第 3 章

插值算法与曲线拟合

数据分析在各个领域有着广泛的应用，尤其是在数学、物理等科学领域和工程领域的实际应用中，经常遇到进行数据分析的情况。例如，在工程领域根据有限的已知数据对未知数据进行推测时经常需要用到数据插值和拟合的方法。在物理或工程领域中则经常需要根据现实情况抽象出微分方程，进行数值求解等。这些常见的数据分析方法在 MATLAB 中都可以实现。

通过本章的内容读者可以了解到各命令的使用场合及内在关系，同时获得综合运用命令解决实际问题的思路和借鉴的经验。在本章中，将分别介绍插值、拟合等各种内容。

3.1 插值

插值(Interpolation)是指在所给定基准数据的情况下，研究如何平滑地估算出基准数据之间其他点的函数数值。每当其他点上函数数值获取的代价比较高时，插值就会发挥作用。

定义 3.1.1 设函数 $y=f(x)$ 在区间 $[a,b]$ 上有定义，且已知在点 $a\leqslant x_0<x_1<\cdots<x_n\leqslant b$ 上的值 $y_0,y_1,\cdots,y_n$，若存在一简单函数，使

$$P(x_i)=y_i,\quad i=0,1,\cdots,n \tag{3-1}$$

成立，则称 $P(x)$ 为 $f(x)$ 的插值函数，点 $x_0,x_1,\cdots,x_n$ 称为插值节点，包含插值节点的区间 $[a,b]$ 称为插值区间，求插值函数 $P(x)$ 的方法称为插值法。若 $P(x)$ 是次数不超过 n 的代数多项式，即 $P(x)=a_0+a_1x+\cdots+a_nx^n$，其中 a_i 为实数，则称 $P(x)$ 为插值多项式，$P(x)=a_0+a_1x+\cdots+a_nx^n$ 相应的插值法称为多项式插值。

定义 3.1.2 设给定数据点 (x_i,y_i)，$i=0,1,\cdots,n$(互异)，欲找二者的近似关系 $P(x)$，满足

(1) $P(x)\in P_n(x)$

(2) $P(x_i)=y_i$，$i=0,1,\cdots,n$

则称 $P(x)$ 为 n 次代数插值多项式。

定理 3.1.1 满足式(3-1)的插值多项式 $P(x)$ 是存在唯一的。

直接求解方程组就可以得到插值多项式 $P(x)$，但这是求插值多项式最繁杂的方法，一般不采用此方法，下面将给出构造插值多项式更简单的方法。

在 MATLAB 中提供了多种插值函数，这些插值函数在获得数据的平滑度、时间复杂度和空间复杂度方面有着完全不同的性能。在本章中，将主要介绍一维插值命令 interp1、二维插值命令 interp2 等 MATLAB 内置函数。同时，还将根据不同的插值方法自行编写 M 文件，完成不同的插值运算，例如拉格朗日插值、牛顿插值等，下面分别进行介绍。

3.1.1　一维插值

在 MATLAB 中，一维多项式插值的方法通过命令 interp1 实现，其具体的调用格式如下：

```
yi = interp1(x,Y,xi)    % 在该命令中,x 必须是向量,Y 可以是向量也可以是矩阵. 如果 Y 是向量,则
                        % 必须与变量 x 具有相同的长度,这时 xi 可以是标量,也可以是向量或者矩阵
yi = interp1(Y,xi)      % 在默认情况下,x 变量选择为 1:n,其中 n 是向量 Y 的长度
yi = interp1(x,Y,xi,method)            % 输入变量 method 用于指定插值的方法
yi = interp1(x,Y,xi,method,'extrap')   % 对超出数据范围的插值运算使用外推方法
yi = interp1(x,Y,xi,method,extrapval)  % 对超出数据范围的插值数据返回 extrapval 数值,一般
                                       % 为 NaN 或者 0
pp = interp1(x,Y,method,'pp')          % 返回数值 pp 为数据 Y 的分段多项式形式,method 用于
                                       % 指定产生分段多项式 pp 的方法
```

在 MATLAB 中，还提供了 interplq 命令作为 interp1 的快速命令，对于数据间隔不均的数据，interp1q 命令运行的速度比 interp1 命令要快，主要原因在于 interplq 命令没有对输入数据进行检测，关于该命令的详细使用方法可查看相应的帮助文件。

在上面的命令中，method 参数的取值和对应的含义如下：

```
nearest   % 最邻近插值方法(Nearest Neighbor Interpolation). 这种插值方法在已知数据的最邻近
          % 点设置插值点,对插值点的数值进行四舍五入,对超出范围的数据点返回 NaN
linear    % 线性插值(Linear Interpolation), 这是 interp1 命令中 method 的默认数值,该方法采用
          % 直线将相邻的数据点相连,对超出数据范围的数据点返回 NaN
spline    % 三次样条插值( Cubic Spline Interpolation ),该方法采用三次样条函数获取插值数据
          % 点,在已知点为端点的情况下,插值函数至少具有相同的一阶和二阶导数
pchip     % 分段三次厄米特多项式插值(Piecewise Cubic Hermite Interpolation)
cubic     % 三次多项式插值,与分段三次厄米特多项式插值方法相同
```

