CONTROL FUZZY EN PYTHON - POSICIÓN DE UNA FAJA

Universidad Nacional de San Antonio Abad del Cusco
Escuela profesional de Ingeniería Electrónica
Inteligencia Artificial

Ing. Ruben Dario Florez Zela
Ingeniero Electrónico
Cusco, Perú
ruben.florez@unsaac.edu.pe

Davis Bremdow Salazar Roa
Estudiante de Ingeniería Electrónica
Cusco, Perú
200353@unsaac.edu.pe

***Abstract*—**
***Index Terms*—Control Fuzzy, Lógica difusa, MATLAB, Fuzzy-LogicDesigner, Mamdani**

I. Sistema

Para aplicar el control Fuzzy se tomo como referencia al sistema mostrado en la figura 1 la cual describe la composición del mismo y actuador sobre el cual se realizará la acción de control (motor DC) para fijar la faja en una posición deseada en función al error entre la posición actual y la posición establecido.

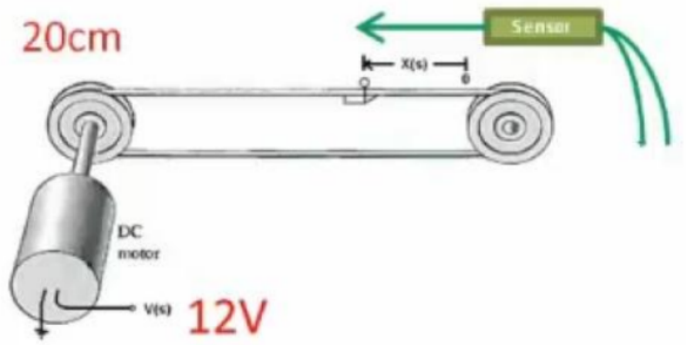

Fig. 1: Planta del sistema

II. Librerías para el controlador Fuzzy en Python

Una alternativa al entorno MATLAB es el uso de Python para simular y configurar el sistema Fuzzy, esto mediante la instalación de la librería **skfuzzy** la cual brinda objetos y funciones correspondientes para este modelado.

Una forma de poder instalar este librería es mediante el empleo del gestor de paquetes de Python**pip** esto en adición a otras librerías como **numpy, matplotlib** para visualizar gráficamente los resultados para las funciones de pertenencia y salida.

```python
import numpy as np
import skfuzzy as fuzz
# control: objeto para aplicar control
from skfuzzy import control as ctrl

import matplotlib.pyplot as plt
```

Fig. 2: Importando las librerías

En la figura 2 se muestra las librerías importadas mediante código Python y al igual que con el uso de MATLAB es posible definir los variables de entrada, funciones de membresía, reglas y comprobar el resultado mediante la puesta en marcha del controlador.

III. Variables de entrada

Mediante la librerías se definen las variables de entrada y salida mediante el uso de las funciones Antecedent y Cosequent respectivamente como se muestra en la figura 3

```python
# entrada :
error = ctrl.Antecedent(np.arange(-20, 20, 0.1), 'error')

# salida :
voltaje = ctrl.Consequent(np.arange(-12, 12, 0.1), 'voltaje')
```

Fig. 3: Variables de entrada y salida para el controlador

Siendo así que estas funciones se definen el universo de discurso, muestreo y el nombre de la variable, ocupándose para este caso los valores de **error y voltaje**.

IV. Funciones de membresía

Una vez definidas las variables de entrada se procede a definir el conjunto de términos para cada caso utilizando para ello las funciones trapezoidales y triangulares para cada caso.

```python
# Entrada
error['ENG'] = fuzz.trapmf(error.universe, [-20, -20, -10, -5])
error['ENP'] = fuzz.trimf(error.universe, [-10, -5, 0])
error['EC'] = fuzz.trimf(error.universe, [-5, 0, 5])
error['EPP'] = fuzz.trimf(error.universe, [0, 5, 10])
error['EPG'] = fuzz.trapmf(error.universe, [5, 10, 20, 20])

# Salida
voltaje['VNG'] = fuzz.trapmf(voltaje.universe, [-12, -12, -6, -3])
voltaje['VNP'] = fuzz.trimf(voltaje.universe, [-6, -3, 0])
voltaje['VC'] = fuzz.trimf(voltaje.universe, [-3, 0, 3])
voltaje['VPP'] = fuzz.trimf(voltaje.universe, [0, 3, 6])
voltaje['VPG'] = fuzz.trapmf(voltaje.universe, [3, 6, 12, 12])
```

Fig. 4: Funciones de pertenencia para las variables definidas

En la figura 4 se muestra el código en el cual se definen las funciones de pertenencia para la variable lingüística error y voltaje, siendo posible además mostrar gráficamente la definición de las mismas mediante las funciones **.view()** en cada caso y cuyas figuras se muestras en 5 y 6 respectivamente.

