Київський національний університет

імені Т.Шевченка

Звіт

до лабораторної роботи №3

на тему:

«Інтерполяція. Сплайни**»**

Студентки 2 курсу

Групи К-24

Факультету комп’ютерних наук

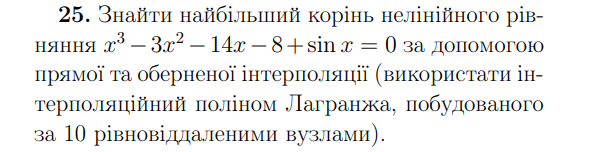
та кібернетики

Віткіної Анни

Київ-2022

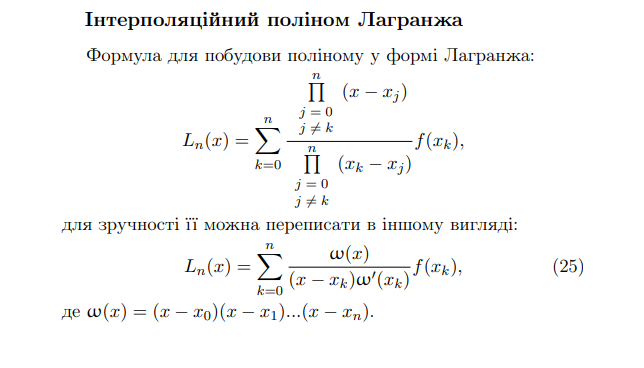
Постановка задачі. Варіант №25

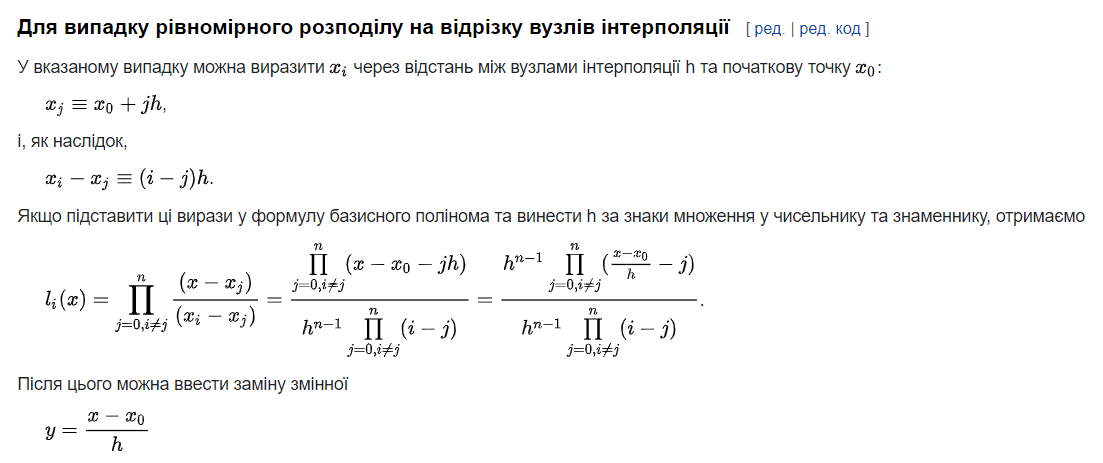
1)

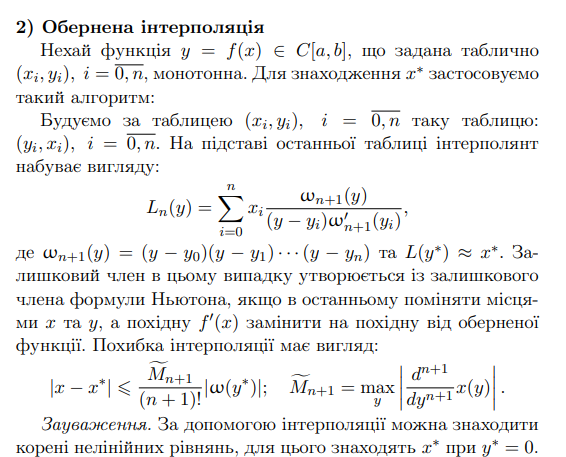


2) **Сплайни**. Умова у всіх однакова: за своїми вузлами (які Ви використовували в першій частині завдання) побудувати природній кубічний інтерполяційний сплайн. Вивести графік із функцією, яку інтерполюєте, вивести інтерполяційні поліноми (з першої частини завдання) та вивести кубічний сплайн.

