PRÁCTICA 2: CONTROL BORROSO

Pablo Acereda García
Department of Computer Science
University of Alcala
28805 - Alcalá de Henares, Madrid, Spain
pablo.acereda@edu.uah.es

Laura Pérez Medeiro
Department of Computer Science
University of Alcala
28805 - Alcalá de Henares, Madrid, Spain
l.perezm@edu.uah.es

September 12, 2020

Introducción

Esta práctica estudiará el diseño de controladores borrosos que permitan controlar la posición de un robot. Esta práctica será dividida en dos partes, una primera parte para el control de la posición de un robot; y una segunda parte para evitar obstáculos con un robot - todo ello mediante el desarrollo de un controlador borroso.

Se hará uso de MATLAB y Simulink para esta práctica. Además, será necesario la herramienta `Fuzzy Logic` para diseñar los controladores.

1 Diseño de un control borroso de posición para un robot móvil

Se parte de un entorno, proporcionado por los profesores de esta asignatura, con dimensiones de 10x10m con origen de coordenadas en el centro de dicho entorno. No hay ningún obstáculo en este entorno.

Se ha de diseñar un controlador borroso de manera que sea capaz de alcanzar una posición descrita mediante las coordenadas de referencia *refx* y *refy*.

El punto de inicio es un esquema previamente proporcionado ([1]). De este esquema se va a sustituir el controlador[1] por un controlador borroso.

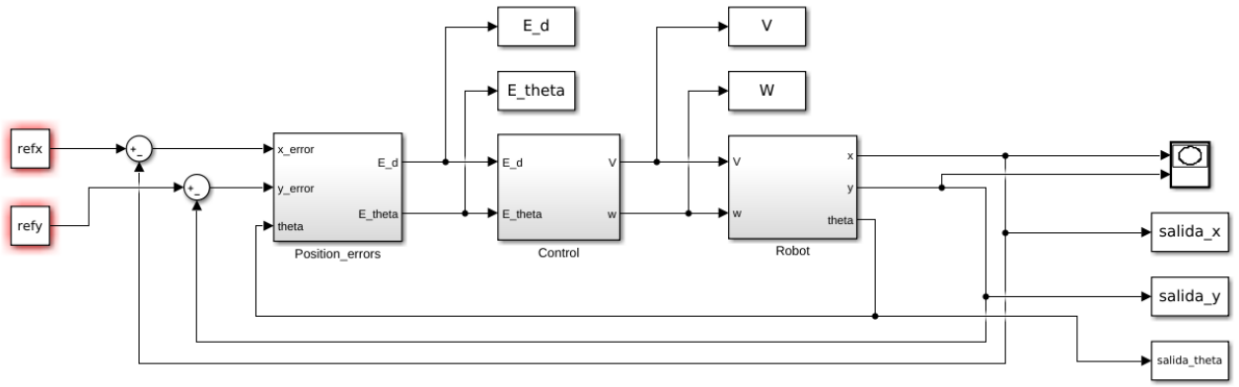


Figure 1: Esquema general de control de posición

[1] Corresponde al bloque *Control* de la figura 1.

Movimiento del robot

Tal y como se ha descrito en práctica anteriores, el robot sigue una serie de funciones para describir su posición (en coordenadas x, y) y su orientación (descrito por θ). Se emplean las siguientes funciones con el objetivo descrito anteriormente:

Posición:

$$X_k = X_{k-1} + V_{k-1} T_s \cos(\theta_{k-1}) \tag{1}$$

$$Y_k = Y_{k-1} + V_{k-1} T_s \sin(\theta_{k-1}) \tag{2}$$

Orientación:

$$\theta_k = Y_{k-1} + \omega_{k-1} T_s \tag{3}$$

Respecto a los operando de cada una de estas funciones se tiene que:

- V_k: velocidad lineal.
- ω: velocidad angular.
- T_s: tiempo de muestreo.

