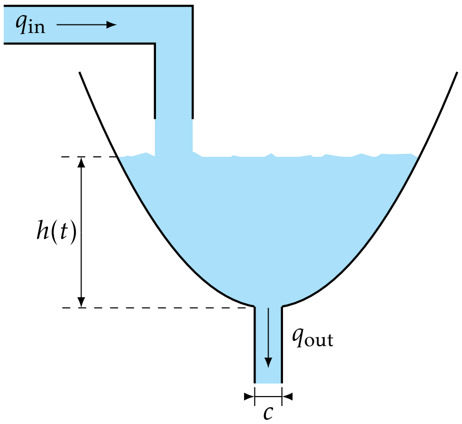
1 Controle de Nível

1.1 Modelagem



A EDO que rege o comportamento do sistema é

$$\begin{aligned}\dot{h}(t) &= \frac{q_{\text{in}} - q_{\text{out}}}{A} \\ &= \frac{16{,}998u(t) - 12{,}714h(t) - 462{,}893}{3019} \qquad \text{(1a)} \\ y(t) &= h(t) \qquad \text{(1b)}\end{aligned}$$

sendo q_{in} a vazão de entrada em cm^3/s, q_{out} a vazão de saída em cm^3/s, $u(t)$ é o sinal de controle dado em porcentagem e limitado ao intervalo $[0, 100]$, e $y(t)$ é a saída de interesse do sistema, limitada a $[0, 70]$ cm.

1.2 Ponto de operação

O ponto de operação escolhido para controle é $y_{\text{op}} = 36$ cm. Assim, o sinal de controle necessário para levar o sistema em regime permanente a este ponto pode ser encontrado, sabendo que em estado estacionário, a variação é nula, i.e, $\dot{h}(t \to \infty) = 0$. Portanto,

$$u_{\text{op}} = \frac{462{,}893 + 12{,}714 y_{\text{op}}}{16{,}998}$$

Aplicando o valor escolhido,

$$u_{\text{op}} = 54{,}16\%$$

1.3 Função transferência

A função transferência do sistema pode ser obtida primeiramente aplicando a transformada de Laplace na (1a) e, em seguida, isolando $Y(s)/U(s)$.

Admitindo-se que o sistema para $t \leq 0^-$ s estava submetido a entrada $u = u_{\text{op}}$ e, destarte a saída era, $y = y_{\text{op}}$ e , segue

$$\begin{aligned}&\mathscr{L}\left\{\dot{h}(t)\right\} = \mathscr{L}\left\{\frac{16{,}998u(t) - 12{,}714h(t) - 462{,}893}{3019}\right\} \\ &\implies H(s) = \frac{16{,}998U(s) - 12{,}714H(s)}{3019} \\ &\implies (s + 12{,}714/3019)H(s) = 16{,}998U(s) \\ &\implies G(s) = \frac{H(s)}{U(s)} = \frac{16{,}998}{s + 12{,}714/3019} \qquad (2)\end{aligned}$$

um sistema de primeira ordem sem atraso, caracterizado pela equação

$$G_{1^{\text{a}}\,\text{ordem}}(s) = \frac{ke^{-\theta s}}{\tau s + 1} \qquad (3)$$

sendo θ o tempo de atraso, k o ganho estático e τ a constante de tempo.

Para sistemas desta forma, o tempo de acomodação t_s é aproximadamente

$$t_s = 4\tau$$

Então, de (2), $\tau = 3019/12{,}714$ e por conseguinte

$$t_s = 4 \cdot \frac{3019}{12{,}714} \approx 949{,}82 \text{ s} \qquad (4)$$

2 Implementação

Para obter os degraus requeridos para o sistema, foi feito o método `degrau` que retorna um vetor com o sinal degrau. Para tanto, deve ser passado o tempo do degrau (`tstep`), o tempo final (`tf`), e o intervalo de tempo (`dt`) entre os valores do vetor de saída, necessariamente. Opcionamente, é possível escolher a amplitude do sinal de saída (`amp`), normalmente em 1 e o tempo inicial (`t0`), por definição 0.

O código utiliza do método `numpy.Heaviside(x1, x2)` provido pela biblioteca `NumPy`, já usada pelo professor no código original. Ao qual, computa

$$\text{numpy.Heaviside(x1, x2)} = \begin{cases} 0 & \text{se x1} < 0 \\ \text{x2} & \text{se x1} = 0 \\ 1 & \text{se x1} > 0 \end{cases}$$

no qual `x1` pode ser um vetor e o processo será feito elemento a elemento.

Deste modo, o método desenvolvido é

```
def degrau(tstep,tf,dt,amp=1,t0=0):
    """ Funçao que retorna um vetor
    ↪ correspondente a um degrau em
    ↪ 'tsep'
        com inicio em 't0' (igual a 0,
    ↪ normalmente) e vai ate 'tf'
```

```
        com espaçamento de 'dt'. Amplitude eh de 'amp'.

    Parameters:
    tstep (float): Tempo no qual iniciara o degrau
    tf    (float): Tempo final do vetor
    dt    (float): Intervalo, em segundos, dos valores (prefira valores de base 2).
    amp   (float): Amplitude do sinal, normalmente eh 1;
    t0    (float): Tempo inicial do vetor, normalmente eh 0.

    Returns:
    vetor[float]: Returnando o vetor u do degrau

   """
    t=np.arange(t0,tf+dt,dt)
    u=amp*np.heaviside(t-tstep,1)
    return u
```

Por fim, para usar diversas funções degraus simultaneamente basta somar os vetor de forma a produzir o comportamento desejado. Por exemplo, caso se deseja um vetor que

- comece em 0 até 3 s;
- em 3 s ele passe para 1;
- em 5 s ele passe para 3;
- o tempo deve ser espaçado em 0,25 s;
- o vetor deve ir até 100 s.

Então, com o método exposto, faz-se

```
tstep, t0, tf, dt = 3, 0, 100, 0.25
t  = np.arange(t0,tf+dt,dt)
u1 = degrau(tstep,tf,dt)
u2 = degrau(5,tf,dt,2)
plt.plot(t,u1)
plt.plot(t,u2)
plt.plot(t,u1+u2)
```