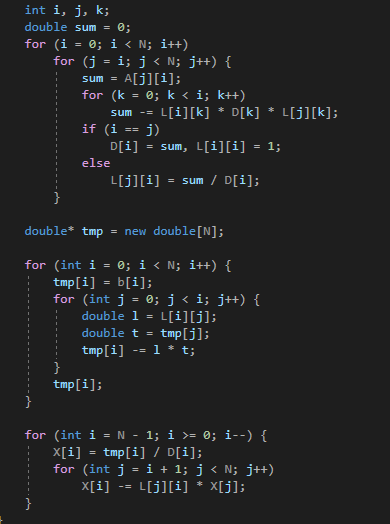
Лабораторная работа 1 Доскоч Роман Дмитриевич 2 курс ПИ

Вариант 14

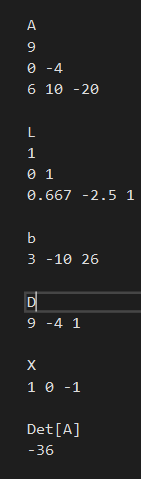
14.1

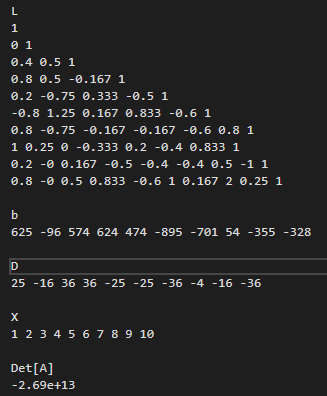
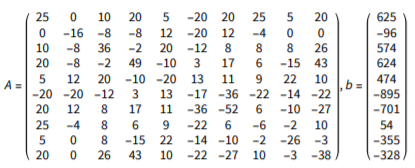


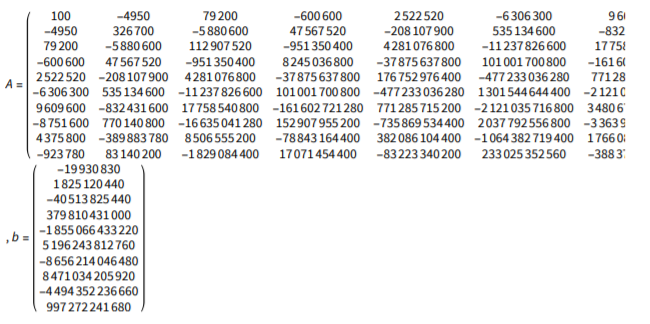
нахождение L и D

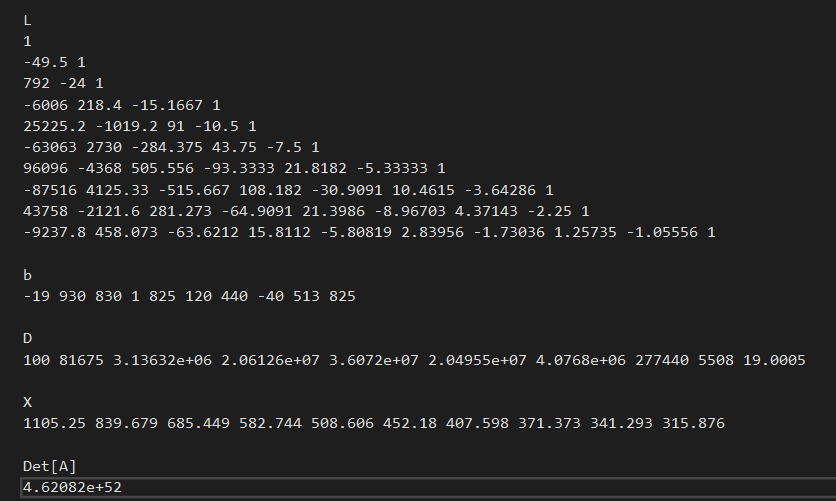
решение СЛАУ Ax = B

Разложение LDLT (полный код в конце отчета)



det = -26 873 856 000 000





Ответ в 3 системе ответ не верен из-за плохой обусловленности (значения матрицы велики).

14.2

Задача называется плохо обусловленной если число обусловленности велико как и в нашем примере.

экспериментальным путем при изменении одного значения вектора B результаты сильно разнятся.

