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

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа

по вычислительной математике №1

Вариант: 16

Преподаватель: Машина Е.А.

Выполнила: Шайхутдинова Н.В.

Группа: P3208

Санкт-Петербург

2024г

Цель работы: изучить прямые и итерационные методы решения систем линейных алгебраических уравнений, выполнить программную реализацию методов.

Описание используемого метода:

Метод Гаусса-Зейделя является модификацией метода простой итерации и обеспечивает более быструю сходимость к решению системы уравнений. Идея метода: при вычислении компонента $x_i^{(k+1)}$ вектора неизвестных на (k+1)-й итерации используются $x_1^{(k+1)}, x_2^{(k+1)}), \ldots, x_{i-1}^{(k+1)}$, уже вычисленные на (k+1)-й итерации. Значения остальных компонент $x_{i+1}^{(k+1)}, x_{i+2}^{(k+1)}, \ldots, x_n^{(k+1)}$ берутся из предыдущей итерации.

Так же как и в методе простых итераций строится эквивалентная СЛАУ и за начальное приближение принимается вектор правых частей (как правило, но может быть выбран и нулевой вектор): $x_i^0 = (d_1, d_2, \ldots, d_n)$.

Тогда приближения к решению системы методом Зейделя определяются следующей системой равенств:

$$x_1^{(k+1)} = c_{11}x_1^{(k)} + c_{12}x_2^{(k)} + \ldots + c_{1n}x_n^{(k)} + d_1$$

$$x_2^{(k+1)} = c_{21}x_1^{(k+1)} + c_{22}x_2^{(k)} + \ldots + c_{2n}x_n^{(k)} + d_2$$

……

$$x_n^{(k+1)} = c_{n1}x_1^{(k+1)} + c_{n2}x_2^{(k+1)} + \ldots + c_{nn-1}x_{n-1}^{(k+1)} + c_{nn}x_n^{(k)} + d_n$$

Расчетные формулы метода:

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^k \qquad i = 1, 2, \ldots, n$$

