МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп'ютерні інформаційні технології»

**Звіт**

**з навчальної практики**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Шаповал І.В.

Дніпро, 2020

# **Постановка завдання**

Ознайомитись з методами інтерполяції функцій та розв’язання системи лінійних алгебраїчних рівнянь. Провести програмну реалізацію методів у відповідності до обраного рівня складності та оформити звіт. Програма повинна забезпечувати введення наступних даних: кількість точок функції/рівнянь, таблиці значень невідомої функції, коефіцієнти рівнянь, степінь багаточлена і точності обчислень (в разі необхідності), вибір метода розрахунку. Відповідь: значення функції в координатах, що задаються користувачем, розв’язок системи рівнянь. Значення виводяться з точністю до шести знаків. В доповнення, для рівнів складності А-С графік функції з позначенням початкових точок та аналітичний вигляд для багаточленів.

Перелік методів інтерполяції:

прості

− метод Лагранжа;

− метод Ньютона;

середньої складності

− метод кубічних сплайнів;

підвищеної складності

− поліноми Чебишева.

Перелік методів розв’язання систем алгебраїчних рівнянь (перераховані за рівнем зростання складності):

− метод Гауса;

− метод прогонки;

− метод Гауса-Зейделя.

|  |  |  |
| --- | --- | --- |
| Рівень | Вимоги до оформлення програми | Вимоги до змісту завдання |
| А | В незалежності від типу інтерфейсу обов’язків контроль вхідної інформації з реалізацією методів і функцій в окремому файлі(ах). Значення функцій та опис рівнянь повинні зчитуватись/записуватись з текстового/бінарного файлу. | Обов’язкове графічне представлення функції та початкових точок. Кількість точок для інтерполяції не менше 10, кількість рівнянь не менше 5 Допускається поєднання методів середньої складності або простих і підвищеної складності. |

# **Зовнішні специфікації**

* 1. *Апроксімація поліномами Чебишева/кубічними сплайнами:*
     1. *Формат вхідних даних:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Умовне позначення | Вимоги | Приклад |
| 1 | Вибір методу | method | 0 або 1(обирается з ListBox’ у) | 1 |
| 2 | Вибір функції | function | Ціле число від 0 до 3 включно(обирается з ListBox’ у), 0 – своя функція | 2 |
| 3 | Кількість точок для функції, яку строїть програма | nodes | Ціле число більше двох | 7 |
| 4 | Шаг для функції, яку строїть програма | step | Дійсне число більше за нуль | 0.3 |
| 5 | Стартова точка для функції, яку строїть програма | start | Дійсне число | 764.835 |
| 6 | Масив точок для своєї функції | x0, y0 | Дійсні числа. Кожен новий x більший за попередній. Для апроксімації ортогональними поліномами Чебишева шаг поміж іксами – константа. | 1; -1  2; 5  3; -7  4; 9 |

* + 1. *Формат вихідних даних:*

*-графік апроксімованої функції, та задані точки*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Умовне позначення | Вимоги | Приклад |
| 1 | Похибка, для функції, яку строїть програма | error | Середне арифметичне абсолютних похибок | 0.005 |

* 1. *Розв’язання СЛАУ методом Гаусса-Зейделя:*
     1. *Формат вхідних даних:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Умовне позначення | Вимоги | Приклад |
| 1 | Розмірність матриці | size | Ціле число від2 до 5 включно(обирается з ListBox’ у) | 3 |
| 2 | Матриця коефіціентів при іксах | a | Умова методу з наступного пункту | 10 3 -3 1  -5 10 3 2  0 0 10 0  4 1 0 10 |
| 3 | Вектор результатів рівнян | b | Дійсні числа | -23;37;0;0 |
| 4 | Вектор наближених значень іксів | x | Дійсні числа | 0;0;0;0 |
| 5 | Точність | epsilon | Ціле число, кілкість знаків після коми | 5 |

* + 1. *Формат вихідних даних:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Умовне позначення | Вимоги | Приклад |
| 1 | Вектор наближених значень іксів | x | Дійсні числа | -3;2;0;1 |

* 1. Функціональні вимоги до програми:

Програма повинна забезпечувати введення наступних даних: кількість точок функції/рівнянь, таблиці значень невідомої функції, коефіцієнти рівнянь, степінь багаточлена і точності обчислень (в разі необхідності), вибір метода розрахунку. Відповідь: значення функції в координатах, що задаються користувачем, розв’язок системи рівнянь. Значення виводяться з точністю до шести знаків. В доповнення, для рівнів складності А-С графік функції з позначенням початкових точок та аналітичний вигляд для багаточленів.

# **Вибір методу рішення задачи**

Для апроксімації функцій був обраний метод апроксімації ортогональними поліномами Чебишева, який полягає у додаванні ортогональних поліномів за формулами:

Де, m – ступінь апроксімуючого полінома; n – кількість равновіддалених точок(-1), відносно яких апроксімуєтся функція; – сама ліва точка; h – шаг поміж точками.

Але, через те, що повноцінно реалізувати не вдалося, додатково був обраний метод апроксімації кубічними сплайнами. Цей метод полягає у тому, що функція, задана на відрізках , має одну функцію

, яка відповідае умовам:

Для розв’язання СЛАУ був обраний ітеративний метод Гаусса – Зейделя, який мае одну умову:

Не зважаючи на цю, достатньо строгу, умову, метод достатньо точний та простий.

- масив початкових наближень, наприклад усі нулі

*; запам’ятати попередні наближення*

*; повторювати доки найбільша абсолютна різниця між новими та попередніми наближеннями не стане менша за потрібну точність.*

# **Розробка алгоритмів рішення програми**

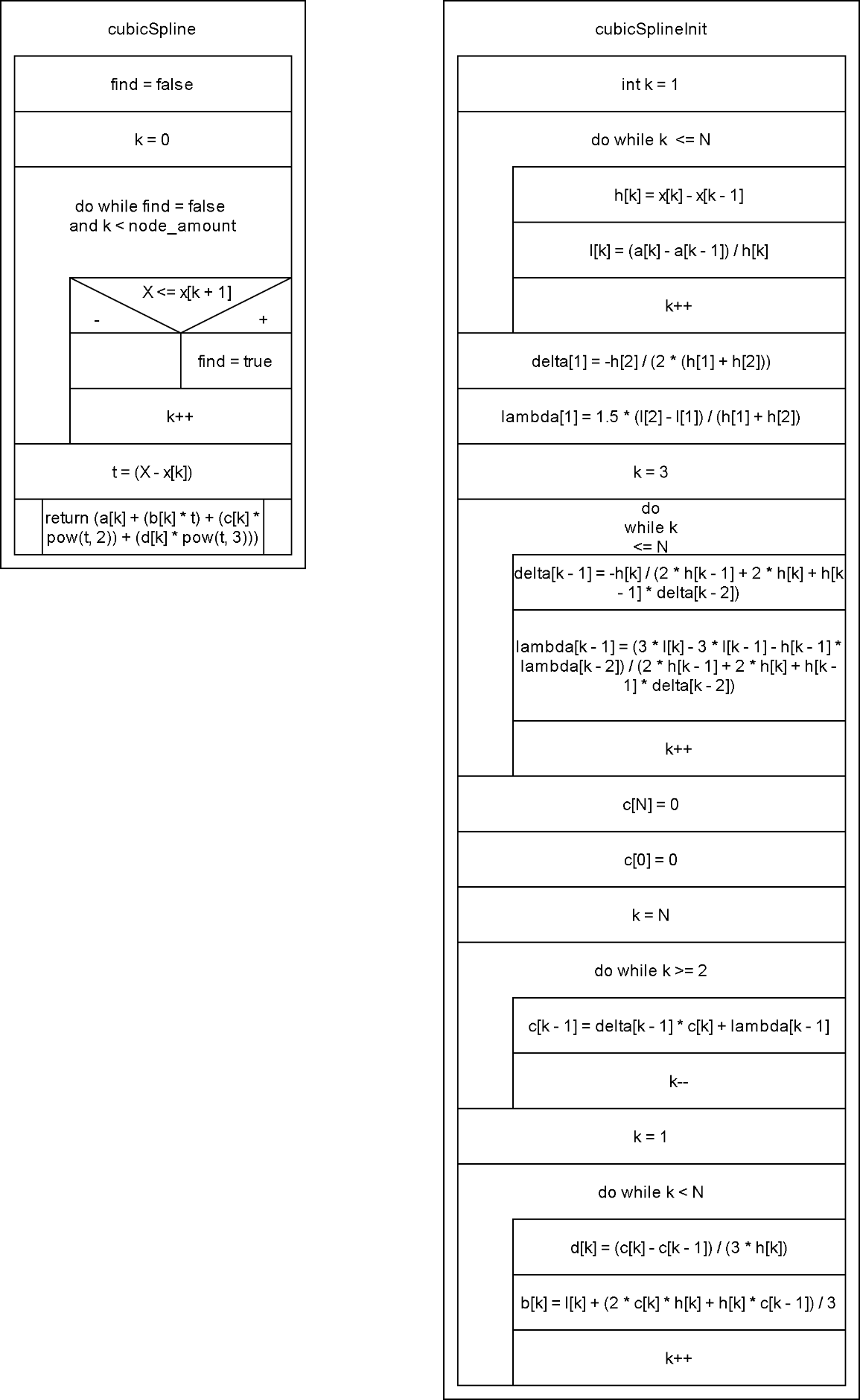


Рисунок 1

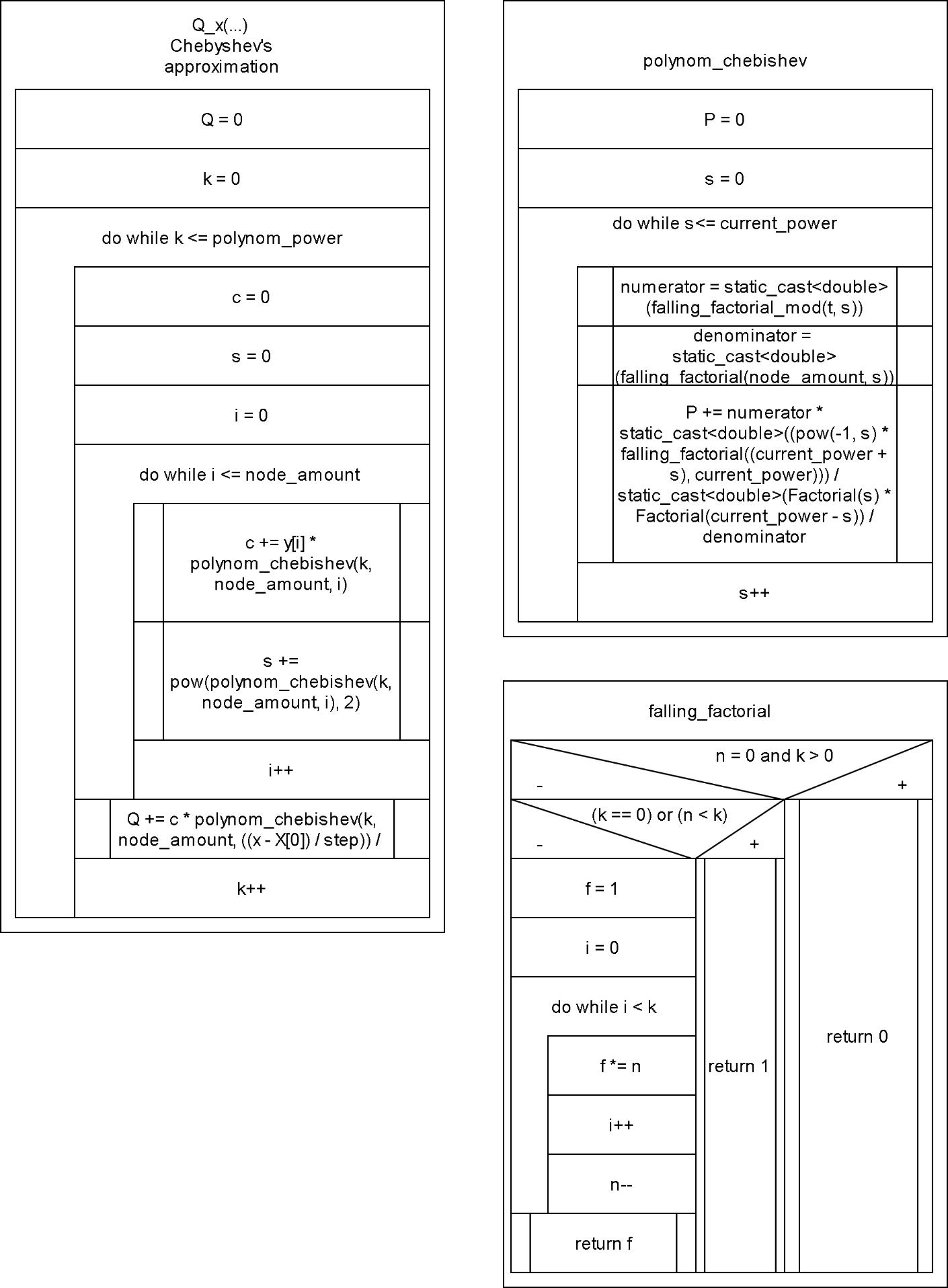


Рисунок 2

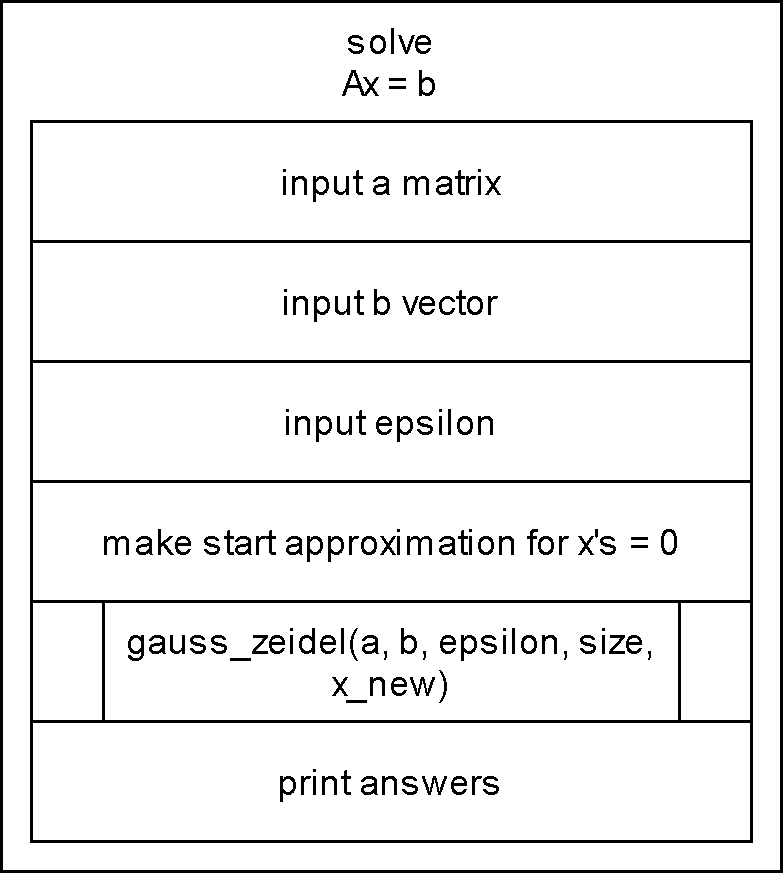


Рисунок 3

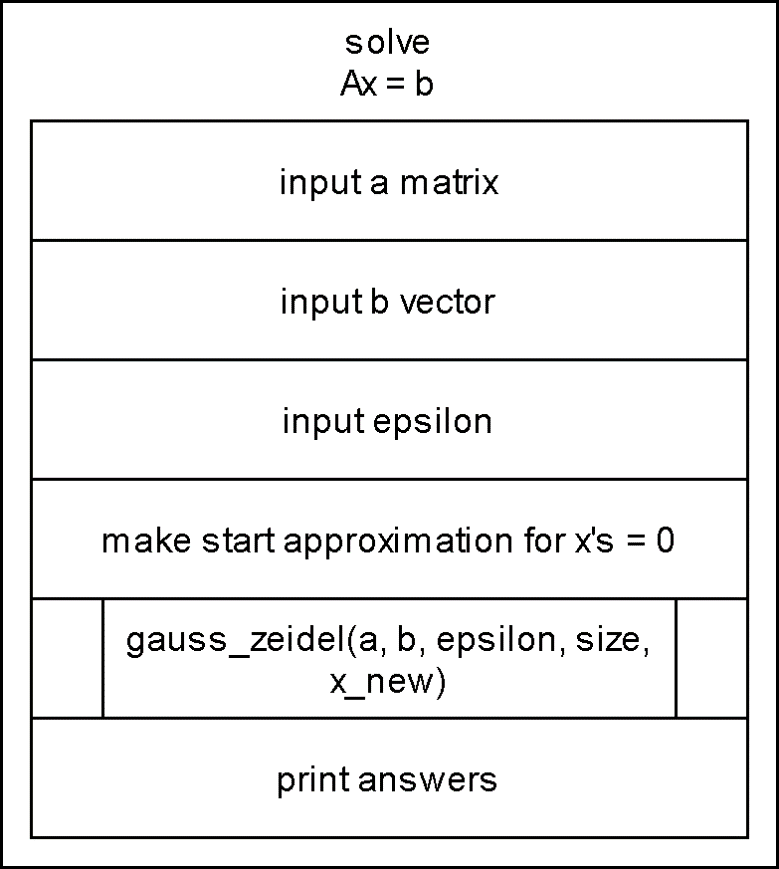


