Teoria Sterowania (639B-ARAUT-MEP-TSTUZ)
Elementy sterowania optymalnego

(opracował: M.T., ostatnia modyfikacja: 18 października 2024)

1 Elementy optymalizacji matematycznej

1.1 Postać standardowa zadania optymalizacji

[1] Rozpatrujemy zadanie optymalizacji w standardowej postaci

$$\begin{array}{ll} \text{minimize} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, p \end{array} \tag{1}$$

gdzie $\boldsymbol{x} \in \mathbb{R}^n$ oznacza wektor zmniennych optymalizacyjnych (decyzyjnych), funkcję $f_0 : \mathbb{R}^n \to \mathbb{R}$ nazywamy funkcją celu (ang. *objective function*), funkcje $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ nazywamy funkcjami ograniczeń (więzów) nierównościowych, zaś funkcje $h_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, p$ nazywamy funkcjami ograniczeń (więzów) równościowych. Zadanie (1) oznacza poszukiwanie wektora, który minimalizuje funkcję celu, spośród wszystkich $\boldsymbol{x}$ spełniających warunki $f_i(\boldsymbol{x}) \leq 0$, $i = 1, \ldots, m$ oraz $h_i(\boldsymbol{x}) = 0$, $i = 1, \ldots, p$. Zbiór punktów $\boldsymbol{x}$, dla których funkcja celu oraz funkcje ograniczeń są określone, nazywamy dziedziną $\mathcal{D}$ zadania (1)

$$\mathcal{D} \equiv \bigcap_{i=0}^{m} \mathbf{dom} f_i \cap \bigcap_{i=1}^{p} \mathbf{dom} h_i.$$

Będziemy zakładać, że $\mathcal{D} \neq \emptyset$ Punkt $\boldsymbol{x} \in \mathcal{D}$ nazywamy dopuszczalnym (ang. *feasible*), jeśli spełnia ograniczenia $f_i(\boldsymbol{x}) = 0$, $i = 1, \ldots, m$ oraz $g_i(\boldsymbol{x}) \leq 0$, $i = 1, \ldots, p$. Zadanie (1) będziemy nazywać *dopuszczalnym* (ang. *feasible*)jeśli istnieje co najmniej jeden dopuszczalny punkt $\boldsymbol{x}$. W innym przypadku, zadanie (1) będziemy nazywać *niedopuszczalnym* (ang. *infeasible*)
Wartość optymalną dla (1) oznaczaną p^* definiujemy

$$p^* \equiv \inf_{\boldsymbol{x} \in \mathcal{D}} \{ f_0(\boldsymbol{x}) \mid f_i(\boldsymbol{x}) = 0 \text{ dla } i = 1, \ldots, m, \ h_i(\boldsymbol{x}) \leq 0 \text{ dla } i = 1, \ldots, p \}.$$

Należy podkreślić, że wartość optymalna p^* nie ma związku z liczbą więzów równościowych p. Będziemy dopuszczać przyjmowanie przez p^* wartości $\pm\infty$. Zgodnie z konwencją $\inf \emptyset = \infty$, w związku z czym dla zadań niedopuszczalnych (ang. *infeasible*) piszemy $p^* = \infty$. Jeśli natomiast dla danego zadania istnieje ciąg punktów dopuszczalnych $\boldsymbol{x}_k$ taki, że $f_0(\boldsymbol{x}_k) \overset{k\to\infty}{\to} -\infty$, to piszemy $p^* = -\infty$ i mówimy, że zadanie jest nieograniczone z dołu (ang. *unbounded below*). Punkt $\boldsymbol{x}^*$ nazywamy *punktem optymalnym* (ang. *optimal point*), lub rozwiązaniem zadania (1) jeśli $\boldsymbol{x}^*$ jest dopuszczalny oraz $f_0(\boldsymbol{x}^*) = p^*$. Zbiór wszystkich punktów optymalnych nazywamy *zbiorem optymalnym* (ang. *optimal set*)

$$\Omega_{\text{optimal}} = \{ \boldsymbol{x} \mid f_i(\boldsymbol{x}) \leq 0, \ i = 1, \ldots, m, \ h_i(\boldsymbol{x}) = 0, \ i = 1, \ldots, p, \ f_0(\boldsymbol{x}) = p^* \}$$

Jeżeli dla zadania (1) zbiór optymalny $\Omega_{\text{optimal}} \neq \emptyset$ to mówimy, że wartość optymalna (funkcji celu) jest osiągana (ang. *attained* lub *achieved*), zaś zadanie optymalizacji jest rozwiązywalne (ang. *solvable*). Jeśli $\Omega_{\text{optimal}} = \emptyset$ to mówimy, że wartosc optymalna nie jest osiągana, sytuacja taka zawsze ma miejsce, jeśli zadanie jest nieograniczone z dołu.

1.2 Zadanie optymalizacji wypukłej w postaci standardowej

Zadanie optymalizacji wypukłej jest postaci

$$\begin{array}{ll} \text{minimize} & f_0(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = b_i, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, p \end{array} \tag{2}$$

gdzie f_0 oraz $h_0, \ldots, h_p$ są funkcjami wypukłymi. Standardowa postać zadania optymalizacji wypukłej charakteryzuje się tym, że funkcja celu oraz funkcje ograniczeń nierównościowych są wypukłe, zaś funkcje ograniczeń równościowych są afiniczne. Zbiór dopuszczalny dla zadania optymalizacji wypukłej jest zbiorem wypukłym, co wynika stąd, że jest on przecięciem wypukłej dziedziny zadania

$$\mathcal{D} = \bigcap_{i=0}^{m} \mathbf{dom} h_i,$$

z p wypukłymi zbiorami subwarstwicowymi (ang. *sublevel sets*) $\{ \boldsymbol{x} \mid h_i(\boldsymbol{x}) \leq 0 \}$ i m hiperpłaszczyznami $\{ \boldsymbol{x} \mid \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = b_i \}$. Bez straty ogólności rozważań można założyć, że $\boldsymbol{a}_i \neq 0$ dla $i = 1, \ldots, m$

Uwaga 1. Zadanie

$$\begin{array}{ll} \text{maximize} & f_0(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = b_i, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, p \end{array} \tag{3}$$

dla *wklęsłej* (ang. *concave*) funckji celu f_0, będziemy traktować jako zadanie optymalizacji wypukłej, ponieważ jest ono w oczywisty sposób równoważne zadaniu optymalizacji wypukłej

$$\begin{array}{ll} \text{minimize} & -f_0(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = b_i, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, p \end{array} \tag{4}$$

Uwaga 2. Będziemy przyjmmować, że zadaniem optymalizacji wypukłej (lub krócej zadaniem wypukłym) jest zadanie optymalizacji wypukłej funkcji celu z wypukłymi funkcjami ograniczeń nierównościowych i *afinicznymi* funkcjami ograniczeń równościowych. Innymi słowy, aby nazwać zadanie wypukłym nie wystarczy aby polegało ono na minimalizacji funkcji wypukłej przy wypukłym zbiorze dopuszczalnym, lecz muszą dodatkowo być spełnione warunnki dotyczące postaci funkcji ograniczeń.

1.3 Przykład - programowanie liniowe

Programowanie liniowe (ang. *linear programming*, w skrócie LP) stanowi niezwykle ważny obszar optymalizacji matematycznej zarówno z teoretycznego jak i praktycznego punktu widzenia [1, 2]. LP jest również ważnym zagadnieniem algorytmicznym [2].

Przykład 1. Rozwiązać zadanie programowania liniowego

$$\begin{aligned} \text{minimize} \quad & x_1 + \frac{1}{2}x_2 \\ \text{subject to} \quad & x_1 + x_2 \le 2 \\ & x_1 + \frac{1}{4}x_2 \le 1 \\ & x_1 - x_2 \le 2 \\ & \frac{1}{4}x_1 + x_2 \ge -1 \\ & x_1 + x_2 \ge 1 \\ & -x_1 + x_2 \le 2 \\ & x_1 + \frac{1}{4}x_2 = \frac{1}{2} \\ & -1 \le x_1 \le \frac{3}{2} \\ & -\frac{1}{2} \le x_2 \le \frac{5}{4} \end{aligned} \tag{5}$$

Zadanie (5) jest stosunkowo proste i można znaleźć rozwiązanie „ręcznie", $x_1^\star = 1/3$, $x_2^\star = 2/3$, jednak w ogólnym przypadku korzysta się z odpowiedniego oprogramowania. Omówimy krótko dwa podejścia, jedno związane ze środowiskiem Matlab, drugie z językiem Python. Zanim przejdziemy do konkretnych rozwiązań, zauważmy, że zadanie (5) jest równoważne zadaniu

$$\begin{aligned} \text{minimize} \quad & c^{\mathrm{T}}x \\ \text{subject to} \quad & Ax \le b \\ & A_{\mathrm{eq}}x = b_{\mathrm{eq}} \\ & x_{\mathrm{LB}} \le x \le x_{\mathrm{UB}} \end{aligned} \tag{6}$$

gdzie

$$A = \begin{bmatrix} 1 & 1 \\ 1 & \frac{1}{4} \\ 1 & -1 \\ -\frac{1}{4} & -1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ -1 \\ 2 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix} \tag{7}$$

$$A_{\mathrm{eq}} = \begin{bmatrix} 1 & \frac{1}{4} \end{bmatrix}, \quad b_{\mathrm{eq}} = \begin{bmatrix} \frac{1}{2} \end{bmatrix} \tag{8}$$

$$x_{\mathrm{LB}} = \begin{bmatrix} -1 \\ -\frac{1}{2} \end{bmatrix}, \quad x_{\mathrm{UB}} = \begin{bmatrix} \frac{3}{2} \\ \frac{5}{4} \end{bmatrix} \tag{9}$$

