

# Control difuso para el seguimiento de nivel en un tanque de agua

Iñigo Martínez Peña y Jon Garzón García  
imartinez280@ikasle.ehu.eus; jgarzon003@ikasle.ehu.eus

Control Inteligente  
Máster en Ingeniería de Control, Automatización y Robótica  
Escuela de Ingeniería de Bilbao, EIB

## Resumen

En este trabajo se describe el ejemplo de MathWorks para simular un *Fuzzy Inference System* (FIS) en Simulink mediante *Fuzzy Logic Toolbox*. El caso de estudio corresponde al control del nivel de agua en un tanque (`sltank`), donde un controlador difuso actúa sobre la apertura/cierre de una válvula para seguir un valor de referencia. Se presenta la estructura del modelo, el uso de los bloques *Fuzzy Logic Controller* y *Fuzzy Logic Controller with Ruleviewer*, el ajuste incremental de la base de reglas para reducir oscilaciones y, de forma breve, opciones avanzadas relacionadas con modos de simulación y acceso a variables intermedias del proceso de inferencia.

**Palabras clave:** control difuso, Mamdani, Simulink, Fuzzy Logic Toolbox, reglas lingüísticas.

## 1. Descripción del proceso

El sistema a controlar es un tanque de agua con una dinámica no lineal. La entrada de control actúa sobre el caudal de entrada mediante una válvula, mientras que el caudal de salida depende de un parámetro geométrico (diámetro del tubo de salida, constante) y de la presión asociada al nivel de agua, que varía con el propio nivel. Esta dependencia hace que el comportamiento global no sea lineal, lo que motiva el uso de un controlador basado en reglas.

El objetivo de control es que el nivel del tanque siga una referencia (setpoint) ante cambios y perturbaciones, con una respuesta estable (sin oscilaciones sostenidas) y con un error estacionario reducido. El ejemplo de Simulink permite comparar una solución inicial simple (reglas basadas solo en error) con una mejora que incorpora la tendencia del nivel (derivada o razón de cambio).

### 1.1. Planteamiento del problema de control

Desde el punto de vista de ingeniería de control, el problema consiste en regular una variable de

proceso (nivel) que responde de manera lenta y no lineal a la acción de mando. Esta no linealidad implica que una misma variación de la válvula no produce siempre la misma variación de nivel, ya que la sensibilidad cambia según la altura de agua acumulada. Por ello, un esquema de control difuso resulta adecuado al permitir traducir conocimiento experto en reglas lingüísticas robustas frente a cambios de régimen.

Además, el sistema debe operar con compromiso entre rapidez y suavidad. Una acción de control demasiado agresiva puede reducir temporalmente el error, pero también aumentar el sobreimpulso y las oscilaciones. En cambio, una acción demasiado conservadora mejora la estabilidad, aunque penaliza el tiempo de establecimiento. El diseño final del controlador busca equilibrar ambos criterios para obtener una respuesta útil en condiciones realistas.

### 1.2. Estrategia de modelado y simulación

El proceso se simula en un entorno de bloques donde se integran el modelo del tanque, el cálculo del error y el controlador difuso. Se usan escenarios con referencias escalón y se observan el nivel, error y señal de control en el tiempo. A partir de estos resultados se identifican problemas (retardo, oscilación, saturación) y se ajustan las funciones de pertenencia o reglas si es necesario.

### 1.3. Criterios de evaluación del desempeño

Para valorar el resultado se emplean indicadores clásicos de control: tiempo de subida, sobreimpulso, tiempo de establecimiento y error en régimen permanente. De forma cualitativa, también se analiza la suavidad de la señal de válvula, ya que oscilaciones rápidas de la actuación pueden no ser deseables en un sistema físico real por desgaste o consumo energético.

Finalmente, la comparación entre la configuración inicial y la configuración mejorada permite justificar técnicamente la inclusión de la variable de tendencia. La versión enriquecida mantiene el seguimiento de referencia y reduce la oscilación alre-

dedor del punto de operación, aportando una respuesta más estable y consistente con los objetivos del proyecto.

## 2. Modelo en Simulink y componentes

### 2.1. Bloques disponibles para simular un FIS

La simulación de un FIS en Simulink se realiza principalmente con los siguientes bloques:

- **Fuzzy Logic Controller:** evalúa un FIS (tipo 1 y, según configuración, tipo 2) y entrega la salida del controlador.
- **Fuzzy Logic Controller with Ruleviewer:** equivalente al anterior, pero permite visualizar durante la simulación la inferencia de reglas.
- **FIS Tree:** permite simular estructuras en árbol de FIS interconectados (útil en diseños jerárquicos).

Cuando se menciona “tipo de sistema” en el contexto difuso, se hace referencia al tipo de sistema de inferencia difusa (FIS). De forma breve: un FIS tipo 1 usa funciones de pertenencia nítidas, mientras que un FIS tipo 2 incorpora incertidumbre adicional en dichas funciones.

En este contexto, “incertidumbre” significa que algunas magnitudes del problema no se conocen con exactitud o varían en el tiempo: ruido de sensor, pequeñas variaciones de parámetros del proceso, perturbaciones externas o ambigüedad al definir términos lingüísticos como “nivel bajo” o “sube rápido”.

**Limitaciones del FIS tipo 1.** En un FIS tipo 1, las funciones de pertenencia se definen de forma determinista (por ejemplo, una función triangular o trapezoidal con parámetros fijos). Esta rigidez puede ser insuficiente cuando el sistema opera bajo condiciones cambiantes o con incertidumbre significativa en las mediciones. Por ejemplo, si el sensor de nivel presenta ruido variable, un tipo 1 no puede adaptar su comportamiento a ese rango de incertidumbre, lo que puede degradar la calidad del control en escenarios reales.

**Ventajas del FIS tipo 2.** Un FIS tipo 2 representa explícitamente esa falta de precisión en las funciones de pertenencia: cada función se describe no con un único contorno, sino con una región de incertidumbre (footprint of uncertainty). Esto permite que el controlador sea más robusto cuando las condiciones reales difieren del modelo nominal, ya

que puede manejar automáticamente variaciones en las entradas sin necesidad de reajustar parámetros.

**Alcance de este trabajo.** En el presente proyecto se utiliza un FIS tipo 1 estándar (Mamdani), que resulta adecuado para el escenario planteado, donde el modelo del tanque es suficientemente preciso y las perturbaciones son moderadas. La mención al tipo 2 se realiza únicamente con fines de contextualización teórica y para justificar futuras extensiones en entornos más exigentes.

### 2.2. Modelo sltank

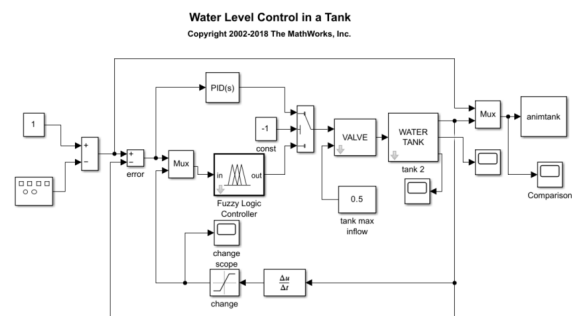


Figura 1: Modelo Simulink del tanque de agua con controlador difuso.

El modelo `sltank` implementa un lazo de control donde el bloque *Fuzzy Logic Controller* recibe dos señales relacionadas con el nivel del agua y genera una acción sobre la válvula. Para que el bloque funcione, el parámetro *FIS name* debe referenciar un objeto FIS existente en el *workspace* de MATLAB (por ejemplo, un `mamfis` llamado `tank`).

### 2.3. Modelo sltankrule (Ruleviewer)

La variante `sltankrule` utiliza el bloque *Fuzzy Logic Controller with Ruleviewer*. La estructura general del lazo es la misma, pero se añade la posibilidad de inspeccionar qué reglas se activan y con qué intensidad. Además, al pausar la simulación se pueden ajustar manualmente las entradas en el visor para observar el efecto sobre la inferencia y la salida del controlador.

```
1 newRules = [rule4 rule5];
2 tank = addRule(tank,newRules);
```

#### 2.3.1. Carga del FIS y configuración inicial

```
1 open_system('sltank')
2
3 rule1 = "If level is okay then valve is
   no_change";
```

```

4 rule2 = "If level is low then valve is open_fast";
5 rule3 = "If level is high then valve is close_fast";
6 rules = [rule1 rule2 rule3];
7 tank.Rules = fisrule(rules);

```

En esta sección se cargan las variables necesarias del archivo FIS y se prepara el espacio de trabajo. La importación del objeto FIS es crítica en el control difuso, ya que contiene la definición completa del sistema: funciones de pertenencia, base de reglas e interpretación de operadores (AND, OR, implicación).

### 2.3.2. Preparación de entradas y simulación

```

1
2 open_system('sltank/Comparison')
3 sim('sltank',100)

```

Aquí se generan los vectores de entrada que alimentarán el controlador difuso. La discretización del espacio de entrada es fundamental para evaluar cómo el FIS responde en diferentes puntos de operación. Esta exploración sistemática del dominio de entrada permite validar la coherencia de las decisiones de control difuso en todo el rango de funcionamiento esperado.

### 2.3.3. Evaluación del sistema de inferencia

```

1
2 rule4 = "If level is okay and rate is positive then valve is close_slow";
3 rule5 = "If level is okay and rate is negative then valve is open_slow";
4 newRules = [rule4 rule5];

```

La evaluación del FIS mediante el método `evalfis()` ejecuta el proceso completo de defuzzificación: fusificación de entradas, activación de reglas mediante los operadores lógicos configurados, agregación de consecuentes y finalmente obtención de la salida nítida. Este módulo es el núcleo del controlador difuso implementado en Simulink.

### 2.3.4. Visualización de resultados

```

1 tank = addRule(tank,newRules);
2
3 sim('sltank',100)
4
5 open_system('sltankrule')

```

Las gráficas de superficie (`plotsurf`) ofrecen una vista tridimensional de la relación entrada-salida del controlador difuso. Esta representación es esencial para validar que el comportamiento del FIS es suave, sin discontinuidades abruptas, y que la ley de control adopta la forma esperada según el diseño de las reglas lingüísticas.

## 3. Diseño y análisis del controlador difuso

### 3.1. Variables del FIS

En el ejemplo se emplean dos entradas y una salida:

- **Entrada 1 (level):** error de nivel (diferencia entre referencia y nivel medido).
- **Entrada 2 (rate):** razón de cambio del nivel (tendencia: subiendo/bajando).
- **Salida (valve):** acción sobre la válvula (velocidad/corrección de apertura o cierre).

### 3.2. Reglas implementadas del controlador difuso

El diseño de reglas evoluciona desde una lógica simple basada únicamente en el error de nivel hacia una estrategia que incorpora la tendencia (*rate*) para mejorar la estabilidad. Las reglas finales son:

- Si el nivel es bajo, abrir la válvula rápido.
- Si el nivel es alto, cerrar la válvula rápido.
- Si el nivel es correcto y está aumentando, cerrar la válvula lentamente.
- Si el nivel es correcto y está disminuyendo, abrir la válvula lentamente.

Esta configuración reduce significativamente las oscilaciones alrededor del setpoint al amortiguar la corrección cerca de la referencia, logrando un seguimiento estable sin comportamientos transitorios pronunciados.

### 3.3. Opciones de simulación y trazabilidad (comentario breve)

La documentación distingue dos modos de simulación para los bloques relacionados: *Interpreted execution* y *Code generation*. Asimismo, el bloque puede exponer señales internas del proceso de inferencia para depuración: entradas fuzzificadas, fuerza de disparo de reglas, salidas por regla y salida agregada. Estas señales permiten entender mejor la contribución de cada regla y localizar configuraciones conflictivas.

### 3.4. Correspondencias con `evalfis` / `evalfisOptions`

El ejemplo también describe una equivalencia conceptual entre parámetros/puertos del

## 4. Conclusiones

Se ha descrito el ejemplo `sltank` de MathWorks para simular un controlador difuso en Simulink mediante *Fuzzy Logic Toolbox*. El flujo propuesto es incremental: partir de una base de reglas mínima basada en el error de nivel y, tras observar oscilaciones alrededor del setpoint, incorporar la variable de tendencia (**rate**) para suavizar la actuación cerca de la referencia.

La variante `sltankrule` con *Ruleviewer* resulta especialmente útil para comprender y depurar la inferencia (qué reglas se activan y con qué intensidad). Finalmente, se han comentado opciones avanzadas: modos de simulación, acceso a señales internas y correspondencias con la evaluación del FIS en MATLAB, que aportan trazabilidad cuando se requiere un análisis más fino del comportamiento.