**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский университет науки и технологий"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Численные методы

**Отчет по лабораторной работе № 3**

**Тема:** «Прямые методы решения систем линейных алгебраических уравнений»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-357 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Акмурзин М.Э. |  |  |  |
| Принял | Гайнетдинова А.А. |  |  |  |

**Уфа 2023**

**Цель:** получить навык численного решения систем линейных алгебраических уравнений (СЛАУ) с использованием различных прямых методов.

**Теоретический материал**

***Задача 1.***

**Метод Гаусса с выбором главного элемента.**

Наиболее известным из точных методов решения систем линейных уравнений является метод исключения Гаусса. В предположении, что , первое уравнение системы

делим на коэффициент , в результате получаем уравнение

Затем из каждого из остальных уравнений вычитается первое уравнение, умноженное на соответствующий коэффициент . В результате эти уравнения преобразуются к виду

Первое неизвестное оказалось исключенным из всех уравнений, кроме первого. Далее в предположении, что , делим второе уравнение на коэффициент и исключаем неизвестное из всех уравнений, начиная со второго и т.д. В результате последовательного исключения неизвестных система уравнений преобразуется в систему уравнений с треугольной матрицей

Совокупность проведенных вычислений называется *прямым ходом метода Гаусса*.

Из -го уравнения системы определяем , из -го – и т.д. до . Совокупность таких вычислений называют *обратным ходом метода Гаусса*.

Чтобы избежать катастрофического влияния вычислительной погрешности, применяют *метод Гаусса с выбором главного элемента*. Его отличие от описанной выше схемы метода Гаусса состоит в следующем. Пусть по ходу исключения неизвестных получена система уравнений

Найдем такое, что и переобозначим и ; далее произведем исключение неизвестной из всех уравнений, начиная с -го. Такое переобозначение приводит к изменению порядка исключения неизвестных и во многих случаях существенно уменьшает чувствительность решения к погрешностям округления при вычислениях.

**Рациональная интерполяция.**

При заданных приближение к ищется в виде

Коэффициенты находятся из совокупности соотношений , которые можно записать в виде

Уравнения (1) образуют систему линейных алгебраических уравнений относительно неизвестных.

***Задача 2.***

Пусть – данная матрица, а и – соответственно нижняя (левая) и верхняя (правая) треугольные матрицы. Справедливо следующее утверждение.

**Теорема**. Если все главные миноры квадратной матрицы отличны от нуля, то существуют такие нижняя и верхняя треугольные матрицы, что . Если элементы диагонали одной из матриц или фиксированы (ненулевые), то такое разложение единственно.

Формулы в случае фиксирования диагонали нижней треугольной матрицы :

***Задача 3.***

**Метод квадратного корня.**

Пусть – данная симметричная матрица, т.е. . Будем строить её представление в виде .

Матрица может быть определена совокупностью формул

**Метод наименьших квадратов.**

Если вещественные функции заданы таблично, т.е. на конечном множестве точек, то их скалярное произведение определяется формулой

где – полное число узлов таблицы. Тогда условие наилучшего среднеквадратичного приближения примет вид

Выберем линейную аппроксимацию

с числом членов . Тогда коэффициенты аппроксимации находятся из уравнений, которые получаются, подставляя обобщённый многочлен в условие наилучшего среднеквадратичного приближения и приравнивая нулю производные по коэффициентам. Описанный способ нахождения аппроксимации называется *методом наименьших квадратов*.

***Задача 4.***

Дана система пятиточечных уравнений:

Text, letter

Description automatically generated

Запишем алгоритм правой прогонки для системы (1)-(5) в следующем виде:

Text, letter

Description automatically generated

Получили решение СЛАУ с пятидиагональной матрицей.

**Индивидуальное задание**

***Задача 1. (2 балла)***

1. Написать вычислительную программу на языке программирования C++ для решения СЛАУ методом Гаусса с выбором ведущего элемента.
2. С использованием написанной программы решить задачу о рациональной интерполяции: выполнить приближение функции , заданной таблично, рациональной функцией вида

где и – многочлены степени и , соответственно. При этом требуется также определить значения и .

1. Построить график интерполирующей функции и исходных данных.

***Задача 2. (3 балла)***

1. Написать вычислительную программу на языке программирования C++ для решения СЛАУ методом LU-разложения. Задачи 1 для квадратурной формулы Симпсона.
2. Выполнить п. 2), 3) Задачи 1.

***Задача 3. (3 балла)***

1. Написать вычислительную программу на языке программирования C++ для решения СЛАУ с симметричной матрицей методом квадратного корня.
2. С использованием написанной программы решить задачу об аппроксимации функции из первой лабораторной работы, заданной на равномерной сетке из узлов, многочленами степени с использованием метода наименьших квадратов.
3. Построить графики аппроксимирующих многочленов и исходных данных.
4. Определить степень многочлена, обеспечивающего наилучшее приближение (соответствующее наименьшему значению суммы квадратов отклонений значений многочлена в узлах сетки от исходных данных).

***Задача 4. (2 балла)***

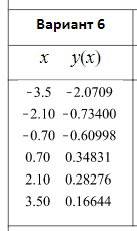
1. Написать вычислительную программу на языке программирования C++ для решения методом прогонки СЛАУ с 5-диагональной матрицей следующего вида:

Scatter chart

Description automatically generated with medium confidence

1. Для отладки программы написать генератор случайных вещественных матриц данного вида с диагональным преобладанием.

Вариант:



**Практическая часть**

***Задача 1.***

Была написана программа на языке для решения СЛАУ методом Гаусса с выбором ведущего элемента. С использованием написанной программы была решена задача о рациональной интерполяции.

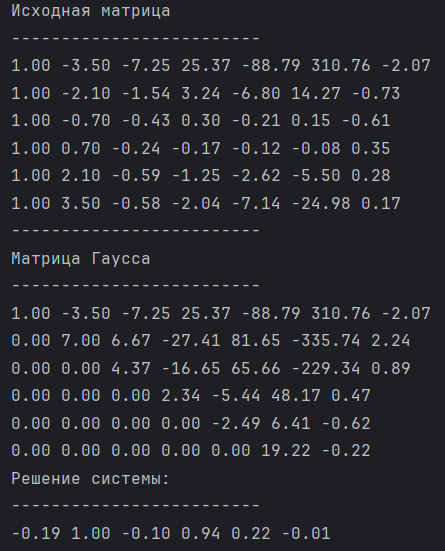


Рисунок 1. Результаты решения СЛАУ для рациональной интерполяции методом Гаусса с выбором ведущего элемента.

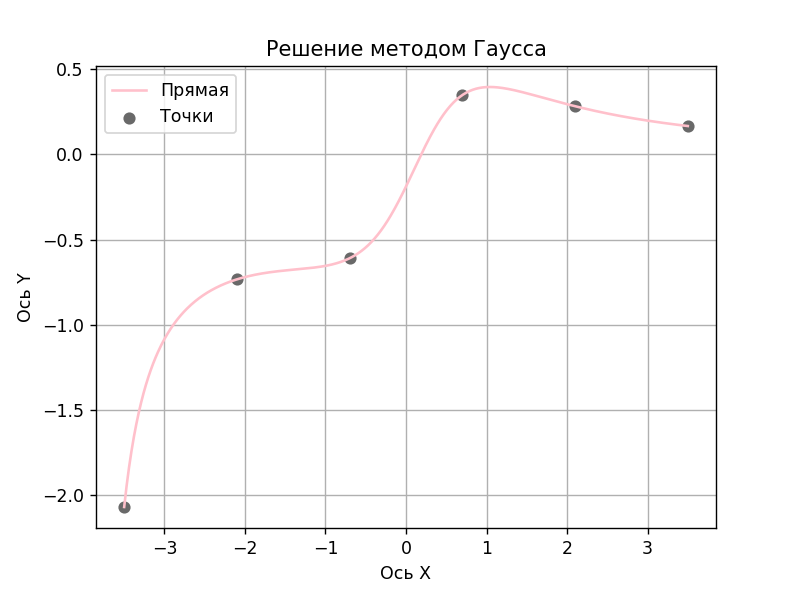


Рисунок 2. График интерполирующей функции и исходных данных для задачи №1.

***Задача 2.***

Была написана программа для решения СЛАУ методом LU-разложения. С использованием написанной программы была решена задача о рациональной интерполяции.

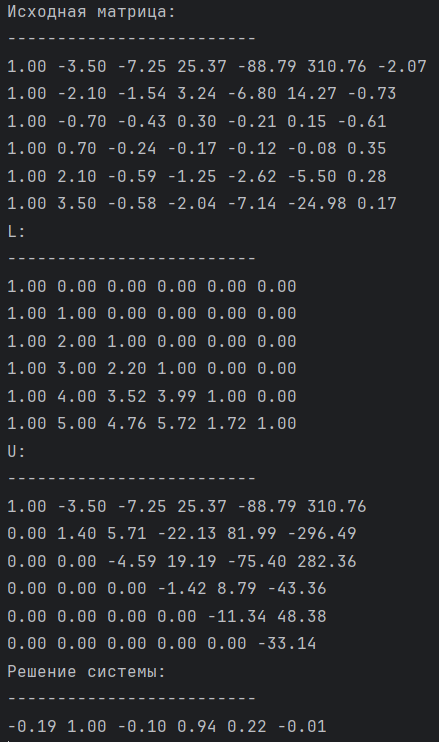
******

Рисунок 3. Результаты решения СЛАУ для рациональной интерполяции методом LU-разложения.

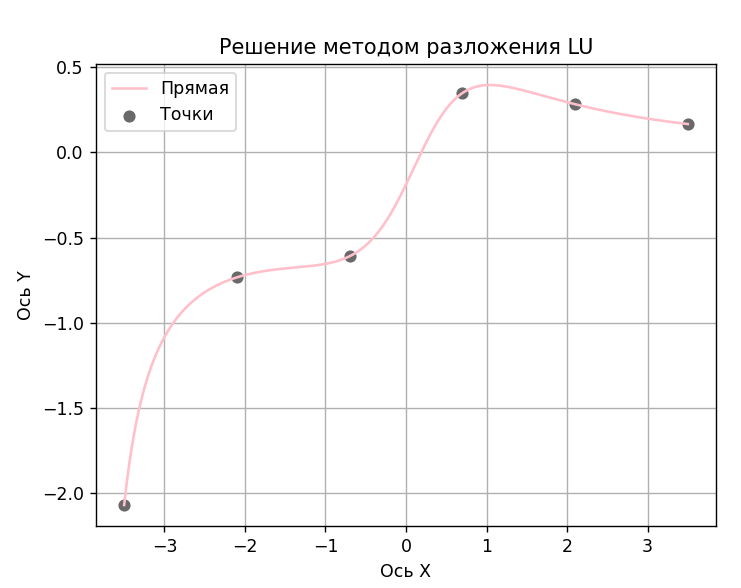


Рисунок 4. График интерполирующей функции и исходных данных для задачи №2.

***Задача 3.***

Была написать программа для решения СЛАУ с симметричной матрицей методом квадратного корня.

С использованием написанной программы решить задачу об аппроксимации функции из первой лабораторной работы, заданной на равномерной сетке из узлов, многочленами степени с использованием метода наименьших квадратов.

Была определена степень многочлена , обеспечивающего наилучшее приближение (соответствующее наименьшему значению суммы квадратов отклонений значений многочлена в узлах сетки от исходных данных).

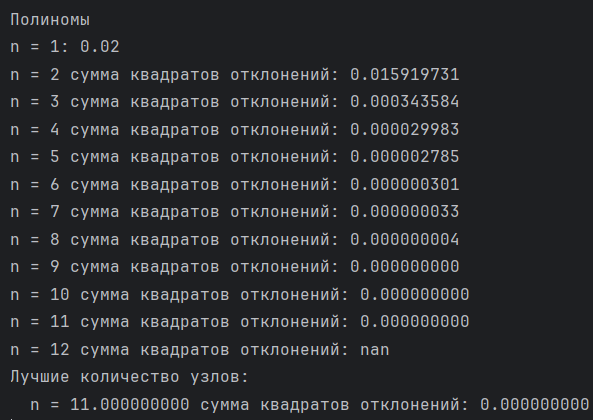


Рисунок 5. Значения суммы квадратов отклонений значений многочлена в узлах сетки от исходных данных для .

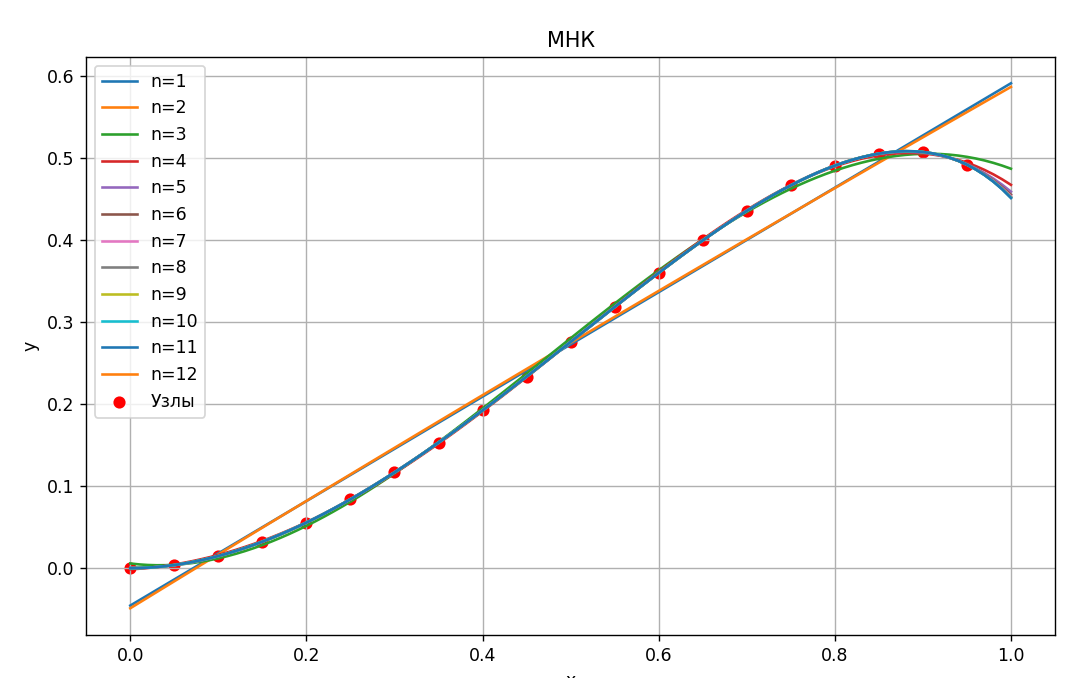
******

Рисунок 6. Графики аппроксимирующих многочленов и исходных данных.

***Задача 4.***

1. Была написана программа для решения методом прогонки СЛАУ с 5-диагональной матрицей. Кроме того, для отладки программы был написан генератор случайных вещественных матриц данного вида с диагональным преобладанием.

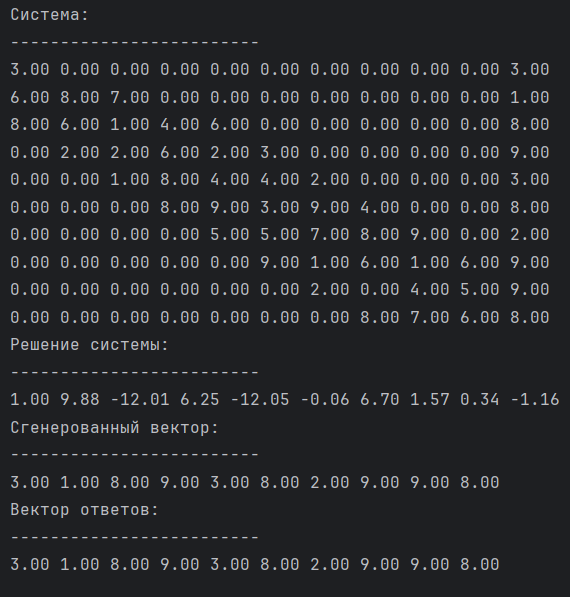


Рисунок 7. Результаты работы метода пяти диагональной прогонки.

Как можно видеть из рисунка 7, программа верно находит решение СЛАУ с пяти диагональной матрицей.

**Вывод**

В ходе проделанной лабораторной работы был изучен теоретический материал необходимый для решения поставленных задач по численному решению систем линейных алгебраических уравнений с использованием различных прямых методов и получен навык проведения вычислительного эксперимента, направленного на их решение.

Для каждой поставленной задачи написана вычислительная программа на языке программирования С++, выполняющая необходимые построения и расчеты по нахождению решения системы линейных алгебраических уравнений.

**Список использованной литературы**

1. Бахвалов Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. — 6-е изд. — М.: БИНОМ. Лаборатория знаний, 2008. — 636 с.: ил.

2. Вержбицкий В. М. Основы численных методов. / В. М. Вержбицкий. — М.: Высш. шк., 2002. — 840 с.: ил.

3. Калиткин Н. Н. Численные методы. — М: «Наука», 1978. — 512 с.

4. Самарский А. А. Методы решения сеточных уравнений. / А. А. Самарский, Е. С. Николаев. — М.: «Наука», 1978. — 592 с.: ил.

**Приложение**

Main.cpp

#include "task.h"

int main(){

SetConsoleOutputCP(CP\_UTF8);

Task3();

return 0;

}

Task.h

#include <iostream>

#include <vector>

#include <windows.h>

#include <iomanip>

#include <fstream>

#include <random>

#include <math.h>

void Task3();

using namespace std;

void print\_matrix(vector<vector<double>> A, bool type);

void print\_vector(vector<double> A, bool type);

vector<vector<double>> set\_rand\_matrix(int n);

vector<double> set\_rand\_vector(int n);

vector<vector<double>> set\_matrix(int n);

void set\_system(vector<vector<double>> &matrix\_coef, vector<double> &d);

vector<double> set\_vector(int n);

int get\_n();

const vector<double> x = {-3.5, -2.1, -0.7, 0.7, 2.1, 3.5};

const vector<double> y = {-2.0709, -0.73400, -0.60998, 0.34831, 0.28276, 0.16644};

namespace task1 {

vector<double> method\_gaussa(vector<vector<double>> matrix\_coef);

void main\_task1();

vector<double> get\_coef();

void point\_1();

double rational\_interpoletion(vector<double> &coef, double x);

void write\_node\_file();

void point\_2\_3(char \*name, vector<double> &coef);

}

namespace task2 {

void main\_task2();

void point\_1();

void LU(vector<vector<double>> A, vector<vector<double>> &L, vector<vector<double>> &U, int n);

vector<double> solution\_LU(vector<vector<double>> &L, vector<vector<double>> &U, vector<double> &d);

vector<double> get\_coef();

}

namespace task4 {

const int N = 9;

void main\_task4();

vector<double> FifthDiag(vector<vector<double>>& M, vector<double>& ansvec);

void set\_rand\_value(int n, vector<vector<double>> &M, vector<double> &b);

vector<double> check\_get\_vector(int n, const vector<vector<double>> &M, const vector<double> &ANS);

}

Task1.cpp

#include "task.h"

#define N\_accuracity 2

using namespace task1;

void print\_matrix(vector<vector<double>> A, bool type) {

if (type) {

cout << fixed;

cout.precision(N\_accuracity);

}

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

for (int i = 0; i < A.size(); i++) {

for (int j = 0; j < A[i].size(); j++) {

cout << A[i][j] << " ";

}

cout << endl;

}

}

void print\_vector(vector<double> A, bool type) {

if (type) {

cout << fixed;

cout.precision(N\_accuracity);

}

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

for (int i = 0; i < A.size(); i++) {

cout << A[i] << " ";

}

cout << endl;

}

vector<double> task1::method\_gaussa(vector<vector<double>> matrix\_coef) {

int n = matrix\_coef.size();

vector<double> x\_answer;

x\_answer.resize(n);

cout << "Исходная матрица " << endl;

print\_matrix(matrix\_coef, 1);

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

double curr\_max = matrix\_coef[i][i];

int index\_max = i;

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

if (abs(matrix\_coef[j][i]) > abs(curr\_max)) {

curr\_max = matrix\_coef[j][i];

index\_max = j;

}

}

if (i != index\_max) {

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

double new\_temp = matrix\_coef[i][j];

matrix\_coef[i][j] = matrix\_coef[index\_max][j];

matrix\_coef[index\_max][j] = new\_temp;

}

}

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

double mu = matrix\_coef[a][i] / matrix\_coef[i][i];

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

matrix\_coef[a][b] -= (matrix\_coef[i][b] \* mu);

}

}

}

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

cout << "Матрица Гаусса" << endl;

print\_matrix(matrix\_coef, 1);

x\_answer[n - 1] = matrix\_coef[n - 1][n] / matrix\_coef[n - 1][n - 1];

for (int i = n - 2; i >= 0; i--) {

x\_answer[i] = matrix\_coef[i][n] / matrix\_coef[i][i];

int raz = abs(i - n + 1);

for (int j = 1; j <= raz; j++) {

x\_answer[i] -= ((matrix\_coef[i][i + j] \* x\_answer[i + j]) / matrix\_coef[i][i]);

}

}

return x\_answer;

}

void task1::main\_task1() {

//point\_1();

vector<double> coef = get\_coef();

point\_2\_3("line\_task1.txt", coef);

system("python D:\\5sem\\numerical\\practicle\\Lab3\\src\\plot\\task1.py");

}

void task1::point\_1() {

vector<vector<double >> matrix\_coef;

vector<double> y;

int count = get\_n();

matrix\_coef = set\_matrix(count);

y = set\_vector(count);

set\_system(matrix\_coef, y);

vector<double> x\_answer = method\_gaussa(matrix\_coef);

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

cout << "Решение сиcтемы " << endl;

print\_vector(x\_answer, true);

}

void set\_system(vector<vector<double>> &matrix\_coef, vector<double> &d) {

int n = matrix\_coef.size();

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

matrix\_coef[i].push\_back(d[i]);

}

}

vector<double> set\_vector(int n) {

vector<double> y;

cout << "Введите у\_i: " << endl;

double y\_answer;

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

cin >> y\_answer;

y.push\_back(y\_answer);

}

return y;

}

vector<vector<double>> set\_matrix(int count) {

vector<vector<double>> matrix\_coef;

matrix\_coef.resize(count);

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

matrix\_coef[i].resize(count);

}

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

cout << "Введите коэфиценты " << i + 1 << " уравнения " << endl;

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

cin >> matrix\_coef[i][j];

}

}

return matrix\_coef;

}

int get\_n() {

int count;

cout << "Введите количество уравнений в системе: " << endl;

cin >> count;

return count;

}

//vector<double> &initial\_polynom(int n,) {

//

//}

double task1::rational\_interpoletion(vector<double> &coef, double x) {

return (coef[0] + x \* coef[1]) /

(1 + x \* coef[2] + x \* x \* coef[3] + x \* x \* x \* coef[4] + x \* x \* x \* x \* coef[5]);

}

vector<double> task1::get\_coef() {

vector<vector<double>> Matrix(6);

for (int i = 0; i < x.size(); i++) {

Matrix[i].push\_back(1);// a0

Matrix[i].push\_back(x[i]);// a1 \* x

Matrix[i].push\_back(-x[i] \* y[i]); // -b1 \* x \* y

Matrix[i].push\_back(-x[i] \* x[i] \* y[i]);// -b2 \* x (^2) \* y

Matrix[i].push\_back(-x[i] \* x[i] \* x[i] \* y[i]);// -b3 \* x(^3) \* y

Matrix[i].push\_back(-x[i] \* x[i] \* x[i] \* x[i] \* y[i]);// -b4 \* x(^4) \* y

Matrix[i].push\_back(y[i]); // = yi

}

vector<double> coef\_inter = method\_gaussa(Matrix);

cout << "Решение системы:\n";

print\_vector(coef\_inter, true);

return coef\_inter;

}

void task1::write\_node\_file() {

ofstream out\_plot;

out\_plot.open("node.txt");

for (int i = 0; i < 6; i++) {

out\_plot << x[i] << " " << y[i] << endl;

}

out\_plot.close();

}

void task1::point\_2\_3(char \*name, vector<double> &coef) {

write\_node\_file();

ofstream outPlot;

outPlot.open(name);

double x\_cur = -3.5, h = 0.01;

while (x\_cur <= 3.5) {

outPlot << x\_cur << " " << rational\_interpoletion(coef, x\_cur) << endl;

x\_cur += h;

}

outPlot.close();

}

Task2.cpp

#include "task.h"

using namespace task2;

void task2::main\_task2() {

//point\_1();

vector<double> coef = get\_coef();

task1::point\_2\_3("line\_task2.txt",coef);

system("python D:\\5sem\\numerical\\practice\\Lab3\\src\\plot\\task2.py");

}

void task2::point\_1() {

int n = get\_n();

vector<vector<double>> A = set\_matrix(n), A\_print;

A\_print = A;

vector<double> d = set\_vector(n), x;

set\_system(A\_print, d);

vector<vector<double>> L(n, vector<double>(n, 0)), U(n, vector<double>(n, 0));

LU(A, L, U, n);

cout << "L:\n";

print\_matrix(L,1);

cout << "U:\n";

print\_matrix(L,1);

x = solution\_LU(L, U, d);

cout << "Решение системы: \n";

print\_vector(x, 1);

}

void task2::LU(vector<vector<double>> A, vector<vector<double>> &L,

vector<vector<double>> &U, int n) {

U = A;

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

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

L[j][i] = U[j][i] / U[i][i];

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

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

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

L[j][i] = U[j][i] / U[i][i];

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

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

U[i][j] = U[i][j] - L[i][k - 1] \* U[k - 1][j];

}

}

vector<double> task2::solution\_LU(vector<vector<double>> &L, vector<vector<double>> &U, vector<double> &d) {

int n = L.size();

vector<double> x(n, 0), y(n, 0);

y[0] = d[0] / L[0][0];

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

y[i] = d[i] / L[i][i];

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

y[i] -= ((L[i][j] \* y[j]) / L[i][i]);

}

}

x[n - 1] = y[n - 1] / U[n - 1][n - 1];

for (int i = n - 2; i >= 0; i--) {

x[i] = y[i] / U[i][i];

int raz = abs(i - n + 1);

for (int j = 1; j <= raz; j++) {

x[i] -= ((U[i][i + j] \* x[i + j]) / U[i][i]);

}

}

return x;

}

vector<vector<double>> set\_rand\_matrix(int n) {

srand(time(0));

vector<vector<double>> matrix(n, vector<double>(n, 0));

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

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

matrix[i][j] = -10 + rand() % 20;

}

}

return matrix;

}

vector<double> set\_rand\_vector(int n) {

vector<double> d(n);

srand(time(0));

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

d[i] = -10 + rand() % 20;

}

return d;

}

vector<double> task2::get\_coef() {

vector<vector<double>> Matrix(6),L(6, vector<double>(6, 0)), U(6, vector<double>(6, 0)),Matrix\_print;

vector<double> d;

for (int i = 0; i < x.size(); i++) {

Matrix[i].push\_back(1);// a0

Matrix[i].push\_back(x[i]);// a1 \* x

Matrix[i].push\_back(-x[i] \* y[i]); // -b1 \* x \* y

Matrix[i].push\_back(-x[i] \* x[i] \* y[i]);// -b2 \* x (^2) \* y

Matrix[i].push\_back(-x[i] \* x[i] \* x[i] \* y[i]);// -b3 \* x(^3) \* y

Matrix[i].push\_back(-x[i] \* x[i] \* x[i] \* x[i] \* y[i]);// -b4 \* x(^4) \* y

d.push\_back(y[i]); // = yi

}

cout << "Исходная матрица:\n";

Matrix\_print=Matrix;

set\_system(Matrix\_print,d);

print\_matrix(Matrix\_print,1);

LU(Matrix,L,U,6);

cout << "L:\n";

print\_matrix(L,1);

cout << "U:\n";

print\_matrix(U,1);

vector<double> coef= solution\_LU(L,U,d);

cout << "Решение системы:\n";

print\_vector(coef,1);

return coef;

}

Task3.cpp

#include "task.h"

std::vector<double> SquareRootMethod(const std::vector<std::vector<double>>& matrix) {

int dim = matrix.size();

std::vector<std::vector<double>> U(dim);

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

U[i].resize(dim);

U[i][i] = matrix[i][i];

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

U[i][i] -= U[k][i] \* U[k][i];

}

U[i][i] = sqrt(U[i][i]);

for (int j = 1; j < dim; ++j) {

if (j > i) {

U[i][j] = matrix[i][j];

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

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

}

U[i][j] /= U[i][i];

}

}

}

std::cout << "U:" << std::endl;

print\_matrix(U,1);

std::vector<double> y(dim);

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

y[i] = matrix[i][dim];

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

y[i] -= U[k][i] \* y[k];

}

y[i] /= U[i][i];

}

std::vector<double> x(dim);

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

x[i] = y[i];

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

x[i] -= U[i][k] \* x[k];

}

x[i] /= U[i][i];

}

return x;

}

double f(double x) {

return x\*x\*acos(0.9\*x);

}

void Task3() {

std::cout << "===== Task 3 =====" << std::endl;

double a = 0;

double b = 1;

int n = 20;

std::vector<double> x\_i;

std::vector<double> y\_i;

double h = (b - a) / n;

ofstream node("node\_task3.txt");

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

double x = a + i \* h;

x\_i.push\_back(x);

y\_i.push\_back(f(x));

node << x << " " << f(x) << endl;

}

node.close();

std::vector<double> answer;

std::vector<std::vector<double>> answers;

std::vector<std::vector<double>> matrix;

//Polynoms with max degree 1...12

for (int m = 2; m < 14; ++m) {

matrix.clear();

matrix.resize(m);

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

matrix[i].resize(m + 1);

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

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

matrix[i][j] += pow(x\_i[k], i + j);

}

}

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

matrix[i][m] += y\_i[k] \* pow(x\_i[k], i);

}

}

std::cout << "Система:" << std::endl;

print\_matrix(matrix,1);

answer = SquareRootMethod(matrix);

std::cout << "Решение системы: " << std::endl;

print\_vector(answer,1);

answers.push\_back(answer);

}

//Graphics

int N = 100;

h = (b - a) / N;

std::ofstream task3("task3.txt");

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

double x = a + i \* h;

double f = 0;

int count = 0;

task3 << x << " ";

for (int k = 0; k < 12; ++k) {

f = 0;

for (int j = 0; j < k + 2; ++j) {

f += answers[k][j] \* pow(x, j);

}

task3 << f << " ";

}

task3 << std::endl;

}

task3.close();

std::cout << "Полиномы" << std::endl;

std::vector<double> sum\_of\_squares\_of\_deviations(12);

for (int i = 0; i < x\_i.size(); ++i) {

for (int k = 0; k < 12; ++k) {

double f = 0;

for (int j = 0; j < k + 2; ++j) {

f += answers[k][j] \* pow(x\_i[i], j);

}

sum\_of\_squares\_of\_deviations[k] += pow(abs(f - y\_i[i]), 2);

}

}

double min\_value = sum\_of\_squares\_of\_deviations[0];

double min\_n = 0;

std::cout << "n = " << 1 << ": " << sum\_of\_squares\_of\_deviations[0] << std::endl;

for (int i = 1; i < 12; ++i) {

if (sum\_of\_squares\_of\_deviations[i] < min\_value) {

min\_value = sum\_of\_squares\_of\_deviations[i];

min\_n = i;

}

std::cout << "n = " << (i + 1) << " сумма квадратов отклонений: " << std::setprecision(9) << sum\_of\_squares\_of\_deviations[i] << std::endl;

}

std::cout << "Лучшие количество узлов: " << std::endl;

std::cout << " n = " << (min\_n + 1) << " сумма квадратов отклонений: " << min\_value << std::endl;

system(" python D:\\5sem\\numerical\\practice\\labs\\Lab3\\src\\plot\\task3.py");

}

Task4.cpp

#include "task.h"

using namespace task4;

void task4::main\_task4() {

int n = 10;

vector<vector<double>> M(n, vector<double>(n, 0)), M\_print;

vector<double> b;

set\_rand\_value(n, M, b);

cout << "Система:\n";

M\_print = M;

set\_system(M\_print, b);

print\_matrix(M\_print, 1);

vector<double> ANS = FifthDiag(M, b);

cout << "Решение системы:\n";

print\_vector(ANS, 1);

vector<double> check = check\_get\_vector(n, M, ANS);

cout << "Сгенерованный вектор:\n";

print\_vector(check, 1);

cout << "Вектор ответов:\n";

print\_vector(b, 1);

}

vector<double> task4::check\_get\_vector(int n, const vector<vector<double>> &M, const vector<double> &ANS) {

vector<double> check(n, 0);

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

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

check[i] += ANS[j] \* M[i][j];

}

}

return check;

}

void task4::set\_rand\_value(int n, vector<vector<double>> &M, vector<double> &b) {

srand(time(0));

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

b.push\_back(rand() % 10);

}

int j = 2;

for (int i = 2; i < n - 2; i++, j++) {

M[i][j - 2] = rand() % 10;

M[i][j - 1] = rand() % 10;

M[i][j] = rand() % 10;

M[i][j + 1] = rand() % 10;

M[i][j + 2] = rand() % 10;

}

j = 0;

for (int i = 0; i < 2; i++, j++) {

if (i == 0) {

// row 1

M[i][j] = rand() % 10;

M[i][j + 1] = rand() % 10;

M[i][j + 2] = rand() % 10;

// row n

M[n - 1][n - 3] = rand() % 10;

M[n - 1][n - 2] = rand() % 10;

M[n - 1][n - 1] = rand() % 10;

} else if (i == 1) {

//row 2

M[i][j - 1] = rand() % 10;

M[i][j] = rand() % 10;

M[i][j + 1] = rand() % 10;

M[i][j + 2] = rand() % 10;

//row n-1

M[n - 2][n - 4] = rand() % 10;

M[n - 2][n - 3] = rand() % 10;

M[n - 2][n - 2] = rand() % 10;

M[n - 2][n - 1] = rand() % 10;

}

}

}

vector<double> task4::FifthDiag(vector<vector<double>> &M, vector<double> &ansvec) {

int n = M.size();

vector<double> a(n), b(n), c(n), d(n), e(n), alpha(n), betta(n), gamma(n), X(n);

a[0] = 0;

b[0] = 0;

a[1] = 0;

e[n - 1] = 0;

d[n - 1] = 0;

//1 rows

c[0] = M[0][0];

d[0] = M[0][1];

e[0] = M[0][2];

//2 rows

b[1] = M[1][0];

c[1] = M[1][1];

d[1] = M[1][2];

e[1] = M[1][3];

//3-n-2 rows

int j = 0;

for (int i = 2; i < n - 2; i++, j++) {

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

b[i] = M[i][j + 1];

c[i] = M[i][j + 2];

d[i] = M[i][j + 3];

e[i] = M[i][j + 4];

}

//n-2 rows

a[n - 2] = M[n - 2][n - 4];

b[n - 2] = M[n - 2][n - 3];

c[n - 2] = M[n - 2][n - 2];

d[n - 2] = M[n - 2][n - 1];

//n-1 line

a[n - 1] = M[n - 1][n - 3];

b[n - 1] = M[n - 1][n - 2];

c[n - 1] = M[n - 1][n - 1];

alpha[0] = -d[0] / c[0];

betta[0] = -e[0] / c[0];

gamma[0] = ansvec[0] / c[0];

alpha[1] = -(b[1] \* betta[0] + d[1]) / (c[1] + b[1] \* alpha[0]);

betta[1] = -e[1] / (c[1] + b[1] \* alpha[0]);

gamma[1] = (ansvec[1] - b[1] \* gamma[0]) / (c[1] + b[1] \* alpha[0]);

double delitel = 0;

double d1 = 0;

double d2 = 0;

double d3 = 0;

for (int i = 2; i < n; i++) {

delitel = ((betta[i - 2] + alpha[i - 1] \* alpha[i - 2]) \* a[i] + alpha[i - 1] \* b[i] + c[i]);

d1 = (a[i] \* alpha[i - 2] \* betta[i - 1] + b[i] \* betta[i - 1] + d[i]);

d2 = (e[i]);

d3 = (ansvec[i] - (a[i] \* alpha[i - 2] \* gamma[i - 1] + a[i] \* gamma[i - 2] + b[i] \* gamma[i - 1]));

alpha[i] =

-d1

/

delitel;

betta[i] =

-d2

/

delitel;

gamma[i] =

d3

/

delitel;

}

X[n - 1] = gamma[n - 1];

X[n - 2] = alpha[n - 2] \* X[n - 1] + gamma[n - 2];

for (int i = n - 3; i >= 0; i--) {

X[i] = betta[i] \* X[i + 2] + alpha[i] \* X[i + 1] + gamma[i];

}

return X;

}

Task1.py

import matplotlib.pyplot as plt

import pandas as pd

def plot\_point():

point = pd.read\_csv('D:/5sem/numerical/practice/labs/Lab3/cmake-build-debug/node.txt', sep=' ', names=['X', 'Y'])

line = pd.read\_csv('D:/5sem/numerical/practice/labs/Lab3/cmake-build-debug/line\_task1.txt', sep=' ', names=['X', 'Y'])

plt.plot(line['X'], line['Y'], color='pink', label='Прямая')

plt.scatter(point['X'], point['Y'], color='dimgray', label='Точки')

plt.grid(True)

plt.title('Решение методом Гаусса')

plt.xlabel('Ось X')

plt.ylabel('Ось Y')

plt.legend()

plt.show()

plot\_point()

Task2.py

import matplotlib.pyplot as plt

import pandas as pd

def plot\_point():

point = pd.read\_csv('D:/5sem/numerical/practice/labs/Lab3/cmake-build-debug/node.txt', sep=' ', names=['X', 'Y'])

line = pd.read\_csv('D:/5sem/numerical/practice/labs/Lab3/cmake-build-debug/line\_task2.txt', sep=' ', names=['X', 'Y'])

plt.plot(line['X'], line['Y'], color='pink',label='Прямая')

plt.scatter(point['X'], point['Y'],color='dimgray' ,label='Точки')

plt.grid(True)

plt.title('Решение методом разложения LU')

plt.xlabel('Ось X')

plt.ylabel('Ось Y')

plt.legend()

plt.show()

plot\_point()

Task3.py

import pandas as pd

import matplotlib.pyplot as plt

file\_path = "D:/5sem/numerical/practice/labs/Lab3/cmake-build-debug/task3.txt"

point=pd.read\_csv("D:/5sem/numerical/practice/labs/Lab3/cmake-build-debug/node\_task3.txt",sep=' ', names=['X', 'Y'])

df = pd.read\_csv(file\_path, delim\_whitespace=True, header=None, names=['x'] + [f'y\_{i}' for i in range(0, 12)])

# print(df.head())

plt.figure(figsize=(10, 6))

for i in range(0, 12):

plt.plot(df['x'], df[f'y\_{i}'], label=f'n={i+1}')

plt.scatter(point["X"],point["Y"],color='red',label="Узлы")

plt.xlabel('x')

plt.ylabel('y')

plt.title('МНК')

plt.legend()

plt.grid(True)

plt.show()