|  |  |
| --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ПЕРМСКОГО КРАЯ | |
| Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования  «Пермский государственный национальный исследовательский университет» | |
| **ЧИСЛЕННЫЕ МЕТОДЫ  «ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ»** | |
| *Лабораторная работа №2* | |
|  | |
|  | Работу выполнили студенты группы ПМИ-1-2015 3 курса механико-математического факультета  Толов Я.Ф., Заманов М.Р. |
|  | Проверил:  Профессор, доктор физико-математических наук  С. В. Русаков  “\_\_\_\_” 20\_\_ г. |
|  |  |
| Пермь 2017 | |

*Задание*

1. Решить заданную систему линейных алгебраических уравнений методом *LU* разложения с выбором главного элемента по столбцу.
2. Используя полученное ранее *LU* разложение, вычислить обратную матрицу к исходной.
3. Вычислить определитель и найти число обусловленности исходной матрицы.

*Исходные данные*

Варианты исходных данных: 14 a) и 20 a).

Дана матрица А.

Вариант 14:

a)┌ ┐ bт = [34.1, -35.2, 11.2, 1.3] +

│ 8.5 -3.3 1.0 -2.4│

│-3.3 6.4 -4.0 0.1│

│ 1.0 -4.0 6.9 1.4│

│-2.4 0.1 1.4 1.2│

└ ┘

Вариант 20:

a)┌ ┐ bт = [-56.5, -27.5, -49.1, -26.3]

│ 4.3 1.9 -2.9 -1.2│

│ 1.9 4.4 2.0 4.3│

│-2.9 2.0 9.8 2.4│

│-1.2 4.3 2.4 9.3│

└ ┘

Дан вектор X.



***Решение***

Представление матрицы A в виде произведения двух матриц P\*A=L\*U, где L – нижняя треугольная матрица, а U – верхняя треугольная матрица, P – матрица перестановок, получаемая из единичной матрицы путём перестановки строк и столбцов.

┌ ┐ ┌ ┐

│ l11 00 0│ │ 1 u12 u13 u14│ ┌ ┐

L = │ l21 l22 0 0│ U = │ 0 1u23 u24│ │ 0 1 0│

│ l31 l32 l33  0│ │ 0 01 u34│ P = │ 1 0 0│

│ l41 l42 l43 l44│ │ 0 00 1│ │ 0 0 1│

└ ┘ └ ┘ └ ┘

Метод обращения матрицы находит массив x, который при умножении на матрицу LU дает массив b. В данном методе два цикла. Первый цикл использует прямую подстановку в нижней части матрицы LU, а второй цикл — обратную подстановку в верхней части той же матрицы.

= |L| = l11\*l22\*l33\*l44.  
Так как единичной матрицы |P| = 1, определитель матрицы |U| так же равен 1.

Число обусловленности – норма A \* норма A^(-1).

Для нахождения числа обусловленности исходной матрицы необходимо сложить модули всех элементов строки и выбрать максимальное значение.

|A||∞=max1<=i<=n()

***Краткие выводы***

Данная лабораторная работа научила нас решать системы линейных алгебраических уравнений новым методом. Также была написана программа, которая решает СЛАУ посредством LUP разложения.

***Результаты тестирования***

Входные данные: №20

N = 4

A =

6.0 -3.1 -9.7 -6.8

4.0 9.4 -5.7 -8.3

8.1 -5.5 -3.8 -8.7

3.0 -9.5 -6.1 2.0

b = -56.5 -27.5 -49.099999 -26.299999+

Выходные данные:determinant = 1802.5

Inverse matrix

-0.6008 0.39001 0.20413 0.46378

-0.17416 0.16908 0.0018918 0.11777

-0.15006 0.010935 0.10644 -0.0018336

-0.38372 0.25145 0.027423 0.35814

condition number = 45.449

>>step: 0

A[0] before swap

6 -3.1 -9.7 -6.8

4 9.4 -5.7 -8.3

8.1 -5.5 -3.8 -8.7

3 -9.5 -6.1 2

>>main element 2

>>swapping 0 and 2 lines

r = 0

A[0] after swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

6 -3.1 -9.7 -6.8

3 -9.5 -6.1 2

L

8.1 0 0 0

-5.5 12.116 0 0

-3.8 -3.8235 -6.8852 0

-8.7 -4.0037 -0.35556 5.2222

U

1 0.49383 0.74074 0.37037

0 1 0.97407 -7.463

0 0 1 -4.6926

0 0 0 1

>>step: 1

A[1] before swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

6 -3.1 -9.7 -6.8

3 -9.5 -6.1 2

>>main element 1

>>no need to swap

r = 1

A[1] after swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

6 -3.1 -9.7 -6.8

3 -9.5 -6.1 2

L

8.1 0 0 0

-5.5 12.116 0 0

-3.8 -3.8235 -6.5778 0

-8.7 -4.0037 -0.033676 2.7561

U

1 0.49383 0.74074 0.37037

0 1 0.080395 -0.61596

0 0 1 -7.0477

0 0 0 1

>>step: 2

A[2] before swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

6 -3.1 -9.7 -6.8

3 -9.5 -6.1 2

>>main element 3

>>swapping 2 and 3 lines

r = 2

A[2] after swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

3 -9.5 -6.1 2

6 -3.1 -9.7 -6.8

L

8.1 0 0 0

-5.5 12.116 0 0

-3.8 -3.8235 -7.0477 0

-8.7 -4.0037 2.7561 -2.606

U

1 0.49383 0.37037 0.74074

0 1 -0.61596 0.080395

0 0 1 0.93333

0 0 0 1

>>step: 3

A[3] before swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

3 -9.5 -6.1 2

6 -3.1 -9.7 -6.8

>>main element 3

>>no need to swap

r = 3

A[3] after swap

8.1 -5.5 -3.8 -8.7

4 9.4 -5.7 -8.3

3 -9.5 -6.1 2

6 -3.1 -9.7 -6.8

L

8.1 0 0 0

-5.5 12.116 0 0

-3.8 -3.8235 -7.0477 0

-8.7 -4.0037 2.7561 -2.606

U

1 0.49383 0.37037 0.74074

0 1 -0.61596 0.080395

0 0 1 0.93333

0 0 0 1

x: 1 2 3 4

fractional error = 1.0658e-14

.

Входные данные: №14

N = 4

A =

-2 3.6 6.7 2.2

-3.2 0.2 -8.0 -2.1

-0.5 9.9 -8.3 4.2

-6.4 -1.9 8.9 -3.8

b = 34.1 -35.2 11.2 1.3

Выходные данные:

determinant = 756.91

Inverse matrix

-0.51552 -0.44485 0.23652 0.2088

-0.4886 -0.40665 0.34998 0.32867

-0.060719 -0.12836 0.040031 0.080026

0.97034 0.65193 -0.47958 -0.59173

condition number = 61.683

>>step: 0

A[0] before swap

-2 3.6 6.7 2.2

-3.2 0.2 -8 -2.1

-0.5 9.9 -8.3 4.2

-6.4 -1.9 8.9 -3.8

>>main element 3

>>swapping 0 and 3 lines

r = 0

A[0] after swap

-6.4 -1.9 8.9 -3.8

-3.2 0.2 -8 -2.1

-0.5 9.9 -8.3 4.2

-2 3.6 6.7 2.2

L

-6.4 0 0 0

-1.9 1.15 0 0

8.9 -12.45 -8.9953 0

-3.8 -0.2 4.4969 3.3875

U

1 0.5 0.078125 0.3125

0 1 10.048 4.1937

0 0 1 3.9188

0 0 0 1

>>step: 1

A[1] before swap

-6.4 -1.9 8.9 -3.8

-3.2 0.2 -8 -2.1

-0.5 9.9 -8.3 4.2

-2 3.6 6.7 2.2

>>main element 2

>>swapping 1 and 2 lines

r = 1

A[1] after swap

-6.4 -1.9 8.9 -3.8

-0.5 9.9 -8.3 4.2

-3.2 0.2 -8 -2.1

-2 3.6 6.7 2.2

L

-6.4 0 0 0

-1.9 10.048 0 0

8.9 -8.9953 -11.421 0

-3.8 4.4969 -0.71465 1.5107

U

1 0.078125 0.5 0.3125

0 1 0.11445 0.41735

0 0 1 7.673

0 0 0 1

>>step: 2

A[2] before swap

-6.4 -1.9 8.9 -3.8

-0.5 9.9 -8.3 4.2

-3.2 0.2 -8 -2.1

-2 3.6 6.7 2.2

>>main element 2

>>no need to swap

r = 2

A[2] after swap

-6.4 -1.9 8.9 -3.8

-0.5 9.9 -8.3 4.2

-3.2 0.2 -8 -2.1

-2 3.6 6.7 2.2

L

-6.4 0 0 0

-1.9 10.048 0 0

8.9 -8.9953 -11.421 0

-3.8 4.4969 -0.71465 1.0306

U

1 0.078125 0.5 0.3125

0 1 0.11445 0.41735

0 0 1 -0.67186

0 0 0 1

>>step: 3

A[3] before swap

-6.4 -1.9 8.9 -3.8

-0.5 9.9 -8.3 4.2

-3.2 0.2 -8 -2.1

-2 3.6 6.7 2.2

>>main element 3

>>no need to swap

r = 3

A[3] after swap

-6.4 -1.9 8.9 -3.8

-0.5 9.9 -8.3 4.2

-3.2 0.2 -8 -2.1

-2 3.6 6.7 2.2

L

-6.4 0 0 0

-1.9 10.048 0 0

8.9 -8.9953 -11.421 0

-3.8 4.4969 -0.71465 1.0306

U

1 0.078125 0.5 0.3125

0 1 0.11445 0.41735

0 0 1 -0.67186

0 0 0 1

x: 1 2 3 4

fractional error = 1.7764e-15

***Код на C++***

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

void fillLU(double\*\*, double\*\*, int);

void print(double\*\*, int);

double\*\* mult(double\*\*, double\*\*, int);

double det(double\*\*, int);

double\*\* trans(double\*\*, int, int);

double\*\* init(int, int, bool=false);

void SwapRows(double\*\*, int, int);

void LUP(double\*\*, double\*\*, int\*, int);

double\* Solve(double\*\*, double\*, int);

double\*\* GetInvert(double\*\*, int);

void del(double\*\*, int);

double norm(double\*\*, int);

void printLU(double\*\*, int);

double fractionalError(double\*\* A, double\* X, double\* b, int n);

bool flag = false;

int main()

{

ifstream fin("input14.txt");

int n = 0;

fin >> n;

double\*\* A = init(n, n);

double\* b = new double[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 >> b[i];

cout << "A:" << endl;

print(A, n);

double deter = det(A, n);

cout << "determinant = " << deter << endl;

double\*\* InvA = GetInvert(A, n);

cout << "Inverse matrix" << endl;

print(InvA, n);

double conditionNumber = norm(A, n) \* norm(InvA, n);

cout << "condition number = " << conditionNumber << endl << endl;

flag = true;

double\* x = Solve(A, b, n);

flag = false;

cout << "x: ";

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

cout << x[i] << " ";

cout << endl;

cout << "fractional error = " << fractionalError(A, x, b, n) <<endl;

cin.get();

cin.get();

return 0;

}

void fillLU(double\*\* LU, double\*\* A, int n)

{

LU[0][0] = A[0][0];

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

{

LU[j][0] = A[j][0];

LU[0][j] = A[0][j] / LU[0][0];

}

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

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

{

double sum = 0;

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

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

LU[j][i] = A[j][i] - sum;

if (i != j)

{

sum = 0;

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

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

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

}

}

}

void print(double\*\* M, int n)

{

cout.setf(ios::left);

for (int i = 0; i < n; i++, cout << endl)

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

cout << setw(8) << setprecision(5) << M[i][j] << " ";

cout.unsetf(ios::left);

}

void printLU(double\*\* LU, int n)

{

cout.setf(ios::left);

double\*\* L = init(n, n, true);

double\*\* U = init(n, n, true);

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

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

{

if (i <= j)

L[j][i] = LU[i][j];

if (i > j)

U[j][i] = LU[i][j];

if (i == j)

U[i][i] = 1;

}

cout << "L" << endl;

print(L, n);

cout << "U" << endl;

print(U, n);

del(L, n);

del(U, n);

cout.unsetf(ios::left);

