МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г.
ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

Лабораторная работа №1

по дисциплине: Вычислительная математика

тема: «**Тема: Решение систем линейных алгебраических уравнений (СЛАУ)** »

Выполнил: студент группы ПВ-233
Мороз Роман Алексеевич

Проверили:

Белгород 2025 г.

Цель работы: Изучить методы решения СЛАУ и особенности их алгоритмизации в современных программных библиотеках NumPy, SciPy языка Python.

Цель работы обуславливает постановку и решение следующих задач: 1) Рассмотреть теоретические основы и классификацию методов решения СЛАУ.

2) Научиться выбирать методы и алгоритмы решения СЛАУ в зависимости от численной ситуации с вниманием к проблемам разрешимости, точности, численной стабильности и эффективности. 3) Изучить особенности применения прямых методов (на примере LU-разложения) решения СЛАУ с использованием библиотек NumPy, SciPy для языка Python.

4) Рассмотреть особенности программной реализации классического метода Гаусса, улучшенного метода Гаусса с частичным выбором ведущего элемента, решения СЛАУ с помощью LU-разложения матрицы.

5) Познакомиться с web-оболочкой интерактивного блокнота Jupyter как современного инструмента объединяющего код, визуализацию и документацию. Использовать блокнот Jupyter как среду для выполнения всех программ и подготовки отчета по данной лабораторной работе.

6) Выполнить индивидуальное задание, закрепляющее на практике полученные знания и практические навыки (номер задания соответствует номеру студента по журналу; если этот номер больше, чем максимальное число заданий, тогда вариант задания вычисляется по формуле: номер по журналу % максимальный номер задания, где % — остаток от деления).

Первая часть данного задания предполагает решение СЛАУ вручную по классическому методу Гаусса.

Вторая часть задания предполагает написание и выполнение коротких программ на языке Python для решения той же СЛАУ (своего индивидуального задания) с использованием разных алгоритмических техник в интерактивном блокноте Jupyter.

Необходимо сравнить вычислительные схемы и полученные результаты для разных алгоритмов между собой и с результатами собственных вычислений вручную.

Третья часть задания — творческая. Нужно скорректировать параметры своего индивидуального задания (внести минимум изменений) так, чтобы наблюдалась численная неустойчивость решения. Проинтерпретировать эту численную ситуацию.

7) Отразить в отчете все полученные результаты. Сделать выводы. К отчету прикрепить расчёты, выполненные вручную.

Ход выполнения практической части лабораторной работы

На ресурсе https://colab.research.google.com/ создать пустой блокнот. Подготовить в нем текстовое описание хода работы и код для следующих программ с комментариями:
1) Программа, демонстрирующая использование библиотек NumPy и SciPy для решения СЛАУ.

```
'''
Использование библиотеки NumPy
Основным методом, используемым в numpy.linalg.solve
является метод LU-разложения (LU decomposition) и его вариации
'''
import numpy as np

# Коэффициенты системы уравнений
A = np.array([[ 4.2, 1.5, -2.1],
              [ 7.1,-4.8, 3.2],
              [-8.1, 0.3, -3.8]
             ], dtype=float)

b = np.array([2.0, -3.2, 0.1], dtype=float)

# Решение системы
x = np.linalg.solve(A, b)

print("Решение системы с использованием numpy.linalg.solve:", x)
```

```
'''
Использование библиотеки SciPy
SciPy предлагает scipy.linalg.lu_solve
для решения системы с помощью LU-разложения
'''
from scipy.linalg import lu_factor, lu_solve

lu, piv = lu_factor(A)
x_lu = lu_solve((lu, piv), b)

print("Решение системы с использованием scipy.linalg:", x_lu)
```

```
Решение системы с использованием numpy.linalg.solve: [ 0.11766805  0.69240325 -0.22247111]
Решение системы с использованием scipy.linalg: [ 0.11766805  0.69240325 -0.22247111]
```

2) Программы, демонстрирующие алгоритмическую реализацию:
а) рассмотренного на лекции метода Гаусса, основанного на преобразовании исходной системы к верхнетреугольной форме с последующим обратным ходом для нахождения решений;
б) улучшенного метода Гаусса с частичным выбором ведущего элемента (метод улучшает точность вычислений за счет минимизации ошибок округления, выбирая в качестве ведущего элемента максимальный по модулю в текущем столбце);
в) решения СЛАУ с помощью LUразложения матрицы.

```
# Коэффициенты системы уравнений
A = np.array([[4.2, 1.5, -2.1],
              [7.1, -4.8, 3.2],
              [-8.1, 0.3, -3.8]], dtype=float)
# Вектор свободных членов
b = np.array([2.0, -3.2, 0.1], dtype=float)

def gauss(A, b):
    """
    Решение СЛАУ методом Гаусса.
    """
    numEquations = len(b)
    # Прямой ход
    for pivotRow in range(numEquations):
        for currentRow in range(pivotRow + 1, numEquations):
            factor = A[currentRow, pivotRow] / A[pivotRow, pivotRow]
```

