Podstawy Sterowania Optymalnego
Labolatorium 3

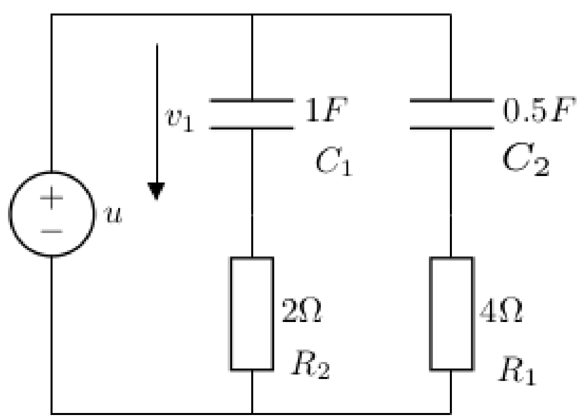
Sterowalność Układów Liniowych.

Prowadzący: mgr inż. Krzysztof Hałas
Wykonał: Ryszard Napierała

23 listopada 2021

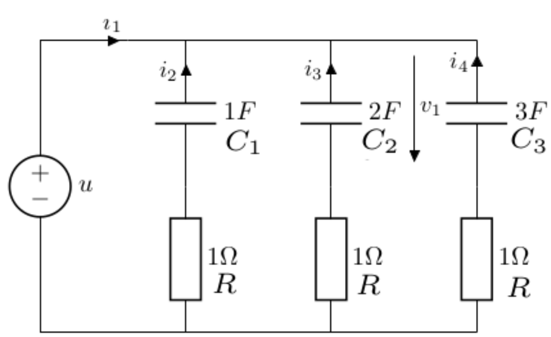
Zadanie 1

1. **Przeanalizować układy. Bez wykonywania obliczeń, na podstawie fizycznej interpretacji układów, określić sterowalność tych układów.**



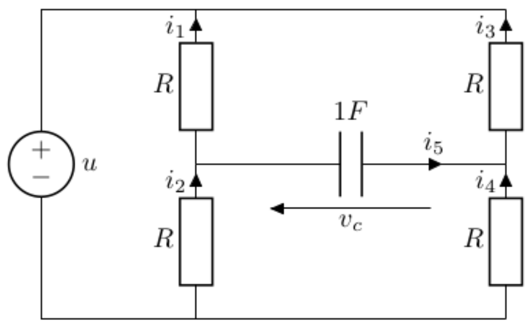
Rysunek 1:

- Układ z rysunku 1 nie jest sterowalny ponieważ $R_1C_1 = R_2C_2$.



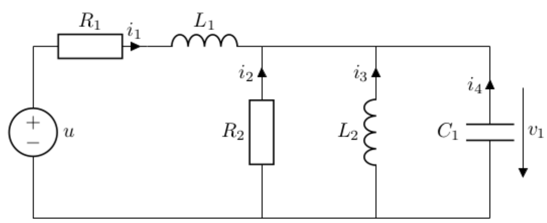
Rysunek 2:

- Układ z rysunku 2 jest sterowalny.



Rysunek 3:

- Układ z rysunku 3 nie jest sterowalny, ponieważ wszystkie rezystancje są sobie równe. Dlatego nie możemy sterować stanem kondensatora.



Rysunek 4:

- Układ z rysunku 4 jest sterowaly.

Jakie cechy układu pozwalają wnioskować o jego sterowalności?
Jeśli dla dowolnych warunków początkowych oraz dowolnej wybranej konfiguracji końcowej znaleźć można sygnał wejściowy pozwalający prze- prowadzić układ ze stanu pozcątkowego do stanu końcowego w skończonym czasie.

Jaki charakter będą miały odpowiedzi układów sterowalnych, a jaki niesterowalnych?
Charakter odpowiedzi będzie niezależny od zadanego sygnału sterującego.

