САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДРОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ
Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия
Дисциплина «Вычислительная математика»

# Отчет

По лабораторной работе №1

Вариант 14

Студент

Федоров Евгений Константинович

Преподаватель

Наумова Надежда Александровна

Санкт-Петербург, 2025 г.

# Оглавление

# Цель работы

Цель работы – изучение вычислительного метода решения системы алгебраических линейных уравнений СЛАУ, а также реализация его на ЭВМ на одном из выбранных языков программирования.

## Описание метода

Метод Гаусса-Зейделя является итерационным методом решения СЛАУ. Он модифицирует метод простых итераций за счет использования на k+1 итерации алгоритма уже полученных на этом же шаге значений.

## Расчетные формулы

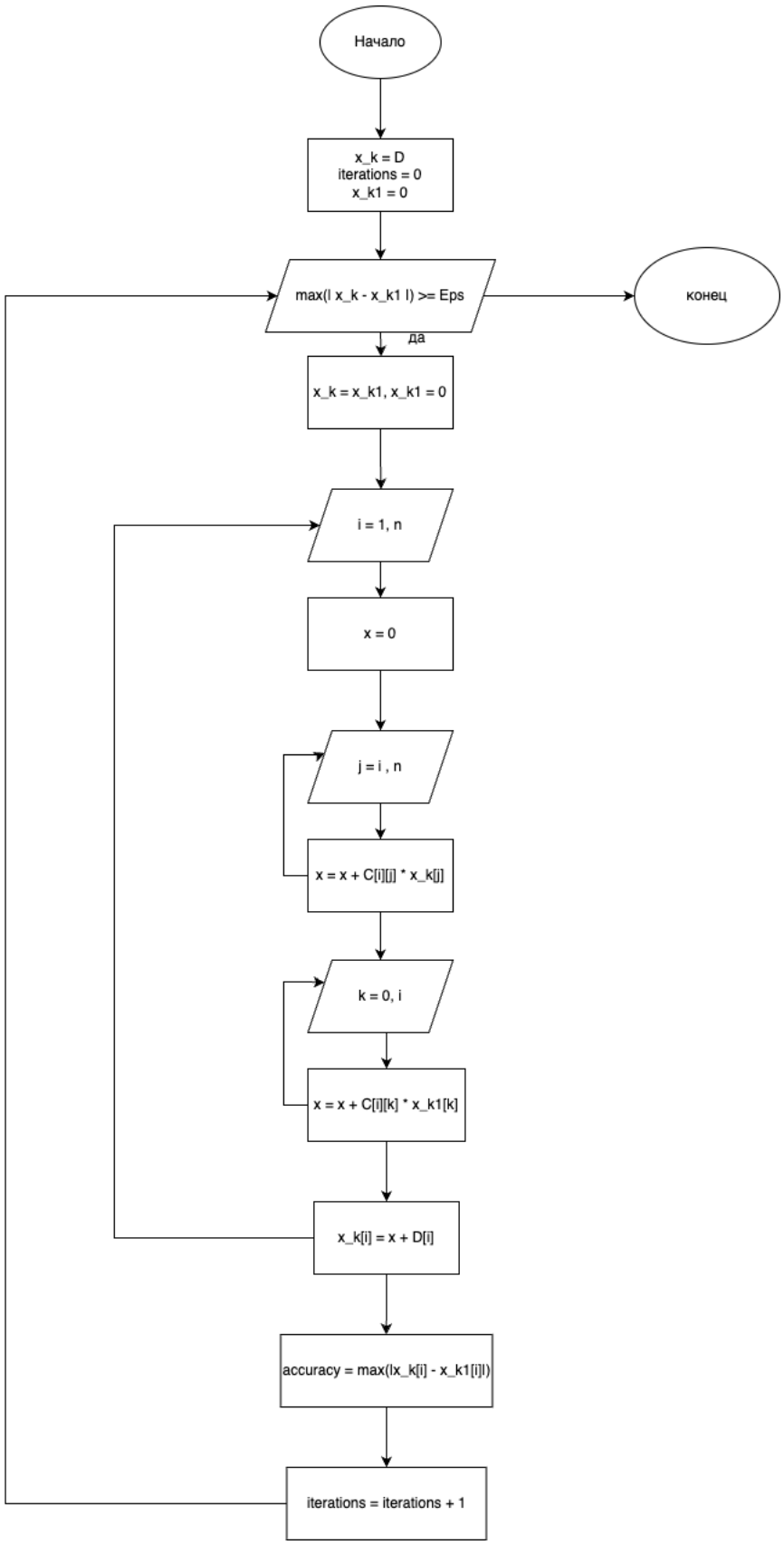
$$x_1^{(k+1)} = c_{11}x_1^{(k)} + c_{12}x_2^{(k)} + \cdots + c_{1n}x_n^{(k)} + d_1$$
$$x_2^{(k+1)} = c_{21}x_1^{(k+1)} + c_{22}x_2^{(k)} + \cdots + c_{2n}x_n^{(k)} + d_2$$
$$x_3^{(k+1)} = c_{31}x_1^{(k+1)} + c_{32}x_2^{(k+1)} + c_{33}x_3^{(k)} \ldots + c_{3n}x_n^{(k)} + d_3$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$x_n^{(k+1)} = c_{n1}x_1^{(k+1)} + c_{n2}x_2^{(k+1)} + \cdots + c_{n\,n-1}x_{n-1}^{(k+1)} + c_{nn}x_n^{(k)} + d_n$$

Рабочая формула метода Гаусса-Зейделя:

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^k \quad i = 1, 2, \ldots, n$$

Итерационный процесс продолжается до тех пор, пока:

$$|x_1^{(k)} - x_1^{(k-1)}| \le \varepsilon, \quad |x_2^{(k)} - x_2^{(k-1)}| \le \varepsilon, \quad |x_3^{(k)} - x_3^{(k-1)}| \le \varepsilon$$


Начало
x_k = D
iterations = 0
x_k1 = 0
max(| x_k - x_k1 |) >= Eps
конец
да
x_k = x_k1, x_k1 = 0
i = 1, n
x = 0
j = i , n
x = x + C[i][j] * x_k[j]
k = 0, i
x = x + C[i][k] * x_k1[k]
x_k[i] = x + D[i]
accuracy = max(|x_k[i] - x_k1[i]|)
iterations = iterations + 1

# Листинг программы

```
def main_calculation(C, D, EPS, n):
    iterations = 0
    x_k = D[:]
    x_k1 = [0] * n

    while iterations == 0 or get_accuracy(x_k, x_k1) >= EPS:
        x_k, x_k1 = x_k1, [0] * n
        for i in range(n):
            x = sum(C[i][j] * x_k[j] for j in range(i, n))
            x += sum(C[i][k] * x_k1[k] for k in range(i))
            x += D[i]
            x_k1[i] = round(x, 6)

        iterations += 1
        printer.printFinalTable(iterations, x_k1,
round(get_accuracy(x_k, x_k1), 5))



    return x_k, iterations
```

Ссылка на git: https://github.com/2BuRy1/Computational-Maths-Lab1

# Примеры и результаты работы программы

## Пример 1

Содержимое файла m:

3
0.01
-3 9 1 5
5 -2 3 10
2 -1 7 7

Вывод программы:

```
Если хотите вводить данные через файл, напишите f, иначе введите любой символ: f
Введите название файла:
m
Изначальная матрица:


x0            x1            x2
-3.0          9.0          1.0          5.0
5.0          -2.0          3.0          10.0
2.0          -1.0          7.0          7.0

Матрица не обладает диагональным преобладанием, попробуем это исправить!!


Матрица с диагональным преобладанием:
x0            x1            x2
5.0          -2.0          3.0          10.0
-3.0          9.0          1.0          5.0
2.0          -1.0          7.0          7.0

Матрица C:
0          0.4          -0.6
0.333          0          -0.111
-0.286          0.143          0
```

```
Матрица D:
2.0
0.5555555555555556
1.0
           Номер итерации           |                x^k                 |     max( | x^(k)_i-x^(k+1) | )
-----------------------------------------------------------------------------------------------------------
                1                   |      [2.0, 1.222222, 0.603175]     |                2.0                 |
                2                   |    [2.126984, 1.197531, 0.563366]  |              0.12698               |
                3                   |    [2.140993, 1.206624, 0.560663]  |              0.01401               |
                4                   |    [2.146252, 1.208677, 0.559453]  |              0.00526               |

ИТОГО

     количество итераций: 4
     ответ с учетом погрешности: [2.140993, 1.206624, 0.560663]
     с округлением: [2.141, 1.207, 0.561]
```

Пример 2

```
Если хотите вводить данные через файл, напишите f, иначе введите любой символ:
Введите размерность матрицы ≤20:
3
Введите точность:
0.01
Введите матрицу:
2 2 10 14
10 1 1 12
2 10 1 13
Изначальная матрица:


x0            x1            x2
2.0          2.0          10.0          14.0
10.0          1.0          1.0          12.0
2.0          10.0          1.0          13.0

Матрица не обладает диагональным преобладанием, попробуем это исправить!!


Матрица с диагональным преобладанием:
x0            x1            x2
10.0          1.0          1.0          12.0
2.0          10.0          1.0          13.0
2.0          2.0          10.0          14.0
```

```
Матрица C:
0         -0.1         -0.1
-0.2          0         -0.1
-0.2         -0.2          0

Матрица D:
1.2
1.3
1.4
           Номер итерации           |                x^k                 |     max( | x^(k)_i-x^(k+1) | )
-----------------------------------------------------------------------------------------------------------
                1                   |          [1.2, 1.06, 0.948]          |                1.2                 |
                2                   |     [0.9992, 1.00536, 0.999088]      |               0.2008               |
                3                   |    [0.999555, 1.00018, 1.000053]     |              0.00518               |

ИТОГО

     количество итераций: 3
     ответ с учетом погрешности: [0.9992, 1.00536, 0.999088]
     с округлением: [0.999, 1.005, 0.999]
```

# Вывод

В ходе выполнения лабораторной работы вспомнил методы решения СЛАУ. Вспомнил свойства матриц, вспомнил про такой прекрасный язык, как Python, заново ознакомился с его синтаксическими особенностями, постарался максимально проникнуться!