Zadanie (6) jest w oczywisty sposób równoważne zadaniu

$$\begin{aligned} \text{minimize} \quad & c^{\mathrm{T}}x \\ \text{subject to} \quad & \begin{bmatrix} A \\ I \\ -I \end{bmatrix} x \le \begin{bmatrix} b \\ x_{\mathrm{UB}} \\ -x_{\mathrm{LB}} \end{bmatrix} \\ & A_{\mathrm{eq}}x = b_{\mathrm{eq}} \end{aligned} \tag{10}$$

Zatem możemy zdefiniować

$$\tilde{A} = \begin{bmatrix} A \\ I \\ -I \end{bmatrix}, \quad \tilde{b} = \begin{bmatrix} b \\ x_{\mathrm{UB}} \\ -x_{\mathrm{LB}} \end{bmatrix}, \tag{11}$$

i rozwiązać zadanie

$$\begin{aligned} \text{maximize} \quad & c^{\mathrm{T}}x \\ \text{subject to} \quad & \tilde{A}x \le \tilde{b} \\ & A_{\mathrm{eq}}x = b_{\mathrm{eq}} \end{aligned} \tag{12}$$

Korzystając ze środowiska Matlab mamy do dyspozycji bibliotekę Optimization Toolbox, oraz dodatkowy, bardzo wygodny pakiet CVX. Jeśli chodzi o język Python, to możliwości jest wiele, dla naszych potrzeb skorzystamy z modułów CVXPY i CVXOPT.

1.3.1 Matlab

```
clear all
close all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x1 = optimvar('x1','LowerBound',-1,'UpperBound',1.5);
x2 = optimvar('x2','LowerBound',-1/2,'UpperBound',1.25);
p = optimproblem('Objective',x1 + x2/2,'ObjectiveSense','min');
p.Constraints.c1 = x1 + x2 <= 2;
p.Constraints.c2 = x1 + x2/4 <= 1;
p.Constraints.c3 = x1 - x2 <= 2;
p.Constraints.c4 = x1/4 + x2 >= -1;
p.Constraints.c5 = x1 + x2 >= 1;
p.Constraints.c6 = -x1 + x2 <= 2;
p.Constraints.c7 = x1 + x2/4 == 1/2;
options = optimoptions('linprog','Algorithm','dual-simplex','OptimalityTolerance',1e-10);
% options = optimoptions('linprog','Algorithm','interior-point','OptimalityTolerance',1e-10);
sol = solve(p,'Options',options);
x1 = sol.x1;
x2 = sol.x2;
disp('-------------------------------')
disp('optimal solution - first method')
disp([x1,x2])
disp('-------------------------------')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
A = [1.0, 1.0;
    1.0, 0.25;
    1.0, -1.0;
    -0.25, -1.0;
    -1.0 -1.0;
    -1.0 1.0];
b = [2; 1; 2; 1; -1; 2];
c = [1; 1.5];
Aeq = [1 0.25];
beq = 0.5;
LB = [-1; -0.5];
UB = [1.5; 1.25];
x = optimvar('x',2,1,'LowerBound',LB,'UpperBound',UB);
p = optimproblem('Objective',c'*x,'ObjectiveSense','min');
p.Constraints.c1 = A*x <= b;
p.Constraints.c2 = Aeq*x == beq;
options = optimoptions('linprog','Algorithm','dual-simplex','OptimalityTolerance',1e-10);
% options = optimoptions('linprog','Algorithm','interior-point','OptimalityTolerance',1e-10);
sol = solve(p,'Options',options);
x = sol.x;
disp('--------------------------------')
disp('optimal solution - second method')
disp(x)
disp('--------------------------------')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A = [1.0, 1.0;
    1.0, 0.25;
    1.0, -1.0;
    -0.25, -1.0;
    -1.0 -1.0;
    -1.0 1.0];
b = [2; 1; 2; 1; -1; 2];
c = [1; 1.5];
Aeq = [1 0.25];
beq = 0.5;
LB = [-1; -0.5];
UB = [1.5; 1.25];
xOpt = linprog(c,A,b,Aeq,beq,LB,UB);
disp('--------------------------------')
disp('optimal solution - third method')
disp(xOpt)
disp('--------------------------------')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cvx_begin
    variables x1 x2
    x1 + x2 <= 2
    x1 + x2/4 <= 1
    x1 - x2 <= 2
    x1/4 + x2 >= -1
    x1 + x2 >= 1
    -x1 + x2 <= 2
    x1 + x2/4 == 1/2
    -1 <= x1 <= 1.5
    -1/2 <= x2 <= 1.25
    minimize ( x1 + 0.5*x2 )
cvx_end
disp('--------------------------------')
disp('optimal solution - fourth method')
disp([x1,x2])
disp('--------------------------------')
```

1.3.2 Python - moduł CVXPY

```
import numpy as np
import cvxpy as cp

################################
## method 1
################################
x1 = cp.Variable()
x2 = cp.Variable()

objective = cp.Minimize(x1 + 0.5*x2)
constraints = [ x1 + x2 <= 2,
                x1 + x2/4 <= 1,
                x1 - x2 <= 2,
                x1/4 + x2 >= -1,
                x1 + x2 >= 1,
                -x1 + x2 <= 2,
                x1 + x2/4 == 1/2,
                -1 <= x1,
                x1 <= 1.5,
                -1/2 <= x2,
                x2 <= 1.25 ]
p1 = cp.Problem(objective, constraints)
p1.solve()
print("--------------")
print(x1.value)
print(x2.value)
print("--------------")

################################
## method 2
################################
n = 2
x = cp.Variable(n)
A = np.array([[1.0,1.0],[1.0,0.25],[1.0,-1.0],[-0.25,-1.0],[-1.0,-1.0],[-1.0,1.0]])
b = np.array([2.0, 1.0, 2.0, 1.0, -1.0, 2.0])
c = np.array([1, 1.5])
Aeq = np.array([1.0, 0.25])
beq = np.array([0.5])
LB = np.array([-1.0, -0.5])
UB = np.array([1.5, 1.25])
objective = cp.Minimize(c.T @ x)
constraints = [ A @ x <= b, Aeq @ x == beq, x <= UB, x >= LB ]
p2 = cp.Problem(objective, constraints)
p2.solve()
print("\n")
print("--------------")
print(x.value)
print("--------------")

################################
## method 3
################################
n = 2
x = cp.Variable(n)
A = np.array([[1.0,1.0],[1.0,0.25],[1.0,-1.0],[-0.25,-1.0],[-1.0,-1.0],[-1.0,1.0]])
b = np.array([2.0, 1.0, 2.0, 1.0, -1.0, 2.0])
c = np.array([1, 1.5])
Aeq = np.array([1.0, 0.25])
beq = np.array([0.5])
LB = np.array([-1.0, -0.5])
UB = np.array([1.5, 1.25])
AA = np.vstack((A,np.eye(n),-np.eye(n)))
bb = np.hstack((b,UB,-LB))
```

```
objective = cp.Minimize(c.T @ x)
constraints = [ AA @ x <= bb, Aeq @ x == beq ]
p3 = cp.Problem(objective, constraints)
p3.solve()
print("\n")
print("--------------")
print(x.value)
print("--------------")
```

1.3.3 Python - moduł CVXOPT

```
import numpy as np
import cvxopt as co
from cvxopt.modeling import  variable, op, dot, matrix

################################
## method 1
################################
x1 = variable()
x2 = variable()

c1 = ( x1 + x2 <= 2 )
c2 = ( x1 + x2/4 <= 1 )
c3 = ( x1 - x2 <= 2 )
c4 = ( x1/4 + x2 >= -1 )
c5 = ( x1 + x2 >= 1 )
c6 = ( -x1 + x2 <= 2 )
c7 = ( x1 + x2/4 == 1/2 )
c8 = ( -1 <= x1 <= 1.5 )
c9 = ( -1/2 <= x2 <= 1.25 )

p1 = op(x1 + 0.5*x2, [c1,c2,c3,c4,c5,c6,c7,c8,c9])
p1.solve()
##print("\n", p1.objective.value())
print("--------------")
print(x1.value)
print(x2.value)
print("--------------")
##print("\n", c1.multiplier.value)

################################
## method 2
################################
n = 2
x = variable(n)
A = np.array([[1.0,1.0],[1.0,0.25],[1.0,-1.0],[-0.25,-1.0],[-1.0,-1.0],[-1.0,1.0]])
A = matrix(A)
b = matrix([2.0, 1.0, 2.0, 1.0, -1.0, 2.0])
c = matrix([1, 1.5])
Aeq = matrix([[1.0], [0.25]])
beq = matrix([0.5])
LB = matrix([-1.0, -0.5])
UB = matrix([1.5, 1.25])

c1 = ( A*x <= b )
c2 = ( Aeq*x == beq)
c3 = ( x <= UB )
c4 = ( x >= LB )
p2 = op(dot(c,x), [c1,c2,c3,c4])
p2.solve()
##print("\n", p2.objective.value())
print("\n")
print("--------------")
print(x.value)
```

```
print("--------------")

##############################
## method 3
##############################
n = 2
A = matrix([[1.0, 1.0, 1.0, -0.25, -1.0, -1.0], [1.0, 0.25, -1.0, -1.0, -1.0, 1.0]])
b = matrix([2.0, 1.0, 2.0, 1.0, -1.0, 2.0]);
c = matrix([1, 1.5])
Aeq = matrix([[1.0], [0.25]])
beq = matrix([0.5])
LB = matrix([-1.0, -0.5])
UB = matrix([1.5, 1.25])

AA = matrix(np.vstack((A,np.eye(n),-np.eye(n))))
bb = matrix(np.vstack((b,UB,-LB)))
sol = co.solvers.lp(c,AA,bb,Aeq,beq)
print("\n", sol['x'][0], sol['x'][1])
```

