Controle digital: Controlador Via Alocação de Polos de Estados Observados.

Bernardo Bresolini * **Ester Queiroz Alvarenga** *

* *Departamento de Engenharia Mecatrônica, CEFET-MG, Campus Divinópolis (e-mail: berbresolini@gmail.com, esterqueirozalvarenga@gmail.com).*

1. INTRODUÇÃO

O conhecimento em Controle Digital por parte de engenheiros que atuarão na área controle é requerido devido à presença massiva dos controladores digitais em ambiente industrial. Por isso, o presente relatório apresenta os estudos e implementações da técnica de projeto de controladores digitais via alocação de polos de estados observados.

Para este estudo, fez-se necessário, selecionar o processo a ser trabalhado, especificar o desempenho desejado, discretizar o sistema, aplicar a técnica de projeto de controlador, obter o sinal de controle em equação a diferença, simular o processo de controle e analisar o desempenho da malha fechada.

Além desta Seção, este trabalho ainda é composto pelas Seções Sistema 1, Sistema 2 e Sistema 3, nas quais são descritos os projetos e as avaliações dos controladores para três sistemas distintos. Dentre os quais trata-se o atraso de tempo com o preditor de Smith. Ainda, tem-se a Conclusão, na qual é recapitulado sinteticamente os resultados do estudo. Nos apêndices é mostrado o código em Python3 para a implementação de cada controlador. No GitHub tem a biblioteca `ctrl.py` com as funções auxiliares desenvolvidas pelos autores usada na implementação.

2. SISTEMA 1

O primeiro processo é descrito matematicamente pela função transferência

$$G_{p1}(s) = \frac{e^{-2{,}8s}}{s+0{,}1}. \tag{1}$$

Vê-se que há um atraso de tempo de 2,8 segundos. Para tratá-lo, será utilizada a técnica de compensação de atraso, denominada preditor de Smith. Esta técnica tenta prever a saída do processo sem atraso e realimentá-la ao controlador. Para tanto, tem-se a estrutura do preditor de Smith apresentada na FIG. 1. Em que $C(z)$ é o controlador, $G_n(s)$ é o modelo da planta sem atraso de tempo, e^{-Ts} é o modelo do atraso e $G_p(s)$ é a representação do processo completo. Nesta configuração, o controlador atua sobre o processo como se não existisse o atraso na dinâmica de malha fechada.

O atraso em `Python3` deve ser tratado exclusivamente pela aproximação de Padè. Neste caso, escolheu-se uma aproximação de 8 ordem.

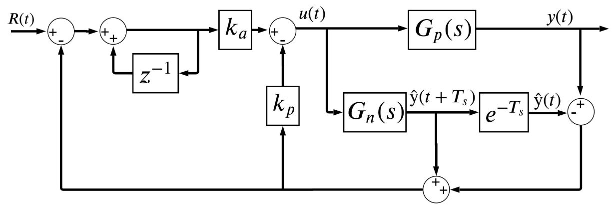

Figura 1. Topologia do Preditor de Smith

Sendo assim, para o projeto do controlador trabalha-se com

$$G_n(s) = \frac{1}{s+0{,}1} \tag{2}$$

cuja representação em espaço de estados é

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= -0{,}1\boldsymbol{x}(t) + u[k \\ y(t) &= \boldsymbol{x}(t) \end{aligned} \tag{3}$$

Para as especificações de desempenho $t_s = 20$ s e $OS = 1\% \approx 0\%$, o polo desejado de malha fechada é $p_{1,\,2} = -0{,}2 \pm 0{,}1364$ no plano s, o que implica em

$$z_{1,\,2} = 0{,}8840 \pm 0{,}0725. \tag{4}$$

note que o tempo de acomodação não computa o atraso, então o tempo real de acomodação deve estar entorno de 22,8 s.

A discretização de espaço de estados por uma taxa de amostragem de T_s é feita por

$$e^{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} T_s} = \begin{bmatrix} \bar{\boldsymbol{A}} & \bar{\boldsymbol{B}} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{5}$$

sendo $\bar{\boldsymbol{A}}$ e $\bar{\boldsymbol{B}}$ as matrizes do sistema discretizado a uma taxa de amostragem T_s. A operação matricial e^{matriz} pode ser feita utilizando o Teorema de Caley-Hamilton e em `Python3` feita por `scipy.linalg.exmp`.

Portanto, para $T_s = 0{,}6$ s, tem-se

$$\begin{aligned} x[k+1] &= 0{,}9418x[k] + 0{,}5825u[k] \\ y[k] &= x[k], \qquad T_s = 0{,}6\,\mathrm{s} \end{aligned} \tag{6}$$

Um décimo do tempo de acomodação de malha fechada é 2 s, contudo pelo critério do tempo de atraso, deve-se escolher uma taxa de amostragem menor que um quarto do atraso (0,7 s) e portanto, optou-se por $T_s = 0{,}6$ s.

Supondo $u(t) = \boldsymbol{K}_p\boldsymbol{x}(t) + k_a\xi(t)$. A forma ampliada das matrizes $\boldsymbol{A}$ e $\boldsymbol{B}$ são

$$\boldsymbol{A}_d = \begin{bmatrix} 0{,}9418 & 0 \\ -0{,}9418 & 1 \end{bmatrix}, \ \boldsymbol{B}_d = \begin{bmatrix} 0{,}5824 \\ -0{,}5824 \end{bmatrix} \tag{7}$$

E portanto, os ganhos projetados $\boldsymbol{K}_p$ e k_a são

$$\begin{bmatrix} \boldsymbol{K}_p \\ k_a \end{bmatrix} = \begin{bmatrix} 0{,}2663 \\ -0{,}03216 \end{bmatrix}. \tag{8}$$

Então, implementando a estrutura de controle proposta na FIG. 3 utilizando o código mostrado no APÊNDICE A, a resposta do sistema e a resposta enviada ao controlador são apresentadas na FIG. 2.

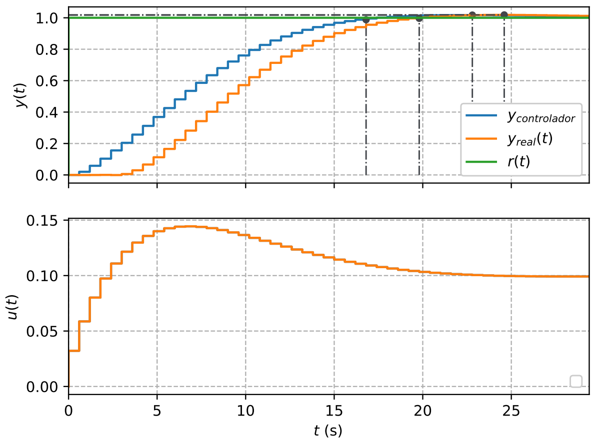

Figura 2. Resposta do sistema 1

A resposta real do sistema mostra que durante 2,8 s, mesmo o sinal de controle tenha sido aplicado, o sistema não apresentou variação na resposta por 2,8 s, mostrando que a aproximação do atraso foi bem sucedida. Além disso, para o sinal enviado ao controlador, o atraso foi removido, fazendo com que o controlador opere normalmente e o sinal de controle não se eleve muito.

Ainda, na TAB. 1 pode-se constatar que o tempo de acomodação obtido é de 3 s a mais que o valor da resposta enviada ao controlador, exatamente como previsto. No entanto, o controlador projetado atingiu a acomodação muito antes, resultado positivo. O sobressinal do sistema foi bem próximo de 1%, o valor do projeto. Sendo assim, o controlador projetado obteve êxito em controlar o sistema. Por fim, o sistema apresentou erro nulo no rastreio devido ao uso do integrador.

Tabela 1. Critérios de desempenho dos controladores

Parâmetro	y_{real}	$y_{\text{controlador}}$
t_s (s)	16,8	19,8
OS (%)	1,033	0,711
US (%)	0,0	0,077
t_r (s)	11,4	10,8

3. SISTEMA 2

A topologia de controle a ser utilizada está apresentada na FIG. 3. Ela utiliza da modelagem em espaço de estados do sistema juntamente com um observador de estados. Ainda, contém um integrador para que o sistema não apresente erro de rastreamento.

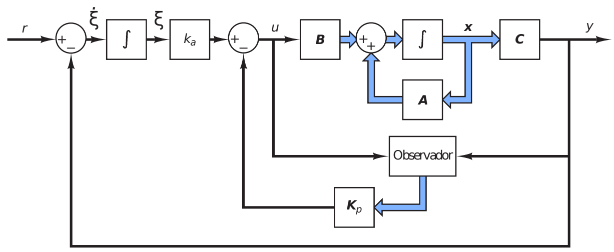

Figura 3. Topologia servossistema do tipo 1 com observador

O modelo do segundo sistema é dado, no plano s, por

$$G_2(s) = \frac{1}{(s+2)(s^2+0{,}2s+0{,}65)}$$

e sua representação em espaço de estados é

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \begin{bmatrix} -2{,}2 & 1{,}05 & -1{,}3 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} u(t) \\ y(t) &= [\,0 \ \ 0 \ \ -1\,]\, \boldsymbol{x}(t) \end{aligned} \tag{9}$$

Calculando as matrizes de controlabilidade $\mathcal{C}$ e observabilidade $\mathcal{O}$ tem-se

$$\mathcal{C} = \begin{bmatrix} -0{,}325 & 0{,}177 & 0{,}266 & 0{,}176 \\ 0{,}214 & 0{,}230 & 0{,}012 & -0{,}196 \\ -0{,}076 & -0{,}308 & -0{,}424 & -0{,}334 \\ -0{,}076 & -0{,}384 & -0{,}807 & -1{,}141 \end{bmatrix}, \tag{10}$$

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & -1 \\ -0{,}214 & 0{,}796 & -0{,}902 \\ -0{,}444 & 1{,}124 & -0{,}501 \end{bmatrix} \tag{11}$$

O sistema é controlável e observável pois as matrizes $\mathcal{C}$ e $\mathcal{O}$ têm posto completo. Sendo assim, é possível controlar e observar o sistema e a implementação da estrutura proposta na FIG. 3 é possível.

Dadas as especificações de $t_{ss} = 10\,\text{s}$ e $OS = 10\%$, os polos dominantes desejados devem se situar em

$$s_{1,\,2} = -0{,}4 \pm j0{,}5458, \tag{12a}$$
$$\implies z_{1,\,2} = 0{,}6152 \pm j0{,}3291. \tag{12b}$$

Ainda, é preciso alocar outros dois polos. Sendo assim, escolhe-se alocá-los em

$$s_3 = -4, \qquad s_4 = -5, \tag{13a}$$
$$\implies z_3 = 0{,}0111, \qquad z_4 = 0{,}0232. \tag{13b}$$

A discretização utilizando o Teorema de Caley-Hamilton e (5) para $T_s = 0{,}9\,\text{s}$ segue

$$\begin{aligned} \dot{\boldsymbol{x}}[k] &= \begin{bmatrix} -0{,}039 & 0{,}620 & -0{,}423 \\ -0{,}325 & 0{,}677 & 0{,}278 \\ 0{,}214 & -0{,}796 & 0{,}902 \end{bmatrix} \boldsymbol{x}[k] + \begin{bmatrix} -0{,}325 \\ 0{,}214 \\ -0{,}076 \end{bmatrix} u[k] \\ y[k] &= [\,0 \ \ 0 \ \ -1\,]\, \boldsymbol{x}[k], \qquad T_s = 0{,}9\,\text{s} \end{aligned} \tag{14}$$

A escolha da taxa de amostragem igual a 0,9 s é para que seu valor seja cerca de 10 vezes menor que o tempo de acomodação da malha fechada.

Supondo $u[k] = \boldsymbol{K}_p \boldsymbol{x}[k] + k_a \xi[k]$. A forma ampliada das matrizes $\boldsymbol{A}$ e $\boldsymbol{B}$ são

$$\boldsymbol{A}_d = \begin{bmatrix} -0{,}039 & 0{,}620 & -0{,}423 & 0 \\ -0{,}325 & 0{,}677 & 0{,}278 & 0 \\ 0{,}214 & -0{,}796 & 0{,}902 & 0 \\ 0{,}214 & -0{,}796 & 0{,}902 & 1 \end{bmatrix}, \ \boldsymbol{B}_d = \begin{bmatrix} -0{,}325 \\ 0{,}214 \\ -0{,}0758 \\ -0{,}0758 \end{bmatrix} \tag{15}$$

E portanto, os ganhos projetados $\boldsymbol{K}_p$ e k_a são

$$\begin{bmatrix} \boldsymbol{K}_p \\ k_a \end{bmatrix} = \begin{bmatrix} -1{,}3414 \\ 3{,}2142 \\ -1{,}1007 \\ -0{,}8329 \end{bmatrix}. \tag{16}$$

Então, implementando a estrutura de controle proposta na FIG. 3 utilizando o código mostrado no APÊNDICE B, a resposta do sistema nos primeiros 18 s é apresentada na FIG. 4.

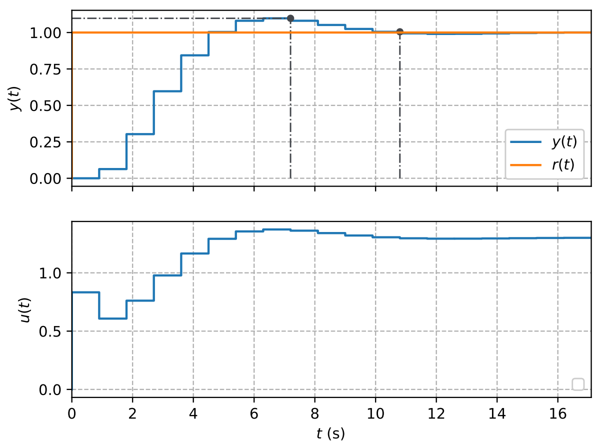

Figura 4. Resposta do sistema 2

Como mostra a TAB. 2, o controlador projetado obteve um desempenho bem próximo dos valores especificados, com um leve desvio. Diferente das abordagens anteriores, a alocação de polos via espaço de estados não adiciona zeros no sistema e portanto, o desempenho da malha fechada tende a ser mais próximo do que foi requerido ao sistema. Além disso, o sistema não apresenta erro no rastreio, justamente por causa da adição do integrador.

Tabela 2. Critérios de desempenho dos controladores

Parâmetro	Valores
t_s (s)	10,8
OS (%)	9,71
US (%)	0,0
t_r (s)	2,7

4. SISTEMA 3

A topologia de controle a ser utilizada está apresentada na FIG. 3. A mesma estrutura utilizada no controlador anterior.

O modelo do terceiro sistema é dado, no plano s, por

$$G_3(s) = \frac{5(s-5)}{(s+2)(s-2)}$$

e sua representação em espaço de estados é

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \begin{bmatrix} 0 & -4 \\ -1 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} -10 \\ 0 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} -0{,}5 & -2{,}5 \end{bmatrix} \boldsymbol{x}(t) \end{aligned} \tag{17}$$

Calculando as matrizes de controlabilidade $\mathcal{C}$ e observabilidade $\mathcal{O}$ tem-se

$$\mathcal{C} = \begin{bmatrix} -18{,}13 & -118{,}32 & -872{,}1 \\ 6{,}91 & 58{,}87 & 436{,}0 \\ 8{,}20 & 96{,}20 & 750{,}2 \end{bmatrix}, \mathcal{O} = \begin{bmatrix} -0{,}5 & -2{,}5 \\ 2{,}652 & -5{,}779 \end{bmatrix} \tag{18}$$

O sistema é controlável e observável pois as matrizes $\mathcal{C}$ e $\mathcal{O}$ têm posto completo. Sendo assim, é possível controlar e observar o sistema e a implementação da estrutura proposta na FIG. 3 é possível.

Dadas as especificações de $t_{ss} = 20\,\text{s}$ e $OS = 10\%$, os polos dominantes desejados devem se situar em

$$s_{1,2} = -0{,}2 \pm j0{,}2729, \tag{19a}$$

$$\Longrightarrow z_{1,2} = 0{,}7884 \pm j0{,}2206. \tag{19b}$$

Vale ressaltar que nas abordagens anteriores, o tempo de acomodação escolhido era $t_s = 40$ s pois era um dos menores valores capazes de estabilizar o sistema sem exigir elevados sinais de controle. No entanto, nesta abordagens, decidiu-se reduzir em 50% o tempo de acomodação de malha fechada.

