

**PRÁCTICA FINAL SISTEMAS DE CONTROL INTELIGENTE**

Diseño de Controladores Borrosos y Neuroborrosos para un Robot Móvil



Sergio Sánchez Campo

David Ramos Fernández

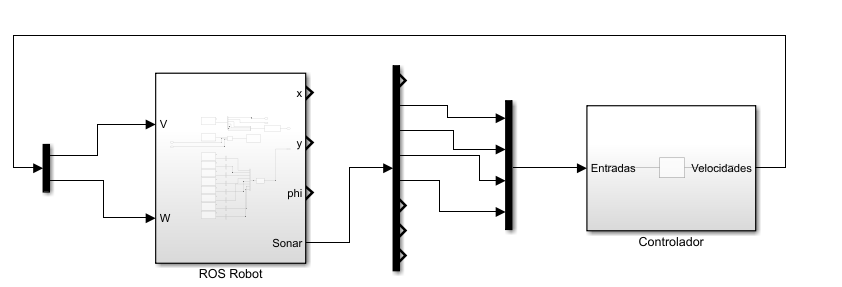
GISI

# **Parte 1. Diseño manual de un control borroso de tipo MAMDANI**

Una vez ya hemos instalado y configurado todos los aspectos referentes a ROS, comenzamos con la primera parte de la práctica, referente al control borroso.

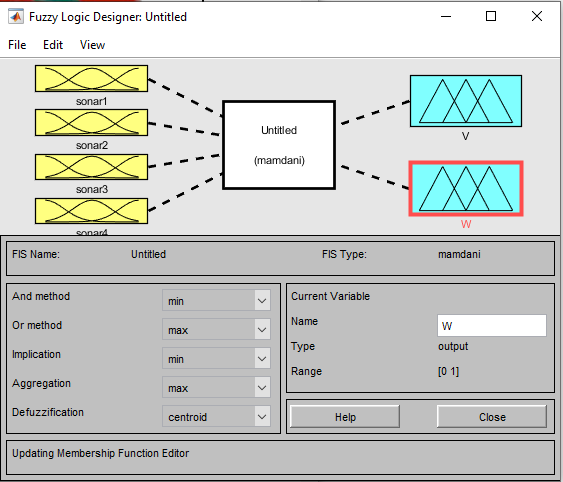
Partimos del archivo de simulink proporcionado con la práctica que contiene el controlador del robot ROS.

Para comenzar decidimos qué sonares emplearemos para el reconocimiento de los obstáculos y las paredes del circuito. Decidimos emplear los sónares 1,2,3 y 4. El 2 y 3 para esquivar los obstáculos del circuito y el 1 y el 4 para mantener la distancia con los bordes del circuito.

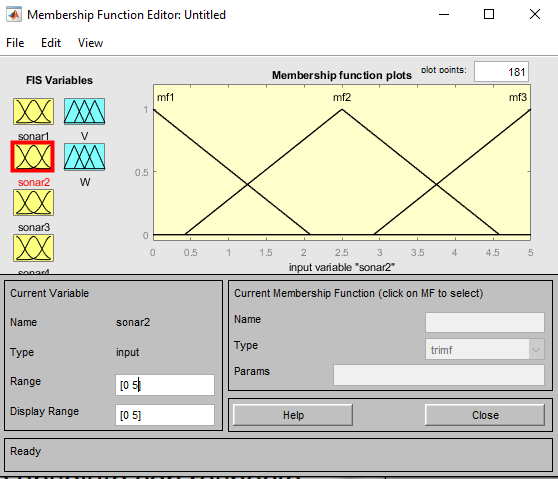


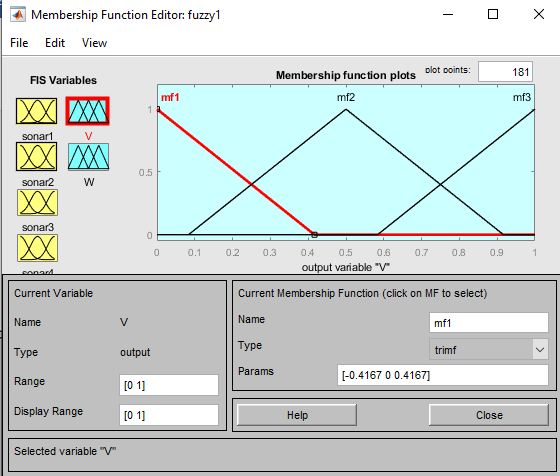
A continuación, pasamos a diseñar el controlador borroso correspondiente:

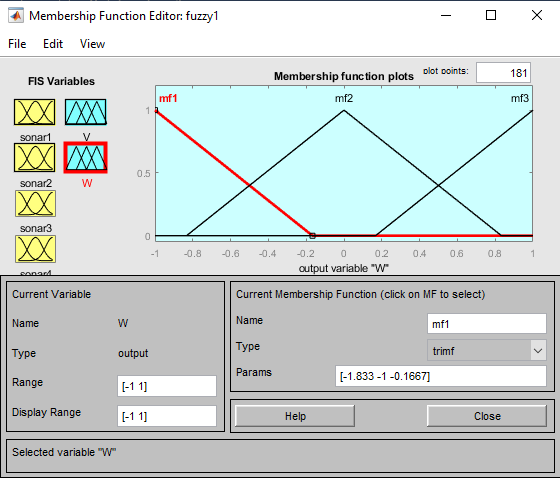
Creamos 4 inputs correspondientes a los 4 sonares y 2 outputs correspondientes a la velocidad(V) y a la velocidad angular(W).



