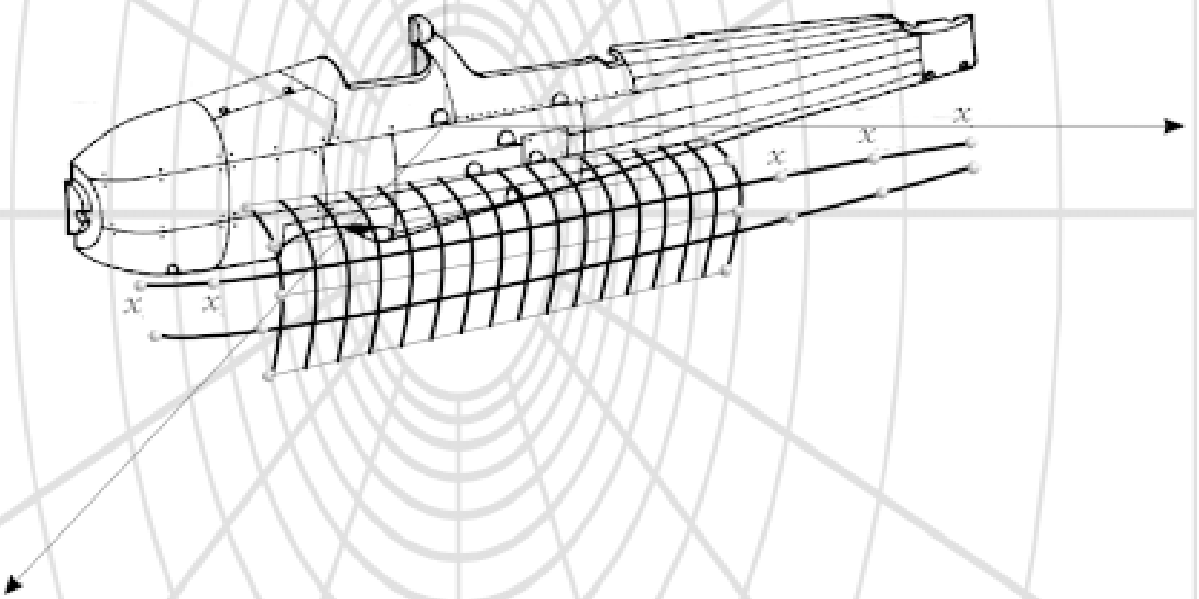
№ 4 (197) январь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Сплайны — инструмент интерполяции

Ю.А. СОКОЛИНСКИЙ



Содержание

Введение

Мы расскажем о современном методе интерполяции, основанном на использовании *сплайнов*. В первом разделе рассматривается классический метод интерполяции, использующий многочлены Лагранжа. Сообщается, какие трудности и проблемы связаны с этим методом. Во втором разделе дается определение сплайна и излагаются идеи, на которых основан метод сплайнов. Третий раздел посвящен алгоритму построения сплайнов. В четвертом разделе демонстрируется эффективность интерполяции с помощью сплайнов. Наконец, в пятом разделе рассказывается о применении параметрических сплайнов для проведения плавных кривых через заданные точки.

Для всех основных алгоритмов интерполяции приводятся соответствующие функции и процедуры на языках программирования Паскаль, Си и Бейсик (версия QBasic). Также приводятся программы, иллюстрирующие применение этих функций и процедур.

1. Интерполяционные многочлены Лагранжа

Напомним постановку задачи интерполяции: имея таблицу значений функции и некоторый класс непрерывных функций, выбрать из него функцию, график которой проходит через заданные точки или, в геометрической трактовке, провести плавную кривую через заданные точки. Класс функций должен быть достаточно узким, чтобы выбор был единственным, и таким, чтобы найденная функция отвечала нашему интуитивному представлению о плавной кривой.

Издавна считалось, что такой класс образуют многочлены. Задача интерполяции с помощью многочленов формулируется следующим образом. Промежуток $[x_0, x_n]$ разбит на n интервалов точками (или, как говорят, узлами):

$$x_0, x_1, x_2, \ldots, x_n$$

— так что $x_{i-1} < x_i$. В узлах заданы значения функции $F(x)$: $y_i = F(x_i)$ (см. *рис.* 1). Требуется построить многочлен возможно меньшей степени, значения которого в узлах равны y_i, т.е. они должны совпадать со значениями функции $F(x)$. Такой многочлен называется интерполяционным многочленом функции $F(x)$. Естественно полагать, что при достаточном числе узлов график интерполяционного многочлена будет похож на график порождающей его функции, однако это всего лишь предположение, гипотеза.

Указанная задача привлекала внимание многих математиков, включая Ньютона и Гаусса, начиная с XVII века. Видимо, наиболее простую формулу интерполяционного многочлена предложил Лагранж [1]. Сначала приведем формулу для i-го "фундаментального" многочлена Лагранжа:

$$L_i(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)}.$$

Сразу видно, что это выражение — многочлен степени n и, кроме того, в i-м узле он равен единице, а в остальных узлах — нулю. Отсюда следует, что интерполяционный многочлен можно записать в виде:

$$L(x) = y_0 L_0(x) + y_1 L_1(x) + \ldots + y_n L_n(x).$$

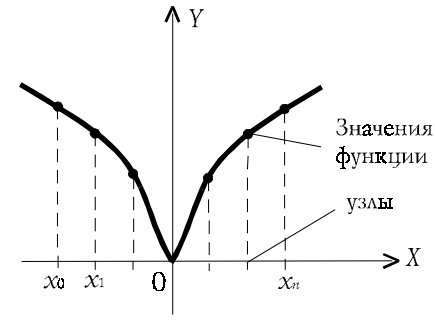


Рис. 1

Приведем запись функции `Lagr(n, ax, ay, x)`, позволяющей находить значение интерполяционного многочлена Лагранжа степени n в точке x, для указанных выше языков программирования. При этом `ax, ay` — массивы узлов и ординат.

Язык Паскаль

Здесь и ниже используются константа и тип:

```
const Nmax=50;{Максимальное число интервалов}
type tArr=array[0..Nmax] of double;
function Lagr(n:byte; var aX,aY:tArr; x:double):double;
var I,J:byte; Sum,Add:double;
begin
  Sum:=0;
  for I:=0 to n do begin
   Add:=aY[I];
   for J:=0 to n do
    if J<>I then Add:=Add*(x-aX[J])/(aX[I]-aX[J]);
   Sum:=Sum+Add;
                  end;
Lagr:=Sum;
end;{Lagr}
```

Язык Си

```
double Lagr(int n, double *ax, double *ay, double x)
{int i,j; double sum=0,add;
 for (i=0; i<=n; i++)
  {add=ay[i];
   for (j=0; j<=n; j++)
   if (i != j) add*=(x-ax[j])/(ax[i]-ax[j]);
   sum+=add;
  }
 return sum;
}
```

Язык Бейсик

```
FUNCTION Lagr# (n%, ax#(), ay#(), x#)
DIM i%, j%, add#, sum#
sum# = 0
FOR i% = 0 TO n%
 add# = ay#(i%)
  FOR j% = 0 TO n%
   IF NOT i% = j% THEN
    add# = add# * (x# - ax#(j%)) / (ax#(i%) - ax#(j%))
   END IF
  NEXT j%
 sum# = sum# + add#
NEXT i%
Lagr# = sum#
END FUNCTION
```

Проведем небольшое исследование "качества" аппроксимации посредством интерполяционных многочленов Лагранжа. Используем в качестве тестов две функции:

а) экспоненту $\exp x$ в промежутке $[0, 2]$;

б) функцию $F(x) = \ln(1 + 100x^2)$ в промежутке $(-1, 1)$. (1)

Составим соответствующие программы. При этом узлы будем предполагать равноотстоящими. Значения функции и интерполяционного многочлена будем вычислять в узлах и посредине каждого интервала между ними. Разумеется, в узлах значения функции и интерполяционного многочлена должны совпасть. Во-первых, не помешает в этом убедиться, а во-вторых, они пригодятся, если мы захотим построить графики функции и ее интерполяционного многочлена. Параметр — число интервалов n — задается непосредственно в программе как константа.

Язык Паскаль

```
{Программа тестирования интерполяционных многочленов Лагранжа}
const Nmax=50; {Максимальное число интервалов}
type
tArr=array[0..Nmax] of double;
Tarr2=array[1..2] of double;
const
NameF:array[1..2] of string[50]=(
'       1. F(x)=exp(x)',
'       2. F(x)=ln(1+100*x*x)');
n=8;{число интервалов}
Xmin:tArr2=(0,-1);  Xmax:tArr2=(2,1);{Граничные узлы}
var
i,NF:byte;
aX,aY:tArr;
h,Xout,Yout,YLagr:double;
Fout:text;
function Lagr(n:byte; var aX,aY:tArr; x:double):double;
...
  end;{Lagr}
function F(NF:byte; x:double):double;
{Функции, для которых находятся интерполяционные многочлены}
begin
 Case NF of
```

```
   1: F:=exp(x);
   2: F:=ln(1+100*x*x);
end;{Case NF}
end;
BEGIN
  {Связываем логический файл Fout с "физическим" - 'Fout' и открываем его на запись}
  assign(Fout,'Fout'); rewrite(Fout);
  for NF:=1 to 2 do begin {Цикл функций}
   h:=(Xmax[NF]-Xmin[NF])/n; {шаг узлов}
   {Вычисляем узлы и значения тестируемой функции в них}
   for i:=0 to n do
   begin aX[I]:=Xmin[NF]+i*h; aY[i]:=F(NF,aX[I]) end;
     {Заголовок выходной таблицы}
     writeln(Fout,NameF[NF]);
      writeln(Fout,'Xout     F     YLagr');
     {Вычисляем точки вывода, а также значения тестируемой
      функции и интерполяционного многочлена в них }
 for i:=0 to 2*n do begin {Цикл точек вывода}
  Xout:=Xmin[NF]+i*h/2;
  Yout:=F(NF,Xout); YLagr:=Lagr(n,aX,aY,Xout);
  {вывод  строки выходной таблицы}
  writeln(Fout,Xout:6:4,'   ',Yout:9:7,' ',YLagr:10:7);
                    end;{Цикл точек вывода}
                  end;{Цикл функций}
close(Fout);
END.
```

Язык Си

```
#include<stdio.h>
#include<math.h>
#define Nmax 50 /*Максимальное число узлов*/
int N=8;/*Число интервалов*/
double Xmin[2]={0,-1}; Xmax[2]={2,1};/*Граничные узлы*/
double Lagr(int n, double *ax, double *ay, double x)
...
}
double F(int NF,double x)
/* Функции, для которых находятся интерполяционные сплайны  */
{switch (NF)
 {case 0: return exp(x);
  case 1: return log(1+100*x*x);
 }
}
void main()
{int i,NF; double ax[Nmax],ay[Nmax];
 double h,xout,yout,yLagr;
 FILE *Fout;
 char *NameF[2]={
 "       1. F(x)=exp(x)",
 "       2. F(x)=ln(1+100*x*x)"};
 /* Связываем логический файл Fout с "физическим" - "Fout" и открываем его на запись */
  Fout=fopen("Fout","w");
 for (NF=0;NF<2; NF++)/*Цикл функций*/
  {h=(Xmax[NF]-Xmin[NF])/N; /* шаг узлов */
  /*Вычисляем узлы и значения тестируемой функции в них*/
  for (i=0; i<=N; i++)
    {ax[i]=Xmin[NF]+i*h; ay[i]=F(NF,ax[i]);}
   /* заголовок выходной таблицы */
   fprintf(Fout,"%.50s\n",NameF[NF]);
     fprintf(Fout,"Xout     Yout      YLagr\n");
   /*Вычисляем точки вывода, а также значения тестируемой
     функции и интерполяционного многочлена в них */
   for (i=0; i<=2*N; i++)/*Цикл точек вывода*/
    {xout=Xmin[NF]+i*h/2;
     yout=F(NF,xout); yLagr=Lagr(N,ax,ay,xout);
    /*Вывод в файл Fout строки выходной таблицы*/
```

```
    fprintf(Fout,"%6.4f    %9.7f %10.7f\n",xout,yout,yLagr);
    }                    /*Цикл точек вывода*/
 }                     /*Цикл функций*/
 fclose(Fout); /* Закрываем выходной файл */
}
```

Язык Бейсик

```
'Программа тестирования интерполяционных
'многочленов Лагранжа
DECLARE FUNCTION Lagr# (n%, ax#(), ay#(), x#)
DECLARE FUNCTION F# (NF%, x#)
DIM Nmax%: Nmax% = 50: 'Максимальное число интервалов
DIM n AS INTEGER: n = 8 'Число интервалов
DIM i AS INTEGER, NF AS INTEGER, nout AS INTEGER
DIM Xout AS DOUBLE, Yout AS DOUBLE, YLagr AS DOUBLE
DIM h AS DOUBLE
DIM ax(0 TO Nmax%) AS DOUBLE, ay(0 TO Nmax%) AS DOUBLE
DIM Xmin(1 TO 2) AS DOUBLE, Xmax(1 TO 2) AS DOUBLE
Xmin(1) = 0: Xmax(1) = 2: Xmin(2) = -1: Xmax(2) = 1
DIM NameF(1 TO 2) AS STRING
NameF(1) = "        1. F(x)=exp(x)"
NameF(2) = "        2. F(x)=ln(1+100*x*x)"
'открываем файл "Fout" на запись и присваиваем
'ему номер #1
OPEN "Fout" FOR OUTPUT AS #1
FOR NF = 1 TO 2 'Цикл функций
 h = (Xmax(NF) - Xmin(NF)) / n 'шаг узлов
 'Вычисляем узлы и значения тестируемой функции  в них
 FOR i = 0 TO n
  ax(i) = Xmin(NF) + i * h: ay(i) = F#(NF, ax#(i))
 NEXT i
 'формат вывода
 format$ = "##.####  #.#######   ###.########"
 'Вывод в файл #1 заголовка выходной таблицы
 PRINT #1, NameF(NF)
 PRINT #1, "Xout     Yout          Lagr"
 'Вычисляем точки вывода, а также значения тестируемой
 'функции и интерполяционного многочлена в них
 FOR i = 0 TO 2 * n 'Цикл точек вывода
  Xout = Xmin(NF) + i * h / 2
  Yout = F(NF, Xout): YLagr = Lagr#(n, ax(), ay(), Xout)
  'Вывод в файл #1 строки выходной таблицы
  PRINT #1, USING format$; Xout; Yout; YLagr
 NEXT i              'Цикл точек вывода
NEXT NF          'Цикл функций
CLOSE 'закрываем файл #1
END

FUNCTION F# (NF%, x#)
'Функции, для которых находятся
'интерполяционные многочлены
SELECT CASE NF%
 CASE 1
  F# = EXP(x#)
 CASE 2
  F# = LOG(1 + 100 * x# * x#)
END SELECT
END FUNCTION
FUNCTION Lagr# (n%, ax#(), ay#(), x#)
...
END FUNCTION
```

Выполнив любую из этих программ при $n=8$, можно убедиться, что расхождение между экспонентой и ее интерполяционным многочленом не превышает двух единиц седьмого дробного разряда, т.е. качество аппроксимации очень высокое.

