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

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**ОТЧЕТ**

| **по лабораторной работе №** | 3 |
| --- | --- |



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.

1. Значение аргумента x в первом интервале, например, при х=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:

𝜑(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:

𝑎𝑖 = 𝑦𝑖−1

𝑑𝑖 = , ℎ𝑖 = 𝑥𝑖 − 𝑥𝑖−1

𝑏𝑖 =

Система уравнений с помощью которой определяется коэффициент 𝑐𝑖:

𝑐1 = 0

ℎ𝑖−1𝑐𝑖−1 + 2(ℎ𝑖−1 + ℎ𝑖)𝑐𝑖 + ℎ𝑖𝑐𝑖+1 = 3

𝑐𝑁+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 𝜂𝑖 и 𝜉𝑖

𝜂𝑖+1 =

𝜉𝑖+1 =

В обратном ходе все 𝑐𝑖, 𝑖 = 1,𝑁 определяются при известном 𝑐𝑁

𝑐1 = 0

𝑐1 = 𝜉2𝑐2 + 𝜂2

𝜉2 = 0

𝜂2 = 0

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

Можем найти 𝑐𝑖 : (обратный ход)

𝑐𝑖 = 𝜉𝑖+1𝑐𝑖+1 + 𝜂𝑖+1

𝑐𝑁+1 = 0

𝑐𝑁 = 𝜂𝑖+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

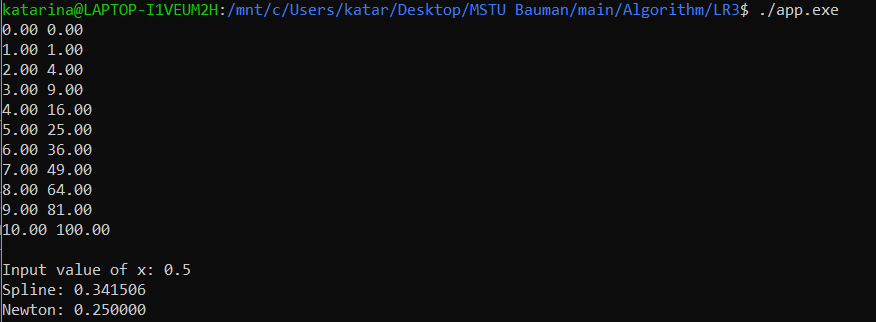
| #include "functions.h" #include "errors.h"  int read\_line(FILE \*f, float \*buf) {  if (fscanf(f, "%f", &buf[0]) != 1)  return ERR\_READ;  if (fscanf(f, "%f", &buf[1]) != 1)  return ERR\_READ;  return OK; }  int read\_file(FILE \*f, float mtx[N][M], int \*n) {  int i = 0;  float buf[2];  int rc = 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; } |
| --- |

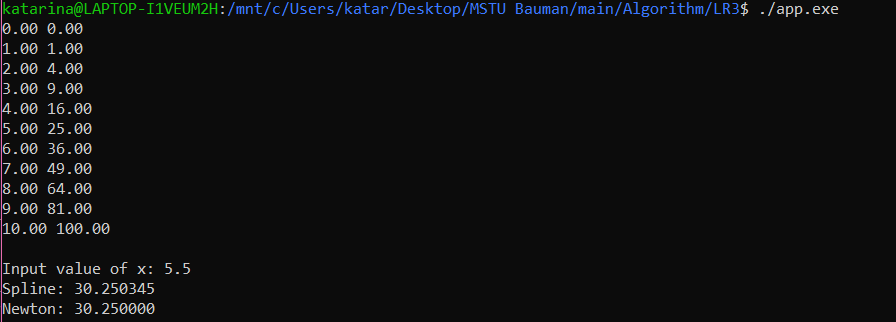
Файл: errors.h

| #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 |





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:

ψ(𝑥) = 𝑎𝑖 + 𝑏𝑖(𝑥 − 𝑥𝑖−1) + 𝑐𝑖(𝑥 − 𝑥𝑖−1)2 + 𝑑𝑖(𝑥 − 𝑥𝑖−1)3, 𝑥𝑖−1 ≤ 𝑥 ≤ 𝑥𝑖 , 1 ≤ 𝑖 ≤ N

Так как у нас только две точки, N = 1

ψ(𝑥0) = **𝑦0 = 𝑎**

ψ(𝑥1) = 𝑦1 = 𝑎 + 𝑏(𝑥1 − 𝑥0) + 𝑐(𝑥1 − 𝑥0)2 + 𝑑(𝑥1 − 𝑥0)3 (\*)

ψ'(𝑥) = 𝑏 + 2𝑐(𝑥 − 𝑥0) + 3𝑑(𝑥 − 𝑥0)2

ψ''(𝑥) = 2𝑐 + 6𝑑(𝑥 − 𝑥0)

ψ''(𝑥0) = 0 = 2𝑐 ⇒ **𝑐 = 0**

ψ''(𝑥1) = 0 = 2𝑐 + 6𝑑(𝑥 − 𝑥0) ⇒ **𝑑 = 0**

Если подставим получение значения в \*:

𝑦1 = 𝑦0 + 𝑏(𝑥1 − 𝑥0) + 0 · (𝑥1 − 𝑥0)2 + 0 · (𝑥1 − 𝑥0)3 ⇒ **𝑏 =**

1. *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

ψ(𝑥0) = 𝑦0 = 𝑎1

ψ(𝑥1) = 𝑦1 = 𝑎1 + 𝑏1(𝑥1 − 𝑥0) + 𝑐1(𝑥1 − 𝑥0)2 + 𝑑1(𝑥1 − 𝑥0)3 = 𝑎2

ψ(𝑥2) = 𝑦2 = 𝑎2 + 𝑏2(𝑥1 − 𝑥0) + 𝑐2(𝑥1 − 𝑥0)2 + 𝑑2(𝑥1 − 𝑥0)3

ψ'(𝑥1) = 𝑏1 + 2𝑐1(𝑥1 − 𝑥0) + 3𝑑1(𝑥1 − 𝑥0)2 = 𝑏2

ψ''(𝑥1) = 2𝑐1 + 6𝑑1(𝑥1 − 𝑥0) = 𝑐2

ψ''(𝑥0) = 0

ψ''(𝑥2) = 0

1. *Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо C1=C2.*

𝑐𝑖−1 = ξ𝑖𝑐𝑖 + η𝑖

Если *C1=C2,* это значит что ξ2 = 1, η2  = 0

1. *Написать формулу для определения последнего коэффициента сплайна СN , 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.*

𝑐N−1 = ξN𝑐N + ηN

𝑘𝑐N−1 + 𝑚𝑐N = *p*