МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №22

Студенты:
Самарина Арина Анатольевна
Суржицкий Арсений Арсентьевич
Группа P3266

Преподаватель:
Машина Екатерина Александровна

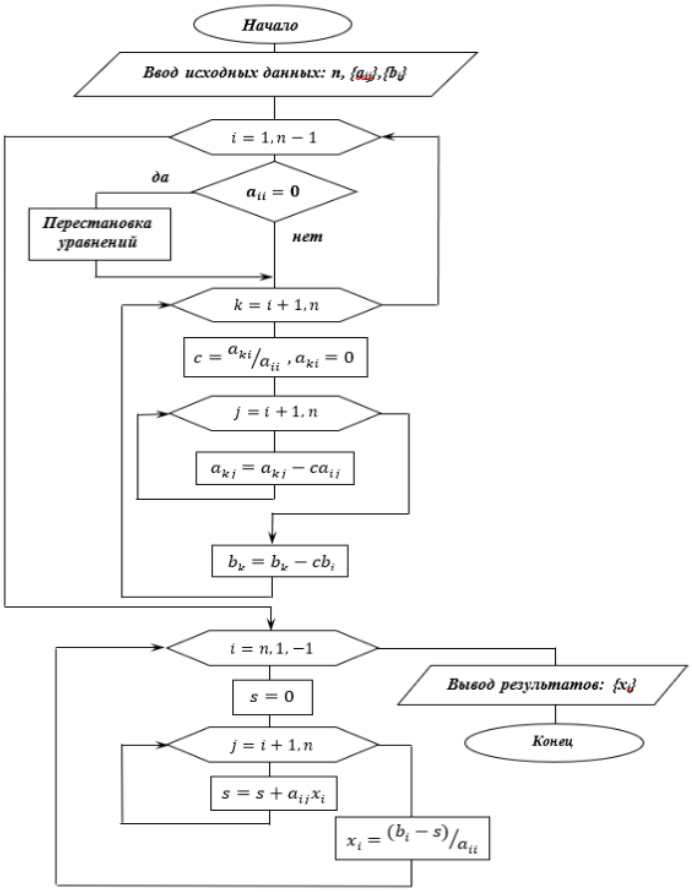
Санкт-Петербург, 2024

1. Цель работы

Цель данной лабораторной работы заключается в изучении и реализации численных методов решения систем линейных алгебраических уравнений (СЛАУ) на примере метода Гаусса.

2. Описание метода, расчётные формулы

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.

3. Листинг программы

Main.py

```
from matrix_input import *
from gauss_method import *
from gauss_method import calculated_residuals
```

```
while True:
    try:
        print("Выберите команду:")
        input_variant = int(
            input(
                "1) Ручной ввод матрицы\n2) Входная матрица из
файла\n3) Создать случайную матрицу\n4) Выйти из программы\n"
            )
        )
        if input_variant in range(1, 5):
            break
        print("Неверная команда")
        continue
    except ValueError:
        print("Введено неверное значение")
        continue

switch_command = {
    1: hand_matrix_input,
    2: open_file_matrix,
    3: random_matrix,
    4: exit,
}

matrix = switch_command.get(input_variant, exit)()
print("Матричное чтение:")
print_matrix(matrix)
first_matrix = matrix[:]

matrix, swap_counts = method_Gauss(matrix)

print("Поменять местами строки = ", swap_counts, "\n")

print("Матрица после прохода вперед:")
print_matrix_float(matrix)

if not is_triangular(matrix):
    print("Не удалось преобразовать матрицу в треугольную
форму.")
    exit()

det = triangular_matrix_determinant(matrix) * ((-1) **
swap_counts)
if det == 0:
    print("Определитель матрицы = 0, невозможно решить обратной
заменой")
    exit()
else:
    print("Определитель матрицы = ", det, "\n")

print_answers(reversal_method(matrix))
```

```
print_residuals(calculated_residuals(first_matrix,
reversal_method(matrix)))
```

matrix_input.py

```
import os
import random


# Позволяет пользователю вводить матрицу вручную #

def hand_matrix_input():
    while True:
        try:
            n = int(input("Введите размерность матрицы, не
превышающую 20\n"))
        except ValueError:
            print("Введено неверное значение")
            continue
        if 0 <= n <= 20:
            break
        else:
            print("Введено неверное значение (Матрица должна
быть от 0 до 20)")

    print("Введенный номер:", n)
    matrix = []
    print("Введите матрицу. Вводите построчно, разделяя элементы
пробелами.")
    for i in range(n):
        while True:
            row = []
            try:
                matrix_row = (
                    input(
                        "Введите {n+1} элементов для {i + 1}-й
строки, разделенных пробелами: "
                    )
                    .replace(",", ".")
                    .split()
                )
                row = list(map(int, matrix_row))
            except ValueError:
                for item in matrix_row:
                    try:
                        number_float = float(item)
                        if number_float.is_integer():
                            row.append(int(number_float))
                        else:
```

```
                            row.append(number_float)
                    except ValueError:
                        if isinstance(item, float):
                            row.append(float(item))
                        else:
                            print("Введен неверный номер")
                            continue

            except UnboundLocalError:
                print("Введен неверный номер")
                continue

            if len(row) != n + 1:
                print("Строка введена неверно")
                continue
            matrix.append(row)
            break

    return matrix


# Проверяет, является ли матрица квадратной #
def is_square_matrix(matrix):
    n = len(matrix)
    for row in matrix:
        if len(row) != n + 1:
            return False
    return True

# Вывод матрицы в консоль #
def print_matrix(matrix):
    for row in matrix:
        for i, elem in enumerate(row):
            if i == len(row) - 1:
                print("|", end=" ")
            print(elem, end=" ")
        print()
    print()


def print_matrix_float(matrix):
    for row in matrix:
        for i, elem in enumerate(row):
            if i == len(row) - 1:
                print("|", end=" ")
            print(float(elem), end=" ")
        print()
    print()

# Загружает матрицу из файла #
def open_file_matrix():
    current_working_directory = os.path.dirname(__file__)
    matrix = []
```

```
    while True:
        try:
            file_name = input("Введите относительный путь к
вашему файлу\n")
            print()
            file_path = os.path.join(current_working_directory,
file_name)
            with open(file_path, "r", encoding="utf-8") as file:
                for line in file:
                    row = []
                    try:
                        row = list(map(int,
line.strip().replace(",", ".").split()))
                    except ValueError:
                        for item in line.strip().replace(",",
".").split():
                            try:
                                number_float = float(item)
                                if number_float.is_integer():

row.append(int(number_float))
                                else:
                                    row.append(number_float)
                            except ValueError:
                                row.append(float(item))
                    matrix.append(row)
                break
        except FileNotFoundError:
            print("Введен неправильный путь к файлу")
            continue
        except ValueError:
            print("Неправильные коэффициенты в указанном файле")
            exit()
    if not is_square_matrix(matrix):
        print("Матрица в файле не квадратная")
        exit()
    return matrix


# Выводит найденные решения системы и соответствующие остатки #
def print_answers(row):
    answer = ""
    for elem in range(len(row)):
        answer += "x" + str(elem + 1) + " = " + str(row[elem]) +
"   "
    print(answer)
def print_residuals(residuales):
    answer = ""
    for elem in range(len(residuales)):
        answer += "r" + str(elem + 1) + " = " +
str(residuales[elem]) + "   "
    print(answer)
```

```
# Создает случайную квадратную матрицу #
def random_matrix():
    while True:
        try:
            n = int(input("Введите размерность матрицы, не
превышающую 20\n"))
        except ValueError:
            print("Введено неверное значение")
            continue
        if 0 <= n <= 20:
            break
        else:
            print("Введено неверное значение (Матрица должна
быть от 0 до 20)")
    matrix = []
    for i in range(n):
        row = []
        for i in range(n + 1):
            if random.random() < 0.9:
                row.append(random.randint(10, 100))
            else:
                row.append(random.randint(10, 100))

        matrix.append(row)
    return matrix
```

gauss_method.py

```
from fractions import Fraction

# Умножение строки на число #
def string_multiplication(matrix, row_index, k, j):
    if row_index < 0 or row_index >= len(matrix):
        raise IndexError("Неверный индекс строки")
    new_row = matrix[row_index][:]

    if (not isinstance(k, int)) and (k == 0):
        try:
            k = int(k)
        except ValueError:
            k = Fraction(k)
    if (not isinstance(j, int)) and (j == 0):
        try:
            j = int(j)
        except ValueError:
            j = Fraction(j)
    k = Fraction(k)
    j = Fraction(j)
    for i in range(len(new_row)):
```

```
        new_row[i] *= Fraction(-k, j)

    return new_row

# Перестановка строк матрицы #
def swap_rows(matrix, i, j):
    if i < 0 or i >= len(matrix) or j < 0 or j >= len(matrix):
        raise IndexError("Неверный индекс строки")
    matrix[i], matrix[j] = matrix[j], matrix[i]

#  Прибавляет к одной строке другую, умноженную на число #
def string_addition(matrix, row_index, adding_row):
    if row_index < 0 or row_index >= len(matrix):
        raise IndexError("Неверный индекс строки")
    for i in range(len(matrix[row_index])):
        matrix[row_index][i] += adding_row[i]
    return matrix

# Производит преобразование матрицы к треугольному виду
(верхнетреугольной).
# Если на текущем шаге на главной диагонали встречается ноль,
строки матрицы переставляются так,
# чтобы ненулевой элемент был на диагонали. Это позволяет
избежать деления на ноль.
# Возвращает преобразованную матрицу и количество перестановок
строк.
def method_Gauss(matrix):
    swap_counts = 0
    for i in range(len(matrix)):
        if matrix[i][i] != 0:
            for j in range(i + 1, len(matrix)):
                buffer_row = string_multiplication(
                    matrix, i, matrix[j][i], matrix[i][i]
                )
                matrix = string_addition(matrix, j, buffer_row)
        else:
            for p in range(i + 1, len(matrix)):
                if matrix[p][i] != 0:
                    swap_counts += 1
                    swap_rows(matrix, i, p)
                    break
                else:
                    raise ValueError("No nonzero diagonal
element")
            for j in range(i + 1, len(matrix)):
                buffer_row = string_multiplication(
                    matrix, i, matrix[j][i], matrix[i][i]
                )
                matrix = string_addition(matrix, j, buffer_row)

    return matrix, swap_counts

# Проверяет, является ли матрица треугольной #
```

```
def is_triangular(matrix):
    n = len(matrix)
    for i in range(n):
        for j in range(i):
            if matrix[i][j] != 0:
                return False
    return True

# Находит решение системы линейных уравнений путем обратной
подстановки #
def reversal_method(matrix):
    n = len(matrix)
    x = [0] * n

    for i in range(n - 1, -1, -1):
        sum = 0
        for j in range(i + 1, n):
            sum += matrix[i][j] * x[j]
        x[i] = (matrix[i][-1] - sum) / matrix[i][i]
    x = list(map(float, x))
    return x

# Вычисляет определитель для треугольной матрицы #
def is_triangular(matrix):
    n = len(matrix)
    for i in range(n):
        for j in range(i):
            if matrix[i][j] != 0:
                return False
    return True

# Вычисляет остатки для найденных решений #
def calculated_residuals(matrix, answers):
    values = []
    for row in matrix:
        sum = 0
        for j in range(len(row) - 1):
            sum += row[j] * answers[j]
        values.append(row[-1] - sum)
    return values

# Определитель треугольной матрицы #
def triangular_matrix_determinant(matrix):
    det = 1
    for i in range(len(matrix)):
        det *= matrix[i][i]
    return det
```

4. Примеры и результаты работы программы

5. Вывод

В ходе работы был успешно реализован метод Гаусса для решения систем линейных уравнений. Программа корректно выполняет все этапы метода, включая вычисление определителя и нахождение вектора неизвестных. Тестирование подтвердило правильность реализации. Работа показала эффективность метода Гаусса и создала основу для дальнейшего изучения численных методов решения СЛАУ.