|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическая работа № 1 | | |
| по дисциплине «Численные методы» | | |
| **ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ** | | |
|  | | |
|  | Бригада | Румянцев Артём |
|  | Уваров Артём |
| Группа ПМ-23 |  |
| Вариант 6 |  |
|  |  |
|  |  |
| Преподаватели | задорожный александр геннадьевич |
|  |  |
| Новосибирск, 2024 | | |

**(Черновой отчёт)**

1. **Цель работы**

Разработать программу решения СЛАУ методом с - разложением и с хранением матрицы в ленточном формате. Исследовать накопление погрешности и ее зависимость от числа обусловленности. Сравнить реализованный метод по точности получаемого решения и количеству действий с методом Гаусса.

1. **Теоретическая часть**

Пусть дана система линейных алгебраических уравненийcсимметричной матрицей:

. (1.1)

Предположим, что нам удалось разложить матрицу:

(1.2)

Подставляя (1.2) в (1.1), получаем:

. (1.3)

Обозначим:

(1.4)

тогда подставляя (1.4) в (1.3), получим:

. (1.5)

Таким образом, решение системы (1.1) сводится к четырем основным этапам:

1. из элементов матрицы найти элементы матриц и;
2. решить систему (1.5) с нижнетреугольной матрицей (прямой ход);
3. решить систему (1.4) с верхнетреугольной матрицей (обратный ход).

Рассмотрим алгоритм получения – разложения. Матрицу будем искать в следующем виде:

(1.6)

Учитывая равенство (1.2) и умножая последовательно строки матрицы на столбцы матрицы, получаем систему, состоящую из уравнений с неизвестными  
 и ( – размерность СЛАУ):

(1.7)

*…*

Решая систему (1.8), можно получить общие формулы для нахождения элементов матрицы :

(1.8)

1. **Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Входные данные** | **Выходные данные** | **Назначение** |
| 1 | size: 4 5  al: 0 0  0 1  3 5  2 4  di: 10  12  23  17 | al: 0.000000  0.000000  0.000000  0.316228  0.948683  1.362462  0.579771  0.713463  di: 3.162278  3.449638  4.499300  4.019308 | Проверка работы -разложения  Изображение выглядит как текст, снимок экрана, Шрифт, линия  Автоматически созданное описание |
| 2 | size: 4 5  al: 0 0  0 1  3 5  2 4  di: 10  12  23  17  b: 38  105  239  176 | y: 12.016655  29.336414  41.702102  32.154463 | Проверка работы поиска вектора y ()  Изображение выглядит как текст, снимок экрана, Шрифт, число  Автоматически созданное описание |
| 3 | size: 4 5  al: 0 0  0 1  3 5  2 4  di: 10  12  23  17  b: 38  105  239  176 | x: 1.000000  4.000000  8.000000  8.000000 | Проверка работы поиска вектора x ()  Изображение выглядит как текст, снимок экрана, Шрифт, число  Автоматически созданное описание |
| 4 | size: 4 5  al: 0 0  0 1  3 5  2 4  di: 10  12  23  17  b: 38  105  239  176 | x: 1.000000  4.000000  8.000000  8.000000 | Проверка работы всей программы (решение СЛАУ)  Изображение выглядит как текст, снимок экрана, Шрифт, дизайн  Автоматически созданное описание |

1. **Исследование на матрицах, число обусловленности которых регулируется за счёт изменения диагонального преобладания:**
2. **Расчет количества действий реализованного метода решения:**

Количество действий, необходимое для получения элементов матрицы U:

Количество действий, необходимое для получения элементов матрицы L:

Прямой ход:

Обратный ход:

Результат:

1. Те**кст программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

#include <iostream>

#include <math.h>

**using** **namespace** std**;**

**typedef** double real**;**

#define realin "%lf"

class matrix **{**

public**:**

int n**,** m**;**

real**\*\*** al**,** **\*\*** au**;**

real**\*\*** l**,** **\*\*** u**;**

real**\*** di**,** **\***b**,** **\*** x**,** **\*** y**;**

//matrix() {};

void input**(**FILE**\*** inmat**,** FILE**\*** invec**)** **{**

//n = 4;

