|  |
| --- |
| БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ |
| Факультет прикладной математики и информатики |
| Кафедра информационных систем управления |
| Лукьянович Александр Сергеевич |
| Лабораторная работа |
|  |
|  |
| **Решение ОСЛАУ методом Гаусса с выбором главного элемента**    **Преподаватель**  *Полевиков Виктор Кузьмич* |
|  |
| Дата сдачи:  22.09.2016 |

Минск, 2016

|  |
| --- |
|  |

Оглавление

[ **Постановка задачи** 3](#_Toc430887395)

[ **Краткая теория** 4](#_Toc430887396)

[ **Листинг программы** 6](#_Toc430887397)

[ **Результаты** 10](#_Toc430887398)

* **Постановка задачи**

Рассматриваем следующую систему: A\*x = f, где A – матрица коэффициентов, x – вектор неизвестных, f – вектор свободных значений.

1. Построить стандартную программу для решения ОСЛАУ методом Гаусса с выбором главного элемента для числа уравнений n ≥ 1.
2. Решить систему с помощью этой программы и получить приближенное значение .
3. Вычислить определитель матрицы A – det A.
4. Найти обратную матрицу .
5. Вычислить меру обусловленности матрицы .

* **Краткая теория**

Имеем систему из n уравнений:

Решая систему по методу Гаусса, приводим матрицу A к нижне-треугольной со следующими коэффициентами:

Вектор неизвестных находится следующим образом:

Определитель матрицы вычисляем следующим образом:

Для нахождения обратной матрицы воспользуемся следующим:

Нормы матриц и вычислены по следующей формуле:

Мера обусловленности матрицы :

* **Листинг программы**

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

#include <iostream>

#include <fstream>

#include <math.h>

#include <exception>

#include <iomanip>

using namespace std;

ofstream fout("output.txt");

ifstream fin("input.txt");

const int n = 10; //размерность квадратной матрицы А

const char\* separator = "\t\t";

void printMatrix(double\*\*);

void printMatrix(double\*);

void readFromFile(double\*\*&, double\*&);

void pryamoyHod(double\*\*, double\*, int&);

void obratnyiyHod(double\*\*, double\*, double\*&);

double determinationOfMatrix(double\*\*);

double\*\* inverseMatrix(double\*\*);

void multiplication(double\*\*, double\*\*);

double\* multiplication(double\*\*, double\*);

double normaOfMatrix(double\*\*);

void cleaningResources(double\*\*&, double\*\*&, double\*&, double\*&, double\*&);

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

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

try {

if (!fout)

throw exception("выходной файл не создан.");

if (!fin)

throw exception("входной файл не отрыт.");

double \*\*matrixA, \*\*inverseMatrixA;

double \*x, \*f, \*x1;

readFromFile(matrixA, x);

fin.close();

f = multiplication(matrixA, x);

int counterOfChanges = 1;

fout << "матрица А:" << endl;

printMatrix(matrixA); fout << endl;

fout << "вектор х(тр):" << endl;

printMatrix(x); fout << endl << endl;

fout << "вектор f(тр):" << endl;

printMatrix(f); fout << endl << endl;

//ходы Гаусса

pryamoyHod(matrixA, f, counterOfChanges);

fout << "матрица А после прямого хода Гаусса:" << endl;

printMatrix(matrixA); fout << endl << endl;

obratnyiyHod(matrixA, f, x1);

fout << "вектор х1(тр):" << endl;

printMatrix(x1); fout << endl << endl;

fout << fixed << setprecision(15) << "определитель A:" << endl

<< (double)counterOfChanges\*determinationOfMatrix(matrixA) << endl << endl;

inverseMatrixA = inverseMatrix(matrixA);

fout << "обратная матрица:" << endl;

printMatrix(inverseMatrixA); fout << endl << endl;

multiplication(matrixA, inverseMatrixA); //принт внутри, хоть и некрасиво

fout << fixed << setprecision(15) << "норма А:" << endl << normaOfMatrix(matrixA) << endl;

fout << fixed << setprecision(15) << "\nнорма A^-1:" << endl << normaOfMatrix(inverseMatrixA) << endl << endl;

fout << fixed << setprecision(15) << "число обусловленности:" << endl

<< normaOfMatrix(matrixA)\*normaOfMatrix(inverseMatrixA) << endl;

fout.close();

cleaningResources(matrixA, inverseMatrixA, x, f, x1);

}

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

return 0;

}

void printMatrix(double\*\* arr) {

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

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

fout << fixed << setprecision(15) << arr[i][j] << separator;

fout << endl;

}

}

void printMatrix(double\* arr) {

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

fout << arr[i] << separator;

}

void readFromFile(double\*\* &matrix, double\* &x) {

matrix = new double\*[n];

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

matrix[i] = new double[n];

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

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

fin >> matrix[i][j];

x = new double[n];

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

fin >> x[i];

}

void pryamoyHod(double\*\* matrixA, double\* f, int &counterOfChanges) {

for (int k(0); k < n - 1; ++k) {

int iMax = k;

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

if (fabs(matrixA[iMax][k]) < fabs(matrixA[i][k]))

iMax = i;

if (iMax != k) {

swap(matrixA[iMax], matrixA[k]);

swap(f[iMax], f[k]);

counterOfChanges \*= -1;

}

for (int i = k + 1; i < n; ++i) {

double temp = (double)matrixA[i][k] / matrixA[k][k];

matrixA[i][k] = 0;

f[i] = f[i] - temp\*f[k];

if (temp)

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

matrixA[i][j] -= temp\*matrixA[k][j];

}

}

}

void obratnyiyHod(double\*\* matrixA, double\* f, double\* &x) {

x = new double[n];

x[n - 1] = (double)f[n - 1] / matrixA[n - 1][n - 1];

//на 1 меньше коэфф(чем в теории) из-за нумерации массива с 0 элемента

for (int i = n - 2; i >= 0; --i) {

double temp = 0;

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

temp += matrixA[i][j] \* x[j];

x[i] = (double)(f[i] - temp) / matrixA[i][i];

}

}

double determinationOfMatrix(double\*\* matrixA) {

double temp = 1.0;

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

temp \*= matrixA[i][i];

return temp;

}

double\*\* inverseMatrix(double\*\* matrix) {

double\*\* inverseMatrix = new double\*[n];

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

inverseMatrix[i] = new double[n];

double\* temp1 = new double[n];

double\* temp2 = new double[n];

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

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

temp2[j] = 0;

temp2[i] = 1.0;

obratnyiyHod(matrix, temp2, temp1);

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

inverseMatrix[j][i] = temp1[j];

}

delete[] temp1;

delete[] temp2;

temp1 = temp2 = nullptr;

return inverseMatrix;

}

void multiplication(double\*\* m1, double\*\* m2) {

double\*\* result = new double\*[n];

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

result[i] = new double[n];

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

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

result[i][j] = 0;

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

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

for (int k(0); k < n; ++k)

result[i][j] += m1[i][k] \* m2[k][j];

fout << "A\*A^-1(должна быть единичная):" << endl;

printMatrix(result); fout << endl << endl;

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

delete[] result[i];

delete[] result;

result = nullptr;

}

double\* multiplication(double\*\* A, double\* x) {

double\* f = new double[n];

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

f[i] = 0;

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

for (int k(0); k < n; ++k)

f[i] += A[i][k] \* x[k];

return f;

}

double normaOfMatrix(double\*\* matrix) {

double sum = 0;

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

double temp = 0;

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

temp += fabs(matrix[i][j]);

if (temp > sum)

sum = temp;

}

return sum;

}

void cleaningResources(double\*\* &a, double\*\* &b, double\* &c, double\* &d, double\* &e) {

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

delete[] a[i];

delete[] b[i];

}

delete[] a;

delete[] b;

delete[] c;

delete[] d;

delete[] e;

a = b = nullptr;

c = d = e = nullptr;

}

* **Результаты**

Матрица :

Вектор точных решений x:

Перемножив матрицу и вектор , получили следующий вектор :

Применив метод Гаусса, получили следующее решение :

По указанной ранее формуле находим определитель матрицы :

По методу Гаусса также нашли следующую обратную матрицу :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -0.1164144353900 | 0.0535946711584 | -0.0444338148617 | -0.0376785251877 | -0.0111558575294 |
| 0.0000000000000 | 0.1011820275276 | 0.0275305274770 | -0.0116277870261 | 0.0047635412676 |
| 0.0000000000000 | 0.0000000000000 | 0.1201304193560 | -0.0622961169832 | 0.0371642893062 |
| 0.0000000000000 | 0.0000000000000 | 0.0000000000000 | 0.1252680079960 | 0.0485611482855 |
| -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0774216552275 |
| -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 |
| 0.0000000000000 | 0.0000000000000 | 0.0000000000000 | 0.0000000000000 | 0.0000000000000 |
| -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 |
| 0.0000000000000 | 0.0000000000000 | 0.0000000000000 | 0.0000000000000 | 0.0000000000000 |
| -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.0344798976839 | -0.0746545147208 | 0.0334346740419 | -0.0680119848780 | -0.4027910210561 |
| -0.0639460262935 | -0.0047776277365 | -0.0178550331903 | 0.1048759419619 | -1.4774357580534 |
| -0.0197170865486 | 0.0335163647775 | -0.0497245272463 | 0.0591869260275 | 0.3888150365323 |
| -0.0462212978324 | 0.0697456377971 | -0.0342835708008 | 0.1548053334678 | 0.5657056735644 |
| 0.1086388961346 | -0.0504566855255 | 0.1015699801046 | -0.0674395235366 | -0.8981667545849 |
| -0.0802582529455 | 0.0526651454156 | -0.0594743257695 | 0.1128929147634 | -0.5709904425099 |
| 0.0000000000000 | 0.0781322282734 | -0.0609825062450 | 0.0208816241760 | 1.6801109398100 |
| -0.0000000000000 | -0.0000000000000 | -0.0323727481437 | 0.0106183907713 | -0.9244641102341 |
| 0.0000000000000 | 0.0000000000000 | 0.0000000000000 | 0.0810469406481 | -0.2997886048489 |
| -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -0.0000000000000 | -1.8405089375810 |

Для проверки полученной обратной матрицы проверим выполнение

:

Далее вычислили нормы матриц и :

Мера обусловленности матрицы :