



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технологический университет «СТАНКИН» (ФГБОУ ВО «МГТУ «СТАНКИН»)**

**Институт Кафедра прикладной**

**информационных систем математики**

**и технологий**

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](#_bookmark1) [кубическим](#_bookmark1) [сплайном](#_bookmark1) [дефекта](#_bookmark1) [1 3](#_bookmark1)

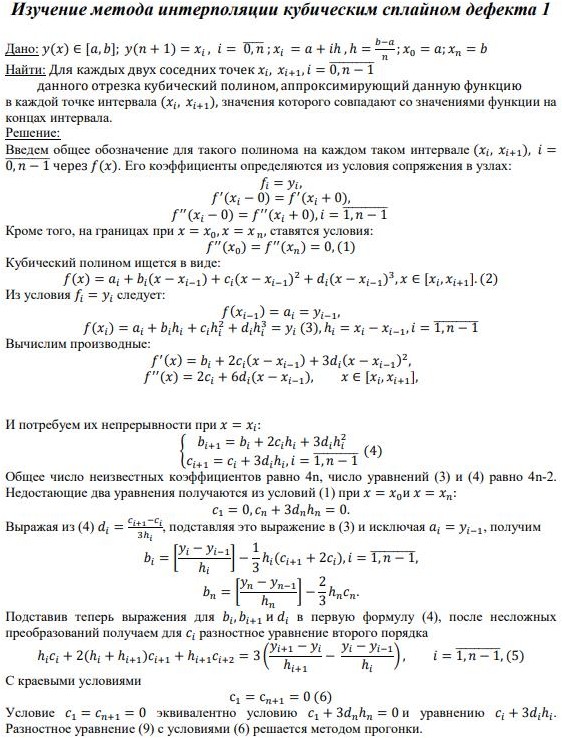
[Табличное задание](#_bookmark0) [функции](#_bookmark0) f(x) [4](#_bookmark0)

[Код программы](#_bookmark2) [на ЭВМ 5](#_bookmark2)

[Блок схема](#_bookmark3) [метода](#_bookmark3) [прогонки 9](#_bookmark3)

[Графики исходной функциональной](#_bookmark4) [зависимости](#_bookmark4) F(x) [и](#_bookmark4) матрица коэффициентов [10](#_bookmark4)

[Вывод 11](#_bookmark5)



## Табличное задание функции f(x)

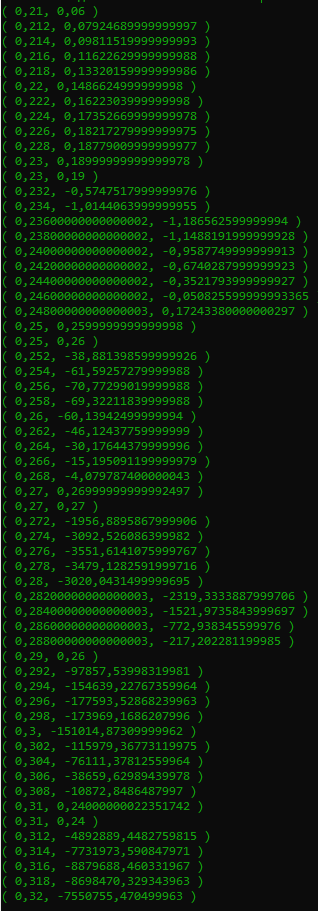
Функция:





# Код программы на ЭВМ

Результат работы программы по поиску коэффициентов

**

*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 filename)

{

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.InvariantCulture))));

}

}

}

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]} )");

}

}

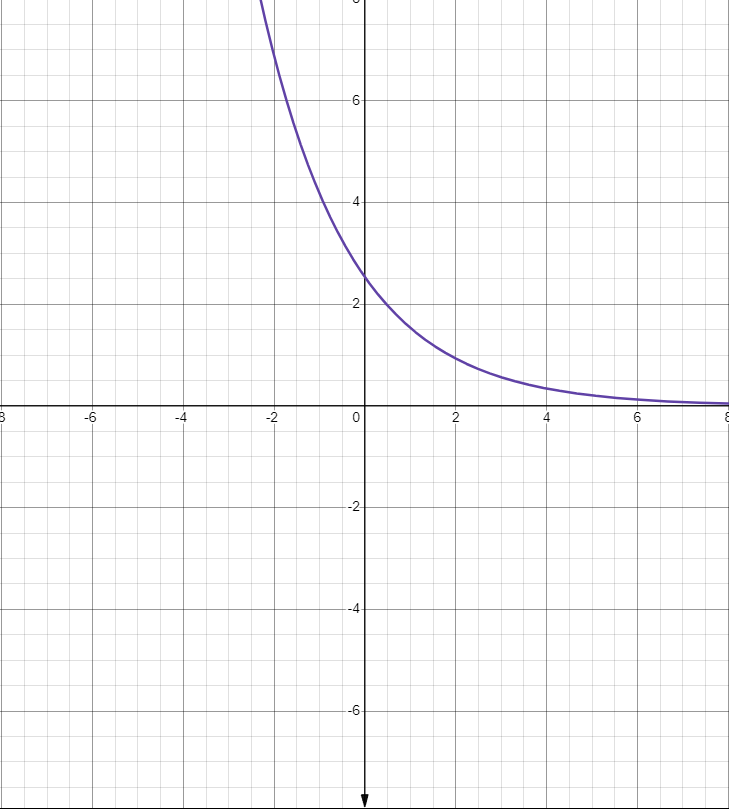
}

}

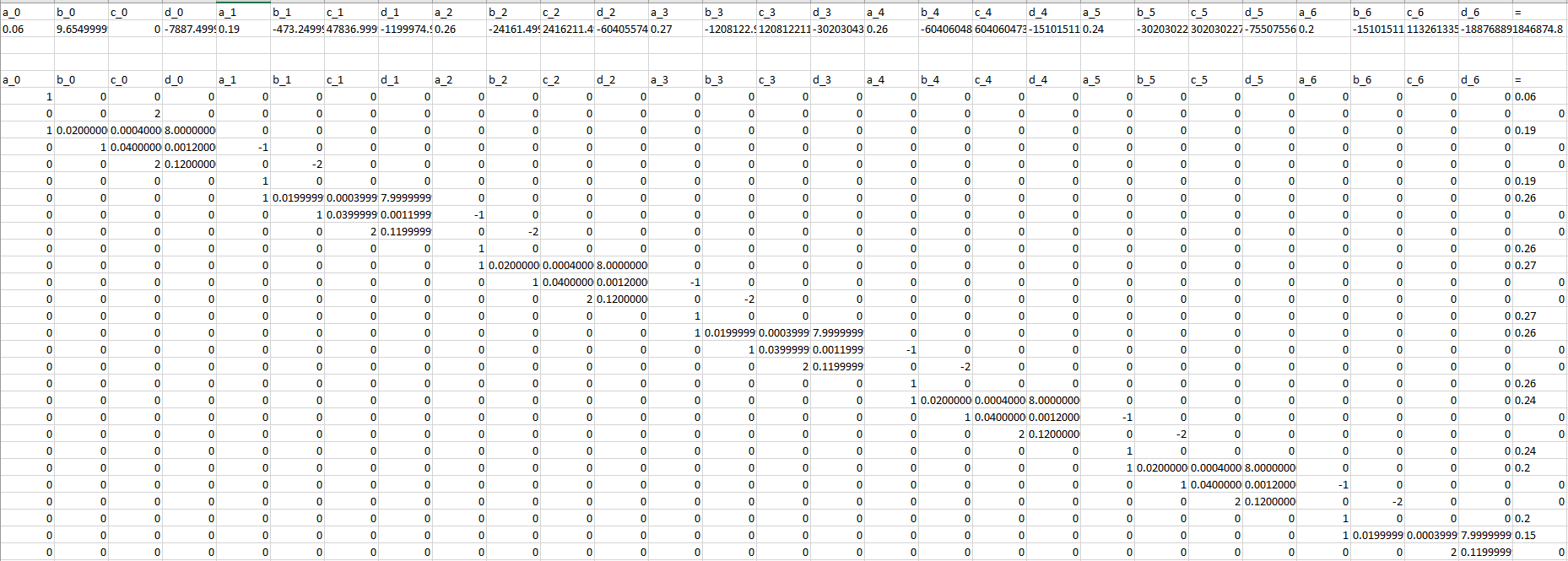
# Блок схема метода прогонки

*Рис 2. Блок -схема к методу прогонки*

# Графики исходной функциональной зависимости F(x) и матрица коэффициентов

****

*Рис 3. График функции и*

**

*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.