Теоретичні матеріали, використані у лабораторній роботі

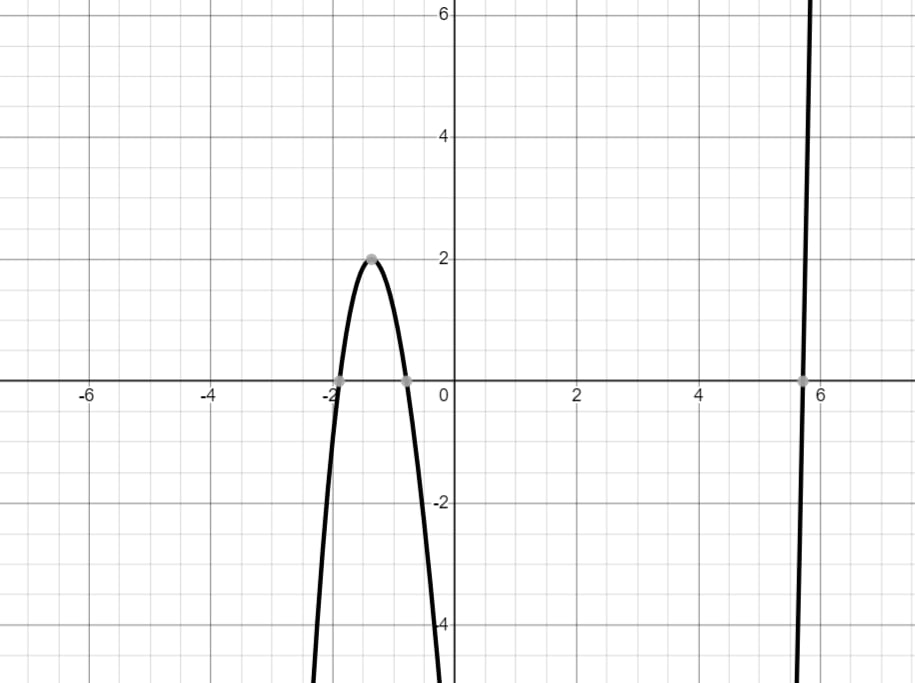
****

****

****

# Дослідження кореня

Для дослідження кореня використаємо графік, який побудуємо на сайті “Desmos”.



[5,6] – перевіримо проміжок.

f(min)\*f(max)=(15.721)\*(-28.959)<0 – підходить.

**Опис програми**

#include <iostream>

#include <iomanip>

#include <math.h>

#include<vector>

#include<algorithm>

#include<cmath>

#define N 9

using namespace std;

float function(float x) {

float y = x \* x \* x - 3 \* x \* x - 14 \* x - 8 + sin(x);

return y;

}

float XLagrange(float x, int number, float x\_array[], float y\_array[]) { **//пряма інтерполяція**

float result = 0;

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

float l = 1;

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

if (j != i)

l \*= (x - x\_array[j]) / (x\_array[i] - x\_array[j]);

result += y\_array[i] \* l;

}

return result;

}

float YLagrange(float y, int number, float x\_array[], float y\_array[]) **{//обернена інтерполяція**

float result = 0;

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

float l = 1;

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

if (j != i)

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

result += x\_array[i] \* l;

}

return result;

}

**//створювач сплайнів**

void SplCreator(float x[N + 1], float y[N + 1], float a1[N], float a2[N], float a3[N], float a4[N])

{

float h[N];

float b[N];

float ftt[N + 1];

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

{

h[i] = (x[i + 1] - x[i]);

b[i] = (y[i + 1] - y[i]) / h[i];

}

ftt[0] = 0;

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

ftt[i + 1] = 3 \* (b[i + 1] - b[i]) / (h[i + 1] + h[i]);

ftt[N] = 0;

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

{

a1[i] = (ftt[i + 1] - ftt[i]) / (6 \* h[i]);

a2[i] = ftt[i] / 2;

a3[i] = b[i] - h[i] \* (ftt[i + 1] + 2 \* ftt[i]) / 6;

a4[i] = y[i];

}

}

int main()

{

setlocale(LC\_ALL,"ukr");

float a = 5;

float b = 6;

float x\_arr[10] = { 5.0 , 5.1 , 5.2 , 5.3 , 5.4 , 5.5 ,5.6 ,5.8 , 5.9 }; **//масив аргументів**

float y\_arr[10]; **//масив значень**

for (int i = 0; i < 10; i++) **//заповнюємо масив значень**

{

y\_arr[i] = function(x\_arr[i]);

}

cout << "Iнтерполяцiя" << endl; **//виведення результатів поліномів**

cout << "Iнтерполяцiйний полiном Лагранжа(прямий) має коефiцiенти:" << endl;

for (int i = 0; i < 10; i++) {

cout << XLagrange(y\_arr[i], N, x\_arr, y\_arr) << " ";

}

cout << endl;

cout << "Iнтерполяцiйний полiном Лагранжа(обернений) має коефiцiенти:" << endl;

for (int i = 0; i < 10; i++) {

cout << YLagrange(x\_arr[i], N, x\_arr, y\_arr) << " ";

}

float a1[N], a2[N], a3[N], a4[N]; **//створюємо сплайни**

SplCreator(x\_arr, y\_arr, a1, a2, a3, a4);

cout << "Cплайни:" << endl;

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

{

cout << "Сплайн " << i + 1 << endl;

cout << "a1 = " << setprecision(10) << a1[i] << " a2 = " << setprecision(10) << a2[i] << " a3 = " << setprecision(10) << a3[i] << " a4 = " << setprecision(10)<< a4[i] << endl;

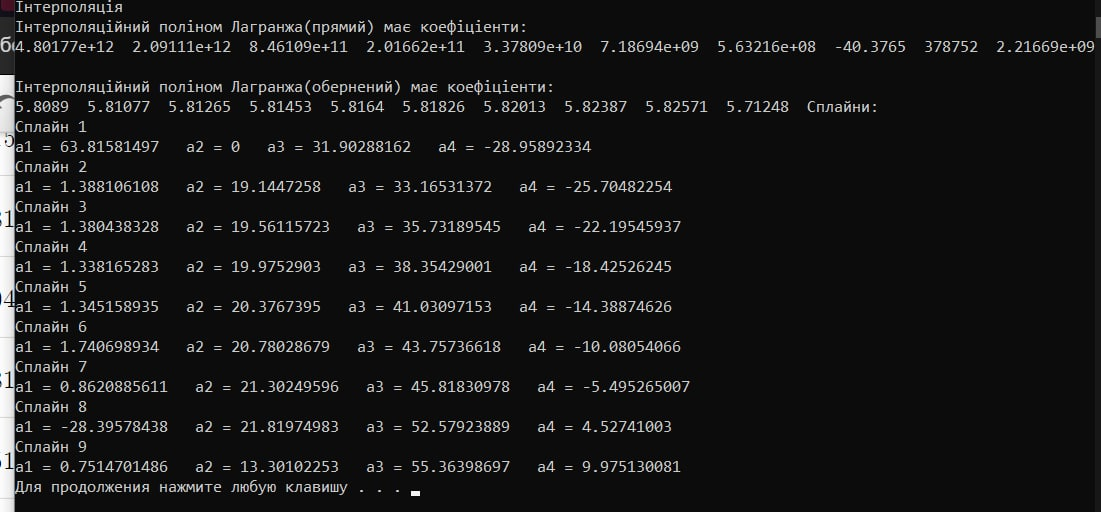
}

system("pause");

return 0;

}

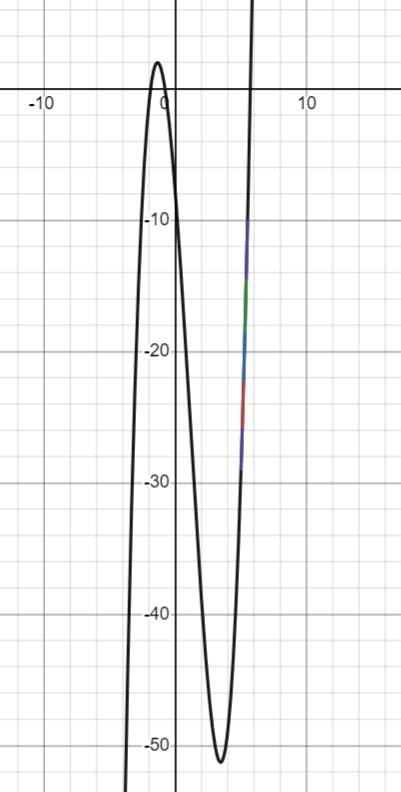
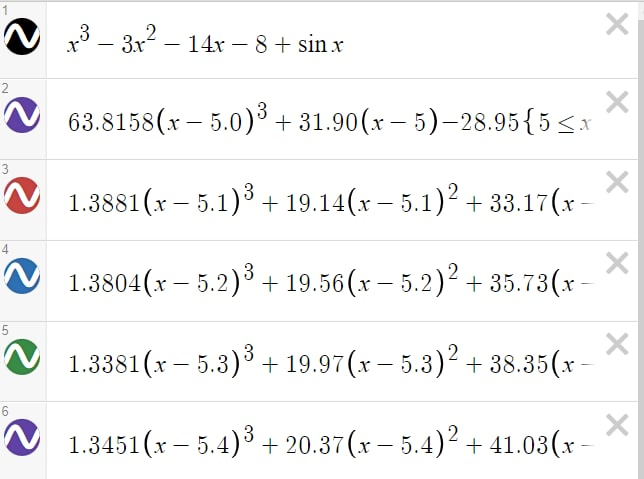
Результати:



З оберненої інтерполяції бачимо що x=5.712

**Графіки**

Будуємо графіки функції і сплайнів:

**Висновки**

Під час написання даної лабораторної роботи я змогла детальніше розглянути інтерполяцію та сплайни на практиці, зрозуміти принцип їхньої роботи та відмінності. Для всіх пунктів роботи було використано 10 рівновіддалених вузлів. Обернена інтерполяція є кращою для обчислень бо одразу видно корінь, він дорівнює вільному члену полінома.