Podstawy Sterowania Optymalnego
Labolatorium 5

Regulator LQR z nieskończonym horyzontem czasowym

Prowadzący: mgr inż. Krzysztof Hałas
Wykonał: Ryszard Napierała

7 grudnia 2021

Zadanie 2

1. **Rozwiązać algebraiczne równanie Riccatiego dla dynamiki układu przyjmując jednostkową macierz Q oraz $R = 1$. Do znalezienia rozwiązania równania Riccatiego wykorzystać funkcję *scipy.linalg.solv* Wyznaczyć i wypisać wartości wzmocnień K.**

```
23 def calculateK(A: np.ndarray, B: np.ndarray, Q: np.ndarray, R: np.ndarray):
24     P = linalg.solve_continuous_are(A, B, Q, R)
25     return linalg.inv(R) @ B.T @ P
26 
27 K = calculateK(A, B, Q, R)
28 print('K = ', K)
```

Output:

```
K =  [[0.23606798 0.65949437]]
```

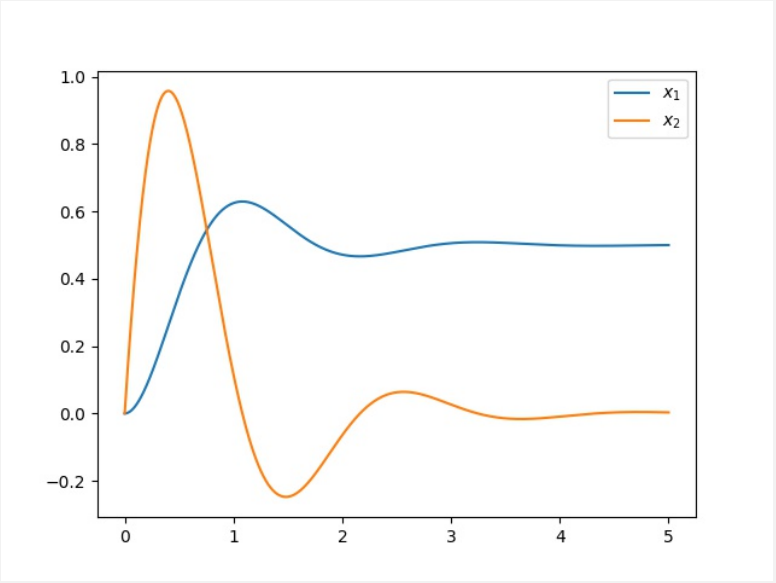
2. **Przygotować funkcję model(x,t) implementującą model dynamiki układu otwartego zgodnie z równaniem. Funkcja powinna przyjmować na wejściu stan układu x oraz aktualną chwilę czasu t**

```
31 def model(
32         x: np.ndarray,
33         t: float,
34         A: np.ndarray,
35         B: np.ndarray,
36         u: Callable):
37     x = x.reshape((2, 1))
38     dx = A@x + B@u(t)
39     return dx.flatten()
```

3. **Przeprowadzić symulację odpowiedzi obiektu na wymuszenie skokowe w czasie $t \in (0, 5)$ s wykorzystując funkcję odeint.**

```
t = np.linspace(0, 5, 1000)
y = integrate.odeint(
    model,
    [0, 0],
    t,
    (A, B, lambda x: np.array([[1]]))
)
plt.plot(t, y)
plt.legend(['$x_1$', '$x_2$'])
plt.show()
plt.close()
```

Output:



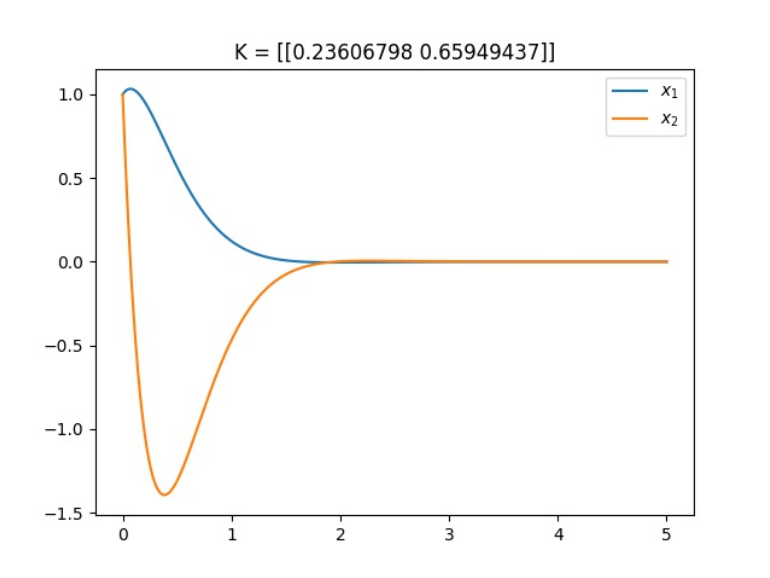
4. **Zmodyfikować funkcję model(x,t) tak, by sygnał sterujący miał postać $u = -Kx$**

```
def modelK(
        x: np.ndarray,
        t: float,
        A: np.ndarray,
        B: np.ndarray,
        K: np.ndarray):
    x = x.reshape((2, 1))
    dx = A@x + -B@K@x
    return dx.flatten()
```

