**PARTE 1 DE LA PRACTICA**

**PARTE 1.1: LLEGAR A UN PUNTO**

En este apartado se plantea el control de posición de un robot móvil dentro de un entorno sin obstáculos de dimensiones 10x10 metros y cuyo origen de coordenadas se encuentra en el centro geométrico del entorno.

El objetivo de esta parte de la práctica es **diseñar un controlador** **borroso** de manera que el robot sea capaz de alcanzar una posición determinada por las referencias de posición refx y refy.

El esquema de control es el mismo que hemos utilizado en anteriores prácticas, con un bloque Position\_errors, un bloque de Control y un bloque Robot. Sin embargo, a diferencia de otras prácticas, el bloque de control lo generaremos utilizando un controlador borroso tal y como hemos aprendido en clase.

Para generar este controlador utilizaremos el comando de Matlab **“fuzzy”**, el cual nos proporciona una interfaz desde la cual podemos configurar nuestro controlador borroso.

Para este controlador mantendremos las operaciones AND/OR y el desborrosificador por defecto. A continuación, añadiremos a nuestro controlador **2 entradas** (E\_d y E\_theta) **y 2 salidas** (V y W), cada una de ellas con distintos rangos:

𝐸𝑑 ∈ [0, 10], 𝐸𝜃 ∈ [−𝜋, 𝜋], 𝑉 ∈ [0, 2], 𝑊 ∈ [−1, 1]

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Figura 1**: Esquema de nuestro controlador

Ahora modificaremos las **funciones de pertenencia** de cada una de las variables como se puede ver en la Figura 2:

Imagen que contiene mapa, texto

Descripción generada automáticamenteImagen que contiene captura de pantalla

Descripción generada automáticamente

Imagen que contiene captura de pantalla

Descripción generada automáticamenteImagen que contiene captura de pantalla

Descripción generada automáticamente

**Figura 2**: Funciones de pertenencia

Para determinar estas funciones de pertenencia hemos realizado diferentes pruebas hasta lograr un recorrido perfecto por parte del robot hasta el objetivo. Inicialmente las pruebas se realizaron controlando solo el giro y manteniendo una **velocidad constante** con valor **V=0.3**. Una vez logrado alcanzar el objetivo correctamente, añadimos las funciones de pertenencia para las variables de velocidad lineal (E\_d y V).

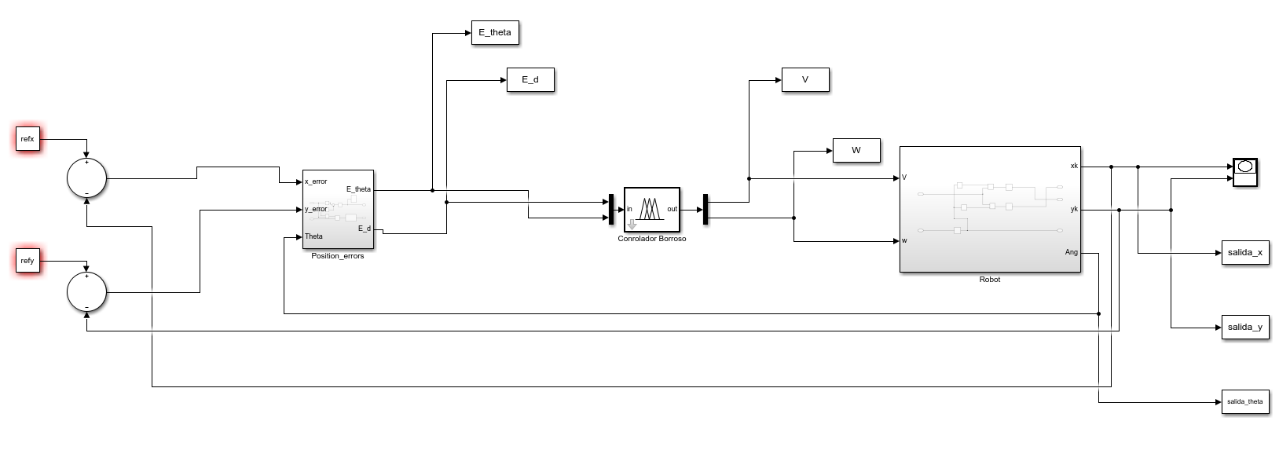
Una vez hemos definido correctamente las funciones de pertenencia para cada uno de los conjuntos es momento de **generar las reglas** que seguirá nuestro controlador. Utilizaremos estas reglas para **determinar el valor de nuestras variables de salida** en función de los valores de entrada:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Figura 3**: Reglas de nuestro controlador borroso

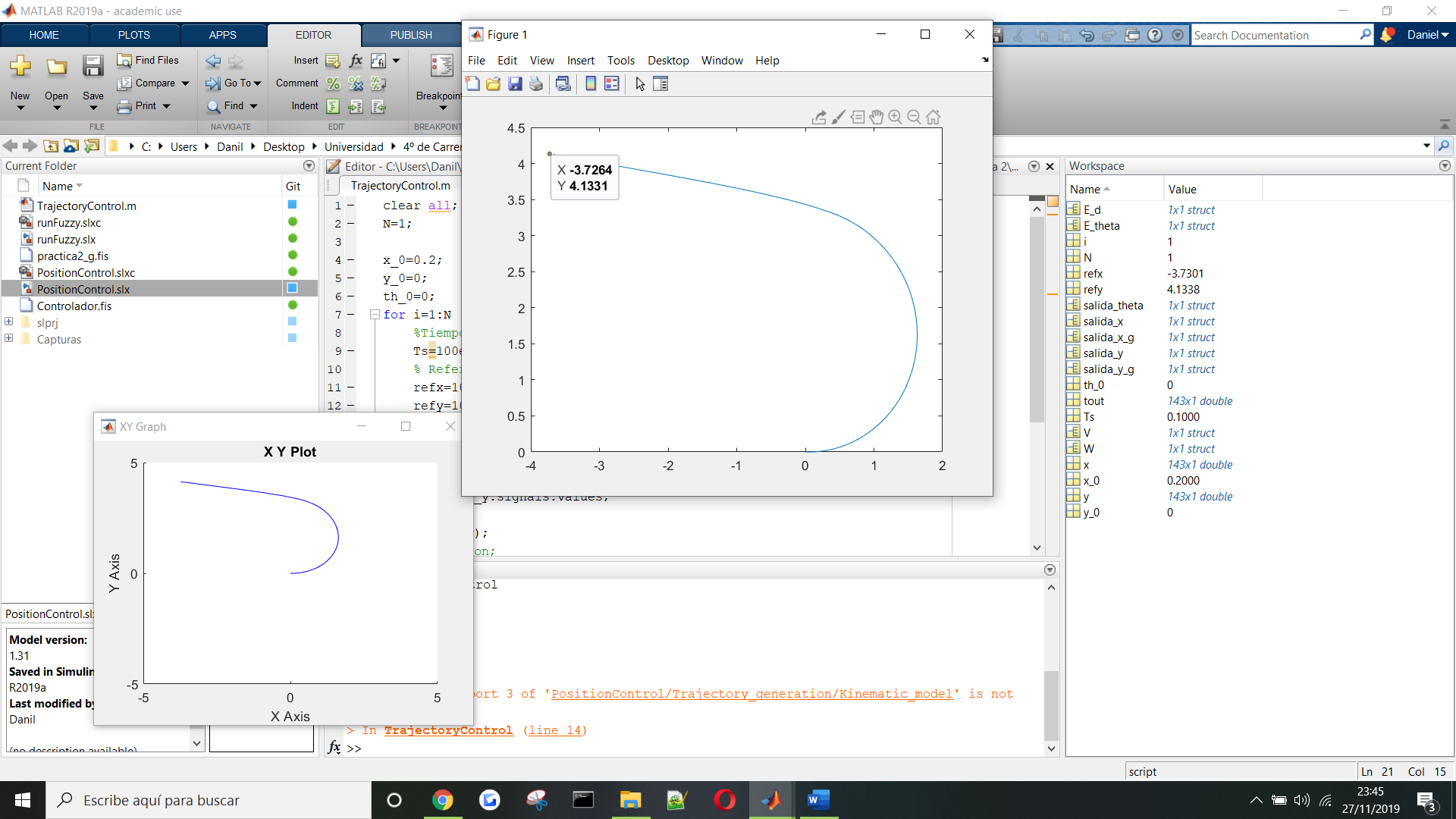
Por último, generaremos un bloque de *Simulink* con el nombre de **“Controlador Borroso”** que introduciremos en nuestro esquema de control mencionado anteriormente sustituyéndolo por el bloque Control tal y como se muestra en la Figura 4:



**Figura 4:** Esquema de control con el bloque Controlador Borroso

Ejecutaremos el código de prueba de la simulación y comprobaremos si nuestro Controlador Borroso es capaz de llevar el robot hasta el punto de destino indicado en las variables refx y refy:





**Figura 5:** Trayectoria resultado de la simulación

Como podemos observar en la Figura 5, los valores objetivo a alcanzar por el robot son **refx=-3.7301** y **refy= 4.1338,** y nuestro Controlador Borroso ha sido capaz de alcanzarlo a la perfección realizando un giro desde la posición inicial.

**PARTE 1.2: COPIAR LA TRAYECTORIA DEL TRAJECTORY\_GENERATION**

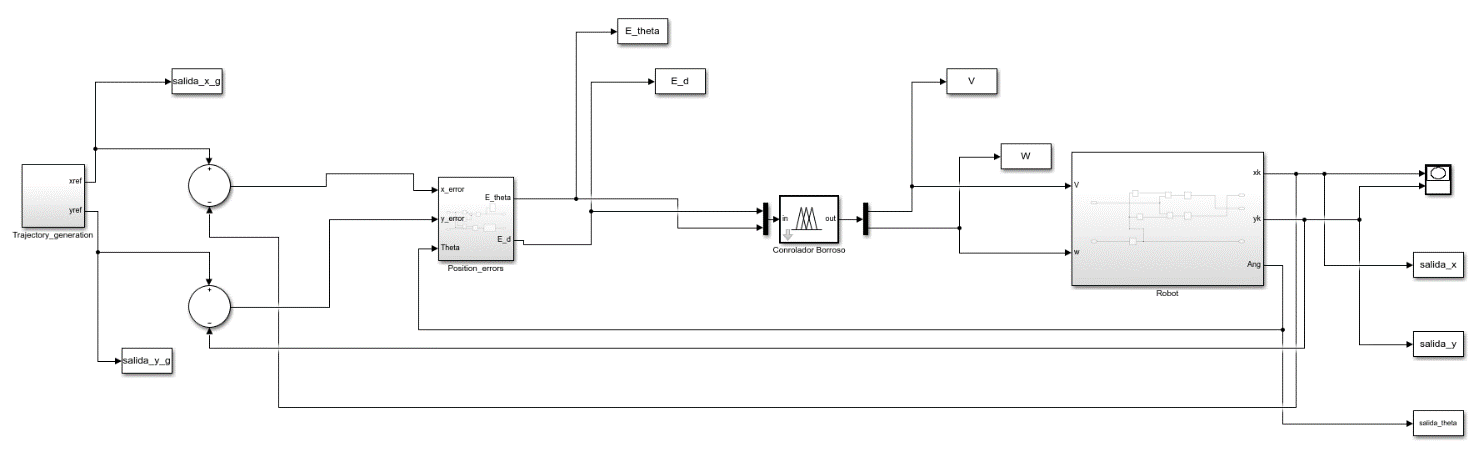
En este apartado vamos a sustituir las referencias de posición de nuestro esquema de control (refx y refy) por el **bloque “Trajectory\_control”** que ya hemos utilizado en anteriores prácticas y genera la siguiente trayectoria:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Figura 6:** Trayectoria generada por el bloque “Trajectory\_generation”

Tras introducir el bloque en nuestro esquema nos encontraremos con el esquema de *Simulink* mostrado en la Figura 7:



**Figura 7:** Esquema con el bloque Trajectory\_generation

Al ejecutar una simulación con este esquema podemos observar como nuestro Controlador Borroso es capaz de imitar perfectamente la trayectoria creada por el bloque recién introducido:

Imagen que contiene mapa

Descripción generada automáticamente

**Figura 8:** Trayectorias resultado de la simulación

Como se puede comprobar en la Figura 8, nuestro script de simulación compara en una misma figura ambas trayectorias en dos colores distintos: **Azul** (Trajectory\_generation) y **Naranja** (Controlador Borroso). Finalmente, nuestro bloque de Controlador Borroso ha sido completamente capaz de **imitar el mismo recorrido** que el bloque generador haciendouso de **las reglas** y **las funciones de pertenencia** generadas anteriormente.

**PARTE 2 DE LA PRACTICA**