Университет ИТМО

Факультет программной инженерии и компьютерной техники

Кафедра вычислительной техники

Вычислительная математика

Лабораторная работа №1

“Решение системы линейный алгебраических уравнений СЛАУ”

Вариант 1

Выполнил: студент группы P32102 Голиков Д.И.

Проверил: преподаватель Рыбаков С.Д.

г. Санкт-Петербург

2023

Цель работы:

Разработать программу для подсчета корней СЛАУ.

Для прямых методов должно быть реализовано:

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 В)   
• Вывод вектора неизвестных: 𝑥1, 𝑥2, … , 𝑥𝑛   
• Вывод вектора невязок: 𝑟1, 𝑟, … , 𝑟𝑛

Описание метода

Метод Основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы

Прямым ходом метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается 𝑥1 из всех последующих уравнений системы. Затем с помощью второго уравнения исключается 𝑥2 из третьего и всех последующих уравнений и т. д.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим единственное в этом уравнении неизвестное 𝑥𝑛. Далее, используя это значение, из предыдущего уравнения вычисляем 𝑥𝑛−1 и т. д. Последним найдем 𝑥1 из первого уравнения.

Код программы:

файл Calculations.py  
import matrix\_io

def calculate\_minor(matrix, dimension, i, j):

return [[matrix[row][col] for col in range(dimension) if col != j] for row in range(dimension) if row != i]

def calculate\_det(matrix, dimension):

if dimension == 1:

return matrix[0][0]

det = 0

sgn = 1

for j in range(dimension):

det += sgn \* matrix[0][j] \* calculate\_det(calculate\_minor(matrix, dimension, 0, j), dimension - 1)

sgn \*= -1

return det

def swap\_columns(matrix, dimension, col1, col2):

for row in range(dimension):

matrix[col1][row], matrix[col2][row] = matrix[col2][row], matrix[col1][row]

def make\_element\_not\_zero(matrix, dimensions, i):

for j in range(i, dimensions):

if matrix[i][j] != 0 and matrix[j][i] != 0:

swap\_columns(matrix, dimensions, i, j)

return

print("Нельзя найти решение")

exit()

def to\_triangle(matrix, dimension):

for i in range(dimension):

if matrix[i][i] == 0:

make\_element\_not\_zero(matrix, dimension, i)

for m in range(i + 1, dimension):

a = -(matrix[m][i] / matrix[i][i])

for j in range(i, dimension):

matrix[m][j] += a \* matrix[i][j]

matrix[m][-1] += a \* matrix[i][-1]

sum = 0

for j in range(dimension):

sum += matrix[i][j]

if sum == 0:

print("Система не имеет решений")

exit()

#matrix\_io.print\_matrix(matrix, dimension)

#print

def calc\_answer(matrix, dimension):

answer = []

answer.append(matrix[dimension - 1][-1]/matrix[dimension - 1][dimension - 1])

i = dimension - 2

while i >= 0:

result = matrix[i][-1]

k = dimension - 1

while k > i:

result -= answer[dimension - 1 - k] \* matrix[i][k]

k -= 1

result /= matrix[i][i]

answer.append(result)

i -= 1

answer.reverse()

return answer

def calc\_discrepancy(matrix, dimension, answer):

discrepancy = []

for i in range(dimension):

sum = 0

for j in range(dimension):

sum += matrix[i][j] \* answer[j]

discrepancy.append([sum, matrix[i][-1], abs(sum - matrix[i][-1])])

return discrepancy

файл matrix\_io.py

import random

def read\_input\_type(types):

input\_num = -1

message = "Введите цифру для выбора варианта ввода "

for i in range(len(types)):

message += types[i] + ": " + str(i) + ", "

message = message[0:-2] + ": "

while input\_num not in range(len(types)):

try:

input\_num = int(input(message))

except Exception:

input\_num = -1

return input\_num

def get\_matrix\_file(filename):

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

try:

dimension = int(file.readline())

matrix = []

for line in file:

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

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

raise ValueError

matrix.append(new\_row)

if len(matrix) != dimension:

raise ValueError

except ValueError:

return None

return matrix, dimension

def get\_matrix\_input():

while True:

try:

dimension = int(input("Порядок матрицы: "))

if dimension <= 0:

print("Порядок матрицы должен быть положительным.")

else:

break

except ValueError:

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

matrix = []

print("Введите коэффициенты матрицы через пробел:")

try:

for i in range(dimension):

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

if len(matrix[i]) != (dimension + 1):

raise ValueError

except ValueError:

return None

return matrix, dimension

def get\_random\_matrix():

while True:

try:

dimension = int(input("Порядок матрицы: "))

if dimension <= 0:

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

else:

break

except ValueError:

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

matrix = [[0]\*(dimension+1) for i in range(dimension)]

for i in range(dimension):

for j in range(dimension + 1):

matrix[i][j] = random.randrange(-20, 20)

return matrix, dimension

def print\_matrix(matrix, dimension):

for i in range(dimension):

output\_string = ""

for j in range(dimension):

output\_string += str(matrix[i][j]) + " "

output\_string += "| " + str(matrix[i][dimension])

print(output\_string)

def print\_answer(answer, dimension):

for i in range(dimension):

print("x" + str(i + 1) + " = " + str(answer[i]))

def print\_discrepancy(discrepancy):

for string in discrepancy:

print(str(string[0]) + " - " + str(string[1]) + " = " + str(string[2]))

#print()

файл main.py

import matrix\_io

import calculations

import copy

input\_type = matrix\_io.read\_input\_type(["с клавиатуры", "из файла", "случайными числами"])

if input\_type == 0:

matrix, dimension = matrix\_io.get\_matrix\_input()

elif input\_type == 1:

matrix, dimension = matrix\_io.get\_matrix\_file("input.txt")

elif input\_type == 2:

matrix, dimension = matrix\_io.get\_random\_matrix()

else:

matrix, dimension = matrix\_io.get\_random\_matrix()

print("Полученная матрица:")

matrix\_io.print\_matrix(matrix, dimension)

det = calculations.calculate\_det(matrix, dimension)

print()

print("Определитель данной матрицы: " + str(det))

if det == 0:

print("Матрица не совместна.")

exit()

triangle\_matrix = copy.deepcopy(matrix)

calculations.to\_triangle(triangle\_matrix, dimension)

print()

print("Матрица в треугольном виде:")

matrix\_io.print\_matrix(triangle\_matrix, dimension)

answer = calculations.calc\_answer(triangle\_matrix, dimension)

print()

print("Полученный ответ: ")

matrix\_io.print\_answer(answer, dimension)

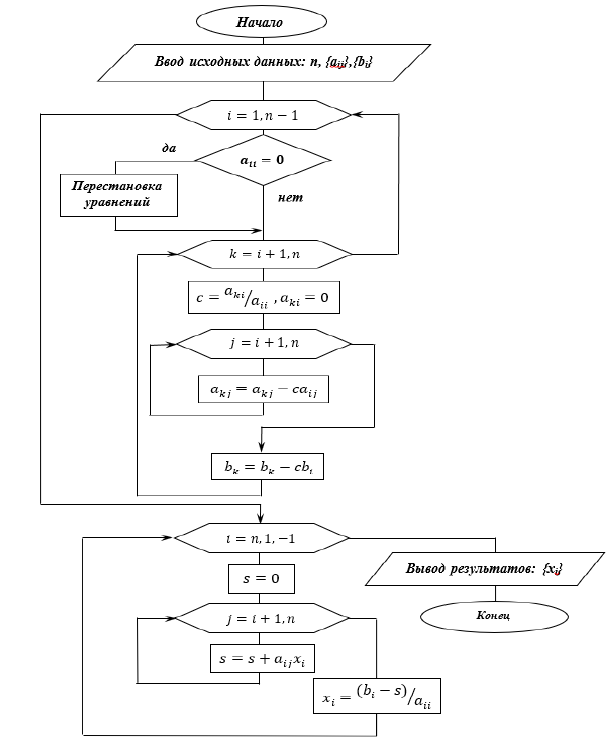
discrepancy = calculations.calc\_discrepancy(matrix, dimension, answer)

print()

print("Полученные невязки: ")

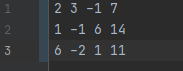
matrix\_io.print\_discrepancy(discrepancy)

Блок схема алгоритма



Пример работы программы

Входной файл input.txt:



Результат работы:

Введите: 1 - для ввода с консоли; 2 - для чтения файла

2

Размерность матрицы:

3

2,00 3,00 -1,00 7,00

1,00 -1,00 6,00 14,00

6,00 -2,00 1,00 11,00

-------

Начало 1й итерации

Максимальный элемент столбца: 6,00

Переставляем строки 3 и 1

Матрица после перестановки:

6,00 -2,00 1,00 11,00

1,00 -1,00 6,00 14,00

2,00 3,00 -1,00 7,00

Матрица после 1го преобразования:

1,00 -0,33 0,17 1,83

0,00 -0,67 5,83 12,17

0,00 3,67 -1,33 3,33

-------

-------

Начало 2й итерации

Максимальный элемент столбца: 3,67

Переставляем строки 3 и 2

Матрица после перестановки:

1,00 -0,33 0,17 1,83

0,00 3,67 -1,33 3,33

0,00 -0,67 5,83 12,17

Матрица после 2го преобразования:

1,00 -0,33 0,17 1,83

0,00 1,00 -0,36 0,91

0,00 0,00 5,59 12,77

-------

-------

Начало 3й итерации

Максимальный элемент столбца: 5,59

Перестановка не требуется

Матрица после перестановки:

1,00 -0,33 0,17 1,83

0,00 1,00 -0,36 0,91

0,00 0,00 5,59 12,77

Матрица после 3го преобразования:

1,00 -0,33 0,17 1,83

0,00 1,00 -0,36 0,91

0,00 0,00 1,00 2,28

-------

Получена треугольная матрица:

1,00 -0,33 0,17 1,83

0,00 1,00 -0,36 0,91

0,00 0,00 1,00 2,28

Определитель матрицы равен:

1.0

Найдены корни СЛАУ:

2,03 1,74 2,28

Вектор невязки:

0,00 0,00 0,00

Вывод

Во время выполнения лабораторной работы я изучил работу прямого метода Гаусса с выбором главного элемента по столбцу. Основной недостаток прямого метода – хранение всей матрицы в памяти. Также метод не учитывает количество нулевых элементов, в результате чего проводятся лишние арифметические операции. Из-за того, что результаты вычисления используются повторно, накапливается вычислительная погрешность.

При решении СЛАУ методом Гаусса может получиться большая погрешность из-за использования маленьких ведущих элементов. Выбор главного максимального элемента позволяет избежать этого.