ЛР №2 «Переходные процессы, свободное движение, устойчивость»

Отчет

Студент
Кирилл Лалаянц
R33352
336700
Вариант - 6

Преподаватель
Пашенко А.В.

Факультет Систем Управления и Робототехники

ИТМО

13.09.2023

Содержание

1 Вводные данные

1.1 Цель работы

В этой работе будет проведенно исследование следующих вопросов:

- Характер свободного поведения системы в зависимости от корней ее характеристического уравнения.
- Исследования границ зоны устойчивости конкретной системы.
- Создание автономного генератора для воссаздания поведения системы.
- Изучение фазовых портретов канонической управляемой формы.

1.2 Инициализация необходимых переменных в Python

Импорт необходимых библиотек (для реализации здесь и далее используется Python Control Systems Library); инициализация массива временных отметок:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import control
4 import sympy
5 import os
6 import math
```

2 Выполнение работы

2.1 Задание 1. Свободное движение.

2.1.1 Теория

Хотим получить ДУ системы вида:

$$\ddot{y} + a_1\dot{y} + a_0 y = u$$

Сначала получим характеристическое уравнение для частного решения вида:

$$\ddot{y} + a_1\dot{y} + a_0 y = 0$$

Имеем два корня – λ_1 и λ_2. Тогда:

$$(p - \lambda_1)(p - \lambda_2) = p^2 - (\lambda_1 + \lambda_2)p + \lambda_1\lambda_2 = 0,$$

умножив которое на y получаем:

$$\ddot{y} + (\lambda_1 + \lambda_2)\dot{y} + \lambda_1\lambda_2 y = 0.$$

Для данного ДУ известно частичное решение:

$$y_0(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t},$$

параметры c_1 и c_2 которого вычисляются из начальных условий.
Также подставим характеристическое уравнение в знаменатель TF $W(p)$:

$$W(p) = \frac{1}{p^2 - (\lambda_1 + \lambda_2)p + \lambda_1\lambda_2}$$

Полученная TF соотвествует системе с желаемыми корнями характеристического уравнения.

2.1.2 Программная реализация

```
1  def task1_output(m1, m2, initial_value, ts, plot_name, save_name):
2      poly = sympy.simplify((p - m1) * (p - m2))
3      coeffs = sympy.Poly(poly, p).all_coeffs()
4
5      ss = control.tf2ss(control.tf(1, np.array(coeffs, dtype=np.float64)))
6      ss_reachable = control.canonical_form(ss, form="reachable")[0]
7
8      tf2_y_0_0 = control.forced_response(ss_reachable, U=0, X0=[0, 0], T=ts)
9      tf2_y_1_1 = control.forced_response(ss_reachable, U=0, X0=initial_value, T=ts)
10     plot_task1(tf2_y_0_0, tf2_y_1_1, initial_value, ts, plot_name, save_name)
```

2.1.3 Результаты

- Устойчивая и неустойчивая апериодические моды

  Выбраны $\lambda_1 = -1$ и $\lambda_2 = 1$. Получено уравнение свободного движения вида:

$$y_0(t) = c_1 e^{-t} + c_2 e^{t}$$

  Из аналитического решения видно, что система, при отличных от 0 начальных условиях, со временем стремится к бесконечности, следовательно – неустойчива.

  Проверим моделированием:

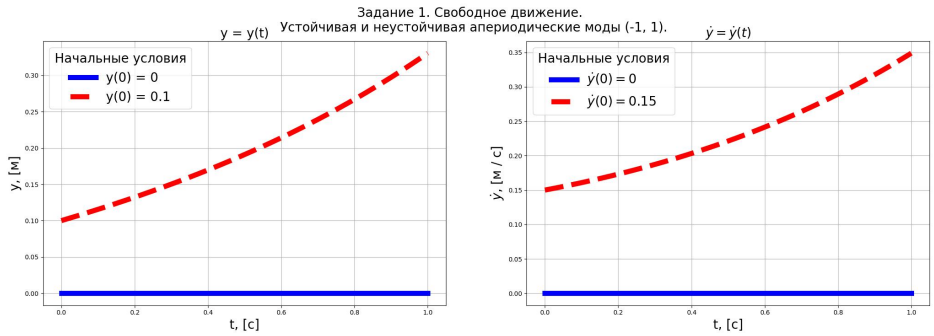

Рис. 1: Результаты устойчивой и неустойчивой апериодических мод.

  Действительно, все так.

- Нейтральная и неустойчивая апериодические моды

  Выбраны $\lambda_1 = 0$ и $\lambda_2 = 1$. Получено уравнение свободного движения вида:

$$y_0(t) = c_1 e^{-t} + c_2$$

  Из аналитического решения видно, что система, при отличных от 0 начальных условиях, со временем стремится к бесконечности, следовательно – неустойчива.

  Проверим моделированием:

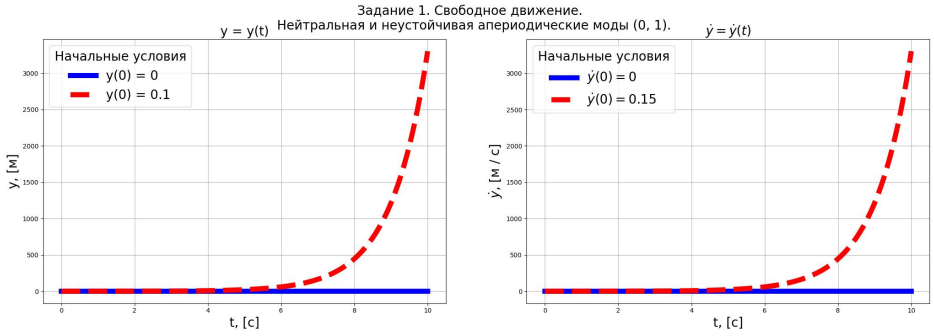

Рис. 2: Результаты нейтральной и неустойчивой апереодических мод.

Действительно, все так.

- Пара консервативных мод

  Выбраны $\lambda_1 = i$ и $\lambda_2 = -i$. Получено уравнение свободного движения вида:

  $$y_0(t) = c_1 cos(t) + c_2 sin(t)$$

  Из аналитического решения видно, что система, при отличных от 0 начальных условиях, устойчива по Ляпунову.

  Проверим моделированием:

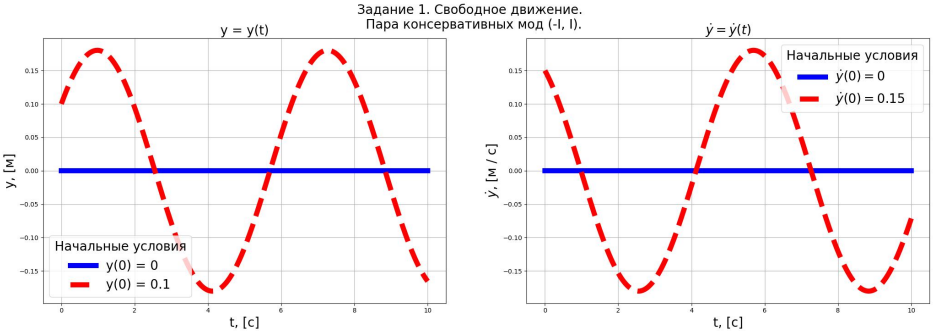

Рис. 3: Результаты пары консервативных мод.

Действительно, все так.

2.2 Задание 2. Переход от формы вход-выход к форме вход-состояние-выход.

2.2.1 Теория

Представление системы можно выбрать другое – в форме State Space. Система в форме SS имеет вид:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases},$$

где t – непрерывное время, $x(t) \in R^n$ – вектор состояния; $u(t) \in R^p$ – вектор управления; $y(t) \in R^l$ – линейный выход системы. $A \in R^{n \times n}$ – матрица системы; $B \in R^{n \times p}$ – матрица управления; $C \in R^{l \times n}$ – матрица наблюдения; $D \in R^{l \times m}$ – матрица связи.

В силу свободы выбора базиса в линейном пространстве состояний такая система может иметь бесконечное количество вариантов заполнения матриц. Для всеобщего удобства приняты три канонических формы представления:

1. Управляемая

$$A = \begin{bmatrix} 0 & & & \\ 0 & & I & \\ \vdots & & & \\ \frac{-a_0}{a_n} & \frac{-a_1}{a_n} & \dots & \frac{-a_{n-1}}{a_n} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, C = \begin{bmatrix} \frac{b_0}{a_n} & \frac{b_1}{a_n} & \dots & \frac{b_{n-1}}{a_n} \end{bmatrix}, D = \begin{bmatrix} \frac{b_n}{a_n} \end{bmatrix};$$

2. Наблюдаемая

$$A = \begin{bmatrix} 0 & 0 & \dots & -\frac{a_0}{a_n} \\ & & & -\frac{a_1}{a_n} \\ & I & & \vdots \\ & & & \frac{-a_{n-1}}{a_n} \end{bmatrix}, B = \begin{bmatrix} \frac{b_0}{a_n} \\ \frac{b_1}{a_n} \\ \vdots \\ \frac{b_{n-1}}{a_n} \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}, D = \begin{bmatrix} \frac{b_n}{a_n} \end{bmatrix};$$

3. Диагональная

   Раскладываем TF на сумму простейших дробей

$$W(p) = \sum_1^n \frac{c_i}{p - \lambda_i} = \sum_1^n \frac{\beta_i \times \gamma_i}{p - \lambda_i},$$

тогда матрицы имеют вид:

$$A = \begin{bmatrix} \lambda_1 & \dots & \dots & 0 \\ \vdots & \lambda_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \lambda_n \end{bmatrix}, B = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, C = \begin{bmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_n \end{bmatrix}, D = \begin{bmatrix} \frac{b_n}{a_n} \end{bmatrix};$$

В задании требуется каноническая управляемая форма, соответсвенно получаем:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -3 & -9 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 7 & 7 & 12 \end{bmatrix}, D = \begin{bmatrix} 0 \end{bmatrix};$$

2.2.2 Программная реализация

Из любопытства были использованы как встроенные средства моделирования системы, так и «ручное» перемножение матриц с маленьким шагом по времени.

```
# Задание начальных условий для ручного моделирования
initial_state = np.array([[0, 0, 0]], dtype=np.float64).T
stateSpaceManual_y = []

# Ручное моделирование
for t in ts:
    initial_state += (desired_form.A @ initial_state + desired_form.B * 1) * dt
    stateSpaceManual_y.append(desired_form.C @ initial_state)

# Авто-моделирование
stateSpaceAuto_y = control.forced_response(desired_form, U=1, X0=0, T=ts).outputs
```

2.2.3 Результаты

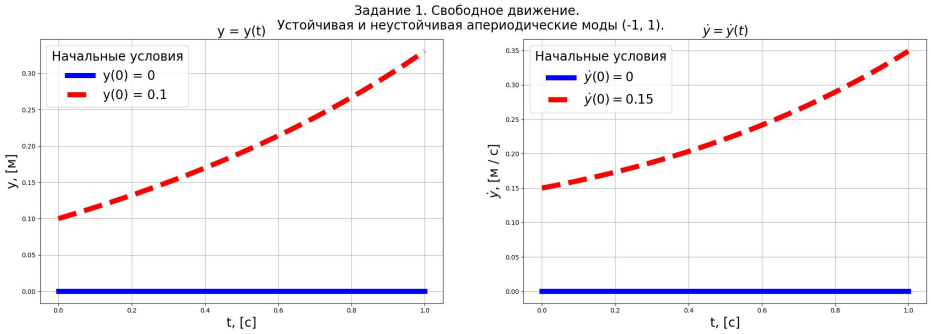

Рис. 4: Результаты второго задания.

По графикам видно, что, независимо от формы представления, при одинаковом входе система выдает одинаковый выход.

2.3 Задание 3. Многоканальная система в форме вход-выход.

2.3.1 Теория

Рассмотрим MIMO систему:

$$A(p)y(t) = B(p)u(t)$$

Для выражения y, перенесем матрицу $A(p)$ в правую часть:

$$y(t) = A(p)^{-1}B(p)u(t) = W(p)u(t),$$

где $W(p)$ - уже не просто TF-дробь, а матрица TF-дробей вида:

$$W(p) = \begin{bmatrix} W_{11}(p) & W_{12}(p) & \dots & W_{1p}(p) \\ W_{21}(p) & W_{22}(p) & \dots & W_{2p}(p) \\ \vdots & \vdots & \ddots & \vdots \\ W_{l1}(p) & W_{l2}(p) & \dots & W_{lp}(p) \end{bmatrix};$$

Подставив данные варианта, получаем:

$$W(p) = \frac{1}{12s+20}\begin{bmatrix} 2s-1 & s-4 \\ -2s+53 & -s+72 \end{bmatrix};$$

2.3.2 Программная реализация

