**Лабораторная работа**

**«Решение систем на основе разложения симметричных матриц»**

**Арсений Анищенко**

**Группа 2**

1. **Постановка задачи.**

Пусть дана система линейных алгебраических уравнений *Ax=b*. Матрица системы симметрическая: *ai,j*=*aj,i*.

Разработать программу численного решения СЛАУ на основе LDLT-разложения.

Матрицу системы сформировать следующим образом:

* недиагональные элементы *ai,j*, *i<j*, выбираются из чисел 0, –1, –2, –3, *–*4 произвольным образом; если *i>j*, то полагается *ai,j*=*aj,i*.
* *ai,i=*, 2≤*i*≤*n*;
* *a*11*=*, *k*≥0.

Правую часть *b* задать умножением матрицы *A* на вектор *x=*(*m*, *m*+1, ... , *n*+*m*–1): *b=Ax*.

Для вычислений выбрать параметры:

* *m* – номер в списке студенческой группы;
* *n* – одно из чисел в пределах от 10 до 12;
* *k* – рассмотреть два случая: *k=*0, *k=*(номер студенческой группы); элементы *ai,j* при фиксированных *i* и *j* в обоих случаях одни и те же (матрицы отличаются только элементом *a*11).

Программно реализовать (в качестве языка программирования выбрать C или C++) вычисления для рассматриваемого примера. Использовать алгоритм (5) файла «LDLt\_RtR разложения», требующий хранения только нижнего треугольника матрицы. В процессе факторизации матрицы *A* (*A=LDLT*) нижняя треугольная матрица *L* (за исключением единиц на главной диагонали) хранится на месте нижнего треугольника матрицы *A,* диагональная матрица *D* хранится на месте главной диагонали матрицы *A*.

При формировании матрицы *A* можно (для простоты формирования) использовать все *n*2 элементов матрицы, при программной реализации LDLT-разложения и решения систем с треугольными матрицами – только нижний треугольник матрицы *A*.

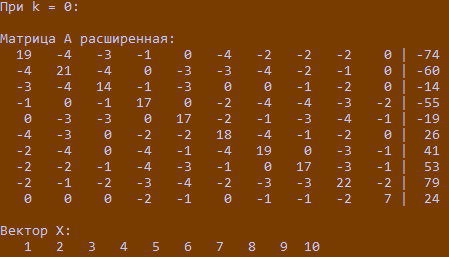
Для обоих случаев выбора *k* в выходных данных отчета должны быть представлены:

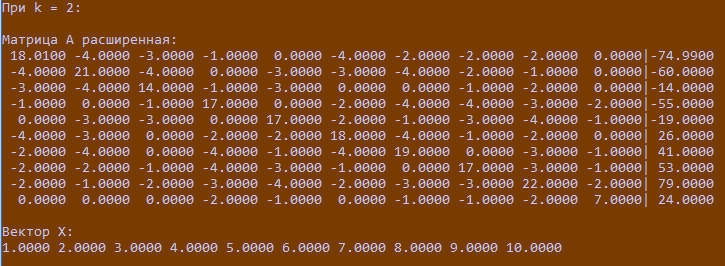
1. Нижняя треугольная матрица *L* (за исключением единиц на главной диагонали), диагональная матрица *D*. Это нижний треугольник преобразованной матрицы *A*, включая главную диагональ.

2. Вектор приближённого решения *x\**.

3. Относительная погрешность вида , где  – точное решение.

**2. Входные данные.**





**3. Листинг программы.**

#include <iostream>

#include <vector>

#include <iomanip>

#include <stdlib.h>

#include <algorithm>

#include <math.h>

#include <time.h>

using namespace std;

const int SIZE = 10;

float \*\*a, s[SIZE], x[SIZE], \*b, \_x[SIZE];

int m = 1, n = SIZE;

float k = 0;

void ldlt(float \*\*a, float \*b){

//Сохранение матрицы А

float \*\*buf = new float\*[SIZE];

for (int i = 0; i < SIZE; ++i){

buf[i] = new float[SIZE];

for (int j = 0; j < SIZE; ++j) {

buf[i][j] = a[i][j];

}

}

//ldlt-разложение

float \*t = new float[SIZE];

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

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

t[i] = a[i][k];

a[i][k] = a[i][k] / a[k][k];

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

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

}

}

}

//Вывод матрицы L

cout << "\nМатрица L:\n";

for (int i = 0; i < SIZE; ++i){

for (int j = 0; j < SIZE; ++j){

float x = a[i][j];

if (i == j)

x = 1;

else if (i < j)

cout << setw(8) << fixed << setprecision(2) << x;

}

cout << endl;

}

//Вывод матрицы D

cout << "Матрица D:" << endl;

for (int i = 0; i < SIZE; ++i){

for (int j = 0; j < SIZE; ++j){

cout.width(5);

cout << fixed << setprecision(2) << setw(8) << ((i == j) ? a[i][j] : 0);

}

cout << endl;

}

//Решение уравнения Ly=b

float \*\_y = new float[SIZE];

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

float sum = 0;

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

sum += a[i][j] \* \_y[j];

}

\_y[i] = b[i] - sum;

}

//Перемножение D и Lt

float \*d = new float[SIZE];

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

d[i] = a[i][i];

a[i][i] = 1;

}

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

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

a[j][i] = d[i] \* a[j][i];

