Universidad Politécnica de Valencia

Escuela Técnica Superior de Ingeniería Informática

Grado en Ingeniería Informática

Asignatura:

Mecatrónica

Asignación:

Práctica No.6

Profesor:

Alejandro Vignoni

Integrantes:

Almengor, Alexander

Silgo, Juan José.

Grupo:

PL-1_OB1

Fecha:

18 de abril del 2021

Tabla de Contenido

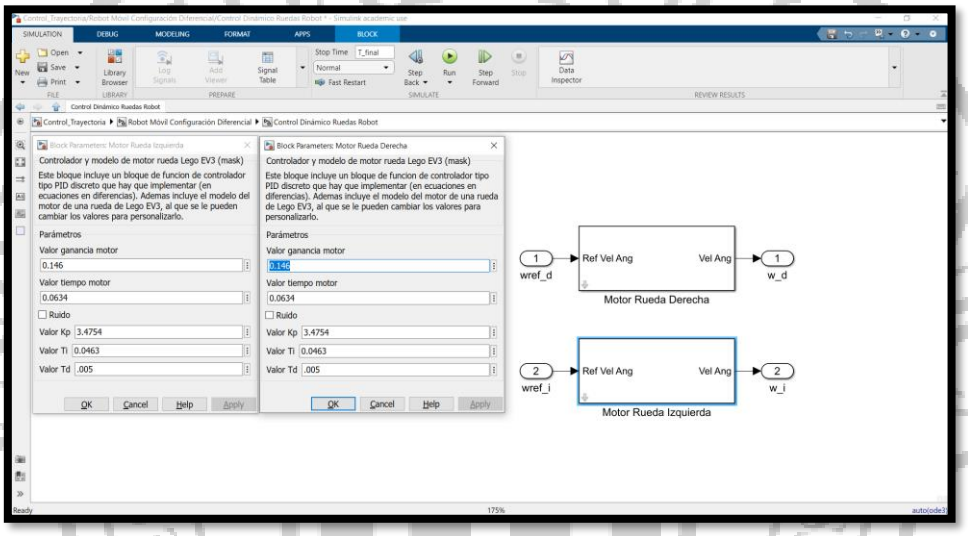
Memoria de la Práctica No. 6

Parte A | Simulink

1. Ejercicio No.1

Integración del control PID de velocidad angular del motor del Lego EV3 (Práctica 5) en el modelo cinemático de un robot con configuración diferencial (creado en la Practica 2).

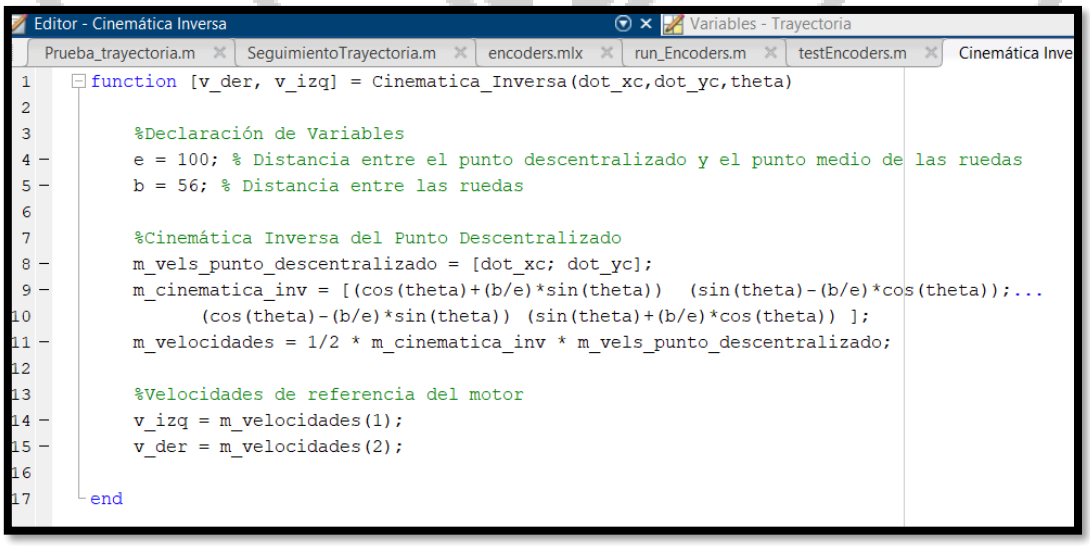
Se agregaron los parámetros obtenidos en la práctica 4 y 5 respectivamente al modelo cinemático del robot.



Parámetros de Ambos Motores

2. Ejercicio No.2

Implementación de la cinemática inversa de punto descentralizado ($x\hat{c}$, $y\hat{c}$) como bloque generador de las referencias de velocidad de las ruedas del robot (vL , vR).



3. Ejercicio No.3

Implementación del control de seguimiento de trayectoria tipo proporcional con velocidad (PV) para generar las derivadas de la posición del punto descentralizado ($\dot{xc}$, $\dot{yc}$) a partir de la configuración actual del robot x, y, θ y de la trayectoria de la referencia ($xref$, $yref$) y sus derivadas ($\dot{xref}$, $\dot{yref}$).

Se decidió implementar este segmento del circuito en una función de Matlab, la cual replica el comportamiento de la ecuación de la Ley de control cinemático proporcional con pre-alimentación de velocidad del punto descentrado, en función de la configuración del robot (x, y, theta).

Editor - Controlador PV | Variables - Trayectoria

Prueba_trayectoria.m | Cinemática Inversa | Controlador PV

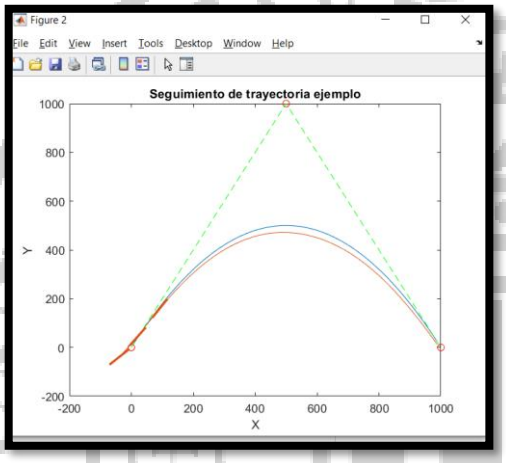
```
1   %Controlador Proporcional Velocidad
2   function [dot_xc, dot_yc] = Controlador_pv(trayectoria,theta, x, y)
3
4       %Ley de Control cinemático Proporcional con pre-alimentación de velocidad
5       %del punto de centrado en función de la configuración del robot (x,y,theta)
6
7       %Declaración de Variables
8       Kx = 4.5; %Ganancia con respecto al error en x
9       Ky = 4.5; %Ganancia con respecto al error en y
10      e = 100;%Distancia entre el punto descentralizado y el punto medio de las ruedas
11      ref = trayectoria(1:2); %Trayectoria de x,y
12      dot_ref = trayectoria(3:4); %Derivadas de x,y de la trayectoría
13      m_ganancia = [Kx 0; 0 Ky]; %Matriz constante de ganancia
14
15      %Matriz de derivadas de x,y de la trayectoría
16      m_dot_ref = [dot_ref(1);dot_ref(2)];
17
18      %Matriz del error del punto de centrado
19      e1 = ref(1) - (x + e*cos(theta)); % Primer elemento de la matriz
20      e2 = ref(2) - (y + e*sin(theta)); % Segundo elemento de la matriz
21      m_error_punto_centrado = [e1;e2];
22
23      % Se múltiplican para obtener los valores de XC, YC
24      m_dot_xyc = m_dot_ref + m_ganancia * m_error_punto_centrado;
25
26      %Derivadas de la trayectoría con respecto a la velocidad para x,y
27      dot_xc = m_dot_xyc(1);
28      dot_yc = m_dot_xyc(2);
29
30  end
```

4. Ejercicio No.4

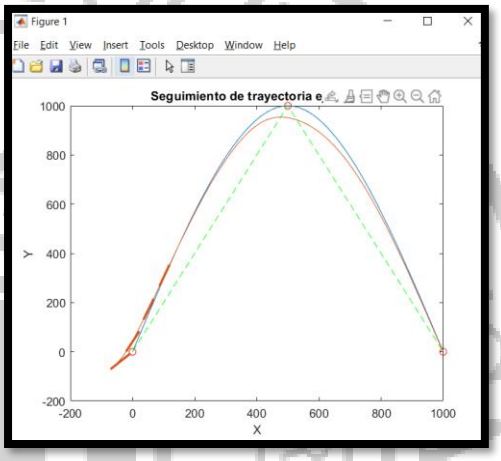
Generación de la trayectoria de la referencia ($xref$, $yref$) y sus derivadas ($x\dot{}ref$, $y\dot{}ref$) utilizando las funciones de la practica 1, cambiando el tiempo real.

Para el desarrollo de este ejercicio se reutilizó las funciones de la práctica No.1 de Curva de Bezier y la Spline Cúbica Natural, tanto para tres puntos y cuatro puntos, se utilizo como puntos de control los brindados en la práctica.

Curva de Bezier y Spline Cúbica Natural para 3pts

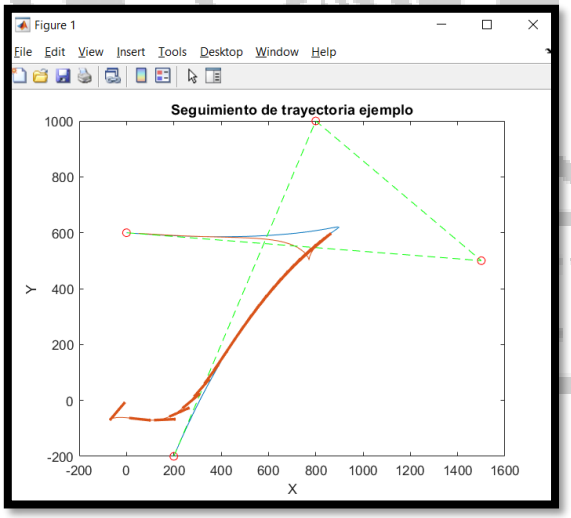


Bezier

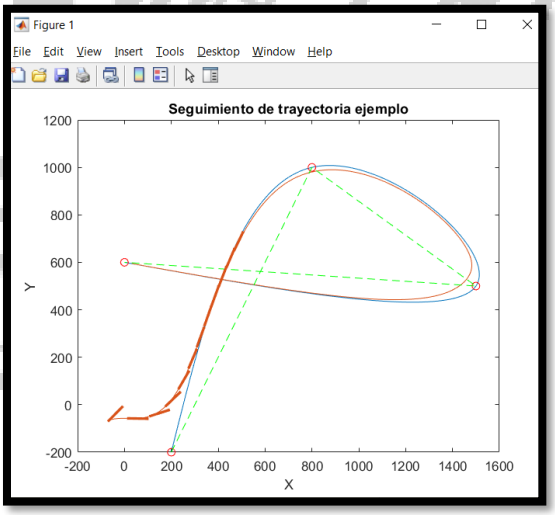


Spline

Curva de Bezier y Spline Cúbica Natural para 4pts



Bezier



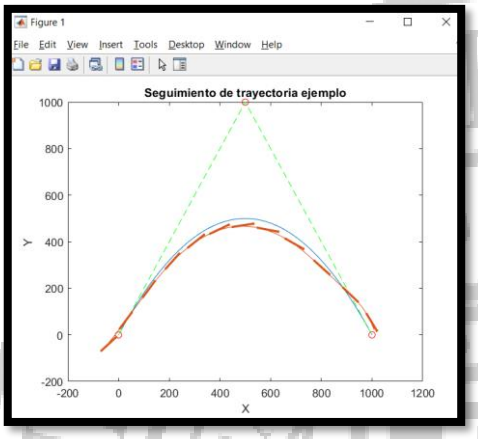
Spline

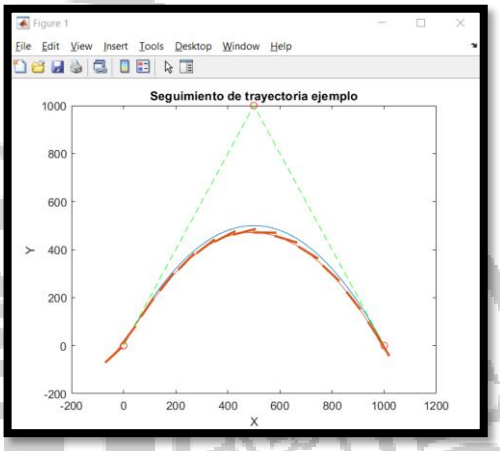
5. Ejercicio No.5

Seguimiento de trayectoria de los puntos propuestos: Realizar un script que defina los puntos de partida y llame a la función correspondiente para obtener las curvas de aproximación. Luego llame a la ejecución del simulink. Calcula el Error (como función del tiempo) y el ITAE de la trayectoria obtenida. Incluye la representación gráfica de los puntos de partida y las curvas generadas

a) PtosControlA [0 0; 1000 500;0 1000]; %mm Total_t = 4; %Segundos

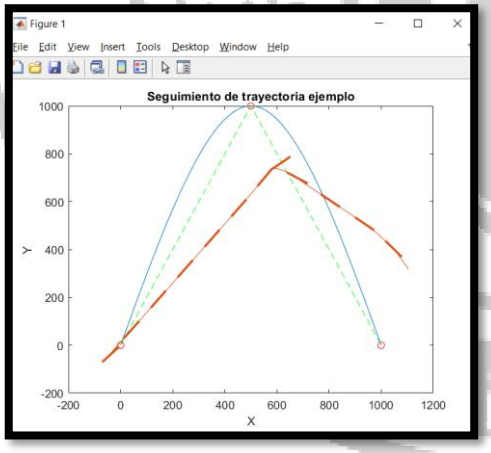
Bezier

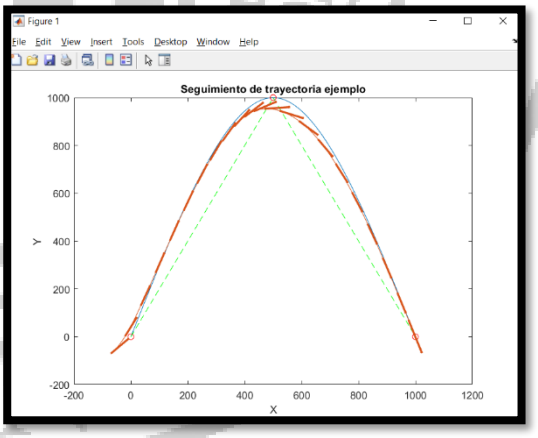




En el gráfico izquierdo podemos ver que al realizar la prueba utilizando el algoritmo de Bezier para tres puntos, empleando 4 segundos se pierde precisión al final del trayecto; sin embargo, si aumentamos el tiempo en un segundo, es decir 5 segundos y utilizando una ganancia de 4.5 para los valores de las variables kx y ky, se puede mejorar la precisión.

