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

**Факультет программной инженерии и компьютерной техники**

**Вычислительная математика**

**Лабораторная работа №1**

**Вариант 7**

Группа: P3267

Выполнила:

Каунова Ю. Д.

Проверил:

Машина Е. А.

Г. Санкт-Петербург

2024

Оглавление

[Цель работы 3](#_Toc159942100)

[Описание метода, расчётные формулы 3](#_Toc159942101)

[Листинг программы 5](#_Toc159942102)

[Примеры и результаты работы программы 9](#_Toc159942103)

[Вывод 9](#_Toc159942104)

# Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

# Описание метода, расчётные формулы

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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

Автоматически созданное описание

# Листинг программы

def make\_diagonal(matrix):

biggest\_el\_index\_list = []

for i in range(len(matrix)):

if matrix[i] == max(matrix[i][:-1], key=abs):

biggest\_el\_index\_list.append(i)

else:

biggest\_el\_index\_list.append(matrix[i].index(max(matrix[i][:-1], key=abs)))

if len(list(set(biggest\_el\_index\_list))) != len(biggest\_el\_index\_list):

print("Matrix was not provided to diagonal-dominated. Simple iteration method might not be applyed correctly")

error\_lines = set()

for i in range(len(biggest\_el\_index\_list)):

if biggest\_el\_index\_list[i] != i:

error\_lines.add(i)

print("These lines are not diagonal-dominated", error\_lines)

result\_matrix = []

try:

for i in range(len(matrix)):

result\_matrix.append(matrix[biggest\_el\_index\_list.index(i)])

print("Matrix was successfully provided to diagonal-dominated:")

except ValueError as e:

print("Matrix was not successfully provided to diagonal-dominated:")

for line in result\_matrix:

for x in line:

print(x, end=" ")

print()

if len(result\_matrix) != len(matrix):

return matrix

return result\_matrix

def simple\_iteration\_method(accuracy, matrix):

max\_iteration = 1000

matrix = make\_diagonal(matrix)

result\_x = []

d = []

for i in range(len(matrix)):

d.append(matrix[i][-1] / matrix[i][i])

last\_x = d

iteration\_amount = 0

print("There are the results of each iteration:")

while True:

iteration\_amount += 1

result\_x = []

accuracy\_vec = []

for i in range(len(matrix)):

summ = last\_x[i] + d[i]

for q in range(len(matrix[i]) - 1):

summ += - matrix[i][q] / matrix[i][i] \* last\_x[q]

result\_x.append(summ)

accuracy\_vec.append(abs(last\_x[i] - summ))

cur\_accuracy = max(accuracy\_vec)

last\_x = result\_x

if cur\_accuracy <= accuracy:

break

if iteration\_amount >= max\_iteration:

raise ValueError("Programm reached the maximum amount of iterations: " + str(max\_iteration) + " . Accuracy on last iteration was " + str(cur\_accuracy))

print(iteration\_amount, result\_x)

residual = []

for i in range(len(matrix)):

sum = - matrix[i][-1]

for q in range(len(matrix)):

sum += matrix[i][q] \* result\_x[q]

residual.append(sum)

return result\_x, iteration\_amount, accuracy\_vec, residual

def main():

filename = "input.txt"

matrix = []

n = 0

accuracy = 0

choose = input(

"Type 0 to enter matrix from file \"" + filename + "\"\nOr type 1 to enter matrix from keyboard\n")

if choose == "0":

file = open(filename, "r")

n = int(file.readline())

accuracy = float(file.readline())

for line in file:

matrix.append([float(x) for x in line.strip().split(" ")])

file.close()

elif choose == "1":

n = int(input())

accuracy = float(input())

for i in range(n):

line = input()

matrix.append([float(x) for x in line.strip().split(" ")])

else:

raise ValueError("You entered invalid number")

if len(matrix) != n:

raise ValueError("Matrix has invalid number of lines")

elif n > 20:

raise ValueError("n can not be greater than 20")

elif len(matrix) != 0:

result = simple\_iteration\_method(accuracy, matrix)

print("x:", result[0])

print("iteration amount:", result[1])

print("accuracy vector:", result[2])

print("residual:", result[3])

try:

main()

except ValueError as e:

print("Error occurred! Error message:", e)

except KeyboardInterrupt as e:

print(e)

except ZeroDivisionError as e:

print("Couldn't apply simple iteration method to the matrix")

# Примеры и результаты работы программы

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

# Вывод

В ходе работы реализован метод простых итераций, позволяющий решать СЛАУ итерационным методом. Метод показал маленькую погрешность, что характерно для итерационных методов.