}

}

float \*\_x = new float[SIZE];

//Получение x\*

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

float sum = 0;

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

sum += a[j][i] \* \_x[j];

}

\_x[i] = (\_y[i] - sum) / a[i][i];

}

for (int i = 0; i < SIZE; ++i){

for (int j = 0; j < SIZE; ++j){

a[i][j] = buf[i][j];

}

}

cout << endl << "Вектор X\*: \n";

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

cout << fixed << setprecision(8) << \_x[i] << " ";

}

cout << endl;

float q = 0, w = 0;

//Подсчет относительной погрешности

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

q += pow(\_x[i] - x[i], 2);

w += x[i] \* x[i];

}

q = sqrt(q);

w = sqrt(w);

double error = q / w;

cout << endl << "Относительная погрешность: " << fixed << setprecision(8) << error << endl;

}

int main(){

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

srand(time(0));

a = new float\*[SIZE];

b = new float[SIZE];

//Генерация матрицы А

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

a[i] = new float[SIZE];

}

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

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

if (i != j){

a[i][j] = rand() % 5 - 4;

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

}

}

}

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

s[i] = 0;

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

if (i != j){

s[i] += a[i][j];

}

a[i][i] = -s[i]; //получение диагональных элементов

}

}

cout << "При k = 0:" << endl;

k = 0;

a[0][0] += pow(10, -k);

//вычисление x и b

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

b[i] = 0;

x[i] = m + i;

}

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

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

b[i] += a[i][j] \* x[j];

}

}

cout << endl << "Матрица A расширенная: \n";

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

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

cout << setw(4) << setprecision(4) << a[i][j] << " ";

}

cout << "|" << setw(4) << b[i] << endl;

}

cout << endl << "Вектор X: \n";

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

cout << setw(4) << x[i];

}

cout << endl;

ldlt(a, b);//Применение ldlt для k=0

cout << endl;

cout << "При k = 2:" << endl;

k = 2;

a[0][0] += -1 + pow(10, -k);

//вычисление x и b

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

b[i] = 0;

x[i] = m + i;

}

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

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

b[i] += a[i][j] \* x[j];

}

}

//Вывод А, b, x

cout << endl << "Матрица A расширенная:" << endl;

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

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

cout << setw(8) <<setprecision(4) << a[i][j];

}

cout << "|" << setw(8) << b[i] << endl;

}

cout << endl << "Вектор X: \n";

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

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

}

cout << endl;

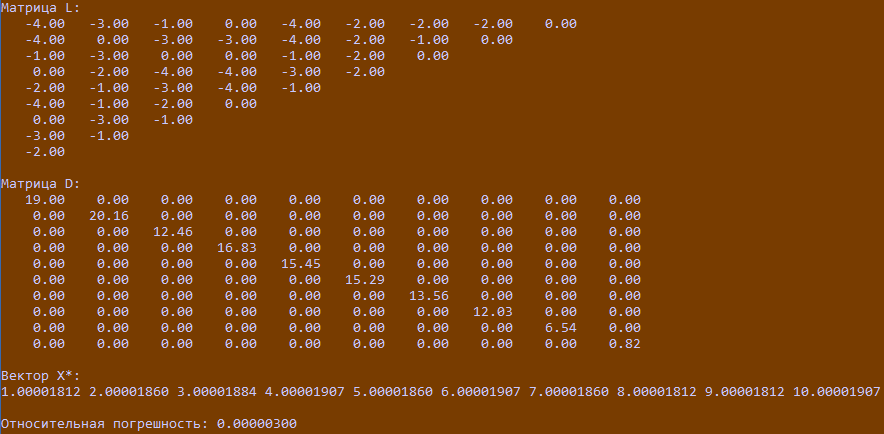
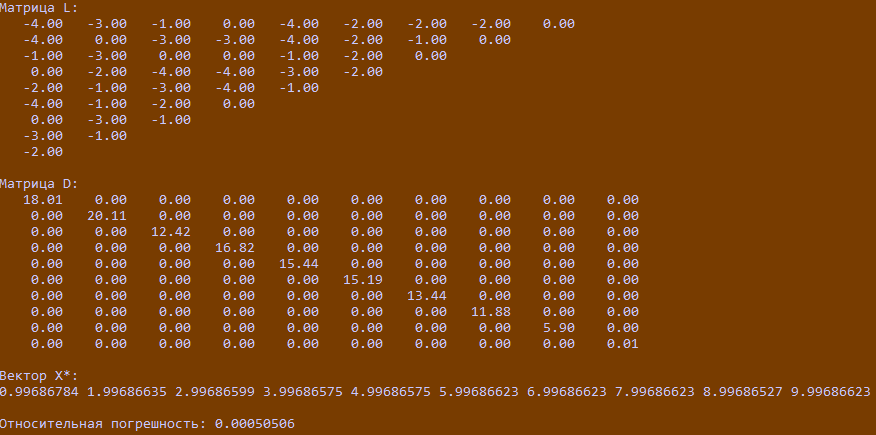
//Применение ldlt для k=2

ldlt(a, b);

return 0;

}

**4. Выходные данные.**

 **5. Вывод.**

Была созданна программа численного решение СЛАУ на основе LDLT-разложения. В случае с k = 0 мы получили погрешность меньше, чем в случае с k = 2. Соответственно 0.00000003% и 0.0000050506%.