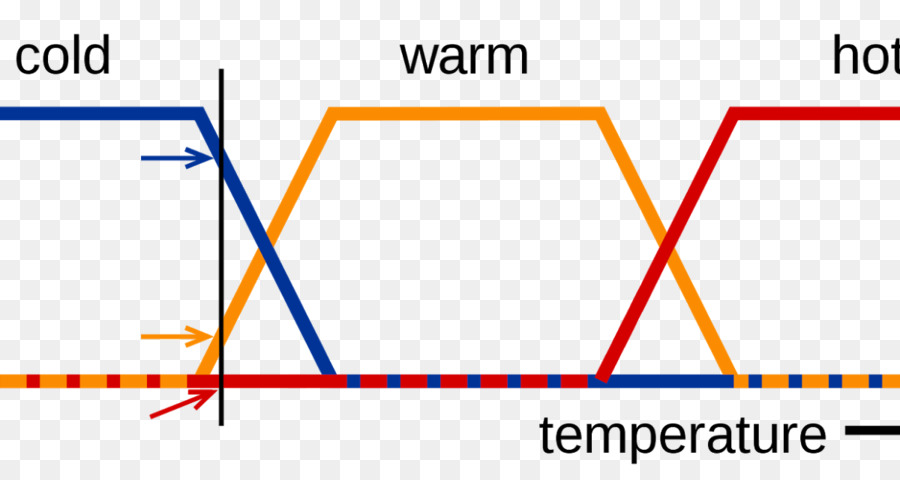
Práctica 2: Control borroso



**SISTEMAS DE CONTROL INTELIGENTE**

**01/12/2019**

**Autores:** Álvaro de las Heras Fernández

Álvaro Maestre Santa

## Índice

[Parte 1. Diseño de un control borroso de posición para un robot móvil 3](#_Toc26007105)

[Parte 2. Diseño de control borroso de posición con evitación de obstáculos 8](#_Toc26007106)

## 

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

El objetivo de esta parte es crear un controlador borroso que permita alcanzar puntos indicados, además de seguir trayectorias. Para su realización hemos seguido los pasos indicados en el enunciado realizando las modificaciones pertinentes para asegurar el correcto funcionamiento del robot en los distintos casos.

1. **Crear controlador borroso con fuzzy.**

Para la velocidad hemos definido la siguiente tabla de reglas para Mandani:

|  |  |  |  |
| --- | --- | --- | --- |
| E\_d  E\_th | P | M | G |
| N | P | M | G |
| C | P | M | G |
| P | P | M | G |

Tabla 1. Reglas para la velocidad lineal.

Básicamente se extrapola que si se encuentra cerca del punto de destino debe de ir más despacio, porque se requiere de una mayor precisión comparado si está en el punto inicial. Al tener todas las columnas un valor se puede ver que únicamente dependen del error a la distancia, E\_d. Esto nos simplifica su escritura en el controlador Mandani como tan solo 3 reglas, en vez de las 9 que se podrían haber tenido.

|  |  |  |  |
| --- | --- | --- | --- |
| E\_d  E\_th | P | M | G |
| N | N | N | N |
| C | C | C | C |
| P | P | P | P |

Tabla 2. Reglas para la velocidad angular.

Este caso es similar, lo único que hay que tener en cuenta que la w se encarga de los giros al ser la velocidad angular. En este caso, sólo depende del error del ángulo en radianes entre el robot y la referencia. Con este dato sabremos si el robot va bien encaminado al destino o se desvía de la trayectoria, esto último implica que se tenga que corregir la trayectoria. Para la corrección de la trayectoria tendremos una w positiva o negativa en el intervalo [-1 1] medido en radianes. Esta velocidad angular será positiva si el E\_theta es positivo porque eso implica que se esta desviando hacia la derecha corrigiendo así hacia la izquierda, y viceversa.

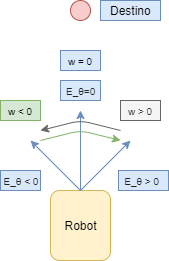


Fig. 1. Diagrama de funcionamiento de la corrección.

Una vez obtenidas las reglas las hemos introducido dentro del controlador, obteniendo el siguiente resultado:

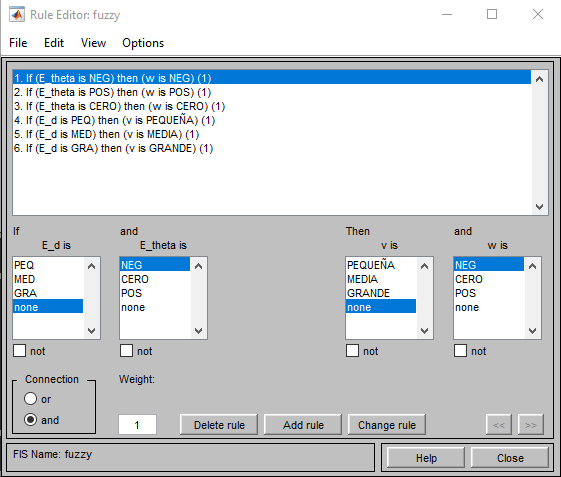


Fig. 2. Reglas del controlador borroso tipo Mandani.

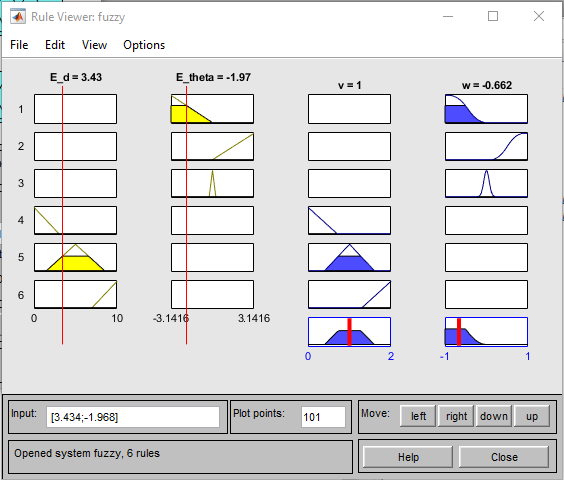


Fig. 3. Vista de las reglas con antecedentes y consecuentes.

Los valores de las funciones de antecedente y consecuente del controlador borroso los hemos puesto conforme a una distribución lógica y simétrica al principio, porque después hemos necesitado calibrarlos visualizando las reglas, además de probándolo con el robot. Como velocidad angular hemos elegido funciones sigmoidales y gaussianas, puesto que hacen que los cambios sean más rápidos lo que hace que gire en menor tiempo. De las funciones cabe destacar que las encargadas de la velocidad angular y theta, tienen picos muy altos en CERO, esto se debe a que nos interesa encontrar la dirección adecuada y por eso cuando tiene una buena está es seguida rápidamente, debido al rápido crecimiento de la función.

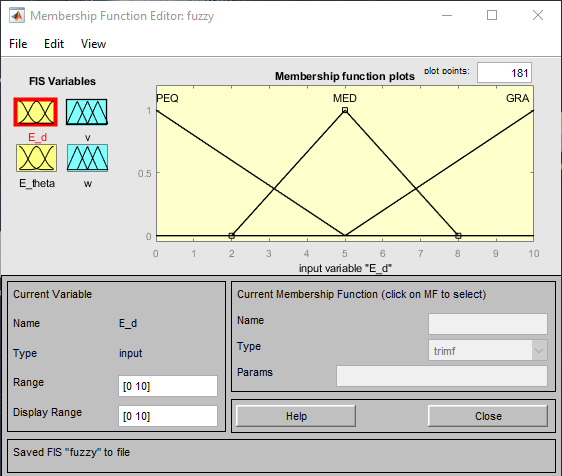
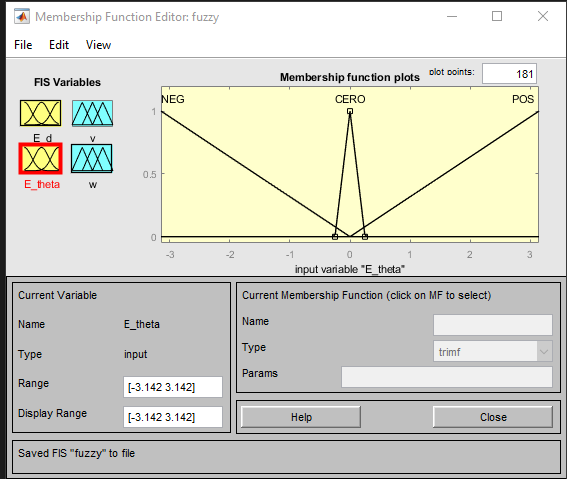
 

Fig. 4. Antecedentes, izquierda error de distancia y derecha error theta.

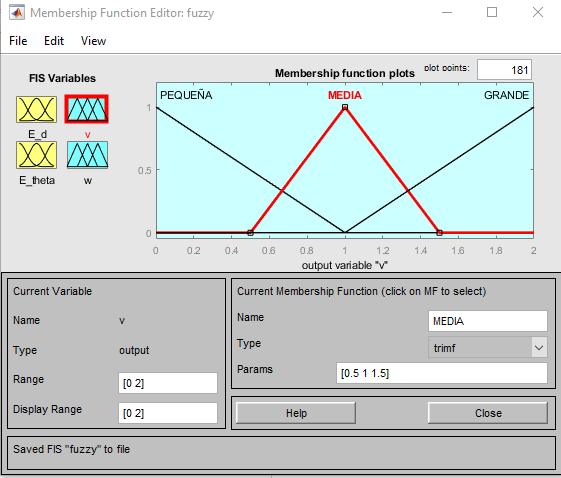
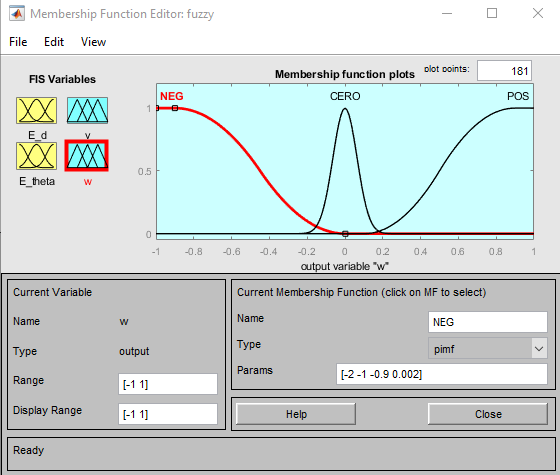
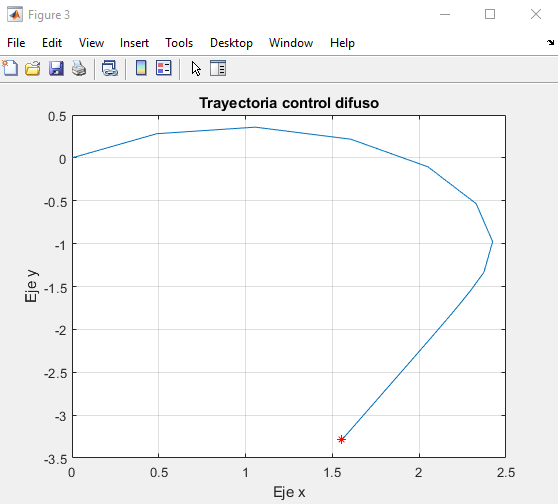
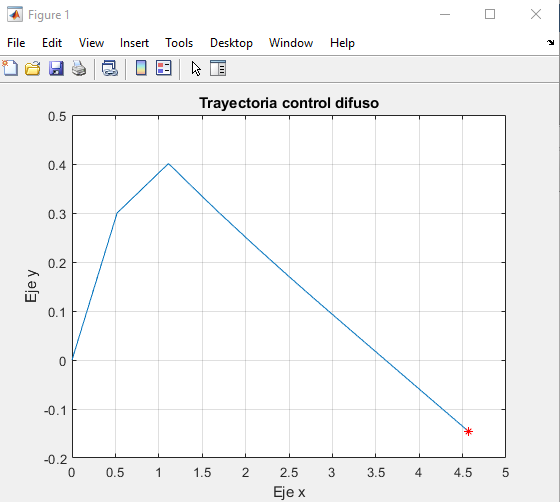
 

Fig. 5. Consecuentes, izquierda velocidad lineal y derecha velocidad angular.

1. **Crear el esquema con el controlador en Simulink.**

El proceso ha sido el mismo que el indicado en el enunciado, añadiendo el módulo, indicando el archivo .fis y conectando los multiplexores y demultiplexores. Finalmente hemos puesto este bloque en el robot.

1. **Probar el controlador con múltiples posiciones.**

Para probar el código con múltiples posiciones hemos desarrollado un script que genera múltiples figuras y señala los puntos. Como se puede observar en las capturas los resultados son bastante buenos y precisos, alcanzando las referencias generadas. ** **

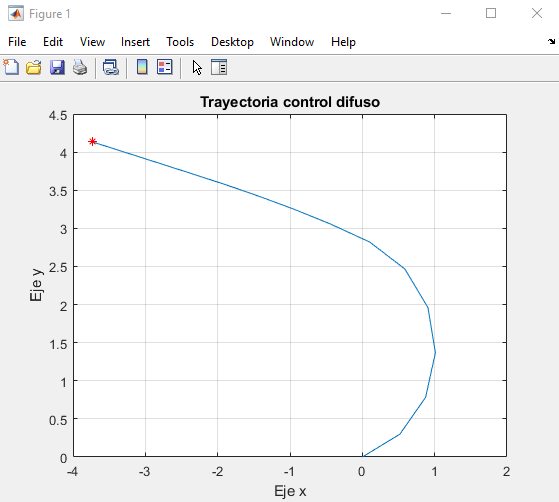
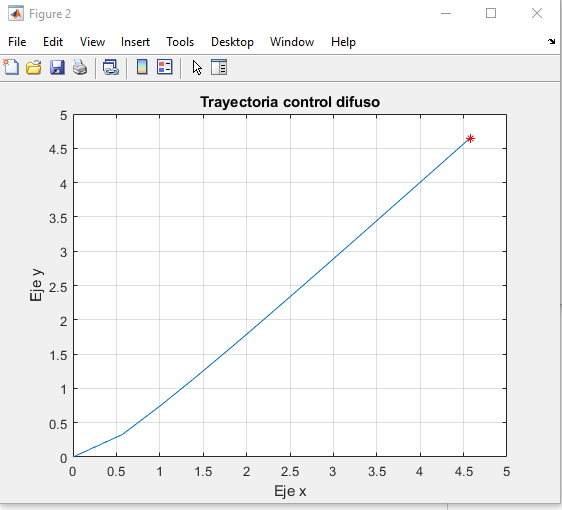
**** ****

Fig. 6. Prueba con diversos puntos generados aleatoriamente.

1. **Aplicar el módulo de trayectorias y comprobar resultado.**

Al igual que en la anterior práctica añadiremos el módulo generador de trayectorias, que genera una especie de 8 desde las coordenadas iniciales que se le indiquen. Como ya teníamos el controlador borroso anterior lo aplicamos directamente, pero causo distintos problemas. El primer problema hacía que se parará la simulación antes de haber dibujado la forma completa, esto se debía a que en alguna ocasión llegaba al punto de referencia de la trayectoria, haciendo que se detuviera por la condición de parada que tiene, por lo que la modificamos para que no se detuviera. Después la trayectoria que dibujaba era sinuosa pero seguía la forma a alcanzar, por lo que el problema estaba con los antecedentes y consecuentes del modelo, no con las reglas. Entonces mediante prueba y error dimos con la configuración adecuada que permitía que funcionase correctamente, tanto para la trayectoria como para ir a puntos aleatorios. Estas pruebas nos hicieron ver que el problema estaba con la entrada del error de distancia y la salida de velocidad lineal, teniendo así que afinarlas.

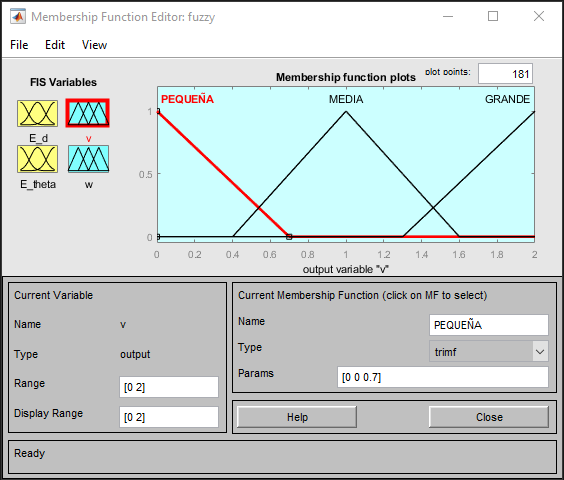
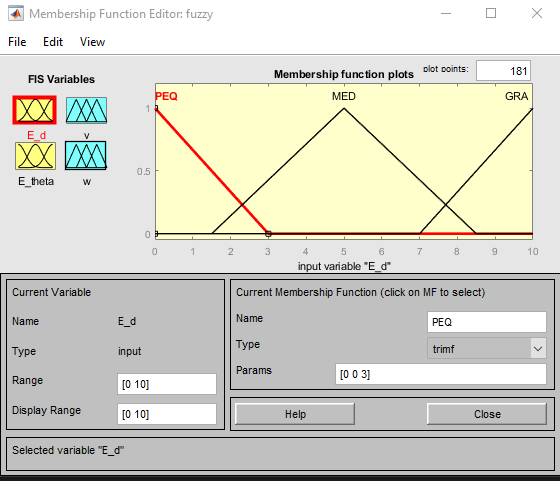
 

Fig. 7. Cambios en los antecedentes y consecuentes.

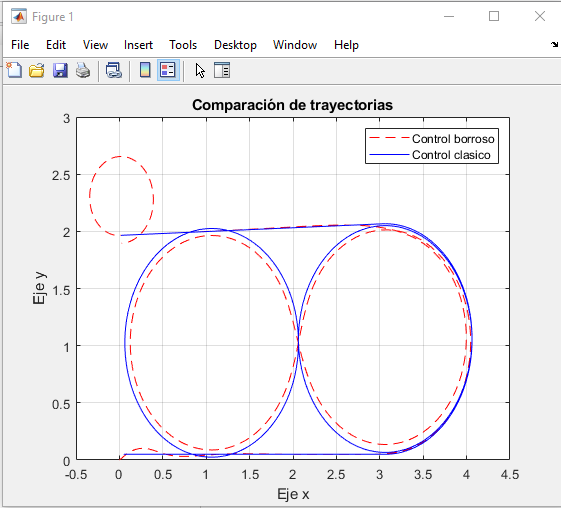


Fig. 8. Comparación entre ambas trayectorias

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

El objetivo de esta parte es modificar el anterior controlador borroso para que permita alcanzar puntos indicados a la vez que esquivar un obstáculo. Para la realización de este apartado, hemos seguido los pasos indicados en el enunciado, realizando las modificaciones oportunas para asegurar el perfecto funcionamiento del robot en los distintos casos.

1. **Crear el controlador borroso.**

En este apartado hemos seguido las indicaciones de la práctica, creando un controlador borroso con cuatro entradas y dos salidas, como se muestra en la siguiente captura. Además hemos reutilizado el controlador del apartado anterior, para así ahorrarnos el afinamiento de uno nuevo, en varios antecedentes y consecuentes.

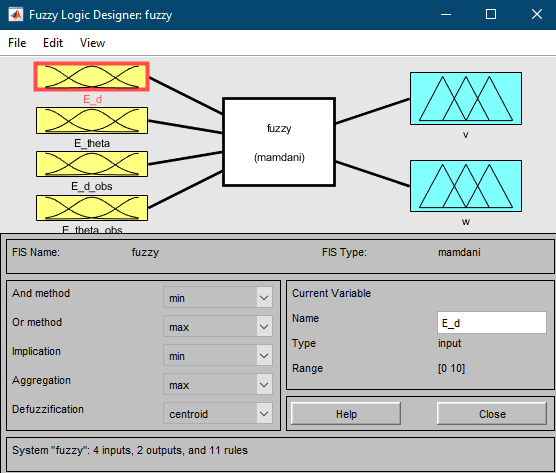
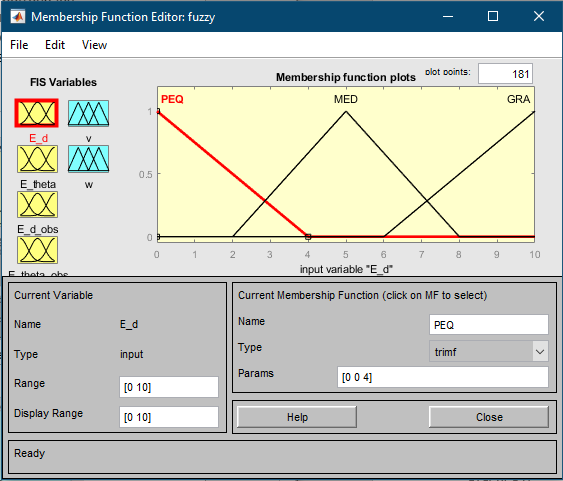
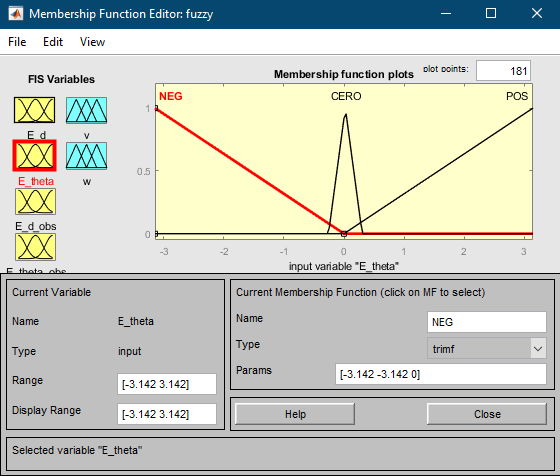


Fig. 9. Diseño del control borroso mediante la toolbox de Matlab

1. **Modificación funciones de entrada y salidas.**

El objetivo de este apartado es modificar las entradas y salidas para que se adecue correctamente nuestro controlador al nuevo problema, el cual, tendrá que resolver. Las modificaciones se pueden apreciar en la siguiente imagen. Muchas se mantienen igual del apartado anterior, excepto las dos nuevas



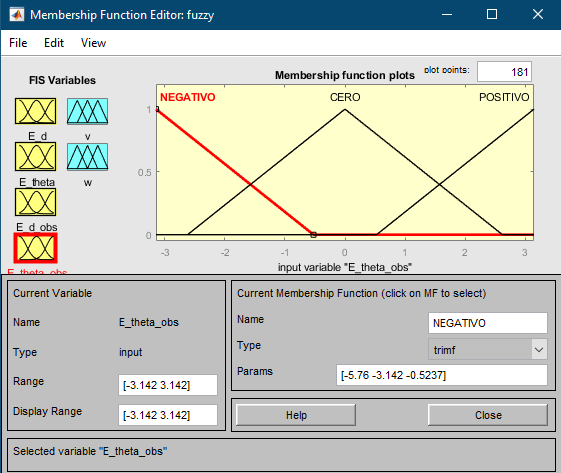
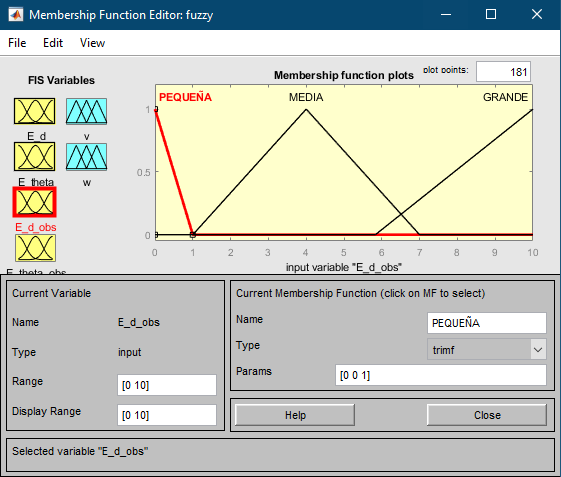


Fig. 10. Entradas: error de distancia, error theta, error distancia obstáculo,

error theta obstáculo.

Estas funciones son el error de distancia del obstáculo y error theta del obstáculo. En un principio, estas gráficas las hemos puesto conforme a una distribución lógica y simétrica, pero después hemos ido calibrándolos, en base a las reglas, además de probándolo en el robot. La primera es el E\_d\_obs, que se caracteriza porque tiene la función pequeña en un intervalo muy corto en comparación a las otras, esto se debe a que se necesita bastante precisión para saber cuándo se encuentra cerca del obstáculo el robot, y así evitarlo. La segunda es similar al resto, sin tener ninguna característica en especial.

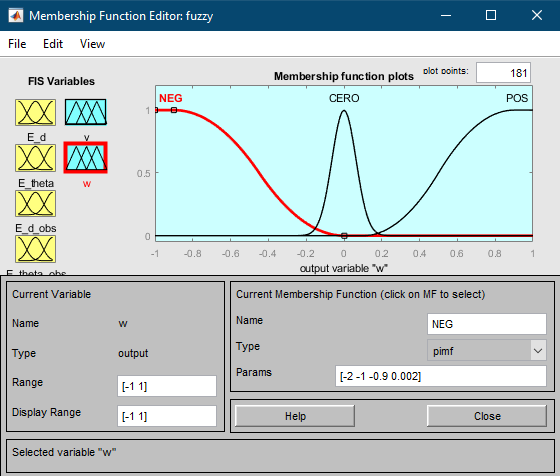
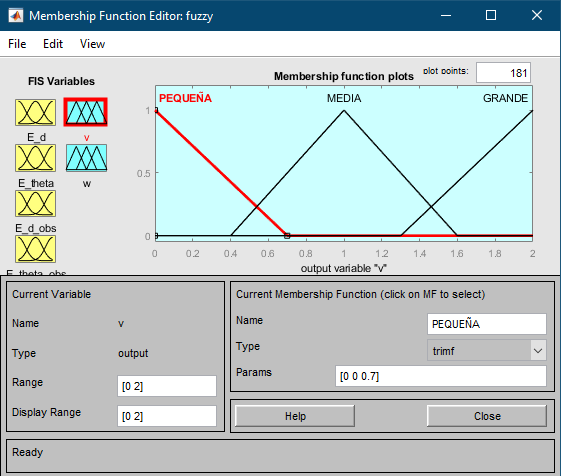
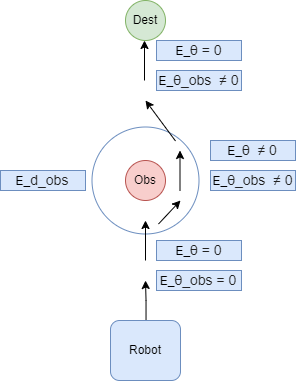


Fig. 11. Salidas: velocidad lineal y velocidad angular.

Como podemos observar, en las salidas, no hemos modificado nada. En esta parte hemos seguido manteniendo las funciones sigmoidales y gaussianas, ya que el robot respondía perfectamente.

1. **Tabla de reglas.**

En esta parte, vamos a poner las tablas de reglas de las nuevas entradas que hemos introducido. Para facilitarnos el trabajo es modelado el problema para ver como varía en función del obstáculo.



Las reglas que hemos deducido a partir de esta modelización son:

|  |  |  |  |
| --- | --- | --- | --- |
| E\_d\_obs  E\_th\_obs | P | M | G |
| N | P | M | G |
| C | P | M | G |
| P | P | M | G |

Fig. 12. Reglas para la velocidad lineal.

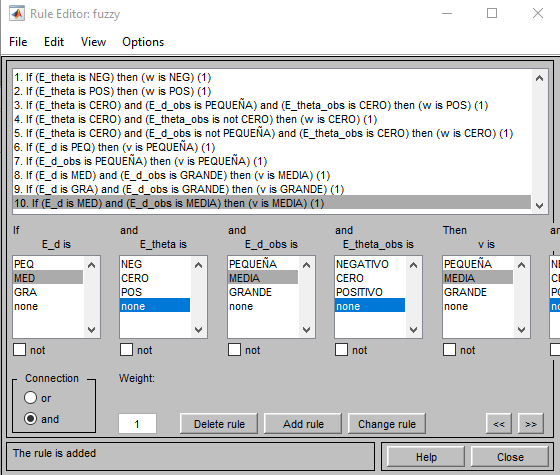
Básicamente, podemos verlo como si se encuentra cerca del obstáculo debe de ir más despacio, ya que requiere de una mayor precisión comparado si está muy lejos del obstáculo. Por lo tanto, podemos ver que esta tabla depende del error a la distancia del obstáculo, E\_d\_obs, generando así sólo 3 reglas.

|  |  |  |  |
| --- | --- | --- | --- |
| E\_d\_obs  E\_th\_obs | P | M | G |
| N | C | C | C |
| C | P | C | C |
| P | C | C | C |

Fig. 13. Reglas para la velocidad angular.

Este caso, es un poco más difícil de entender, ya que hay que tener en cuenta que la velocidad angular, es la que se encarga de los giros, por lo tanto, lo podemos simplificar, a que, si está cerca del obstáculo, que empiece a girar, en este caso hemos puesto que gire positivamente, pero también podríamos haber puesto que gire de forma negativa o que girará a una distancia media. Posteriormente, la corrección se encargarán las tablas ya explicadas en la parte 1 de esta práctica. Sin embargo, las reglas anteriores no valen tal y como estaban y necesitaban de una serie de modificaciones para que esquivará correctamente, en el caso de que E\_theta fuera 0. Por lo que teniendo en cuenta los errores theta de ambas funciones, añadimos un par de reglas para que fuera lo más recto posible hasta llegar al obstáculo.

Con todos estos datos y reglas las implementamos quedando el siguiente resultado.



Aunque para la segunda parte la modificamos reduciendo las reglas a las siguientes, los cambios que se pueden ver es el control de velocidades, que daba problemas con respecto al caso anterior.

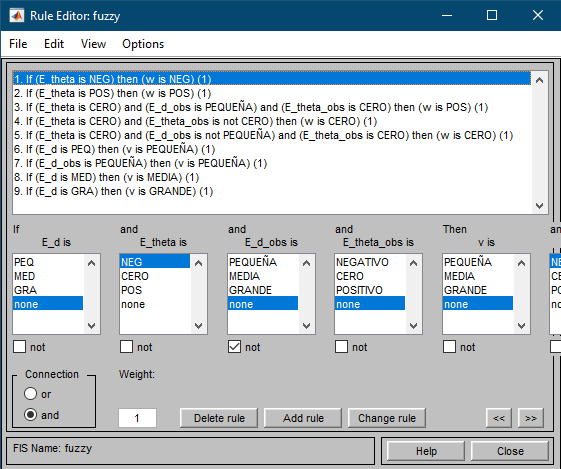


Fig. 14. Reglas del controlador borroso tipo Mandani.

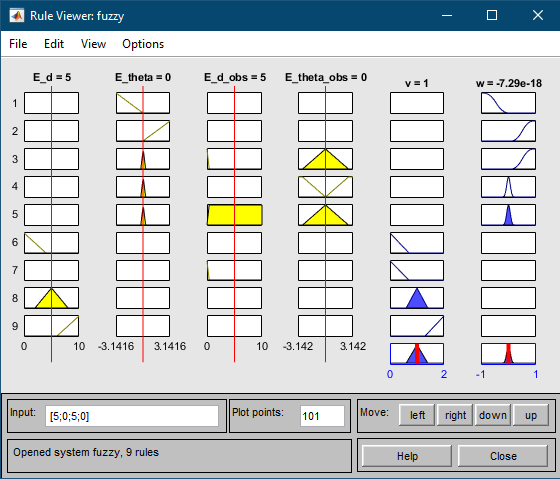
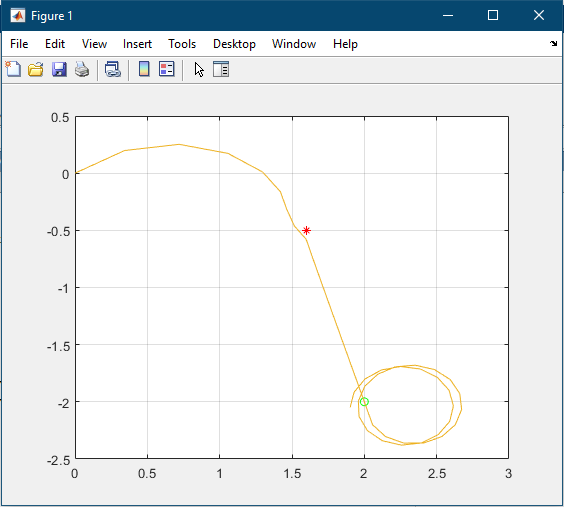
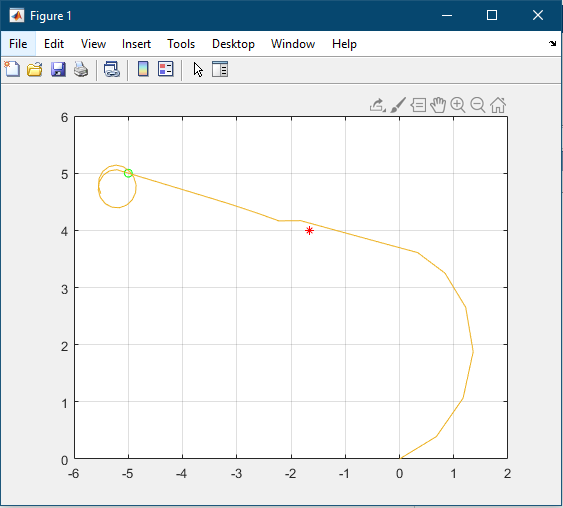
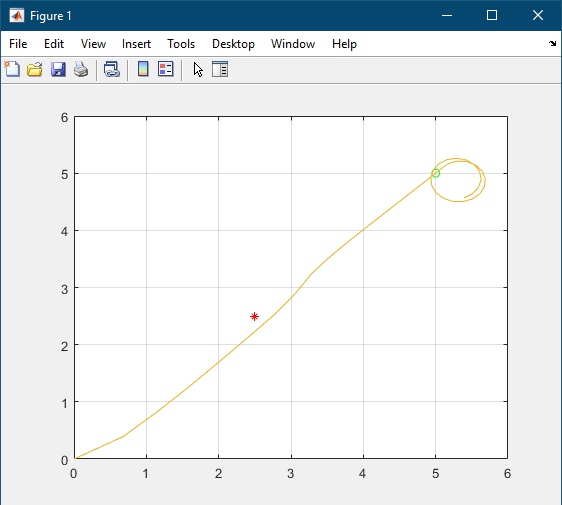
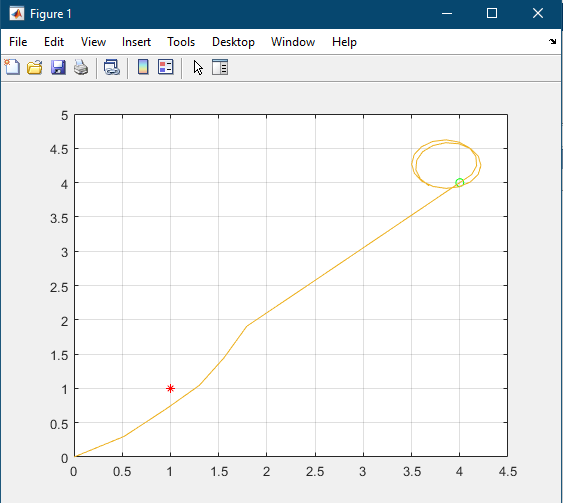


Fig. 15. Vista de las reglas con antecedentes y consecuentes.

