БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №1

Решение систем линейных уравнений, нахождение определителя и обратной матрицы методом Гаусса

Выполнил:
Студент 2 курса 5 группы ФПМИ
Карасик Семён

Руководитель:
Радкевич Елена Владимировна

Минск, 2017 г.

Оглавление

Постановка задачи

Решить систему линейных уравнений методом Гаусса. Вычислить определитель системы. Найти ей обратную матрицу. Проверить обратную матрицу. Вывести невязку.

Описание метода нахождения решений системы алгебраических уравнений методом Гаусса с выбором главного элемента

Метод Гаусса также называется методом последовательного исключения переменных или схемой единственного деления. В процессе решения системы мы последовательно исключаем данную переменную из других уравнений, получая в результате новую систему из n-1 уравнения с n-1 неизвестной, из которой опять исключаем переменную. В результате чего матрица приводится к верхнетреугольному виду, в котором легко находятся корни уравнений. Его применение возможно, если определитель и все угловые миноры матрицы не равны нулю.

Рассмотрим систему вида:

$$\begin{cases} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{12}x_2+\ldots+a_{2n}x_n=b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n=b_n \end{cases} (1)$$

i = $\overline{1,n}$

1. поменять строки местами так, чтобы на месте $a_{i,i}$ стоял максимальный по модулю ненулевой элемент i-го столбца(выбор опорного элемента по столбцу). Если сделать это невозможно, то система либо не имеет решения, либо имеет бесконечно много решений.
2. поделить i-ую строку на опорный элемент: $a_{i,j}^{(i)}=a_{i,j}^{(i-1)}/a_{i,i}^{(i-1)}$, j = $\overline{i,n}$, $f_i^{(i)}=f_i^{(i-1)}/a_{i,i}^{(i-1)}$
3. исключить переменную x_i из последующих уравнений: $a_{k,j}^{(i)}=a_{k,j}^{(i-1)}-a_{i,j}^{(i)}*a_{k,i}^{(i-1)}$, $f_k^{(i)}=f_k^{(i-1)}-f_i^{(i)}*a_{k,i}^{(i-1)}$, $k=\overline{i+1,n}, j=\overline{i,n}$

Тогда:

$x_n=f_n$, $x_i=f_i-a_{i,i}*x_i-\ldots-a_{i,n}*x_n, i=\overline{n-1,1}$

Нахождение определителя матрицы

Метод Гаусса приводит исходную матрицу A к матрицу A` эквивалентными преобразованиями, которые изменяют определитель с точностью до значений, на которые делятся строки. При поиске опорного элемента мы меняем местами строки, что изменяет знак определителя на обратный. Матрица A' является верхнетреугольной с 1 на главной диагонали, поэтому ее определитель равен 1. Откуда:

$detA = a_{1,1}^{(0)} * ... * a_{n,n}^{(n-1)} * (-1)^p$ где p - число перестановок строк и столбцов.

Нахождение обратной матрицы

Для нахождения обратной матрицы достаточно решить методом Гаусса n-линейных уравнений, у каждого из которых в столбце свободных членов на k-ом месте стоит 1, а на остальных - нули, т е. решить систему с расширенной матрицей вида:

$AX = E \Rightarrow X = A^{-1}$.

