

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕТОДЫ ЧИСЛЕННОГО АНАЛИЗА

ЛАБОРАТОРНАЯ РАБОТА 1

ИНТЕРПОЛЯЦИОННЫЙ МНОГОЧЛЕН ЛАГРАНЖА

Рак Алексей

25 мая 2017 г.

Постановка Задачи

Для заданной функции $f(x)$ на равномерной сетке узлов построить интерполяционный многочлен Лагранжа $P_n(x)$ и вычислить:

- 1) $P_n(x^*), P_n(x^{**}), P_n(x^{***})$
- 2) Погрешность $R_n(x^*), R_n(x^{**}), R_n(x^{***})$
- 3) Абсолютные погрешности $R_{true}(x^*), R_{true}(x^{**}), R_{true}(x^{***})$

Данные

Функция: $f(x) = 1.3 * e^x - 0.3 * \sin(x)$

Равномерная сетка узлов: $[1, 2]$

Шаг 0.1

$x^* = \frac{31}{30}, x^{**} = \frac{46}{30}, x^{***} = \frac{59}{30}$

Алгоритмы решения и формулы

Рассмотрим многочлены $l_i(x_i)$ степени n , удовлетворяющие условиям:

$$l_i(x_i) = \sigma_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

которые называются многочленами Лагранжа. Их можно записать в виде:

$$l_i(x_i) = \sigma_{ij} = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}$$

Тогда интерполяционный многочлен в форме Лагранжа принимает вид: $P_n(x) = \sum_{i=0}^n l_i(x) f_i$. Введём следующее обозначение: $w(x) = \prod_{j=0}^n (x - x_j)$. Легко заметить, что $\prod_{j=0, j \neq i}^n (x_i - x_j) = w'(x_i)$. Тогда вид многочлена Лагранжа будет следующим:

$$P_n(x) = \sum_{i=0}^n \frac{w(x)}{(x - x_i)w'(x_i)} f_i$$

Формулы оценки для погрешности:

- 1) $R_{true}(x) = f(x) - P_n(x)$
- 2) $|R_n(x)| \leq \frac{M}{(n+1)!} |w(x)|$
- 3) $|P_n(x)| \leq \frac{Mh^{n+1}}{4(n+1)}$

Листинг

```
#include <cmath>
#include <iomanip>
#include <iostream>

const double x0 = 1;
const double step = 0.1;
const int n = 10;
const double alpha = 1.3;
const double df11max = alpha * exp(2) - (1 - alpha) * cos(1);

double func(double x) {
    return alpha * exp(x) + (1 - alpha) * sin(x);
}

double Lagranj(double x) {
    double ans = 0;
    for (int i = 0; i <= n; i++) {
```

```

        double term = 1, xi = x0 + i * step;
        for (int j = 0; j <= n; j++) {
            if (i == j) {
                continue;
            }
            double xj = x0 + j * step;
            term = term * (x - xj) / (xi - xj);
        }
        term *= func(xi), ans += term;
    }
    return ans;
}

double calcRn(double x) {
    double ans = df11max;
    for (int i = 0; i <= n; i++)
        ans = ans * fabs(x0 + step * i - x) / (i + 1);
    return ans;
}

double calcR_true(double x) {
    return fabs(Lagranj(x) - func(x));
}

int main() {
    double x = 31.0 / 30;
    double xx = 46.0 / 30;
    double xxx = 59.0 / 30;

    std::cout << "P1 " << std::setprecision(15) << Lagranj(x) << std::endl;
    std::cout << "P2 " << std::setprecision(15) << Lagranj(xx) << std::endl;
    std::cout << "P3 " << std::setprecision(15) << Lagranj(xxx) << std::endl;
    std::cout << "R1 " << std::setprecision(15) << calcR_true(x) << std::endl;
    std::cout << "R2 " << std::setprecision(15) << calcR_true(xx) << std::endl;
    std::cout << "R3 " << std::setprecision(15) << calcR_true(xxx) << std::endl;
    std::cout << std::setprecision(15) << df11max * pow(0.1, 11) / 44 << std::endl;
    std::cout << "R1 " << std::setprecision(15) << calcRn(x) << std::endl;
    std::cout << "R2 " << std::setprecision(15) << calcRn(xx) << std::endl;
    std::cout << "R3 " << std::setprecision(15) << calcRn(xxx) << std::endl;
    return 0;
}

```

Результаты и вывод

Выходные данные:

$$P_n(x^*) = 3.39584070365707$$

$$P_n(x^{**}) = 5.72388585590385$$

$$P_n(x^{***}) = 9.01405984385184$$

$$R_n(x^*) \leq 1.00345724778056e^{-12}$$

$$R_n(x^{**}) \leq 9.91156290249787e^{-15}$$

$$R_n(x^{***}) \leq 1.00345724778056e^{-12}$$

$$R_{true}(x^*) = 5.90194559890733e^{-13}$$

$$R_{true}(x^{**}) = 5.32907051820075e^{-15}$$

$$R_{true}(x^{***}) = 6.21724893790088e^{-13}$$

Вывод:

Метод Лагранжа нахождения интерполяционного многочлена позволяет достаточно точно интерполировать функцию, при этом абсолютная погрешность не превосходит $1e^{-12}$ в рассматриваемых точках. Наиболее точные значения получаются вблизи центрального узла таблицы.