Capítulo 5

Control borroso directo aplicado a la planta

Visto el control clásico es hora de probar técnicas de control borroso. En este capítulo se desarrollará un control borroso incremental a partir de los resultados obtenidos en los controladores clásicos del capítulo precedente. Es de esperar que al tratarse de un sistema no lineal, el controlador borroso mejore las prestaciones de su homólogo clásico.

La lógica borrosa ha llegado a ser rápidamente una de las tecnologías más exitosas en la actualidad para desarrollar sistemas de control. Con su ayuda, los requerimientos muy complejos pueden ser implementados en un control simple, barato y de fácil mantenimiento. Haciendo un breve resumen, el control borroso tiene como objetivo realizar el análisis y diseño de sistemas de control basados en sistemas de inferencia difusos, los cuales permiten transformar el conocimiento sobre el control de una planta, expresado en forma de regla SI-ENTONCES, en una relación matemática para el cálculo de acciones de control a partir de mediciones de la planta. Una regla borrosa puede reemplazar varias reglas convencionales. Y puesto que la lógica borrosa crea un sistema de control al combinar reglas y conjuntos borrosos, permitiendo a los diseñadores construir controladores aun cuando no tengan un entendimiento total del sistema.

Al igual que ocurrió con el controlador PID, primero se utilizará una herramienta en MATLAB para el diseño de controladores borrosos, para más tarde, implementar un controlador borroso en un PLC. Volveremos a utilizar el software Unity, con lo que se conseguirá un controlador que cumpla con la norma IEC 1131-7.

Por último, una vez se realicen una serie de pruebas, se compararán los resultados obtenidos con los que se obtuvieron en el capítulo anterior con los controladores clásicos.

5.1. Control Borroso Incremental

Se diseñará un control directo pseudo-lineal incremental. Un controlador de este tipo es comparable a un PI, de hecho parte de la sintonización de un PI clásico para luego, a través de sus constantes, calcular las del controlador borroso (ver [16, 8]). La estructura de cualquier control borroso es la de la Figura 5.1.1.

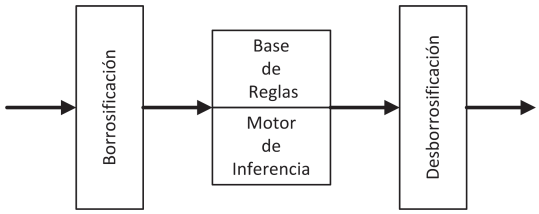

Figura 5.1.1: Estructura general de un controlador borroso.

En el caso de un controlador directo, las entradas serán funciones del error y la salida estará orientada a la variable manipulada. En este proyecto se diseñará un controlador incremental, cuyas entradas son el error y la derivada del error, moduladas por las constantes GE y GCE respectivamente. La salida será la derivada de la variable manipulada que, previa modulación con una constante GCU, se integra antes de actuar.

A este controlador también se le conoce con el nombre de *Fuzzy* PI, por su similitud con la acción del controlador clásico, cuya salida es:

$$u_n = u_{n-1} + \triangle u_n \tag{5.1.1}$$

$$\triangle u_n = K_p \left(e_n - e_{n-1} + \frac{1}{T_i} e_n T_s \right) \tag{5.1.2}$$

El esquema básico de control de un controlador borroso incremental *(Fuzzy Incremental Controller)* es el de la Figura 5.1.2.

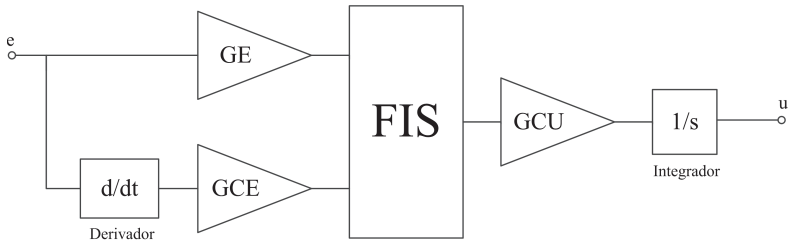

Figura 5.1.2: Estructura de un controlador borroso incremental.

El controlador tiene la siguiente salida:

$$u_n = \sum_i (cu_i \cdot GCU \cdot T_s) \tag{5.1.3}$$

La aproximación lineal para éste controlador es:

$$u_n = \sum_i (E_i + CE_i) \cdot GCU \cdot T_s = GCU \cdot \sum_{i=1}^{n} \left[GE \cdot e_i + GCE \cdot \frac{e_i - e_{i-1}}{T_s} \right] \cdot T_s = \tag{5.1.4}$$

$$= GCU \cdot \left[GE \cdot \sum_{i=1}^{n} e_i \cdot T_s + GCE \cdot \sum_{i=1}^{n} (e_i - e_{i-1} \right] = GCE \cdot GCU \cdot \left[\frac{GE}{GCE} \sum_{i=1}^{n} e_i \cdot T_i + e_n \right]$$

Comparando con las constantes del PI clásico, tenemos que:

$$K_p = GCE \cdot GCU \tag{5.1.5}$$

$$\frac{1}{T_i} = \frac{GE}{GCE} \tag{5.1.6}$$

Basta con fijar GE para obtener las demás a través de los valores de las constantes clásicas.

Se establecen, de la manera mas simple, dos funciones de pertenencia triangulares para cada entrada con vértices de -1 y 1 respectivamente y tres para la salida con vértices -1, 0 y 1 (aunque en nuestro caso la salida sera con singletons, no con triángulos). Los nombres pueden ser, para las entradas: Negativo y Positivo y para la salida: Negativo, Cero y Positivo. Como veremos más adelante, la elección de las funciones de pertenencia de la salida de tipo singleton no es aleatoria, el desborrosificador que utilizará el PLC utiliza este tipo de funciones de pertenencia.

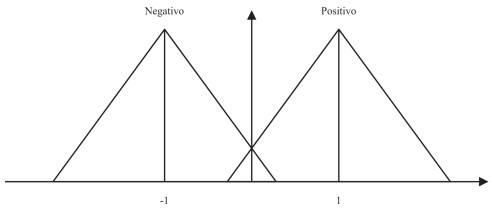

Figura 5.1.3: Funciones de pertenencia de las entradas.

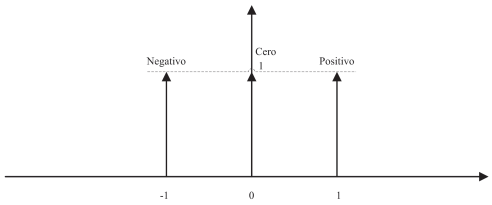

Figura 5.1.4: Funciones de pertenencia de la salida.

La base de reglas para este tipo de controlador está formado por cuatro:

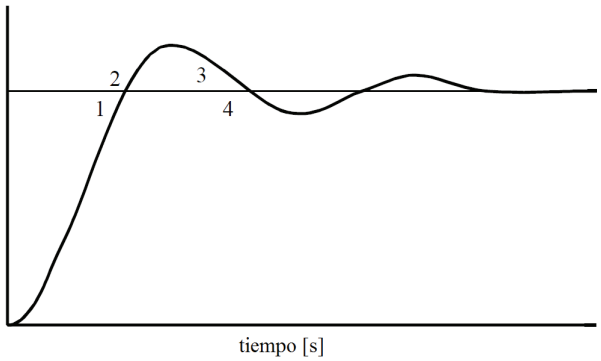

Figura 5.1.5: Respuesta ante entrada escalón y número de estado del proceso.

Estado del proceso	Error	Variación del error
1	Positivo	Negativo
2	Negativo	Negativo
3	Negativo	Positivo
4	Positivo	Positivo

Cuadro 5.1: Números de estado y su correspondencia con las variables de entrada.

- SI E es P y $\triangle E$ es P, ENTONCES $\triangle u$ es *Positiva*.
- SI E es P y $\triangle E$ es N, ENTONCES $\triangle u$ es *Cero*.
- SI E es N y $\triangle E$ es P, ENTONCES $\triangle u$ es *Cero*.
- SI E es N y $\triangle E$ es N, ENTONCES $\triangle u$ es *Negativa*.

Un controlador borroso directo tiene la ventaja de que las entradas del algoritmo son simplemente el error y su derivada. Su ajuste se basa en un procedimiento (como los PID). En concreto, se debe ajustar las constantes de un PI (K_p, T_i) y, a través de ellas, se obtienen las GCU, y GCE (eligiendo una GE determinada).

Para comprender mejor la base de reglas se presenta la Figura 5.1.5 y la Tabla 5.1.

El método de desborrosificación elegido para la salida es:

$$u = \frac{\sum_{i=1}^{n} \mu_i \cdot S_i}{\sum_{i=1} \mu_i} \tag{5.1.7}$$

donde n es el número de singletons (en nuestro caso tres: -1, 0 y 1) y μ_i es el grado de pertenencia asociado al singleon i.

Una vez explicada el procedimiento de diseño, utilizaremos la *Fuzzy Logic Toolbox™* para aplicar en Simulink el controlador borroso incremental al modelo de la planta.

5.2. Control borroso incremental en Simulink

El proceso es muy similar al seguido con el controlador PID. Primero diseñaremos controladores borrosos en Simulink y luego los implementaremos en Unity para el PLC. Para comenzar a diseñar el controlador en Simulink debemos seleccionar la *Fuzzy Logic Toolbox* en el navegador de librerías de Simulink y seleccionar la *toolbox* que nos interesa, tal y como se muestra en la Figura 5.2.1.

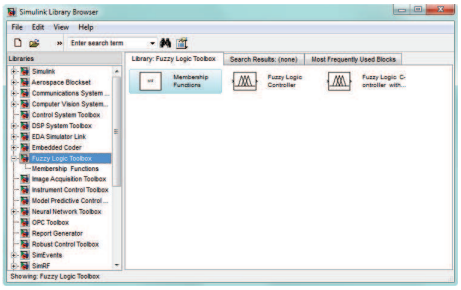

Figura 5.2.1: *Fuzzy Logic Toolbox*.

La librería la componen un bloque con funciones de pertenencia y y el bloque del controlador borroso (con y sin visualizador de reglas). Para implementar el controlador borroso incremental con el modelo de la planta, únicamente necesitaremos el bloque *Fuzzy Logic Controller with Ruleviewer*.

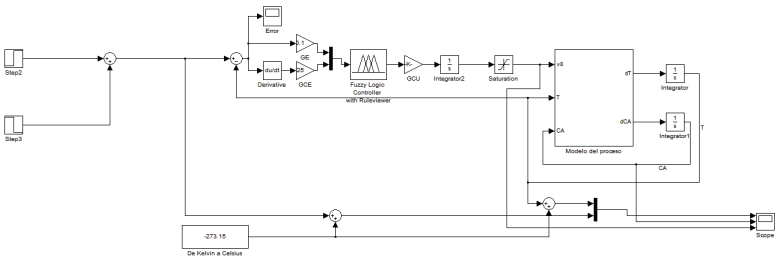

Figura 5.2.2: Controlador Borroso Incremental en Simulink.

Siguiendo la estructura de la Figura 5.1.2, necesitaremos varios bloques de ganancia para los parámetros GE, GCE y GCU. Además como las entradas son el error y su derivada, se restará el valor de la temperatura actual con el de la referencia para sacar el error y utilizaremos un bloque de derivada para conseguir la variación del error. Como se comentó en el apartado anterior, la salida de la base de reglas necesita ser integrada, por lo que usaremos un integrador para conseguir la señal de control y limitaremos la salida del mismo entre los valores 0 y 100.

Para configurar el bloque del controlador borroso es necesario configurar un fichero .fis, es posible realizarlo manualmente pero no es aconsejable. Tenemos a nuestra disposición una serie de editores que nos facilitarán la configuración de nuestro sistema de inferencia borroso (FIS).

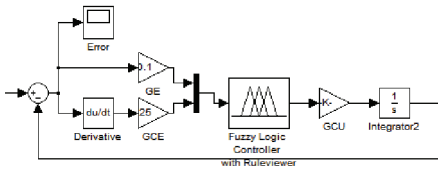

Figura 5.2.3: Bloques que forman el controlador borroso.

Para utilizarlo basta con escribir en la línea de comando de MATLAB: *fuzzy*, y se abrirá el Editor de FIS, como en la Figura 5.2.4. A partir de esta pantalla podemos acceder al resto de editores gráficos y visualizadores de la *toolbox*.

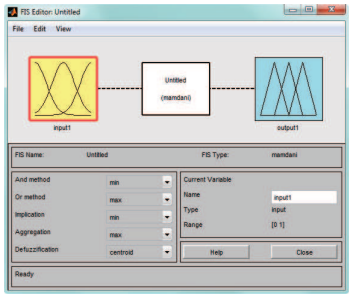

Figura 5.2.4: Editor de Sistemas de Inferencia Borrosos.

Lo primero que debemos seleccionar es el tipo de FIS que queremos, si de tipo Mamdani o de tipo Sugeno. Seleccionamos *File* en el menú superior, luego *New FIS* y por último Sugeno. Ahora vamos a definir el número de entradas. Por defecto viene una única entrada, puesto que vamos a tener dos entradas (error y su derivada) tendremos que añadir una nueva entrada. Para ello seleccionamos *Edit*, *Add variable* y escogemos *Input*. Desde la misma pantalla podemos modificar el nombre y cambiar el funcionamiento de los operadores.

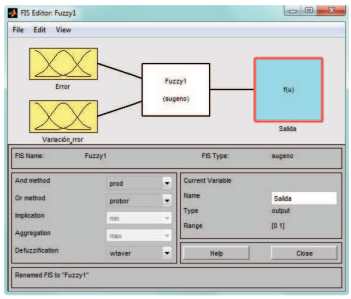

Figura 5.2.5: Configuración para el control borroso incremental.

Una vez hecho esto definiremos las funciones de pertenencia tanto de entrada como de salida. Las funciones de pertenencia de las entradas eran de tipo triangular mientras que las funciones de pertenencia de la salida son de tipo singleton. Para esto basta con hacer doble clic sobre el recuadro de las entradas o salida, o seleccionar *Membership Functions* en el menú *Edit*.

Ya en el Editor de Funciones de Pertenencia configuramos las variables tal y como las definimos en el primer apartado. Podemos elegir el tipo de función de pertenencia (triangular, rectangular, etc.) y definir los puntos de la misma.

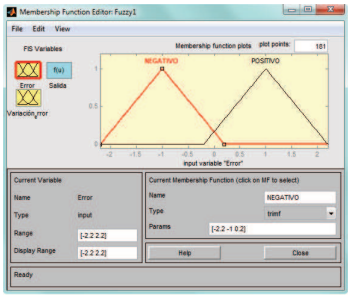

Figura 5.2.6: Ventana del Editor de Funciones de Pertenencia.

Definidas el número de entradas y salidas, las funciones de pertenencia de las entradas y salida queda por definir la Base de las Reglas. Al igual que en el caso anterior podemos hacer doble clic con el ratón sobre el bloque centrar o seleccionar el Editor de Base de Reglas en el menú superior.

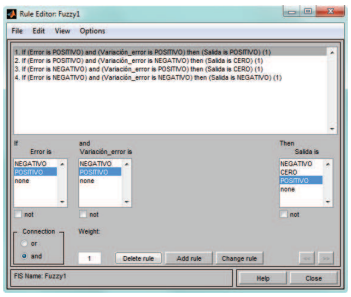

Figura 5.2.7: Ventana del Editor de Reglas.

A través de este editor creamos la base de reglas del sistema de inferencia borroso. Basta con ir combinando adecuadamente los valores de las entradas y los valores de la salida.

Para terminar tenemos que guardar los cambios que hemos realizado, para ello escogemos la opción de *Export* en el menú *File*. Podemos exportar nuestro FIS bien a un archivo .fis o al *Workspace* de MATLAB. Haremos ambas cosas para poder recuperar en cualquier momento nuestra configuración.

Por último configuramos el controlador lógico borroso de Simulink para que utilice el FIS que acabamos de diseñar. Tendremos que hacer doble clic sobre el bloque y escribir el nombre con el que hemos exportado el archivo .fis al *Workspace*. Este bloque buscará el nombre que le pongamos en el *Workspace* por lo que será necesario exportar el FIS al espacio de trabajo de MATLAB cada vez que queramos utilizar el controlador borroso.

Ya podemos simular el controlador borroso incremental en Simulink. Cada vez que realicemos una simulación aparecerá el Visualizador de Reglas, donde podemos observar qué valores van tomando las entrada y la salida.

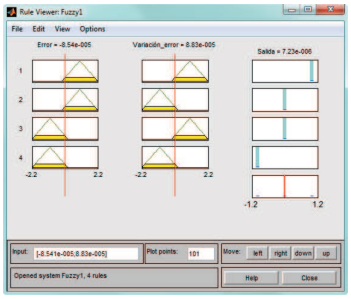

Figura 5.2.8: Ventana del Visualizador de Reglas.

5.2.1. Simulaciones

Se presentan a continuación varias simulaciones con distintos valores de referencia. En las mismas gráficas aparecerán los resultados que obtuvimos con los PIs clásicos, diseñados en el capítulo anterior.

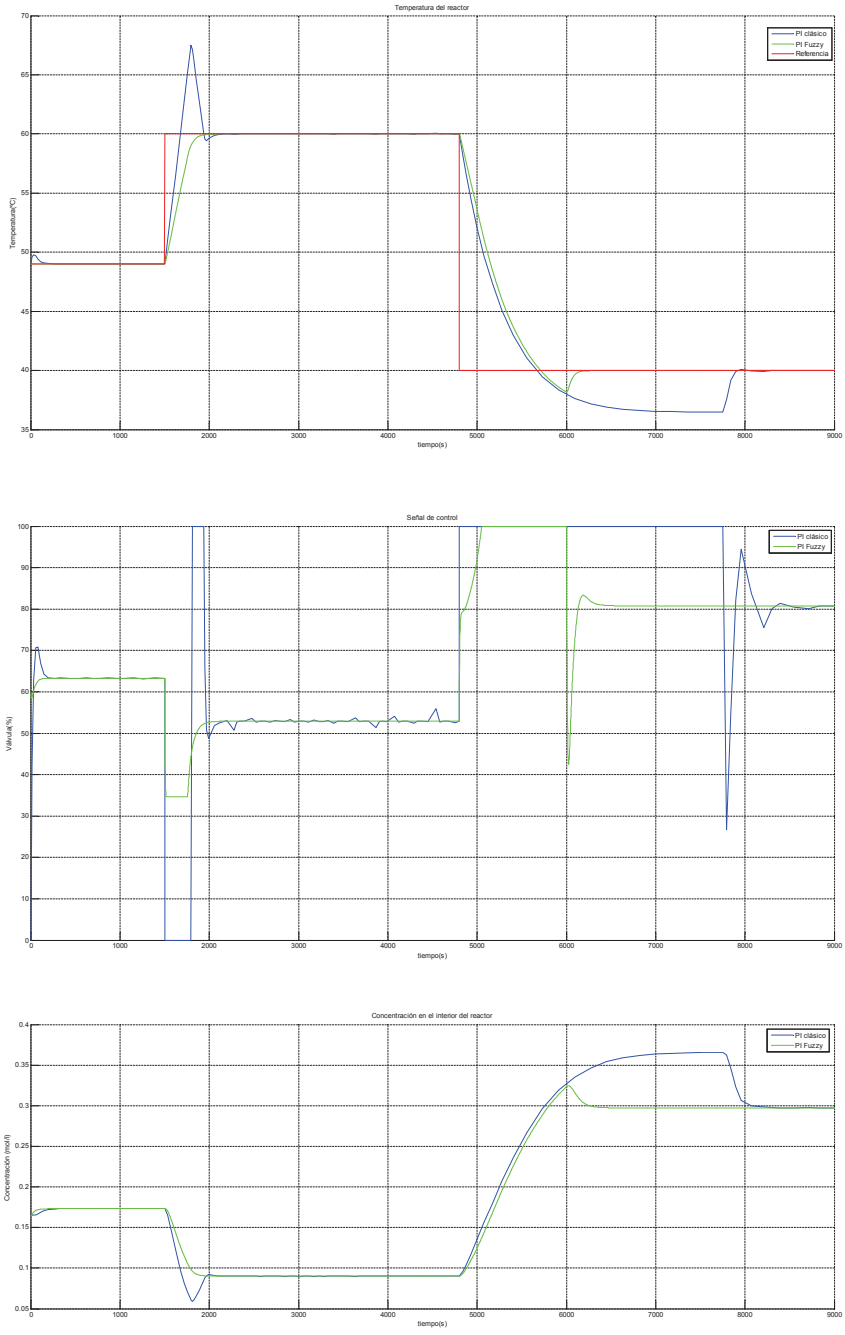

Figura 5.2.9: Comparación 1 entre el PI clásico y el PI Fuzzy.

K_p	$1/T_i$	GE	GCE	GCU
-58.98	0.016	0.1	6	-9.83

Cuadro 5.2: Parámetros del controlador clásico y el borroso de la comparación 1.

Como era de esperar el controlador borroso incremental mejora al PI clásico. Sobre todo a la hora de cambiar de punto de trabajo.

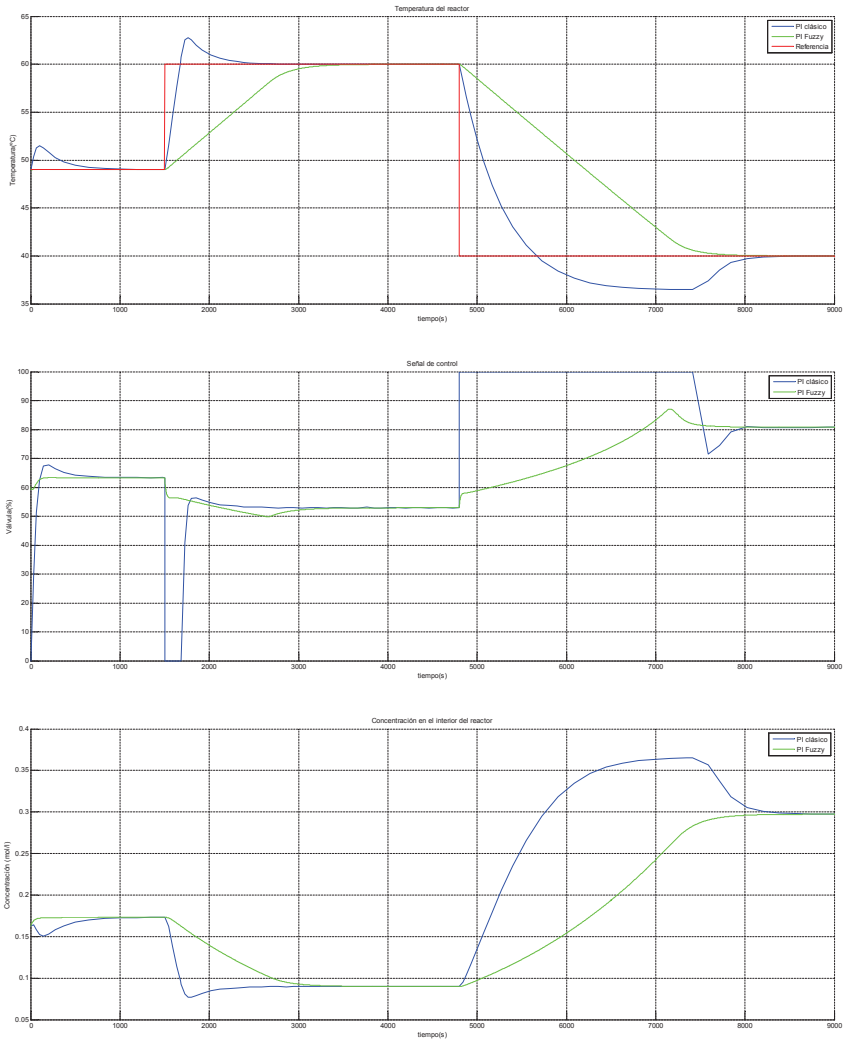

Figura 5.2.10: Comparación 2 entre el PI clásico y el PI Fuzzy.

K_p	$1/T_i$	GE	GCE	GCU
-19.957	0.004	0.1	25	-0.79828

Cuadro 5.3: Parámetros del controlador clásico y el borroso de la comparación 2.

El sistema con el controlador borroso prácticamente no sobreoscila, mientras que el controlador clásico sobreoscila en ambos casos, además al cambiar el punto de trabajo no responde adecuadamente. En cambio el controlador clásico obtiene una respuesta más rápida.

5.3. Implementación en Unity

Lo primero que hay que tener en cuenta antes de comenzar a diseñar el controlador borroso incremental en Unity, es que esta librería no viene por defecto en el software Unity Pro, es una librería que se consigue a parte del programa principal. El ordenador que se encuentra conectado a la planta ya botón cuenta con dicha librería así como una de las últimas versiones del software Unity Pro.

La aplicación es exactamente la misma que la utilizada en el controlador PID. Por ello, nos centraremos en explicar cómo se ha diseñado el controlador borroso en Unity. Para el desarrollo del algoritmo se ha usado el lenguaje FBD de la norma IEC 1131-3, con las funciones de la librería *Fuzzy* de Unity. Para seleccionar la librería pulsamos en el menú superior "Herramientas" y seleccionamos "Explorador de librería de tipos". Una vez abierta la ventana, elegimos el nombre de la biblioteca: Fuzzy_lib.

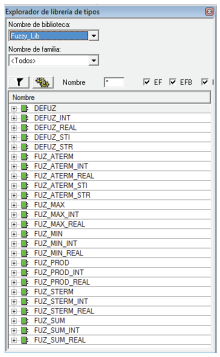

Figura 5.3.1: Explorador de librería de tipos.

La librería incluye todo lo necesario para programar el controlador borroso incremental.

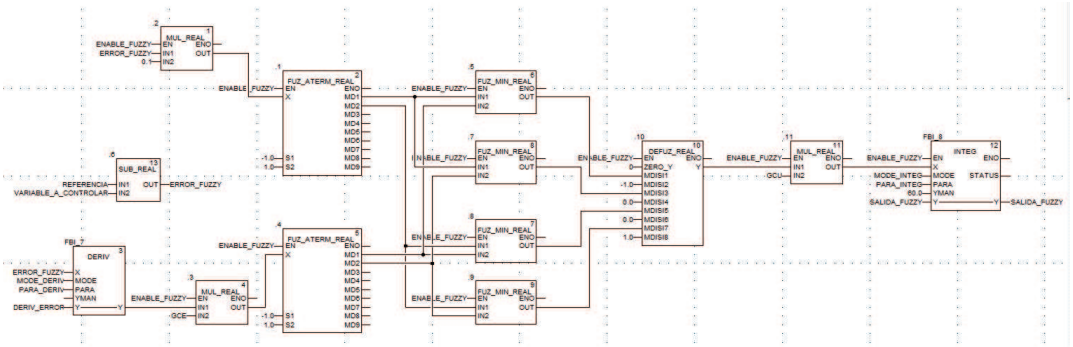

Figura 5.3.2: Control borroso incremental implementado en Unity.

Comenzaremos diseñando el borrosificador. De todas las funciones de bloque que hay para utilizar como borrosificador utilizaremos la FUZ_ATERM de tipo REAL. Este bloque es capaz de borrosificar hasta 9 términos lingüísticos por entrada y especificar cada uno de los grados de pertenencia (salidas de MD1 a MD9).

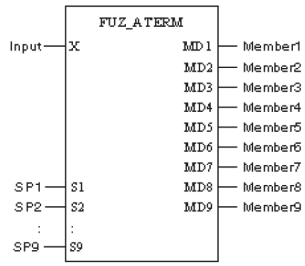

Figura 5.3.3: Representación borrosificador FUZ_ATERM en FBD.

Para definir las entradas triangulares que necesitamos, utilizamos las salidas MD1 y MD2, y establecemos S1=-1 y S2=1. Para definir los distintos tipos de función de pertenencia:

- Rampas para la primera y la última función de pertenencia.
- Triángulos para las funciones de pertenencia entre la primera y la última.

El número de puntos de referencia (de S1 a Sx) se puede incrementar hasta un máximo de 9 mediante la modificación del tamaño vertical del bloque. El número de grados de pertenencia calculados corresponde al número de funciones de pertenencia. Los tipos de datos de los valores de entrada y el valor de salida deben ser iguales. Necesitaremos un bloque FUZ_ATERM por cada una de las entradas, en este caso dos (error y derivada del error).

Pero a diferencia del bloque PIDFF que calculaba el error cometido, al diseñar el controlador borroso debemos calcular el error a parte. Utilizaremos un bloque SUB_REAL para realizar la diferencia entre las dos señales (la referencia y la variable a controlar) y el resultado será el error buscado. También necesitaremos un bloque DERIV para calcular la derivada del error. Habrá que configurarlo de forma que no añada ninguna ganancia y no introduzca retardo.

Para implementar los parámetros del PI borroso GE, GCE y GCU será suficiente con utilizar un bloque MUL_REAL que nos multiplica dos entradas. En una fijaremos alguno de los parámetros de diseño y en la otra irá una de las entradas o la salida.

Con esto tendríamos preparado la primera parte del controlador borroso: el borrosificador.

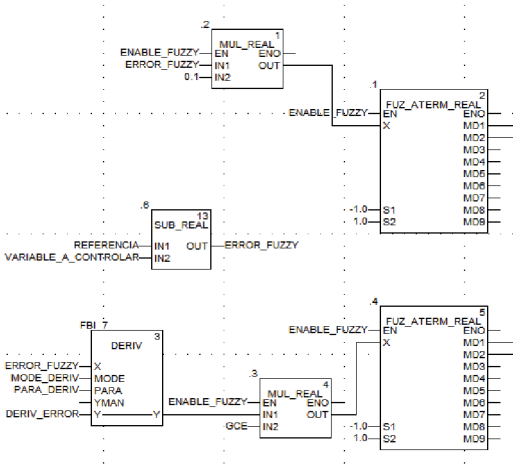

Figura 5.3.4: Borrosificador en Unity.

El siguiente paso será diseñar la Base de las Reglas. Utilizaremos el bloque FUZ_MIN, el cual reconoce el valor de entrada mas pequeño y lo envía a la salida. Puesto que vamos a trabajar con variables reales escogemos el bloque de tipo REAL.

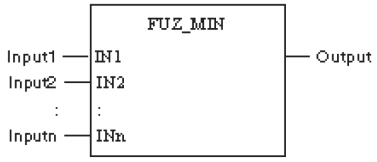

Figura 5.3.5: Representación del bloque FUZ_MIN en FBD.

Combinaremos las salidas del borrosificador entre los distintos bloques FUZ_MIN hasta completar la base de Reglas que definimos al principio de este capítulo:

- SI E es P y $\triangle E$ es P, ENTONCES $\triangle u$ es *Positiva*.
- SI E es P y $\triangle E$ es N, ENTONCES $\triangle u$ es *Cero*.
- SI E es N y $\triangle E$ es P, ENTONCES $\triangle u$ es *Cero*.
- SI E es N y $\triangle E$ es N, ENTONCES $\triangle u$ es *Negativa*.

Como tenemos cuatro reglas serán necesarios cuatro bloques FUZ_MIN para poder completar la base.

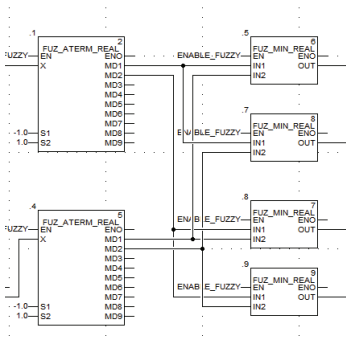

Figura 5.3.6: Conexión borrosificador - base de Reglas.

Por último, para completar el controlador borroso, necesitaremos un desborrosificador. Para este fin está el bloque DEFUZ.

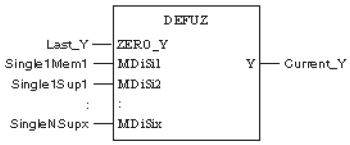

Figura 5.3.7: Representación desborrosificador DEFUZ en FBD.

Este bloque realizara una desborrosificación de los términos lingüísticos que se representan mediante singletons, según el método de máximo valor medio. La posición de los singletons se define mediante puntos de referencia (de S1 a S9). La importancia de cada término se establece mediante el correspondiente grado de pertenencia (de MD1 a MD9).

La fórmula utilizada es:

$$Y = \frac{\sum_{i=1}^{n} MD_i \cdot S_i}{\sum_{i=1}^{n} MD_i} \tag{5.3.1}$$

donde n=número de singletons.

La forma cómo se conectaron las salidas de la base de Reglas con el desborrosificador se muestra en la Figura 5.3.8.

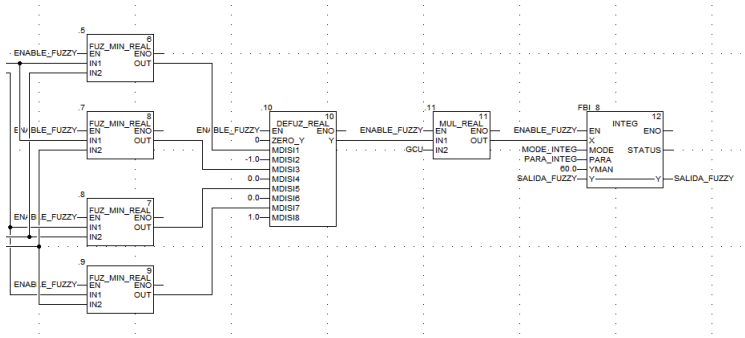

Figura 5.3.8: Desborrosificador en Unity.

Para completar el diseño del controlador borroso incremental, la salida del desborrosificador se multiplicará por el parámetro GCU, y el resultado se integrará para obtener la señal de control. Utilizaremos el bloque INTEG para integrar la salida y configuraremos los parámetros del integrador de tal forma que sólo permita valores de salida entre 0 y 100.

5.4. Simulación

Una vez está diseñado el controlador borroso incremental en Unity estamos en disposición de probar su funcionamiento. Al igual que ocurría con el controlador clásico se puede utilizar el SCADA que se ha creado para vigilar el proceso. Únicamente habrá que elegir el control borroso (*Fuzzy*) en lugar del PID pulsando el botón FUZZY. Además, habrá que pulsar el botón de ENABLE para que el controlador pueda arrancar.

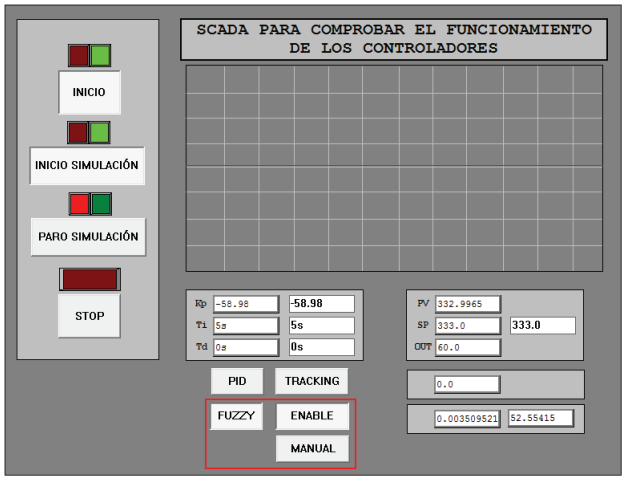

Figura 5.4.1: Selección del controlador borroso en el SCADA.

Podremos alternar entre modo manual y automático con el botón MANUAL.

Utilizando los datos de los PI clásicos, obtuvimos las siguientes simulaciones:

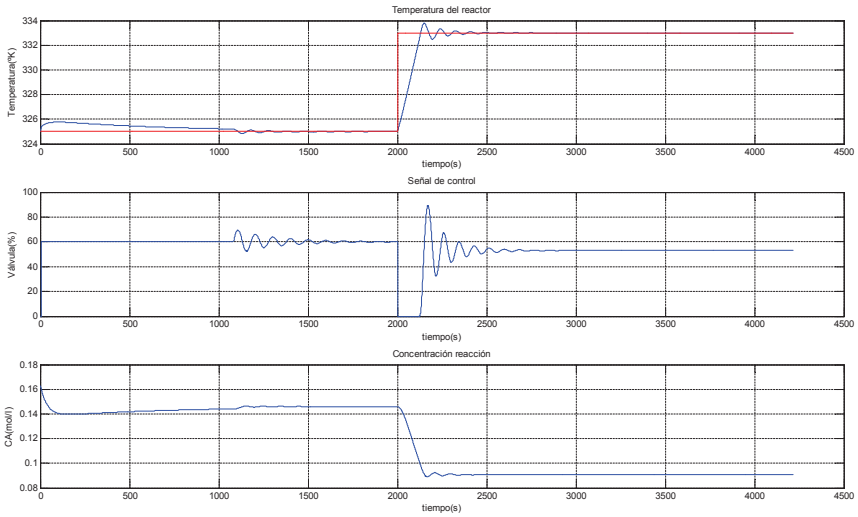

Figura 5.4.2: Simulación 1 con GE=0.1, GCE=0.5 y GCU=-117.96.

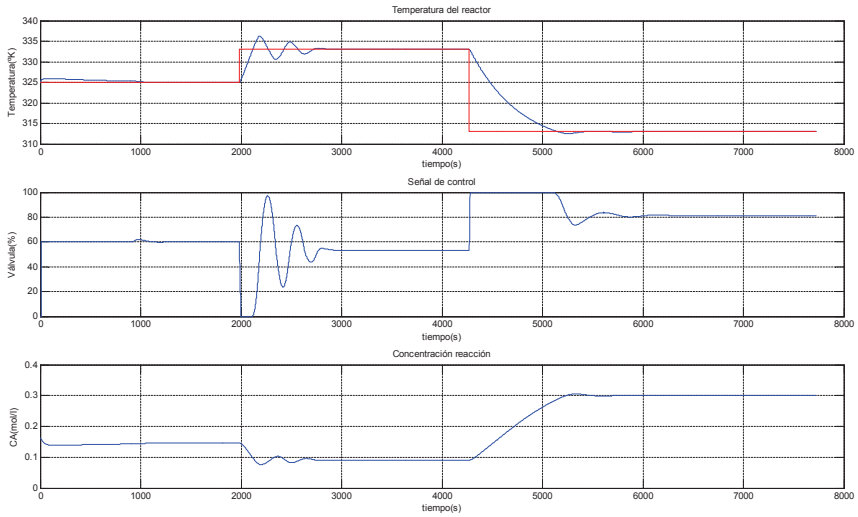

Figura 5.4.3: Simulación 2 con GE=0.1, GCE=6 y GCU=-9.83.

