МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
“Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра проектирования и безопасности компьютерных систем

ЛАБОРАТОРНАЯ РАБОТА №8
по дисциплине
"ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА"

по теме:
«ПРИБЛИЖЁННОЕ ДИФФЕРЕНЦИРОВАНИЕ»

Выполнил: Нгуен Ле Минь
Группа: N3251
Преподаватель: Гришенцев А.Ю.

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург
2021

Задание 8.1: Конечные разности

1. Задание

Наименование задачи: дифференцирование функции, заданной в равноотстоящих точках, с помощью конечных разностей.

Вид решения: программа и отчёт.

Реализация решения: язык C или C++.

Разработать алгоритм и написать программу реализующую: дифференцирование функции (найти производную заданного порядка), заданной в равноотстоящих точках, с помощью правых и левых конечных разностей. Результаты, для производных первого, второго и третьего порядков одной и той же последовательности, отобразить на графике и сравнить.

2. Теория

В случаях, когда аналитически производную найти нельзя, производную в точках можно найти следующим образом:

1.Задают некоторое конечное значение Δx

2.Вычисляют $f(x)$ и $f(x + \Delta x)$

3. Находят $\Delta y = f(x + \Delta x)\breve{\ } f(x)$

4.Значение производной полагают равным $y' \approx \Delta y/\Delta x$ Это соотношение называют аппроксимацией производной функции спомощью отношения конечных разностей (Δy и Δx–конечные, в отличие от бесконечно малых в определении производной)

3. Код

```
/*
 * Author : Nguyen Le Minh
 * Group : N3251
 * Lab : 8.1
 */
#include <iostream>
#include <cmath>
#include "matplotlibcpp.h"
using namespace std;
namespace plt = matplotlibcpp;

const int a = 0;
const int b = 10;
const int steps = 10;

double f(double x){
    return 4*pow(x,3) + pow(x,2) - 2*x + 3;
}

void solver(){
    double d1[2][steps - 1];
    double d2[2][steps - 2];
    double d3[2][steps - 3];
    double step = (double)(b - a)/steps;
    for(int i = 0; i < steps - 1; i++){
        d1[0][i] = (-f(a + i*step) + f(a + (i + 1)*step))/step;
        d1[1][i] = a + i*step + step/2;
    }
    for(int i = 0; i < steps - 2; i++){
        d2[0][i] = (-d1[0][i] + d1[0][i+1])/step;
        d2[1][i] = a + i*step + step/2;
    }
    for(int i = 0; i < steps - 3; i++){
        d3[0][i] = (-d2[0][i] + d2[0][i+1])/step;
        d3[1][i] = a + i*step + step/2;
```

```
    }
    plt::plot(d1[1], d1[0], "r-");
    plt::plot(d2[1], d2[0], "k-");
    plt::plot(d3[1], d3[0], "b-");
    plt::show();
    for (int i = 0; i < steps-2; i++)
        cout << "d1("<< d1[1][i] << ", " << d1[0][i] << ") d2(" << d2[1][i] << ", " << d2[0][i] <<
}

int main(){
    solver();
    return 0;
}
```

4. Выводы программы

d1(0.5, 3) d2(0.5, 26) d3(0.5, 24)
d1(1.5, 29) d2(1.5, 50) d3(1.5, 24)
d1(2.5, 79) d2(2.5, 74) d3(2.5, 24)
d1(3.5, 153) d2(3.5, 98) d3(3.5, 24)
d1(4.5, 251) d2(4.5, 122) d3(4.5, 24)
d1(5.5, 373) d2(5.5, 146) d3(5.5, 24)
d1(6.5, 519) d2(6.5, 170) d3(6.5, 24)
d1(7.5, 689) d2(7.5, 194) d3(26, 0.5)

При выполнении этого задания мы научились средствами языка C++ реализовывать дифференцирование функции с помощью правых и левых конечных разностей.

Задание 8.2: Производная с помощью преобразования Фурье

1. Задание

Наименование задачи: производная с помощью преобразования Фурье.
Вид решения: программа и отчет.
Реализация решения: язык C или C++.

Разработать алгоритм и написать программу реализующую: вычисление производной с помощью преобразования Фурье. Результаты, для производной одной и той же последовательности, полученной с помощью преобразования Фурье и правых и левых конечных разностей отобразить на графике и сравнить. Оценить вычислительную сложность вычисления производной с помощью преобразования Фурье.

2. Теория

Theorem 1 *Пусть функция f абсолютно интегрируема на $(-\infty;\infty)$ и f' непрерывна и абсолютно интегрируема на $(-\infty;\infty)$. Тогда*

$F[f'](y) = (iy)F[f](y), y \in (-\infty;\infty)$

Доказательство :

Представим функцию f в виде

$$f(x) = f(0) + \int_0^x f'(t)dt$$

из сходимости интеграла $\int_0^{+\infty} f'(t)dt$ следует существование пределов $\lim_{x\to+\infty} f(x)$, $\lim_{x\to-\infty} f(x)$. Они не могут быть отличными от нуля в силу сходимости интеграла $\int_{-\infty}^{\infty} |f(x)|dx$. С помощью интегрирования по частям получаем

$$F[f'](y) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f'(x)e^{-ixy}dx = \frac{1}{\sqrt{2\pi}}f(x)e^{-ixy}\Big|_{-\infty}^{\infty} + \frac{iy}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)e^{-ixy}dy = iyF[f](y)$$

.

Theorem 2 *Пусть функции f непрерывна на $(-\infty;\infty)$, а функция $f_1 = xf(x)$ абсолютно интегрируема на $(-\infty;\infty)$. Тогда :*

$\frac{d}{dy}F[f](y) = F[-if_1](y) = F[-ixf(x)](y)$

Доказательство :

Дифференцируя интеграл $F[f](y) := v.p.\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)e^{-iyx}dx$ по параметру y,

получаем на основании теоремы

$$\frac{d}{dy}F[f](y) = \frac{1}{\sqrt{2\pi}}\frac{d}{dy}\int_{-\infty}^{\infty} f(x)e^{-iyx}dx = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}(-ix)f(x)e^{-iyx}dx$$

Заметим, что последний интеграл сходится равномерно на $(-\infty, \infty)$ по признаку Вейерштрасса с мажорорантом $\phi(x) = | xf(x) |$.

3. Код

4. Выводы программы

Задание 8.3: Интерполяционные формулы Ньютона

1. Задание

Наименование задачи: дифференцирование функции, заданной в равноотстоящих точках, с помощью конечных разностей.

Вид решения: программа и отчёт.

Реализация решения: язык C или C++.

Разработать алгоритм и написать программу реализующую: дифференцирование функции (найти производную заданного порядка), заданной в равноотстоящих точках, с помощью правых и левых конечных разностей. Результаты, для производных первого, второго и третьего порядков одной и той же последовательности, отобразить на графике и сравнить.

2. Теория

Функция $y = f(x)$ задана в равноотстоящих точках $x_i, (i = 0, 1, 2..., n)$ отрезка $[a, b]$. Для нахождения производных на отрезке $[a, b]$ функцию $y = f(x)$ приближенно заменим интерполяционным полиномом Ньютона, построенным для системы узлов $x_0, x_1, ..., x_k (k \leq n)$ и получим первую интерполяционную формулу :

$$y(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_0 + ... + \frac{t(t-1)...(t-n+1)}{n!}\Delta^n y_0, (1)$$

где $\Delta^k y_0 = \Delta^{k-1} y_1 - \Delta^{k-1} y_0$ - конечная разность $k-$ ого порядка,

$$t = \frac{x - x_0}{h}, h = x_{i+1} - x_i, (i = 0, 1, ...)(2)$$

Перемножив биномы получим

$$y(x) = y_0 + t\Delta y_0 + \frac{t^2 - t}{2}\Delta^2 y_0 + \frac{t^3 - 3t^2 + 2t}{6}\Delta^3 y_0 + ... + \frac{t(t-1)...(t-n+1)}{n!}\Delta^n y_0 (3)$$

Выражение (3) содержит в правой части функцию $t(x) = \frac{x - x_0}{h}$

Метод нахождения производной правой конечной разностью :

Формула :

$$y'(i) = \frac{y_{i+1} - y_i}{h} + O(h) = \frac{\Delta y_i}{h} + O(h)$$

Метод нахождения производной левой конечной разностью :

Формула :

$$\overline{y'(i)} = \frac{y_i - y_{i-1}}{h} + O(h) = \frac{\Delta \overline{y_i}}{h} + O(h)$$

3. Код

```
/*
 * Author : Nguyen Le Minh
 * Group : N3251
 * Lab : 8.3
 */
#include <iostream>
#include <vector>
using namespace std;
//factorial
int fac(int x) {
    if (x <= 1) return 1;
    int result = 1;
    for (int i = 2; i <= x; i++) {
        result *= i;
    }
    return result;
}
```

```cpp
double trlen_k(float q,int k,double delta_k) {
    //k k-ой член в понином P(x)
    //delta_k k-ой разносmь x0
    double result = 1;
    for (int i = k-1; i >=0; i--)
    {
        result *= (q - i);
    }
    result = result * delta_k / fac(k);
    return result;
}

void solver(){
    int n;

    cout << "Input N:";//размер последовательности
    cin >> n;
    int num;
    cout << "Input number of pairs value(x,y):";
    cin >> num;
    float start;
    float h;//шаг
    cout << "Input start value x0: ";
    cin >> start;
    cout << "Input step value h:";
    cin >> h;
    auto *x = (float*)malloc(sizeof(float)*num);
    auto *y = (double*)malloc(sizeof(double)*num);
    cout << "Input yi:";
    x[0] = start;
    cin >> y[0];

    //вычислит конечные разности

    vector<vector<double>> delta(n);
    int prev = 0;
    for (int i = 1; i < num; i++)
    {
        x[i] = start + i * h;
        cin >> y[i];
        delta[0].push_back(y[i] - y[prev]);//delta[0][i] первой разносmь
        prev = i;
    }
    int prev_k;
    for (int k = 1; k < n; k++)
    {
        prev = 0;
        prev_k = k-1;
        for (int i = 1; i < (num - k); i++)
        {
            //delta[k][i] k+1-mu разносmь
            delta[k].push_back(delta[prev_k][i] - delta[prev_k][prev]);
            prev = i;

        }

    }
    cout << "x y delta_y delta_2y delta_3y" << endl;
    for (int i = 0; i < num-3; i++)
    {
        cout << x[i]<<" " << y[i] << " " << delta[0][i] << " " << delta[1][i]
```

```
        << " " << delta[2][i];
        cout << endl;
    }

    // Интерполяционные формулы Ньютона

    float x0,x_new;//x0 наболее близки к x_new по сравнению с заданной xi,x_new>x0
    cout << "Input x_new to find y_new:";
    cin >> x_new;
    cout << "Input x0 and number of x0 in series:";
    cin >> x0;
    int k;//положение x0 в заданной ряд x
    cin >> k;
    k -= 1;
    float q=(x_new-x0)/h;//число шагов для достижения точки x0
    double Px=y[k];
    for (int i = 1; i <= n; i++)
    {
        cout << trlen_k(q, i, delta[i - 1][k]) << endl;
        Px += trlen_k(q, i, delta[i - 1][k]);
    }
    cout <<"Value Y_new: "<< Px<<endl;

    //производной левой разностью y'=f(x)-f(x0)/(x-x0) при x>x0
    double        df_left = (Px - y[k]) / (x_new - x0);
    cout << "derivative left: " << df_left << endl;
    //производной правой разностью y'=f(x)-f(x0)/(x-x0) при x<x0
    int idx_right = 0;//index x0 where x0>x_new
    for (int i = 0; i < num; i++)
    {
        if (x[i] > x_new) {
            idx_right = i;
            break;
        }
    }
    double        df_right = (Px - y[idx_right]) / (x_new - x[idx_right]);
    cout << "derivative right: " << df_right << endl;
    free(x);
    free(y);

}

int main() {
    solver();
    return 0;
}
```

4. Выводы программы

Input N:3
Input number of pairs value(x,y):11
Input start value x0: 1
Input step value h:0.1
Input yi:0.8753
0.9903
0.6996
0.7855
0.3696
0.4144
0.5255
0.6963
0.1055
0.29963
0.7417
x y delta_y delta_2y delta_3y
1 0.8753 0.115 -0.4057 0.7823
1.1 0.9903 -0.2907 0.3766 -0.8784
1.2 0.6996 0.0859 -0.5018 0.9625
1.3 0.7855 -0.4159 0.4607 -0.3944
1.4 0.3696 0.0448 0.0663 -0.0066
1.5 0.4144 0.1111 0.0597 -0.8213
1.6 0.5255 0.1708 -0.7616 1.54653
1.7 0.6963 -0.5908 0.78493 -0.53699
Input x_new to find y_new:1.43
Input x0 and number of x0 in series:1.4
5
0.01344
-0.0069615
-0.0003927
Value Y_new: 0.375686
derivative left: 0.20286
derivative right: 0.55306

Мы научились средствами языка c++ реализовывать вычисления производной с помощью интерполяционных формул Ньютона.