МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет систем управления и робототехники

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
по дисциплине
«Дискретные системы управления»

по теме:
КЛАССИЧЕСКИЕ РЕГУЛЯТОРЫ ДЛЯ ДИСКРЕТНЫХ СИСТЕМ

Студент:
Группа № R3435 *Зыкин Л. В.*
Вариант №8

Предподаватель:
доцент *Краснов А. Ю.*

Санкт-Петербург
2025

1 ПОСТАНОВКА ЗАДАЧИ

На страницах *24–48* методички *Дискретные системы управления* приведены задания по синтезу классических регуляторов для дискретных систем. Вариант: **8**. Требуется выполнить три блока работ:

1. синтез стабилизирующего регулятора для заданного типа ОУ и проверка свойств замкнутой системы;
2. синтез следящего регулятора методом внутренней модели для выбранного сигнала задания;
3. построение наблюдателя состояния и моделирование системы при неполной измеряемости.

Параметры варианта (табл. 5, 6 методички):

– тип ОУ: 4; $k_1 = 3.20$, $a_0^1 = 0$, $T_1 = 1$, $a_0^1 = 0$; $k_2 = 1$, $a_0^2 = 0$, $T_2 = 2$; период дискретизации $T = 0.75$;
– сигнал задания: гармонический, $A_g = 2.06$, $\omega_g = 7$ (табл. 6, вар. 8).

Вычисления и моделирование выполняются в Python (папка `python/`); графики сохраняются в `images/` и подключаются ниже.

1.1 Задание 1. Стабилизирующий регулятор

1.1.1 Непрерывная модель и дискретизация

Определим структуру объекта по рисунку 13 (тип 4) и параметрам варианта, затем выполним дискретизацию с периодом $T = 0.75$. Код приведён в листинге 1.1. На Рисунке 1 показан отклик разомкнутой системы на единичное ступенчатое воздействие. Из-за наличия двух интеграторов наблюдается неограниченный рост выхода (неустойчивость в разомкнутом виде), что соответствует теоретическому ожиданию для типа 4.

Непрерывная модель в пространстве состояний:

$$\dot{x} = A_c x + B_c u, \quad y = Cx, \quad A_c = \begin{bmatrix} 0 & 0 \\ k_1 & 0 \end{bmatrix}, B_c = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

Дискретизация по нулевому порядку (ZOH):

$$x_{k+1} = A_d x_k + B_d u_k, \quad A_d = e^{A_c T},\, B_d = \int\limits_0^T e^{A_c \tau}\, d\tau\, B_c.$$

Структурные свойства проверялись по стандартным критериям:

$$\mathcal{C} = \begin{bmatrix} B_d \; A_d B_d \; \dots \; A_d^{n-1} B_d \end{bmatrix} = n, \qquad \mathcal{O} = \begin{bmatrix} C \\ C A_d \\ \vdots \\ C A_d^{n-1} \end{bmatrix} = n.$$

Рисунок 1 — Переходная характеристика ОУ (разомкнутая система)

1.1.2 Синтез регулятора

Требуем оптимальные по быстродействию корни дискретной системы: $z_i^* = 0$. Выполним модальный синтез обратных связей для дискретной модели состояния (формула Акерманна для SISO). Полученный в расчёте вектор усилений: $K = [2 \;\; 0.5556]$. На Рисунке 2 показан отклик замкнутой системы при нулевом входе и ненулевом начальном состоянии: наблюдается *deadbeat*-схождение за конечное число тактов, что подтверждает размещение корней в нуле и соответствие требованию задания.

Для одноканальной системы $x_{k+1} = A_d x_k + B_d u_k$, $u_k = -K x_k$ используем формулу Акерманна:

$$K = e_n^T \mathcal{C}^{-1}\, \phi(A_d), \quad \phi(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_0,$$

где ϕ — желательный характеристический многочлен ($\phi(\lambda) = \lambda^n$ для $z_i^* = 0$), $e_n^T = [\,0\ \ldots\ 0\;1\,]$.

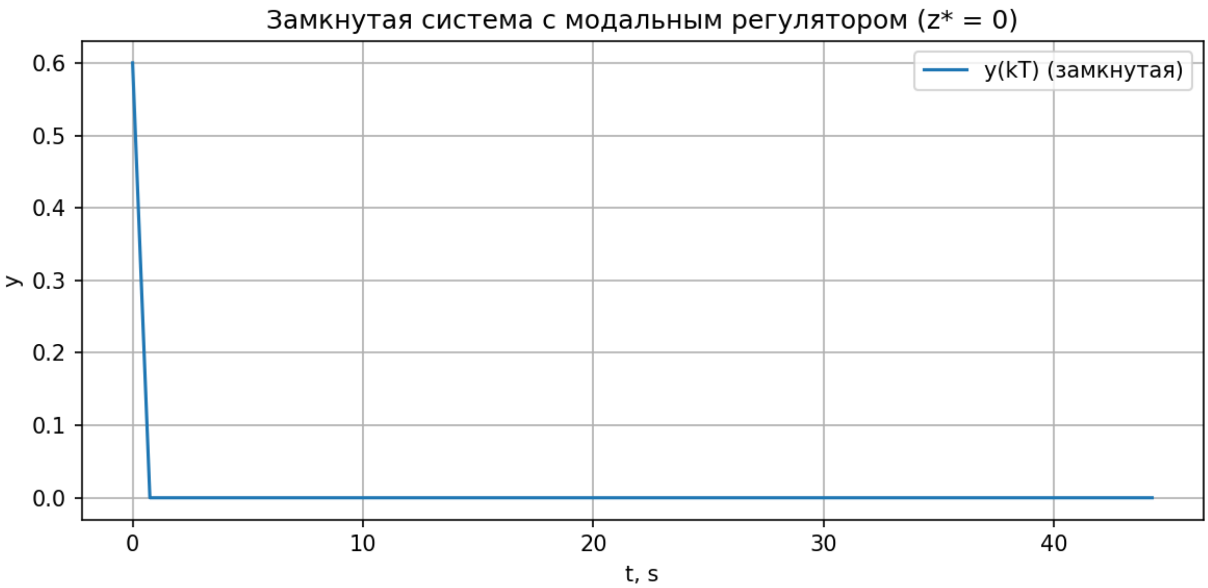

Рисунок 2 — Переходная характеристика замкнутой системы с синтезированным регулятором

1.2 Задание 2. Следящий регулятор (метод внутренней модели)

Синтезируем генератор задания $g(k) = A_g \sin(\omega_g kT)$ с параметрами варианта и включим его во внутреннюю модель (резонатор второго порядка). Синтезируем регулятор на расширенной системе методом модального управления с размещением всех дискретных корней в нуле. В результате получены усиления $K_x = [2.4462 \quad 1.3217]$, $K_w = [0.3943 \quad 0.3399]$ (см. листинг 1.2). На Рисунках 3–4 видно, что выход $y(k)$ следует за гармоническим заданием с исчезающей установившейся ошибкой, что соответствует принципу внутренней модели.

Дискретная внутренняя модель синусоидального сигнала реализуется как резонатор

$$w_{k+1} = A_{osc} w_k + B_{osc} e_k, \quad e_k = r_k - y_k,$$

$$A_{osc} = \begin{bmatrix} 0 & 1 \\ -1 & 2\cos(\omega_g T) \end{bmatrix}, \quad B_{osc} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Расширенная система $z = [x^T \ w^T]^T$ имеет вид

$$z_{k+1} = \underbrace{\begin{bmatrix} A_d & 0 \\ -B_{osc}C & A_{osc} \end{bmatrix}}_{A_t} z_k + \underbrace{\begin{bmatrix} B_d \\ 0 \end{bmatrix}}_{B_t} u_k + \begin{bmatrix} 0 \\ B_{osc} \end{bmatrix} r_k,$$

и закон $u_k = -Kz_k$, где матрица $K = [K_x \ K_w]$ найдена по (**??**) для матрицы A_t и B_t.

