

---

ЛР №1 «Формы представления линейных систем»

---

Отчет

Студент

Кирилл Лалаянц

R33352

336700

Вариант - 5

Преподаватель

Пашенко А.В.

Факультет Систем Управления и Робототехники

ИТМО

13.09.2023

## Содержание

1	Вводные данные	1
1.1	Цель работы . . . . .	1
1.2	Данные варианта. Инициализация необходимых переменных в Python	1
2	Выполнение работы	3
2.1	Задание 1. Одноканальная система в форме вход-выход. . . . .	3
2.1.1	Теория . . . . .	3
2.1.2	Программная реализация . . . . .	3
2.1.3	Результаты . . . . .	4
2.2	Задание 2. Переход от формы вход-выход к форме вход-состояние-выход. . . . .	5
2.2.1	Теория . . . . .	5
2.2.2	Программная реализация . . . . .	6
2.2.3	Результаты . . . . .	7
2.3	Задание 3. Многоканальная система в форме вход-выход. . . . .	8
2.3.1	Теория . . . . .	8
2.3.2	Программная реализация . . . . .	8
2.3.3	Результаты . . . . .	9
2.4	Задание 4. Одноканальная система в форме вход-состояние-выход. . . . .	10
2.4.1	Теория . . . . .	10
2.4.2	Программная реализация . . . . .	10
2.4.3	Результаты . . . . .	10
2.5	Задание 5. Многоканальная система в форме вход-состояние-выход. . . . .	11
2.5.1	Теория . . . . .	11
2.5.2	Программная реализация . . . . .	11
2.5.3	Результаты . . . . .	11

3	Заключение	12
3.1	Выводы . . . . .	12

## 1 Вводные данные

### 1.1 Цель работы

В этой работе пройдет изучение представления линейных систем в форме Transfer Function и State Space, а так же их дальнейшее моделирование.

### 1.2 Данные варианта. 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
6 sympy.init_printing()
7 p = sympy.Symbol("s")
8
9 dt = 0.001 # шаг симуляции
10 max_t = 50 # время симуляции
11 ts = np.linspace(0, max_t, int(max_t / dt))
```

---

Ввод данных варианта:

---

```
1 var_num = 6
2
3 # Task 1 & 2
4 a2, a1, a0, b2, b1, b0 = [9, 3, 6, 12, 7, 7]
5 # Task 3
6 A11 = p + 19
7 A12 = p + 3
8 A21 = p + 6
9 A22 = p + 2
10 task3_A = sympy.Matrix([[A11, A12], [A21, A22]])
11 B11 = 7
12 B12 = 7
13 B21 = 5
14 B22 = 6
15 task3_B = sympy.Matrix([[B11, B12], [B21, B22]])
16 # Task 4
```

```
17 task4_A = np.array([[0, -9], [1, -6]])
18 task4_B = np.array([[1], [5]])
19 task4_C = np.array([2, 5])
20 # Task 5
21 task5_A = np.array([[0, -9], [1, -6]])
22 task5_B = np.array([[1, 4], [3, 5]])
23 task5_C = np.array([[2, 7], [4, 6]])
```

---

## 2 Выполнение работы

### 2.1 Задание 1. Одноканальная система в форме вход-выход.

#### 2.1.1 Теория

Для моделирования системы сначала получим её TF из ДУ. Имеем:

$$\ddot{y} + a_2\dot{y} + a_1y + a_0y = b_2\ddot{u} + b_1\dot{u} + b_0u$$

Введем оператор дифференцирования по времени:

$$s \doteq \frac{d}{dt},$$

тогда получим:

$$s^3y + s^2a_2y + sa_1y + a_0y = s^2b_2u + sb_1u + b_0u$$

$$y = \frac{b_2s^2 + b_1s + b_0}{s^3 + a_2s^2 + a_1s + a_0}u = W(p)u$$

$$W(p) = \frac{b_2s^2 + b_1s + b_0}{s^3 + a_2s^2 + a_1s + a_0} = \frac{12s^2 + 7s + 7}{s^3 + 9s^2 + 3s + 6}$$

Полученная TF соответствует критерию физической реализуемости. Динамический порядок = 3; относительный динамический порядок = 1.

#### 2.1.2 Программная реализация

---

