Белорусский Государственный Университет

Методы Численного Анализа

Лабораторная работа 6

Интерполирование сплайном третьего порядка

Рак Алексей

28 мая 2017 г.

Постановка Задачи

По заданной равномерной сетке узлов $x_i \in [1,2], x_i = 1 + ih, h = \frac{2-1}{n}, i = \overline{0,n}, n = 10$ для функции $f(x) = 1.3e^x - 0.3\sin x$ требуется построить интерполяционный кубический сплайн. Рассчитать истинную погрешность в точках:

$$x^* = x_0 + \frac{h}{3},\ x^{**} = x_5 + \frac{h}{3},\ x^{***} = x_{10} - \frac{h}{3}$$

Алгоритм решения и формулы

Строим систему следующего вида для вычитания моментов $M_i = S''(x_i)$

$$\begin{cases} 2M_0 - \lambda_0 M_1 = d_0, i = 0 \\ -\nu M_{i-1} + 2M_i - \lambda_i M_{i+1} = d_i, i = \overline{1, n-1} \\ -\nu_n M_{n-1} + 2M_n = d_n, i = n \end{cases}$$

2 дополнительных условия зададим в виде:

$$\begin{cases} d_0 = 2f''(x_0) \\ d_n = 2f''(x_n) \end{cases}$$

откуда

$$\begin{cases} \lambda_0 = 0 \\ \nu_n = 0 \end{cases}$$

Остальные коэффициенты находим по формулам:

$$\lambda_i = -\frac{h_i + 1}{h_i + h_{i+1}} = -0.5, i = \overline{1, n-1}$$
$$\nu_i = -\frac{h_i}{h_i + h_{i+1}} = -0.5, i = \overline{1, n-1}$$
$$d_i = \frac{3}{h}\left(\frac{f_{i+1} - f_i}{h} - \frac{f_i - fi - 1}{h}\right), i = \overline{1, n-1}$$

Решим данную систему методом левой прогонки и найдём моменты M_i.
Интерполяционный сплайн на отрезке $[x_{i-1}, x_i], i = \overline{1,n}$ строим по формуле:
$S_{i-1}(x) = M_{i-1}\frac{(x_i - x)^3}{6h} + M_i\frac{(x - x_{i-1})^3}{6h} + \frac{x_i - x}{h}\left(f_{i-1} - \frac{M_{i-1}h^2}{6}\right) + \frac{x - x_{i-1}}{h}\left(f_i - \frac{M_i h^2}{6}\right)$
Для точек x^*, x^{**}, x^{***} считаем истинную погрешность:
$R_{true}(x) = |f(x) - s_k(x)|$, где $x_k \leq x \leq x_{k+1}$

Листинг

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

class Spline {
public:
    const double coef = 1.3;
    const int N = 10;
    double A, B, h;
    double *x, *f, *l, *v, *d, *c;
```

```
    Spline() {
        A = 1, B = 2, h = 0.1;
        x = new double[N + 1];
        f = new double[N + 1];
        l = new double[N];
        v = new double[N];
        d = new double[N + 1];
        c = new double[N + 1];
    }

    void initSweepCoefs() {
        for (int i = 0; i < N + 1; i++) {
            x[i] = A + i * h;
            f[i] = coef * exp(Spline::x[i]) + (1 - coef) * sin(Spline::x[i]);
        }
        l[0] = 0;
        v[N - 1] = 0;
        c[0] = c[N] = 2;
        d[0] = 2 * (coef * exp(A) - (1 - coef) * sin(A));
        d[N] = 2 * (2.1 * exp(B) - (1 - coef) * sin(B));
        for (int i = 0; i < N - 1; i++) {
            l[i + 1] = v[i] = -0.5;
            c[i + 1] = 2;
            d[i + 1] = (3 / h) * (((f[i + 2] - f[i + 1]) / h) - ((f[i + 1] - f[i]) / h));
        }
    }

    double S(int i, double x, double *mom) {
        return (mom[i - 1] * pow(Spline::x[i] - x, 3) / 6 * h) +
               (mom[i] * pow(x - Spline::x[i - 1], 3) / 6 * h) +
               ((Spline::x[i] - x) / h) * (f[i - 1] - ((mom[i - 1] * pow(h, 2)) / 6)) +
               ((x - Spline::x[i - 1]) / h) * (f[i] - ((mom[i] * pow(h, 2)) / 6));
    }

    double *solveMatrix() {
        double *x = new double[N + 1];
        double m;
        for (int i = 1; i < N + 1; i++) {
            m = v[i] / c[i - 1];
            c[i] = c[i] - m * l[i - 1];
            d[i] = d[i] - m * d[i - 1];
        }
        x[N] = d[N] / c[N];
        for (int i = N - 1; i >= 0; i-)
            x[i] = (d[i] - l[i] * x[i + 1]) / c[i];
        return x;
    }
};

int main() {
    Spline *spline = new Spline();
    spline->initSweepCoefs();
    double *moments = spline->solveMatrix();
    double *xch = new double[3];
    double *fch = new double[3];
    double *Rtrue = new double[3];
    double *Sch = new double[3];
    xch[0] = spline->x[0] + (spline->h / 3);
    xch[1] = spline->x[5] + (spline->h / 3);
    xch[2] = spline->x[10] - (spline->h / 3);
    int ind[3] = {1, 6, 10};
    for (int i = 0; i < 3; i++) {
        fch[i] = spline->coef * exp(xch[i]) + (1 - spline->coef) * sin(xch[i]);
```

```
        Sch[i] = spline->S(ind[i], xch[i], moments);
        Rtrue[i] = fabs(fch[i] - Sch[i]);
        std::cout « "f[" « i « "]: " « fch[i] « " " « "S[" « i « "]: " « Sch[i] « " "
                  « "Pogr[" « i « "]: " « Rtrue[i] « std::endl;
    }
    return 0;
}
```

Результаты и вывод

Выходные данные:
f[0]: 3.39584 S[0]: 3.3902 Pogr[0]: 0.00564207
f[1]: 5.72389 S[1]: 5.69945 Pogr[1]: 0.0244403
f[2]: 9.01406 S[2]: 8.99745 Pogr[2]: 0.0166138
Вывод:
Исходя из полученных результатов, можно заметить, что погрешность вычислений имеет большую величину. Это объясняется тем, что на каждом отрезке строится полином третьей степени (например, в других работах строились полиномы больших степеней).
Преимущество данного подхода заключается в том, что строится несколько полиномов для всех маленьких отрезков, а не один для всего отрезка интерполирования. Этот метод намного лучше отражает поведение функции на каждом из маленьких отрезков.