PRÁCTICA 2

SISTEMAS DE CONTROL INTELIGENTE

Control Borroso

Grado en Ingeniería Informática

David Barreiro Zambrano

Natalia Montoya Gómez

ÍNDICE

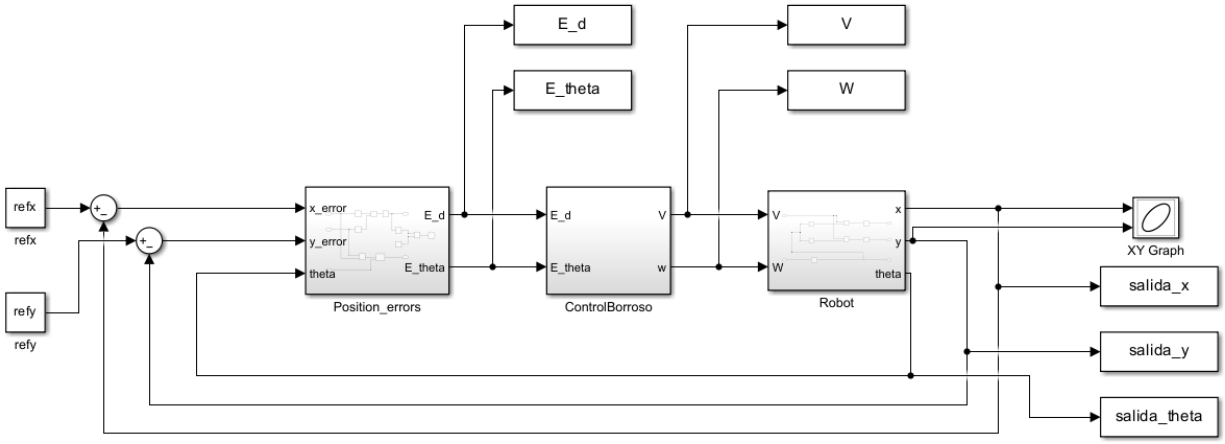
PARTE I – Diseño de un control borroso de posición para un robot móvil

1. Objetivo y descripción del sistema

Se utiliza el modelo de robot de la práctica anterior, puesto que esta es una continuación de la anterior.

El objetivo de esta práctica es diseñar un controlador borroso, como se ha visto en las clases de teoría, de tal forma que el robot, que se ha especificado que será el de la práctica anterior, sea capaz de llegar a una posición determinada por las referencias que se especifican de posiciones, las cuales son: *refx* y *refy*.

El modelo con el que se va a empezar a trabajar es el modelo de la práctica anterior, pero con el bloque de control como controlador borroso.

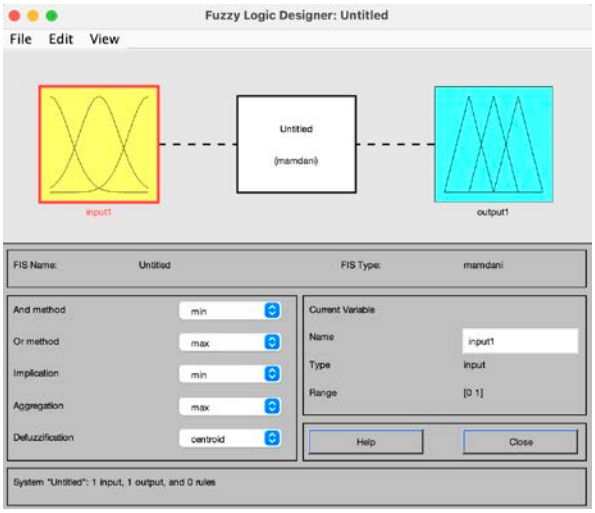


El controlador borroso que se creará tendrá dos entradas, los cuales serán los errores de distancia y ángulo. Tendrá también dos salidas, la velocidad lineal y la angular. Todo esto contará con sus propias funciones de pertenencia y reglas específicas.

