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

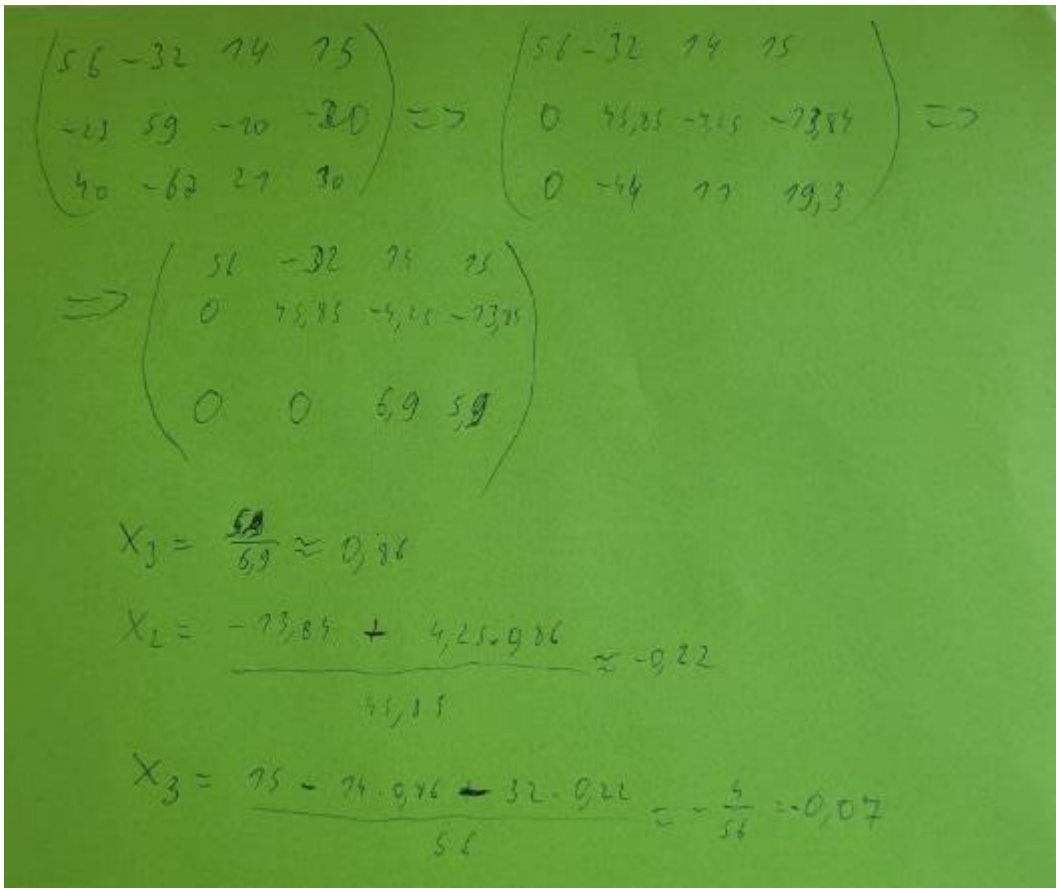
Выполнил: ст. группы ПВ-233
Ситников Алексей Павлович
Проверил:
Горбов Даниил Игоревич

Белгород 2025 г.

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

Вариант 13

13. $$\begin{cases} 56x_1 - 32x_2 + 14x_3 = 15 \\ -23x_1 + 59x_2 - 10x_3 = -20 \\ 40x_1 - 67x_2 + 21x_3 = 30 \end{cases}$$



Программа:

```
import numpy as np
# Коэффициенты системы уравнений
A = np.array([[ 56, -32, 14],
[ -23,59, -10],
[40, -67, 21]], dtype=float)
# Вектор свободных членов
b = np.array([15, -20, 30], dtype=float)
def gauss(A, b):
```

```
    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("Метод Гауса: ", gauss(A.copy(), b.copy()))

def gauss_elimination_with_partial_pivoting(matrix, vector):

    matrix_size = len(matrix)
    # Прямой ход
    for current_column in range(matrix_size):
    # Поиск максимального элемента в текущем столбце
        max_index = np.argmax(np.abs(matrix[current_column:,
current_column])) + current_column
# Обмен строк в матрице и векторе свободных членов
        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]
    # 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("Метод Гауса улучшенный: ", gauss_elimination_with_partial_pivoting(A,
b))

def lu_decomposition(matrix, vector):
    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-разложение: ", lu_decomposition(A, b))
```

Решение программы:

```
Метод Гауса:  [-0.07468155 -0.2217901   0.86320595]
Метод Гауса улучшенный:  [-0.07468155 -0.2217901   0.86320595]
LU-разложение:  [-0.07468155 -0.2217901   0.86320595]
```

Все решения совпали, как и вычисленное вручную.

Теперь попробуем выбрать такие коэффициенты чтобы была разница.

Если взять числа разного порядка, где разница в порядке существенна, то получится что первый метод не справился.

```
Метод Гауса:  [0.         0.         1.07142857]
Метод Гауса улучшенный:  [7.50000000e-001 1.34987893e-101 1.07142857e+000]
LU-разложение:  [7.50000000e-001 1.34987893e-101 1.07142857e+000]
```

Была выбрана такая матрица:

```
A = np.array([[56 * 1e-100, -32, 14],
              [-23, 59 * 1e+100, -10],
              [40, -67, 21 * 1e-100]], dtype=float)
```

Улучшенный метод Гауса справился, так как при выборе ключевого элемента брали максимальный в столбце.

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