

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

Расчетно-графическая работа №3
Вариант №3

По дисциплине
Линейная алгебра

Выполнили:
Стафеев И.А., Борисевич А.В.,
Румянцев М.В., Корчагин Р.П.

Поток: ЛИН АЛГ ИКТ 17.3

Проверила
Шиманская Г.С.

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ	3
1 Техническое задание	4
2 Теоретическая информация.....	5
3 Программа для решения СЛАУ методом Гаусса	7
3.1 Ввод пользователя	7
3.2 Преобразование матрицы к ступенчатому виду	7
3.3 Проверка на наличие решений	9
3.4 Получение свободных переменных	9
3.5 Получение решений для системы с единственным реше- нием	10
3.6 Получение решений для системы с множеством решений	11
3.7 Итоговая функция	13
4 Примеры работы программы.....	15
4.1 Пример 1	15
4.2 Пример 2	16
4.3 Пример 3	17
4.4 Пример 4	18
4.5 Пример 5	19
4.6 Пример 6	20
4.7 Пример 7	21
4.8 Пример 8	22
5 Оценка эффективности алгоритма	23
ЗАКЛЮЧЕНИЕ	25
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	26

ВВЕДЕНИЕ

В расчетно-графической работе №3 необходимо создать программу для решения систем линейных алгебраических уравнений методом Гаусса.

Требуется написать техническое задание для программы, написать и прокомментировать код, привести примеры выполнения программы при решении систем линейных уравнений и произвести оценку эффективности написанного алгоритма.

1 Техническое задание

В расчётно-графической работе №3 требуется написать программу, решающую систему линейных алгебраических уравнений методом Гаусса и поддерживающую систему до 4 уравнений с 4 неизвестными. В ходе анализа предметной области были выявлены следующие требования к разрабатываемой программе:

1. Программа должна поддерживать пользовательский ввод систем линейных уравнений;
2. Программа должна обрабатывать все возможные случаи систем линейных уравнений;
3. Программа должна решать произвольные введенные пользователем системы; соответствующие любому из определенных случаев систем уравнений;
4. Программа должна выводить промежуточные и конечные результаты выполнения программы

Для удовлетворения требований необходимо решить следующие задачи:

1. Ознакомиться с теоретическими выкладками относительно устройства и особенностей применения метода Гаусса;
2. Написать код программы, реализующий решение системы линейных уравнений, введенную пользователем, методом Гаусса;
3. Провести тестирование программы на входных данных, соответствующих возможным сценариям работы, сверить результаты с достоверными и при необходимости исправить ошибки.

Программа будет написана на языке программирования Python и будет работать в консольном режиме. При желании пользователь может заносить системы уравнений в исходный код программы и запускать его с помощью интерпретатора. Исходный код программы доступен в репозитории [1].

2 Теоретическая информация

Метод Гаусса:

Матрицей системы линейных алгебраических уравнений является матрица, состоящая из коэффициентов при неизвестных уравнения. Столбец свободных членов содержит свободные члены системы. Матрицу, состоящую из матрицы системы и столбца свободных членов, называют расширенной матрицей системы - именно с ней работает метод Гаусса и написанная программа. Пример СЛАУ и соответствующей ей расширенной матрицы представлен ниже.

$$\begin{cases} x_1 + x_2 + x_3 = 0 \\ 2x_1 + x_2 = 4 \\ x_1 - x_2 - 2x_3 = 5 \end{cases} \quad \left(\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 4 \\ 1 & -1 & -2 & 5 \end{array} \right)$$

Метод Гаусса для решения СЛАУ заключается в приведении матрицы к ступенчатому виду, путём элементарных преобразований строк данной матрицы (привести к ступенчатому виду можно любую матрицу). Под элементарными преобразованиями понимается перемена мест двух строк в данной матрице, умножение строки на произвольное число, отличное от нуля, прибавление к одной строке другой строки, умноженной на некоторое число.

В процессе приведения матрицы к ступенчатому виду производятся преобразования строк матрицы с целью уменьшения числа неизвестных (то есть обнуления элементов строк) в строках до момента, пока не удастся однозначно определить значение одной из переменных (или установить, что какие-то переменные могут принимать любые значения).

Пример эквивалентных преобразований, с помощью которых достигается ступенчатый вид расширенной матрицы, представлен ниже.

$$\left(\begin{array}{cccc|c} 1 & 2 & 3 & -2 & 1 \\ 2 & -1 & -2 & -3 & 2 \\ 3 & 2 & -1 & 2 & -5 \\ 2 & -3 & 2 & 1 & 11 \end{array} \right) \sim \left(\begin{array}{cccc|c} 1 & 2 & 3 & -2 & 1 \\ 0 & -5 & -8 & 1 & 0 \\ 0 & -4 & -10 & 8 & -8 \\ 0 & -7 & -4 & 5 & 9 \end{array} \right) \sim$$

$$\sim \left(\begin{array}{cccc|c} 1 & 2 & 3 & -2 & 1 \\ 0 & -5 & -8 & 1 & 0 \\ 0 & 0 & -\frac{18}{5} & \frac{36}{5} & -8 \\ 0 & 0 & \frac{36}{5} & \frac{18}{5} & 9 \end{array} \right) \sim \left(\begin{array}{cccc|c} 1 & 2 & 3 & -2 & 1 \\ 0 & -5 & -8 & 1 & 0 \\ 0 & 0 & -\frac{18}{5} & \frac{36}{5} & -8 \\ 0 & 0 & 0 & 18 & -7 \end{array} \right)$$

Матрица такого вида соответствует более простой системе уравнений, решение которой начинается с решения последнего уравнения, затем результат подставляется в предпоследнее уравнение и т.д.

Однако возможны случаи, когда система не имеет решений или имеет бесконечное число решений. Первому случаю соответствует наличие строки вида $(0 \ 0 \ \dots \ 0 \ | \ a)$, $a \neq 0$ в приведенной матрице. Во втором случае у системы будут свободные переменные (в приведенной матрице это будут соответствующие столбцы таких элементов, лежащих на главной диагонали матрицы, что эти элементы равны 0). В этом случае столбцы свободных переменных необходимо перенести в конец матрицы, поменяв знаки их элементов (что соответствует перенесению членов в правую часть исходных уравнений), а затем теми же равносильными преобразованиями привести матрицу к виду, когда подматрица до столбца свободных членов будет единичной. Тогда все несвободные (базисные) переменные можно выразить через свободные переменные и свободные члены.

Если в исходной системе уравнений количество переменных не превышает количества уравнений, то у системы может не быть решения, может быть одно решение или быть множество решений. Если в исходной системе количество неизвестных больше количества уравнений, то эта система либо не имеет решений, либо имеет бесконечно много решений (единственного решения быть не может, так как там всегда будут свободные переменные). Значит, всего существует 8 различных случаев систем уравнений, которые программа должна корректно обрабатывать.

3 Программа для решения СЛАУ методом Гаусса

3.1 Ввод пользователя

При запуске программы пользователь должен ввести количество уравнений и количество неизвестных в системе, а затем ввести уравнения системы в формате $a_1 \ a_2 \ \dots \ a_n \mid b_1$, где a_i - коэффициент при x_i , а b_j - свободный член j -го уравнения. Чтобы в консоли все дробные числа отображались в виде обыкновенных дробей, а не вещественных чисел (что нагляднее и удобнее), было принято решение ограничить допустимые вводимые числа только целыми. Однако, если пользователю необходимо решить СЛАУ с нецелыми коэффициентами, то он должен запустить программу из исходного кода, внося в него самостоятельно данные, а не как исполняемый exe-файл. Листинг этой части программы не приводится (поскольку к самому алгоритму решения СЛАУ отношения не имеет), однако в репозитории разработки [1] он присутствует (как и весь код целиком). Пример пользовательского ввода представлен на рисунке 1

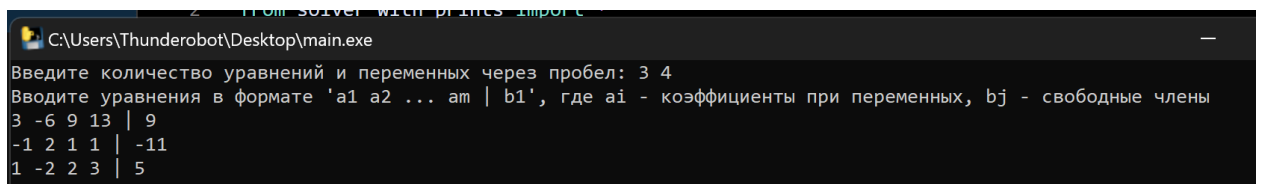


Рисунок 1 — Пример пользовательского ввода

3.2 Преобразование матрицы к ступенчатому виду

Первым этапом работы программы является приведение матрицы к ступенчатому виду. В цикле, идущем по номерам столбцов, на каждой итерации нужно из строк, лежащих ниже текущей, вычесть текущую строку, умноженную на коэффициент так, чтобы все элемент в этом столбце ниже текущей строки обнулились. Для этого в соответствующей функции (1) на каждой итерации находится строка с ненулевым элементом в текущем столбце (чтобы принять ее за текущую и вычитать ее из других строк). Если такая строка не найдена, то программа перейдет к рассмотрению

следующего столбца матрицы. Иначе выполняются описанные ранее действия, и номер текущей строки увеличивается на 1. В результате работы функции исходная матрица приводится к ступенчатой.

```
def get_row_echelon_form(matrix):  
    """Функция для приведения матрицы к ступенчатому виду"""  
    n, m = matrix.shape  
    cur_row = 0  
    for i in range(m):  
        # Найдем строку с ненулевым элементом, чтоб вычитать ее из строк ниже  
        non_zero_row = find_non_zero_row(matrix, cur_row, i)  
        if non_zero_row == -1:  
            continue  
        # Поставим строку с ненулевым элементом на место текущей строки  
        matrix[[cur_row, non_zero_row]] = matrix[[non_zero_row, cur_row]]  
        # Поделим строку на число так, чтобы на диагонали был единичный элемент  
        matrix[cur_row] /= matrix[cur_row][i]  
        for j in range(cur_row + 1, n):  
            # Из всех строк ниже текущей вычтем текущую, умноженную на коэффициент  
            matrix[j] -= matrix[j][i] * matrix[cur_row]  
        cur_row += 1  
    return matrix
```

Listing 1: Функция для преобразования матрицы к ступенчатому виду

Функция для нахождения строки с ненулевым элементом в текущем столбце приведена в листинге 2. В этой функции в цикле рассматриваются все строки не выше текущей, и для каждой проверяется элемент в текущем столбце.

```
def find_non_zero_row(matrix, cur_row, col):  
    """Функция для нахождения строки не выше cur_row, где в столбце col  
    содержится ненулевой элемент"""  
    for i in range(cur_row, matrix.shape[0]):  
        if matrix[i][col] != 0:  
            return i  
    return -1
```

Listing 2: Функция для нахождения строки с ненулевым элементом в текущем столбце

3.3 Проверка на наличие решений

Следующим этапом работы программы является проверка на наличие решений для полученной ранее ступенчатой матрицы. Эта функция приведена в листинге 3

```
def check_if_any_solutuion_exists(matrix):  
    """Функция для проверки наличия решений СЛАУ"""  
    for i in range(matrix.shape[0]):  
        # Если в приведенной матрице есть строка вида  
        # '0 0 ... 0 | n', n != 0, то решений нет  
        if np.count_nonzero(matrix[i]) == 1:  
            return False  
    return True
```

Listing 3: Функция для проверки наличия решений СЛАУ

Функция в цикле проходит по всем строкам матрицы, и если какая-то строка состоит только из нулей и последнего элемента, не равного нулю, то система решений не имеет, для обозначения чего возвращается соответствующий булевый флаг. Если решений системы нет, то программа прекращает свою работу

3.4 Получение свободных переменных

Если система имеет решения, необходимо определить свободные переменные, чтобы установить, сколько решений имеет система. Соответствующая функция приведена в листинге 4.

```

def find_non_basic_variables(matrix):
    """Функция для нахождения свободных переменных - тех,
    которые не лежат на ступеньках матрицы"""
    n, m = matrix.shape
    m -= 1
    row = 0
    non_basic = []
    for col in range(m):
        if matrix[row][col] != 0:
            row += 1
        else:
            non_basic.append(col)
    if row == matrix.shape[0]:
        # случай, когда уравнений меньше, чем переменных
        non_basic.extend(list(range(n, m)))
        break
    return non_basic

```

Listing 4: Функция для получения свободных переменных в СЛАУ

Функция проходит в цикле по номерам столбцов. Переменные, которым соответствуют элементы, с которых начинается "ступенька" матрицы, являются базисными, остальные являются свободными. Отдельно необходимо учесть случай, когда количество переменных больше числа уравнений, так как тогда в свободные переменные надо внести еще и те номера столбцов, когда "ступеньки" матрицы закончились.

3.5 Получение решений для системы с единственным решением

Если количество свободных переменных, найденное на предыдущем шаге, равно нулю, то система имеет единственное решение. Для нахождения корней написана функция, приведенная в листинге 5.

```

def back_substitution_for_one_solution(matrix):
    """Функция для нахождения корней, когда СЛАУ имеет единственное решение"""
    sm = matrix[np.any(matrix != 0, axis=1)]
    n = sm.shape[0]
    x = np.zeros(n, dtype=np.int64) + Fraction()
    for m in range(n - 1, -1, -1):
        # Обычное нахождение корней: переносим все известное в "правую часть"
        x[m] = Fraction(sm[m][-1] - np.dot(sm[m][m:-1], x[m:]))
    return x

```

Listing 5: Функция для получения корней системы с единственным решением

В функции создается массив для корней, который заполняется с конца. В цикле, начиная с последнего уравнений системы, последовательно находятся корни. Из последней строки однозначно можно установить последнюю переменную, затем при помощи подстановки из предпоследней однозначно устанавливается предпоследняя переменная и так далее.

3.6 Получение решений для системы с множеством решений

Если количество свободных переменных не равно нулю, то систем имеет множество решений, которые выражаются через свободные переменные. В соответствии с определенным в теоретической части алгоритмом сначала необходимо перенести столбцы свободных переменных в конец матрицы, что делает функция в листинге 6.

```

def shift_columns_to_the_end(matrix, columns):
    """Функция для перемещения определенных столбцов в конец матрицы.
    По сути это перемещение чисел в правую часть уравнений"""
    wthoc = np.delete(matrix, columns, axis=1)
    wthc = -matrix[:, columns]
    return np.concatenate([wthoc, wthc], axis=1)

```

Listing 6: Функция для перемещения столбцов свободных переменных в конец матрицы

Затем приведенную матрицу необходимо еще раз преобразовать так, чтобы в "левой" части матрицы, где находятся столбцы базисных переменных, была единичная подматрица. За это отвечает функция, представленная в листинге 7.

```
def back_substitution_for_infty_solutions(matrix, n, non_basic_variables):  
    """Функция для нахождения решения СЛАУ с бесконечным количеством решений"""  
    # Перенесем свободные переменные в правую часть и оставим ненулевые строки  
    sm = shift_columns_to_the_end(matrix, non_basic_variables)  
    sm = sm[np.any(sm != 0, axis=1)]  
    # Приведем матрицу так, чтобы "левая" часть (с неизвестными переменными)  
    # стала единичной  
    m = sm.shape[0]  
    for i in range(m - 2, -1, -1):  
        for j in range(i + 1, m):  
            sm[i] -= sm[i][j] * sm[j]  
  
    return sm
```

Listing 7: Функция для преобразования матрицы для определения базисных переменных

Эта функция после вызова функции для перемещения столбцов в конец матрицы в цикле, начиная с последней строки, вычитает из нее все строки ниже, умноженные на соответствующие коэффициенты так, чтобы после итерации эта строка стала строкой единичной матрицы.

Функция возвращает матрицу, из которой можно выразить все базисные переменные через свободные. Для этого создана функция, приведенная в листинге 8.

```

def represent_root(coef, free_vars):
    """Функция, выводящая корень уравнения, зависящий от свободных переменных"""
    ans = [str(coef[0])]
    for i, j in zip(coef[1:], free_vars):
        if i == 0:
            continue
        sign = "-" if i < 0 else "+"
        elem = f"{abs(i)}*x{j+1}" if abs(i) != 1 else f"x{j+1}"
        ans.extend([sign, elem])
    return " ".join(ans)

```

Listing 8: Функция для перемещения столбцов свободных переменных в конец матрицы

Функция принимает список коэффициентов для неизвестной переменной ("правую" часть строки матрицы) и список номеров свободных переменных. Затем функция составляет строку, представляющую собой запись корня системы уравнений через свободный член и базисные переменные.

3.7 Итоговая функция

Затем все шаги алгоритма необходимо объединить в одну функцию, которая будет принимать массив коэффициентов и массив свободных членов, и будет выполнять упомянутые шаги - приведение матрицы к ступенчатому виду, проверку наличия решений, определение свободных переменных, нахождение корней - и затем будет выводить найденные корни (или сообщать об отсутствии решений). Код этой функции приведен в листинге 9.

```

def gaussian_elimination(A, b=None):
    """Функция, решающая СЛАУ методом Гаусса"""
    if b is None:
        ext_matrix = A.copy()
    else:
        ext_matrix = np.concatenate([A, b], axis=1)

    ext_matrix = ext_matrix.astype(np.int64) + Fraction()
    # Приведем матрицу к ступенчатому виду
    REM = get_row_echelon_form(ext_matrix)

    # Проверка, есть ли решения у СЛАУ
    if not check_if_any_solutuion_exists(REM):
        print("Решений нет!")
        return

    # Найдём свободные переменные
    non_basic_variables = find_non_basic_variables(REM)
    if not non_basic_variables:
        # Если свободных переменных нет, то решение одно
        print("Решение одно")
        roots = back_substitution_for_one_solution(REM)
        for i in range(len(roots)):
            print(f"x{i+1} = {roots[i]}")
    else:
        # Если свободные переменные есть, то решений бесконечно много
        print("Решений бесконечно много")
        str_free_vars = ", ".join([f"x{i+1}" for i in non_basic_variables])
        print("Свободные переменные -", str_free_vars)
        coefs = back_substitution_for_infty_solutions(
            REM, REM.shape[1] - 1, non_basic_variables
        )
        basic_variables = [i for i in range(REM.shape[1] - 1)
                           if i not in non_basic_variables]
        for i in range(coefs.shape[0]):
            root = represent_root(
                coefs[i][coefs.shape[1] - 1 - len(non_basic_variables):],
                non_basic_variables
            )
            print(f"x{basic_variables[i] + 1} = {root}")

```

Listing 9: Функция для решения СЛАУ методом Гаусса

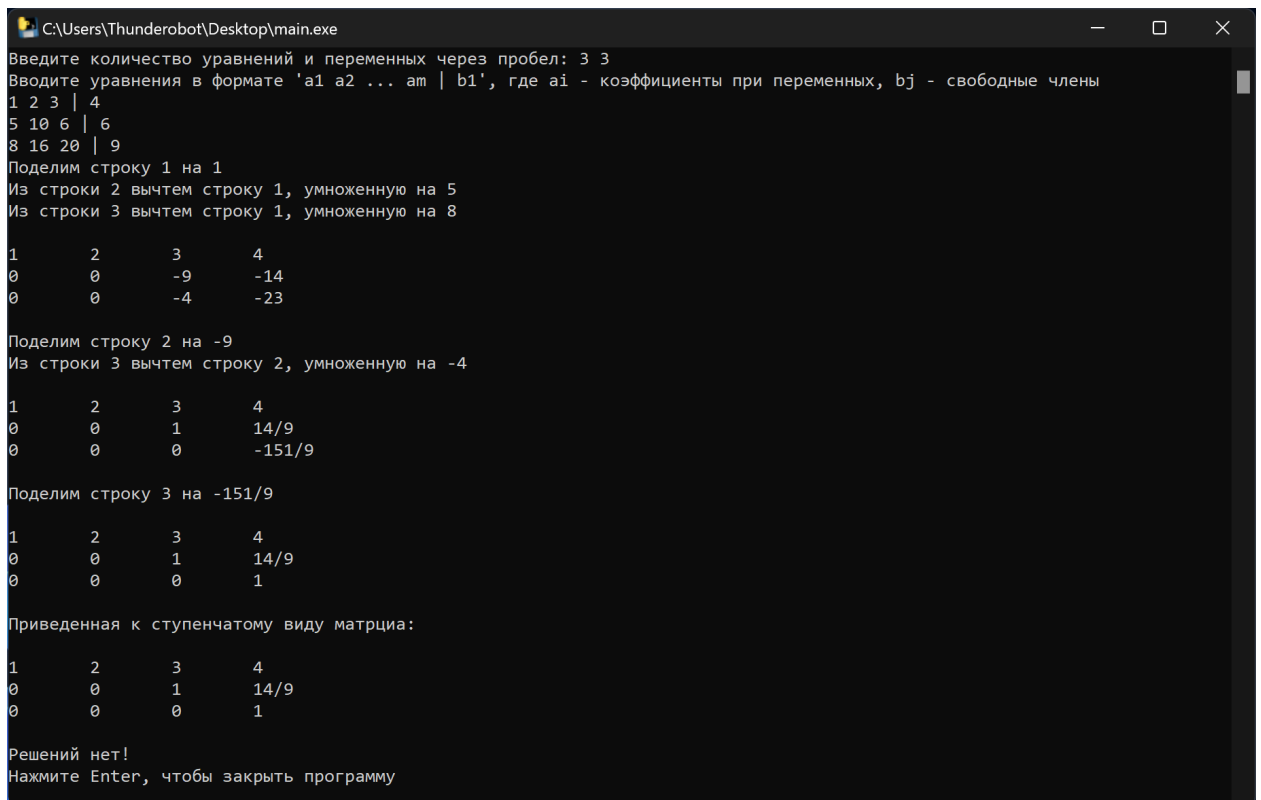
4 Примеры работы программы

Здесь будут приведены примеры работы программы для всех возможных случаев СЛАУ (всего их 8). Пусть n - количество уравнений в системе, m - количество неизвестных.

4.1 Пример 1

На рисунке 2 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n = m$, решений у системы нет):

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4 \\ 5x_1 + 10x_2 + 6x_3 = 6 \\ 8x_1 + 16x_2 + 20x_3 = 9 \end{cases}$$



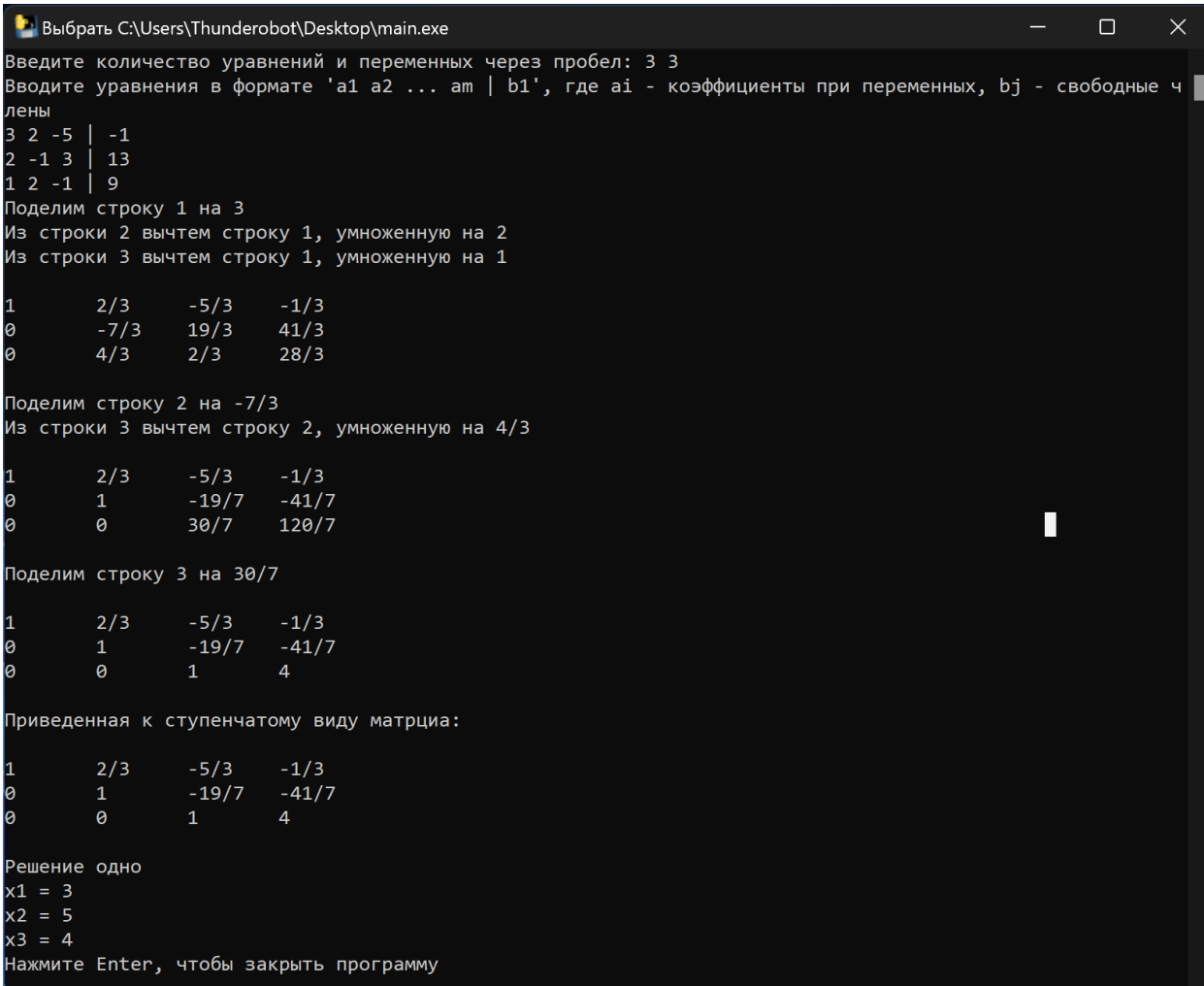
```
C:\Users\Thunderobot\Desktop\main.exe
Введите количество уравнений и переменных через пробел: 3 3
Введите уравнения в формате 'a1 a2 ... am | b1', где ai - коэффициенты при переменных, bj - свободные члены
1 2 3 | 4
5 10 6 | 6
8 16 20 | 9
Поделите строку 1 на 1
Из строки 2 вычтем строку 1, умноженную на 5
Из строки 3 вычтем строку 1, умноженную на 8
1      2      3      4
0      0     -9     -14
0      0     -4     -23
Поделите строку 2 на -9
Из строки 3 вычтем строку 2, умноженную на -4
1      2      3      4
0      0      1     14/9
0      0      0    -151/9
Поделите строку 3 на -151/9
1      2      3      4
0      0      1     14/9
0      0      0      1
Приведенная к ступенчатому виду матрица:
1      2      3      4
0      0      1     14/9
0      0      0      1
Решений нет!
Нажмите Enter, чтобы закрыть программу
```

Рисунок 2 — Пример №1 работы программы при $n = m$ и отсутствии решений

4.2 Пример 2

На рисунке 3 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n = m$, система имеет одно решение):

$$\begin{cases} 3x_1 + 2x_2 - 5x_3 = -1 \\ 2x_1 - x_2 + 3x_3 = 13 \\ x_1 + 2x_2 - x_3 = 9 \end{cases}$$



```
Выбрать C:\Users\Thunderobot\Desktop\main.exe
Введите количество уравнений и переменных через пробел: 3 3
Вводите уравнения в формате 'a1 a2 ... am | b1', где ai - коэффициенты при переменных, bj - свободные члены
3 2 -5 | -1
2 -1 3 | 13
1 2 -1 | 9
Поделим строку 1 на 3
Из строки 2 вычтем строку 1, умноженную на 2
Из строки 3 вычтем строку 1, умноженную на 1

1      2/3      -5/3      -1/3
0      -7/3      19/3      41/3
0      4/3       2/3       28/3

Поделим строку 2 на -7/3
Из строки 3 вычтем строку 2, умноженную на 4/3

1      2/3      -5/3      -1/3
0      1       -19/7     -41/7
0      0       30/7      120/7

Поделим строку 3 на 30/7

1      2/3      -5/3      -1/3
0      1       -19/7     -41/7
0      0        1         4

Приведенная к ступенчатому виду матрица:

1      2/3      -5/3      -1/3
0      1       -19/7     -41/7
0      0        1         4

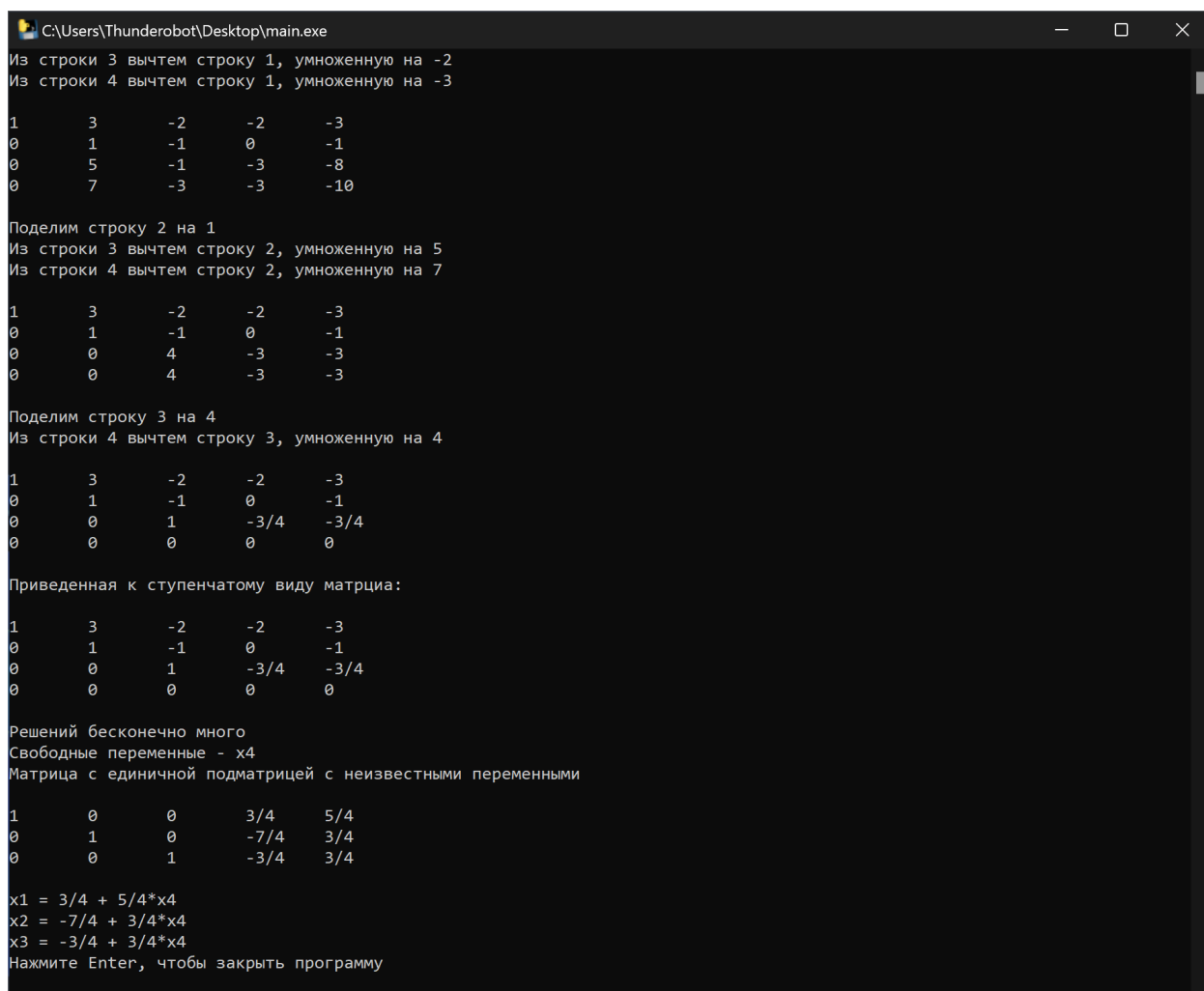
Решение одно
x1 = 3
x2 = 5
x3 = 4
Нажмите Enter, чтобы закрыть программу
```

Рисунок 3 — Пример №2 работы программы при $n = m$ и единственном решении

4.3 Пример 3

На рисунке 4 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n = m$, система имеет бесконечно много решений):

$$\begin{cases} x_1 + 3x_2 - 2x_3 - 2x_4 = -3 \\ -x_1 - 2x_2 + x_3 + 2x_4 = 2 \\ -2x_1 - x_2 + 3x_3 = 3x_4 = -2 \\ -3x_1 - 2x_2 + 3x_3 + 3x_4 = -1 \end{cases}$$



```
C:\Users\Thunderobot\Desktop\main.exe
Из строки 3 вычтем строку 1, умноженную на -2
Из строки 4 вычтем строку 1, умноженную на -3

1      3      -2      -2      -3
0      1      -1      0      -1
0      5      -1      -3      -8
0      7      -3      -3     -10

Поделим строку 2 на 1
Из строки 3 вычтем строку 2, умноженную на 5
Из строки 4 вычтем строку 2, умноженную на 7

1      3      -2      -2      -3
0      1      -1      0      -1
0      0      4      -3      -3
0      0      4      -3      -3

Поделим строку 3 на 4
Из строки 4 вычтем строку 3, умноженную на 4

1      3      -2      -2      -3
0      1      -1      0      -1
0      0      1     -3/4     -3/4
0      0      0      0      0

Приведенная к ступенчатому виду матрица:

1      3      -2      -2      -3
0      1      -1      0      -1
0      0      1     -3/4     -3/4
0      0      0      0      0

Решений бесконечно много
Свободные переменные - x4
Матрица с единичной подматрицей с неизвестными переменными

1      0      0      3/4      5/4
0      1      0     -7/4      3/4
0      0      1     -3/4      3/4

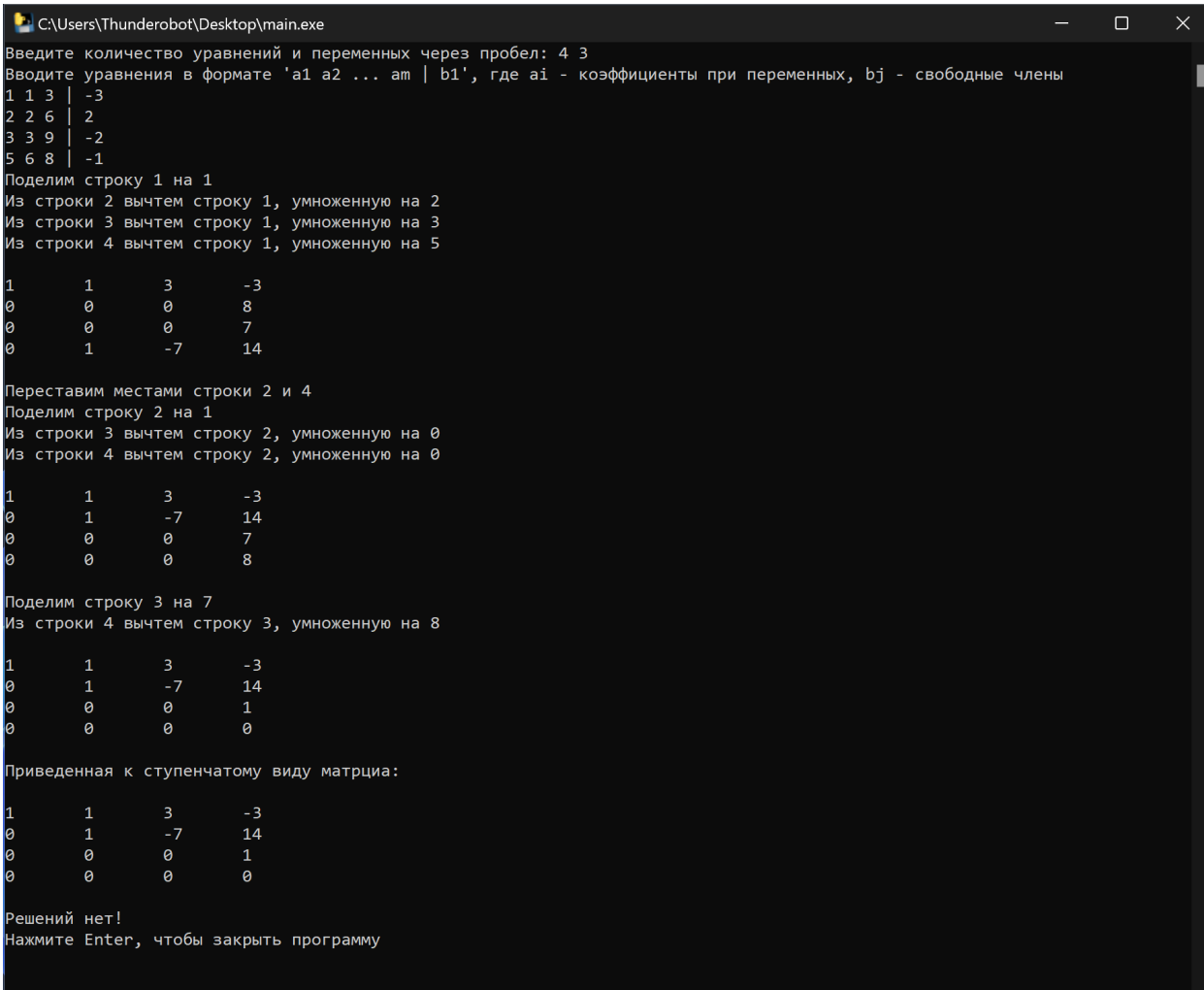
x1 = 3/4 + 5/4*x4
x2 = -7/4 + 3/4*x4
x3 = -3/4 + 3/4*x4
Нажмите Enter, чтобы закрыть программу
```

Рисунок 4 — Пример №3 работы программы при $n = m$ и бесконечном количестве решений

4.4 Пример 4

На рисунке 5 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n > m$ и решений у системы нет):

$$\begin{cases} x_1 + x_2 + 3x_3 = -3 \\ 2x_1 + 2x_2 + 6x_3 = 2 \\ 3x_1 + 3x_2 + 9x_3 = -2 \\ 5x_1 + 6x_2 + 8x_3 = -1 \end{cases}$$



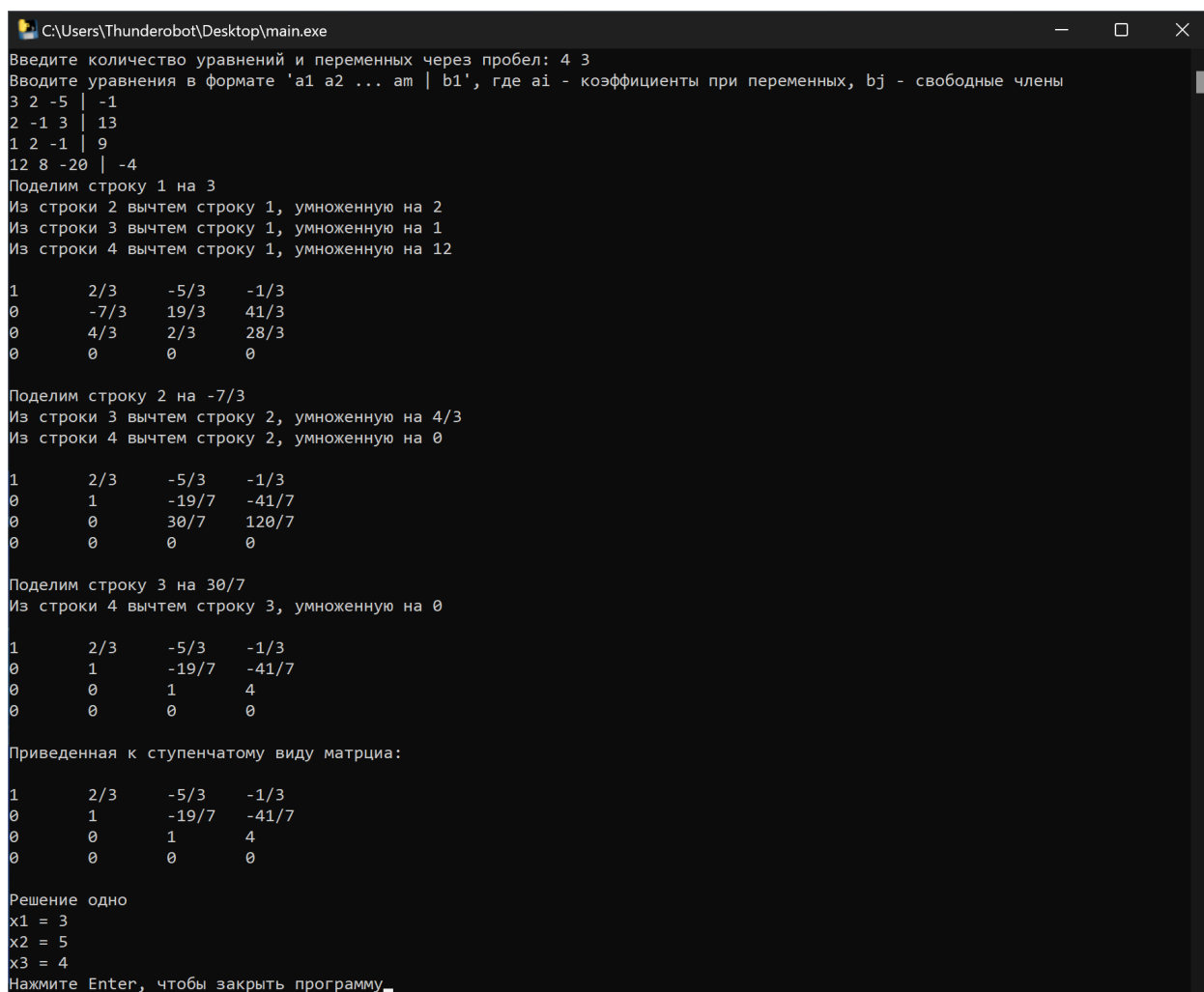
```
C:\Users\Thunderobot\Desktop\main.exe
Введите количество уравнений и переменных через пробел: 4 3
Введите уравнения в формате 'a1 a2 ... am | b1', где ai - коэффициенты при переменных, bj - свободные члены
1 1 3 | -3
2 2 6 | 2
3 3 9 | -2
5 6 8 | -1
Поделим строку 1 на 1
Из строки 2 вычтем строку 1, умноженную на 2
Из строки 3 вычтем строку 1, умноженную на 3
Из строки 4 вычтем строку 1, умноженную на 5
1      1      3      -3
0      0      0      8
0      0      0      7
0      1     -7     14
Переставим местами строки 2 и 4
Поделим строку 2 на 1
Из строки 3 вычтем строку 2, умноженную на 0
Из строки 4 вычтем строку 2, умноженную на 0
1      1      3      -3
0      1     -7     14
0      0      0      7
0      0      0      8
Поделим строку 3 на 7
Из строки 4 вычтем строку 3, умноженную на 8
1      1      3      -3
0      1     -7     14
0      0      0      1
0      0      0      0
Приведенная к ступенчатому виду матрица:
1      1      3      -3
0      1     -7     14
0      0      0      1
0      0      0      0
Решений нет!
Нажмите Enter, чтобы закрыть программу
```

Рисунок 5 — Пример №4 работы программы при $n > m$ и отсутствии решений

4.5 Пример 5

На рисунке 6 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n > m$, система имеет одно решение):

$$\begin{cases} 3x_1 + 2x_2 - 5x_3 = -1 \\ 2x_1 - x_2 + 3x_3 = 13 \\ x_1 + 2x_2 - x_3 = 9 \\ 12x_1 + 8x_2 - 20x_3 = -4 \end{cases}$$



```
C:\Users\Thunderobot\Desktop\main.exe
Введите количество уравнений и переменных через пробел: 4 3
Введите уравнения в формате 'a1 a2 ... am | b1', где ai - коэффициенты при переменных, bj - свободные члены
3 2 -5 | -1
2 -1 3 | 13
1 2 -1 | 9
12 8 -20 | -4
Поделим строку 1 на 3
Из строки 2 вычтем строку 1, умноженную на 2
Из строки 3 вычтем строку 1, умноженную на 1
Из строки 4 вычтем строку 1, умноженную на 12
1      2/3      -5/3      -1/3
0      -7/3      19/3      41/3
0      4/3       2/3      28/3
0      0        0        0

Поделим строку 2 на -7/3
Из строки 3 вычтем строку 2, умноженную на 4/3
Из строки 4 вычтем строку 2, умноженную на 0
1      2/3      -5/3      -1/3
0      1      -19/7     -41/7
0      0      30/7      120/7
0      0        0        0

Поделим строку 3 на 30/7
Из строки 4 вычтем строку 3, умноженную на 0
1      2/3      -5/3      -1/3
0      1      -19/7     -41/7
0      0        1        4
0      0        0        0

Приведенная к ступенчатому виду матрица:
1      2/3      -5/3      -1/3
0      1      -19/7     -41/7
0      0        1        4
0      0        0        0

Решение одно
x1 = 3
x2 = 5
x3 = 4
Нажмите Enter, чтобы закрыть программу_
```

Рисунок 6 — Пример №5 работы программы при $n > m$ и единственном решении

4.6 Пример 6

На рисунке 7 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n > m$, система имеет бесконечное число решений):

$$\begin{cases} x_1 + 3x_2 - 2x_3 - 2x_4 = -3 \\ -x_1 - 2x_2 + x_3 + 2x_4 = 2 \\ -2x_1 - x_2 + 3x_3 + x_4 = -2 \\ -3x_1 - 2x_2 + 3x_3 + 3x_4 = -1 \\ -6x_1 - 4x_2 + 6x_3 + 6x_4 = -2 \end{cases}$$

```

C:\Users\Thunderobot\Desktop\main.exe
0      5      -1      -3      -8
0      7      -3      -3     -10
0     14     -6     -6     -20

Поделим строку 2 на 1
Из строки 3 вычтем строку 2, умноженную на 5
Из строки 4 вычтем строку 2, умноженную на 7
Из строки 5 вычтем строку 2, умноженную на 14

1      3      -2      -2      -3
0      1     -1       0      -1
0      0       4     -3      -3
0      0       4     -3      -3
0      0       8     -6      -6

Поделим строку 3 на 4
Из строки 4 вычтем строку 3, умноженную на 4
Из строки 5 вычтем строку 3, умноженную на 8

1      3      -2      -2      -3
0      1     -1       0      -1
0      0       1    -3/4    -3/4
0      0       0       0       0
0      0       0       0       0

Приведенная к ступенчатому виду матрица:

1      3      -2      -2      -3
0      1     -1       0      -1
0      0       1    -3/4    -3/4
0      0       0       0       0
0      0       0       0       0

Решений бесконечно много
Свободные переменные - x4
Матрица с единичной подматрицей с неизвестными переменными

1      0       0       3/4    5/4
0      1       0      -7/4    3/4
0      0       1     -3/4    3/4

x1 = 3/4 + 5/4*x4
x2 = -7/4 + 3/4*x4
x3 = -3/4 + 3/4*x4
Нажмите Enter, чтобы закрыть программу

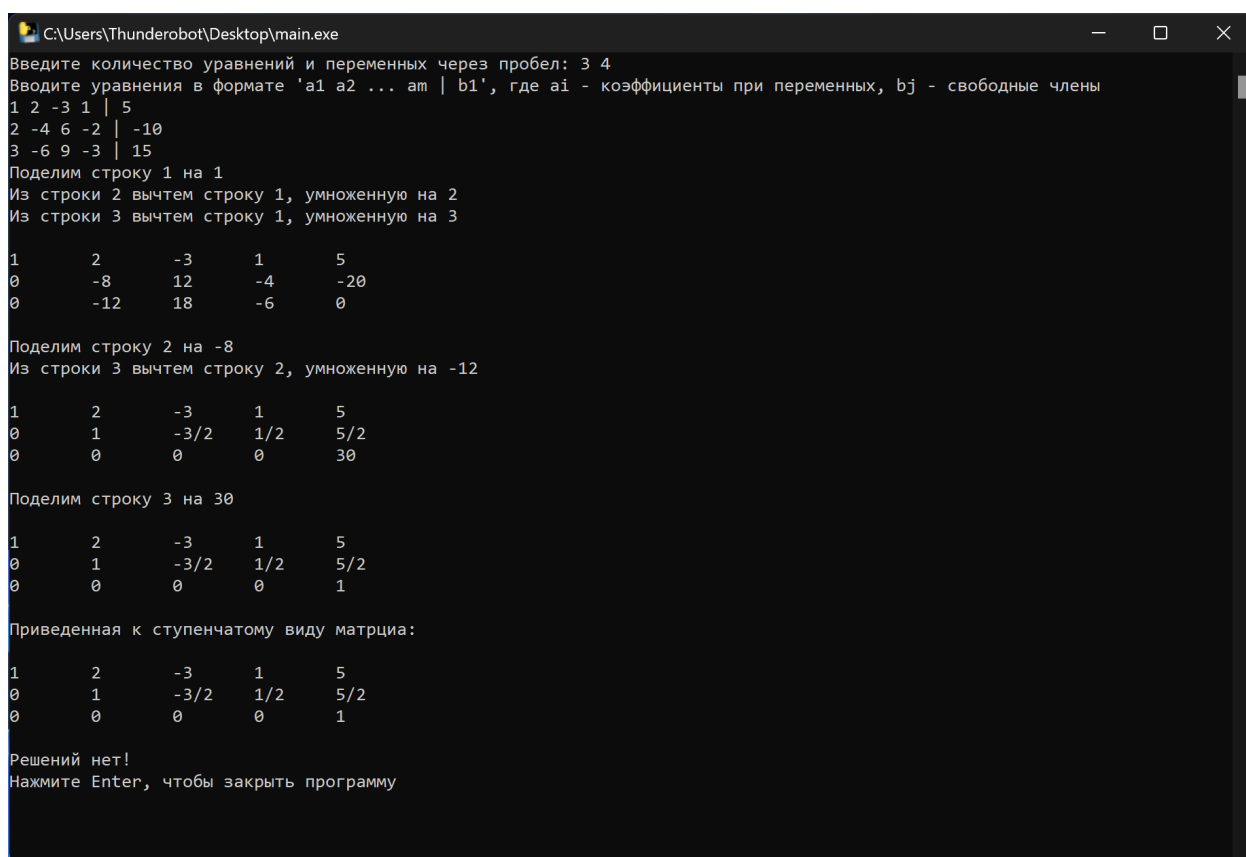
```

Рисунок 7 — Пример №6 работы программы при $n > m$ и бесконечном количестве решений

4.7 Пример 7

На рисунке 8 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n < m$, у системы решений нет):

$$\begin{cases} x_1 + 2x_2 - 3x_3 + x_4 = 5 \\ 2x_2 - 4x_3 + 6x_4 = -10 \\ 3x_1 - 6x_2 + 9x_3 - 3x_4 = 15 \end{cases}$$



```
C:\Users\Thunderobot\Desktop\main.exe
Введите количество уравнений и переменных через пробел: 3 4
Введите уравнения в формате 'a1 a2 ... am | b1', где ai - коэффициенты при переменных, bj - свободные члены
1 2 -3 1 | 5
2 -4 6 -2 | -10
3 -6 9 -3 | 15
Поделим строку 1 на 1
Из строки 2 вычтем строку 1, умноженную на 2
Из строки 3 вычтем строку 1, умноженную на 3

1      2      -3      1      5
0      -8      12     -4     -20
0      -12     18     -6      0

Поделим строку 2 на -8
Из строки 3 вычтем строку 2, умноженную на -12

1      2      -3      1      5
0      1     -3/2    1/2    5/2
0      0      0      0     30

Поделим строку 3 на 30

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

Приведенная к ступенчатому виду матрица:

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

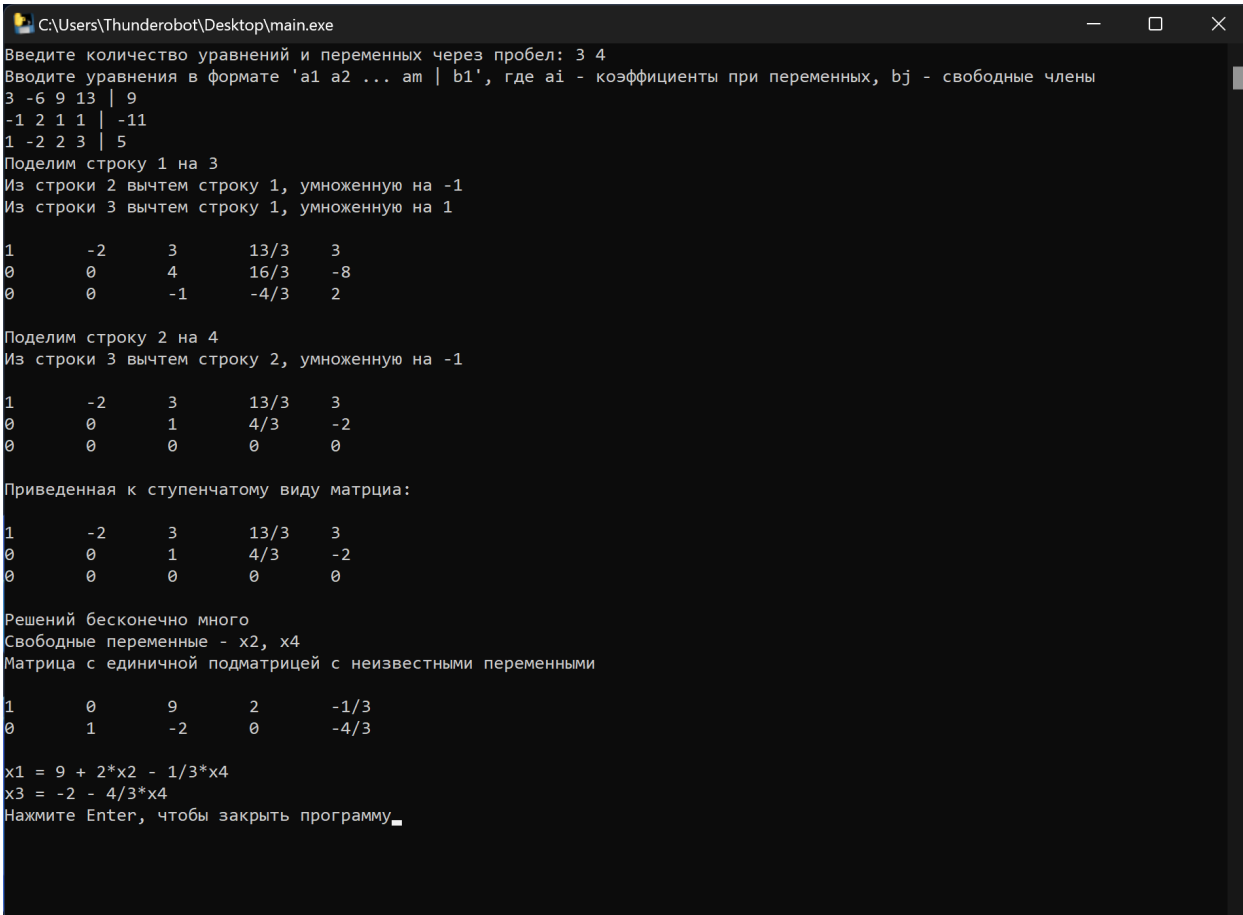
Решений нет!
Нажмите Enter, чтобы закрыть программу
```

Рисунок 8 — Пример №7 работы программы при $n < m$ и отсутствии решений

4.8 Пример 8

На рисунке 9 приведен пример выполнения написанной программы при решении следующей СЛАУ ($n < m$, решений у системы бесконечно много):

$$\begin{cases} 3x_1 - 6x_2 + 9x_3 + 13x_4 = 9 \\ -x_1 + 2x_2 + x_3 + x_4 = -11 \\ x_1 - 2x_2 + 2x_3 + 3x_4 = 5 \end{cases}$$



```
C:\Users\Thunderobot\Desktop\main.exe
Введите количество уравнений и переменных через пробел: 3 4
Вводите уравнения в формате 'a1 a2 ... am | b1', где ai - коэффициенты при переменных, bj - свободные члены
3 -6 9 13 | 9
-1 2 1 1 | -11
1 -2 2 3 | 5
Поделит строку 1 на 3
Из строки 2 вычтем строку 1, умноженную на -1
Из строки 3 вычтем строку 1, умноженную на 1
1      -2      3      13/3      3
0      0      4      16/3      -8
0      0      -1     -4/3      2

Поделит строку 2 на 4
Из строки 3 вычтем строку 2, умноженную на -1
1      -2      3      13/3      3
0      0      1      4/3      -2
0      0      0      0      0

Приведенная к ступенчатому виду матрица:
1      -2      3      13/3      3
0      0      1      4/3      -2
0      0      0      0      0

Решений бесконечно много
Свободные переменные - x2, x4
Матрица с единичной подматрицей с неизвестными переменными
1      0      9      2      -1/3
0      1     -2      0      -4/3

x1 = 9 + 2*x2 - 1/3*x4
x3 = -2 - 4/3*x4
Нажмите Enter, чтобы закрыть программу.
```

Рисунок 9 — Пример №8 работы программы при $n < m$ и бесконечном количестве решений

5 Оценка эффективности алгоритма

Проанализируем эффективность каждой функции программы использованием O -символики. Пусть n - количество уравнений, m - количество неизвестных.

Функция для получения строки с ненулевым текущим столбцом (2) выполняется в худшем случае за $O(n)$. Когда $\text{cur_row} = 0$ и все элементы столбца col равны 0.

Функция для приведения матрицы к ступенчатому виду (1) выполняется в худшем случае за $O(m^2 \cdot n)$, поскольку внутри основного цикла последовательно выполняются поиск строки с ненулевым столбцом, перемена мест текущей строки и найденной, деление текущей строки на первый ее ненулевой элемент (все они выполняются линейно). Затем идет вложенный цикл, в котором из строк вычитается текущая. Эта часть выполняется в худшем случае за $O((n-1)m)$, когда $\text{cur_row} = 0$. Итого сложность составляет $O(m \cdot (n + n + n + (n-1) \cdot (m + m))) = O(m \cdot (3n + 2m \cdot (n-1))) = O(3nmt + 2m^2(n-1)) = O(m^2 \cdot n)$.

Функция для проверки наличия решений (3) выполняется за $O(nm)$, поскольку идет проверка по все строкам и всем столбцам.

Функция для получения свободных переменных выполняется за $O(m)$, поскольку цикл проходит по всем номерам столбцов (номер строки же инкрементируется, сложность чего равна $O(1)$, что не влияет на общую сложность выполнения функции).

Функция для нахождения корней системы с единственным решением (5) выполняется за $O(1 + 2 + \dots + n) = O(\frac{1+n}{2} \cdot n) = O(n^2)$, поскольку на каждом шаге количество переменных, которые нужно перенести в правую часть и умножить на коэффициенты, чтобы найти текущую переменную, увеличивается на 1.

Функция для нахождения корней системы со множеством решений выполняется за $O(nm + (n-1) \cdot (1 + 2 + \dots + m)) = O(nm + (n-1) \cdot \frac{1+m}{2} \cdot m) = O(nm + nm^2) = O(m^2 \cdot n)$, поскольку по сути здесь рассматриваются все пары строк (сложность $O(nm)$), и для каждой пары происходит вычитание одной из другой (сложность $O(m)$).

Сложность функций для вывода корней учитываться не будет, так как в сущности частью алгоритма они не являются, однако их сложность не превышает $O(n)$ для системы с единственным решением и $O(nm)$ для системы со множеством решений.

Сложность итоговой функции (9) составляет $O(n \cdot m^2 + nm) = O(n \cdot m^2)$ для системы без решений, $O(n \cdot m^2 + nm + m + n^2) = O(n \cdot m^2 + n^2)$ для системы с единственным решением и $O(n \cdot m^2 + nm + m + n \cdot m^2) = O(n \cdot m^2)$. Для наиболее частого случая применения (когда количество уравнений совпадает с количеством неизвестных) сложность составляет $O(n \cdot n^2) = O(n^3)$.

ЗАКЛЮЧЕНИЕ

В ходе выполнения расчетно-графической работы №3 была написана программа для решения систем линейных алгебраических уравнений методом Гаусса: создано техническое задание для программы, написан и прокомментирован код, приведены примеры выполнения программы при решении систем линейных уравнений и произведена оценка эффективности написанного алгоритма. Полученные навыки написания математических программ будут полезны в дальнейшей профессиональной деятельности. Исходный код программы и исполняемый файл доступны в репозитории разработки [1].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. GitHub. Репозиторий разработки программы для решения СЛАУ методом Гаусса [Электронный ресурс]. - URL: https://github.com/staffeev/Gaussian_elimination (дата обращения: 07.01.2024)