3.1.2　人口数量预测

为了让读者直观地看到上面各个不同插值方法的差别，下面举例演示不同方法的插值。用不同的方法来对基础数据进行插值运算，并比较各种方法的不同。

【例 3-1】 某城市在 1900—1990 年，每隔 10 年统计一次该城市的人口数量，得到的结果如下：

75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505 249.633

上面数据的单位是百万，以上面的数据为基础，对没有进行人口统计的年限的人口数量进行推测，分别使用不同的插值方法。

代码如下：

```
% chapter3/test1.m
t = 1900:10:1990
p = [75.995 91.972 105.711 123.203  131.669  150.697  179.323  203.212
226.505  249.633]

x = 1900:0.01:1990
y1 = interp1(t,p,x,'linear')
y2 = interp1(t,p,x,'spline')
y3 = interp1(t,p,x,'cubic')
y4 = interp1(t,p,x,'nearest')

subplot(2,1,1)
plot(t,p,'ko')
hold on
plot(x,y1,'r')
hold on
plot(x,y2,'b')
title('linear vs spline')
grid on

subplot(2,1,2)
plot(x,y3,'r')
hold on
plot(x,y4,'b')
title('cubic vs nearest')
grid on
```

运行结果如图 3-1 所示。

比较计算结果，代码如下：

```
% chapter3/test2.m
msg = '      year       linear   spline     cubic      nearest    '
for i = 0:8
    n = 8 * i
    year = 1904 + n
    pop(i + 1,1) = year
    pop(i + 1,2) = y1((year - 1900)/0.01 + 1)
    pop(i + 1,3) = y2((year - 1900)/0.01 + 1)
```

```
        pop(i+1,4) = y3((year-1900)/0.01+1)
        pop(i+1,5) = y4((year-1900)/0.01+1)
    end
    p = round(pop)
    disp(msg)
    disp(p)
```

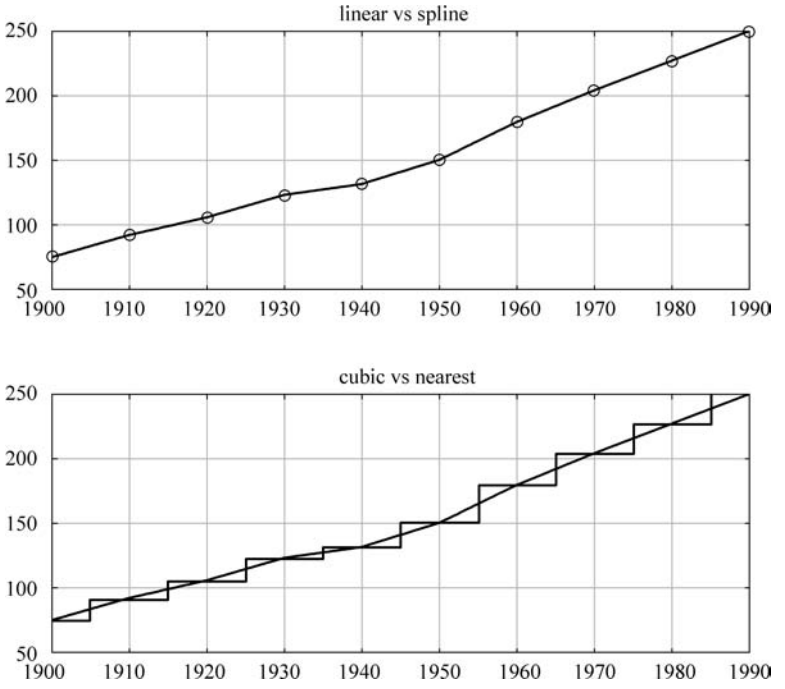

图 3-1　不同插值方法得到的结果

运行结果如下。从结果可以看出，使用不同的插值方法得到的预测数据会略有差别，上面的结果中 nearest 方法得到的结果是偏差最大的一种，其他方法则差别不大。

year	linear	spline	cubic	nearest
1904	82	84	83	76
1912	95	94	95	92
1920	106	106	106	106
1928	120	120	120	123
1936	128	128	128	132
1944	139	137	138	132
1952	156	156	156	151
1960	179	179	179	179
1968	198	199	199	203

根据上面的实例，下面简要比较各种插值方法在执行速度、占用内存大小及获得数据的平滑度方面的性能，如表 3-1 所示。

表 3-1 各种插值方法的比较

方 法	说 明
nearest	最快的插值方法，但是数据平滑方面最差，数据不是连续的
linear	执行速度较快，有足够的精度。最为常用，而且为默认设置
cubic	较慢，精度高，平滑度好，当希望得到平滑的曲线时可以使用该选项
spline	执行速度最慢，精度高，最平滑

3.1.3 二维插值

二维插值是高维插值的一种，主要应用于图像处理和数据的可视化，其基本思想与一维插值大致相同，它是对两个变量的函数 $z=f(x,y)$ 进行插值。

在 MATLAB 中，二维插值的常用函数是 interp2，其调用格式如下：

```
Z1 = interp2(X,Y,Z,XI,Y1)          % 原始数据 x、y 和 z 决定插值函数 z = f(x,y)，返回的数值 Z1 是
                                   % (X1,Y1)：在函数 z = f(x, y)上的数值
Z1 = interp2(Z, X1, Y1)            % 其中如果 Z 的维度是 n × m, 则有 x = 1:n, y = 1:m
Z1 = interp2(Z,ntimes)             % 在两点之间递归地进行插值 ntimes 次
Z1 = interp2(X,Y,Z,X1,Yl,method)   % 使用选定的插值方法进行二维插值
```