2 Elementy sterowania optymalnego dla układów SISO)

Weźmy pod uwagę dyskretny układ dynamiczny SISO, opisany równaniami

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k \tag{13a}$$
$$y_k = \boldsymbol{C}\boldsymbol{x}_k \tag{13b}$$

gdzie $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times 1}$, $\boldsymbol{C} \in \mathbb{R}^{1\times n}$, $\boldsymbol{x}_k \in \mathbb{R}^n$, $k = 0, 1, \ldots$

Problem wyznaczenia sygnału sterujacego dla osiągnięcia zadanego stanu docelowego można sformułować jako zadanie optymalizacji [1, 3, 4], które polega na znalezieniu ciągu sterowań u_k, $k = k_\mathrm{i}, \ldots, k_\mathrm{f} - 1$, który przeprowadzi układ z określonego stanu początkowego $\boldsymbol{x}_{k_\mathrm{i}} = \boldsymbol{x}_\mathrm{i}$ do zadanego stanu końcowego $\boldsymbol{x}_{k_\mathrm{f}} = \boldsymbol{x}_\mathrm{f}$, dla $k_\mathrm{f} > k_\mathrm{i}$, $k_\mathrm{f} \in \mathbb{N}$. W dalszym ciągu bez straty ogólności rozważań będziemy zakładać, że $k_\mathrm{i} = 0$. Zgodnie z równaniem stanu (13) dla $k_\mathrm{i} = 0$ mamy

$$\begin{aligned}
\boldsymbol{x}_1 &= \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}u_0 \\
\boldsymbol{x}_2 &= \boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}u_1 \\
&= \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}u_0) + \boldsymbol{B}u_1 \\
&= \boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{B}u_0 + \boldsymbol{B}u_1 \\
\boldsymbol{x}_3 &= \boldsymbol{A}\boldsymbol{x}_2 + \boldsymbol{B}u_2 \\
&= \boldsymbol{A}(\boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{B}u_0 + \boldsymbol{B}u_1) + \boldsymbol{B}u_2 \\
&= \boldsymbol{A}^3\boldsymbol{x}_0 + \boldsymbol{A}^2\boldsymbol{B}u_0 + \boldsymbol{A}\boldsymbol{B}u_1 + \boldsymbol{B}u_2 \\
&\vdots
\end{aligned} \tag{14}$$

Ogólnie, rozwiązaniem równania (13) dla $k > 0$ jest

$$\boldsymbol{x}_k = \underbrace{\boldsymbol{A}^k\boldsymbol{x}_0}_{\substack{\text{składowa}\\ \text{swobodna}}} + \underbrace{\sum_{j=0}^{k-1} \boldsymbol{A}^{k-j-1}\boldsymbol{B}u_j}_{\text{składowa wymuszona}} . \tag{15}$$

W języku teorii obwodów składową swobodną nazywa się składową nieustaloną, natomiast składową wymuszoną – składową ustaloną. Składowa swobodna zależy od warunku początkowego, nie zależny natomiast od wymuszenia. Składowa wymuszona zależy od wymuszenia (tzn. ciągu sterowań), nie zależny natomiast od warunku początkowego. Podstawiając $k = k_\mathrm{f} = N$ (N będziemy nazywać horyzontem czasowym) otrzymujemy

$$\begin{aligned}
\boldsymbol{x}_N &= \boldsymbol{A}^N\boldsymbol{x}_0 + \sum_{j=0}^{N-1} \boldsymbol{A}^{N-1+j}\boldsymbol{B}u_j \\
&= \boldsymbol{A}^N\boldsymbol{x}_0 + \begin{bmatrix}\boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{A}^1\boldsymbol{B} & \boldsymbol{B}\end{bmatrix}\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-2} \\ u_{N-1}\end{bmatrix}
\end{aligned} \tag{16}$$

Wprowadzając oznaczenia

$$\boldsymbol{R} \equiv \begin{bmatrix}\boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B}\end{bmatrix} \tag{17}$$

oraz

$$\boldsymbol{\mu} \equiv \begin{bmatrix}u_0 & u_1 & \ldots & u_{N-2} & u_{N-1}\end{bmatrix}^\mathrm{T} \tag{18}$$

możemy napisać zależność (16) zwięźle

$$\boldsymbol{x}_\mathrm{f} = \boldsymbol{A}^N\boldsymbol{x}_0 + \boldsymbol{R}\boldsymbol{\mu} . \tag{19}$$

Zatem postawione na początku zadanie wyznaczenia sterowania sprowadza się do znalezienia takiego wektora $\boldsymbol{\mu}$, który spełnia zależność

$$\boldsymbol{R}\boldsymbol{\mu} = \boldsymbol{x}_\mathrm{f} - \boldsymbol{A}^N\boldsymbol{x}_0 , \tag{20}$$

gdzie $\boldsymbol{R} \in \mathbb{R}^{n\times N}$ oraz $\boldsymbol{\mu} \in \mathbb{R}^N$ są określone przez (17) i (18). Wyznaczenie ciągu sterowań $\boldsymbol{\mu}$ sprowadza się zatem do rozwiązania układ równań liniowych (20). Załóżmy, że $N \geq n$, oraz $\operatorname{rank}\boldsymbol{R} = n$, czyli układ jest N-sterowalny. Jeśli $N > n$ to istnieje nieskończenie wiele rozwiązań równania (20). Daje to możliwość nałożenia dodatkowych ograniczeń na $\boldsymbol{\mu}$.

2.1 Sterowanie z minimalną energią

Sterowanie z minimalną energią (ang. *minimum energy control*) polega na wyznaczeniu ciągu sterowań, który minimalizuje wskaźnik

$$E \equiv \sum_{k=0}^{N-1} u_k^2 \tag{21}$$

co prowadzi do zadania optymalizacji

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\mu}\|_2^2 \tag{22a}$$
$$\text{subject to} \quad \boldsymbol{R}\boldsymbol{\mu} = \boldsymbol{x}_\mathrm{f} - \boldsymbol{A}^N\boldsymbol{x}_0 \tag{22b}$$

Rozwiązaniem powyższego zadania optymalizacji jest

$$\begin{aligned}\boldsymbol{\mu}^* &= \boldsymbol{R}^\dagger \left(\boldsymbol{x}_\mathrm{f} - \boldsymbol{A}^N \boldsymbol{x}_0\right) \\ &= \boldsymbol{R}^\mathrm{T} \left(\boldsymbol{R}\boldsymbol{R}^\mathrm{T}\right)^{-1} \left(\boldsymbol{x}_\mathrm{f} - \boldsymbol{A}^N \boldsymbol{x}_0\right)\end{aligned} \tag{23}$$

gdzie

$$\boldsymbol{G} \equiv \boldsymbol{R}\boldsymbol{R}^\mathrm{T} \tag{24}$$

nazywamy gramianem N-osiągalności (N-reachability grammian), zaś $\dagger$ oznacza pseudoodwrotność Moore'a–Penrose'a macierzy. Założenie $\operatorname{rank} \boldsymbol{R} = n$ implikuje, że $\operatorname{rank} \boldsymbol{R}\boldsymbol{R}^\mathrm{T} = n$, zatem gramian N-osiągalności $\boldsymbol{G}$ jest macierzą odwracalną. Zadanie (22) jest równoważne zadaniu

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\mu}\|_2 \tag{25a}$$

$$\text{subject to} \quad \boldsymbol{R}\boldsymbol{\mu} = \boldsymbol{x}_\mathrm{f} - \boldsymbol{A}^N \boldsymbol{x}_0 \tag{25b}$$

2.2 Sterowanie z minimalnym wydatkiem

Sterowanie z minimalnym wydatkiem (ang. *minimum fuel control*, zwane również sterowaniem optymalnym ze względu na wydatek) polega na wyznaczeniu ciągu sterowań, który minimalizuje wskaźnik

$$F \equiv \sum_{k=0}^{N-1} |u_k| = \|\boldsymbol{\mu}\|_1 , \tag{26}$$

co prowadzi do zadania optymalizacji

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\mu}\|_1 \tag{27a}$$

$$\text{subject to} \quad \boldsymbol{R}\boldsymbol{\mu} = \boldsymbol{x}_\mathrm{f} - \boldsymbol{A}^N \boldsymbol{x}_0 \tag{27b}$$

