|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | «Теоретическая информатика и компьютерные технологии» |

**Отчет по лабораторной работе № 1**

**«Сплайн-интерполяция»**

***по курсу***

***«Численные методы»***

Выполнил:

Студент группы ИУ9-62Б

Караник А.А.

Преподаватель:

Домрачева А.Б.

1. *г.*

## Цель работы

Реализация программы для нахождения интерполяционной функции, проходящей через заданные точки.

## Постановка задачи

Протабулировать функцию на отрезке с шагом , где , , , .

Для полученных узлов построить кубический сплайн. Вычислить значения и сплайнов в точках . Вычислить значения и сплайнов в узлах интерполяции. Вычислить значение и сплайна в произвольной точке.

## Теоретические сведения

Для каждого отрезка разбиения отыскиваем кубический сплайн в виде

На частные многочлены накладываются условия:

(сплайн проходит через все узлы)

(непрерывность сплайна и его первых двух производных в промежуточных узлах)

(краевые условия)

Эти условия приводят к трехдиагональной СЛАУ относительно коэффициентов :

где – постоянный шаг интерполяции.

Для решения системы используется метод прогонки.

Остальные коэффициенты выражаются через по следующим формулам:

## Реализация

/\*\*

 \* Laboratory work: 1

 \* Discipline: Numerical methods

 \* Copyright (c) 2024, Andrey Karanik

\*/

#include <iostream>

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <iomanip>

float\* solve(float \*a, float \*b, float \*c, float \*d, int n) {

    float \*alpha = new float[n - 1];

    float \*beta = new float[n - 1];

    float \*x = new float[n];

    for (int i = 1; i < n - 1; i++) {

        if (std::fabs(b[i]) < std::fabs(a[i - 1]) + std::fabs(c[i])) {

            std::cout << "The condition is not satisfied." << std::endl;

            break;

        }

        if (std::fabs(b[i]) / (std::fabs(c[i])) < 1) {

            std::cout << "The condition is not satisfied." << std::endl;

            break;

        }

        if (std::fabs(b[i]) / (std::fabs(a[i - 1])) < 1) {

            std::cout << "The condition is not satisfied." << std::endl;

            break;

        }

    }

    if (b[0] != 0) {

        alpha[0] = -c[0] / b[0];

    } else {

        std::cout << "ERROR: b[0] = 0";

        return x;

    }

    beta[0] = d[0] / b[0];

    for (int i = 1; i < n - 1; i++) {

        alpha[i] = -(c[i]) / (alpha[i - 1] \* a[i - 1] + b[i]);

        beta[i] = (d[i] - a[i - 1] \* beta[i - 1]) / (alpha[i - 1] \* a[i - 1] + b[i]);

    }

    beta[n - 1] = (d[n - 1] - a[n - 2] \* beta[n - 2]) / (alpha[n - 2] \* a[n - 2] + b[n - 1]);

    x[n - 1] = beta[n - 1];

    std::cout << "";

    for (int i = n - 2; i >= 0; i--) {

        x[i] = alpha[i] \* x[i + 1] + beta[i];

    }

    delete[] alpha;

    delete[] beta;

    return x;

}

float calculateSplineValue(float x, float \*a, float \*b, float \*c, float \*d, float \*\_x, float left, float h) {

    int i = (x - left) / h;

    float difference = x - \_x[i];

    return a[i] + b[i] \* difference + c[i] \* difference \* difference + d[i] \* difference \* difference \* difference;

}

float f(float x) {

    return log(4\*x) / x;

}

int main(int argc, char \*argv[]) {

    int n = 10;

    float left = 0.25f;

    float right = 4.0f;

    float \*x = new float[n + 1];

    float \*y = new float[n + 1];

    float \*a = new float[n + 1];

    float \*b = new float[n + 1];

    float \*c = new float[n + 1];

    float \*d = new float[n + 1];

    float \*spline = new float[n + 1];

    float \*delta = new float[n + 1];

    float h = (right - left) / (float)n;

    x[0] = left;

    for (int i = 1; i <= n; i++) {

        x[i] = x[i - 1] + h;

    }

    for (int i = 0; i <= n; i++) {

        y[i] = f(x[i]);

    }

    float \*diag\_a = new float[n - 2];

    float \*diag\_b = new float[n - 1];

    float \*diag\_c = new float[n - 2];

    float \*free\_d = new float[n - 1];

    for (int i = 0; i < n - 2; i++) {

        diag\_a[i] = 1.0f;

        diag\_c[i] = 1.0f;

    }

    for (int i = 0; i < n - 1; i++) {

        diag\_b[i] = 4.0f;

    }

    for (int i = 1; i < n - 1; i++) {

        free\_d[i - 1] = (y[i + 1] - 2.0f \* y[i] + y[i - 1]) / (h \* h);

    }

    float\* solution = solve(diag\_a, diag\_b, diag\_c, free\_d, n - 1);

    for (int i = 1; i <= n - 1; i++) {

        c[i] = solution[i - 1];

    }

    c[n] = 0;

    c[0] = 0;

    for (int i = 0; i <= n; i++) {

        a[i] = y[i];

    }

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

        b[i] = ((y[i + 1] - y[i]) / h) - (h / 3.0f) \* (c[i + 1] + 2.0f \* c[i]);

        d[i] = (c[i + 1] - c[i]) / (3.0f \* h);

    }

    b[n - 1] = ((y[n] - y[n - 1]) / h) - (2.0f / 3.0f) \* h \* c[n - 1];

    d[n - 1] = -c[n] / (3.0f \* h);

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

        spline[i] = calculateSplineValue(x[i], a, b, c, d, x, left, h);

    }

    std::cout << "x: ";

    for (int i = 0; i <= n; i++) {

        std::cout << std::fixed << std::setprecision(3) << x[i] << " ";

    }

    std::cout << std::endl;

    std::cout << "S(x): ";

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

        std::cout << std::fixed << std::setprecision(3) << spline[i] << " ";

    }

    std::cout << std::endl;

    std::cout << "f(x): ";

    for (int i = 0; i <= n; i++) {

        std::cout << std::fixed << std::setprecision(3) << y[i] << " ";

    }

    std::cout << std::endl;

    std::cout << "|S(x)-f(x)|: ";

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

        std::cout << std::fixed << std::setprecision(3) << fabs(spline[i] - y[i]) << " ";

    }

    std::cout << std::endl;

    for (int i = 1; i <= n; i++) {

        x[i] = left + (i - 0.5) \* h;

    }

    for (int i = 1; i <= n; i++) {

        y[i] = f(x[i]);

    }

    for (int i = 1; i <= n; i++) {

        spline[i] = calculateSplineValue(x[i], a, b, c, d, x, left, h);

    }

    for (int i = 1; i <= n; i++) {

        delta[i] = fabs(y[i] - spline[i]);

    }

    std::cout << "x: ";

    for (int i = 1; i <= n; i++) {

        std::cout << std::fixed << std::setprecision(3) << x[i] << " ";

    }

    std::cout << std::endl;

    std::cout << "S(x): ";

    for (int i = 1; i <= n; i++) {

        std::cout << std::fixed << std::setprecision(3) << spline[i] << " ";

    }

    std::cout << std::endl;

    std::cout << "f(x): ";

    for (int i = 1; i <= n; i++) {

        std::cout << std::fixed << std::setprecision(3) << y[i] << " ";

    }

    std::cout << std::endl;

    std::cout << "|S(x)-f(x)|: ";

    for (int i = 1; i <= n; i++) {

        std::cout << std::fixed << std::setprecision(3) << fabs(spline[i] - y[i]) << " ";

    }

    std::cout << std::endl;

    delete[] x;

    delete[] y;

    delete[] a;

    delete[] b;

    delete[] c;

    delete[] d;

    delete[] solution;

    delete[] spline;

    delete[] delta;

    return 0;

}

## Тестирование

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 0 | 0.250 | 0.000 | 0.000 | 0.000 | - | - | - | - |
| 1 | 0.625 | 1.466 | 1.466 | 0.000 | 0.438 | 0.784 | 1.279 | 0.495 |
| 2 | 1.000 | 1.386 | 1.386 | 0.000 | 0.812 | 1.386 | 1.451 | 0.064 |
| 3 | 1.375 | 1.240 | 1.240 | 0.000 | 1.188 | 1.240 | 1.312 | 0.072 |
| 4 | 1.750 | 1.112 | 1.112 | 0.000 | 1.562 | 1.112 | 1.173 | 0.061 |
| 5 | 2.125 | 1.007 | 1.007 | 0.000 | 1.938 | 1.007 | 1.057 | 0.050 |
| 6 | 2.500 | 0.921 | 0.921 | 0.000 | 2.312 | 0.921 | 0.962 | 0.041 |
| 7 | 2.875 | 0.850 | 0.850 | 0.000 | 2.688 | 0.850 | 0.884 | 0.034 |
| 8 | 3.250 | 0.789 | 0.789 | 0.000 | 3.062 | 0.789 | 0.818 | 0.029 |
| 9 | 3.625 | 0.738 | 0.738 | 0.000 | 3.438 | 0.738 | 0.762 | 0.025 |
| 10 | 4.000 | - | 0.693 | - | 3.812 | 0.693 | 0.715 | 0.022 |

## Вывод

В процессе выполнения лабораторной работы была разработана программа на языке C++, предназначенная для создания интерполяционной функции с использованием кубических сплайнов. Результаты работы программы включают в себя значения функции и сплайна в точках разбиения и интерполяции. Анализ данных показывает, что погрешность приближения в точках интерполяции на краях функции минимальна, приближаясь к нулю.