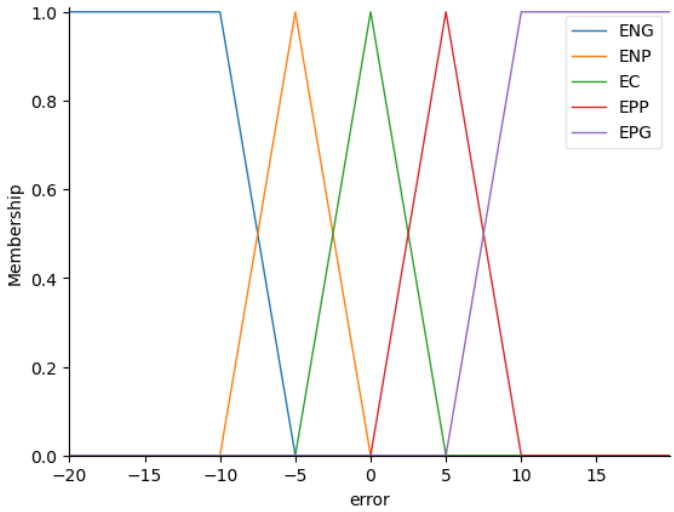

Fig. 5: Funciones de pertenencia para la variable error

V. Reglas Fuzzy

Con las funciones ya definidas es posible definir las reglas que se aplicaran para aplicar el control en la posición, generando cinco reglas en función al sistema mostrado y las cuales se muestran en la figura 7

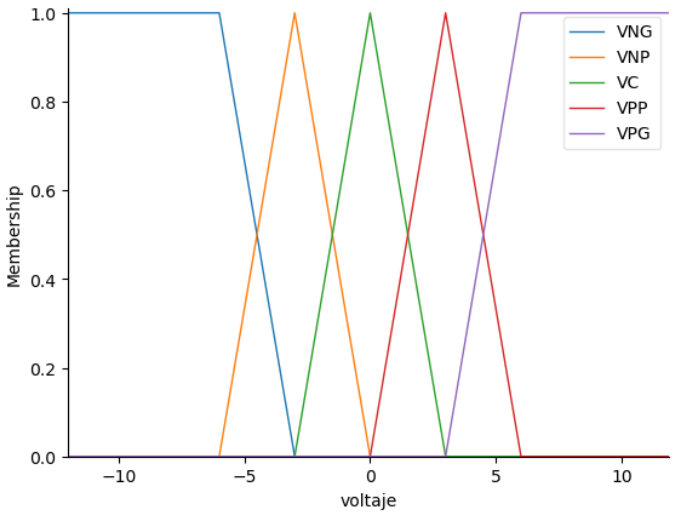

Fig. 6: Funciones de pertenencia para la variable voltaje

```python
rule1 = ctrl.Rule(error['ENG'], voltaje['VNG'])
rule2 = ctrl.Rule(error['ENP'], voltaje['VNP'])
rule3 = ctrl.Rule(error['EC'], voltaje['VC'])
rule4 = ctrl.Rule(error['EPP'], voltaje['VPP'])
rule5 = ctrl.Rule(error['EPG'], voltaje['VPG'])
```

Fig. 7: Reglas para los valores Fuzzy

Ahora estas reglas es posible aplicar el control para validar su funcionamiento, mediante la unión de reglas y aplicar la simulación mediante las funciones propuestas para **skfuzzy**, como se muestra en la figura 8

```python
# Unificando la reglas definidas
reglas_gen = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])

# clip_to_bounds: Que no se salga de los límites del antecedente y consecuente
reglas = ctrl.ControlSystemSimulation(control_system=reglas_gen, clip_to_bounds=True)
```

Fig. 8: Unión de reglas y simulación

VI. Resultados

Finalmente el sistema propuesto es un sistema en lazo abierto y para poder aplicar cierto control es necesario aplicar una retroalimentación considerando para ello que se debe utilizar una variable adicional que emule el voltaje de salida obtenido, por lo tanto esto se hace definiendo la variable **volt-retro** en conjunto con la variable **set-point** para emular tal error.

```python
set_point = 11.5
volt_retro = 11.5
# Calcular la salida en función a los parámetros definidos
reglas.input['error'] = set_point - volt_retro

reglas.compute()
output = reglas.output['voltaje']

print(output)

error.view(sim=reglas)
voltaje.view(sim=reglas)
```

Fig. 9: Simulación del sistema retroalimentado

Siendo así que para validar el funcionamiento del sistema modelado se emula inicialmente un error de 0 10 y el cual se debe corresponder con este valor de compensación para el voltaje de salida.

