

**Министерство науки и высшего образования Российской
Федерации
Федеральное государственное автономное образовательное
учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной
техники**



**Вариант №13
Лабораторная работа №1
по дисциплине
Вычислительная математика**

Выполнил студент группы Р3212
Соколов Анатолий Владимирович
Преподаватель:
Наумова Надежда Александровна

Содержание

1	Задание	1
1.1	Вариант	1
1.2	Цель работы	1
1.3	Описание метода, расчетные формулы	2
1.4	Реализация численного метода	2
1.5	Блок-схема реализованного алгоритма	5
1.6	Ссылка на GitHub с основной реализацией	5
1.7	Примеры и результаты работы программы	6
2	Заключение	7
3	Список литературы	7

1 Задание

В программе реализуемый численный метод решения системы линейных алгебраических уравнений (СЛАУ) должен быть реализован в виде отдельного класса /метода/функции, в который исходные/выходные данные передаются в качестве параметров.

Задавать размерность матрицы ($n < 20$) из файла или с клавиатуры – по выбору конечного пользователя. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

Сформировать не менее 3 файлов (тестов) с различным набором данных.

Программа должна быть протестирована на различных наборах данных, в том числе и некорректных.

1.1 Вариант

Для итерационных методов должно быть реализовано:

- Точность задается с клавиатуры/файла
- Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
- Вывод вектора неизвестных: x_1, x_2, \dots, x_n
- Вывод количества итераций, за которое было найдено решение.
- Вывод вектора погрешностей: $|x_i^K - x_i^{k-1}|$

Метод	№ варианта
Метод Гаусса	1, 3, 5, 8, 21, 24, 26, 28, 31, 39
Метод Гаусса с выбором главного элемента по столбцам	11, 17, 19, 22, 25, 27, 30, 34, 40
Метод простых итераций	2, 4, 6, 7, 10, 13, 15, 23, 32, 36, 38
Метод Гаусса-Зейделя	9, 12, 14, 16, 18, 20, 29, 33, 35, 37

1.2 Цель работы

Рассчитать систему линейных алгебраических уравнений СЛАУ и изучить МПИ.

1.3 Описание метода, расчетные формулы

Описание метода

Итерационные методы это методы последовательных приближений.

Задается некоторое начальное приближение. Далее с помощью определенного алгоритма проводится один цикл вычислений - итерация. В результате итерации находят новое приближение. Итерации проводятся до получения решения с требуемой точностью.

Метод простых итераций — это численный метод решения уравнений, включая алгебраические и дифференциальные уравнения. Основная идея метода состоит в том, чтобы преобразовать исходное уравнение таким образом, чтобы решение могло быть представлено как предел последовательности, получаемой итеративным способом.

Для алгебраических уравнений метод простых итераций можно описать следующим образом:

1. Преобразование уравнения: Исходное уравнение $f(x) = 0$ преобразуется в эквивалентное уравнение вида $x = \phi(x)$, где функция $\phi(x)$ должна быть выбрана таким образом, чтобы последовательность, порождаемая этой функцией, сходилась к корню исходного уравнения.
2. Выбор начального приближения: Выбирается начальное приближение к корню уравнения, обозначаемое как x_0 .
3. Итерационный процесс: На каждом шаге вычисляется следующее приближение к корню по формуле $x_{n+1} = \phi(x_n)$, где n — номер текущей итерации. Этот процесс повторяется до тех пор, пока разность между последовательными приближениями не станет меньше заданной точности ε , т.е. $|x_{n+1} - x_n| < \varepsilon$.
4. Критерий остановки: Итерационный процесс продолжается до тех пор, пока не будет достигнута заданная точность или не будет выполнено максимально допустимое количество итераций.

Расчетные формулы

$$x_i^{k+1} = \frac{b_i}{a_{ii}} - \sum_{j=1, j \neq i}^n \frac{a_{ij}}{a_{ii}} x_j^k, i = 1, 2, \dots, n$$
$$c_{ij} = \begin{cases} 0, & i = j \\ -\frac{a_{ij}}{a_{ii}} \end{cases}$$
$$d_i = \frac{b_i}{a_{ii}}, i = 1, 2, \dots, n$$
$$x^{k+1} = Cx^k + d$$

