Sistemas de Control Inteligente

PF – Diseño de Controladores Borrosos y Neuronales para la maniobra de aparcamiento de un vehículo

Grado en Ingeniería Informática
Universidad de Alcalá

Raúl Moratilla Núñez – 03215952T

Sergio Lorenzo Montiel – 03217363P

14/01/2024

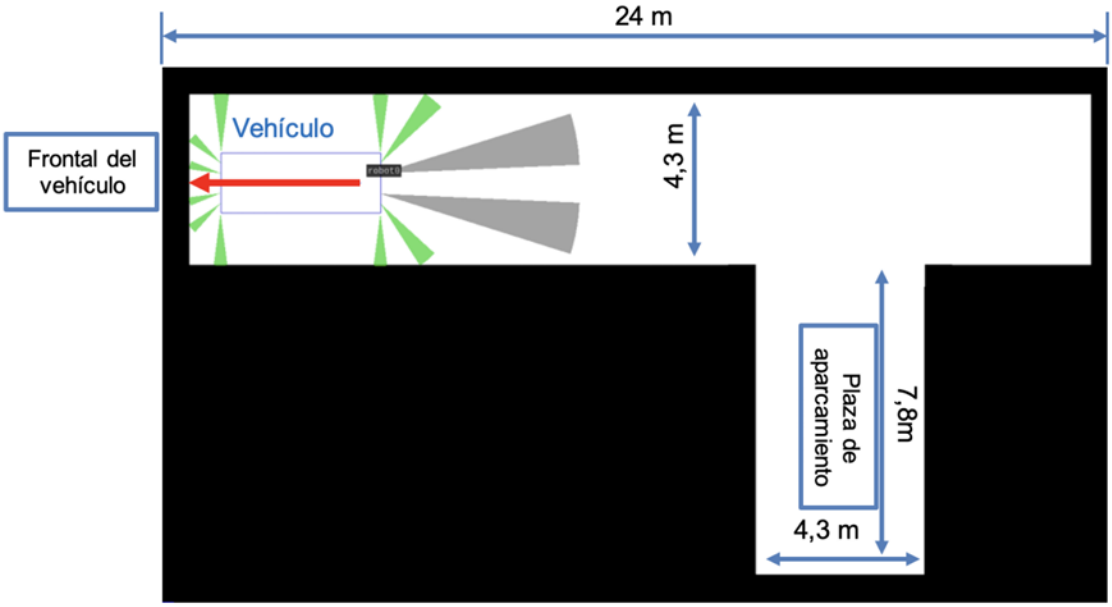
Índice

1. Introducción

La práctica consiste en diseñar un controlador borroso y otro neuronal para resolver un problema de aparcamiento de un coche en batería. Se establece un entorno de unas dimensiones concretas sobre el que se debe trabajar. Primeramente, será necesario realizar un controlador borroso, y sobres ese se generarán los datos de entrenamiento para diseñar el controlador neuronal.

2. Análisis del entorno

El entorno sobre el que se realiza la práctica es el siguiente:



Además, se tiene que el coche mide 1.5 metros de ancho y 4.5 de largo, y es capaz de tomar velocidades en el intervalo [-30, 30] y girar el volante en el ángulo [-90, 90].

Tras realizar varias pruebas, se obtienen varias conclusiones:

- La máxima medición que pueden obtener los sensores es 5.

- La máxima medición que pueden obtener los sensores 0, 5, 6 y 11 con el coche recto y sin estar en la maniobra de giro es aproximadamente 2.8. La máxima distancia que puede obtener el sensor 10 en la misma situación es aproximadamente 3.95. Sin embargo, esta situación no debería ocurrir dados los rangos que se han dado en los que se debe poner el coche, por lo que se puede presuponer que estos valores son menores.

- Con el robot girando a una “alta” velocidad, si un sensor se acerca a la pared a una distancia menor a 1 y se sigue acercando, el coche se queda pillado unos segundos hasta seguir con la ejecución.

- El ángulo de giro entre 80º y 90º y entre -80º y -90º no funciona igual que el resto de los ángulos de giro, ya que realiza una rotación en la que el coche prácticamente no avanza en el espacio. Se puede aprovechar esto en favor de realizar un giro más rápido.

- La comunicación entre ROS y Simulink tiene un límite, y al poner velocidades muy altas, no da tiempo a que se pase toda la información necesaria y el coche se pasa de frenada.

- En las ejecuciones se obtiene un fallo en los sensores, y es que aleatoriamente, los primeros valores de estos son igual a 0, algo que es imposible dado que aún se encuentra en la posición inicial. En nuestro caso estos fallos se producen aproximadamente hasta entre el segundo 0.5 y el segundo 1.5.

Tras haberlo analizado de forma detenida, se ha concluido que se usarán 7 sensores, el 0, el 5, el 6, el 8, el 9, el 10 y el 11, que son los que creemos necesarios para el correcto funcionamiento del diseño.

En cuanto al entorno de programación utilizado, se usará la versión de Matlab 2022b.

3. Condición de parada

Hay que determinar una condición de parada para poner el flag_parada a 1 y que la ejecución finalice correctamente. En nuestro caso, hemos aprovechado este bloque para solucionar el error de comunicación entre ROS y Simulink en el que aparecen varios ceros al inicio de la ejecución. La solución implementada es que, mientras eso ocurra, el coche no se moverá, es decir tendrá velocidad lineal 0. El problema de esta solución es que aumenta el tiempo delta sin que el coche no haga nada entre 0.5 y 1.5 segundos, por lo que al dato final de "delta_time" habría que restarle esta cantidad de tiempo. La condición de parada en sí será parar la ejecución cuando tanto el sensor 8 como el sensor 9 detecten una medición menor a 1.5. Esta será la condición tanto para el neuronal como para el borroso. El código es el siguiente:

```
function [flag_parada, filter_speed] = condicion_parada(raw_speed, sens)

if ~ismember(0, sens) && sens(4) <= 1.5 && sens(5) <= 1.5
    filter_speed = 0.0;
    flag_parada=1;

elseif ismember(0, sens)
    filter_speed = 0.0;
    flag_parada=0;

else
    filter_speed = raw_speed;
    flag_parada=0;
end
```

Además, se han añadido en el modelo Simulink dos variables To Workspace para saber el valor de los sensores utilizados y de tanto la velocidad lineal como el ángulo de giro del volante en cada momento de la ejecución. Además, permite saber hasta cuándo ha ocurrido el problema de los ceros. En estas variables, cada fila representa una décima de tiempo transcurrido.

4. Controlador Borroso

4.1. Introducción a Borroso

Los controladores borrosos, también conocidos como sistemas de lógica difusa o control borroso, son un tipo de controlador utilizado en sistemas de automatización y control. A diferencia de los controladores convencionales que emplean lógica booleana y operaciones precisas, los controladores borrosos incorporan la lógica difusa, que permite manejar la imprecisión y la incertidumbre presentes en muchos sistemas del mundo real.

La base del control borroso es que, en lugar de asignar valores binarios (verdadero o falso) a las variables de entrada y salida, la lógica difusa permite representar la verdad como una función de membresía difusa, que asigna grados de pertenencia en un rango de 0 a 1. Esto refleja la naturaleza más flexible y gradual de muchas situaciones del mundo real.

Los controladores borrosos constan de tres partes fundamentales: la borrosificación, que convierte las entradas precisas en valores difusos; el motor de inferencia, que aplica reglas lógicas difusas para generar salidas borrosas; y la desborrosificación, que convierte las salidas borrosas en valores precisos. Estos controladores son especialmente útiles en sistemas complejos donde las reglas convencionales y las relaciones matemáticas no son fácilmente modeladas.

La capacidad de los controladores borrosos para manejar la imprecisión y la incertidumbre los hace aplicables en una amplia gama de campos, desde sistemas de control industrial hasta electrodomésticos, robótica y vehículos autónomos. Su flexibilidad y adaptabilidad a situaciones no lineales y variables cambiantes han contribuido a su popularidad en la ingeniería de control.

4.2. Aparcamiento en batería con controlador borroso

En este trabajo se va a desarrollar un controlador borroso que sea capaz de realizar un aparcamiento en batería en el entorno proporcionado por los profesores de la asignatura, analizado en el apartado 2, **Análisis del entorno.**

En cuanto a las características del controlador, es necesario destacar las siguientes:

- Controlador de tipo Mamdani.

Un controlador de tipo Mamdani es un tipo específico de controlador borroso, nombrado en honor a su creador, Ebrahim Mamdani. Este tipo de controlador borroso se caracteriza por su estructura y método de inferencia. Sus componentes principales son:

- Borrosificación**.** La entrada del sistema se convierte en valores borrosos mediante funciones de membresía que asignan grados de pertenencia en un rango de 0 a 1. Estas funciones capturan la imprecisión y la incertidumbre asociadas con las variables de entrada.

- Base de Reglas. El conocimiento experto se codifica mediante reglas if-then que relacionan las entradas borrosas con las salidas borrosas. Cada regla describe cómo la combinación de ciertos valores de entrada afecta a ciertos valores de salida.

- Motor de Inferencia. Las reglas se aplican mediante operadores lógicos borrosos (como AND, OR) para determinar las contribuciones individuales de cada regla a la salida borrosa. Este paso implica la combinación de los grados de verdad asociados con cada regla.

- Desborrosificación. Los resultados borrosos se transforman en una salida precisa aplicando un proceso de desborrosificación. Esto generalmente implica determinar el valor central o la media ponderada de las salidas borrosas.

- Conectiva AND: Mínimo.

En nuestro caso, la conectiva AND será el mínimo, lo que quiere decir que, en caso de cumplirse una regla, se escogerá como conjunto de esa regla aquel que tenga una altura mínima en la gráfica.

- Agregación de Reglas: Máximo.

Es decir, cuando se cumplan varias reglas, el conjunto final será aquel formado por la unión de los conjuntos obtenidos con cada regla.

- Proceso de Desborrosificación: Centroid.

Este método consiste en calcular el centro de masa de la figura generada por la agregación de reglas, siguiendo la siguiente fórmula:

$$\frac{\int_{-\infty}^{\infty} \mu_A(y) \cdot y \, dy}{\int_{-\infty}^{\infty} \mu_A(y) \, dy}$$

Donde:

- $\mu_{A(}y)$ es la función de membresía del conjunto difuso resultante.

- y es la variable de salida.

- La integral se realiza a lo largo del rango completo de la variable de salida.

- Inputs:

Para realizar el controlador borroso, se hará uso de 7 sensores, (0, 5, 6, 8, 9, 10, 11), por lo que habrá 7 gráficas de input. Se explicará el uso de cada sensor más adelante.

- Outputs:

Como se desea obtener la velocidad lineal y el ángulo de giro del volante, será necesario que haya dos gráficas en el output.

- Estrategia.

La estrategia que se va a buscar con el controlador será la siguiente.

1. Centrar el coche en el pasillo inicial desde la posición en la que inicie hasta una posición en la que los sensores 0 y 11 midan entre 1 y 1.4 y los sensores 5 y 6 midan entre 1.4 y 1.8. El coche no quedará centrado, sino desplazado ligeramente hacia abajo. Esto permite realizar el giro de una forma más directa y haciendo que el coche se quede centrado en el aparcamiento.

2. Maniobra de giro. Aprovechar el giro de 90 grados hacia la izquierda cuando el sensor 10 no detecte nada hasta antes de que el coche se choque con la esquina superior (sensor 5), momento en el que se comenzará a girar menos hasta que se salve el obstáculo para volver a girar 90.

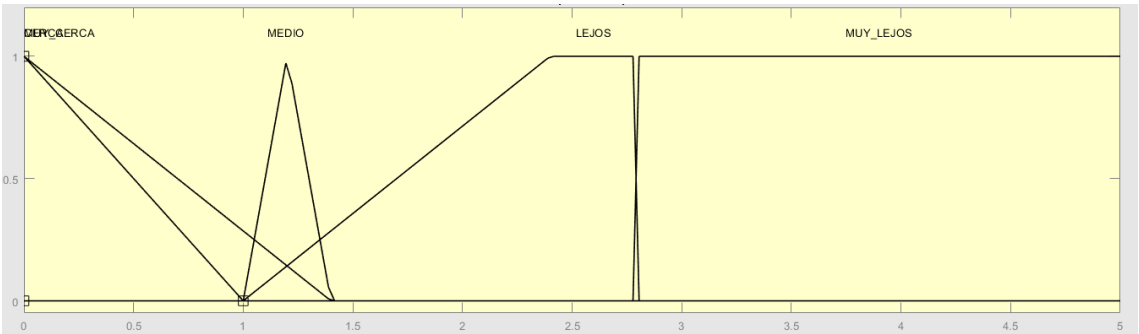
3. Enderezar el coche. En el aparcamiento, cuando el coche ya ha entrado, debido a los dos pasos anteriores entra aproximadamente por el centro, pero siempre está ligeramente girado. Por ello, es necesario girar el volante en la dirección correcta para que el coche se ponga recto.

4. Fin de la maniobra. Se considera que el coche ha aparcado cuando tanto el sensor 8 como el sensor 9 realicen una medición menor a 1.5.

4.3. Gráficas de inputs y outputs

Las gráficas utilizadas por cada input y cada output son las siguientes:

- Sensor 0. Input.



Como se observa, este sensor cuenta con 5 funciones de pertenencia:

- MUY_CERCA (linzmf, [0, 1]).

Esta función sirve principalmente para que cuando el coche esté cerca del borde inferior se pueda reducir tanto el giro como la velocidad para evitar el choque.

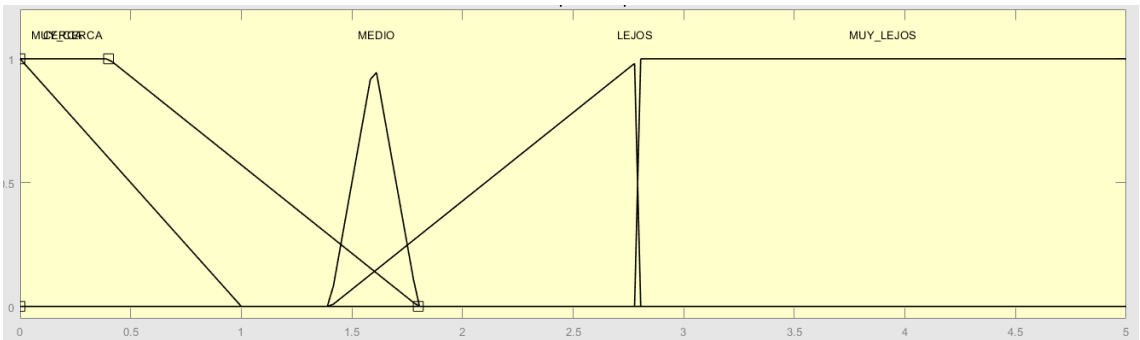
- CERCA (linzmf, [0, 1.4], MEDIO (trimf, [1, 1.2, 1.4]) y LEJOS (trapmf, [1, 2.4, 2.8, 2.8]).

Estas 3 funciones se complementan para mantener el coche centrado en el rango [1, 1.4]. LEJOS termina en 2.8 ya que es lo máximo que podría medir el sensor estando el coche recto y sin tener la zona de aparcamiento próxima.

- MUY_LEJOS (linsmf, [2.8, 2.8]).

Se considera muy lejos a partir de 2.8 por la razón explicada anteriormente. En caso de medir este sensor más de 2.8 se considera que el coche está aparcando, pero aún no ha entrado el coche entero en el aparcamiento y queda el morro delantero fuera.

- Sensor 5. Input.



Como se observa, este sensor cuenta con 5 funciones de pertenencia, todas ellas con equivalencia en el sensor 0:

- MUY_CERCA (linzmf, [0, 1]).

Esta función sirve principalmente para que cuando la esquina superior izquierda del coche se acerque demasiado al borde en la maniobra de giro en el aparcamiento se pueda reducir tanto el giro como la velocidad para evitar el choque.

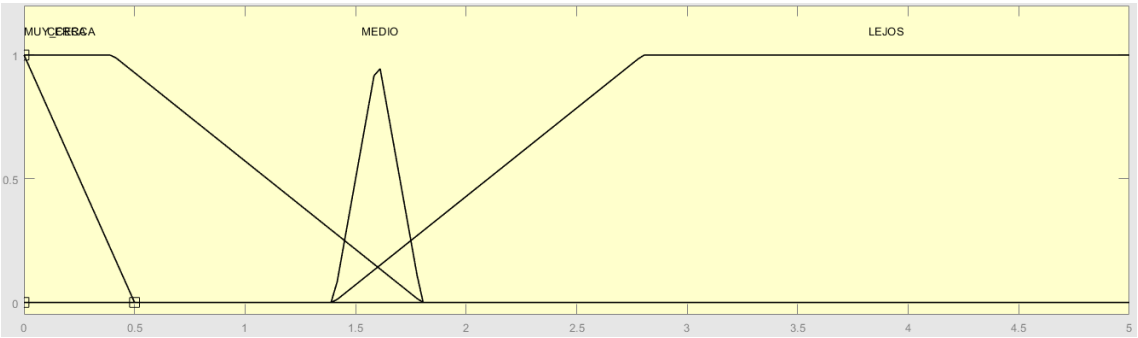
- CERCA (linzmf, [0.4, 1.8], MEDIO (trimf, [1.4, 1.6, 1.8]) y LEJOS (trimf, [1.4, 2.8, 2.8]).

Estas 3 funciones se complementan para mantener el coche centrado en el rango [1.4, 1.8]. LEJOS termina en 2.8 ya que es lo máximo que podría medir el sensor estando el coche recto y sin tener la zona de aparcamiento próxima.

- MUY_LEJOS (linsmf, [2.8, 2.8]).

Se considera muy lejos a partir de 2.8 por la razón explicada anteriormente. En caso de medir este sensor más de 2.8 se considera que el coche está aparcando, pero aún no ha entrado el coche entero en el aparcamiento y queda el morro delantero fuera.

- Sensor 6. Input.



Como se observa, este sensor cuenta con 4 funciones de pertenencia:

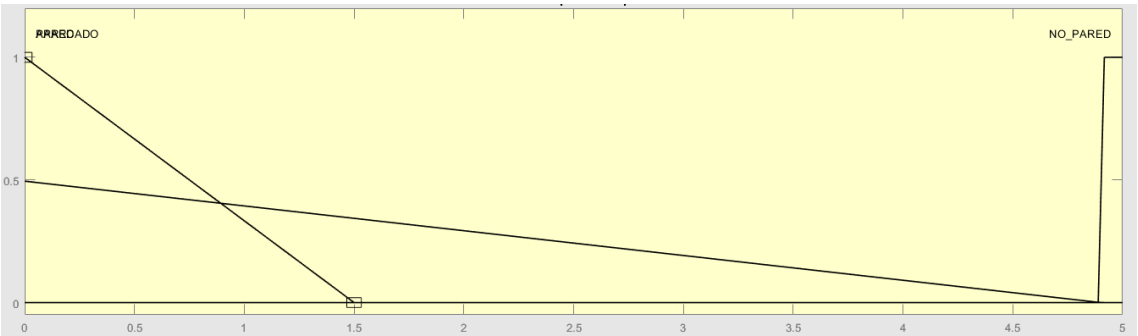
- MUY_CERCA (linzmf, [0, 0.5]).

Esta función sirve para que, en caso de que el giro se haya ido demasiado lejos en el aparcamiento en algún caso, el coche no se choque con la esquina más lejana del aparcamiento, reduciendo el giro y la velocidad.

- CERCA (linzmf, [0.4, 1.8], MEDIO (trimf, [1.4, 1.6, 1.8]) y LEJOS (trimf, [1.4, 2.8, 2.8]).

Estas 3 funciones se complementan para mantener el coche centrado en el rango [1.4, 1.8]. En este caso no es necesaria una función MUY_LEJOS.

- Sensor 8 y Sensor 9. Input.



Ambas gráficas son iguales, con 3 funciones de pertenencia en los mismos rangos:

- APARCADO (linzmf, [0, 1.5]).

Esta función sirve para reducir tanto la velocidad como el giro a 0 cuando el coche esté aparcado.

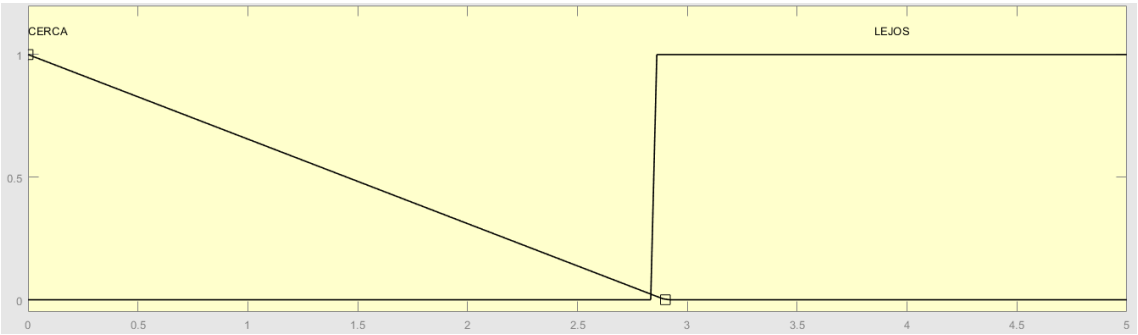
- PARED (linzmf, [-5, 4.9]).

Esta función sirve para saber cuándo el coche está de frente con alguna pared. La función va desde 0 hasta 0.5 en altura porque se quiere que las reglas en las que se usa tengan menos influencia que el resto de las reglas.

- NO_PARED (linsmf, [4.89, 4.89]).

Esta función sirve para detectar cuando el sensor 8 no toca la pared. Tiene una forma cuadrada para poder hacer el not, de tal forma que no haya que crear otra función complementaria, funcionando en este caso de forma binaria.

- Sensor 10. Input.



Este sensor sirve para saber cuándo realizar la maniobra de aparcamiento, y tiene 2 funciones de pertenencia

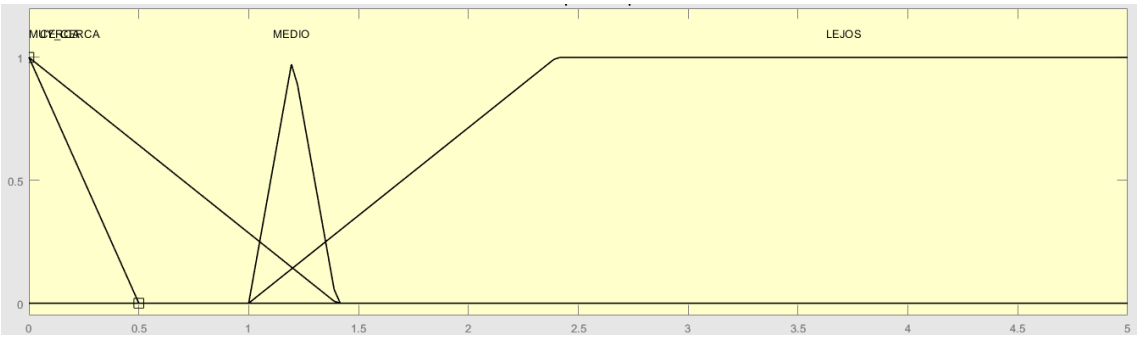
- CERCA (linzmf, [0, 2.9]).

Lo cual significa que el coche no debe iniciar la maniobra de aparcamiento.

- LEJOS (linsmf, [2.89, 5]).

Lo que significa que el coche debe realizar la maniobra de aparcamiento.

- Sensor 11. Input.



El sensor 11 sirve para mantener el coche centrado y que no se choque con la pared al estar muy cerca:

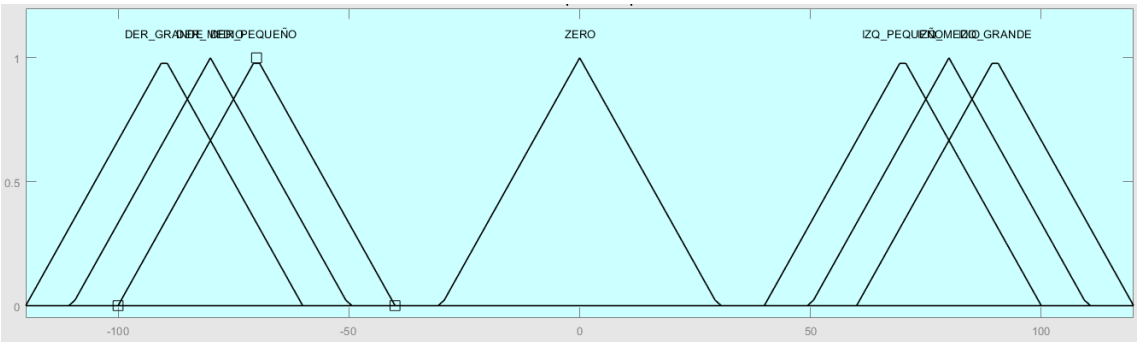
- MUY_CERCA (linzmf, [0, 0.5]).

Esta función sirve para que el coche no se choque con la pared inferior.

- CERCA (linzmf, [0, 1.4], MEDIO (trimf, [1, 1.2, 1.4]) y LEJOS (linsmf [1, 2.4]).

Estas 3 funciones se complementan para mantener el coche centrado en el rango [1, 1.4]. En este caso no es necesaria una función MUY_LEJOS.

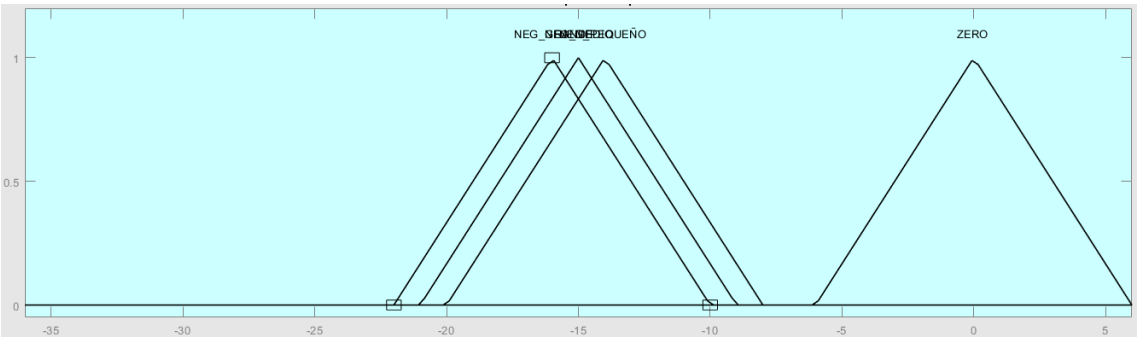
- W (Ángulo de Giro). Output.



Consta de 3 funciones de giro a la izquierda, otras 3 de giro a la derecha y 1 para no girar. Todas las funciones son triangulares, con el valor máximo en el centro y de rango 60º. Las funciones son:

- DER_GRANDE (centrada en -90)
- DER_MEDIO (centrada en -80)
- DER_PEQUEÑO (centrada en -70)
- ZERO (centrada en 0)
- IZQ_PEQUEÑO (centrada en 70)
- IZQ_MEDIO (centrada en 80)
- IZQ_GRANDE (centrada en 90)

- V (Velocidad lineal). Output.



Consta de 4 funciones, todas ellas triangulares, con el valor máximo en el centro y de rango 12. No ha sido necesario crear funciones de velocidad positiva. Las funciones son:

- NEG_GRANDE (centrado en -16)
- NEG_MEDIO (centrado en -15)
- NEG_PEQUEÑO (centrado en -14)
- ZERO (centrado en 0)

4.4. Reglas

Se han creado un total de 27 reglas, aunque en su mayoría son reglas similares que simplemente obedecen a todas las combinaciones posibles o que son simétricas. Para explicar las reglas, estas se organizarán en grupos para que sea más fácil y rápida su comprensión. Se explicarán según a qué parte de la estrategia explicada anteriormente pertenecen.

- Centrado del coche.

Hay un total de 9 reglas que se encargan de la tarea de centrar el coche en el lugar adecuado. Este lugar, como ya se ha explicado, no está exactamente en el centro, si no que está ligeramente abajo para facilitar el giro posteriormente. Estas reglas salen de combinar las funciones CERCA y LEJOS de los sensores 6 y 11. De estas combinaciones se obtienen 2 reglas. Es necesario que se activen estas reglas cuando el sensor 8 o el sensor 9 no detecten una pared delante. Por ello, es necesario duplicarlas, una por sensor. Se obtendrían 4. En caso de que tanto el 11 como el 6 midan MEDIO, es necesario mirar en este caso qué miden tanto el 0 como el 5. De aquí salen otras 4 combinaciones. La novena sale de la combinación en la que todos son MEDIO, en la que también es necesario que ni el 8 ni el 9 estén tocando pared. En las 4 primeras reglas se comprueba que ni el sensor 0 ni el sensor 5 detecten MUY_LEJOS.

1. If (sensor0 is not MUY_LEJOS) and (sensor5 is not MUY_LEJOS) and (sensor6 is CERCA) and (sensor8 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is LEJOS) then (W is IZQ_MEDIO)(V is NEG_PEQUEÑO) (1)
2. If (sensor0 is not MUY_LEJOS) and (sensor5 is not MUY_LEJOS) and (sensor6 is CERCA) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is LEJOS) then (W is IZQ_MEDIO)(V is NEG_PEQUEÑO) (1)
3. If (sensor0 is not MUY_LEJOS) and (sensor5 is not MUY_LEJOS) and (sensor6 is LEJOS) and (sensor8 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is CERCA) then (W is DER_MEDIO)(V is NEG_PEQUEÑO) (1)
4. If (sensor0 is not MUY_LEJOS) and (sensor5 is not MUY_LEJOS) and (sensor6 is LEJOS) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is CERCA) then (W is DER_MEDIO)(V is NEG_PEQUEÑO) (1)
5. If (sensor0 is CERCA) and (sensor5 is LEJOS) and (sensor6 is MEDIO) and (sensor8 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is MEDIO) then (W is IZQ_GRANDE)(V is NEG_PEQUEÑO) (1)
6. If (sensor0 is CERCA) and (sensor5 is LEJOS) and (sensor6 is MEDIO) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is MEDIO) then (W is IZQ_GRANDE)(V is NEG_PEQUEÑO) (1)
7. If (sensor0 is LEJOS) and (sensor5 is CERCA) and (sensor6 is MEDIO) and (sensor8 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is MEDIO) then (W is DER_GRANDE)(V is NEG_PEQUEÑO) (1)
8. If (sensor0 is LEJOS) and (sensor5 is CERCA) and (sensor6 is MEDIO) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is MEDIO) then (W is DER_GRANDE)(V is NEG_PEQUEÑO) (1)
9. If (sensor0 is MEDIO) and (sensor5 is MEDIO) and (sensor6 is MEDIO) and (sensor8 is NO_PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is MEDIO) then (W is ZERO)(V is NEG_GRANDE) (1)

La lógica que siguen las 9 reglas es la siguiente:

Sensor 0	Sensor 5	Sensor 6	Sensor 11	W	V
NOT_MUY_LEJOS	NOT_MUY_LEJOS	CERCA	LEJOS	IZQ_MEDIO	MEDIA
NOT_MUY_LEJOS	NOT_MUY_LEJOS	LEJOS	CERCA	DER_MEDIO	MEDIA
CERCA	LEJOS	MEDIO	MEDIO	IZQ_GRANDE	MEDIA
LEJOS	CERCA	MEDIO	MEDIO	DER_GRANDE	MEDIA
MEDIO	MEDIO	MEDIO	MEDIO	ZERO	GRANDE

Cuando la parte trasera del coche está centrada, es necesario poner el giro de 90º o -90º ya que es la forma más rápida de centrarlo, y en el resto de los casos, se debe ir al lado contrario al que esté el morro con ángulos de 80º o -80º. Sólo se pone ángulo 0º y velocidad grande cuando el coche está centrado.

- Maniobra de giro.

Para realizar la maniobra de giro son necesarias tan solo 2 reglas.

```
10. If (sensor10 is LEJOS) then (W is IZQ_GRANDE)(V is NEG_PEQUEÑO) (1)
11. If (sensor5 is MUY_CERCA) and (sensor10 is LEJOS) then (W is IZQ_PEQUEÑO)(V is ZERO) (1)
```

La primera de ellas indica que cuando el sensor 10 detecte lejos, quiere decir que el aparcamiento está próximo y que hay que girar todo lo que sea posible.

La segunda controla que el coche no se choque con la esquina superior derecha, reduciendo tanto el ángulo del volante como la velocidad. Aunque en esta regla la velocidad sea 0, se realizará una media con la regla anterior ya que se activan simultáneamente.

- Enderezar el coche.

La fase en la que se endereza el coche consta nuevamente de varias reglas que son muy similares pero que se multiplican ya que es necesario indicar todas las combinaciones. A su vez, consta de varias fases.

```
12. If (sensor5 is MUY_LEJOS) and (sensor8 is PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) then (W is IZQ_MEDIO)(V is NEG_MEDIO) (1)
13. If (sensor0 is MUY_LEJOS) and (sensor8 is PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) then (W is IZQ_MEDIO)(V is NEG_MEDIO) (1)
14. If (sensor0 is MUY_LEJOS) and (sensor8 is NO_PARED) and (sensor9 is PARED) and (sensor10 is CERCA) then (W is DER_MEDIO)(V is NEG_MEDIO) (1)
15. If (sensor5 is MUY_LEJOS) and (sensor8 is NO_PARED) and (sensor9 is PARED) and (sensor10 is CERCA) then (W is DER_MEDIO)(V is NEG_MEDIO) (1)
16. If (sensor0 is MUY_LEJOS) and (sensor6 is CERCA) and (sensor8 is NO_PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is LEJOS) then (W is DER_MEDIO)(V is NEG_MEDIO) (1)
17. If (sensor5 is MUY_LEJOS) and (sensor6 is CERCA) and (sensor8 is NO_PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is LEJOS) then (W is IZQ_MEDIO)(V is NEG_MEDIO) (1)
18. If (sensor0 is MUY_LEJOS) and (sensor6 is LEJOS) and (sensor8 is NO_PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is CERCA) then (W is IZQ_MEDIO)(V is NEG_MEDIO) (1)
19. If (sensor5 is MUY_LEJOS) and (sensor6 is LEJOS) and (sensor8 is NO_PARED) and (sensor9 is NO_PARED) and (sensor10 is CERCA) and (sensor11 is CERCA) then (W is DER_MEDIO)(V is NEG_MEDIO) (1)
20. If (sensor5 is not MUY_LEJOS) and (sensor8 is PARED) and (sensor9 is not NO_PARED) and (sensor10 is not LEJOS) then (W is DER_MEDIO)(V is NEG_PEQUEÑO) (1)
21. If (sensor5 is not MUY_LEJOS) and (sensor8 is not NO_PARED) and (sensor9 is PARED) and (sensor10 is not LEJOS) then (W is IZQ_MEDIO)(V is NEG_PEQUEÑO) (1)
22. If (sensor8 is PARED) and (sensor9 is PARED) and (sensor10 is CERCA) then (W is ZERO)(V is NEG_MEDIO) (1)
```

Las 8 primeras se aplican cuando la parte trasera del coche ha entrado en el aparcamiento, pero el morro aún no. Estas enderezan y centran el coche parcialmente. Es decir, puede que el coche no esté enderezado del todo, pero no se dispone de información para enderezarlo más. La lógica de estas reglas se resume en que, si se toca una pared con el sensor 8 o 9 y el otro sensor no detecta nada, se gira en el sentido en el que está el sensor que no detecte nada. En caso de que ni el 8 ni el 9 detecten algo, se intenta centrar en base a la información de los sensores 0 y 11.

Las 3 siguientes reglas se aplican cuando el coche ya ha entrado entero, y se centran únicamente en enderezar el coche, ya que se considera que al llegar a este punto ya está lo suficientemente centrado. Estas funciones utilizan la sentencia NOT sobre la función NO_PARED de los sensores 8 y 9, MUY_LEJOS del sensor 5 y LEJOS del sensor 10, ya

que interesa que estos valores funcionen como booleanos para que la lógica borrosa se aplique a los valores de los sensores 8 y 9 en la función de PARED. La lógica de estas reglas consiste en girar en la dirección en la que la medición sea más grande entre el sensor 8 y 9.

- Aparcar.

Esta fase consiste en una simple regla en la que, si el sensor 8 y el 9 están en APARCADO, se pone la velocidad en ZERO. Esta función se complementa con la condición de parada, explicada anteriormente.

```
23. If (sensor8 is APARCADO) and (sensor9 is APARCADO) then (W is ZERO)(V is ZERO) (1)
```

- Auxiliares.

Son 4 funciones que vigilan que, en cualquier momento, el coche no se choque con las paredes, poniendo ángulo de giro de volante ZERO para reducir el grado de giro (hace media con el resto de las reglas) y reduciendo la velocidad.

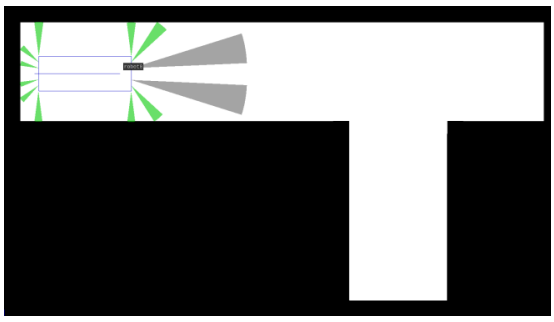
```
24. If (sensor0 is MUY_CERCA) and (sensor10 is CERCA) then (W is IZQ_PEQUEÑO)(V is NEG_PEQUEÑO) (1)
25. If (sensor5 is MUY_CERCA) and (sensor10 is CERCA) then (W is ZERO)(V is NEG_PEQUEÑO) (1)
26. If (sensor6 is MUY_CERCA) then (W is IZQ_MEDIO)(V is NEG_PEQUEÑO) (1)
27. If (sensor11 is MUY_CERCA) then (W is DER_MEDIO)(V is NEG_PEQUEÑO) (1)
```

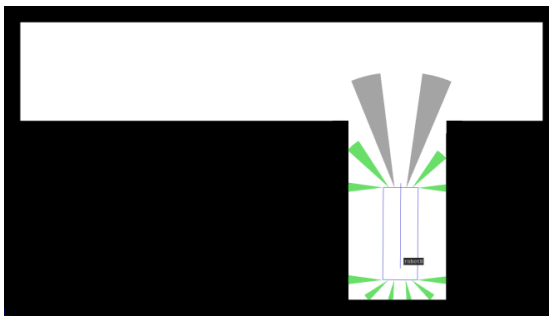
4.5. Ejecución

Se mostrará la posición inicial, la posición final y la salida de evaluación maniobra en cada una de las ejecuciones. Es necesario recordar que el tiempo delta en realidad es menor al tiempo efectivo en el que aparca el robot.

4.5.1. Entorno Inicial

El entorno inicial es el que comienza en la posición (5, 10.5, 3.14159). La posición inicial, final y evaluación de la maniobra son las siguientes:

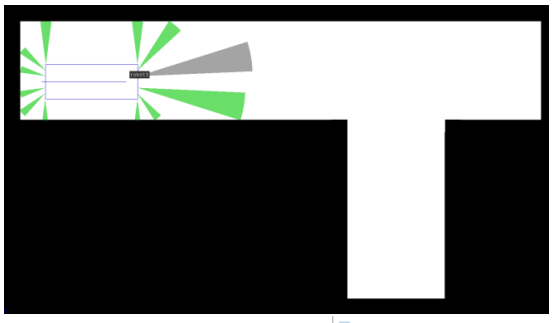


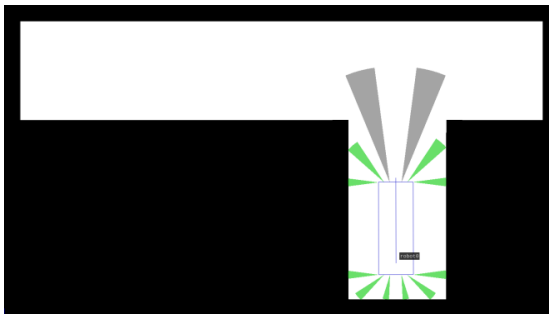


```
parking_time =
    6.5000
parking_position_x =
   17.1333
parking_position_y =
    2.0671
parking_position_theta =
    1.5634
         0
         0
```

4.5.2. Entorno Definitivo

El entorno definitivo es el que comienza en la posición (5.3, 10.1, 3.14159). La posición inicial, final y evaluación de la maniobra son las siguientes:





```
parking_time =
    6.5000
parking_position_x =
   16.9528
parking_position_y =
    2.2757
parking_position_theta =
    1.5727
         0
         0
```

5. Controlador Neuronal

5.1. Introducción

Una vez hemos diseñado el controlador borroso, es hora de pasar a desarrollar el controlador neuronal.

Las redes neuronales son un componente central en el campo de la inteligencia artificial y el aprendizaje profundo. Están inspiradas en el funcionamiento del cerebro humano y se utilizan para realizar tareas complejas de procesamiento de información. Estas redes consisten en capas de unidades llamadas neuronas, que están interconectadas a través de conexiones ponderadas.

Las redes neuronales se pueden clasificar en diferentes tipos, siendo las redes neuronales artificiales y las redes neuronales profundas (también conocidas como redes neuronales profundas) las más comunes. En las redes neuronales artificiales, las neuronas están organizadas en capas de entrada, capas ocultas y capas de salida. La información fluye a través de la red desde la capa de entrada hasta la capa de salida durante el proceso de aprendizaje.

Por otro lado, las redes neuronales profundas son una forma más avanzada que incluye múltiples capas ocultas, permitiendo la extracción de características más complejas y abstractas. Estas redes han demostrado ser particularmente efectivas en tareas como el reconocimiento de imágenes, procesamiento de lenguaje natural y juegos.

El aprendizaje en las redes neuronales se logra ajustando los pesos de las conexiones entre las neuronas. Esto se realiza mediante algoritmos de optimización que minimizan la diferencia entre las predicciones de la red y las salidas deseadas, proceso conocido como entrenamiento.

5.2. Detalles sobre la red neuronal

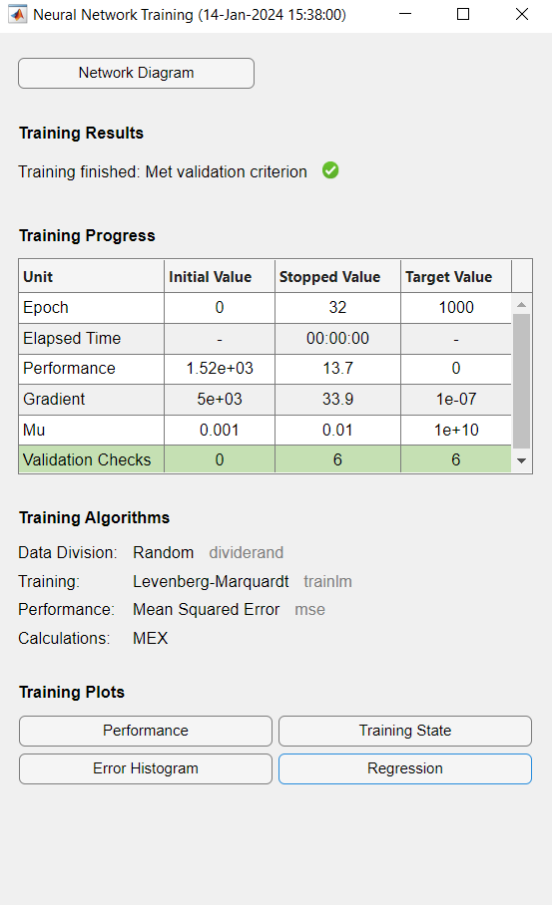
Nuestra red neuronal utiliza el algoritmo de entrenamiento de Levenberg-Marquadt, el cual es el algoritmo de aprendizaje más rápidos para redes neuronales con un número de parámetros moderado. La red es de tipo "Feed-Forward", es decir, no tiene ningún tipo de retroalimentación y siempre van desde la capa de entrada hasta la de salida. Posee dos capas ocultas, una con 8 y otra con 4 neuronas, con las que tras varias pruebas han dado los mejores resultados posibles. También hemos dividido los parámetros de entrenamiento, validación y test de la siguiente manera respectivamente: [70 15 15].

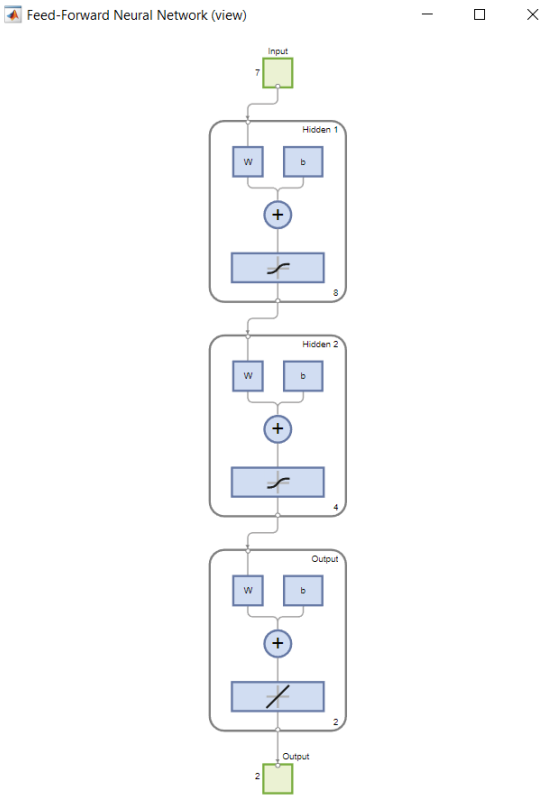
5.3. Datos de entrenamiento

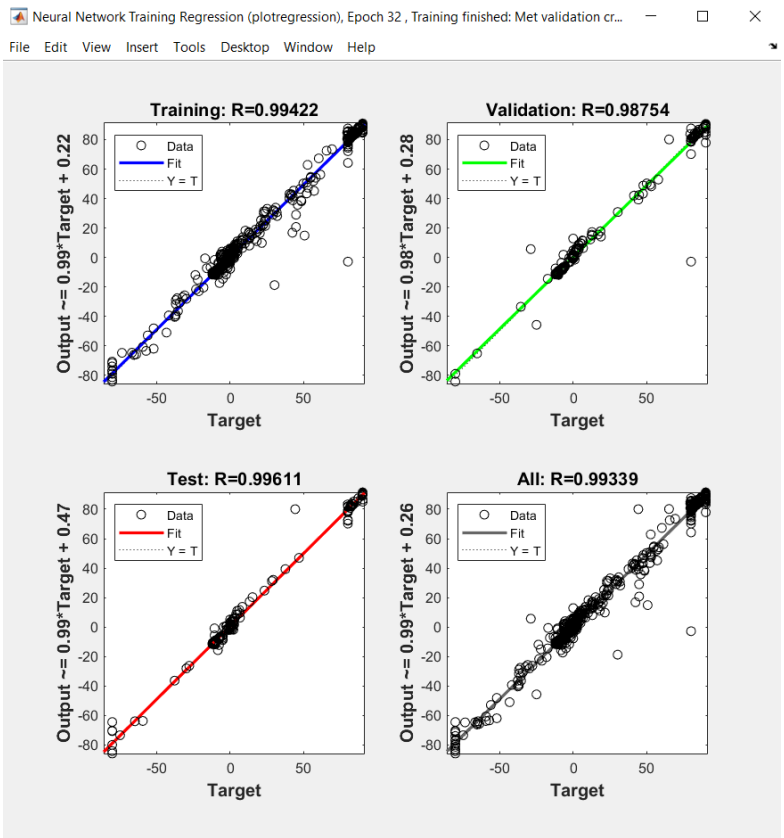
Para entrenar el controlador neuronal hemos decidido obtener los datos de entrenamiento a través de una versión antigua del controlador borroso diseñado en la parte 1. Esta versión funcionaba correctamente, pero era más lenta que la versión final. En cuanto al número de datos de entrenamiento, hemos realizado 15 ejecuciones de las cuales se han obtenido los datos de entrenamiento. Estas ejecuciones se han realizado variando las posiciones iniciales entre 4.5 y 5.5 en el eje de las X y entre 10 y 11 en el eje de las Y.

```
N_DATASETS = 15;
dataset = []
for i = 1:N_DATASETS
    datos = csvread("datos_neuronal/ej"+i+".txt");
    % Elimina las filas con al menos un 0 en las primeras 5 columnas
    filas_con_cero = any(datos(:,1:7) == 0, 2);
    datos(filas_con_cero, :) = [];
    dataset = [dataset ; datos];
end
```

Una vez hemos obtenido nuestros datos de entrenamiento, pasamos a entrenar a nuestra red y obtenemos los siguientes resultados:


Neural Network Training (14-Jan-2024 15:38:00)
Network Diagram
Training Results
Training finished: Met validation criterion
Training Progress
Unit | Initial Value | Stopped Value | Target Value
Epoch | 0 | 32 | 1000
Elapsed Time | - | 00:00:00 | -
Performance | 1.52e+03 | 13.7 | 0
Gradient | 5e+03 | 33.9 | 1e-07
Mu | 0.001 | 0.01 | 1e+10
Validation Checks | 0 | 6 | 6
Training Algorithms
Data Division: Random dividerand
Training: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX
Training Plots
Performance
Training State
Error Histogram
Regression

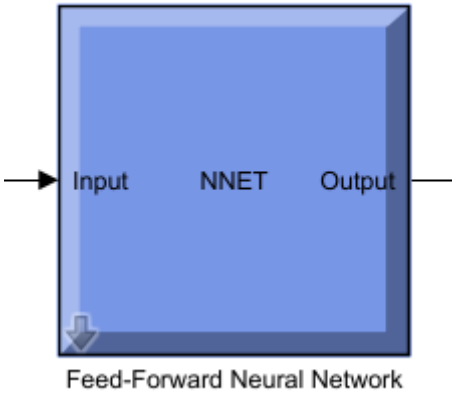

Feed-Forward Neural Network (view)
Input
Hidden 1
Hidden 2
Output
Output


Neural Network Training Regression (plotregression), Epoch 32 , Training finished: Met validation cr...
File Edit View Insert Tools Desktop Window Help
Training: R=0.99422
Validation: R=0.98754
Test: R=0.99611
All: R=0.99339
Data
Fit
Y = T
Output ~= 0.99*Target + 0.22
Output ~= 0.98*Target + 0.28
Output ~= 0.99*Target + 0.47
Output ~= 0.99*Target + 0.26
Target

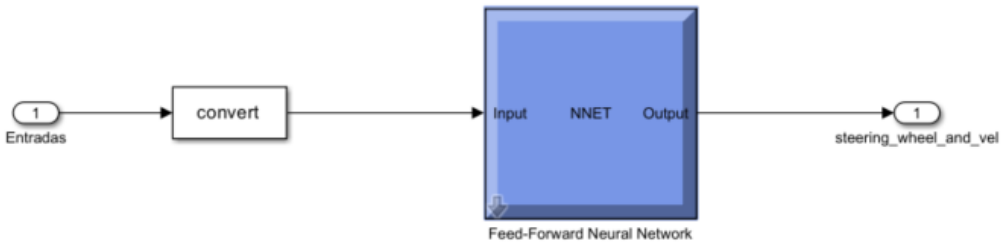
Como se puede comprobar, el valor de “performance” es de 13.7 y los valores de entrenamiento, validación y test están todos muy próximos a la recta de regresión por lo que podemos decir que este modelo es bastante bueno.

5.4. Ejecución

Una vez ejecutamos la función “gensim”, se nos generará una red neuronal basada en los resultados del entrenamiento anterior:



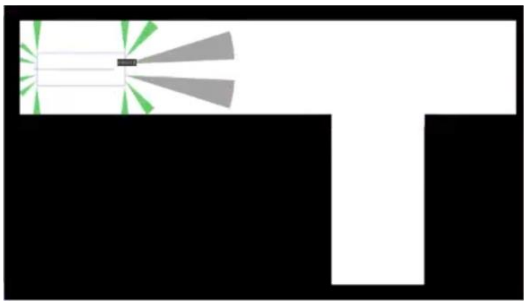
Tras haber obtenido la red neuronal, deberemos añadirla a nuestro esquema de Simulink, además de añadir el bloque de “Data Type Conversion”:

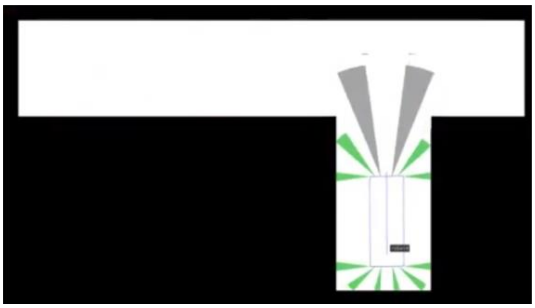


Tras todo esto ejecutamos para poder ver el resultado final y ver como aparca el coche.

5.4.1. Entorno Inicial

El entorno inicial es el que comienza en la posición (5, 10.5, 3.14159). La posición inicial, final y evaluación de la maniobra son las siguientes:



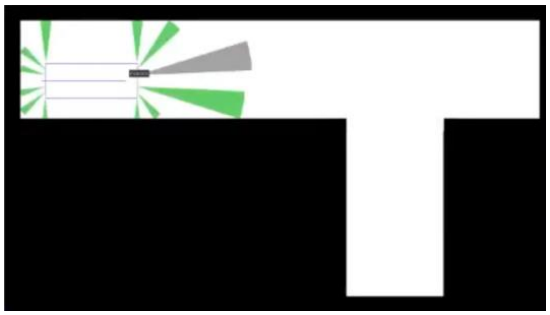


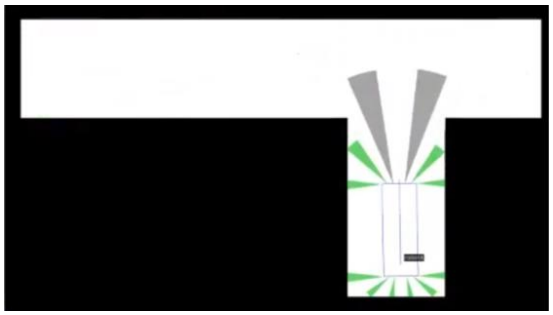
```
parking_time =
     8
parking_position_x =
   17.1600
parking_position_y =
    2.2921
parking_position_theta =
    1.5776
         0
         0
```

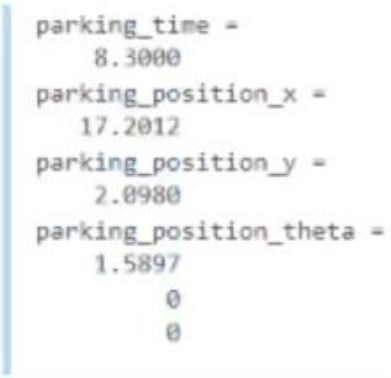
Cabe recalcar que habrá cierto tiempo desde que se ejecutará el programa hasta que arranque el robot en el cual algunos sensores tendrán datos residuales representados con 0 por lo que el tiempo real en aparcar será algo menor. Todo el funcionamiento se puede ver en el vídeo adjuntado donde se muestra el funcionamiento completo.

5.4.2. Entorno definitivo

El entorno definitivo es el que comienza en la posición (5.3, 10.1, 3.14159). La posición inicial, final y evaluación de la maniobra son las siguientes:







Cabe recalcar que habrá cierto tiempo desde que se ejecutará el programa hasta que arranque el robot en el cual algunos sensores tendrán datos residuales representados con 0 por lo que el tiempo real en aparcar será algo menor. Todo el funcionamiento se puede ver en el vídeo adjuntado donde se muestra el funcionamiento completo.

6. Otras Ejecuciones

Además de las ejecuciones que se nos pide para la práctica, hemos realizado otras ejecuciones desde otras posiciones iniciales en las cuales se puede comprobar que aparca de igual manera recto y centrado. Esto puede comprobarse en uno de los vídeos adjuntados.

7. Estructura de la entrega

La carpeta de entrega consta de la siguiente estructura:

- Entrega
    - Controlador Borroso
        - ackerman_ROS_fuzzy_controller.slx →Archivo Simulink a ejecutar.
        - ackerman_ROS_fuzzy_controller.slxc → Caché del archivo Simulink a ejecutar.
        - parking_bateria.fis → Controlador Borroso.
        - video_parking_borroso.mp4 → Vídeo del aparcamiento borroso.
    - Controlador Neuronal
        - datos_entrenamiento → Carpeta con todos los datos de entrenamiento utilizados

        - ackerman_ROS_neural_controller.slx →Archivo Simulink a ejecutar.
        - ackerman_ROS_neural_controller.slxc → Caché del archivo Simulink a ejecutar.
        - entrenamiento.m → Script para entrenar la red neuronal.
        - video_parking_neuronal.mp4 → Vídeo del aparcamiento neuronal.
    - Punto Extra
        - video_parking_punto_extra.mp4 → Vídeo con las ejecuciones para la competición del punto extra.
    - Memoria