5. АППРОКСИМАЦИЯ И СГЛАЖИВАНИЕ ФУНКЦИЙ

Не всегда можно удовлетвориться прямым выводом информации на графическое устройство. Во многих случаях предварительная ее обработка (восполнение, приближение, сглаживание) позволяет выделить наиболее существенное, улучшить не только математическое, но и эстетическое представление информации. Важную роль методы аппроксимации играют и при анализе функций.

Современные устройства графического ввода и вывода в совокупности с быстродействующими ЭВМ позволяют математику не только видеть результаты, но и оперативно вмешиваться в процесс вычислений, выбирая другие узлы, меняя класс приближающих функций, определяя критерий согласия, контролируя точность приближения. На практике проблема приближения функций возникает во многих случаях, в частности, при вводе в машину функциональных зависимостей, полученных в результате эксперимента, при выборе представления таких зависимостей в памяти ЭВМ, при выводе информации на графические устройства.

В системах автоматизации проектирования и производства методы аппроксимации применяются прежде всего в целях конструирования кривых и поверхностей. При построении кривой в этих случаях утрачивает (или почти утрачивает) смысл такой математический критерий как точность аппроксимации и главную роль начинают играть такие критерии как внешний вид и гладкость кривой, отсутствие осцилляций и т.п.

В этой главе мы рассмотрим реализацию нескольких включенных в Графор методов аппроксимации и восполнения функций. Эти методы хорошо известны в математике и поэтому их описание приводится лишь постольку, поскольку оно необходимо для понимания смысла и структуры программ.

5.1. Сплайн - аппроксимация

Математические *сплайны* берут свое начало от тонких гибких стержней, которыми пользовались чертежники для проведения плавных кривых через заданные точки. Стержень закреплялся в точках (x_i, y_i) и принимал форму кривой $y(x)$ с минимальной "энергией натяжения", пропорциональной $\int y''^2\, dx$.

Если перейти к математическому описанию сплайна, то *сплайн-функцией* степени κ с точками соединения $x_0 < x_1 < ... < x_n$ будет функция $y(x)$, которая на отрезке $[x_0, x_n]$ имеет непрерывные производные до κ-1 включительно и на каждом из отрезков $[x_{i-1}, x_i]$ равна многочлену степени κ. Далее нас будут интересовать лишь кубические сплайны ($\kappa = 3$). Такой сплайн обеспечивает совпадение в узлах с исходной функцией и непрерывность первой и второй производных в точках соединения.

Прежде всего определяются первые производные во всех точках соединения. Решается трехдиагональная система n-1 уравнений с доминирующей главной диагональю. Она может быть решена, если задать два краевых условия. Программы, включенные в Графор, позволяют задавать четыре типа краевых условий:

а) известны значения первых производных на концах сетки (y'_0, y'_n);

б) известны значения вторых производных на концах сетки (y''_0, y''_n);

в) при построении периодического сплайна значения функции в первой и последней точках совпадают, а также $y'_0 = y'_n$ и $y''_0 = y''_n$;

г) если краевые условия не заданы, то считается, что $y''_0 = y''_n = 0$.

После того как построена трехдиагональная матрица (или дополненная трехдиагональная матрица в случае в)) и вектор свободных членов, система уравнений может быть решена. Для

решения такой системы существует эффективный алгоритм, который в отечественной литературе называется методом прогонки. В результате решения системы уравнений получается вектор первых производных y_i' в точках соединения.

Теперь значение $y(x)$ для $x_{i-1} \le x \le x_i$ определяется из многочлена

$$y(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \qquad (5.1)$$

коэффициенты которого легко найти, поскольку известны значения функции и первые производные в точках соединения.

Таким образом, метод позволяет выделить ряд самостоятельных функций:

- формирование трехдиагональной матрицы и вектора свободных членов в соответствии с заданными краевыми условиями - подпрограмма TDMP,
- решение системы уравнений, заданных матрицей и вектором свободных членов (метод прогонки) - подпрограмма TRIDIG,
- вычисление коэффициентов кубических многочленов для всех отрезков - подпрограмма SPLINE,
- вычисление значений сплайна на заданной равномерной сетке - подпрограмма SPLINT.

Отметим, что программа SPLINE при своей работе использует подпрограммы TDMP и TRIDIG. Эти программы универсальны и их применение не ограничено сплайн-аппроксимацией. Их универсальность имеет и более глубокий смысл. В частности, коэффициенты для кубических парабол можно вычислить, задав вектор производных полностью. Это означает, что программист по своему усмотрению может модифицировать вектор производных. При интерактивной работе таким образом можно подбирать подходящую форму кривой. Математическое обоснование сплайнов и их анализ изложен в [11].

Программа SPLINE(X, Y, U, N, A, B, C, D, KODE, IER) вычисляет коэффициенты кубического сплайна. Попутно вычисляются (если не заданы) значения первых производных в точках соединения. Программа имеет следующие параметры:

X, Y, U - векторы значений аргумента, функции и первых производных (длины N);

N - количество точек;

A, B, C - векторы коэффициентов кубического многочлена при переменных x^3, x^2, и x (длины N);

D - вектор свободных членов кубического многочлена (длины N);

KODE - признак задания краевых условий:

KODE = -2 - на концах сетки заданы вторые производные; перед вызовом программы y_1'' и y_n'' должны быть занесены соответственно в D(1) и D(N),

KODE = -1 - на концах сетки заданы первые производные; перед вызовом программы y_1' и y_n' должны быть занесены соответственно в D(1) и D(N),

KODE = 0 - краевые условия не заданы, что равносильно $y_1'' = y_n'' = 0$,

KODE = 1 - задан периодический сплайн, т. е. $y_1' = y_n'$ и $y_1'' = y_n''$,

KODE = 2 - вектор значений производных U задан полностью;

IER - код ошибки; этот код для программы SPLINE является транзитным и передается в том виде, в каком он получен в подпрограмме TRIDIG.

Коэффициенты A(I), B(I), C(I) и D(I) соответствуют отрезку от X(I) до X(I+1), I = 1, ..., N-1.

Программа SPLINT(X,N,A,B,C,D,Y,M) позволяет вычислить значения кубического сплайна на равномерной сетке с заданным шагом. Параметры программы:

X - вектор значений аргумента (длины N);

N - количество точек;

A, B, C - векторы коэффициентов кубического сплайна при переменных x^3, x^2, и x (длины N);

D - вектор свободных членов кубического сплайна (длины N);

Y - вектор значений кубического сплайна (длины M);

M - количество узлов равномерной сетки на отрезке [X(1), X(N)].

На рис. 5.1 построен сплайн для функций sin *x* и cos *x*, значения которых определены на сетке с шагом $\pi/6$, но для функции cos *x* значения первых производных в узлах заданы:

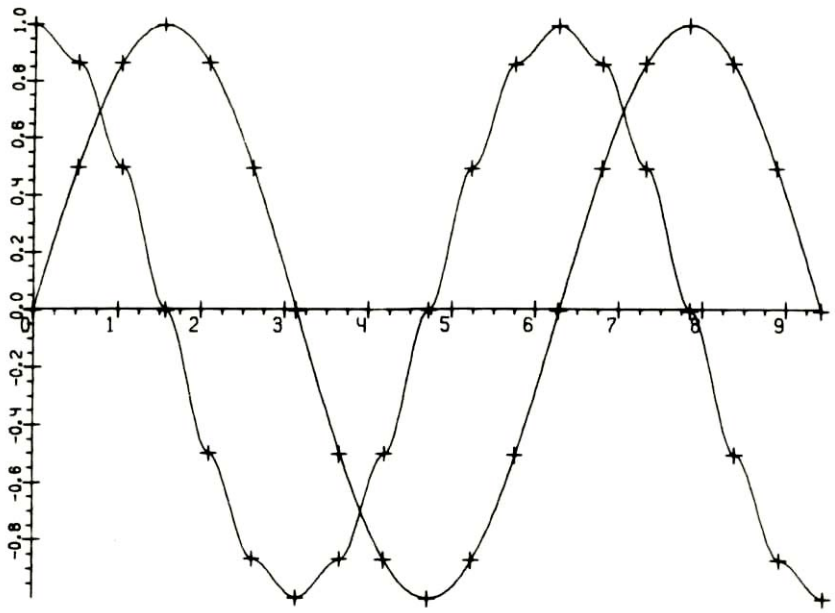

Рис. 5.1. Пример построения сплайнов для функций sin *x* и cos *x*

равными нулю. Пример показывает как, управляя производными, можно изменять форму кривой. Рисунок получен с помощью следующей программы:

```
      DIMENSION X(20), Y(20), U(20)
      DIMENSION A(20), B(20), C(20), D(20),YM(200)
      DX = 3.141593/6
      X(1) = 0.
      DO 1 I = 1, 19
      Y(I) = SIN(X(I))
    1 X(I+1) = X(I) + DX
      CALL PAGE(17., 26., 0, 0, 0)
      CALL LIMITS(0., X(19), -1., 1.)
      CALL REGION(1.5, 14., 14., 10., 0, 0, 0)
      CALL LINEMO(X, Y, 19, 1, -1)
      CALL SPLINE(X, Y, U, 19, A, B, C, D, 0, IER)
      CALL SPLINT(X, 19, A, B, C, D, YM, 200)
      DX = (X(19) - X(1)) / 199.
      CALL INCLIN(X(1), DX, 0, YM, 200, 0, 0)
      DO 2 I = 1, 19
    2 Y(I) = 0.
      CALL SPLINE(X, U, Y, 19, A, B, C, D, 2, IER)
      CALL SPLINT(X, 19, A, B, C, D, YM, 200)
      CALL INCLIN(X(1), X(19), 1, YM, 200, 0, 0)
      CALL AXES(0, 0, 0., 4, 0, 0, 0., 4, 0)
      CALL ENDPG('5.1')
      END
```