Для второй функции (1) ситуация совсем иная. На *рис.* 2 представлены графики функции $F(x) = \ln(1 + 100x^2)$ (сплошная линия) и ее интерполяционного многочлена $Lagr(x)$ (пунктир). Видно, что интерполяционный многочлен хотя бы качественно воспроизводит функцию $F(x)$ лишь при малых x, а

при $|x| > 0{,}25$ он начинает осциллировать (колебаться), т.е. проходит через чередующиеся минимумы и максимумы, расположенные внутри шагов интерполяции. При этом с ростом $|x|$ амплитуда колебаний нарастает.

Может быть, число шагов интерполяции $n=8$ слишком мало? Попробуем выполнить какую-нибудь из приведенных программ при $n=16$. Действительно, осцилляции появляются лишь при $|x| > 0{,}5$, но их амплитуда резко увеличивается. Это затрудняет построение графиков, и мы продемонстрируем сравнительную таблицу функций $F(x)$ и $Lagr(x)$ при $|x| > 0{,}5$.

функция \ $\|x\|$	0,5625	0,6875	0,8125	0,9375
F	3,49	3,89	4,20	4,49
Lagr	3,18	4,88	− 0,76	49,3

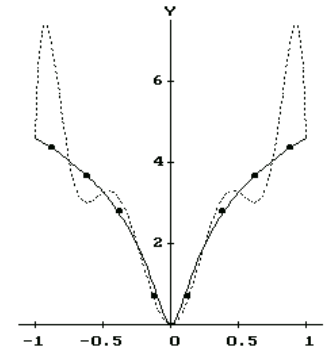


Рис. 2

Видно, что колебания интерполяционного многочлена растут просто "катастрофически".

Даже из этого примера видно, что использовать многочлены Лагранжа следует с большой осторожностью.

2. Что такое сплайн и откуда он взялся

В 60-х годах нашего столетия был предложен совершенно новый метод интерполяции, основанный на использовании класса *сплайн-функций,* или просто *сплайнов* [1, 2]. Сплайн можно определить как функцию, обладающую следующими свойствами:

а) график сплайна проходит через все заданные точки (x_i, y_i), $i = 0, 1, ..., n$;

б) на каждом интервале $[x_{i-1}, x_i]$ сплайн является кубическим многочленом;

в) во всех внутренних узлах x_i, $i = 1, 2, ..., n - 1$ первая и вторая производные сплайна непрерывны, т.е. сплайн — гладкая функция.

Это не слишком длинное определение вызывает целую гамму вопросов:

— Что означает слово *сплайн*?

— Почему сплайн состоит из многочленов, и притом кубических, а не из каких-либо других функций?

— Предыдущий вопрос, в сущности, означает: какая идея лежит в основе метода сплайнов?

— Является ли приведенное определение сплайна полным, т.е. достаточно ли информации о заданных точках для построения сплайна?

— Как построить сплайн, т.е. как определить коэффициенты кубических многочленов, из которых состоит сплайн?

Начнем с терминологии. Английское слово *spline* означает здесь длинную и тонкую линейку, изгибая которую проводят плавную кривую через заданные точки.

Ответ на вопрос об идейных истоках метода сплайнов потребует значительно больше места. Как и во времена Ньютона и Эйлера, изобретатели метода сплайнов вдохновлялись идеями, заимствованными из механики. В данном случае речь идет о разделе механики, известном под названием *сопротивление материалов* [3]. На студенческом жаргоне этот предмет имеет сокращенное название *сопромат,* и отношение к нему достаточно ярко характеризует поговорка: "Сдал сопромат — можешь жениться".

Мы рассмотрим (разумеется, очень кратко и бегло) раздел сопротивления материалов, известный как *теория балок.* Под балкой подразумевается прямолинейный стержень, обладающий определенной жесткостью, поперечные размеры которого значительно меньше его длины. Это означает, что изгиб балки (а также ее растяжение) требует заметных усилий, в отличие от *струны* (нити), которая свободно изгибается, но сопротивляется растяжению. Пусть балка направлена горизонтально и определенным способом закреплена на опорах, расположенных на ее концах (см. *рис.* 3). Направим ось x вдоль балки, а ось y — вертикально. Координаты концов балки обозначим через x_0 и x_n. На балку действуют силы, которые мы будем считать вертикальными. Координаты точек приложения сил обозначим : $x_1, x_2, ..., x_{n-1}$.

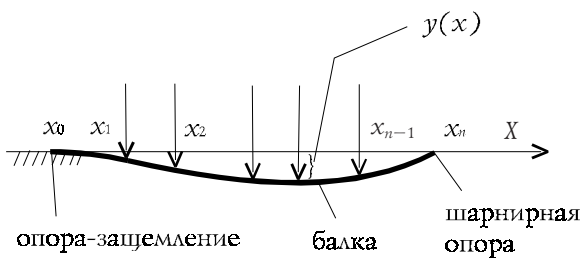


Рис. 3

Жесткость балки не абсолютна — под действием приложенных сил она деформируется. Различают *линейные* деформации — перемещения точек балки по вертикали — и *угловые* деформации — углы поворота поперечных сечений балки. Обозначим линейное перемещение в точке x через $y(x)$. Нетрудно заметить, что угловое перемещение — это угол наклона касательной к функции $y(x)$. Сформулируем ***основную гипотезу*** теории балок: абсолютные значения линейных и угловых перемещений весь-

ма малы. Отсюда следует, что в качестве углового перемещения с достаточной точностью можно принять производную функции $y(x)$, т.е. $y'(x)$.

Выше были упомянуты опоры балки. Различают два типа опор: *шарнирная опора* и *защемление*. Шарнирная опора фиксирует линейное перемещение балки в точке приложения опоры, но не препятствует угловому перемещению в ней, а защемление фиксирует как линейное, так и угловое перемещение. В опорах возникают *реактивные усилия* (реакции), приложенные к балке. Шарнирной опоре соответствует реактивная сила, а защемлению — еще и реактивный изгибающий момент. Мы не будем заниматься вопросом, как определить реакции в опорах, нам достаточно знать, что они существуют и зависят от типа опоры.

Одним из центральных понятий теории балок является внутренний изгибающий момент $M(x)$, возникающий в каждой точке (каждом сечении) балки. Он определяется как сумма моментов всех сил, приложенных левее рассматриваемой точки с координатой x, т.е. сил, точки приложения которых $x_i < x$. В эту сумму входит момент реактивной силы, приложенной к точке x_0, а также реактивный изгибающий момент, приложенный к этой точке, если он существует, т.е. опора является защемлением. Напомним, что момент силы есть произведение силы на "плечо" — расстояние от точки приложения силы до выбранной точки (с соответствующим знаком). Из этого определения следует важный факт: внутренний момент $M(x)$ — *непрерывная* функция x и на каждом интервале $[x_{i-1}, x_i]$ она *линейна*.

Запишем теперь ***основное уравнение*** теории балки, связывающее силы и моменты, действующие на балку, с деформациями:

$$y''(x) = \frac{M(x)}{Z},$$

где Z — так называемая изгибная жесткость балки, определяемая ее материалом и геометрией поперечного сечения.

Отсюда с учетом указанных выше свойств $M(x)$ — непрерывности и линейности на интервалах $[x_{i-1}, x_i]$ — следует, что функция $y(x)$ на каждом интервале $[x_{i-1}, x_i]$ является кубическим многочленом. Очевидно также, что ее первая и вторая производные непрерывны. Другими словами, кривая $y(x)$, описывающая линейную деформацию балки, — это сплайн! Вот путь, по которому шли изобретатели сплайна.

Давайте разберемся, что мы, собственно, получили: сплайны возникают естественным образом в некоторой прикладной дисциплине, а именно — сопротивлении материалов. Есть ощущение, что этого недостаточно. Все-таки определение сплайна носит чисто математический характер, и было бы вполне разумным, чтобы сплайны обладали каким-то свойством, выделяющим их среди остальных функций. Такое свойство есть, но подойти к нему поможет та же механическая аналогия. В сопротивлении материалов широко используется энергетический метод: балка деформируется таким образом, чтобы потенциальная энергия деформации была минимальной. Указанная энергия пропорциональна интегралу вида

$$\int_{x_0}^{x_n} y''^2 dx.$$

Отметим, что вторая производная характеризует кривизну в данной точке. Поэтому сплайн является такой гладкой кривой, проходящей через заданные точки, которая имеет минимальную среднюю кривизну.

На языке математики это означает, что среди всех гладких функций, проходящих через заданные точки, сплайн обладает минимальным значением интеграла от квадрата второй производной или, что то же самое, сплайн обладает минимальной среднеквадратичной второй производной. Это утверждение, доказываемое чисто математически, и есть то свойство сплайнов, которое мы искали.

3. Построение сплайна*

Перейдем к вопросам построения сплайна, т.е. определения коэффициентов кубических многочленов, из которых он состоит. Главную роль будут играть вторые производные сплайна в узлах. Их принято обозначать $m_i = y''(x_i)$ и называть *моментами* по причинам, понятным из сказанного выше. Вторая производная кубического многочлена линейна. Следовательно, на i-м интервале вторую производную сплайна можно выразить как

$$y''(x) = \frac{m_{i-1}(x_i - x) + m_i(x - x_{i-1})}{h_i},$$

где $h_i = x_i - x_{i-1}$ шаг i-го интервала.

* В третьей главе обсуждаются достаточно сложные вопросы, связанные с выводом уравнения сплайна. В следующих главах в программах используются только уравнения, приведенные в конце главы. Таким образом, при первом чтении вывод уравнений можно пропустить, познакомившись лишь с окончательными результатами.

Интегрируя это выражение дважды, получаем:

$$y(x) = \frac{m_{i-1}(x_i - x)^3 + m_i(x - x_{i-1})^3}{6h_i} + C_1 x + C_2 \qquad \text{(i)}$$

— где C_1 и C_2 — константы интегрирования. Используем значения сплайна в $(i-1)$-м и i-м узлах. Обозначим

$$g(x) = C_1 x + C_2 = y(x) - \frac{m_{i-1}(x_i - x)^3 + m_i(x - x_{i-1})^3}{6h_i}. \qquad \text{(ii)}$$

Эта функция также линейная. Выразим ее значения в $(i-1)$-м и i-м узлах.

$$g(x_{i-1}) = y_{i-1} - \frac{m_{i-1}}{6} h_i^2, \qquad \text{(iii)}$$

$$g(x_i) = y_i - \frac{m_i}{6} h_i^2. \qquad \text{(iv)}$$

Из (ii), (iii), (iv) можно получить (v)

$$g(x) = \left(\frac{y_{i-1}}{h_i} - \frac{m_{i-1}h_i}{6}\right)(x_i - x) + \left(\frac{y_i}{h_i} - \frac{m_i h_i}{6}\right)(x - x_{i-1}). \qquad \text{(v)}$$

Подставляя (v) в (i), получим:

$$y(x) = \frac{m_{i-1}(x_i - x)^3 + m_i(x - x_{i-1})^3}{6h_i} + \left(\frac{y_{i-1}}{h_i} - \frac{m_{i-1}h_i}{6}\right)(x_i - x) + \left(\frac{y_i}{h_i} - \frac{m_i h_i}{6}\right)(x - x_{i-1}). \qquad (2)$$

Чтобы вычислять сплайн с помощью этого выражения, надо иметь значения моментов. Займемся этой задачей. Воспользуемся условием непрерывности первой производной сплайна (т.е. отсутствием излома) во внутренних узлах. Дифференцируя уравнение сплайна, находим выражение его производной:

$$y'(x) = -\frac{m_{i-1}}{2h_i}(x_i - x)^2 + \frac{m_i}{2h_i}(x - x_{i-1})^2 - \frac{h_i}{6}(m_i - m_{i-1}) + k_i \qquad (3)$$

— где $k_i = (y_i - y_{i-1})/h_i$ — угловой коэффициент i-й хорды.

Подставим сюда правую границу i-го интервала:

$$y'(x_i) = \frac{m_i h_i}{3} + \frac{m_{i-1}h_i}{6} + k_i \qquad (4)$$

Записав уравнение (3) для $(i+1)$-го интервала и подставив в него левую границу этого интервала, найдем другое выражение для этой же производной:

$$y'(x_i) = -\frac{m_i h_{i+1}}{3} - \frac{m_{i+1}h_{i+1}}{6} + k_{i+1}. \qquad (5)$$

Приравнивая выражения (4) и (5), получаем уравнение, связывающее моменты в трех последовательных узлах — $(i-1)$-м, i-м, $(i+1)$-м:

$$h_i m_{i-1} + 2(h_i + h_{i+1})m_i + h_{i+1} = 6(k_{i+1} - k_i). \qquad (6)$$

Сколько таких уравнений мы имеем? Их количество равно числу внутренних узлов, т.е. оно составляет $n-1$. А моменты нам надо знать во всех узлах, число которых $n+1$. Значит, двух уравнений не хватает. Таким образом, ответ на вопрос, является ли приведенное выше определение сплайна полным, *отрицательный*. Такой же ответ подсказывает и наша аналогия: для нахождения деформаций балки надо знать, помимо прочего, как она закрепляется на концах, т.е. нужны еще граничные условия.

Для сплайнов используют три варианта граничных условий.

1) "Свободный" конец с нулевым моментом: $m_0 = 0$ и/или $m_n = 0$. Название подсказано аналогией: шарнирно закрепленный конец балки свободно поворачивается, и изгибающий момент в нем равен нулю. Это условие можно использовать, если известно, что:

а) Граничный узел — это точка перегиба (такова, например, точка $x = 0$ для функций $y = \sin x$ и $y = x^3$);

б) Начальная (конечная) часть сплайна — прямая линия.

