***Санкт-Петербургский национальный исследовательский университет ИТМО***

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

**Отчет**

По лабораторной работе №1

Вариант 9

Выполнила:

Миличенко А.Е.

P3210

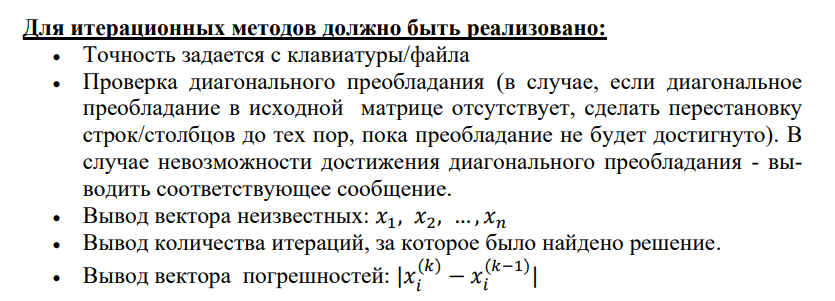
Преподаватель:

Малышева Т.А.

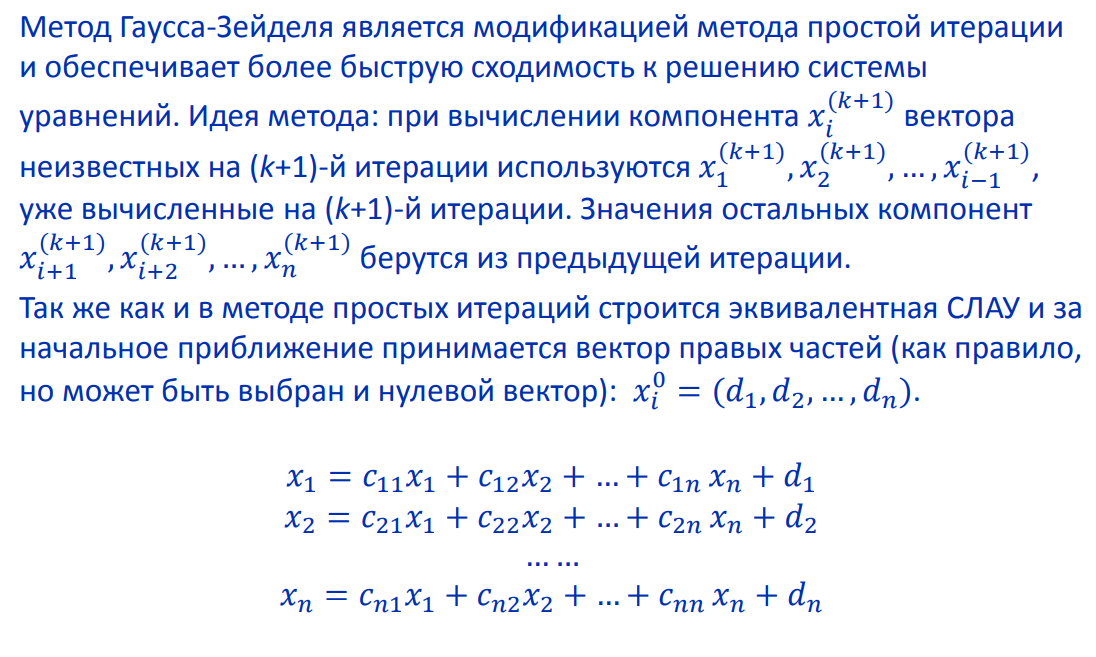
Санкт-Петербург

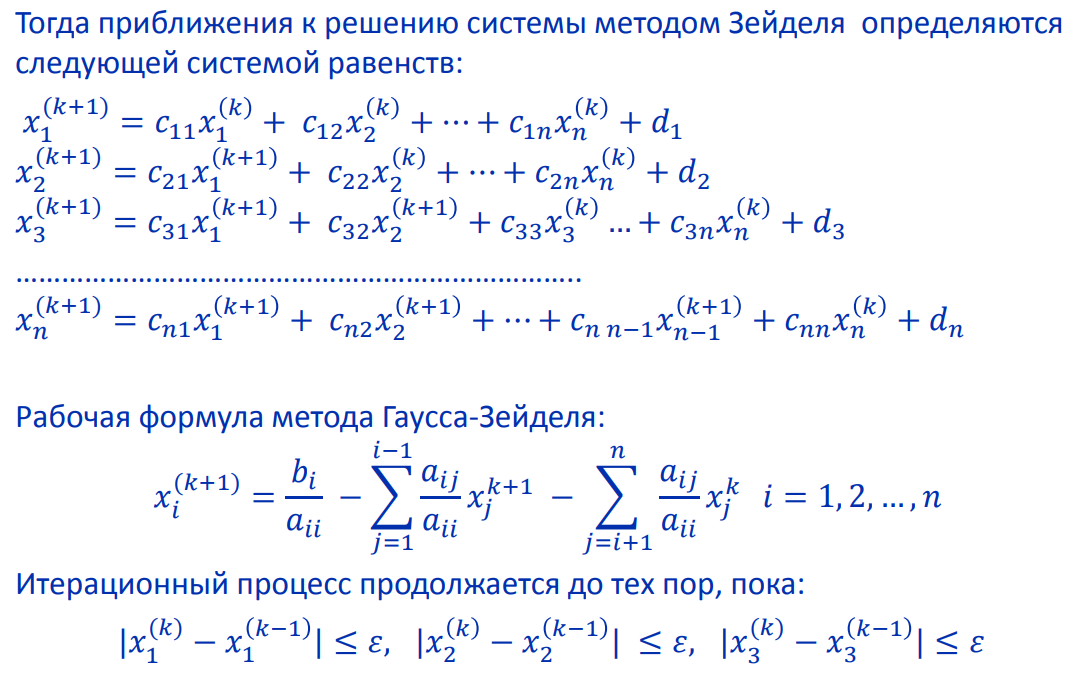
2024 г.

**Цель работы**

Реализовать программу, которая бы решала систему линейных алгебраических уравнений методом Гаусса – Зейделя. Также должно быть реализовано: 

**Описание метода и расчётные формулировки**





**Листинг программы**

mport math

import random

# Функция для получения матрицы из файла

def get\_matrix\_file(inp='matrix.txt'):

with open(inp, 'rt') as file:

try:

n = int(file.readline())

eps = float(file.readline())

max\_iterations = int(file.readline())

matrix = []

for line in file:

new\_row = list(map(float, line.strip().split()))

if len(new\_row) != (n + 1):

raise ValueError

matrix.append(new\_row)

if len(matrix) != n:

raise ValueError

except ValueError:

return None

return matrix, n, eps, max\_iterations

# Функция для получения матрицы с клавиатуры

def get\_matrix\_keyboard():

while True:

try:

n = int(input('Введите порядок матрицы: '))

if n <= 0:

print('Порядок матрицы должен быть натуральным числом')

else:

break

except ValueError:

print('Порядок матрицы должен быть целым числом')

while True:

try:

eps = float(input('Введите точность ответа: '))

if eps <= 0:

print('Точность ответа не может быть меньше или равна нулю')

else:

break

except ValueError:

print('Точность матрицы должна быть числом больше нуля')

while True:

try:

max\_iterations = int(input('Введите максимальное число итераций: '))

if max\_iterations <= 0:

print('Максимальное количество итераций должно быть натуральным числом')

else:

break

except ValueError:

print('Максимальное количество итераций должно быть целым натуральным числом')

matrix = []

print('Введите коэффициенты матрицы вместе с вектором b через пробел: ')

try:

for \_ in range(n):

matrix.append(list(map(float, input().strip().split())))

if len(matrix[-1]) != (n + 1):

raise ValueError

except ValueError:

return None

return matrix, n, eps, max\_iterations

# Функция для создания рандомной матрицы

def get\_matrix():

while True:

try:

n = int(input("Введите порядок матрицы: "))

if n <= 0:

print("Порядок матрицы должен быть натуральным числом.")

else:

break

except ValueError:

print("Порядок матрицы должен быть натуральным числом.")

while True:

try:

eps = float(input("Введите точность ответа: "))

if eps <= 0:

print("Точность матрицы должна быть положительным числом.")

else:

break

except ValueError:

print("Точность матрицы должна быть числом.")

while True:

try:

max\_iterations = int(input("Максимальное количество итераций: "))

if max\_iterations <= 0:

print("Максимальное количество итераций должно быть положительным числом.")

else:

break

except ValueError:

print("Максимальное количество итераций должно быть целым числом.")

matrix = [[0] \* (n + 1) for \_ in range(n)]

for i in range(n):

for j in range(n + 1):

matrix[i][j] = random.randrange(1, 100)

return matrix, n, eps, max\_iterations

# Функция для печати матрицы

def print\_arr(matrix):

for row in matrix:

output = ' '.join(map(str, row[:-1])) + ' | ' + str(row[-1])

print(output)

# Функция для проверки матрицы на диагональное преобладание

def check\_diagonal\_dominance(matrix):

not\_equals = 0

for i in range(len(matrix)):

summ = 0

for j in range(len(matrix)):

if i == j:

continue

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

if summ > abs(matrix[i][i]):

return False

if summ < abs(matrix[i][i]):

not\_equals = 1

if not\_equals == 0:

return False

return True

# Основная функция решения уравнения

def solve\_equation(matrix, n, epsilon, max\_iterations):

vector\_old\_ans = [0] \* n

vector\_ans = [0] \* n

difference = epsilon + 1

num = 0

found\_answer = True

errors = [0] \* n

while difference > epsilon:

for i in range(n):

summ = 0

for j in range(n):

if i != j:

summ += matrix[i][j] / matrix[i][i] \* vector\_ans[j]

vector\_ans[i] = matrix[i][n] / matrix[i][i] - summ

for i in vector\_ans:

if i == None or i == math.inf or i == -math.inf:

print("Значения расходятся, невозможно найти ответ")

found\_answer = False

return None

max\_difference = 0.0

for i in range(n):

errors[i] = abs(vector\_old\_ans[i] - vector\_ans[i])

if abs(vector\_old\_ans[i] - vector\_ans[i]) > max\_difference:

max\_difference = abs(vector\_old\_ans[i] - vector\_ans[i])

difference = max\_difference

for i in range(n):

vector\_old\_ans[i] = vector\_ans[i]

num += 1

if num >= max\_iterations:

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

found\_answer = False

break

if found\_answer:

print("Ответ найден за " + str(num) + " итераций:")

print([vector\_ans[i] for i in range(n)])

print("Вектор погрешностей:")

print(errors)

def ensure\_diagonal\_dominance(matrix, n):

if check\_diagonal\_dominance(matrix):

print("Диагональное преобладание выполнено")

else:

print("Диагональное преобладание не выполнено")

for i in range(n):

max\_element = matrix[i][0]

max\_index = 0

for j in range(n):

if matrix[j][i] > max\_element:

max\_element = matrix[j][i]

max\_index = j

if max\_index != i and max\_element != matrix[i][i]:

for j in range(n):

matrix[j][i], matrix[j][max\_index] = matrix[j][max\_index], matrix[j][i]

if check\_diagonal\_dominance(matrix):

print("Матрицу удалось привести к виду, в котором выполняется условие преобладания диагонали")

else:

print("Матрицу не удалось привести к виду, в котором выполняется условие преобладания диагонали")

print("Новый вид матрицы:")

print\_arr(matrix)

your\_input = ""

while your\_input not in ("1", "2", "3"):

your\_input = input("Введите 1, чтобы ввести матрицу с клавиатуры, 2, чтобы взять матрицу из файла, или 3, чтобы сгенерировать матрицу: ")

if your\_input == "1":

array, n, epsilon, max\_iterations = get\_matrix\_keyboard()

elif your\_input == "2":

array, n, epsilon, max\_iterations = get\_matrix\_file()

elif your\_input == '3':

array, n, epsilon, max\_iterations = get\_matrix()

else:

print('Напоминаю, вам нужно ввести 1 - чтобы ввести матрицу с клавиатуры, 2 - чтобы взять матрицу из файла, 3 - чтобы сгенерировать матрицу')

print("Полученный массив:")

print\_arr(array)

print("Полученная точность: " + str(epsilon))

print("Полученное количество итераций: " + str(max\_iterations))

ensure\_diagonal\_dominance(array, n) # Вызываем функцию для проверки и приведения матрицы к диагональному преобладанию

if all(array[i][i] != 0 for i in range(n)):

solve\_equation(array, n, epsilon, max\_iterations)

else:

print("На диагонали есть 0, поэтому ответ вычислить не получится")

**Результат работы программы**

Введите '1' для работы через консоль, или введите '2' для работы с файлом:

2

Чтение данных файла ...

Матрица в треугольном виде:

[6.0, 9.0, 3.0]|5.0

[0, 3.5, 0.5]|-1.5

[0, 0, 3.142857142857143]|4.904761904761905

Определитель матрицы: 66.0

Вектор невязки: [-1.7763568394002505e-15, 0.0, 0.0]

Полученные неизвестные векторы: [1.03030303030303, -0.6515151515151515, 1.5606060606060608]

**Вывод**

В результате выполнения лабораторной работы была создана программа, способная эффективно решать системы линейных уравнений методом Гаусса – Зейделя. Этот метод является универсальным и простым для реализации на ЭВМ. Так же этот метод обеспечивает более быструю сходимость в отличие от метода простой итерации. Из недостатков этого метода могу сказать, что он достаточно трудоемкий.