МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Поволжский государственный университет телекоммуникаций и информатики»

**Лабораторная работа №3**

**Решение плохо обусловленных систем линейных алгебраических уравнений.** **В-13**

Выполнил:

Селихов А.С.

Проверила:

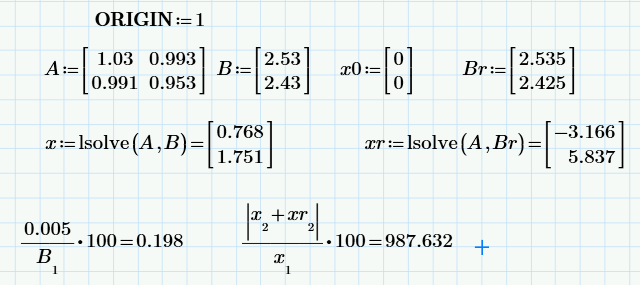
Ахметшина Э.Г.

Самара

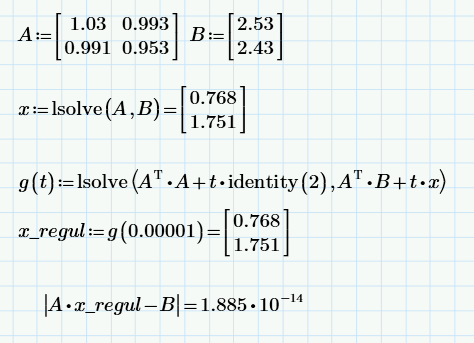
2021

**Цель работы:** Изучить известные методы решения систем плохо обусловленных систем линейных алгебраических уравнений.

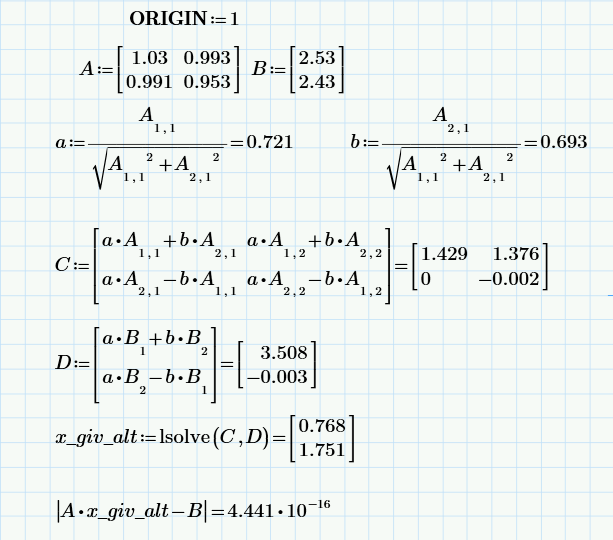
1. Оценка влияния погрешности при возмущении правой части



1. Метод Регуляризации.



1. Метод Гивенса.



1. Код программы

use methods::\*;

use std::{io, vec};

fn main() {

    let mut matrix = Vec::<Vec<f64>>::new();

    let mut free\_elements = Vec::<f64>::new();

    let mut size = usize::MIN;

    println!("Ручной ввод?\n(1)Да;\n(2)Нет");

    let mut choice = String::new();

    io::stdin().read\_line(&mut choice).expect("Failed to read");

    let mut inp = 0;

    match choice.trim().parse::<usize>() {

        Ok(v) => {

            inp = v;

        }

        Err(..) => eprintln!("Не подходящий формат: {}", choice.trim()),

    };

    if inp == 1 {

        println!("Введите размерность для квадратной матрицы");

        let mut input\_text = String::new();

        io::stdin()

            .read\_line(&mut input\_text)

            .expect("Failed to read");

        let trimmed = input\_text.trim();

        match trimmed.parse::<usize>() {

            Ok(v) => {

                size = v;

            }

            Err(..) => println!("Не подходящий формат: {}", trimmed),

        };

        for \_i in 0..size {

            matrix.push(vec![0.0; size]);

            free\_elements.push(0.0);

        }

        println!("Введите коэфы и свободные члены: ");

        for i in 0..size {

            for j in 0..size {

                let mut input\_text = String::new();

                io::stdin()

                    .read\_line(&mut input\_text)

                    .expect("Failed to read");

                let trimmed = input\_text.trim();

                match trimmed.parse::<f64>() {

                    Ok(v) => {

                        matrix[i][j] = v;

                    }

                    Err(..) => println!("Не подходящий формат: {}", trimmed),

                };

            }

            let mut input\_text = String::new();

            io::stdin()

                .read\_line(&mut input\_text)

                .expect("Failed to read");

            let trimmed = input\_text.trim();

            match trimmed.parse::<f64>() {

                Ok(v) => {

                    free\_elements[i] = v;

                }

                Err(..) => println!("Не подходящий формат: {}", trimmed),

            };

        }

    } else {

        matrix = vec![vec![1.03, 0.993], vec![0.991, 0.051]];

        free\_elements = vec![2.53, 2.43];

        size = 2;

