Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1 по Вычислительной Математике

Решение системы линейных алгебраических
уравнений СЛАУ

Вариант №5

Группа: P3214
Выполнил:
Минкова Алина Андреевна
Проверил:
Малышева Татьяна Алексеевна

Г. Санкт-Петербург
2024

Оглавление

Цель работы

Используя известные методы вычислительной математики, написать программу, осуществляющий решение СЛАУ прямым методом.

Описание метода, расчётные формулы

Метод Гаусса основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы.

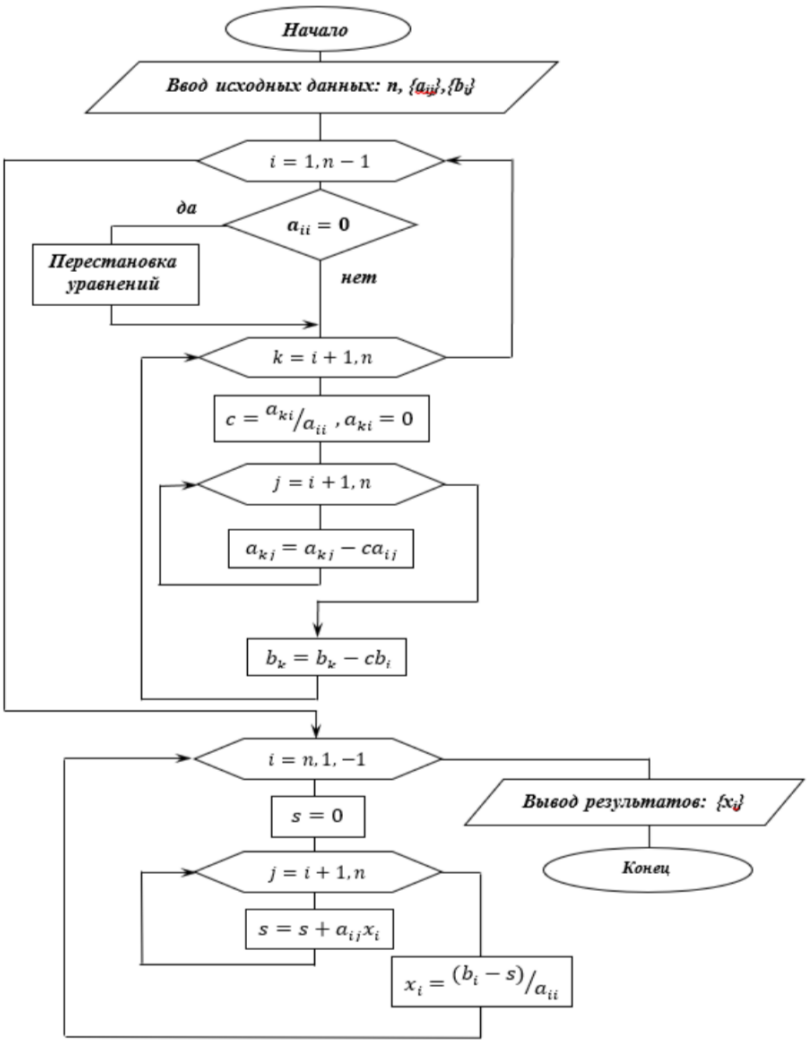
Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается $x1$ из всех последующих уравнений системы. Затем с помощью второго уравнения исключается $x2$ из третьего и всех последующих уравнений и т.д. Этот процесс продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным xn , т. е. матрица системы будет приведена к треугольному виду.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим неизвестное xn . Далее, из предыдущего уравнения вычисляем $xn-1$ и т. д. Последним найдем $x1$ из первого уравнения.

Метод имеет много различных вычислительных схем.

Основное требование - $\det A \neq 0$.

Блок-схема метода Гаусса



Первый цикл с переменной цикла i реализует прямой ход, а второй – обратный ход метода.

i – номер неизвестного, которое исключается из оставшихся n – 1 уравнений при прямом ходе (а также номер уравнения, из которого исключается xi) и номер неизвестного, которое определяется из i -го уравнения при обратном ходе;

k – номер уравнения, из которого исключается неизвестное xi при прямом ходе;

j – номер столбца при прямом ходе и номер уже найденного неизвестного при обратном ходе.

Листинг программы

```
import numpy as np

def read_data():
    size = 0
    dat = []
    matrix = []
    b = []
    inp_type = input("Выберете способ ввода данных (file/console): ")
    if inp_type == "file":
        filename = input("Введите имя файла: ")
        with open(filename, "r") as f:
            size = int(f.readline())
            for i in range(size):
                matrix.append(list(map(float, f.readline().split(" "))))
    elif inp_type == "console":
        size = int(input("Введите размерность: "))
        print("Введите матрицу:")
        for i in range(size):
            matrix.append(list(map(float, input().split())))
    else:
        print("Некорректный ввод")
        read_data()

    for i in range(size):
        b.append(matrix[i].pop())

    if size > 20 or size <= 1:
        print("Задан некорректный размер")
        read_data()
    dat.append(size)
    dat.append(matrix)
    dat.append(b)
    return dat


class Solver:
    def __init__(self, size, matrix, b):
        self.size = size
        self.matrix = matrix
        self.b = b
        self.x = [0] * size
        self.row_swaps = 0

    def print_matrix(self):
        for i, row in enumerate(self.matrix):
            for element in row:
                print("{:8}".format(element), end='')
            print(" | ", end='')
            print("{:8}".format(self.b[i]))
        print()

    def det_with_libr(self, matrix):
        return np.linalg.det(matrix)

    def det(self, matrix, row_swaps):
        det = -1 ** row_swaps
        for i in range(len(matrix)):
            det *= matrix[i][i]
        # if row_swaps % 2 != 0:
        #     det *= -1
        return det
```

```
    def upper_triangular(self):
        for i in range(self.size):
            if self.matrix[i][i] == 0:
                non_zero_row = i + 1
                while non_zero_row < self.size and
self.matrix[non_zero_row][i] == 0:
                    non_zero_row += 1

                if non_zero_row == self.size:
                    print("Система несовместна.")
                    exit()

                self.matrix[i], self.matrix[non_zero_row] =
self.matrix[non_zero_row], self.matrix[i]
                self.b[i], self.b[non_zero_row] = self.b[non_zero_row],
self.b[i]
                self.row_swaps += 1
                # print(f"Матрица после перестановки {self.row_swaps}:")
                # self.print_matrix()

            for j in range(i + 1, self.size):
                coef = self.matrix[j][i] / self.matrix[i][i]
                for k in range(self.size):
                    self.matrix[j][k] -= coef * self.matrix[i][k]
                self.b[j] -= coef * self.b[i]
            # print(f"Матрица после итерации {i + 1}:")
            # self.print_matrix()

    def back_substitution(self):
        for i in range(self.size - 1, -1, -1):
            self.x[i] = self.b[i]
            for j in range(i + 1, self.size):
                self.x[i] -= self.matrix[i][j] * self.x[j]
            self.x[i] /= self.matrix[i][i]

    def compute_residuals(self):
        residuals = [0] * len(self.b)
        for i in range(len(self.b)):
            sum_val = 0
            for j in range(len(self.x)):
                sum_val += self.matrix[i][j] * self.x[j]
            residuals[i] = sum_val - self.b[i]
        return residuals

if __name__ == '__main__':
    data = read_data()
    size, matrix, b = data[0], data[1], data[2]
    solver = Solver(size, matrix, b)

    print("Получена матрица:")
    solver.print_matrix()

    det_with_libr = solver.det_with_libr(matrix)
    print("Определитль матрицы, посчитанный библиотекой:")
    print(det_with_libr)
    print()

    solver.upper_triangular()
    print("Верхнетреугольная матрица:")
    solver.print_matrix()
    print("Количество перестановок строк матрицы:", solver.row_swaps)
    print()
```

```
    det = solver.det(matrix, solver.row_swaps)
    print("Определитель матрицы:")
    print(det)
    print()

    solver.back_substitution()
    print("Вектор неизвестных:")
    for i in range(size):
        print(f"x{i + 1} = {solver.x[i]}")
    print()

    residuals = solver.compute_residuals()
    print("Вектор невязок:")
    for i in range(size):
        print('{0:.17e}'.format(residuals[i]))
    print()
```

Примеры и результаты работы программы

Работа из файла

```
Введите имя файла: inp.txt
Получена матрица:
    34.0    54.0 |    789.0
    32.0    12.0 |     56.0

Определитль матрицы, посчитанный библиотекой:
-1320.0

Верхнетреугольная матрица:
    34.0    54.0 |    789.0
     0.0-38.8235294117647 | -686.5882352941177

Количество перестановок строк матрицы: 0

Определитель матрицы:
1320.0

Вектор неизвестных:
x1 = -4.881818181818185
x2 = 17.684848484848487

Вектор невязок:
0.00000000000000000e+00
0.00000000000000000e+00
```

Работа из консоли

```
Выберете способ ввода данных (file/console): console
Введите размерность: 3
Введите матрицу:
34 32 56 78
65 45 78 65
34 2 34 56
Получена матрица:
    34.0    32.0    56.0 |     78.0
    65.0    45.0    78.0 |     65.0
    34.0     2.0    34.0 |     56.0

Определитль матрицы, посчитанный библиотекой:
-17540.000000000004

Верхнетреугольная матрица:
    34.0    32.0    56.0 |     78.0
     0.0-16.176470588235297-29.058823529411768 | -84.11764705882354
     0.0     0.031.89090909090909 |    134.0

Количество перестановок строк матрицы: 0

Определитель матрицы:
17540.000000000004
```

```
Вектор неизвестных:
x1 = -2.4166476624857465
x2 = -2.348004561003422
x3 = 4.2018244013683015

Вектор невязок:
0.00000000000000000e+00
0.00000000000000000e+00
0.00000000000000000e+00
```

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.