**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)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №1**

по дисциплине: Вычислительная математика

тема: «Решение Систем Линейных Алгебраических Уравнений(СЛАУ)»

Выполнил: ст. группы ВТ-231

Масленников Даниил

Проверили:

Островский Алексей Мичеславович

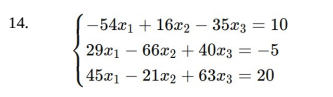
Белгород 2025 г.

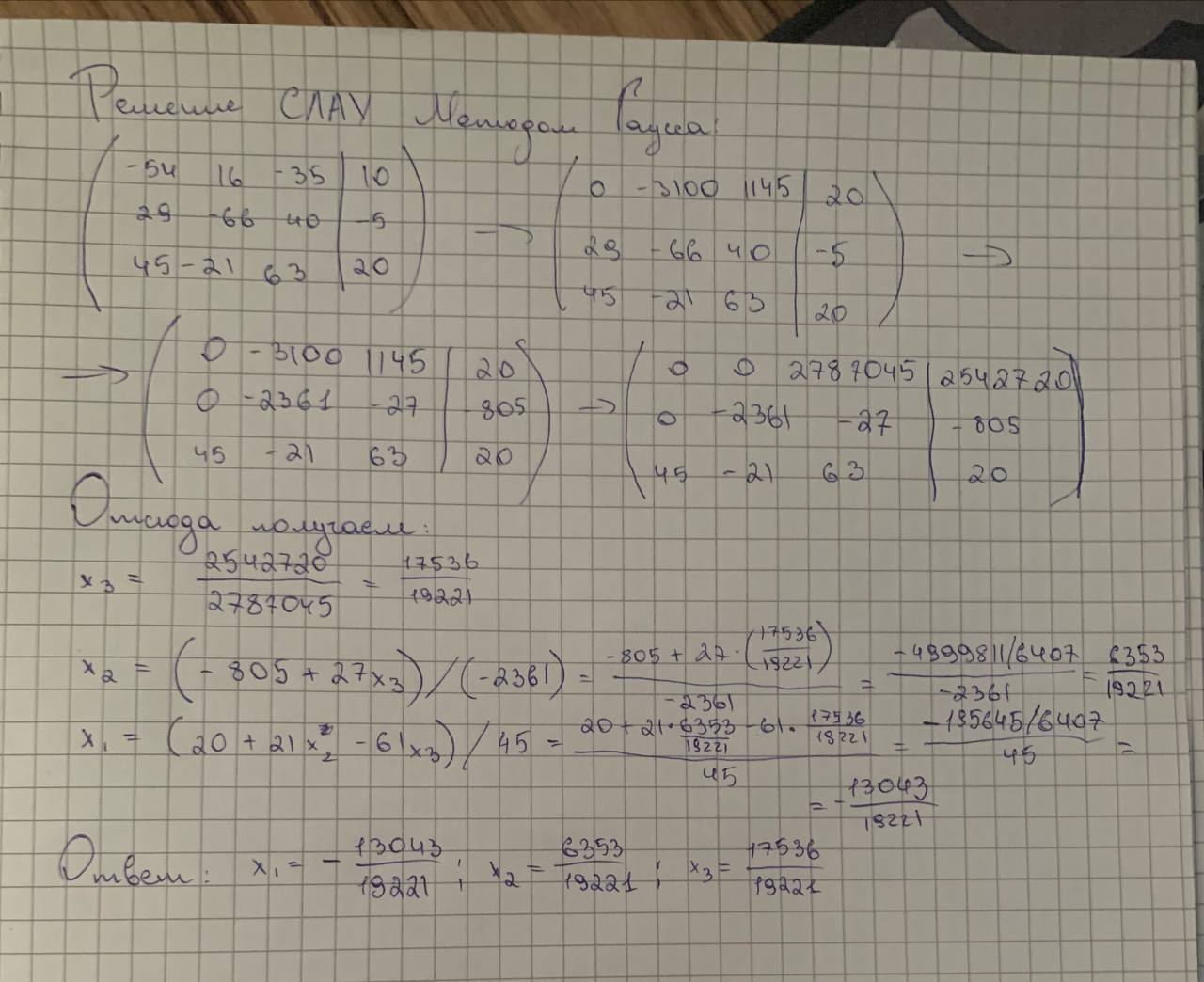
Л а б о р а т о р н а я р а б о т а № 0

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

**Рассмотреть теоретические основы и классификацию методов решения СЛАУ.**  
В рамках данной задачи необходимо изучить основные подходы к решению систем линейных алгебраических уравнений (СЛАУ), включая прямые и итерационные методы. Прямые методы, такие как метод Гаусса, LU-разложение и метод Крамера, позволяют найти точное решение за конечное число шагов. Итерационные методы, такие как метод Якоби и метод Гаусса-Зейделя, используются для приближенного решения больших систем. Также важно рассмотреть классификацию методов по их применимости, точности и вычислительной сложности.

* **Научиться выбирать методы и алгоритмы решения СЛАУ в зависимости от численной ситуации.**  
  При решении СЛАУ важно учитывать такие аспекты, как разрешимость системы, точность вычислений, численная стабильность и эффективность алгоритмов. Например, для плохо обусловленных систем следует использовать методы с частичным выбором ведущего элемента или регуляризацию. Для больших разреженных систем предпочтение отдается итерационным методам. Задача заключается в освоении навыков выбора оптимального метода в зависимости от характеристик системы.
* **Изучить особенности применения прямых методов (на примере LU-разложения) решения СЛАУ с использованием библиотек NumPy, SciPy для языка Python.**  
  LU-разложение — это один из наиболее эффективных прямых методов решения СЛАУ. В данной задаче необходимо изучить, как библиотеки NumPy и SciPy реализуют этот метод, и научиться применять их для решения систем уравнений. Особое внимание уделяется удобству использования этих библиотек и их интеграции в Python-код.
* **Рассмотреть особенности программной реализации классического метода Гаусса, улучшенного метода Гаусса с частичным выбором ведущего элемента, решения СЛАУ с помощью LU-разложения матрицы.**  
  Классический метод Гаусса основан на приведении системы к верхнетреугольному виду с последующим обратным ходом. Улучшенный метод Гаусса с частичным выбором ведущего элемента повышает точность вычислений за счет минимизации ошибок округления. LU-разложение позволяет разложить матрицу на две треугольные матрицы, что упрощает решение системы. В рамках задачи необходимо реализовать эти методы программно и сравнить их эффективность.
* **Познакомиться с web-оболочкой интерактивного блокнота Jupyter как современного инструмента, объединяющего код, визуализацию и документацию.**  
  Jupyter Notebook — это мощный инструмент для разработки и документирования кода. Он позволяет объединять текстовые описания, математические формулы, код и визуализации в одном документе. В данной задаче необходимо освоить работу с Jupyter Notebook, использовать его для выполнения программ и подготовки отчета по лабораторной работе.
* **Выполнить индивидуальное задание, закрепляющее на практике полученные знания и практические навыки.**  
  Индивидуальное задание предполагает решение конкретной СЛАУ вручную и с использованием различных алгоритмических техник в Python. Номер задания соответствует номеру студента по журналу. Если номер превышает количество заданий, вариант вычисляется по формуле: номер по журналу % максимальный номер задания, где % — остаток от деления. Задание включает три части:
* Решение СЛАУ вручную методом Гаусса.
* Написание и выполнение программ на Python для решения той же СЛАУ с использованием разных методов.
* Корректировка параметров системы для демонстрации численной неустойчивости и интерпретация результатов.

**Вариант 14**  
  
  
  


**Задание 1: Вручную решить СЛАУ методом Гаусса**

**Задание 2: Написание и выполнение программ на Python для решения той же СЛАУ с использованием разных методов.**

**from scipy.linalg import lu\_factor, lu\_solve**

**import numpy as np**

**A = np.array([[-54, 16, -35],**

**[29, -66, 40],**

**[45, -21, 63]], dtype=float)**

**b = np.array([10, -5, 20.00000001], dtype=float)**

**x = np.linalg.solve(A, b)**

**print("решение системы с использованием numpy.linalg.solve:", x)**

**lu, piv = lu\_factor(A)**

**x\_lu = lu\_solve((lu, piv), b)**

**print("решение системы с использованием scipy.linalg:", x\_lu)**

**def gauss(A, b):**

**numEquations = len(b)**

**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]**

**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("решение системы методом Гаусса:", gauss(A.copy(), b.copy()))**

**def gauss\_elimination\_with\_partial\_pivoting(matrix, vector):**

**matrix\_size = len(matrix)**

**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]], vector[[current\_column, max\_index]] = \**

**matrix[[max\_index, current\_column]], 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]**

**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("решение системы методом Гаусса с частичным выбором ведущего элемента:",**

**gauss\_elimination\_with\_partial\_pivoting(A.copy(), b.copy()))**

**def lu\_decomposition(matrix, vector):**

**matrix\_size = len(matrix)**

**L = np.zeros((matrix\_size, matrix\_size))**

**U = np.zeros((matrix\_size, matrix\_size))**

**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]**

**y = np.zeros(matrix\_size)**

**for row in range(matrix\_size):**

**y[row] = vector[row] - np.dot(L[row, :row], y[:row])**

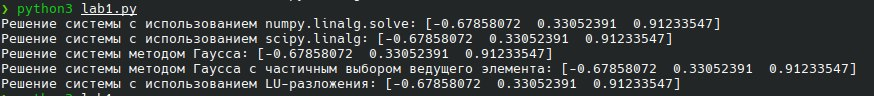
**x = np.zeros(matrix\_size)**

**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("решение системы с использованием LU-разложения:", lu\_decomposition(A.copy(), b.copy()))**

 **Задание 3: Корректировка параметров системы для демонстрации численной неустойчивости и интерпретация результатов.**

**A = np.array([[-54.00000001, 16.00000001, -35.00000001],**

**[29.00000001, -66.00000001, 40.00000001],**

**[45.00000001, -21.00000001, 63.00000001]], dtype=float)**

**b = np.array([10, -5, 20.00000001], dtype=float)**