1.4 Реализация численного метода

```
1 fn shuffle(&mut self) -> (bool, Vec<Vec<f64>>) {
2     let mut biggest = vec![-1; self.n];
3     let mut biggest_set = HashSet::new();
4     let mut found_strict = false;
5
6     for i in 0..self.n {
7         let sum: f64 = self.a[i].iter().sum();
8         for j in 0..self.n {
9             if 2.0 * self.a[i][j] ≥ sum {
10                 if 2.0 * self.a[i][j] > sum {
11                     found_strict = true;
12                 }
13                 biggest[i] = j as isize;
14                 biggest_set.insert(j);
15                 break;
16             }
17         }
18         if biggest[i] == -1 {
19             return (false, vec![vec![0.0]]);
20         }
21     }
22
23     if !found_strict || biggest.len() != biggest_set.len() {
24         return (false, vec![vec![0.0]]);
25     }
26 }
```

```

25     }
26
27     let mut shuffled_a = vec![vec![]; self.n];
28     let mut shuffled_b = vec![0.0; self.n];
29
30     for i in 0..self.n {
31         let index = biggest[i] as usize;
32         shuffled_a[index] = self.a[i].clone();
33         shuffled_b[index] = self.b[i];
34     }
35
36     self.a = shuffled_a.clone();
37     self.b = shuffled_b;
38
39     (true, shuffled_a)
40 }
41
42 fn find_c_and_d(coefficients: Vec<Vec<f64>>>) -> Vec<Vec<f64>> {
43     let n = coefficients.len(); // The number of rows, assuming a square
44     ↪ matrix for coefficients
45     let mut c = vec![vec![0.0; n]; n]; // Initialize C matrix with zeros
46
47     for i in 0..n {
48         // Diagonal element of the current row
49         let diag_elem = coefficients[i][i];
50
51         for j in 0..n {
52             // Check if the current element is not on the diagonal
53             if i != j {
