МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных систем и технологий

Кафедра прикладной математики

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ № 2

ПО ДИСЦИПЛИНЕ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА»

СТУДЕНТА 2 КУРСА бакалавриата ГРУППЫ ИДБ-22-05

(уровень профессионального образования)

Моряков Антон

НА ТЕМУ

Интерполирование кубическим сплайном дефекта вариант №13

Направление:
Профиль подготовки:

Отчет сдан «_» ___ 2024г.

Оценка _____

Проверил: преподаватель Стихова О.В. _____

(Ф.И.О., должность, степень, звание) *(подпись)*

МОСКВА 2024

Оглавление

Изучение метода интерполяции кубическим сплайном дефекта 1

Дано: $y(x) \in [a,b];\ y(n+1) = x_i,\ i = \overline{0,n}; x_i = a + ih, h = \frac{b-a}{n}; x_0 = a; x_n = b$

Найти: Для каждых двух соседних точек $x_i,\ x_{i+1}, i = \overline{0,n-1}$ данного отрезка кубический полином, аппроксимирующий данную функцию в каждой точке интервала $(x_i,\ x_{i+1})$, значения которого совпадают со значениями функции на концах интервала.

Решение:

Введем общее обозначение для такого полинома на каждом таком интервале $(x_i,\ x_{i+1}),\ i = \overline{0,n-1}$ через $f(x)$. Его коэффициенты определяются из условия сопряжения в узлах:

$$f_i = y_i,$$
$$f'(x_i - 0) = f'(x_i + 0),$$
$$f''(x_i - 0) = f''(x_i + 0), i = \overline{1,n-1}$$

Кроме того, на границах при $x = x_0, x = x_n$, ставятся условия:

$$f''(x_0) = f''(x_n) = 0, (1)$$

Кубический полином ищется в виде:

$$f(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, x \in [x_i, x_{i+1}]. (2)$$

Из условия $f_i = y_i$ следует:

$$f(x_{i-1}) = a_i = y_{i-1},$$
$$f(x_i) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i\ (3), h_i = x_i - x_{i-1}, i = \overline{1,n-1}$$

Вычислим производные:

$$f'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$
$$f''(x) = 2c_i + 6d_i(x - x_{i-1}), \qquad x \in [x_i, x_{i+1}],$$

И потребуем их непрерывности при $x = x_i$:

$$\begin{cases} b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 \\ c_{i+1} = c_i + 3d_i h_i, i = \overline{1,n-1} \end{cases} (4)$$

Общее число неизвестных коэффициентов равно 4n, число уравнений (3) и (4) равно 4n-2. Недостающие два уравнения получаются из условий (1) при $x = x_0$ и $x = x_n$:

$$c_1 = 0, c_n + 3d_n h_n = 0.$$

Выражая из (4) $d_i = \frac{c_{i+1} - c_i}{3h_i}$, подставляя это выражение в (3) и исключая $a_i = y_{i-1}$, получим

$$b_i = \left[\frac{y_i - y_{i-1}}{h_i}\right] - \frac{1}{3}h_i(c_{i+1} + 2c_i), i = \overline{1,n-1},$$
$$b_n = \left[\frac{y_n - y_{n-1}}{h_n}\right] - \frac{2}{3}h_n c_n.$$

Подставив теперь выражения для b_i, b_{i+1} и d_i в первую формулу (4), после несложных преобразований получаем для c_i разностное уравнение второго порядка

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = 3\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right), \qquad i = \overline{1,n-1}, (5)$$

С краевыми условиями

$$c_1 = c_{n+1} = 0\ (6)$$

Условие $c_1 = c_{n+1} = 0$ эквивалентно условию $c_1 + 3d_n h_n = 0$ и уравнению $c_i + 3d_i h_i$. Разностное уравнение (9) с условиями (6) решается методом прогонки.

Табличное задание функции f(x)

Функция:

$$f(x) = e^{-\frac{(x-\frac{N}{N-5})^2}{2}}$$

x	0.21,	0.23,	0.25,	0.27,	0.29,	0.31,	0.33,	0.35
y(x)	0.06,	0.19,	0.26,	0.27,	0.26,	0.24,	0.20,	0.15

Код программы на ЭВМ

Результат работы программы по поиску коэффициентов

```
( 0,21, 0,06 )
( 0,212, 0,07924689999999997 )
( 0,214, 0,09811519999999993 )
( 0,216, 0,11622629999999988 )
( 0,218, 0,13320159999999986 )
( 0,22, 0,1486624999999998 )
( 0,222, 0,1622303999999998 )
( 0,224, 0,17352669999999978 )
( 0,226, 0,18217279999999975 )
( 0,228, 0,18779009999999977 )
( 0,23, 0,18999999999999978 )
( 0,23, 0,19 )
( 0,232, -0,5747517999999976 )
( 0,234, -1,0144063999999955 )
( 0,23600000000000002, -1,186562599999994 )
( 0,23800000000000002, -1,1488191999999928 )
( 0,24000000000000002, -0,9587749999999913 )
( 0,24200000000000002, -0,6740287999999923 )
( 0,24400000000000002, -0,3521793999999927 )
( 0,24600000000000002, -0,050825599999993365 )
( 0,24800000000000003, 0,17243380000000297 )
( 0,25, 0,2599999999999998 )
( 0,25, 0,26 )
( 0,252, -38,881398599999926 )
( 0,254, -61,59257279999988 )
( 0,256, -70,77299019999988 )
( 0,258, -69,32211839999988 )
( 0,26, -60,13942499999994 )
( 0,262, -46,12437759999999 )
( 0,264, -30,17644379999996 )
( 0,266, -15,195091199999979 )
( 0,268, -4,079787400000043 )
( 0,27, 0,26999999999992497 )
( 0,27, 0,27 )
( 0,272, -1956,8895867999906 )
( 0,274, -3092,526086399982 )
( 0,276, -3551,6141075999767 )
( 0,278, -3479,1282591999716 )
( 0,28, -3020,0431499999695 )
( 0,28200000000000003, -2319,3333887999706 )
( 0,28400000000000003, -1521,9735843999697 )
( 0,28600000000000003, -772,938345599976 )
( 0,28800000000000003, -217,202281199985 )
( 0,29, 0,26 )
( 0,292, -97857,53998319981 )
( 0,294, -154639,22767359964 )
( 0,296, -177593,52868239963 )
( 0,298, -173969,1686207996 )
( 0,3, -151014,87309999962 )
( 0,302, -115979,36773119975 )
( 0,304, -76111,37812559964 )
( 0,306, -38659,62989439978 )
( 0,308, -10872,8486487997 )
( 0,31, 0,24000000022351742 )
( 0,31, 0,24 )
( 0,312, -4892889,4482759815 )
( 0,314, -7731973,590847971 )
( 0,316, -8879688,460331967 )
( 0,318, -8698470,329343963 )
( 0,32, -7550755,470499963 )
```

Рис 1. Результаты работы программы по нахождению коэффициентов

Программа нахождения коэффициентов СЛАУ

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;

class Program
{
    static void Main(string[] args)
    {
        // Точки данных
        List<double> x = new List<double> { 0.21, 0.23, 0.25, 0.27, 0.29, 0.31, 0.33, 0.35,
0.37, 0.39 };
        List<double> y = new List<double> { 0.06, 0.19, 0.26, 0.27, 0.26, 0.24, 0.20, 0.15,
0.10, 0.06 };

        // Вычисление коэффициентов сплайна
        ComputeSplineCoefficients(x, y);
    }

    // Функция для печати вектора
    static void PrintVector(List<double> v)
    {
        foreach (var el in v)
        {
            Console.Write(el + " ");
        }
        Console.WriteLine();
    }

    // Функция Гаусса для решения системы уравнений
    static List<double> Gauss(List<List<double>> matrix)
    {
        int n = matrix.Count;
        List<double> x = new List<double>(new double[n]);

        for (int i = 0; i < n; ++i)
        {
            // Поиск максимального элемента в текущем столбце
            double maxEl = Math.Abs(matrix[i][i]);
            int maxRow = i;
            for (int k = i + 1; k < n; ++k)
            {
                if (Math.Abs(matrix[k][i]) > maxEl)
                {
                    maxEl = Math.Abs(matrix[k][i]);
                    maxRow = k;
                }
            }

            // Перестановка строк
            for (int k = i; k < n + 1; ++k)
            {
                var temp = matrix[maxRow][k];
                matrix[maxRow][k] = matrix[i][k];
                matrix[i][k] = temp;
            }

            // Обнуление элементов ниже диагонали
            for (int k = i + 1; k < n; ++k)
            {
```

```
                double c = -matrix[k][i] / matrix[i][i];
                for (int j = i; j < n + 1; ++j)
                {
                    if (i == j)
                    {
                        matrix[k][j] = 0;
                    }
                    else
                    {
                        matrix[k][j] += c * matrix[i][j];
                    }
                }
            }
        }

        // Обратный ход
        for (int i = n - 1; i >= 0; --i)
        {
            x[i] = matrix[i][n] / matrix[i][i];
            for (int k = i - 1; k >= 0; --k)
            {
                matrix[k][n] -= matrix[k][i] * x[i];
            }
        }

        return x;
    }

    // Функция для сохранения матрицы в CSV файл
    static void SaveMatrixToCSV(List<List<double>> matrix, List<string> headers, string file-
name)
    {
        using (StreamWriter file = new StreamWriter(filename))
        {
            // Записываем заголовки столбцов
            file.WriteLine(string.Join(";", headers));

            foreach (var row in matrix)
            {
                file.WriteLine(string.Join(";", row.Select(e => e.ToString(CultureInfo.Invari-
antCulture))));
            }
        }
    }

    static bool GetUserConsent(string message)
    {
        Console.Write(message + " (y/n): ");
        char response = Console.ReadKey().KeyChar;
        Console.WriteLine();
        return response == 'y' || response == 'Y';
    }

    static void ComputeSplineCoefficients(List<double> x, List<double> y)
    {
        int n = x.Count - 1;
        List<List<double>> matrix = new List<List<double>>();

        for (int i = 0; i < n - 1; i++)
        {
            List<double> line = new List<double>(new double[4 * n + 1]);
            line[i * 4] = 1;
            line[4 * n] = y[i];
            matrix.Add(line);

            if (i == 0)
            {
```

```
        line = new List<double>(new double[4 * n + 1]);
        line[i * 4 + 2] = 2;
        matrix.Add(line);
    }

    line = new List<double>(new double[4 * n + 1]);
    line[i * 4] = 1;
    line[i * 4 + 1] = x[i + 1] - x[i];
    line[i * 4 + 2] = Math.Pow(x[i + 1] - x[i], 2);
    line[i * 4 + 3] = Math.Pow(x[i + 1] - x[i], 3);
    line[4 * n] = y[i + 1];
    matrix.Add(line);

    line = new List<double>(new double[4 * n + 1]);
    line[i * 4 + 1] = 1;
    line[i * 4 + 2] = 2 * (x[i + 1] - x[i]);
    line[i * 4 + 3] = 3 * Math.Pow(x[i + 1] - x[i], 2);
    line[(i + 1) * 4] = -1;
    matrix.Add(line);

    line = new List<double>(new double[4 * n + 1]);
    line[i * 4 + 2] = 2;
    line[i * 4 + 3] = 6 * (x[i + 1] - x[i]);
    line[(i + 1) * 4 + 1] = -2;
    matrix.Add(line);
}

List<double> lastLine = new List<double>(new double[4 * n + 1]);
lastLine[(n - 1) * 4] = 1;
lastLine[4 * n] = y[y.Count - 2];
matrix.Add(lastLine);

lastLine = new List<double>(new double[4 * n + 1]);
lastLine[(n - 1) * 4] = 1;
lastLine[(n - 1) * 4 + 1] = x[x.Count - 1] - x[x.Count - 2];
lastLine[(n - 1) * 4 + 2] = Math.Pow(x[x.Count - 1] - x[x.Count - 2], 2);
lastLine[(n - 1) * 4 + 3] = Math.Pow(x[x.Count - 1] - x[x.Count - 2], 3);
lastLine[4 * n] = y[n];
matrix.Add(lastLine);

lastLine = new List<double>(new double[4 * n + 1]);
lastLine[(n - 1) * 4 + 2] = 2;
lastLine[(n - 1) * 4 + 3] = 6 * (x[x.Count - 1] - x[x.Count - 2]);
matrix.Add(lastLine);

