Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет ПИиКТ



ОТЧЁТ

По лабораторной работе № 1

По предмету: Вычислительная математика

Вариант: Метод Гаусса с выбором главного элемента

Студент:

Андрейченко Леонид Вадимович

Группа P3230

Преподаватель:

Перл Ольга Вячеславовна

Санкт – Петербург

2022

**Описание метода**

Метод Гаусса с выбором главного элемента по столбцам.

Данный метод можно применять, если определитель матрицы не равен 0.

Схема с выбором главного элемента является одной из модификаций метода Гаусса. Идеей является такая перестановка уравнений, чтобы на k-ом шаге исключения ведущим элементом 𝑎𝑖𝑖 оказывался наибольший по модулю элемент k-го столбца. Главной идеей является привести матрицу к треугольному виду, и после найти все коэффициенты.

* Изображение выглядит как текст

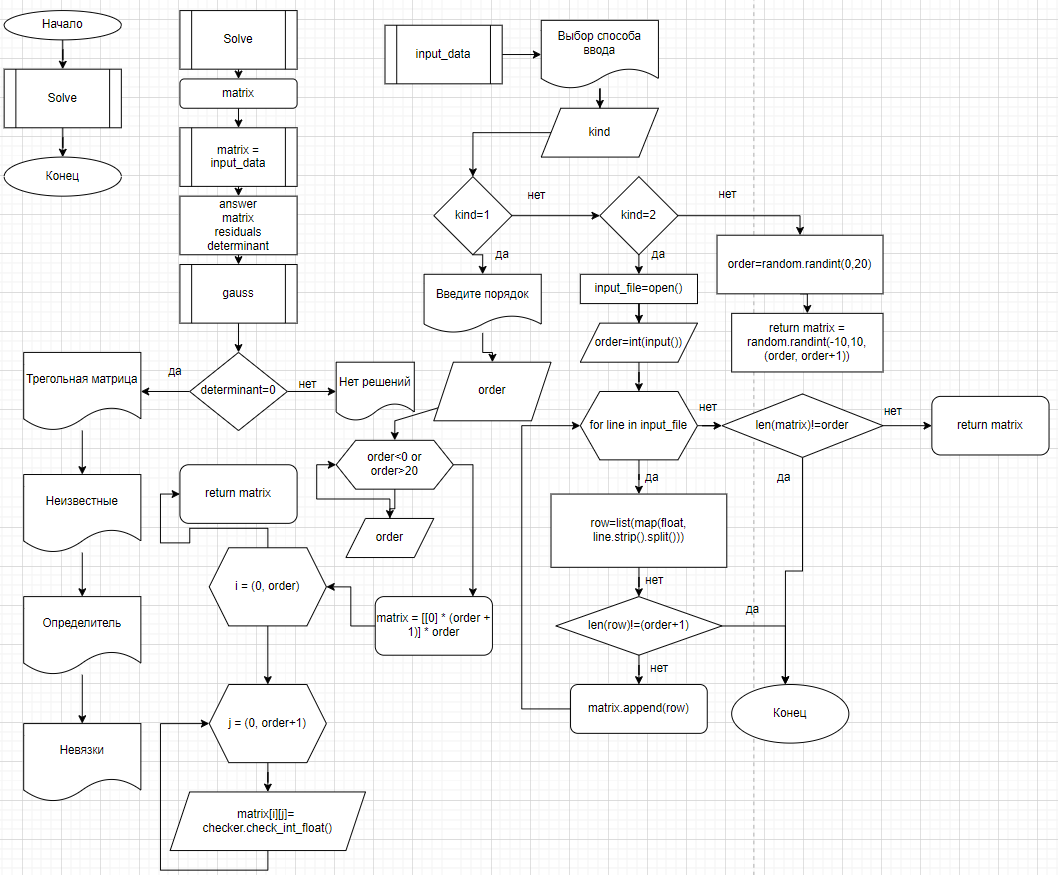
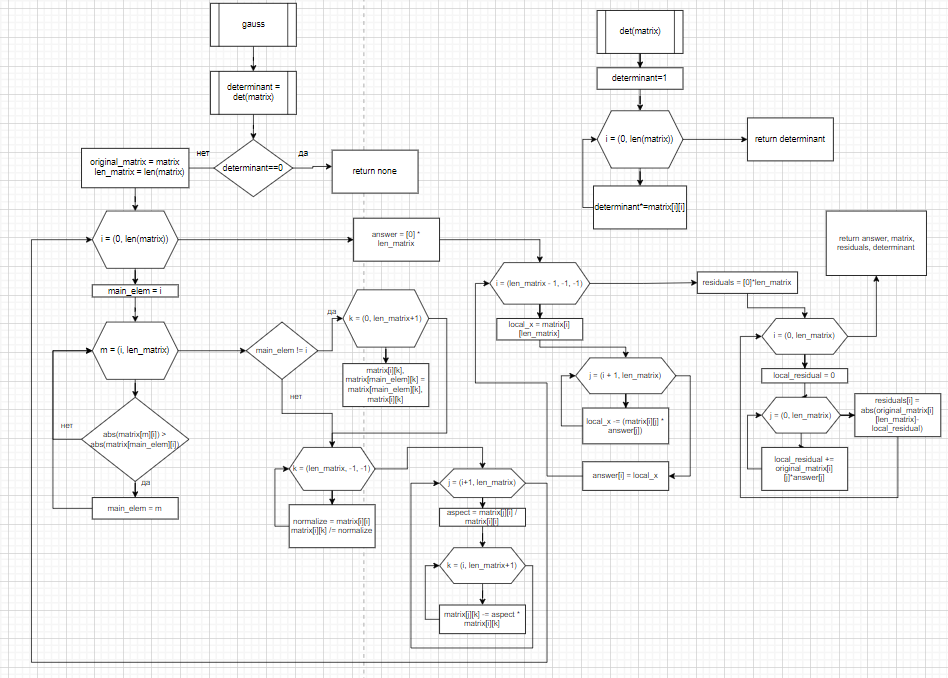
  Автоматически созданное описаниеПрямой ход
  + Выбор ненулевого наибольшего по модулю элемента, и вычисление его множителя(где p – строка, q – столбец):
  + К каждой строке прибавим главную строку, умноженную на соответствующий множитель mi для этой строки. В результате мы получим новую матрицу, у которой q-й столбец состоит из нулей.
  + Продолжаем алгоритм пока не дойдем до последнего члена
* Обратный ход
  + Теперь, зная последний коэффициент и имея треугольную матрицу мы шаг за шагом начинаем подставлять его на строку выше – тем самым находя новые.
  + Продолжаем алгоритм пока не найдем все коэффициенты

Блок схема

**Листинг программы**

import checker

import get\_data  
from numpy import \*  
  
  
def input\_data():  
 print("Здравствуйте, выберите способ ввода данных: \n Введите 1, для ввода данных вручную \n Введите 2, "  
 "для ввода данных из файла \n Введите 3, для случайной генерации чисел")  
 kind = get\_data.get\_int()  
 while (kind > 3 or kind < 1):  
 print("Некорректный ввод")  
 kind = get\_data.get\_int()  
 if (kind == 1):  
 return manual\_input()  
 elif (kind == 2):  
 return file\_input()  
 else:  
 return random\_input()  
  
  
def manual\_input():  
 print("Введите порядок матрицы не более 20")  
 order = get\_data.get\_int()  
 while (order < 0 or order > 20):  
 print("Некорректный ввод")  
 order = get\_data.get\_int()  
 matrix = []  
 print("Начинайте вводить значения по одному (если число дробное - используйте точку):")  
 for i in range(order):  
 row = []  
 for j in range(order + 1):  
 row.append(get\_data.get\_int\_float())  
 matrix.append(row)  
 return matrix  
  
  
def file\_input():  
 try:  
 print("Введить относительный путь к файлу...") # data/input.txt  
 input\_file = open(input(), 'r')  
 order = int(input\_file.readline())  
 matrix = []  
 for line in input\_file:  
 row = list(map(float, line.strip().split()))  
 if (len(row) != (order + 1)):  
 raise ValueError  
 matrix.append(row)  
 if (len(matrix) != order):  
 raise IndexError  
 return matrix  
 except (IndexError):  
 print("Неправильный ввод данных(")  
 except (TypeError, ValueError, IndexError):  
 print("Не удалось считать данные из файла")  
 sys.exit()  
  
  
def random\_input():  
 order = random.randint(0, 10)  
 matrix = random.randint(-10, 10, (order, order + 1))  
 while (det(matrix) == 0):  
 matrix = random.randint(-10, 10, (order, order + 1))  
 return matrix  
  
  
def gauss(matrix):  
 original\_matrix = matrix  
 len\_matrix = len(matrix)  
 for i in range((len(matrix))):  
 # Поиск максимального элемента  
 main\_elem = i  
 for m in range(i, len\_matrix):  
 if abs(matrix[m][i]) > abs(matrix[main\_elem][i]):  
 main\_elem = m  
 # Перестановка строк  
 if main\_elem != i:  
 for k in range(len\_matrix + 1):  
 matrix[i][k], matrix[main\_elem][k] = matrix[main\_elem][k], matrix[i][k]  
  
 # Нормирование  
 for k in range(len\_matrix, -1, -1):  
 try:  
 normalize = matrix[i][i]  
 matrix[i][k] /= normalize  
 except ZeroDivisionError:  
 return 0, 0, 0, 0  
 # Исключение  
 for j in range(i + 1, len\_matrix):  
 aspect = matrix[j][i] / matrix[i][i]  
 for k in range(i, len\_matrix + 1):  
 matrix[j][k] -= aspect \* matrix[i][k]  
  
 determinant = det\_triangle(matrix)  
  
 # Обратный ход  
 answer = [0] \* len\_matrix  
 for i in range(len\_matrix - 1, -1, -1):  
 local\_x = matrix[i][len\_matrix]  
 for j in range(i + 1, len\_matrix):  
 local\_x -= (matrix[i][j] \* answer[j])  
 answer[i] = local\_x  
 return residual(original\_matrix, len\_matrix, answer, determinant, matrix)  
  
  
def residual(original\_matrix, len\_matrix, answer, determinant, matrix):  
 residuals = [0] \* len\_matrix  
 for i in range(len\_matrix):  
 local\_residual = float64(0)  
 for j in range(len\_matrix):  
 local\_residual += original\_matrix[i][j] \* answer[j]  
 residuals[i] = abs(float64(original\_matrix[i][len\_matrix] - local\_residual))  
 return answer, matrix, residuals, determinant  
  
  
def det\_triangle(matrix):  
 determinant = float64(1)  
 for i in range(len(matrix)):  
 determinant \*= matrix[i][i]  
 return determinant  
  
  
def det(matrix):  
 if len(matrix) == 1:  
 return matrix[0][0]  
 else:  
 determinant = float64(0)  
 for i in range(len(matrix)):  
 m = delete(matrix, [i], 1)  
 if i % 2 == 0:  
 determinant += (matrix[0][i] \* det(delete(m, [0], 0)))  
 else:  
 determinant -= (matrix[0][i] \* det(delete(m, [0], 0)))  
 return determinant  
  
  
def solve():  
 matrix = input\_data()  
 print("Исходная матрица:")  
 for row in matrix:  
 print(' '.join([str(elem) for elem in row]))  
  
 answer, matrix, residuals, determinant = gauss(matrix)  
 if answer is None or determinant == 0:  
 print("Система не имеет решений!")  
 else:  
 print("Вычисленная треугольная матрица равна:")  
 for row in matrix:  
 print(' '.join([str(elem) for elem in row]))  
  
 print("\n Вычисленные неизвестные:")  
 for i in range(len(answer)):  
 print("X" + str(i + 1) + " = " + str(answer[i]))  
  
 print("\n Определитель матрицы: " + str(determinant))  
  
 print("\n Вычисленные невязки:")  
 for i in range(len(residuals)):  
 print("Delta" + str(i + 1) + " = " + str(residuals[i]))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 solve()

 **Блок-Схема метода**

Изображение выглядит как текст

Автоматически созданное описание **Пример работы**

Изображение выглядит как текст

Автоматически созданное описание

**Вывод**

В результате выполнения данной лабораторной работой я познакомился с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав на языке программирования Python метод Гаусса с выбором главного элемента по столбцам. В сравнении с итерационными методами данный является более эффективным (особенно при большем количестве элементов), а в сравнении с обычным методом Гаусса, у данного меньше погрешность.