2. Desarrollo de la práctica

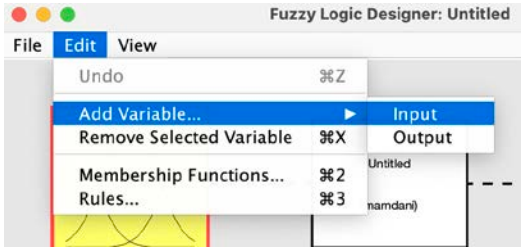
a. *Ejecute el comando "fuzzy" en la consola de Matlab para crear un controlador borroso mediante la toolbox de Matlab de control borroso.*
Mantener las opciones por defecto para las operaciones de AND/OR y el desborrosificador (Defuzzification) y siga los siguientes pasos:

Nada más ejecutar el comando que se indica en el enunciado, aparece la siguiente ventana:

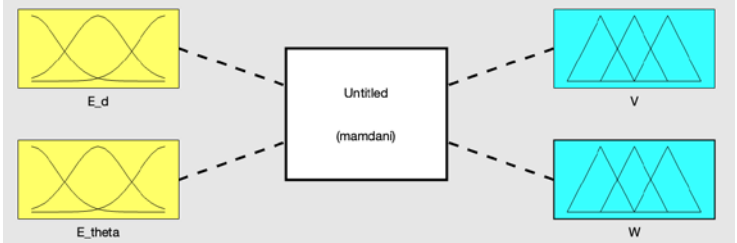


i. *Cree un sistema borroso con dos entradas y dos salidas mediante el menú "Edit/Add Variable/input" y "Edit/Add Variable/output". Nombre las entradas como "E_d" y "E_theta" y las salidas como "V" y "W".*
Inicialmente considere fijar la velocidad lineal V a una velocidad constante y trabaje sobre el control de la velocidad angular W. Una vez ajustado este, proceda a implementar el control de la velocidad lineal.

El menú desde el cual se añaden entradas y salidas, es el que aparece en la siguiente imagen:



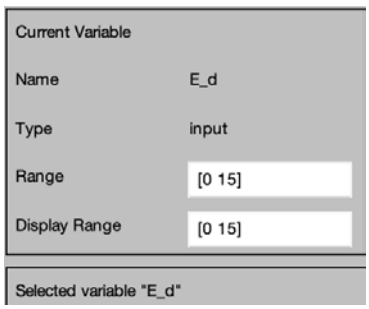
Las entradas y salidas creadas se cambian de nombre de la siguiente manera, donde la V hace referencia a la velocidad lineal y la W a la velocidad angular.

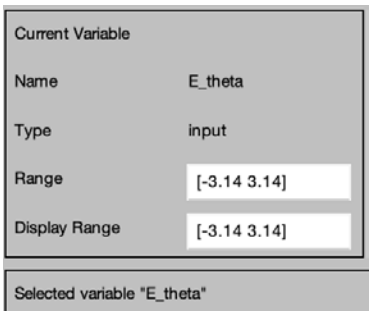


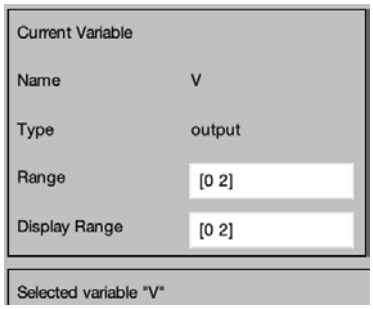
ii. Ajuste las variables de entrada para que dispongan de los siguientes rangos:

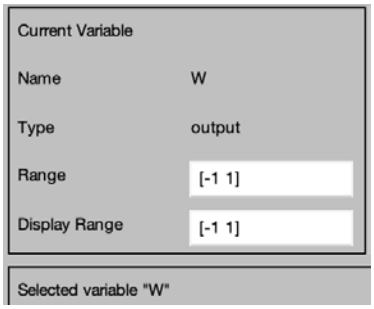
$$E_d \in [0,15],\ E_\theta \in [-\pi, \pi],\ V \in [0, 2],\ W \in [-1,1]$$

Se ubica cada una de las entradas y salidas, por lo que se escribe el rango deseado en el enunciado de la siguiente manera:





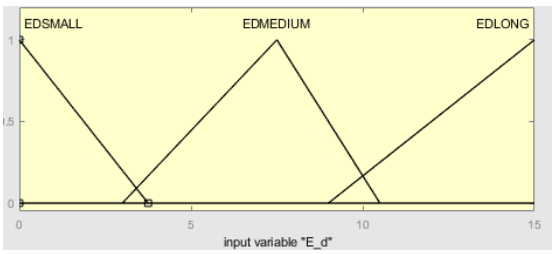


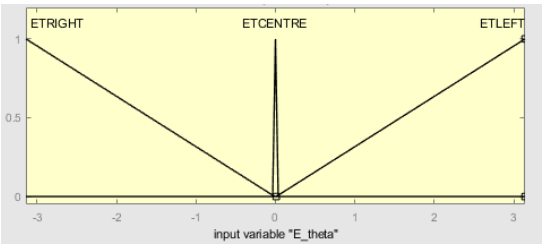


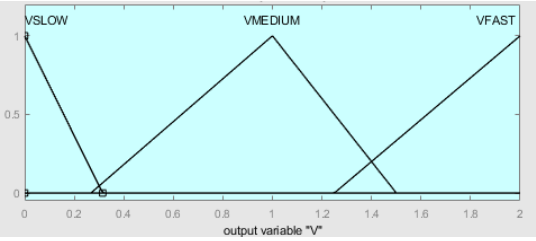
iii. Cree las funciones de pertenencia del tipo que considere necesarias en cada una de las variables definidas en la entrada y la salida. Utilice normas para los conjuntos borrosos que tengan un significado claro como NEGATIVO, POSITIVO, GRANDE, PEQUEÑO, etc.

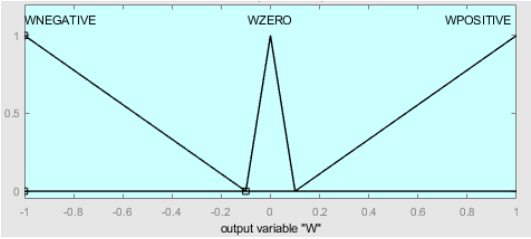
- Para la entrada *E_d* utilizamos los conjuntos: EDSMALL, EDMEDIUM, EDLONG.
- Para la entrada *E_theta* utilizamos: ETLEFT, ETCENTER, ETRIGHT.
- Para la salida *V* utilizamos: VSLOW, VMEDIUM, VFAST.
- Para la salida *W* utilizamos: WNEGATIVE, WZERO, WPOSITIVE.

A continuación, se muestran las funciones de pertenencia de las entradas y salidas, las cuales se han ido creando mediante la interfaz que se puede observar en las siguientes imágenes, ajustando los parámetros:









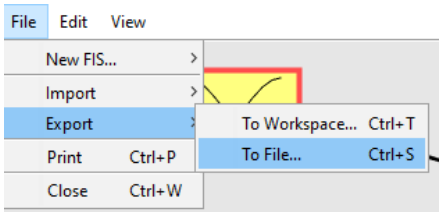
iv. Diseñe la tabla de reglas del controlador mediante la interfaz de diseño de reglas.

Para poder añadir las reglas, nos ubicamos en la misma pestaña anterior, clicamos en *Edit* y en el desplegable, pulsamos la opción *Rules*. Se añaden las reglas que se consideren necesarias:

1. If (E_d is EDSMALL) and (E_theta is ETLEFT) then (V is VSLOW)(W is WPOSITIVE) (1)
2. If (E_d is EDSMALL) and (E_theta is ETCENTRE) then (V is VSLOW)(W is WZERO) (1)
3. If (E_d is EDSMALL) and (E_theta is ETRIGHT) then (V is VSLOW)(W is WNEGATIVE) (1)
4. If (E_d is EDMEDIUM) and (E_theta is ETRIGHT) then (V is VMEDIUM)(W is WNEGATIVE) (1)
5. If (E_d is EDMEDIUM) and (E_theta is ETCENTRE) then (V is VMEDIUM)(W is WZERO) (1)
6. If (E_d is EDMEDIUM) and (E_theta is ETLEFT) then (V is VSLOW)(W is WPOSITIVE) (1)
7. If (E_d is EDLONG) and (E_theta is ETLEFT) then (V is VFAST)(W is WPOSITIVE) (1)
8. If (E_d is EDLONG) and (E_theta is ETCENTRE) then (V is VFAST)(W is WZERO) (1)
9. If (E_d is EDLONG) and (E_theta is ETRIGHT) then (V is VFAST)(W is WNEGATIVE) (1)