2. **Wyznaczyć modele układów z Rys. 1-4. Przedstawić modele w przestrzeni zmiennych stanu.**

   - Rysunek 1

$$i_c = C\frac{d}{dt}u_c$$

$$u = i_c R + v_c$$

$$u = RC\frac{d}{dt}v_c + v_c$$

$$RC\frac{d}{dt}v_c = u - v_c$$

$$\frac{d}{dt}v_c = -\frac{1}{RC}v_c + \frac{1}{RC}u$$

$$\begin{cases} x_1 = v_{C_1} \\ x_2 = v_{C_2} \\ y_1 = x_1 \\ y_2 = x_2 \end{cases}$$

$$\begin{cases} \dot{x}_1 = -\frac{1}{R_1 C_1} x_1 + \frac{1}{R_1 C_1} u \\ \dot{x}_2 = -\frac{1}{R_2 C_2} x_2 + \frac{1}{R_2 C_2} u \end{cases}$$

$$\dot{x} = \begin{bmatrix} -\frac{1}{R_1 C_1} & 0 \\ 0 & -\frac{1}{R_2 C_2} \end{bmatrix} x + \begin{bmatrix} \frac{1}{R_1 C_1} \\ \frac{1}{R_2 C_2} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \end{bmatrix} u$$

- Rysunek 2

$$\dot{x} = \begin{bmatrix} -\frac{1}{RC_1} & 0 & 0 \\ 0 & -\frac{1}{RC_2} & 0 \\ 0 & 0 & -\frac{1}{RC_3} \end{bmatrix} x + \begin{bmatrix} \frac{1}{RC_1} \\ \frac{1}{RC_2} \\ \frac{1}{RC_3} \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \end{bmatrix} u$$

- Rysunek 3

$$i_3 + i_4 = i_1 + i_2$$

$$i_1 R = v_c + i_3 R, \;\; i_2 R = -v_c + i_4$$

$$i_5 = C \frac{d}{dt} v_c = i_2 - i_1 = i_3 - i_4$$

$$i_1 = \frac{1}{R} v_c + i_2, \;\; i_2 = -\frac{1}{R} v_c + i_4$$

$$i_5 = -\frac{1}{R} v_c - \frac{1}{R} v_c + i_4 + i_3$$

$$i_5 = -\frac{1}{R} v_c - i_5$$

$$2C \frac{d}{dt} v_c = -\frac{1}{R} v_c - \frac{1}{R} v_c$$

$$\frac{d}{dt} v_c = -\frac{1}{CR} v_c$$

$$\begin{cases} x = v_c \\ \dot{x} = -\frac{1}{CR} x \\ y = x \end{cases}$$

$$\dot{x} = \begin{bmatrix} -\frac{1}{CR} \end{bmatrix} x + \begin{bmatrix} 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 \end{bmatrix} x + \begin{bmatrix} 0 \end{bmatrix} u$$

