Лабораторная работа

«Итерационный степенной метод»

1. **Постановка Задачи.**

Необходимый для выполнения работы теоретический материал и формулы имеются в файле «Степенной метод».

**Задание 1.** Разработать программу вычисления наибольшего и второго по величине модуля собственных значений и соответствующих им собственных векторов симметричной матрицы.

Матрицу задать следующим образом (воспользоваться программой лабораторной работы «Решение систем на основе разложения симметричных матриц»):

* недиагональные элементы *ai,j*, *i<j*, выбираются из чисел 0, –1, –2, –3, *–*4 произвольным образом; если *i>j*, то полагается *ai,j*=*aj,i*.
* *ai,i=*, 2≤*i*≤*n*;
* *a*11*=*.

Для вычислений выбрать *n* – одно из чисел в пределах от 10 до 12. В качестве языка программирования выбрать C или C++, использовать тип float.

Для вычисления наибольшего по модулю собственного значения и соответствующего собственного вектора использовать формулы из пункта Случай 1 (файл «Степенной метод»). В формуле λ1≈ (формула вспомогательная, не для счета) выбирать такое *i*, для которого достигается ). В этом случае справедливо (это уже для счета) λ1≈.

Для вычисления наибольшего по модулю собственного значения использовать также формулу λ1≈.

Вывести на печать *uk* на итерациях 46–50 (*k=*46,…,50).

Для обоих случаев: вывести на печать приближенное λ1 для *k=*46,…,50; вычислить вектор *vk+*1–λ1*uk* (это, для проверки насколько оказались хороши вычисленные λ1 и *uk*, вектор «погрешности» *Auk*–λ1*uk*) для *k=*50, вычислить и вывести на печать кубическую норму (максимум-норму) этого вектора.

Для вычисления второго по величине модуля собственного значения использовать формулу λ2≈. Выбирать такое *i*, для которого достигается ).

Рассмотреть три случая:

1) *m=*30, λ1 берется при *k=*50 по формуле для не обязательно симметричной матрицы;

2) *m=*50, λ1 берется при *k=*50 по формуле для не обязательно симметричной матрицы;

3) *m=*50, λ1 берется при *k=*50 по формуле для симметричной матрицы.

Во всех случаях λ1 уже было вычислено ранее.

Для трех случаев: вывести на печать приближенное λ2, вычислить и вывести на печать собственный вектор *u*, соответствующий собственному значению λ2, вычислить вектор *Au*–λ2*u*, вычислить и вывести на печать кубическую норму этого вектора.

Содержание работы должно включать следующие пункты.

1. Постановка задачи.
2. Расчетные формулы.
3. Входные данные.
4. Листинг программы.
5. Выходные данные.
6. Выводы.
7. **Расчетные формулы**

*y*0*=*[1, 0, …, 0]

*u*0*=y*0,

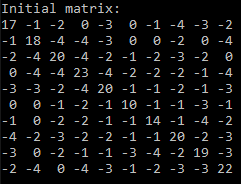
*vk=Auk*−1,  *uk=*, *k =* 1, 2, …

λ1≈

λ1≈

λ2≈

1. **Входные данные**



1. **Листинг программы**

Файл SymmetricMatrix.h:

#pragma once

#include <vector>

#include <iostream>

class SymmetricMatrix

{

public:

SymmetricMatrix();

~SymmetricMatrix();

float get(size\_t i, size\_t j) const;

void power\_iteration() const;

std::vector<float> multiply\_by\_vector(const std::vector<float>& v) const;

friend std::ostream& operator<<(std::ostream&, const SymmetricMatrix&);

private:

std::vector<std::vector<float>> A;

static const int n = 10;

};

Файл SymmetricMatrix.cpp:

#include "SymmetricMatrix.h"

#include <ctime>

#include <iomanip>

#include "Vector\_operations.h"

#include <cmath>

SymmetricMatrix::SymmetricMatrix()

