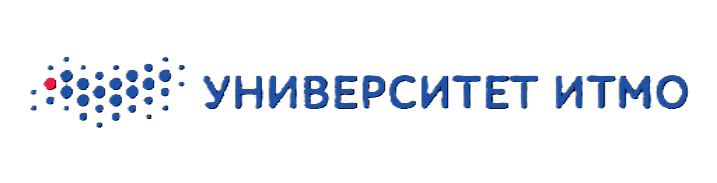
Университет ИТМО

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техники



ЛАБОРАТОРНАЯ РАБОТА №1

РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Вариант №10

Группа: P3211

Студент: Орчиков Даниил Валерьевич

Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург

2022

Оглавление

[Цель работы 2](#_Toc159799039)

[Описание используемого метода 2](#_Toc159799040)

[Расчетные формулы 2](#_Toc159799041)

[Листинг программы 2](#_Toc159799042)

[Примеры и результаты работы программы 5](#_Toc159799043)

[Пример 1 5](#_Toc159799044)

[Пример 2 5](#_Toc159799045)

[Пример 3 6](#_Toc159799046)

Цель работы

Изучить прямые и итерационные методы решения систем линейных алгебраических уравнений, выполнить программную реализацию методов.

# Описание используемого метода

# Расчетные формулы

# Листинг программы

Представлен только код, непосредственно выполняющий вычисления

Весь код можно посмотреть [тут, на GitHub](https://github.com/DaniilOrchikov/Computational_mathematics/tree/master/laba1)

function checkDiagonalDominance(matrix) {  
 // Смотрим в каждой ли строке есть элемент модуль которого больше суммы моделей остальных элементов в строке  
 // Также смотрим возможно ли будет переставить строки так что эти элементы встанут на диагональ  
 let f = new *Array*(*n*).fill(0);  
 for (let i = 0; i < *n*; i++) {  
 let max\_value = 0  
 let ind = -1  
 for (let j = 0; j < *n*; j++) {  
 if (*Math*.abs(matrix[i][j]) > max\_value) {  
 max\_value = *Math*.abs(matrix[i][j])  
 ind = j  
 }  
 }  
 if (ind !== -1) f[ind] = true  
 if (matrix[i].reduce((partialSum, a) => partialSum + *Math*.abs(a), 0) - *Math*.abs(max\_value) > *Math*.abs(max\_value)) {  
 return false; // выполнение достаточного условия невозможно  
 }  
 }  
 return f.reduce((flag, a) => flag && a) // имеет ли смысл пытаться переставлять строки?  
}

function calc() {  
 readMatrix() // тут мы прочитаем со страницы matrix и right\_parts  
 let variable\_matrix = []  
 let variable\_right\_parts = []  
 for (let i = 0 ; i < *n*; i ++){  
 variable\_matrix.push([...*matrix*[i]])  
 }  
 variable\_right\_parts.push(...*right\_parts*)  
 // -----  
 // Смотрим возможно ли диагональное преобладание и если возможно - добиваемся этого с помощью переставления строк  
 // Иначе пытаемся убрать с диагонали все нули так как в противном случае на следующем шаге будет деление на ноль  
 if (!checkDiagonalDominance(variable\_matrix)) {  
 *document*.getElementById("message").innerHTML = "Достигнуть диагонального преобладания невозможно<br>"  
 // -----  
 // Пробуем убрать нули с диагонали  
 for (let i = 0; i < *n*; i++) {  
 let ind = 0  
 for (let j = 0; j < *n*; j++) {  
 if (variable\_matrix[i][j] !== 0 && variable\_matrix[j][i] !== 0) {  
 ind = j  
 break  
 }  
 }  
 variable\_matrix[i] = [variable\_matrix[ind], variable\_matrix[ind] = variable\_matrix[i]][0]  
 variable\_right\_parts[i] = [variable\_right\_parts[ind], variable\_right\_parts[ind] = variable\_right\_parts[i]][0]  
 }  
 let f = true  
 for (let i = 0; i < *n*; i++)  
 if (!variable\_matrix[i][i]) f = false  
 if (!f) { // Не получилось убрать с диагонали все нули :(  
 clear()  
 *document*.getElementById("message").innerHTML += "Матрица несовместна"  
 return  
 }  
 // -----  
 } else {  
 for (let i = 0; i < *n*; i++) {  
 let max\_value = 0  
 let ind = -1  
 for (let j = 0; j < *n*; j++) { // Находим индекс максимального элемента в строке  
 if (*Math*.abs(variable\_matrix[i][j]) >= max\_value) {  
 max\_value = *Math*.abs(variable\_matrix[i][j])  
 ind = j  
 }  
 }  
 // переставляем строку так чтобы максимальный элемент в ней встал на диагональ  
 if (ind !== -1) {  
 variable\_matrix[i] = [variable\_matrix[ind], variable\_matrix[ind] = variable\_matrix[i]][0]  
 variable\_right\_parts[i] = [variable\_right\_parts[ind], variable\_right\_parts[ind] = variable\_right\_parts[i]][0]  
 }  
 if (ind !== i)  
 i = *Math*.min(i, ind) - 1  
 }  
 }  
 // -----  
 // Рисуем матрицу с переставленными строками  
 let m = ""  
 for (let i = 0; i < *n*; i++) {  
 m += "<div class='row'>"  
 for (let j = 0; j < *n*; j++) {  
 m += `<input type='number' class='cell' value='${variable\_matrix[i][j]}' readonly>`  
 }  
 m += `<input type='number' class='cell right\_part' value='${variable\_right\_parts[i]}' readonly></div>`  
 }  
 *document*.getElementById("new\_matrix").innerHTML = m  
 // -----  
 // Выражаем неизвестные x1, x2, ..., xn (делим строку на элемент на диагонали)  
 for (let i = 0; i < *n*; i++) {  
 let mid = variable\_matrix[i][i]  
 for (let j = 0; j < *n*; j++) {  
 variable\_matrix[i][j] /= -mid  
 }  
 variable\_matrix[i][i] = 0  
 variable\_right\_parts[i] /= mid  
 }  
 // -----  
 // Итерационно вычисляем значения  
 let x\_n = []  
 *document*.getElementById("init\_approx").querySelectorAll("\*").forEach(input => {  
 x\_n.push(parseInt(input.value, 10))  
 })  
 let x\_n1 = new *Array*(*n*)  
  