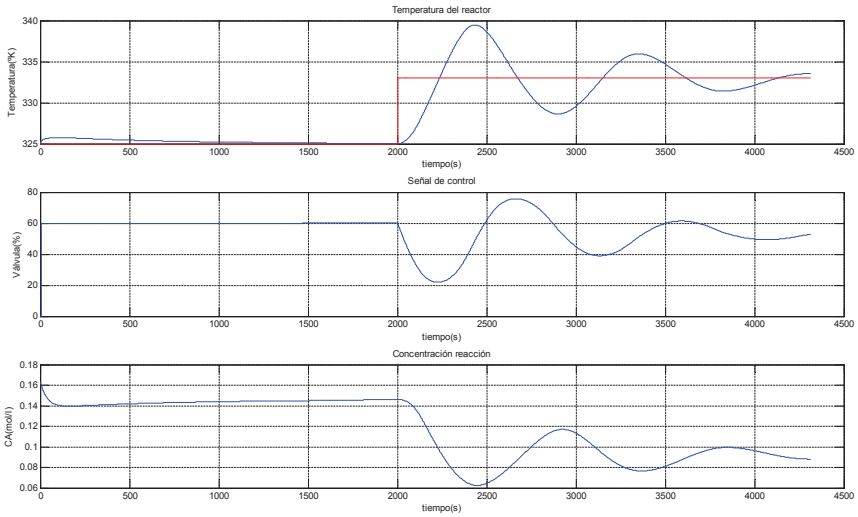

Figura 5.4.4: Simulación 3 con GE=0.1, GCE=25 y GCU=-0.79828.

Vemos como si los parámetros no son adecuados, la respuesta no es buena.

Comparando el PI clásico equivalente con el PI Fuzzy obtenido a partir de sus parámetros:

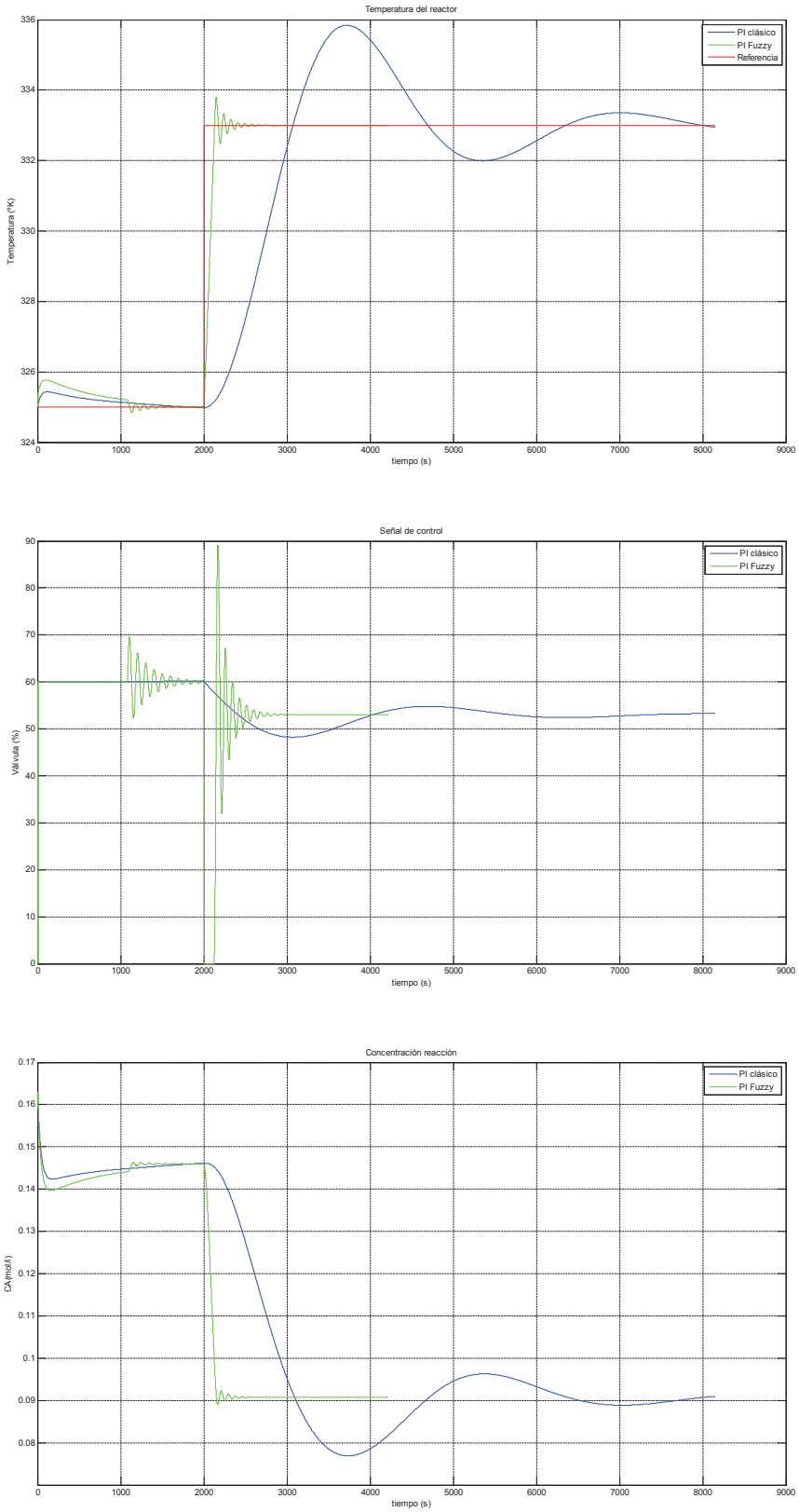

Figura 5.4.5: Comparativa entre controladores clásico y borroso equivalentes.