**Лабораторная работа №1**

**Интерполяция алгебраическими многочленами**

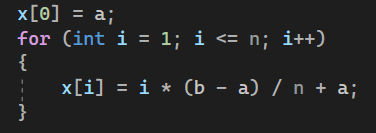
Башев Ян Дмитриевич 11 группа

Вариант 2



Способы выбора узлов:

1. Равноотстоящие узлы(расстояние между любыми двумя соседними точками одинаково) выбирались следующим образом:



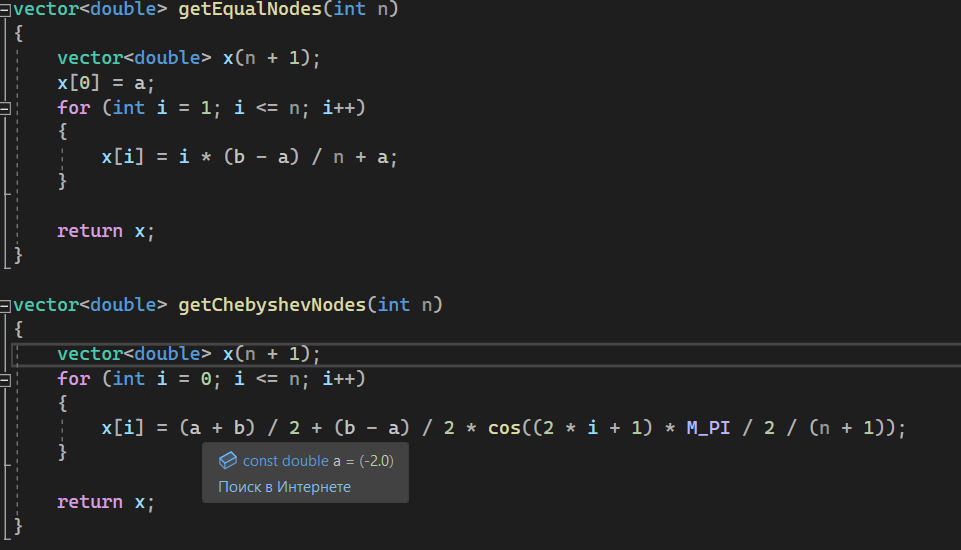
1. Чебышевские узлы вычислялись по формуле:



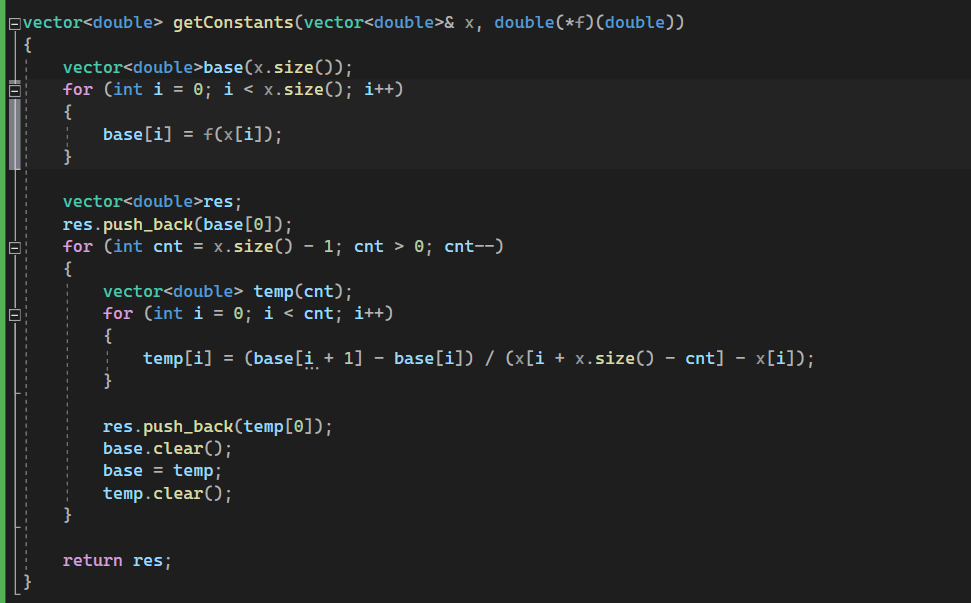
Представление, использованное при построении интерполяции многочленов

При построении интерполяции многочленов использовались:

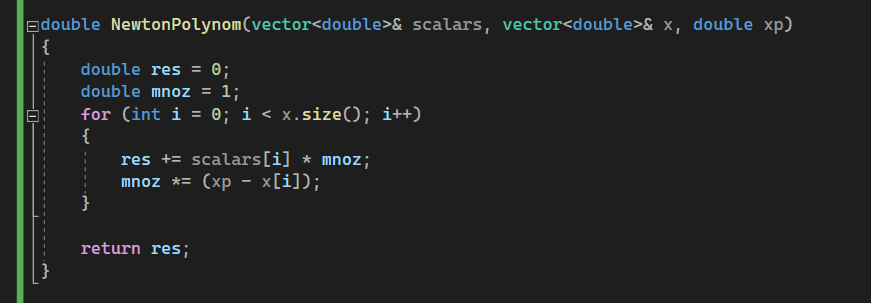
1. Функции получения вектора узлов описанными выше способами



1. Функция получения всех



1. Непосредственно сама функция вычисления полинома в точке



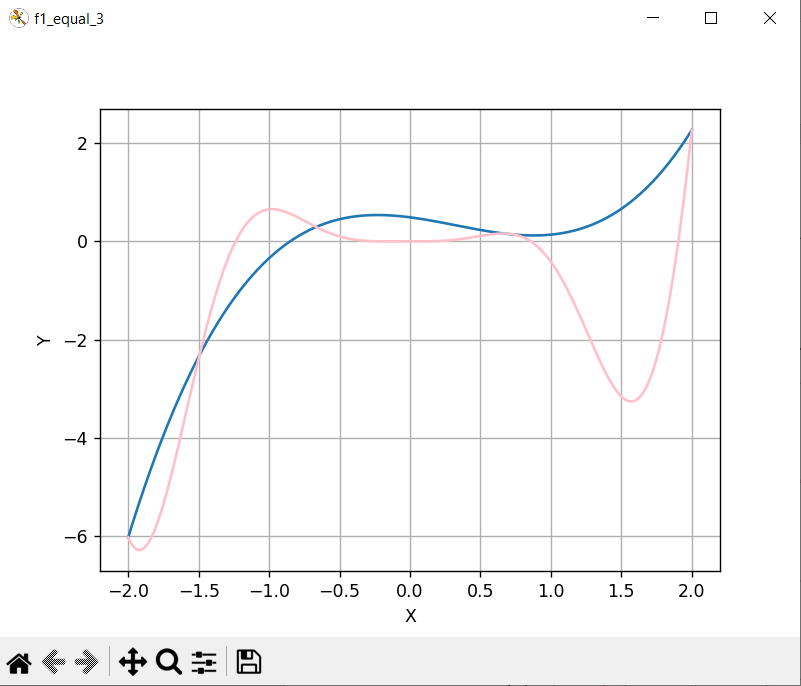
**Таблица погрешностей для функции 1)**

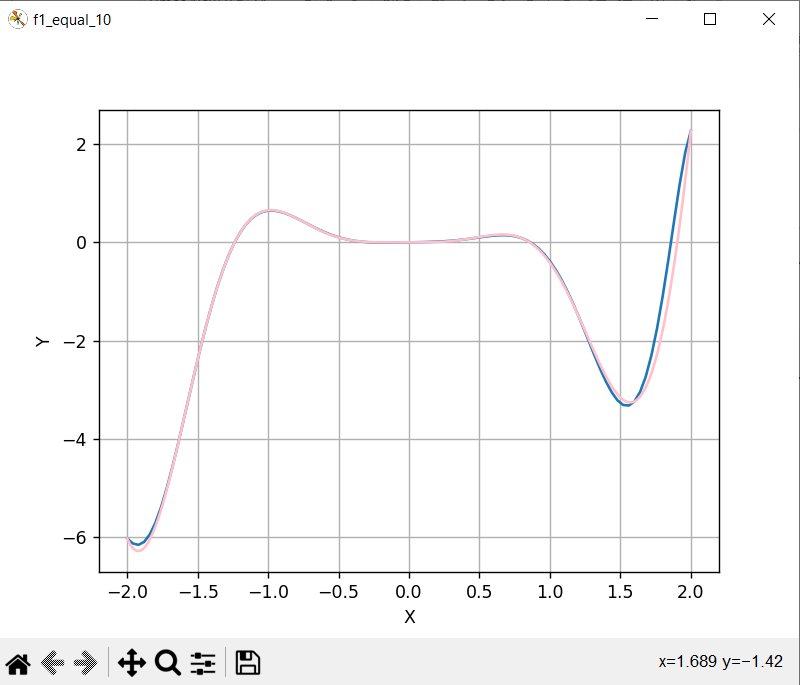
|  |
| --- |
| n Равноотстоящие узлы Чебышевские узлы |
| 3 4.12512 3.27213 |
| 4 4.10145 2.46963 |
| 5 3.19041 1.60894 |
| 6 1.51405 0.54151 |
| 7 1.03857 0.391995 |
| 8 1.85339 0.313566 |
| 9 1.41301 0.199272 |
| 10 0.891228 0.0792641 |
| 11 0.489626 0.0354085 |
| 12 0.20678 0.0101618 |
| 13 0.0956468 0.00353947 |
| 14 0.0307953 0.000810207 |
| 15 0.0120534 0.000232827 |
| 16 0.00310756 4.41263e-05 |
| 17 0.00105985 1.10107e-05 |
| 18 0.000236755 1.76486e-06 |
| 19 7.5773e-05 3.92725e-07 |
| 20 1.48644e-05 5.43171e-08 |
| 21 4.29792e-06 1.09507e-08 |
| 22 7.38597e-07 1.32995e-09 |
| 23 1.94962e-07 2.47447e-10 |
| 24 2.97996e-08 4.55431e-11 |
| 25 7.23942e-09 4.18749e-11 |
| 26 9.94186e-10 2.60663e-11 |
| 27 2.23025e-10 4.73843e-11 |
| 28 3.4811e-11 2.41922e-11 |
| 29 2.50582e-11 2.26237e-11 |
| 30 2.24007e-11 4.56648e-11 |

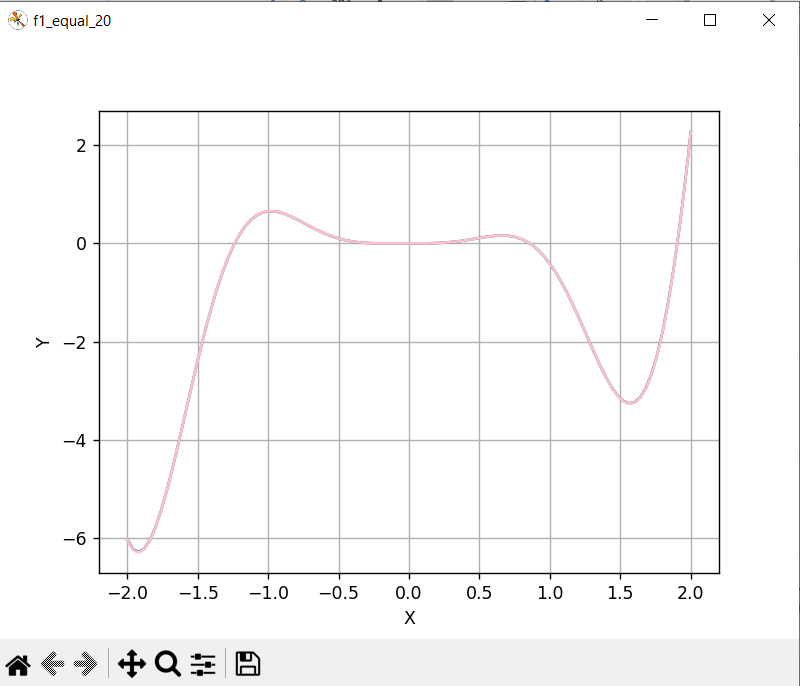
**Таблица погрешностей для функции 2)**

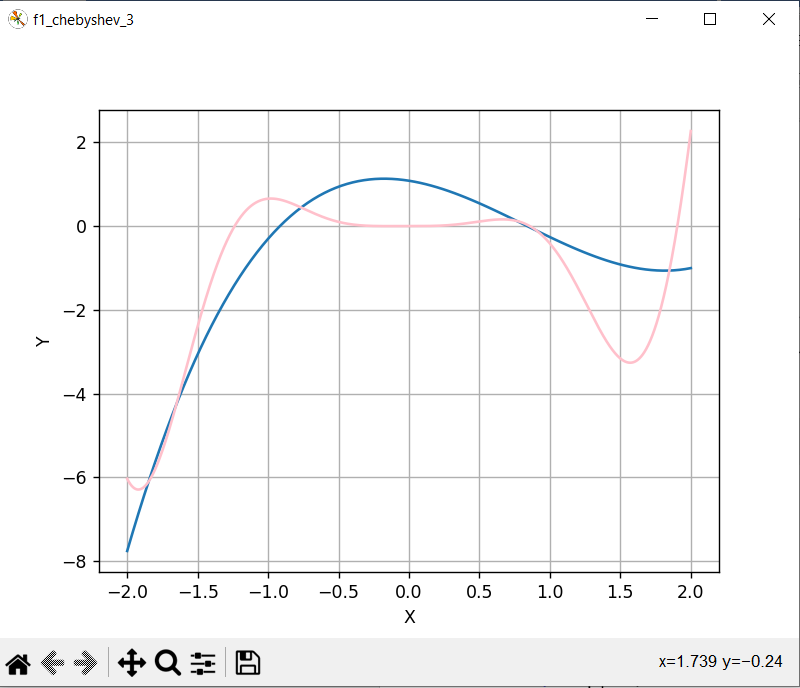
|  |
| --- |
| n Равноотстоящие узлы Чебышевские узлы |
| 3 7.94093 9.00366 |
| 4 4.14281 4.5129 |
| 5 3.22255 3.73981 |
| 6 4.33285 2.47277 |
| 7 2.20164 3.0711 |
| 8 10.9413 3.17221 |
| 9 7.69212 2.9131 |
| 10 2.35104 2.25655 |
| 11 8.77435 2.04637 |
| 12 41.6615 1.82223 |
| 13 23.3406 1.10199 |
| 14 31.1349 1.55376 |
| 15 44.9435 1.43417 |
| 16 157.351 0.94878 |
| 17 9.61235 0.938441 |
| 18 313.262 0.736069 |
| 19 280.518 0.502749 |
| 20 98.8846 0.795094 |
| 21 459.312 0.487587 |
| 22 3004.54 0.606728 |
| 23 364.695 0.595527 |
| 24 5490.81 0.723672 |
| 25 5636.46 0.696574 |
| 26 6192.74 0.697593 |
| 27 6733.91 0.472944 |
| 28 61005.6 0.606958 |
| 29 23873.7 0.560365 |
| 30 73721.6 0.539263 |

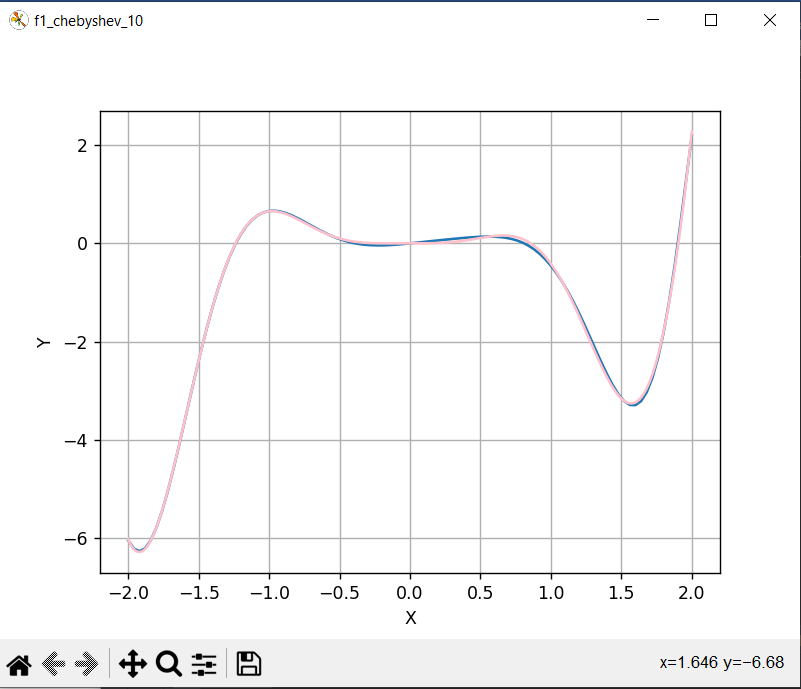
Получены следующие графики:

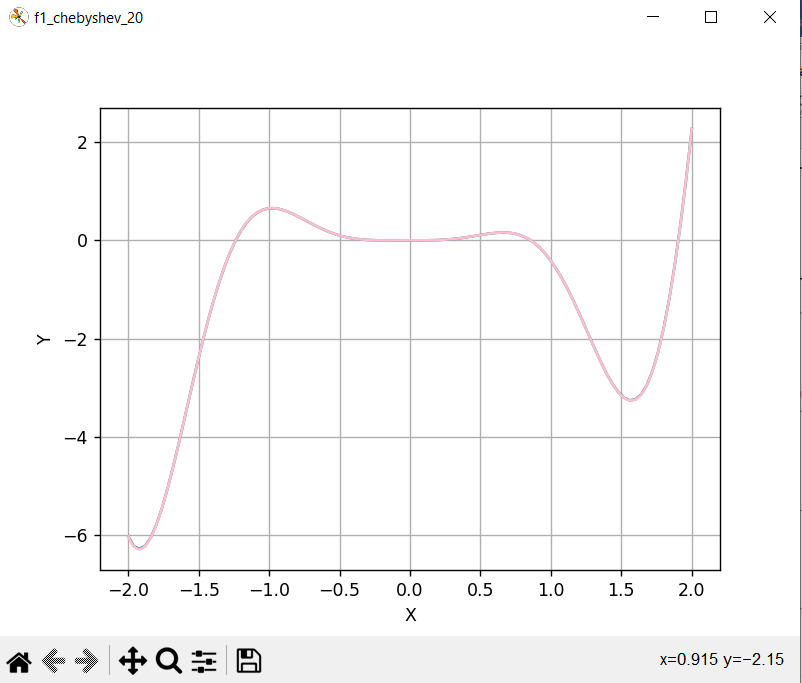


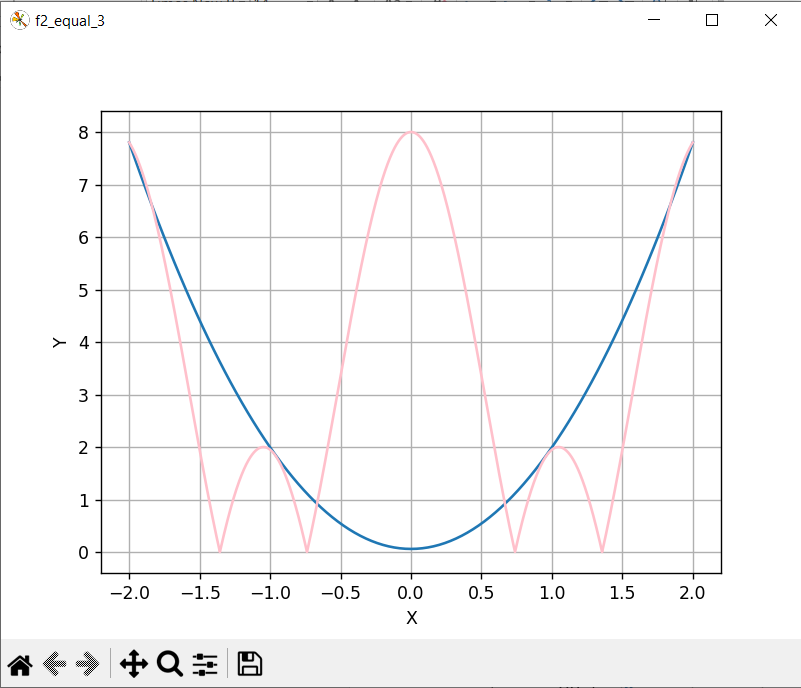


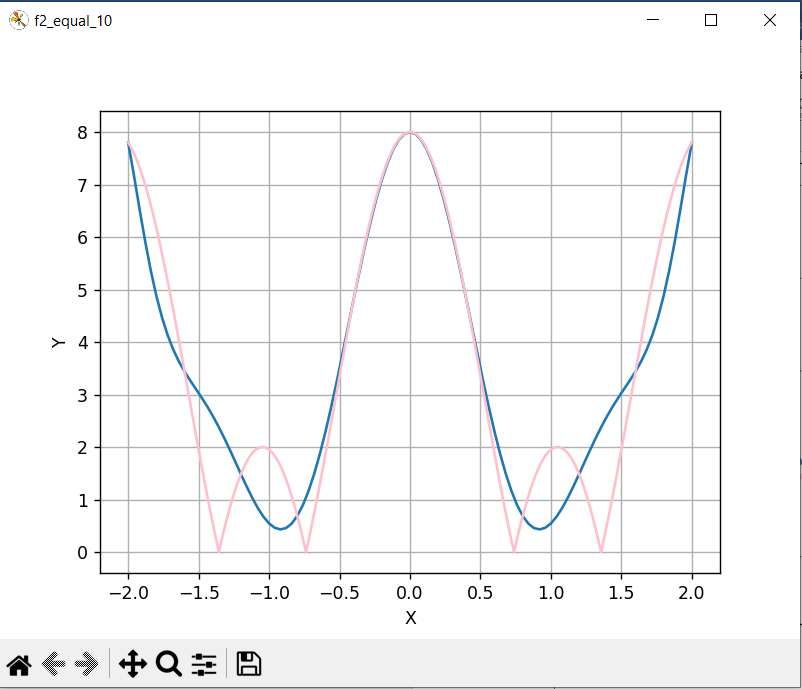


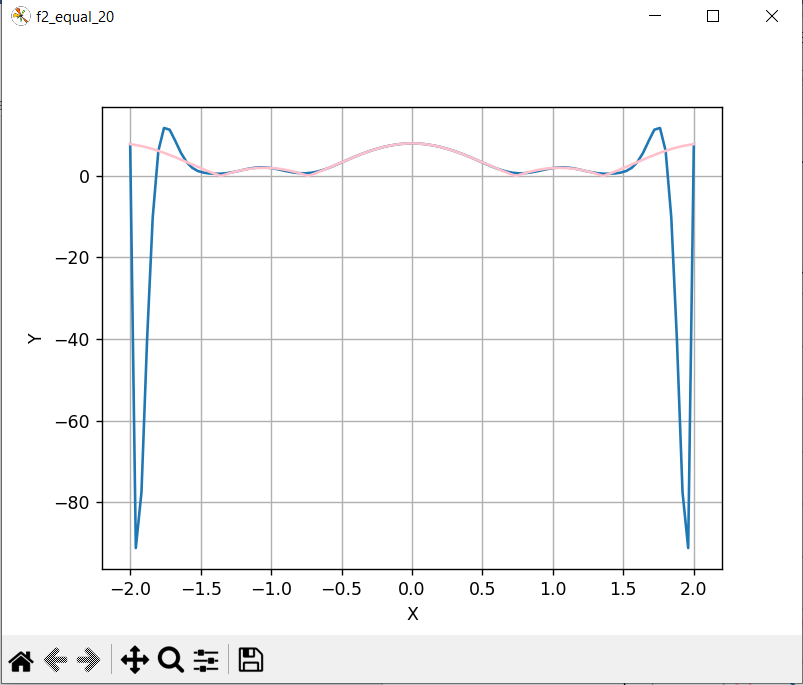


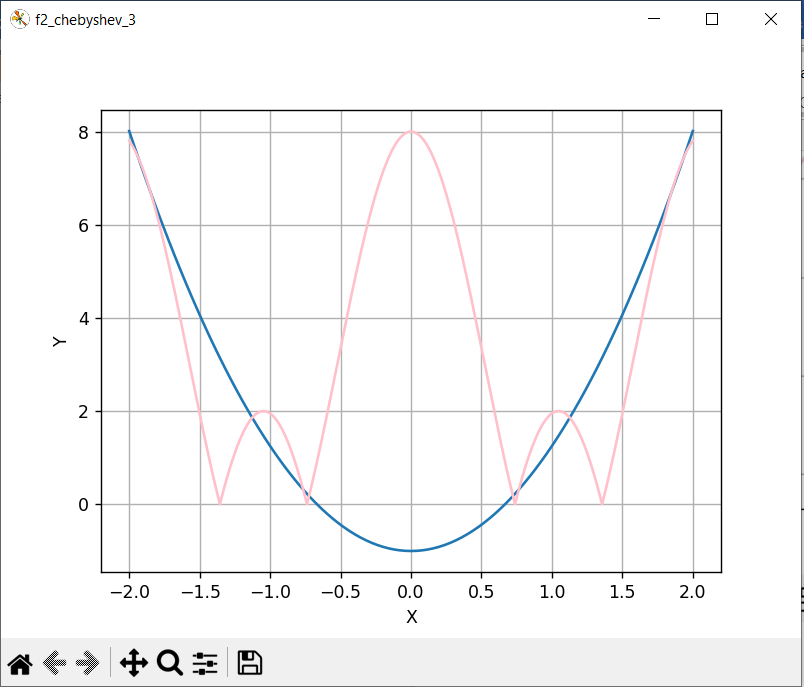


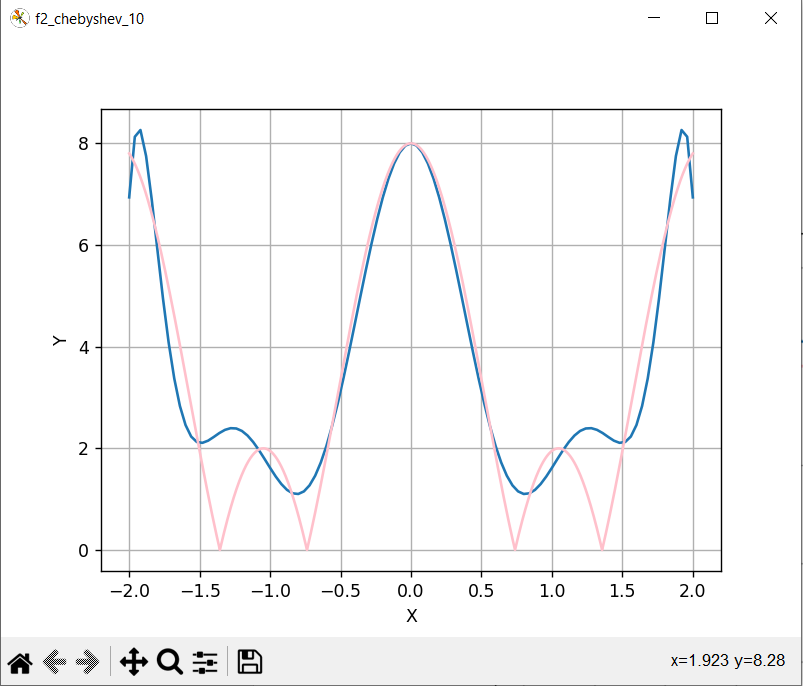


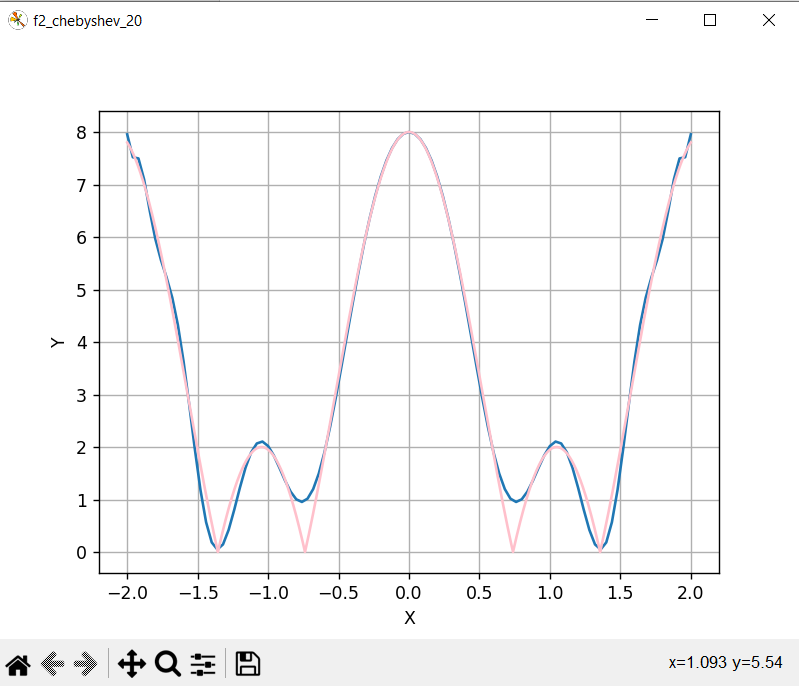












Исходный код программы:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <fstream>

#include <vector>

#include <cmath>

#include <math.h>

#include <string>

#include <map>

using namespace std;

const double a = -2;

const double b = 2;

double f1(double x)

{

return x \* x \* x \* cos(3 \* x - 1);

}

double f2(double x)

{

return abs(5 \* cos(3 \* x) + 3);

}

vector<double> getEqualNodes(int n)

{

vector<double> x(n + 1);

x[0] = a;

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

{

x[i] = i \* (b - a) / n + a;

}

return x;

}

vector<double> getChebyshevNodes(int n)

{

vector<double> x(n + 1);

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

{

x[i] = (a + b) / 2 + (b - a) / 2 \* cos((2 \* i + 1) \* M\_PI / 2 / (n + 1));

}

return x;

}

vector<double> getConstants(vector<double>& x, double(\*f)(double))

{

vector<double>base(x.size());

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

{

base[i] = f(x[i]);

}

vector<double>res;

res.push\_back(base[0]);

for (int cnt = x.size() - 1; cnt > 0; cnt--)

{

vector<double> temp(cnt);

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

{

temp[i] = (base[i + 1] - base[i]) / (x[i + x.size() - cnt] - x[i]);

}

res.push\_back(temp[0]);

base.clear();

base = temp;

temp.clear();

}

return res;

}

double NewtonPolynom(vector<double>& scalars, vector<double>& x, double xp)

{

double res = 0;

double mnoz = 1;

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

{

res += scalars[i] \* mnoz;

mnoz \*= (xp - x[i]);

}

return res;

}

void tableLine(int n, double(\*f)(double))

{

cout << n << '\t';

double res1 = 0;

double res2 = 0;

for (int i = 0; i <= 100; i++)

{

double xp = a + i \* (b - a) / 100;

vector<double>temp = getEqualNodes(n);

vector<double>temp2 = getConstants(temp, f);

res1 = max(res1, abs(NewtonPolynom(temp2, temp, xp) - f(xp)));

temp.clear();

temp2.clear();

temp = getChebyshevNodes(n);

temp2 = getConstants(temp, f);

res2 = max(res2, abs(NewtonPolynom(temp2, temp, xp) - f(xp)));

}

cout << res1 << "\t\t\t\t" << res2 << '\n';

}

void outFile(ostream& out, vector<double>& scalars, vector<double>& x)

{

for (int i = 0; i <= 100; i++)

{

double xp = a + i \* (b - a) / 100;

out << xp << ' ' << NewtonPolynom(scalars, x, xp) << '\n';

}

}

const vector<int> N\_DATA = { 3, 10, 20 };

const vector<double(\*)(double)> FUNC = { f1, f2 };

const map<double(\*)(double), string> FILE\_IDENTITY\_FUNC = { {f1, "f1"}, {f2, "f2"} };

const vector<vector<double>(\*)(int)> NODES = { getEqualNodes, getChebyshevNodes };

const map<vector<double>(\*)(int), string> FILE\_IDENTITY\_NODES = { {getEqualNodes, "equal"}, {getChebyshevNodes, "chebyshev"} };

int main()

{

for (auto n : N\_DATA)

for(auto f: FUNC)

for (auto nodes: NODES)

{

vector<double> x = nodes(n);

vector<double> scalars = getConstants(x, f);

ofstream out(FILE\_IDENTITY\_FUNC.at(f) + '\_' + FILE\_IDENTITY\_NODES.at(nodes) + '\_' + to\_string(n) + ".txt");

outFile(out, scalars, x);

out.close();

}

for (auto f: FUNC)

{

cout << FILE\_IDENTITY\_FUNC.at(f);

cout << "-----------TABLE------------\n";

cout << "n\tmax difference(equal nodes)\tmax difference(chebyshev nodes)\n";

for (int k = 3; k <= 30; k++)

{

tableLine(k, f);

}

cout << "-----------TABLE------------\n\n";

}

}