

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Сибирский государственный университет телекоммуникаций и информатики»  
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"  
профиль "Программное обеспечение средств  
вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

**Лабораторная работа № 2**  
**по дисциплине**  
**«Алгоритмы и вычислительные методы оптимизации»**

Бригада 4

Выполнил:

студент гр. ИП- 111

«12» февраля 2024 г.

Корнилов А.А.

ФИО студента

Новосибирск 2024 г.

## Содержание

<b>1. Постановка задачи .....</b>	<b>3</b>
<b>2. Результат работы программы.....</b>	<b>4</b>
2.1. Предложенная система.....	4
2.2. Система для тестирования.....	8
<b>Приложение. Листинг .....</b>	<b>10</b>
Класс простых дробей.....	10
Основная программа .....	12

## 1. Постановка задачи

Написать программу, находящую все базисные решения системы линейных уравнений методом Жордана-Гаусса. Программа должна выводить промежуточные матрицы после каждого шага исключений и найденные базисные решения системы.

Должна иметься возможность быстро ввести входные данные для различного количества переменных и уравнений. Начальную работу программу необходимо продемонстрировать на предложенной ниже системе:

$$4. \begin{cases} 3x_1 - 8x_2 + 5x_3 - 6x_4 + 15x_5 = 35 \\ -9x_1 + 11x_2 + 19x_3 + 26x_4 + 21x_5 = 59 \\ 7x_1 + 4x_2 - 3x_3 + 16x_4 + 9x_5 = 19 \\ 36x_1 - 15x_2 - 18x_3 + 10x_4 + 36x_5 = 68 \end{cases}$$

Для получения максимальной оценки необходимо, чтобы все вычисления выполнялись в простых дробях. Для этого использовать класс простых дробей, реализованный в *лабораторной работе №1*.

## 2. Результат работы программы

### 2.1. Предложенная система

```
PS D:\Documents\SibSUTISIVI Semestr\AI\WMO\LR82> python .\main.py
| Метод Жордана-Гаусса |

Выберите способ матрицы:
1) Ввод матрицы с клавиатуры
2) Чтение матрицы из файла в директории "./matrices/"
Выход из программы - любой неперечисленный ввод
> 2
Введите название файла в формате .txt: 4
| НАЧАЛЬНАЯ МАТРИЦА |

3  -8  5  -6  15  35
-9  11  19  26  21  59
7   4  -3  16   9   19
36  -15 -18  18  36  68

| ПРОИЗШЛА ПЕРЕСТАНОВКА |

36  -15 -18  18  36  68
-9  11  19  26  21  59
7   4  -3  16   9   19
3  -8  5  -6  15  35

1  -5/12  -1/2  5/18  1  17/9
-9  11  19  26  21  59
7   4  -3  16   9   19
3  -8  5  -6  15  35

1  -5/12  -1/2  5/18  1  17/9
0  29/4  29/2  57/2  38  76
0  83/12  1/2  253/18  2  52/9
0  -27/4  13/2  -41/6  12  88/3

1  -5/12  -1/2  5/18  1  17/9
0  1  2  114/29  128/29  384/29
0  83/12  1/2  253/18  2  52/9
0  -27/4  13/2  -41/6  12  88/3

1  0  1/3  588/261  79/29  1633/261
0  1  2  114/29  128/29  384/29
0  0  -40/3  -3428/261  -772/29  -17416/261
0  0  20  1714/87  1158/29  8788/87

| ПРОИЗШЛА ПЕРЕСТАНОВКА |

1  0  1/3  588/261  79/29  1633/261
0  1  2  114/29  128/29  384/29
0  0  20  1714/87  1158/29  8788/87
0  0  -40/3  -3428/261  -772/29  -17416/261

1  0  1/3  588/261  79/29  1633/261
0  1  2  114/29  128/29  384/29
0  0  1  857/878  579/298  2177/435
0  0  -40/3  -3428/261  -772/29  -17416/261

1  0  0  1381/878  597/298  1996/435
0  1  0  853/435  21/145  286/435
0  0  1  857/878  579/298  2177/435
0  0  0  0  0  0

(0, 1, 2)
| НАЧАЛЬНАЯ МАТРИЦА |

1  0  0  1996/435
0  1  0  286/435
0  0  1  2177/435

1  0  0  1996/435
0  1  0  286/435
0  0  1  2177/435

1  0  0  1996/435
0  1  0  286/435
0  0  1  2177/435

1  0  0  1996/435
0  1  0  286/435
0  0  1  2177/435

Решение:
x1 = 1996/435 x2 = 286/435 x3 = 2177/435

(0, 1, 3)
| НАЧАЛЬНАЯ МАТРИЦА |

1  0  1381/878  1996/435
0  1  853/435  286/435
0  0  857/878  2177/435

1  0  1381/878  1996/435
0  1  853/435  286/435
0  0  857/878  2177/435

1  0  1381/878  1996/435
0  1  853/435  286/435
0  0  857/878  2177/435

1  0  1381/878  1996/435
0  1  853/435  286/435
0  0  1  4354/857

1  0  0  -2979/857
0  1  0  -8132/857
0  0  1  4354/857

Решение:
x1 = -2979/857 x2 = -8132/857 x3 = 4354/857
```

Скриншот 2.1.1. – Вычисление матрицы и начало нахождения базисов

(0, 1, 4)			
	НАЧАЛЬНАЯ	МАТРИЦА	
1	0	597/290	1996/435
0	1	21/145	206/435
0	0	579/290	2177/435
1	0	597/290	1996/435
0	1	21/145	206/435
0	0	579/290	2177/435
1	0	597/290	1996/435
0	1	21/145	206/435
0	0	579/290	2177/435
1	0	597/290	1996/435
0	1	21/145	206/435
0	0	1	4354/1737
1	0	0	-331/579
0	1	0	64/579
0	0	1	4354/1737
Решение:			
$x_1 = -331/579 \quad x_2 = 64/579 \quad x_3 = 4354/1737$			
(0, 2, 3)			
	НАЧАЛЬНАЯ	МАТРИЦА	
1	0	1381/870	1996/435
0	0	853/435	206/435
0	1	857/870	2177/435
1	0	1381/870	1996/435
0	0	853/435	206/435
0	1	857/870	2177/435
ПРОИЗОШЛА ПЕРЕСТАНОВКА			
1	0	1381/870	1996/435
0	1	857/870	2177/435
0	0	853/435	206/435
1	0	1381/870	1996/435
0	1	857/870	2177/435
0	0	853/435	206/435
1	0	1381/870	1996/435
0	1	857/870	2177/435
0	0	1	206/853
1	0	0	3587/853
0	1	0	4066/853
0	0	1	206/853
Решение:			
$x_1 = 3587/853 \quad x_2 = 4066/853 \quad x_3 = 206/853$			
(0, 2, 4)			
	НАЧАЛЬНАЯ	МАТРИЦА	
1	0	597/290	1996/435
0	0	21/145	206/435
0	1	579/290	2177/435
1	0	597/290	1996/435
0	0	21/145	206/435
0	1	579/290	2177/435
ПРОИЗОШЛА ПЕРЕСТАНОВКА			
1	0	597/290	1996/435
0	1	579/290	2177/435
0	0	21/145	206/435
1	0	597/290	1996/435
0	1	579/290	2177/435
0	0	21/145	206/435
1	0	597/290	1996/435
0	1	579/290	2177/435
0	0	1	206/63
1	0	0	-15/7
0	1	0	-32/21
0	0	1	206/63
Решение:			
$x_1 = -15/7 \quad x_2 = -32/21 \quad x_3 = 206/63$			
(0, 3, 4)			
	НАЧАЛЬНАЯ	МАТРИЦА	
1	1381/870	597/290	1996/435
0	853/435	21/145	206/435
0	857/870	579/290	2177/435
1	1381/870	597/290	1996/435
0	853/435	21/145	206/435
0	857/870	579/290	2177/435
1	1381/870	597/290	1996/435
0	1	63/853	206/853
0	857/870	579/290	2177/435

Скриншот 2.1.2. – Нахождение базисов (продолжение)

```

1 0 1656/853 3587/853
0 1 63/853 206/853
0 0 1641/853 4866/853

1 0 1656/853 3587/853
0 1 63/853 206/853
0 0 1 4866/1641

1 0 0 -331/547
0 1 0 32/547
0 0 1 4866/1641

Решение:
x1 = -331/547 x2 = 32/547 x3 = 4866/1641

(1, 2, 3)
НАЧАЛЬНАЯ МАТРИЦА
0 0 1381/870 1996/435
1 0 853/435 206/435
0 1 857/870 2177/435

ПРОИЗОШЛА ПЕРЕСТАНОВКА
1 0 853/435 206/435
0 0 1381/870 1996/435
0 1 857/870 2177/435

1 0 853/435 206/435
0 0 1381/870 1996/435
0 1 857/870 2177/435

ПРОИЗОШЛА ПЕРЕСТАНОВКА
1 0 853/435 206/435
0 1 857/870 2177/435
0 0 1381/870 1996/435

1 0 853/435 206/435
0 1 857/870 2177/435
0 0 1381/870 1996/435

1 0 853/435 206/435
0 1 857/870 2177/435
0 0 1 3992/1381

1 0 0 -7174/1381
0 1 0 2979/1381
0 0 1 3992/1381

Решение:
x1 = -7174/1381 x2 = 2979/1381 x3 = 3992/1381

(1, 2, 4)
НАЧАЛЬНАЯ МАТРИЦА
0 0 597/290 1996/435
1 0 21/145 206/435
0 1 579/290 2177/435

ПРОИЗОШЛА ПЕРЕСТАНОВКА
1 0 21/145 206/435
0 0 597/290 1996/435
0 1 579/290 2177/435

1 0 21/145 206/435
0 0 597/290 1996/435
0 1 579/290 2177/435

ПРОИЗОШЛА ПЕРЕСТАНОВКА
1 0 21/145 206/435
0 1 579/290 2177/435
0 0 597/290 1996/435

1 0 21/145 206/435
0 1 579/290 2177/435
0 0 597/290 1996/435

1 0 21/145 206/435
0 1 579/290 2177/435
0 0 1 3992/1791

1 0 0 30/199
0 1 0 331/597
0 0 1 3992/1791

Решение:
x1 = 30/199 x2 = 331/597 x3 = 3992/1791

(1, 3, 4)
НАЧАЛЬНАЯ МАТРИЦА
0 1381/870 597/290 1996/435
1 853/435 21/145 206/435
0 857/870 579/290 2177/435

ПРОИЗОШЛА ПЕРЕСТАНОВКА
1 853/435 21/145 206/435
0 1381/870 597/290 1996/435
0 857/870 579/290 2177/435

```

Скриншот 2.1.3. – Нахождение базисов (продолжение)

```

1 853/435 21/145 286/435
0 1381/870 597/290 1996/435
0 857/870 579/290 2177/435

```

```

1 853/435 21/145 286/435
0 1 1791/1381 3992/1381
0 857/870 579/290 2177/435

```

```

1 0 -3312/1381 -7174/1381
0 1 1791/1381 3992/1381
0 0 993/1381 2979/1381

```

```

1 0 -3312/1381 -7174/1381
0 1 1791/1381 3992/1381
0 0 1 3

```

```

1 0 0 2
0 1 0 -1
0 0 1 3

```

Решения:  
 $x_1 = 2 \quad x_2 = -1 \quad x_3 = 3$

(2, 3, 4)

```

0 1381/870 597/290 1996/435
0 853/435 21/145 286/435
1 857/870 579/290 2177/435

```

ПРОИЗОШЛА ПЕРЕСТАНОВКА

```

1 857/870 579/290 2177/435
0 853/435 21/145 286/435
0 1381/870 597/290 1996/435

```

```

1 857/870 579/290 2177/435
0 853/435 21/145 286/435
0 1381/870 597/290 1996/435

```

```

1 857/870 579/290 2177/435
0 1 63/853 286/853
0 1381/870 597/290 1996/435

```

```

1 0 1641/853 4866/853
0 1 63/853 286/853
0 0 1656/853 3587/853

```

```

1 0 1641/853 4866/853
0 1 63/853 286/853
0 0 1 3587/1656

```

```

1 0 0 331/552
0 1 0 15/184
0 0 1 3587/1656

```

Решения:  
 $x_1 = 331/552 \quad x_2 = 15/184 \quad x_3 = 3587/1656$

СОКРАЩЕННАЯ МАТРИЦА

```

1 0 0 1381/870 597/290 1996/435
0 1 0 853/435 21/145 286/435
0 0 1 857/870 579/290 2177/435
0 0 0 0 0 0

```

Результат вычислений

Общие решения:  
 $x_1 = 1381/870 * x_4 + 597/290 * x_5 = 1996/435$   
 $x_2 = 853/435 * x_4 + 21/145 * x_5 = 286/435$   
 $x_3 = 857/870 * x_4 + 579/290 * x_5 = 2177/435$

Базисные решения:  
 $x_1 = 1996/435$   
 $x_2 = 286/435$   
 $x_3 = 2177/435$   
 $x_4 = 0$   
 $x_5 = 0$

```

x1 = -2979/857
x2 = -8132/857
x3 = 0
x4 = 4354/857
x5 = 0

```

```

x1 = -331/579
x2 = 64/579
x3 = 0
x4 = 0
x5 = 4354/1737

```

```

x1 = 3587/853
x2 = 0
x3 = 4866/853
x4 = 286/853
x5 = 0

```

```

x1 = -15/7
x2 = 0
x3 = -32/21
x4 = 0
x5 = 286/63

```

```

x1 = -331/547
x2 = 0
x3 = 0
x4 = 32/547
x5 = 4866/1641

```

```

x1 = 0
x2 = -7174/1381
x3 = 2979/1381
x4 = 3992/1381
x5 = 0

```

```

x1 = 0
x2 = 38/199
x3 = 331/597
x4 = 0
x5 = 3992/1791

```

```

x1 = 0
x2 = 2
x3 = 0
x4 = -1
x5 = 3

```

```

x1 = 0
x2 = 0
x3 = 331/552
x4 = 15/184
x5 = 3587/1656

```

Скриншот 2.1.4. – Завершение нахождения базисов и вывод результатов вычисления программы

## 2.2. Система для тестирования

```
PS D:\Documents\5ibSUITIS\VI Semestr\AI\VMC\LR02> python .\main.py
Метод Жордана-Гаусса

Выберите способ матрицы:
1) Ввод матрицы с клавиатуры
2) Чтение матрицы из файла в директории "./matrices/"
Выход из программы - любой неперечисленный ввод
> 2
Введите название файла в формате .txt: task1
НАЧАЛЬНАЯ МАТРИЦА

2 3 -1 1 1
8 12 -9 8 3
4 6 3 -2 3
2 3 9 -7 3

ПРОИЗВОЛ ПЕРЕСТАНОВКА

8 12 -9 8 3
2 3 -1 1 1
4 6 3 -2 3
2 3 9 -7 3

1 3/2 -9/8 1 3/8
2 3 -1 1 1
4 6 3 -2 3
2 3 9 -7 3

1 3/2 -9/8 1 3/8
0 0 5/4 -1 1/4
0 0 15/2 -6 3/2
0 0 45/4 -9 9/4

ПРОИЗВОЛ ПЕРЕСТАНОВКА

1 3/2 -9/8 1 3/8
0 0 5/4 -1 1/4
0 0 45/4 -9 9/4
0 0 15/2 -6 3/2

1 3/2 -9/8 1 3/8
0 0 5/4 -1 1/4
0 0 1 -4/5 1/5
0 0 15/2 -6 3/2

1 3/2 0 1/10 3/5
0 0 0 0 0
0 0 1 -4/5 1/5
0 0 0 0 0

(0, 1)
НАЧАЛЬНАЯ МАТРИЦА

1 3/2 3/5
0 0 1/5

1 3/2 3/5
0 0 1/5

Нет решения

(0, 2)
НАЧАЛЬНАЯ МАТРИЦА

1 0 3/5
0 1 1/5

1 0 3/5
0 1 1/5

1 0 3/5
0 1 1/5

Решения:
x1 = 3/5 x2 = 1/5

(0, 3)
НАЧАЛЬНАЯ МАТРИЦА

1 1/10 3/5
0 -4/5 1/5

1 1/10 3/5
0 -4/5 1/5

1 1/10 3/5
0 1 -1/4

1 0 5/8
0 1 -1/4

Решения:
x1 = 5/8 x2 = -1/4
```

Скриншот 2.2.1. – Вычисление матрицы и нахождение базисов



```

(1, 2) | НАЧАЛЬНАЯ МАТРИЦА |
3/2  0  3/5
0    1  1/5

1  0  2/5
0  1  1/5

1  0  2/5
0  1  1/5

1  0  2/5
0  1  1/5

Решение:
x1 = 2/5  x2 = 1/5

(1, 3) | НАЧАЛЬНАЯ МАТРИЦА |
3/2  1/10  3/5
0    -4/5  1/5

1  1/5  2/5
0 -4/5  1/5

1  1/5  2/5
0 -4/5  1/5

1  1/5  2/5
0  1  -1/4

1  0  5/12
0  1 -1/4

Решение:
x1 = 5/12  x2 = -1/4

(2, 3) | НАЧАЛЬНАЯ МАТРИЦА |
0  1/10  3/5
1  -4/5  1/5

ПРОИЗВОД ПЕРЕСТАНОВКА

1  -4/5  1/5
0  1/10  3/5

1  -4/5  1/5
0  1/10  3/5

1  -4/5  1/5
0  1/10  3/5

1  -4/5  1/5
0  1  6

1  0  5
0  1  6

Решение:
x1 = 5  x2 = 6

СОКРАЩЕННАЯ МАТРИЦА

1  3/2  0  1/10  3/5
0  0  0  0  0
0  0  1  -4/5  1/5
0  0  0  0  0

РЕЗУЛЬТАТ ВЫЧИСЛЕНИЙ

Общие решения:
x1 = 3/2 * x2 + 1/10 * x4 - 3/5
x3 = 4/5 * x4 + 1/5

Базисные решения:
x1 = 3/5
x2 = 0
x3 = 1/5
x4 = 0

x1 = 5/8
x2 = 0
x3 = 0
x4 = -1/4

x1 = 0
x2 = 2/5
x3 = 1/5
x4 = 0

x1 = 0
x2 = 5/12
x3 = 0
x4 = -1/4

x1 = 0
x2 = 0
x3 = 5
x4 = 6

```

Скриншот 2.2.2. – Нахождение базисов (продолжение)  
и вывод результатов вычисления программы

## Приложение. Листинг

### Класс простых дробей

*fraction.py*

```
import math

class Fraction:
    __slots__ = ("_numerator", "_denominator")

    def __init__(self, numerator=0, denominator=1):
        if type(numerator) is not int or type(denominator) is not int:
            raise TypeError(
                "Fraction(%s, %s) - значения числителя и знаменателя должны быть
целыми числами"
                % (numerator, denominator)
            )
        if denominator == 0:
            raise ZeroDivisionError("Fraction(%s, 0)" % numerator)
        g = math.gcd(numerator, denominator)
        if denominator < 0:
            g = -g
        numerator //= g
        denominator //= g
        self._numerator = numerator
        self._denominator = denominator

    # Сумма двух дробей
    def __add__(self, other):
        if isinstance(other, Fraction):
            return Fraction(
                self._numerator * other._denominator
                + other._numerator * self._denominator,
                self._denominator * other._denominator,
            )
        return NotImplemented

    # Разность двух дробей
    def __sub__(self, other):
        if isinstance(other, Fraction):
            return Fraction(
                self._numerator * other._denominator
                - other._numerator * self._denominator,
                self._denominator * other._denominator,
            )
        return NotImplemented

    # Произведение двух дробей
    def __mul__(self, other):
        if isinstance(other, Fraction):
            return Fraction(
                self._numerator * other._numerator,
                self._denominator * other._denominator,
            )
        return NotImplemented
```

```

# Частное двух дробей
def __truediv__(self, other):
    if isinstance(other, Fraction):
        return Fraction(
            self._numerator * other._denominator,
            self._denominator * other._numerator,
        )
    return NotImplemented

# x < y
def __lt__(self, other):
    if isinstance(other, Fraction):
        return (
            self._numerator * other._denominator
            < other._numerator * self._denominator
        )
    return NotImplemented

# x <= y
def __le__(self, other):
    if isinstance(other, Fraction):
        return (
            self._numerator * other._denominator
            <= other._numerator * self._denominator
        )
    return NotImplemented

# x == y
def __eq__(self, other):
    if isinstance(other, Fraction):
        return (
            self._numerator * other._denominator
            == other._numerator * self._denominator
        )
    return NotImplemented

# x != y
def __ne__(self, other):
    if isinstance(other, Fraction):
        return (
            self._numerator * other._denominator
            != other._numerator * self._denominator
        )
    return NotImplemented

# x > y
def __gt__(self, other):
    if isinstance(other, Fraction):
        return (
            self._numerator * other._denominator
            > other._numerator * self._denominator
        )
    return NotImplemented

# x >= y
def __ge__(self, other):
    if isinstance(other, Fraction):

```

```

        return (
            self._numerator * other._denominator
            >= other._numerator * self._denominator
        )
    return NotImplemented

def __repr__(self):
    if self._denominator == 1:
        return "Fraction(%s)" % self._numerator
    else:
        return "Fraction(%s, %s)" % (self._numerator, self._denominator)

def __str__(self):
    if self._denominator == 1:
        return str(self._numerator)
    else:
        return "%s/%s" % (self._numerator, self._denominator)

def getABS(self):
    return Fraction(abs(self._numerator), abs(self._denominator))

```

## Основная программа

*main.py*

```

from lib.fraction import Fraction
import itertools

def printMatrix(matrix):
    s = [str(e) for e in row] for row in matrix]
    lens = [max(map(len, col)) for col in zip(*s)]
    fmt = "{}:{}".format(x) for x in lens
    table = [fmt.format(*row) for row in s]
    print("\n".join(table))
    print("\n")

def methodJordanGauss(matrix, flag=True):
    print(f"{'-'*14} НАЧАЛЬНАЯ МАТРИЦА {'-'*14}\n")
    printMatrix(matrix)
    # По столбцам
    for c in range(len(matrix)):
        index = c
        # По элементам столбца
        for i in range(c + 1, len(matrix)):
            if matrix[index][c].getABS() < matrix[i][c].getABS():
                index = i
        if index != c:
            matrix[index], matrix[c] = matrix[c], matrix[index]
            print(f"{'-'*12} ПРОИЗОШЛА ПЕРЕСТАНОВКА {'-'*12}\n")
            printMatrix(matrix)
        if matrix[c][c] == Fraction(0):
            continue

```

```

# Сокращение строки
if matrix[c][c] != Fraction(1):
    matrix[c] = [i / matrix[c][c] for i in matrix[c]]
    printMatrix(matrix)
# По всем строкам для обнуления
for i in range(len(matrix)):
    if matrix[i][c] == Fraction(0) or i == c:
        continue
    coefficient = matrix[i][c] * Fraction(-1)
    # По элементам строк, начиная с c-ого
    for j in range(c, len(matrix[0])):
        matrix[i][j] = matrix[i][j] + matrix[c][j] * coefficient
    printMatrix(matrix)

countNullStr = 0
for i in matrix:
    nullSumFlag = True
    for j in i[:-1]:
        if j != Fraction(0):
            nullSumFlag = False
            break
    # Если элементы строки нулевые и значение не нулевое
    if nullSumFlag and i[-1] != Fraction(0):
        countNullStr = 0
        break
    # Если строка нулевая
    elif nullSumFlag and i[-1] == Fraction(0):
        continue
    countNullStr += 1

if not countNullStr:
    return None
elif countNullStr == len(matrix) and countNullStr == len(matrix[0]) - 1:
    return [[matrix[i][-1] for i in range(len(matrix))]]
else:
    result = [[], []]
    # Общее решение
    for i in matrix:
        tmpSum = Fraction(0)
        for j in i:
            tmpSum += j.getABS()
        if tmpSum != Fraction(0):
            result[0].append(i)
    matrix = result[0]
    # Базисное решение
    if flag:
        for i in itertools.combinations(
            [i for i in range(len(matrix[0]) - 1)], countNullStr
        ):
            print("—" * 50)
            print(" ", i)
            tmpResult = [0 for i in range(len(matrix[0]) - 1)]
            tmpMatrix = []
            # Получение необходимых столбцов, как строк
            for j in i:
                tmpMatrix.append([x[j] for x in matrix])
            tmpMatrix.append([x[-1] for x in matrix])

```

```

# Транспонирование
tmpMatrix = [list(j) for j in zip(*tmpMatrix)]
result_r = methodJordanGauss(tmpMatrix, False)
if not result_r:
    print("Нет решения")
elif len(result_r) == 1:
    print("Решение:")
    ans = result_r[0]
    for j in range(len(ans)):
        print(" x{} = {}".format(j + 1, ans[j]), end="")
    t = 0
    for j in i:
        tmpResult[j] = ans[t]
        t += 1
    print("\n")
    result[1].append(tmpResult)
elif len(result_r) == 2:
    print("\nСЛАУ имеет множество решений")
return result

def main__choice(matrix):
    print(
        f"{'-'*8}- Метод Жордана-Гаусса -{'-'*8}\n",
        "\n Выберите способ матрицы:\n",
        " 1) Ввод матрицы с клавиатуры\n",
        " 2) Чтение матрицы из файла в директории './matrices/'\n",
        " Выход из программы - любой неперечисленный ввод",
    )
    answer = input(" > ")
    if answer == "1":
        R = int(input("Введите количество строк: "))
        for row in range(R):
            a = list(map(int, input().strip().split(" ")))
            matrix.append(list(map(Fraction, a)))
        return methodJordanGauss(matrix)
    elif answer == "2":
        filename = str(input("Введите название файла в формате .txt: "))
        try:
            f = open("matrices/" + filename + ".txt", "r")
        except:
            print(' \033[91m\033[1m[-]\033[0m Не удалось открыть файл "input.txt".')
            return
        for line in f:
            a = list(map(int, line.strip().split(" ")))
            matrix.append(list(map(Fraction, a)))
        return methodJordanGauss(matrix)

def main():
    matrix = []
    result = main__choice(matrix)

    print(f"{'-'*14}- СОКРАЩЕННАЯ МАТРИЦА -{'-'*14}\n")
    printMatrix(matrix)
    print(f"{'-'*14}- Результат вычислений -{'-'*14}\n")

```

```

if not result:
    print("Нет решения")
elif len(result) == 1:
    print("Решение:")
    ans = result[0]
    for i in range(len(ans)):
        print("  x{} = {}".format(i + 1, ans[i]))
    print("\n")
elif len(result) == 2:
    o_ans = result[0]
    ans = result[1]

    print("Общее решение:")
    for i in o_ans:
        tmp_str = " "
        for j in range(len(i) - 1):
            tmp = i[j]
            if tmp != Fraction(0):
                if tmp.getABS() != Fraction(1):
                    if tmp < Fraction(0):
                        tmp_str += " - "
                    else:
                        tmp_str += " + "
                tmp_str += str(tmp.getABS()) + " * "
            tmp_str += "x" + str(j + 1)
        tmp_str += " = " + str(i[-1])
        print(tmp_str)

    print("\nБазисные решения:")
    for i in ans:
        for j in range(len(i)):
            print("  x{} = {}".format(j + 1, i[j]))
        print()
    else:
        print("\033[91m\033[1m[-]\033 Произшла не предвиденная ошибка!")
    return

if __name__ == "__main__":
    main()

```