- Rysunek 4

$$v_L = L \frac{d}{dt} i_L$$

$$v_1 = L_2 \frac{d}{dt} i_3$$

$$v_1 = i_2 R_2$$

$$v_1 = -u + i_1 R_1 + L_1 \frac{d}{dt} i_1$$

$$i_4 = -i_1 - i_2 - i_3$$

$$C_1 \frac{d}{dt} v_1 = -i_1 - \frac{1}{R_2} v_1 - i_3$$

$$\begin{cases} x_1 = i_1 \\ x_2 = i_2 \\ x_3 = v_1 \end{cases}$$

$$\dot{x} = \begin{bmatrix} -\frac{R_1}{L_1} & 0 & \frac{1}{L_1} \\ 0 & 0 & \frac{1}{L_2} \\ -\frac{1}{C_1} & -\frac{1}{C_1} & -\frac{1}{C_1 R_2} \end{bmatrix} x + \begin{bmatrix} \frac{1}{L_1} \\ 0 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \end{bmatrix} u$$

```
import numpy as np
from scipy import signal, integrate
from dataclasses import astuple, dataclass
import matplotlib.pyplot as plt

#1.2
@dataclass
class Model:
    A: np.ndarray
    B: np.ndarray
    C: np.ndarray
    D: np.ndarray

    def get(self):
        return astuple(self)

    @property
    def sys(self):
        return signal.StateSpace(*self.get())

C1 = 1
C2 = 0.5
R1 = 2
R2 = 4

rys1 = Model(
    np.array([
        [-1/(R1*C1), 0],
        [0, -1/(R2*C2)]
    ]),
    np.array([
        [1/(R1*C1)],
        [1/R2*C2]
```

```
    ]),
    np.array([[1, 0]]),
    np.array([[0]])
)

C1 = 1
C2 = 2
C3 = 3
R = 1

rys2 = Model(
    np.array([
        [-1/(R*C1), 0, 0],
        [0, -1/(R*C2), 0],
        [0, 0, -1/(R*C3)]
    ]),
    np.array([
        [1/(R*C1)],
        [1/(R*C2)],
        [1/(R*C3)]
    ]),
    np.array([[0, 0, 1]]),
    np.array([[0]])
)

C = 1
R = 1

rys3 = Model(
    np.array([[-1/(C*R)]]),
    np.array([[0]]),
    np.array([[1]]),
    np.array([[0]])
)

C1 = 2
R1 = 2
R2 = 1
L1 = 0.5
L2 = 1

rys4 = Model(
    np.array([
        [-R1/L1, 0, 1/L1],
        [0, 0, 1/L2],
        [-1/C1, -1/C1, -1/(C1*R2)]
    ]),
    np.array([
        [1/L1],
```

```
            [0],
            [0]
        ]),
        np.array([[0, 0, 1]]),
        np.array([[0]])
)
```

Czy możliwe jest uzyskanie innych modeli w przestrzeni zmiennych stanu?
Jest możliwe uzyskanie innych modeli w przestrzeni zmiennych stanu.

3. **Wyznaczyć macierze Kalmana i formalnie zbadać sterowalność układów (wykorzystać na przykład *numpy.linalg.matrix_rank*).**

```
def kalman(rys: Model):
    K = np.zeros_like(rys.A)
    for i in range(rys.A.shape[0]):
        K[:, i:i+1] = np.linalg.matrix_power(rys.A, i)@rys.B
    print(K)
    rank = np.linalg.matrix_rank(K)
    print(f'rank={rank}')
    print(f'Uklad{"" if rank==K.shape[0] else " nie"} sterowalny\n')

print('rys1:')
kalman(rys1)
print('rys2:')
kalman(rys2)
print('rys3:')
kalman(rys3)
print('rys4:')
kalman(rys4)
```

Output:

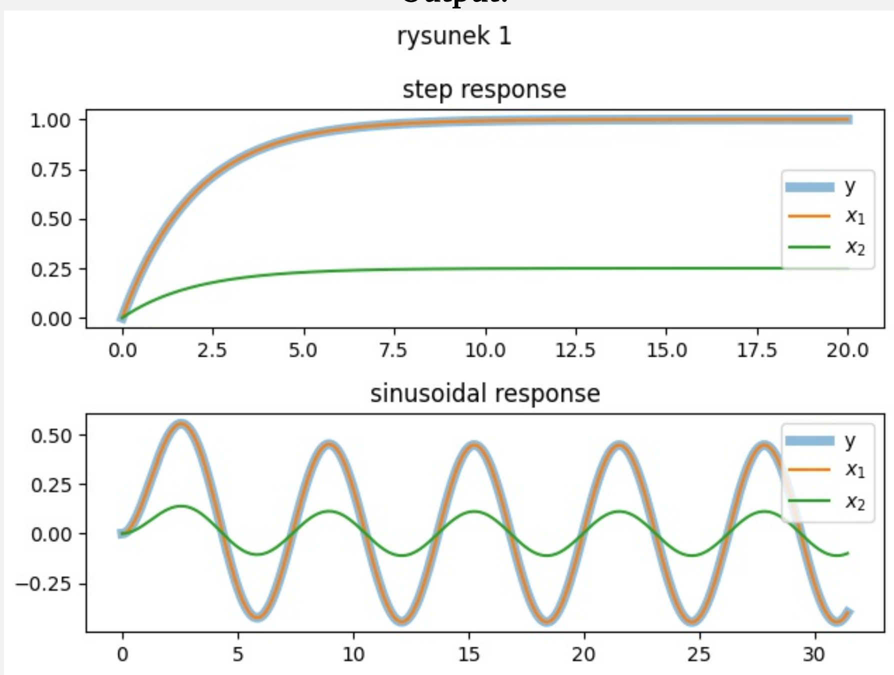
```
rys1:
[[ 0.5     -0.25   ]
 [ 0.125   -0.0625]]
rank=1
Uklad nie sterowalny

rys2:
[[ 1.          -1.           1.         ]
 [ 0.5         -0.25         0.125      ]
 [ 0.33333333 -0.11111111   0.03703704]]
rank=3
Uklad sterowalny

rys3:
[[0.]]
rank=0
Uklad nie sterowalny

rys4:
[[ 2.   -8.   30. ]
 [ 0.    0.   -1. ]
 [ 0.   -1.    4.5]]
rank=3
Uklad sterowalny
```

