|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
|  | | |
| Практическое задание № 5 | | |
| по дисциплине «Численные методы» | | |
|  | | |
| **Собственные числа** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМ-71 |
| Студент: | Востриков Вячеслав |
|  |  |
| Преподаватели: | Патрушев И.И. |
|  | Задорожный А.Г. |
|  | Персова М.Г. |
|  |  |
|  | | |
| Новосибирск | | |
| 2020 | | |

1. **Цель**

Реализовать степенной метод для поиска максимума и модифицировать его для поиска минимума (без обращения матрицы, т.е.через LU-разложение).

1. **Анализ**

Для нахождения максимального собственного значения будем использовать степенной метод. Для того нахождение минимального собственного значения будем находить максимальное собственное значение обратной матрицы.

**Поиск модуля максимального значения:**

Генерируем случайный ненулевой вектор . В данном случае берём вектор .

Для k = 1, 2, … вычисляются:

Чтобы не было переполнения, каждые 5 итераций цикла

если < eps и k < max\_iter, то продолжаем итерации

Результат итерационного процесса:

**Поиск модуля минимального значения:**

Строим обратную матрицу и делаем поиск по модулю максимального значения.

1. **Текст программы**

#include <string>

#include <fstream>

#include <locale>

#include <iostream>

#include <iomanip>

using namespace std;

#define N 10

typedef double T;

int n, max\_iter = 10000;

T eps = 1e-14, max, min;

T\* x1, \*x;

T\*\* A;

void Hylbert\_m(const int size)

{

A = new T\* [size];

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

A[i] = new T[size];

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

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

A[i][j] = 1. / T(i + j + 1);

ofstream fout("hylb.txt");

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

{

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

fout << setprecision(15) << A[i][j] << " ";

fout << endl;

}

}

void input()

{

ifstream fcin("matrix\_info.txt");

A = new T\* [n];

x = new T[n];

x1 = new T[n];

fcin >> n;

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

{

A[i] = new T[n];

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

fcin >> A[i][j];

}

fcin.close();

}

void output()

{

ofstream fout("result.txt");

fout << setprecision(15) << "MAX\_VALUE: " << to\_string(max) << endl<< "MIN\_VALUE: "<< to\_string(min);

fout.close();

}

T norm\_vec(T\* vec)

{

T sum = 0;

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

sum += vec[i] \* vec[i];

return sqrt(sum);

}

void LU\_dec()

{

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

{

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

{

T sum = 0;

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

sum += A[j][k] \* A[k][i];

A[j][i] -= sum;

}

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

{

T sum = 0;

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

sum += A[i][k] \* A[k][j];

A[i][j] = (A[i][j] - sum) / A[i][i];

}

}

}

void solve\_LU(T\* res)

{

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

{

T sum = 0;

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

sum += A[i][j] \* res[j];

res[i] = (x[i] - sum) / A[i][i];

}

for (int i = n - 1; i >= 0; i--)

{

T sum = 0;

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

sum += A[i][j] \* res[j];

res[i] -= sum;

}

}

void MAX\_eingenvalue()

{

int iter;

T norm\_1, norm\_2, supp\_val;

T lambda\_1, lambda = 0;

bool end = false;

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

x[i] = i;

norm\_1 = norm\_vec(x);

for (iter = 0; iter < max\_iter && !end; iter++)

{

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

{

x1[i] = 0;

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

x1[i] += A[i][j] \* x[j];

}

norm\_2 = norm\_vec(x1);

lambda\_1 = norm\_2 / norm\_1;

supp\_val = fabs((lambda\_1 - lambda) / lambda\_1);

if (supp\_val < eps)

end = true;

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

x[i] = x1[i];

lambda = lambda\_1;

norm\_1 = norm\_2;

if (iter % 5 == 0)

{

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

x[i] /= norm\_1;

norm\_1 = 1;

}

}

cout << "iter\_max = " << to\_string(iter) << endl;

max = lambda;

}

void MIN\_eingenvalue()

{

int iter;

T norm\_1, norm\_2, supp\_val;

T lambda\_1, lambda = 0;

bool end = false;

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

x[i] = i;

norm\_1 = norm\_vec(x);

LU\_dec();

for (iter = 0; iter < max\_iter && !end; iter++)

{

solve\_LU(x1);

norm\_2 = norm\_vec(x1);

lambda\_1 = norm\_2 / norm\_1;

supp\_val = fabs((lambda\_1 - lambda) / lambda\_1);

if (supp\_val < eps)

end = true;

x = x1;

lambda = lambda\_1;

norm\_1 = norm\_2;

if (iter % 5 == 0)

{

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

x[i] /= norm\_1;

norm\_1 = 1;

}

}

cout << "iter\_min = " << to\_string(iter);

min = 1. / lambda;

}

int main()

{

int choice;

setlocale(LC\_ALL, "rus");

cout << "Сгенерировать матрицу Гильберта? n - размерность матрицы, \* <= 0 - нет.\n";

cin >> choice;

if (choice > 0)

Hylbert\_m(choice);

input();

MAX\_eingenvalue();

MIN\_eingenvalue();

output();

return 1;

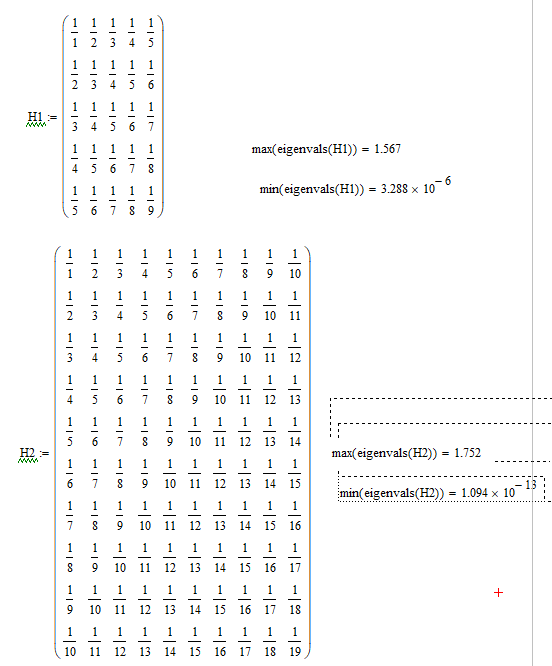
}

1. **Исследования**

Во всех тестах eps = 1e-14, max\_iter = 10000.

***Матрица Гильберта (n = 5, n = 10)***

Результаты Маткада:



Результаты программы:

Для матрицы размерности k = 5

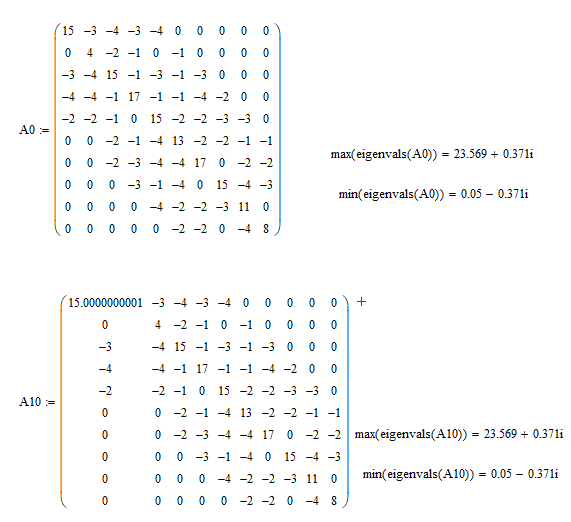


Для матрицы размерности k = 10



***Матрица (k = 0, k = 10)***

Результаты Маткада:



Результаты программы:

Для матрицы k = 0

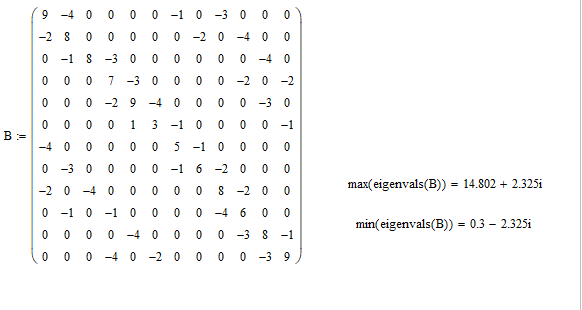


Для матрицы k = 10



***Матрица с диагональным преобладанием***

Результаты Маткада:

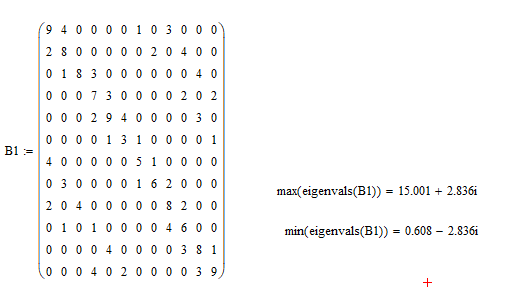


Результаты программы:



***Матрица с положительным знаком вне диагональных элементов***

Результаты Маткада:



Результаты работы программы:

