

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных технологий и вычислительных систем** |

ОТЧЕТ О ВЫПОЛНЕНИИ   
ЛАБОРАТОРНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ   
«*ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА*»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТА | *2* | КУРСА | | *бакалавриата* | ГРУППЫ | *ИДБ-21-01* |
|  | | | *(уровень профессионального образования)* | |  | |

|  |
| --- |
| **Лямцева Кирилла Игоревича** |
| *(ФИО)* |

НА ТЕМУ

|  |
| --- |
| Интерполирование кубическим сплайном дефекта 1 |

Вариант 12

|  |  |
| --- | --- |
| Направление: | 09.03.01 Информатика и вычислительная техника |
| Профиль подготовки: | «Программное обеспечение вычислительной техники и автоматизированных систем» |

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет сдан «25» мая 2023 г. | | | |
|  |  |  |  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
| Преподаватель | Ряшенцев К.В. |  |  |
|  | *(Ф.И.О., должность, степень, звание.)* |  | *(подпись)* |

МОСКВА 2023

**ОГЛАВЛЕНИЕ**

[ЗАДАНИЯ 3](#_Toc101125961)

[ХОД РАБОТЫ 4](#_Toc101125962)

[ЗАКЛЮЧЕНИЕ 10](#_Toc101125963)

**ЗАДАНИЯ**

1. Изучить метод интерполяции кубическим сплайном.

2. По номеру варианта № для функции f(x) выбрать отрезок [а;5] и разбить

его на 5 локальных отрезков. Задать функцию f(x) таблично. Составить свою

систему линейных алгебраических уравнений для нахождения коэффициентов

кубического полинома.

3. Написать программу на ЭВМ метода матричной прогонки.

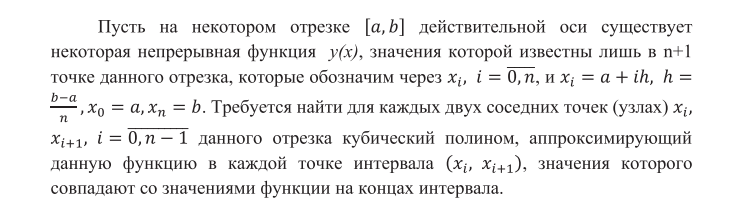
Построить на одной координатной плоскости функциональную

зависимость исходных данных f(x) и кубический сплайн S(x).

4. Вычислить погрешность сплайн - интерполяции. Провести анализ

полученных результатов и сделать выводы.

**ХОД РАБОТЫ**



Задана функция по номеру варианта (12):

==

Табличное задание функции f(x) = :



**Код программы на С++ для вычисления коэффициентов a, b, c, d:**

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <array>

#include <math.h>

class TridiagonalMatrix //Класс трехдиагональной матрицы для метода прогонки (для вычисления коэффициентов при квадратах)

{

private:

int n;

std::vector<double> A, B, C; //A - поддиагональные элементы, В - диагональные, С -наддиагональные, n - размер матрицы

public:

TridiagonalMatrix(const std::vector<double>&A1, const std::vector<double>&B1,

const std::vector<double>&C1) : A{ A1 }, B{ B1 }, C{ C1 } {

n = size(B1);

}

int GetN() { //размер матрицы

return n;

}

//геттеры для обращения к элементам матрицы

const double GetA(int i) {

return A[i];

}

const double GetB(int i) {

return B[i];

}

const double GetC(int i) {

return C[i];

}

};

//Прогонка, принимает на вход трехдиагональную матрицу и столбец свободных членов

//Возвращает столбец решения x

std::vector<double> Sweep(TridiagonalMatrix& Matrix, const std::vector<double>& D)

{

double denom; //переменная для хранения знаменателя

const int N = Matrix.GetN(); //размер входной матрицы

std::vector<double> x(N), p(N - 1), q(N - 1); // х - результат прогонки, p q -векторы для вычисления х

p[0] = -Matrix.GetC(0) / Matrix.GetB(0);

q[0] = D[0] / Matrix.GetB(0);

for (int i = 0; i < N - 2; ++i) {

denom = Matrix.GetA(i) \* p[i] + Matrix.GetB(i + 1);

p[i + 1] = -Matrix.GetC(i + 1) / denom;

q[i + 1] = (D[i + 1] - Matrix.GetA(i) \* q[i]) / denom;

}

x[N - 1] = (D[N - 1] - Matrix.GetA(N - 2) \* q[N - 2]) / (Matrix.GetA(N - 2) \*

p[N - 2] + Matrix.GetB(N - 1));

for (int i = 1; i < N; ++i) {

x[N - i - 1] = x[N - i] \* p[N - i - 1] + q[N - i - 1];

}

return x;

}

//интерполирующая функция, принимает на вход точки, значения функции в этих точках, значения вторых производных на концах отрезка(гран условия, по умолчанию == 0)

//возвращает вектор размера 4n - 4 - коэффициенты a, b, c, d, идущие подряд.

std::vector<double> Interpolation(std::vector<double>& x, std::vector<double>& y,

double sec\_der\_begin = 0, double sec\_der\_end = 0) {

//a, b, c, f - векторы для Трехдиагональной матрицы и прогонки (состоят из h\_i), A\_koef, B\_koef, C\_koef, D\_koef - векторы для хранения коэф.интерполяции

std::vector<double> a(size(x) - 1), b(size(x)), c(size(x) - 1), f(size(x)),

C\_koef(size(x)), B\_koef(size(x) - 1), A\_koef(size(x) - 1), D\_koef(size(x) - 1);

//вектор для записи результата

std::vector<double> res(4 \* size(x) - 4);

//заносим в векторы трехдиагональной матрицы краевые значения.

b[0] = 1;

b.back() = 1;

a.back() = 0;

c[0] = 0;

f[0] = sec\_der\_begin;

f.back() = sec\_der\_end;

//заносим в векторы трехдиагональной матрицы значения h\_i

for (int i = 0; i < size(c) - 1; ++i) {

b[i + 1] = 2 \* (x[i + 2] - x[i]);

c[i + 1] = x[i + 2] - x[i + 1];

a[i] = x[i + 1] - x[i];

f[i + 1] = 6 \* ((y[i + 2] - y[i + 1]) / (x[i + 2] - x[i + 1]) - (y[i + 1] -

y[i]) / (x[i + 1] - x[i]));

}

//инициализируем трехдиагональную матрицу

TridiagonalMatrix A(a, b, c);

//прогонка

C\_koef = Sweep(A, f);

//вычисляем коэффициенты интерполяции

for (int i = 0; i < size(B\_koef); ++i) {

A\_koef[i] = y[i + 1];

D\_koef[i] = (C\_koef[i + 1] - C\_koef[i]) / (x[i + 1] - x[i]);

B\_koef[i] = (y[i + 1] - y[i]) / (x[i + 1] - x[i]) + (x[i + 1] - x[i]) \*

C\_koef[i + 1] / 3 + (x[i + 1] - x[i]) \* C\_koef[i] / 6;

}

//заносим результат и возрващаем его

for (int i = 0; i < int(size(res) / 4); ++i) {

res[i \* 4] = A\_koef[i];

res[i \* 4 + 1] = B\_koef[i];

res[i \* 4 + 2] = C\_koef[i + 1];

res[i \* 4 + 3] = D\_koef[i];

}

return res;

}

//функция для записи результата работы интерполяции в файл.

void WritingToFile(std::vector<double> res, std::vector<double> x,

std::vector<double> y) {

std::ofstream fout;

fout.open("Interpolation.txt");

for (int i = 0; i < size(x); ++i) {

fout << x[i] << ";";

}

fout << std::endl;

for (int i = 0; i < size(x); ++i) {

fout << y[i] << ";";

}

fout << std::endl;

for (int i = 0; i < int(size(res) / 4); ++i) {

fout << res[i \* 4] << ";" << res[i \* 4 + 1] << ";" << res[i \* 4 + 2] / 2 <<

";" << res[i \* 4 + 3] / 6 << std::endl;

}

}

double Y(double x)

{

double y = sin(1.5 \* x) + cos (1.5 \* x);

return y;

}

int main()

{

std::vector<double> x, res;

double x0 = 0.0;

double xn = 3.0;

int n = 5;

double h = (xn - x0) / (n + 1);

for (int i = 0; i <= n + 1; i++)

{

x.push\_back(h \* i);

}

// std::vector<double> x = { 0, 0.3, 0.8, 1, 1.2 ,1.7}, res;

std::vector<double> y(size(x));

for (int i = 0; i < size(x); ++i) {

y[i] = Y(x[i]);

}

double x1 = 1, x2 = 0;

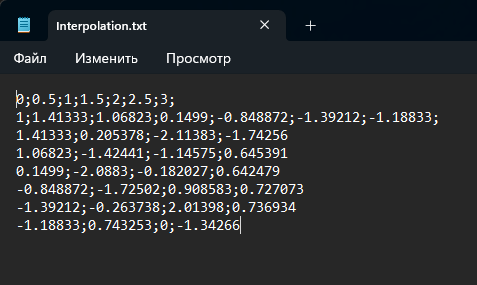
res = Interpolation(x, y, x1, x2);

WritingToFile(res, x, y);

return 0;

}

**Результат работы программы:**



**Код программы на Python для получения графика:**

import matplotlib.pyplot as plt

import numpy as np

#Тут путь к файлу, куда записаны результаты интерполяции. Введи свой.

path = r"C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.11\vc.txt"

a, b, c, d = [], [], [], []

with open(path) as f:

lines = f.readlines()

x = np.array(list(map(float, lines[0][:-2].split(';'))))

y = np.array(list(map(float, lines[1][:-2].split(';'))))

for i in range(len(lines) - 2):

koefs = list(map(float, lines[i + 2][:-2].split(';')))

a.append(koefs[0])

b.append(koefs[1])

c.append(koefs[2])

d.append(koefs[3])

X, Y, std, Sum\_std = np.array([]), np.array([]), 0, 0

x1 = np.linspace(0, 3.0, 250)

y1 = np.sin(1.5 \* x1) + np.cos(1.5 \* x1)

for i in range(len(a)):

X = np.append(X, np.linspace(x[i], x[i + 1], 50))

j = a[i] + b[i] \* (X[i \* 50:i \* 50 + 51] - x[i + 1]) + c[i] \* (X[i \* 50:i \* 50 + 51] - x[i + 1])\*\*2 + d[i] \* (X[i \* 50:i \* 50 + 51] - x[i + 1])\*\*3

Y = np.append(Y, j)

for i in range(25):

Sum\_std += np.abs(Y[i\*10] - y1[i\*10])

std = Sum\_std/25

plt.style.use('seaborn-v0\_8-pastel')

figure, ax = plt.subplots()

ax.scatter(x, y)

ax.plot(X, Y, color='red', label=f'Cubic Spline, Sum\_std = {round(Sum\_std, 2)}, std = {round(std, 2)}')

ax.plot(x1, y1, label='Function')

plt.xlabel('x')

plt.ylabel('y')

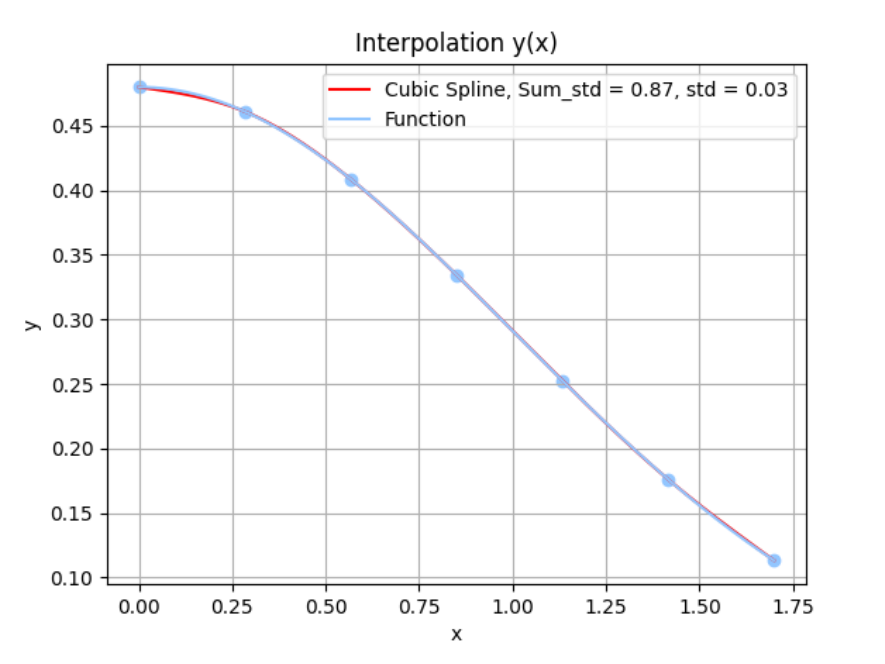
plt.title('Interpolation y(x)')

plt.legend()

plt.grid()

plt.show()

**Результат работы программы:**

****

**ЗАКЛЮЧЕНИЕ**

Метод интерполирования кубическим сплайном дефекта 1 - это численный метод для интерполяции функции на отрезке [a, b] с помощью кусочнокубической функции, которая проходит через заданные узлы интерполяции и имеет непрерывную первую производную на всем интервале [a, b]. Этот метод часто используется в математическом моделировании, приближении данных, а также в других областях, где требуется интерполяция функции. Таким образом, метод интерполирования кубическим сплайном дефекта 1 позволяет получить кусочно-кубическую функцию, которая аппроксимирует исходную функцию на отрезке [a, b] с высокой точностью и имеет непрерывную первую производную, что делает ее полезной во многих приложениях, где требуется интерполяция функции.