**Лабораторная работа**

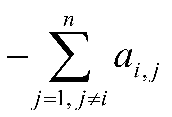
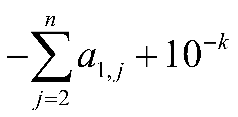
**«Решение систем на основе разложения симметричных матриц»**

Необходимый для выполнения работы теоретический материал и формулы (программируется алгоритм (5)) имеются в файле «LDLt\_RtR разложения».

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

**Задание.** 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).

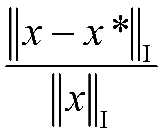
Программно реализовать (в качестве языка программирования выбрать C или C++) вычисления для рассматриваемого примера. Использовать алгоритм (5) файла «LDLt\_RtR разложения», требующий хранения только нижнего треугольника матрицы. В процессе факторизации матрицы *A* (*A=LDLT*) нижняя треугольная матрица *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 https://lh4.googleusercontent.com/GxMCWKkUYcFm56GbgbAdROD7Wn0NdRG4KyQ5vVRtTu-CHHyKtF69Zr7QxQg8FgxAtp73TCjylBB-62kp3go4GIqo71QerAfOZGfXwT8zTF1el0tfvxSnVKCT74KJyZ3X15gWn4U – точное решение.

Отчет должен включать следующие пункты:

1. Постановка задачи.
2. Входные данные.
3. Листинг программы. Обязательны подробные комментарии.
4. Выходные данные.
5. Выводы.

**Входные данные**

1. n=12, m=13, k=0

2. n=12, m=13, k=3

**Листинг программы**

#include <iostream>

#include <random>

#include <cmath>

#include <iomanip>

using namespace std;

//Размерность матрицы

int n = 12;

//Номер в списке группы

int m = 13;

//Номер группы или 0

int k = 0;

//Нахождение случайного числа

int Rand(int L, int R) {

static random\_device rd;

static mt19937 gen(rd());

uniform\_int\_distribution<> dis(L, R);

return dis(gen);

}

//Вывод матрицы

void PrintMatrix(float\*\* A)

{

cout << "A:" << endl;

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

{

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

{

cout << fixed << setprecision(3) << setw(10) << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

//Вывод столбца

void PrintArray(float\* x)

{

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

cout << fixed << setprecision(7) << setw(10) << x[i] << " ";

cout << endl << endl;

}

//Вывод матриц L и D

void PrintLD(float\*\* A)

{

cout << "L:" << endl;

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

{

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

{

if (i == j)

cout << fixed << setprecision(3) << setw(10) << 1 << " ";

else if (j > i)

cout << fixed << setprecision(3) << setw(10) << 0 << " ";

else

cout << fixed << setprecision(3) << setw(10) << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

cout << "D:" << endl;

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

{

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

{

if (i == j)

cout << fixed << setprecision(3) << setw(10) << A[i][j] << " ";

else

cout << fixed << setprecision(3) << setw(10) << 0 << " ";

}

cout << endl;

}

cout << endl;

}

//Умножение матрицы на столбец

void Multiplication(float\*\* A, float\* x, float\* b)

{

float s;

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

{

s = 0;

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

{

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

}

b[i] = s;

}

}

//Нахождение LDLt разложения

void LDLt(float\*\* A)

{

float\* t = new float[n];

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[k][k];

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

{

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

}

}

}

delete[]t;

}

//Решение системы Ly=b

void System1(float\*\* A, float\* b)

{

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

{

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

{

b[j] -= b[i] \* A[j][i];

}

}

}

//Нахождение произведения DLt

void DLt(float\*\* A)

{

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

{

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

{

A[i][j] = A[j][i] \* A[i][i];

}

}

}

//Решение системы DLtx=y

void System2(float\*\* A, float\* b)

{

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

{

b[i] /= A[i][i];

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

{

b[j] -= A[j][i] \* b[i];

