**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

**ЧИСЛЕННОЕ ИНТЕРПОЛИРОВАНИЕ И ДИФФЕРЕНЦИРОВАНИЕ**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

Ходосевич Данила

**Численное интерполирование и дифференцирование**

Пусть на отрезке [*а; b*] заданы точки и значения функции в этих точках: *.*

Интерполяционный многочлен Лагранжа:

.

Оценка погрешности формулы Лагранжа:

,

где

Для равномерной сетки с шагом h и известных узлов yi-1, yi, yi+1 производные в точке xi вычисляются следующим образом

|  |  |  |  |
| --- | --- | --- | --- |
| Левая | Правая | Центральная | Вторая |
| (yi-yi-/h | (yi+1-yi)/h | (yi+1-yi-/2h | (yi-1-2yi+yi+/h2 |

**Задание 1. Интерполирование**

Известна функция . Заполнить таблицу значениями yk для указанных *xk*, с точностью 10-4. Составить по таблице интерполяционный многочлен Лагранжа. Привести его окончательный вид. Вычислить значение функции в заданной точке аналитически и с помощью многочлена Лагранжа. Оценить погрешность формулы Лагранжа и абсолютную погрешность вычислений.

**Задание 2. Дифференцирование**

2) Вычислить таблицу на отрезке [a,b] на равномерной сетке (5 узлов), и в этих узлах найти значение первой производной функции по формулам 1-го (левая и правая) и 2-го порядка точности (центральная) и значение второй производной по формулам 2-го порядка точности, где это возможно Во всех точках найти точные значения производных. Оценить погрешность. Результаты свести в таблицу. Точность – 4 значащих цифры.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | численно | | | | точно | |
| *xk, m* | *f’(x)* слева | *f’(x)* справа | *f’(x)* центр | *f’’(x)* | *f’(x)* | *f’’(x)* |

**Задание 1. Интерполирование**

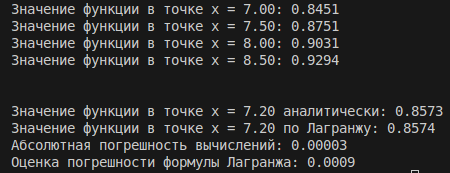
,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 7.0 | 7.5 | 8.0 | 8.5 |
|  | 0.8451 | 0.8751 | 0.9031 | 0.9294 |

Интерполяционный метод Лагранжа:

Интерполяционный метод Лагранжа:

, *,*

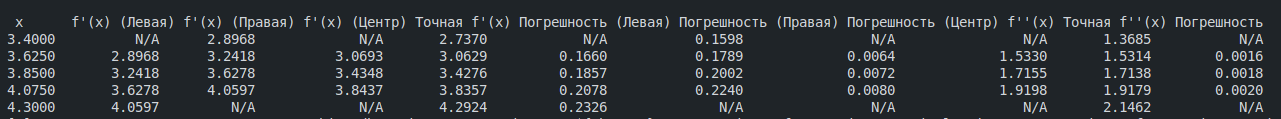


**Задание 2. Дифференцирование**

|  |  |  |  |
| --- | --- | --- | --- |
| Левая | Правая | Центральная | Вторая |
| (yi-yi-/h | (yi+1-yi)/h | (yi+1-yi-/2h | (yi-1-2yi+yi+/h2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Погрешность f’(x) слева | Погрешность f'(x) справа | Погрешность f'(x) центр | Погрешность f''(x) |
| 3.400 | Не сущ | 0.1599 | Не сущ | Не сущ |
| 3.625 | 0.1660 | 0.1789 | 0.0065 | 0.0016 |
| 3.850 | 0.1858 | 0.2002 | 0.0072 | 0.0018 |
| 4.075 | 0.2079 | 0.2241 | 0.0081 | 0.0020 |
| 4.300 | 0.2326 | Не сущ | Не сущ | Не сущ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Численно | | | | Точно | |
| xk, m | f’(x) слева | f’(x) справа | f’(x) центр | f’’(x) | f’(x) | f’’(x) |
| 3.400 | - | 2.8969 | - | - | 2.7370 | 1.3685 |
| 3.625 | 2.8969 | 3.2418 | 3.0693 | 0.3642 | 3.0629 | 1.5314 |
| 3.850 | 3.2418 | 3.6278 | 3.4348 | 0.4127 | 3.4276 | 1.7138 |
| 4.075 | 3.6278 | 4.0598 | 3.8438 | 0.4677 | 3.8357 | 1.9179 |
| 4.300 | 4.0598 | - | - | - | 4.2924 | 2.1462 |



**Приложения**

**Задание 1**

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

#include <algorithm>

// Функция для вычисления логарифма

double f(double x) {

return std::log10(x);

}

// Производная функции f

double df(double x) {

return 1 / (x \* std::log(10));

}

// Функция для вычисления полинома Лагранжа

double lagrangePolynomial(double x, const std::vector<double>& xk, const std::vector<double>& yk) {

int n = xk.size();

double p = 0.0;

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

double l = 1.0;

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

if (i != j) {

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

}

}

p += yk[i] \* l;

}

return p;

}

// Функция для оценки погрешности интерполяции

double errorEstimate(double x, const std::vector<double>& xk, double (\*df)(double)) {

int n = xk.size();

// Находим максимальное значение производной на интервале

double M = 0.0;

for (double xi = xk.front(); xi <= xk.back(); xi += (xk.back() - xk.front()) / 1000) {

M = std::max(M, std::abs(df(xi)));

}

// Вычисляем произведение (x - xk[i])

double w = 1.0;

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

w \*= (x - xk[i]);

}

return (M \* std::abs(w)) / std::tgamma(n + 1); // std::tgamma(n+1) = n!

}

int main() {

// Узлы интерполяции

std::vector<double> xk = {7, 7.5, 8, 8.5};

std::vector<double> yk;

// Заполняем yk значениями функции f

for (double x : xk) {

yk.push\_back(std::round(f(x) \* 10000) / 10000); // округляем до 4 знаков

}

// Точка, в которой будем вычислять значение функции

double x = 7.2;

// Вычисляем аналитическое значение функции в точке x

double yAnalytical = f(x);

// Вычисляем значение полинома Лагранжа в точке x

double yLagrange = lagrangePolynomial(x, xk, yk);

// Оценка погрешности и абсолютная погрешность

double error = errorEstimate(x, xk, df);

double absoluteError = std::abs(yAnalytical - yLagrange);

// Вывод результатов

std::cout << std::fixed << std::setprecision(4);

for (size\_t i = 0; i < xk.size(); ++i) {

std::cout << "Значение функции в точке x = " << xk[i] << ": " << yk[i] << std::endl;

}

std::cout << "\n";

std::cout << "Значение функции в точке x = " << x << " Аналитически: " << yAnalytical << std::endl;

std::cout << "Значение функции в точке x = " << x << " По ЛАгранжу: " << yLagrange << std::endl;

std::cout << "Абсолютная погрешность вычислений: " << absoluteError << std::endl;

std::cout << "Оценка погрешности формулы Лагранжа: " << error << std::endl;

return 0;

}

**Задание 2**

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

#include <string>

// Определяем функцию f(x) = exp(x/2)

double f(double x) {

return std::exp(x / 2);

}

// Определяем первую производную функции f(x)

double df(double x) {

return std::exp(x / 2) / 2;

}

// Определяем вторую производную функции f(x)

double d2f(double x) {

return std::exp(x / 2) / 4;

}

// Структура для хранения результатов

struct Result {

double x;

double f\_prime\_left;

double f\_prime\_right;

double f\_prime\_center;

double exact\_first\_derivative;

double error\_left;

double error\_right;

double error\_center;

double f\_second;

double exact\_second\_derivative;

double error\_second;

};

int main() {

// Задаем границы интервала и количество точек

double a = 3.4;

double b = 4.3;

int n = 5;

// Вычисляем шаг между точками

double h = (b - a) / (n - 1);

// Генерируем n точек на интервале [a, b]

std::vector<double> x(n);

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

x[i] = a + i \* h;

}

// Вектор для хранения результатов

std::vector<Result> results;

// Вычисляем производные и ошибки для каждой точки

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

double xi = x[i];

// Вычисляем точные значения первой и второй производных

double exact\_first\_derivative = df(xi);

double exact\_second\_derivative = d2f(xi);

// Вычисляем приближенные значения первой производной

double f\_prime\_left = (i > 0) ? (f(xi) - f(xi - h)) / h : NAN;

double f\_prime\_right = (i < n - 1) ? (f(xi + h) - f(xi)) / h : NAN;

double f\_prime\_center = (i > 0 && i < n - 1) ? (f(xi + h) - f(xi - h)) / (2 \* h) : NAN;

// Вычисляем приближенное значение второй производной только для промежуточных точек

double f\_second = (i > 0 && i < n - 1) ? (f(xi + h) - 2 \* f(xi) + f(xi - h)) / (h \* h) : NAN;

// Вычисляем ошибки между точными и приближенными значениями производных

double error\_left = (i > 0) ? std::abs(exact\_first\_derivative - f\_prime\_left) : NAN;

double error\_right = (i < n - 1) ? std::abs(exact\_first\_derivative - f\_prime\_right) : NAN;

double error\_center = (i > 0 && i < n - 1) ? std::abs(exact\_first\_derivative - f\_prime\_center) : NAN;

double error\_second = (i > 0 && i < n - 1) ? std::abs(exact\_second\_derivative - f\_second) : NAN;

results.push\_back({

xi,

f\_prime\_left,

f\_prime\_right,

f\_prime\_center,

exact\_first\_derivative,

error\_left,

error\_right,

error\_center,

f\_second,

exact\_second\_derivative,

error\_second

});

}

// Выводим результаты в таблице

std::cout << std::fixed << std::setprecision(4);

std::cout << " x f'(x) (Левая) f'(x) (Правая) f'(x) (Центр) Точная f'(x) Погрешность (Левая) Погрешность (Правая) Погрешность (Центр) f''(x) Точная f''(x) Погрешность\n";

for (const auto& result : results) {

std::cout << std::setw(6) << result.x << " "

<< std::setw(12) << (std::isnan(result.f\_prime\_left) ? "N/A" : std::to\_string(result.f\_prime\_left).substr(0, 6)) << " "

<< std::setw(11) << (std::isnan(result.f\_prime\_right) ? "N/A" : std::to\_string(result.f\_prime\_right).substr(0, 6)) << " "

<< std::setw(15) << (std::isnan(result.f\_prime\_center) ? "N/A" : std::to\_string(result.f\_prime\_center).substr(0, 6)) << " "

<< std::setw(12) << result.exact\_first\_derivative << " "

<< std::setw(14) << (std::isnan(result.error\_left) ? "N/A" : std::to\_string(result.error\_left).substr(0, 6)) << " "

<< std::setw(16) << (std::isnan(result.error\_right) ? "N/A" : std::to\_string(result.error\_right).substr(0, 6)) << " "

<< std::setw(18) << (std::isnan(result.error\_center) ? "N/A" : std::to\_string(result.error\_center).substr(0, 6)) << " "

<< std::setw(18) << (std::isnan(result.f\_second) ? "N/A" : std::to\_string(result.f\_second).substr(0, 6)) << " "

<< std::setw(12) << result.exact\_second\_derivative << " "

<< std::setw(13) << (std::isnan(result.error\_second) ? "N/A" : std::to\_string(result.error\_second).substr(0, 6)) << "\n";

}

return 0;

}