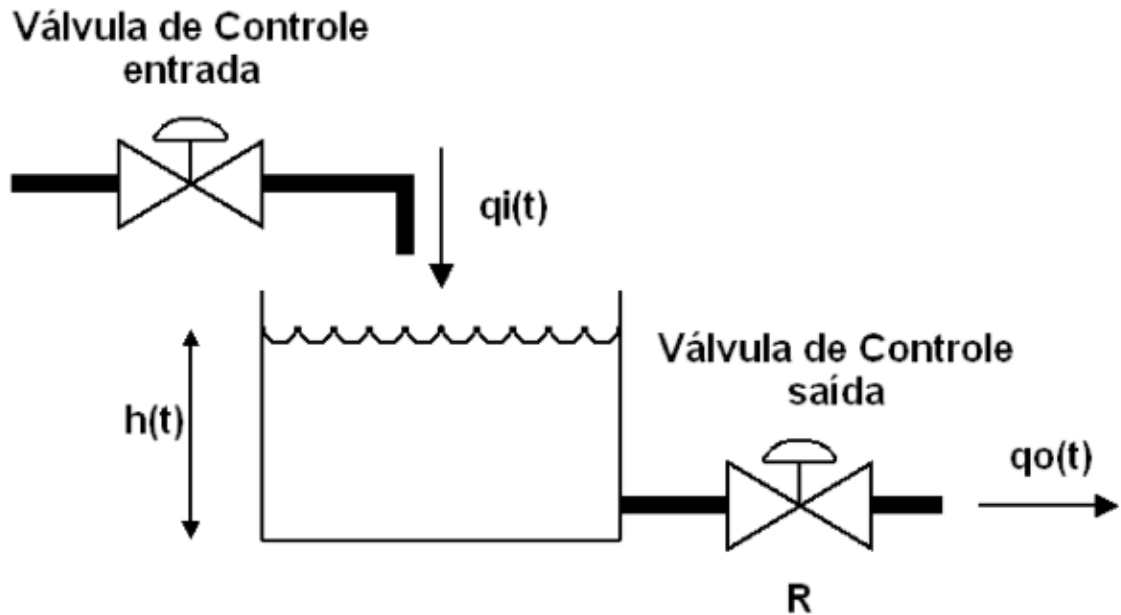


Controle Fuzzy

Autor: Jorge Vitor Gonçalves de Souza

O objetivo deste trabalho é desenvolver um controlador fuzzy para o sistema apresentado na figura abaixo.



O controlador deve manter a água em um limiar pré definido. Para isso temos o controle da válvula de saída R , que possui uma vazão q_0 sendo que: $0 \leq q_0 \leq 15 \text{ l/s}$. A entrada possui um fluxo contínuo q_i sendo que: $0 > q_i \leq 15 \text{ l/s}$. A altura h do tanque é de 100 cm e o limiar L pode assumir qualquer valor maior que 0 e menor que 80.

Bibliotecas Utilizadas

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz
import random as rd
```

Função de Pertinencia

A função de pertinência utilizada foi a trapezoidal.

```
In [2]: def trapezoidal(i, a, m, n, b):
    y = list()
    for x in i:
        y.append(max(min((x - a)/(m - a), 1, (b - x)/(b - n)), 0))
    return y
```

Simulação da Vazão de Entrada

A fim de simular a vazão da válvula de entrada no tanque foi sorteado um valor aleatório entre 0 e 15 por meio da função `random.randint()`.

```
In [3]: def valvEntrada():  
        return rd.randint(1, 15) # um número para simular a vazão de entrada
```

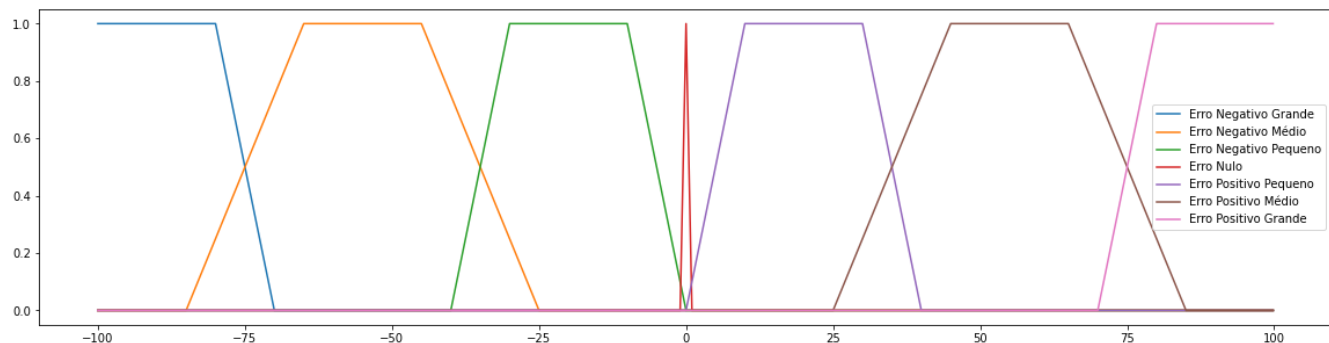
Antecedentes

Antecedentes utilizados:

- Erro Negativo Grande
- Erro Negativo Médio
- Erro Negativo Pequeno
- Erro Nulo
- Erro Positivo Pequeno
- Erro Positivo Médio
- Erro Positivo Grande

```
In [4]: erro = np.linspace(-100, 100, 1000, False) # declara o intervalo do antecedente  
ante = list()  
  
ante.append(trapezoidal(erro, -110, -100, -80, -70)) # negativo grande  
ante.append(trapezoidal(erro, -85, -65, -45, -25)) # negativo médio  
ante.append(trapezoidal(erro, -40, -30, -10, 0)) # negativo pequeno  
ante.append(trapezoidal(erro, -1, 0, 0, 1)) # nulo  
ante.append(trapezoidal(erro, 0, 10, 30, 40)) # positivo pequeno  
ante.append(trapezoidal(erro, 25, 45, 65, 85)) # positivo médio  
ante.append(trapezoidal(erro, 70, 80, 100, 110)) # positivo grande  
  
plt.figure(1, figsize=((20,5)))  
plt.plot(erro, ante[0], label="Erro Negativo Grande")  
plt.plot(erro, ante[1], label="Erro Negativo Médio")  
plt.plot(erro, ante[2], label="Erro Negativo Pequeno")  
plt.plot(erro, ante[3], label="Erro Nulo")  
plt.plot(erro, ante[4], label="Erro Positivo Pequeno")  
plt.plot(erro, ante[5], label="Erro Positivo Médio")  
plt.plot(erro, ante[6], label="Erro Positivo Grande")  
plt.legend(loc="center right")
```

Out[4]: <matplotlib.legend.Legend at 0x7fbaa23d5f40>



Consequentes

Consequentes utilizados:

- Fechar Muito
- Fechar
- Fechar Pouco
- Manter
- Abrir Pouco
- Abrir

- Abrir Muito

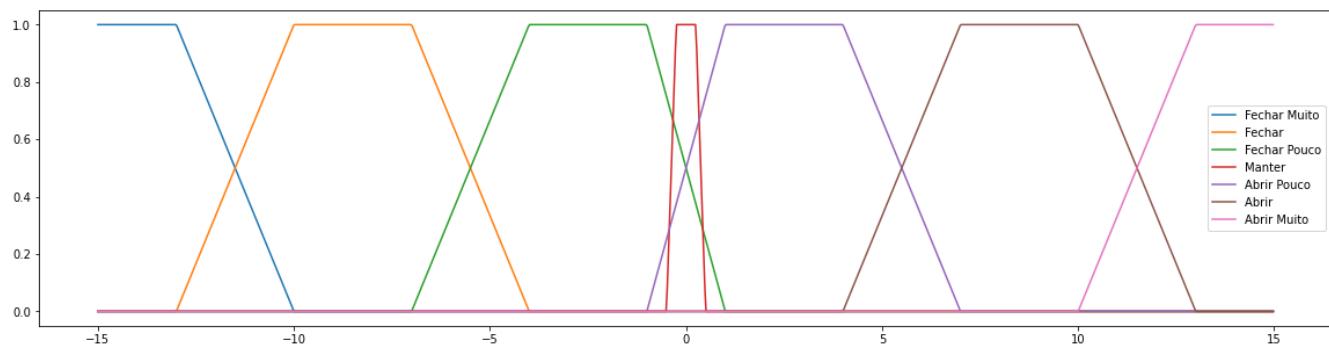
```
In [5]: saida = np.linspace(-15, 15, 1000, False) # declara o intervalo do consequente