5. **Przeprowadzić symulację układu dla niezerowych warunków początkowych. Zbadać wpływ macierzy Q oraz R na przebieg odpowiedzi układu.**

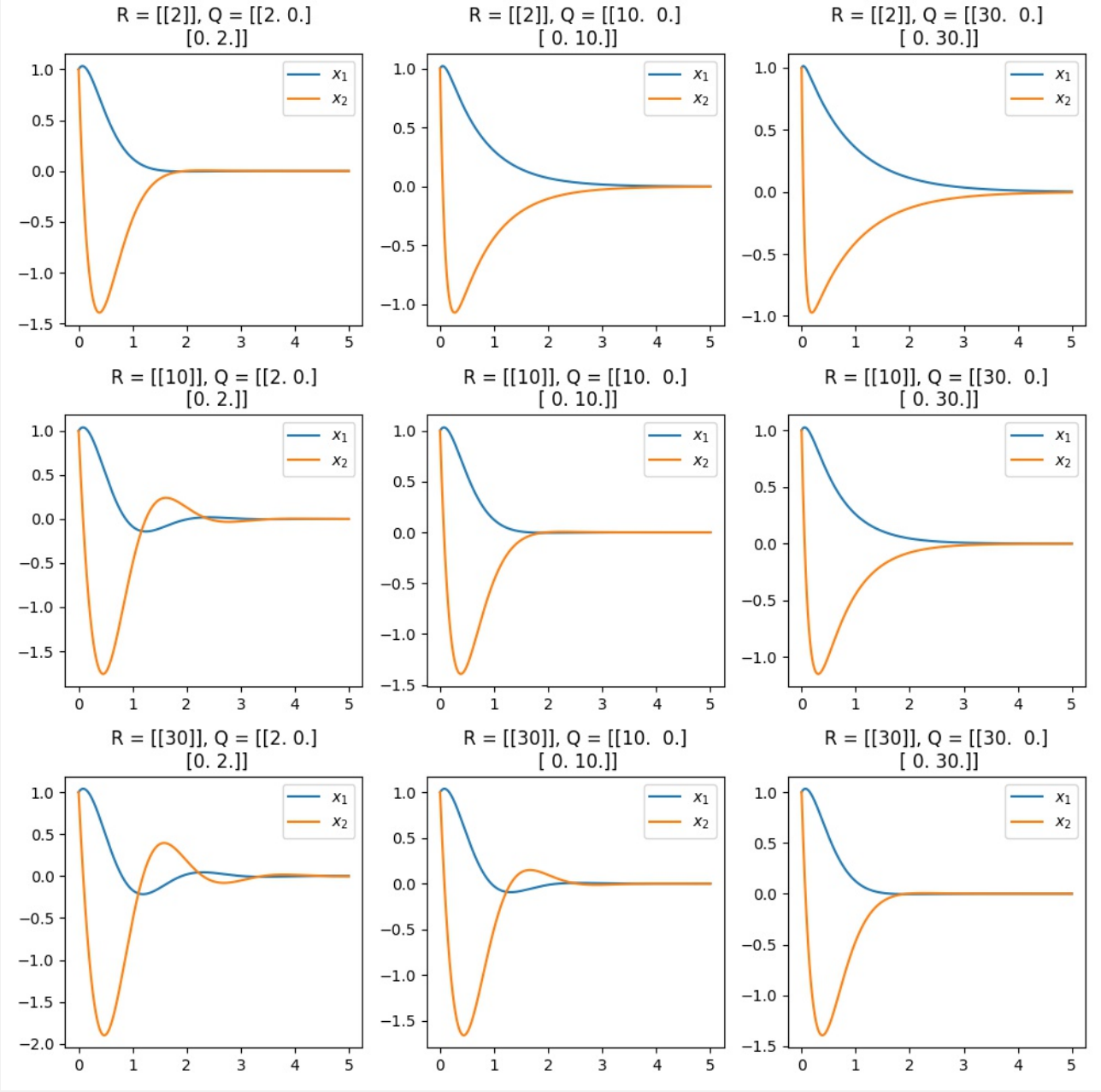
```
66  def plotLQR(
67          ax: plt.Axes,
68          t: np.ndarray,
69          A: np.ndarray,
70          B: np.ndarray,
71          K: np.ndarray,
72          title: str):
73      y = integrate.odeint(
74          modelK,
75          [1, 1],
76          t,
77          (A, B, K)
78      )
79      ax.plot(t, y)
80      ax.legend(['$x_1$', '$x_2$'])
81      ax.set_title(title)
82
83  fig, ax = plt.subplots(1)
84  plotLQR(ax, t, A, B, K, f'K = {K}')
85  plt.show()
86  plt.close()
87
88  fig, axes = plt.subplots(3, 3)
89  fig.set_size_inches(10, 10)
90  for i, r in enumerate((2, 10, 30)):
91      for j, q in enumerate((2, 10, 30)):
92          k = calculateK(A, B, Q*q, R*r)
93          plotLQR(axes[i][j], t, A, B, k, f'R = {R*r}, Q = {Q*q}')
94  plt.tight_layout()
95  plt.show()
96  plt.close()
```

Output:



Czy macierze Q oraz R pozwalają dowolnie kształtować przebieg uchybu regulacji? Czy istnieje jakaś zależność między doborem macierzy Q oraz R?
Większe wartości Q wydłużają czas regulacji i zmniejszają przeregulowanie, natomiast większe wartości R skracają czas regulacji i zwiększają przeregulowanie. Co jest widoczne na poniższych przebiegach odpowiedzi skokowej układu.

Output:



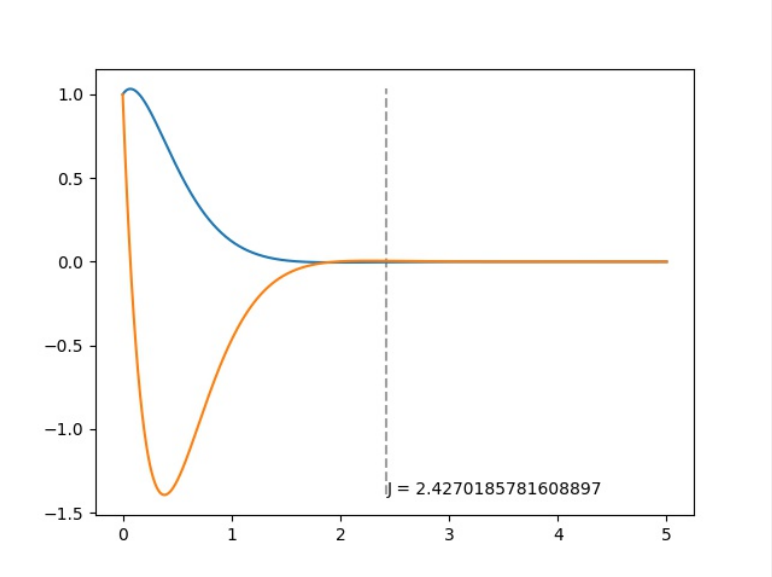
6. **Rozszerzyć funkcję model(x,t) o wyznaczanie wartości wskaźnika jakości J. Funkcja model(x,t) powinna wyznaczać pochodną (tj. wyrażenie podcałkowe) wskaźnika J jako dodatkową zmienną stanu – zostanie ona scałkowana przez odeint, a jej wartość zwrócona po zakończeniu symulacji.**

```
99  def modelKJ(
100         x: np.ndarray,
101         t: float,
102         A: np.ndarray,
103         B: np.ndarray,
104         K: np.ndarray,
105         Q: np.ndarray,
106         R: np.ndarray):
```

```
    j = x[1]
    x = x[1:].reshape((2, 1))
    u = -K@x
    j += x.T@Q@x + u.T@R@u
    dx = A@x + B@u
    return np.append(j.flatten(), dx.flatten())

y = integrate.odeint(
    modelKJ,
    [0, 1, 1],
    t,
    (A, B, K, Q, R)
)
plt.plot(t, y[:,1:])
plt.vlines(y[-1,:1], np.min(y[:,1:]), np.max(y[:,1:]), '#A0A0A0', '--')
plt.annotate(f'J = {y[-1, 0]}', (y[-1, 0], np.min(y[:,1:])))
plt.show()
plt.close()
```

Output:



Czy wyznaczona wartość rzeczywiście odpowiada minimalizowanemu wyrażeniu? W jakim horyzoncie czasu została ona wyznaczona?
Wartość odpowiada minimalnemu wyrażniu, została zaznaczona na powyższym wykresie.

