Самарский национальный исследовательский университет

имени академика С.П. Королева

Естественнонаучный институт

Механико-математический факультет

Кафедра информатики и вычислительной математики

Отчет по лабораторной работе №1 «Вычисление машинного эпсилон. Решение систем линейных алгебраических уравнений с треугольными матрицами» по дисциплине «Методы вычислений»

Выполнил:

студент группы 4345-020303D

Р.Е. Ильдияров

Проверил:

доцент В.П. Сироченко

Самара-2024

**1. ВЫЧИСЛЕНИЕ МАШИННОГО ЭПСИЛОН**

**1.1 Постановка задачи**

Реализовать алгоритм вычисления машинного эпсилон

**1.2 Краткое описание численного метода**

Машинное эпсилон представляет собой наименьшее положительное число ε, такое что в представлении с плавающей точкой. Для его вычисления используется метод последовательного уменьшения значения до достижения точности меньше, чем машинное эпсилон.

Асимптотическая сложность

Количество действий для вычисления элипсоида: 212

**1.3 Листинг программы**

Среда разработки: RustRover 2023.3 EAP, Build #RR-233.14015.152, built on February 16, 2024.

Язык программирования: Rust (rustc 1.75.0 (82e1608df 2023-12-21)

|  |
| --- |
| mod elips;  fn main() -> std::io::Result<()> {  let elips\_res\_32 = elips::elipsoid32(1.);  let elips\_res\_64 = elips::elipsoid64(1.);  println!("Accuracy at 32 bit: {}", elips\_res\_32);  println!("Accuracy at 64 bit: {}", elips\_res\_64);  Ok(())  } |

Листинг 1 - src/main.rs

|  |
| --- |
| pub fn elipsoid64(eps:f64) ->f64{  let mut new\_eps:f64 = eps;  let mut new\_eps1:f64 = 0.0;   loop{  new\_eps /= 2.0;  new\_eps1 = new\_eps + 1.0;  if new\_eps1 <= 1.0{  break;  }  }  new\_eps }  pub fn elipsoid32(eps:f32) ->f32{  let mut new\_eps:f32 = eps;  let mut new\_eps1:f32 = 0.0;   loop{  new\_eps /= 2.0;  new\_eps1 = new\_eps + 1.0;  if new\_eps1 <= 1.0  {  break;  }  }  new\_eps } |

Листинг 2 - src/elips.rs

**1.4 Результаты расчетов**

Таблица 1- Расчет элипсоида для одинарной точности

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1.0 | 0.000000059604645 |

Таблица 2 - Расчет элипсоида для двойной точности

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1.0 | 0.00000000000000011102230246251565 |

**2. Решение систем линейных алгебраических уравнений с нижней треугольной матрицей**

**2.1 Постановка задачи**

Пусть дана система линейных алгебраических уравнений вида:

Нужно найти решение системы методом подстановки.

**2.2 Краткое описание численного метода**

Метод обратной подстановки применяется в численном анализе для решения систем линейных уравнений с треугольными матрицами. Этот метод часто используется в задачах, где матрицы обладают специальными структурами, такими как нижняя треугольность, так как он обычно эффективнее общих методов решения систем.

Используя свойства нижней треугольной матрицы, метод обратной подстановки решает систему уравнений, начиная с последнего уравнения и последовательно находя неизвестные от последней к первой.

Этапы алгоритма:

1. Начнем с последнего уравнения;
2. Решим уравнение для первой неизвестной:
3. Подставим найденное значение ​ в предпоследнее уравнение и найдем следующую неизвестную:
4. Продолжим этот процесс до тех пор, пока не найдем все неизвестные .

Асимптотическая сложность:

**2.3 Листинг программы**

Среда разработки: RustRover 2023.3 EAP, Build #RR-233.14015.152, built on February 16, 2024.

Язык программирования: Rust (rustc 1.75.0 (82e1608df 2023-12-21)

|  |
| --- |
| pub fn lower\_matrix(a: Vec<Vec<f32>>, b: Vec<f32>) ->Vec<f32> {  let N:usize = b.len();   let mut x=vec![0.0; N];  x[0] = b[0]/a[0][0];  let mut summ:f32;   for i in (1..N){  summ = 0.0;  for j in (0..i){  summ += x[j] \* a[i][j];  }  x[i] = (b[i]-summ)/a[i][i];  }   return x; } |