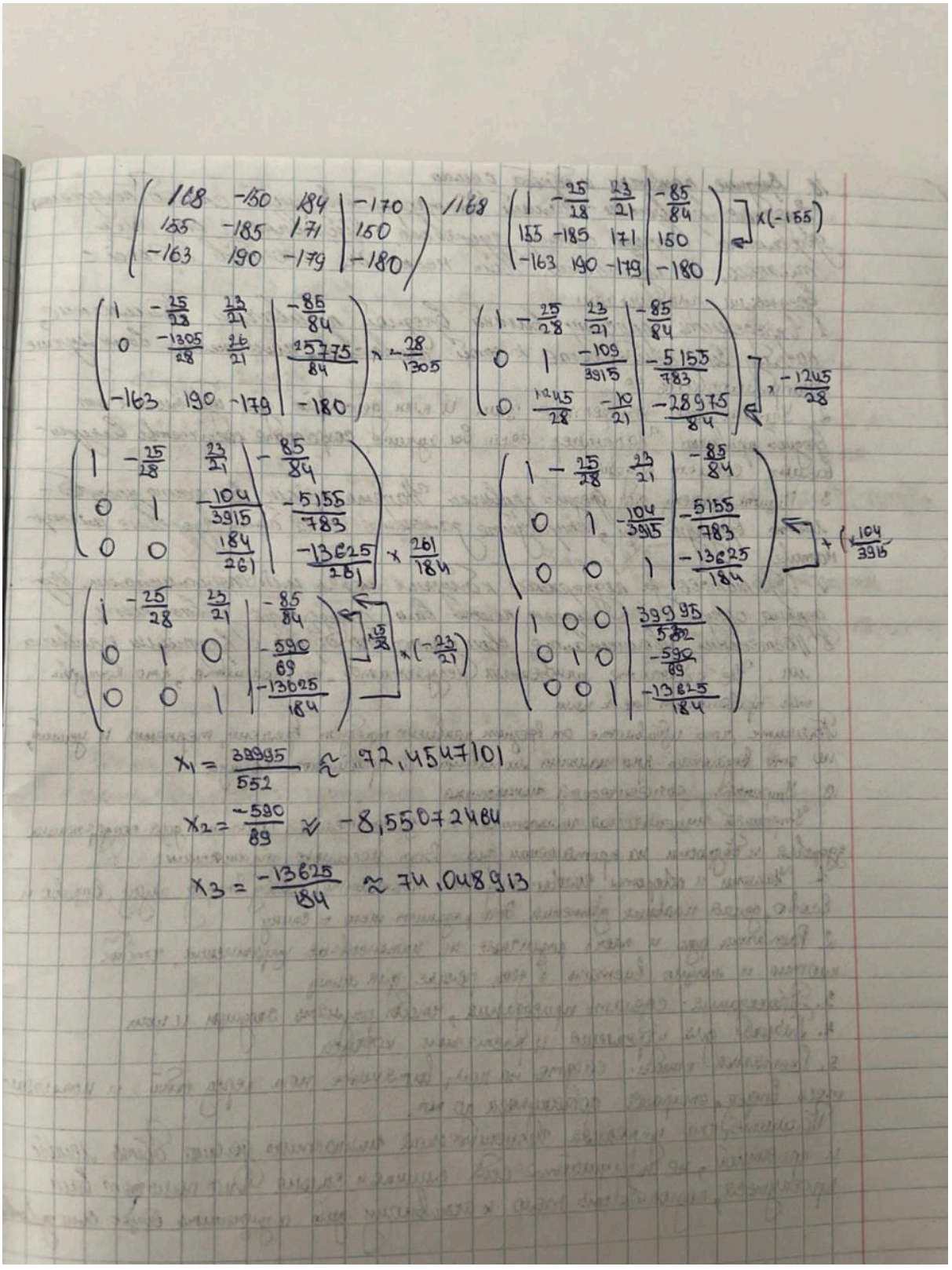
```
            for currentCol in range(pivotRow, numEquations):
                A[currentRow, currentCol] -= factor * A[pivotRow,
currentCol]
            b[currentRow] -= factor * b[pivotRow]

    # Обратный ход
    solutionVector = np.zeros(numEquations)
    for currentRow in range(numEquations - 1, -1, -1):
        sum_ax = 0
        for currentCol in range(currentRow + 1, numEquations):
            sum_ax += A[currentRow, currentCol] *
solutionVector[currentCol]
        solutionVector[currentRow] = (b[currentRow] - sum_ax) /
A[currentRow, currentRow]

    return solutionVector

print(gauss(A.copy(), b.copy()))

def gauss_elimination_with_partial_pivoting(matrix, vector):
    """
    Решает СЛАУ методом Гаусса с частичным выбором ведущего элемента.
    """
    matrix_size = len(matrix)

    # Прямой ход
    for current_column in range(matrix_size):
        max_index = np.argmax(np.abs(matrix[current_column:,
current_column])) + current_column
        matrix[[current_column, max_index]] = matrix[[max_index,
current_column]]
        vector[[current_column, max_index]] = vector[[max_index,
current_column]]

        for i in range(current_column + 1, matrix_size):
            factor = matrix[i][current_column] /
matrix[current_column][current_column]
            matrix[i, current_column:] -= factor *
matrix[current_column, current_column:]
            vector[i] -= factor * vector[current_column]

    # Обратный ход
    solution = np.zeros(matrix_size)
```

```
    for i in range(matrix_size - 1, -1, -1):
        solution[i] = (vector[i] - np.dot(matrix[i, i + 1:], solution[i
+ 1:])) / matrix[i][i]

    return solution

print(gauss_elimination_with_partial_pivoting(A.copy(), b.copy()))

def lu_decomposition(matrix, vector):
    """
    Выполняет LU-разложение матрицы и решает СЛАУ с помощью этого
разложения.
    """
    matrix_size = len(matrix)
    L = np.zeros((matrix_size, matrix_size))
    U = np.zeros((matrix_size, matrix_size))

    # LU разложение
    for row in range(matrix_size):
        L[row, row] = 1
        for col in range(row, matrix_size):
            sum_upper = sum(L[row, sum_index] * U[sum_index, col] for
sum_index in range(row))
            U[row, col] = matrix[row, col] - sum_upper
        for col in range(row + 1, matrix_size):
            sum_lower = sum(L[col, sum_index] * U[sum_index, row] for
sum_index in range(row))
            L[col, row] = (matrix[col, row] - sum_lower) / U[row, row]

    # Решение Ly = b для y
    y = np.zeros(matrix_size)
    for row in range(matrix_size):
        y[row] = vector[row] - np.dot(L[row, :row], y[:row])

    # Решение Ux = y для x
    x = np.zeros(matrix_size)
    for row in range(matrix_size - 1, -1, -1):
        x[row] = (y[row] - np.dot(U[row, row + 1:], x[row + 1:])) /
U[row, row]

    return x

print(lu_decomposition(A.copy(), b.copy()))
```

```
[ 0.11766805  0.69240325 -0.22247111]
[ 0.11766805  0.69240325 -0.22247111]
[ 0.11766805  0.69240325 -0.22247111]
```

Индивидуальные задания

9. $$\begin{cases} 168x_1 - 150x_2 + 184x_3 = -170 \\ 155x_1 - 185x_2 + 171x_3 = 150 \\ -163x_1 + 190x_2 - 179x_3 = -180 \end{cases}$$



$$x_1 = \frac{39995}{552} \approx 72,45471014$$

$$x_2 = \frac{-590}{69} \approx -8,55072464$$

$$x_3 = \frac{-13625}{184} \approx 74,04891304$$

Решение программы:

```
Решение системы с использованием numpy.linalg.solve: [ 72.45471014  -8.55072464 -74.04891304]
Решение системы с использованием scipy.linalg: [ 72.45471014  -8.55072464 -74.04891304]
```

```
[ 72.45471014  -8.55072464 -74.04891304]
[ 72.45471014  -8.55072464 -74.04891304]
[ 72.45471014  -8.55072464 -74.04891304]
```

Численные неустойчивости:

```
A = np.array([[ 168, -150, 11231284],
              [ 0,-185, 171],
              [-163123123, 190, -1]
             ], dtype=float)

b = np.array([-1711110, 777, -0], dtype=float)
```

```
[-5.05516350e-06 -4.34087636e+00 -1.52410101e-01]
[-5.05516376e-06 -4.34087636e+00 -1.52410101e-01]
[-5.05516350e-06 -4.34087636e+00 -1.52410101e-01]
```

```
# Коэффициенты системы уравнений
A = np.array([[ 1118, -13540, 1125284],
              [ 0,-185, 171],
              [-163123123, 7777, -1]
             ], dtype=float)

# Вектор свободных членов
b = np.array([-1711110, 777, -34535], dtype=float)
```

```
[-5.85320113e-05 -5.66857571e+00 -1.58880998e+00]
[-5.85320114e-05 -5.66857571e+00 -1.58880998e+00]
[-5.85320113e-05 -5.66857571e+00 -1.58880998e+00]
```

Вывод: Изучили методы решения СЛАУ и особенности их алгоритмизации в современных программных библиотеках NumPy, SciPy языка Python.