Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа №1**

**по дисциплине «Вычислительная математика»**

Вариант: **13 - Метод Гаусса с выбором главного элемента по столбцам**

Преподаватель:   
Малышева Татьяна Алексеевна

Выполнил: Телегин Даниил Евгеньевич

Группа: Р3211

Санкт-Петербург, 2025 г

# Цель работы

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них средствами программирования.

Целью данной лабораторной работы является изучение и реализация метода Гаусса с выбором главного элемента по столбцам для решения системы линейных алгебраических уравнений. Также требуется вычислить определитель матрицы, вывести треугольную форму матрицы, найти вектор неизвестных и вектор невязок, а затем сравнить результат с библиотечными функциями.

# Описание метода

**Метод Гаусса с выбором главного элемента по столбцам**

Метод Гаусса предназначен для решения систем линейных уравнений вида:

где:

* — матрица коэффициентов (размерность ),
* — вектор неизвестных (-мерный),
* — вектор свободных членов.

Метод состоит из двух этапов:

1. **Прямой ход** (приведение к треугольной форме)
2. **Обратный ход** (нахождение неизвестных методом подстановки)

**Выбор главного элемента**

На каждом шаге выбирается наибольший по модулю элемент текущего столбца в качестве ведущего. Это уменьшает вычислительные погрешности.

Формула преобразования элементов в прямом ходе:

**Определитель матрицы**

После приведения к треугольному виду определитель вычисляется как произведение диагональных элементов:

**Вектор невязок**

После нахождения решения x, вектор невязок r вычисляется как разница между левыми и правыми частями уравнения:

# Листинг программы

<https://github.com/quwiier/4_computional_mathematics/tree/master/labs/lab1>

## main.py

import numpy as np

def swap\_rows(A, B, row1, row2):

    """Меняет местами две строки в матрице A и векторе B"""

    A[row1], A[row2] = A[row2], A[row1]

    B[row1], B[row2] = B[row2], B[row1]

def gauss\_elimination\_with\_pivoting(A, B):

    """Метод Гаусса с выбором главного элемента по столбцам и подсчётом перестановок"""

    n = len(A)

    swap\_count = 0

    for i in range(n):

        max\_row = i

        for k in range(i + 1, n):

            if abs(A[k][i]) > abs(A[max\_row][i]):

                max\_row = k

        if max\_row != i:

            swap\_rows(A, B, i, max\_row)

            swap\_count += 1

        for j in range(i + 1, n):

            factor = A[j][i] / A[i][i]

            for k in range(i, n):

                A[j][k] -= factor \* A[i][k]

            B[j] -= factor \* B[i]

    return A, B, swap\_count

def compute\_determinant(U, swap\_count):

    """Вычисляет определитель с учетом количества перестановок"""

    det = (-1) \*\* swap\_count

    for i in range(len(U)):

        det \*= U[i][i]

    return det

def compute\_residual(A, x, B):

    """Вычисляет вектор невязок"""

    residuals = [sum(A[i][j] \* x[j] for j in range(len(A))) - B[i] for i in range(len(A))]

    return residuals

def back\_substitution(A, B):

    """Обратный ход метода Гаусса"""

    n = len(A)

    x = [0] \* n

    for i in range(n - 1, -1, -1):

        sum\_ax = sum(A[i][j] \* x[j] for j in range(i + 1, n))

        x[i] = (B[i] - sum\_ax) / A[i][i]

    return x

def main():

    n = int(input("Введите размерность матрицы: "))

    if n > 20:

        print("n > 20, эффективность и точность решения снижена")

    print("Введите коэффициенты матрицы построчно:")

    A = [list(map(float, input().split())) for \_ in range(n)]

    print("Введите свободные члены:")

    B = list(map(float, input().split()))

    determinant\_numpy = np.linalg.det(np.array(A))

    print(determinant\_numpy) # вывод до выполнения алгоритма, чтобы убедиться, что равен 0, тогда будет ошибка "Система несовместная (0 на главной диагонали)"

    x\_numpy = np.linalg.solve(np.array(A), np.array(B))

    A\_copy = [row[:] for row in A]

    B\_copy = B[:]

    U, B\_transformed, swap\_count = gauss\_elimination\_with\_pivoting(A\_copy, B\_copy)

    x\_gauss = back\_substitution(U, B\_transformed)

    determinant = compute\_determinant(U, swap\_count)

    residuals = compute\_residual(A, x\_gauss, B)

    print("\nТреугольная матрица после преобразований:")

    for i in range(n):

        print(" ".join(f"{num:.2f}" for num in U[i]) + f" | {B\_transformed[i]:.2f}")

    print("\nОпределитель матрицы:", determinant)

    print("Определитель матрицы (NumPy):", determinant\_numpy)

    print(swap\_count) # кол-во перестановок для определения знака детерминанта -1^k, k-колво перестановок

    print("\nРешение системы:", " ".join(f"x{i+1} = {xi:.15f}" for i, xi in enumerate(x\_gauss)))

    print("Решение системы (NumPy):", " ".join(f"x{i+1} = {xi:.15f}" for i, xi in enumerate(x\_numpy)))

    print("\nВектор невязок:", " ".join(f"{r:.30f}" for r in residuals))

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except ZeroDivisionError:

        print("Система несовместная (0 на главной диагонали)")

    except Exception as e:

        print(f"Произошла ошибка: {e}")

# Примеры работы программы

# Введите размерность матрицы: 3

# Введите коэффициенты матрицы построчно:

# 2 3 -1

# 4 -1 2

# -3 2 5

# Введите свободные члены:

# 5 3 -2

# Треугольная матрица после преобразований:

# 4.00 -1.00 2.00 | 3.00

# 0.00 3.50 -2.00 | 3.50

# 0.00 0.00 7.21 | -1.00

# Определитель матрицы: -101.0

# Определитель матрицы (NumPy): -101.00000000000003

# Решение системы: x1 = 1.049505 x2 = 0.920792 x3 = -0.138614

# Решение системы (NumPy): x1 = 1.049505 x2 = 0.920792 x3 = -0.138614

# Вектор невязок: 0.000000 0.000000 0.000000

# Вывод:

В результате выполнения данной лабораторной работой я познакомился с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав на языке программирования Python метод Гаусса с выбором главного элемента по столбцам, а также сравнив результат с решением через стороннюю библиотеку NumPy.