54                 // C matrix is -1 times the original coefficient matrix
55                 ↪ divided by the diagonal element
56                 c[i][j] = -coefficients[i][j] / diag_elem;
57             }
58         }
59         // The diagonal elements of C are set to zero
60         c[i][i] = 0.0;
61     }
62
63     c
64 }
65
66 fn iterate(&mut self) {
67     let mut new_sol = vec![0.0; self.n];
68     for i in 0..self.n {
69         new_sol[i] = self.b[i] / self.a[i][i] - self.sum_sol_row(i);
70         self.sol_acc[i] = (new_sol[i] - self.sol[i]).abs();
71     }
72     self.sol = new_sol;
73     self.sol_iter += 1;
74 }
75
76 pub fn solve(&mut self) -> Json<serde_json::Value> {
77     let mut err = String::new();
78
79     if !self.shuffle().0 {
80         err = String::from("Невозможно
81         ↪ привести к диагональному преобладанию.")
82     }
83
84     self.shuffled_matrix = self.shuffle().1;
85 }

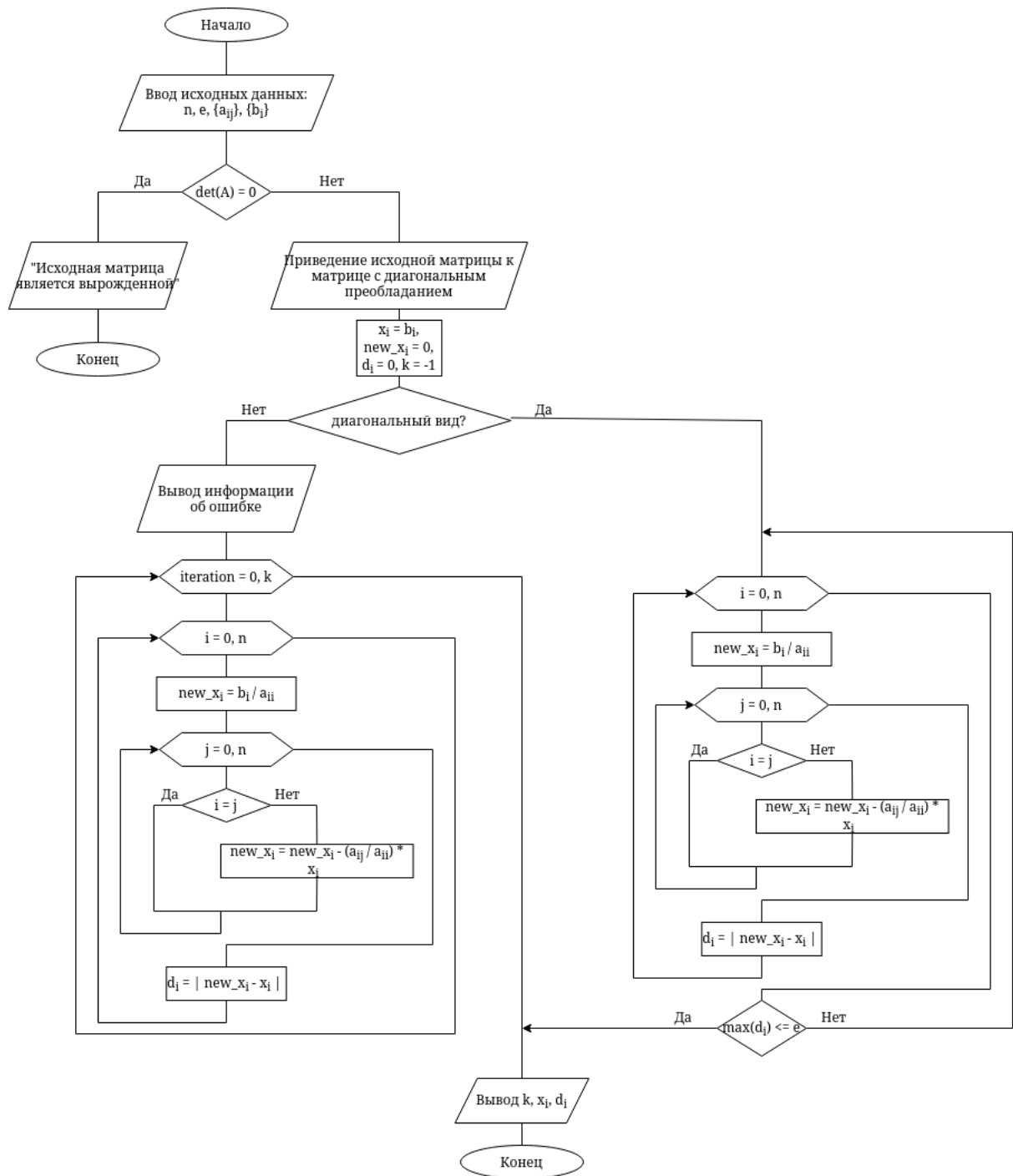
```

```

83     while self.sol_acc.iter().max_by(|a, b| a.total_cmp(b)).unwrap() > &
      ↪ self.acc
84         && self.sol_iter < self.max_iter
85     {
86         self.iterate();
87     }
88
89     if !self.shuffled_matrix.is_empty() {
90         self.c = Matrix::find_c_and_d(self.shuffle().1);
91         return Json(json!({
92             "sol": self.sol,
93             "acc": self.sol_acc,
94             "iter": self.sol_iter,
95             "c": self.c,
96             "mtrx": self.shuffled_matrix,
97             "err": err,
98         }));
99     }
100
101     Json(json!({
102         "sol": self.sol,
103         "acc": self.sol_acc,
104         "iter": self.sol_iter,
105         "mtrx": self.shuffled_matrix,
106         "err": err,
107     }))
108 }

