Universidad de Alcalá

Práctica 2. Control Borroso (I)

Sistemas de Control Inteligente

Grado en Ingeniería Computadores
Grado en Ingeniería Informática
Grado en Sistemas de Información
Universidad de Alcalá

En esta práctica, se estudiará el diseño de controladores borrosos para el control de posición de robots móviles. Se hará uso de Matlab y Simulink como en prácticas anteriores, siendo necesaria la instalación en este caso de la "Fuzzy Logic Toolbox". La práctica consta de dos partes:

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

Parte 2. Diseño de control borroso de posición con evitación de obstáculos

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

1. Objetivo y descripción del sistema.

En este apartado se plantea el control de posición de un robot móvil dentro de un entorno sin obstáculos de dimensiones 10x10 metros y cuyo origen de coordenadas se encuentra en el centro geométrico del entorno.

El objetivo de esta parte de la práctica es diseñar un controlador borroso de manera que el robot sea capaz de alcanzar una posición determinada por las referencias de posición refx y refy.

El esquema de control propuesto se describe mediante el siguiente diagrama de bloques en el entorno Matlab/Simulink:

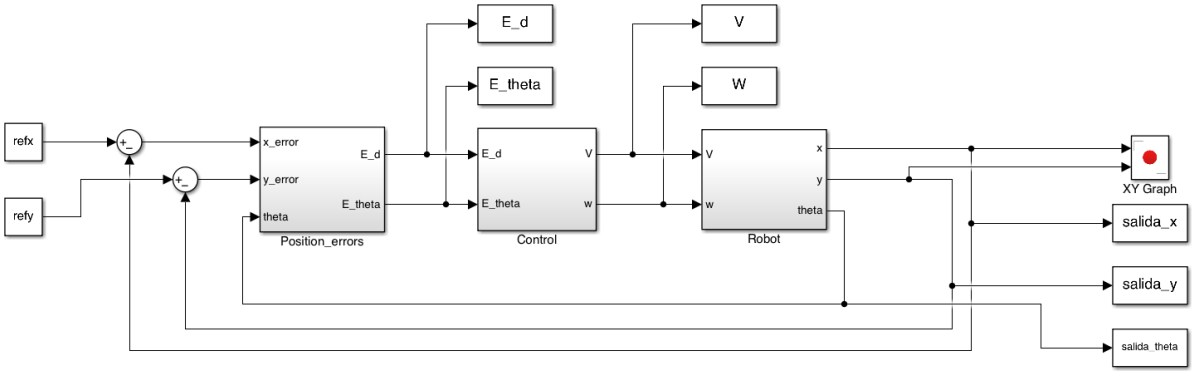

Figura 1. Esquema general de un control de posición.

1.1. Subsistemas que modelan el robot y calculan el error en la posición del robot respecto a una consigna dada: Robot y Position_errors.

Los subsistemas que describen el robot y los errores de posición serán los mismos utilizados en las prácticas anteriores.

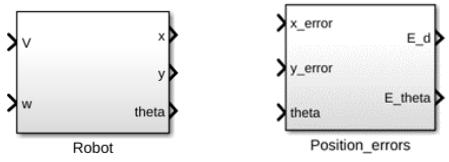

Figura 2. Subsistemas Robot y Cálculo de errores diseñados en prácticas anteriores

Como se describió en la práctica anterior, el movimiento del robot se rige por el siguiente modelo dinámico:

$$x_k = x_{k-1} + V_{k-1}\mathrm{T_s}\cos(\theta_{k-1})$$
$$y_k = y_{k-1} + V_{k-1}T_s\sin(\theta_{k-1})$$
$$\theta_k = \theta_{k-1} + \omega_{k-1}T_s$$

donde $\mathrm{T_s}$ es el tiempo de muestreo utilizado en el sistema, ω_k es la velocidad angular del robot y $\mathrm{V_k}$ es la velocidad lineal.

La referencia de posición del robot, es decir la posición destino u objetivo del robot, se determina mediante las coordenadas x_r, y_r (bloques constantes *refx* y *refx* del esquema general de control de la Figura 1).

Para realizar el control de posición del robot se calculan dos errores:

- Error de distancia al objetivo:

$$E_d = sqrt((x_r - x_k)^2 + (y_r - y_k)^2)$$

- Error de ángulo al objetivo:

$$E_\theta = atan2(y_r - y_k, x_r - x_k) - \theta_k$$

donde atan2 representa la función arcotangente que es sensible a los diferentes cuadrantes del ángulo.

Como en la prática anterior, se deberá añadir una comprobación para que E_theta siempre tenga un valor comprendido en el intervalo [-π,π] y, en caso de que el ángulo esté fuera de estos límites, se deberá realizar la transformación del ángulo obtenido (E_theta) a su ángulo equivalente dentro del intervalo [-π,π].

2. Desarrollo de la práctica

Como se indicó al comienzo del documento, el objetivo de esta primera parte de la práctica es diseñar un controlador de posición basado en la teoría de control borroso. Dicho controlador tiene como entradas los errores de distancia y ángulos comentados anteriormente y genera a su salida los valores de velocidad (angular y lineal) que serán entradas en el bloque de simulación del robot. El controlador borroso diseñado se implementará en un bloque de Simulink como el mostrado en la Figura 3.

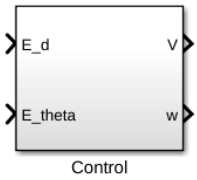

Figura 3. Subsistema de control.

Para ello se piden los siguientes apartados:

a) Ejecute el comando “fuzzy” en la consola de Matlab para crear un controlador borroso mediante la toolbox de Matlab de control borroso. Se abrirá la interfaz que se muestra en la Figura 4.

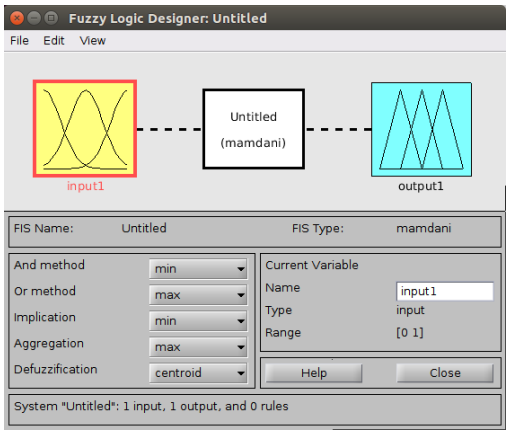

Figura 4. Interfaz de creación de controladores borrosos.

Mantenga las opciones por defecto para las operaciones de AND/OR y el desborrosificador (*Defuzzification*) y siga los siguientes pasos:

i. Cree un sistema borroso con dos entradas y dos salidas mediante el menú "Edit/Add Variable/input" y "Edit/Add Variable/output". Nombre las entradas como "*E_d*" y "*E_theta*" y las salidas como "*V*" y "*W*" tal y como se muestra en la Figura 5.

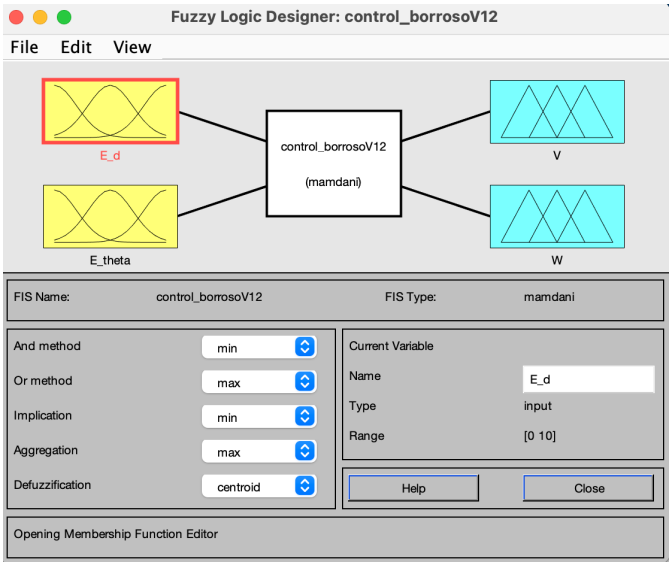

Figura 5. Entradas y salidas.

Inicialmente, considere fijar la velocidad lineal *V* a una velocidad constante y trabaje sobre el control de la velocidad angular *W*. Una vez ajustado este, proceda a implementar el control de la velocidad lineal.

ii. Ajuste las variables de entrada para que dispongan de los siguientes rangos (véase Figura 6):

$$E_d \in [0{,}15], E_\theta \in [-\pi, \pi], V \in [0, 2], W \in [-1{,}1]$$

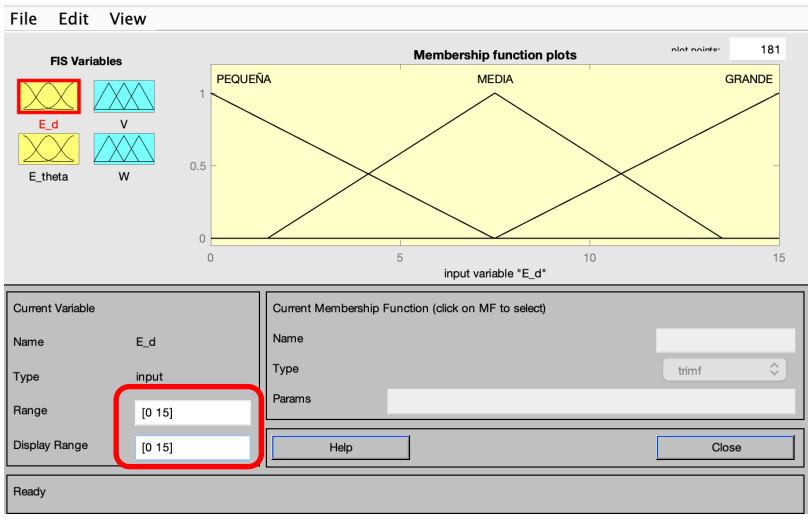

Figura 6. Ajuste de los rangos de las variables de entrada.

iii. Cree las funciones de pertenencia del tipo que considere necesarias en cada una de las variables definidas en la entrada y la salida (véase Figura 7). Utilice nombres para los conjuntos borrosos que tengan un significado claro como NEGATIVO, POSITIVO, GRANDE, PEQUEÑO, etc.

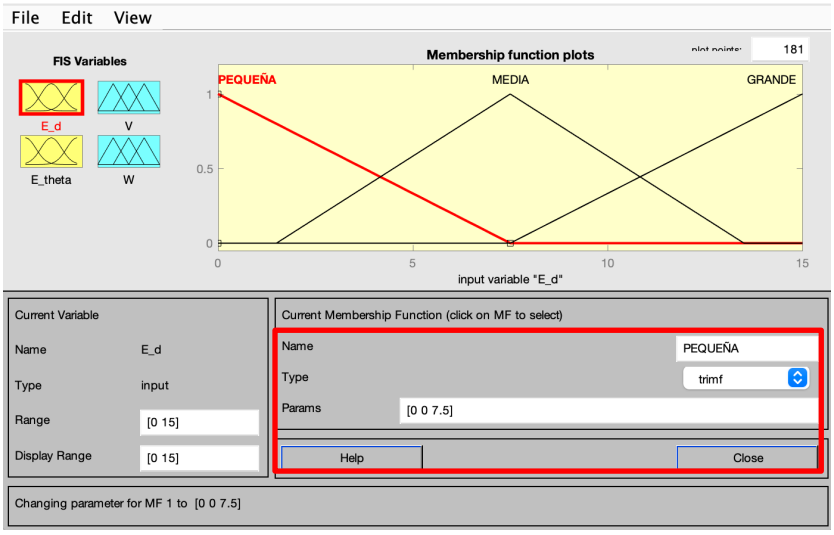

Figura 7. Definición de funciones de pertenencia de cada variable.

iv. Diseñe la tabla de reglas del controlador mediante la interfaz de diseño de reglas mostrada en la Figura 8.

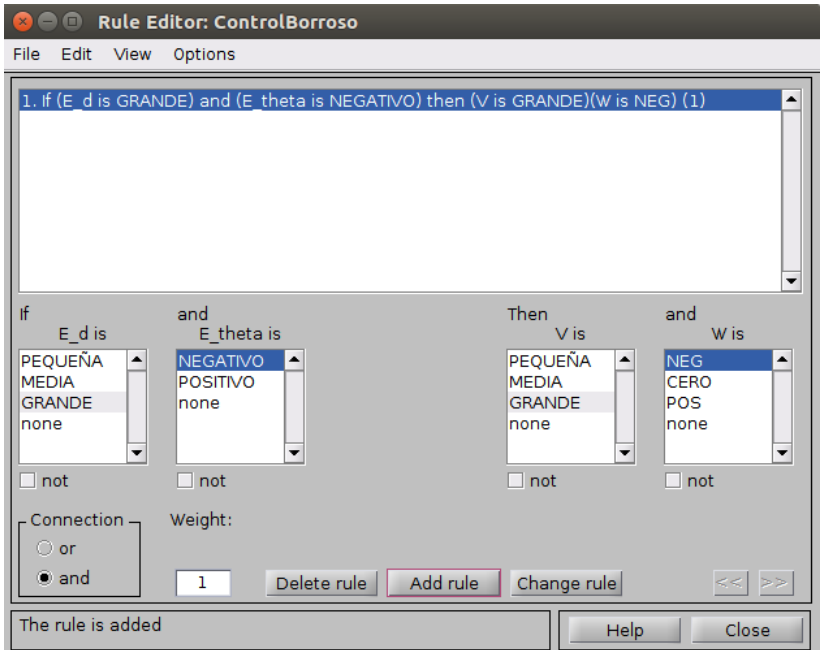

Figura 8. Tabla de reglas del controlador.

v. Exporte el controlador diseñado a un fichero .fis mediante el menú “File/Export to File”

b) Una vez diseñado el controlador, se creará el esquema mostrado en la Figura 9 en el entorno Simulink para generar el bloque controlador mediante la opción “Create Subsystem from Selection” (véase Figura 10).

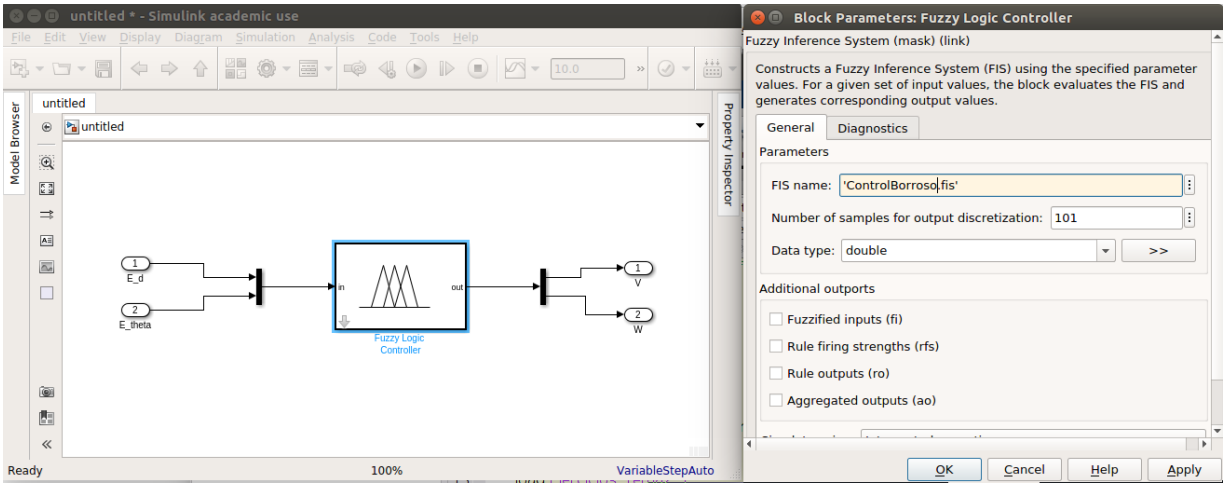

Figura 9. Creación del bloque controlador.

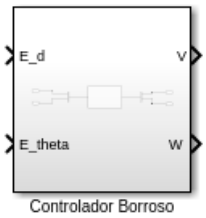

Figura 10. Bloque controlador borroso.

c) Introduzca el bloque controlador en el esquema de Simulink de la Figura 1 y guarde el modelo con el nombre “PositionControl.slx”. Realice pruebas con posiciones *refx* y *refy* aleatorias para comprobar que el controlador se comporta adecuadamente. Para ello genere un script de Matlab que permita automatizar ese proceso y mostrar mediante la función plot la trayectoria seguida por el robot. Ejecute dicho script varias veces o introduzca un bucle en el código proporcionado.

```
%Tiempo de muestreo
Ts=100e-3
% Referencia x-y de posicion
refx=10*rand-5;
refy=10*rand-5;
% Ejecutar Simulacion
sim('PositionControl.slx')
% Mostrar
x=salida_x.signals.values;
y=salida_y.signals.values;
figure;
plot(x,y);
grid on;
hold on;
```

d) Sustituya las referencias de posición (bloques refx y refy) por el generador de trayectorias proporcionado ("Trajectory_generator.slx") y compruebe el funcionamiento del sistema que se muestra en la Figura 11. Realice los cambios en el diseño del controlador borroso que sean necesarios para que el robot sea capaz de seguir la trayectoria.

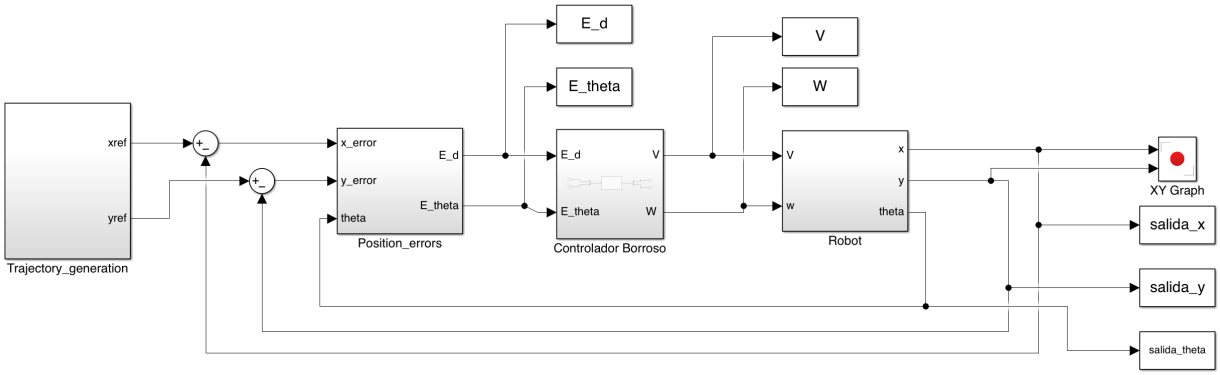

Figura 11. Seguimiento de una trayectoria.

La Figura 13 muestra la arquitectura del subsistema "Trajectory_generation", que generará la trayectoria que debe seguir el robot móvil. Este subsistema se proporciona dentro de la documentación de la práctica en el fichero "Trajectory_generator.slx". Para utilizar este bloque es necesario inicializar los parámetros iniciales de la trayectoria (x_0, y_0, th_0) y el periodo de muestro (Ts). Debe tener en cuenta que, si la trayectoria y el robot comienzan lo suficientemente cerca como para que se cumpla la condición de parada, se terminará la simulación.

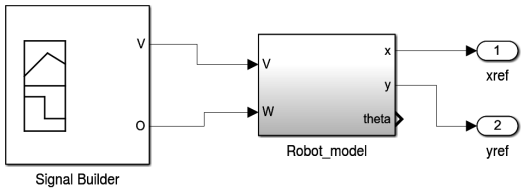

Figura 12. Seguimiento de una trayectoria.

El bloque "Signal Builder" se utiliza para generar las señales de velocidad lineal y angular. El bloque "Robot_model" es idéntico al utilizado para modelar el robot, añadiendo variables a los integradores para cambiar la posición (variables x0 e y0) y orientación (theta0) iniciales de la trayectoria. Este bloque y la trayectoria generada se muestran a continuación:

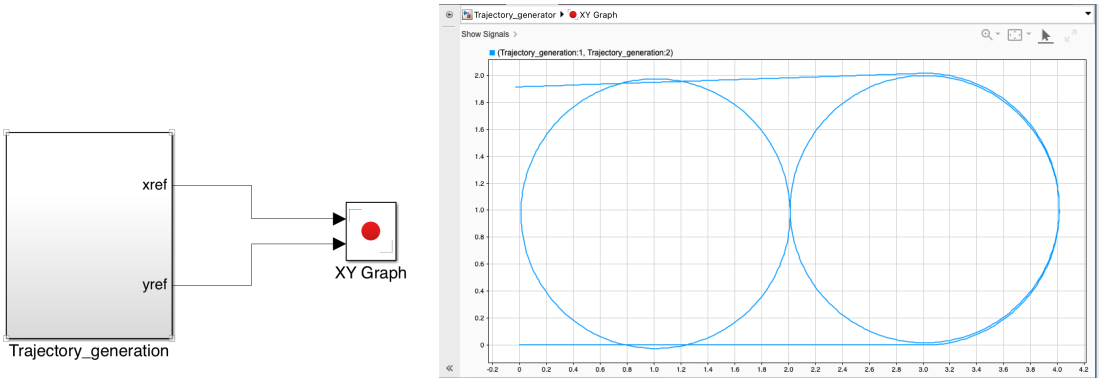

Figura 13. Trayectoria generada por el sistema "Trajectory_generation".