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

Факультет программной инженерии и компьютерных технологий

Вычислительная математика

Лабораторная работа 1

«Решение системы линейных алгебраических уравнений СЛАУ»

Вариант 13

Выполнила:

Павличенко Софья Алексеевна, P3215

Проверила:
Малышева Татьяна Алексеевна

Санкт-Петербург 2025г.

Оглавление

Цель

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

Описание метода

Метод Гаусса с выбором главного элемента по столбцу — это модификация классического метода Гаусса, где на каждом шаге для повышения точности и стабильности выбирается наибольший по модулю элемент в текущем столбце (ведущий элемент), и соответствующая строка меняется местами с текущей.

Этапы метода:

1. Выбор главного элемента:
    - В текущем столбце (начиная с главной диагонали) находим элемент с наибольшим модулем.
    - Меняем строки местами, чтобы этот элемент стал ведущим.
2. Прямой ход (приведение к верхнетреугольному виду):
    - Для всех строк ниже ведущей зануляем элементы под диагональю, используя элементарные преобразования.
3. Обратный ход (нахождение решения):
    - Подставляем найденные значения переменных, двигаясь снизу вверх, решая треугольную систему.

Расчётные формулы

$det(A) = (-1)^k * \prod_{i=1}^{n} a_{ii}$ – определитель после приведения матрицы к треугольному виду, где k – число перестановок строк (или столбцов) матрицы при ее приведении к треугольному виду.

$r_i = \sum_{j=0}^{n-1} a_{ij} * x_j - b_i$ – вектор невязки.

Листинг программы

```
import copy
import numpy as np


def to_upper_triangular(matrix, n):
    """Прямой ход метода Гаусса — приведение к верхнетреугольному виду"""
    triangular_matrix = copy.deepcopy(matrix)  # Создаём копию, чтобы не изменять
исходную матрицу
    det = 1  # Определитель
    swap_count = 0  # Количество перестановок строк

    for i in range(n):
        # Выбор главного элемента по столбцу (поиск максимума по модулю)
        pivot_row = i
        for row in range(i + 1, n):
            if abs(triangular_matrix[row][i]) >
abs(triangular_matrix[pivot_row][i]):
                pivot_row = row

        # Меняем строки, если найден другой главный элемент
        if pivot_row != i:
            triangular_matrix[i], triangular_matrix[pivot_row] =
triangular_matrix[pivot_row], triangular_matrix[i]
            swap_count += 1

        # Если найден нулевой элемент на диагонали — система вырожденная
        if triangular_matrix[i][i] == 0:
            return triangular_matrix, 0, swap_count

        det *= triangular_matrix[i][i]

        # Зануление элементов под главной диагональю
        for elimination_row in range(i + 1, n):
            k = triangular_matrix[elimination_row][i] / triangular_matrix[i][i]
            for col in range(i, n + 1):
                triangular_matrix[elimination_row][col] -= k *
triangular_matrix[i][col]

    # Корректируем знак определителя в зависимости от количества перестановок
    return triangular_matrix, det * (-1) ** swap_count, swap_count


def solve_slae(triangular_matrix, n):
    """Обратный ход метода Гаусса - находим переменные"""
    solution = [0] * n
    for i in range(n - 1, -1, -1):
        # Находим значение x_i
        solution[i] = triangular_matrix[i][n] / triangular_matrix[i][i]

        # Вычитаем найденное значение из предыдущих уравнений
        for j in range(i - 1, -1, -1):
            triangular_matrix[j][n] -= triangular_matrix[j][i] * solution[i]

    return solution


def compute_residual(matrix, n, solution):
```

```
    """Вычисляет вектор невязки: разницу между Ax и b."""
    residual = [0] * n
    for i in range(n):
        Ax_i = sum(matrix[i][j] * solution[j] for j in range(n))
        residual[i] = Ax_i - matrix[i][n]

    return residual


# --- Ввод данных ---

. . .

# --- Решение СЛАУ ---

# print_matrix(matrix, "Исходная матрица:")

triangular_matrix, determinant, swap_count = to_upper_triangular(matrix, n)
print_matrix(triangular_matrix, "Треугольная матрица:")

print(f"Количество перестановок: {swap_count}")
print(f"Определитель матрицы: {determinant:.3f}")

if determinant == 0:
    print("Система несовместна или имеет бесконечно много решений.")
else:
    solution = solve_slae(triangular_matrix, n)
    print("Решение системы:" + ", ".join(f"x{i+1} = {x:.3f}" for i, x in
enumerate(solution)))

    residual = compute_residual(matrix, n, solution)
    print("Вектор невязки:" + ", ".join(f"r{i+1} = {r:.30e}" for i, r in
enumerate(residual)))

print("\n\nПроверим результаты, используя библиотеки:\n")

A = np.array([row[:-1] for row in matrix], dtype=float)
b = np.array([row[-1] for row in matrix], dtype=float)

print(f"Определитель матрицы (NumPy): {np.linalg.det(A):.3f}")

# Проверка рангов
rank_A = np.linalg.matrix_rank(A)                  # Ранг коэффициентной матрицы
rank_Ab = np.linalg.matrix_rank(np.column_stack((A, b)))  # Ранг расширенной
матрицы

if rank_A == rank_Ab and rank_A == n:
    print("Решение системы (NumPy):", ", ".join(f"x{i+1} = {xi:.3f}" for i, xi in
enumerate(np.linalg.solve(A, b))))
else:
    print("Система несовместна или имеет бесконечно много решений.")
```

Результат работы программы

```
Как вы хотите задать коэффициенты матрицы?
1 - с консоли
2 - с файла
1

Введите размерность матрицы:
4
Введите коэффициенты матрицы (каждая строка через пробел):
3 4 -9 5 -14
-15 -12 50 -16 44
-27 -36 73 8 142
9 12 -10 -16 -76

Исходная матрица:
3.000  4.000  -9.000  5.000  -14.000
-15.000  -12.000  50.000  -16.000  44.000
-27.000  -36.000  73.000  8.000  142.000
9.000  12.000  -10.000  -16.000  -76.000

Треугольная матрица:
-27.000  -36.000  73.000  8.000  142.000
0.000  8.000  9.444  -20.444  -34.889
0.000  0.000  14.333  -13.333  -28.667
0.000  0.000  0.000  5.062  -0.000

Определитель матрицы: -15672.000
Решение системы:x1 = -8.000, x2 = -2.000, x3 = -2.000, x4 = -0.000
Вектор невязки:r1 = -1.776e-15, r2 = -7.105e-15, r3 = 0.000e+00, r4 = 0.000e+00

 Проверим результаты, используя библиотеки:

 Определитель матрицы (NumPy): -15672.000
 Решение системы (NumPy): x1 = -8.000, x2 = -2.000, x3 = -2.000, x4 = -0.000
```

Выводы

В результате выполнения данной лабораторной работой я познакомилась с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав на языке программирования Python метод Гаусса с выбором главного элемента по столбцам.

Результаты, полученные программой, совпали с библиотечными (NumPy) с высокой точностью, что говорит о корректности реализованного алгоритма.