关于上面命令中的参数，应注意下面的内容：

对于其他高维插值的命令，使用格式与此大致相同。在上面的命令中，X、Y 和 Z 是进行插值的基准数据。X1、Y1 是待求插补函数值 Z1 的自变量对组，虽然随着维度的增加，插值的具体操作变得越来越复杂，但是基本原理和一维插值的情况相同，调用名称也很类似。

参数 X 和 Y 必须满足一定的条件，首先 X 和 Y 必须是同维矩阵，同时矩阵中的元素，无论是行向还是列向，都必须是单调排列。最后，X 和 Y 必须是 Plaid 格式的矩阵，所谓 Plaid 格式的矩阵，是指 X 矩阵每一行的元素依照单调次序排列，并且任何两行都是相同的；Y 矩阵每一列的元素依照单调次序排列，并且任何两列都是相同的。

对于 X1 和 Y1 数据系列，一般需要使用 meshgrid 命令来创建。具体的方法为给定两个变量的分向量，然后通过 meshgrid 命令生成对应的数据系列。

关于该命令的 method 参数，其含义和前面小节中 interp1 命令的 method 参数类似，只是参数数值 linear 表示的是双线性插值方法。

3.1.4 绘制二元函数图形

【例 3-2】 以二元函数 $z(x,y)=\dfrac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$ 为例，首先经过“仿真”获取基础数据，生

成一组基础的稀疏“测量数据”，然后使用二维插值得到更细致的数据，完成绘图。

代码如下：

```
%chapter3/test3.m
%绘制基础数据图形
x = -3*pi:3*pi
y = x
[X,Y] = meshgrid(x,y)
r = sqrt(X.^2 + Y.^2) + eps
F = sin(r)./r
[FX,FY] = gradient(F)
dzdr = sqrt(FX.^2 + FY.^2)
subplot(1,2,1)
surf(X,Y,F,abs(dzdr))
colormap(spring)
alphamap('rampup')
%进行二维插值运算
xi = linspace(-3*pi,3*pi,100)
yi = linspace(-3*pi,3*pi,100)
[XI,YI] = meshgrid(xi,yi)
ZI = interp2(X,Y,F,XI,YI,'cubic')
%绘制插值后的数据图形
subplot(1,2,2)
surf(XI,YI,ZI)
colormap(spring)
alphamap('rampup')
```

运行结果如图 3-2 所示。

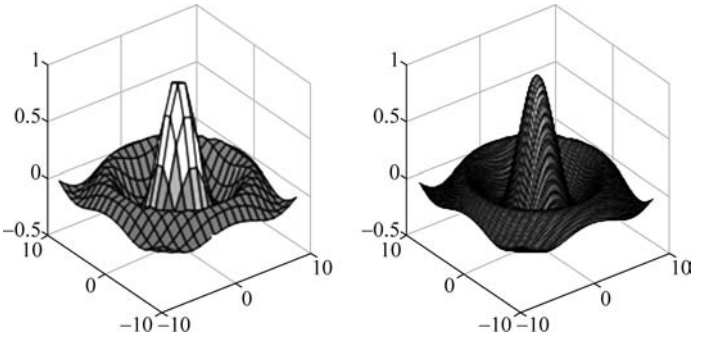

图 3-2　基础数据图形和插值后图形

【例 3-3】 在三维图形表面上添加等高线，使用二维插值的方法来计算等高线的数据点坐标，同时使用 streamslice 命令绘制三维图形表面等高线数据点的梯度。

代码如下：

```
%chapter3/test4.m
%生成数据
z = peaks
surf(z)
shading interp
hold on
%添加三维图形的等高线
[c ch] = contour3(z,20)
set(ch,'edgecolor','b')
%计算三维图形表面的数值梯度
[u v] = gradient(z)
h = streamslice( - u, - v)
set(h,'color','k')
%对等高线进行二维数值插值
for i = 1:length(h)
      zi = interp2(z,get(h(i),'xdata'),get(h(i),'ydata'))
    set(h(i),'zdata',zi)
end
view(30,50)
axis tight
```

运行结果如图 3-3 所示。

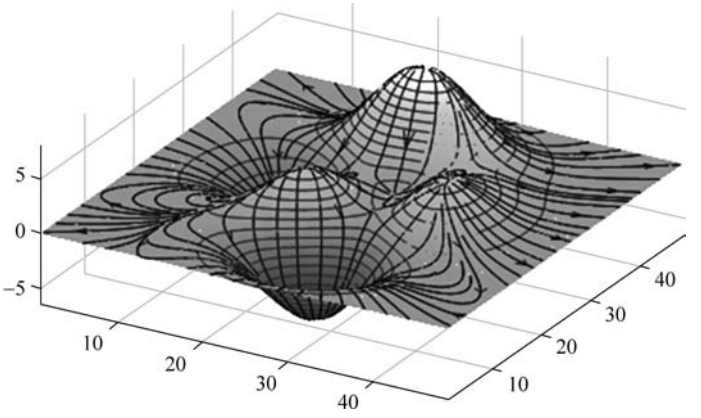

图 3-3　三维图形

在上面的两个例子中，多次用到了关于 MATLAB 图形对象操作的命令，主要涉及了图形颜色、角度等属性。

3.2　插值算法


41min

3.2.1　拉格朗日插值

拉格朗日(Lagrange)插值是 n 次多项式插值，其成功地用构造插值基函数的方法，解决了求 n 次多项式插值函数问题。拉格朗日插值的基本思想是将待求的 n 次多项式插

值函数改写成另一种表示方式，再利用插值条件确定其中的待定函数，从而求出插值多项式。

1. 基函数

为了构造插值多项式，我们先定义插值基函数。

定义 3.2.1　设 $x_0,x_1,\cdots,x_n$ 是给定的彼此互异的 $n+1$ 个插值结点，$y_i=f(x_i)$，$i=0,1,\cdots,n$ 为给出的函数值，则 $P_n(x)=y_0l_0(x)+y_1l_1(x)+\cdots+y_nl_n(x)$ 是唯一的次数不超过 n 的，并且满足 $P_n(x_i)=y_i$，$i=0,1,\cdots,n$ 的多项式。其中

$$l_i(x)=\frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

为拉格朗日插值基函数，$P_n(x)$ 为拉格朗日插值函数。

下面介绍基函数 $l_i(x)$ 的性质。

性质 3.2.1　$l_i(x_j)=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases}$

性质 3.2.2　$l_i(x)\in P_n(x)$，　$i=0,1,\cdots,n$

性质 3.2.3　$\sum\limits_{i=0}^{n}l_i(x)\equiv 1$

2. 拉格朗日插值公式

定理 3.2.1　n 次代数插值问题的解为 x_0,x_1 称为拉格朗日插值多项式。特殊化，得到如下插值公式。

1) 线性插值 $L_1(x)$

设已知 x_0,x_1 及 $y_0=f(x_0)$，$y_1=f(x_1)$，$L_1(x)$ 为 f 不超过一次多项式，且满足 $L_1(x_0)=y_0$，$L_1(x_1)=y_1$，几何上，$L_1(x)$ 为过 (x_0,y_0) 和 (x_1,y_1) 的直线，从而得到

$$L_1(x)=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0) \tag{3-2}$$

为了推广到高阶问题，我们将式(3-2)变成对称式：

$$L_1(x)=l_0(x)y_0+l_1(x)y_1 \tag{3-3}$$

其中，$l_0(x)=\dfrac{x-x_1}{x_0-x_1}$ 和 $l_1(x)=\dfrac{x-x_0}{x_1-x_0}$ 均为 1 次多项式且满足 $l_0(x)=1$ 且 $l_1(x)=0$ 或 $l_0(x)=0$ 且 $l_1(x)=1$。两关系式可统一写成

$$l_i(x_j)=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \tag{3-4}$$

2) 抛物线插值 $L_2(x)$

假定插值结点为 x_{i-1},x_i,x_{i+1}，要求抛物线插值(二次插值)多项式 $L_2(x)$，使它满足

$$L_2(x_j)=y_i,\quad j=i-1,i,i+1 \tag{3-5}$$

我们知道，在几何上就是通过三点 (x_{i-1},y_{i-1})、(x_i,y_i) 和 (x_{i+1},y_{i+1}) 确定抛物线。为了求出 $L_2(x)$ 的表达式，可采用基函数法，此时基函数 $l_{i-1}(x)$、$l_i(x)$ 及 $l_{i+1}(x)$ 是二次

函数，且在节点上分别满足条件

$$\begin{cases} l_{i-1}(x_{i-1})=1, & l_{i-1}(x_j)=0, \quad j=i,i+1 \\ l_i(x_i)=1, & l_i(x_j)=0, \quad j=i-1,i+1 \\ l_{i+1}(x_{i+1})=1, & l_{i+1}(x_j)=0, \quad j=i-1,i \end{cases} \tag{3-6}$$

满足条件式(3-6)的插值基函数可以很容易地求出，例如求 $l_{i-1}(x)$，因它有两个零点 x_i 及 x_{i+1}，故可表示为

$$l_{i-1}(x)=A(x-x_i)(x-x_{i+1}) \tag{3-7}$$

其中 A 为待定系数，可由条件 $l_{i-1}(x_{i-1})=1$ 定出：

$$A=\frac{1}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} \tag{3-8}$$

于是

$$l_{i-1}(x)=\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} \tag{3-9}$$

同理可得

$$l_i(x)=\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} \tag{3-10}$$

$$l_{i+1}(x)=\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} \tag{3-11}$$

利用二次插值基函数 $l_{i-1}(x)$、$l_i(x)$和 $l_{i+1}(x)$可立即得到二次插值多项式

$$L_2(x)=y_{i-1}l_{i-1}(x)+y_il_i(x)+y_{i+1}l_{i+1}(x) \tag{3-12}$$

显然，它满足条件 $L_2(x_j)=y_j(j=i-1,i,i+1)$。将上面求得的 $l_{i-1}(x)$、$l_i(x)$和 $l_{i+1}(x)$代入式(3-12)，得

$$\begin{aligned} L_2(x)=y_{i-1}&\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}+y_i\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}+ \\ &y_{i+1}\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} \end{aligned} \tag{3-13}$$

3. 余项与误差估计

拉格朗日插值用来求 n 个节点的$(n-1)$次插值多项式，它就是线性插值和抛物线插值的推广和延伸。我们设有 n 个节点，则拉格朗日插值的表达式表示为

$$\begin{aligned} g(x)&=\sum_{k=1}^{n}\frac{(x-x_1)(x-x_2)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_1)(x_k-x_2)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}y_k \\ &=\sum_{k=1}^{n}\left(\prod_{j=1,j\neq k}^{n}\frac{x-x_j}{x_k-x_j}\right)y_k \end{aligned} \tag{3-14}$$