Рисунок 4

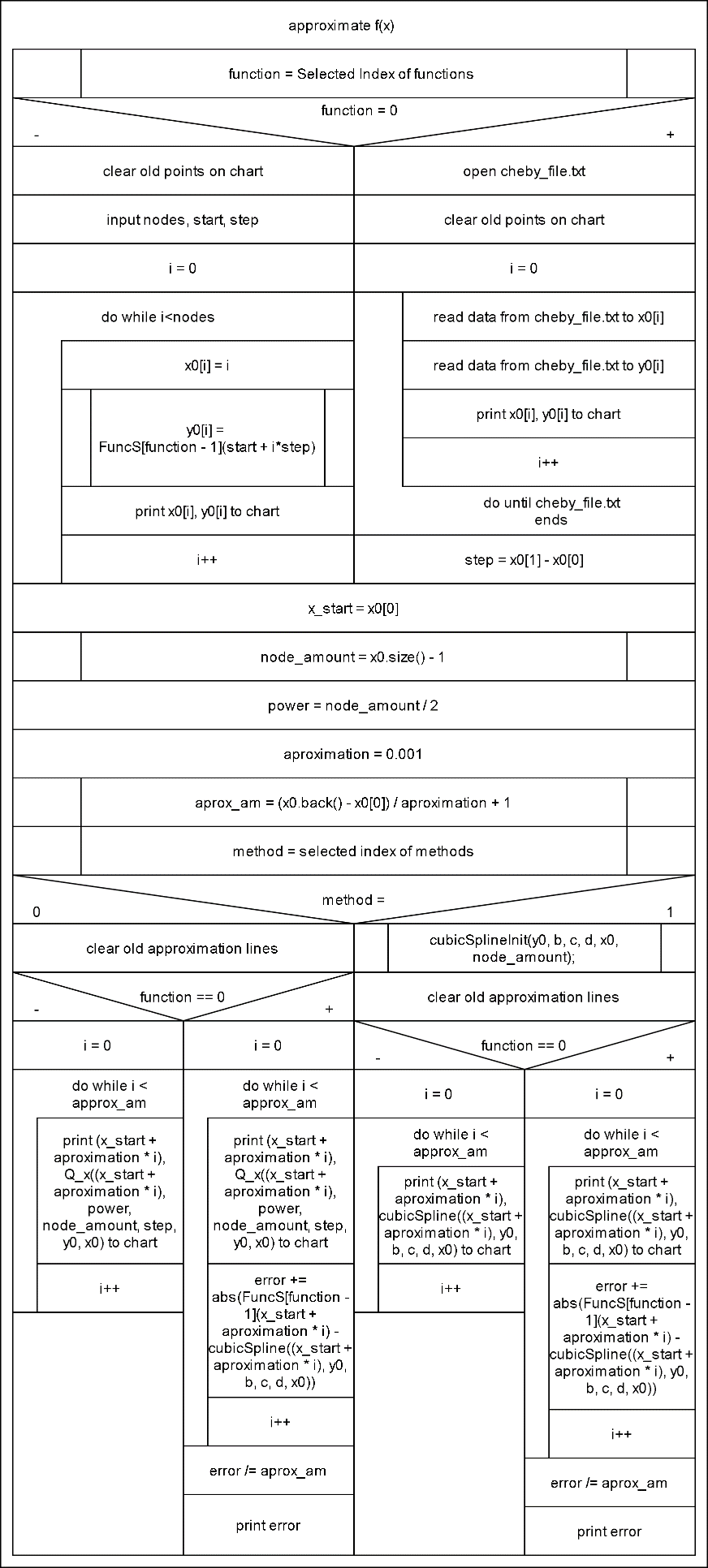


Рисунок 5

# **Текст програми**

“MyForm.h”

#include "approximation.h"

#include <cmath>

#include <fstream>

#include <string>

#include <vector>

#include "gauss\_zeidel.h"