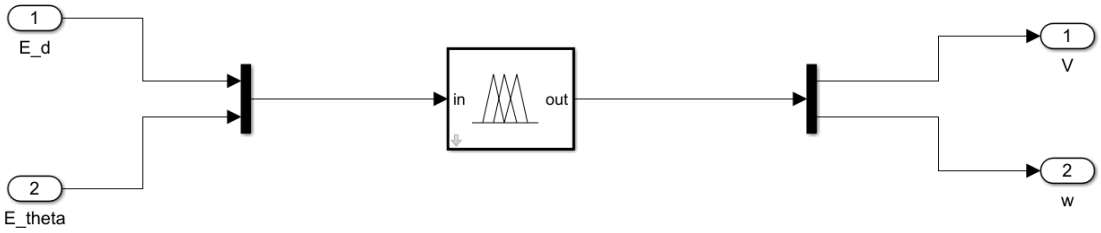
v. Exporte el controlador diseñado a un fichero *.fis* mediante el menú “File/Export to File”

Exportamos el fichero con el nombre de ‘controlador.fis’.

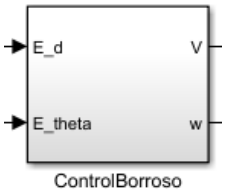


b. *Una vez diseñado el controlador, se creará el esquema en el entorno Simulink para generar el bloque controlador mediante la opción "Create Subsystem from Selection".*

Como se muestra en el enunciado, creamos el controlador borroso con la siguiente estructura:



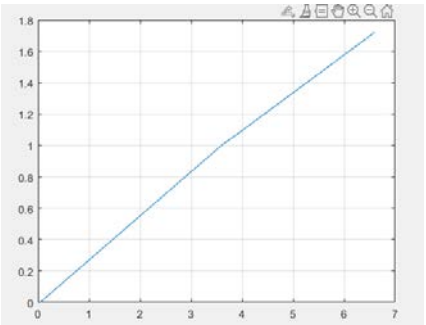
En el modelo completo, tiene el siguiente aspecto:



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.*

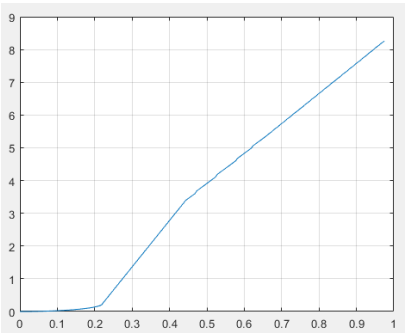
Como pide el enunciado, se hacen pruebas con posiciones aleatorias para comprobar que el sistema se comporta adecuadamente.

