ФГБОУ ВО «КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

КАФЕДРА ЮНЕСКО ПО ИНФОРМАЦИОННЫМ ВЫЧИСЛИТЕЛЬНЫМ ТЕХНОЛОГИЯМ

ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине «Языки программирования»

Семестр 2

Тема: Динамические многомерные массивы. Решение СЛАУ методом Гаусса

Выполнил:

студент группы ФИТ-194

Максименко Р.В.

Кемерово, 2020

Задача 2. «Решение СЛАУ методом Гаусса» (10 баллов). Напишите программу, реализующую решение системы линейных алгебраических уравнений методов Гаусса с выбором главного элемента по строке, по столбцу либо по всей матрице. Для найденного решения необходимо определить точность  (определить максимальную по модулю разницу между невязкой решения и полученным решением). Инструкция: а) находим решение СЛАУ Ax=B методом Гаусса (вектор х) б) находим вектор B1= Ax. в) сравниваем исходный вектор B и полученный вектор B1, для этого находим погрешность вычислений   max Bi  B1i , i  0,1,2,....N 1 ). Значение погрешности  вывести на экран. Также необходимо проводить тестирование программного кода для различных матриц: a. Матрица, заданная явно (тестовый вариант с известным решением); b. Матрица, заданная случайным образом; c. Нулевая или вырожденная матрицы (2 коллинеарные строки, или одна нулевая строка) d. Матрица Гильберта: 6 , где , 0,1,2..... 1 1 1 ,      i j N i j Ai j Для матрицы заданной случайным образом провести оценку времени выполнения программной реализации и построить графики зависимости времени выполнения от порядка матрицы СЛАУ. Сравнить теоретическую и практическую временную сложности алгоритма метода Гаусса (на графиках).

#include"pch.h"

#include <locale>

#include <iostream>

#include <ctime>

using namespace std;

const int n = 4;

int i, j, l, k;

double a1, a2;

double a[n][n + 1] =

{ { 1,1,1,1,1}, //coefficient matrix, where the last column is the right side of the system

{1,2,1,3,0},

{1,1,2,2,3},

{ 1,1,1,3,3} };

int main()

{

setlocale(LC\_ALL, "Russian");

double \*x;

x = (double\*)malloc(n \* sizeof(double));

for (k = 0; k < n; k++) //Finding the maximum item in the first column

{

a1 = abs(a[k][k]);

i = k;

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

if (abs(a[l][k]) > a1)

{

i = l;

a1 = abs(a[l][k]);

}

if (a1 == 0) //zero element check

{

cout << "The system has no solutions." << endl;

}

if (i != k) // permutation of the i-th row containing the main element of the k-th row

{

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

{

a2 = a[k][j];

a[k][j] = a[i][j];

a[i][j] = a2;

}

}

a1 = a[k][k];//k-th row conversion (Calculation of scaling factors)

a[k][k] = 1;

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

a[k][j] = a[k][j] / a1;

for (i = k + 1; i < n; i++)//string conversion using k-th string

{

a2 = a[i][k];

a[i][k] = 0;

if (a2 != 0)

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

a[i][j] = a[i][j] - a2 \* a[k][j];

}

}

for (i = n - 1; i >= 0; i--) //Finding SLAU Solutions

{

x[i] = 0;

a1 = a[i][n];

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

a1 = a1 - a[i][j] \* x[j];

x[i] = a1;

}

cout << "System solution:" << endl; //conclusion of decisions

int t = clock();

cout << ((float)t) / CLOCKS\_PER\_SEC << " seconds.n" << endl;

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

{

cout << "x[" << i + 1 << "]=" << x[i];

cout << endl;

}

system("PAUSE");

return 0;

}