```
# Входы от времени
u1 = np.zeros_like(ts) + 1
u2 = 2 * np.sin(ts)
mimo_Us = np.array([u1, u2])

# Симуляция
mimo_TF_y = control.forced_response(
    mimo_TF, U=mimo_Us, X0=0, T=ts
).outputs
```

2.3.3 Результаты

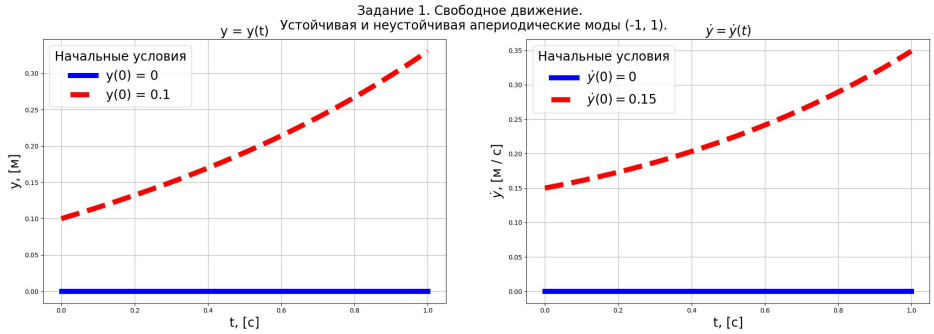

Рис. 5: Результаты третьего задания.

Видно, что графики выхода начинаются не в 0. Это связано с тем, что система не strictly-proper (то есть степени полиномов числителя и знаменятеля равны). Следствие этого заметно при переводе системы в формат SS:

$$A = \begin{bmatrix} -1.67 \end{bmatrix}, B = \begin{bmatrix} -0.607 & -0.794 \end{bmatrix}, C = \begin{bmatrix} 0.594 \\ -7.73 \end{bmatrix}, D = \begin{bmatrix} 0.167 & 0.0833 \\ -0.167 & -0.0833 \end{bmatrix};$$

Матрица связи D получается не нулевой, в связи с чем выход начинает зависеть не только от внутреннего состояния системы через матрицу наблюдения C, но и от входного взаимодействия через матрицу связи D.

2.4 Задание 4. Одноканальная система в форме вход-состояние-выход.

2.4.1 Теория

Необходимо промоделировать SISO систему в формате SS:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases},$$

где:

$$A = \begin{bmatrix} 0 & -9 \\ 1 & -6 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 5 \end{bmatrix}, C = \begin{bmatrix} 2 & 5 \end{bmatrix}, D = \begin{bmatrix} 0 \end{bmatrix};$$

2.4.2 Программная реализация

```
# Создание системы из матриц B-C-B
siso_SS = control.ss(task4_A, task4_B, task4_C, [0])