2) Параболический (квадратичный) граничный интервал:

$$m_0 = m_1 \quad \text{и/или} \quad m_n = m_{n-1}.$$

Название означает, что для граничного интервала вместо кубического многочлена берется параболический (второй степени). Как известно, вторая производная параболы постоянна. Это граничное условие используется, когда отсутствует информация о значениях первой и второй производной в граничной точке.

3) Защемление: в граничной точке задается значение производной.

а) Граничная точка — левая: $d_0 = y'(x_0)$. Используя формулу (5) при $i = 0$, запишем граничное условие в виде:

$$m_0 = -\frac{1}{2} m_1 + \frac{3(k_1 - d_0)}{h_1}.$$

б) Граничная точка — правая: $d_n = y'(x_n)$. Теперь используем формулу (4) при $i = n$. Граничное условие имеет вид:

$$m_{n-1} = -2m_n + \frac{6(d_n - k_n)}{h_n}.$$

Для всех трех вариантов граничные условия можно представить как

$$m_0 = rm_1 + s, \tag{7}$$

$$m_{n-1} = um_n + v. \tag{8}$$

Итак, граничные условия позволили получить два недостающих уравнения. Система уравнений (6) (7), (8) при $i = 1, 2, ..., n$, которую нам предстоит решить, — "трехдиагональная". Это означает, что в матрице коэффициентов элементы, отличные от нуля, располагаются на главной диагонали и двух соседних с ней. Для таких систем существует эффективный метод решения, известный как "метод прогонки", которым мы и воспользуемся. Идея заключается в следующем. Уравнение (7) выражает нулевой момент через первый. Подставив его в уравнение (6) при $i = 1$, мы выразим первый момент через второй. Продолжая этот процесс, последовательно выражаем каждый момент через его правого соседа и закончим, выразив m_{n-1} через m_n. Но то же самое делает уравнение (8). Приравнивая эти выражения, находим последний момент m_n. Теперь находим предпоследний момент m_{n-1} — ведь он выражен через последний, затем момент m_{n-2} и т.д. вплоть до нулевого момента m_0. Перейдем к технической реализации описанного алгоритма.

Пусть каждый момент, кроме последнего, представлен в виде:

$$m_{i-1} = P_i m_i + Q_i, \qquad i = 1, 2, ..., n \tag{8}$$

— где P_i, Q_i — "прогоночные" коэффициенты, причем из уравнения (7) следует, что $P_1 = r$, $Q_1 = s$.

Подставив это выражение в уравнение (6), получим:

$$m_i[h_i P_i + 2(h_i + h_{i+1})] + m_{i+1} h_{i+1} = 6(k_{i+1} - k_i) - h_i Q_i.$$

Видно, что для прогоночных коэффициентов выполняются соотношения:

$$P_{i+1} = -\frac{h_{i+1}}{h_i P_i + 2(h_i + h_{i+1})},$$

$$Q_{i+1} = \frac{6(k_{i+1} - k_i) - h_i Q_i}{h_i P_i + 2(h_i + h_{i+1})}.$$

Они позволяют последовательно вычислить все прогоночные коэффициенты, начиная со второго и кончая n-м. Это "прямой ход" прогонки.

Записав уравнение (8) при $i = n$ и сопоставив его с уравнением (7), находим последний момент m_n:

$$m_n = \frac{Q_n - v}{u - P_n}.$$

А теперь выполняем "обратный ход" прогонки: по формуле (8) находим все моменты, начиная с $(n-1)$-го и кончая первым.

Таким образом, сплайн вычисляется по формуле (2)*, но, прежде чем ею воспользоваться, надо найти моменты, следуя указанному алгоритму.*

Представим теперь процедуру
MomSpline(n,aX,aY,Mom,BegT,EndT,DerB,DerE)
и функцию Spline(aX,aY,Mom,x).

Процедура MomSpline находит массив моментов сплайна Mom на основании заданного числа интервалов n, массивов узлов aX и ординат aY, кодов условий на левом и правом конце BegT,EndT и производных в граничных узлах DerB,DerE. Коды граничных условий следующие:

1 — свободный конец с нулевым моментом;
2 — квадратичный (параболический) концевой отрезок;
3 — в концевом узле задана производная.

Если код граничного условия отличен от 3, то значение производной в соответствующем узле — произвольное.

Функция Spline находит значение сплайна, заданного массивами узлов, ординат и моментов, в точке x.

Язык Паскаль

```
procedure MomSpline(n:byte; var aX,aY,Mom:tArr;
              begT,EndT:byte; DerB,DerE:double);
{Нахождение массива моментов сплайна Mom;
 aX,aY - массивы узлов и ординат;
 BegT,EndT - коды условия на левом и правом конце;
 DerB,DerE - производная в соответствующем концевом узле.
 Если код граничного условия отличен от 3, то значение производной - произвольное}
var
I:byte;
h,k,P,Q:tArr;{h,k - массивы шагов и угловых коэффициентов.
              P,Q - массивы прогоночных коэффициентов.    }
Z,r,s,u,v:double;{Mom[0]=r*Mom[1]+s;  Mom[n-1]=u*Mom[n]+v}
begin
for I:=1 to n do begin
 {Шаг и угловой коэффициент i-го интервала}
 h[I]:=aX[I]-aX[I-1];
 k[I]:=(aY[I]-aY[I-1])/h[I];
                  end;
Case BegT of
1:{Свободный нач. узел: нач. момент=0}
  begin r:=0; s:=0 end;
2:{Квадратичный нач. отрезок: Mom[0]=Mom[1]}
  begin r:=1; s:=0 end;
3:{В нач. узле задана производная DerB}
  begin r:=-0.5; s:=3*(k[1]-DerB)/h[1] end;
end;{Case BegT}
Case EndT of
1:{Свободный кон. узел: кон. момент=0}
  begin u:=0; v:=0 end;
2:{Квадратичный кон. отрезок: Mom[n-1]=Mom[n]}
  begin u:=1; v:=0 end;
3:{В кон. узле задана производная DerE}
  begin u:=-2; v:=6*(DerE-k[n])/h[n]; end;
end;{Case EndT}
P[1]:=r; Q[1]:=s;
for I:=1 to n-1 do begin{Прямой ход прогонки}
Z:=h[I]*P[I]+2*(h[I]+h[I+1]);
P[I+1]:=-h[I+1]/Z;
Q[I+1]:=(6*(k[I+1]-k[I])-h[I]*Q[I])/Z;
                    end;{Прямой ход прогонки}
Mom[n]:=(Q[n]-v)/(u-P[n]);{Момент на правом конце}
{Обратный ход прогонки - определение моментов}
For I:=n-1 downto 0 do
Mom[I]:=P[I+1]*Mom[I+1]+Q[I+1];
end;{MomSpline}

function Spline(var aX,aY,Mom:tArr; x:double):double;
{Нахождение значения сплайна в точке x, лежащей внутри промежутка aX[0],aX[n]. Сплайн
задан массивами узлов aX, ординат aY, моментов Mom}
var
I:byte;
h,h1,h2,a3,b3,a1,b1:double;
begin
{Поиск интервала, содержащего точку x}
```

```
i:=1; while not ((aX[I-1]<=x) and (x<=aX[I])) do Inc(I);
{Вычисление значения сплайна}
h:=aX[I]-aX[I-1];
h1:=aX[I]-x;  h2:=x-aX[I-1];
a3:=Mom[I-1]/(6*h);  b3:=Mom[I]/(6*h);
a1:=aY[I-1]/h-Mom[I-1]*h/6;
b1:=aY[I]/h-Mom[I]*h/6;
Spline:=a3*h1*h1*h1+b3*h2*h2*h2+a1*h1+b1*h2;
end;{Spline}
```

Язык Си

```
void MomSpline(int n, double *ax, double *ay, double *mom,
               int begT, int endT, double derB, double derE)
/* Нахождение массива моментов сплайна Mom;
   ax,ay - массивы узлов и ординат;
   begT,endT - коды условия на левом и правом конце;
   derB,derE - производная в соответствующем концевом узле.
 Если код граничного условия отличен от 3, то значение производной - произвольное */
{int i; double h[Nmax],k[Nmax],P[Nmax],Q[Nmax];
 /* h,k - массивы шагов и угловых коэффициентов
    P,Q - массивы прогоночных коэффициентов */
 double Z,r,s,u,v;
 /*mom[0]=r*mom[1]+s;  mom[n-1]=u*mom[n]+v*/
 for (i=1; i<=n; i++)
  /* Шаг и угловой коэффициент i-го интервала */
  {h[i]=ax[i]-ax[i-1];
   k[i]=(ay[i]-ay[i-1])/h[i];
  }
  switch(begT)
  {case 1: /* Свободный нач. узел: нач. момент=0 */
   r=0; s=0; break;
   case 2: /* Квадратичный нач. отрезок: Mom[0]=Mom[1] */
   r=1; s=0; break;
   case 3: /* В нач. узле задана производная derB */
   r=-0.5; s=3*(k[1]-derB)/h[1]; break;
  }
  switch(endT)
  {case 1: /* Свободный кон. узел: кон. момент=0 */
   u=0; v=0; break;
   case 2: /* Квадратичный кон. отрезок: mom[n-1]=mom[n] */
   u=1; v=0; break;
   case 3: /* В кон. узле задана производная derE */
   u=-2; v=6*(derE-k[n])/h[n]; break;
  }
  P[1]=r;   Q[1]=s;
  for (i=1; i<n; i++) /* Прямой ход прогонки */
   {Z=h[i]*P[i]+2*(h[i]+h[i+1]);
    P[i+1]=-h[i+1]/Z;
    Q[i+1]=(6*(k[i+1]-k[i])-h[i]*Q[i])/Z;
   }                  /* Прямой ход прогонки */
  mom[n]=(Q[n]-v)/(u-P[n]); /*Момент на правом конце*/
  /*Обратный ход прогонки - определение моментов*/
  for (i=n-1; i>=0; i—) mom[i]=P[i+1]*mom[i+1]+Q[i+1];
}

double Spline(double *ax, double *ay, double *mom, double x)
/* Нахождение значения сплайна в точке x, лежащей внутри промежутка ax[0],ay[n].
Сплайн задан массивами узлов ax, ординат ay, моментов mom  */
{int i; double h,h1,h2,a3,b3,a1,b1;
 /* Поиск интервала, содержащего точку x */
 i=1; while (! ( ax[i-1]<=x && x<=ax[i])) i++;
 /* Вычисление значения сплайна */
 h=ax[i]-ax[i-1];
 h1=ax[i]-x;  h2=x-ax[i-1];
 a3=mom[i-1]/(6*h);  b3=mom[i]/(6*h);
 a1=ay[i-1]/h-mom[i-1]*h/6;
 b1=ay[i]/h-mom[i]*h/6;
 return a3*h1*h1*h1+b3*h2*h2*h2+a1*h1+b1*h2;
}
```

Язык Бейсик

```
SUB MomSpline (n%, aX#(), aY#(), Mom#(), BC%(), Der#())
'Нахождение моментов сплайна Mom.
'n% - число интервалов;
'aX,aY - массивы узлов и ординат;
'BC%(1 TO 2) - коды условия на концах;
'Der#(1 TO 2) - производные в концевых узлах.
'Если код граничного условия отличен от 3, то значение производной - произвольное
DIM i AS INTEGER, Z#, r#, S#, u#, v#
DIM BegT%, EndT%, DerB#, DerE#
DIM Q#(1 TO n%), k#(1 TO n%), P#(1 TO n%), H#(1 TO n%)
'h,k - массивы шагов и угловых коэффициентов
'P,Q - массивы прогоночных коэффициентов
BegT% = BC%(1): EndT% = BC%(2)
DerB# = Der#(1): DerE# = Der#(2)
FOR i = 1 TO n%
'Шаг и угловой коэффициент на i-м интервале
 H#(i) = aX#(i) - aX#(i - 1)
 k#(i) = (aY#(i) - aY#(i - 1)) / H#(i)
NEXT i
SELECT CASE BegT%
CASE 1 'Свободный нач. узел: нач. момент=0
r# = 0: S# = 0
CASE 2 'Квадратичный нач. отрезок: mom[0]=mom[1]
r# = 1: S# = 0
CASE 3 'В нач. узле задана производная derB
r# = -.5: S# = 3 * (k#(1) - DerB#) / H#(1)
END SELECT
SELECT CASE EndT%
CASE 1 'Свободный кон. узел: кон. момент=0
u# = 0: v# = 0
CASE 2 'Квадратичный кон. отрезок: mom[n-1]=mom[n]
u# = 1: v# = 0
CASE 3 'В кон. узле задана производная derE
u# = -2: v# = 6 * (DerE# - k#(n%)) / H#(n%)
END SELECT
P#(1) = r#: Q#(1) = S#
FOR i = 1 TO n% - 1 'Прямой ход прогонки
 Z# = H#(i) * P#(i) + 2 * (H#(i) + H#(i + 1))
 P#(i + 1) = -H#(i + 1) / Z#
 Q#(i + 1) = (6 * (k#(i + 1) - k#(i)) - H#(i) * Q#(i)) / Z#
NEXT i               'Прямой ход прогонки
'Момент на правом конце
Mom#(n%) = (Q#(n%) - v#) / (u# - P#(n%))
'Обратный ход прогонки - определение моментов
FOR i = n% - 1 TO 0 STEP -1
 Mom#(i) = P#(i + 1) * Mom#(i + 1) + Q#(i + 1)
NEXT i
END SUB

FUNCTION Spline# (aX#(), aY#(), Mom#(), X#)
'Нахождение значения сплайна в точке x, лежащей внутри
'промежутка  aX(0),aX(n). Сплайн задан массивами узлов aX,
'ординат aY, моментов Mom.
DIM i AS INTEGER, H AS DOUBLE, h1 AS DOUBLE, h2 AS DOUBLE
DIM a1 AS DOUBLE, a3 AS DOUBLE, b1 AS DOUBLE, b3 AS DOUBLE
'поиск интервала, содержащего точку x
i = 1
DO WHILE NOT (aX#(i - 1) <= X# AND X# <= aX#(i))
i = i + 1
LOOP
H = aX#(i) - aX#(i - 1)
h1 = aX#(i) - X#: h2 = X# - aX#(i - 1)
a3 = Mom#(i - 1) / (6 * H): b3 = Mom#(i) / (6 * H)
a1 = aY#(i - 1) / H - Mom#(i - 1) * H / 6
b1 = aY#(i) / H - Mom#(i) * H / 6
h13# = h1 * h1 * h1: h23# = h2 * h2 * h2
Spline = a3 * h13# + b3 * h23# + a1 * h1 + b1 * h2
END FUNCTION
```

4. Зачем нужны сплайны

Казалось бы, мы выяснили все вопросы, касающиеся сплайнов, но остался один, и весьма существенный: зачем они нужны? Ответ подсказывается уже заглавием статьи — сплайны используются для решения задач интерполяции. Если сплайн построен со значениями функции $F(x)$, т.е. ординаты $y_i = F(x_i)$ при $i = 0, 1, ..., n$, то он является интерполяционным для функции $F(x)$. Естественно ожидать, что при достаточном числе узлов интерполяционный сплайн будет хорошо аппроксимировать порождающую его функцию, однако (как и в случае с интерполяционными многочленами Лагранжа) это предположение, гипотеза.

Опять выполним исследование качества аппроксимации функций соответствующими интерполяционными сплайнами. Проверять будем те же функции, что и в разделе 1, для чего составим аналогичные программы. Значения функции и интерполяционного сплайна будем вычислять только посредине между узлами. Дополнительно будем находить процент отклонения значений функции и интерполяционного сплайна.

Для сплайнов надо выбрать тип граничных условий. Производные выбранных нами функций легко находятся:

а) $\exp'x = \exp x$;

б) Производная функции (1) имеет вид:

$$F'(x) = \frac{200x}{1 + 100x^2}.$$

Поэтому для обеих функций на каждом конце используем граничное условие 3.

Приведем указанные программы:

Язык Паскаль

```
{Программа тестирования интерполяционных сплайнов}
const Nmax=50;{Максимальное число интервалов}
type
tArr=array[0..Nmax] of double;
tArr2=array[1..2] of double;
const
NameF:array[1..2] of string[50]=(
'       1. F(x)=exp(x)',
'       2. F(x)=ln(1+100*x*x)');
n:byte=8;{Число интервалов}
Xmin:tArr2=(0,-1); Xmax:tArr2=(2,1); {Граничные узлы}
var
i,NF:byte; DerB,DerE:double;
aX,aY,Mom:tArr;
Der:array[1..2] of tArr2;{матрица производных в гран. узлах}
h,Xout,Yout,YSpline,dY:double;
Fout:text;
procedure MomSpline(n:byte; var aX,aY,Mom:tArr;
             begT,EndT:byte; DerB,DerE:double);
...
end;{MomSpline}
function Spline(var aX,aY,Mom:tArr; x:double):double;
...
end;{Spline}
function F(NF:byte; x:double):double;
{Функции, для которых находятся интерполяционные сплайны}
begin
Case NF of
1: F:=exp(x);
2: F:=ln(1+100*x*x);
end;{Case NF}
end;{F}
Procedure BoundDer(NF:byte);
{Производные в граничных точках}
begin
Case NF of
1: begin
    Der[1][1]:=exp(Xmin[1]); Der[1][2]:=exp(Xmax[1]);
   end;
2: begin
    Der[2][1]:=200*Xmin[2]/(1+100*sqr(Xmin[2]));
    Der[2][2]:=200*Xmax[2]/(1+100*sqr(Xmax[2]));
   end;
```

```
end;{Case NF}
end;{BoundDer}
BEGIN
{Связываем логический файл Fout с "физическим" - 'Fout' и открываем его на запись}
assign(Fout,'Fout'); rewrite(Fout);
for NF:=1 to 2 do begin{Цикл функций}
 h:=(Xmax[NF]-Xmin[NF])/n; {шаг узлов}
 {Вычисляем узлы и значения тестируемой функции в них}
 for i:=0 to n do
  begin aX[I]:=Xmin[NF]+i*h; aY[i]:=F(NF,aX[I]) end;
 {Граничные условия}
 BoundDer(NF);
 DerB:=Der[NF][1]; DerE:=Der[NF][2];
 {Нахождение моментов}
 MomSpline(n,aX,aY,Mom,3,3,DerB,DerE);
 {Заголовок выходной таблицы}
 writeln(Fout,NameF[NF]);
 writeln(Fout,'Xout     F          YSpline   dY');
 {Для каждой точки вывода находим значения тестируемой функции, интерполяционного
сплайна и относительного отклонения этих величин в %  }
  for i:=1 to n do begin{Цикл точек вывода}
   Xout:=(aX[i-1]+aX[i])/2;
    Yout:=F(NF,Xout); YSpline:=Spline(aX,aY,Mom,Xout);
    dY:=(Yout-YSpline)/Yout*100;
    {вывод  строки выходной таблицы}
   writeln
    (Fout,Xout:6:4,'   ',Yout:9:7,' ',YSpline:10:7,' ',dY:7:4);
                   end;{Цикл точек вывода}
                 end;{Цикл функций}
close(Fout);{Закрываем файл Fout}
END.
```

Язык Си

```
/*Программа тестирования интерполяционных сплайнов*/
#include<stdio.h>
#include<math.h>
#define Nmax 50 /*Максимальное число узлов*/
int N=16; /*Число интервалов*/
double Xmin[2]={0,-1}; Xmax[2]={2,1};/*Граничные узлы*/
double Der[2][2];/*матрица производных в гран. узлах*/
void MomSpline(int n, double *ax, double *ay, double *mom,
              int begT, int endT, double derB, double derE)
...
}
double Spline(double *ax, double *ay, double *mom, double x)
...
}
double F(int NF,double x)
/* Функции, для которых находятся интерполяционные сплайны */
{switch (NF)
 {case 0: return exp(x);
  case 1: return log(1+100*x*x);
 }
}
void BoundDer(int NF)
/*Производные в граничных точках*/
{switch (NF)
 {case 0:
    Der[0][0]=exp(Xmin[0]); Der[0][1]=exp(Xmax[0]);
   break;
  case 1:
    Der[1][0]=200*Xmin[1]/(1+100.0*Xmin[1]*Xmin[1]);
    Der[1][1]=200*Xmax[1]/(1+100.0*Xmax[1]*Xmax[1]);
   break;
 }
}
void main()
{int i,NF; double DerB,DerE;
 double ax[Nmax],ay[Nmax],mom[Nmax];
```

```
 double  h,Xout,Yout,YSpline,dY;
 FILE *Fout;
 char *NameF[2]={
 "        1. F(x)=exp(x)",
 "        2. F(x)=ln(1+100*x*x)"};
 /*Связываем логический файл Fout с "физическим" - "Fout" - и открываем его на запись */
  Fout=fopen("Fout","w");
 for (NF=0; NF<2; NF++)/*Цикл функций*/
  {h=(Xmax[NF]-Xmin[NF])/N; /* шаг узлов */
   /*Вычисляем узлы и значения тестируемой функции в них*/
   for (i=0; i<=N; i++)
     {ax[i]=Xmin[NF]+i*h; ay[i]=F(NF,ax[i]);}
   /* Граничные условия */
   BoundDer(NF);
    DerB=Der[NF][0]; DerE=Der[NF][1];
   /* Нахождение моментов */
    MomSpline(N,ax,ay,mom,3,3,DerB,DerE);
   /* Заголовок выходной таблицы */
    fprintf(Fout,"%.50s\n",NameF[NF]);
   fprintf(Fout,"nXout      F          YSpline    dY\n");
   /* Для каждой точки вывода находим значения тестируемой функции,
      интерполяционного сплайна и относительного отклонения этих величин в % */
   for (i=1; i<=N; i++) /*Цикл точек вывода*/
    {Xout=(ax[i-1]+ax[i])/2;
     Yout=F(NF,Xout); YSpline=Spline(ax,ay,mom,Xout);
     dY=(Yout-YSpline)/Yout*100;
     /* вывод  строки выходной таблицы */
     fprintf(Fout,"%6.4f  %8.6f  %10.6f  %6.4f\n",
                          Xout,Yout,YSpline,dY);
    }                    /*Цикл точек вывода*/
 }                      /*Цикл функций*/
  fclose(Fout);
}
```

Язык Бейсик

```
'Программа тестирования интерполяционных сплайнов
DECLARE SUB BoundDer (NF%)
DECLARE SUB MomSpline (n%, ax#(), ay#(), mom#(), BC%(), Der#())
DECLARE FUNCTION F# (NF%, x#)
DECLARE FUNCTION Spline# (ax#(), ay#(), mom#(), x#)
DIM Nmax%: Nmax% = 50: 'Максимальное число интервалов
DIM n AS INTEGER: n = 16 'Число интервалов
DIM i AS INTEGER, NF AS INTEGER, nout AS INTEGER
DIM Xout AS DOUBLE, Yout AS DOUBLE, YSpline AS DOUBLE
DIM h AS DOUBLE, Der(1 TO 2) AS DOUBLE
DIM ax(0 TO Nmax%) AS DOUBLE, ay(0 TO Nmax%) AS DOUBLE
DIM mom(0 TO Nmax%) AS DOUBLE, BC(1 TO 2) AS INTEGER
DIM SHARED Der22(1 TO 2, 1 TO 2) AS DOUBLE
'Граничные узлы
DIM SHARED Xmin(1 TO 2) AS DOUBLE, Xmax(1 TO 2) AS DOUBLE
Xmin(1) = 0: Xmax(1) = 2: Xmin(2) = -1: Xmax(2) = 1
DIM NameF(1 TO 2) AS STRING
NameF(1) = "        1. F(x)=exp(x)"
NameF(2) = "        2. F(x)=ln(1+100*x*x)"
'Открываем файл "Fout" на запись
'и даем ему номер #1
OPEN "Fout" FOR OUTPUT AS #1
FOR NF = 1 TO 2 'Цикл функций
 h = (Xmax(NF) - Xmin(NF)) / n 'шаг узлов
 'Вычисляем узлы и значения тестируемой функции в них
 FOR i = 0 TO n
  ax(i) = Xmin(NF) + i * h: ay(i) = F#(NF, ax(i))
 NEXT i
 'Граничные условия
 CALL BoundDer(NF)
 Der(1) = Der22(NF, 1): Der(2) = Der22(NF, 2)
 BC(1) = 3: BC(2) = 3
 'Нахождение моментов
 CALL MomSpline(n, ax(), ay(), mom(), BC(), Der())
```

```
 'формат вывода
 format$ = "##.####   #.#######    ##.#######    ##.####"
 'Заголовок выходной таблицы
 PRINT #1, NameF(NF)
 PRINT #1, "Xout      Yout          YSpline         dY"
 'Для каждой точки вывода находим значения тестируемой функции, интерполяционного
 ' сплайна и относительного отклонения этих величин в %
 FOR i = 1 TO n 'Цикл точек вывода
  Xout = (ax(i - 1) + ax(i)) / 2
  Yout = F(NF, Xout)
  YSpline = Spline(ax(), ay(), mom(), Xout)
  dY = (Yout - YSpline) / Yout * 100
  'вывод  строки выходной таблицы
  PRINT #1, USING format$; Xout; Yout; YSpline; dY
 NEXT i          'Цикл точек вывода
NEXT NF          'Цикл функций
CLOSE 'Закрываем файл #1
END

SUB BoundDer (NF%)
'Производные в граничных точках
SELECT CASE NF%
 CASE 1
  Der22(1, 1) = EXP(Xmin(1)): Der22(1, 2) = EXP(Xmax(1))
 CASE 2
  Der22(2, 1) = 200 * Xmin(2) / (1 + 100 * Xmin(2) * Xmin(2))
  Der22(2, 2) = 200 * Xmax(2) / (1 + 100 * Xmax(2) * Xmax(2))
END SELECT
END SUB

FUNCTION F# (NF%, x#)
'Функции, для которых находятся интерполяционные многочлены
SELECT CASE NF%
 CASE 1
  F# = EXP(x#)
 CASE 2
  F# = LOG(1 + 100 * x# * x#)
END SELECT
END FUNCTION

SUB MomSpline (n%, ax#(), ay#(), mom#(), BC%(), Der#())
...
END SUB

FUNCTION Spline# (ax#(), ay#(), mom#(), x#)
...
END FUNCTION
```

Выполнив любую из этих программ при $n = 8$, найдем, что расхождение между значениями экспоненты и ее интерполяционного сплайна не превышает нескольких единиц пятого дробного разряда, или 0,001%. Такая точность вполне достаточна для практических целей. Напомним, однако, что при интерполяции многочленами Лагранжа расхождения начинались лишь в седьмом дробном разряде.

Перейдем к изучению интерполяционных сплайнов функции (1). Это функция $F(x) = \ln(1 + 100x^2)$ в промежутке $[-1, 1]$. Как мы помним, она оказалась "твердым орешком" для интерполяционных многочленов Лагранжа. Результат вычислений также при $n = 8$ нанесен на *рис.* 2 в виде точек — маленьких кружков. Видно, что интерполяционный сплайн вполне удовлетворительно воспроизводит ход порождающей его функции; осцилляции, характерные для интерполяционных многочленов Лагранжа, полностью отсутствуют. Правда, о количественном согласии говорить трудно: расхождение составляет от 24,5% (при значениях функции $F(x)$, близких к нулю) до 0,1%. Поэтому увеличим число интервалов вдвое: $n = 16$. Расхождение заметно уменьшается, находясь в пределах от 5,5% в области минимума до 0,0002%.

Можно подвести итоги. ***Интерполяция с помощью сплайнов является надежной, явление осцилляции здесь не наблюдается, с ростом числа узлов точность интерполяции возрастает.***

Как уже говорилось, приведенные программы (так же как и программы раздела 1) носят чисто исследовательский характер. Ведь если известно аналитическое выражение какой-либо функции, то незачем прибегать к интерполяции. Она требуется, когда мы располагаем лишь таблицей значений функции, а ее значения в промежуточных точках либо вообще неизвестны, либо алгоритм их вычисления слишком сложный и трудоемкий. Приведем следующий пример. Решение ряда прикладных задач связано с использованием температуры кипения воды, которая, как известно из курса физики, зависит от давления, возрастая с его увеличением.

Имеются подробные таблицы температуры кипения воды в зависимости от давления [4]. Они составлены на компьютере, алгоритм расчета опубликован, однако он весьма громоздкий и разобраться в нем под силу только специалисту в области термодинамики. Поэтому если нам приходится разрабатывать программу, в которой фигурирует температура кипения воды как функция давления, то проще всего включить в программу выборку из этих таблиц по возможности меньшего объема и использовать какой-нибудь метод интерполяции. Какой именно? Первое, что приходит в голову, — это линейная интерполяция — простая и хорошо знакомая. Но интересующая нас зависимость существенно нелинейная, и поэтому в программу придется включить всю таблицу, что, во-первых, трудоемко, а во-вторых, может вызвать проблемы с памятью. Выбор между многочленами Лагранжа и сплайнами делаем в пользу сплайнов из соображений надежности. В приводимых ниже программах решается "обратная" задача: определяется давление кипящей воды для заданной температуры. Диапазон задания температуры принимаем в пределах от 100°С до 200°С. Этому диапазону температур соответствует диапазон давлений (с некоторым запасом) от 1 до 16 бар: при давлении 1 бар температура кипения воды — 99,63°С, а при 16 барах — 201,37°С (напомним, что 1 бар = 10^5 Н/м^2). Поиск давления выполняется методом "деления пополам", при этом учитывается, что температура кипения с ростом давления возрастает. Приводим соответствующие программы. Обратите внимание на выбор узлов давлений: первый и последний узлы лежат вне рабочего диапазона, начальные и конечные шаги меньше основных. Это связано с тем, что, не имея информации о производных в узлах, мы вынуждены использовать на обоих концах второе граничное условие: квадратичность граничного отрезка. Указанный выбор узлов позволяет компенсировать меньшую точность этого граничного условия и является, таким образом, типичным. Как показывает специальный анализ, ошибка при нахождении давления по приведенным программам не превышает одной-двух единиц третьего десятичного разряда, что вполне достаточно для практических целей.

Язык Паскаль

```
{Определение давления, при котором кипит вода с заданной температурой}
const n=13;{Число интервалов}
type
tArr=array[0..n] of double;
const
Invitation1='Укажите температуру кипения воды ';
Invitation2='в диапазоне от 100 до 200° C ';
Invitation=Invitation1+Invitation2;
aP:tArr={массив давлений}
(0.7,1,1.4,1.9,2.5,3.3,4.3,5.5,7,8.8,11,13.5,16,17);
aTb:tArr={массив температур}
(89.96,99.63,109.32,118.62,127.43,136.82,146.25,
155.47,164.96,174.40,184.06,193.35,201.37,204.30);
procedure MomSpline(n:byte; var aX,aY,Mom;
         BegT,EndT:byte; DerB,DerE:double);
...
end;{MomSpline}
function Spline(var aX,aY,Mom; x:double):double;
...
var
mom:tArr; Tb,Tb_mid,Pleft,Pright,Pmid :double;
BEGIN
{Определяем моменты} MomSpline(n,aP,aTb,mom,2,2,0,0);
repeat  {Ввод температуры кипения}
write(Invitation); readln(Tb);
until (100<=Tb) and (Tb<=200);
{Задаем левую и правую границы поиска}
Pleft:=1; Pright:=16;
{Поиск делением пополам с учетом роста температуры кипения с давлением}
while Pright-Pleft>0.001 do begin
Pmid:=(Pright+Pleft)/2;
Tb_mid:=Spline(aP,aTb,mom,Pmid);
if Tb_mid<Tb then Pleft:=Pmid else Pright:=Pmid;
                              end;
writeln('Давление составляет ',(Pright+Pleft)/2:6:3,' бар');
END.
```

Язык Си

```
/* Определение давления, при котором кипит вода с заданной температурой */
#include<stdio.h>
#include<string.h>
#define Nmax 50 /*Максимальное число узлов*/
#define N 14 /*Число узлов*/
```

```
void MomSpline(int n, double *ax, double *ay, double *mom,
               int begT, int endT, double derB, double derE)
{
...
}
double Spline(double *ax, double *ay, double *mom, double x)
{
...
}
void main ()
{char *Invitation1="\nУкажите температуру кипения воды",
      *Invitation2="в диапазоне от 100 до 200° C";
 double aP[N]=/* массив давлений */
  {0.7,1,1.4,1.9,2.5,3.3,4.3,5.5,7,8.8,11,13.5,16,17};
 double aTb[N]=/* массив температур */
  {89.96,99.63,109.32,118.62,127.43,136.82,146.25,
    155.47,164.96,174.40,184.06,193.35,201.37,204.30};
 double mom[N],Tb,Tb_mid,Pleft,Pright,Pmid;
 /*Определяем моменты*/ MomSpline(N-1,aP,aTb,mom,2,2,0,0);
 strcat(Invitation1,Invitation2);
 do /* Ввод температуры кипения */
   {printf(Invitation1); scanf("%lf",&Tb);}
 while ( 100>Tb || Tb>200);
 /*Задаем левую и правую границы поиска*/
 Pleft=1; Pright=16;
 /*Поиск делением пополам с учетом роста температуры кипения с давлением */
 while (Pright-Pleft>0.001)
  {Pmid=(Pright+Pleft)/2;
   Tb_mid=Spline(aP,aTb,mom,Pmid);
   if (Tb_mid<Tb) Pleft=Pmid; else Pright=Pmid;
  }
 printf("\nДавление составляет %6.3f бар",(Pright+Pleft)/2);
}
```

Язык Бейсик

```
'Определение давления, при котором кипит вода с заданной температурой
DECLARE SUB MomSpline (n%, ax#(), ay#(), mom#(), BC%(), Der#())
DECLARE FUNCTION Spline# (ax#(), ay#(), mom#(), x#)
DIM n AS INTEGER: n = 13 'число интервалов
Invitation1$ = "Укажите температуру кипения воды "
Invitation2$ = "в диапазоне от 100 до 200° C "
Invitation$ = Invitation1$ + Invitation2$
DIM aP(0 TO n) AS DOUBLE, aTb(0 TO n) AS DOUBLE
DIM mom(0 TO n) AS DOUBLE, i AS INTEGER
DATA 0.7,1,1.4,1.9,2.5,3.3,4.3,5.5,7,8.8,11,13.5,16,17
FOR i = 0 TO n: READ aP(i): NEXT i
'aTb(0)-aTb(6)
DATA 89.96,99.63,109.32,118.62,127.43,136.82,146.25
FOR i = 0 TO 6: READ aTb(i): NEXT i
'aTb(7)-aTb(n)
DATA 155.47,164.96,174.40,184.06,193.35,201.37,204.30
FOR i = 7 TO n: READ aTb(i): NEXT i
DIM Tb AS DOUBLE, TbMid AS DOUBLE
DIM Pleft AS DOUBLE, Pright AS DOUBLE, Pmid AS DOUBLE
DIM BC(1 TO 2) AS INTEGER, Der(1 TO 2) AS DOUBLE
BC(1) = 2: BC(2) = 2 'Граничные условия
'Определяем моменты
CALL MomSpline(n, aP(), aTb(), mom(), BC(), Der())
DO 'Ввод температуры кипения
PRINT Invitation$: INPUT Tb
LOOP UNTIL 100 <= Tb AND Tb <= 200
'Задаем левую и правую границы поиска
Pleft = 1: Pright = 16
'Поиск делением пополам с учетом роста температуры кипения с давлением
DO WHILE Pright - Pleft > .001
 Pmid = (Pright + Pleft) / 2
 TbMid = Spline(aP(), aTb(), mom(), Pmid)
 IF TbMid < Tb THEN Pleft = Pmid ELSE Pright = Pmid
LOOP
```

```
Pmid = (Pright + Pleft) / 2
PRINT USING "& ##.### &"; "Давление составляет "; Pmid; " бар"
END

SUB MomSpline (n%, ax#(), ay#(), mom#(), BC%(), Der#())
...
END SUB

FUNCTION Spline# (ax#(), ay#(), mom#(), x#)
...
END FUNCTION
```

5. Использование сплайнов в графике

Кажется вполне естественной и очевидной идея использовать сплайны для проведения гладких кривых через заданные точки. Однако здесь возникают некоторые трудности. Так, кривая может иметь вертикальную касательную, а график кубического (да и любого) многочлена — нет. Для ряда кривых зависимость ординаты от абсциссы не выражается однозначной функцией — одной абсциссе может соответствовать несколько ординат. Такие кривые, как спирали (Архимеда или логарифмические), имеют обе указанные особенности. Наконец, кривые могут быть замкнутыми и самопересекающимися.

Все эти трудности позволяют преодолеть *параметрические сплайны*. Если для каждой точки (x_i, y_i), $i = 0, 1, ..., n$ известно значение некоторого параметра t_i, причем последовательность t_i является возрастающей, то можно построить два сплайна: *X* и *Y* — сплайны, выражающие зависимость абсциссы x и ординаты y от параметра t, т.е. получить функции $x(t)$ и $y(t)$. Пусть параметр t пробегает значения от t_0 до t_n с достаточно малым шагом. Соединяя соседние точки $(x(t), y(t))$ прямолинейным отрезком, получаем интересующую нас кривую.

Как выбрать параметр? Для параметризации кривых часто используют полярный угол и полярную систему координат. Этот способ может подойти и для параметрических сплайнов, но здесь надо правильно выбрать положение полюса так, чтобы полярные углы увеличивались при переходе к следующей заданной точке кривой, что иногда затруднительно. В принципе параметр может быть выбран произвольно, достаточно каждой заданной точке сопоставить какое-то его значение. Однако интуиция и вычислительные эксперименты подсказывают, что разность значений параметра в соседних заданных точках должна быть пропорциональна расстоянию между этими точками. Мы приходим к параметру, который имеет довольно длинное название — *накопленная длина хорды*. В узлах значение этого параметра – сумма длин хорд всех предыдущих интервалов. Пусть d_i — длина хорды i–го интервала:

$$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}\,.$$

Тогда параметр в нулевой точке $t_0 = 0$, а при $i > 0$

$$t_i = t_{i-1} + d_i, \qquad i = 1, ..., n.$$

Достоинством такого выбора параметра является его универсальность — возрастание параметра с ростом номера точки здесь гарантировано.

Посмотрим теперь, как выбирать граничные условия для параметрических сплайнов при выборе в качестве параметра накопленной длины хорды.

Пусть φ – угол наклона граничной хорды (первой или последней). Тогда производная от x по параметру t в граничной точке выражается как

$$\frac{dx}{dt} = \cos\varphi.$$

Это означает, что для *X*-сплайна производные в граничных точках известны и можно использовать граничное условие 3 для обоих концов. Допустим теперь, что в какой-либо граничной точке кривой известен тангенс угла наклона касательной g. Поскольку

$$g = \frac{dy}{dx},$$

то с учетом предыдущего выражения производная от y по параметру t имеет вид:

$$\frac{dy}{dt} = g\cos\varphi$$

— и для *Y*-сплайна в соответствующей граничной точке также можно использовать граничное условие 3. На практике встречаются случаи, когда начальный или конечный участок кривой – прямолинейный отрезок и, значит, совпадает с хордой. Производная от y по параметру t в этом случае есть синус угла φ:

$$\frac{dy}{dt} = \sin\varphi$$

— и опять-таки можно использовать граничное условие 3. В остальных случаях для *Y*-сплайна придется использовать граничное условие 2 – квадратичность граничного участка.

Сейчас мы приведем тексты процедуры `Curve`, которая с помощью параметрического сплайна строит на экране кривую. Прежде чем перейти к исходным данным этой процедуры, коснемся вопроса о системах координат, применяемых в графических программах. Как известно, положение пикселя на экране характеризуется экранной системой координат: начало координат расположено в левом верхнем углу экрана, ось *X* направлена вправо, ось *Y* — сверху вниз. Мы будем также использовать центральную систему координат: начало координат расположено в центре экрана, ось *X* направлена вправо, ось *Y* – вверх (см. [5]).

Вернемся к исходным данным процедуры `Curve`. Они включают:

– Число интервалов `N`.

– Массивы координат заданных точек `aX,aY`. При этом имеется в виду произвольная система координат, в которой работает пользователь.

– Центральные координаты точки привязки `X0,Y0`. Под "точкой привязки" подразумевается точка на экране, соответствующая началу пользовательской системы координат.

– Масштаб `M`, т.е. количество пикселей в единице длины пользователя.

– Коды граничных условий `Tb,Te`. Подчеркнем во избежание недоразумений, что эти коды отличаются от кодов граничных условий, используемых в процедуре `MomSpline`. (Там они имеют обозначения `BegT, EndT.`) Здесь значения кодов следующие:

1. Граничный участок является прямолинейным отрезком.
2. Граничный участок – параболический (квадратичный).
3. Касательная в граничной точке горизонтальна.

Читатель может возразить, что, как показано выше, можно было бы охватить и случай наклонной касательной, и будет совершенно прав. Просто на практике при проведении кривых через точки о значениях наклона касательных (т.е. о производных) обычно ничего не известно. Исключение составляют замкнутые кривые, у которых начальная (она же конечная) точка самая верхняя или самая нижняя; тогда касательная в ней заведомо горизонтальна. Поэтому мы и ограничились случаем горизонтальной касательной. При желании можно модифицировать приводимые процедуры так, чтобы учесть общий случай.

Язык Паскаль

```
procedure Curve(var Ax,aY:tArr; X0,Y0:integer;
                  M:double; N:byte; Tb,Te:byte);
{С помощью параметрического сплайна строится на экране кривая по входным величинам:
aX,aY - массивы координат узловых точек;
X0,Y0 - центральные координаты точки привязки;
M - масштаб; N - число интервалов;
Tb,Te - коды граничных условий}
const NStep=10;{Число шагов на интервале}
var
aChord,aAcCh,MomX,MomY:tArr;
DerBy,DerEy,X,Y,t,DerBx,DerEx:double;
Xb,Xe,Yb,Ye:integer;
BegTy,EndTy,j,i:byte;
begin
aAcCh[0]:=0;{Накопленная хорда в начальной точке}
{Вычисление массивов хорд aChord и накопленных хорд aAcCh}
for i:=1 to N do begin
aChord[i]:=sqrt(sqr(aX[i]-aX[i-1])+sqr(aY[i]-aY[i-1]));
aAcCh[i]:=aAcCh[i-1]+aChord[i];
                    end;
{Производные x по параметру в граничных точках - косинусы угла наклона хорды}
DerBx:=(aX[1]-aX[0])/aChord[1];
DerEx:=(aX[N]-aX[N-1])/aChord[N];
Case Tb of
1: begin {Нач. участок - прямая линия}
    BegTy:=3;
     DerBy:=(aY[1]-aY[0])/aChord[1];{sin угла наклона хорды}
    end; {Нач. участок - прямая линия}
 2: BegTy:=2;{Параболический нач. участок}
 3: begin{Горизонтальная касательная}
     BegTy:=3; DerBy:=0
     end;{Горизонтальная касательная}
end;{Case}
Case Te of
1: begin{Кон. участок - прямая линия}
    EndTy:=3;
```

```
     DerEy:=(aY[N]-aY[N-1])/aChord[N];{sin угла наклона хорды}
     end;{Кон. участок - прямая линия}
 2: EndTy:=2;{Параболический кон. участок}
 3: begin{Горизонтальная касательная в кон. точке}
     EndTy:=3; DerEy:=0
     end;{Горизонтальная касательная в кон. точке}
end;{Case}
{Моменты сплайна абсцисс}
MomSpline(N,aAcCh,aX,MomX,3,3,DerBx,DerEx);
{Моменты сплайна ординат}
MomSpline(N,aAcCh,aY,MomY,BegTy,EndTy,DerBy,DerEy);
{Экранные координаты начальной точки}
Xb:=Xscr(X0+round(M*aX[0]));
Yb:=Yscr(Y0+round(M*aY[0]));
for i:=1 to N do {Цикл участков кривой}
 for j:=1 to Nstep do begin{Цикл шагов интервала}
  t:=aAcCh[i-1]+aChord[i]/Nstep*j;{Параметр}
  X:=Spline(aAcCh,aX,MomX,t);{Координаты очередной}
  Y:=Spline(aAcCh,aY,MomY,t);{точки кривой         }
  Xe:=Xscr(X0+round(M*X));{Экранные координаты    }
  Ye:=Yscr(Y0+round(M*Y));{очередной точки кривой}
  line(Xb,Yb,Xe,Ye);{Соединяем предыдущую и очередную точки}
  Xb:=Xe; Yb:=Ye; {Очередная точка становится предыдущей}
                    end;{Цикл шагов интервала}
end;{Curve}
```

Язык Си

```
void Curve (int X0,int Y0, double M, int N, int Tb, int Te)
/* С помощью параметрического сплайна строится на экране кривая по входным величинам:
   X0,Y0 - центральные координаты начала координат;
   M - масштаб; N - число интервалов;
   aX,AY - массивы узлов и ординат;
   Tb,Te - коды граничных условий.   */
{double  aChord[Nmax],aAcCh[Nmax],MomX[Nmax],MomY[Nmax];
 double  X,Y,t,DerBx,DerEx,DerBy,DerEy,dx,dy;
 int  NStep=10,Xb,Xe,Yb,Ye,i,j,BegTy,EndTy;
 aAcCh[0]=0;/*Накопленная хорда в нач. точке*/
 /*Вычисление массивов хорд aChord  и накопленных хорд aAcCh*/
 for (i=1; i<=N; i++)
  {dx=aX[i]-aX[i-1]; dy=aY[i]-aY[i-1];
   aChord[i]=sqrt(dx*dx+dy*dy);
   aAcCh[i]=aAcCh[i-1]+aChord[i];
  }
 /*Производные x по параметру в граничных точках - косинусы угла наклона хорды */
 DerBx=(aX[1]-aX[0])/aChord[1];
 DerEx=(aX[N]-aX[N-1])/aChord[N];
 switch(Tb)
 {case 1:/* Нач. участок - прямая линия*/
   BegTy=3;
   DerBy=(aY[1]-aY[0])/aChord[1];/*sin угла наклона хорды*/
   break;
  case 2: BegTy=2; break;/*Параболический нач. участок*/
  case 3:/*Горизонтальная касательная в нач. точке*/
   BegTy=3; DerBy=0; break;
 }
 switch(Te)
 {case 1:/* Кон. участок - прямая линия*/
   EndTy=3;
   DerEy=(aY[N]-aY[N-1])/aChord[N];/*sin угла наклона хорды*/
   break;
  case 2: EndTy=2; break;/*Параболический кон. участок*/
  case 3:/*Горизонтальная касательная в кон. точке*/
   EndTy=3; DerEy=0; break;
 }
 /*Моменты сплайна абсцисс*/
 MomSpline(N,aAcCh,aX,MomX,3,3,DerBx,DerEx);
 /*Моменты сплайна ординат*/
 MomSpline(N,aAcCh,aY,MomY,BegTy,EndTy,DerBy,DerEy);
 /*Экранные координаты начальной точки*/
```

```
  Xb=Xscr(X0+round(M*aX[0]));
  Yb=Yscr(Y0+round(M*aY[0]));
 for (i=1; i<=N; i++) /*Цикл участков кривой*/
 for (j=1; j<=NStep; j++)/*Цикл шагов интервала*/
   {t=aAcCh[i-1]+aChord[i]/NStep*j;/*Параметр*/
   X=Spline(aAcCh,aX,MomX,t);/*Координаты очередной*/
   Y=Spline(aAcCh,aY,MomY,t);/*точки кривой*/
   Xe=Xscr(X0+round(M*X));/*Экранные координаты*/
   Ye=Yscr(Y0+round(M*Y));/*очередной точки кривой*/
   line(Xb,Yb,Xe,Ye);/*Соединяем предыдущую и очередную точки*/
   Xb=Xe; Yb=Ye; /*Очередная точка становится предыдущей*/
   }                        /*Цикл шагов интервала*/
}
```

Язык Бейсик

```
SUB Curve (aX#(), aY#(), M#, Param%())
'С помощью параметрического сплайна строится на экране кривая по входным величинам:
'aX,AY - координаты узловых точек
'M - масштаб;
'Элементами массива Param являются:
'X0,Y0 - центральные координаты точки привязки;
'N - число интервалов;
'Tb,Te - коды граничных условий; ColorC - цвет кривой.
DIM X0%, Y0%, n%, Tb%, Te%, ColorC%
Nstep% = 10 'Число шагов на интервале
X0% = Param%(1): Y0% = Param%(2): n% = Param%(3)
Tb% = Param%(4): Te% = Param%(5): ColorC% = Param%(6)
DIM aChord#(1 TO n%)
DIM aAcCh#(0 TO n%), MomX#(0 TO n%), MomY#(0 TO n%)
DIM DerX#(1 TO 2), DerY#(1 TO 2), X#, Y#, t#, dx#, dy#
DIM BCx%(1 TO 2), BCy%(1 TO 2), Xb%, Xe%, Yb%, Ye%, j%, i%
aAcCh#(0) = 0 'Накопленная хорда в нач. точке
'Вычисление массивов хорд aChord  и накопленных хорд aAcCh
FOR i% = 1 TO n%
 dx# = aX#(i%) - aX#(i% - 1): dy# = aY#(i%) - aY#(i% - 1)
 aChord#(i%) = SQR(dx# * dx# + dy# * dy#)
 aAcCh#(i%) = aAcCh#(i% - 1) + aChord#(i%)
NEXT i%
'Производные x по параметру в граничных точках - косинусы угла наклона хорды
DerX#(1) = (aX#(1) - aX#(0)) / aChord#(1)
DerX#(2) = (aX#(n%) - aX#(n% - 1)) / aChord#(n%)
BCx%(1) = 3: BCx%(2) = 3 'гран. условия X-сплайна
SELECT CASE Tb%
 CASE 1 'Нач. участок - прямая линия
  BCy%(1) = 3
  DerY#(1) = (aY#(1) - aY#(0)) / aChord#(1)
 CASE 2 'Параболический нач. участок
  BCy%(1) = 2
 CASE 3 'Горизонтальная касательная
  BCy%(1) = 3: DerY#(1) = 0
END SELECT
SELECT CASE Te%
 CASE 1 'Кон. участок - прямая линия
  BCy%(2) = 3
  DerY#(2) = (aY#(n%) - aY#(n% - 1)) / aChord#(n%)
 CASE 2 'Параболический кон. участок
  BCy%(2) = 2
 CASE 3 'Горизонтальная касательная
  BCy%(2) = 3: DerY#(2) = 0
END SELECT
'Моменты сплайна абсцисс
CALL MomSpline(n%, aAcCh#(), aX#(), MomX#(), BCx%(), DerX#())
'Моменты сплайна ординат
CALL MomSpline(n%, aAcCh#(), aY#(), MomY#(), BCy%(), DerY#())
'Экранные координаты начальной точки
Xb% = Xscr%(X0% + Round%(M# * aX#(0)))
Yb% = Yscr%(Y0% + Round%(M# * aY#(0)))
FOR i% = 1 TO n% 'Цикл участков кривой
```

```
 FOR j% = 1 TO Nstep% 'Цикл шагов интервала
  t# = aAcCh#(i% - 1) + aChord#(i%) / Nstep% * j% 'Параметр
  'Координаты очередной точки кривой
  X# = Spline(aAcCh#(), aX#(), MomX#(), t#)
  Y# = Spline(aAcCh#(), aY#(), MomY#(), t#)
  'Экранные координаты очередной точки кривой
  Xe% = Xscr%(X0% + Round%(M# * X#))
  Ye% = Yscr%(Y0% + Round%(M# * Y#))
  'Соединяем предыдущую и очередную точки
  LINE (Xb%, Yb%)-(Xe%, Ye%), ColorC%
  'Очередная точка становится предыдущей
  Xb% = Xe%: Yb% = Ye%
 NEXT j%               'Цикл шагов интервала
NEXT i%           'Цикл участков кривой
END SUB
```

Чтобы проверить работу этой процедуры, составим программу, изображающую с помощью параметрических сплайнов две кривые: эллипс и спираль Архимеда. Используя известные уравнения этих кривых, вычислим координаты массива точек, после чего, обращаясь к процедуре `Curve`, проведем на экране сами кривые. На экран выведем и вычисленный массив точек — это поможет оценить качество изображения кривых. Какие граничные условия мы используем? В качестве первой и последней точки эллипса используем его нижнюю точку, где касательная горизонтальна. Поэтому оба граничных условия эллипса имеют код 3. А для спирали Архимеда оба граничных условия эллипса имеют код 2 — ничего другого о граничных условиях мы сказать не можем.

Приводим программы:

Язык Паскаль

```
{изображение эллипса и спирали Архимеда с помощью параметрического сплайна}
uses GRAPH,CRT;
const
Nmax=50;{Максимальное число интервалов}
PATH='';{файлы *.BGI находятся в рабочем каталоге}
type
tArr=array[0..Nmax] of double;
tArr2=array[1..2] of byte;
const {Параметры для каждой кривой}
n:tArr2=(12,21);{число интервалов}
Tb:tArr2=(3,2);{Код начального условия}
Te:tArr2=(3,2);{Код конечного условия}
M:array[1..2] of double=(1.5,1);{Масштаб}
Var
Xc,Yc,X0,Y0,W,H,gd,gm:integer;
Nc,i:byte;
aX,aY:tArr;
procedure Points(Ncurve:byte; n:byte; var aX,aY:tarr);
{Для эллипса и спирали Архимеда находятся массивы узлов и ординат aX,aY}
var i:byte; a,b,Angle0,Angle:double;
begin
Case Ncurve of
1: begin{эллипс}
  a:=150; b:=100;{полуоси}
Angle0:=270;
for i:=0 to n do begin
 Angle:=Angle0+360/n*i;
 aX[i]:=a*cos(Angle/180*Pi);
 aY[i]:=b*sin(Angle/180*Pi);
                  end;
    end;{эллипс}
2: {спираль Архимеда}
for i:=0 to n do begin
Angle:=630/n*i;
aX[i]:=Angle/4*cos(Angle/180*Pi);
aY[i]:=Angle/4*sin(Angle/180*Pi);
                  end;
end;{Case}
end;{Points}
procedure MomSpline(n:byte; var aX,aY,Mom:tArr;
...
```

```
end;{MomSpline}
function Spline(var aX,aY,Mom:tArr; x:double):double;
...
end;{Spline}
{Переход от центральных координат к экранным}
procedure WH(var W,H:integer);{ширина и высота экрана}
begin
{Функции GetMaxX и GetMaxY возвращают максимальные значения соответствующих экранных координат}
W:=(GetMaxX+1) ;{ширина}
H:=(GetMaxY+1) ;{высота}
end;
Function Xscr(X:Integer):Integer;
Begin Xscr:=X + W div 2 end;
Function Yscr(Y:Integer):Integer;
Begin Yscr:=H div 2 - Y end;
procedure Curve(var Ax,aY:tArr; X0,Y0:integer;
                M:double; N:byte; Tb,Te:byte);
...
end;{Curve}
BEGIN
X0:=0; Y0:=0; gd:=DETECT;
for Nc:=1 to 2 do begin{цикл кривых}
 initgraph(gd,gm,path);{переход в графический режим}
 WH(W,H); {определяем ширину и высоту экрана}
 setcolor(white);{Цвет линии - белый}
 {Находим узловые точки очередной кривой}
 Points(Nc,n[Nc],aX,aY);
 {Изображаем кривую}
 Curve(aX,aY,X0,Y0,M[Nc],n[Nc],Tb[Nc],Te[Nc]);
 setcolor(LightRed);{Цвет линии - розовый}
 setfillstyle(SolidFiLL,LightRed);{Закраска - розовая}
 {В узловых точках рисуем розовые маленькие окружности}
 for i:=0 to n[Nc] do begin
  Xc:=Xscr(X0+round(M[Nc]*aX[i]));
  Yc:=Yscr(Y0+round(M[Nc]*aY[i]));
  pieslice(Xc,Yc,0,360,3);{рисование и закраска сектора}
                          end;
 readln;{Выход - нажатием клавиши Enter}
 CloseGraph;{Переход в текстовый режим}
                  end;{цикл кривых}
END.
```