2.3 Sterowanie nadążne

Zadanie syntezy sterowania nadążnego (ang. *control for trajectory tracking*) polega na znalezieniu takiego ciągu sterowań aby sygnał wyjściowy y_k układu „śledził" zadaną trajektorię odniesienia (referencyjną) w pewnym horyzoncie czasowym $k \in \{1, \ldots, N\}$. Załóżmy, że $\boldsymbol{x}_0 = 0$. Wówczas, korzystając z (16), otrzymujemy

$$\boldsymbol{x}_k = \boldsymbol{A}^{k-1}\boldsymbol{B}u_0 + \ldots + \boldsymbol{A}\boldsymbol{B}u_{k-2} + \boldsymbol{B}u_{k-1} , \tag{28}$$

zaś biorąc pod uwagę równanie wyjścia $y_k = \boldsymbol{C}\boldsymbol{x}_k$ otrzymujemy

$$y_k = \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B}u_0 + \ldots + \boldsymbol{C}\boldsymbol{A}\boldsymbol{B}u_{k-2} + \boldsymbol{C}\boldsymbol{B}u_{k-1} , \tag{29}$$

dla $k = 1, \ldots, N$, lub równoważnie w postaci macierzowej

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{B} & 0 & \ldots & 0 & 0 \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}\boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \tag{30}$$

Przyjmując oznaczenia

$$\boldsymbol{\xi} \equiv \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \qquad \boldsymbol{\mu} \equiv \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \tag{31}$$

oraz

$$\boldsymbol{\Phi} \equiv \begin{bmatrix} \boldsymbol{C}\boldsymbol{B} & 0 & \ldots & 0 & 0 \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}\boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} \end{bmatrix}, \tag{32}$$

możemy napisać

$$\boldsymbol{\xi} = \boldsymbol{\Phi}\boldsymbol{\mu} . \tag{33}$$

Macierz tranzycji $\boldsymbol{\Phi}$ posiada szczególną strukturę, tzw. strukturę Toeplitza. Jest macierzą dolnotrójkątną, odwracalną dla $\boldsymbol{C}\boldsymbol{B} \neq 0$. Oznaczmy trajektorię referencyjną przez $\boldsymbol{\xi}_\mathrm{ref}$

$$\boldsymbol{\xi}_\mathrm{ref} = \begin{bmatrix} y_1^\mathrm{ref} & y_2^\mathrm{ref} & \ldots & y_N^\mathrm{ref} \end{bmatrix}^\mathrm{T} . \tag{34}$$

Wyznaczenie poszukiwanego sterowania sprowadza się do rozwiązania zadania

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\Phi}\boldsymbol{\mu} - \boldsymbol{\xi}_\mathrm{ref}\|_2 . \tag{35}$$

Innymi słowy, dla zadanej referencyjnej sekwencji wyjść (odpowiedzi) $\boldsymbol{\xi}_\mathrm{ref}$, minimalizujemy pierwiastek z sumy kwadratów uchybów śledzenia $e_k = y_k - y_k^\mathrm{ref}$ (ang. *tracking error*) dla $k = 1, \ldots, N$,

$$\sqrt{\sum_{k=1}^{N} e_k^2} = \sqrt{\sum_{k=1}^{N} \left[y_k - y_k^\mathrm{ref}\right]^2} \tag{36}$$

Możemy nakładać dodatkowe ograniczenia na ciąg sterowań $\boldsymbol{\mu}$. Nawet jeśli macierz $\boldsymbol{\Phi}$ jest odracalna, jest dobrą praktyką rozpatrywać tzw. wersję zregularyzowaną zadania śledzenia, w której wprowadza się wyraz kary (ang. *penalty term*) proporcjonalny do energii sygnału sterującego, co prowadzi do zadania optymalizacji

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\Phi}\boldsymbol{\mu} - \boldsymbol{\xi}_\mathrm{ref}\|_2 + \gamma \|\boldsymbol{\mu}\|_2 , \tag{37}$$

gdzie $\gamma > 0$ jest dodatnim współczynnikiem kary, zaś $\gamma \|\boldsymbol{\mu}\|_2^2$ wspomnianym wyrazem kary. Postawienie zadania w ten sposób umożliwia znalezienie kompromisu (ang. *tradeoff*) pomiędzy uchybem śledzenia a energią sygnału wejściowego. Można rozpatrywać rozmaite modyfikacje zadania (37) np.

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\Phi}\boldsymbol{\mu} - \boldsymbol{\xi}_\mathrm{ref}\|_2 + \gamma \|\boldsymbol{\mu}\|_1 , \tag{38}$$

lub

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\Phi}\boldsymbol{\mu} - \boldsymbol{\xi}_\mathrm{ref}\|_2 + \gamma \|\boldsymbol{\mu}\|_2 \tag{39a}$$

$$\text{subject to} \quad \|\boldsymbol{\mu}\|_\infty \leq u^\mathrm{max} \tag{39b}$$

gdzie $\gamma > 0$ jest ustalonym parametrem zadania.
Stosunkowo łatwo można uwzględnić ograniczenia na szybkość zmian sygnału (ang. *slew rate*). Zauważmy, że dla

$$\boldsymbol{D} = \begin{bmatrix} -1 & 1 & 0 & \ldots & 0 \\ 0 & -1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \in \mathbb{R}^{(N-1)\times N} \tag{40}$$

mamy

$$\boldsymbol{D}\boldsymbol{\mu} = \begin{bmatrix} u_1 - u_0 \\ u_2 - u_1 \\ \vdots \\ u_{N-1} - u_{N-2} \end{bmatrix} . \tag{41}$$

Oznaczając przez s^max maksymalną dopuszczalną szybkość zmian sygnału, formułujemy zadanie optymalizacji

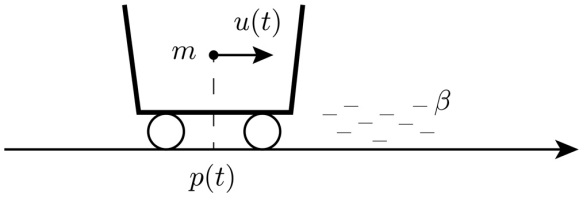
$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \|\boldsymbol{\Phi}\boldsymbol{\mu} - \boldsymbol{\xi}_\mathrm{ref}\|_2 + \gamma \|\boldsymbol{\mu}\|_2 \quad \gamma > 0 \tag{42a}$$

$$\text{subject to} \quad \|\boldsymbol{\mu}\|_\infty \leq u^\mathrm{max} \tag{42b}$$

$$\|\boldsymbol{D}\boldsymbol{\mu}\|_\infty \leq s^\mathrm{max} \tag{42c}$$

3 Przykłady

Rozpatrujemy wózek o masie m (Rys. 1), w położeniu $p(t)$, poruszający się w ustalonym kierunku, na który działa siła wymuszająca $u(t)$ oraz siła oporów ruchu scharakteryzowana współczynnikiem β.



Rysunek 1: Obiekt sterowania. Źródło: [4].

Zgodnie z drugą zasadą dynamiki Newtona, ruch wózka jest opisany równaniem

$$u(t) - \beta\dot{p}(t) = m\ddot{p}(t). \tag{43}$$

Oznaczając $x_1(t) = p(t)$, $x_2(t) = \dot{p}(t)$, można napisać równanie (43) w postaci układu dwóch równań pierwszego rzędu

$$\dot{x}_1(t) = x_2(t), \tag{44a}$$
$$\dot{x}_2(t) = \alpha x_2(t) + bu(t), \tag{44b}$$

gdzie

$$\alpha = -\frac{\beta}{m}, \qquad b = \frac{1}{m}. \tag{45}$$

Wprowadzając dalej oznaczenia

$$\boldsymbol{A}_\mathrm{c} = \begin{bmatrix} 0 & 1 \\ 0 & \alpha \end{bmatrix}, \quad \boldsymbol{B}_\mathrm{c} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{46}$$

można napisać

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_\mathrm{c}\boldsymbol{x}(t) + \boldsymbol{B}_\mathrm{c}u(t), \tag{47}$$

Jeżeli przyjmiemy, że interesującym nas sygnałem odpowiedzi rozpatrywanego układu jest położenie wózka, to równanie wyjścia będzie postaci

$$y(t) = \boldsymbol{C}\boldsymbol{x}(t), \quad \text{gdzie} \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{48}$$

Równania (47) oraz (48) opisują układ z czasem ciągłym. Dla układów tego typu można wyznaczać odpowiedniki dyskretne. Rozważmy ogólny układ LTI z czasem ciągłym

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_\mathrm{c}\boldsymbol{x}(t) + \boldsymbol{B}_\mathrm{c}\boldsymbol{u}(t), \tag{49a}$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{49b}$$

gdzie $\boldsymbol{x}(t) \in \mathbb{R}^n$ jest wektorem stanu, $\boldsymbol{u}(t) \in \mathbb{R}^m$ jest wektorem wymuszeń, zaś $\boldsymbol{y}(t) \in \mathbb{R}^p$ jest wektorem odpowiedzi. Rozwiązanie równania stanu (49a), dla $t \geq t_0$, wyraża się wzorem

$$\boldsymbol{x}(t) = \mathrm{e}^{\boldsymbol{A}_\mathrm{c}(t-t_0)}\boldsymbol{x}(t_0) + \int_{t_0}^{t} \mathrm{e}^{\boldsymbol{A}_\mathrm{c}(t-\tau)}\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(\tau)\mathrm{d}\tau\,. \tag{50}$$

Układ (49) można zdyskretyzować. Zakładając, że wymuszenie $\boldsymbol{u}(t)$ jest stałe między kolejnymi chwilami próbkowania

$$\boldsymbol{u}(t) = \boldsymbol{u}(k\Delta), \quad \text{dla każdego} \quad t \in [k\Delta, (k+1)\Delta[\tag{51}$$

oraz znając stan w chwili $k\Delta$, można wyznaczyć stan w chwili $(k+1)\Delta$

$$\begin{aligned}\boldsymbol{x}((k+l)\Delta) &= \mathrm{e}^{\boldsymbol{A}_\mathrm{c}\Delta}\boldsymbol{x}(k\Delta) + \int_{k\Delta}^{(k+1)\Delta} \mathrm{e}^{\boldsymbol{A}_\mathrm{c}((k+1)\Delta-\tau)}\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(\tau)\mathrm{d}\tau \\ &= \mathrm{e}^{\boldsymbol{A}_\mathrm{c}\Delta}\boldsymbol{x}(k\Delta) + \int_{k\Delta}^{(k+1)\Delta} \mathrm{e}^{\boldsymbol{A}_\mathrm{c}((k+1)\Delta-\tau)}\mathrm{d}\tau\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(k\Delta) \\ &= \mathrm{e}^{\boldsymbol{A}_\mathrm{c}\Delta}\boldsymbol{x}(k\Delta) + \int_{0}^{\Delta} \mathrm{e}^{\boldsymbol{A}_\mathrm{c}\tau}\mathrm{d}\tau\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(k\Delta)\,. \end{aligned} \tag{52}$$

Oznaczając

$$\boldsymbol{x}(k) = \boldsymbol{x}(k\Delta), \quad \boldsymbol{u}(k) = \boldsymbol{u}(k\Delta), \tag{53}$$

$$\boldsymbol{A}_\Delta = \mathrm{e}^{\boldsymbol{A}_\mathrm{c}\Delta}, \quad \boldsymbol{B}_\Delta = \int_0^\Delta \mathrm{e}^{\boldsymbol{A}_\mathrm{c}\tau}\boldsymbol{B}_\mathrm{c}\mathrm{d}\tau, \tag{54}$$

możemy napisać równania odpowiednika dyskretnego

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_\Delta\boldsymbol{x}(k) + \boldsymbol{B}_\Delta\boldsymbol{u}(k), \tag{55a}$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k). \tag{55b}$$

W sytuacjach gdy nie będzie to prowadzić do wątpliwości, będziemy opuszczać indeks Δ przy macierzach $\boldsymbol{A}$ i $\boldsymbol{B}$ w równaniu (55a), ponadto, będziemy pisać $\boldsymbol{x}_k$ oraz $\boldsymbol{u}_k$ zamiast, odpowiednio, $\boldsymbol{x}(k)$ oraz $\boldsymbol{u}(k)$.

Korzystając ze wzorów (53–55) oraz uwzględniając (46) otrzymujemy macierze

$$\boldsymbol{A} = \begin{bmatrix} 1 & \frac{1}{\alpha}(\mathrm{e}^{\alpha\Delta} - 1) \\ 0 & \mathrm{e}^{\alpha\Delta} \end{bmatrix}, \tag{56}$$

$$\boldsymbol{B} = -\frac{1}{\beta}\begin{bmatrix} \frac{1}{\alpha}(\mathrm{e}^{\alpha\Delta} - 1) - \Delta \\ \mathrm{e}^{\alpha\Delta} - 1 \end{bmatrix} \tag{57}$$

odpowiednika dyskretnego

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{58a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k, \tag{58b}$$

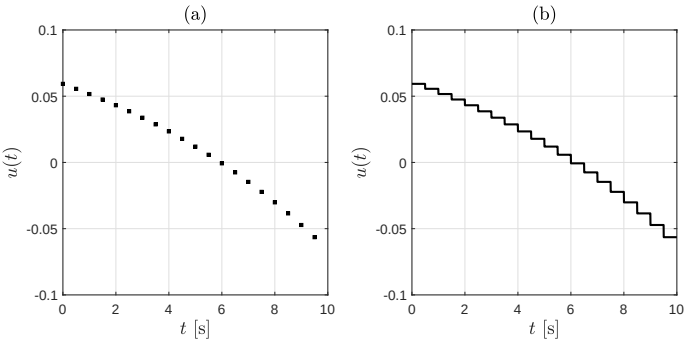
dla rozpatrywanego obiektu tzn. wózka.

4 Zadania

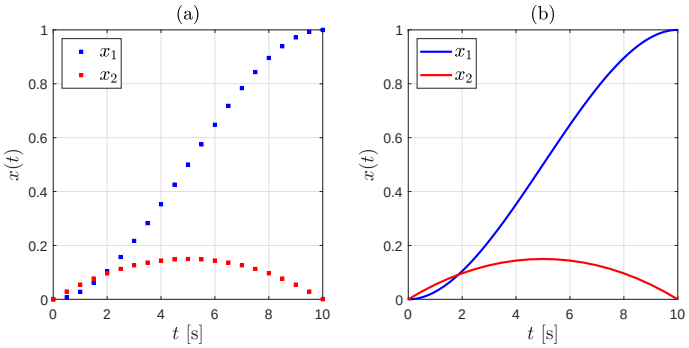
Zadanie 1. Zakładając, że masa rozpatrywanego wózka wynosi $m = 1\,[\mathrm{kg}]$, $\beta = 0.1\,[\mathrm{Ns/m}]$, przedział próbkowania $\Delta = 0.5\,[\mathrm{s}]$, stan początkowy $x_0 = [0\ 0]^\mathrm{T}$, $t_\mathrm{f} = 10\,[\mathrm{s}]$ wyznaczyć optymalny [w sensie sterowania z minimalną energią (25)] ciąg sterowań

$$\boldsymbol{\mu}_{k_\mathrm{f}} = [u_0, \ldots, u_{N-1}]^\mathrm{T} \tag{59}$$

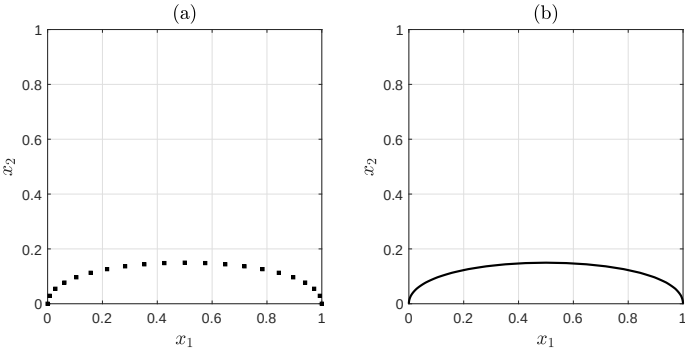
który przeprowadzi układ do stanu końcowego $x_\mathrm{f} = [1\ 0]^\mathrm{T}$ (który odpowiada położeniu końcowemu $p(k_\mathrm{f}) = 1$ i prędkości końcowej $\dot{p}(k_\mathrm{f}) = 0$). Wyznaczyć trajektorię stanu układu odpowiadającą temu ciągowi sterowań oraz wygenerować wykresy przebiegu $u(t)$ i $x(t)$ przedstawione na Rys. 2–4. Horyzont wynosi $N = t_\mathrm{f}/\Delta = 20$. Trajektorię w przestrzeni stanu dla zadanego sterowania można wyznaczyć korzystając z funkcji `lsim` środowiska Matlab. Zadanie (25) jest równoważne zadaniu (22), które jest zadaniem optymalizacji kwadratowej, dlatego można je rozwiązać korzystając z funkcji `quadprog` środowiska Matlab.



Rysunek 2: Sterowanie z minimalną energią - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 1]



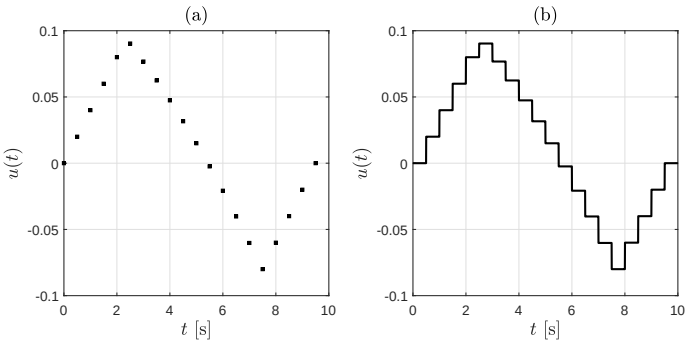
Rysunek 3: Sterowanie z minimalną energią - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 1]



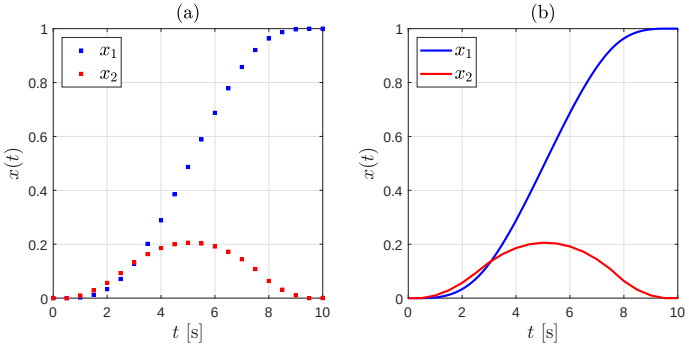
Rysunek 4: Sterowanie z minimalną energią - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 1]

Zadanie 2. Tak jak w Zadaniu 1, przy dodatkowych warunkach $u_0 = u_{N-1} = 0$ oraz

