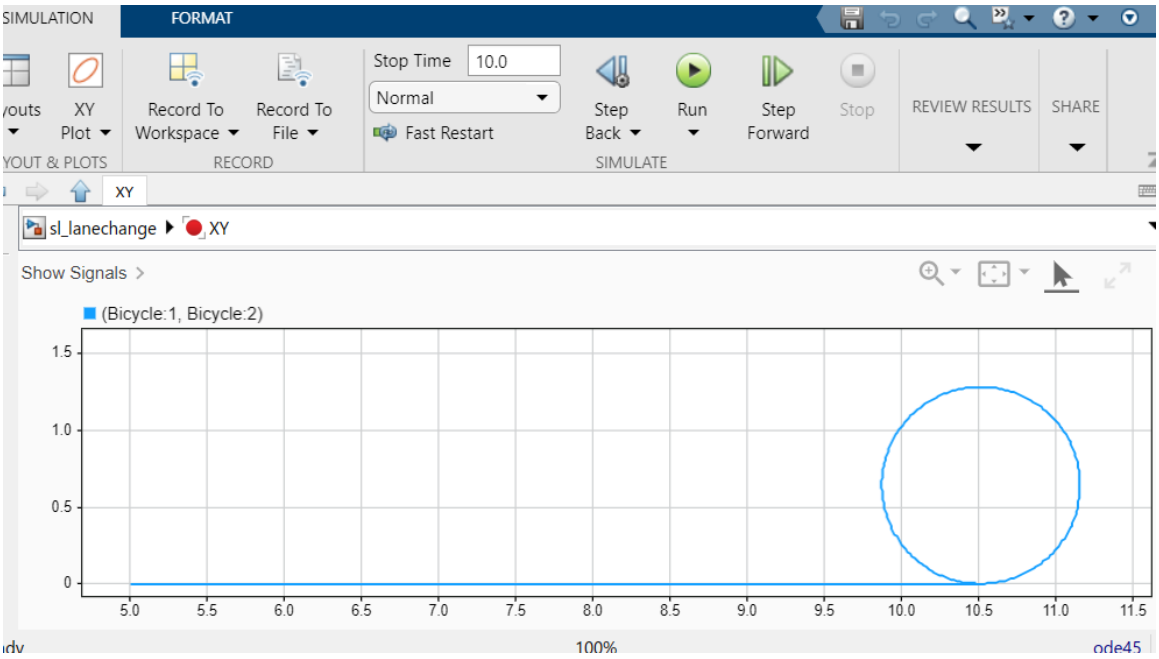


Figura 37: Trayectoria circular