4. **Zaimplementować przedstawione układy i zbadać ich odpowiedzi na wybrane wymuszenia (np. wymuszenie skokowe, wymuszenie sinusoidalne, wykorzystać na przykład *scipy.signal.lsim* lub *scipy.signal.lsim2*).**

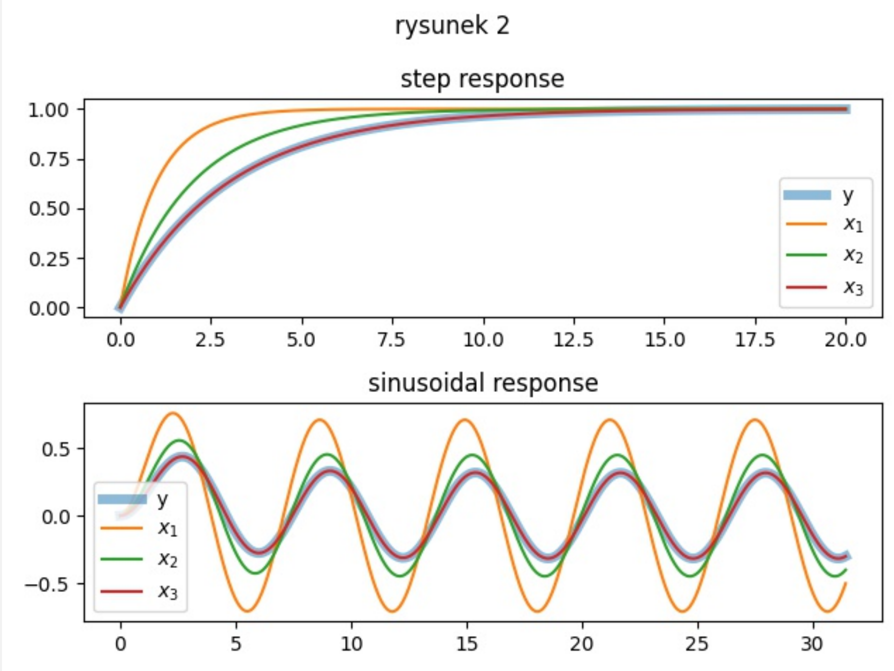
```
def plot(rys:Model, ax:plt.Axes, u:np.ndarray, t:np.ndarray, title:str):
    t, y, x = signal.lsim2(rys.sys, u, t)
    ax.plot(t, y, linewidth=5, alpha=0.5)
    states = x.shape[1]
    ax.set_title(title)
    for i in range(states):
        ax.plot(t, x.T[i])
    ax.legend(['y'] + [f'$x_{s+1}$' for s in range(states)])

def sim(rys: Model, title: str):
    fig, (ax1, ax2) = plt.subplots(2, 1)
    fig.suptitle(title)

    t = np.linspace(0, 20, 1000)
    u = np.ones((1000,))
    plot(rys, ax1, u, t, 'step response')

    t = np.linspace(0, 10*np.pi, 1000)
```

```
    u = np.sin(t)
    plot(rys, ax2, u, t, 'sinusoidal response')

    fig.tight_layout()
    plt.show()

sim(rys1, 'rysunek 1')
sim(rys2, 'rysunek 2')
sim(rys3, 'rysunek 3')
sim(rys4, 'rysunek 4')
```