若在$[a,b]$上用 $L_n(x)$近似 $f(x)$，则其截断误差为 $R_n(x)=f(x)-L_n(x)$，也称为

插值多项式的余项。关于插值余项估计有以下定理。下面的定理说明了用插值多项式 $L_n(x)$ 近似代数 $f(x)$ 时的余项。

定理 3.2.2(余项定理)　设 $f^{(n)}(x)$ 在 $[a,b]$ 上连续，$f^{(n+1)}(x)$ 在 (a,b) 内存在节点 $a\leqslant x_0<x_1<\cdots<x_n\leqslant b$，$L_n(x)$ 是满足条件 $L_n(x_j)=y_j$，$j=0,1,\cdots,n$ 的插值多项式，则对任何 $x\in[a,b]$，插值余项

$$R_n(x)=f(x)-L_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) \tag{3-15}$$

这里 $\xi\in(a,b)$ 且依赖于 x，$\omega_{n+1}(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$。

这里需要说明如下两点：

(1) 在插值多项式的余项公式中，包含 $f^{(n+1)}(\xi)$，其中，ξ 一般与插值结点 $x_j(j=0,1,\cdots,n)$ 及插值点 x 有关，因此，ξ 一般是无法知道的，这就对余项的估计带来了困难。

(2) 由插值多项式的余项公式可以看出，当被插值的函数 $f(x)$ 为次数不高于 n 的多项式时，其 n 次插值多项式就是它本身，因为此时 $f^{(n+1)}(\xi)=0$，即余项为 0。

应当指出，余项表达式只有在 $f(x)$ 的高阶导数存在时才能应用。ξ 在 (a,b) 内的具体位置通常不可能给出，如果我们可以求出 $\max\limits_{a\leqslant x\leqslant b}|f^{(n+1)}(x)|=M_{n+1}$，则插值多项式 $L_n(x)$ 逼近 $f(x)$ 的截断误差限是

$$|R_n(x)|\leqslant\frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)| \tag{3-16}$$

当 $n=1$ 时，线性插值余项为

$$R_1(x)=\frac{1}{2}f''(\xi)\omega_2(x)=\frac{1}{2}f''(\xi)(x-x_0)(x-x_1),\quad \xi\in[x_0,x_1] \tag{3-17}$$

当 $n=2$ 时，抛物线插值的余项为

$$R_2(x)=\frac{1}{6}f'''(\xi)(x-x_0)(x-x_1)(x-x_2),\quad \xi\in[x_0,x_2] \tag{3-18}$$

利用余项表达式(3-15)，当 $f(x)=x^k(k\leqslant n)$ 时，由于 $f^{(n+1)}(x=0)$，于是有

$$R_n(x)=x^k-\sum_{i=0}^{n}x_i^k l_i(x)=0 \tag{3-19}$$

由此得

$$\sum_{i=0}^{n}x_i^k l_i(x)=x^k,\quad k=0,1,\cdots,n \tag{3-20}$$

特别当 $k=0$ 时，有

$$\sum_{i=0}^{n}l_i(x)=1 \tag{3-21}$$

式(3-20)和式(3-21)也是插值基函数的性质。

4. 拉格朗日插值的 MATLAB 实现

常用的几个函数：

1）poly 函数

调用格式：Y＝poly(V)返回矩阵特征多项式的系数。

代码如下：

```
>> Y = poly(3);
```

结果为

```
1   -3 %即其特征多项式为 x-3
```

2）polyval 函数

调用格式：Y＝polyval(p,x) 返回 n 次多项式 p 在 x 处的值。输入变量 p＝[p0　p1　p2　…　pn]是一个长度为 n＋1 的向量，其元素为按降排列的多项式系数。

如对多项式 $p(x)=3x^2+2x+1$，计算当 $x=5$、7、9 时的值。

代码如下：

```
>> p = [3 2 1]
   x = [5,7,9]
   polyval(p,x)
```

结果为

```
ans =
86    162    262
```

3）poly2sym 函数

调用格式一：poly2sym(C)把多项式的系数向量转化为符号多项式。

代码如下：

```
>> c = [1 -3]
   y = poly2sym(c)
```

结果为

```
y = x-3
```

调用格式二：f1＝poly2sym(C,'V') 或 f2＝poly2sym(C,sym('V'))。

代码如下：

```
     >> c = [1 -3];y = poly2sym(c,'s')
或者 >> c = [1 -3];y = poly2sym(c,sym('s'))
```

结果为

```
y = s - 3
```

拉格朗日插值多项式和基函数的主程序代码如下：

```
% chapter3/test5.m
function [C, L,L1,l] = lagran1(X,Y)
m = length(X); L = ones(m,m);
for k = 1: m
V = 1;
for i = 1:m
if k~ = i
V = conv(V,poly(X(i)))/(X(k) - X(i));
end
end
L1(k,:) = V; l(k,:) = poly2sym (V)
end
C = Y * L1;L = Y * l
return
```

【例 3-4】 给出节点数据 $f(-2.15)=17.03$，$f(-1.00)=7.24$，$f(0.01)=1.05$，$f(1.02)=2.03$，$f(2.03)=17.06$，$f(3.25)=23.05$，作五次拉格朗日插值多项式和基函数，并写出估计其误差的公式，代码如下：

```
>>  X = [ - 2.15   - 1.00   0.01   1.02   2.03   3.25];
    Y = [17.03   7.24   1.05   2.03   17.06   23.05];
    [C, L ,L1,l] = lagran1(X,Y)
```

输入的量：$n+1$ 个节点 $(x_i,y_i)(i=1,2,\cdots,n+1)$ 横坐标向量 $\boldsymbol{X}$，纵坐标向量 $\boldsymbol{Y}$；

输出的量：n 次拉格朗日插值多项式 L 及其系数向量 $\boldsymbol{C}$，基函数 l 及其系数矩阵 $\boldsymbol{L}_1$。

运行后输出五次拉格朗日插值多项式 L 及其系数向量 $\boldsymbol{C}$，基函数 l 及其系数矩阵 $\boldsymbol{L}_1$ 如下：

```
C =  - 0.2169     0.0648     2.1076     3.3960    - 4.5745     1.0954
L = 1.0954 - 4.5745 * x + 3.3960 * x^2 + 2.1076 * x^3 + 0.0648 * x^4 - 0.2169 * x^5
```

拉格朗日插值及其误差估计的 MATLAB 程序代码如下：

```
% chapter3/test6.m
function [y,R] = lagranzi(X,Y,x,M)
n = length(X); m = length(x);
```

```
for i = 1:m
    z = x(i);s = 0.0;
    for k = 1:n
        p = 1.0; q1 = 1.0; c1 = 1.0;
for j = 1:n
            if j~ = k
p = p * (z - X(j))/(X(k) - X(j));
            end
        q1 = abs(q1 * (z - X(j)));c1 = c1 * j;
        end
        s = p * Y(k) + s;
    end
    y(i) = s;
end
R = M * q1/c1;
return
```

【例 3-5】 已知 $\sin30^\circ=0.5$，$\sin45^\circ=0.7071$，$\sin60^\circ=0.8660$，用拉格朗日插值及其误差估计的 MATLAB 函数求 $\sin40^\circ$的近似值，并估计其误差。

代码如下：

```
 % chapter3/test7.m
>> x = 2 * pi/9; M = 1;
      X = [pi/6 ,pi/4, pi/3];
      Y = [0.5,0.7071,0.8660];
      [y,R] = lagranzi(X,Y,x,M)
```

输入的量：$\boldsymbol{X}$ 是 $n+1$ 个节点$(x_i,y_i)(i=1,2,\cdots,n+1)$的横坐标向量，$\boldsymbol{Y}$ 是纵坐标向量，x 是以向量形式输入的 m 个插值点，M 在$[a,b]$上满足$|f^{(n+1)}(x)|\leqslant M$。

输出的量：$\boldsymbol{y}$ 为 m 个插值构成的向量，R 是误差限。

运行后输出的插值 $\boldsymbol{y}$ 及其误差限 R 为

```
        y =                     R =
                0.6434                  8.8610e-004
```

23min

3.2.2 牛顿插值

拉格朗日插值的优点是插值多项式特别容易建立，缺点是增加节点时原有多项式不能利用，必须重新建立，即所有基函数都要重新计算，这就造成了计算量的浪费；牛顿(Newton)插值多项式是代数插值的另一种表现形式，当增加节点时它具有所谓的“承袭性”，这要用到差商的概念。在本节中，将详细介绍如何使用牛顿插值方法进行数据插值。

1. 差商的定义与性质

定义 3.2.2　已知函数 $f(x)$ 的 $n+1$ 个插值点为 (x_i,y_j)，$y_i=f(x_i)$，$i=0,1,\cdots,n$，$\dfrac{f(x_i)-f(x_j)}{x_i-x_j}$ 称为 $f(x)$ 在点 (x_i,y_j) 的一阶差商，记为 $f[x_i,y_j]$，即

$$f[x_i,y_j]=\frac{f(x_i)-f(x_j)}{x_i-x_j} \tag{3-22}$$

一阶差商的差商 $\dfrac{f[x_i-x_j]-f[x_j,x_k]}{x_i-x_k}$ 称为 $f(x)$ 在点 x_i,x_j,x_k 的二阶差商，记为 $f[x_i,x_j,x_k]$，即

$$f[x_i,x_j,x_k]=\frac{f[x_i-x_j]-f[x_j,x_k]}{x_i-x_k} \tag{3-23}$$

一般地，$k-1$ 阶差商的差商 $\dfrac{f[x_0,x_1,\cdots,x_{k-1}]-f[x_1,x_2,\cdots,x_k]}{x_0-x_k}$ 称为 $f(x)$ 在点 x_0，$x_1,\cdots,x_k$ 的 k 阶差商，记为 $f[x_0,x_1,\cdots,x_k]$，即

$$f[x_0,x_1,\cdots,x_k]=\frac{f[x_0,x_1,\cdots,x_{k-1}]-f[x_1,x_2,\cdots,x_k]}{x_0-x_k} \tag{3-24}$$

差商具有以下性质。

性质 3.2.4　n 阶差商可以表示成 $n+1$ 个函数值 $f(x_0),f(x_1),\cdots,f(x_n)$ 的线性组合，即

$$f[x_0,x_1,\cdots,x_k]=\sum_{i=0}^{n}\frac{f(x_i)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \tag{3-25}$$

