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

Изображение выглядит как текст, Шрифт, белый, рукописный текст

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, рукописный текст, бумага, Самоклеющийся листок

Контент, сгенерированный ИИ, может содержать ошибки.

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:  
Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.  
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.  
Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

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.