Ainda, é preciso alocar outro polo. Sendo assim, escolhe-se alocá-lo em

$$s_3 = -5 \tag{20}$$

$$\Longrightarrow z_3 = 0{,}00674. \tag{21}$$

Utilizando o Teorema de Caley-Hamilton e (5) para encontrar as matrizes do sistema discreto para $T_s = 1\,\text{s}$, tem-se

$$\begin{aligned} \dot{\boldsymbol{x}}[k] &= \begin{bmatrix} 3{,}7622 & -7{,}2537 \\ -1{,}8134 & 3{,}7622 \end{bmatrix} \boldsymbol{x}[k] + \begin{bmatrix} -18{,}1343 \\ 6{,}9055 \end{bmatrix} u[k] \\ y[k] &= \begin{bmatrix} -0{,}5 & -2{,}5 \end{bmatrix} \boldsymbol{x}[k], \qquad T_s = 1\,\text{s} \end{aligned} \tag{22}$$

A taxa de amostragem escolhida teve que ser 20 vezes o tempo de acomodação, pois com a presença do observador num sistema com polos e zeros de fase não mínima foi necessário elevar a frequência de amostragem para garantir a estabilidade e a adequação da resposta aos critérios de especificação.

Supondo $u[k] = \boldsymbol{K}_p \boldsymbol{x}[k] + k_a \xi[k]$. A forma ampliada das matrizes $\boldsymbol{A}$ e $\boldsymbol{B}$ são

$$\boldsymbol{A}_d = \begin{bmatrix} 3{,}7622 & -7{,}2537 & 0 \\ -1{,}8134 & 3{,}7622 & 0 \\ -2{,}6525 & 5{,}7786 & 1 \end{bmatrix}, \ \boldsymbol{B}_d = \begin{bmatrix} -18{,}1343 \\ 6{,}9055 \\ 8{,}1966 \end{bmatrix} \tag{23}$$

E portanto, os ganhos projetados $\boldsymbol{K}_p$ e k_a são

$$\begin{bmatrix} \boldsymbol{K}_p \\ k_a \end{bmatrix} = \begin{bmatrix} -0{,}2182 \\ 0{,}4289 \\ 0{,}002688 \end{bmatrix}. \tag{24}$$

Então, implementando a estrutura de controle proposta na FIG. 3 utilizando o código mostrado no APÊNDICE C, a resposta do sistema nos primeiros 24 s é apresentada na FIG. 5. Visualmente, o sistema apresentou um *overshoot* próximo do valor de projeto e um tempo de acomodação ligeiramente menor.

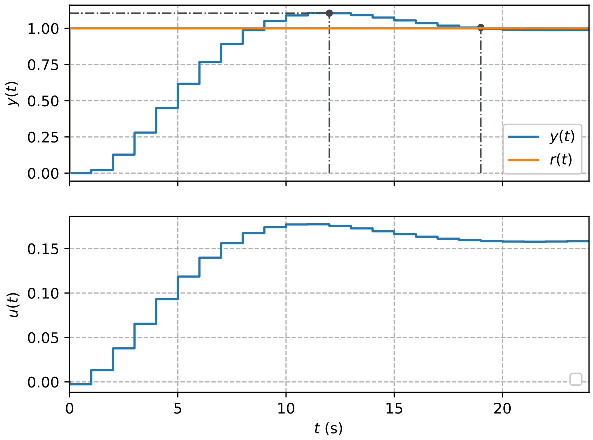

Figura 5. Resposta do sistema 3

Como mostra a TAB. 3, os critérios de desempenho foram atendidos. Vale ressaltar que o zero de fase não mínima não teve seu efeito percebido, diferente das abordagens anteriores. Ainda, mesmo com a redução em 50% no tempo de acomodação, foi possível estabilizar o sistema. Juntamente com o erro nulo em regime permanente, o controlador projetado foi de grande êxito no controle do processo.

Tabela 3. Critérios de desempenho dos controladores

Parâmetro	Valor
t_s (s)	19,0
OS (%)	11,584
US (%)	0,0
t_r (s)	5,0

5. CONCLUSÃO

A utilização da estrutura em espaço de estados para modelar e projetar controladores se mostrou bastante eficiente em todos os casos. Sem a adição de novos zeros como a abordagem polinomial e LGR, o controle do sistema se torna mais simples e fácil de analisar e tratá-lo.