Листинг программы:

```
import random

def checkDiagonalDominance(matrix, n):

   counter = 0

   sum = 0

   for i in range(n):

       for j in range(n):

           sum += abs(matrix[i][j])

       if sum - 2 * abs(matrix[i][i]) < 0:
```

```
            counter += 1
        sum = 0
    if counter == n:
        return True
    else:
        return False
def printMatrix(matrix, n):
    for i in range(n):
        for j in range(n + 1):
            print(matrix[i][j], end=" ")
        print("\n")
def calculateSum(line, n, matrix, xArray):
    sum = 0
    for i in range(n):
        sum += xArray[i] * matrix[line][i]
    sum -= xArray[line] * matrix[line][line]
    return sum
def generalMathFuntion(n, e, M, matrix):
    for i in range(n - 1):
        if checkDiagonalDominance(matrix, n) == True:
            print("Матрица прошла проверку на диаганальное преобладание :)")
            break
        else:
            firstLine = matrix[i]
            matrix.pop(i)
            matrix.append(firstLine)
            print("Проверка на диаганальное преобладание не пройдена :(")
            print("Новая матрица:")
            printMatrix(matrix, n)
    eArray = []
    counter = 0
    xArray = []
    xArrayLast = []
    eResult = False
    err = False
    for i in range(n):
```

```
        x = matrix[i][-1] / matrix[i][i]
        xArray.append(x)
        xArrayLast.append(x)
    for i in range(n):
        eArray.append(0)
    print("Итерация -> x1, x2, ... xn -> Максимальное абсолютное отклонение")
    while counter < M and eResult == False and err == False:
        for i in range(n):
            xArray[i] = (matrix[i][-1] - calculateSum(i, n, matrix, xArray)) / matrix[i][i]
        for i in range(n):
            eArray[i] = abs(xArray[i] - xArrayLast[i])
        if max(eArray) < e:
            eResult = True
        for i in range(n):
            xArrayLast[i] = xArray[i]
        counter += 1
        print(counter, "->", xArray, "->", max(eArray))
        if eArray[-1] > eArray[-2]:
            err = True
    answer = []
    answer.append(counter)
    answer.append(xArray)
    answer.append(eArray)
    answer.append(err)
    print(answer)
    return answer
def checkN(n):
    try:
        n = int(n)
        if n > 0 and n <= 20:
            checkerForN = True
            return checkerForN
        else:
            print("Порядок матрицы должен быть больше нуля и <= 20!")
    except Exception:
        print("Порядок матрицы должен быть целым числом!")
```

```
def checkE(e):
    try:
        e = float(e)
        if e > 0:
            checkerForE = True
            return checkerForE
        else:
            print("Погрешность вычислений должна быть больше нуля!")
    except Exception:
        print("Погрешность вычислений должна быть числом!")
def checkM(M):
    try:
        M = int(M)
        if M > 0:
            checkerForM = True
            return checkerForM
        else:
            print("Максимальное число итераций должно быть больше нуля!")
    except Exception:
        print("Максимальное число итераций должно быть целым числом!")
def checkNumbers(numbers, n, i):
    try:
        numbers = numbers.split()
        numbers = list(map(float, numbers))
        if len(numbers) == n + 1:
            checkerForNumbers = True
            return checkerForNumbers
        else:
            print("Количество введённых чисел в строке", i + 1, "должно быть равно", n + 1)
    except Exception:
        print("Коэффициенты и правая часть уравнения должны быть числами!")
def consoleInput(n, matrix):
    for i in range(n):
        checkerForNumbers = False
        while checkerForNumbers == False:
            print("Введите через пробел коэффициенты и правую часть уравнения для", i + 1, "строки:")
```

```
            numbers = input()
            if checkNumbers(numbers, n, i) == True:
                checkerForNumbers = True
                numbers = numbers.split()
                numbers = list(map(float, numbers))
        matrix.append(numbers)
    return matrix
def randomInput(n, matrix):
    for i in range(n):
        numbers = []
        for i in range(n + 1):
            number = round(random.uniform(-100, 100), 5)
            numbers.append(number)
        matrix.append(numbers)
    return matrix
def consoleInputForNME(inputSelection):
    checkerForN = False
    while checkerForN == False:
        print("Введите порядок матрицы:")
        n = input()
        if checkN(n) == True:
            checkerForN = True
            n = int(n)
    checkerForE = False
    while checkerForE == False:
        print("Введите погрешность вычислений:")
        e = input()
        if checkE(e) == True:
            checkerForE = True
            e = float(e)
    checkerForM = False
    while checkerForM == False:
        print("Введите максимальное число итераций:")
        M = input()
        if checkM(M) == True:
            checkerForM = True
```

```
            M = int(M)
    matrix = []
    if inputSelection == "1":
        matrix = consoleInput(n, matrix)
    elif inputSelection == "3":
        matrix = randomInput(n, matrix)
    print("Исходная матрица:")
    printMatrix(matrix, n)
    answer = generalMathFuntion(n, e, M, matrix)
    if answer[0] >= M:
        print("Не удалось получить ответ за указанное количество итераций")
    elif answer[3] == True:
        print("Погрешность увеличивается")
    else:
        print("Ответ:")
        print(answer[0], "->", answer[1], "->", max(answer[2]))
def readFile(file):
    try:
        while (True):
            n = file.readline().strip()
            n = int(n)
            e = file.readline().strip()
            e = float(e)
            M = file.readline().strip()
            M = int(M)
            matrix = []
            for i in range(n):
                numbers = file.readline().strip()
                if checkNumbers(numbers, n, i) == True:
                    numbers = numbers.split()
                    numbers = list(map(float, numbers))
                    matrix.append(numbers)
            if checkM(M) and checkN(n) and checkE(e) and len(matrix) == n:
                print("Исходная матрица:")
                printMatrix(matrix, n)
                counter, xArray, eArray, err = generalMathFuntion(n, e, M, matrix)
```

```
            if counter >= M:

                print("Не удалось получить ответ за указанное количество итераций")

            elif err == True:

                print("Погрешность увеличивается")

            else:

                print("Ответ:")

                print(counter, "->", xArray, "->", max(eArray))

    except Exception:

        print("Некорректные данные в файле или данных не хватает!")

def fileInput():

    checkerForFile = False

    while checkerForFile == False:

        print("Введите путь к файлу:")

        fileInput = input()

        try:

            file = open(fileInput, "r")

            checkerForFile = True

            readFile(file)

        except Exception:

            print("Не получилось найти файл с таким именем!")

while (True):

    checkerForInputSelection = False

    while checkerForInputSelection == False:

        print(

            "Введите \"1\" для ввода входных данных через консоль, \"2\" для чтения из файла, \"3\" для
генерации рандомной матрицы, \"4\" для выхода из программы:")

        inputSelection = input()

        if inputSelection == "1":

            checkerForInputSelection = True

            consoleInputForNME(inputSelection)

        elif inputSelection == "2":

            checkerForInputSelection = True

            fileInput()

        elif inputSelection == "3":

            checkerForInputSelection = True

            consoleInputForNME(inputSelection)
```

```
    elif inputSelection == "4":

        checkerForInputSelection = True

        exit()

    else:

        print("Я Вас не понимаю :(")
```

Примеры и результаты работы программы:

1 пример.

input (test1):

```
3

0.01

10

5 -1 3 5

1 -4 2 20

2 -1 5 10
```

output:

Введите "1" для ввода входных данных через консоль, "2" для чтения из файла, "3" для генерации рандомной матрицы:

2

Введите путь к файлу:

test1

Исходная матрица:

5.0 -1.0 3.0 5.0

1.0 -4.0 2.0 20.0

2.0 -1.0 5.0 10.0

Матрица прошла проверку на диагональное преобладание :)

Итерация -> x1, x2, ... xn -> Максимальное абсолютное отклонение

1 -> [-1.2, -4.3, 1.6199999999999999] -> 2.2

2 -> [-0.8320000000000001, -4.398000000000001, 1.4532] -> 0.3679999999999999

3 -> [-0.75152, -4.4612799999999995, 1.408352] -> 0.08048000000000011

4 -> [-0.7372672000000001, -4.4801408, 1.3988787200000001] -> 0.018860800000000566

5 -> [-0.7353553920000003, -4.484399487999999, 1.3972622592000004] -> 0.004258687999999289

Ответ:

5 -> [-0.7353553920000003, -4.484399487999999, 1.3972622592000004] -> 0.004258687999999289

2 пример.

Введите "1" для ввода входных данных через консоль, "2" для чтения из файла, "3" для генерации рандомной матрицы:

3

Введите порядок матрицы:

3

Введите погрешность вычислений:

0.01

Введите максимальное число итераций:

10

Исходная матрица:

58.50867 -53.04924 88.29016 -62.63536

-58.08987 -40.61872 92.87043 -67.12536

72.31872 -1.74887 73.59774 -52.14268

Проверка на диагональное преобладание не пройдена :(

Новая матрица:

-58.08987 -40.61872 92.87043 -67.12536

72.31872 -1.74887 73.59774 -52.14268

58.50867 -53.04924 88.29016 -62.63536

Проверка на диагональное преобладание не пройдена :(

Новая матрица:

-58.08987 -40.61872 92.87043 -67.12536

58.50867 -53.04924 88.29016 -62.63536

72.31872 -1.74887 73.59774 -52.14268

Итерация -> x1, x2, ... xn -> Максимальное абсолютное отклонение

1 -> [-0.8027267386009342, -0.8837659153473649, 0.059293920560884124] -> 2.064468221355896

2 -> [1.8683025484462865, 3.3399597619221417, -2.464950329283761] -> 4.223725677269506

3 -> [-5.120695353424042, -8.569407847662598, 4.119592166376747] -> 11.90936760958474

4 -> [13.733754849960068, 23.18409353048909, -13.652651370683904] -> 31.75350137815169

5 -> [-36.882686206807676, -62.219860829854746, 34.05473495012861] -> 85.40395436034383

6 -> [99.10668321070973, 167.16419651392562, -94.12058991059656] -> 229.38405734378037

7 -> [-266.20631082622293, -449.06701348690126, 250.20057880850345] -> 616.2312100008269

8 -> [715.165793661705, 1206.3562816684307, -674.7796488734184] -> 1655.423295155332

9 -> [-1921.1714333575633, -3240.74300049393, 1810.0675111723601] -> 4447.09928216236

10 -> [5161.032482610647, 8705.85388059389, -4865.1765533108755] -> 11946.59688108782

Не удалось получить ответ за указанное количество итераций

3 пример.

Введите "1" для ввода входных данных через консоль, "2" для чтения из файла, "3" для генерации рандомной матрицы:

1

Введите порядок матрицы:

2

Введите погрешность вычислений:

0.01

Введите максимальное число итераций:

8

Введите через пробел коэффициенты и правую часть уравнения для 1 строки:

1 2 3

Введите через пробел коэффициенты и правую часть уравнения для 2 строки:

6 5 4

Исходная матрица:

1.0 2.0 3.0

6.0 5.0 4.0

Проверка на диагональное преобладание не пройдена :(

Новая матрица:

6.0 5.0 4.0

1.0 2.0 3.0

Итерация -> x1, x2, ... xn -> Максимальное абсолютное отклонение

1 -> [-0.5833333333333334, 1.7916666666666667] -> 1.25

2 -> [-0.826388888888889, 1.9131944444444444] -> 0.24305555555555558

3 -> [-0.9276620370370369, 1.9638310185185184] -> 0.10127314814814792

4 -> [-0.9698591820987653, 1.9849295910493827] -> 0.04219714506172845

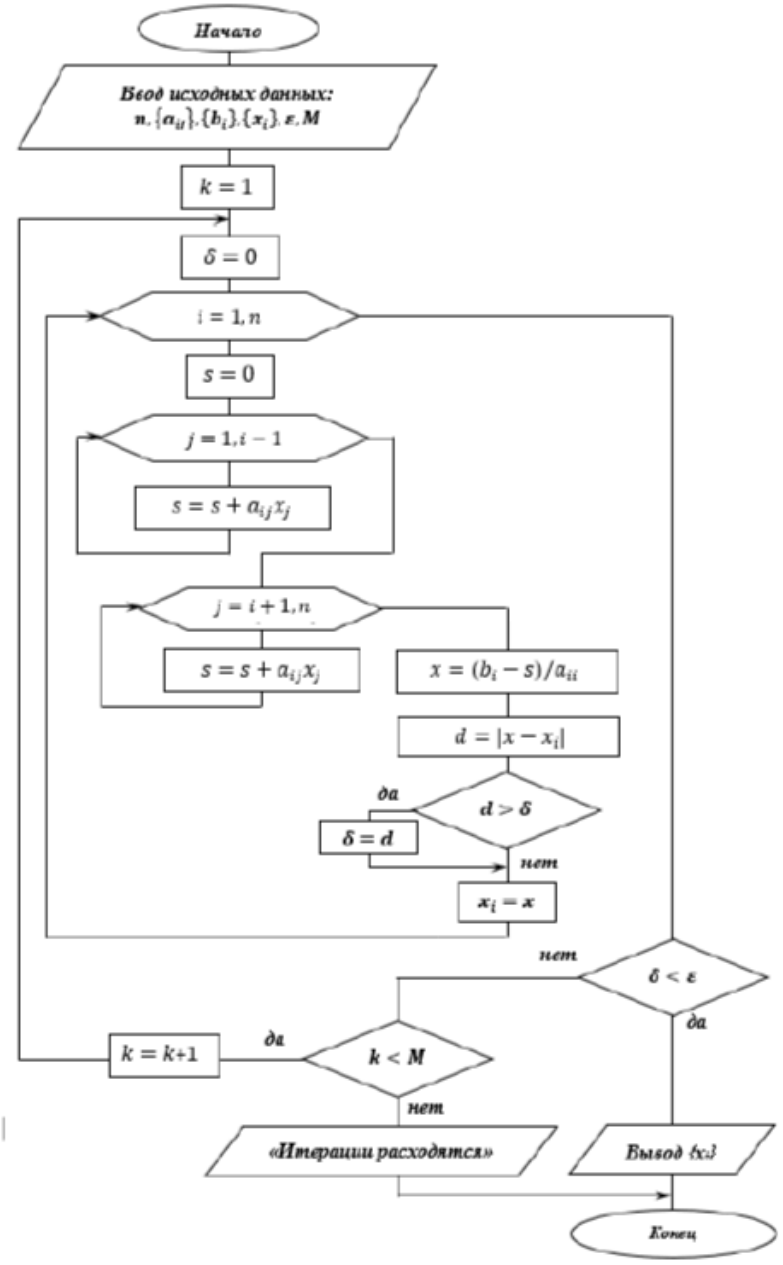
5 -> [-0.9874413258744855, 1.9937206629372428] -> 0.017582143775720205

6 -> [-0.9947672191143692, 1.9973836095571846] -> 0.007325893239883641

Ответ:

6 -> [-0.9947672191143692, 1.9973836095571846] -> 0.007325893239883641

Блок-схема алгоритма:



Вывод:

В ходе выполнения лабораторной работы я познакомилась с методом Гаусса-Зейделя и написала его реализация по питоне.

Достоинства метода:

Является универсальным и простым для реализации на ЭВМ.

Обеспечивает более быструю сходимость (по сравнению с методом простой итерации)

Недостатки метода:

Является трудоемким