Один из векторов

X

**(4.00003 -0.999968 -0.999969 5.00003 5.00003 -4.99997 -3.99997 4.00002 2.00002 -3.99998)Т**

B

(-19930830 1825120440 -40513825440 379810431000 -185506643322**0** 5196243812760 ….**)Т**

X

(3.80004 -1.16663 -1.14282 4.87503 4.88892 -5.09997 -4.09088 3.91669 1.9231 -4.0714**)Т**

B

(-19930830 1825120440 -40513825440 379810431000 -185506643322**1** 5196243812760 ….**)Т**

X B = ...**2**...

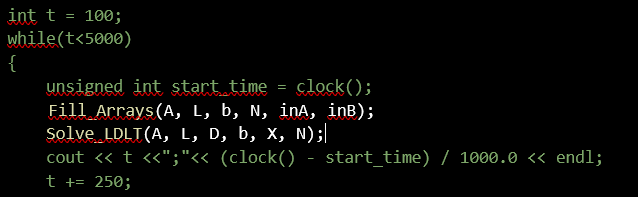
(3.60004 -1.33329 -1.28568 4.75004 4.77781 -5.19997 -4.18179 3.83336 1.84618 -4.14283**)Т**

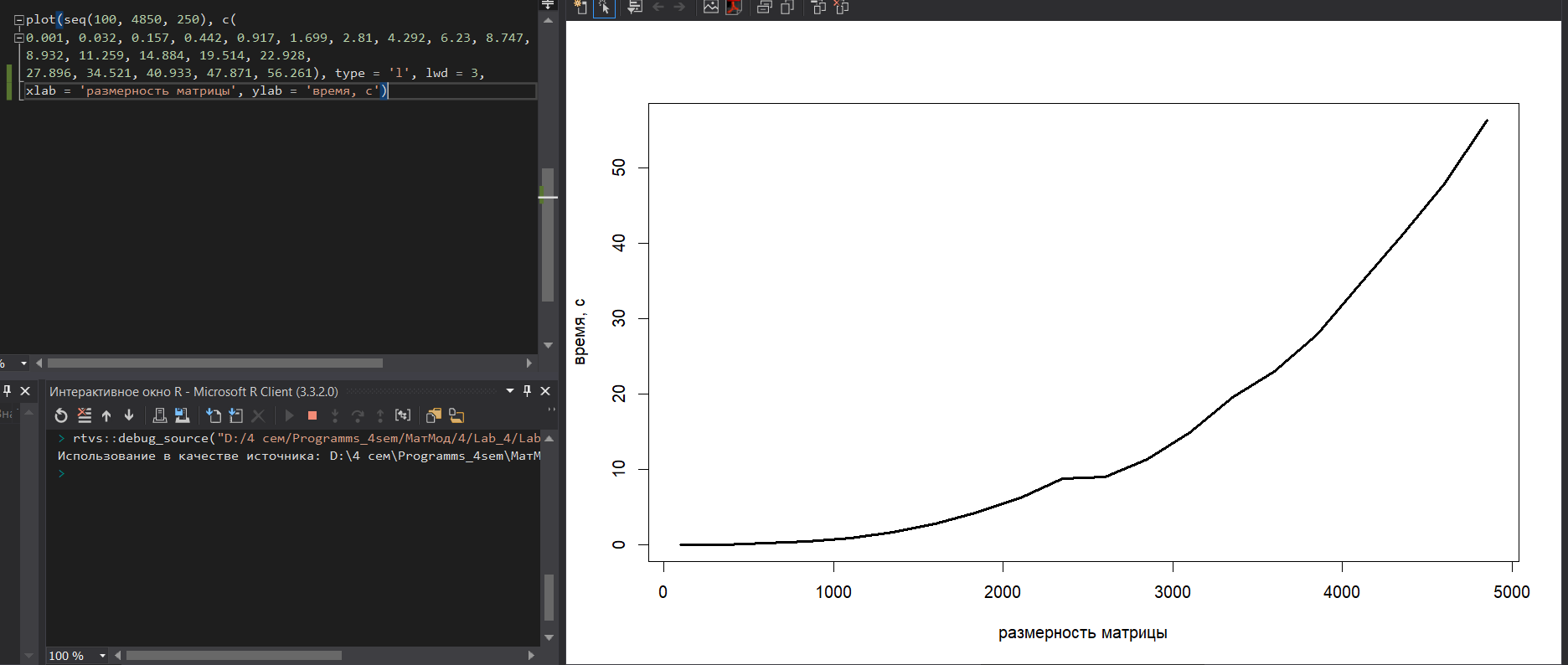
X B = ...**10**...