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int aprox\_am, node\_amount, power;

double x\_start, aproximation, step;//x1-x0;

std::vector<double> y0;

std::vector<double> x0;

pointFunc FuncS[3]{ f\_x, f\_x0, f\_x1 };

int function = listBox1->SelectedIndex;

if (function == 0) {

std::ifstream cheby\_input;

cheby\_input.open("D:\\1uno1\\diit\\year1\\prac\\Done\\week1\_full\_95\\WEEK1\\cheby\_file.txt");

std::string trash;

std::getline(cheby\_input, trash);

this->chart1->Series["known f(x)"]->Points->Clear();

for (int i = 0; !cheby\_input.eof(); i++) {//

cheby\_input >> x\_start;

x0.push\_back(x\_start);

cheby\_input >> x\_start;

y0.push\_back(x\_start);

this->chart1->Series["known f(x)"]->Points->AddXY(x0[i], y0[i]);

}

step = x0[1] - x0[0];

this->textBox1->Text = Convert::ToString(" ");

this->maskedTextBox1->Text = Convert::ToString(" ");

this->maskedTextBox2->Text = Convert::ToString(" ");

this->maskedTextBox3->Text = Convert::ToString(" ");

}

else {

this->chart1->Series["known f(x)"]->Points->Clear();

int nodes = Convert::ToInt16(maskedTextBox1->Text);

double start = Convert::ToDouble(maskedTextBox3->Text);

step = Convert::ToDouble(maskedTextBox2->Text);

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

x0.push\_back(i);

y0.push\_back(FuncS[function - 1](start + i \* step));//add step for i, add f(x) choosing

this->chart1->Series["known f(x)"]->Points->AddXY(x0[i], y0[i]);

}

}

x\_start = x0[0];

node\_amount = x0.size() - 1;

power = node\_amount / 2;

aproximation = 0.001;

aprox\_am = (x0.back() - x0[0]) / aproximation + 1;

int method = listBox2->SelectedIndex;

switch (method) {

case 0:

this->chart1->Series["Approximation"]->Points->Clear();

if (function == 0) {

for (int i = 0; i < aprox\_am; i++) {

this->chart1->Series["Approximation"]->Points->AddXY((x\_start + aproximation \* i), Q\_x((x\_start + aproximation \* i), power, node\_amount, step, y0, x0));

}

}

else {

double error = 0;

for (int i = 0; i < aprox\_am; i++) {

this->chart1->Series["Approximation"]->Points->AddXY((x\_start + aproximation \* i), Q\_x((x\_start + aproximation \* i), power, node\_amount, step, y0, x0));

error += abs(FuncS[function - 1](x\_start + aproximation \* i) - Q\_x((x\_start + aproximation \* i), power, node\_amount, step, y0, x0));

}

error /= aprox\_am;

this->textBox1->Text = Convert::ToString(error);

}

break;

case 1:

double\* b = new double[node\_amount];

double\* c = new double[node\_amount];

double\* d = new double[node\_amount];

cubicSplineInit(y0, b, c, d, x0, node\_amount);

this->chart1->Series["Approximation"]->Points->Clear();

if (function == 0) {

for (int i = 0; i < aprox\_am; i++) {

this->chart1->Series["Approximation"]->Points->AddXY((x\_start + aproximation \* i), cubicSpline((x\_start + aproximation \* i), y0, b, c, d, x0));

}

}

else {

double error = 0;

for (int i = 0; i < aprox\_am; i++) {

this->chart1->Series["Approximation"]->Points->AddXY((x\_start + aproximation \* i), cubicSpline((x\_start + aproximation \* i), y0, b, c, d, x0));

error += abs(FuncS[function - 1](x\_start + aproximation \* i) - cubicSpline((x\_start + aproximation \* i), y0, b, c, d, x0));

}

error /= aprox\_am;

this->textBox1->Text = Convert::ToString(error);

}

break;

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

const int n = 5;

double\*\* a = new double\* [n];

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

a[i] = new double[n];

#pragma region A\_matrix\_init

a[0][0] = Convert::ToDouble(maskedTextBox4->Text);

a[0][1] = Convert::ToDouble(maskedTextBox13->Text);

a[0][2] = Convert::ToDouble(maskedTextBox18->Text);

a[0][3] = Convert::ToDouble(maskedTextBox23->Text);

a[0][4] = Convert::ToDouble(maskedTextBox28->Text);

a[1][0] = Convert::ToDouble(maskedTextBox10->Text);

a[1][1] = Convert::ToDouble(maskedTextBox9->Text);

a[1][2] = Convert::ToDouble(maskedTextBox8->Text);

a[1][3] = Convert::ToDouble(maskedTextBox7->Text);

a[1][4] = Convert::ToDouble(maskedTextBox6->Text);

a[2][0] = Convert::ToDouble(maskedTextBox17->Text);

a[2][1] = Convert::ToDouble(maskedTextBox16->Text);

a[2][2] = Convert::ToDouble(maskedTextBox15->Text);

a[2][3] = Convert::ToDouble(maskedTextBox14->Text);

a[2][4] = Convert::ToDouble(maskedTextBox12->Text);

a[3][0] = Convert::ToDouble(maskedTextBox25->Text);

a[3][1] = Convert::ToDouble(maskedTextBox24->Text);

a[3][2] = Convert::ToDouble(maskedTextBox22->Text);

a[3][3] = Convert::ToDouble(maskedTextBox21->Text);

a[3][4] = Convert::ToDouble(maskedTextBox20->Text);

a[4][0] = Convert::ToDouble(maskedTextBox32->Text);

a[4][1] = Convert::ToDouble(maskedTextBox31->Text);

a[4][2] = Convert::ToDouble(maskedTextBox30->Text);

a[4][3] = Convert::ToDouble(maskedTextBox29->Text);

a[4][4] = Convert::ToDouble(maskedTextBox27->Text);

