|  |
| --- |
| БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ |
| Факультет прикладной математики и информатики |
| Кафедра информационных систем управления |
| Лукьянович Александр Сергеевич  10 группа, 2 курс |
| Лабораторная работа |
|  |
|  |
| **Итерационный метод вращений. Итерационно-степенной метод**    **Преподаватель**  *Полевиков Виктор Кузьмич* |
|  |
| Дата сдачи:  08.12.2016 |

Оглавление

[Постановка задачи 3](#_Toc468960293)

[Итерационный метод вращений: 3](#_Toc468960294)

[Итерационно-степенной метод: 3](#_Toc468960295)

[Краткая теория 4](#_Toc468960296)

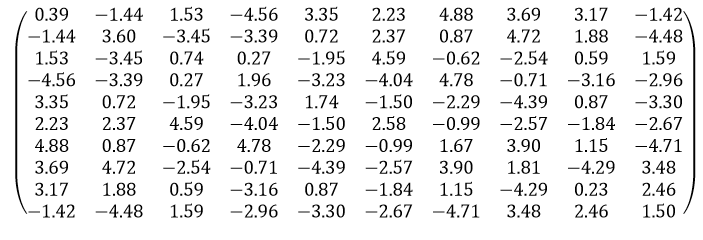
[Итерационный метод вращений: 4](#_Toc468960297)

[Итерационно-степенной метод: 5](#_Toc468960298)

[Листинг программы 6](#_Toc468960299)

[Результаты 8](#_Toc468960300)

# Постановка задачи

Рассматриваем следующую матрицу : , где – вещественная матрица.

## Итерационный метод вращений:

1. Построить стандартную программу для поиска спектра матрицы по методу вращений Якоби для числа уравнений с заданной точностью
2. Найти с помощью этой программы все собственные значения . Останавливать итерационный процесс по апостериорной оценке, т.е. заканчивать вычисления при .

## Итерационно-степенной метод:

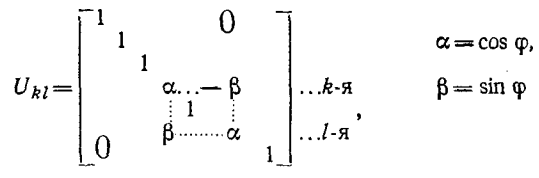
1. Построить стандартную программу для поиска спектрального радиуса по итерационно-степенному методу для числа уравнений с заданной точностью .
2. Решить систему с помощью этой программы и получить . Останавливать итерационный процесс по апостериорной оценке, т.е. заканчивать вычисления при .

# Краткая теория

## Итерационный метод вращений:

Строим итерационную последовательность (у подобных матриц спектры равны), где – матрица вращений, а – индексы максимального по модулю внедиагонального элемента.

Матрица имеет следующий вид:



Непосредственно перемножением легко проверить, что , поэтому итерационная последовательность приобретает вид .

Для упрощения формул

)

)

Построим матрицу , которая отличается от матрицы только элементами столбцов, остальные элементы совпадают:

Аналогично, матрица отличается от матрицы только элементами строк:

Полученная матрица .

Останавливаем итерационный процесс при .

## Итерационно-степенной метод:

Перенумеровываем в порядке убывания модули собственных значений:

Таким образом, искомое собственное значение . Выбираем произвольное начальное итерационное приближение .

Строим итерационную последовательность . Последовательность собственных значений сходится к искомому собственному значению . Останавливаем итерационный процесс при . Чтобы не получать переполнений или исчезновений чисел, после каждой итерации нормируем вектор:

# Листинг программы

Программа выполнена на языке программирования C++, в среде Microsoft Visual Studio Community 2015.

#include "resources.h"

int main(int argc, char argv[]) {

setlocale(LC\_ALL, ".1251");

const int n = 10;

const double eps = pow(10, -5);

try {

ifstream fin("input.txt");

matrix<double> A(create(n, fin));

fin.close();

ofstream fout("output.csv", ios\_base::trunc);

fout << "матрица А:" << endl;

print(A, fout);

fout << "\nматрица собственных значений:" << endl;

printf(IMV(A, eps), fout);

pair<double, vector<double>> maxL(ISM(A, eps));

fout << "\nмаксимальное собственное значение:\n" << maxL.first

<< "\n\nсобственный вектор для него:" << endl;

printf(maxL.second, fout);

fout << "\nA\*y:\n";

printf(prod(A, maxL.second), fout);

fout << "\nл\*y:\n";

printf(maxL.first\*maxL.second, fout);

fout.close();

