Интерполяционные многочлены

Выполнил: студент 1-ого курса Глазырин Алексей Андреевич

5 мая 2024 г.

Содержание

1 Введение

Интерполяционные многочлены Лагранжа и Ньютона заинтересовали меня еще со времен подготовки к экзамену по алгебре. Я нашел в этих многочленах что-то особенное, захотел в них разобраться и стать "ассом"своего дела.

Но чтобы разбираться было интересно, я решил написать проект по программированию с графической визуализацией. Я посчитал это очень благородным поступком, ведь мой труд когда-нибудь понадобится таким же первокурсникам, как и мне в данный момент.

2 Выбор инструментов

2.1 Среда разработки

Для того, чтобы реализовать свой проект, я выбрал среду разработки "Visual Studio 2022". Для реализации графического интерфейса было использовано дополнительное расширение "Windows Forms которое широко распространено благодаря своей простоте и удобству.

2.2 Библиотеки

include <sstream> - для работы со строками
include <vector> - для работы с динамическими массивами
include <math.h> - для работы с математическими операциями
include <msclr/marshal.cppstd.h> - для конвертации типа Windows::String в String

3 Документация по коду

Проект состоит из 3 файлов:

1. **MyForm.cpp**

   В этом файле описаны функции, которые взаимодействуют с элементами графического интерфейса.

   - *Функция create_ chart_ click* строит график по точкам. Сначала переводит строки из TextBox(в которые требуется вводить координаты точек) в обычный string, а затем функция convert переводит их в значения double. Далее идет поточечное построение при помощи цикла for, аргументы(argumentX) которого находятся в диапозоне [-10, 10].

```
 System::Void Project::MyForm::create_chart_Click(System::Object^ sender)
{
std::string str1 = msclr::interop::marshal_as<std::string>(arguments->Text);
std::string str2 = msclr::interop::marshal_as<std::string>(values->Text);

std::vector<float> x, y;

convert(str1, x);
convert(str2, y);

this->chart1->Series[0]->Points->Clear();
for (int argumentX = -10; argumentX <= 10; argumentX++)
{
this->chart1->Series[0]->Points->AddXY(argumentX, search(x, y, argumentX));
}

return System::Void();
}
```

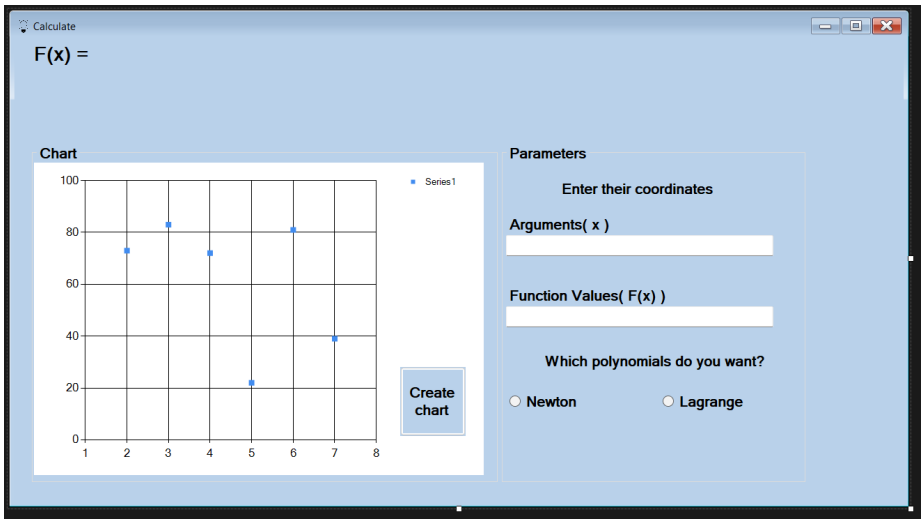
- Фунцкция radioButton(1 or 2)_CheckedChanged занимается выводом одного из полиномов:

```
System::Void Project::MyForm::radioButton1_CheckedChanged(System::Object^ sender)
{
std::string str1 = msclr::interop::marshal_as<std::string>(arguments->Text);
std::string str2 = msclr::interop::marshal_as<std::string>(values->Text);

std::vector<float> x, y;

convert(str1, x);
convert(str2, y);

std::string result = lagrangePolynomial(x, y);

this->function->Text = gcnew String(result.c_str());

return System::Void();
}
```

В данном случае многочлен Лагранжа(для Ньютона вызываем функцию NewtonPolynomial)

2. **MyForm.h**

Данный заголовочный файл инициализирует те объекты, которые были созданы в его конструкторе:



3. **Header.h**

(a) Функция для вывода в строковом типе многочлена Лагранжа. На вход подается два векторных динамических массива, которые хранят в себе отформатированные элементы из соответствующих им TextBox-ов. Далее, при помощи строковых потоков ostringstream функция накапливает результат в переменной res при помощи вывода output.

```
std::string lagrangePolynomial(std::vector<float> x, std::vector<float> y)
{
    std::ostringstream output;

    int n = x.size();
    float coef = 1;

    output << "L(x) = ";
    for (int i = 0; i < n; i++)
    {
        coef = coef * y[i];
        for (int j = 0; j < n; j++)
        {
            if (j != i)
            {
                coef = coef / (x[i] - x[j]);
            }
        }

        if (coef < 0)
        {
            output << "(" << coef << ")";
        }
        else
        {
            output << coef;
        }

        for (int j = 0; j < n; j++)
        {
            if (j != i)
            {
                if (x[j] >= 0)
                {
                    output << " * (x - " << x[j] << ")";
                }
                else
                {
                    output << " * (x + " << (-1) * x[j] << ")";
                }
            }
        }
        if (i != n - 1)
        {
            output << " + ";
        }
        coef = 1;
    }

    return output.str();
}
```

(b) Функция newtonPolynomial работает по такому же принципу, сохраняя свой многочлен в строке.

```
std::string newtonPolynomial(std::vector<float> x, std::vector<float> y)
{
    int n = x.size();
    std::vector<std::vector<float>> divided_differences(n, std::vector<float>(n));
    std::ostringstream output;

    for (int i = 0; i < n; i++)
    {
        divided_differences[i][0] = y[i];
    }

    for (int j = 1; j < n; j++)
    {
        for (int i = 0; i < n - j; i++)
        {
            divided_differences[i][j] = (divided_differences[i + 1][j - 1] - divided_differences[i][j - 1]) / (x[i + j] - x[i]);
        }
    }

    output << "N(x) = " << divided_differences[0][0];
    for (int j = 1; j < n; j++)
    {
        if (divided_differences[0][j] != 0)
        {
            output << " + ";
            output << divided_differences[0][j];
            for (int k = 0; k < j; k++)
            {
                if (x[k] >= 0)
                {
                    output << " * (x - " << x[k] << ")";
                }
                else
                {
                    output << " * (x + " << (-1) * x[k] << ")";
                }
            }
        }
    }

    return output.str();
}
```

(c) Функция convert получает на вход строку из TextBox в формате string, а потом делит эту строку на элементы, стоящие между запятыми и добавляет их в массив.

```
void convert(const std::string& str, std::vector<float>& arr) {
    std::istringstream iss(str);
    std::string token;

    while (std::getline(iss, token, ',')) {
        arr.push_back(std::stod(token));
    }
}
```

(d) Эта функция схожа с lagrangePolynomial, только здесь она считает значение многочлена в определенной точке(argumentX)

```
float search(std::vector<float> x, std::vector<float> y, double argumentX)
{
    float res = 0;
    for (int i = 0; i < x.size(); i++)
    {
        float term = y[i];
        for (int j = 0; j < x.size(); j++)
        {
            if (j != i) {
                term *= (argumentX - x[j]) / (x[i] - x[j]);
            }
        }
        res += term;
    }
    return res;
}
```