Spline



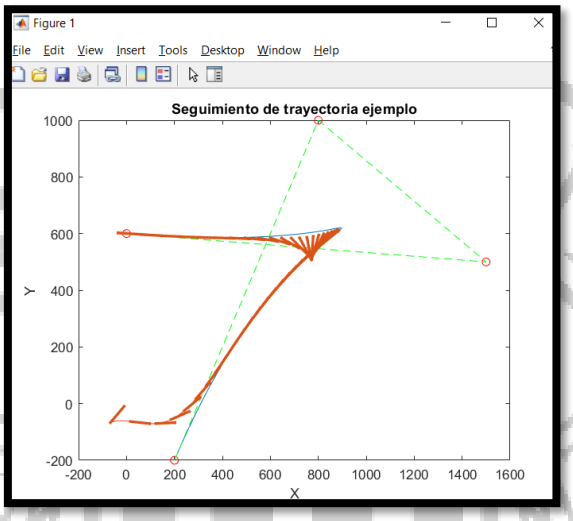


En el caso del cálculo de las trayectorias utilizando Spline, podemos observar que para un tiempo de 4 segundos y una ganancia de 4.5 para las variables kx y ky, el seguimiento de la trayectoria perdía precisión; sin embargo, al ajustar los parámetros del tiempo a 8 segundos, es decir el doble, se obtuvo un mejor resultado más preciso.

b) PtosControlB=[200 -200; 800 1000; 1500 500; 0 600]; %mm Total_t = 20; %Segundos

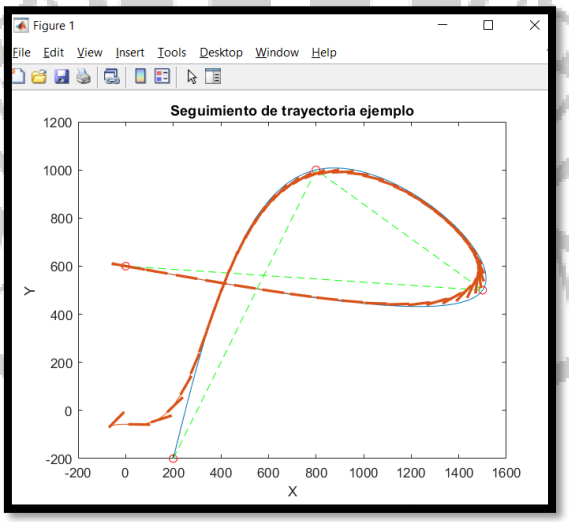
Bezier

Al realizar el seguimiento de trayectoria utilizando el algoritmo de Bezier para cuatro puntos, y una ganancia para las variables kx y ky de 5, obtuvimos un seguimiento de la trayectoria bastante preciso, esto también se debe a que el tiempo es de 20 segundos, a mayor tiempo mejor la precisión.



Spline

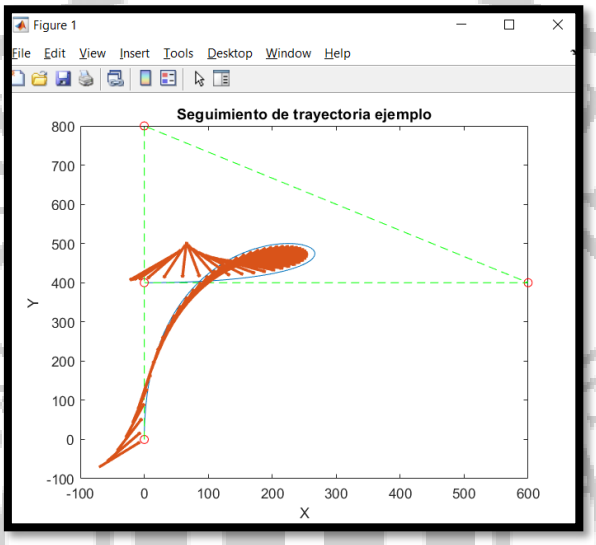
Utilizando el algoritmo de Spline para cuatro puntos obtuvimos un seguimiento de trayectoria preciso.



c) PtosControlC= [3 3;3 6;5 5;3 5]; mm Total_t = 20; %Segundos (Banderín de Golf);

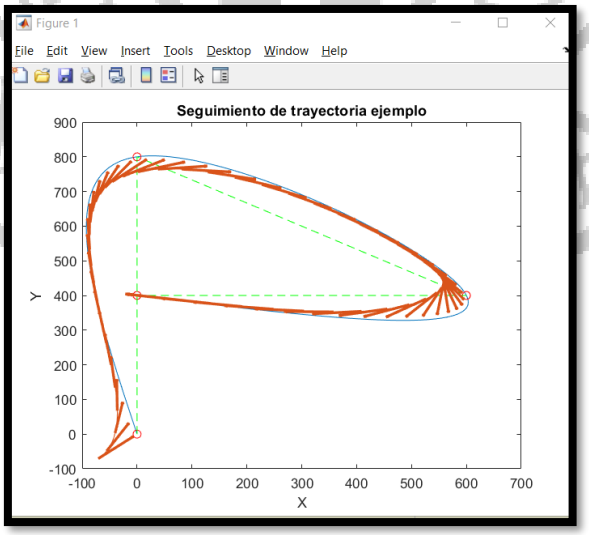
Bezier

Para la prueba realizada con Bezier intentado seguir una trayectoria en forma de banderín de Golf, los resultados nos muestran que a 20 segundos con una ganancia en las variables kx y ky de 3, la trayectoria seguida se aproxima bastante al último punto, pero como el comportamiento natural de Bezier es de una aproximación, se puede ver que el recorrido termina antes.

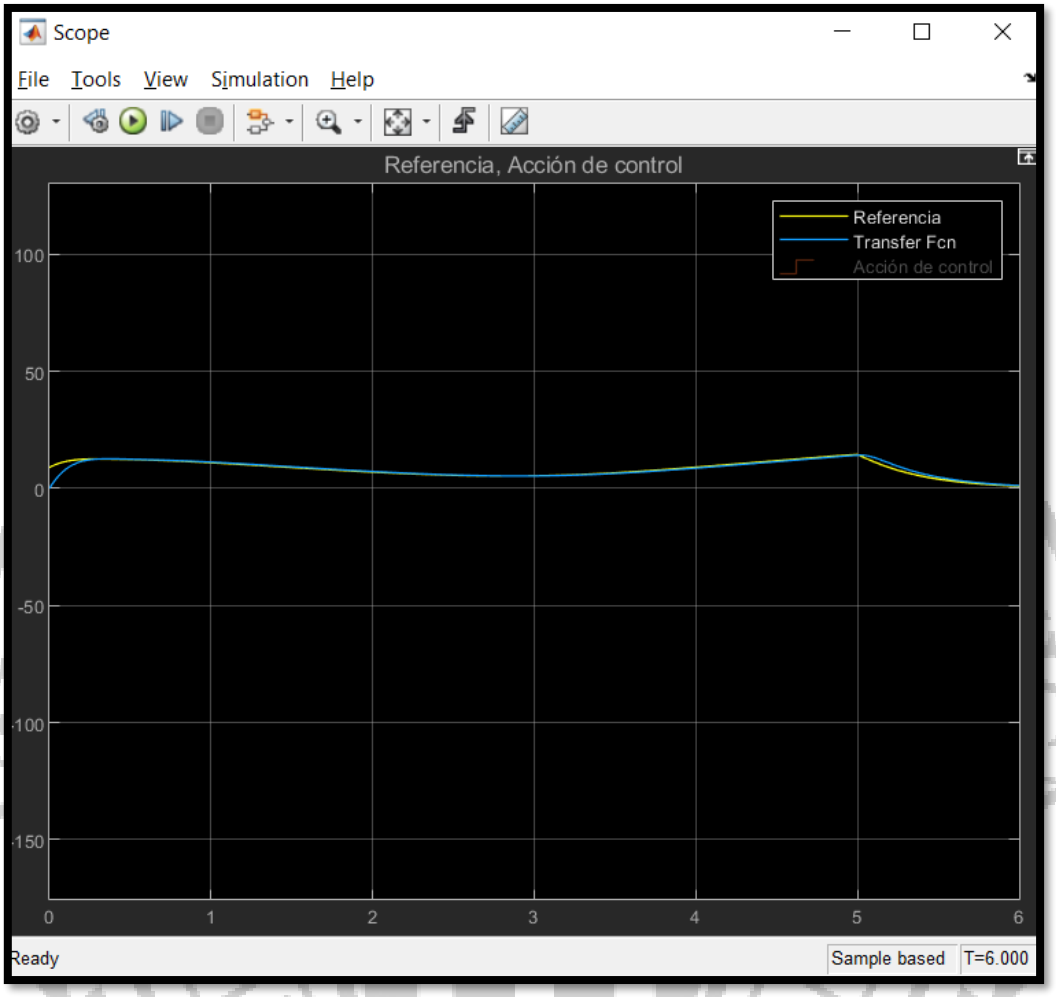


Spline

Al repetir el mismo ejercicio con los mismos parámetros salvo que los valores de las variables kx y ky, fueron de 4.5, y empleando un algoritmo de generación de Trayectorias de movimiento como Spline, el recorrido fue mucho más preciso y se recorre por completo la figura.



ITAE



Como se puede observar en la imagen el robot apenas tiene un retardo inicial en llegar a la referencia, manteniéndose a una distancia despreciable de ella durante el resto de la trayectoria. Con esto concluimos que, aunque se precisaría de un cálculo exhaustivo o una medición dedicada para visualizar correctamente este valor, el ITAE es mínimo en este caso, entendiendo que es aceptable.

Parte B | CoppeliaSim

1. Algoritmo del Punto descentrado en Matlab

Implementar al algoritmo del punto descentrado en Matlab siguiendo el esqueleto del archivo SeguimientoTrayectoria.m completando las líneas correspondientes.

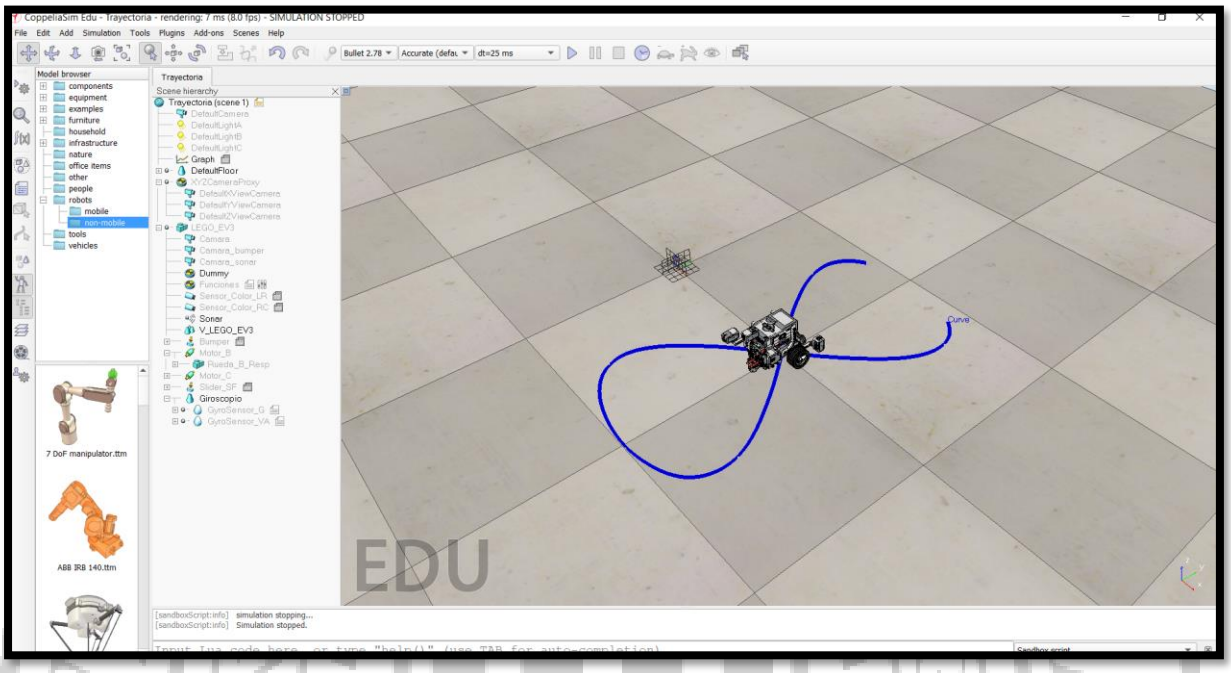
Se implementó el algoritmo de punto descentrado en Matlab.

```
%% Seguimiento trayectoria punto descentralizado

    % obtenemos el valor de los encoders (grados) para ver cuanto se ha movido
    posruedaDerecha = double(MotorRotationCount(OUT_C));
    posruedaIzquierda = double(MotorRotationCount(OUT_A));

    % calculamos las velocidades angulares (rad/s) de las ruedas // CONVERTIR A RADIANES
    wdk= (posruedaDerecha - posruedaDerecha1)*(pi/180)*(1/Ts);
    wik= (posruedaIzquierda- posruedaIzquierda1)*(pi/180)*(1/Ts);%derivada Euler

    % calculamos las velocidades lineales (mm/s) de las ruedas: v = w*radio
    vdk= wdk * radiorueda;
    vik= wik * radiorueda;

   % calculamos la velocidad lineal del robot (mm/s)
    vk =(vdk+vik)/2; %promedio

    % calculamos la velocidad angular del robot (rad/s)
    wk = (vdk-vik)/(2*b);

    % estimamos la posicion X-Y (mm) y la orientacion del robot (rad)
    x=x + vk*Ts*cos(theta);
    y=y+vk*Ts*sin(theta);
    theta=theta+wk*Ts;

    % calculamos la velocidad del punto descentralizado a partir del control cinematico del
robot (mm/s) %pag 18
    velxp= velxref + kx * (xref-(x+e*cos(theta)));
    velyp= velyref + ky * (yref-(y+e*sin(theta)));



    % calculamos las velocidades lineales de la ruedas que debera aplicar el robot a partir
del modelo cinematico inverso del robot (mm/s) %pag17
    vi= (((e*cos(theta)+b*sin(theta))*velxp)+((e*sin(theta)-b*cos(theta))*velyp))*(1/e);
    vd= (((e*cos(theta)-b*sin(theta))*velxp)+((e*sin(theta)+b*cos(theta))*velyp))*(1/e);

    % calculamos las velocidades angulares de referencia para el control dinamico (rad/s)
    wref_d= vd/radiorueda;
    wref_i= vi/radiorueda;

    % mandamos las acciones de control a aplicar a cada rueda (en % actuación)
    OnFwd(OUT_C,((100*wref_d)/18.32)); % motor derecha
    OnFwd(OUT_A,((100*wref_i)/18.32)); % motor izquierda

    % almacenamos los valores de los encoder para la proxima iteracion
    posruedaDerecha1 = posruedaDerecha;
    posruedaIzquierda1 = posruedaIzquierda;

    % iteración
    i=i+1;

    % calculo indice integral error cuadratico
    ex= xref - x;
    ey= yref - y;
    errorcua= errorcua + (ex * ex+ ey * ey);



    %Guardar datos
    datos=[datos;xref yref x y errorcua];

    t2=double(CurrentTick());
    % Espera hasta el siguiente periodo
    espera=(max(0,Ts*1000-(t2-t1)));
    Wait(espera);
```

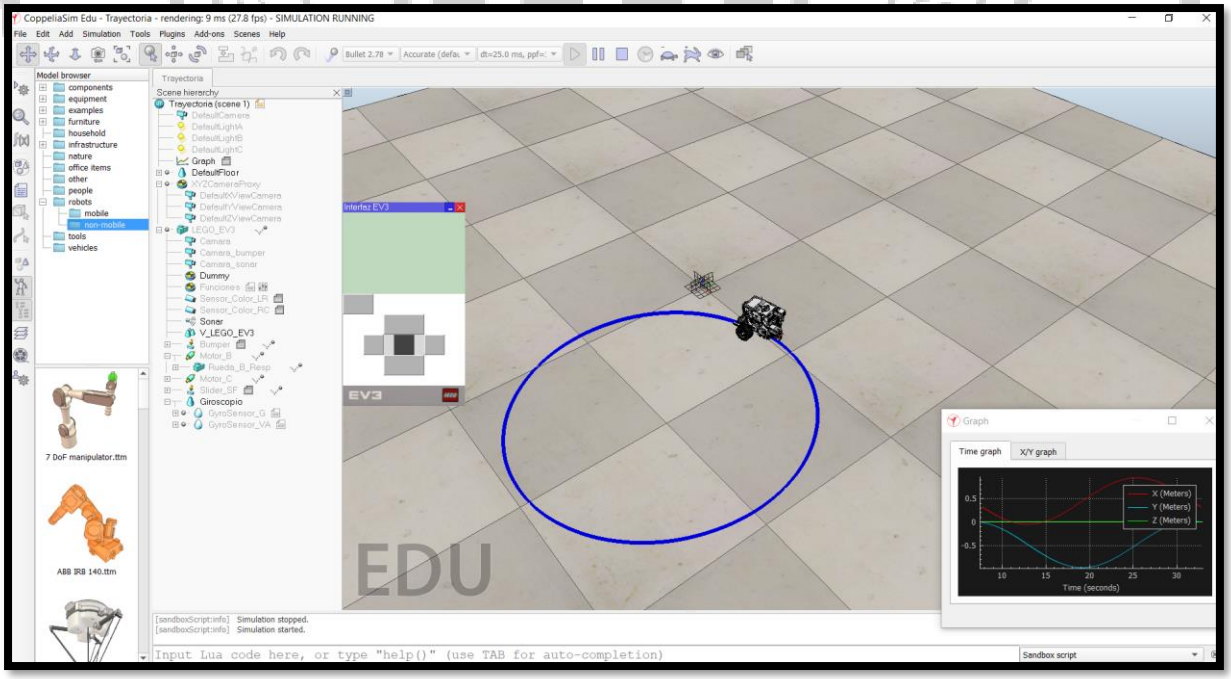
2. Simulación en CoppeliaSim

Simular en CoppeliaSim utilizando la escena Trayectoria.ttt y el código de comienzo StartSeguimientoTrayectoria.m

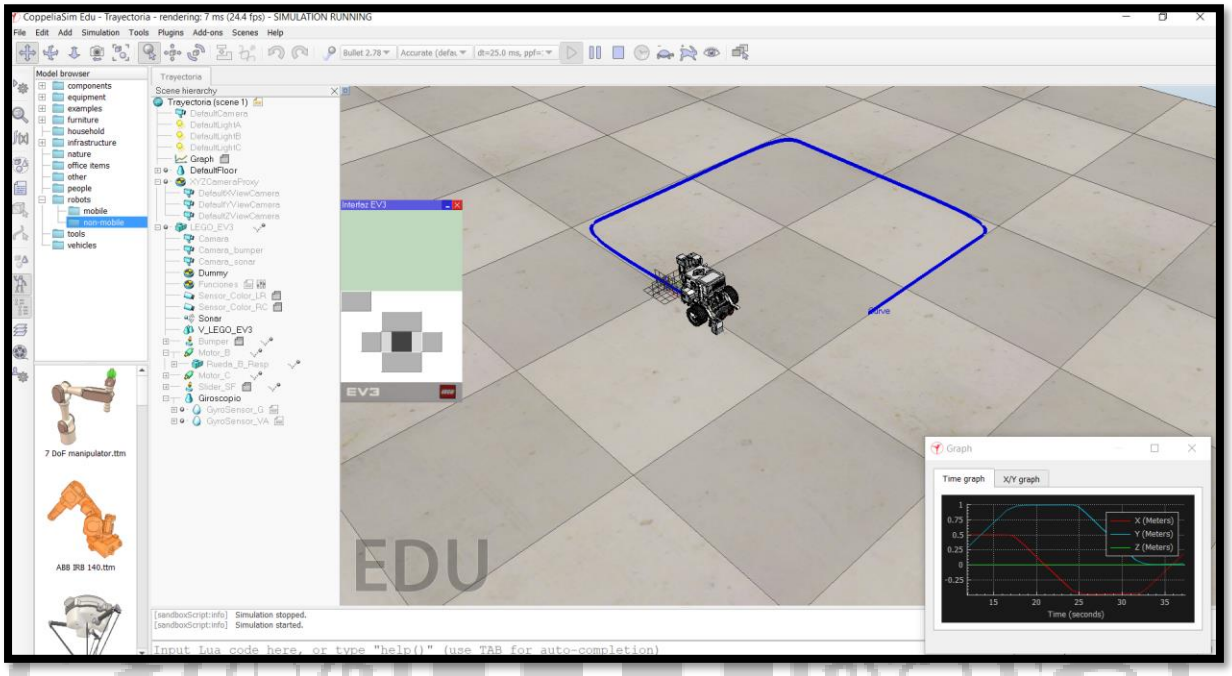
2.1 Trayectoria Infinito



2.2 Trayectoria Circular



2.3 Trayectoria Cuadrada

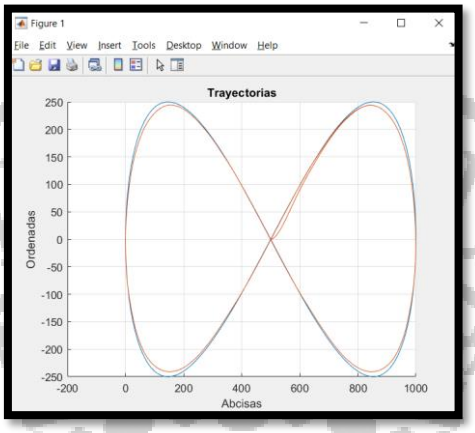


3. Gráficos de Trayectorias

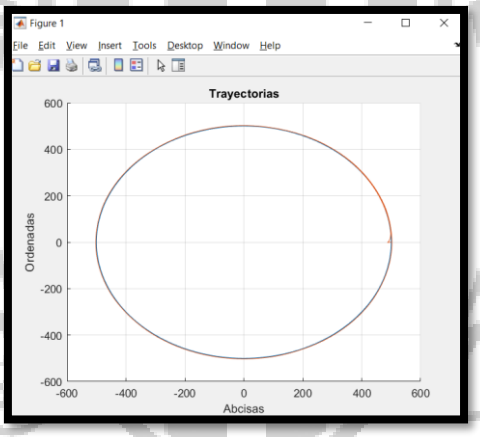
Graficar en la misma gráfica la trayectoria deseada del robot y la obtenida por odometría.

Se muestra en los gráficos la trayectoria de referencia representada con la línea roja, y con la azul la obtenida por el robot mediante odometría.

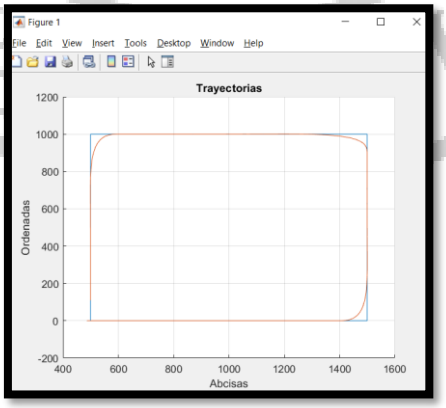
3.1 Trayectoria de Infinito



3.2 Trayectoria Circular



3.3 Trayectoria Cuadrada



4. Error Cuadrático Medio de Seguimiento

Calcular el error cuadrático medio del seguimiento de trayectoria obtenido

Al desarrollar la práctica se cálculo el error cuadrático medio que utiliza la función mean, indexando los datos recopilados de la matriz datos en la columna número cinco.

```
%Declaración de Variables
xref = datos(:,1);
yref = datos(:,2);
x = datos(:,3);
y = datos(:,4);
error_cua_medio = mean(datos(:,5));
```

4.1 Trayectoria de Infinito

```
error_cua_medio =

   7.5097e+05
```

4.2 Trayectoria Circular

```
error_cua_medio =

   7.7595e+05
```

4.3 Trayectoria Cuadrada

```
error_cua_medio =

   3.4471e+06
```