}

}

}

//Нахождение относительной погрешности

float Fault(float\* x, float\* b)

{

float\* dx = new float[n];

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

dx[i] = x[i] - b[i];

float ndx = 0;

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

ndx += dx[i] \* dx[i];

ndx = sqrt(ndx);

float nx = 0;

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

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

nx = sqrt(nx);

cout << fixed << setprecision(7) << "Относительная погрешность: " << 100 \* ndx / nx << "%" << endl << endl;

return ndx / nx;

}

int main()

{

setlocale(LC\_ALL, "rus");

//k=0

//Создание матрицы n\*n

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

float\*\* save = new float\* [n];

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

{

A[i] = new float[n];

save[i] = new float[n];

}

//Инициализация

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

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

if (i != j)

{

if (i > j)

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

else

A[i][j] = Rand(-4, 0);

}

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

{

float s = 0;

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

if (i != j)

s -= A[i][j];

A[i][i] = s;

}

//Сохранение матрицы A

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

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

save[i][j] = A[i][j];

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

//Вывод матрицы

PrintMatrix(A);

//Создание и инициализация вектора x

float\* x = new float[n];

cout << "x:" << endl;

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

x[i] = m + i;

PrintArray(x);

//Создание и нахождение вектора b

float\* b = new float[n];

Multiplication(A, x, b);

cout << "b:" << endl;

PrintArray(b);

//Нахождение LDLt разложения

LDLt(A);

PrintLD(A);

//Решение системы Ly=b

System1(A, b);

//Нахождение произведения DLt

DLt(A);

//Решение системы DLtx=y

System2(A, b);

cout << "Вектор приближенного значения x\*:" << endl;

PrintArray(b);

//Расчет погрешности

Fault(x, b);

cout << "------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------" << endl << endl;

//k=3

k = 3;

//Восстановление матрицы A

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

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

A[i][j] = save[i][j];

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

//Вывод матрицы и вектора x

PrintMatrix(A);

PrintArray(x);

//Нахождение вектора b

Multiplication(A, x, b);

cout << "b:" << endl;

PrintArray(b);

//Нахождение LDLt разложения

LDLt(A);

PrintLD(A);

//Решение системы Ly=b

System1(A, b);

//Нахождение произведения DLt

DLt(A);

//Решение системы DLtx=y

System2(A, b);

cout << "Вектор приближенного значения x\*:" << endl;

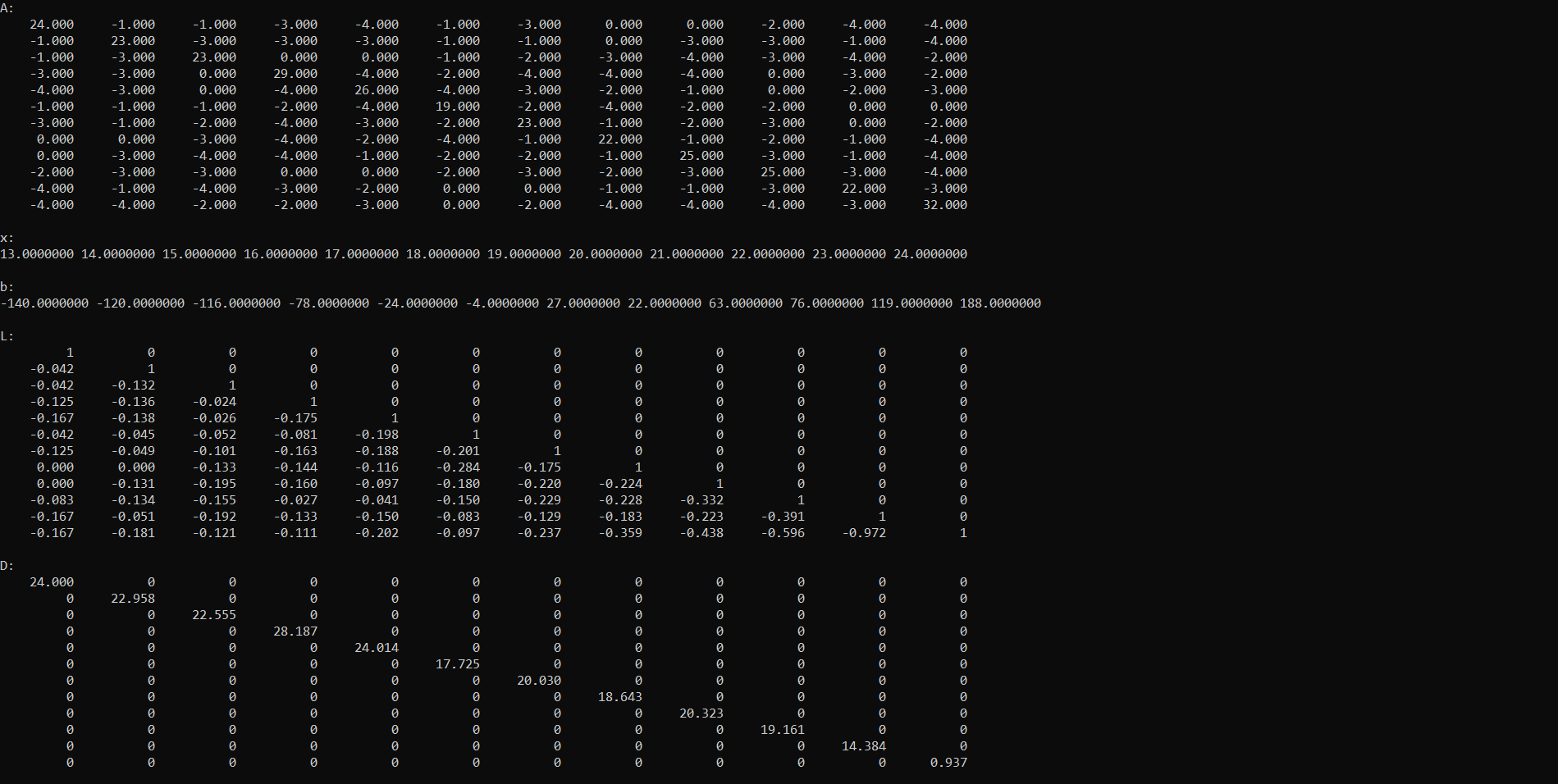
PrintArray(b);