Программа TDMP(X,Y,N,A,B,C,D,KODE) позволяет вычислить элементы трехдиагональной матрицы или дополненной трехдиагональной матрицы (в случае периодического сплайна) и вектор свободных членов. Программа имеет следующие параметры:

X, Y - векторы значений аргумента и функции (длины N);

N - количество точек;

A, B, C - поддиагональный, диагональный и наддиагональный векторы матрицы (длины N);

D - вектор свободных членов (длины N);

KODE - признак задания краевых условий:

KODE = -2 - на концах сетки заданы вторые производные; перед вызовом программы y_1'' и y_n'' должны быть занесены соответственно в D(1) и D(N),

KODE = -1 - на концах сетки заданы первые производные, перед вызовом программы y_1' и y_n' должны быть занесены соответственно в D(1) и D(N),

KODE = 0 - краевые условия не заданы, что равносильно $y_1'' = y_n'' = 0$,

KODE = 1 - задан периодический сплайн, т. е. $y_1' = y_n'$ и $y_1'' = y_n''$.

Программа TRIDIG(U, N, A, B, C, D, KODE, IER) позволяет найти решение системы уравнений, заданной трехдиагональной матрицей или дополненной трехдиагональной матрицей, методом прогонки. В частности, для кубического сплайна решением системы является вектор первых производных в точках соединения. Параметры программы:

U - вектор результатов;

N - количество уравнений;

A, B, C - поддиагональный, диагональный и наддиагональный векторы матрицы;

D - вектор свободных членов;

KODE - характеристика матрицы: KODE = 0 – трехдиагональная матрица, KODE = 1 - дополненная трехдиагональная матрица;

IER - код ошибки: IER = 0 - ошибки нет, IER = 1 – если B(1) = 0, IER = 2 – если B(J) + C(J) * A(J-1) = 0.

Замечание. Способ задания матрицы.

а) Трехдиагональная матрица (KODE = 0):

$$\begin{bmatrix} r_{11} & r_{12} & 0 & \dots & 0 & 0 & 0 \\ r_{21} & r_{22} & r_{23} & \dots & 0 & 0 & 0 \\ 0 & r_{32} & r_{33} & \dots & 0 & 0 & 0 \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & \dots & r_{n-2,n-2} & r_{n-2,n-1} & 0 \\ 0 & 0 & 0 & \dots & r_{n-1,n-2} & r_{n-1,n-1} & r_{n-1,n} \\ 0 & 0 & 0 & \dots & 0 & r_{n,n-1} & r_{n,n} \end{bmatrix}$$

задается следующим образом:

A(1) = 0, A(2) = r_{21}, ..., A(N) = $r_{n,n-1}$ - поддиагональный вектор,

B(1) = r_{11}, B(2) = r_{22}, ..., B(N) = $r_{n,n}$ - диагональный вектор,

C(1) = r_{12}, C(2) = r_{23}, ..., C(N) = 0 - наддиагональный вектор;

б) дополненная трехдиагональная матрица (KODE = 1);

$$\begin{bmatrix} r_{11} & r_{12} & 0 & \ldots & 0 & 0 & r_{1,n} \\ r_{21} & r_{22} & r_{23} & \ldots & 0 & 0 & 0 \\ 0 & r_{32} & r_{33} & \ldots & 0 & 0 & 0 \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & \ldots & r_{n-2,n-2} & r_{n-2,n-1} & 0 \\ 0 & 0 & 0 & \ldots & r_{n-1,n-2} & r_{n-1,n-1} & r_{n-1,n} \\ 0 & r_{n,2} & 0 & \ldots & 0 & r_{n,n-1} & r_{n,n} \end{bmatrix}$$

задается следующим образом:

A(1) = $r_{1,n}$, A(2) = r_{21}, ..., A(N) = $r_{n,n-1}$ - поддиагональный вектор,

B(1) = r_{11}, B(2) = r_{22}, ..., B(N) = $r_{n,n}$ - диагональный вектор,

C(1) = r_{12}, C(2) = r_{23}, ..., C(N) = $r_{n,2}$ - наддиагональный вектор.

5.2. Параметрическое задание функций

Для вычисления сплайн-функции, заданной на сетке $x_1 < x_2 < \ldots < x_n$, требуется упорядоченная монотонно возрастающая последовательность x_i. Это означает, что сплайн-функция $y = y(x)$ может быть построена только для однозначной функции.

Однако, используя параметрическое задание функции, можно построить неоднозначную, в том числе замкнутую, кривую (периодический сплайн). Для этого выбирается независимая переменная t, удовлетворяющая указанным выше требованиям, и вычисляются сплайн-приближения $x(t)$ и $y(t)$, а затем строится кривая, проходящая через точки (x_i, y_i), соответствующие выбранным t_i.

Предлагается два способа задания параметра t. В простейшем случае t задается как последовательность целых чисел: $t_j = j$. Другая возможность состоит в том, что t_j соответствует суммарной длине хорд, которая является аппроксимацией длины дуги между первой и j-й точками. В этом случае

$$t_1 = 1,\ t_2 = 2 \text{ и } t_j = t_{j-1} + \sqrt{\frac{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \qquad (5.2)$$

Вычислив приближенно длины отрезков получившейся кривой $x = x(t)$, $y = y(t)$, можно построить более точные сплайны. Однако, это не приводит к заметному изменению самой кривой [11]. В комплекс Графор включена программа TCALC, которая позволяет вычислить вектор независимой переменной t, используя для этого соотношение (5.2).

Следует заметить, что параметрическое представление можно использовать не только при построении сплайнов. С тем же успехом им можно воспользоваться и при других способах аппроксимации, описанных в других разделах этой главы.

Программа TCALC(X, Y, T, N) позволяет вычислить вектор T для параметрического задания функции $x = x(t)$, $y = y(t)$. Значение t_j определяется как приведенное расстояние от точки (x_1, y_1) до точки (x_j, y_j), причем $t_1 = 1$, $t_2 = 2$. Параметры программы следующие:

X - вектор значений аргумента;

Y - вектор значений функции;

T - вектор значений параметра;

N - число точек.

На рис. 5.2 показан пример построения сплайнов с использованием параметрического представления кривых. Исходное задание рисунка содержит около 100 точек. На рис. 5.2, *а*) показано изображение, построенное путем соединения этих точек отрезками прямых. Рис. 5.2, *б*) построен с применением сплайн-интерполяции. На рис. 5.2, *в*) показано преобразование

а) б) с)

Рис. 5.2. Сплайны для таблично заданных неоднозначных функций:
а) исходное задание (около 100 точек); б) гладкое восполнение (около 3000 точек);
в) изменены вычисленные производные в точках соединения.

изображения: изменены вычисленные производные в точках соединения. Ниже приведен фрагмент программы, который применялся при построении этого рисунка.Здесь X, Y - массивы, содержащие узлы участков ломаных, а XN, YN - массивы, в которые заносятся значения гладких кривых, соответствующие этим ломаным.

```
      DIMENSION X(30),Y(30), A(30), B(30), C(30), D(30), T(30)
      DIMENSION UX(30), UY(30), XN(150), YN(150)
      . . . . . .
      GOTO (10, 20, 30) L
   10 CALL LINEO(X, Y, M)
      GOTO 5
   20 CALL TCALC(X, Y, T, M)
      CALL SPLINE(T, X, UX, M, A, B, C, D, 0, IER)
      CALL SPLINT(T, M, A, B, C, D, XN, N)
      CALL SPLINE(T, Y, UY, M, A, B, C, D, 0, IER)
      CALL SPLINT(T, M, A, B, C, D, YN, N)
      CALL LINEO(XN, YN, N)
      GOTO 5
   30 CALL TCALC(X, Y, T, M)
      CALL SPLINE(T, X, UX, M, A, B, C, D, 0, IER)
      DO 8 I = 1, M
    8 UX(I) = FX * UX(I)
      CALL SPLINE(T, X, UX, M, A, B, C, D, 2, IER)
      CALL SPLINT(T, M, A, B, C, D, XN, N)
      CALL SPLINE(T, Y, UY, M, A, B, C, D, 0, IER)
      DO 9 I = 1, M
    9 UY(I) = FY * UY(I)
      CALL SPLINE(T, Y, UY, M, A, B, C, D, 2, IER)
      CALL SPLINT(T, M, A, B, C, D, YN, N)
      CALL LINEO(XN, YN, N)
      GOTO 5
```

5.3. Локальные сплайны

В 5.1 проведение лекальной кривой основывалось на аналогии длинного гибкого стержня, т. е. сплайн строился с учетом положения всех n точек и поэтому требовалось решение n уравнений. Было отмечено, что, например, влияние краевых условий быстро затухает и сказывается лишь на ближайших к концам интервалах. На практике чертежники используют этот факт, применяя лекала - фигурные линейки для проведения плавных кривых. Так, когда строится участок кривой, то учитывается положение лишь четырех-пяти точек, ближайших к этому участку. Построенный таким образом сплайн будем называть *локальным сплайном*.

Непрерывность второй производной имеет значение при проектировании контура механического инструмента (например, кулачка), но несущественна для графического представления кривой. Поэтому можно заметно упростить алгоритм вычисления первых производных в точках соединения. При построении локального сплайна будем определять производные в точках соединения по 3, 4 или 5 смежным точкам, пользуясь известными интерполяционными формулами Ньютона и Стирлинга для равномерной сетки. Вычисление по этим формулам выполняет подпрограмма-функция DERIV5.

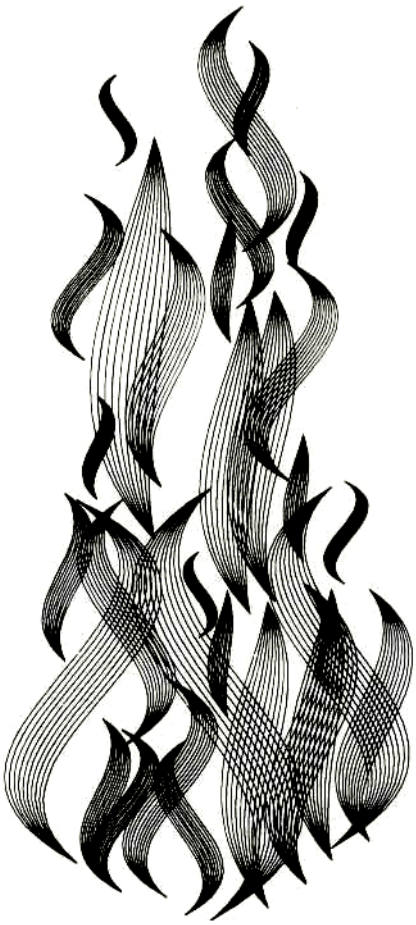

Рис. 5.3. Кубические параболы

Функция DERIV5(DX, Y, N, I) позволяет вычислить приближенное значение производной в i-ом узле для функции, заданной на сетке с постоянным шагом. Параметры функции задают:

DX - размер постоянного шага сетки;

Y - вектор значений исходной функции;

N - число точек;

I - номер точки, в которой вычисляется производная.

Таким образом, для вычисления первых производных в точках соединения не требуется решать систему уравнений, как это было в случае полного сплайна. Более того, не требуется знать производные во всех точках одновременно. Как только вычислены производные на концах участка, подпрограмма CUBPOL определяет коэффициенты кубического многочлена для этого участка, и они немедленно используются для рисования части кривой.

Программа CUBPOL(X1, X2, Y1, Y2, DY1, DY2, A) позволяет вычислить коэффициенты кубического многочлена $y(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3$. Программа имеет следующие параметры:

X1, X2 - значения аргумента на левом и правом концах отрезка;

Y1, Y2 - значения функции на левом и правом концах отрезка;

DY1, DY2 - производные в узлах X1, X2;

A - вектор коэффициентов кубического многочлена, причем A(1) = a_1, A(2) = a_2, A(3) = a_3, A(4) = a_4.

Внешне, с точки зрения программиста, рисование лекальной кривой ничуть не сложнее, чем рисование ломаной. Достаточно обратиться к программе SINCL, которая с помощью подпрограммы DERIV5 вычисляет производные, затем, обращаясь к CUBPOL, вычисляет

коэффициенты кубического многочлена, и, наконец, строит участок кривой так, чтобы длина составляющих его отрезков прямых не превышала 1,5 - 2 мм.

При построении локального сплайна экономней используется как время, так и пространство в памяти машины. Тем не менее, кривая, которая при этом получается, практически не отличается от кривой, построенной как полный сплайн.

Программа SINCL(XBEG, DX, Y, N) позволяет провести плавную кривую для функции, заданной на равномерной сетке. В узлах сетки обеспечивается совпадение с исходной функцией и непрерывность первой производной. Программа имеет следующие параметры:

XBEG - начальное значение аргумента;

DX - размер постоянного шага сетки;

Y - вектор значений заданной функции;

N - число точек.

На рис. 5.3 каждый "язычок" пламени образован десятью кубическими параболами. Для вычисления коэффициентов многочлена $y = a_i x^3 + b_i x^2 + c_i x + d_i$ задавались координаты двух точек и первые производные в этих точках.

5.4. Сглаживание методом наименьших квадратов

Метод наименьших квадратов используется в том случае, когда требуется функцию, заданную в m точках $x_j, y_j (j = 1,2,...,m)$ приблизить многочленом степени $n < m$:

$$y(x) = \sum_{i=0}^{n} a_i x^i . \qquad (5.3)$$

Поскольку вычисленная кривая $y(x)$, вообще говоря, не проходит через заданные точки и дает сглаженное множество значений $y(x)$, необходимо определить меру близости исходной и приближающей функций. Из метода наименьших квадратов следует, что необходимо выбрать многочлен, который обеспечивает минимум суммы

$$\sum_{j=1}^{m} \rho_j (y_j - \sum_{i=0}^{n} a_i x_j^i)^2 . \qquad (5.4)$$

(В этом критерии имеется коэффициент $0 \le \rho_j \le 1$, который определяет значимость точки или вес, с которым она влияет на выбор многочлена.) Если $\rho_j = 0$, то отклонение в j-ой точке не учитывается при выборе многочлена.) Если рассматривать выражение (5.4) как функцию переменной a_k, то для нахождения минимума надо продифференцировать это выражение по каждой из неизвестных a_k. В результате получим

$$2\sum_{j=1}^{m} \rho_j (y_j - \sum_{i=0}^{n} a_i x_j^i) x_j^k = 0, \qquad k = 0, 1, \ldots, n,$$

или после преобразования

$$\sum_{j=1}^{m} \rho_j y_j x_j^k = \sum_{i=0}^{n} a_i \left[\sum_{j=1}^{m} \rho_j x_j^{i+k} \right], \qquad k = 0, 1, ..., n.$$

То есть для определения коэффициентов необходимо решить систему $n + 1$ линейных уравнений, которые называют нормальными уравнениями, с $n + 1$ неизвестными a_i. B матричной форме систему можно записать как B x $\bar{a} = \bar{c}$, где B - симметрическая матрица размерности $n + 1$. Заметим, что матрица B не зависит от значений исходной функции и поэтому при обработке нескольких функций, заданных на одной сетке, достаточно построить и обратить матрицу B только один раз.

Программа LESQ(X, Y, RO, M, A, N) позволяет вычислить коэффициенты аппроксимирующего многочлена $y(x) = a_1 + a_2 x + ... + a_n x^{n-1}$ заданной степени методом наименьших квадратов с учетом весовых коэффициентов. Программа строит исходную матрицу A, а затем обращает ее на своем поле, используя подпрограмму SMINV. Программа имеет следующие параметры:

X, Y - векторы значений аргумента и аппроксимирующей функции (длины |M|);

RO - вектор значений весовых коэффициентов (длины |M|);

M - число точек (если M < 0, то считается, что все точки имеют одинаковый вес, равный 1, и число точек принимается равным |M|; в этом случае в качестве формального параметра RO может быть задан любой вектор, например X, причем он не влияет на результаты вычислений и сохраняется неизменным);

A - вектор коэффициентов аппроксимирующего многочлена длины |N| (коэффициенты заносятся в порядке возрастания индекса, т. е. A(1) = a_1, A(2) = a_2, ..., A(N) = a_n);

N - степень аппроксимирующего многочлена плюс 1, (если N < 0, программа использует вычисленную предыдущим обращением к ней матрицу B^{-1} порядка |N|).

Программа SMINV(B, V, N) позволяет найти матрицу, обратную симметрической, методом квадратных корней. Программа использует наддиагональный треугольник исходной матрицы, т. е. те элементы b_{ij}, для которых $i \le j$. Обратная матрица помещается на место исходной и является полной матрицей, для которой $b_{ij} = b_{ji}$. Параметры программы:

B - исходная симметрическая матрица порядка N;

V - рабочий вектор длины N;

N - порядок матрицы.

Итак, мы рассмотрели программы, которые позволяют получить многочлен, аппроксимирующий функцию, заданную таблично. По найденным коэффициентам можно вычислять значения аппроксимирующего многочлена с помощью программы PVAL. В том случае, когда нас интересует лишь графическое представление аппроксимирующей кривой, можно воспользоваться программой LSFIT.

Программа PVAL(RES, ARG, A, N) позволяет вычислить значения многочлена

$$y(x) = a_1 + a_2 x + ... + a_n x^{n-1}$$

(n - 1)-ой степени для заданного аргумента. Параметры программы следующие:

RES - значение многочлена от заданного аргумента;

ARG - значение аргумента;

A - вектор коэффициентов многочлена (длины N);

N - степень многочлена плюс 1.

Программа LSFIT(X, Y, RO, M, N, MPTS) позволяет начертить кривую, аппроксимирующую функцию методом наименьших квадратов с учетом весовых коэффициентов. Для вычисления коэффициентов аппроксимирующего многочлена используется подпрограмма LESQ. Программа LSFIT имеет следующие параметры:

X, Y - векторы значений аргумента и аппроксимируемой функции (длины |M|);

RO - вектор значений весовых коэффициентов (длины |M|);

M - число точек (если M < 0, то считается, что все точки имеют одинаковый вес, равный 1, и число точек принимается равным |M|, в этом случае в качестве формального параметра RO может быть задан любой вектор, причем он не влияет на результаты вычислений и сохраняется неизменным);

N - степень аппроксимирующего многочлена плюс 1 (этот параметр передается в программу LESQ, и если N < 0, то подпрограмма LESQ использует вычисленную предыдущим обращением к ней матрицу B^{-1} порядка |N|);

|MPTS| - количество узлов равномерной сетки на отрезке [X(1),X(M)]: если MPTS > 0, аппроксимирующая кривая проводится непрерывной линией, если MPTS < 0 - штриховой.

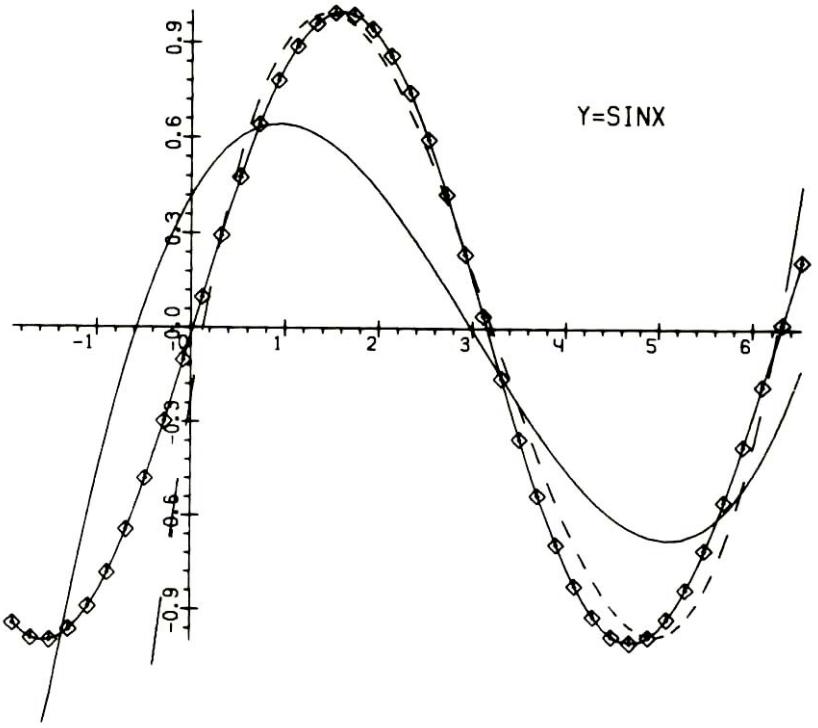

Рис. 5.4. Аппроксимация синусоидальной кривой многочленом третьей степени

На рис. 5.4 показана аппроксимация синусоидальной кривой многочленом третьей степени. Исходная кривая помечена маркером номер 3. Сплошной линией показана кривая, принадлежащая многочлену, который получен для точек, взятых с равными весами. Штриховая кривая принадлежит многочлену, полученному при условии, что точки на отрезке [-1.9, 0.] имеют вес равный 0, точки на отрезке [0.1, 3.1] - вес, равный 1 и все остальные точки взяты с весом 0.2. Очевидно, улучшение приближения на выделенном отрезке [0.1, 3.1] получено в ущерб приближению для всей кривой. Ниже приведена программа, выполняющая этот рисунок.

```
      DIMENSION X(100),Y(100),RO(100)
      X(1) = -1.9
      DO 1 I = 1, 85
      Y(I) = SIN(X(I))
    1 X(I+1) = X(I)+.1
      CALL PAGE(17., 26., 0, 0, 0)
      CALL MINMAX(Y, 85, YN, YX)
      CALL LIMITS(-1.9, X(85), YN, YX)
      CALL REGION(1.5, 1.5, 14., 11., 0, 0, 0)
      CALL LINEMO(X, Y, 85, 3, 1)
      DO 2 I = 1, 85
    2 RO(I) = 1.
      CALL LSFIT(X, Y, RO, 85, 4, 100)
      DO 3 I = 1, 20
```

```
  3 RO(I) = 0.
    DO 4 I = 51, 85
  4 RO(I) =.2
    CALL LSFIT(X, Y, RO, 85, 4, -100)
    CALL SYMBOL(11.5, 10.5, .4, 'Y = SINX', 6, 0)
    CALL AXES(0, 0, .1, 5, 0, 0, .3, 5, 0)
    CALL ENDPG('5.4')
    END
```

Описанный метод не рекомендуется использовать, если степень многочлена выше пяти, так как матрица становится плохо обусловленной. Это обстоятельство снижает ценность прямого решения системы нормальных уравнений. Можно избежать затруднений воспользовавшись для представления функции ортогональными многочленами, т. е. вычислять не многочлен (5.3), а некоторый его эквивалент. Тогда наилучшей аппроксимацией

$$y(x) = \sum_{i=0}^{n} s_i p_i(x)$$

будет такая, при которой обеспечивается минимум суммы

$$\sum_{j=1}^{m} \rho_j \left[y_j - \sum_{i=0}^{n} s_i p_i(x_j) \right]^2 .$$

Для образования ортогональных многочленов используется трехчленное рекуррентное соотношение (см. [12]).

Программа APPOLY(X, Y, RO, M, A, N, C) позволяет вычислить коэффициенты аппроксимирующего многочлена (5.3) заданной степени методом наименьших квадратов с использованием ортогональных многочленов. Параметры программы:

X, Y - векторы значений аргумента и аппроксимируемой функции (длины |M|);

RO - вектор значений весовых коэффициентов (длины |M|);

M - количество точек (если M < 0, то считается, что все точки имеют одинаковый вес, равный 1, и количество точек принимается равным |M|; в этом случае в качестве формального параметра RO может быть задан любой вектор, причем он не влияет на результаты вычислений и сохраняется неизменным);

A - вектор коэффициентов аппроксимирующего многочлена (5.3) длины |N| (коэффициенты заносятся в порядке возрастания индекса, т. е. A(1) = a_1, ..., A(N) = a_n);

N - степень аппроксимирующего многочлена плюс 1;

C - рабочий вектор длины 2 * N

По вычисленным коэффициентам можно получить значения аппроксимирующего многочлена, воспользовавшись подпрограммой PVAL, и затем начертить аппроксимирующую кривую с помощью одной из имеющихся подпрограмм проведения линий (LINEO, LINEMO, INCLIN и т. п.).

5.5. Аппроксимация функции конечным рядом Фурье

Если функция $f(x)$ задана на отрезке $[0,\ 2\pi]$ в 2N + 1 дискретных точках, т. е. с шагом $2\pi/(2N+1)$, то она может быть представлена как

$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{M} (A_k \cos kx + B_k \sin kx) \qquad (5.5)$$

причем $0 \le M \le N$ и $0 \le x \le 2\pi$.

Поскольку система функций, участвующих в разложении (5.5), ортогональна на заданном дискретном множестве точек, коэффициенты разложения представляются формулами

$$A_k = \frac{2}{2N+1}\sum_{l=0}^{2N} f(\frac{2\pi l}{2N+1})\cos\frac{2\pi k}{2N+1}l, \qquad k = 0, 1, \ldots, N,$$

$$B_k = \frac{2}{2N+1}\sum_{l=0}^{2N} f(\frac{2\pi l}{2N+1})\sin\frac{2\pi k}{2N+1}l, \qquad k = 0, 1, \ldots, N.$$

Если в разложении (5.5) $M = N$, то $2N + 1$ значений $f(x)$ ($x = 0, 1, ..., 2N$) определяют $2N + 1$ коэффициентов A_k, B_k. Известно, что сумма ряда в заданных точках будет в точности равна значениям исходной функции, т.е. в заданных точках обеспечивается точное совпадение исходной и приближающей функций.

Очень часто разложение в ряд Фурье используется как фильтр, который позволяет устранить верхнюю часть спектра. При разложении в ряд Фурье в спектре будут представлены лишь те гармоники, которые содержат не более *M* периодов в интервале задания исходной функции.

Реализация метода состоит из трех подпрограмм. Подпрограмма FORIT вычисляет указанное количество коэффициентов *M* для функции, определенной на дискретном множестве точек, а подпрограмма FORIF - заданное число коэффициентов *M* ряда Фурье, аппроксимирующего исходную функцию, значения которой определяются с помощью подпрограммы-функции на отрезке $[0, 2\pi]$ в $2N+1$ точках с шагом $2\pi/(2N+1)$. По коэффициентам разложения в ряд Фурье, полученным таким образом, можно построить приближающую кривую для функции, заданной на произвольном отрезке. Для этой цели предназначена подпрограмма FORFIT. Кривая может быть вычерчена как непрерывной, так и штриховой линией. (Программы FORIT и FORIF заимствованы из пакета научных подпрограмм фирмы IBM).

Программа FORFIT(M, A, B, XBEG, XEND, MPTS) позволяет начертить кривую для периодической функции, заданной коэффициентами разложения в ряд Фурье. Коэффициенты могут быть получены с помощью подпрограмм FORIT или FORIF. Параметры программы:

M - количество гармоник, участвующих в разложении;

A - вектор коэффициентов при косинусах в разложении функции длины M + 1 (коэффициенты располагаются в порядке возрастания индекса);

B - вектор коэффициентов при синусах в разложении функции длины M + 1 (коэффициенты располагаются в порядке возрастания индекса, причем B(1) = 0);

XBEG, XEND - значения аргументов, соответствующих левому и правому концу отрезка, на котором определена функция;

|MPTS| - число интервалов, покрывающих аппроксимирующую кривую (ширина интервала = (XEND - XBEG)/|MPTS|, MPTS $\neq 0$); если MPTS > 0, аппроксимирующая кривая проводится непрерывной линией, если MPTS < 0 - штриховой.

Программа FORIT(FNT, N, M, A, B, IER) позволяет вычислить коэффициенты ряда Фурье, аппроксимирующего исходную табулированную функцию, заданную на отрезке $[0, 2\pi]$ в $2N + 1$ равноотстоящих точках. Параметры программы:

FNT - вектор, задающий $2N + 1$ значений табулированной функции;

N - определяет отрезок так, что на отрезке $[0, 2\pi]$ значения заданы в $2N + 1$ равноотстоящих точках с шагом $2\pi/(2N + 1)$;

M - количество гармоник, участвующих в разложении;

A - вектор коэффициентов при косинусах в разложении функции длины M + 1 (коэффициенты располагаются в порядке возрастания индекса);

B - вектор коэффициентов при синусах в разложении функции длины M + 1 (коэффициенты располагаются в порядке возрастания индекса, причем B(1) = 0);

IER - код ошибки: IER = 0 - ошибки нет, IER = 1, если N < M, IER = 2, если M < 0.

Программа FORIF(FUN, N, M, A, B, IER) позволяет вычислить коэффициенты ряда Фурье, аппроксимирующего исходную функцию, значения которой вычисляются с помощью подпрограммы-функции на отрезке $[0, 2\pi]$ в $2N+1$ точках с шагом $2\pi/(2N+1)$. Параметры программы:

FUN - имя подпрограммы-функции, которая используется для вычисления значения функции в заданных точках.

Остальные параметры имеют тот же смысл, что и аналогичные параметры подпрограммы FORIT.

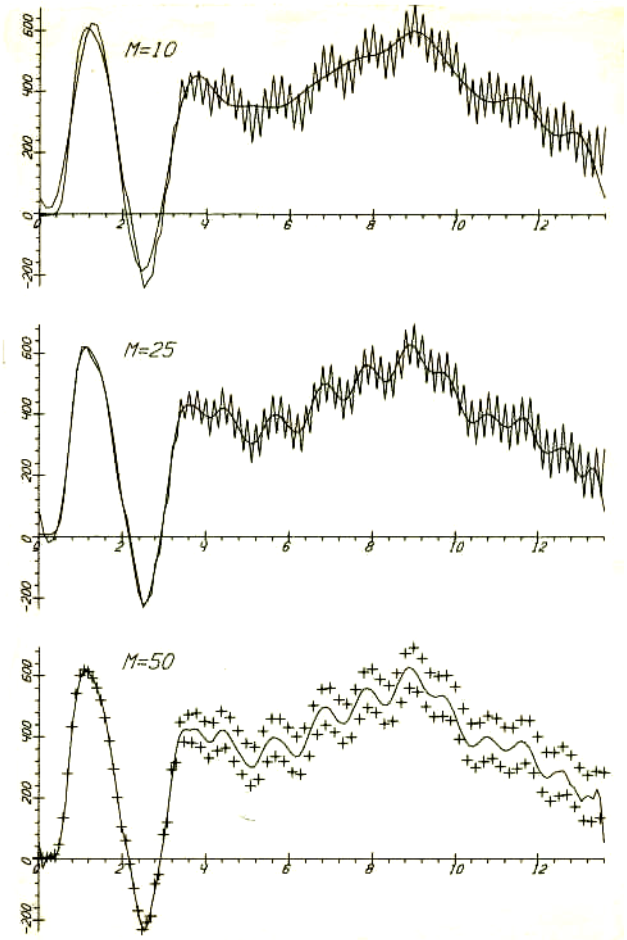

Рис. 5.5. Пример, иллюстрирующий использование разложения в ряд Фурье как фильтра

Использование разложения функции в ряд Фурье как фильтра иллюстрируется на рис. 5.5, где показана исходная функция, помеченная маркером + на нижнем рисунке, и приближение, полученное при разложении в ряд Фурье. На рисунках указано число M - количество гармоник в разложении.

Ниже приводится программа, с помощью которой получен график, изображенный в нижней части рис. 5.5 (M = 50).

```
      DIMENSION X(140),Y(140), A(100), B(100)
      READ 1, Y
    1 FORMAT(7F10.3)
      X(1) = 0.
      DO 2 I = 1,137
    2 X(I+1) = X(I) +.1
      CALL PAGE(17., 25.5, 0, 0, 0)
      CALL MINMAX(Y, 137, YN, YX)
```

```
CALL LIMITS(0., X(137), YN, YX)
CALL REGION(1., .5, 15., 7.5, 0, 0, 0)
CALL LINEMO(X, Y, 137, 1, -1)
CALL FORIT(Y, 68, 50, A, B, IER)
CALL FORFIT(50, A, B, 0., X(137), 150)
CALL ITALIC(1)
CALL SYMBOL(3.2, 7.3, .5, 'M=50', 4, 0)
CALL AXES(0, 0, 0., 5, 0, 0, 200., 5, 0)
CALL ENDPG('5.5')
END
```

5.6. Линейный фильтр

В некоторых случаях сглаживание функции, связанное с устранением высокочастотных составляющих, можно выполнить более простым способом по сравнению с аппроксимацией рядами Фурье. Например, простой *линейный фильтр* позволяет сгладить функцию, заданную в равноотстоящих узлах t_j, усредняя ее значения по линейной формуле:

$$\tilde{f}_j = \left(\sum_{i=j-m}^{j+m} f_i\right) \Big/ (2m+1), \qquad j = 1, 2, ..., n \qquad (5.6)$$

Заметим, что суммируется нечетное число точек. Если суммируется четное количество точек, то используется формула:

$$\tilde{f}_j = \left(\sum_{i=j-m+1}^{j+m} f_i\right) \Big/ (2m), \qquad j = 1, 2, ..., n \qquad (5.7)$$

Вообще говоря, формулы (5.6), (5.7) верны лишь в том случае, если $m < j \le n - m$, т.е. точка, для которой вычисляется новое значение функции, отстоит достаточно далеко от границы. Во всех других случаях соответствующим образом изменяются нижний и верхний пределы суммирования и знаменатель. Такой фильтр уменьшает амплитуды гармоник в верхней половине спектра. Более подробный анализ линейных фильтров можно найти в монографии Хемминга [12].

Описываемый метод сглаживания реализован в программе LINFIL. Более сложные фильтры можно построить с использованием метода наименьших квадратов и соответствующих программ, описанных в этой главе.

Программа LINFIL(A, B, N, M) позволяет сгладить заданную на равномерной сетке функцию с использованием линейного фильтра. Параметры программы следующие:

A - вектор значений функции (длины |N|);

B - вектор усредненных значений функции в узлах (длины |N|);

N - число узлов сетки;

M - параметр, определяющий число узлов, участвующих в суммировании (индекс *m* из формул (5.6) и (5.7).

При N > 0 суммирование ведется по формуле (5.6), при N < 0 используется формула (5.7). Число узлов берется равным |N|.

Если в обращении к программе в качестве формальных параметров A и B указан один и тот же вектор, то в сумму войдут вновь вычисленные значения функции в узлах, расположенных слева от текущего узла. Этот способ может быть полезен, если используется формула (5.7).

На рис. 5.6 показан пример сглаживания функции с помощью линейного фильтра (ср. с рис. 5.5).

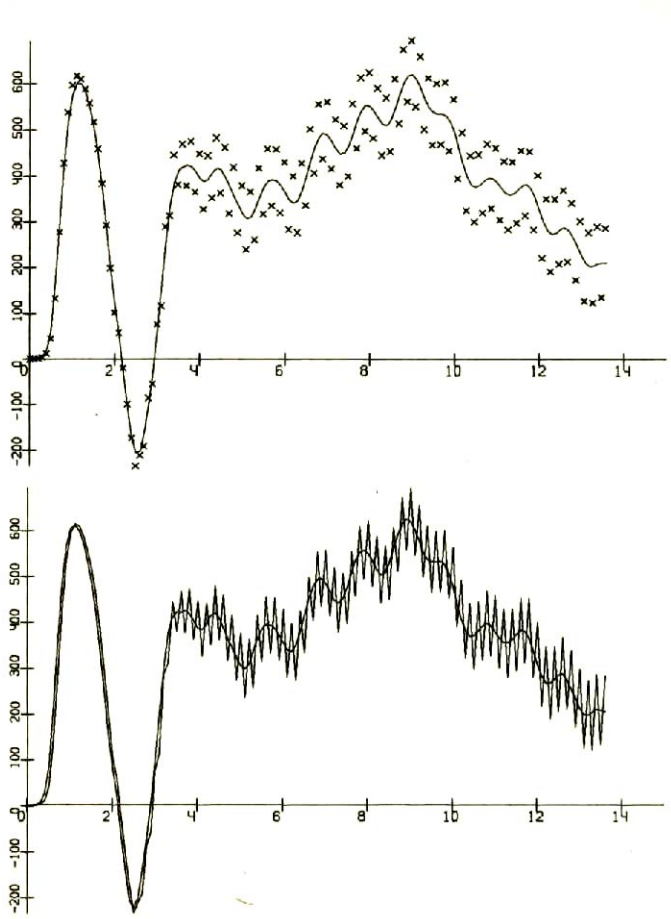

Рис. 5.6. Пример использования линейного фильтра

```
   DIMENSION Y(140), Z(140)
   READ 5, Y
 5 FORMAT(7F10.3)
   CALL PAGE(16., 25., 0, 0, 0)
   CALL LINFIL(Y, Z, -137, 1)
   CALL MINMAX(Y, 137, ZMN, ZMX)
   CALL REGION(1., .5, 15., 10., 0, 0, 0)
   CALL LIMITS(0., 15., ZMN, ZMX)
   CALL INCLIN(0.,.1, 0, Y, 137, 0, 0)
   CALL INCLIN(0.,.1, 0, Z, 137, 0, 0)
   CALL AXES(0, 0, 0.,0, 0, 0, 0.,0, 0)
   CALL REGION(1.,11.,15.,10.,0, 0, 0)
   CALL INCLIN(0.,.1, 0, Y, 137,-2,-1)
   CALL LINFIL(Y, Y, 137, 1)
   CALL INCLIN(0.,.1, 0, Y, 137, 0, 0)
   CALL AXES(0, 0, 0.,0, 0, 0, 0.,0, 0)
   CALL ENDPG('5.6')
   END
```

5.7. Аппроксимация функций ортогональными многочленами Чебышева

Для приближения непериодических функций используется ортогональная система функций - многочлены Чебышева, которые, в сущности, являются функциями Фурье $\cos n\theta$, замаскированными простым преобразованием переменной $\theta = \arccos x$. Таким образом, $T_n(x) = \cos(n \arccos x)$. Многочлены Чебышева связаны рекуррентным соотношением:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

причем $T_0 = 1,\ T_1 = x.$

Приближающая функция ищется в виде суммы многочленов Чебышева, т. е.

$$f(x) \approx \sum_{i=0}^{M} c_i T_i(x), \qquad (5.8)$$

Используя узловые точки многочленов, т. е. систему точек, на которой многочлены Чебышева ортогональны, получаем следующую формулу для вычисления коэффициентов c_i:

$$c_i = \left(\sum_{j=1}^{N} f(x_j)T_i(x_j)\right) \Big/ \left(\sum_{j=1}^{N} T_i^2(x_j)\right). \qquad (5.9)$$

Узловые точки многочленов распределены неравномерно (они сгущаются к концам интервала [-1, 1]), а именно

$$x_j = \cos\left(\frac{2j-1}{2N}\pi\right), \quad j = 1, 2, ..., N. \qquad (5.10)$$

Программа CHENSP вычисляет коэффициенты c_i для дискретной функции, заданной в равноотстоящих точках, используя при этом соответствующее преобразование переменной (5.10). После того, как коэффициенты известны, программа позволяет вычислить значения функции на заданной равномерной сетке, опять-таки с преобразованием переменной.

Следует заметить, что коэффициенты c_i (см. (5.9)) не зависят от *M*, т. е. от количества многочленов, входящих в сумму (5.8). Этот факт используется в примере, который приводится вслед за описанием программы.

Программа CHENSP(YM, N, YH, K, C, M) позволяет вычислить коэффициенты приближающей функции C (если N > 0) и ее значения YH (если K > 0) на множестве равноотстоящих точек, используя ортогональную систему функций - многочлены Чебышева. Если N = 0, то значения YH вычисляются в предположении, что коэффициенты C вычислены ранее (см. пример). Программа имеет следующие параметры:

YM - вектор значений дискретной функции длины |N|;

YH - вектор значений приближающей функции длины |K|;

C - вектор коэффициентов приближающей функции длины |M|.

На рис. 5.7 изображена часть синусоиды, на которую наложен шум (Y=SIN(X)+RAND(-0.1, 0.1, 1937)), и приближение этой функции многочленами Чебышева (два случая: M = 12 и M = 6).

```
DIMENSION YM(100), YH(100), C(12)
KR = 1937
XM = 0.
YM(1) = RAND(-1.01, 0.01, KR)
```

```
   DO 1 I = 2, 100
   XM = XM + 0.0314159
 1 YM(I) = SIN(XM) + RAND(-0.1,0.1, KR)
   CALL CHENSP(YM, 100,YH, 100, C, 12)
   CALL PAGE(17., 26., 0, 0, 0)
   CALL REGION(1.,1.,13.,10., 0, 0, 0)
   CALL LIMITS(0.,3.3,-0.2, 1.15)
   CALL AXES('M=12',4, 0.,0, 0, 0, 0.,0, 0)
   CALL INCLIN(0.,0.0314159, 0, YM, 100, 0, 0)
   CALL INCLIN(0.,0.0314159, 0,YH, 100, 0, 0)
   CALL CHENSP(0.,0,YH, 100, C, 6)
```

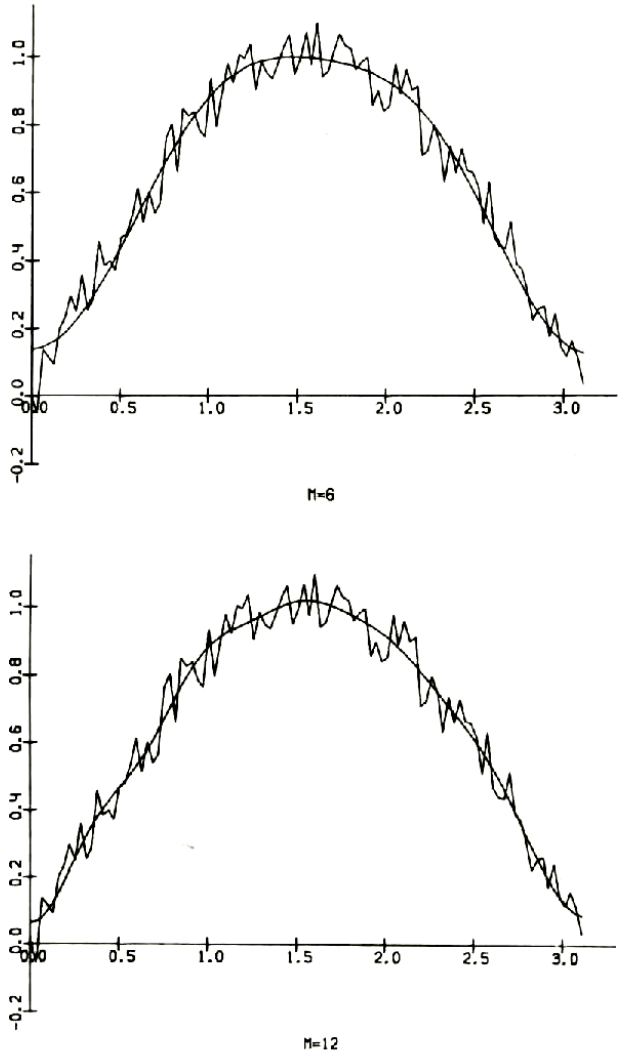

Рис. 5.7. Пример, иллюстрирующий аппроксимацию функции многочленами Чебышева

```
   CALL REGION(1.,13.,13.,10.,0, 0, 0)
   CALL AXES('M=6', 3,0.,0, 0, 0, 0.,0, 0)
   CALL INCLIN(0.,0.0314159, 0,YM, 100, 0, 0)
   CALL INCLIN(0.,0.0314159, 0,YH, 100, 0, 0)
   CALL ENDPG('5.7')
   END
```

Функция RAND(A, B, KR) позволяет генерировать псевдослучайные числа с равномерным распределением в интервале (A, B), где A < B. Функция имеет следующие параметры:

A, B - нижняя и верхняя границы интервала;

KR - целая переменная, принимающая значения в диапазоне от 1 до 67108863.

Перед первым обращением к функии RAND необходимо занести в KR нечетное число в указанных пределах. Значение KR не следует изменять между обращениями к функции.

5.8 Аппроксимация методом Безье

В системах автоматизации проектирования и производства для конструирования кривых и поверхностей применяется аппроксимация методом Безье. Сущность метода заключается в следующем.

Пусть задана совокупность из $n+1$ точек $\overline{P}_0, ..., \overline{P}_n$, которую будем называть *ломаной Безье*. *Кривая Безье*, соответствующая этой ломаной, описывается в виде функции параметра t следующим полиномом:

$$\overline{p}(t) = \sum_{i=0}^{n} \overline{P}_i J_{ni}(t), \qquad (5.11)$$

где $\overline{p}(t)$ - радиус-вектор точек на кривой, а $J_{ni}(t)$ - аппроксимирующие многочлены Бернштейна, равные

$$J_{ni}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}. \qquad (5.12)$$

Здесь $0 \le t \le 1$ и, кроме того, предполагается, что $t^i = 1$ при $i = 0$ и $t = 0$.

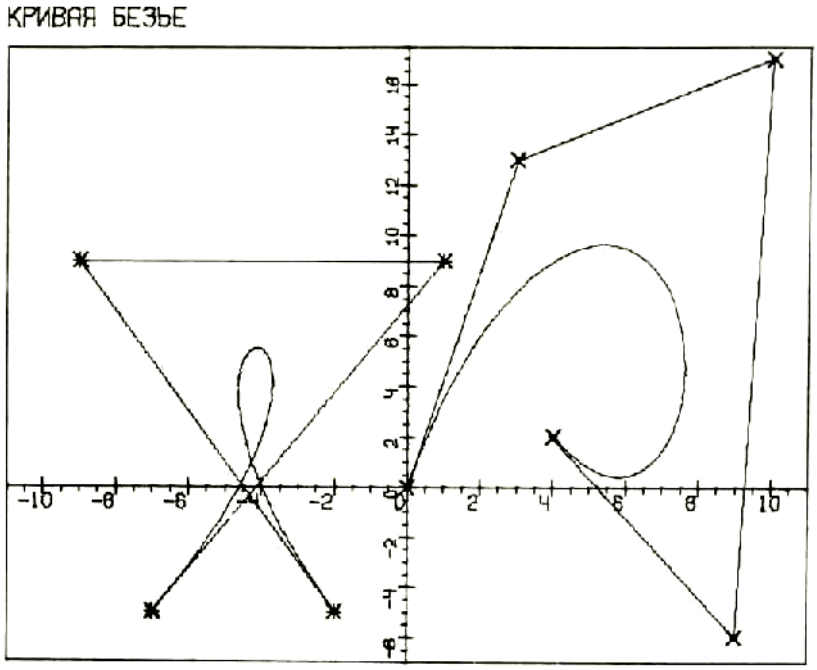

Рис. 5.8. Пример аппроксимации методом Безье

Ломаная Безье однозначно определяет форму кривой Безье. Изменяя положения вершин ломаной, можно управлять формой соответствующей кривой Безье. При этом следует иметь в виду следующее:

1) самой кривой в общем случае будут принадлежать только первая и последняя вершины ломаной Безье, остальные вершины будут лишь оказывать влияние на вид и гладкость кривой (рис. 5.8);

2) наклоны касательных векторов в крайних точках кривой Безье и ломаной Безье совпадают (рис. 5.8), поэтому при сопряжении двух кривых Безье, заданных ломаными $\overline{P}_0$, ..., $\overline{P}_n$ и $\overline{Q}_0$, …, $\overline{Q}_m$ одинаковый наклон кривых в точке соединения получается в том случае, если точки $\overline{P}_{n-1}$, $\overline{P}_n$ (которая совпадает с $\overline{Q}_0$) и $\overline{Q}_1$ лежат на одной прямой;

3) как видно из выражений (5.11) и (5.12), степень аппроксимирующего полинома равна n (т. е. числу звеньев в ломаной Безье), поэтому для увеличения порядка кривой Безье достаточно лишь задать дополнительные вершины в соответствующей ломаной Безье;

4) кривая Безье всегда целиком лежит внутри выпуклой оболочки ломаной Безье.

Метод аппроксимации Безье реализован в программах BEZ2 и BEZ3, с помощью которых можно построить соответственно двумерные и трехмерные кривые Безье.

Программа BEZ2(XP, YP, NPLG, XC, YC, NPTS) позволяет вычислить значения координат точек плоской кривой, аппроксимирующей заданную совокупность точек методом Безье. Программа имеет следующие параметры:

XP, YP - векторы значений соответственно X-, и Y-координат заданной совокупности точек (ломаной Безье) длины NPLG;

NPLG - число точек в ломаной Безье;

XC, YC - векторы значений соответственно X-, и Y-координат аппроксимирующей кривой Безье длины NPTS.

NPTS - число точек в аппроксимирующей кривой Безье.

Программа BEZ3(XP, YP, ZP, NPLG, XC, YC, ZC, NPTS) позволяет вычислить значения координат точек пространственной кривой, аппроксимирующей заданную совокупность точек методом Безье. Параметры программы следующие:

XP, YP, ZP - векторы значений соответственно X-, Y- и Z-координат заданной совокупности точек (ломаной Безье) длины NPLG;

XC, YC, ZC - векторы значений соответственно X-, Y- и Z-координат аппроксимирующей кривой Безье (длины NPTS).

Параметры NPLG и NPTS имеют то же значение, что и в программе BEZ2.

На рис. 5.8 показаны ломаные Безье и соответствующие им кривые Безье. Построение выполнялось с помощью следующего фрагмента программы.

```
PARAMETER(NL=5, NS=41, NL1=4, NS1=61)
REAL XC(NS),YC(NS), XP(NL),YP(NL)
REAL XC1(NS1),YC1(NS1), XP1(NL1),YP1(NL1)
DATA XP/0., 3., 10., 9., 4./,YP/0.,13.,17.,-6., 2./
DATA XP1/-2.,-9.,1.,-10./, YP1/-5.,7.,7.,-5./
CALL GRINIT
CALL BEZ2(XP, YP, NL, XC, YC, NS)
CALL BEZ2(XP1,YP1, NL1, XC1,YC1, NS1)
CALL PAGE(18.,14.,0, 0, 0)
CALL REGION(1.,1.,16.,12.,'КРИВАЯ БЕЗЬЕ',12, 1)
CALL LIMITS(-11.,11.,-7.,18.)
CALL AXES('X',1,0.,4,'Y',1,0.,4,0)
CALL LINEMO(XP, YP, NL, 2, 1)
CALL LINEO(XC, YC, NS)
CALL LINEMO(XP1,YP1, NL1,-7, 1)
CALL LINEO(XC1,YC1, NS1)
CALL ENDPG('5.8')
END
```

Для изображения пространственных кривых Безье можно воспользоваться программами, описанными в п. 8.1.3. При этом могут оказаться полезными также средства повышения наглядности изображения (см. п. 8.4.4, рис. 8.33).

5.9. Аппроксимация на основе B-сплайнов

Основу метода Безье составляет аппроксимация многочленами Бернштейна. Однако можно выбрать и другие наборы базисных функций, например *B-сплайны*, по отношению к которым базис Бернштейна является частным случаем. В сравнении с многочленами Бернштейна *B*-сплайны обладают следующими дополнительными преимуществами [16-18].

1. Аппроксимация *B*-сплайнами обеспечивает более точное приближение ломаной, чем аппроксимация многочленами Бернштейна.
2. При аппроксимации методом Безье на значения координат каждой точки кривой оказывают влияние все вершины ломаной Безье. Это затрудняет корректирование отдельных участков кривой. В то же время аппроксимация *B*-сплайнами обладает желательной локальностью. Этим способом можно производить локальные изменения кривой без полного пересчета.
3. Единственным путем увеличения (уменьшения) порядка кривой Безье является увеличение (уменьшение) числа вершин и соответствующей ломаной Безье. В методе *B*-сплайнов эти два параметра независимы и, следовательно, могут быть выбраны произвольно.

Кривая, построенная на основе *B*-сплайн-базиса, описывается следующим образом [16,18]:

$$\overline{p}(t) = \sum_{i=0}^{n} \overline{P}_i N_{ik}(t), \qquad (5.13)$$

где $\overline{p}(t)$ - радиус-вектор точек на кривой, $\overline{P}_i$ - вершины аппроксимируемой ломаной (всего вершин $n + 1$), а $N_{ik}(t)$ - весовая функция i-й нормализованной B-сплайн базисной кривой порядка k (т. е. степени k - 1), задаваемая рекуррентными соотношениями:

$$N_{i1}(t) = \begin{cases} 1, & \textit{если} \quad x_i \le t \le x_{i+1} \\ 0 & \textit{в} \qquad \textit{остальных} \quad \textit{случаях} \end{cases},$$

и

$$N_{ik}(t) = \frac{(t - x_i) N_{i,k-1}(t)}{x_{i+k-1} - x_i} - \frac{(x_{i+k} - t) N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}. \qquad (5.14)$$

Здесь x_i - элементы *узлового вектора*, а t - параметр, изменяющийся в диапазоне от 0 до $t_{\max} = n - k + 2$.

Узловой вектор, длина которого $n + k + 1$, вводится для учета собственной кривизны *B*-сплайн-кривых и представляет собой неубывающую последовательность целых чисел - *параметрических узлов*. Узловой вектор определяется числом точек в аппроксимируемой ломаной, порядком кривой, а также наличием сложных (кратных) узлов.

Из (5.13) и (5.14) следует, что *B*-сплайн-кривая является полиномом степени k -1 на каждом интервале (x_i, x_{i+1}) и что все ее производные до (k -2)-го порядка включительно непрерывны вдоль всей кривой. То есть эта кривая представляет собой сплайн-функцию порядка k (степени k -1).

На рис. 5.9 изображены несколько *B*-сплайн-кривых различного порядка, аппроксимирующих ломаную с шестью вершинами. При этом кривая пятого порядка является кривой Безье для заданной ломаной, кривая четвертого порядка - кубическим сплайном, а кривая второго порядка -

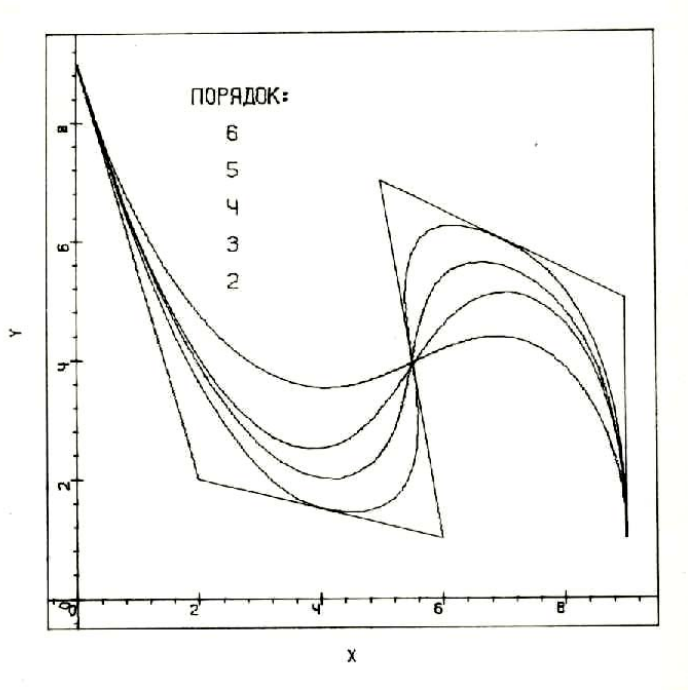

Рис. 5.9. Изображение нескольких *B*-сплайн-кривых различного порядка.

последовательностью связанных отрезков, совпадающих с исходной ломаной. Представляет интерес также кривая третьего порядка - она касается внутренних отрезков ломаной в их средних точках. Заметим, что с ростом порядка *B*-сплайн-кривые как бы спрямляются, их форма все в большей степени становится отличной от аппроксимируемой ломаной.

В формируемые кривые можно вносить изломы путем включения в соответствующую ломаную кратных вершин. Так, для образования излома в кривой третьего порядка необходима сдвоенная вершина (рис. 5.10). А чтобы создать излом в кривой четвертого порядка, потребуется вершина с кратностью три и т. д.

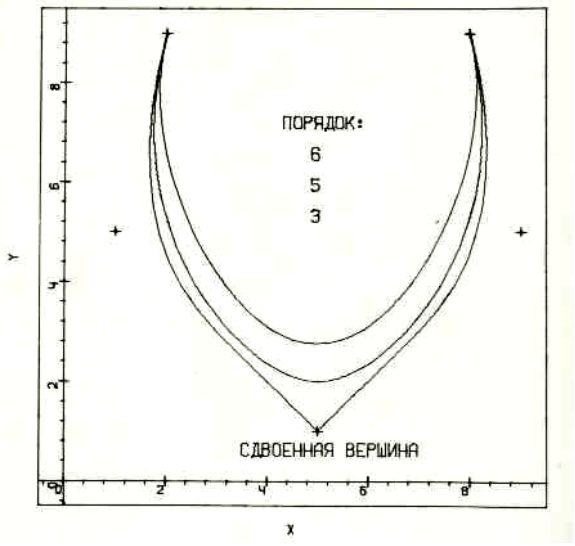

Рис. 5.10. *B*-сплайн-кривые со сдвоенной вершиной. В этой вершине в кривой третьего порядка образуется излом

Большое количество примеров, иллюстрирующих различные способы построения *B*-сплайн-кривых, читатель может найти в [16].

Аппроксимация *B*-сплайнами реализована в программах BSPLN2 и BSPLN3. Первая из них предназначена для формирования плоских, а вторая – пространственных *B*-сплайн-кривых.

Программа BSPLN2(K, XP, YP, NPLG, XC, YC, NPTS, WFUN, NWF, VKNOTS,

NKNOTS) позволяет вычислить значения координат точек плоской *B*-сплайн-кривой, аппроксимирующей заданную совокупность точек. Программа имеет следующие параметры:

K - порядок аппроксимирующей кривой (ее степень равна K - 1), $2 \le K \le NPLG$;

XP, YP - векторы значений соответственно X-, и Y-координат заданной совокупности точек (аппроксимируемой ломаной) длины NPLG;

NPLG - число точек в аппроксимируемой ломаной;

XC, YC - векторы значений соответственно X-, и Y-координат аппроксимирующей *B*-сплайн-кривой (длины NPTS);

NPTS - число точек в аппроксимирующей *B*-сплайн-кривой;

WFUN - двумерный рабочий массив, предназначенный для размещения весовых функций *B*-сплайн-базиса, размером (NWF, K);

NWF - максимальное значение, принимаемое первым индексом рабочего массива WFUN, NWF = NPLG + K - 1;

VKNOTS - одномерный рабочий массив, предназначенный для размещения элементов узлового вектора, длины NKNOTS;

NKNOTS - длина узлового вектора, NKNOTS = NPLG + K.

Программа BSPLN3(K, XP,YP, ZP, NPLG, XC,YC, ZC, NPTS, WFUN, NWF, VKNOTS, NKNOTS) позволяет вычислить значения координат точек пространственной *B*-сплайн-кривой, аппроксимирующей заданную совокупность точек. Параметры программы следующие:

XP, YP, ZP - векторы значений соответственно X-, Y- и Z-координат заданной совокупности точек аппроксимируемой ломаной длины NPLG;

XC, YC, ZC - векторы значений соответственно X- Y- и Z-координат аппроксимирующей *B*-сплайн-кривой длины NPTS.

Остальные параметры имеют то же значение, что и в программе BSPLN2.

В программах BSPLN2 и BSPLN3 используются служебные программы KNOTV2(K, XP, YP, NPLG, VKNOTS, NKNOTS) и KNOTV3(K, XP,YP, ZP, NPLG, VKNOTS, NKNOTS) с помощью которых строятся узловые векторы соответственно для двумерного и трехмерного случаев. Параметры этих программ имеют то же значение, что и в программах BSPLN2 и BSPLN3.

Ниже приведена программа, с помощью которой выполнялось построение рис. 5.10.

```
REAL XP(6),YP(6)
REAL XC1(41),YC1(41), XC2(41),YC2(41), XC3(41),YC3(41)
REAL WF1(8,3), WF2(10,5), WF3(11,6), VK(13)
DATA XP/2., 1., 2*5., 9., 8./
DATA YP/9., 5., 2*1., 5., 9./, NP, NS/6, 41/
CALL BSPLN2(3, XP,YP, NP, XC1,YC1, NS, WF1, 8, VK, 9)
CALL BSPLN2(5, XP,YP, NP, XC2,YC2, NS, WF2,10, VK, 11)
CALL BSPLN2(6, XP,YP, NP, XC3,YC3, NS, WF3,11, VK, 12)
CALL PAGE(18.,18., 0, 0, 0)
CALL REGION(1.,1.,16.,16.,0, 0,1)
CALL AXES('X',1, 0.,5,'Y',1, 0.,5, 0)
CALL LINEMO(XP,YP, NP, 1,-1)
CALL LINEO(XC1,YC1, NS)
CALL LINEO(XC2,YC2, NS)
CALL LINEO(XC3,YC3, NS)
CALL ENDPG('5.10')
END
```