    }

    println!("--------------------------------------------------------------------------------");

    println!("Изменение данных в ходе вычислений: ");

    let answer\_for\_givens = givens(matrix.clone(), free\_elements.clone(), size.clone());

    let answer\_for\_regular = regular(matrix.clone(), free\_elements.clone(), size.clone());

    println!("--------------------------------------------------------------------------------");

    println!("Исходные данные: ");

    println!("");

    for i in 0..size {

        for j in 0..size {

            if j != 0 {

                if matrix[i][j] < 0.0 {

                    print!(" - {}x{}", matrix[i][j].abs(), j);

                } else {

                    print!(" + {}x{}", matrix[i][j], j);

                }

            } else {

                print!(" {}x{}", matrix[i][j], j);

            }

        }

        print!(" = {}\n", free\_elements[i]);

    }

    println!("--------------------------------------------------------------------------------");

    println!("Корни системы по Гивенсу:");

    for i in 0..size {

        print!("x{} = {} ", i, answer\_for\_givens[i]);

    }

    println!("Проверка");

    for i in 0..size {

        let mut ans = 0.0;

        for j in 0..size {

            ans += matrix[i][j] \* answer\_for\_givens[j];

        }

        println!("x[{}] = {}", i, ans - free\_elements[i]);

    }

    println!("\n---------------------------------------------------------------------------------");

    println!("Корни системы по Регулярному:");

    for i in 0..size {

        print!("x{} = {} ", i, answer\_for\_regular[i]);

    }

    println!("Проверка");

    for i in 0..size {

        let mut ans = 0.0;

        for j in 0..size {

            ans += matrix[i][j] \* answer\_for\_regular[j];

        }

        println!("x[{}] = {}", i, ans - free\_elements[i]);

    }

}

pub fn givens(matrix: Vec<Vec<f64>>, free\_elements: Vec<f64>, size: usize) -> Vec<f64> {

    let mut answer = vec![0.0; size];

    let mut indent\_matrix = matrix.clone();

    let mut indent\_free\_elements = free\_elements.clone();

    let mut l;

    let mut r;

    for i in 0..(size - 1) {

        for j in (i + 1)..size {

            let mut m = (matrix[i][i].powi(2) \* matrix[j][i].powi(2)).sqrt();

            l = matrix[j][i] / m; // A12

            m = matrix[i][i] / m; //B12

            for k in 0..size {

                r = matrix[i][k];

                indent\_matrix[i][k] = m \* indent\_matrix[i][k] + l \* indent\_matrix[j][k]; // a1j

                indent\_matrix[j][k] = m \* indent\_matrix[j][k] - l \* r;

            }

            r = indent\_free\_elements[i];

            indent\_free\_elements[i] = m \* indent\_free\_elements[i] + l \* indent\_free\_elements[j];

            indent\_free\_elements[j] = m \* indent\_free\_elements[j] - l \* r;

        }

    }

    println!("Данные после вращения: \n");

    for i in 0..size {

        for j in 0..size {

            if j != 0 {

                if indent\_matrix[i][j] < 0.0 {

                    print!(" - {}", indent\_matrix[i][j].abs());

                } else {

                    print!(" + {}", indent\_matrix[i][j]);

                }

            } else {

                print!("{}", indent\_matrix[i][j]);

            }

        }

        print!(" = {}\n", indent\_free\_elements[i]);

    }

    answer = simple\_gaus(indent\_matrix.clone(), indent\_free\_elements.clone());

    answer

}

pub fn regular(matrix: Vec<Vec<f64>>, free\_elements: Vec<f64>, size: usize) -> (usize, Vec<f64>) {

    let mut answer = vec![0.0; size];

    let mut a1 = vec![vec![0.0; size]; size];

    let mut b1 = vec![0.0; size];

    let mut x0 = simple\_gaus(matrix.clone(), free\_elements.clone());

    for i in 0..size {

        for k in 0..size {

            let mut s = 0.0;

            for j in 0..size {

                s += matrix[j][i] \* matrix[j][k];

            }

            a1[i][k] = s;

        }

    }

    for i in 0..size {

        let mut s = 0.0;

        for j in 0..size {

            s += matrix[j][i] \* free\_elements[j]

        }

        b1[i] = s;

    }

    let mut alfa = 0.0;

    let mut b2 = vec![EPS; size];

    let mut max = f64::MAX;

    let mut count = 0;

    while max > EPS {

        alfa += 0.00000001;

        let mut a2 = a1.clone();

        for i in 0..size {

            a2[i][i] = a1[i][i] + alfa;

            b2[i] = b1[i] + alfa \* x0[i];

        }

        a1 = a2.clone();

        b1 = b2.clone();

        b2 = gaus(a2.clone(), b2.clone());

        a2 = a1.clone();

        answer = b2.clone();

        x0 = answer.clone();

        b2 = b1.clone();

        b2 = gaus(a2, b2);

        max = (b2[0] - answer[0]).abs();

        for i in 1..size {

            if (b2[i] - answer[i]).abs() > max {

                max = (b2[i] - answer[i]).abs();

            }

        }

        count += 1;

    }

    (count, answer)

}