# Симуляция
small_ts = ts[:5000]
siso_SS_y = control.forced_response(siso_SS, U=1, X0=[0, 0], T=small_ts).outputs
```

2.4.3 Результаты

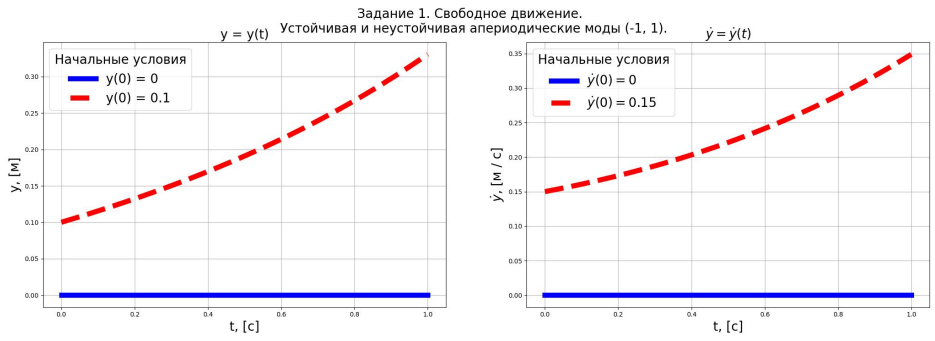

Рис. 6: Результаты четвертого задания.

2.5 Задание 5. Многоканальная система в форме вход-состояние-выход.

2.5.1 Теория

Необходимо промоделировать MIMO систему в формате SS:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases},$$

где:

$$A = \begin{bmatrix} 0 & -9 \\ 1 & -6 \end{bmatrix}; B = \begin{bmatrix} 1 & 4 \\ 3 & 5 \end{bmatrix}; C = \begin{bmatrix} 2 & 7 \\ 4 & 6 \end{bmatrix}; D = \begin{bmatrix} 0 \end{bmatrix};$$

2.5.2 Программная реализация

```
# Создание системы из матриц B-C-B
mimo_SS = control.ss(task5_A, task5_B, task5_C, 0)

# Симуляция
mimo_SS_y = control.forced_response(mimo_SS, U=mimo_Us, X0=[0, 0], T=ts).outputs
```

2.5.3 Результаты

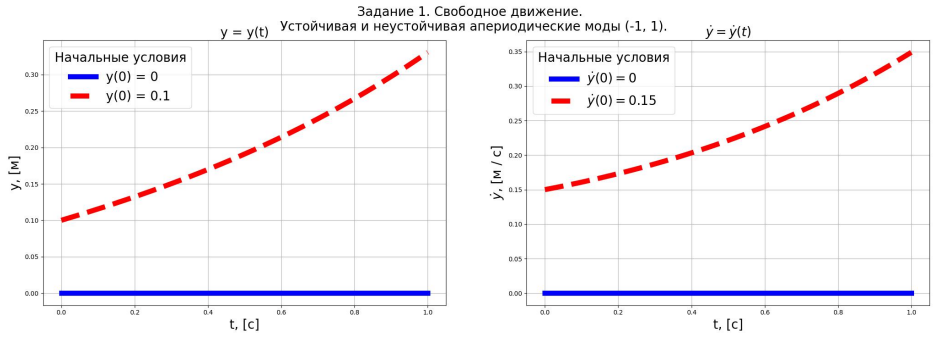

Рис. 7: Результаты пятого задания.

3 Заключение

В процессе работы была проведена симуляция в среде Python при помощи Python Control Systems Library для SISO и MIMO систем, данных в двух основных представлениях: TF и SS.

3.1 Выводы

1. Проведено моделирование SISO и MIMO Transfer Function систем с различным ненулевым входным воздействием.
2. Проведено моделирование SISO и MIMO State Space систем с различным ненулевым входным воздействием.
3. Выход системы не зависит от формы представления.
4. Не strictly proper системы могу давать выход отличный от 0 в момент времени t = 0 при 0 начальных условиях из-за ненулевой матрицы связи D.