Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра прикладной математики

Лабораторная работа №5

по дисциплине «Численные методы»

Группа ПМ-63

Студент Майер В. А.

Преподаватель Задорожный А.Г.

Новосибирск, 2018

1. **Цель работы**  
   Найти минимальное и максимальное собственное значение.
2. **Анализ**

Поиск максимального по модулю собственного значения степенным методом:

Упорядочим собственные значения:

|𝜆1|> |𝜆1| ≥⋯ ≥|𝜆𝑛|

Возьмём произвольный вектор y. Запишем его разложение по базису:

𝑦0=𝑐1𝑥1+ 𝑐2𝑥2+⋯+ 𝑐𝑛𝑥𝑛

Вычислим

𝑦1=𝐴𝑦0=𝑐1𝐴𝑥1+ 𝑐2𝐴𝑥2+⋯+ 𝑐𝑛𝐴𝑥𝑛

Так как {𝜆𝑖,𝑥𝑖} при всех 𝑖= {1,2,…,𝑛} по предположению являются собственными парами матрицы A, то, в силу 𝐴𝑥=𝜆𝑥, можно переписать

𝑦1=𝐴𝑦0=𝑐1𝜆1𝑥1+ 𝑐2𝜆2𝑥2+⋯+ 𝑐𝑛𝜆𝑛𝑥𝑛

Для второй итерации:

𝑦2=𝐴𝑦1=𝑐1𝜆12𝑥1+ 𝑐2𝜆22𝑥2+⋯+ 𝑐𝑛𝜆𝑛2𝑥𝑛

Для k-ой итерации

𝑦𝑘=𝐴𝑦(𝑘−1)=𝑐1𝜆1𝑘𝑥1+ 𝑐2𝜆2𝑘𝑥2+⋯+ 𝑐𝑛𝜆𝑛𝑘𝑥𝑛

𝑦𝑘=𝑐1𝜆1𝑘[𝑥1+ 𝑐2𝑐1(𝜆2𝜆1)𝑘𝑥2+⋯+ 𝑐𝑛𝑐1(𝜆𝑛𝜆1)𝑘𝑥𝑛]

Т.к |𝜆𝑖𝜆1|𝑘𝑘(𝑖≠1)→∞→0, в фигурной скобке начинает преобладать первое слагаемое.

Поиск минимального по модулю собственного значения методом обратных итераций:

Применение степенного метода к матрице 𝐴−1

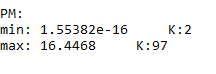
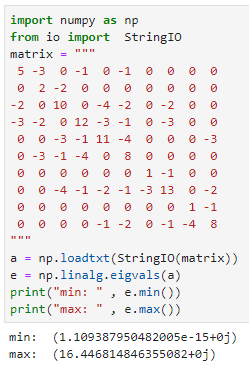
|𝜆1|> |𝜆1| ≥⋯ ≥|𝜆𝑛| ⇒ |1𝜆𝑛|> |1𝜆𝑛−1| ≥⋯ ≥|1𝜆1|

Чтобы избежать обращения матрицы 𝐴 строятся обратные итерации, вместо умножения матрицы на вектор решают систему

𝐴𝑦(𝑘)=𝑦(𝑘−1)

1. **Матрицы из ЛР1**

Заданная точность e = 1e-10 для относительной погрешности и e0 = 1e-20 при сравнении с 0



1. **Матрицы из ЛР2**

|  |  |  |
| --- | --- | --- |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
| Матрица Гильберта | | |
| N | Результат numpy | Результат работы программы |
| 1 |  |  |
| 2 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |

1. **Нулевая матрица 5\*5**

|  |  |
| --- | --- |
| Результат numpy | Результат работы программы |
|  |  |

1. **Диогональные матрицы**

10^I 0 0 0 0 10^-I + 1 0 0 0 0

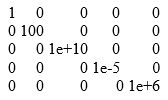
0 1 0 0 0 0 1 0 0 0

0 0 1 0 0 0 0 1 0 0

0 0 0 1 0 0 0 0 1 0

0 0 0 0 1 0 0 0 0 1

|  |  |  |
| --- | --- | --- |
| I | Результат работы программы | Результат работы программы |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |

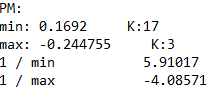


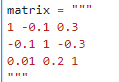
1. **Матрицы с комплексными собственными значениями**



Сравнение с обратной матрицей





1. **Код программы**

#include <iostream>

#include <vector>

#include <fstream>

#include <windows.h>

#include <string>

#include <random>

#include <chrono>

#include <math.h>

#pragma region 1

typedef void(\*Gause)(std::vector<std::vector<double>> &A, std::vector<double> &F, std::vector<double> &X);

typedef void(\*LUInIt)(std::vector<std::vector<double>> &A);

typedef void(\*LU)(std::vector<double> &F, std::vector<double> &X);

using namespace std;

class Matrix

{

public:

Matrix() { n = m = 0; e = 1e-10; e0 = 1e-20; maxiter = 100000; }

~Matrix() {}

vector<vector<double>> A1;

vector<vector<double>> A;

vector<double> y;

vector<double> buf;

vector<double> x;

int k;

void HouseholderTransformation()

{

int n = A.size();

int l, k, j, i;

double scale, hh, h, g, f;

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

{

l = i - 1; h = scale = 0;

if (l > 0)

{

for (k = 0; k <= l; k++)

scale += abs(A[i][k]);

if (!scale)

y[i] = A[i][l];

else

{

for (k = 0; k <= l; k++)

{

A[i][k] /= scale;

h += A[i][k] \* A[i][k];

}

f = A[i][l];

g = (f >= 0 ? -sqrt(h) : sqrt(h));

y[i] = scale \* g; h -= f \* g;

A[i][l] = f - g;

f = 0;

for (j = 0; j <= l; j++)

{

A[j][i] = A[i][j] / h;

g = 0.;

for (k = 0; k <= j; k++)

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

for (k = j + 1; k <= l; k++)

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

y[j] = g / h;

f += y[j] \* A[i][j];

}

hh = f / (h + h);

for (j = 0; j <= l; j++)

{

f = A[i][j];

y[j] = g = y[j] - hh \* f;

for (k = 0; k <= j; k++) A[j][k] -= (f\*y[k] + g \* A[i][k]);

}

}

}

else

y[i] = A[i][l];

x[i] = h;

}

x[0] = 0;

y[0] = 0;

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

{

l = i - 1;

if (x[i]) {

for (j = 0; j <= l; j++) {

g = 0;

for (k = 0; k <= l; k++)

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

for (k = 0; k <= l; k++)

A[k][j] -= g \* A[k][i];

}

}

x[i] = A[i][i];

A[i][i] = 0.;

for (j = 0; j <= l; j++)

A[j][i] = A[i][j] = 0;

}

}

void QL() {

int n = A.size();

int m, l, iter, i, k;

double s, r, p, g, f, dd, c, b;

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

y[i - 1] = y[i];

y[n - 1] = 0;

for (l = 0; l < n; l++)

{

iter = 0;

do {

for (m = l; m < n - 1; m++)

{

dd = abs(x[m]) + abs(x[m + 1]);

if ((double)(abs(y[m] + dd) == dd))

break;

}

if (m != l)

{

if (++iter >= maxiter)

{

cout << "Error!!!";

return;

}

g = (x[l + 1] - x[l]) / (2 \* y[l]);

r = hypot(1., g);

if (g >= 0.)

g += fabs(r);

else

g -= fabs(r);

g = x[m] - x[l] + y[l] / g;

s = c = 1; p = 0;

for (i = m - 1; i >= l; i--)

{

f = s \* y[i]; b = c \* y[i];

y[i + 1] = r = hypot(f, g);

if (!r)

{

x[i + 1] -= p; y[m] = 0;

break;

}

s = f / r; c = g / r; g = x[i + 1] - p; r = (x[i] - g)\*s + 2 \* c\*b; x[i + 1] = g + (p = s \* r); g = c \* r - b;

}

if (r == 0. && i >= l) continue;

x[l] -= p; y[l] = g;

y[m] = 0;

}

} while (m != l);

}

}

bool Open(string path)

{

try

{

ifstream info("info.txt");

info >> n >> maxiter ;

info.close();

A.resize(n);

ifstream in("A.txt");

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

{

A[i].resize(n);

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

in >> A[i][j];

}

in.close();

y.resize(n);

/\*in.open("Y.txt");

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

in >> y[i];

in.close();\*/

}

catch (const std::exception& e)

{

return false;

}

return true;

}

void CreateGilbert(UINT N)

{

A.resize(N);

n = N;

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

{

A[i].resize(n);

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

A[i][j] = 1 / 1.0/(i + j + 1);

}

y.resize(n);

}

double MaxEigen(bool t = 0)

{

default\_random\_engine generator(chrono::system\_clock::to\_time\_t(chrono::system\_clock::now()));

uniform\_int\_distribution<int> distribution(1, 100000000);

vector<double> y1(n);

k = 0;

if (1)

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

y1[i] = 1;

else

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

y1[i] = distribution(generator);

x.resize(n);

lamda.resize(n);

double NormY;

buf.resize(n);

bool c = false, he;

double last;

int ce;

while (maxiter > k && !c)

{

NormY = Norm(y1);

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

y1[i] /= NormY;

mult(y1, x);

c = true;

he = false;

for (ce = 0; ce < n; ce++)

{

if (abs(y1[ce]) > e0 && abs(x[ce]) > e0)

{

buf[ce] = x[ce] / y1[ce];

last = buf[ce];

he = true;

break;

}

}

if (!he)

return 0;

for (; ce < n; ce++)

if (abs(y1[ce]) > e0)

{

buf[ce] = x[ce] / y1[ce];

if (abs(buf[ce] - last)/last > e)

c = false;

if (last < buf[ce])

last = buf[ce];

}

x.swap(y1);

k++;

}

return last;

}

double MinEigen(bool t = 0)

{

auto hinstLib = LoadLibrary(TEXT("LDU.dll"));

LUInIt create = (LUInIt)GetProcAddress(hinstLib, "LUFInit");

LU lu = (LU)GetProcAddress(hinstLib, "LUF");

Gause gause = (Gause)GetProcAddress(hinstLib, "Gause");

create(A);

default\_random\_engine generator(chrono::system\_clock::to\_time\_t(chrono::system\_clock::now()));

uniform\_int\_distribution<int> distribution(-10000000, 10000000);

vector<double> y1(n);

k = 0;

if(t)

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

y1[i] = 1;

else

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

y1[i] = distribution(generator);

x.resize(n);

buf.resize(n);

del(y1, Norm(y1), x);

int I;

double sSSum = 0;

double c = false, hElem;

double last;

size\_t ce;

while (maxiter > k && !c)

{

//gause(A, x, y1);

lu(x, y1);

c = true;

I = 0;

sSSum = 0;

hElem = false;

for (ce = 0; ce < n; ce++)

{

if (abs(x[ce]) > e0 && abs(y1[ce]) > e0)

{

buf[ce] = y1[ce] / x[ce];

last = buf[ce];

hElem = true;

break;

}

}

ce++;

if (!hElem)

return false;

for (; ce < n; ce++)

if (abs(x[ce]) > e0)

{

buf[ce] = y1[ce] / x[ce];

if (abs(buf[ce] - last) / last > e)

c = false;

if (buf[ce] > last)

last = buf[ce];

}

del(y1, Norm(y1), x);

k++;

}

return 1/last;

}

double FindNearestEigen()

{

auto hinstLib = LoadLibrary(TEXT("LDU.dll"));

Gause gause = (Gause)GetProcAddress(hinstLib, "Gause");

if(x.size() < n)

x.resize(n);

if (buf.size() < n)

{

buf.resize(n);

y.resize(n);

}

if (A1.size() < n)

A1 = A;

default\_random\_engine generator(chrono::system\_clock::to\_time\_t(chrono::system\_clock::now()));

uniform\_int\_distribution<int> distribution(-10000000, 10000000);

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

x[i] = distribution(generator);

double normX = Norm(x);

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

x[i] /= normX;

mult(x, buf);

double p = Scolar(buf, x);

k = 0;

auto pCheck = p + 1;

while (maxiter > k && abs(p - pCheck) > 1e-13)

{

pCheck = p;

CheckEigen(p);

gause(A1, x, y);

del(y, Norm(y), x);

mult(x, buf);

p = Scolar(buf, x);

k++;

}

return p;

}

size\_t n;

private:

double e, e0;

size\_t m;

size\_t maxiter;

vector<double> lamda;

double max;

inline void CheckEigen(double p)

{

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

A1[i][i] = A[i][i] - p;

}

inline double sum(vector<double> x)

{

double sum = 0;

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

sum += x[i];

return sum;

}

inline void mult(vector<double> x, vector<double> &y)

{

double sum;

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

{

sum = 0;

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

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

y[i] = sum;

}

}

inline void del(vector<double> x, double y, vector<double> &res)

{

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

res[i] = x[i] / y;

}

inline double Scolar(vector<double> a, vector<double> b)

{

double sum = 0;

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

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

return sum;

}

inline double Norm(vector<double> a)

{

return sqrt(Scolar(a, a));

}

};

#pragma region Перебор единичного вектора в n мерной гиперфере(не доделано)

#pragma endregion

int main()

{

Matrix matrix;

double min, max, mmin, mmax;

if (!matrix.Open(""))

return -1;

//matrix.CreateGilbert(12);

//ищет случайное собственное значение поэтому нужен перебор все единичных векторов

//cout << "Реэля: \n";

//min = max = matrix.FindNearestEigen();

//double p;

//vector<int> index;

//vector<double> result;

//for (size\_t i = 0; i < pow(2, matrix.n); i++)

//{

// p = matrix.BackCircle();

// if (p < min)

// min = p;

// if (p > max)

// max = p;

//}

/\*cout << "min: " << min << endl;

cout << "max: " << max << endl;

cout << endl << endl;\*/

int k1, k2;

double mm = matrix.MaxEigen(1), mmm = matrix.MaxEigen();

mmax = abs(mm) > abs(mmm) ? mm: mmm;

k1 = matrix.k;

mm = matrix.MinEigen(1), mmm = matrix.MinEigen();

mmin = abs(mm) < abs(mmm) ? mm : mmm;

k2 = matrix.k;

cout << "PM: \n";

cout << "min: "<< mmin << " K:" << k2 << endl;

cout << "max: "<< mmax << " K:" << k1 << endl;

cout << endl << endl;

cout << "QL: \n";

matrix.HouseholderTransformation();

matrix.QL();

sort(matrix.x.begin(), matrix.x.end());

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

cout << matrix.x[i] << endl;

system("pause");

return 0;

}

#pragma endregion