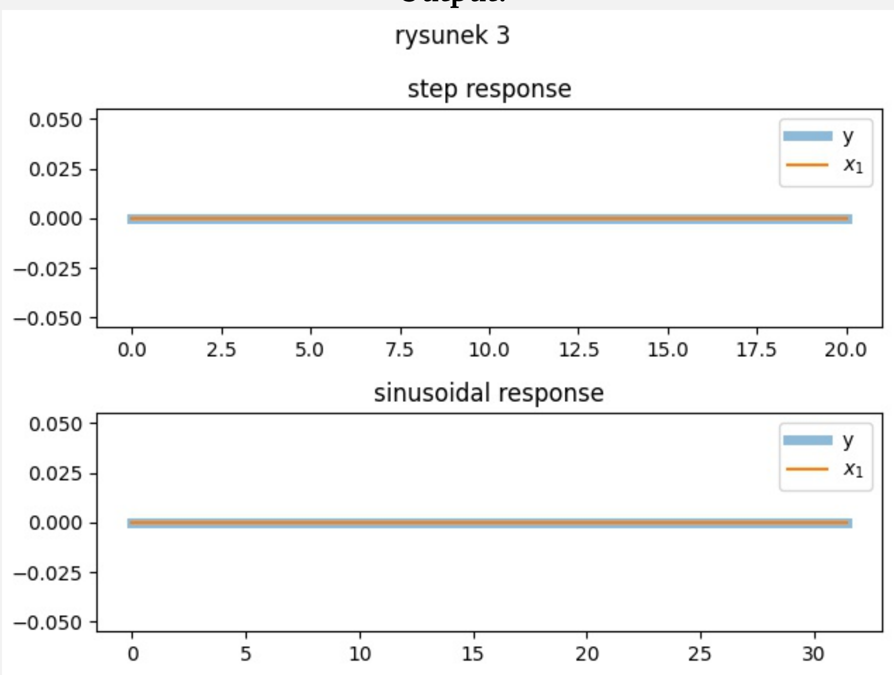
Output:



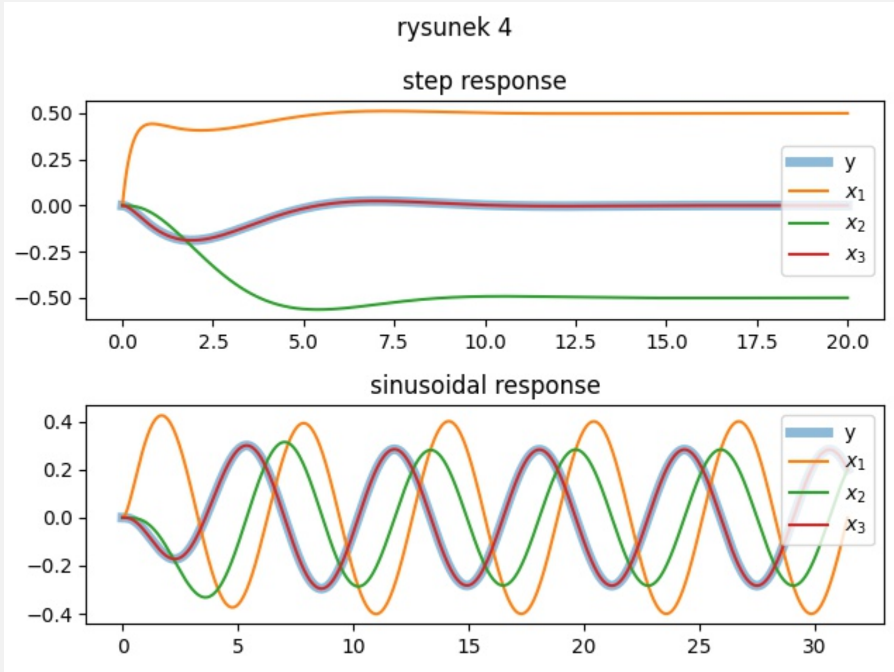
Output:



Output:



Output:



Czy uzyskane przebiegi potwierdzają wcześniejsze przypuszczenia?
Odpowiedzi dla układu z rysunku 1 nie potwierdzają wcześniejszych przypuszczeń.

