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 **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

ОТЧЕТ

по лабораторной работе № 3

Название: Сплайн-интерполяции

Дисциплина: Вычислительные алгоритмы

Студент ИУ7И - 46Б _____ Андрич К.
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ В.М. Градов
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Цель работы

Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

Исходные данные

1. Таблица функции с количеством узлов N. Задать с помощью формулы 2 y x в диапазоне [0..10] с шагом 1.

2. Значение аргумента x в первом интервале, например, при x=0.5 и в середине таблицы, например, при x= 5.5. Сравнить с точным значением.

Описание алгоритма

Кубический сплайн — это кривая, состоящая из состыкованных полиномов третьей степени ($y^{(IV)}(x) = 0$). В точках стыковки значения и производные двух соседних полиномов равны.

Интерполяционный полином на участке между каждой пары соседних точек выглядит так:

$$\varphi(x) = ai + bi(x - xi{-}1) + ci(x - xi{-}1)^2 + di(x - xi{-}1)^3$$

Значение многочлена и интерполирующей функции совпадает в узлах:

$$f(xi{-}1) = yi{-}1$$

$$f(xi) = yi = ai + bi(x - xi{-}1) + ci(x - xi{-}1)^2 + di(x - xi{-}1)^3$$

Формула определения коэффициентов:

$$a_i = y_{i-1}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad h_i = x_i - x_{i-1}$$

$$b_i = \frac{(y_i - y_{i-1})}{h_i} - \frac{h_i(c_{i+1} + 2c_i)}{3}$$

Система уравнений с помощью которой определяется коэффициент c_i:

$$c_1 = 0$$

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

$$c_{N+1} = 0$$

Матрица будет трехдиагональная. Для решения такой системы пользуем метод прогонки.

Алгоритм метода прогонки:
Есть прямой ход и обратный ход.

В прямом ходе все прогоночные коэффициенты определяются при заданных начальных значениях прогоночных коэффициентов η_i и ξ_i

$$\eta_{i+1} = \frac{F_i + A_i \eta_i}{B_i - A_i \xi_i}$$
$$\xi_{i+1} = \frac{D_i}{B_i - A_i \xi_i}$$

В обратном ходе все c_i, $i = 1,N$ определяются при известном c_N

$$c_1 = 0$$
$$c_1 = \xi_2 c_2 + \eta_2$$
$$\xi_2 = 0$$
$$\eta_2 = 0$$

Можем найти начальные коэффициенты если у нас есть граничные условия. (прямый ход)
Можем найти c_i: (обратный ход)

$$c_i = \xi_{i+1} c_{i+1} + \eta_{i+1}$$
$$c_{N+1} = 0$$
$$c_N = \eta_{i+1}$$

Код программы

В программе есть 4 файлов: 2 заголовочных файла (functions.h и errors.h) и 2 файла кода в СИ (main.c, functions.c)

Файл: errors.h

```
#ifndef ERRORS_H
#define ERRORS_H

#define OK 0
#define ERR_IO 1
#define ERR_READ 2

#endif //FUNCTIONS_H
```

Файл: functions.h

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
#define N 20
#define M 2
#define EPS 1e-6

typedef struct
{
      float A[N];
      float B[N];
      float C[N];
      float D[N];
} struct_t;

int read_line(FILE *f, float *buf);
int read_file(FILE *f, float mtx[N][M], int *n);
int get_x(float *x);
void print_table(float mtx[N][M], int n);
void find_A(float *arr, int n, float *ys);
void find_C(float *arr, int n, float *xs, float *ys);
void find_BD(float *arr1, float *arr2, int n, float *xs, float *ys, float
*array);
void get_coef(float *xs, float *ys, struct_t *coef, int n);
int index(float *xs, float x, int n);
float get_res_sp(float *arr, float x, struct_t coef, int indx);
float spline(float mtx[N][M], float x, int n);
int find_index(float mtx[N][M], float x, float n, float t);
float newton(float mtx[N][M], float x, float n, float t);

#endif //FUNCTIONS_H
```

Файл: functions.c

```
#include "functions.h"
#include "errors.h"

int read_line(FILE *f, float *buf)
{
    if (fscanf(f, "%f", &buf[0]) != 1)
        return ERR_READ;
    if (fscanf(f, "%f", &buf[1]) != 1)
        return ERR_READ;
    return OK;
}

int read_file(FILE *f, float mtx[N][M], int *n)
{
    int i = 0;
    float buf[2];
```

```
    int rc = OK;
    while(rc == OK)
    {
        rc = read_line(f, buf);
        if (rc != ERR_READ)
        {
            mtx[i][0] = buf[0];
            mtx[i][1] = buf[1];
            i += 1;
        }

    }
    *n = i;
    if (i != 0)
        rc = OK;
    return rc;
}

int get_x(float *x)
{
    float n;
    printf("Input value of x: ");
    if (scanf("%f", &n) != 1)
        return ERR_READ;
    *x = n;
    return OK;
}

void print_table(float mtx[N][M], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 2; j++)
            printf("%.2f ", mtx[i][j]);
        printf("\n");
    }
    printf("\n");
}

void find_A(float *arr, int n, float *ys)
{
    for (int i = 0; i < n - 1; i++)
        arr[i] = ys[i];
}

void find_C(float *arr, int n, float *xs, float *ys)
{
    float xi[N], theta[N];
    float h1, h2, xi_n, theta_n, phi;
    for (int i = 0; i < 2; i++)
    {
```

```
        xi[i] = 0;
        theta[i] = 0;
    }
    for (int i = 2; i < n; i++)
    {
        h1 = xs[i] - xs[i - 1];
        h2 = xs[i - 1] - xs[i - 2];
        phi = 3 * ((ys[i] - ys[i - 1]) / h1 - (ys[i - 1] - ys[i - 2]) / h2);
        xi_n = 0 - (h1 / (h2 * xi[i - 1] + 2 * (h2 + h1)));
        theta_n = (phi - h1 * theta[i - 1]) / (h1 * xi[i - 1] + 2 * (h2 + h1));
        xi[i] = xi_n;
        theta[i] = theta_n;
    }
    arr[n - 2] = theta[n - 2];
    for (int i = n - 2; i > 0; i--)
        arr[i - 1] = xi[i] * arr[i] + theta[i];
}

void find_BD(float *arr1, float *arr2, int n, float *xs, float *ys, float *array)
{
    float h;
    for (int i = 1; i < n - 1; i++)
    {
        h = xs[i] - xs[i - 1];
        arr1[i - 1] = (ys[i] - ys[i - 1]) / h - (h * (array[i] + 2 * array[i - 1]))
/ 3;
        arr2[i - 1] = (array[i] - array[i - 1]) / (3 * h);
    }
    h = xs[n - 1] - xs[n - 2];
    arr1[n - 2] = (ys[n - 1] - ys[n - 2]) / h - (h * 2 * array[n - 2]) / 3;
    arr2[n - 2] = 0 - array[n - 2] / (3 * h);
}

void get_coef(float *xs, float *ys, struct_t *coef, int n)
{
    find_A(coef->A, n, ys);
    find_C(coef->C, n, xs, ys);
    find_BD(coef->B, coef->D, n, xs, ys, coef->C);
}

int index(float *xs, float x, int n)
{
    int index = 1;
    while (index < n && xs[index] < x)
        index += 1;
    return index -= 1;
}

float get_res_sp(float *arr, float x, struct_t coef, int indx)
{
    float h = x - arr[indx];
```

```
    float y = coef.A[indx] + coef.B[indx] * h + coef.C[indx] * h * h + coef.D[indx]
* h * h * h;
    return y;
}

float spline(float mtx[N][M], float x, int n)
{
    float xs[n], ys[n], res;
    int indx;
    struct_t coef;
    for (int i = 0; i < n; i++)
    {
        xs[i] = mtx[i][0];
        ys[i] = mtx[i][1];
    };
    get_coef(xs, ys, &coef, n);
    indx = index(xs, x, n);
    res = get_res_sp(xs, x, coef, indx);
    return res;
}

int find_index(float mtx[N][M], float x, float n, float t)
{
    int inx = 0;
    int lborder, rborder;
    for (int i = 0; i < n; i++)
    {
        if (mtx[i][0] > x)
            break;
        inx++;
    }
    if (inx >= (n - t))
        return (n - t - 1);
    lborder = inx;
    rborder = inx;
    while (t > 0)
    {
        if ((rborder - inx) == (inx - lborder))
        {
            if (lborder > 0)
                lborder -= 1;
            else
                rborder += 1;
        }
        else
        {
            if (rborder < (n - 1))
                rborder += 1;
            else
                lborder -= 1;
        }
```

```
        t--;
    }
    return lborder;
}

float newton(float mtx[N][M], float x, float n, float t)
{
    int indx = find_index(mtx, x, n, t);
    float newt = mtx[indx][1];
    int mult;
    for (int i = 0; i < t; i++)
    {
        for (int j = 0; j < (t - i); j++)
        {
            if (fabs(mtx[indx + j][0] - mtx[indx + j + i + 1][0]) > EPS)
                mtx[indx + j][1] = (mtx[indx + j][1] - mtx[indx + j + 1][1]) /
(mtx[indx + j][0] - mtx[indx + j + i + 1][0]);
        }
        mult = 1;
        for (int k = 0; k < (i + 1); k++)
            mult *= (x - mtx[indx + k][0]);
        mult *= mtx[indx][1];
        newt += mult;
    }
    return newt;
}
```

Файл: errors.h

```
#include "functions.h"
#include "errors.h"

int main(void)
{
    FILE *f = fopen("data.txt", "r");
    int n, rc = OK;
    float x;
    float res1, res2;
    float mtx[N][M];
    if (f == NULL)
    {
        printf("Error openning file \n");
        rc = ERR_READ;
    }
    else
    {
        rc = read_file(f, mtx, &n);
        if (rc == OK)
        {
            print_table(mtx, n);
            rc = get_x(&x);
```

```
            if (rc == OK)
            {
                res1 = spline(mtx, x, n);
                printf("Spline: %f\n", res1);
                res2 = newton(mtx, x, n, 3);
                printf("Newton: %f\n", res2);
            }
            else
                printf("Error input \n");
        }
        else
            printf("Error reading file \n");
        fclose(f);
    }
    return rc;
}
```

Результаты работы

x	y	Сплайн	Ньютон
0.5	0.25	0.341506	0.250000
5.5	30.25	30.250345	30.25000

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/main/Algorithm/LR3$ ./app.exe
0.00 0.00
1.00 1.00
2.00 4.00
3.00 9.00
4.00 16.00
5.00 25.00
6.00 36.00
7.00 49.00
8.00 64.00
9.00 81.00
10.00 100.00

Input value of x: 0.5
Spline: 0.341506
Newton: 0.250000
```

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/main/Algorithm/LR3$ ./app.exe
0.00 0.00
1.00 1.00
2.00 4.00
3.00 9.00
4.00 16.00
5.00 25.00
6.00 36.00
7.00 49.00
8.00 64.00
9.00 81.00
10.00 100.00

Input value of x: 5.5
Spline: 30.250345
Newton: 30.250000
```

Вопросы при защите лабораторной работы

1. *Получить выражения для коэффициентов кубического сплайна, построенного на двух точках.*
   Пусть у нас (x0, x1) и (y0, y1)
   Тогда Кубический сплайн имеет вид:
   $\psi(x) = a_i + bi(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3,\ x_{i-1} \le x \le x_i\ ,\ 1 \le i \le N$

   Так как у нас только две точки, N = 1

   $\psi(x_0) = y_0 = a$
   $\psi(x_1) = y_1 = a + b(x_1 - x_0) + c(x_1 - x_0)^2 + d(x_1 - x_0)^3$ (*)
   $\psi'(x) = b + 2c(x - x_0) + 3d(x - x_0)^2$
   $\psi''(x) = 2c + 6d(x - x_0)$

   $\psi''(x_0) = 0 = 2c \Rightarrow c = \mathbf{0}$
   $\psi''(x_1) = 0 = 2c + 6d(x - x_0) \Rightarrow d = \mathbf{0}$

   Если подставим получение значения в *:
   $y_1 = y_0 + b(x_1 - x_0) + 0 \cdot (x_1 - x_0)^2 + 0 \cdot (x_1 - x_0)^3 \Rightarrow b = \frac{y1 - y0}{x1 - x0}$

2. *Выписать все условия для определения коэффициентов сплайна, построенного на 3-х точках*
   - Значение Сплайна и интерполируемой функции совпадают во всех точках
   - В внутренем узле, между левой и правой части, совпадают значения 1. и 2. производных
   - Если у нас нехватки условия, можем положить величину второй производной равной 0 на краях участка интерполирования

   $\psi(x_0) = y_0 = a_1$
   $\psi(x_1) = y_1 = a_1 + b_1(x_1 - x_0) + c_1(x_1 - x_0)^2 + d_1(x_1 - x_0)^3 = a_2$
   $\psi(x_2) = y_2 = a_2 + b_2(x_1 - x_0) + c_2(x_1 - x_0)^2 + d_2(x_1 - x_0)^3$
   $\psi'(x_1) = b_1 + 2c_1(x_1 - x_0) + 3d_1(x_1 - x_0)^2 = b_2$
   $\psi''(x_1) = 2c_1 + 6d_1(x_1 - x_0) = c_2$
   $\psi''(x_0) = 0$
   $\psi''(x_2) = 0$

3. *Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо $C_1=C_2$.*
   $c_{i-1} = \xi_i c_i + \eta_i$
   Если $C_1=C_2$, это значит что $\xi_2 = 1$, $\eta_2 = 0$

4. *Написать формулу для определения последнего коэффициента сплайна CN , чтобы можно было выполнить обратный ход метода прогонки, если в качестве граничного условия задано kCN-1+mCN=p, где k,m и p - заданные числа.*
   $c_{N-1} = \xi_N c_N + \eta_N$

   $kc_{N-1} + mc_N = p$

   $$c_{N-1} = \frac{p - k\eta_N}{k\xi_N + m}$$