Continuamos introduciéndoles los rangos a las variables, de entre 0 y 5 metros los sónares, que es su rango máximo al que son capaces de detectar. De entre 0 y 1 la velocidad, ya que es la velocidad máxima que alcanza el robot. Y de entre -1 y 1 radianes/segundo que es el giro máximo que es capaz de realizar el robot.

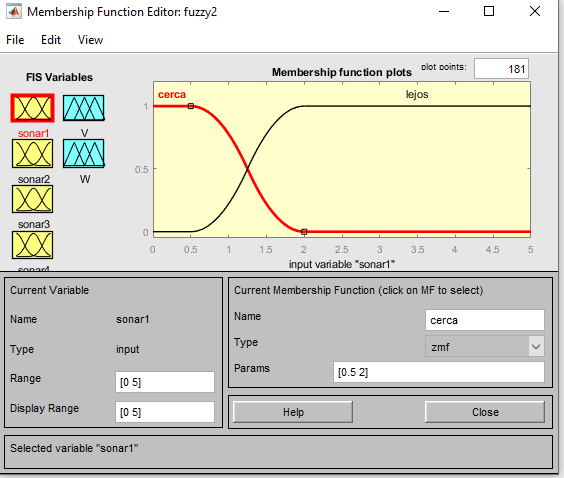


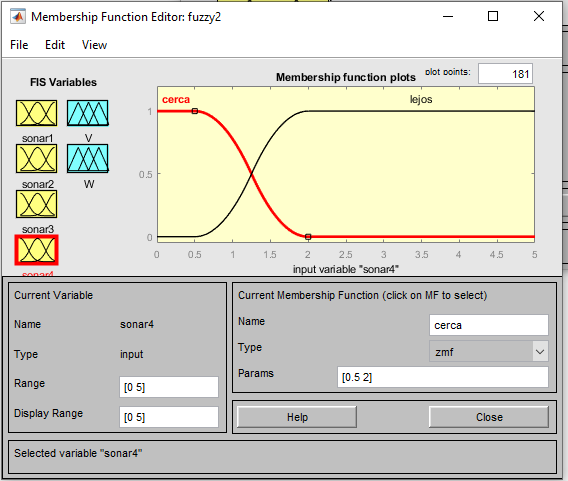




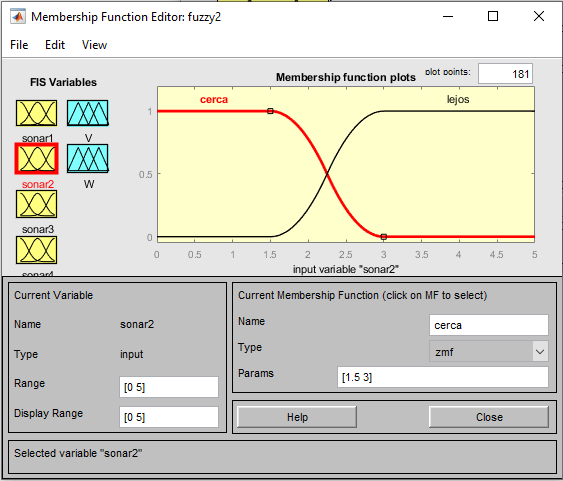
Seguimos con las variables, configurando ahora las funciones de pertenencia de los conjuntos borrosos.

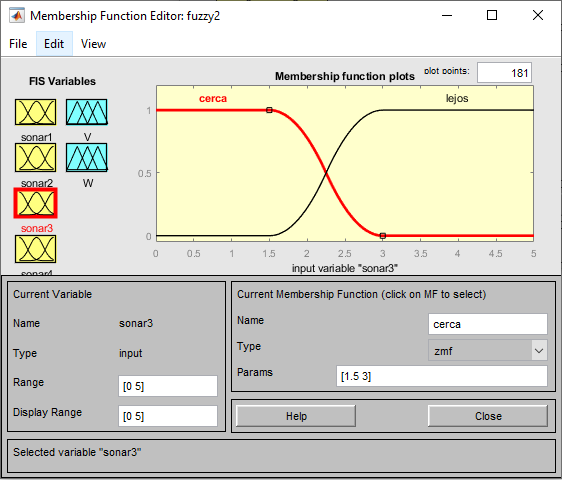
Para los sónares 1 y 4, empleamos dos rectas una para cerca y otra para lejos que se cortan en el 1,25.



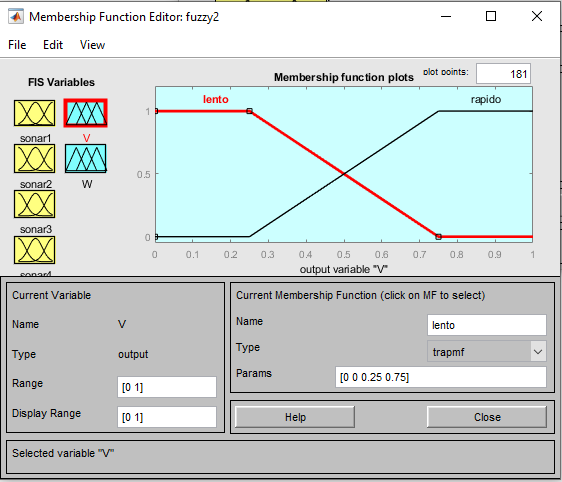


Y para los sónares 2 y 3 también empleamos dos rectas, pero en este caso el punto de corte lo ponemos a una mayor distancia, a 2,25. Ya que son los sensores que se encuentran en la parte frontal y queremos que detecten objetos a una mayor distancia.

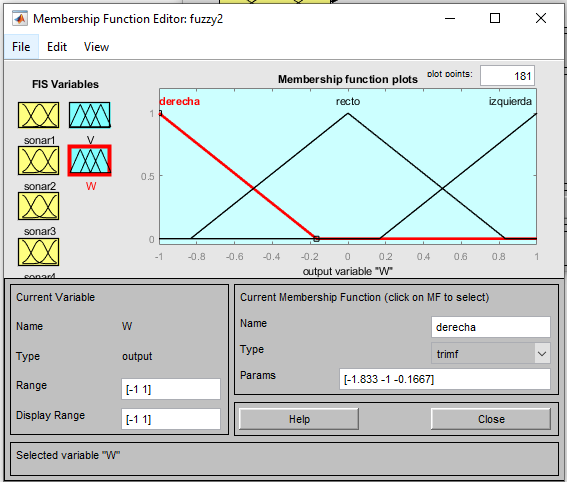




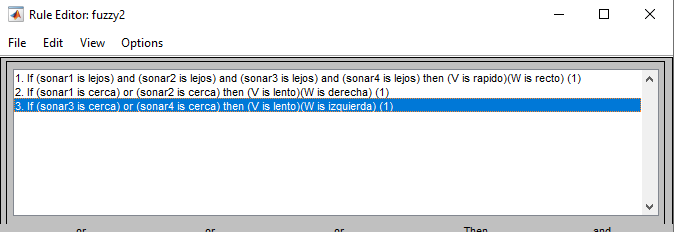
Respecto a la velocidad establecemos una recta para velocidad lenta y otra para rápido que se cortan justo en el centro.



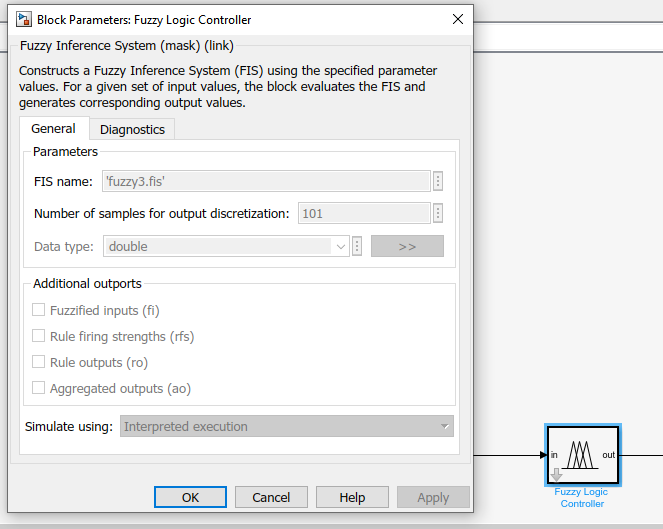
Y para la velocidad angular, empleamos 3 rectas, una para girar a la derecha, otra para girar a la izquierda y otra para ir recto. Para el lado izquierdo y derecho seleccionamos funciones con pendiente constante y que tuviesen su pico en la máxima velocidad angular del robot. Y en el caso de la función recto seleccionamos una función triangular con su pico máximo en 0, para que no gire el robot.



Después pasamos a establecer las reglas, comenzamos estableciendo 3 reglas básicas: la primera para que siga recto si ningún sónar detecta un objeto cercano. La segunda que hace que el robot gire a la derecha cuando el sonar 1 o 2 (que se encuentra en el lado izquierdo del robot) detecten algún obstáculo cerca. Y la tercera regla que hace que el robot gire a la izquierda cuando el sonar 3 o 4 (que se encuentra en el lado derecho del robot) si detectan algún obstáculo cerca.