```

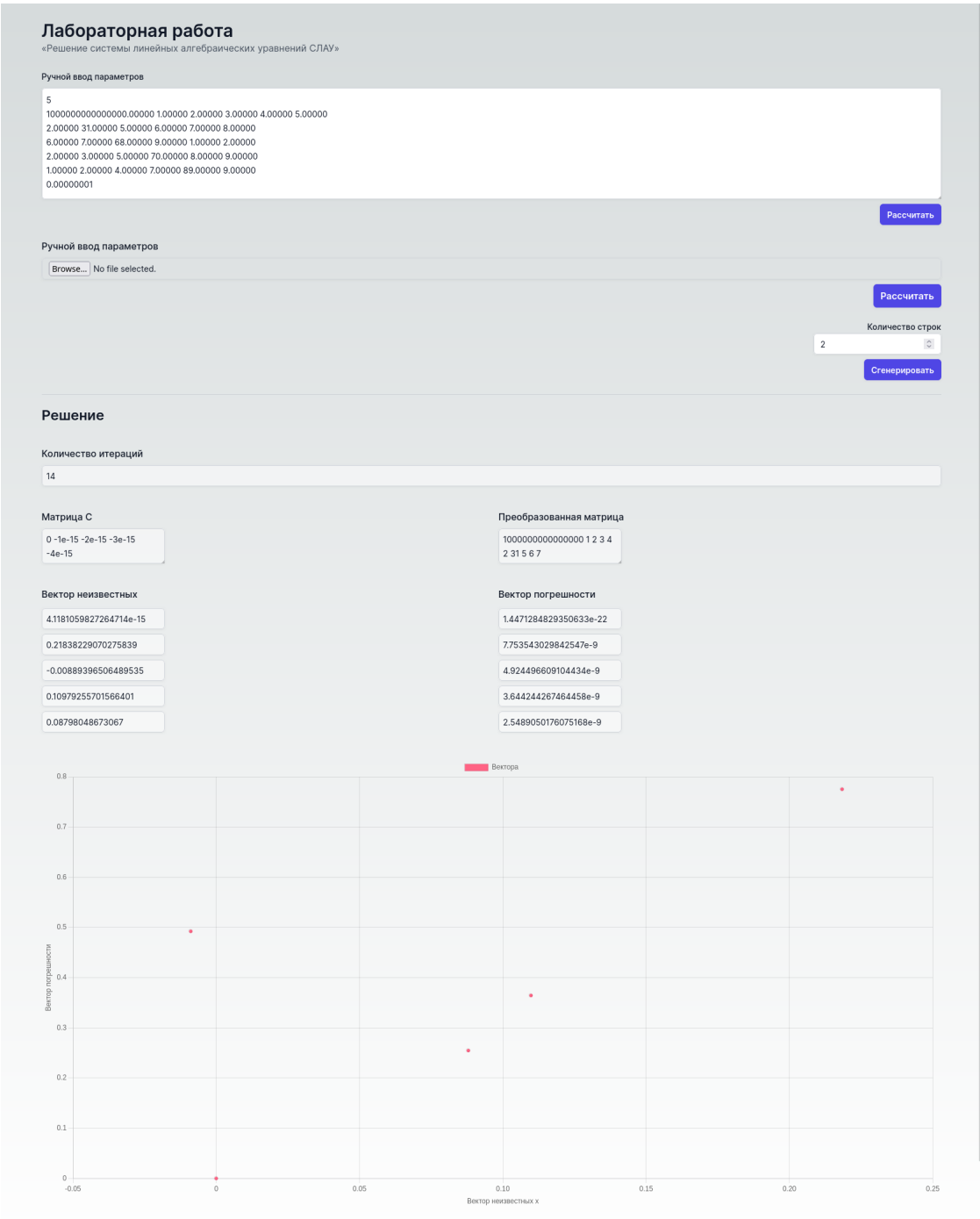
1.5 Блок-схема реализованного алгоритма



1.6 Ссылка на GitHub с основной реализацией

[Github](#)

1.7 Примеры и результаты работы программы



Лабораторная работа

«Решение системы линейных алгебраических уравнений СЛАУ»

Ручной ввод параметров

5 -4 -5 -5 -2 7 7 4 -4 4
19 2 9 3 -4 -2 -6 -12 5
-9 -4 4 -8 -18 -10 -8 -7 -10 6
8 -6 4 -9 -13 2 6 -3 6 7
6 -3 7 5 8 1 4 -10 -8 -18
4 6 -10 3 -7 -2 1 -7 6 0 9
6 0 4 3 -2 -3 -6 0 -1 -10 10
0.0001

Рассчитать

Ручной ввод параметров

Browse... No file selected.

Рассчитать

Количество строк

2

Сгенерировать

Решение

Внимание!
Невозможно привести к диагональному преобладанию.

Количество итераций

100

Матрица C

0

Преобразованная матрица

0

Вектор неизвестных

-2.5852534493149344e+54
6.245122285354683e+54
8.807613597598216e+54
9.372293610621543e+54
1.783136596044143e+55
8.404664905195151e+54
-1.8009960373030533e+55
-5.829813486957118e+54
-6.958701616613679e+54
-2.8074619225183042e+54

Вектор погрешности

3.296978774620167e+54
7.964416651378534e+54
1.1232366828129646e+55
1.1952504351933615e+55
2.274037584582705e+55
1.0718485573483243e+55
2.29668137649126413e+55
7.434772529406436e+54
8.874445766626415e+54
3.580361674620655e+54

Вектор погрешности

2 Заключение

Я познакомился с новым для меня и крайне необыкновенным вычислением СЛАУ на языке rust.

3 Список литературы

- [1] Слайды с лекций (2023). // Кафедра информатики и вычислительной техники – Малышева Татьяна Алексеевна, к.т.н., доцент.