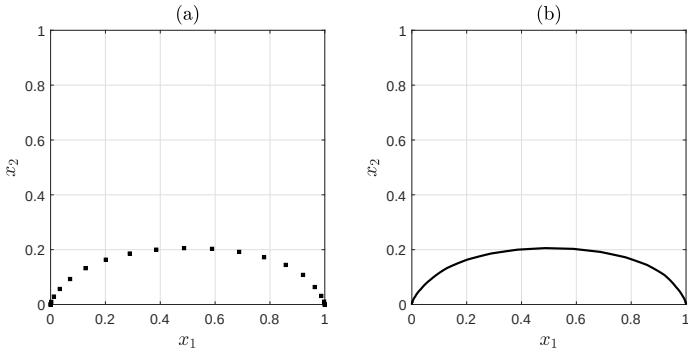
$$|u_{k+1} - u_k| \leq 0.02, \quad k = 0, \ldots, N-2. \tag{60}$$



Rysunek 5: Sterowanie z minimalną energią - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 2]



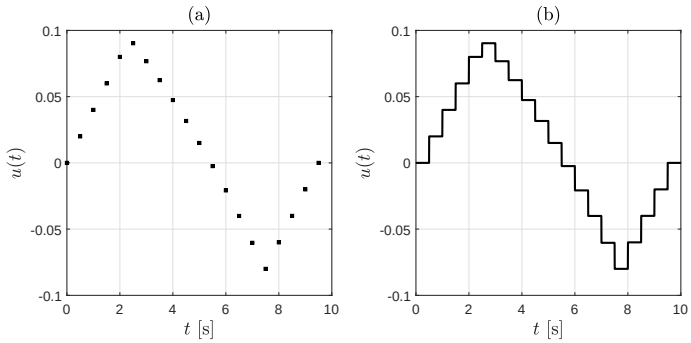
Rysunek 6: Sterowanie z minimalną energią - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 2]



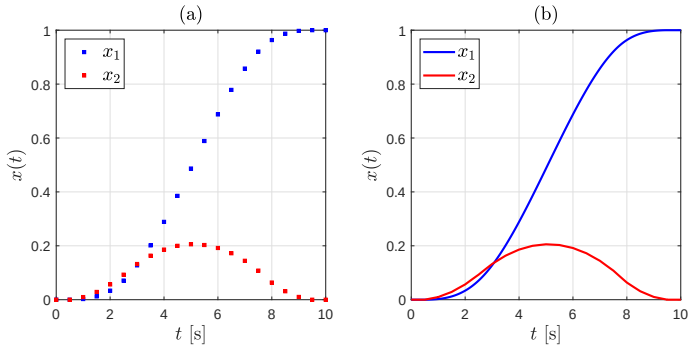
Rysunek 7: Sterowanie z minimalną energią - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 2]

Zadanie 3. Tak jak w Zadaniu 1, przy dodatkowych warunkach $u_0 = u_{N-1} = 0$ oraz

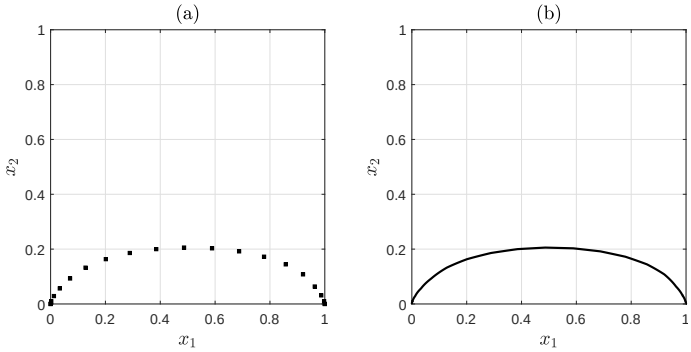
$$u_{k+1} - u_k \leq 0.02, \quad k = 0, \ldots, N-2. \tag{61}$$



Rysunek 8: Sterowanie z minimalną energią - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 3]

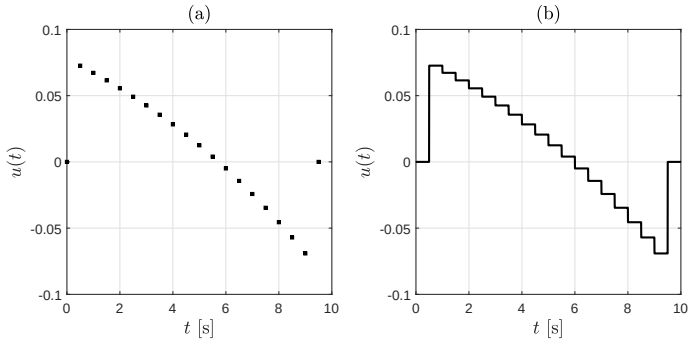


Rysunek 9: Sterowanie z minimalną energią - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 3]

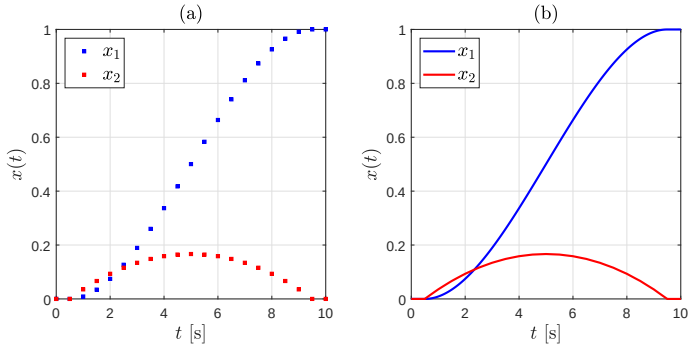


Rysunek 10: Sterowanie z minimalną energią - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 3]

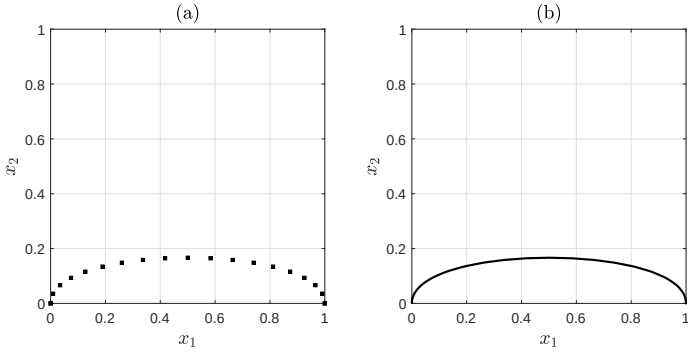
Zadanie 4. Tak jak w Zadaniu 1, przy dodatkowych warunkach $u_0 = u_{N-1} = 0$ (bez warunków na sterowanie).



Rysunek 11: Sterowanie z minimalną energią - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 4]



Rysunek 12: Sterowanie z minimalną energią - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 4]

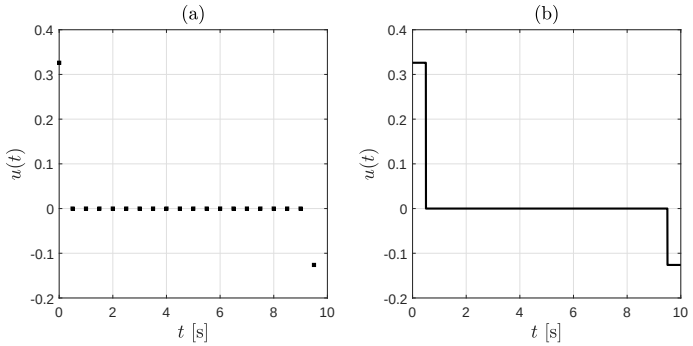


Rysunek 13: Sterowanie z minimalną energią - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 4]

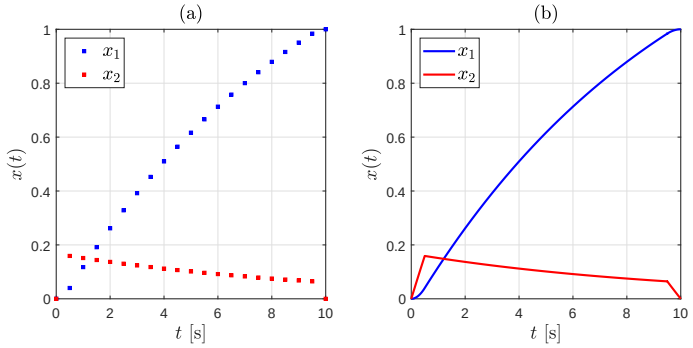
Zadanie 5. Zakładając, że masa rozpatrywanego wózka wynosi $m = 1\,[\mathrm{kg}]$, $\beta = 0.1\,[\mathrm{Ns/m}]$, przedział próbkowania $\Delta = 0.5\,[\mathrm{s}]$, stan początkowy $x_0 = [0\ 0]^{\mathrm{T}}$, $t_{\mathrm{f}} = 10\,[\mathrm{s}]$ wyznaczyć optymalny [w sensie sterowania z minimalnym wydatkiem (27)] ciąg sterowań

$$\boldsymbol{\mu}_{k_{\mathrm{f}}} = [u_0, \ldots, u_{N-1}]^{\mathrm{T}} \tag{62}$$

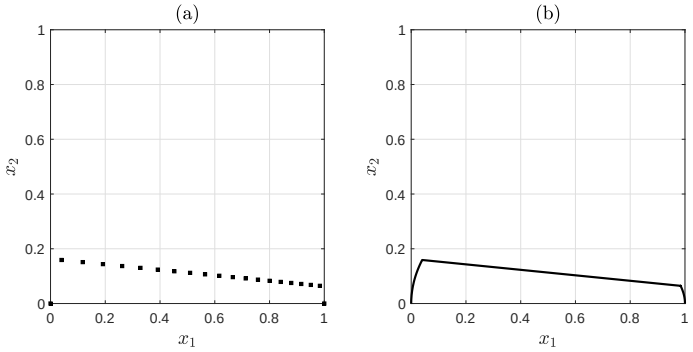
który przeprowadzi układ do stanu końcowego $x_{\mathrm{f}} = [1\ 0]^{\mathrm{T}}$ (który odpowiada położeniu końcowemu $p(k_{\mathrm{f}}) = 1$ i prędkości końcowej $\dot{p}(k_{\mathrm{f}}) = 0$). Wyznaczyć trajektorię stanu układu odpowiadającą temu ciągowi sterowań oraz wygenerować wykresy przebiegu $u(t)$ i $x(t)$ przedstawione na Rys. 14–16. Horyzont wynosi $N = t_{\mathrm{f}}/\Delta = 20$. Trajektorię w przestrzeni stanu dla zadanego sterowania można wyznaczyć korzystając z funkcji `lsim` środowiska Matlab.



Rysunek 14: Sterowanie z minimalnym wydatkiem - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 5]



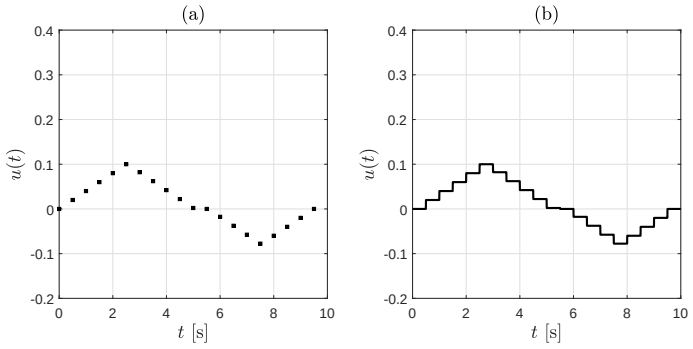
Rysunek 15: Sterowanie z minimalnym wydatkiem - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 5]



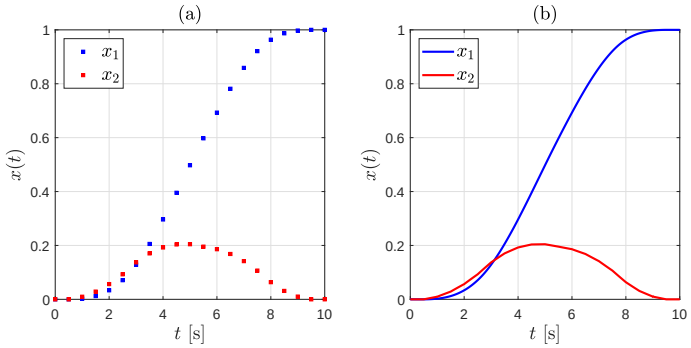
Rysunek 16: Sterowanie z minimalnym wydatkiem - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 5]

Zadanie 6. Tak jak w Zadaniu 5, przy dodatkowych warunkach $u_0 = u_{N-1} = 0$ oraz

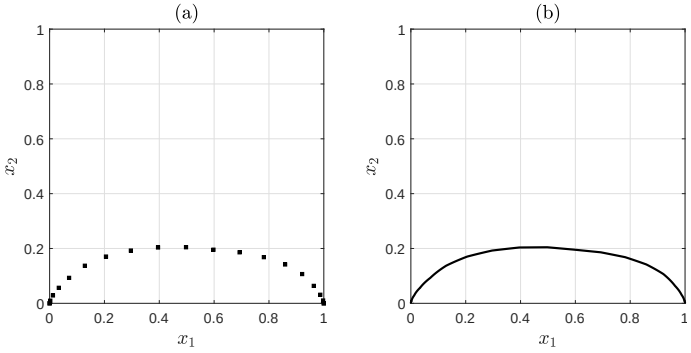
$$|u_{k+1} - u_k| \leq 0.02, \quad k = 0, \ldots, N-2. \tag{63}$$



Rysunek 17: Sterowanie z minimalnym wydatkiem - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 6]



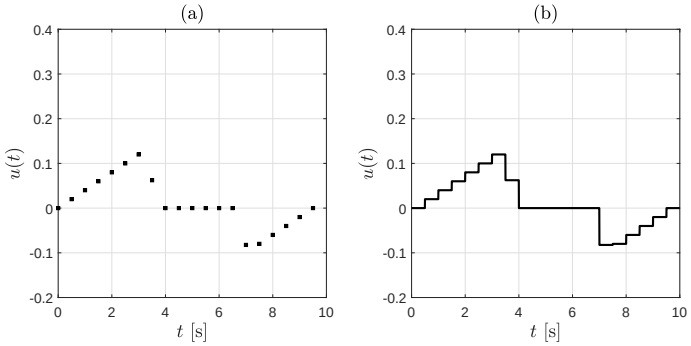
Rysunek 18: Sterowanie z minimalnym wydatkiem - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 6]



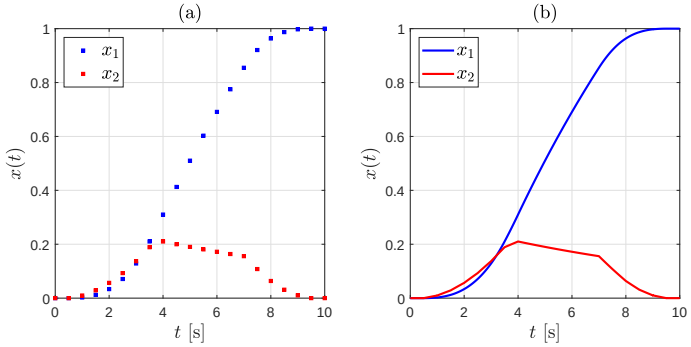
Rysunek 19: Sterowanie z minimalnym wydatkiem - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 6]

Zadanie 7. Tak jak w Zadaniu 5, przy dodatkowych warunkach $u_0 = u_{N-1} = 0$ oraz

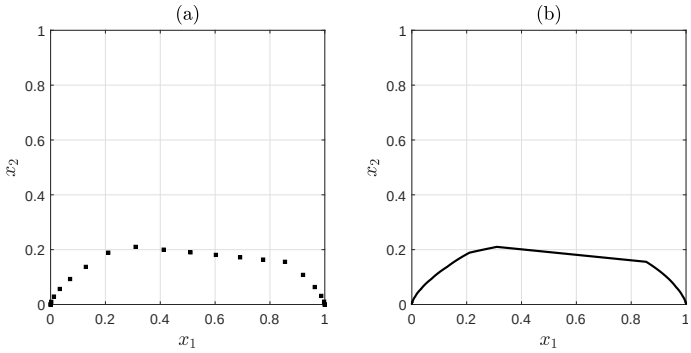
$$u_{k+1} - u_k \leq 0.02, \quad k = 0, \ldots, N-2. \tag{64}$$



Rysunek 20: Sterowanie z minimalnym wydatkiem - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 7]

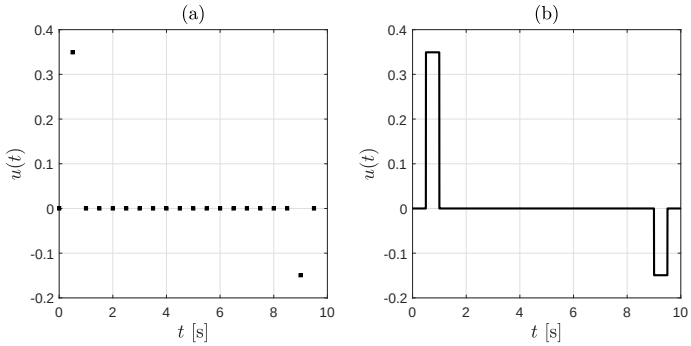


Rysunek 21: Sterowanie z minimalnym wydatkiem - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 7]

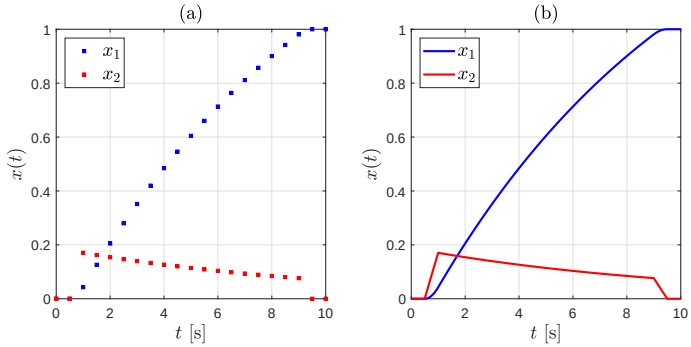


Rysunek 22: Sterowanie z minimalnym wydatkiem - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 7]

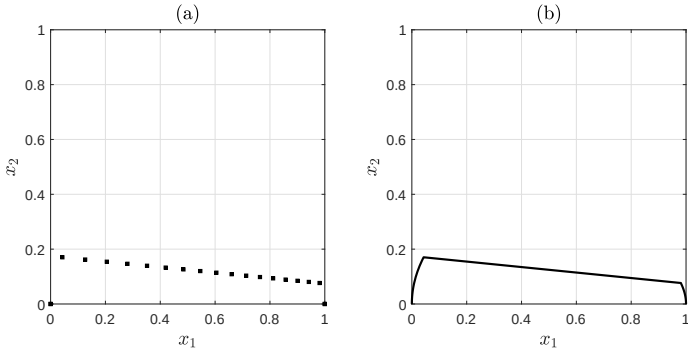
Zadanie 8. Tak jak w Zadaniu 5, przy dodatkowych warunkach $u_0 = u_{N-1} = 0$ (bez warunków na sterowanie).