PRIMERA PRUEBA



refx = 6.5548
refy = 8.2346

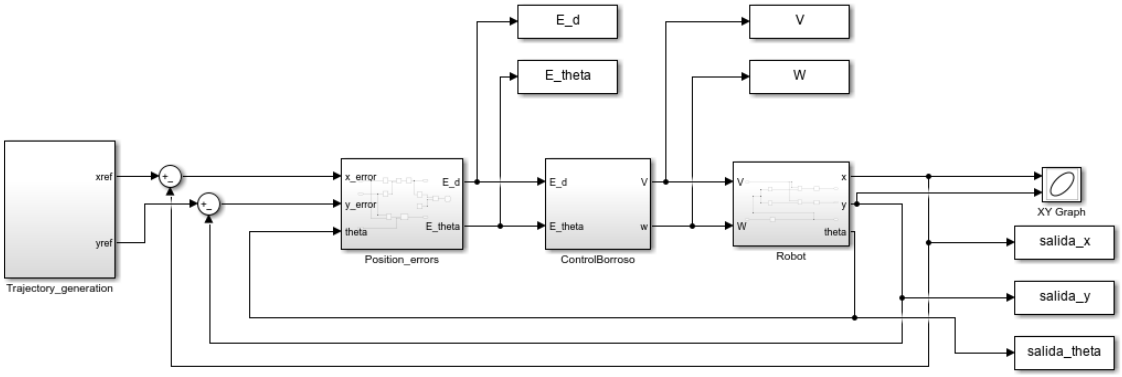
SEGUNDA PRUEBA



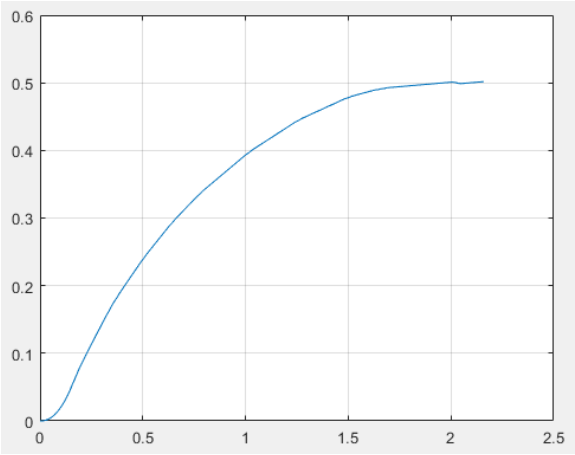
refx = 0.9713
refy = 1.7118

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. Realice los cambios en el diseño del controlador borroso que sean necesarios para que el robot sea capaz de seguir la trayectoria.*

Como bien especifica el enunciado, hemos cambiado las entradas *refx* y *refy* por el generador que se proporciona "*Trajectory_generator.slx*", de la siguiente manera:



El resultado final es el siguiente:



PARTE II – Diseño de un control borroso de posición con evitación de obstáculos

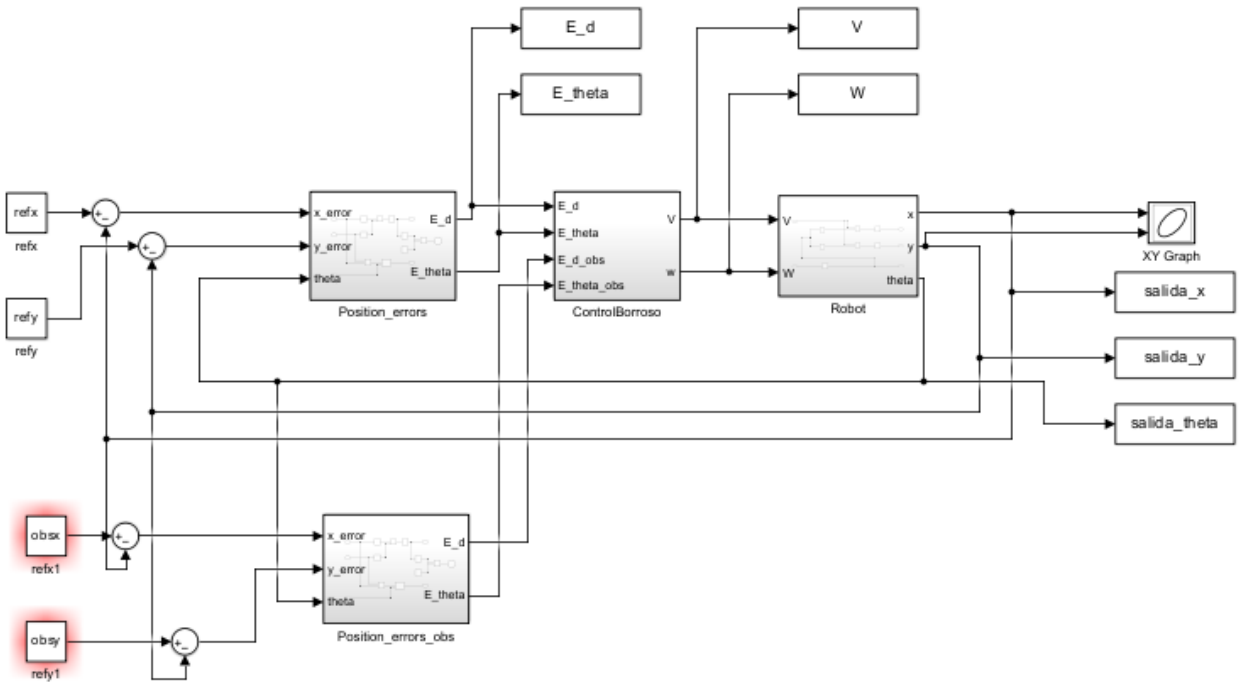
1. Objetivo y descripción del sistema

En este caso, se utiliza el modelo del anterior apartado de esta misma práctica, pues la segunda parte es una ampliación de la primera.

