Nombre de la practica waxin

# 1 – Introduccion

# 2- Parte 1 – Controlador Borroso

## 2.1 – Sensores disponibles

# Sensores disponibles

Imagen que contiene Icono

Descripción generada automáticamente

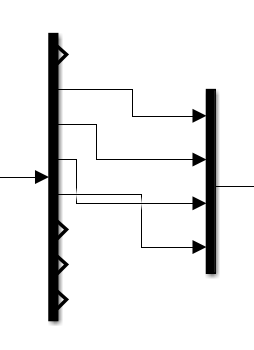
El 0 es el de abajo y siguen en orden en sentido horario

Podemos usar el 4 y el 5 para que se pegue bien a la pared pa correr velosidad altísima

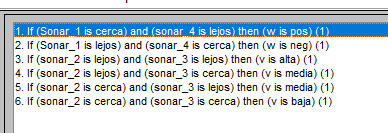
Aunque deberíamos usar mínimo el 0 para que no se pegue con esa pared

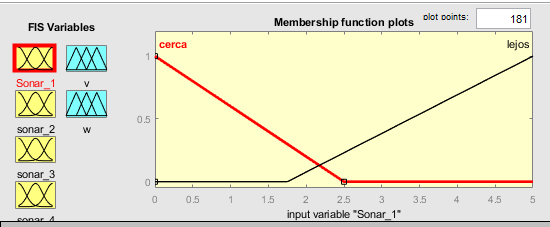
Voa probar con 4 5 y 0

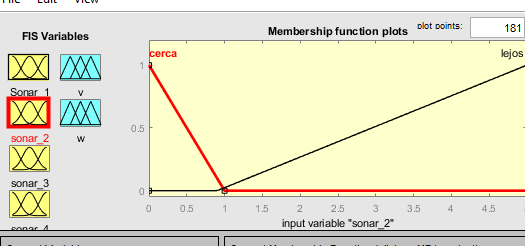
Entradas son sonar 1 2 3 y 4



Configuracion inicial (soanr 1 y 4 = y sonar 2 y 3 =)

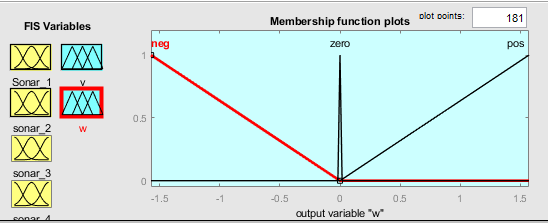






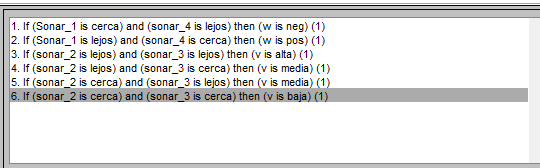
Interfaz de usuario gráfica

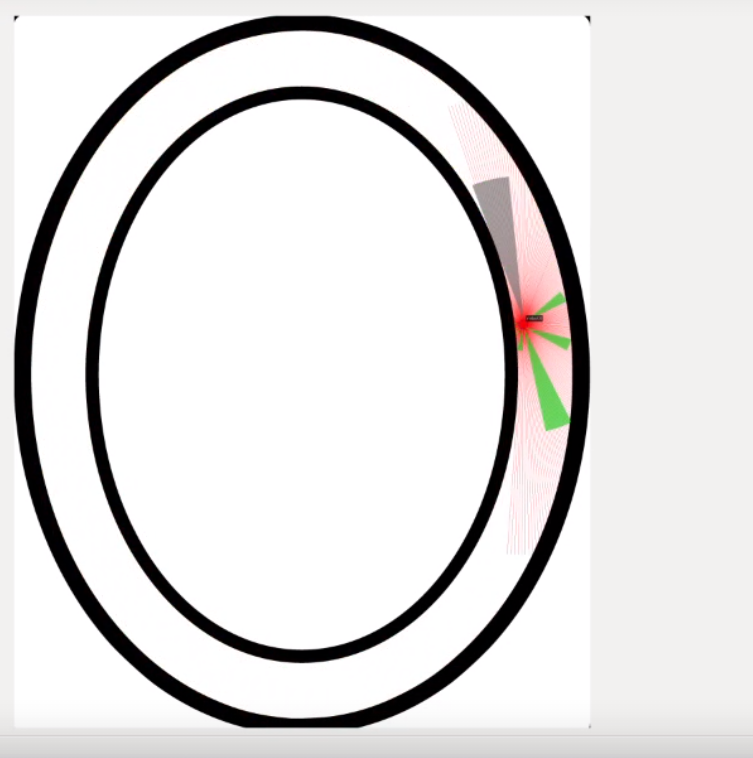
Descripción generada automáticamente



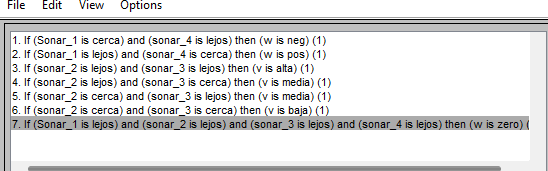
Sigue recto y hace un giro hacia la izquierda y se choca

Las primeras dos reglas eran erradas, ya que confundimos la orientación de la velocidad angular, teniendo que cambiar el valor positivo por negativo y viceversa. Tras esto, el robot fue capaz de casi completar el circuito, chocando contra la pared interior en la zona de los cuadrantes 3 y 4. Las rglas en cuestión y la imagen del simulador son las siguientes

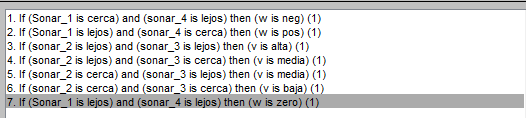




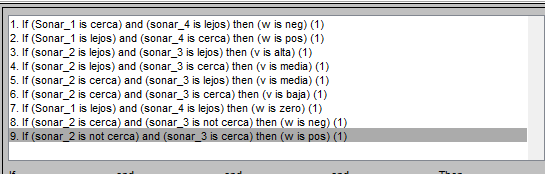
Tras este intento, se añadió una nueva regla que establecía W nula cuando todos los sonares ofrecían lecturas lejanas. Choco un poco antes que en el caso anterior.



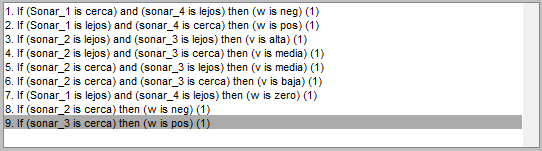
Tras esto se modifico dicha regla, teniendo en cuenta únicamente los sonares 1 y 4. El resultado fue similar.



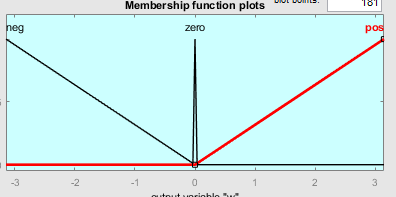
A continuación, se introdujeron dos nuevas reglas, estableciendo que cuando uno de estos dos sonares (2 y 3) están cerca y el otro no, realice un giro en la dirección pertinente. Choco a la misma altura en la pared contraria.



Tras esto, se eliminaron de dichas reglas los sonares con cercanía negada. Choco en la pred contraria en la anterior.

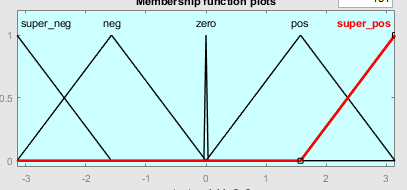


Tras esto procedimos a aumentar el rango de valores de la velocidad angular de (-pi/2 pi/2) a (-pi pi)



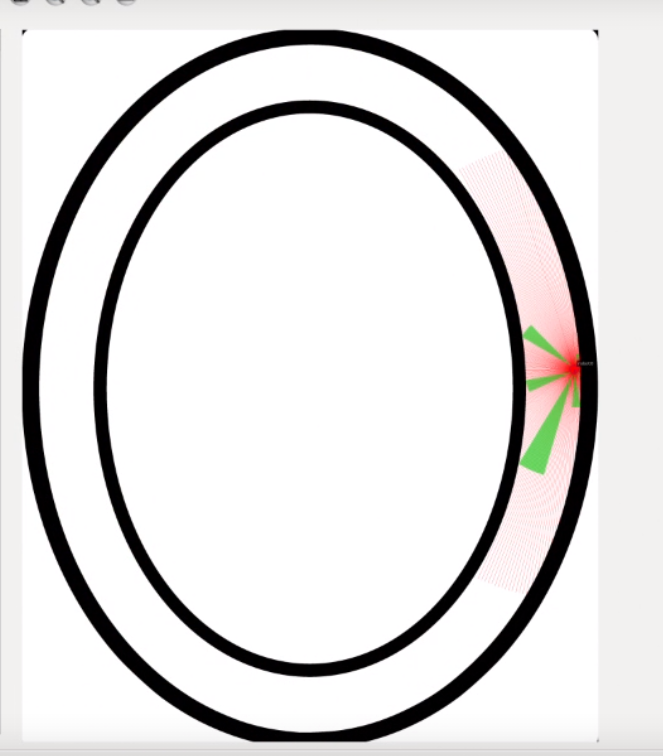
Al haber aumentado el rango, el robot se volvió inestable, aunque choco en la misma zona que en experimentos anteriores. Esto nos llevó a razonar que deberíamos establecer nuevos conjuntos borrosos, haciendo referencia a un giro más o menos brusco.

Así, obtenemos el siguiente resultado:

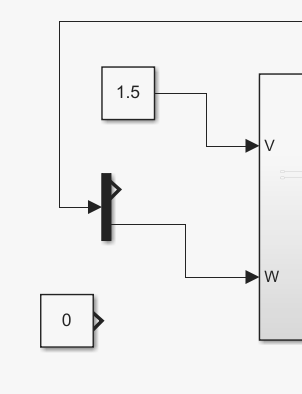


De esta forma, las ultimas reglas añadidas, implican giros mas burscos (super?neg y supr?pos), mientras que el resto continúan igual. Su comportamiento se volvió mucho mas estable, pero vovlvio a chocar en la misma zona, por lo que establecimos las reglas originales (sonar i is not cerca)

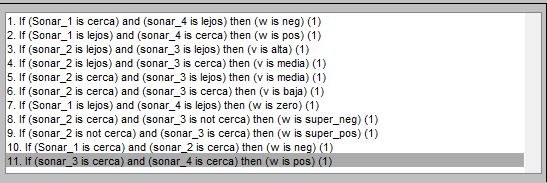
Volvió a chocar en el mismo lugar, por lo que redujimos el valor de velocidad baja para dar tiempo a realizar el giro pertinente. Realizo el recorrido mas largo hasta el momento



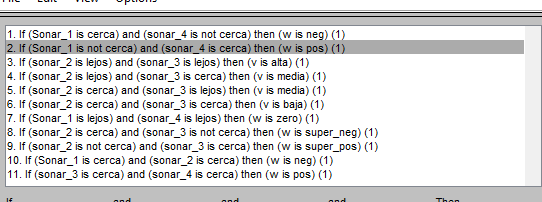
Esto nos llevo a pensar en establecer una velocidad lineal a 1.5 con un modulo constante en el simulink.



El resultado fue el peor hasta el momento, por lo que se decidió ampliar el rango de cercanía de los sonares. El comportamiento fue similar, por lo que se amplio el conjunto de reglas al siguiente



El resultado no aporto demasiado mejoría por lo que se cambiaron las dos primeras reglas:

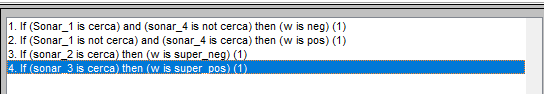


Empezamos de 0:

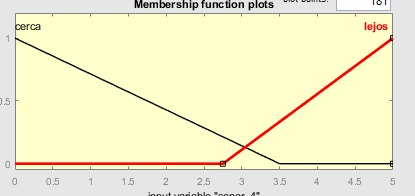
Solo las 2 primeras reglas a v constante

Llego a completar los dos primeros cuadrantes antes de chcoar

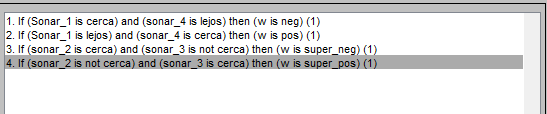
Añadimos reglas de superneg y superpos con 2 y 3



Se redujo el conjunto lejos de los sonares 1 y 4

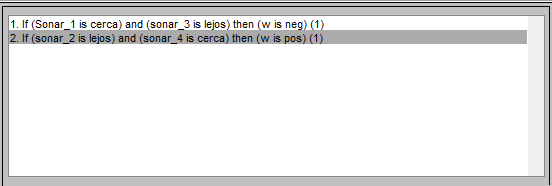


El comportamiento no mejoro por lo que se cambiaron las dos ultimas reglas:

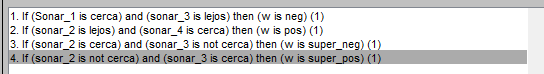


Descubrimos un error en el conjunto lejos del sonar 1, y procedimos a arreglarlo. No hubo mejoría.

Tras esto razonamos que debíamos gestionar los giros no muy grandes usando un sonar interno y otro externo, de tal manera que con el interno comprobamos no tener cerca un obstáculo y con el externo estar suficientemente lejos de la pared:

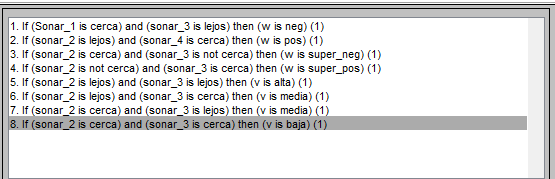


De nuevo, debido a que volvió a fallar aunque mejor, añadimos las reglas de giro bruusco, únicamente teniendo encuenta los sonares frontales



Descrubrimos que el fallo era que el stop time estaba seteado a 10, tras cambiarlo a 100, con esta configuración fue capaz de completar la vuelta. Tras esto, decidicmos hacer pruebas sobre cual era la velocidad máxima a la que podía completar el circuito. La primera prueba con velocidad constante 5 fue fructífera. 10 fue una velocidad demasiado rápida.Ningun otro valor mayor de 5 fue aceptable.

Tras esto añadimos reglas referentes a la velocidad, las mismas con las que comenzamos a intentar resolver el problemañ



Con esta configuración logramos completar el circuito en 14 segundos. CIRCUITO SIN OBSTACULOS WEY

En el de obstáculos se chokio

así que añadimos una regla que cuando todos los sonares están lejos w es 0

Da una vuelta y luego se choca. Cambiamos el conjunto de velocidad centrando los limites del conjunto media, de esta forma las velocidades medias se acercan mas al centro, así cuando los sonares 2 y 3 esten en cerca y lejos respectivamente se frena un poco mas y se puede evitar a tiempo los obstaculos

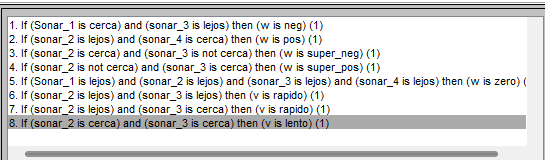
Gráfico, Gráfico de líneas

Descripción generada automáticamente

AHORA HEMOS VISTO QUE VA SOBRADO, ASIQUE HAY QUE HACER QUE VAYA MAS RAPIDO JODER

Hemos deciddio usar únicamente dos conjuntos para la velocidad, ya que el rango de valores de velocidad lineal es tan pequeño que distinguir entre tres clases de velocidad puede ser excesivo y únicamente con velocidades rápida y lenta será suficiente

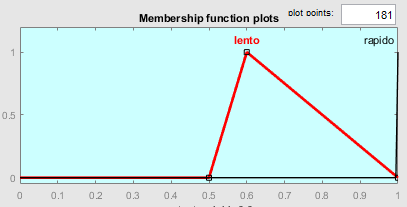
En consecuencia, rehicimos las reglas en base a los nuevos conjuntos:

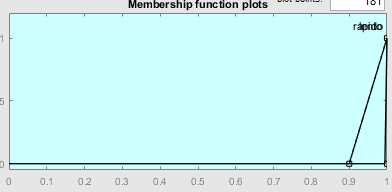


Aun así, el robot sigue moviéndose muy despacio, por lo que procedemos a cambiar los conjuntos con el fin de aumentarla. El primer intento se realizó ajustando la velocidad baja de tal manera que el valor más bajo sea 0.3:



Sigue siendo muy lento, por lo que acotamos aún mas los conjuntos:

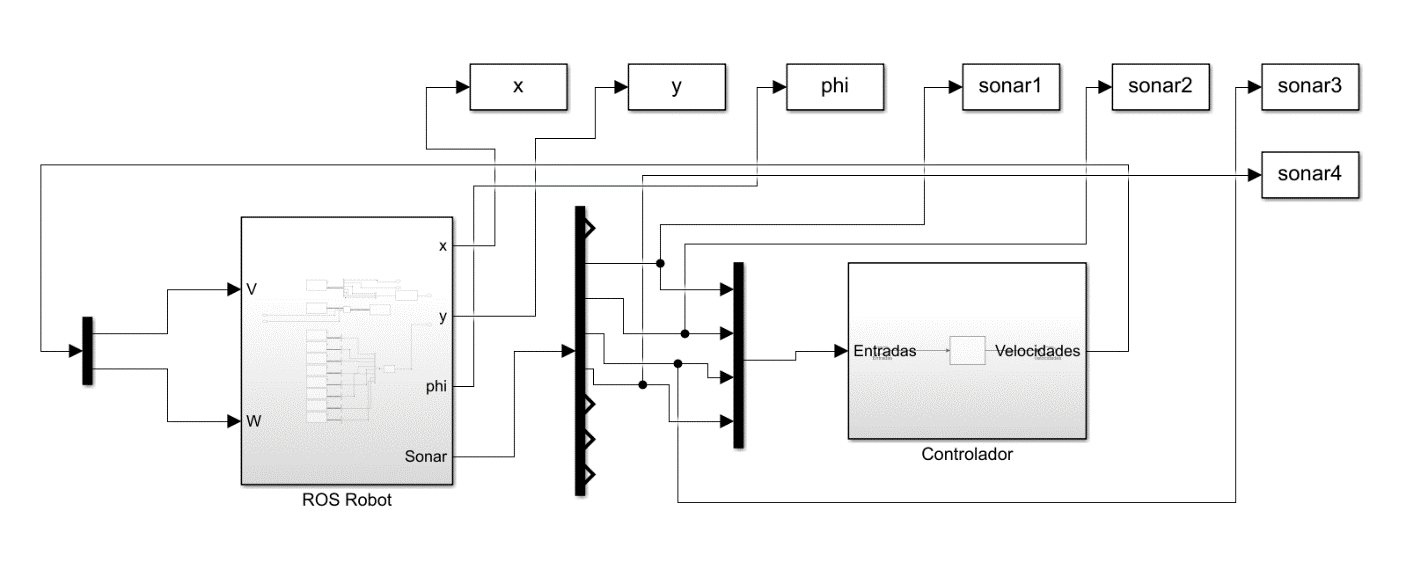
  
otra vez lo mismo

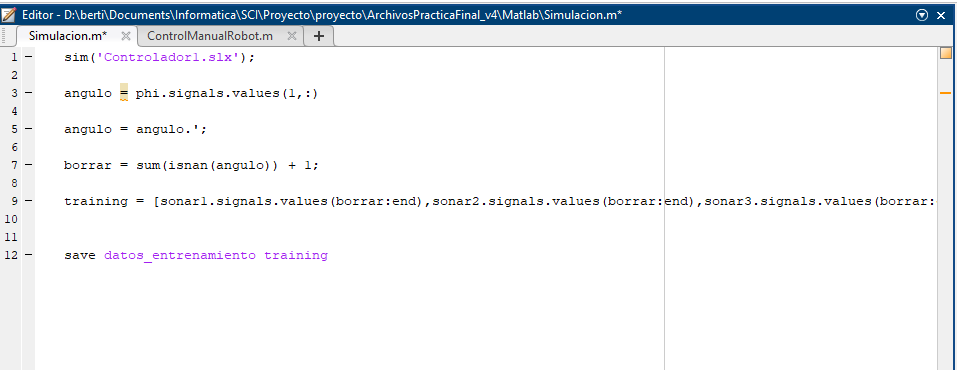


NEUROBORROSO

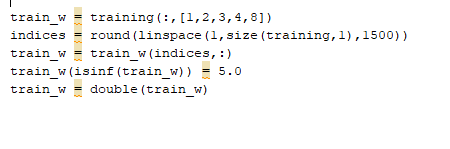
A la hora de recolectar datos para el entrenamiento del controlador neuroborroso, pensamos que seria mejor idea tomar los datos de una trayectoria de nuestro controlador anterior que de una ejecución realizada por nosotros, debido a la dificultad de realizar el circuito de una manera rápida con el control manual.

Así creamos el siguiente script de Matlab que toma los datos directamente del controlador, y a partir de los cuales construimos por nuestra cuenta la matriz training a partir del comportamiento del controlador anterior. Para ello tuvimos que añadir una serie de módulos al modelo slx como en prácticas anteriores





Una vez tomados los datos, obtenemos los pertinentes para el entrenamiento de cada uno de los controladores