(2.00008 -2.66659 -2.4285 3.75006 3.88895 -5.99995 -4.90904 3.16672 1.23082 -4.71424 **)Т**

14.3

Функцией Fill\_Arrays заполняю таблицу случайными числами.

В цикле по размерностям пробегаю и замеряю время вывожу получившиеся точки для графика 

график построен с помощью языка R.

**За минуту ~5000 размерность матрицы.**

Main.cpp

#include <fstream>

#include <iostream>

#include <vector>

#include "Functions.h"

#include <ctime>

using namespace std;

int main() {

    srand(time(0));

    ifstream inA("A.txt");

    ifstream inB("b.txt");

    ofstream out("output.txt");

    int N;

    inA >> N;

    double\*\* A = new double\* [N], \*\* L = new double\* [N];

    double\* b = new double[N], \* D = new double[N], \* X = new double[N];

        Fill\_Arrays(A, L, b, N, inA, inB);

        Solve\_LDLT(A, L, D, b, X, N);

        PrintInfo(A, L, D, X, b, N,out);

    /\*

    int t = 100;

    while(t<5000)

    {

        unsigned int start\_time = clock();

Fill\_Arrays(A, L, b, N, inA, inB);

        Solve\_LDLT(A, L, D, b, X, N);

        cout << t <<";"<< (clock() - start\_time) / 1000.0 << endl;

        t += 250;

    }

        \*/

}

Functions.h

#pragma once

#include <fstream>

void PrintInfo(double\*\* A, double\*\* L, double\* D, double\* X, double\* b, int N, std::ofstream& out);

void Fill\_Arrays(double\*\* A, double\*\* L, double\* b, int N, std::ifstream& inA, std::ifstream& inB);

void Solve\_LDLT(double\*\* A, double\*\* L, double\* D, double\* b, double\* X, int N);

Functions.cpp

#pragma once

#include "Functions.h"

#include <fstream>

#include <iomanip>

using namespace std;

void PrintInfo(double\*\* A, double\*\* L, double\* D, double\* X, double\* b, int N, ofstream& out) {

    out << "\nA\n";

    for (int i = 0; i < N; i++) {

        for (int j = 0; j <= i; j++) out << A[i][j] << " ";

        out << "\n";

    }

    out << "\nL\n";

    for (int i = 0; i < N; i++) {

        for (int j = 0; j <= i; j++) out <<L[i][j] << " ";

        out << "\n";

    }

    out << "\nb\n";

    for (int j = 0; j < N; j++) out << b[j] << " ";

    out << "\n\nD\n";

    for (int j = 0; j < N; j++) out << D[j] << " ";

    out << "\n\nX\n";

    for (int j = 0; j < N; j++) out << X[j] << " ";

    out << "\n\nDet[A]\n";

    long double det = 1.0;

    for (int j = 0; j < N; j++) det \*= D[j];

    out << det;

}

void Fill\_Arrays(double\*\* A, double\*\* L, double\* b, int N, std::ifstream& inA, std::ifstream& inB) {

    for (int i = 0; i < N; i++)

        A[i] = new double[i + 1], L[i] = new double[i + 1], inB >> b[i];

    for (int i = 0; i < N; i++)

        for (int j = 0; j <= i; j++)

            inA >> A[i][j];

}

void Solve\_LDLT(double\*\* A, double\*\* L, double\* D, double\* b, double\* X, int N) {

    int i, j, k;

    double sum = 0;

    for (i = 0; i < N; i++)

        for (j = i; j < N; j++) {

            sum = A[j][i];

            for (k = 0; k < i; k++)

                sum -= L[i][k] \* D[k] \* L[j][k];

            if (i == j)

                D[i] = sum, L[i][i] = 1;

            else

                L[j][i] = sum / D[i];

        }

    double\* tmp = new double[N];

    for (int i = 0; i < N; i++) {

        tmp[i] = b[i];

        for (int j = 0; j < i; j++) {

            double l = L[i][j];

            double t = tmp[j];

            tmp[i] -= l \* t;

        }

        tmp[i];

    }

    for (int i = N - 1; i >= 0; i--) {

        X[i] = tmp[i] / D[i];

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

            X[i] -= L[j][i] \* X[j];

    }

}