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.

- Прямой ход
  - Выбор ненулевого наибольшего по модулю элемента, и вычисление его множителя(где p – строка, q – столбец): $m_i = -\frac{a_{iq}}{a_{pq}}$
  - 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

Листинг программы

```
import checker
```

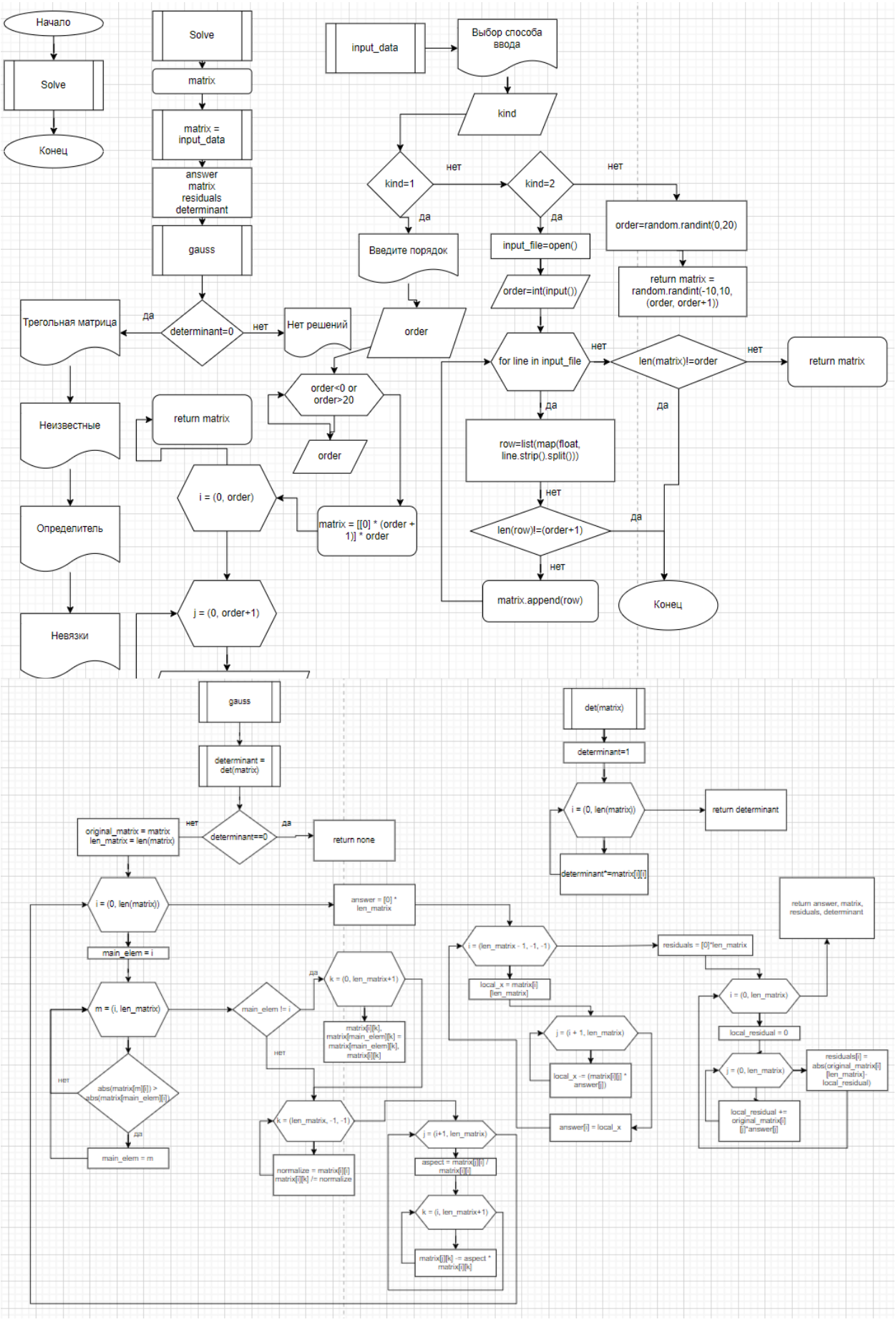
```
import get_data
from numpy import *


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



Пример работы

```
Здравствуйте, выберите способ ввода данных:
 Введите 1, для ввода данных вручную
 Введите 2, для ввода данных из файла
 Введите 3, для случайной генерации чисел
>> 1
Введите порядок матрицы не более 20
>> 3
Начинайте вводить значения по одному (если число дробное - используйте точку):
>> 2
>> 3
>> -1
>> 7
>> 1
>> -1
>> 6
>> 14
>> 6
>> -2
>> 1
>> 11
```

```
Исходная матрица:
2.0 3.0 -1.0 7.0
1.0 -1.0 6.0 14.0
6.0 -2.0 1.0 11.0
Вычисленная треугольная матрица равна:
1.0 -0.3333333333333333 0.16666666666666666 1.8333333333333333
0.0 1.0 -0.36363636363636365 0.9090909090909092
0.0 0.0 1.0 2.284552845528455

 Вычисленные неизвестные:
X1 = 2.032520325203252
X2 = 1.7398373983739837
X3 = 2.284552845528455

 Определитель матрицы: 1.0

 Вычисленные невязки:
Delta1 = 2.220446049250313e-16
Delta2 = 1.1102230246251565e-16
Delta3 = 0.0
```

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.