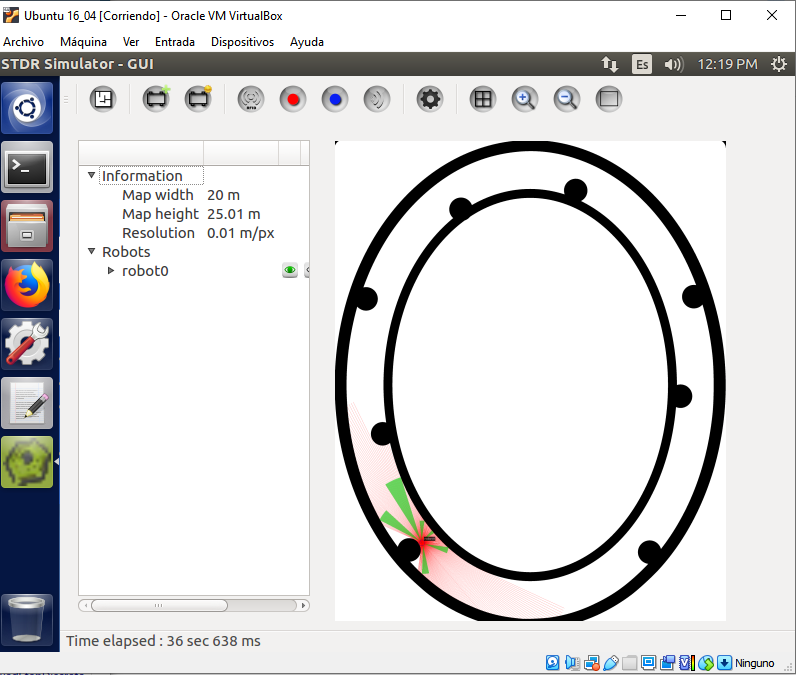


Finalmente incorporamos este fuzzy al controlador del robot.



Probamos en el circuito sin obstáculos y lo recorrió perfectamente, girando adecuadamente y sin chocarse en ningún momento y a una buena velocidad.

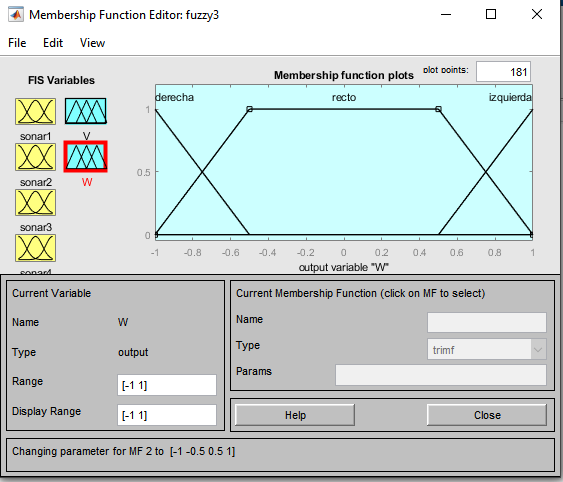
A continuación, probamos en el circuito con obstáculos y al detectar el primer obstáculo giro en la dirección adecuada pero no lo suficiente y no le dio tiempo a esquivar el obstáculo y se chocó, el robot giraba adecuadamente pero no lo suficiente para darle tiempo a esquivar el obstáculo.



Para resolver esto podíamos o variar el rango de detección o incrementar la velocidad de giro.

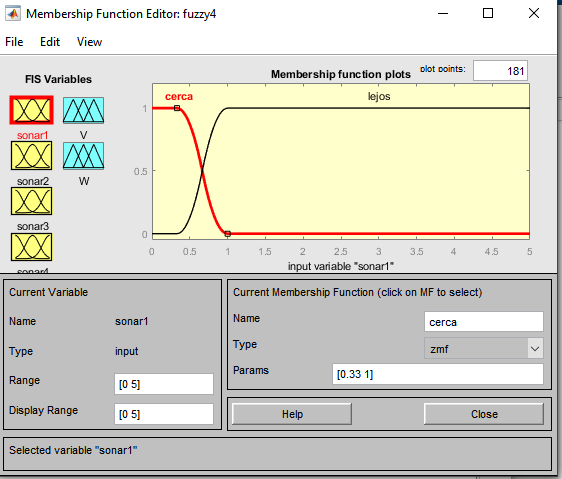
Probamos incrementando la velocidad de giro, modificando las funciones de pertenencia del output W.

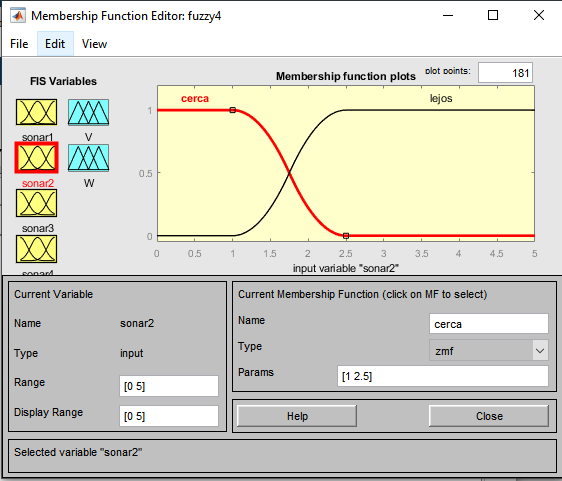
Moviendo las rectas de derecha e izquierda hacia los bordes, aumentando su velocidad angular para lograr mayores giros.



Realizamos otra vez la prueba y observamos que gira bien pero no lo esquiva ya que entra en conflicto con la norma de girar al otro lado ya que detecta el otro lado del circuito también cerca.

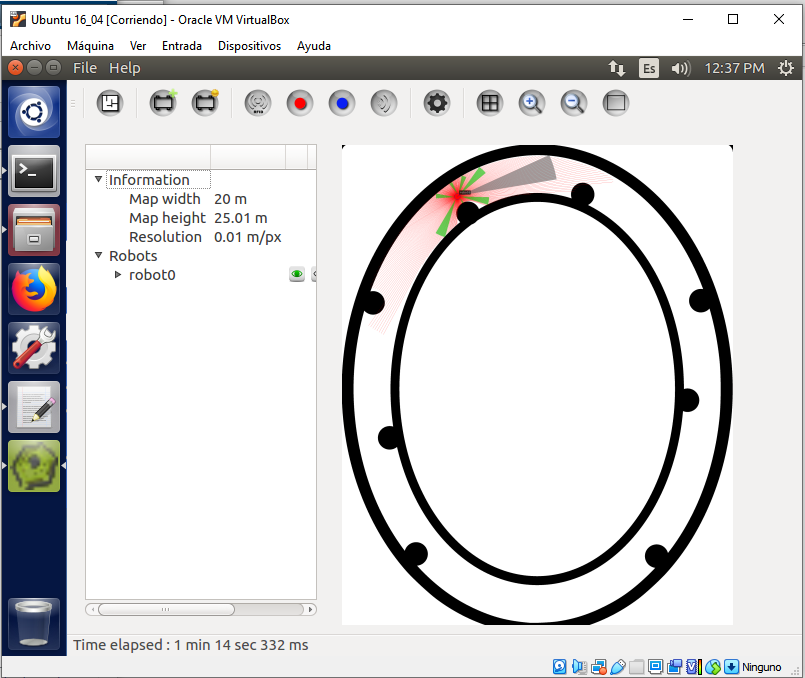
Para resolverlo reducimos la distancia a la que el sensor comienza a detectar que un obstáculo está cerca. Desplazando ligeramente hacia el 0 las rectas de los sonares.



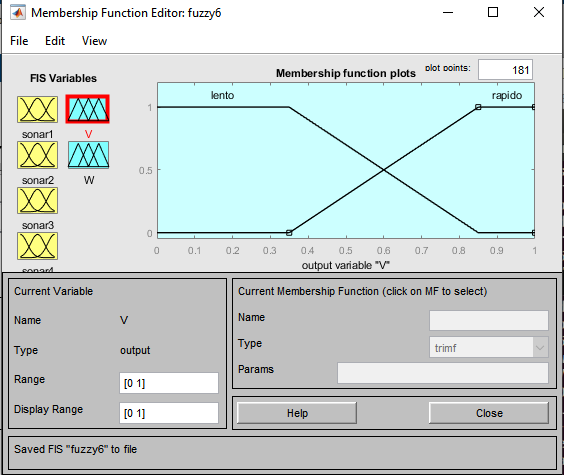


Volvemos a probar el funcionamiento del robot con estas configuraciones.

En este caso, el robot ya es capaz de recorrer de manera adecuada el circuito de obstáculos y de obstáculos2, sin chocarse contra ningún muro ni ningún obstáculo y a una velocidad decente, aunque a la hora de esquivar los obstáculos reduce bastante su velocidad. Así que para finalizar vamos a tratar de incrementar la velocidad a la que el robot es capaz de recorrer los circuitos sin chocarse en ningún momento.

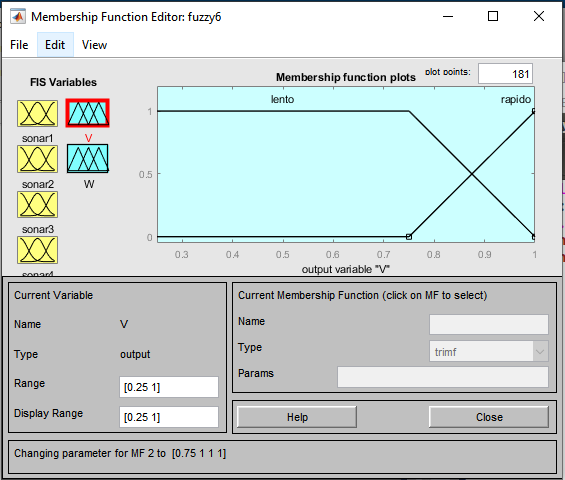


Para este objetivo vamos a variar la función de la velocidad:



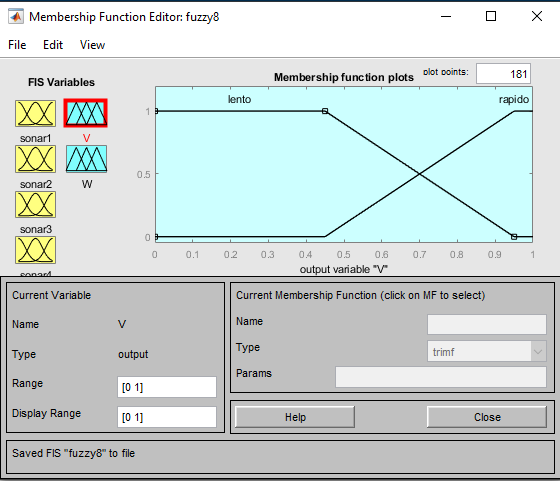
Desplazamos en 0.1 las funciones hacia la derecha.

Al volver a probarlo, y el robot sigue recorriendo bien el circuito y aumenta ligeramente su velocidad. Vamos a tratar de aumentar más la velocidad, en este caso variamos el rango, estableciendo el mínimo en 0.25 y desplazamos la función de lento y rápido bastante hacia la derecha.



Ahora va a demasiada velocidad y el robot se choca con el obstáculo ya que no le da tiempo a girar lo suficiente para esquivarlo.

Así que partimos del fuzzy anterior y le aumentamos la velocidad, pero un menor rango, le volvemos a aumentar en 0.1 la velocidad, desplazando las funciones a la derecha:



Volvemos a ejecutarlo y recorre adecuadamente el circuito, a mayor velocidad y sin chocarse. En este punto, el robot ya recorre el circuito a una velocidad bastante elevada teniendo en cuenta su velocidad máxima. Y por lo tanto nos quedamos con esta versión.

# **Parte 2. Diseño automático de un controlador neuroborroso de tipo SUGENO**

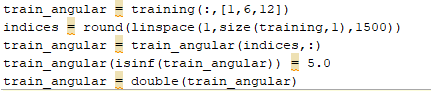
Para la parte de control mediante controlador Sugeno, comenzamos estableciendo nuestras IPs para ROS en el archivo ControlManualRobot:



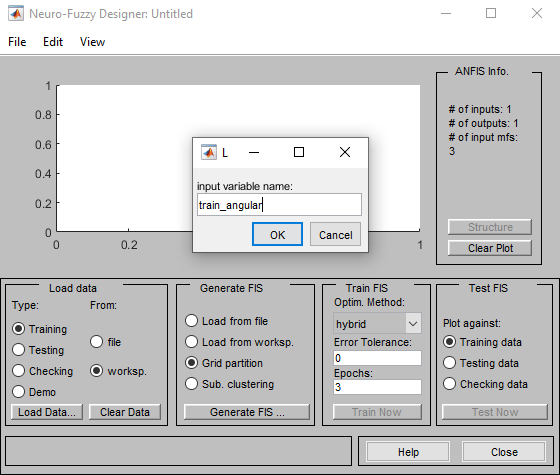
A continuación, ejecutamos el programa y recorremos el circuito empleando el controlador manual. Tras varios intentos, logramos realizar adecuadamente el circuito y obtener unos buenos datos. Obtenemos la matriz entrenamiento que contiene dichos datos, se nos pasa un poco del tamaño máximo de 1.500 filas que queremos utilizar para el entrenamiento del neuro controlador, así que eliminamos algunos datos del principio y del final que son los que contenían datos menos relevantes, ya que pasaba un tiempo desde que empezaba a grabar datos y nosotros empezábamos a controlar el robot y tras acabar de conducirlo.



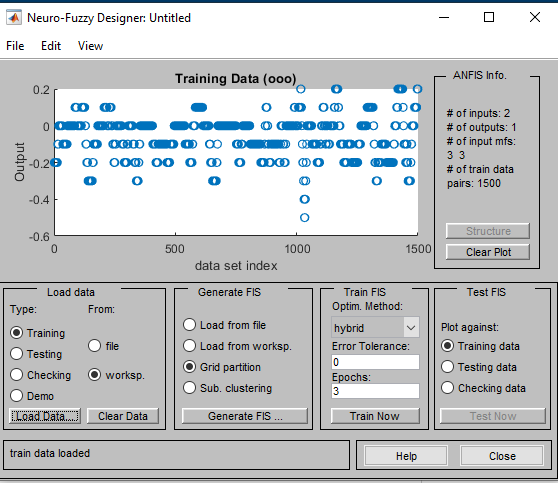
Vamos a comenzar realizando el entrenamiento del controlador de la velocidad angular. Vamos a emplear como inputs los sonares 0 y 5 para la detección de obstáculos adyacentes al robot.



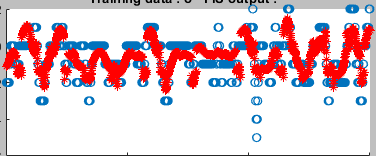
Y escribimos “Anfisedit” en Matlab para comenzar con el entrenamiento.



Comenzamos cargando los datos desde el workspace de Matlab.

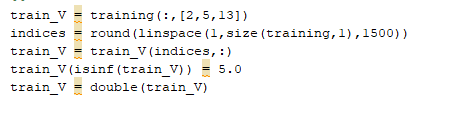


A continuación, generamos el FIS mediante *Grid partition*. Y realizamos el entrenamiento de tipo híbrido y con 3 épocas. Finalmente, le damos a *Test Now* para comprobar como ha sido el resultado del entrenamiento.

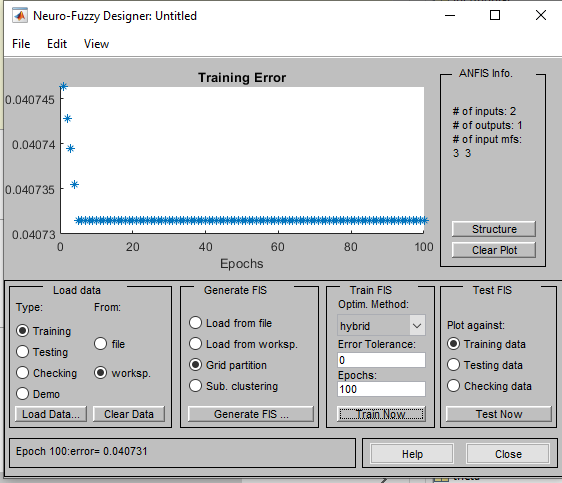


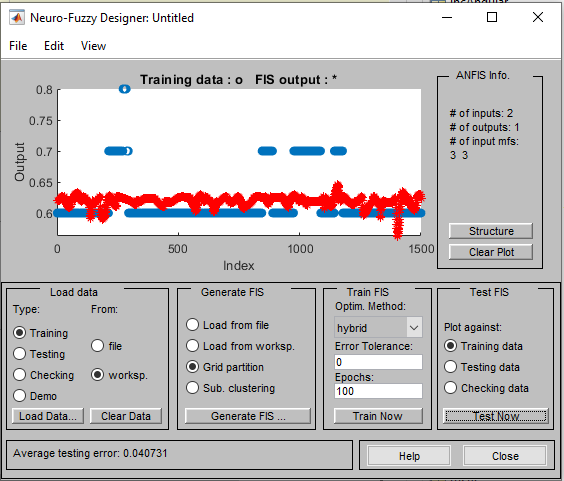
Observamos que no es especialmente preciso, pero si que consigue una cierta aproximación.

En segundo lugar, realizamos el entrenamiento de la velocidad. Para este entrenamiento vamos a emplear los sonares 3 y 4, que son los frontales del robot y los que creemos que son más importantes a la hora de regular la velocidad del robot.



Seguimos los mismos pasos que en el caso anterior, aunque aumentamos el número de épocas para el entrenamiento FIS. En este caso, observamos que no hubiese sido necesarias tantas épocas, sobre todo porque existe poca variación de los datos.

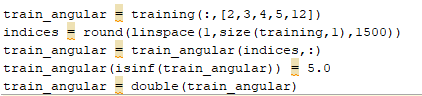




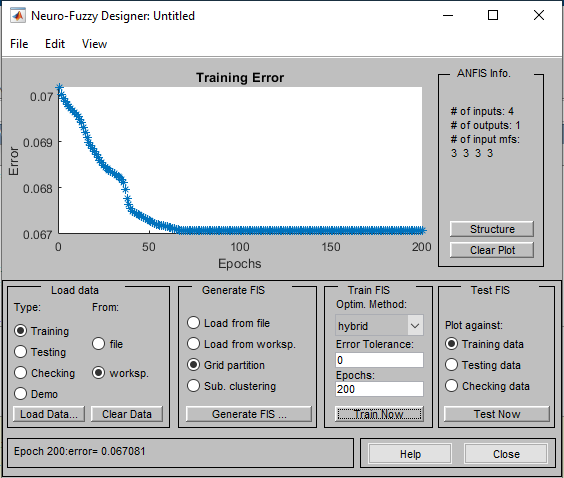
El resultado de los datos es moderadamente preciso, aunque en este caso tampoco había mucha menor variedad de datos diferentes ya que el piloto mantuvo una velocidad más o menos constante sin demasiadas variaciones y sin adaptarse demasiado al circuito, ya que no fuimos del todo capaces de ir moderando la velocidad mientras hacíamos que el robot no se chocase.

Al probarlo, en el circuito sin obstáculos lo recorre perfectamente sin chocarse. En cambio, al probarlo en el circuito con obstáculos no gira lo suficiente y se choca con los obstáculos.

Para solucionar esto vamos a modificar el número de sonares que tiene en cuenta para el control de la velocidad angular:

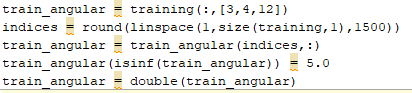


Y aumentamos el número de épocas del entrenamiento, pasando de 3 a 200. Asegurándonos de minimizar el error lo máximo posible.

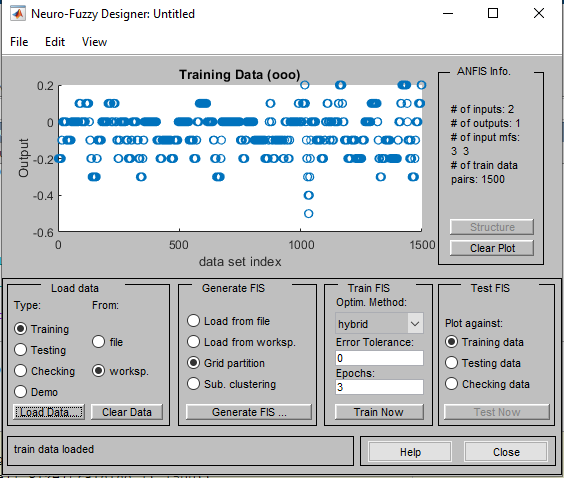


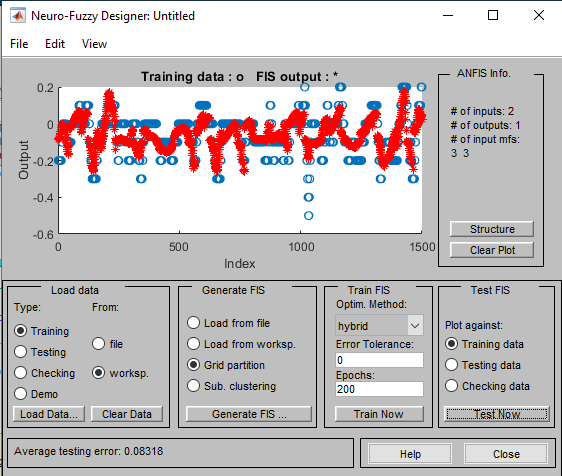
Al volver a probarlo en el circuito con obstáculos, tampoco esquiva adecuadamente los obstáculos.

Probamos volviendo a modificar los sonares, empleando los sensores 2 y 3 para el entrenamiento angular, que son los que empleamos para evitar obstáculos en la primera parte de la práctica.

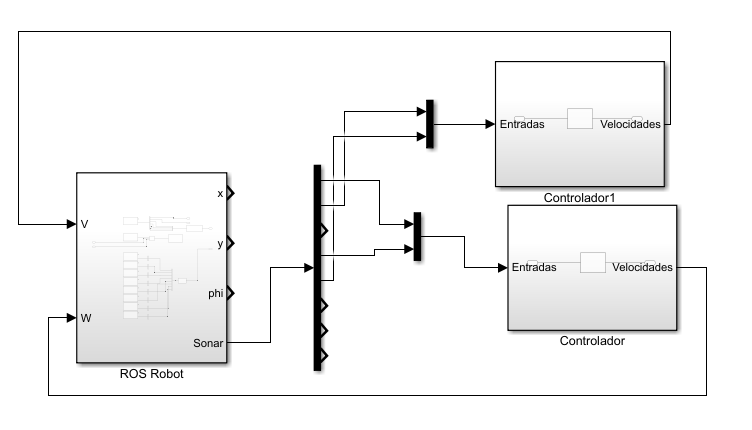


Y volvemos a realizar el entrenamiento.





Probamos con esta configuración y en este caso el robot recorre adecuadamente el circuito y es capaz de evitar los obstáculos, tanto del circuito de obstáculos 1 pero no del circuito con obstáculos 2. Finalmente estuvimos probando con diferentes combinaciones de sonares y empleando el sonar 0 (lateral izquierdo) y el sonar 3(frontal derecho), el robot era capaz de recorrer adecuadamente todos los circuitos recorriendo adecuadamente el circuito y esquivando todos los obstáculos. Siendo capaza de recorrer los 3 circuitos diferentes con obstáculos sin problemas.



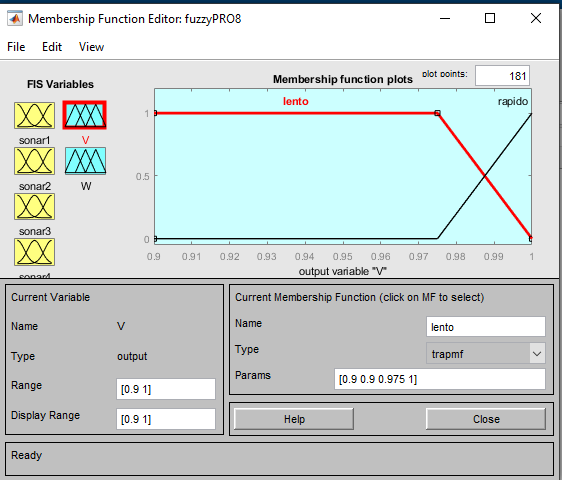
El mayor problema en este caso es que la velocidad es bastante reducida durante todo el recorrido ya que está basado en nuestros datos de conducción y no hemos sido capaces de recorrer mejor el circuito.

# **Parte 3. Competición**

Para la competición hemos escogido el controlador borroso ya que es con el que mejor resultados hemos obtenidos. Sin embargo, el controlador Sugeno tiene mejor trazada, el problema es que no alcanza altas velocidades ya que no somos capaces de realizar el circuito a mayor velocidad.

Con el objetivo de reducir el tiempo que tarda el robot en recorrer el circuito, hemos modificado las funciones de velocidad en el fuzzy y de los sonares.

Respecto a la velocidad, tras realizar varias pruebas hemos observado que el rango óptimo de velocidad es entre 0.9 y 1, que es la velocidad máxima con la que el robot es capaz de realizar los giros adecuadamente.



También hemos modificado ligeramente los sonares, reduciendo el rango de detección de obstáculos para que el robot pierda menos tiempo al esquivar los obstáculos pero que no se choque en ningún momento. Además, el rango de detección del lado derecho es ligeramente inferior al del lado izquierdo para que el robot se acerque más al lado interno del circuito.