Листинг 3 - src/matrix\_count.rs

|  |
| --- |
| use std::io::prelude::\*; use std::io::BufReader; use std::fs::File;  mod matrix\_count;  fn read\_matrix\_file(filename: &str) ->(i32, Vec<Vec<f32>>, Vec<f32>){  let file = File::open(filename).expect("File error");  let mut reader = BufReader::new(file);  let mut line = String::new();   // get matrix size  reader.read\_line(&mut line).expect("Line error");  let n = line.trim().parse::<i32>().unwrap();  line.clear();   //get matrix from file line  let mut a:Vec<Vec<f32>> = Vec::new();  for i in (0..n) {  reader.read\_line(&mut line).expect("Error when parse matrix");  let k:Vec<f32> = line.split(" ").map(|c| c.trim().parse::<f32>().expect("Is not of num")).collect();  a.push(k);  line.clear();  }   // get answer vector  reader.read\_line(&mut line).expect("Error when parse \"b\"");  let b:Vec<f32> = line.split(" ").map(|c| c.trim().parse::<f32>().expect("Is not of num")).collect();  line.clear();   (n, a, b) }  fn main() -> std::io::Result<()> {  //примеры решения матричных уравнений  //Нижнетреугольная матрица  //1  let (\_, a, b) = read\_matrix\_file("cache/matrix/down/1.txt");  let res = matrix\_count::lower\_matrix(a, b);  println!("Решение нижнетреугольной матрицы 1:");  println!("{:?}", res);   //2  let (\_, a, b) = read\_matrix\_file("cache/matrix/down/2.txt");  let res = matrix\_count::lower\_matrix(a, b);  println!("Решение нижнетреугольной матрицы 2:");  println!("{:?}", res);    Ok(()) } |

Листинг 4 - src/main.rs

**2.4 Результаты расчетов**

Таблица 3 - результаты тестирования решения нижней треугольной матрицы

|  |  |  |
| --- | --- | --- |
| Входные данные | Результат | Ожидаемы результат |
| 3 1 0 0 2 3 0 4 5 6 1 8 32 | [1.0, 2.0, 3.0] | [1.0, 2.0, 3.0] |
| 5 8 0 0 0 0 4 5 0 0 0 1 9 1.2 0 0 7.8 9 12.32 23.5 0 4 7.9 1.1 0.6 5 18.3 34 98 1.01 4 | [2.2875,4.9700003,42.485413, -24.892885,-15.242246] | [2.2875,4.9700003,42.485413, -24.892885,-15.242246] |

**3. Решение систем линейных алгебраических уравнений с верхней треугольной матрицей**

**3.1 Постановка задачи**

Пусть дана система линейных алгебраических уравнений вида:

Нужно найти решение системы методом подстановки.

**3.2 Краткое описание численного метода**

Метод обратной подстановки применяется в численном анализе для решения систем линейных уравнений с треугольными матрицами. Этот метод часто используется в задачах, где матрицы обладают специальными структурами, такими как нижняя треугольность, так как он обычно эффективнее общих методов решения систем.

Используя свойства нижней треугольной матрицы, метод обратной подстановки решает систему уравнений, начиная с последнего уравнения и последовательно находя неизвестные от последней к первой.

Этапы алгоритма:

1. Начнем с последнего уравнения;
2. Решим уравнение для первой неизвестной:
3. Подставим найденное значение ​ в предпоследнее уравнение и найдем следующую неизвестную:
4. Продолжим этот процесс до тех пор, пока не найдем все неизвестные .

Асимптотическая сложность:

**3.3 Листинг программы**

Среда разработки: RustRover 2023.3 EAP, Build #RR-233.14015.152, built on February 16, 2024.

Язык программирования: Rust (rustc 1.75.0 (82e1608df 2023-12-21)

|  |
| --- |
| pub fn upper\_matrix(a: Vec<Vec<f32>>, b: Vec<f32>) ->Vec<f32> {  let N:usize = b.len();   let mut x=vec![0.0; N];  x[N-1] = b[N-1]/a[N-1][N-1];  let mut summ:f32;   for i in (0..N-1).rev(){  summ = 0.0;  for j in (i+1..N){  summ += x[j] \* a[i][j];  }  x[i] = (b[i]-summ)/a[i][i];  }   return x; } |

Листинг 5 - src/matrix\_count.rs

|  |
| --- |
| use std::io::prelude::\*; use std::io::BufReader; use std::fs::File;  mod matrix\_count;  fn read\_matrix\_file(filename: &str) ->(i32, Vec<Vec<f32>>, Vec<f32>){  let file = File::open(filename).expect("File error");  let mut reader = BufReader::new(file);  let mut line = String::new();   // get matrix size  reader.read\_line(&mut line).expect("Line error");  let n = line.trim().parse::<i32>().unwrap();  line.clear();   //get matrix from file line  let mut a:Vec<Vec<f32>> = Vec::new();  for i in (0..n) {  reader.read\_line(&mut line).expect("Error when parse matrix");  let k:Vec<f32> = line.split(" ").map(|c| c.trim().parse::<f32>().expect("Is not of num")).collect();  a.push(k);  line.clear();  }   // get answer vector  reader.read\_line(&mut line).expect("Error when parse \"b\"");  let b:Vec<f32> = line.split(" ").map(|c| c.trim().parse::<f32>().expect("Is not of num")).collect();  line.clear();   (n, a, b) }  fn main() -> std::io::Result<()> {  //верхне треугольная матрица  let (\_, a, b) = read\_matrix\_file("cache/matrix/upper/1.txt");  let res = matrix\_count::upper\_matrix(a, b);  println!("Решение треугольной матрицы 1:");  println!("{:?}", res);   Ok(()) } |

Листинг 6 - src/main.rs

**3.4 Результаты расчетов**

Таблица 4 - результаты тестирования решения нижней треугольной матрицы

|  |  |  |
| --- | --- | --- |
| Входные данные | Результат | Ожидаемы результат |
| 3 1 0 0 2 3 0 4 5 6 1 8 32 | [1.0, 2.0, 3.0] | [1.0, 2.0, 3.0] |