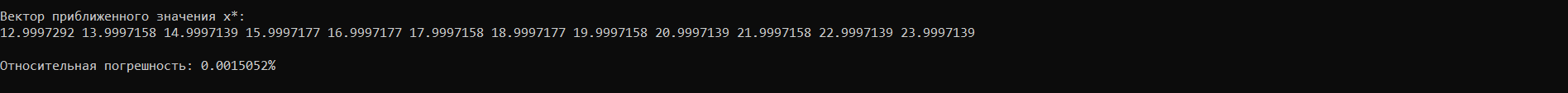
//Расчет погрешности

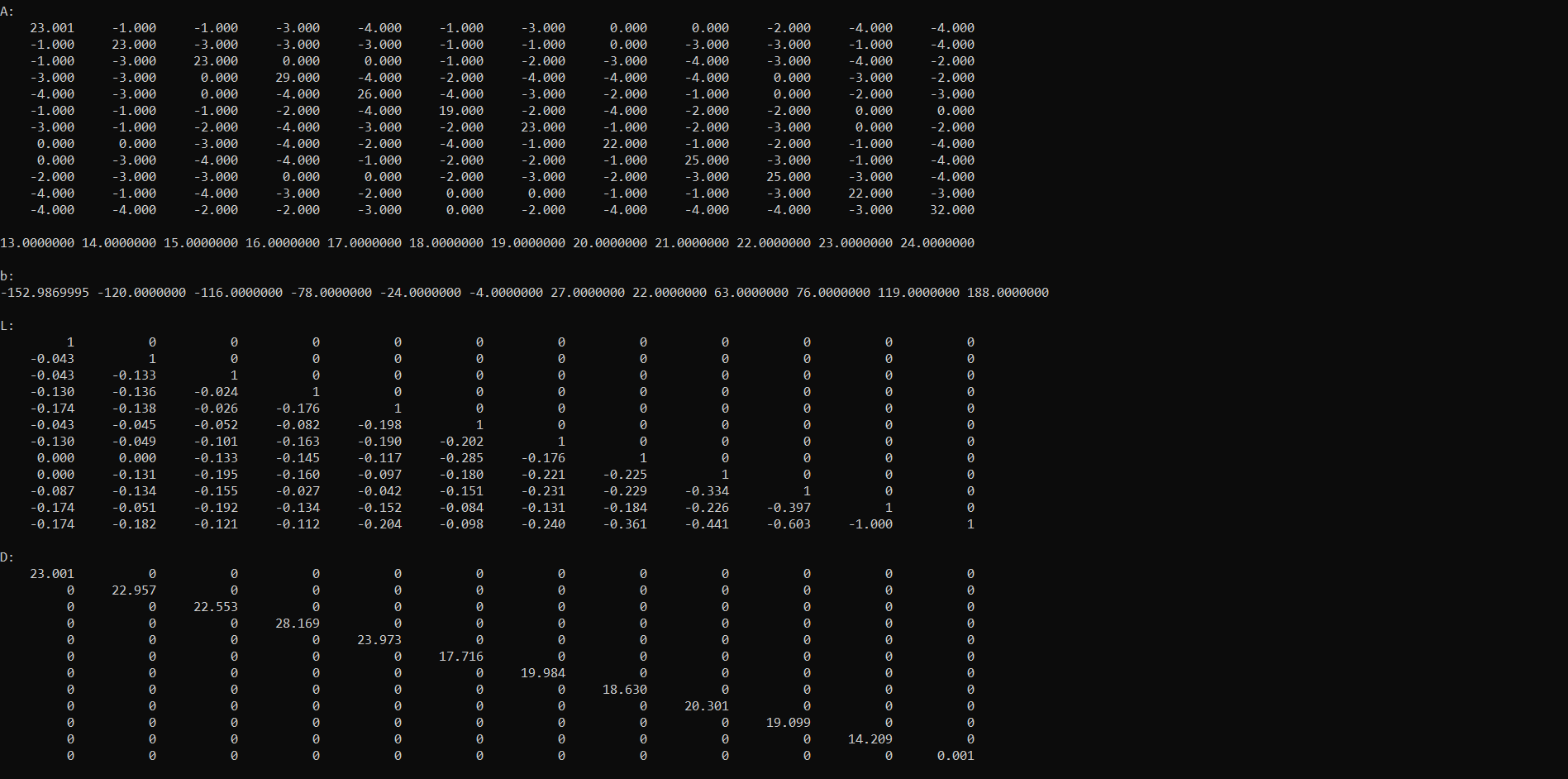
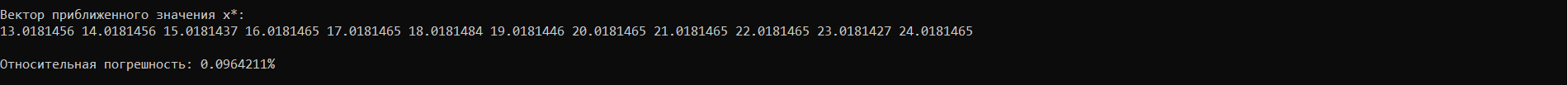
Fault(x, b);

}

**Выходные данные**

1. При k=0



1. При k=3

**Выводы**

При k=3 погрешность получилась в разы больше, чем при k=0, связано это с тем, что при k=3 происходит потеря значащих цифр.