БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Лабораторная работа 12

Интерполирование функции с использованием кубического сплайна

Вариант 7

Выполнил:
Журик Никита Сергеевич
2 курс, 6 группа
Преподаватель:
Будник Анатолий Михайлович

Минск, 2019

Содержание

1. Постановка задачи

1. Интерполировать исходную функцию кубическим сплайном;
2. Вычислить теоретическую оценку и действительную невязку интерполирования;
3. Проанализировать результаты и сравнить с методом интерполирования по равномерной сетке.

2. Алгоритм решения

- Рассмотрим задачу интерполирования исходной функции $f(x)$ при помощи естественного кубического сплайна на равномерной сетке узлов. Отсюда $M_0 = M_N = 0; h_i = \delta = 0.1, i = \overline{1, N}; N = 10$.
- Кубический сплайн может быть построен по формуле:

$$S_3(x) = \frac{M_{i-1}}{6\delta}(x_i - x)^3 + \frac{M_i}{6\delta}(x - x_{i-1})^3 + \frac{(x - x_{i-1})}{\delta}\left(f_i - M_i\frac{\delta^2}{6}\right) + \frac{x_i - x}{6}\left(f_{i-1} - M_{i-1}\frac{\delta^2}{6}\right) \quad (1)$$

- Для нахождения моментов M_i рассмотрим следующую СЛАУ:

$$\frac{\delta}{6}M_{i-1} + \frac{2\delta}{3}M_i + \frac{\delta}{6}M_{i+1} = \frac{f_{i+1} - f_i}{\delta} - \frac{f_i - f_{i-1}}{\delta}, i = \overline{1, N-1} \quad (2)$$

  Решив данную СЛАУ, построим искомый кубический сплайн.

- Невязку оценим по формуле

$$|r_3(x)| \leq \delta^4 \max_{x\in[a,b]} |f^{(4)}(x)| \quad (3)$$

3. Листинг программы

Для реализации алгоритма был использован Python и библиотеки numpy и matplotlib.

```
#CubicSpline.py

import numpy as np
from math import exp, log, factorial
import matplotlib.pyplot as plt

a = 1.0
b = 2.0
N = 10
delta = (b - a) / N
alpha = 1.7

points = [a + i * delta for i in range(N + 1)]

def f(x):
    return alpha * exp(-x) + (1 - alpha) * log(x)

def fDeriv4(x):
    return alpha * exp(-x) - (1 - alpha) * factorial(3) / x ** 4

def maxDeriv4(samples):
    space = np.linspace(a, b, samples)
    return np.max(np.abs(np.array([(fDeriv4(x)) for x in space], dtype=np.double)))

def S3(M, k):
    return lambda x: (M[k - 1] * (points[k] - x) ** 3 / (6 * delta) + M[k] * (x - points[k - 1])
        ** 3 / (6 * delta)
    + (x - points[k - 1]) * (f(points[k]) - M[k] * delta ** 2 / 6) / delta
    + (points[k] - x) * (f(points[k - 1]) - M[k - 1] * delta ** 2 / 6) / delta)

def findMoments():
    A = np.zeros((N + 1, N + 1), dtype = np.double)
    b = np.zeros(N + 1, dtype = np.double)
```

```
    A[0][0] = A[N][N] = 1
    b[0] = b[N] = 0
    for i in range(1, N):
        A[i][i - 1] = delta / 6
        A[i][i] = 2 * delta / 3
        A[i][i + 1] = delta / 6
        b[i] = (f(points[i + 1]) - f(points[i])) / delta - (f(points[i]) - f(points[i - 1])) / delta
    return np.linalg.solve(A, b)

def deficiency():
    return maxDeriv4(10000) * delta ** 3

if __name__ == "__main__":
    M = findMoments()

    check = [points[0] + delta / 2.6]

    [print("S3({0}) = {1}".format(x, S3(M, 1)(x))) for x in check]
    print()

    [print("r3({0}) = {1}".format(x, f(x) - S3(M, 1)(x))) for x in check]
    print()

    print("Expected deficiency on whole interval: " + str(deficiency()))

    print("Real deficiency in control points: " +
          str(np.max(np.abs(np.array([(f(x) - S3(M, 1)(x)) for x in check], dtype=np.double)))))
```

4. Вывод программы

```
S3(1.0384615384615385) = 0.5760276330461487

r3(1.0384615384615385) = -0.0006477719097222057

Expected deficiency on whole interval: 0.0048253950499914525
Real deficiency in control points: 0.0006477719097222057
```

5. Выводы

- В результате применения формул для равномерной сетки узлов удалось интерполировать исходную функцию многочленом третьей степени с точностью $r_{EqSp_{control}} = 1.328510657672144e-05$ (в рассматриваемой точке x^*). При помощи кубического сплайна была достигнута точность $r_{Cube_{control}} = 6.477719097222057e-04$.
- Таким образом, погрешность интерполирования при использовании кубического сплайна больше. Преимуществом данного метода по сравнению с рассмотренным ранее является отсутствие необходимости рассматривать различные случаи местоположения точки интерполирования, что делает интерполирование при помощи кубического сплайна более универсальным методом.