1. **Comprobación del robot.**

En esta parte, vamos a comprobar que le robot se comporta de la forma esperada, viendo cómo esquiva el obstáculo. Antes de ver las capturas con los diferentes comportamientos del robot, haré una breve explicación de lo que vais a ver en la gráfica. Lo primero que vemos será una línea y dos puntos, la línea indica la trayectoria del robot, el punto rojo indica el obstáculo y el punto verde, la meta. Se puede observar como corrige cerca del obstáculo en los casos en los que se interpone y como no lo hace cuando no esta en su ruta.



Fig. 16. Prueba con diversos puntos.

1. **Aplicar el módulo de trayectorias y comprobar resultado.**

Al igual que en la anterior parte de esta práctica, añadiremos el módulo generador de trayectorias, que generará los dos círculos desde las coordenadas iniciales que se le indiquen. Pero a diferencia de la anterior parte, deberá de esquivar un obstáculo a demás de segur el ocho volteado. Esto lo conseguimos mediante prueba y error a la hora de configurar el controlador para que hiciera la trayectoria perfecta además de esquivar el obstáculo. Con esto nos dimos cuenta, de que el error lo tuvimos en las precisión de las entradas y quitando las reglas de velocidad, que distorsionaban el dibujo. El resultado obtenido es el siguiente:

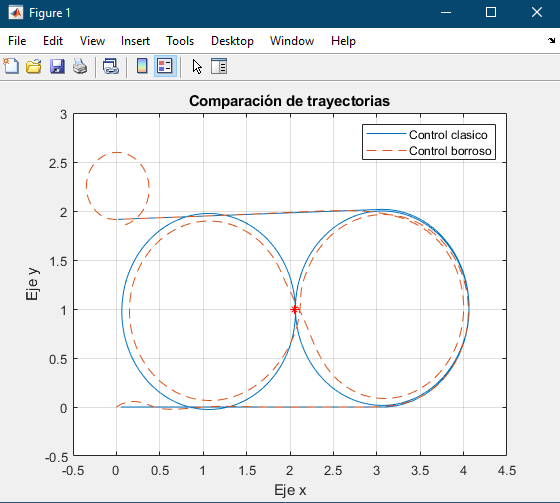


Fig. 17. Comparaciones entre ambas trayectorias evitando el obstáculo.

Haciendo zoom, se puede ver mejor como esquiva el obstaculo:

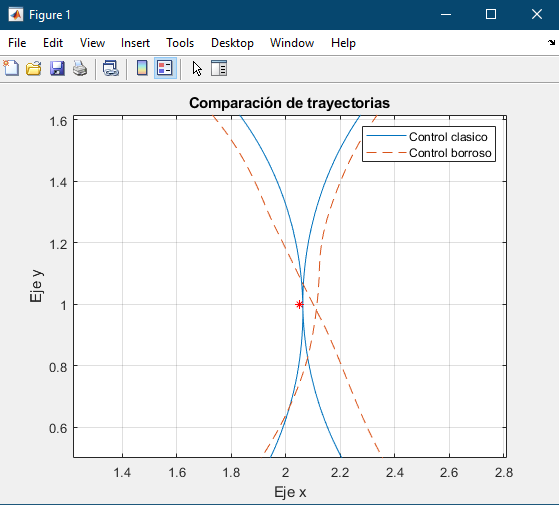


Fig. 18. Zoom gráfico de comparación entre ambas trayectorias.

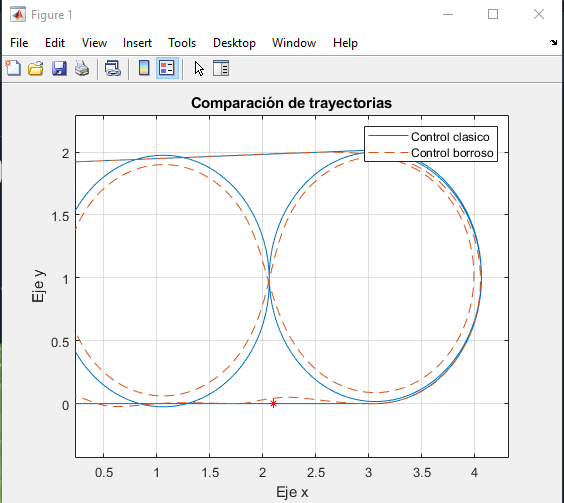


Fig. 19. Otro ejemplo más con el obstáculo en otra posición.