Белорусский Государственный Университет

Методы Численного Анализа

Лабораторная работа 4

Интерполирование с помощью многочлена Чебышева

Рак Алексей

25 мая 2017 г.

Постановка Задачи

Минимизировать остаток интерполирования, используя многочлен Чебышева. Для этого с его помощью выберем узлы интерполирования на нужном отрезке $[1, 2]$ и проведём по ним интерполяцию многочленом Лагранжа. Вычислим его значения в контрольных точках x^*, x^{**}, x^{***}. Оценим результат и сравним его с предыдущим.

Данные

Функция: $f(x) = 1.3 * e^x - 0.3 * sin(x)$
$x^* = \frac{31}{30}$, $x^{**} = \frac{46}{30}$, $x^{***} = \frac{59}{30}$

Алгоритм решения и формулы

Идея метода состоит в выборе таких узлов для построения интерполяционного полинома, чтобы в оценке:

$$\|f(x) - P_n(x)\| \leq \frac{\|f^{(n+1)}(x)w(x)\|}{(n+1)!}$$

многочлен $w(x)$ был наименее отклоняющимся от нуля, что достигается на Чебышёвском наборе узлов.

$$x_m = \frac{a+b}{2} + \frac{b-a}{2}cos\left(\frac{\pi}{n+1}\left(m + \frac{1}{2}\right)\right) \qquad m = \overline{0, n}$$

В таком случае оценка погрешности принимает вид:

$$\|f(x) - P_n(x)\| \leq \frac{\|f^{(n+1)}(x)\|(b-a)^{n+1}2^{1-2(n+1)}}{(n+1)!}$$

Листинг

```
#include <cmath>
#include <iostream>
#include <vector>

const int n = 11;
const double a = 1.3;
const double x1 = 31.0 / 30;
const double x2 = 46.0 / 30;
const double x3 = 59.0 / 30;
std::vector<double> nodes(n);
std::vector<double> fx(n);
const double right = 2;
const double left = 1;

double f(double x) {
    return a * exp(x) + (1 - a) * sin(x);
}

double f1(double x) {
    return a * exp(x) + (1 - a) * cos(x);
}

double f2(double x) {
    return a * exp(x) - (1 - a) * sin(x);
}

double f3(double x) {
```

```
    return a * exp(x) - (1 - a) * cos(x);
}

int factorial(int x) {
    if (x == 1 || x == 0)
        return 1;
    return x * factorial(x - 1);
}

double Lagranj(double x) {
    double ans = 0;
    for (int i = 0; i < n; i++) {
        double term = 1, xi = nodes[i];
        for (int j = 0; j < n; j++) {
            if (i == j) continue;
            double xj = nodes[j];
            term = term * (x - xj) / (xi - xj);
        }
        term *= f(xi);
        ans += term;
    }
    return ans;
}

void createNodes() {
    double add = (right + left) / 2., mul = (right - left) / 2.;
    for (int i = 0; i < n; ++i) {
        nodes[i] = add + mul * cos(M_PI * (2 * (n - i) + 1) / 2 / (n + 1));
        fx[i] = f(nodes[i]);
    }
}

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

double findTheoreticalResidual() {
    double mul = (right - left) / 2.;
    for (int i = 1; i < n; ++i)
        mul *= ((right - left) / (i + 1) / 4.);
    switch (n % 4) {
        case 0:
            return f1(right) * mul;
        case 1:
            return f2(right) * mul;
        case 2:
            return f3(right) * mul;
        case 3:
            return f(right) * mul;
    }
    return 0;
}

int main() {
    createNodes();
    std::cout << "nodes: " << std::endl;
    for (int i = 0; i < n; i++) {
        std::cout << nodes[i] << " ";
    }
    std::cout << std::endl;
    std::cout << "expected: " << findTheoreticalResidual() << std::endl;
    std::cout << "x*:" << std::endl;
    std::cout << "resalt: " << Lagranj(x1) << std::endl;
```

```
    std::cout « "true: " « calcR_true(x1) « std::endl;
    std::cout « "x**:" « std::endl;
    std::cout « "resalt: " « Lagranj(x2) « std::endl;
    std::cout « "true: " « calcR_true(x2) « std::endl;
    std::cout « "x***:" « std::endl;
    std::cout « "resalt: " « Lagranj(x3) « std::endl;
    std::cout « "true: " « calcR_true(x3) « std::endl;
    return 0;
}
```

Результаты и вывод

Выходные данные:
Ноды:
1.004281.038061.103321.195621.308661.434741.565261.691341.804381.896681.96194
Ожидаемая погрешность: $1.1149e^{-13}$
Точка x^{*}:
Результат: 3.39584
Истинная погрешность: $4.44089e^{-15}$
Точка x^{**}:
Результат: 5.72389
Истинная погрешность: $2.4869e^{-14}$
Точка x^{***}:
Результат: 9.01406
Истинная погрешность: $1.84741e^{-13}$
Вывод:
Данный метод является одним из самых точных методов из числа рассмотренных, в силу того, что в этом методе происходит выбор самих узлов. Выбранные узлы являются корнями многочлена Чебышева, а многочлен Чебышева есть наиболее близкая к нулю функция, то есть мы приближаем остаток интерполирования к нулю наилучшим образом.
Во всех точках погрешность стала меньше, это наиболее заметно в первой точке, чуть менее заметно во второй точке и практически не заметно в третей. Это можно списать на то, что вначале рассматриваемого отрезка "концентрация узлов"вышла более "плотной поэтому точность понижается при приближении к третьей точке.