Tras estop entrenamos el modelo

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

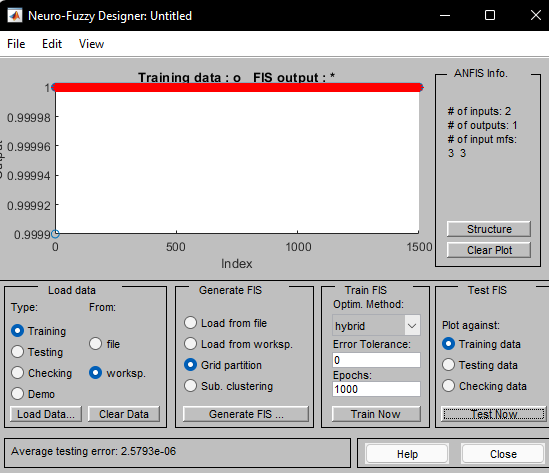
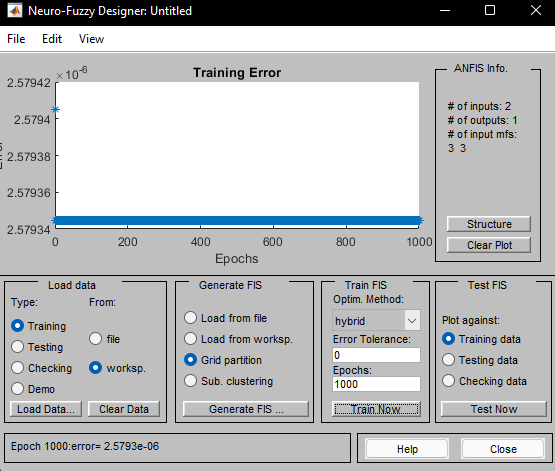
Al ver durante cientos de épocas una tendencia casi nula de mejoría decidimos parar el entrenamiento en 500 epocas. El error final es de menos de 0.28, lo cual es un gran resultado.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Podemos apreciar en este grafico, que el aprendizaje ha sido más que aceptable.

Entrenamos ahora el controlador referente a la velocidad lineal



Podemos comprobar que el error mínimo se alcanza en las primeras épocas casi instantáneamente, esto se debe a que el controlador borroso del que hemos extraído los datos de entrenamiento intenta siempre mantener la velocidad del robot lo más cerca del máximo para así tardar lo menos posible, por ello da la sensación en el grafico debido a la baja resolución de este, de que es un valor constante, pero se debe al redondeo.

Ahora exportamos los controladores generados de ambos entrenamientos y los usamos para controlar al robot intergandolos de esta manera en simulink

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente