§35. Лабораторная работа № 7.2
Интерполяция функций сплайном третьего порядка.

Ц е л ь р а б о т ы: изучить сплайн-интерполяцию различных порядков.

С о д е р ж а н и е р а б о т ы: на основании экспериментальных данных построить сплайн функцию, и найти значение функции в указанной точке,

Теоретический материал

Опишем сначала, что такое интерполяция и как эта задача решена в данном случае.

Пусть на некотором отрезке в точках x_0, x_1, x_2, ... x_N нам известны значения некоторой функции $f(x)$, а именно y_0, y_1, y_2, ... y_N.

Требуется построить интерполирующую функцию F(x), такую, что она принимает в указанных точках те же значения, т.е. $F(x_0) = y_0$, $F(x_1) = y_1$, ... $F(x_N) = y_N$.

Геометрически это значит, что нужно найти кривую $y = F(x)$ определенного типа, проходящую через систему заданных точек. В такой общей постановке задача может иметь бесчисленное множество решений или совсем не иметь решений. В случае интерполяции сплайном кривая $F(x)$ состоит из кусочков, а именно, на каждом из отрезков $[x_{k-1}; x_k]$ функция $F(x)$ является кубическим полиномом

$F_k(x) = a_k + b_k(x-x_k) + c_k(x-x_k)^2 + d_k(x-x_k)^3$

$F = F_1$ на интервале $[x_0, x_1]$

$F = F_2$ на интервале $[x_1, x_2]$

...

$F = F_N$ на интервале $[x_{N-1}, x_N]$

При этом, на каждом из отрезков $[x_{k-1}; x_k]$ коэффициенты полинома a_k, b_k, c_k, d_k разные. Чтобы узнать эти коэффициенты, кроме условия непрерывности функции на полиномы налагают дополнительные условия, а именно непрерывности первой и второй производной функции $F(x)$, а также равенства вторых производных функции на концах отрезка $[x_0; x_N]$, т.е.

$F_{k-1}(x_{k-1}) = F_k(x_{k-1})$,

$F'_{k-1}(x_{k-1}) = F'_k(x_{k-1})$,

$F''_{k-1}(x_{k-1}) = F''_k(x_{k-1})$,

при $k=2,3,..N$

$F''(x_0)=0$, $F''(x_N)=0$.

Найдем выражения для производных функций F_k

$F'_k(x) = b_k + 2c_k(x - x_k) + 3d_k(x-x_k)^2$

$F''_k(x) = 2c_k + 6\,d_k(x-x_k)$

Подставив их в условия непрерывности получим систему:

$a_1 - b_1h_1 + c_1h_1^2 - d_1h_1^3 = y_0$

$a_k = y_k,\ k=1,2,..N$

$a_{k-1} = a_k - b_kh_k + c_kh_k^2 - d_kh_k^3,\ k=2,3...N$

$b_{k-1} = b_k - 2c_kh_k + 3d_kh_k^2,\ k=2,3...N$

$c_{k-1} = c_k - 3d_kh_k,\ k=2,3...N$

$c_1 - 3d_1h_1 = 0$

$c_N = 0$

Здесь введено обозначение $h_k = x_k - x_{k-1},\ k=1,2,...N$

Введем еще $l_k = (y_k - y_{k-1})/h_k,\ k=1,2,...N$, а также $c_0 = 0$.

Упомянутую выше систему уравнений можно решить с помощью некоторых преобразований. При этом используется так называемый метод прогонки.

Вводятся прогоночные коэффициенты

$\delta_1 = - h_2/(2(h_1+h_2))$

$\lambda_1 = 3(l_2 - l_1)/(2(h_1+h_2))$

$\delta_{k-1} = - h_k/(2h_{k-1}+2h_k+h_{k-1}\delta_{k-2}),\ k=3,4,\ ...\ N$

$\lambda_{k-1} = (3l_k - 3l_{k-1} - h_{k-1}\lambda_{k-2})/(2h_{k-1}+2h_k+h_{k-1}\delta_{k-2})$

Далее следует найти коэффициенты c_k по формулам обратной прогонки

$c_{k-1} = \delta_{k-1}c_k + \lambda_{k-1},\ k = N, N-1, N-2,\ ...\ 2$

После нахождения c_k нужно найти b_k и d_k по формулам

$b_k = l_k + (2c_kh_k + h_kc_{k-1})/3,\ k=1,2,...N$

$d_k = (c_k - c_{k-1})/(3h_k),\ k=1,2,...N$

Алгоритм реализован в виде программы на языке C.

Приведем программу, где реализован этот алгоритм:

```
//Cubic spline interpolation program
//when we have two columns of data x and y in input file:
//
//x0 y0
//x1 y1
```

```
//...
//xn yn
//
//and we want to find such function f(x)
//where f(xi) = yi
//and f(x) is cubic function on every [x_k-1, x_k] segment
//and f(x), f'(x), f''(x) are continual
//the result is four columns of cubic polinom coefficients

#include <math.h>
#include <stdio.h>
#include <process.h>
float *x, *y, *h, *l, *delta, *lambda, *c, *d, *b;
int N;
char filename[256];
FILE* InFile=NULL;
void count_num_lines(){
  //count number of lines in input file - number of equations
  int nelf=0;       //non empty line flag
  do{
     nelf = 0;
     while(fgetc(InFile)!='\n' && !feof(InFile)) nelf=1;
     if(nelf) N++;
  }while(!feof(InFile));
  N--;
}
void readmatrix(){
  int i=0;
  //read matrixes a and b from input file
  for(i=0; i<N+1; i++){
     fscanf(InFile, "%f", &x[i]);
     fscanf(InFile, "%f", &y[i]);
```

```
    }
}

void allocmatrix(){
    //allocate memory for matrixes
    x = new float[N+1];
    y = new float[N+1];
    h = new float[N+1];
    l = new float[N+1];
    delta = new float[N+1];
    lambda = new float[N+1];
    c = new float[N+1];
    d = new float[N+1];
    b = new float[N+1];
}
void freematrix(){
    delete [] x;
    delete [] y;
    delete [] h;
    delete [] l;
    delete [] delta;
    delete [] lambda;
    delete [] c;
    delete [] d;
    delete [] b;
}

void printresult(){
    int k=0;
    printf("\nA[k]\tB[k]\tC[k]\tD[k]\n");
    for(k=1; k<=N; k++){
        printf("%f\t%f\t%f\t%f\n", y[k], b[k], c[k], d[k]);
```

```
  }
}
void testresult(){
  float start = x[0];
  float end = x[N];
  float step = (end - start)/20;
  FILE* OutFile = fopen("test.txt", "wt");
  for(float s = start; s<=end; s+= step){
    //find k, where s in [x_k-1; x_k]
    for(int k=1; k<=N; k++){
        if(s>=x[k-1] && s<=x[k]){
          break;
        }
    }
    float F = y[k] + b[k]*(s-x[k]) + c[k]*pow(s-x[k], 2) + d[k]*pow(s-x[k], 3);
    fprintf(OutFile, "%f\t%f\n", s,  F);
  }
  fclose(OutFile);
}
void cls(){
  for(int i=0; i<25; i++) printf("\n");
}
void main(){
  int k=0;
  cls();
  do{
    printf("\nInput filename: ");
    scanf("%s", filename);
    InFile = fopen(filename, "rt");
  }while(InFile==NULL);
  count_num_lines();
  rewind(InFile);
```

```
  allocmatrix();
  readmatrix();
  for(k=1; k<=N; k++){
     h[k] = x[k] - x[k-1];
     if(h[k]==0){
          printf("\nError, x[%d]=x[%d]\n", k, k-1);
          return;
     }
     l[k] = (y[k] - y[k-1])/h[k];
  }
  delta[1] = - h[2]/(2*(h[1]+h[2]));
  lambda[1] = 1.5*(l[2] - l[1])/(h[1]+h[2]);
  for(k=3; k<=N; k++){
    delta[k-1] = - h[k]/(2*h[k-1] + 2*h[k] + h[k-1]*delta[k-2]);
    lambda[k-1] = (3*l[k] - 3*l[k-1] - h[k-1]*lambda[k-2]) /
                  (2*h[k-1] + 2*h[k] + h[k-1]*delta[k-2]);
  }
  c[0] = 0;
  c[N] = 0;
  for(k=N; k>=2; k--){
    c[k-1] = delta[k-1]*c[k] + lambda[k-1];
  }
  for(k=1; k<=N; k++){
    d[k] = (c[k] - c[k-1])/(3*h[k]);
    b[k] = l[k] + (2*c[k]*h[k] + h[k]*c[k-1])/3;
  }
  printresult();
  testresult();
  freematrix();
}
```

Чтобы воспользоваться этой программой, нужно запустить скомпилированный исполняемый файл. В первую очередь программа спросит, откуда ей брать данные для

интерполяции. Создайте в любом текстовом редакторе (но только не в Word-е а, например в notepad-е) файл, где напишите значения x_k, y_k, построчно через пробел, приблизительно так:

x_0	y_0
x_1	y_1
x_2	y_2
...	
x_N	y_N

Например,

1	5
2	3
3	2.5
4	2
5	0

Этот файл необходимо создать в той директории, где лежит программа, иначе она его не найдет. В результате работы программы, она выдаст нечто вроде:

A[k]	*B[k]*	*C[k]*	*D[k]*
3.000000	-1.250000	1.125000	0.375000
2.500000	-0.125000	0.000000	-0.375000
2.000000	-1.250000	-1.125000	-0.375000
0.000000	-2.375000	0.000000	0.375000

Это и есть решение системы, т.е. набор коэффициентов a_k, b_k. c_k, d_k на четырех отрезках. Кроме того, программа создаст файл test.txt в котором запишет подсчитанные значения интерполирующей функции в 20-точках на отрезке $[x_0; x_N]$. Вы сможете убедиться, что значения интерполирующей функции плавно меняются от точки к точке, и в точках x_k совпадают со значениями y_k.

Коротко опишем, для чего служит такое большое количество подпрограмм в данной программе:

void count_num_lines() - подсчитывает количество точек, где задана функция

void allocmatrix() - выделяет память для массивов b, c, d, x, y, delta, lambda

void readmatrix() - прочитывает из файла координаты x, y точек

void testresult() - на основании вычисленных коэффициентов *a, b, c, d* вычисляет значение интерполирующей функции в 20 точках на интервале $[x_0; x_N]$ с равномерным шагом

void printresult() - распечатывает столбцы коэффициентов a, b, c, d

void freematrix() - освобождает память, которая была выделена ранее

void cls() - стирает экран в начале работы программы

void main() - основная функция из которой последовательно вызываются все вышеперечисленные функции, и проходит процесс вычисления коэффициентов по формулам прямой и обратной прогонки.

Варианты заданий к лабораторной работе № 8

Варианты № 1 – № 5

По приведенной ниже таблице значений функции требуется построить квадратичный сплайн, который в точке x=0.3 равен 2.67 и который в каждой данной точке имеет первую производную, равную результату численного дифференцирования данной функции в данной точке

x	0	0.3	0.6	0.9	1.2	1.5	1.8	2.1	2.4	2.7
f(x)	3.76	2.67	2.84	1.17	2.39	4.98	5.28	5.91	4.27	3.44

Ответ вывести в аналитическом и графическом вариантах.

Варианты № 6 – № 10

По приведенной ниже таблице значений функции требуется построить квадратичный сплайн, который в точке x=0.91 равен 4.12 и который в каждой данной точке имеет первую производную, равную результату численного дифференцирования данной функции в данной точке:

x	0	0.25	0.76	0.91	1.83	2.35	4.72	4.99	5.87	6.45
f(x)	-1.34	-3.14	3.83	4.12	-3.12	-2.43	-2.65	-1.23	0.76	1.28

Ответ вывести в аналитическом и графическом вариантах.

Варианты № 11 – № 15

По приведенной ниже таблице значений функции требуется построить квадратичный сплайн, который в точке x=1.67 равен -5.23 и который в каждой данной точке имеет первую производную, равную результату численного дифференцирования данной функции в данной точке:

x	1.23	1.67	2.04	2.34	2.56	2.99	3.34	4.54	4.87	5.11
f(x)	-3.21	-5.23	-0.23	-1.17	0.32	0.43	0.99	1.54	4.34	9.12

Ответ вывести в аналитическом и графическом вариантах.

Варианты № 16 – № 20

По приведенной ниже таблице значений функции требуется построить квадратичный сплайн, который в точке x=1.5 равен 0.43 и который в каждой данной точке имеет первую производную, равную результату численного дифференцирования данной функции в данной точке:

x	-1.2	-0.5	0.4	1.5	2.1	2.9	3.3	3.9	4.3	5.1
f(x)	-2.11	-2.33	-0.14	0.43	1.34	2.65	6.23	9.23	7.65	4.23

Ответ вывести в аналитическом и графическом вариантах.

Варианты № 21 – № 25

По приведенной ниже таблице значений функции требуется построить квадратичный сплайн, который в точке x=4.12 равен 1.23 и который в каждой данной точке имеет первую производную, равную результату численного дифференцирования данной функции в данной точке:

x	2.1	2.67	3.65	4.12	4.78	5.32	5.99	6.13	6.56	7.01
f(x)	-4.21	-1.23	-0.45	1.23	1.01	2.03	2.76	2.43	-3.34	4.12

Ответ вывести в аналитическом и графическом вариантах.

Варианты № 26 – № 30

По приведенной ниже таблице значений функции требуется построить квадратичный сплайн, который в точке x=1.67 равен -5.23 и который в каждой данной точке имеет первую производную, равную результату численного дифференцирования данной функции в данной точке:

x	1.23	1.67	2.04	2.34	2.56	2.99	3.34	4.54	4.87	5.11
f(x)	-3.21	-5.23	-0.23	-1.17	0.32	0.43	0.99	1.54	4.34	9.12

Ответ вывести в аналитическом и графическом вариантах.