 **МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ** 

**имени М.В.Ломоносова**

**Факультет вычислительной математики и кибернетики**

**Практическое задание № 2 по учебному курсу**

**«Численные методы линейной алгебры»**

**Метод Чебышева с оптимальным набором итерационных параметров**

**Матрица №2**

**ОТЧЁТ**

**о выполненном задании**

студента **301** учебной группы факультета ВМК МГУ

**Матвеевой Софьи Вячеславовны**

гор. Москва 2023г.

# Постановка задачи

Требуется методом Чебышева с оптимальным набором итерационных параметров

приближенно решить систему линейных алгебраических уравнений

x + Ax = F

с симметричной положительно определенной матрицей. Элементы матрицы являются вещественными числами, расположенными на отрезке Матрица предоставляется в виде файла в формате csv.

С помощью теоремы Гершгорина оценить спектр матрицы системы уравнений.

Подобрать наименьший показатель степени, при котором погрешность решения на последней итерации в среднеквадратической норме не превосходит погрешность прямого метода(метода Холецкого). Построить график среднеквадратической нормы погрешности решения как функции номера итерации метода Чебышева.

Для проверки метода требуется случайно сгенерировать вектор-столбец ***x*** с равномерно распределенными на отрезке [-1; 1] компонентами, вычислить правую часть по формуле .

**Описание метода решения задачи :**

1. *Оценка спектра с помощью теоремы Гершгорина.*

*Теорема Гершгорина: Каждое характеристическое число матрицы A всегда расположено в одном из кругов:*

*Так как матрица вещественна и является матрицей с диагональным преобладанием , то все собственные числа являются вещественными. Тогда для оценки спектра матрицы A найдем:*

1. *Решение СЛАУ с помощью итерационного метода Чебышева*

*Обозначим A = Ax + x.*

*Для решения* Ax = f методом Чебышева проведём последовательные приближения вектора x. По заданию , (n+1)-е приближение найдём по формуле:

*Выразим:*

*, ,*

*Где m – число итераций,*

*- итерационный параметр, который находится следующим образом:*

*Далее для m являющихся степенью двойки:*

**Листинг программы**

Программa написанa на языке «C++14».

В программе были использованы следующие библиотеки и using:

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <random>

#include <chrono>

#include <cmath>

#include <fstream>

using namespace **std**;

using namespace **chrono**;

В программе используются следующие псевдонимы для типов данных:

typedef **vector**<**vector**<double>> **matrix**;

typedef **vector**<double> **vec**;

**matrix** – тип матриц вещественных чисел

**vec** – тип векторов вещественных чисел

Функция, которая производит умножение матрицы на вектор-столбец:

**vec** **MulMV**(const **matrix** &m, const **vec** &x)

{

**vec** res = **vec**(m.**size**(), 0.);

for (**size\_t** i = 0; i < m.**size**(); i++)

{

for (**size\_t** j = 0; j < m.**size**(); j++)

{

res**[**i**]** += m**[**i**][**j**]** \* x**[**j**]**;

}

}

return res;

}

Оператор разности двух векторов:

**vec** **operator-**(const **vec** &x, const **vec** &y)

{

**vec** **ans**(x.**size**(), 0);

for (**size\_t** i = 0; i < x.**size**(); i++)

{

ans**[**i**]** = x**[**i**]** - y**[**i**]**;

}

return ans;

}

Оператор суммы двух векторов:

**vec** **operator+**(const **vec** &x, const **vec** &y)

{

**vec** **ans**(x.**size**(), 0);

for (**size\_t** i = 0; i < x.**size**(); i++)

{

ans**[**i**]** = x**[**i**]** + y**[**i**]**;

}

return ans;

}

Оператор умножения вектора на число:

**vec** **operator\***(const **vec** &x, double &y)

{

**vec** **ans**(x.**size**(), 0);

for (**size\_t** i = 0; i < x.**size**(); i++)

{

ans**[**i**]** = x**[**i**]** \* y;

}

return ans;

}

Класс системы линейных алгебраических уравнений:

struct **Equasions**

{

long N = 0; *//Размерность матрицы*

**matrix** Matrix; *//Матрица*

**vec** x; *//Правильный ответ*

**vec** res; *//Полученный ответ*

**vec** f; *//Соответствущая x правая часть*

double h1, h2;*//ограничение на спектор матрицы*

**vector**<**uint64\_t**> it\_par = {1};*//вектор итерационных параметров*

**vec** error;*//вектор среднеквадратической нормы погрешности*

Конструктор класса: считывает из заданного файла матрицу, считает размерность матрицы:

**Equasions**(**string** filename)

{

**string** line;

**ifstream** **in**(filename);

if (in.**is\_open**())

{

while (**getline**(in, line))

{

if (line.**empty**())

break;

Matrix.**emplace\_back**();

for (**size\_t** i = 0; i < line.**length**(); i++)

{

**string** x = "";

while (line**[**i**]** != ',' && i != line.**length**())

{

x **+=** line**[**i**]**;

++i;

}

Matrix**[**N**]**.**push\_back**(**stod**(x, nullptr));

}

++N;

}

}

in.**close**();

}

Функция, которая случайным образом генерирует вектор-столбец решений x с равномерно распределёнными на отрезке [-1; 1] компонентами:

void **Equasions**::**CalculateRandX**(const unsigned int seed)

{

x **=** **vec**(N, 0);

**default\_random\_engine** **rand**(seed);

**uniform\_real\_distribution**<double> **dist**(0.0, 1.0);

for (**size\_t** i = 0; i < N; i++)

{

x**[**i**]** = **dist(**rand**)**;

}

}

Функция, вычисляющая правую часть системы уравнений по вектору x:

void **Equasions**::**CalculateF**()

{

f **=** **MulMV**(Matrix, x);

}

Функция, вычисляющая среднеквадратическую норму погрешности решения:

double **RMSE**()

{

double ans = 0;

for (int i = 0; i < N; ++i)

{

ans += (res**[**i**]** - x**[**i**]**) \* (res**[**i**]** - x**[**i**]**);

}

return **sqrt**(ans / N);

}

Функция, вычисляющая относительную погрешность решения:

double **Relative\_Error**()

{

double ans = 0;

double nx = 0;

for (int i = 0; i < N; ++i)

{

ans += (res**[**i**]** - x**[**i**]**) \* (res**[**i**]** - x**[**i**]**);

nx += x**[**i**]** \* x**[**i**]**;

}

return **sqrt**(ans) / **sqrt**(nx);

}

Функция, оценивающая спектр матрицы:

void **Interval\_Characteristics**()

{

double sum = 0;

double min = **numeric\_limits**<double>::**max**();

double max = 0;

for (**size\_t** i = 0; i < N; i++)

{

double center = **fabs**(Matrix**[**i**][**i**]**);

sum = 0;

for (**size\_t** j = 0; j < N; j++)

{

sum += **fabs**(Matrix**[**i**][**j**]**);

}

sum -= center;

if (center - sum < min)

{

min = center - sum ;

}

if (center + sum > max)

{

max = center + sum ;

}

}

h1 = min;

h2 = max;

}

Функция, которая находит итерационные параметры:

void **Iteration\_Parameters**()

{

**vector**<**uint64\_t**> new\_par;

**uint64\_t** m = it\_par.**size**();

for (**size\_t** i = 0; i < it\_par.**size**(); i++)

{

new\_par.**push\_back**(it\_par**[**i**]**);

**uint64\_t** new\_p = 4 \* m - it\_par**[**i**]**;

new\_par.**push\_back**(new\_p);

}

it\_par **=** new\_par;

}

Функция, котрая реализует метод Чебышёва:

void **Chebyshev\_method**()

{

double tau0 = 2 / (h1 + h2);

double r0 = (h2 - h1) / (h2 + h1);

res **=** **vec**(N, 0);

**uint64\_t** m = it\_par.**size**();

error **=** **vec**(m, 0);

for (**size\_t** i = 0; i < m; i++)

{

double tauk = tau0 / (1 - r0 \* **cos**(**M\_PI** \* it\_par**[**i**]**/ (2 \* m)));

res **=** (f **-** **MulMV**(Matrix, res)) **\*** tauk **+** res;

error**[**i**]** = **RMSE**();

}

}

Main:

int **main**()

{

**Equasions** m = **Equasions**("SLAU\_var\_2.csv"); *//Считывает матрицу из файла*

for (**size\_t** i = 0; i < m.N; i++)*//Аx+x=f <=> (A+I)x=f*

{

m.Matrix**[**i**][**i**]** += 1;

}

if (!**CheckMatrix**(m)) *//Проверка на симметричность*

{

cout **<<** "Матрица не является симметричной" **<<** **endl**;

return 0;

}

**Equasions** copy\_m = m;*// Сохраняет копию матрицы*

double norm = 0; *//Переменная для подсчета средней среднеквадратической нормы погрешности*

for (**size\_t** i = 0; i < 100; i++) *//Производим 100 итераций вычисления решения системы уравнений*

{

m.**CalculateRandX**(i);

m.**CalculateF**();m.**CalculateL**();

m.**FirstGaussBackward**();

m.**SecondGaussBackward**();

norm += m.**RMSE**(); *//Считаем среднеквадратическую погрешность*

m **=** copy\_m; *//возвращаем матрице ее первоначальные значения*

}

double Kholetsky\_error = norm / 100;

cout **<<** "Среднеквадратическую погрешность методом Холецкого = " **<<** Kholetsky\_error **<<** **endl**;

m.**Interval\_Characteristics**();

cout **<<** "Оценки спектра: h1 = " **<<** m.h1 **<<** " h2 = " **<<** m.h2 **<<** **endl**;

copy\_m **=** m;*// Сохраняет копию матрицы*

double Chebyshev\_error = **numeric\_limits**<double>::**max**();

int n = -1;*//степень двойки количества итераций*

while(Chebyshev\_error > Kholetsky\_error)

{

++n;

Chebyshev\_error = 0;

for (**size\_t** i = 0; i < 10; i++)

{

m.**CalculateRandX**(i);

m.**CalculateF**();

m.**Chebyshev\_method**();

Chebyshev\_error += m.**RMSE**();

}

Chebyshev\_error /= 10;

cout **<<** "Среднеквадратической отклонение при числе итераций " **<<** (1 << n) **<<** " = "**<<** Chebyshev\_error **<<** **endl**;

m.**Iteration\_Parameters**();

}

cout **<<** "Наименьший показатель степени двойки = " **<<** n **<<** **endl**;

cout **<<** "Относительная погрешность решения, полученного методом Чебышева = " **<<** m.**Relative\_Error**() **<<** **endl**;

**ofstream** file;

file.**open**("Errors.txt");

file **<<** '[';

for (auto elem : m.error) {

file **<<** **std**::**fixed** **<<** **std**::**setprecision**(20) **<<** elem **<<** ", ";

}

file **<<** ']';

file.**close**();

return 0;

}

**Полученные результаты**

Среднеквадратическую погрешность методом Холецкого = 2.19438e-16

Оценки спектра: h1 = 1 h2 = 153.4

Среднеквадратической отклонение при числе итераций 1 = 0.0920647

Среднеквадратической отклонение при числе итераций 2 = 0.531929

Среднеквадратической отклонение при числе итераций 4 = 0.407347

Среднеквадратической отклонение при числе итераций 8 = 0.223949

Среднеквадратической отклонение при числе итераций 16 = 0.0615709

Среднеквадратической отклонение при числе итераций 32 = 0.00501245

Среднеквадратической отклонение при числе итераций 64 = 2.58137e-05

Среднеквадратической отклонение при числе итераций 128 = 8.01435e-10

Среднеквадратической отклонение при числе итераций **256** = **1.7709e-16**

Наименьший показатель степени двойки = 8

Относительная погрешность решения, полученного методом Чебышева = 3.29051e-16

График среднеквадратической нормы погрешности решения:

Изображение выглядит как текст, линия, График, снимок экрана

Автоматически созданное описание

Среднеквадратическая норма погрешности

Номер итерации