федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №1
«Системы линейных алгебраических уравнений»

по дисциплине «**Вычислительная математика**»

Автор: Кулаков Н. В.

Факультет: ПИиКТ

Группа: P3230

Преподаватель: Перл О.В.

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2021

1. Описание метода. Расчетные формулы.

Метод последовательных итераций — метод, позволяющий находить значения вектора неизвестных с заданной точностью. В его основе лежит итерационный подход: для нахождения значений на k + 1 шаге необходимо знать значения на k шаге. Также его можно оптимизировать, взяв в качестве неизвестных уже вычисленные значения на данном шаге.

Расчетные формулы:

Пусть дана СЛАУ:

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ \ldots \ + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ \ldots \ + a_{2n}x_n = b_2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \ \ldots \ + a_{nn}x_n = b_n. \end{array}\right\}$$

Для вычисления системы решений необходимо привести СЛАУ к виду: $x = \beta + \alpha x.$

$$\alpha = \begin{bmatrix} \alpha_{11} & \alpha_{12} \ldots \alpha_{1n} \\ \alpha_{21} & \alpha_{22} \ldots \alpha_{2n} \\ \cdots & \cdots \\ \alpha_{n1} & \alpha_{n2} \ldots \alpha_{nn} \end{bmatrix} \quad \text{и} \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \quad \beta_i = \frac{b_i}{a_{ii}}; \quad \alpha_{ij} = -\frac{a_{ij}}{a_{ii}} \quad \text{при } i \neq j$$

Для этого запишем исходное СЛАУ в матричном виде:

$$A = \begin{bmatrix} a_{11} & a_{12} \ldots a_{1n} \\ a_{21} & a_{22} \ldots a_{2n} \\ \cdots & \cdots \\ a_{n1} & a_{n2} \ldots a_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

После того, как привели матрицу к виду, где максимальные элементы стоят на главной диагонали, разделим по строкам матрицу A:

$$[a_{i,1}, a_{i,2}, \ldots, a_{i,n}]$$

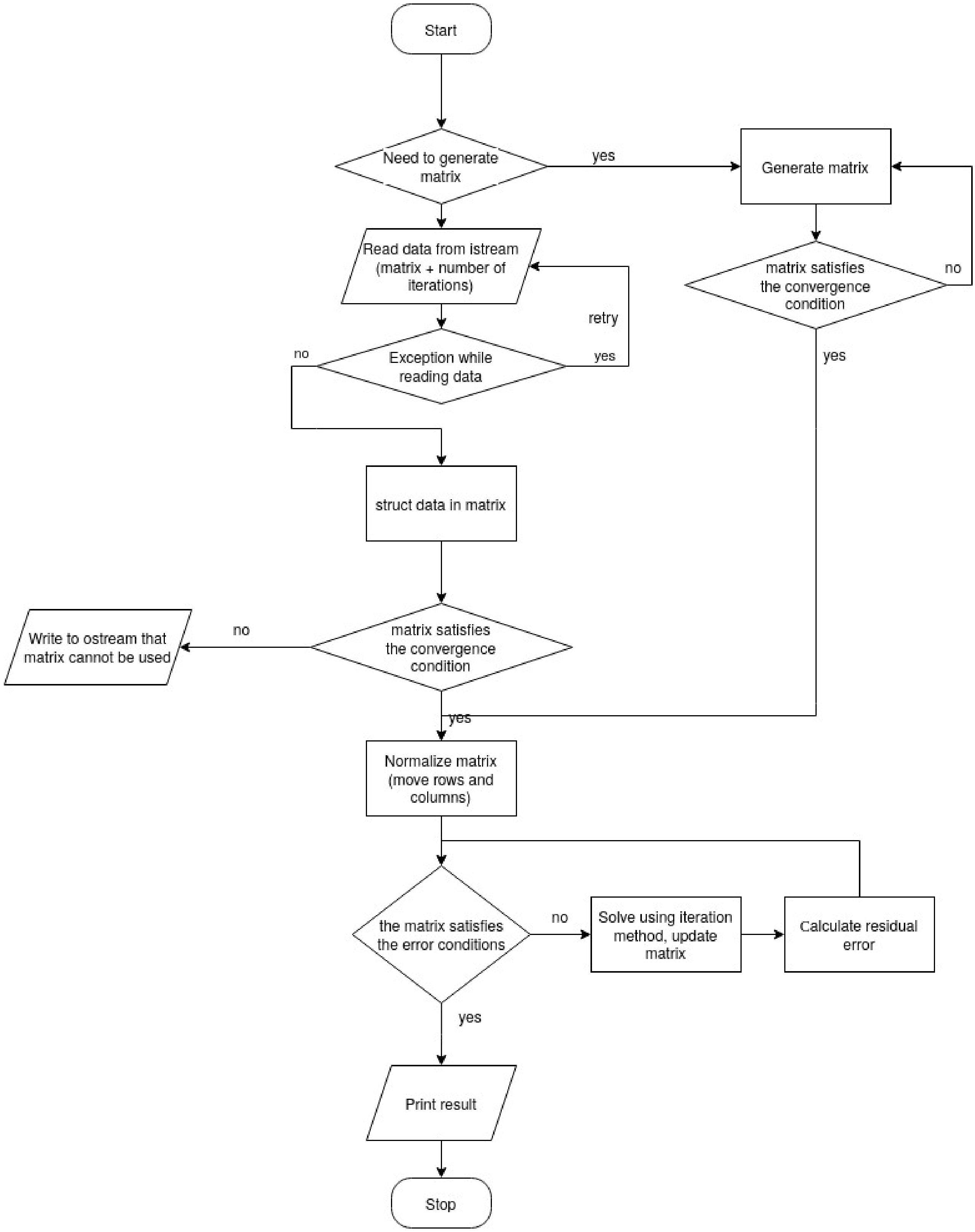
Сохраним в переменную значение $a_{i,i}$ и занулим данный элемент в строке. Затем каждый элемент матрицы умножим на $-a_{i,i}^{-1}$. Затем произведем операцию матричного умножения:

$$x^{(k+1)} = \beta + \alpha x^{(k)}$$

В результате получим значения матрицы неизвестных на k-ом шаге.

Для вычисления столбца погрешностей необходимо найти разность столбца неизвестных на k-ом и k+1 шагах.

2. Блок-схема численного метода.



3. Листинг реализованного численного метода программы.

```
def iterate(matrix_a, matrix_b, matrix_x):
    new_matrix_x = Matrix(matrix_x.rows, matrix_x.columns).init(0)
    for i in range(matrix_a.rows):
```

```
        row = copy(matrix_a[i])
        x_main = - row[i]
        new_matrix_x[i][0] = matrix_b[i][0] / (-x_main)
        row[i] = 0
        for j in range(matrix_a.columns):
            new_matrix_x[i][0] += row[j] / x_main * matrix_x[j][0]
    return new_matrix_x
```

4. **Примеры и результаты работы программы на разных данных.**

```
Enter the matrix by rows:
[1]: 4 1
[2]: 3 9
Enter matrix_b:
Enter the matrix by rows:
[1]: 6
[2]: 1
INFO: matrix_b == matrix_x
Do you want to diagonalize matrix? Y/N: n
Enter difference between iterations: 0.001
Result matrix_x:
[[1.6060434349279835]
[-0.4243130572702332]]
Result matrix_err:
[[-0.000229070216049454]
[-0.0001393175582991013]]
Number of iterations:  9
Elapsed time (ms):  0.2949430054286495
```

```
python3 src/main.py
Enter matrix dimensions: 4
Generate matrix? Y/N: y
Generate pseudo? Y/N: y
Generated matrix:
[[208.82552778217533, 28.147049230385317, 80.31115298657966, 86.35190745127878]
[72.08177957115434, 917.105911435043, 109.44617993796938, 65.41180466648946]
[33.42046055037232, 98.59036655111846, 551.5167048464543, 18.833261117044433]
[22.893657947451935, 101.4870790004234, 16.23905038808694, 327.2210906243729]]
Enter matrix_b:
Enter the matrix by rows:
[1]: 7
[2]: 5
[3]: 1.2
[4]: 6
INFO: matrix_b == matrix_x
Do you want to diagonalize matrix? Y/N: y
Enter difference between iterations: 0.0001
Result matrix_x:
[[0.026831848230920087]
[0.0022592390716969983]
[-0.0003986657520028253]
[0.015769308056537694]]
Result matrix_err:
[[-4.1755772764481836e-05]
[-1.6335060172751898e-05]
[-1.5228908641793477e-05]
[-2.152656338038489e-05]]
Number of iterations:  15
Elapsed time (ms):  1.0004869982367381
```

5. **Вывод.**

В результате выполнения данной лабораторной работы я разобрался как работают итерационные методы, такие как итерационный метод и метод Зейделя. Сложность асимптотическая сложность данного метода равна O(k * n^2), где k — количество итераций, n — объем входных данных.

Итерационные методы в основном применяются для нахождения решений СЛАУ для матриц большого размера с заданной точностью, поскольку сложность точных методов O(n^3), однако для таких матриц необходимо условие сходимости: одно из достаточных условий — норма матрицы альфа < 1, либо при помощи ее следствий.

В корректности работы убедился. Полученные результаты отличались от посчитанных вручную: результаты приблизительно отличались на величину, норма которой (евклидова) была меньше допустимой ошибки между итерациями.