//m = 2;

fscanf**(**inmat**,** "%d %d"**,** **&**n**,** **&**m**);**

allocateMemory**();**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

**for** **(**int j **=** 0**;** j **<** m**;** **++**j**)** **{**

fscanf**(**inmat**,** realin**,** **&**al**[**i**][**j**]);**

l**[**i**][**j**]** **=** al**[**i**][**j**];**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

**for** **(**int j **=** 0**;** j **<** m**;** **++**j**)** **{**

fscanf**(**inmat**,** realin**,** **&**au**[**i**][**j**]);**

u**[**i**][**j**]** **=** au**[**i**][**j**];**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)**

fscanf**(**inmat**,** realin**,** **&**di**[**i**]);**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)**

fscanf**(**invec**,** realin**,** **&**b**[**i**]);**

fclose**(**inmat**);**

fclose**(**invec**);**

**}**

void allocateMemory**();**

void calcLU**();**

void output**(**FILE**\*** res**);**

void getY**();**

void getX**();**

void Gilbert**();**

**};**

void matrix**::**calcLU**()** **{**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** n**;** j**++)** **{**

real sumL **=** 0**;**

real sumU **=** 0**;**

real sumD **=** 0**;**

**for** **(**int ki **=** 0**;** ki **<=** i **-** 2 **;** ki**++)**

**{**

sumU **+=** al**[**i**][**ki **+** m **-** i**]** **\*** au**[**j**][**ki **+** m **-** j**];**

**}**

**for** **(**int kj **=** 0**;** kj **<=** j **-** 2**;** kj**++)**

**{**

sumL **+=** al**[**i**][**kj **+** m **-** i**]** **\*** au**[**j**][**kj **+** m **-** j**];**

**}**

printf**(**"sumU: %f %d %d "**,** sumU**,** i**,** j**);**

printf**(**"sumL: %f %d %d\n"**,** sumL**,** i**,** j**);**

u**[**i**][**j**]** **=** al**[**i**][**j **+** m **-** i**]** **-** sumU**;**

l**[**i**][**j**]** **=** **(**au**[**j**][**i **+** m **-** j**]** **-** sumL**)** **/** di**[**j**];**

//sumD += al[i][j] \* au[k];

**}**

**}**

**}**

void matrix**::**allocateMemory**()** **{**

al **=** **new** real **\*** **[**n**];**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

al**[**i**]** **=** **new** real**[**m**];**

**}**

au **=** **new** real **\*** **[**n**];**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

au**[**i**]** **=** **new** real**[**m**];**

**}**

l **=** **new** real **\*** **[**n**];**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

l**[**i**]** **=** **new** real**[**m**];**

**}**

u **=** **new** real **\*** **[**n**];**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

u**[**i**]** **=** **new** real**[**m**];**

**}**

di **=** **new** real**[**n**];**

b **=** **new** real**[**n**];**

x **=** b**;**

y **=** b**;**

**}**

int main**()** **{**

FILE**\*** inmat**,** **\*** invec**;**

matrix matrix**;**

fopen\_s**(&**inmat**,** "inmat.txt"**,** "r"**);**

fopen\_s**(&**invec**,** "invec.txt"**,** "r"**);**

matrix**.**input**(**inmat**,** invec**);**

matrix**.**calcLU**();**

**for** **(**int i **=** 0**;** i **<** matrix**.**n**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** matrix**.**n**;** j**++)** **{**

matrix**.**al**[**i**][**j **+** matrix**.**m **-** i**]** **!=** **NULL** **?** printf**(**"%f "**,** matrix**.**al**[**i**][**j **+** matrix**.**m **-** i**])** **:** printf**(**"%d "**,** 0**);**

**}**

printf**(**"\n"**);**

**}**

**for** **(**int i **=** 0**;** i **<** matrix**.**n**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** matrix**.**n**;** j**++)** **{**

matrix**.**au**[**j**][**i **+** matrix**.**m **-** j**]** **!=** **NULL** **?** printf**(**"%f "**,** matrix**.**au**[**j**][**i **+** matrix**.**m **-** j**])** **:** printf**(**"%d "**,** 0**);**

**}**

printf**(**"\n"**);**

**}**

**}**