Jakie są różnice między różnymi funkcjami symulującymi układy w Pythonie?
Funkcje *scipy.signal.lsim* i *scipy.signal.lsim2* dla układów liniowych, z tym że *lsim2* nie wymaga podawania wektorów czasu i wymuszenia. W przypadku nie podania wektora wymuszenia, domyślne wymuszenie to zero. W przypadku nie podania wektora czasu, domyśly wektor czasu to *numpy.linspace(0, 10, 101)*. Obie te funkcje przyjmują układ liniowy w postaci klasy *lti* lub interpretacji układu w postaci *tuple* 2,3, lub 4 elementowej.
Funkcja *scipy.integrate.solve_ivp* różni się od powyższych funkcji głównie tym, że reprezentacja układu podawana jest w postaci ręcznie zdefiniowanej funkcji różniczkowej.
Funkcje *scipy.signal.step* oraz *scipy.signal.impulse* przyjmują układ liniowy w postaci klasy *lti* lub interpretacji układu w postaci *tuple* 2,3, lub 4 elementowej. Obliczają kolejno odpowiedź skokową i impulsową układu. Jeżeli wektor czasu nie zostanie podany, jest on automatycznie wyliczany.

5. **Dla układów sterowalnych z poprzednich zadań wyznaczyć postać sterowalną równań dynamiki. Do wykonania obliczeń wykorzystać funkcjonalności Pythona.**

```
def to_controllable_canonical(rys: Model) -> Model:
    [num], den = signal.ss2tf(*rys.get())
    A = np.zeros_like(rys.A)
    A[:-1,1:] = np.eye(rys.A.shape[0] - 1)
    A[-1] = den[::-1][:-1]*-1
    B = np.zeros_like(rys.B)
    B[-1] = 1
    C = np.array([num[::-1][:-1]])
    return Model(A,B,C,np.array([[0]]))

rys2cc = to_controllable_canonical(rys2)
rys4cc = to_controllable_canonical(rys4)
```

```python

print('rys2:\n', rys2cc)
print('rys4:\n', rys4cc)
```

Output:

```
rys2:
 Model(A=array([[ 0.        ,  1.        ,  0.        ],
       [ 0.        ,  0.        ,  1.        ],
       [-0.16666667, -1.        , -1.83333333]]),
       B=array([[0.], [0.], [1.]]),
       C=array([[0.16666667, 0.5       , 0.33333333]]),
       D=array([[0]]))
rys4:
 Model(A=array([[ 0. ,  1. ,  0. ],
       [ 0. ,  0. ,  1. ],
       [-2. , -3.5, -4.5]]),
       B=array([[0.], [0.], [1.]]),
       C=array([[ 6.66133815e-16, -1.00000000e+00, -3.55271368e-15]]),
       D=array([[0]]))
```