cons = list()

cons.append(trapezoidal(saida, -19, -15, -13, -10)) # fechar muito
cons.append(trapezoidal(saida, -13, -10, -7, -4)) # fechar
cons.append(trapezoidal(saida, -7, -4, -1, 1)) # fechar pouco
cons.append(trapezoidal(saida, -0.5, -0.25, 0.25, 0.5)) # manter
cons.append(trapezoidal(saida, -1, 1, 4, 7)) # abrir pouco
cons.append(trapezoidal(saida, 4, 7, 10, 13)) # abrir
cons.append(trapezoidal(saida, 10, 13, 15, 19)) # abrir muito

plt.figure(2, figsize=((20, 5)))
plt.plot(saida, cons[0], label="Fechar Muito")
plt.plot(saida, cons[1], label="Fechar ")
plt.plot(saida, cons[2], label="Fechar Pouco")
plt.plot(saida, cons[3], label="Manter")
plt.plot(saida, cons[4], label="Abrir Pouco")
plt.plot(saida, cons[5], label="Abrir ")
plt.plot(saida, cons[6], label="Abrir Muito")
plt.legend(loc="center right")
```

Out[5]: <matplotlib.legend.Legend at 0x7fbaa16903d0>



Declaração de Variáveis Utilizadas

```
In [6]: limiar = 70 # define o limiar
deff_out = 0 # variável que guarda o valor defuzificado
nivel_at = 0 # nível atual
out_val = 0 # vazão da válvula de saída
corr = list() # lista que armazena o nível após cada iteração
erro_at = 0 # variável que guarda o erro atualizado
```

Mamdani

Regras Fuzzy Utilizadas:

- Se A é Negativo Grande, então C é Abrir Muito
- Se A é Negativo Médio, então C é Abrir
- Se A é Negativo Pequeno, então C é Abrir Pouco
- Se A é Nulo, então C é Manter
- Se A é Positivo Pequeno, então C é Fechar Pouco
- Se A é Positivo Médio, então C é Fechar
- Se A é Positivo Grande, então C é Fechar Muito

```
In [7]: for i in range(len(erro)):
        in_val= valvEntrada() # recebe o valor da vazão na válvula de ent
```

```

nível_at = nível_at + in_val      # atualiza o nível atual de acordo com os v
erro_at = int(nível_at - limiar) # atualiza o erro atual

b1 = list()
b2 = list()
b3 = list()
b4 = list()
b5 = list()
b6 = list()
b7 = list()
out = list()

for j in range(len(saida)):
    b1.append(min(ante[0][erro_at] , cons[6][j]))
    b2.append(min(ante[1][erro_at] , cons[5][j]))
    b3.append(min(ante[2][erro_at] , cons[4][j]))
    b4.append(min(ante[3][erro_at] , cons[3][j]))
    b5.append(min(ante[4][erro_at] , cons[2][j]))
    b6.append(min(ante[5][erro_at] , cons[1][j]))
    b7.append(min(ante[6][erro_at] , cons[0][j]))
    out.append(max(b1[j], b2[j], b3[j], b4[j], b5[j], b6[j], b7[j]))

deff_out = fuzz.defuzz(saida, np.array(out), 'centroid')

if(out_val + deff_out > 15):      # verificação da vazão de saída
    out_val = 15
elif(out_val + deff_out < 0):
    out_val = 0
else:
    out_val = out_val + deff_out

corr.append(nível_at)            # armazena o nível atual
nível_at = nível_at - out_val    # atualiza o nível de acordo com a vazão

```

Resultado Gráfico

```

In [8]: x = list()
        y_lim = list()

        for i in range(1000):
            x.append(i)
            y_lim.append(70)

        plt.figure(3, figsize = ((20, 5)))
        plt.yticks(np.arange(0, 100, 10))
        plt.ylim(0,100)
        plt.xlim(0, 1000)
        plt.plot(x, corr, color='m', label="Nível")
        plt.plot(x, y_lim, color = 'b', label="Limiar")
        plt.legend(loc="center right")

```

Out[8]: <matplotlib.legend.Legend at 0x7fbaa1b3d2b0>