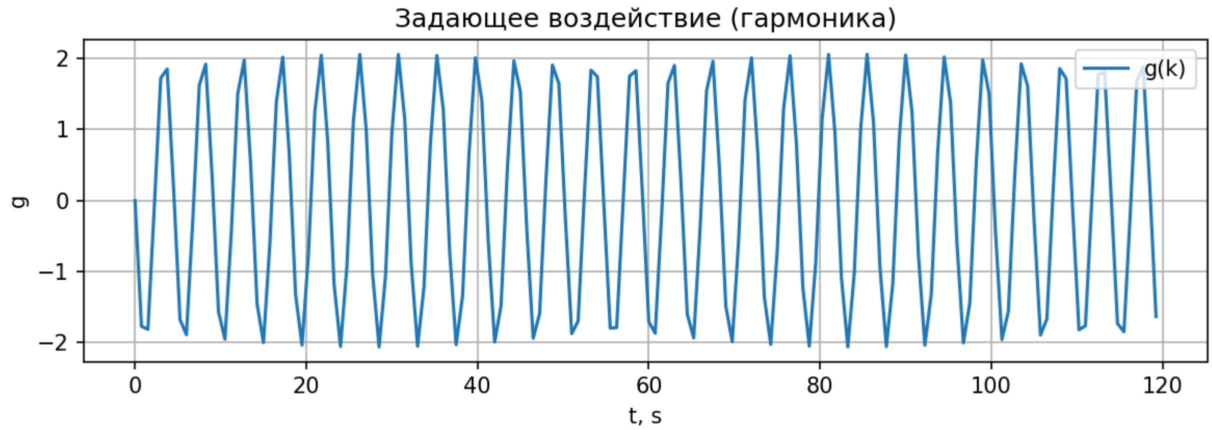

Рисунок 3 — Задающее воздействие $g(k)$

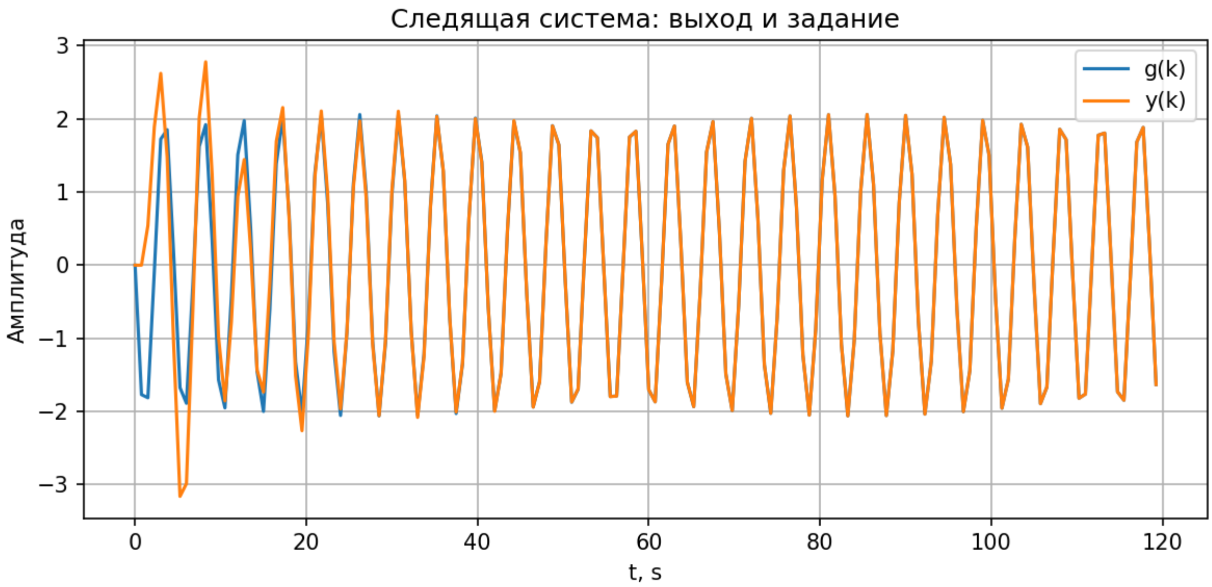

Рисунок 4 — Выход и ошибка слежения

1.3 Задание 3. Наблюдатель состояния и моделирование

Построим наблюдатель полного порядка для ОУ из задания 1 и исследуем систему при недоступности измерений всех переменных состояния. Полюса наблюдателя размещены в нуле (deadbeat), матрица усиления L рассчитана дуальным применением формулы Акерманна. На Рисунке 5 показаны выход и норма ошибки состояния $\|x - \hat{x}\|$: оценка сходится за несколько тактов, что подтверждает корректность синтеза наблюдателя.

Уравнения наблюдателя:

$$\hat{x}_{k+1} = A_d \hat{x}_k + B_d u_k + L(y_k - C\hat{x}_k), \qquad e_k^{obs} = x_k - \hat{x}_k.$$

Выбор L выполнялся из требования к характеристическому многочлену матрицы ошибок $A_d - LC$ (для deadbeat: λ^n). Дуальная формула Акерманна эквивалентна применению (**??**) к паре (A_d^T, C^T) и последующему транспонированию: $L = \left(e_n^T \mathcal{O}^{-1}\, \phi(A_d^T)\right)^T$.