El objetivo de esta práctica es extender el apartado anterior para que esta vez el robot sea capaz de esquivar un obstáculo situado en una determinada posición y con unas dimensiones también conocidas.

El obstáculo se encontrará en *obsx* y *obsy*.

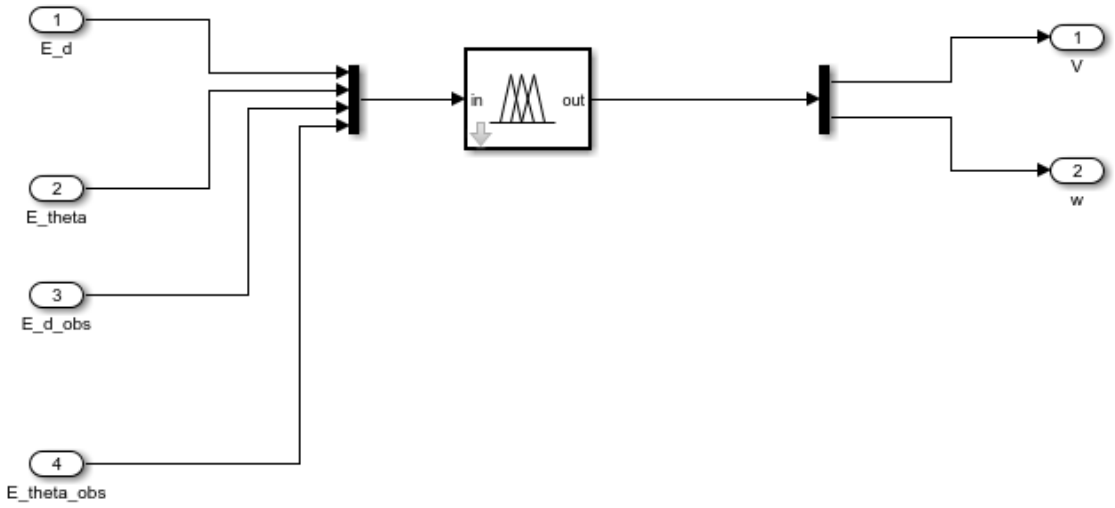
El modelo se ha modificado para que ahora se puedan tener en cuenta los obstáculos, y tengan un sitio en el sistema.



2. Desarrollo de la práctica

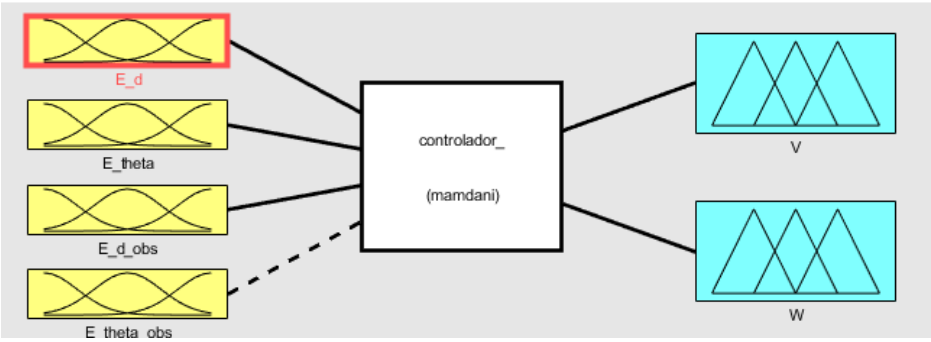
El controlador realizado tendrá como entradas los errores de distancia y ángulo a la referencia de posición (E_d, E_Θ) y los errores de distancia y ángulo a la posición del obstáculo ($E_{d,obs}$, $E_{\Theta,obs}$). El controlador genera a su salida los valores de velocidad (angular y lineal) que serán entradas en el bloque de simulación del robot. El controlador borroso diseñado se implementará en un bloque de Simulink.