List<string> headers = new List<string>();
for (int i = 0; i < n; i++)
{
    headers.Add($"a_{i}");
    headers.Add($"b_{i}");
    headers.Add($"c_{i}");
    headers.Add($"d_{i}");
}
headers.Add("=");

if (GetUserConsent("Сохранить систему неравенств в файл matrix.csv?"))
{
    SaveMatrixToCSV(matrix, headers, "matrix.csv");
}

List<double> solution = Gauss(matrix);
List<List<double>> sol = new List<List<double>> { solution };

if (GetUserConsent("Сохранить найденные коэффициенты в файл coeff.csv?"))
{
    SaveMatrixToCSV(sol, headers, "coeff.csv");
}
```

```
        List<List<double>> tests = new List<List<double>>();
        for (int i = 0; i < x.Count - 1; i++)
        {
            for (double j = x[i]; j <= x[i + 1]; j += (x[i + 1] - x[i]) / 10)
            {
                tests.Add(new List<double>
                {
                    j,
                    solution[i * 4] +
                    solution[i * 4 + 1] * (j - x[i]) +
                    solution[i * 4 + 2] * Math.Pow(j - x[i], 2) +
                    solution[i * 4 + 3] * Math.Pow(j - x[i], 3)
                });
            }
        }

        if (GetUserConsent("Сохранить найденные тестовые точки в файл testPoints.csv?"))
        {
            SaveMatrixToCSV(tests, new List<string> { "x", "y" }, "testPoints.csv");
        }

        if (GetUserConsent("Вывести найденные тестовые точки в терминал?"))
        {
            foreach (var point in tests)
            {
                Console.WriteLine($"( {point[0]}, {point[1]} )");
            }
        }
    }
}
```

Блок схема метода прогонки

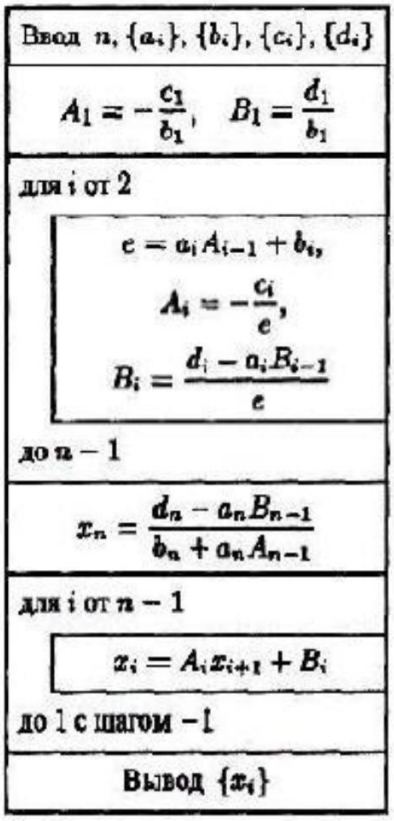

Рис 2. Блок -схема к методу прогонки

Графики исходной функциональной зависимости F(x) и матрица коэффициентов

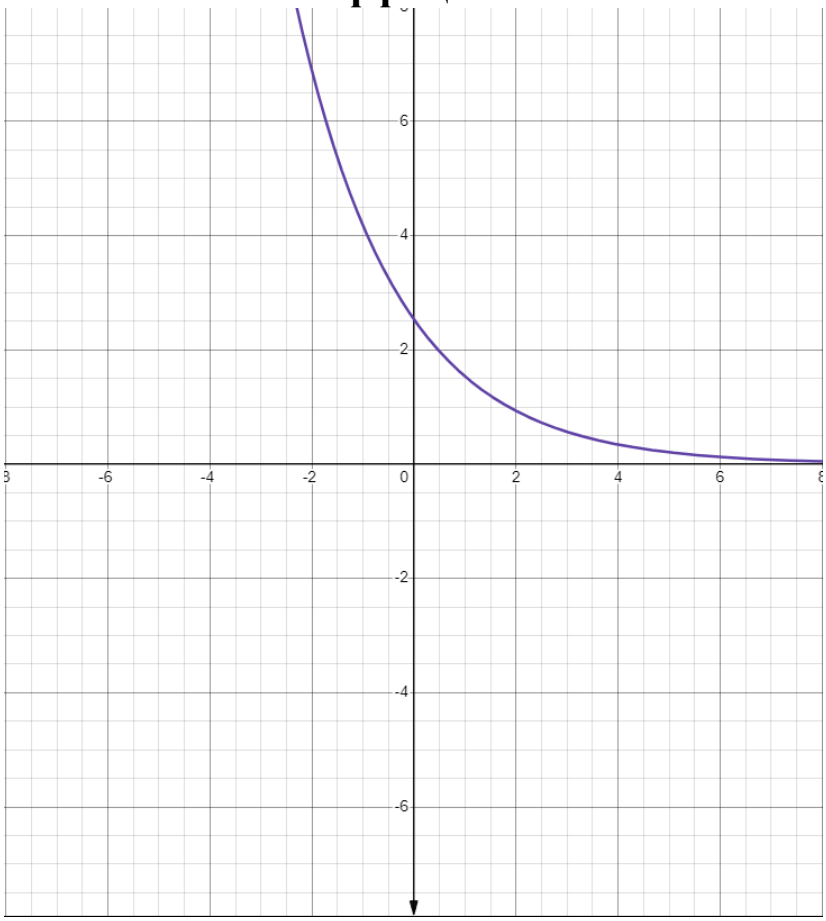

Рис 3. График функции и

a_0	b_0	c_0	d_0	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2	a_3	b_3	c_3	d_3	a_4	b_4	c_4	d_4	a_5	b_5	c_5	d_5	a_6	b_6	c_6	d_6	=
0.06	9.6549999		0	-7887.499	0.19	-473.2499	47836.999	-1199974.9	0.26	-24161.49	2416211.4	-60405574	0.27	-1208122.9	120812211	-30203043	0.26	-60406048	6040604	73 -15101511	0.24	-30203022	3020302	27 -75507556	0.2	-15101511	1132613	35 -1887688	91846874.8

a_0	b_0	c_0	d_0	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2	a_3	b_3	c_3	d_3	a_4	b_4	c_4	d_4	a_5	b_5	c_5	d_5	a_6	b_6	c_6	d_6	=
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.06
0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0.0200000	0.0004000	8.0000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.19
0	1	0.0400000	0.0012000	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0.1200000	0	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.19
0	0	0	0	1	0.0199999	0.0003999	7.9999999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.26
0	0	0	0	0	1	0.0399999	0.0011999	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	2	0.1199999	0	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.26
0	0	0	0	0	0	0	0	1	0.0200000	0.0004000	8.0000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.27
0	0	0	0	0	0	0	0	0	1	0.0400000	0.0012000	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0.1200000	0	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.27
0	0	0	0	0	0	0	0	0	0	0	0	1	0.0199999	0.0003999	7.9999999	0	0	0	0	0	0	0	0	0	0	0	0	0.26
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0399999	0.0011999	-1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.1199999	0	-2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.26
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0200000	0.0004000	8.0000000	0	0	0	0	0	0	0	0	0.24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0400000	0.0012000	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.1200000	0	-2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0.24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0200000	0.0004000	8.0000000	0	0	0	0.2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0400000	0.0012000	-1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.1200000	0	-2	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0.2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0199999	0.0003999	7.9999999	0.15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.1199999	0

Рис 4. Таблица кубического сплайна и погрешности

Вывод

Преимуществом сплайнов перед обычной интерполяцией является их сходимость и устойчивость процесса вычислений, обеспечивающие достаточную точность построения графиков.