САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ

ИССЛЕДРОВАТЕЛЬСКИЙ

УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,

МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

**Отчет**

По лабораторной работе №1

Вариант 14

Студент

Федоров Евгений Константинович

Преподаватель

Наумова Надежда Александровна

Санкт-Петербург, 2025 г.

Оглавление

[Цель работы 3](#_Toc191135346)

[Описание метода 4](#_Toc191135347)

[Расчетные формулы 4](#_Toc191135348)

[Листинг программы 5](#_Toc191135349)

[Примеры и результаты работы программы 11](#_Toc191135350)

[Пример 1 11](#_Toc191135351)

[Пример 2 12](#_Toc191135352)

# Цель работы

Цель работы – изучение вычислительного метода решения системы

алгебраических линейных уравнений СЛАУ, а также реализация его на ЭВМ на

одном из выбранных языков программирования.

## 

# Описание метода

Метод Гаусса-Зейделя является итерационным методом решения

СЛАУ. Он модифицирует метод простых итераций за счет

использования на k+1 итерации алгоритма уже полученных на этом

же шаге значений.

## Расчетные формулы

Изображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Автоматически созданное описание

# Листинг программы

Файл matrixReader.py:

from os.path import exists  
  
  
def readMatrixFromCIN():  
 print("Введите размерность матрицы ≤20: ")  
 size = int(input());  
 if size > 20:  
 print("Вы хотите ввести слишком большую СЛАУ, ОШИБКА")  
 exit(1)  
 print("Введите точность: ")  
 accuracy = float(input())  
 print("Введите матрицу: ")  
 matrix = [input().strip().split(" ") for i in range(size)]  
 for i in range(size):  
 try:  
 #fprint(len(matrix[i]))  
 if len(matrix) != size or len(matrix[i]) != size+1:  
 print("Данные чета невалидны")  
 exit(1)  
 except IndexError:  
 print("Данные чета невалидны")  
 print("Данные чета невалидны")  
 exit(1)  
 return matrix, accuracy, size  
  
  
def readMatrixFromTXT(FilePath):  
 if not exists(FilePath):  
 print("Такого файла нет:(")  
 exit(1)  
 with open(FilePath) as f:  
 size = int(f.readline());  
 if size > 20:  
 print("Вы хотите ввести слишком большую СЛАУ, ОШИБКА")  
 exit(1)  
 accuracy = float(f.readline())  
 matrix = [i.strip().split(" ") for i in f.readlines()]  
 for i in range(size):  
 try:  
 if matrix.\_\_len\_\_() != size or matrix[i].\_\_len\_\_() != size+1:  
 print("Данные чета невалидны")  
 exit(1)  
 except IndexError:  
 print("Данные чета невалидны")  
 exit(1)  
 return matrix, accuracy, size

Файл matrixValidator.py:

def check\_diagonal\_dominance(matrix, n):  
 for i in range(n):  
 sum\_non\_diag = sum(abs(matrix[i][j]) for j in range(n) if j != i)  
 if abs(matrix[i][i]) <= sum\_non\_diag:  
 return False  
 return True  
  
  
def parseMatrixValues(matrix, precision=3):  
 if matrix is None:  
 print("Ошибка: матрица не была загружена.")  
 exit(1)  
  
 try:  
 float\_matrix = [[round(float(element), precision) for element in row] for row in matrix]  
 except ValueError:  
 print("Данные в матрице невалидны")  
 exit(1)  
  
 return float\_matrix  
  
  
def swap\_rows(matrix, row1, row2):  
 matrix[row1], matrix[row2] = matrix[row2], matrix[row1]  
  
  
def count\_norm(M):  
 res = 0  
 for i in M:  
 res = max(res, sum([abs(j) for j in i]))  
 return res  
  
def do\_diagonal\_dominance(matrix):  
  
  
 for i in range(len(matrix)):  
 max\_row = i  
 for j in range(i + 1, len(matrix)):  
 if abs(matrix[j][i]) > abs(matrix[max\_row][i]):  
 max\_row = j  
 if max\_row != i:  
 swap\_rows(matrix, max\_row, i)  
  
  
 if not check\_diagonal\_dominance(matrix, len(matrix)):  
 print("Матрицу нельзя привести к диагонально доминирующему виду.")  
 return None  
  
 return matrix

Файл calculations.py:

import solution.matrixPrinter as printer  
  
  
def make\_C\_and\_D\_matrix(matrix):  
 C = []  
 D = []  
  