El bloque de control, el cual es un controlador borroso está ligeramente cambiado para poder contemplar las nuevas entradas:



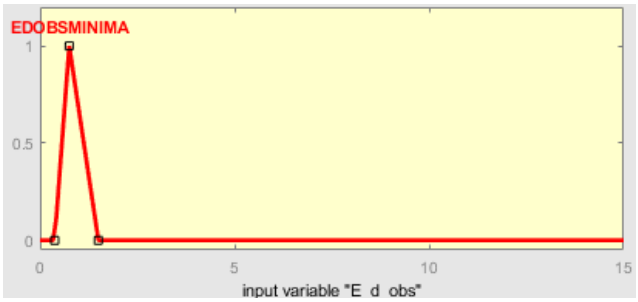
a. *Defina un controlador de 4 entradas y 2 salidas como el mostrado a continuación. Se recomienda partir del controlador diseñado en la primera parte de la práctica.*

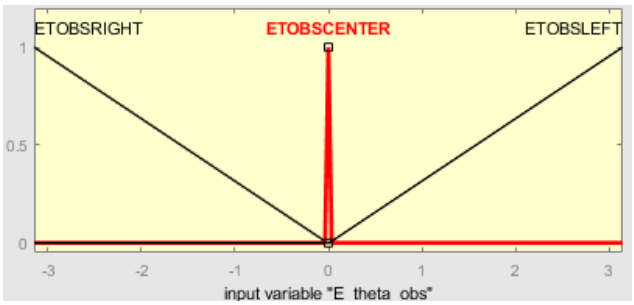
A partir del controlador borroso de la parte anterior de la práctica, se añaden las otras dos entradas.



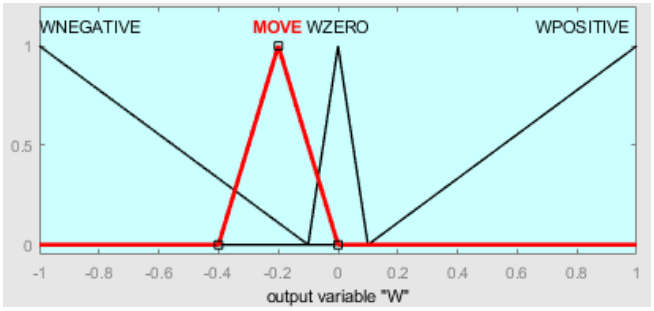
b. *Diseñe las funciones de pertenencia de las nuevas entradas "E_d_obs" y "E_theta_obs" y modifique las que crea convenientes del resto de entradas y salidas del controlador.*

Se muestran a continuación las funciones de pertenencia de las nuevas entradas:





Hay una ligera modificación en las funciones de pertenencia de la salida de la velocidad angular, en la que se añade la nueva función de *MOVE* en caso de que detecte algún obstáculo.

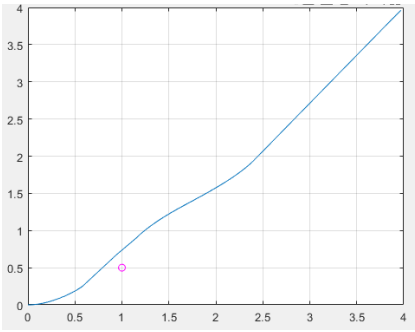


c. *Cree la tabla de reglas del nuevo controlador. Se puede utilizar la opción "none" para ignorar algunas de las entradas del controlador en una regla.*

Se ha utilizado la tabla de reglas de la práctica anterior, agregando una regla más para esquivar el obstáculo:

10. If (E_d_obs is EDOBSMINIMA) then (V is VSLOW)(W is MOVE) (1)

d. *Evalúe el controlador diseñado utilizando el diagrama de bloques de la Figura 2. Llame a ese diagrama 'EvitarObstaculo.slx' y proponga valores de las variables refx, refy obsx y obsy en los que el robot esté obligado a esquivar el obstáculo y muestre las trayectorias seguidas por el robot.*



Como se puede observar, hace el trabajo satisfactoriamente, ya que tiene un recorrido, y a su vez esquiva un obstáculo del camino.

El obstáculo es el círculo rojo y el recorrido es la línea azul.

e. *Sustituya los bloques refx y refy por el sistema generador de trayectorias utilizado en la primera parte de la práctica. Proponga valores de obsx y obsy en los que el robot esté obligado a esquivar el obstáculo y muestre las trayectorias seguidas por el robot en este caso.*

Sustituimos al igual que en la primera parte, las entradas *refx* y *refy* por el generador de trayectorias, por lo que se quedará de la siguiente manera: