Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Московский государственный технический университет имени Н.Э. Баумана

Лабораторная работа №3
по курсу «Численные методы»
«Интерполяция B-сплайнами»

Выполнил:
студент группы ИУ9-62Б
Головкин Дмитрий

Проверила:
Домрачева А.Б.

Москва, 2023

Цель:

Анализ метода интерполяции функции, основанный на построении кубического в контрольных точках.

Постановка задачи:

Дано: Функция $y_i = \phi(x_i), i = \overline{1,n}$ задана таблично, исходные данные включают ошибки измерения.

x_1	x_2	...	x_{n-1}	x_n
y_1	y_2	...	y_{n-1}	y_n

Найти: Функцию (интерполянту) $f(x)$, совпадающую с значениями $y_i, i = \overline{1,n}$ в контрольных точках $x_i, i = \overline{1,n}$:

$$f(x_i) = y_i$$

Тестовый пример:

Зададим некоторую функцию $\phi(x)$ таблично:

0.5	0.75	1.0	1.25	1.5	1.75	2.0	2.25	2.5	2.75
2.78	2.95	3.13	3.31	3.c51	3.72	3.94	4.18	4.43	4.69

3.0	3.25	3.5	3.75	4.0	4.25	4.5	4.75	5.0	5.25	5.5
4.97	5.27	5.58	5.92	6.27	6.65	7.04	7.47	7.91	8.38	8.89

Теоретические сведения:

Интерполяция, интерполирование — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.

Основная цель интерполяции — получить быстрый (экономичный) алгоритм вычисления значений $y(x)$ для значений x, не содержащихся в таблице данных. Интерполируюшие функции $f(x)$, как правило строятся в виде линейных комбинаций некоторых элементарных функций:

$$f(x) = \sum_{k=0}^{N} c_k \Phi_k(x)$$

где $\{\Phi_k(x)\}$ — фиксированный линейно независимые функции, $c_0, c_1, \cdots, c_n$ — не определенные пока коэффициенты.

Один из способов интерполирования на всем отрезке $[a, b]$ является интерполирование сплайнами. Сплайном называется кусочно-полиномиальная функция, определенная на отрезке $[a, b]$ и имеющая на этом отрезке некоторое количество непрерывных производных. Преимущества интерполяции сплайнами по сравнению с обычными методами интерполяции – в сходимости и устойчивости вычислительного процесса.

Рассмотрим один из наиболее распространенных в практике случаев – интерполирование функции кубическим сплайном.

Пусть на отрезке $[a, b]$ задана непрерывная функция $y(x)$. Введем разбиение отрезка:

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

и обозначим $y_i = y(x_i), i = \overline{1, n}$

Сплайном, соответствующим данной функции узлам интерполяции называется функция $s(x)$, удовлетворяющая следующим условиям:

1. На каждом отрезке $[x_{i-1}; x_i], i = \overline{2, n}$ функция $s(x)$ является кубическим многочленом;
2. Функция $s(x)$, а также ее первая и вторая производные непрерывны на отрезке $[a, b]$;
3. $s(x_i) = y_i, i = \overline{2, n}$ - условие интерполирования.

Сплайн, определяемый условиями 1 – 3, называется интерполяционным кубическим сплайном и имеет вид:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Заметим несколько важных условий: $S_i'(x_i) = S_{i+1}'(x_i)$, $S_i'(x_i) = S_{i+1}'(x_i)$, $S_1''(a) = 0$, $S_n''(b) = 0$. Последние два условия называют условиями гладкости на краях. Пусть n - число разбиений на отрезке $[a; b]$. Тогда зададим параметр h следующим образом : $h = \frac{b-a}{n}$

Теперь, используя данные условия, можем получить следующие соотношения:

1. $d_i = \frac{c_(i+1)-c_i}{3h}, i = \overline{1, n}$;
2. Используя условия гладкости на краях получаем: $c_1 = 0$, $c_{n+1} = 0$;
3. $a_i = y_{i-1}, i = \overline{1, n}$;
4. $b_i = \frac{y_i - y_{i-1}}{h} - \frac{h}{3}(c_{i+1} + 2c_i)$;

Следующим шагом необхожимо вычислить коэффициенты c_i. Преобразуя далее функцию сплайна, получаем следующую систему:

$$\begin{cases} 4c_2 + c_3 = \frac{3}{h^2}(y_2 - 2y_1 - y_0) \\ \dots \\ c_i + 4c_{i+1} + c_{i+2} = \frac{3}{h^2}(y_{i+1} - 2y_i - y_{i-1}) \\ \dots \\ c_{n-1} + 4c_n = \frac{3}{h^2}(y_n - 2y_{n-1} - y_{n-2}) \end{cases}$$

Данные коэффициенты будем искать с помощью метода прогонки.

Практическая реализация: Листинг 1. Интерполяция кубическими сплайнами

```
#!python
# -*- coding: utf-8 -*-
import numpy as np

def calculate_x_n(a, b, c, d, n):
    #alpha = [0.0] * n
    #beta = [0.0] * n
    alpha = np.zeros(n)
    beta = np.zeros(n)
    alpha[0] = -c[0] / b[0]
    beta[0] = d[0] / b[0]
    for i in range(1,n-1):
        #print(i)
        alpha[i] = -c[i] / (a[i-1]*alpha[i-1] + b[i])
        beta[i] = (d[i] - a[i-1]*beta[i-1]) / (a[i-1]*alpha[i-1] + b[i])
    x_n = (d[n-1] - a[n-2]*beta[n-2]) / (a[n-2]*alpha[n-2] + b[n-1])
    #print(alpha, beta)
    return x_n, alpha, beta

def calculate_c_n_vector(x_n, alpha, beta, n):
    x = [0.0] * (n-1)
    x.append(x_n)
    for i in range(n-2, -1, -1):
        x[i] = alpha[i]*x[i+1] + beta[i]
    return x

def calculate_d(y_n, h, n):
    d = [y_n[0] / 3 * (h*h)]
    for i in range(2, n+1):
        d.append(y_n[i] - 2*y_n[i-1] + y_n[i-2])
    return (3/(h*h)) * np.asarray(d)



a = 0
b = 1
n = 9
a_coefs = [1.0] * (n-1)
b_coefs = [4.0] * n
```

```
c_coefs = [1.0] * (n-1)
x_n = np.arange(a, b + b/18, b/9)
y_n = np.exp(x_n)
print('X: ',x_n)
print('Y: ',y_n)

h = (b - a) / n
d_coefs = calculate_d(y_n, h, n)
#print("---", d_coefs)
spl, alpha, beta = calculate_x_n(a_coefs, b_coefs, c_coefs, d_coefs, n)
#print(spl, alpha, beta)

c_n = calculate_c_n_vector(spl, alpha, beta, n)
c_n.append(0)
c_n = np.asarray(c_n)
print('c_coefs: ', c_n)
d_n = (c_n[1:n+1] - c_n[:n]) / (3*h)
print('d_coefs: ',d_n)
a_n = y_n[:n]
print('a_coefs: ',a_n)
b_n = (y_n[1:n+1] - y_n[:n]) / h - (h / 3) * (c_n[1:n+1] + 2*c_n[:n])
print('b_coefs: ',b_n)

x_n_new = np.arange(a, b + b/9, b/18)
y_n_new = np.exp(x_n_new)
spline_extended = [a_n[(idx-1)//2] + b_n[(idx-1)//2] * (x_n_new[idx] -x_n[(idx
                   d_n[(idx-1)//2] * (x_n_new[idx] - x_n[(idx-1)//2])**3 for
spline_extended = np.concatenate((a_n[:1], spline_extended))

spline = a_n + b_n*(x_n[1:n+1] - x_n[:n]) + c_n[:n]*(x_n[1:n+1] - x_n[:n])**2
         d_n*(x_n[1:n+1] - x_n[:n])**3
spline = np.concatenate((a_n[:1], spline))

for i in range(0,n+1):
    print("x=", x_n[i], " | f(x)=", y_n[i], " | spl(x)=",spline[i], " | delta=

for i in range(0,2*n+1):
    print("x=", x_n_new[i], " | f(x)=", y_n_new[i], " | spl(x)=",spline_extend
```

Результаты:

Для тестирования полученной программы была задана табличная функция:

0.5	0.75	1.0	1.25	1.5	1.75	2.0	2.25	2.5	2.75
2.78	2.95	3.13	3.31	3.c51	3.72	3.94	4.18	4.43	4.69

3.0	3.25	3.5	3.75	4.0	4.25	4.5	4.75	5.0	5.25	5.5
4.97	5.27	5.58	5.92	6.27	6.65	7.04	7.47	7.91	8.38	8.89

В результате работы программы (Листинг 1) получаем значения:

Значение x_n	Значение y_n	Значение сплайна $s^3(x_n)$	Разница значений $s^3(x_n) - y_n$
0.5	2.78	2.7799999999999994	-4.440892098500626e-16
0.75	2.95	2.952961581335141	0.002961581335140906
1.0	3.13	3.13	0.0
1.25	3.31	3.3148652559945737	0.004865255994573658
1.5	3.51	3.51	0.0
1.75	3.72	3.7175773946865607	-0.0024226053134395187
2.0	3.94	3.94	0.0
2.25	4.18	4.178575165259181	-0.0014248347408187811
2.5	4.43	4.43	0.0
2.75	4.69	4.691871944276711	0.0018719442767105576
3.0	4.97	4.97	0.0
3.25	5.27	5.268937057633973	-0.0010629423660262205
3.5	5.58	5.58	0.0
3.75	5.92	5.901129825187394	-0.01887017481260589
4.0	6.27	6.27	0.0
4.25	6.65	6.69654364161645	0.04654364161644953
4.5	7.04	7.039999999999999	-8.881784197001252e-16
4.75	7.47	7.265195608346805	-0.20480439165319453
5.0	7.91	7.909999999999998	-1.7763568394002505e-15
5.25	8.38	9.118923924996324	0.7389239249963229
5.5	8.89	8.889999999999999	-1.7763568394002505e-15

Выводы:

В ходе выполнения лабораторной работы был рассмотрен метод интерполяции функции, основанный на построении кубического сплайна в контрольных точках, так же для метода была написана реализация на языке программирования Python.

Анализируя результаты полученной программы, можно заметить то, что метод интерполяции функции, основанный на построении кубического в контрольных точках, имеет высокую точность вследствие низкой погрешности. Повышая степень кусочно-интерполяционного многочлена, можно добиться еще лучших результатов аппроксимации функции.