 for i in range(len(matrix)):  
 a\_ii = matrix[i][i]  
 C\_row = [0 if i == j else (-matrix[i][j] / a\_ii) for j in range(len(matrix))]  
 C.append(C\_row)  
 D\_row = matrix[i][len(matrix)] / a\_ii  
 D.append(D\_row)  
  
 return C, D  
  
  
def get\_accuracy(x\_k, x\_k1):  
 return max(abs(x\_k[i] - x\_k1[i]) for i in range(len(x\_k)))  
  
  
def main\_calculation(C, D, EPS, n):  
 iterations = 0  
 x\_k = D[:]   
 x\_k1 = [0] \* n   
  
 while iterations == 0 or get\_accuracy(x\_k, x\_k1) >= EPS:  
 x\_k, x\_k1 = x\_k1, [0] \* n   
 for i in range(n):  
 x = sum(C[i][j] \* x\_k[j] for j in range(i, n))  
 x += sum(C[i][k] \* x\_k1[k] for k in range(i))  
 x += D[i]  
 x\_k1[i] = round(x, 6)  
  
 iterations += 1  
 printer.printFinalTable(iterations, x\_k1, round(get\_accuracy(x\_k, x\_k1), 5))   
  
  
  
 return x\_k, iterations

Файл matrixPrinter.py:

def printMainMatrix(matrix):  
 for i in range(len(matrix)):  
 print(f"x{i}", end=" " \* 12)  
 print()  
 for i in range(len(matrix)):  
 for j in range(len(matrix[i])):  
 print(matrix[i][j], end=" " \* 10)  
 print()  
 print()  
  
  
def printSupportiveMatrix(matrix):  
 for i in range(len(matrix)):  
 for j in range(len(matrix[i])):  
 print(matrix[i][j], end=" " \* 10)  
 print()  
 print()  
  
  
def printFinalTable(iteration, xk, accuracy):  
 print(f"{iteration}".center(36) + " | " + f"{xk}".center(36) + " | " + f"{accuracy}".center(36) + " |")

Файл main.py:

import solution.matrixReader as reader  
import solution.matrixValidator as validator  
import solution.calculations as calculations  
import solution.matrixPrinter as printer  
  
def getData():  
 flag = input("Если хотите вводить данные через файл, напишите f, иначе введите любой символ: ") == "f"  
  
 if flag:  
 print("Введите название файла: ")  
 matrix, accuracy, n = reader.readMatrixFromTXT(input().strip())  
 else:  
 matrix, accuracy, n = reader.readMatrixFromCIN()  
 return matrix, accuracy, n  
  
  
def main():  
 matrix, accuracy, n = getData()  
  
 matrix = validator.parseMatrixValues(matrix)  
 print("Изначальная матрица: \n\n")  
 printer.printMainMatrix(matrix)  
  
 if not validator.check\_diagonal\_dominance(matrix, n):  
  
 print("Матрица не обладает диагональным преобладанием, попробуем это исправить!!\n\n")  
 matrix = validator.do\_diagonal\_dominance(matrix)  
 if (matrix == None):  
 print("К сожалению, нам не получится привести ее к нужному виду:(")  
 exit(1)  
  
 print("Матрица с диагональным преобладанием: ")  
 printer.printMainMatrix(matrix)  
 C, D = calculations.make\_C\_and\_D\_matrix(matrix)  
  
 print("Матрица C:")  
 printer.printSupportiveMatrix(C)  
  
 print("Матрица D:")  
 for i in range(len(D)):  
 print(D[i], end="")  
 print()  
 print("Номер итерации".center(37) + "|" + " x^k ".center(38) + "|" + " max( | x^(k)\_i-x^(k+1) | ) ".center(36))  
 print("-"\*116)  
 xk, k = calculations.main\_calculation(C, D , accuracy, n)  
  
 print("\nИТОГО\n")  
 print(f"\t количество итераций: {k}\n\t ответ с учетом погрешности: {xk}\n\t с округлением: {list([round(i, 3) for i in xk])}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Примеры и результаты работы программы

## Пример 1

Содержимое файла m:

3  
0.01  
-3 9 1 5  
5 -2 3 10  
2 -1 7 7

Вывод программы:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, черный

Автоматически созданное описание

## Пример 2

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, черный

Автоматически созданное описание

# Вывод

В ходе выполнения лабораторной работы вспомнил методы решения СЛАУ. Вспомнил свойства матриц, вспомнил про такой прекрасный язык, как Python, заново ознакомился с его синтаксическими особенностями, постарался максимально проникнуться!