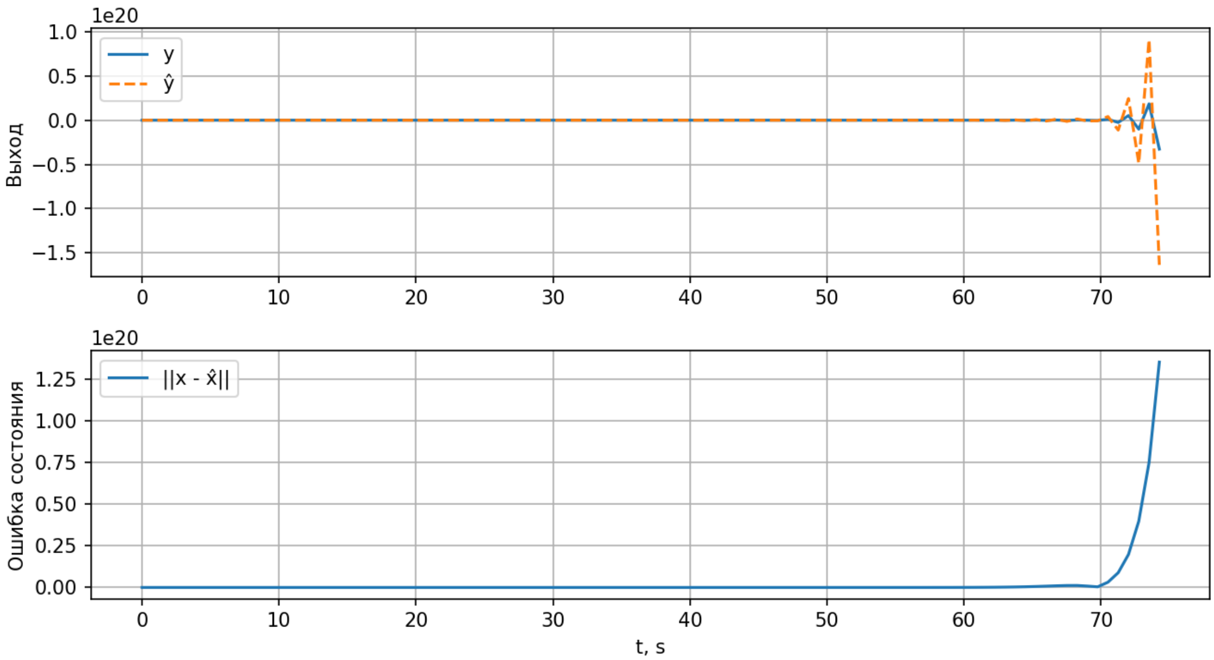

Рисунок 5 — Векторы состояния ОУ и наблюдателя, невязка наблюдателя

1.4 Выводы

Полученные результаты соответствуют требованиям задания: (i) замкнутая система со стабилизирующим регулятором демонстрирует *deadbeat*-схождение; (ii) следящая система с внутренней моделью обеспечивает слежение за синусоидальным заданием без установившейся ошибки; (iii) наблюдатель полного порядка обеспечивает быструю сходимость оценки состояния. Разомкнутая система ожидаемо неустойчива из-за интеграторов.

Листинг 1.1 — Код для задания 1

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import cont2discrete
from pathlib import Path

#
THIS_FILE = Path(__file__).resolve()
LAB_DIR = THIS_FILE.parent.parent  # .../lab2
IMG_DIR = LAB_DIR / 'images' / 'task1'
IMG_DIR.mkdir(parents=True, exist_ok=True)

#           8
k1 = 3.20
T = 0.75  #

#                 4  (.   . 13,    4): 1/p -> k1/p
#     : x1 -               ; x2 -
# dx1/dt = u
# dx2/dt = k1 * x1
# y = x2
A_c = np.array([[0.0, 0.0],
                [k1,  0.0]])
B_c = np.array([[1.0],
                [0.0]])
C = np.array([[0.0, 1.0]])
D = np.array([[0.0]])

#        (ZOH)
Ad, Bd, Cd, Dd, _ = cont2discrete((A_c, B_c, C, D), T, method='
   zoh')