 let v = 0  
 while (true) {  
 v += 1  
 for (let i = 0; i < *n*; i++) {  
 let a = 0  
 for (let j = 0; j < *n*; j++) {  
 a += variable\_matrix[i][j] \* x\_n[j]  
 }  
 a += variable\_right\_parts[i]  
 x\_n1[i] = a  
 }  
 let max\_deviation = 0  
 for (let i = 0; i < *n*; i++) {  
 if (*Math*.abs(x\_n[i] - x\_n1[i]) > max\_deviation)  
 max\_deviation = *Math*.abs(x\_n[i] - x\_n1[i])  
 }  
 if (max\_deviation < *document*.getElementById("accuracy").value)  
 break  
 *document*.getElementById("answer").innerHTML = ""  
 if (v >= *document*.getElementById("maximum\_number\_of\_iterations").value) {  
 *document*.getElementById("answer").innerHTML = "Не удалось достигнуть требуемой точности за отведенное количество итераций<br>"  
 break  
 }  
 x\_n = [...x\_n1]  
 x\_n1 = new *Array*(*n*)  
 }  
 // -----  
 printAnswer(x\_n, x\_n1, v)  
}

function printAnswer(x\_n, x\_n1, v) {  
 let m = ""  
 for (let i = 0; i < *n*; i++)  
 m += `x${i + 1} = ${x\_n1[i]}<br>`  
 m += "Число итераций: " + v + "<br>"  
 m += "Вектор погрешностей: ["  
 for (let i = 0; i < *n*; i++) {  
 m += `${x\_n[i] - x\_n1[i]}`  
 if (i + 1 < *n*) m += ", "  
 }  
 m += "]"  
 *document*.getElementById("answer").innerHTML += m  
}

# Примеры и результаты работы программы

## Пример 1

Размер матрицы - 4  
Точность - 0.00001  
Максимальное число итераций - 200  
Начальные приближения - 1 2 3 4  
1 0 1 4 5  
6 1 1 3 0  
0 9 4 4 6  
1 1 3 0 1

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

## Пример 2

Размер матрицы - 4  
Точность - 0.00001  
Максимальное число итераций - 200  
Максимальное число итераций - 1 1 1 1  
-3 1 2 2 -5  
3 0 -3 3 -1  
-4 2 -5 3 -3  
-3 0 3 2 1

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

## Пример 3

Размер матрицы - 3  
Точность - 0.001  
Максимальное число итераций - 200  
Максимальное число итераций - 0 0 0  
1 2 0 1  
3 4 0 2  
5 6 0 1

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание