**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Численные методы

**Отчет по лабораторной работе № 3**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа МКН-315 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Акмурзин М.Э. |  |  |  |
| Принял | Гайнетдинова А.А. |  |  |  |

**Уфа 2022**

**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)***

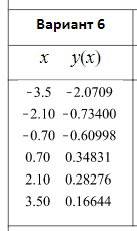
1. Написать вычислительную программу на языке программирования C++ для решения методом прогонки СЛАУ с 5-диагональной матрицей следующего вида:

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.

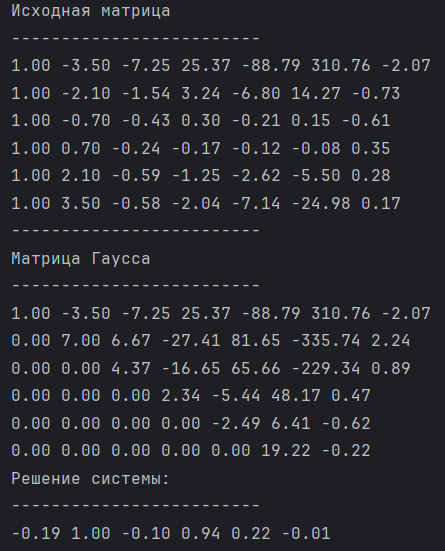


Рисунок 1. Результаты решения СЛАУ для рациональной интерполяции методом Гаусса с выбором ведущего элемента.

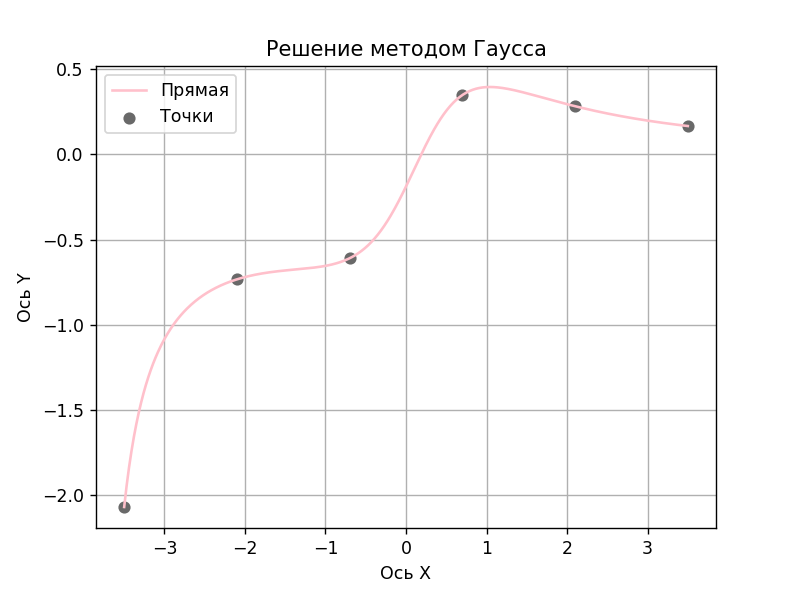


Рисунок 2. График интерполирующей функции и исходных данных для задачи №1.

***Задача 2.***

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

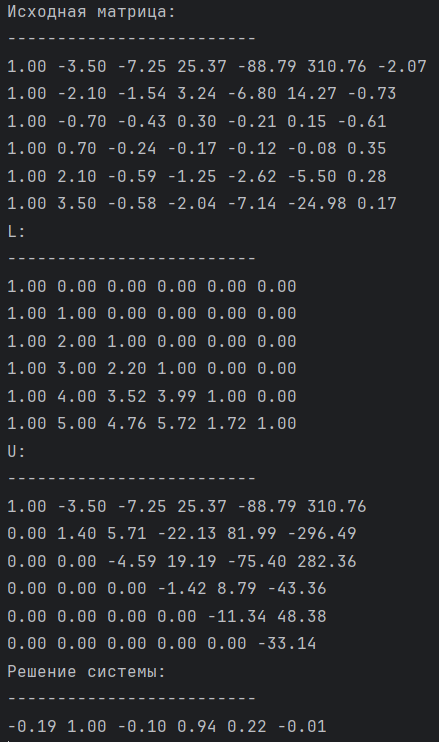
******

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

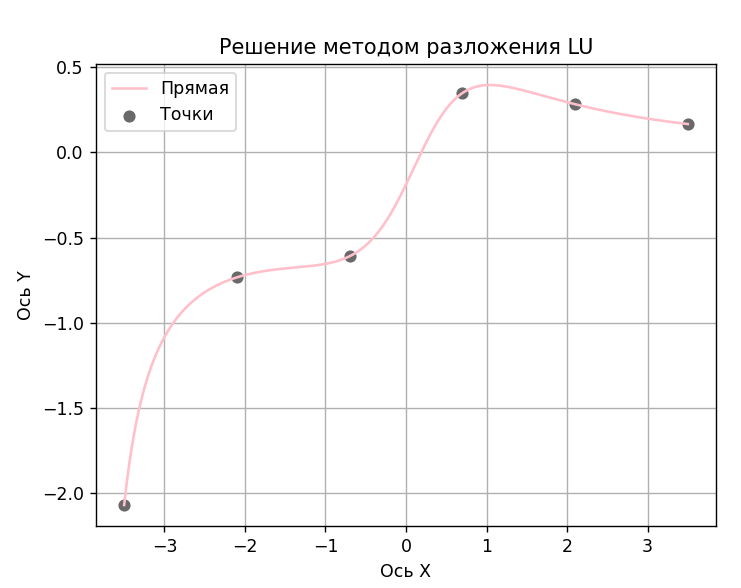


Рисунок 4. График интерполирующей функции и исходных данных для задачи №2.

***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).

***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.

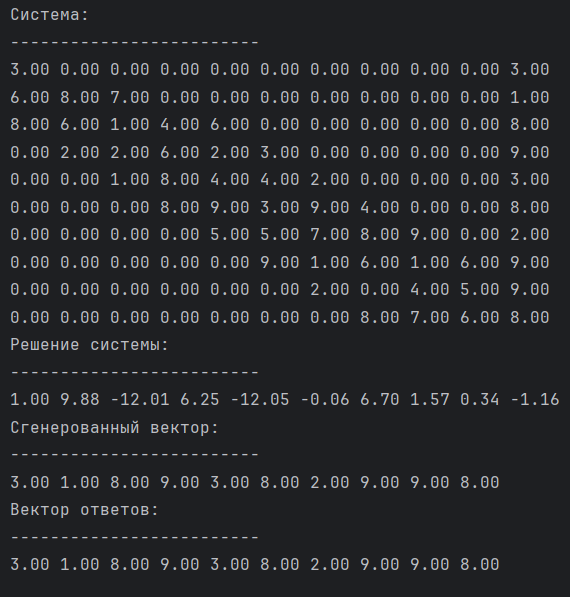


Рисунок 7. Результаты работы метода пяти диагональной прогонки.

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 <iostream>

#include "Tasks12.h"

#include "Task3.h"

#include "Task4.h"

int main() {

//Task1();

//Task2();

//Task3();

Task4();

return 0;

}

**Tasks12.h**

#pragma once

#include <vector>

#include <iostream>

#include <fstream>

#include <sstream>

#include "Methods.h"

std::vector<double> x\_i = { -3.5, -2.10, -0.70, 0.70, 2.10, 3.50 };

std::vector<double> y\_i = { -2.0709, -0.73400, -0.60998, 0.34831, 0.28276, 0.16644 };

const int dim = 6;

std::vector<std::vector<double>> A;

void InitMatrix() {

A.resize(dim);

int p = 2;

int q = 3;

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

A[i].resize(dim + 1);

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

if (j <= p) {

A[i][j] = pow(x\_i[i], j);

}

else {

A[i][j] = -y\_i[i] \* pow(x\_i[i], j - p);

}

}

//b\_0 = 1

A[i][dim] = y\_i[i];

}

}

std::vector<double> Gauss(std::vector<std::vector<double>> matrix) {

int dim = matrix.size();

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

//выбор главного элемента

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

double max = 0;

double m\_index = 0;

for (int m = k; m < dim; ++m) {

if (abs(matrix[m][k]) > max) {

max = abs(matrix[m][k]);

m\_index = m;

}

}

if (max == 0) {

std::cout << "There is no single solution" << std::endl;

return { -1 };

}

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

double temp = matrix[k][i];

matrix[k][i] = matrix[m\_index][i];

matrix[m\_index][i] = temp;

}

}

//исключение

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

double coeff = -matrix[j][i] / matrix[i][i];

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

matrix[j][k] += coeff \* matrix[i][k];

}

}

}

//обратный ход

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

answer[dim - 1] = matrix[dim - 1][dim] / matrix[dim - 1][dim - 1];

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

answer[i] = matrix[i][dim];

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

answer[i] -= matrix[i][j] \* answer[j];

}

answer[i] /= matrix[i][i];

}

return answer;

}

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

int dim = matrix.size();

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

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

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

L[i].resize(dim);

U[i].resize(dim);

L[i][i] = 1;

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

if (i <= j) {

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

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

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

}

}

else {

L[i][j] = matrix[i][j];

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

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

}

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

}

}

}

std::cout << "L:" << std::endl;

PrintMatrix(L);

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

PrintMatrix(U);

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

}

}

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;

}

void Task1() {

std::cout << "Task 1" << std::endl;

InitMatrix();

std::cout << "Extended matrix of the system: " << std::endl;

PrintMatrix(A);

std::vector<double> answer = Gauss(A);

std::cout << "Gauss answer: " << std::endl;

PrintVector(answer);

int N = 100;

double a = x\_i[0];

double b = x\_i[dim - 1];

double h = (b - a) / N;

std::ofstream task1("task1.txt");

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

double x\_i = a + i \* h;

double f = (answer[0] + answer[1] \* x\_i + answer[2] \* pow(x\_i, 2)) / (1 + answer[3] \* x\_i + answer[4] \* pow(x\_i, 2) + answer[5] \* pow(x\_i, 3));

task1 << x\_i << " " << f << std::endl;

}

task1.close();

std::stringstream graphic\_info;

graphic\_info << "\"" + PATH + "task1.txt\" using 1:2 title \"Interpolation task 1\" with lines, '-' using 1:2 title \"Points\"\n";

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

graphic\_info << x\_i[i] << " " << y\_i[i] << std::endl;

}

graphic\_info << "e\n";

DrawGraphic(graphic\_info.str());

}

void Task2() {

std::cout << "Task 2" << std::endl;

InitMatrix();

std::cout << "Extended matrix of the system: " << std::endl;

PrintMatrix(A);

std::vector<double> answer = LUDecomposition(A);

std::cout << "LUDecomposition answer: " << std::endl;

PrintVector(answer);

int N = 100;

double a = x\_i[0];

double b = x\_i[dim - 1];

double h = (b - a) / N;

std::ofstream task2("task2.txt");

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

double x\_i = a + i \* h;

double f = (answer[0] + answer[1] \* x\_i + answer[2] \* pow(x\_i, 2)) / (1 + answer[3] \* x\_i + answer[4] \* pow(x\_i, 2) + answer[5] \* pow(x\_i, 3));

task2 << x\_i << " " << f << std::endl;

}

task2.close();

std::stringstream graphic\_info;

graphic\_info << "\"" + PATH + "task2.txt\" using 1:2 title \"Interpolation task 2\" with lines, '-' using 1:2 title \"Points\"\n";

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

graphic\_info << x\_i[i] << " " << y\_i[i] << std::endl;

}

graphic\_info << "e\n";

DrawGraphic(graphic\_info.str());

}

**Task3.h**

#pragma once

#include <vector>

#include <iostream>

#include "Methods.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;

PrintMatrix(U);

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;

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

double x = a + i \* h;

x\_i.push\_back(x);

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

}

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 << "Matrix:" << std::endl;

PrintMatrix(matrix);

answer = SquareRootMethod(matrix);

std::cout << "Square root method answer: " << std::endl;

PrintVector(answer);

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::stringstream graphic\_info;

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

graphic\_info << "\"" << PATH << "task3.txt\" using 1:" << i + 2 << " title \"Approximation with n = " << i + 1 << "\" with lines,";

}

graphic\_info << " '-' using 1:2 title \"Points\"\n";

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

graphic\_info << x\_i[i] << " " << y\_i[i] << std::endl;

}

graphic\_info << "e\n";

DrawGraphic(graphic\_info.str());

//Find polynom degree with best approximation

std::cout << "Polynom degree with best approximation" << 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: " << 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) << " Sum of squares of deviations: " << sum\_of\_squares\_of\_deviations[i] << std::endl;

}

std::cout << "Best: " << std::endl;

std::cout << " n = " << (min\_n + 1) << " Sum of squares of deviations: " << min\_value << std::endl;

std::cout << "===== Polynom degree with best approximation =====" << std::endl;

}

**Task4.h**

#pragma once

#include <vector>

#include "Methods.h"

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

void InitFiveDiagonalMatrix(int dim) {

matrix.resize(dim);

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

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

double sum = 0;

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

if (abs(i - j) <= 2 && i != j) {

double value = RandomDouble(5, 10);

matrix[i][j] = value;

sum += abs(value);

}

}

matrix[i][i] = 4 \* pow(-1, RandomInt(0, 1)) \* sum;

matrix[i][dim] = RandomDouble(-1, 1);

}

PrintMatrix(matrix);

}

std::vector<double> FiveDiagonalMethod(std::vector<std::vector<double>> matrix) {

int dim = matrix.size();

std::vector<double> alpha(dim + 3), beta(dim + 3), gamma(dim + 3);

alpha[1] = -matrix[0][0+1] / matrix[0][0];

beta[1] = matrix[0][0 + 2] / matrix[0][0];

gamma[1] = matrix[0][matrix.size()] / matrix[0][0];

double delta1 = matrix[1][1] - -matrix[1][0] \* alpha[1];

alpha[2] = (-matrix[1][2] - beta[1] \* -matrix[1][0]) / delta1;

gamma[2] = (matrix[1][matrix.size()] + -matrix[1][0] \* gamma[1]) / delta1;

beta[2] = matrix[1][1 + 2] / delta1;

for (int i = 2; i <= dim - 1; ++i) {

double delta = matrix[i][i] - (i<2?0:matrix[i][i - 2]) \* beta[i - 1] + alpha[i] \* ((i<2?0:matrix[i][i - 2]) \* alpha[i - 1] - (i<1?0:-matrix[i][i - 1]));

if (i <= dim - 2) {

alpha[i + 1] = ((i>matrix.size() - 2?0:-matrix[i][i + 1]) + beta[i] \* ((i<2?0:matrix[i][i - 2]) \* alpha[i - 1] - (i<1?0:-matrix[i][i - 1]))) / delta;

}

if (i <= dim - 3) {

beta[i + 1] = (i > matrix.size() - 3?0:matrix[i][i + 2]) / delta;

}

gamma[i + 1] = (matrix[i][matrix.size()] - (i<2?0:matrix[i][i - 2]) \* gamma[i - 1] - gamma[i] \* ((i<2?0:matrix[i][i - 2]) \* alpha[i - 1] - (i<1?0:-matrix[i][i - 1]))) / delta;

}

std::vector<double> answer(dim + 3);

answer[dim - 1] = gamma[dim];

answer[dim - 2] = alpha[dim - 1] \* answer[dim - 1] + gamma[dim - 1];

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

answer[i] = alpha[i + 1] \* answer[i + 1] - beta[i + 1] \* answer[i + 2] + gamma[i + 1];

}

answer.pop\_back();

answer.pop\_back();

answer.pop\_back();

return answer;

}

void CheckAnswer(const std::vector<double>& answer) {

int dim = matrix.size();

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

double sum = 0;

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

sum += matrix[i][j] \* answer[j];

}

std::cout << "i = " << i << " sum = " << sum << " f = " << matrix[i][dim] << std::endl;

}

}

void Task4() {

std::srand(std::time(nullptr));

std::cout << "Task 4" << std::endl;

std::cout << "Extended matrix of the system: " << std::endl;

InitFiveDiagonalMatrix(7);

std::cout << "Answer: " << std::endl;

std::vector<double> answer = FiveDiagonalMethod(matrix);

PrintVector(answer);

std::cout << "Check answer: " << std::endl;

CheckAnswer(answer);

}