A topologia adotada utiliza de um integrador garantindo erro nulo no rastreio, mas a técnica para o projeto continua sem muita alteração, facilitando em muito esta etapa.

No terceiro caso, a melhora foi muito evidente, mesmo com a redução em 50% do tempo de acomodação, foi possível alcançar os critérios de desempenho, o que não foi possível nas abordagens anteriores.

Por fim, o uso de observadores de estados é útil quando não é possível medir os estados do sistema e quando o modelo utilizado é próximo do modelo do processo, seu uso quase não impacta na performance do sistema.

```
import numpy as np
import matplotlib.pyplot as plt
import control as ct
import ctrl

plt.close('all')

# Função usada para simular o sistema continuamente
def simCont(sys, t0, tf, qnt, u, x0):
    t = np.linspace(t0,tf+t0,qnt)
    io_sys = ct.tf2io(sys)
    t, y, x = ct.input_output_response(io_sys, t, u, x0, return_x=True)
    return y[-1], x.T[-1]

N = 8  # Ordem do atraso
# Defina o modelo do sistema e a taxa de amostragem
G  = ct.tf([1],[1,0.1])
nd, dd  = ct.pade(2.8, N) # Aproximação do atraso por Padè
delay = ct.tf(nd,dd)
Gn = G*delay # Sistema com atraso

Ts = 0.6 # Taxa de amostragem
qnt = 25 # Quantidade de pontos para a simulação contínua

# Quantidade de amostras
m = 50

# Defina os critérios de desempenho
ts = 20   # Tempo de acomodação
OS = 0.01  # Sobressinal (%/100)

# Polos de malha fechada
z, w, pd = ctrl.param(ts, OS) # Polo desejado
pnd = np.array([])       # Polo não dominante

# Polos em z
P = np.exp(Ts*np.concatenate((pd,pnd)))
P.sort() # Organiza em ordem crescente

# Conversão para espaço de estados contínuo e depois discreto
sys = ct.tf2ss(G)      # Sistema em espaço de estados
sysz = sys.sample(Ts) # Sistema discretizado
n = len(G.den[0][0])-1 # Ordem do sistema

# Forma ampliada
Ad = np.vstack((np.hstack((sysz.A, np.zeros((n,1)))),
                np.hstack((-sysz.C*sysz.A, np.eye(1)))))
Bd = np.concatenate((sysz.B, -sysz.C*sysz.B))

# Controlabilidade
Ctrb = ct.ctrb(Ad,Bd)

# Verifica a controlabilidade e observabilidade
assert np.linalg.matrix_rank(Ctrb)==n+1, 'Não há controlabilidade'

# Projeto do controladdor com integrador
K  = ct.place(Ad,Bd,P) # projeto
Kp = K[0,0:-1]          # Ganho para os estados do sistema
Ka = K[0,-1]            # Ganho do integrador

# Defina as condições iniciais dos estados
x0sys = np.array([0]) # Cond. inicial do sistema
xa0   = 0  # Cond. inicial do integrador

# Pré-alocação
r  = np.ones(m)        # Referência
r[0] = 0
```

```python
u  = np.zeros_like(r)  # Sinal de controle
xa = np.zeros_like(r)  # Estado do integrador

yn = np.zeros(m+1)      # Saída do sistema
y  = np.zeros_like(yn)
xm = np.zeros((m+1,n)) # Estados do sistema
xd = np.zeros((m+1,N))
xn = np.zeros((m+1,n+N))

# Adicionando as condições iniciais
xm[0], xa[0] = x0sys, xa0

t0 = 0 # Tempo de início da simulação
t  = np.asarray(range(m))*Ts # Tempo da simulação

for k in range(m)[1:]:
    xa[k] = xa[k-1] + r[k] - y[k]
    uk = -Kp*xm[k] - Ka*xa[k]
    u[k] = uk[0,0]

    # Modelo de projeto
    ymk, xmk = simCont(G,     t0, Ts, qnt, u[k], xm[k])
    # Sistema c/ atraso
    ynk, xnk = simCont(Gn,    t0, Ts, qnt, u[k], xn[k])
    # Atraso
    ydk, xdk = simCont(delay, t0, Ts, qnt, ymk, xd[k])

    yn[k+1] = ynk # Saída do sistema
    y[k+1] = ynk + ymk - ydk # Saída p/ o controlador

    xm[k+1] = xmk # Modelo
    xd[k+1] = xdk # Atraso
    xn[k+1] = xnk # Sistema

    t0 += Ts

ylabels=['$y_{controlador}$','$y_{real}(t)$']
ctrl.PlotData(t, [y[:-1], yn[:-1]], r, [u,u], True, ylabels)
```

```python
import numpy as np
import matplotlib.pyplot as plt
import control as ct
import ctrl

plt.close('all')

# Função usada para simular o sistema continuamente
def simCont(sys, t0, tf, qnt, u, x0):
    t = np.linspace(t0,tf+t0,qnt)
    io_sys = ct.LinearIOSystem(sys)
    t, y, x = ct.input_output_response(io_sys, t, u, x0, return_x=True)
    return y[-1], x.T[-1]

# Defina o modelo do sistema e a taxa de amostragem
G  = ct.tf(1, np.convolve([1,2],[1,0.2,0.65]))
Ts = 0.9

# Quantidade de amostras
m = 20

# Defina os critérios de desempenho
ts = 10   # Tempo de acomodação
OS = 0.1  # Sobressinal (%/100)

# Polos de malha fechada
z, w, pd = ctrl.param(ts, OS) # Polo desejado
pnd = np.array([-5,-4])       # Polo não dominante

# Polos em z
P = np.exp(Ts*np.concatenate((pd,pnd)))
P.sort() # Organiza em ordem crescente

# Conversão para espaço de estados contínuo e depois discreto
sys = ct.tf2ss(G)     # Sistema em espaço de estados
sysz = sys.sample(Ts) # Sistema discretizado
n = len(G.den[0][0])-1 # Ordem do sistema

# Forma ampliada
Ad = np.vstack((np.hstack((sysz.A, np.zeros((n,1)))),
                np.hstack((-sysz.C*sysz.A, np.eye(1)))))
Bd = np.concatenate((sysz.B, -sysz.C*sysz.B))

# Controlabilidade e Observabilidade
Ctrb, Obsv = ct.ctrb(Ad,Bd), ct.obsv(sysz.A, sysz.C)

# Verifica a controlabilidade e observabilidade
assert np.linalg.matrix_rank(Ctrb)==n+1, 'Não há controlabilidade'
assert np.linalg.matrix_rank(Obsv)==n, 'Não há observabilidade'

# Projeto do controladdor com integrador
K  = ct.place(Ad,Bd,P) # projeto
Kp = K[0,0:-1]         # Ganho para os estados do sistema
Ka = K[0,-1]           # Ganho do integrador

# Polos desejados do Observador
Pobs = np.array([Ts/4, Ts/4.1, Ts/4.2])
Ke   = ct.place(sysz.A.T, sysz.C.T, Pobs) # Proj. Observador

# Defina as condições iniciais dos estados
x0sys = np.array([0, 0, 0]) # Cond. inicial do sistema
x0obs = np.array([0, 0, 0]) # Cond. inicial do observador
xa0   = 0  # Cond. inicial do integrador

# Pré-alocação
r  = np.ones(m)        # Referência
r[0] = 0
```

```python
u  = np.zeros_like(r)  # Sinal de controle
xa = np.zeros_like(r)  # Estado do integrador

y  = np.zeros(m+1)      # Saída do sistema
x  = np.zeros((m+1,n)) # Estados do sistema
xe = np.zeros_like(x)  # Estados do observador

# Adicionando as condições iniciais
x[0], xe[0], xa[0] = x0sys, x0obs, xa0

t0 = 0 # Tempo de início da simulação
t  = np.asarray(range(m))*Ts # Tempo da simulação

for k in range(m)[1:]:
    xek = xe[k].reshape((n,1))
    xa[k] = xa[k-1] + r[k] - y[k]
    uk = -Kp*xek - Ka*xa[k]
    u[k] = uk[0,0]

    y1, x1 = simCont(sys, t0, Ts, 100, u[k], x[k])