Pero estas funciones de por sí no son suficientes para controlar el estado del robot. También se necesitará calcular los errores de las funciones calculadas anteriormente: error de distancia (E_d) y error de ángulo (E_θ). Se busca tener un mayor control de la posición del robot. Dichos errores se calculan a partir de las siguientes fórmulas:

$$E_d = \sqrt{(x_r - x_k)^2 + (y_r - y_k)^2} \tag{4}$$

$$E_\theta = atan2(y_r - y_k, x_r - x_k) - \theta_k \tag{5}$$

En E_θ, *atan2* hace referencia a la función arco-tangente sensible a los diferentes cuadrantes que tenga el ángulo que se le proporciona.

Diseño del controlador

El objetivo es diseñar un controlar borroso que toma como entrada los errores mencionados anteriormente (distancia y ángulo). Como salida, proporciona la velocidad lineal y angular que el robot deba usar en ese momento.

El bloque queda representado por la figura 2.

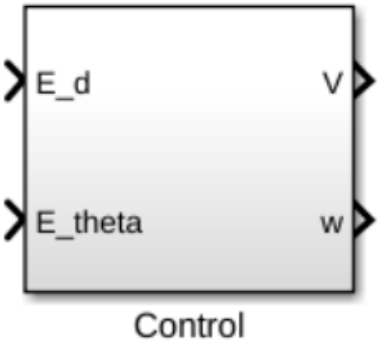


Figure 2: Subsistema de control

Para diseñar este bloque en Simulink se deben seguir los siguientes pasos:

1. Se teclea dentro de la terminal de MATLAB el comando `fuzzy` para abrir así la ventana de diseño de nuestro controlador borroso ([3]).

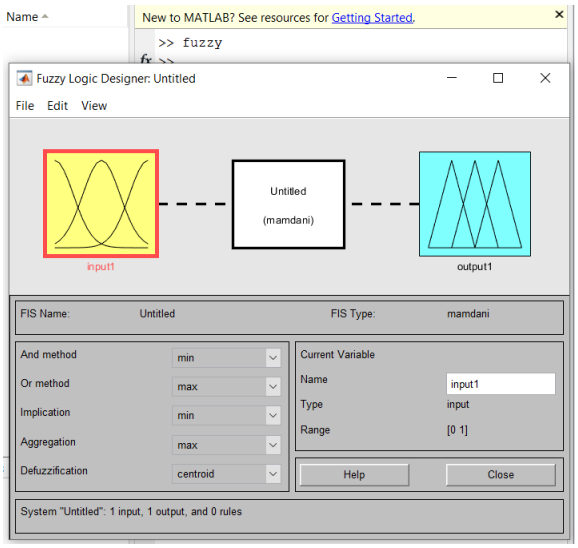


Figure 3: Comando fuzzy y ventana de diseño Fuzzy Logic

2. Dentro de la ventana que aparece, y en los selectores de la parte izquierda, se mantienen: los valores de las operaciones *AND* como mínimos, las operaciones *OR* como máximos y el desborrosificador como *centroid*. El resto de valores también se mantienen. Debería tenerse en cuenta que la desborrosificación es el proceso de representar un conjunto difuso con un número nítido y el método *centroid* o método de *centro de área (COA)* es el método más extendido. Otros métodos que se utilizan para la desborrosificación son el *centro de sumas (COS)* y el método de la *media del máximo*. Además MATLAB ofrece también la opción de añadir un método de desborrosificación personalizado. Se eligieron estos valores por defecto por cuestión de simplicidad y de preferencia de los desarrolladores de esta práctica.
3. Para mantener la lógica proporcionada por el enunciado y tal y como se muestra en el bloque *Control* del esquema, se deben añadir una variable de entrada y de salida adicionales a las proporcionadas por defecto por la plantilla de *fuzzy*. Tal y como se muestra en la figura 4:

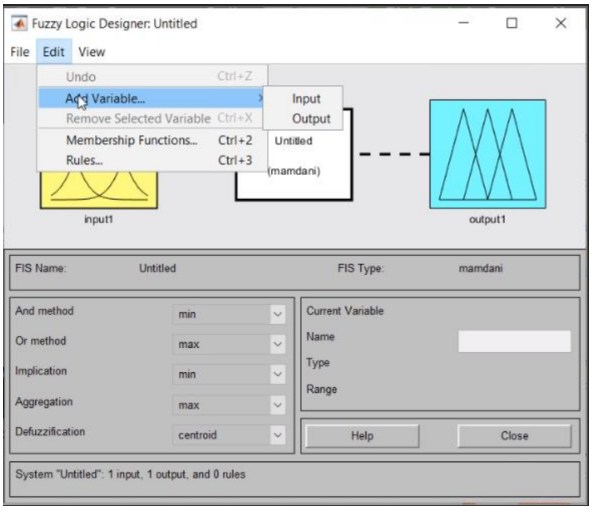


Figure 4: Añadir variables de entrada y salida

4. Posteriormente se renombran las variables de manera que sean más identificables y que tengan el mismo nombre que las proporcionadas en el esquema, bloque de *Control*. Tal y como se muestra a continuación (figura [5]):

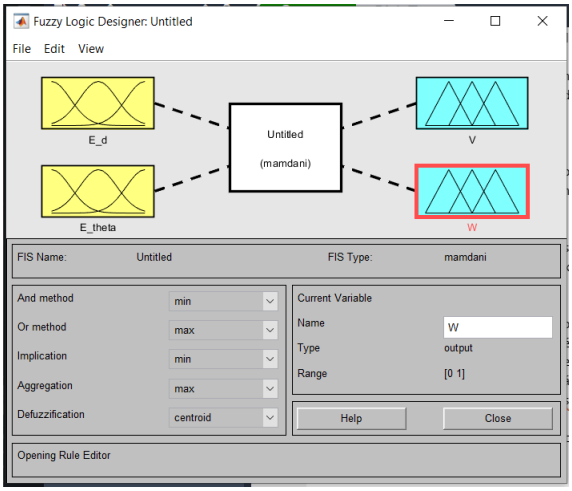


Figure 5: Renombrar las variables

5. El siguiente paso es ajustar los rangos de las funciones de pertenencia de cada variable de manera que el robot sea capaz de seguir un circuito.

Table 1: Rangos de pertenencia.

Variable	Min Range	Max Range
E_d	0	10
E_{theta}	$-\pi$	π
V	0	2
W	-1	1

6. Ahora que las funciones ya tienen asignados unos rangos, es momento de ajustar las funciones de pertenencia. Para ello, se sigue la siguiente tabla:

Table 2: Functions de pertenencia.

Variable	Función
E_d (PEQUEÑA)	[0 0 5]
E_d (MEDIA)	[0 5 10]
E_d (GRANDE)	[0 10 10]
E_{theta} (NEGATIVO)	[-3.14 -3.14 0]
E_{theta} (POSITIVO)	[0 3.14 3.14]
V (PEQUEÑA)	[0 0 1]
V (MEDIA)	[0 1 2]
V (GRANDE)	[1 2 2]
W (NEGATIVO)	[-1 -1 0]
W (POSITIVO)	[0 1 1]

Si se quieren más detalles, echar un vistazo a las imágenes a continuación:

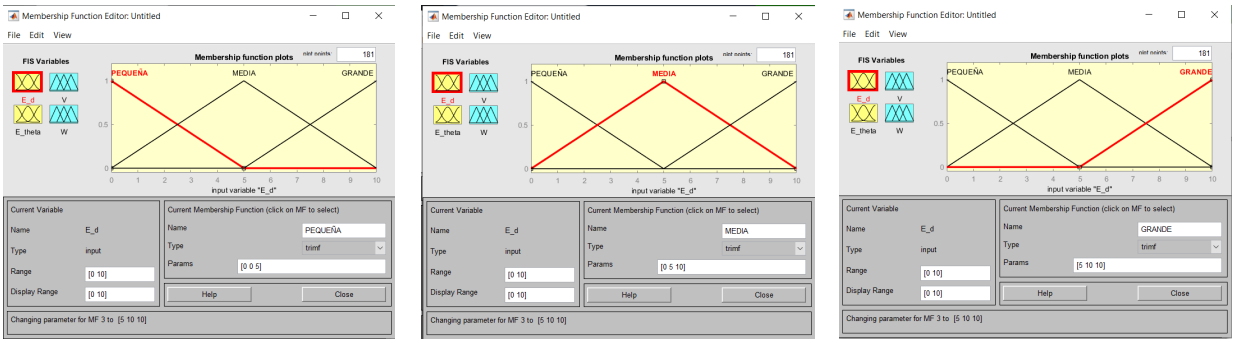


Figure 6: Valores para el error de distancia

Figure 7: Valores para el error angular

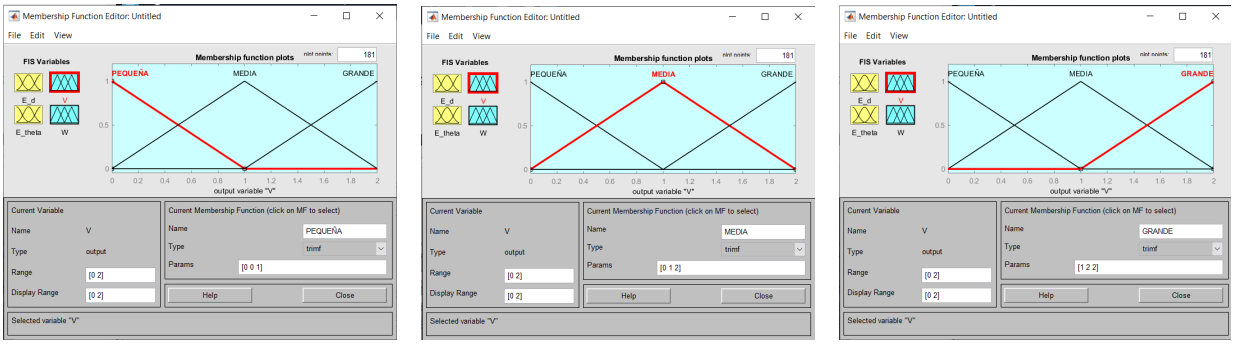


Figure 8: Valores para la velocidad lineal

Figure 9: Valores para la velocidad angular

7. Una vez se tienen creadas las funciones de pertenencia para cada variable, se crean las reglas para el controlador. Se decide crear reglas simples, ya que son más sencillas e intuitivas a la hora de predecir el comportamiento que seguirá el robot. Las reglas creadas son las siguientes (figura [10]):

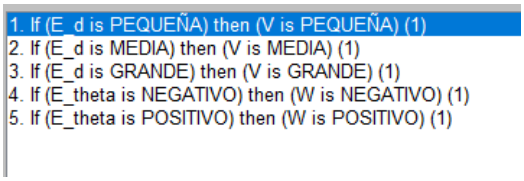


Figure 10: Añadir variables de entrada y salida

8. Después se creará el subsistema que conformará el bloque del controlador borroso. Posteriormente deberá introducirse en el esquema con el resto de componentes. Este fichero es el nombrado como *PositionControl.slx*. El resultado se muestra en la siguiente figura ([11]):

