ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет программной инженерии и компьютерной техники

Дисциплина:
«Вычислительная математика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Выполнил:
Студент гр. P32151
Понамарев Степан Андреевич

Проверил:
Машина Екатерина Алексеевна

Санкт-Петербург
2023г.

1. Задание лабораторной работы:

Лабораторная работа 1. «Решение системы линейных алгебраических уравнений СЛАУ»

1. № варианта определяется как номер в списке группы согласно ИСУ.
2. В программе численный метод должен быть реализован в виде отдельной подпрограммы/метода/класса, в который исходные/выходные данные передаются в качестве параметров.
3. Размерность матрицы n<=20 (задается из файла или с клавиатуры - по выбору конечного пользователя).
4. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

Для прямых методов должно быть реализовано:

- Вычисление определителя
- Вывод треугольной матрицы (включая преобразованный столбец B)
- Вывод вектора неизвестных: $x_1,\ x_2,\ \dots, x_n$
- Вывод вектора невязок: $r_1,\ r,\ \dots, r_n$

Для итерационных методов должно быть реализовано:

- Точность задается с клавиатуры/файла
- Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
- Вывод вектора неизвестных: $x_1,\ x_2,\ \dots, x_n$
- Вывод количества итераций, за которое было найдено решение.
- Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$

Содержание отчета:

- Цель работы,
- Описание метода, расчетные формулы,
- Листинг программы (по крайне мере, где реализован сам метод)
- Примеры и результаты работы программы,
- Выводы.
- Отчет предоставляется в электронном/бумажном виде.

2. Листинг программы:

Код программы на языке Python:

Main.py:

```
from MartixManager import MatrixContainer
from InputManager import InputManager
import numpy as np
import os, sys

if __name__ == "__main__":
    print("Добро пожаловать в решатель СЛАУ методом Гаусса-Зейделя!\n")

    matrix_container = MatrixContainer()

    if InputManager.yes_or_no_input("Хотите считать матрицу из csv-файла?
[yes/no]: "):
```

```
        filename = InputManager.string_input('Введите название файла без
".csv": ')
        try:
            m = np.genfromtxt(filename + '.csv', delimiter=',')
        except ValueError:
            print("Матрица некорректна.")
            os.system("pause")
            sys.exit()
        except OSError:
            print("Файл не найден.")
            os.system("pause")
            sys.exit()

    else:
        m = InputManager.matrix_input()

    matrix_container.set_matrix(m)

    print("Хотите ввести точность? По умолчанию epsilon равно 10^-8.")
    if InputManager.yes_or_no_input("Ввести epsilon? [yes/no]: "):
        matrix_container.set_epsilon(InputManager.float_input("epsilon="))

    if matrix_container.find_determinant() == 0:
        print("Нахождение корней невозможно: определитель матрицы равен
нулю")
    else:
        matrix_container.solve()

    os.system("pause")
```

InputManager.py:

```
import numpy as np


class InputManager:
    @staticmethod
    def string_input(message=""):
        buf = ""
        while buf == "":
            buf = input(message).strip()
        return buf

    @staticmethod
    def _check_number(buf):
        try:
            float(buf.replace(',', '.'))
            return True
        except ValueError:
            return False

    @staticmethod
    def _convert_to_number(num):
        if InputManager._check_number(num):
            return float(num.replace(',', '.'))
        return None

    @staticmethod
    def float_input(message=""):
        while True:
            buf = InputManager.string_input(message)
            if InputManager._check_number(buf):
                return InputManager._convert_to_number(buf)
```

```
    @staticmethod
    def int_input(message=""):
        return int(InputManager.float_input(message))

    @staticmethod
    def yes_or_no_input(message=""):
        answer = "0"
        while answer[0].lower() not in ["y", "n"]:
            answer = InputManager.string_input(message)
        return answer[0].lower() == "y"

    @staticmethod
    def matrix_input():
        lines_number = InputManager.int_input("Введите количество строк
матрицы: ")
        print("Введите матрицу вида\n"
              "a_11, a_12, ..., a_1n, b_1\n"
              "...\n"
              "a_n1, a_n2, ..., a_nn, b_n\n")
        matrix = [[0] * (lines_number + 1) for _ in range(lines_number)]
        for i in range(lines_number):
            line = InputManager.string_input().split()
            while len(line) != lines_number + 1 or not
all([InputManager._check_number(t) for t in line]):
                print("Некорректный ввод строки")
                line = InputManager.string_input().split()
            for j in range(lines_number + 1):
                matrix[i][j] = InputManager._convert_to_number(line[j])
        print()
        return np.array(matrix)
```

MatrixManager.py:

```
import numpy as np
from MatrixError import MatrixError


class MatrixContainer:
    def __init__(self):
        self.matrix = None
        self.constants = None
        self.lines_number = None
        self.determinant = None
        self.epsilon = None
        self._drop_context()

    def _drop_context(self):
        self.lines_number = 0
        self.constants = []  # b vector
        self.matrix = []  # coefficients matrix
        self.solutions = []  # x vector
        self.lines_swapped = 0
        self.iterations_count = 0
        self.epsilon = 0.00000001

    def set_matrix(self, matrix):
        self._drop_context()
        self.matrix = matrix[:, :-1]
        self.constants = matrix[:, -1]
        self.lines_number = matrix.shape[0]
        self.solutions = np.zeros(self.lines_number)
```

```
    def set_epsilon(self, epsilon):
        self.epsilon = epsilon

    def calculate_discrepancies(self):
        try:
            return max(abs(self.matrix.dot(self.solutions) - self.constants))
        except ValueError:
            return np.inf

    def _add_any_line(self, matrix, target_string, target_element):
        """
        Добавляет к строке target_string любую строку так,
        чтобы в позиции target_element не было нуля
        """
        if matrix[target_string][target_element] == 0:
            for i in range(self.lines_number):
                if matrix[target_string][target_element] +
matrix[i][target_element] != 0:
                    matrix[target_string] += matrix[i]
                    break

        return matrix

    def find_determinant(self):
        if self.determinant is not None:
            return self.determinant
        self.determinant = round(np.linalg.det(self.matrix), 11)
        return self.determinant

    def _start_calculating(self):
        n = self.lines_number
        # Начальные значения вектора X задаются нулями
        self.solutions = np.zeros(n)  # X vector

        coeffs_matrix = np.zeros((n, n))  # C's matrix
        # заполнение coeffs_matrix
        for i in range(n):
            if self.matrix[i][i] == 0:
                # эта функция заполняет нули на диагонали
                self._add_any_line(self.matrix, i, i)
            coeffs_matrix[i] = self.matrix[i] / self.matrix[i][i]
        np.fill_diagonal(coeffs_matrix, 0)  # заполнение диагонали
coeffs_matrix нулями

        # на данный момент на диагонали matrix нет нулей
        free_members = self.constants / np.diag(self.matrix)  # D's vector

        # Начало решения
        k = self.iterations_count
        self.solutions = np.zeros(n)
        last_discrepancies = self.calculate_discrepancies()
        while last_discrepancies > self.epsilon and k < 99999:
            # Если бы мы использовали метод простой итерации:
            # self.solutions = free_members -
coeffs_matrix.dot(self.solutions)

            # Метод Гаусса-Зейделя заключается в том, что элементы вектора
иксов берутся из предыдущей итерации
            for i in range(n):
                self.solutions[i] = free_members[i] - sum(coeffs_matrix[i] *
self.solutions)

            k += 1
```

```
            if last_discrepancies - self.calculate_discrepancies() < 0:
                raise MatrixError("Итерационный метод расходится.")
            last_discrepancies = self.calculate_discrepancies()

        self.iterations_count += k

        return True

    def solve(self):
        if self.matrix == [] or self.lines_number == 0:
            print("Матрица не задана.")
            return
        if self.find_determinant() == 0:
            print("Определитель матрицы равен нулю. Матрица имеет бесконечное
число решений.")
            return
        try:
            self._start_calculating()
            print("_____________РЕШЕНИЕ_____________")
            print("Определитель:", self.determinant)
            print("Количество итераций:", self.iterations_count)
            print("Вектор решений:", self.solutions)
            print("Расхождение:", self.calculate_discrepancies())
            print("________________________________")
        except MatrixError:
            print("У данной матрицы итерационный метод расходится. \n"
                  "Попробуйте поменять местами столбцы так, чтобы в каждой
строке диагональные "
                  "элементы были больше суммы остальных.")
        except Exception:
            print("Не удалось найти решение для данной матрицы. Попробуйте
другую")
        return
```

MatrixError.py:

```
class MatrixError(ValueError):
    def __init__(self, message):
        # Call the base class constructor with the parameters it needs
        super().__init__(message)
```

3. Пример работы:

```
Добро пожаловать в решатель СЛАУ методом Гаусса-Зейделя!

Хотите считать матрицу из csv-файла? [yes/no]: yes
Введите название файла без ".csv": matrix
Хотите ввести точность? По умолчанию epsilon равно 10^-8.
Ввести epsilon? [yes/no]: no
_____________РЕШЕНИЕ_____________
Определитель: 54.6
Количество итераций: 12
Вектор решений: [1.4 2.3]
Расхождение: 4.272811437999735e-09
________________________________
Для продолжения нажмите любую клавишу . . .
```

```
Добро пожаловать в решатель СЛАУ методом Гаусса-Зейделя!

Хотите считать матрицу из csv-файла? [yes/no]: no
Введите количество строк матрицы: 4
Введите матрицу вида
a_11, a_12, ..., a_1n, b_1
...
a_n1, a_n2, ..., a_nn, b_n

56 3 12 7 90
1 43 2 4 5
0 0 1 0 0
0 2 0 3 0

Хотите ввести точность? По умолчанию epsilon равно 10^-8.
Ввести epsilon? [yes/no]: no
_____________РЕШЕНИЕ_____________
Определитель: 6781.0
Количество итераций: 8
Вектор решений: [ 1.6096446   0.0840584   0.         -0.05603893]
Расхождение: 6.861951007408607e-10
_________________________________
Для продолжения нажмите любую клавишу . . .
```