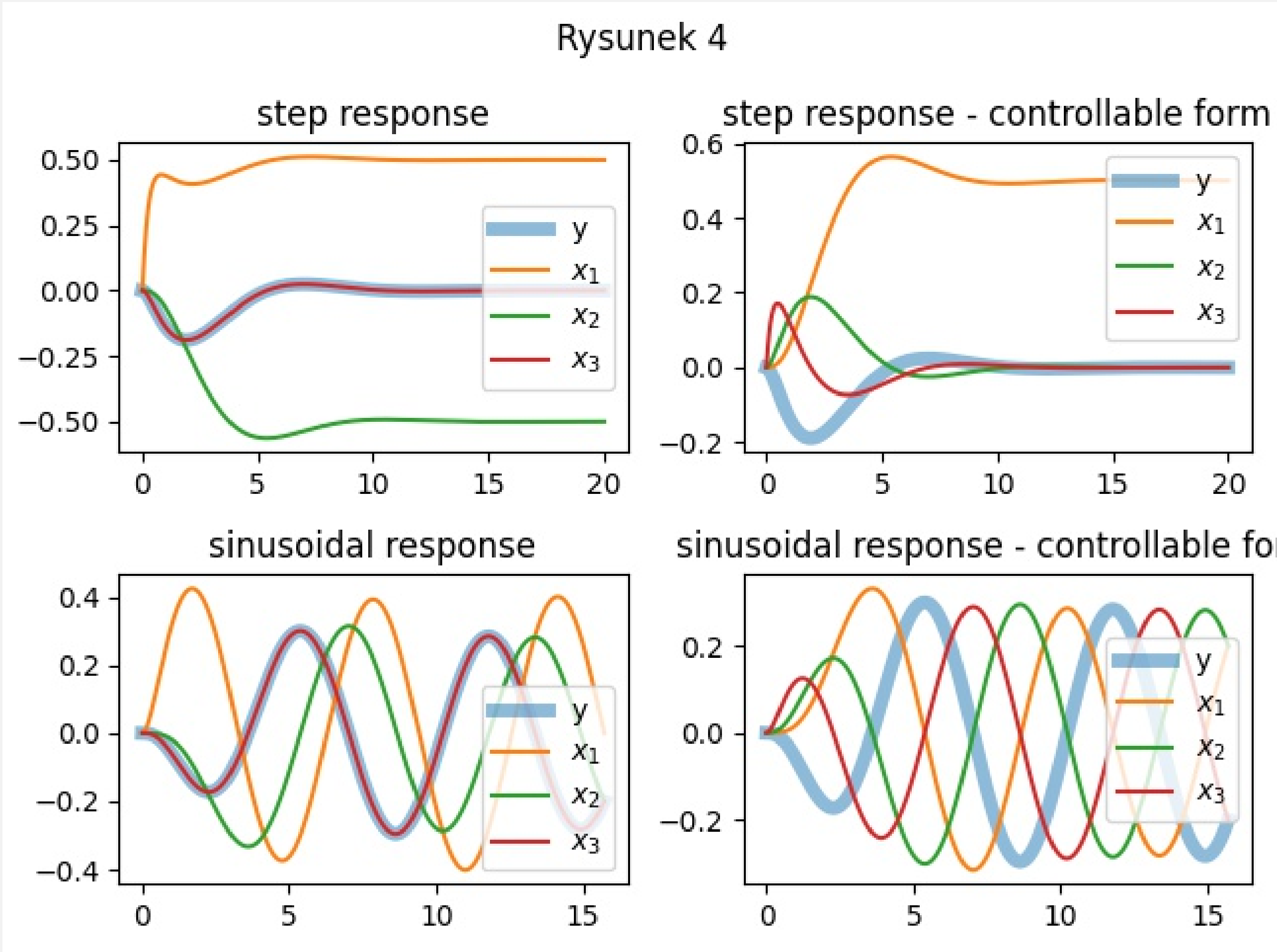
Czy dla układów niesterowalnych można wyznaczyć postać normalną regulatorową dynamiki? Dlaczego?

Można wyznaczyć postać normalną regulatorową dla układów niesterowalnych, jeżeli stopień licznika jest mniejszy niż stopień mianownika w transmitancji. Ponieważ jest to jedynie inna reprezentacja tego samego układu.

6. **Dla wybranego przypadku przeprowadzić symulację obiektu dla obu reprezentacji (tj. wyznaczonej samodzielnie w zadaniu 1.2 oraz normalnej regulatorowej).**

```python
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
fig.suptitle('Rysunek 4')

t = np.linspace(0, 20, 1000)
u = np.ones((1000,))
plot(rys4, ax1, u, t, 'step response')
plot(rys4cc, ax2, u, t, 'step response - controllable form')

t = np.linspace(0, 5*np.pi, 1000)
u = np.sin(t)
plot(rys4, ax3, u, t, 'sinusoidal response')
plot(rys4cc, ax4, u, t, 'sinusoidal response - controllable form')

fig.tight_layout()
plt.show()
```

Output:



Czy obie reprezentacje są równoważne?
Tak

Czy przebiegi dla obu reprezentacji są jednakowe? Dlaczego? Jakie będzie miało to znaczenie przy projektowaniu układu regulacji w oparciu o postać sterowalną?
Przebiegi nie są jednakowe dla obu reprezentacji. Jest to związane z tym że inaczej zostały przyjęte równania stanów przejściowych. Jednak wyjście układu jest jednakowe dla obu reprezentacji. Przy projektowaniu układu regulacji w oparciu o postać sterowalną mamy tą zaletę że każdy stan jest pochodną stanu poprzedniego, a wyjście jest sumą stanów pośrednich pomnożonych przez współczynniki, co sprowadza dostrajanie układu do regulacji współczynników.