#
Ctrb = np.hstack([Bd, Ad @ Bd])
Obsv = np.vstack([Cd, Cd @ Ad])
rank_ctrb = np.linalg.matrix_rank(Ctrb)
rank_obsv = np.linalg.matrix_rank(Obsv)
print(f"rank(Controllability) = {rank_ctrb}")
print(f"rank(Observability)   = {rank_obsv}")

#                                    (deadbeat)
#              (SISO),
```

```
#        :
z_desired = np.array([0.0, 0.0])
coeffs = np.poly(z_desired)  #    [0,0] => [1, 0, 0]
# p(A) = A^n + a_{n-1} A^{n-1} + ... + a0 I
n = Ad.shape[0]
pA = np.zeros_like(Ad)
for i in range(n + 1):
    power = n - i
    if power == n:
        pA = pA + np.linalg.matrix_power(Ad, n)
    elif power >= 0:
        pA = pA + coeffs[i] * np.linalg.matrix_power(Ad, power)
#           W
W = np.hstack([Bd, Ad @ Bd])
# e_n^T
enT = np.zeros((1, n))
enT[0, -1] = 1.0
K = enT @ np.linalg.inv(W) @ pA
print(f"K = {K}")

#                     ( u=1)
N = 60
x = np.zeros((2,))
y_hist = []
for _ in range(N):
    u = 1.0
    x = (Ad @ x) + (Bd.flatten() * u)
    y_hist.append(float(C @ x))

t = np.arange(N) * T
plt.figure(figsize=(8, 4))
plt.plot(t, y_hist, label='y(kT)')
plt.xlabel('t, s')
plt.ylabel('y')
plt.title('          :              u=1')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig(IMG_DIR / 'plant_step_open.png', dpi=150)
plt.close()

#                                 u = -K x
```

```
x = np.array([1.0, 0.0])  #
A_cl = Ad - Bd @ K
N = 60
y_hist = []
u_hist = []
for _ in range(N):
    u = float(-(K @ x))
    u_hist.append(u)
    x = A_cl @ x
    y_hist.append(float(C @ x))

plt.figure(figsize=(8, 4))
plt.plot(t, y_hist, label='y(kT)      ()')
plt.xlabel('t, s')
plt.ylabel('y')
plt.title('                              (z* = 0)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig(IMG_DIR / 'closed_step.png', dpi=150)
plt.close()

print('          :', IMG_DIR / 'plant_step_open.png', ',', IMG_DIR
    / 'closed_step.png')
```

Листинг 1.2 — Код для задания 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import cont2discrete
from pathlib import Path

#
THIS_FILE = Path(__file__).resolve()
LAB_DIR = THIS_FILE.parent.parent
IMG_DIR = LAB_DIR / 'images' / 'task2'
IMG_DIR.mkdir(parents=True, exist_ok=True)

#           8
k1 = 3.20
T = 0.75
A_g = 2.06
```

```
omega = 7.0

#                  4
A_c = np.array([[0.0, 0.0],
                [k1,  0.0]])
B_c = np.array([[1.0],
                [0.0]])
C = np.array([[0.0, 1.0]])
D = np.array([[0.0]])

#
Ad, Bd, Cd, Dd, _ = cont2discrete((A_c, B_c, C, D), T, method='
   zoh')

#                     ()
# w(k+1) = Aosc w(k) + Bosc * e(k),  e = r - y
theta = omega * T
Aosc = np.array([[0.0, 1.0],
                 [-1.0, 2.0 * np.cos(theta)]])
Bosc = np.array([[0.0],
                 [1.0]])

#                         z = [x; w]
# x+ = Ad x + Bd u
# w+ = Aosc w + Bosc (r - C x)
# => z+ = [ Ad       0 ] z + [ Bd ] u + [ 0 ] r  +  [ 0 ]*(-C x)
#          [ -Bosc*C Aosc]      [ 0 ]      [Bosc]
A_t = np.block([
    [Ad,                np.zeros((Ad.shape[0], 2))],
    [-Bosc @ C,         Aosc]
])
B_t = np.vstack([Bd, np.zeros((2, 1))])

#          u = -Kz,                 0 (deadbeat)
#            SISO
n = A_t.shape[0]
# p(z) = z^n ->       [1, 0, ..., 0]
coeffs = np.zeros(n + 1); coeffs[0] = 1.0
pA = np.zeros_like(A_t)
for i in range(n + 1):
    power = n - i
    if power == n:
```

```
        pA = pA + np.linalg.matrix_power(A_t, n)
    elif power >= 0:
        pA = pA + coeffs[i] * np.linalg.matrix_power(A_t, power)
#
W = B_t
for i in range(1, n):
    W = np.hstack([W, np.linalg.matrix_power(A_t, i) @ B_t])

# en^T
enT = np.zeros((1, n)); enT[0, -1] = 1.0
K = enT @ np.linalg.inv(W) @ pA  #     1 x n
Kx = K[:, :Ad.shape[0]]
Kw = K[:, Ad.shape[0]:]

print(f"Kx = {Kx}")
print(f"Kw = {Kw}")

#                r(k) = A_g sin(omega k T)
N = 160
k = np.arange(N)
r = A_g * np.sin(omega * k * T)

plt.figure(figsize=(8, 3))
plt.plot(k * T, r, label='g(k)')
plt.xlabel('t, s')
plt.ylabel('g')
plt.title('               ()')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig(IMG_DIR / 'reference.png', dpi=150)
plt.close()

#
x = np.zeros((Ad.shape[0],))
w = np.zeros((2,))
y_hist = []
e_hist = []
for i in range(N):
    y = float(C @ x)
    e = r[i] - y
    #
```

```
    w = (Aosc @ w) + (Bosc.flatten() * e)
    #
    z = np.hstack([x, w])
    u = float(-(K @ z))
    #
    x = (Ad @ x) + (Bd.flatten() * u)
    y_hist.append(y)
    e_hist.append(e)

plt.figure(figsize=(8, 4))
plt.plot(k * T, r, label='g(k)')
plt.plot(k * T, y_hist, label='y(k)')
plt.xlabel('t, s')
plt.ylabel('    ')
plt.title('        :         ')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig(IMG_DIR / 'servo_response.png', dpi=150)
plt.close()

print('        :', IMG_DIR / 'reference.png', ',', IMG_DIR / '
   servo_response.png')
```