Rysunek 23: Sterowanie z minimalnym wydatkiem - sygnał sterowania $u(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 8]



Rysunek 24: Sterowanie z minimalnym wydatkiem - zmienne stanu $x_1(t)$ i $x_2(t)$. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 8]



Rysunek 25: Sterowanie z minimalnym wydatkiem - trajektoria w przestrzeni stanu. Panel (a): układ z czasem dyskretnym, panel (b): układ z czasem ciągłym. [Zadanie 8]

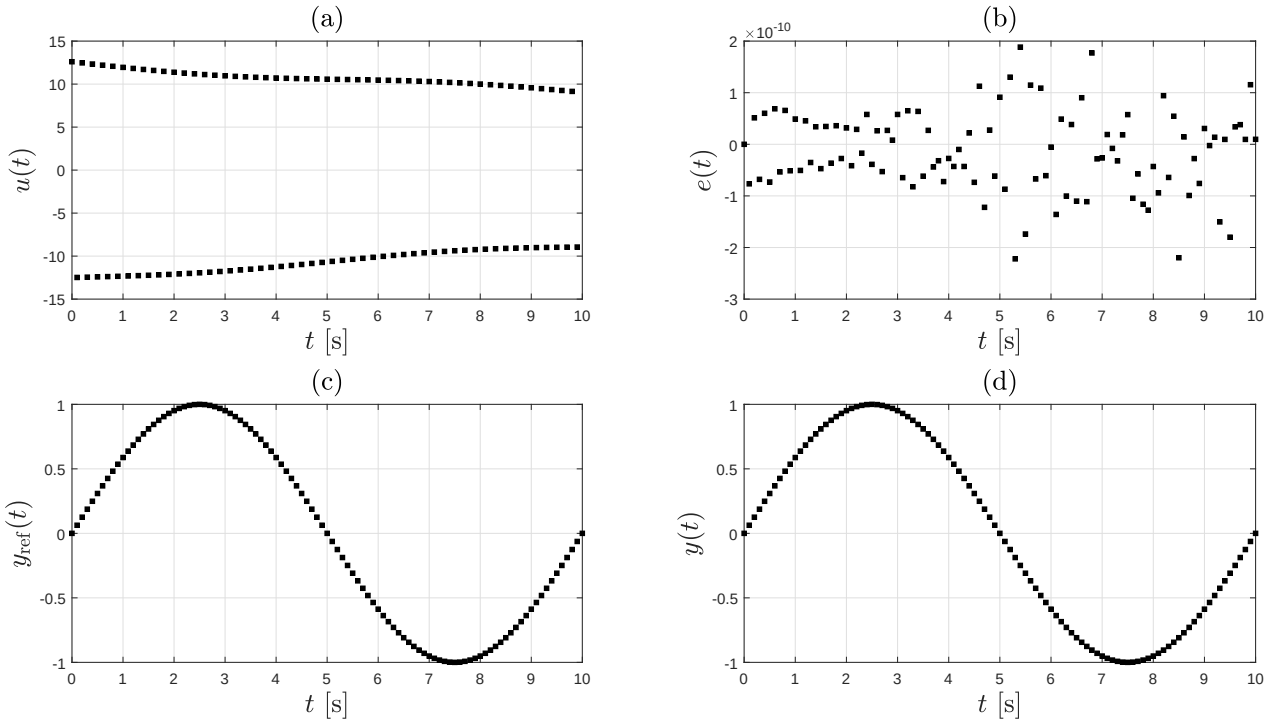
Zadanie 9. Dla $x_0 = [0\ 0]^{\mathrm{T}}$, $\Delta = 0.1\,[\mathrm{s}]$, $k_{\mathrm{f}} = 100$, $\gamma = 0$ wyznaczyć optymalny [w sensie sterowania nadążnego, zgodnie ze wzorem (37)] ciąg sterowań

$$\mu_{k_{\mathrm{f}}} = [u_0, \ldots, u_{k_{\mathrm{f}}-1}]^{\mathrm{T}} \tag{65}$$

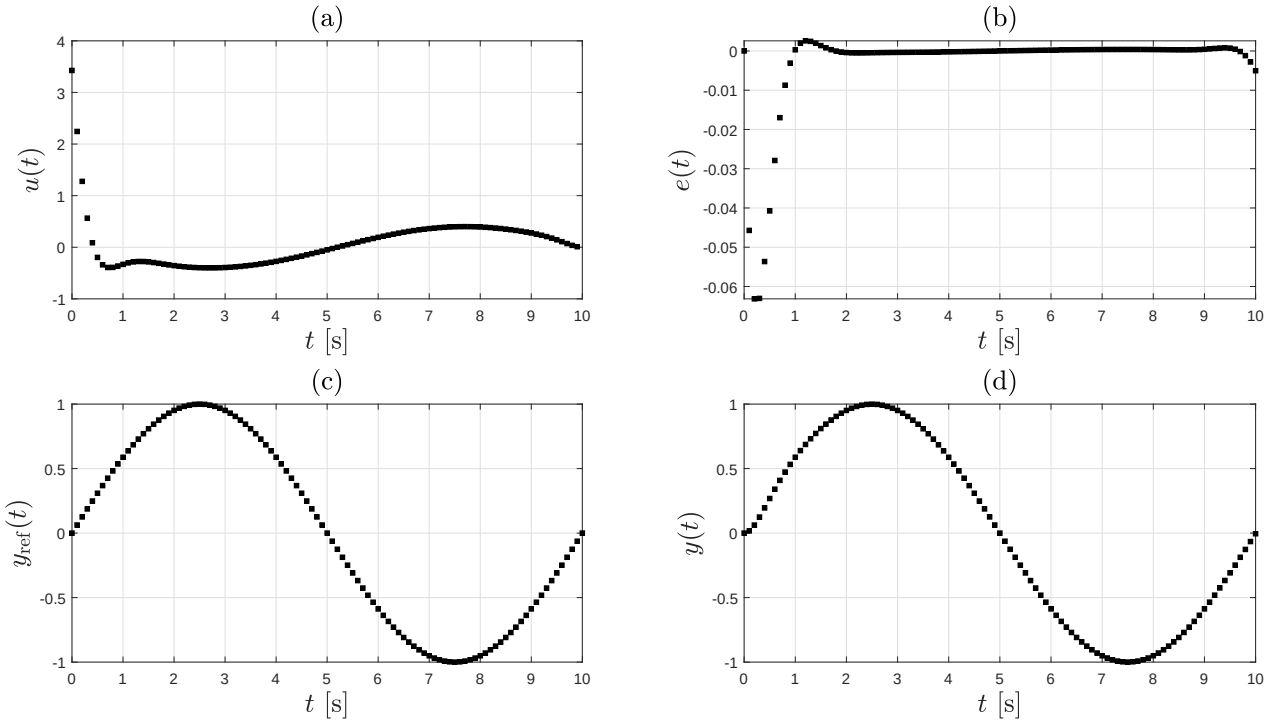
dla trajektorii referencyjnej

$$y_{\mathrm{ref}} = \sin(\omega k \Delta), \quad \omega = \frac{2\pi}{10}, \quad k = 1, \ldots, k_{\mathrm{f}}. \tag{66}$$

Wyznaczyć trajektorię stanu układu odpowiadającą temu ciągowi sterowań. Sporządzić rysunki przebiegu $u(t)$ i $x(t)$ tak jak na Rys. 26. Powtórzyć zadanie dla $\gamma = 0.1$ i porównać wyniki (odpowiednie przebiegi dla $\gamma = 0.1$ na Rys. 27).



Rysunek 26: Sterowanie nadążne dla $\gamma = 0$ [wzór (37)]. Panel (a): sterowanie $u(t)$, panel (b): uchyb śledzenia $e(t) = y(t) - y_{\mathrm{ref}}(t)$, panel (c): odpowiedź referencyjna $y_{\mathrm{ref}}(t)$, panel (d): odpowiedź układu $y_{\mathrm{ref}}(t)$ na sterowanie $u(t)$. [Zadanie 9]



Rysunek 27: Sterowanie nadążne dla $\gamma = 0.1$ [wzór (37)]. Panel (a): sterowanie $u(t)$, panel (b): uchyb śledzenia $e(t) = y(t) - y_{\mathrm{ref}}(t)$, panel (c): odpowiedź referencyjna $y_{\mathrm{ref}}(t)$, panel (d): odpowiedź układu $y_{\mathrm{ref}}(t)$ na sterowanie $u(t)$. [Zadanie 9]

Literatura

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. `http://web.stanford.edu/~boyd/cvxbook/`.

[2] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani. *Algorytmy*. Wydawnictwo Naukowe PWN, Warsza-

wa, 2012.

[3] Stephen Boyd and Lieven Vandenberghe. *Additional Exercises for Convex Optimization*. 2004. https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook_extra_exercises.pdf.

[4] G.C. Calafiore and L. El Ghaoui. *Optimization Models*. Control systems and optimization series. Cambridge University Press, 2014.