    y[k+1] = y1
    x[k+1] = x1
    xe[k+1] = (sysz.A*xek + sysz.B*uk + Ke.T*(y[k] -sysz.C*xek)).reshape(n)

    t0 += Ts

ylabels=['$y(t)$']
ctrl.PlotData(t, y[:-1], r, u, True, ylabels)
```

```python
import numpy as np
import matplotlib.pyplot as plt
import control as ct
import ctrl

plt.close('all')

# Função usada para simular o sistema continuamente
def simCont(sys, t0, tf, qnt, u, x0):
    t = np.linspace(t0,tf+t0,qnt)
    io_sys = ct.LinearIOSystem(sys)
    t, y, x = ct.input_output_response(io_sys, t, u, x0, return_x=True)
    return y[-1], x.T[-1]

# Defina o modelo do sistema e a taxa de amostragem
G  = ct.tf([5,-25], np.convolve([1,2],[1,-2]))
Ts = 1

# Quantidade de amostras
m = 25

# Defina os critérios de desempenho
ts = 20   # Tempo de acomodação
OS = 0.1  # Sobressinal (%/100)

# Polos de malha fechada
z, w, pd = ctrl.param(ts, OS) # Polo desejado
pnd = np.array([-5])       # Polo não dominante

# Polos em z
P = np.exp(Ts*np.concatenate((pd,pnd)))
P.sort() # Organiza em ordem crescente

# Conversão para espaço de estados contínuo e depois discreto
sys = ct.tf2ss(G)     # Sistema em espaço de estados
sysz = sys.sample(Ts) # Sistema discretizado
n = len(G.den[0][0])-1 # Ordem do sistema

# Forma ampliada
Ad = np.vstack((np.hstack((sysz.A, np.zeros((n,1)))),
                np.hstack((-sysz.C*sysz.A, np.eye(1)))))
Bd = np.concatenate((sysz.B, -sysz.C*sysz.B))

# Controlabilidade e Observabilidade
Ctrb, Obsv = ct.ctrb(Ad,Bd), ct.obsv(sysz.A, sysz.C)

# Verifica a controlabilidade e observabilidade
assert np.linalg.matrix_rank(Ctrb)==n+1, 'Não há controlabilidade'
assert np.linalg.matrix_rank(Obsv)==n, 'Não há observabilidade'

# Projeto do controladdor com integrador
K  = ct.place(Ad,Bd,P) # projeto
Kp = K[0,0:-1]         # Ganho para os estados do sistema
Ka = K[0,-1]           # Ganho do integrador

Acl = Ad-Bd*K
#eigAcl = np.linalg.eig(Acl)[0]
#eigAcl.sort()

# Polos desejados do Observador
Pobs = np.array([Ts/4, Ts/4.1])
Ke   = ct.place(sysz.A.T, sysz.C.T, Pobs) # Proj. Observador

# Defina as condições iniciais dos estados
x0sys = np.array([0, 0]) # Cond. inicial do sistema
x0obs = np.array([0, 0]) # Cond. inicial do observador
xa0   = 0  # Cond. inicial do integrador
```

```python

# Pré-alocação
r  = np.ones(m)         # Referência
r[0] = 0
u  = np.zeros_like(r)  # Sinal de controle
xa = np.zeros_like(r)  # Estado do integrador

y  = np.zeros(m+1)      # Saída do sistema
x  = np.zeros((m+1,n)) # Estados do sistema
xe = np.zeros_like(x)  # Estados do observador

# Adicionando as condições iniciais
x[0], xe[0], xa[0] = x0sys, x0obs, xa0

t0 = 0 # Tempo de início da simulação
t  = np.asarray(range(m))*Ts # Tempo da simulação

for k in range(m)[1:]:
    xek = xe[k].reshape((n,1))
    xa[k] = xa[k-1] + r[k] - y[k]
    uk = -Kp*xek - Ka*xa[k]
    u[k] = uk[0,0]

    y1, x1 = simCont(sys, t0, Ts, 100, u[k], x[k])
    y[k+1] = y1
    x[k+1] = x1
    xe[k+1] = (sysz.A*xek + sysz.B*uk + Ke.T*(y[k] -sysz.C*xek)).reshape(n)

    t0 += Ts

ylabels=['$y(t)$']
ctrl.PlotData(t, y[:-1], r, u, True, ylabels)
```