Листинг программы

```
//lab1, v20. Simon Karasik, course 2, group 5.

#include <fstream>
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <cmath>
#include <stdexcept>

using namespace std;

typedef vector<double> Vector;
typedef vector<Vector> Matrix;

const double EPS = 10E-4;

bool eq(double x, double y) {
	return abs(x - y) < EPS;
}

Matrix loadIdentity(int n) {
	Matrix mat(n);
	for (int i = 0; i < n; i++) {
		mat[i] = Vector(n, 0);
		mat[i][i] = 1;
	}
	return mat;
}

Matrix loadMatrix(int n, int m) {
	Matrix mat(n);
	for (int i = 0; i < n; i++)
		mat[i] = Vector(m, 0);
	return mat;
}

Matrix operator*(const Matrix & a, const Matrix & b)
{
	if (a[0].size() != b.size())
		throw invalid_argument("Bad size of
matricies.");
	int n = a.size(), m = b[0].size(), l = a[0].size();
	Matrix p = loadMatrix(n, m);
	for (int i = 0; i < n; i++)
		for (int j = 0; j < m; j++) {
			double sum = 0;
			for (int k = 0; k < l; k++)
				sum += a[i][k] * b[k]
[j];

			p[i][j] = sum;
		}
	return p;
}

Vector operator*(const Matrix & a, const Vector & v)
{
	if (a[0].size() != v.size())
		throw invalid_argument("Bad size.");
	int n = a.size(), m = v.size();
	Vector res = Vector(v.size(), 0);
	for (int i = 0; i < n; i++)
		for (int j = 0; j < m; j++)
			res[i] += a[i][j] * v[j];
	return res;
}

void printVector(const Vector & v, ostream & os, bool
useScientific = false) {
	if (useScientific) {
		os << scientific;
		for (int i = 0; i < v.size(); i++)
			os << v[i] << '\t';
	}
	else {
		os << setprecision(4) << fixed;
		for (int j = 0; j < v.size(); j++)
			os << setw(8) << v[j];
	}
	os << endl;
}

void printMatrix(const Matrix & mat, ostream & os,
bool useScientific = false) {
	for (int i = 0; i < mat.size(); i++)
		printVector(mat[i], os, useScientific);
}

Vector Gauss(Matrix a, Vector f, double & det) {
	int n = a.size();
	if (n != f.size())
		throw invalid_argument("Size of
matrix and f must be equal.");
	det = 1;

	for (int step = 0; step < n; step++) {
		int maxStep = step;
		for (int i = step + 1; i < n; i++)
			if (abs(a[i][i]) >
abs(a[maxStep][maxStep]))
				maxStep = i;
		if (maxStep != step) {
			a[step].swap(a[maxStep]);
			swap(f[step], f[maxStep]);
```

```
                det *= -1;
            }
            double anchor = a[step][step];

            det *= anchor;
            f[step] /= anchor;
            for (int j = step; j < n; j++)
                a[step][j] /= anchor;
            for (int i = step + 1; i < n; i++) {
                double k = a[i][step];
                for (int j = step; j < n; j++)
                    a[i][j] -= a[step][j] * k;
                f[i] -= f[step] * k;
            }

            for (int i = 0; i < n; i++) {
                Vector v = a[i];
                v.push_back(f[i]);
            }
        }

        Vector x(n, 0);
        for (int i = n - 1; i >= 0; i--) {
            x[i] = f[i];
            for (int j = n - 1; j > i; j--)
                x[i] -= a[i][j] * x[j];
        }

        return x;
}

Matrix inverse(const Matrix & a) {
        int n = a.size();
        Matrix res = loadMatrix(n, n);
        Vector f(n, 0);
        double det;
        for (int i = 0; i < n; i++) {
            f[i] = 1;
            Vector x = Gauss(a, f, det);
            for (int j = 0; j < n; j++)
                res[j][i] = x[j];
            f[i] = 0;
        }
        return res;
}

int main() {
        ifstream fin("input.txt");
        ofstream fout("output.txt");

        int n;
        fin >> n;
        Matrix a = loadMatrix(n, n);
        Vector f = Vector(n);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                fin >> a[i][j];
        for (int i = 0; i < n; i++)
            fin >> f[i];

        double det;
        Vector x = Gauss(a, f, det);
        Matrix inversed = inverse(a);
        Vector error = a * x;
        for (int i = 0 ; i< x.size(); i++)
            error[i] -= f[i];
        Matrix inversedError = inversed * a;
        for (int i = 0; i < a.size(); i++)
            inversedError[i][i] -= 1;

        fout << "A:" << endl;
        printMatrix(a, fout);
        fout << "f:" << endl;
        printVector(f, fout);
        fout << "x:" << endl;
        printVector(x, fout);
        fout << "Ax - f:" << endl;
        printVector(error, fout, true);
        fout << "A^-1:" << endl;
        printMatrix(inversed, fout);
        fout << "A*A^-1 - E:" << endl;
        printMatrix(inversedError, fout, true);

        return 0;
}
```

Входные данные

```
input.txt:
5
0.4974 0.0000 -0.1299 0.0914 0.1523
-0.0305 0.3284 0.00000 -0.0619 0.0203
0.0102 -0.0914 0.5887 0.0112 0.0355
0.0305 0.0000 -0.0741 0.5887 0.0000
0.0203 -0.0305 0.1472 -0.0122 0.4263
1.5875 -1.7590 1.4139 1.7702 -2.0767S
```

Выходные данные

```
output.txt
A:
  0.4974  0.0000 -0.1299  0.0914  0.1523
 -0.0305  0.3284  0.0000 -0.0619  0.0203
  0.0102 -0.0914  0.5887  0.0112  0.0355
  0.0305  0.0000 -0.0741  0.5887  0.0000
  0.0203 -0.0305  0.1472 -0.0122  0.4263
f:
  1.5875 -1.7590  1.4139  1.7702 -2.0767
x:
  5.0010 -3.9554  2.0057  3.0003 -5.9993
Ax - f:
-3.1086e-15 -8.8818e-16 2.2204e-16  1.1102e-15  0.0000e+00
A^-1:
  2.0568  0.0961  0.6090 -0.3372 -0.7901
  0.1764  3.0609  0.1300  0.2874 -0.2196
 -0.0009  0.4693  1.7419  0.0129 -0.1671
 -0.1067  0.0541  0.1877  1.7177  0.0199
 -0.0881  0.0539 -0.6158  0.0813  2.4259
A*A^-1 - E:
-1.1102e-16 5.1348e-16  -3.1364e-15 7.8063e-17  -3.8858e-16
9.5410e-18  0.0000e+00  -7.2858e-16 1.3444e-17  4.1633e-17
0.0000e+00  -1.9949e-17 0.0000e+00  2.1684e-18  0.0000e+00
-2.3852e-18 6.7221e-17  2.0426e-16  -1.1102e-16 -3.1225e-17
-6.9389e-18 1.1102e-16  -1.1102e-16 -1.3878e-17 -1.1102e-16
```