#pragma endregion

double b[n]{Convert::ToDouble(maskedTextBox33->Text), Convert::ToDouble(maskedTextBox5->Text), Convert::ToDouble(maskedTextBox11->Text), Convert::ToDouble(maskedTextBox19->Text), Convert::ToDouble(maskedTextBox26->Text)};

double epsilon = Convert::ToDouble(maskedTextBox34->Text);

double x\_new[n]{0, 0, 0, 0, 0};//start approx

int size = listBox3->SelectedIndex + 2;

gauss\_zeidel(a, b, epsilon, size, x\_new);

textBox2->Text = Convert::ToString(x\_new[0]);

textBox3->Text = Convert::ToString(x\_new[1]);

textBox4->Text = Convert::ToString(x\_new[2]);

textBox5->Text = Convert::ToString(x\_new[3]);

textBox6->Text = Convert::ToString(x\_new[4]);

}

private: System::Void listBox3\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

#pragma region all\_0

maskedTextBox4->Text = Convert::ToString("0");

maskedTextBox13->Text = Convert::ToString("0");

maskedTextBox18->Text = Convert::ToString("0");

maskedTextBox23->Text = Convert::ToString("0");

maskedTextBox28->Text = Convert::ToString("0");

maskedTextBox33->Text = Convert::ToString("0");

maskedTextBox10->Text = Convert::ToString("0");

maskedTextBox9->Text = Convert::ToString("0");

maskedTextBox8->Text = Convert::ToString("0");

maskedTextBox7->Text = Convert::ToString("0");

maskedTextBox6->Text = Convert::ToString("0");

maskedTextBox5->Text = Convert::ToString("0");

maskedTextBox17->Text = Convert::ToString("0");

maskedTextBox16->Text = Convert::ToString("0");

maskedTextBox15->Text = Convert::ToString("0");

maskedTextBox14->Text = Convert::ToString("0");

maskedTextBox12->Text = Convert::ToString("0");

maskedTextBox11->Text = Convert::ToString("0");

maskedTextBox25->Text = Convert::ToString("0");

maskedTextBox24->Text = Convert::ToString("0");

maskedTextBox22->Text = Convert::ToString("0");

maskedTextBox21->Text = Convert::ToString("0");

maskedTextBox20->Text = Convert::ToString("0");

maskedTextBox19->Text = Convert::ToString("0");

maskedTextBox32->Text = Convert::ToString("0");

maskedTextBox31->Text = Convert::ToString("0");

maskedTextBox30->Text = Convert::ToString("0");

maskedTextBox29->Text = Convert::ToString("0");

maskedTextBox27->Text = Convert::ToString("0");

maskedTextBox26->Text = Convert::ToString("0");

textBox2->Text = Convert::ToString("0");

textBox3->Text = Convert::ToString("0");

textBox4->Text = Convert::ToString("0");

textBox5->Text = Convert::ToString("0");

textBox6->Text = Convert::ToString("0");

maskedTextBox34->Text = Convert::ToString("0.001");

#pragma endregion

#pragma region all invisible

maskedTextBox18->Visible = false;

maskedTextBox23->Visible = false;

maskedTextBox28->Visible = false;

maskedTextBox8->Visible = false;

maskedTextBox7->Visible = false;

maskedTextBox6->Visible = false;

maskedTextBox17->Visible = false;

maskedTextBox16->Visible = false;

maskedTextBox15->Visible = false;

maskedTextBox14->Visible = false;

maskedTextBox12->Visible = false;

maskedTextBox11->Visible = false;

maskedTextBox25->Visible = false;

maskedTextBox24->Visible = false;

maskedTextBox22->Visible = false;

maskedTextBox21->Visible = false;

maskedTextBox20->Visible = false;

maskedTextBox19->Visible = false;

maskedTextBox32->Visible = false;

maskedTextBox31->Visible = false;

maskedTextBox30->Visible = false;

maskedTextBox29->Visible = false;

maskedTextBox27->Visible = false;

maskedTextBox26->Visible = false;

textBox4->Visible = false;

textBox5->Visible = false;

textBox6->Visible = false;

label9->Visible = false;

label10->Visible = false;

label11->Visible = false;

label20->Visible = false;

label19->Visible = false;

label18->Visible = false;

label25->Visible = false;

label24->Visible = false;

label23->Visible = false;

label32->Visible = false;

label31->Visible = false;

label30->Visible = false;

label29->Visible = false;

label28->Visible = false;

label37->Visible = false;

label36->Visible = false;

label35->Visible = false;

label34->Visible = false;

label33->Visible = false;

label42->Visible = false;

label41->Visible = false;

label40->Visible = false;

label39->Visible = false;

label38->Visible = false;

label14->Visible = false;

label13->Visible = false;

label7->Visible = false;

#pragma endregion

int size = listBox3->SelectedIndex;

switch (size) {

case 3:

maskedTextBox28->Visible = true;

maskedTextBox6->Visible = true;

maskedTextBox12->Visible = true;

maskedTextBox20->Visible = true;

maskedTextBox27->Visible = true;

maskedTextBox29->Visible = true;

maskedTextBox30->Visible = true;

maskedTextBox31->Visible = true;

maskedTextBox32->Visible = true;

maskedTextBox26->Visible = true;

label11->Visible = true;

label18->Visible = true;

label23->Visible = true;

label28->Visible = true;

label33->Visible = true;

label38->Visible = true;

label39->Visible = true;

label40->Visible = true;

label41->Visible = true;

label42->Visible = true;

textBox6->Visible = true;

label7->Visible = true;

//rows/columns 1-5 visible(labels/maskedTextBoxes)

case 2:

maskedTextBox23->Visible = true;

maskedTextBox7->Visible = true;

maskedTextBox14->Visible = true;

maskedTextBox21->Visible = true;

maskedTextBox22->Visible = true;

maskedTextBox24->Visible = true;

maskedTextBox25->Visible = true;

maskedTextBox19->Visible = true;

label10->Visible = true;

label19->Visible = true;

label24->Visible = true;

label29->Visible = true;

label34->Visible = true;

label35->Visible = true;

label36->Visible = true;

label37->Visible = true;

textBox5->Visible = true;

label13->Visible = true;

//rows/columns 1-4 visible(labels/maskedTextBoxes)

case 1:

maskedTextBox18->Visible = true;

maskedTextBox8->Visible = true;

maskedTextBox15->Visible = true;

maskedTextBox16->Visible = true;

maskedTextBox17->Visible = true;

maskedTextBox11->Visible = true;

label9->Visible = true;

label20->Visible = true;

label25->Visible = true;

label30->Visible = true;

label31->Visible = true;

label32->Visible = true;

textBox4->Visible = true;

label14->Visible = true;

//rows/columns 1-3 visible(labels/maskedTextBoxes)

case 0:

//rows/columns 1-2 visible(labels/maskedTextBoxes)

break;

}

label21->Text = "x2 +";

label26->Text = "x2 +";

label20->Text = "x3 +";

label25->Text = "x3 +";

label30->Text = "x3 +";

label19->Text = "x4 +";

label24->Text = "x4 +";

label29->Text = "x4 +";

label34->Text = "x4 +";

switch (size) {

case 0:

label21->Text = "x2 =";

label26->Text = "x2 =";

case 1:

label20->Text = "x3 =";

label25->Text = "x3 =";

label30->Text = "x3 =";

break;

case 2:

label19->Text = "x4 =";

label24->Text = "x4 =";

label29->Text = "x4 =";

label34->Text = "x4 =";

break;

}

}

“approximation.h”

#ifndef approximation\_H

#define approximation\_H

#include <vector>

typedef double (\*pointFunc)(double);

double f\_x(double x);

double f\_x0(double x);

double f\_x1(double x);

double Q\_x(double x, int polynom\_power, int node\_amount, double step, std::vector<double>& y, std::vector<double>& X);

int Factorial(int n);

double polynom\_chebishev(int current\_power, int node\_amount, double t);

int falling\_factorial(int n, int k);

int falling\_factorial\_mod(int n, int k);

double cubicSpline(double X, std::vector<double> &a, double\* b, double\* c, double\* d, std::vector<double> &x);

void cubicSplineInit(std::vector<double>& a, double\* b, double\* c, double\* d, std::vector<double>& x, int N);

#endif

“approximation.cpp”

#include <cmath>

#include "approximation.h"

#include <iostream>

#include <vector>

double f\_x(double x) {

return(sin(x));//return actual function

}

double f\_x0(double x) {

return (3 \* pow(x, 5) - pow(x, 4) + 12 \* pow(x, 3) - 1.1 \* x);//return actual function

}

double f\_x1(double x) {

return(pow(x, (1 / x)));//return actual function

}

double Q\_x(double x, int polynom\_power, int node\_amount, double step, std::vector<double> &y, std::vector<double>& X) {

double Q = 0;

double c, s;

for (int k = 0; k <= polynom\_power; k++) {

c = 0;

s = 0;

for (int i = 0; i <= node\_amount; i++) {

c += y[i] \* polynom\_chebishev(k, node\_amount, i);

s += pow(polynom\_chebishev(k, node\_amount, i), 2);

}

Q += c \* polynom\_chebishev(k, node\_amount, ((x - X[0]) / step)) / s;

}

return Q;

}

double polynom\_chebishev(int current\_power, int node\_amount, double t) {

double P = 0, numerator, denominator;

for (int s = 0; s <= current\_power; s++) {

numerator = static\_cast<double>(falling\_factorial\_mod(t, s));

denominator = static\_cast<double>(falling\_factorial(node\_amount, s));

P += numerator \* static\_cast<double>((pow(-1, s) \* falling\_factorial((current\_power + s), current\_power))) / static\_cast<double>(Factorial(s) \* Factorial(current\_power - s)) / denominator;

}

return P;

}

int Factorial(int n) {//

if (n < 0)

return 0;

if (n == 0)

return 1;

return (n \* Factorial(n - 1));

}

int falling\_factorial(int n, int k) {

if ((n == 0) && (k > 0))

return 0;

if ((k == 0) || (n < k))

return 1;

int f = 1;

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

f \*= n;

return f;

}

int falling\_factorial\_mod(int n, int k) {//unused atm

if (k > n)

return 0;

if ((k == 0) || (n < k))

return 1;

int f = 1;

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

f \*= n;

return f;

}

double cubicSpline(double X, std::vector<double>& a, double\* b, double\* c, double\* d, std::vector<double>& x) {

bool find = false;//found\*

int k = 0;

for (k; !find && (k < a.size() - 2); k++)

if (X <= x[k + 1])

find = true;

double t = (X - x[k]);

return (a[k] + (b[k] \* t) + (c[k] \* pow(t, 2)) + (d[k] \* pow(t, 3)));

}

void cubicSplineInit(std::vector<double>& a, double\* b, double\* c, double\* d, std::vector<double> &x, int N) {

double\* h = new double[N];

double\* l = new double[N];

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

h[k] = x[k] - x[k - 1];

l[k] = (a[k] - a[k - 1]) / h[k];

}

double\* delta = new double[N], \*lambda = new double[N];

delta[1] = -h[2] / (2 \* (h[1] + h[2]));

lambda[1] = 1.5 \* (l[2] - l[1]) / (h[1] + h[2]);

for (int k = 3; k <= N; k++) {

delta[k - 1] = -h[k] / (2 \* h[k - 1] + 2 \* h[k] + h[k - 1] \* delta[k - 2]);

lambda[k - 1] = (3 \* l[k] - 3 \* l[k - 1] - h[k - 1] \* lambda[k - 2]) / (2 \* h[k - 1] + 2 \* h[k] + h[k - 1] \* delta[k - 2]);

}

c[N] = 0;

c[0] = 0;

for (int k = N; k >= 2; k--) {

c[k - 1] = delta[k - 1] \* c[k] + lambda[k - 1];

}

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

d[k] = (c[k] - c[k - 1]) / (3 \* h[k]);

b[k] = l[k] + (2 \* c[k] \* h[k] + h[k] \* c[k - 1]) / 3;

}

}

“gauss\_zeidel.h”

bool zeidel\_check(double\*\* a, int n);

void gauss\_zeidel(double\*\* a, double\* b, double epsilon, int n, double\* x\_new);

double max(double\* array, int n);

“gauss\_zeidel.cpp”

#include "gauss\_zeidel.h"

#include <cmath>

#include <iostream>

void gauss\_zeidel(double\*\* a, double\* b, double epsilon, int n, double\* x\_new){

double\* x\_old = new double[n], \* error\_ar = new double[n], error;

do {

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

x\_old[i] = x\_new[i];

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

x\_new[i] = b[i];

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

x\_new[i] -= a[i][j] \* x\_new[j];

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

x\_new[i] -= a[i][j] \* x\_old[j];

x\_new[i] /= a[i][i];

}

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

error\_ar[j] = abs(x\_new[j] - x\_old[j]);

error = max(error\_ar, n);

} while (error > epsilon);

}

double max(double\* array, int n) {

double\* temp = new double[n];

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

temp[i] = array[i];

}

bool sorted = false;

for (int i = 0; !sorted && (i < (n - 1)); i++) {//i - #of iteration

bool sorted = true;

for (int j = 0; j < (n - i - 1); j++) {//j - #element being compared to next

if (temp[j] < temp[j + 1]) {

float TEMP = temp[j];

temp[j] = temp[j + 1];

temp[j + 1] = TEMP;

sorted = false;

}

}

}

return(temp[0]);

}

bool zeidel\_check(double\*\* a, int n) {

bool ok = true;

double row\_sum;

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

row\_sum = 0;

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

row\_sum += a[i][j];

}

if (abs(a[i][i]) < abs(row\_sum));

}

return ok;

}

# **Контрольний приклад**

Апроксімація ортогональними поліномами Чебишева:

Приклад взятий з книжки «Численные методы анализа» Б.П. Медович, И.А. Марон, Э.З. Шувалова. Стр. 29

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 0.3 | 0.6 | 0.9 | 1.2 | 1.5 | 1.8 | 2.1 | 2.4 | 2.7 |
|  | 1.3 | 1.245 | 1.095 | 0.855 | 0.514 | 0.037 | -0.6 | -1.295 | -1.767 | -1.914 |
|  | 1.31 | 1.236 | 1.098 | 0.868 | 0.514 | 0.017 | -0.602 | -1.263 | -1.793 | -1.908 |

*Де – задані точки; – апроксімоване значення в заданих точках*

*Інтурполяція кубічними сплайнами:*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *0.2* | *1.2* | *2.2* |
|  | *7* | *5* | *5* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a | b | c | d |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

# **Тестування програми**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| 1 | Апроксімація функції з файлу | 0.2; 7  1.2; 5  2.2; 5  3.2; 7  4.2; 11  5.2; 13  6.2; 9 | Співпадіння графіку у вказаних точках |
| 2 | Апрохімаія | nodes =12  step = 0.7  start = 0 | Співпадіння графіку у вказаних точках, мінімальна похибка |
| 3 | Апроксімація | nodes = 7  step = 0.3  start = -1 | Співпадіння графіку у вказаних точках, мінімальна похибка |
| 4 | Апроксімація | nodes = 10  step = 0.2  start = 0.001 | Співпадіння графіку у вказаних точках, мінімальна похибка |
| 5 | СЛАУ 1 |  |  |
| 6 | СЛАУ 2 |  |  |

# **Аналіз результатів тестування і роботи програми**

Реалізація апроксимації ортогональними поліномами Чебишева має дуже малу похибку в заданих точках, при добре підібраному ступеню полінома(на час написання звіту програма встановлює ступінь ), але проміжкові значення повторюють значення останнього вузлу апроксимацій. Через це графік апроксимованої функції виглядає, як графік кусочної функції.

Через це була додатково реалізована інтерполяція кубічними сплайнами. Її графік у свою чергу співпадає із значеннями вузлів інтерполяції, проміжні значення також плавно з’єднують вузли. Але останній відрізок має трохи більшу похибку(скоріш за все похибка реалізації) і не співпадає значення в останньому вузлі. Взагалі, похибка невелика, але більша ніж теоретична похибка ортогональними апроксимації поліномами Чебишева.

Для розв’язання СЛАУ був реалізований метод Гаусса – Зейделя. При виконанні умови вказаної у п.3, результат відповідає очікуваному з похибкою, яка менша або дорівнює вказаній точності.

# **Література**

«Численные методы анализа» Б.П. Медович, И.А. Марон, Э.З. Шувалова.