}

double\*\* mult(double\*\* A, double\*\* B, int n)

{

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

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

C[i] = new double[n];

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

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

{

double sum = 0;

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

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

C[i][j] = sum;

}

return C;

}

double\*\* trans(double\*\* A, int n,int m)

{

double\*\* B = new double\*[m];

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

B[i] = new double[n];

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

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

B[j][i] = A[i][j];

return B;

}

double\*\* init(int n, int m, bool fillByZero)

{

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

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

A[i] = new double[m];

if (fillByZero)

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

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

A[i][j] = 0.0;

return A;

}

void del(double\*\* M, int n)

{

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

delete[] M[i];

delete[] M;

}

double det(double\*\* A, int n)

{

double\*\* LU = init(n, n);//для экономии памяти марици L и U объеденены

int\* P = new int[n];

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

P[i] = i;

LUP(A, LU, P, n);

double res = 1;

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

res \*= LU[i][i];

delete[] P;

delete[] LU;

return res;

}

double\* Solve(double\*\* A, double\* b, int n)

{

double\*\* LU = init(n, n);//для экономии памяти марици L и U объеденены

int\* P = new int[n];

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

P[i] = i;

LUP(A, LU, P, n);

double\* y = new double[n];

double\* x = new double[n];

double s = 0;

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

{

s = 0;

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

s += LU[i][j] \* y[j];

y[i] = b[P[i]] - s;

}

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

{

s = 0;

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

s += LU[i][j] \* x[j];

x[i] = (y[i] - s) / LU[i][i];

}

delete[] y;

delete[] P;

del(LU, n);

return x;

}

void LUP(double\*\* A, double\*\* LU, int\* P, int n)

{

double\*\* debug = NULL;

if (flag)

{

debug = init(n, n);

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

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

debug[i][j] = A[i][j];

}

for (int i = 0; i < n; i++) //LU = A;

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

LU[i][j] = A[i][j];

int r = 0;

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

{

if (flag)

{

cout << ">>step: " << i << endl;

cout << "A[" << i << "] before swap" << endl;

print(debug, n);

}

//поиск опорного элемента

double pivotValue = 0;

int pivot = -1;

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

{

if (fabs(LU[row][i]) > pivotValue)

{

pivotValue = fabs(LU[row][i]);

pivot = row;

}

}

if (pivotValue != 0)

{

//меняем местами i-ю строку и строку с опорным элементом

int t = P[pivot]; P[pivot] = P[i]; P[i] = t;

SwapRows(LU,pivot, i);

if (flag)

{

SwapRows(debug, pivot, i);

cout << ">>main element " << pivot << endl;

if (pivot != i)

cout << ">>swapping " << i << " and " << pivot << " lines" << endl;

else

cout << ">>no need to swap" << endl;

cout << "r = " << r++ << endl;

}

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

{

LU[j][i] /= LU[i][i];

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

LU[j][k] -= LU[j][i] \* LU[i][k];

}

}

if (flag)

{

cout << "A[" << i << "] after swap" << endl;

print(debug, n);

printLU(LU, n);

cout << endl;

}

}

}

void SwapRows(double\*\* M, int a, int b)

{

double\* t= M[a];

M[a] = M[b];

M[b] = t;

}

double\*\* GetInvert(double\*\* A, int n)

{

double\*\* Invert = init(n, n);

double\* E = new double[n];

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

E[i] = 0;

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

{

E[i]++;

Invert[i] = Solve(A, E, n);

E[i]--;

}

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

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

{

double t = Invert[i][j];

Invert[i][j] = Invert[j][i];

Invert[j][i] = t;

}

delete[] E;

return Invert;

}

double norm(double\*\* M, int n)

{

double sum = 0;

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

sum += fabs(M[0][i]);

double norm = sum;

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

{

sum = 0;

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

sum += fabs(M[i][j]);

if (sum > norm)

norm = sum;

}

return norm;

}

double fractionalError(double\*\* A, double\* X, double\* b, int n)

{

double error = 0;

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

{

double c = b[i];

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

c -= A[i][j] \* X[j];

if (fabs(c) > error)

error = fabs(c);

}

return error;

}