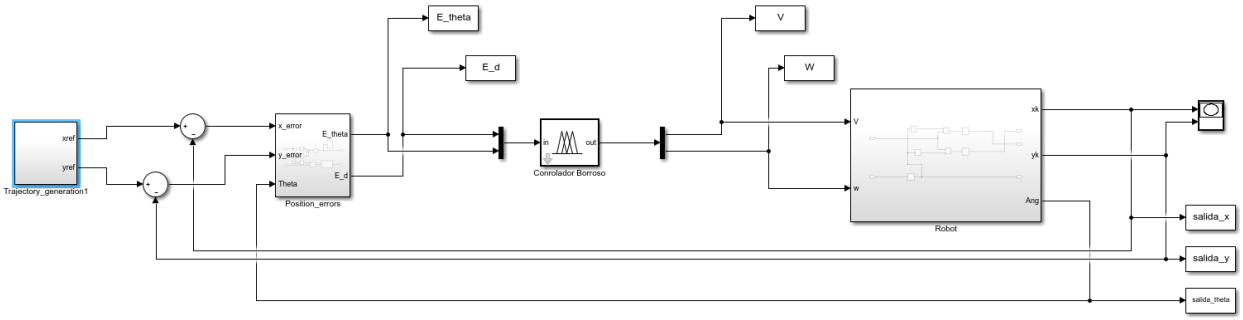


Figure 11: Esquema de seguimiento de trayectoria con controlador borroso

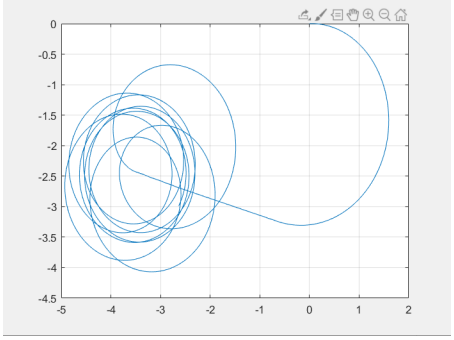
9. Por último se creará un script de MATLAB que permita realizar pruebas con posiciones *refx* y *refy* aleatorias que sirva para comprobar el correcto funcionamiento del controlador antes de introducir el generador de trayectorias. El script a utilizar será el siguiente:

```
%  PRACTICE 3. FUZZY CONTROL
% The aim of the practice is strudying fuzzy control design in order to
% control robots which are in move.
% The practice will be divide in two parts:
        % PART 1. Designing a fuzzy control to control the position
        % PART 2. Designing a fuzzy control to avoid abstacles
% LINES THAT ALWAYS HAVE TO BE:
% clear variables,
% close opened windows
% and clean the comand windows
%--------------------------------------------------------------------------%
clear variables;
close all;
clc;
%--------------------------------------------------------------------------%
%Tiempo de muestreo
Ts=100e-3;
% Referencia x-y de posicion
x_0 = 0;
y_0 = 0;
th_0 = 0;
num=5;
for c = 1:1
    xref=10*rand-5;
    yref=10*rand-5;
    % Ejecutar Simulacion
    sim('PositionControl.slx')
```

```
    % Mostrar
    x=salida_x.signals.values;
    y=salida_y.signals.values;
    v = V.signals.values;
    w = W.signals.values;
    %figure;
    plot(x,y);
    %plot(v,w);
    grid on;
    hold on;
end
```

fuzzyController/PositionControlerCode.m

Siguiendo estos pasos se debería obtener un controlador. Si se ejecuta el script mencionado en el último paso, se deberían obtener los siguientes resultados:

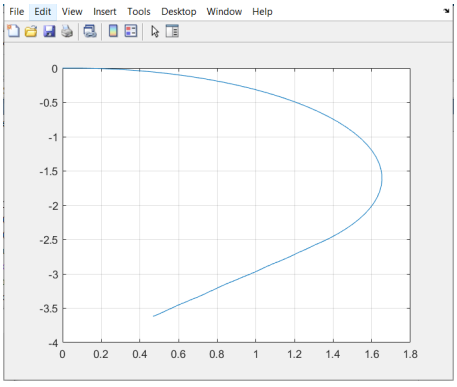


(a) Movimiento realizado

xref	-3.5071
y	*1001x1 double*
y_0	0
yref	-2.4249

(b) Coordenadas x,y para indicar el destino

Figure 12: Primera prueba de los valores



(a) Movimiento realizado

xref	0.4722
y	*67x1 double*
y_0	0
yref	-3.6138

(b) Coordenadas x,y para indicar el destino

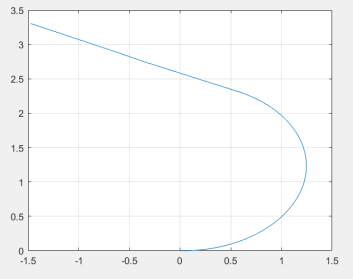
Figure 13: Segunda prueba de los valores

Por lo que se puede observar, este controlador no otorga los resultados esperados, no es completamente funcional. Por lo tanto, será necesario replantear las funciones de pertenencia o reglas usadas hasta ahora. Para ello, se plantea la idea de crear una tercera función de pertenencia para las variables E_θ y W. Esta nueva función corresponda a cuando a cuando el valor es cero ya que se sospecha que es en esos valores las funciones de pertenencia pueden encontrarse con conflictos. Gracias a la inclusión de esta nueva función de pertenencia se puede crear una nueva regla que indique la actuación a seguir por parte del controlador en ese supuesto, por tanto, las funciones de pertenencia finales serían las siguientes:

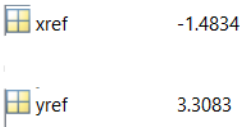
Table 3: Functions de pertenencia.

Variable	Función
E_d (PEQUEÑA)	[0 0 5]
E_d (MEDIA)	[0 5 10]
E_d (GRANDE)	[5 10 10]
E_{theta} (NEGATIVO)	[-3.14 -3.14 0]
E_{theta} (CERO)	[-0.005 0 0.005]
E_{theta} (POSITIVO)	[0 3.14 3.14]
V (PEQUEÑA)	[0 0 1]
V (MEDIA)	[0 1 2]
V (GRANDE)	[1 2 2]
W (NEGATIVO)	[-1 -1 0]
W (CERO)	[-0.005 0 0.005]
W (POSITIVO)	[0 1 1]

Estos valores pueden encontrarse dentro del fichero *ControlBorroso1.fis* y han sido obtenidos mediante experimentación. Con este nuevo controlador se vuelve a ejecutar el script de prueba obteniendo (figuras 14 y 14):

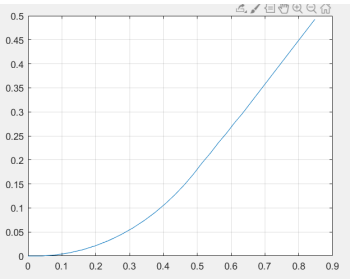


(a) Movimiento realizado



(b) Coordenadas x,y para indicar el destino

Figure 14: Primera prueba nuevos valores



(a) Movimiento realizado

(b) Coordenadas x,y para indicar el destino

Figure 15: Segunda prueba nuevos valores

Se obtienen los resultados esperados dadas las entradas utilizadas por el controlador. Para completar la primera parte de esta práctica, se sustituyen las entradas por *workspace* por entradas introducidas a partir de un bloque generador de trayectoria (proporcionado por los profesores de la asignatura). Sustituyendo las entradas *xref* e *yref* generadas por el script por las salidas producidas por el generador de trayectorias se obtienen los siguientes resultados:

Como se puede observar, el recorrido que sigue el robot es muy similar a la trayectoria creada aunque existen ciertos puntos con desviaciones, sobre todo en la parte del tramo final.

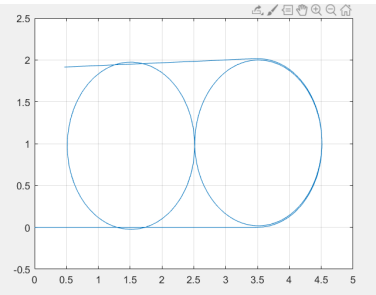


Figure 16: Comportamiento al utilizar el generador de trayectoria

2 Controlador borroso de posición para evitar obstáculos

Con el fin de que nuestro controlador borroso sea capaz de detectar objetos y así poder esquivarlos, se añadirá al esquema de simulink un nuevo bloque que devuelva el error de posicionamiento del obstáculo. Además, partiendo del anterior controlador se deberá añadir dos nuevas entradas y, en consecuencia, se tendrán que crear nuevas reglas. Las entradas nuevas aportarán información acerca del lugar en el que se encuentra el obstáculo para que, en consecuencia, se actúe según las siguientes reglas:

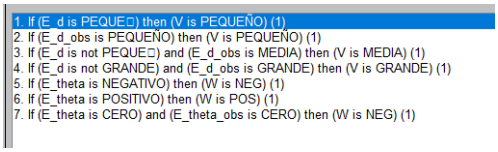


Figure 17: Reglas con las nuevas variables

Esas reglas se obtienen tras estudiar cuál debería ser la actuación en las distintas situaciones. Esto se puede comprender mejor con las siguientes tablas:

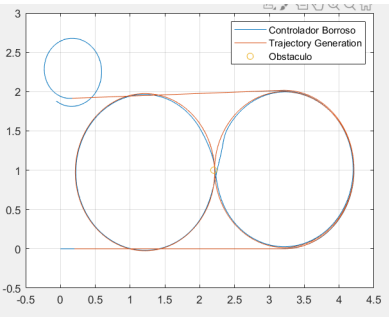
E_d\E_d_pos	Pequeña	Mediana	Grande
Pequeña	Pequeña	Pequeña	Pequeña
Mediana	Pequeña	Mediana	Mediana
Grande	Pequeña	Mediana	Grande

(a) Reglas de velocidad lineal

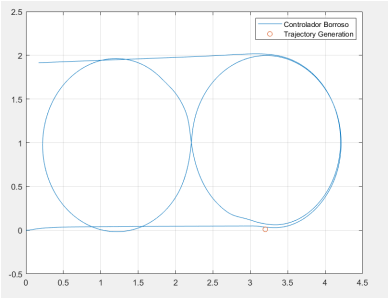
E_theta\E_theta_pos	Negativo	Cero	Positivo
Negativo	Negativo	Negativo	Negativo
Cero	Cero	Negativo	Cero
Positivo	Positivo	Positivo	Positivo

(b) Reglas de velocidad angular

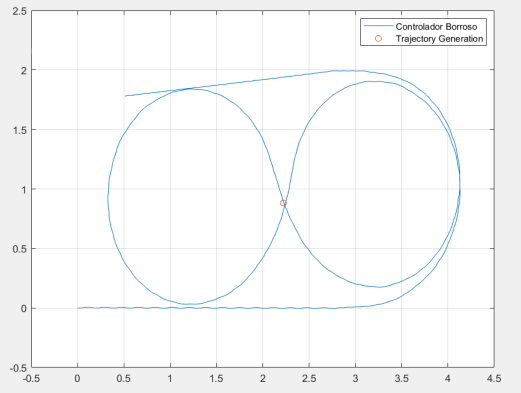
Los valores que toman las funciones de pertenencia de las variables E_theta_obs y E_d_obs son análogos a los que toman las variables E_θ y E_d respectivamente. Con ello se obtienen los siguientes resultados:



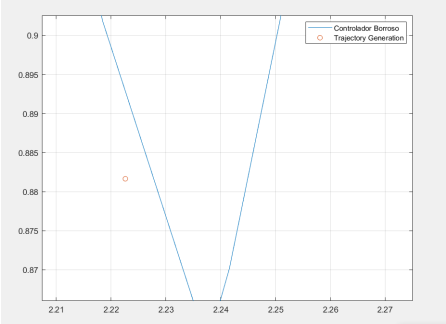
(a) Prueba nº1



(b) Prueba nº2



(c) Prueba nº3



(d) Ampliación prueba nº3

Figure 19: Prueba de las reglas y funciones de pertenencia en un entorno con obstáculos

Por último comentar que el código utilizado para las pruebas de esta parte es el proporcionado por los profesores de la asignatura con la salvedad de añadir en la gráfica una 'o' que represente el lugar en el que se haya el objeto. De esta manera se puede visualizar más fácilmente si la trayectoria es la correcta. También, en la primera prueba se decide mostrar la trayectoria generada por el bloque generador de trayectorias para así comprobar que se sigue realiazando en un inicio la trayectoria deseada.