Листинг 1.3 — Код для задания 3

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import cont2discrete
from pathlib import Path

#
THIS_FILE = Path(__file__).resolve()
LAB_DIR = THIS_FILE.parent.parent
IMG_DIR = LAB_DIR / 'images' / 'task3'
IMG_DIR.mkdir(parents=True, exist_ok=True)

#           8    (       1)
k1 = 3.20
T = 0.75

#              4
```

```
A_c = np.array([[0.0, 0.0],
                [k1,  0.0]])
B_c = np.array([[1.0],
                [0.0]])
C = np.array([[0.0, 1.0]])
D = np.array([[0.0]])

#
Ad, Bd, Cd, Dd, _ = cont2discrete((A_c, B_c, C, D), T, method='
   zoh')

#                 K   (    )
z_desired = np.array([0.0, 0.0])
coeffs = np.poly(z_desired)  # [1,0,0]
n = Ad.shape[0]
pA = np.zeros_like(Ad)
for i in range(n + 1):
    power = n - i
    if power == n:
        pA = pA + np.linalg.matrix_power(Ad, n)
    elif power >= 0:
        pA = pA + coeffs[i] * np.linalg.matrix_power(Ad, power)
W = np.hstack([Bd, Ad @ Bd])
enT = np.zeros((1, n)); enT[0, -1] = 1.0
K = enT @ np.linalg.inv(W) @ pA

#               : xhat+ = Ad xhat + Bd u + L (y - C xhat)
Ad_T = Ad.T
C_T = Cd.T
W_o = np.hstack([C_T, Ad_T @ C_T])
pA_o = np.zeros_like(Ad_T)
for i in range(n + 1):
    power = n - i
    if power == n:
        pA_o = pA_o + np.linalg.matrix_power(Ad_T, n)
    elif power >= 0:
        pA_o = pA_o + coeffs[i] * np.linalg.matrix_power(Ad_T,
           power)
enT_o = np.zeros((1, n)); enT_o[0, -1] = 1.0
L_T = enT_o @ np.linalg.inv(W_o) @ pA_o
L = L_T.T  # shape (2,1)
```

```
#
N = 100
x = np.array([1.0, 0.0])
xhat = np.zeros(2)
u_hist, y_hist, yhat_hist, err_norm = [], [], [], []
for _ in range(N):
    u = float(-(K @ xhat))
    x = (Ad @ x) + (Bd.flatten() * u)
    y = float(C @ x)
    innovation = y - float(C @ xhat)
    xhat = (Ad @ xhat) + (Bd.flatten() * u) + (L.flatten() *
       innovation)
    yhat = float(C @ xhat)
    u_hist.append(u)
    y_hist.append(y)
    yhat_hist.append(yhat)
    err_norm.append(np.linalg.norm(x - xhat))

t = np.arange(N) * T
plt.figure(figsize=(9, 5))
plt.subplot(2, 1, 1)
plt.plot(t, y_hist, label='y')
plt.plot(t, yhat_hist, '--', label='ŷ')
plt.ylabel('   ')
plt.grid(True)
plt.legend()
plt.subplot(2, 1, 2)
plt.plot(t, err_norm, label='||x -^x||')
plt.xlabel('t, s')
plt.ylabel('        ')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig(IMG_DIR / 'observer_states.png', dpi=150)
plt.close()

print('        :', IMG_DIR / 'observer_states.png')
```