Язык Си

```
/*изображение эллипса и спирали Архимеда с помощью параметрического сплайна*/
#include<graphics.h>
#include<math.h>
#include<conio.h>
#define PI 3.1415926
#define Nmax 50 /* максимальное число узлов */
#define PATH "" /*файлы *.BGI находятся в рабочем каталоге*/
int W,H;
double aX[Nmax],aY[Nmax];
int round(double f) /* Округление до ближайшего целого */
{return (int)floor(f+0.5);}
/* Переход от центральных координат к экранным */
void WH(int *W, int *H) /* ширина и высота экрана */
/* Функции getmaxx и getmaxy возвращают максимальные значения соответствующих экранных координат */
{*W=getmaxx()+1; *H=getmaxy()+1;}
int Xscr(int x){ return x+W/2;}
int Yscr(int y){ return H/2-y;}
void MomSpline(int n, double *ax, double *ay, double *mom,
               int begT, int endT, double derB, double derE)
{
...
}
double Spline(double *ax, double *ay, double *mom, double x)
{
...
}
```

```
void Points(int Ncurve, int n, double *aX, double *aY)
/* Для эллипса и спирали Архимеда находятся массивы узлов и ординат aX,aY */
{int i; double a,b,Angle0,Angle;
 switch(Ncurve)
 {case 0: /* эллипс */
   a=150; b=100; /* полуоси */
   Angle0=270;
   for (i=0; i<=n; i++)
    {Angle=Angle0+360.0/n*i;
     aX[i]=a*cos(Angle/180*PI);
     aY[i]=b*sin(Angle/180*PI);
    }
     break;
  case 1: /* Спираль Архимеда */
   for (i=0; i<=n; i++)
    {Angle=630.0/n*i;
     aX[i]=Angle/4*cos(Angle/180*PI);
     aY[i]=Angle/4*sin(Angle/180*PI);
    }
     break;
  }
}
void Curve (int X0,int Y0, double M, int N, int Tb, int Te)
{
...
}
void main()
{int  Xc,Yc,X0=0,Y0=0,gd,gm,Nc,i;
 /*Параметры для каждой кривой*/
 double M[2]={1.5,1};/*Масштаб*/
 int n[2]={12,21};/*число интервалов*/
 int Tb[2]={3,2};/*Код начального условия*/
 int Te[2]={3,2};/*Код конечного условия*/
 gd=DETECT;
 for (Nc=0; Nc<2; Nc++)/*цикл кривых*/
 {initgraph(&gd,&gm,PATH); /*Переход в графический режим*/
  WH(&W,&H); /*определение ширины и высоты экрана*/
  setcolor(WHITE);/*Цвет линии - белый*/
  /*Находим характеристики очередной кривой*/
  Points(Nc,n[Nc],aX,aY);
  /*Изображаем кривую*/
   Curve(X0,Y0,M[Nc],n[Nc],Tb[Nc],Te[Nc]);
  setcolor(LIGHTRED);/*Цвет линии - розовый*/
  setfillstyle(SOLID_FILL,LIGHTRED);/*закраска - розовая*/
  /*В заданных точках рисуем розовые маленькие окружности*/
  for (i=0; i<=n[Nc]; i++)
   {Xc=Xscr(X0+round(M[Nc]*aX[i]));
    Yc=Yscr(Y0+round(M[Nc]*aY[i]));
    pieslice(Xc,Yc,0,360,3);/*рисование и закраска сектора*/
  }
  getch(); /* Выход - нажатием любой клавиши */
  closegraph(); /* Переход в текстовый режим */
 }                      /*цикл кривых*/
}
```

Язык Бейсик

```
'изображение эллипса и спирали Архимеда с помощью параметрического сплайна
DECLARE SUB Curve (aX#(), aY#(), M#, Param%())
DECLARE SUB MomSpline (n%, aX#(), aY#(), Mom#(), BC%(), Der#())
DECLARE SUB Points (Ncurve%, n%, aX#(), aY#())
DECLARE FUNCTION Round% (A#)
DECLARE FUNCTION Spline# (aX#(), aY#(), Mom#(), X#)
DECLARE FUNCTION Xscr% (X%)
DECLARE FUNCTION Yscr% (Y%)
DIM SHARED Pi AS DOUBLE: Pi = 3.1415926#
DIM SHARED Ws AS INTEGER, Hs AS INTEGER
Ws = 640: Hs = 480 'Ширина и высота экрана_
Nmax% = 50 'Максимальное число интервалов
DIM aX(0 TO Nmax%)  AS DOUBLE, aY(0 TO Nmax%) AS DOUBLE
```

```
DIM n%(1 TO 2): n%(1) = 12: n%(2) = 21 'число интервалов
  'Коды граничных условий
DIM Tb%(1 TO 2): Tb%(1) = 3: Tb%(2) = 2
DIM Te%(1 TO 2): Te%(1) = 3: Te%(2) = 2
'Масштаб
DIM M#(1 TO 2): M#(1) = 1.5: M#(2) = 1
X0% = 0: Y0% = 0'центральные координаты точки привязки
White% = 15 'белый цвет
LightRed% = 12 'розовый цвет
DIM par%(1 TO 2, 1 TO 6)
'Первая строка двумерного массива par соответствует
'массиву параметров эллипса, вторая - спирали Архимеда
par%(1, 1) = X0%: par%(1, 2) = Y0%: par%(1, 3) = n%(1)
par%(1, 4) = Tb%(1): par%(1, 5) = Te%(1): par%(1, 6) = White%
par%(2, 1) = X0%: par%(2, 2) = Y0%: par%(2, 3) = n%(2)
par%(2, 4) = Tb%(2): par%(2, 5) = Te%(2): par%(2, 6) = White%
DIM Param%(1 TO 6)
DIM Xc AS INTEGER, Yc AS INTEGER, Nc AS INTEGER, i AS INTEGER
FOR Nc = 1 TO 2 'цикл кривых
SCREEN 12 'Переход в графический режим для монитора VGA
'Находим узловые точки очередной кривой
CALL Points(Nc, n%(Nc), aX(), aY())
'Изображаем кривую
FOR i = 1 TO 6: Param%(i) = par%(Nc, i): NEXT i
CALL Curve(aX(), aY(), M#(Nc), Param%())
'В узловых точках рисуем розовые маленькие окружности
FOR i = 0 TO n%(Nc)
Xc = Xscr%(X0% + Round%(M#(Nc) * aX(i)))
Yc = Yscr%(Y0% + Round%(M#(Nc) * aY(i)))
CIRCLE (Xc, Yc), 3, LightRed% 'рисование окружности
PAINT (Xc, Yc), LightRed%, LightRed% 'закраска окружности
NEXT i
'Продолжение - нажатием клавиши Enter
DO: LOOP UNTIL INKEY$ = CHR$(13)
SCREEN 0 'Переход в текстовый режим
NEXT Nc          'цикл кривых
END

SUB Curve (aX#(), aY#(), M#, Param%())
...
END SUB

SUB MomSpline (n%, aX#(), aY#(), Mom#(), BC%(), Der#())
...
END SUB

SUB Points (Ncurve%, np%, aX#(), aY#())
'Для эллипса и спирали Архимеда находятся массивы узлов и ординат aX,aY
DIM i%, A#, b#, Angle0#, Angle#
SELECT CASE Ncurve%
CASE 1 'эллипс
 A# = 150: b# = 100 'полуоси
 Angle0# = 270
 FOR i% = 0 TO np%
  Angle# = Angle0# + 360! / np% * i%
  aX#(i%) = A# * COS(Angle# / 180 * Pi)
  aY#(i%) = b# * SIN(Angle# / 180 * Pi)
 NEXT i%
CASE 2 'спираль Архимеда
 FOR i% = 0 TO np%
  Angle# = 630! / np% * i%
  aX#(i%) = Angle# / 4 * COS(Angle# / 180 * Pi)
  aY#(i%) = Angle# / 4 * SIN(Angle# / 180 * Pi)
 NEXT i%
END SELECT
END SUB

FUNCTION Round% (A#)
'округление до ближайшего целого
Round% = INT(A# + .5)
END FUNCTION
```

```
FUNCTION Spline# (aX#(), aY#(), Mom#(), X#)
...
END FUNCTION

FUNCTION Xscr% (X%)
'Переход от центральной координаты X к экранной
Xscr% = X% + Ws / 2
END FUNCTION

FUNCTION Yscr% (Y%)
'Переход от центральной координаты Y к экранной
Yscr% = Hs / 2 - Y%
END FUNCTION
```

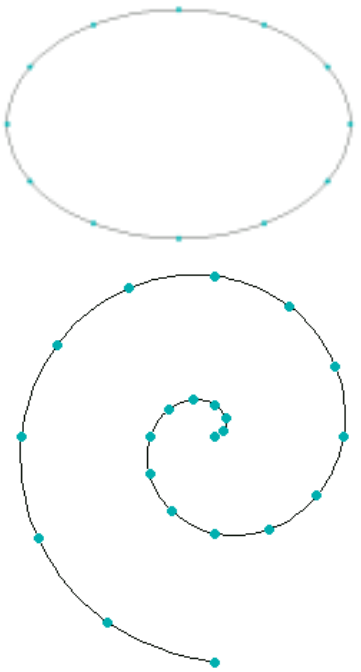


Рис. 4

На *рис.* 4 представлены эллипс и спираль Архимеда, полученные с помощью этой программы. Как видим, параметрические сплайны работают вполне удовлетворительно. Конечно, располагая уравнениями эллипса и спирали Архимеда, мы вполне могли построить эти кривые, не используя параметрические сплайны.

Сейчас мы перейдем к более интересной задаче. Требуется воспроизвести на экране небольшой рукописный текст на русском языке — слово или фразу. Предполагается, что текст первоначально написан на миллиметровке. Каждую букву разбиваем на элементы — гладкие кривые: овалы, различные “крючки” и т.д. так, как нас когда-то учили на уроках чистописания. Для каждого элемента выбираем несколько характерных точек, записываем их координаты и, кроме того, указываем граничные условия. Так, для овала (элемент рукописных букв “о”, “а”, “р”, “д” и т.д.) удобно выбрать в качестве первой (и последней) точки нижнюю или верхнюю. Тогда касательная в этой точке горизонтальна и можно использовать в начале и конце граничное условие 3. Крючок, загнутый внизу (элемент рукописных букв “а”, “и”, “ш”, “м” и т.д.), вверху имеет прямолинейный участок. Поэтому если начать с верхней точки, то код начального условия — 1, а конечного — 2. Учитывая, что объем этой информации довольно велик, помещаем ее в текстовый файл. Для элемента отводим строку файла целиком. Вначале указываем начальное условие в виде `Tb=...`, например, для овала `Tb=3`. Далее (пропустив один пробел) записываем координаты точек элемента, разделяя их также одним пробелом. После ординаты последней точки пропускаем пробел и указываем конечное условие в виде `Te=...`, например, для овала `Te=3`. Параметры этой задачи:

а) Центральные координаты точки привязки — `X0,Y0`. Точка привязки соответствует началу координат на миллиметровке.

б) Масштаб `M` — число пикселей в одном миллиметре.

Алгоритм решения этой задачи заключается в следующем. Последовательно читаются строки файла. Для каждой прочитанной строки выполняется последовательность действий:

— Находится код начального условия `Tb`, он содержится в 4-й позиции строки (для Си — в 3-й), а также код конечного условия `Te`, который содержится в последней позиции строки.

— Счетчику точек `n` присваивается значение `−1`, индекс первой позиции записи абсциссы точки `Ind` получает значение `6` (для Си — `5`), а индекс подстроки `"Te="`, обозначаемый `IndE`, — `(L-3)`, где `L` — длина строки.

— Выполняется цикл чтения координат точек элемента: наращивается счетчик точек `n`; находится позиция пробела, обрамляющего справа запись абсциссы точки `pB`; выделяется подстрока указанной записи и переводится в `aX[n]`; увеличивая на единицу `pB`, получаем индекс первой позиции записи ординаты точки; действуя аналогичным образом, находим `aY[n]` и индекс первой позиции записи абсциссы следующей точки `Ind`; если `Ind=IndE`, то цикл закончен.

— На экране с помощью процедуры `Curve` изображается элемент.

Приводим программы.

Язык Паскаль

```
{Изображение "рукописного" текста с помощью параметрического сплайна}
uses GRAPH,CRT;
const
Nmax=50;{Максимальное число интервалов}
PATH='';{Файлы *.BGI находятся в рабочем каталоге}
type tArr=array[0..Nmax] of double;
Var
Fin:text;
```

```
Number,Sin:string;
n,X0,Y0,W,H,gd,gm,kode:integer;
M:double;
Tb,Te,L,i,j,Ind,IndE,pB:byte;
aX,aY:tArr;
procedure MomSpline(n:byte; var aX,aY,Mom:tArr;
             begT,EndT:byte; DerB,DerE:double);
...
end;{MomSpline}
function Spline(var aX,aY,Mom:tArr; x:double):double;
...
end;{Spline}
procedure WH(var W,H:integer);{ширина и высота экрана}
begin
{Функции GetMaxX и GetMaxY возвращают максимальные значения соответствующих экранных
координат}
W:=(GetMaxX+1);{Ширина}
H:=(GetMaxY+1);{Высота}
end;
Function Xscr(X:Integer):Integer;
{Переход от центральной координаты X к экранной}
Begin Xscr:=X + W div 2 end;
Function Yscr(Y:Integer):Integer;
{Переход от центральной координаты Y к экранной}
Begin Yscr:=H div 2 - Y end;
procedure Curve(X0,Y0:integer; var aX,AY:tArr;
                  M:real; N:byte; Tb,Te:byte);
...
end;{Curve}
function PosBlank(S:string; Beg:byte):byte;
{Определение позиции первого пробела, находящегося в строке S правее позиции Beg,
считая, что такой пробел существует}
var i:byte;
begin
i:=Beg+1; while S[i]<>' ' do inc(i); PosBlank:=i
end;
BEGIN
{Связываем логический файл Fin с "физическим" файлом "word" и открываем его на чтение}
assign(Fin,'word'); reset(Fin);
write('Укажите X0  в центральных координатах ');
readln(X0);
write('Укажите Y0  в центральных координатах ');
readln(Y0);
write('Укажите масштаб'); readln(M);
gd:=DETECT;
initgraph(gd,gm,path);{Переход в графический режим}
WH(W,H); {Определяем ширину и высоту экрана}
setcolor(white);{Цвет линии - белый}
repeat          {Цикл чтения файла}
 readln(Fin,Sin);
 L:=length(Sin);{Длина строки Sin}
 Tb:=ord(Sin[4])-ord('0');{Код начала}
 Te:=ord(Sin[L])-ord('0');{Код конца}
 Ind:=6; IndE:=L-3; n:=-1;
 repeat          {Цикл чтения координат}
  inc(n);
  for j:=1 to 2 do begin{Координаты очередной точки}
  pB:=PosBlank(Sin,Ind);
  {Number - строковая запись очередной координаты}
  Number:=copy(Sin,Ind,pB-Ind);
  {Перевод строки Number в элемент массива}
  if j=1 then val(Number,aX[n],kode)
  else        val(Number,aY[n],kode);
  Ind:=pB+1;
                    end;{Координаты очередной точки}
 until Ind=IndE;{Цикл чтения координат}
 Curve(X0,Y0,aX,AY,M,n,Tb,Te);{Изображаем кривую}
until EoF(Fin);{Цикл чтения файла}
readln;{Выход - нажатием клавиши Enter}
CloseGraph;{Переход в текстовый режим}
close(Fin);{Закрываем файл Fin}
END.
```

Язык Си

```
/* изображение "рукописного" текста с помощью параметрического сплайна */
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<string.h>
#define Nmax 50 /* Максимальное число интервалов */
#define PATH "" /*Файлы *.BGI находятся в рабочем каталоге*/
int W,H;
int round(double f) /* Округление до ближайшего целого */
{return  (int)floor(f+0.5);}
void WH(int *W, int *H) /* ширина и высота экрана */
/* Функции getmaxx и getmaxy возвращают максимальные значения соответствующих экранных
координат */
{*W=getmaxx()+1;  *H=getmaxy()+1;}
/*Переход от центральных координат X и Y к экранной*/
int Xscr(int x){ return x+W/2;}
int Yscr(int y){ return H/2-y;}
void MomSpline(int n, double *ax, double *ay, double *mom,
{
...
}
double Spline(double *ax, double *ay, double *mom, double x)
{
...
}
void Curve (int X0,int Y0, double *aX, double *aY,
                  double M, int N, int Tb, int Te)
{
...
}
int PosBlank(char *S,int Beg)
/* Определение позиции первого пробела, находящегося в строке S правее позиции Beg,
считая, что такой пробел существует */
{int i;
 i=Beg+1; while(S[i] != ' ') i++; return i;
}
void Substr(char S[],int Start, int L, char *Sub)
/* Выделение из строки S подстроки Sub длиной L с начальной позиции Start */
{int i;
 for (i=0; i<L; i++) Sub[i]=S[Start+i]; Sub[L]='\0';
}
void main()
{char  Number[20],Sin[100];
  int  n,X0,Y0,gd,gm,kode,Tb,Te,L,i,j,Ind,IndE,pB;
 double aX[Nmax],aY[Nmax];
 double M;
 FILE *Fin;
 /* Связываем логический файл Fin с "физическим" файлом "word" и открываем его на чтение */
  Fin=fopen("word","r");
 printf("\nУкажите X0  в центральных координатах ");
 scanf("%d",&X0);
 printf("\nУкажите Y0  в центральных координатах ");
 scanf("%d”,&Y0);
 printf("\nУкажите масштаб "); scanf("%lf",&M);
 gd=DETECT;
 initgraph(&gd,&gm,PATH); /* Переход в графический режим */
 WH(&W,&H); /* Определение ширины и высоты экрана */
 setcolor(WHITE) ;/*Цвет линии - белый*/
 while (feof(Fin) == 0) /*Цикл чтения файла*/
  {fgets(Sin,100,Fin);/* Читаем очередную строку файла Fin*/
   L=strlen(Sin);/* Длина строки Sin */
   /* Если строка - не последняя, то в ее конце
      находится символ '\n', от которого надо избавиться */
   if (feof(Fin) == 0) {L—; Sin[L]='\0';}
   Tb=Sin[3]-'0'; Te=Sin[L-1]-'0';/*Коды начала и конца*/
   Ind=5; IndE=L-4; n=-1;
   do                   /* Цикл чтения координат */
```

```
    {n++;
     for (j=1; j<=2; j++)/*Координаты очередной точки*/
      {pB=PosBlank(Sin,Ind);
       /*Number - строковая запись очередной координаты*/
       Substr(Sin,Ind,pB-Ind,Number);
       /*Перевод строки Number в элемент массива*/
       if (j==1)  aX[n]=atof(Number);
       else       aY[n]=atof(Number);
       Ind=pB+1;
      }                  /*Координаты очередной точки*/
    }
   while (Ind != IndE); /* Цикл чтения координат */
    Curve(X0,Y0,aX,aY,M,n,Tb,Te);/*Изображаем кривую*/
  }                      /*Цикл чтения файла*/
 getch(); /* Выход - нажатием любой клавиши */
 closegraph(); /* Переход в текстовый режим */
 fclose(Fin); /* Закрываем файл Fin */
}
```

Язык Бейсик

```
'Изображение "рукописного" текста с помощью параметрического сплайна
DECLARE SUB Curve (aX#(), aY#(), M#, Param%())
DECLARE SUB MomSpline (n%, aX#(), aY#(), Mom#(), BC%(), Der#())
DECLARE FUNCTION PosBlank% (S$, Beg%)
DECLARE FUNCTION Round% (A#)
DECLARE FUNCTION Spline# (aX#(), aY#(), Mom#(), X#)
DECLARE FUNCTION Xscr% (X%)
DECLARE FUNCTION Yscr% (Y%)
DIM SHARED Ws AS INTEGER, Hs AS INTEGER
Ws = 640: Hs = 480 'ширина и высота экрана_
Nmax% = 50 'Максимальное число интервалов
DIM aX(0 TO Nmax%)  AS DOUBLE, aY(0 TO Nmax%) AS DOUBLE
DIM X0 AS INTEGER, Y0 AS INTEGER
DIM White AS INTEGER: White = 15
DIM Sf AS STRING, Number AS STRING
DIM Param(1 TO 6) AS INTEGER
DIM n%, Tb%, Te%, L%, i%, j%, Ind%, IndE%, pB%, M#
'Открываем физический файл "word" на чтение и даем ему номер #1
OPEN "word" FOR INPUT AS #1
INPUT "Укажите X0  в центральных координатах ", X0
INPUT "Укажите Y0  в центральных координатах ", Y0
INPUT "Укажите масштаб ", M#
SCREEN 12 'Переход в графический режим для монитора VGA
DO                  'Цикл чтения файла
 LINE INPUT #1, Sf
 L% = LEN(Sf) 'Длина строки Sf
 Tb% = ASC(MID$(Sf, 4, 1)) - ASC("0")'Код начала
 Te% = ASC(RIGHT$(Sf, 1)) - ASC("0") 'Код конца
 Ind% = 6: IndE% = L% - 3: n% = -1
 DO                     'Цикл чтения координат
  n% = n% + 1
  FOR j% = 1 TO 2 'Координаты очередной точки
   pB% = PosBlank%(Sf, Ind%)
   'Number - строковая запись очередной координаты
   Number = MID$(Sf, Ind%, pB% - Ind%)
   IF j% = 1 THEN
    aX(n%) = VAL(Number)
   ELSE
    aY(n%) = VAL(Number)
   END IF
   Ind% = pB% + 1
  NEXT j%          'Координаты очередной точки
 LOOP UNTIL Ind% = IndE% 'Цикл чтения координат
 Param(1) = X0: Param(2) = Y0: Param(3) = n%:
 Param(4) = Tb%: Param(5) = Te%: Param(6) = White
 'Изображаем кривую
 CALL Curve(aX(), aY(), M#, Param%())
```

```
LOOP UNTIL (EOF(1)) 'Цикл чтения файла
'Выход - нажатием клавиши Enter
DO: LOOP UNTIL INKEY$ = CHR$(13)
SCREEN 0 'Переход в текстовый режим
CLOSE #1 'Закрываем файл "word"
END

SUB Curve (aX#(), aY#(), M#, Param%())
...
END SUB

SUB MomSpline (n%, aX#(), aY#(), Mom#(), BC%(), Der#())
...
END SUB

FUNCTION PosBlank% (S$, Beg%)
'Определение позиции первого пробела, находящегося в строке S правее позиции Beg,
'считая, что такой пробел существует
DIM i AS INTEGER
i = Beg% + 1
DO WHILE MID$(S$, i, 1) <> " ": i = i + 1: LOOP
PosBlank% = i
END FUNCTION

FUNCTION Round% (A#)
'Округление до ближайшего целого
Round% = INT(A# + .5)
END FUNCTION

FUNCTION Spline# (aX#(), aY#(), Mom#(), X#)
...
END FUNCTION

FUNCTION Xscr% (X%)
'Переход от центральной координаты X к экранной
Xscr% = X% + Ws / 2
END FUNCTION

FUNCTION Yscr% (Y%)
'Переход от центральной координаты Y к экранной
Yscr% = Hs / 2 - Y%
END FUNCTION
```

Приведем пример файла WORD:

```
Tb=2 3 2 4.5 0 8 3 10 6.5 12 10 13.5 13.5 15 15 Te=2
Tb=1 15 15 12 10 9 5 9 1 11 0 16 3 Te=2
Tb=1 22 15 19 10 16 5 16 1 18 0 23 3 Te=2
Tb=3 25 0 23.5 3.5 25 6.5 27 9.5 30 10 31.5 8 30.5 5 29 2 25 0 Te=3
Tb=1 33 10 30 5 29 3.3 29 1.7 31 0 33 2 Te=2
Tb=1 37 10 34 5 33 3.3 33 1.7 35 0 38 3 Te=2
Tb=1 42 10 39 5 38 3.3 38 1.7 40 0 43 3 Te=2
Tb=1 47 10 44 5 43 3.3 43 1.7 45 0 48 3 Te=2
Tb=3 50 0 48.5 3.5 50 6.5 52 9.5 55 10 56.5 8 55.5 5 54 2 50 0 Te=3
Tb=1 58 10 55 5 54 3.3 54 1.7 56 0 58 2 Te=2
Tb=1 72 10 69 5 68 3.3 68 1.7 70 0 73 3 Te=2
Tb=1 77 10 74 5 73 3.3 73 1.7 75 0 78 3 Te=2
Tb=1 98 10 95 5 93.5 2.5 91 0 89.5 2.5 Te=2
Tb=1 98 10 95 5 94 3.3 94 1.7 96 0 99 3 Te=2
Tb=1 103 10 100 5 99 3.3 99 1.7 101 0 104 3 Te=2
Tb=2 104 3 108 5 110 7 110 10 108 9 106 7 104 3 106 0 110 2 112 5 Te=2
Tb=3 113 0 111.5 3.5 113 6.5 115 9.5 118 10 119.5 8 118.5 5 117 2 113 0 Te=3
Tb=1 121 10 118 5 115 0 112 -5 110.5 -7 109 -5 110 -4 115 0 123 6.4 Te=1
Tb=1 123 6.4 126.6 10 130.6 14 129 17 127.6 14 125 9.73 123 6.4 Te=1
Tb=3 124 0 122.5 3.5 124 6.5 126 9.5 129 10 130.5 8 129.5 5 128 2 124 0 Te=3
Tb=1 128 2 132 3 Tb=1
Tb=2 132 3 136 5 138 7 138 10 136 9 134 7 132 3 134 0 138 2 140 5 Te=2
Tb=3 141 0 139.5 3.5 141 6.5 143 9.5 146 10 147.5 8 146.5 5 145 2 141 0 Te=3
Tb=1 149 10 146 5 143 0 140 -5 138.5 -7 137 -5 138 -4 143 0 151 6.4 Te=1
Tb=1 153 10 150 5 149 3.3 149 1.7 151 0 154 1.7 154 4 152 6 150 5 Te=2
```

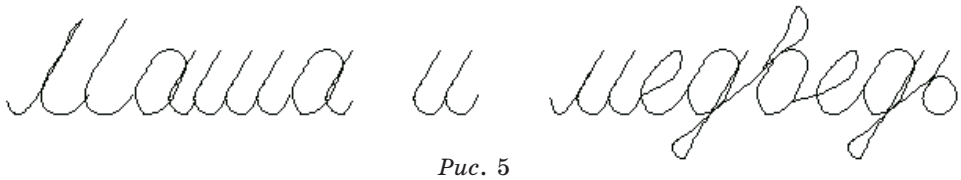


Рис. 5

Текст, закодированный в этом файле, представлен на *рис.* 5. Надеемся, читателя не затруднит его чтение и он согласится, что посредством параметрических сплайнов можно вполне прилично изобразить рукописные буквы. Конечно, приведенное решение можно значительно улучшить. Во-первых, программа не содержит контроля и диагностики. Есть у нее и более серьезный недостаток. Файл исходных данных неизбежно будет содержать одни и те же элементы. Их запись отличается только абсциссами точек, причем разность абсцисс соответствующих точек двух одинаковых элементов постоянна. Составление подобного файла представляет собой довольно трудоемкую и нудную работу.

Более эффективный, но зато и более сложный способ решения заключается в следующем. Каждый элемент записываем лишь один раз, и необязательно в файле, а просто в программе. Для каждого элемента может использоваться своя система координат, но каждая из них должна быть сдвигом какой-либо другой по оси `X` (но не `Y`!). Каждая буква задается (также непосредственно в программе) двумя массивами: один содержит номера элементов, а другой — сдвиги по оси `X` от последней точки предыдущего элемента до первой точки следующего. Мы помним из уроков чистописания, что рукописный текст — слитный. Как это обеспечить? Пусть `Xe,Ye` — координаты последней точки очередной, уже изображенной буквы. Нам надо знать, куда поместить следующую. Используя параметрические сплайны первого элемента, найдем в нем самую левую точку с ординатой `Ye`. Обозначим ее абсциссу через `Xnext` и положим `dx=Xnext–Xe`. Сдвинем теперь следующую букву влево на `dx`, тем самым слитность будет обеспечена. Мы не будем составлять подобную программу, предоставляя это заинтересованному читателю.

Заключение

Пора подводить итоги. Конечно, мы не исчерпали тему сплайнов. За пределами изложения остались многие вопросы, и в частности "фундаментальные", или *B*–сплайны, позволяющие удобнее строить сплайны; периодические сплайны, предназначенные для построения замкнутых кривых; построение пространственных кривых с помощью сплайнов и ряд других вопросов.

Однако мы надеемся, что читатель получил некоторое представление о сплайнах как удобном и надежном инструменте интерполяции и проведения кривых.

Литература

1. *Фокс А., Пратт М.* Вычислительная геометрия. М.: Мир, 1982.

2. *Алберг Дж., Нильсон Э., Уолш Дж.* Теория сплайнов и ее приложения. М.: Мир, 1972.

3. *Гастев В.А.* Краткий курс сопротивления материалов. М.: Наука, 1977.

4. *Ривкин С.Л., Александров А.А.* Термодинамические свойства воды и водяного пара. М.: Энергия, 1975.

5. *Соколинский Ю.А., Островский С.Л.* Задачи по теме "Компьютерная графика". Информатика, 1998, № 30, 31.

Гл. редактор
Е.Б.Докшицкая
Зам. гл. редактора
С.Л.Островский
Редакция:
Л.Н.Картвелишвили,
М.С Мачнев,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П. Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л. Володина,
С.М.Подберезина

выходит четыре раза в месяц
При перепечатке ссылка на ИНФОРМАТИКУ обязательна, рукописи не возвращаются

121165, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 240 1041

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Объединение педагогических изданий "ПЕРВОЕ СЕНТЯБРЯ"
Отпечатано в типографии "Пресса"
125865, Москва, ул. Правды, 24
Тираж 7000 экз.
Заказ №

Internet: infosef@glasnet.ru
Fidonet: 2:5020/69.32
WWW: http://www.glasnet.ru/~infosef
FTP: ftp://ftp.glasnet.ru/users/infosef

тел./факс (095)249 3138
факс (095)249 3184
тел. (095)249 3386
13.01