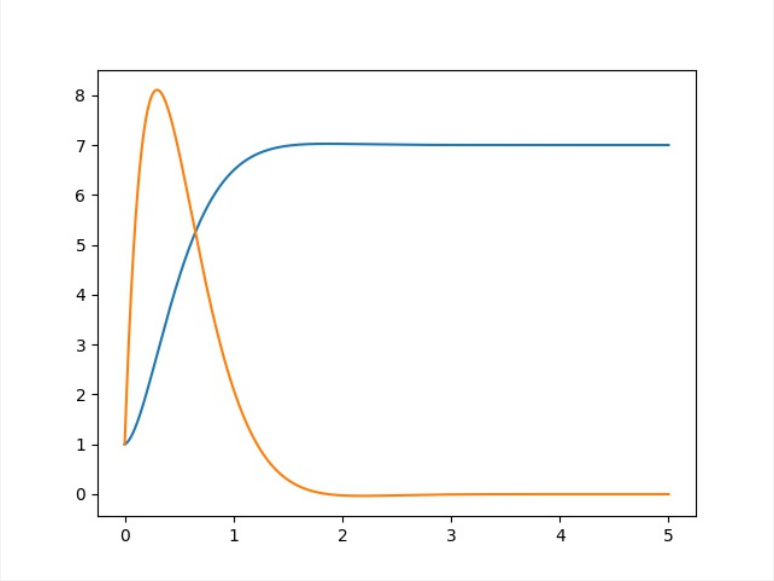
Zadanie 3

1. **Zmodyfikować funkcję model(x,t) tak, by sygnał sterujący wyznaczany był zgodnie ze schematem przedstawionym na rysunku. Wyznaczyć wzmocnienia regulatora zgodnie z algorytmem** *LQR*

```python
def modelKe(
        x: np.ndarray,
        t: float,
        A: np.ndarray,
        B: np.ndarray,
        K: np.ndarray,
        qd: np.ndarray,
        Cap: float):
    qd_over_C = qd/Cap
    L = x[0]
    x = x[1:].reshape((2, 1))
    xd = np.array([[qd, 0]]).T
    e = xd - x
    ue = -K@e
    u = -ue+qd_over_C
    dx = A@x + B@u
    L += e.T@Q@e + ue.T@R@ue
    return np.append(L.flatten(), dx.flatten())

qd = 7
y = integrate.odeint(
    modelKe,
    [0, 1, 1],
    t,
    (A, B, K, qd, C)
)
plt.plot(t, y[:, 1:])
plt.show()
plt.close()
```

Output:

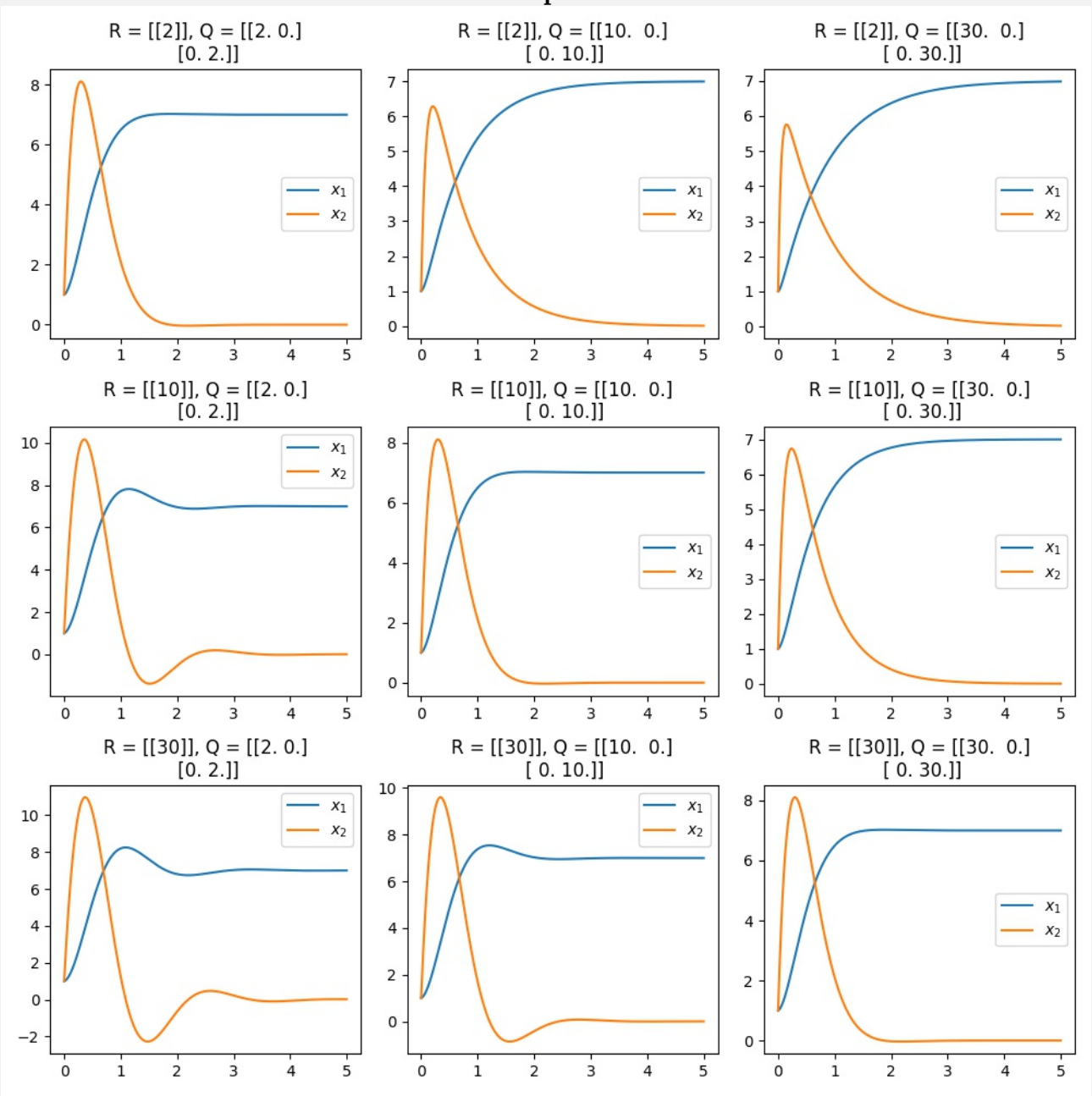


2. **Przeprowadzić symulację układu zamkniętego dla wybranej wartości zadanej qd. Zbadać wpływ macierzy Q oraz R na przebieg odpowiedzi układu.**

```
158 def plotLQR(
159         ax: plt.Axes,
160         t: np.ndarray,
161         A: np.ndarray,
162         B: np.ndarray,
163         K: np.ndarray,
164         qd: float,
165         C: float,
166         title: str):
167     y = integrate.odeint(
168         modelKe,
169         [0, 1, 1],
170         t,
171         (A, B, K, qd, C)
172     )
173     ax.plot(t, y[:, 1:])
174     ax.legend(['$x_1$', '$x_2$'])
175     ax.set_title(title)
176 
177 fig, axes = plt.subplots(3, 3)
178 fig.set_size_inches(10, 10)
179 for i, r in enumerate((2, 10, 30)):
180     for j, q in enumerate((2, 10, 30)):
181         k = calculateK(A, B, Q*q, R*r)
182         plotLQR(axes[i][j], t, A, B, k, qd, C, f'R = {R*r}, Q = {Q*q}')
183 plt.tight_layout()
184 plt.show()
185 plt.close()
```

Wpływ macierzy Q i R jest analogiczny jak w zadaniu 2, podpunkcie 5.

Output:


R = [[2]], Q = [[2. 0.]
[0. 2.]]
R = [[2]], Q = [[10. 0.]
[0. 10.]]
R = [[2]], Q = [[30. 0.]
[0. 30.]]
R = [[10]], Q = [[2. 0.]
[0. 2.]]
R = [[10]], Q = [[10. 0.]
[0. 10.]]
R = [[10]], Q = [[30. 0.]
[0. 30.]]
R = [[30]], Q = [[2. 0.]
[0. 2.]]
R = [[30]], Q = [[10. 0.]
[0. 10.]]
R = [[30]], Q = [[30. 0.]
[0. 30.]]
x_1
x_2