```
1 # Создание TF
2 den = [1, a2, a1, a0] # denominator
3 num = [b2, b1, b0] # numerator
4 transferFunction = control.tf(num, den)
5
6 # Моделирование с 0 начальными значениями
7 transferFunction_y = control.forced_response(transferFunction, U=1, X0=0, T=ts).outputs
```

---

## 2.1.3 Результаты

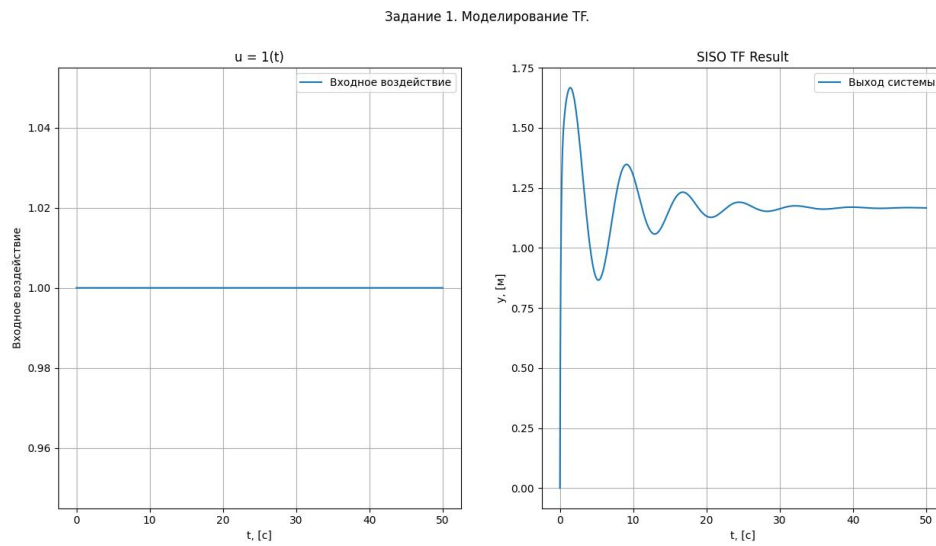


Рис. 1: Результаты первого задания.

На графике видно поведение системы.

## 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:

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

В задании требуется каноническая управляемая форма, соответственно получаем:

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

### 2.2.2 Программная реализация

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

---

```
1 # Задание начальных условий для ручного моделирования
2 initial_state = np.array([[0, 0, 0]], dtype=np.float64).T
3 stateSpaceManual_y = []
4
5 # Ручное моделирование
6 for t in ts:
7     initial_state += (desired_form.A @ initial_state + desired_form.B * 1) * dt
8     stateSpaceManual_y.append(desired_form.C @ initial_state)
9
10 # Авто-моделирование
11 stateSpaceAuto_y = control.forced_response(desired_form, U=1, X0=0, T=ts).outputs
```

---

### 2.2.3 Результаты

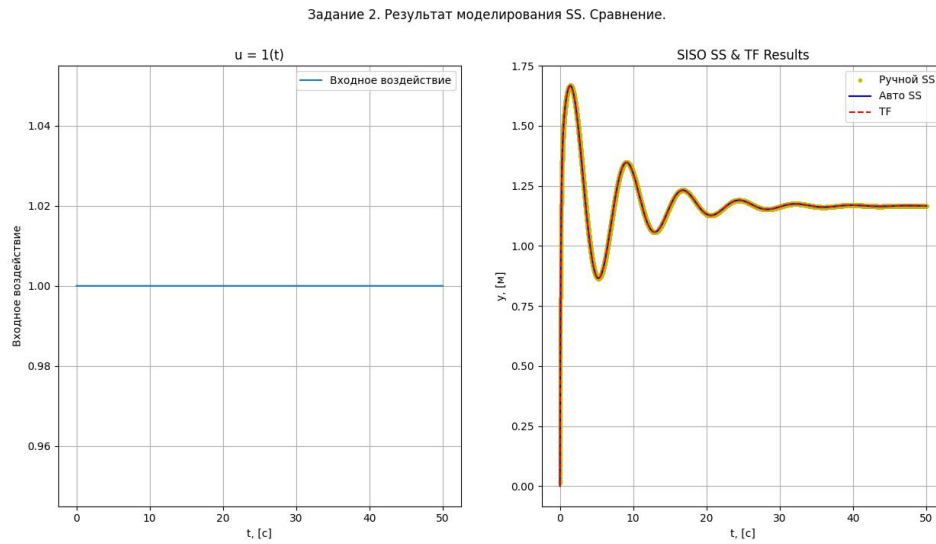


Рис. 2: Результаты второго задания.

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

---

```
1 # Входы от времени
2 u1 = np.zeros_like(ts) + 1
3 u2 = 2 * np.sin(ts)
4 mimo_Us = np.array([u1, u2])
5
6 # Симуляция
7 mimo_TF_y = control.forced_response(
8     mimo_TF, U=mimo_Us, X0=0, T=ts
9 ).outputs
```

---

## 2.3.3 Результаты

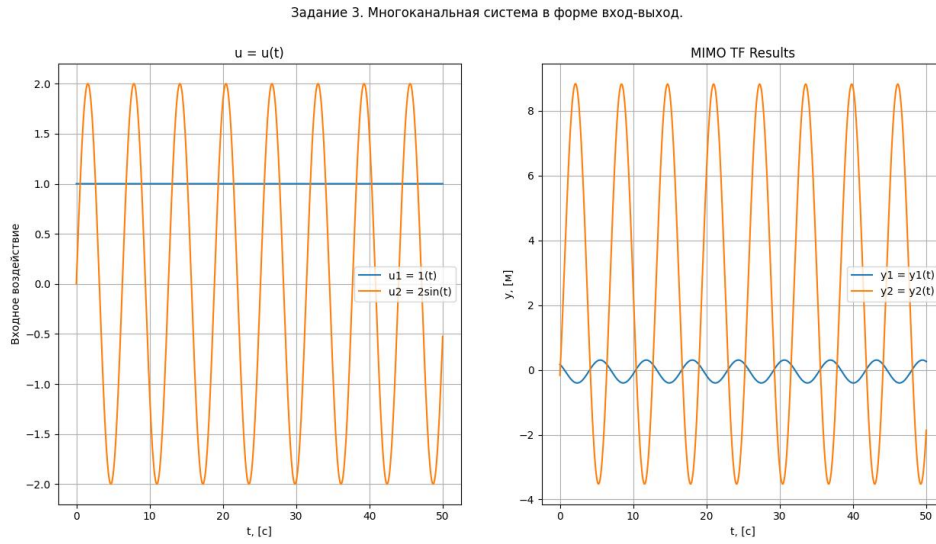


Рис. 3: Результаты третьего задания.

Видно, что графики выхода начинаются не в 0. Это связано с тем, что система не strictly-proper (то есть степени полиномов числителя и знаменателя 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

---

```

1 # Создание системы из матриц B-C-B
2 siso_SS = control.ss(task4_A, task4_B, task4_C, [0])
3
4 # Симуляция
5 small_ts = ts[:5000]
6 siso_SS_y = control.forced_response(siso_SS, U=1, X0=[0, 0], T=small_ts).outputs

```

---

### 2.4.3 Результаты

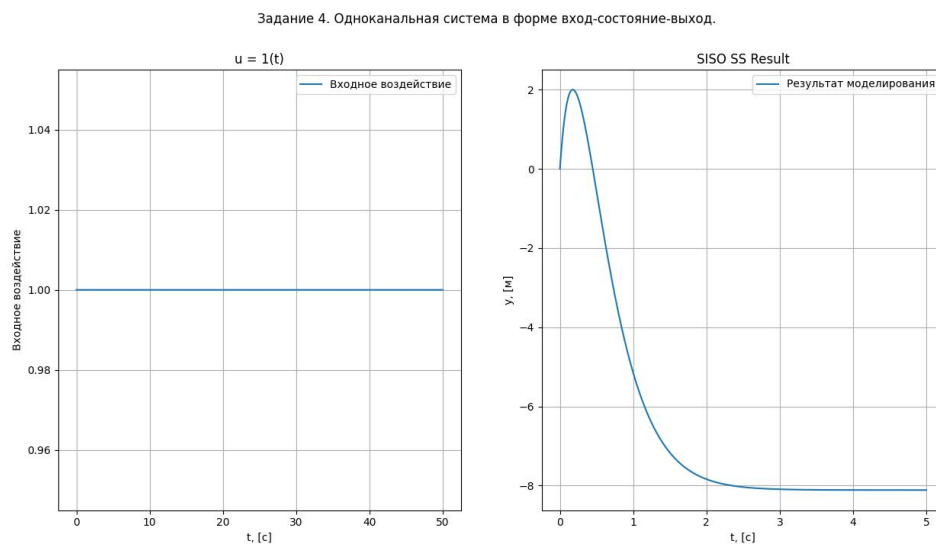


Рис. 4: Результаты четвертого задания.

## 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

---

```

1 # Создание системы из матриц B-C-B
2 mimo_SS = control.ss(task5_A, task5_B, task5_C, 0)
3
4 # Симуляция
5 mimo_SS_y = control.forced_response(mimo_SS, U=mimo_Us, X0=[0, 0], T=ts).outputs

```

---

### 2.5.3 Результаты

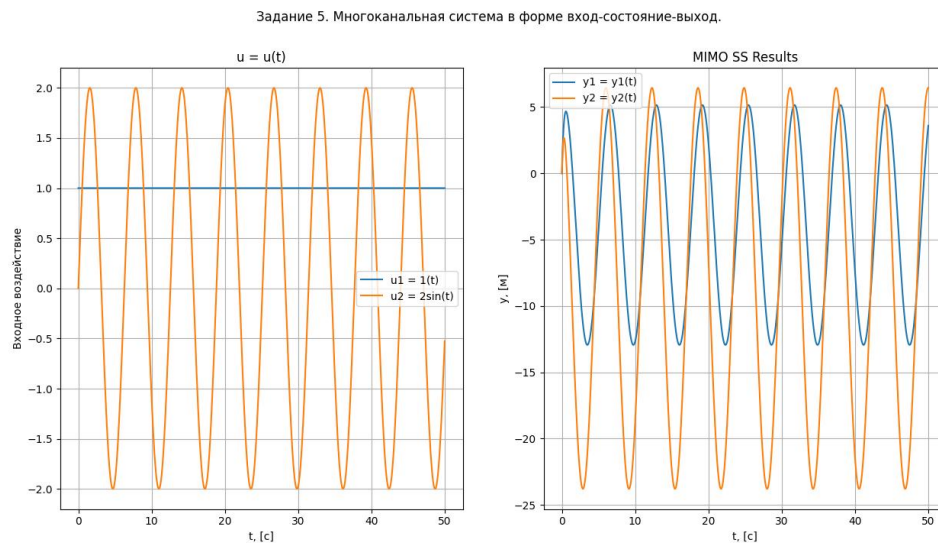


Рис. 5: Результаты пятого задания.

### 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 могут давать выход отличный от 0 в момент времени  $t = 0$  при 0 начальных условиях из-за ненулевой матрицы связи  $D$ .