{

//srand(time(NULL));

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

{

A.emplace\_back(i + 1);

for (int j = 0; j < i + 1; ++j)

{

if (i != j)

A[i][j] = -rand() % 5;

}

}

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

{

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

if (i != j)

A[i][i] -= get(i, j);

}

A[0][0] += 1;

}

SymmetricMatrix::~SymmetricMatrix()

{

}

float SymmetricMatrix::get(size\_t i, size\_t j) const

{

return i < j ? A[j][i] : A[i][j];

}

void SymmetricMatrix::power\_iteration() const

{

std::vector<float> u(n); // u\_k

u[0] = 1;

auto v = multiply\_by\_vector(u); // v\_k+1

float lambda1\_1, lambda1\_2;

std::vector<float> u\_29, v\_30, u\_30, v\_31, u\_49, v\_50;

for (int k = 1; k <= 50; ++k)

{

u = v / max\_norm(v);

v = multiply\_by\_vector(u);

size\_t i = index\_of\_max\_abs\_el(v);

lambda1\_1 = v[i] \* (u[i] > 0 ? 1 : -1);

lambda1\_2 = dot\_product(v, u) / dot\_product(u, u);

if (k >= 46)

{

std::cout << "k = " << k << ": " << u << std::endl;

std::cout << "1. lambda\_1 = " << lambda1\_1 << std::endl;

std::cout << "2. lambda\_1 = " << lambda1\_2 << std::endl;

}

if (k == 29)

{

u\_29 = u;

v\_30 = v;

}

if (k == 30)

{

u\_30 = u;

v\_31 = v;

}

if (k == 49)

{

u\_49 = u;

v\_50 = v;

}

}

std::cout << "\n1. ||v^51 - lambda\_1 \* u^50|| = " << max\_norm(v - u \* lambda1\_1) << std::endl;

std::cout << "2. ||v^51 - lambda\_1 \* u^50|| = " << max\_norm(v - u \* lambda1\_2) << std::endl;

size\_t i = index\_of\_max\_abs\_el(v\_30 - lambda1\_1 \* u\_29);

auto tmp\_u = v\_31 - lambda1\_1 \* u\_30;

float lambda2 = (v\_31[i] \* max\_norm(v\_30) - lambda1\_1 \* v\_30[i]) / (v\_30[i] - lambda1\_1 \* u\_29[i]);

std::cout << "\n1) lambda\_2 = " << lambda2 << std::endl

<< "u = " << tmp\_u << std::endl

<< "||A\*u - lambda\_2\*u|| = " << max\_norm(multiply\_by\_vector(tmp\_u) - lambda2 \* tmp\_u) << std::endl;

i = index\_of\_max\_abs\_el(v\_50 - lambda1\_1 \* u\_49);

tmp\_u = v - lambda1\_1 \* u;

lambda2 = (v[i] \* max\_norm(v\_50) - lambda1\_1 \* v\_50[i]) / (v\_50[i] - lambda1\_1 \* u\_49[i]);

std::cout << "\n2) lambda\_2 = " << lambda2 << std::endl

<< "u = " << tmp\_u << std::endl

<< "||A\*u - lambda\_2\*u|| = " << max\_norm(multiply\_by\_vector(tmp\_u) - lambda2 \* tmp\_u) << std::endl;

i = index\_of\_max\_abs\_el(v\_50 - lambda1\_2 \* u\_49);

tmp\_u = v - lambda1\_2 \* u;

lambda2 = (v[i] \* max\_norm(v\_50) - lambda1\_2 \* v\_50[i]) / (v\_50[i] - lambda1\_2 \* u\_49[i]);

std::cout << "\n2) lambda\_2 = " << lambda2 << std::endl

<< "u = " << tmp\_u << std::endl

<< "||A\*u - lambda\_2\*u|| = " << max\_norm(multiply\_by\_vector(tmp\_u) - lambda2 \* tmp\_u) << std::endl;

}

std::vector<float> SymmetricMatrix::multiply\_by\_vector(const std::vector<float>& v) const

{

size\_t n = v.size();

std::vector <float> ans(n);

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

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

{

ans[i] += get(i, j) \* v[j];

}

return ans;

}

std::ostream & operator<<(std::ostream & os, const SymmetricMatrix & m)

{

for (int i = 0; i < m.n; ++i)

{

for (int j = 0; j < m.n; ++j)

os << std::setw(2) << m.get(i, j) << ' ';

os << '\n';

}

return os;

}

Файл Vector\_operations.cpp:

#pragma once

#include <vector>

#include <iomanip>

inline float max\_norm(std::vector<float> v)

{

float max = 0;

for (float el : v)

if (abs(el) > max)

max = abs(el);

return max;

}

inline std::vector<float> operator-(const std::vector<float>& a, const std::vector<float>& b)

{

std::vector<float> tmp(a.size());

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

tmp[i] = a[i] - b[i];

return tmp;

}

inline std::vector<float> operator\*(const std::vector<float>& v, float f)

{

auto tmp = std::move(v);

for (auto& el : tmp)

el \*= f;

return tmp;

}

inline std::vector<float> operator\*(float f, const std::vector<float>& v)

{

return v\*f;

}

inline std::vector<float> operator/(const std::vector<float>& v, float f)

{

return v \* (1 / f);

}

inline size\_t index\_of\_max\_abs\_el(const std::vector<float>& v)

{

size\_t tmp = 0;

float max = 0;

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

if (abs(v[i]) > max)

{

tmp = i;

max = abs(v[i]);

}

return tmp;

}

inline float dot\_product(const std::vector<float>& a, const std::vector<float>& b)

{

float tmp = 0;

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

tmp += a[i] \* b[i];

return tmp;

}

std::ostream& operator<<(std::ostream& os, const std::vector<float>& v)

{

os << '(';

for (size\_t i = 0; i < v.size() - 1; ++i)

{

os << std::fixed << std::setprecision(4) << v[i] << ", ";

}

os << v[v.size()-1] << ')';

return os;

}

Файл main.cpp:

#include <iostream>

#include "SymmetricMatrix.h"

using namespace std;

int main()

{

SymmetricMatrix s0, s1;

cout << "Initial matrix:\n"<< s0 << endl;

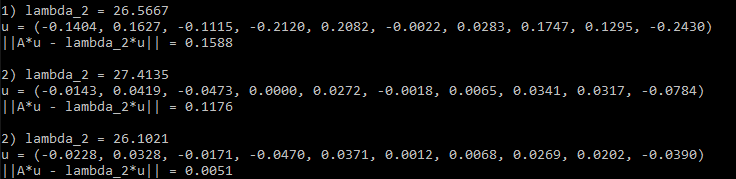
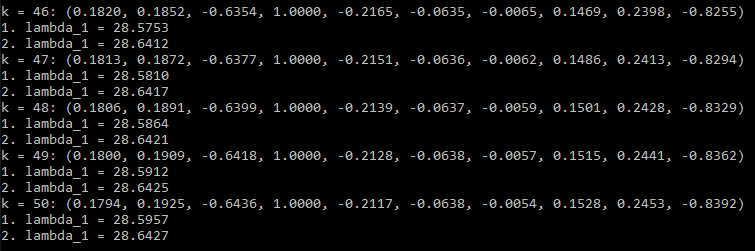
s0.power\_iteration();

system("pause");

return 0;

}

1. **Выходные данные**



1. **Выводы**

При вычисления наибольшего по модулю собственного значения симметричной матрицы предпотчительней использовать формулу λ1≈ чем λ1≈, так как сходимость в таком случае более быстрая.

При вычисления второго по величине модуля собственного значения симметричной матрицы предпотчительней вычислять λ1 по формуле для симметричной матрицы и при это использовать наибольшее m.