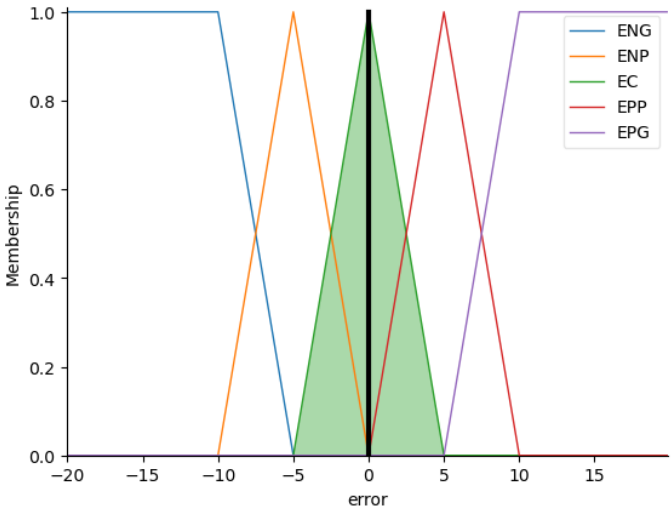

Fig. 10: Valor de entrada 0 en la variable error

Esto se aprecia en la figura 11 la cual tiene un valor de 8.351999640086417e-16 el cual se aproxima a cero, validando el código propuesto.

Finalmente para una error de entrada de -2.7 se tiene un salida aproximada de -1.841296 que se aproxima al voltaje a compensar de -2.7V mediante el modelado retroalimentado y que se muestra en la figura 12

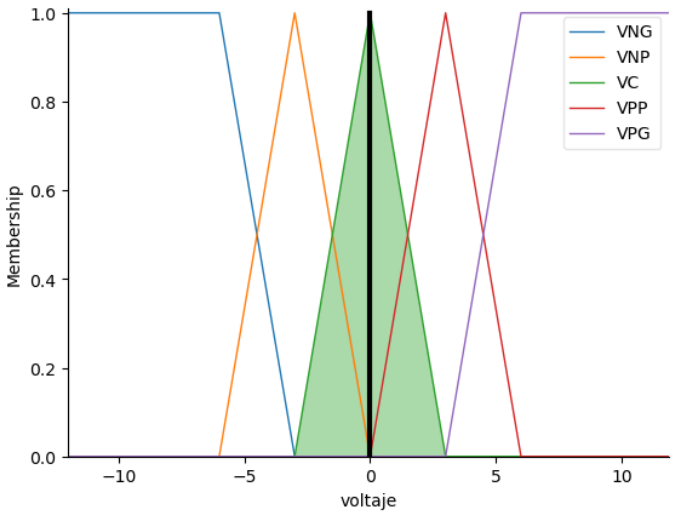

Fig. 11: Voltaje de salida o respuesta del sistema

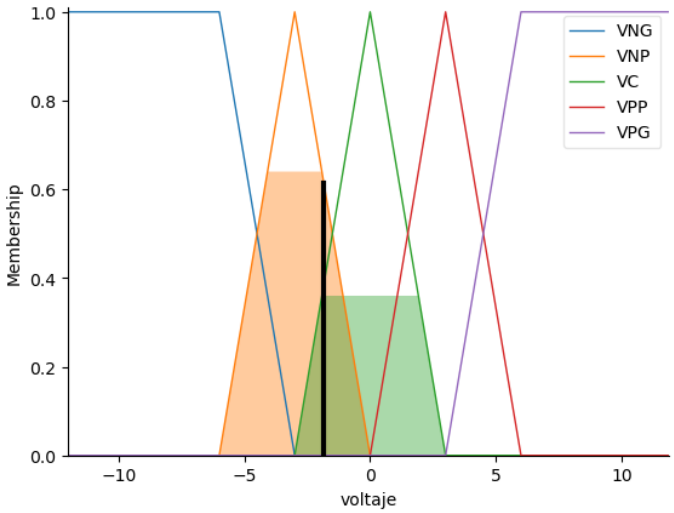

Fig. 12: Respuesta del sistema para un error de -2.7V

VII. Conclusiones

La incorporación de la retroalimentación transforma el comportamiento del sistema originalmente en lazo abierto en un esquema con capacidad de corrección dinámica del error. Al definir explícitamente la variable de error como la diferencia entre el set-point y la señal retroalimentada (volt-retro), el controlador Fuzzy actúa compensando progresivamente las desviaciones de posición de la faja.