}

catch (exception& e) { cerr << e.what() << endl; }

return 0;

}

template<typename T>

int sign(T val) {

if (val > 0)

return 1;

else if (val == 0)

return 0;

return -1;

}

pair<size\_t, size\_t> findMaxNotDiagElement(matrix<double> A) {

size\_t k = 0, l = 0;

double max = INT\_MIN;

for (size\_t i(0); i < A.size1(); i++)

for (size\_t j(i + 1); j < A.size2(); j++)

if (fabs(A(i, j)) > max) {

k = i;

l = j;

max = fabs(A(i, j));

}

return pair<size\_t, size\_t>(k, l);

}

bool checkMatrix(matrix<double> A, const double eps) {

for (size\_t i(0); i < A.size1(); i++)

for (size\_t j(0); j < A.size2(); j++)

if (i == j)

continue;

else if (fabs(A(i, j)) > eps)

return false;

return true;

}

matrix<double> IMV(matrix<double> A, const double eps) {

while (!checkMatrix(A, eps)) {

pair<size\_t, size\_t> temp(findMaxNotDiagElement(A));

size\_t k = temp.first;

size\_t l = temp.second;

double mju = 2 \* A(k, l) / (A(k, k) - A(l, l));

double alpha = sqrt(0.5\*(1 + 1 / sqrt(1 + mju\*mju))); //cos

double betta = sign(mju)\*sqrt(0.5\*(1 - 1 / sqrt(1 + mju\*mju))); //sin

matrix<double> B(A.size1(), A.size2());

for (size\_t i(0); i < B.size1(); i++) {

B(i, k) = A(i, k)\*alpha + A(i, l)\*betta;;

B(i, l) = -A(i, k)\*betta + A(i, l)\*alpha;

for (size\_t j(0); j < B.size2(); j++)

if (j != k && j != l)

B(i, j) = A(i, j);

}

matrix<double> C(A.size1(), A.size2());

for (size\_t i(0); i < C.size1(); i++) {

C(k, i) = B(k, i)\*alpha + B(l, i)\*betta;

C(l, i) = -B(k, i)\*betta + B(l, i)\*alpha;

for (size\_t j(0); j < C.size2(); j++)

if (j != k && j != l)

C(j, i) = B(j, i);

}

A = C;

}

return A;

}

pair<double, vector<double>> ISM(matrix<double> A, const double eps) {

vector<double> y(A.size1(), 1);

double lambda = 0;

while (true) {

vector<double> yNext(prod(A, y));

double lambdaNext = inner\_prod(yNext, y) / inner\_prod(y, y);

if (fabs(lambdaNext - lambda) <= eps)

break;

lambda = lambdaNext;

y = yNext;

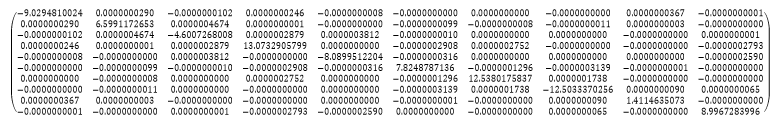
y /= norm\_inf(y);

}

return pair<double, vector<double>>(lambda, y / norm\_inf(y));

}

# Результаты

С помощью метода вращений получаем диагональную матрицу собственных значений :

Находя по итерационно-степенному методу максимальное собственное значение:

Собственный вектор для него:

|  |
| --- |
| -0.24473 |
| 0.126833 |
| -0.45441 |
| 1 |
| -0.67297 |
| -0.66527 |
| 0.863819 |
| 0.856866 |
| -0.50628 |
| -0.19239 |

|  |
| --- |
| -3.19941 |
| 1.658115 |
| -5.94052 |
| 13.07317 |
| -8.79782 |
| -8.69723 |
| 11.29286 |
| 11.20196 |
| -6.61873 |
| -2.51514 |

Выполним проверку:

|  |
| --- |
| -3.22578 |
| 1.631502 |
| -5.9362 |
| 13.07471 |
| -8.79086 |
| -8.68991 |
| 11.28579 |
| 11.22613 |
| -6.59912 |
| -2.52687 |