事实上，由式(3-25)，当 $n=1$ 时

$$f[x_0,x_1]=\frac{f(x_0)-f(x_1)}{x_0-x_1}=\frac{f(x_0)}{x_0-x_1}+\frac{f(x_1)}{x_1-x_0} \tag{3-26}$$

当 $n=2$ 时

$$\begin{aligned}
f[x_0,x_1,x_2]&=\frac{f[x_0,x_1]-f[x_1,x_2]}{x_1-x_2}\\
&=\frac{f[x_0,x_1]}{x_0-x_1}+\frac{f[x_1,x_2]}{x_2-x_0}\\
&=\frac{1}{x_0-x_2}\left(\frac{f(x_0)}{x_0-x_1}+\frac{f(x_1)}{x_1-x_0}\right)+\frac{1}{x_2-x_0}\left(\frac{f(x_1)}{x_1-x_2}+\frac{f(x_2)}{x_2-x_1}\right)\\
&=\frac{f(x_0)}{(x_0-x_1)(x_0-x_2)}+\frac{f(x_1)}{x_0-x_2}\left(\frac{1}{x_1-x_0}-\frac{1}{x_1-x_2}\right)+\\
&\quad\frac{f(x_2)}{(x_2-x_0)(x_2-x_1)}
\end{aligned}$$

$$=\frac{f(x_0)}{(x_0-x_1)(x_0-x_2)}+\frac{f(x_1)}{(x_1-x_0)(x_1-x_2)}+\frac{f(x_2)}{(x_2-x_0)(x_2-x_1)} \tag{3-27}$$

一般地有

$$f[x_0,x_1,\cdots,x_k]=\sum_{i=0}^{n}\frac{f(x_i)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \tag{3-28}$$

性质 3.2.5 （对称性）差商与节点的顺序无关，如

$$\begin{cases}f[x_0,x_1]=f[x_1,x_0]\\ f[x_0,x_1,x_2]=f[x_1,x_0,x_2]=f[x_0,x_2,x_1]\end{cases} \tag{3-29}$$

这一点可以从性质 3.2.4 看出。

性质 3.2.6 若 $f(x)$是 x 的 n 次多项式，则一阶差商 $f[x,x_0]$是 x 的 $n-1$ 次多项式，二阶差商 $f[x,x_0,x_1]$是 x 的 $n-2$ 次多项式；一般地，函数 $f(x)$的 k 阶差商 $f[x,x_0,\cdots,x_{k-1}]$是 x 的 $n-k$ 次多项式($k\leqslant n$)，而 $k>n$ 时，k 阶差商为零。

2. 牛顿插值多项式

设 x 是$[a,b]$上任一点，则由 $f(x)$的一阶差商定义式 $f[x,x_0]=\dfrac{f(x)-f(x_0)}{x-x_0}$得

$$f(x)=f(x_0)+f[x,x_0](x-x_0) \tag{3-30}$$

同理由 $f(x)$的二阶差商定义式 $f[x,x_0,x_1]=\dfrac{f[x,x_0]-f[x,x_1]}{x-x_1}$得

$$f[x,x_0]=f[x_0,x_1]+f[x,x_0,x_1](x-x_1) \tag{3-31}$$

一般地，由 $f(x)$的 $n+1$ 阶差商定义式

$$f[x_0,x_1,\cdots,x_n]=\frac{f[x_0,x_1,\cdots,x_{n-1}]-f[x_1,x_2,\cdots,x_n]}{x_0-x_n} \tag{3-32}$$

有

$$f[x,x_0,x_1,\cdots,x_{n-1}]=f[x_0,x_1,\cdots,x_n]+f[x,x_0,x_1,\cdots,x_n](x-x_n) \tag{3-33}$$

即可得到一系列等式：

$$\begin{cases}f(x)=f(x_0)+f[x,x_0](x-x_0)\\ f[x,x_0]=f[x_0,x_1]+f[x,x_0,x_1](x-x_1)\\ f[x,x_0,x_1]=f[x_0,x_1,x_2]+f[x,x_0,x_1,x_2](x-x_2)\\ f[x,x_0,x_1,\cdots,x_{n-1}]=f[x_0,x_1,\cdots,x_n]+f[x,x_0,x_1,\cdots,x_n](x-x_n)\end{cases} \tag{3-34}$$

依次将后式代入前式，最后得

$$\begin{aligned}f(x)=\ & f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots+\\ & f[x_0,x_1,\cdots,x_n](x-x_0),\cdots,(x-x_{n-1})+\\ & f[x,x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_n)\end{aligned} \tag{3-35}$$

式(3-35)可以写为

$$f(x)=N_n(x)+R_n(x)$$

其中，

$$N_n(x)=f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots+f[x_0,x_1,\cdots,x_n](x-x_0),\cdots,(x-x_{n-1}) \tag{3-36}$$

$$R_n(x)=f[x,x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_n) \tag{3-37}$$

可以看出，$N_n(x)$是关于 x 的次数不超过 n 的多项式，并且当 $x=x_i$ 时，有

$$R_n(x_i)=0,\quad i=0,1,\cdots,n \tag{3-38}$$

因而有

$$N_n(x_i)=f(x_i),\quad i=0,1,\cdots,n \tag{3-39}$$

即 $N_n(x)$满足插值条件，称为牛顿插值多项式，再由插值多项式的唯一性可知，$L_n(x)=N_n(x)$，因而两个多项式对应的余项是相等的，即

$$f[x,x_0,x_1,\cdots,x_n]\omega_{n+1}(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) \tag{3-40}$$

由此得到差商与导数的关系。

性质 3.2.7　若 $f(x)$在$[a,b]$上存在 n 阶导数，且 $x_i\in[a,b](i=0,1,\cdots,n)$，则

$$f[x_0,x_1,\cdots,x_n]=\frac{f^{(n)}(\xi)}{n!},\quad \xi\in[a,b] \tag{3-41}$$

3. 牛顿插值的 MATLAB 实现

1) 牛顿插值多项式、差商和误差公式的 MATLAB 程序

输入参数：$n+1$ 个节点$(x_i,y_i)(i=1,2,\cdots,n+1)$的横坐标向量 $\boldsymbol{X}$ 和纵坐标向量 $\boldsymbol{Y}$。

输出参数：n 阶牛顿插值多项式 L 及其系数向量 $\boldsymbol{C}$，差商的矩阵 $\boldsymbol{A}$，插值余项公式 wcgs 及其$\dfrac{R_n(x)}{f^{(n+1)}(\xi)}$(其中 $\xi\in(\min\limits_{1\leqslant i\leqslant n+1}(x_i),\max\limits_{1\leqslant i\leqslant n+1}(x_i))$)的系数向量 $\boldsymbol{C}_{\mathrm{W}}$。

求牛顿插值多项式和差商的代码如下：

```
% chapter3/test8.m
function [A,C,L,wcgs,Cw] = newpoly(X,Y)
n = length(X); A = zeros(n,n); A(:,1) = Y';
  s = 0.0; p = 1.0; q = 1.0; c1 = 1.0;
      for  j = 2:n
                for i = j:n
              A(i,j) = (A(i,j-1) - A(i-1,j-1))/(X(i) - X(i-j+1));
                end
              b = poly(X(j-1));q1 = conv(q,b); c1 = c1 * j;  q = q1;
      end
```

```
  C = A(n,n); b = poly(X(n)); q1 = conv(q1,b);
for k = (n - 1): - 1:1
C = conv(C,poly(X(k))); d = length(C); C(d) = C(d) + A(k,k);
end
L(k,:) = poly2sym(C); Q = poly2sym(q1);
syms M
wcgs = M * Q/c1; Cw = q1/c1;
return
```

【例 3-6】 给出节点数据 $f(-2.15)=17.03$，$f(-1.00)=7.24$，$f(0.01)=1.05$，$f(1.02)=2.03$，$f(2.03)=17.06$，$f(3.25)=23.05$ 作五阶牛顿插值多项式和差商，并写出估计误差的公式。

代码如下：

```
% chapter3/test8.m
X = [ - 2.15   - 1.00   0.01   1.02   2.03   3.25];
Y = [17.03   7.24   1.05   2.03   17.06   23.05];
[A,C,L,wcgs,Cw] = newpoly (X,Y)
```

运行结果如下：

```
A =

    17.0300         0         0         0         0         0
     7.2400   - 8.5130         0         0         0         0
     1.0500   - 6.1287    1.1039         0         0         0
     2.0300    0.9703    3.5144    0.7604         0         0
    17.0600   14.8812    6.8866    1.1129    0.0843         0
    23.0500    4.9098   - 4.4715   - 3.5056   - 1.0867   - 0.2169


C =

    - 0.2169    0.0648    2.1076    3.3960   - 4.5745    1.0954
L =

 - (7813219284746629 * x^5)/36028797018963968 +
(583849564517807 * x^4)/9007199254740992 +
(593245028711263 * x^3)/281474976710656 +
(3823593773002357 * x^2)/1125899906842624 -
(321902673270315 * x)/70368744177664 + 308328649211299/281474976710656

wcgs =
```

```
(M*(x^6 - (79*x^5)/25 - (14201*x^4)/2500 +
(4934097026900981*x^3)/281474976710656 +
(154500712237335*x^2)/35184372088832 -
(8170642380559269*x)/562949953421312 +
5212760744134241/36028797018963968))/720


Cw =

    0.0014   -0.0044   -0.0079    0.0243    0.0061   -0.0202    0.0002
```

2）牛顿插值及其误差估计的 MATLAB 程序

输入参数：$n+1$ 个节点$(x_i, y_i)(i=1,2,\cdots,n+1)$的横坐标向量 $\boldsymbol{X}$ 和纵坐标向量$\boldsymbol{Y}$，x 是以向量形式输入的 m 个插值点，M 在$[a,b]$上满足$|f^{(n+1)}(x)|\leqslant M$。

输出参数：向量 $\boldsymbol{x}$ 的插值向量 $\boldsymbol{y}$ 及其误差限 R。

牛顿插值及其误差估计的代码如下：

```
% chapter3/test9.m
function [y,R] = newcz(X,Y,x,M)
n = length(X); m = length(x);
for t = 1:m
    z = x(t); A = zeros(n,n);A(:,1) = Y';
s = 0.0; p = 1.0; q1 = 1.0; c1 = 1.0;
        for  j = 2:n
          for i = j:n
           A(i,j) = (A(i,j-1) - A(i-1,j-1))/(X(i) - X(i-j+1));
          end
         q1 = abs(q1*(z - X(j-1)));c1 = c1*j;
        end
        C = A(n,n);q1 = abs(q1*(z - X(n)));
for k = (n-1):-1:1
C = conv(C,poly(X(k)));d = length(C); C(d) = C(d) + A(k,k);
end
  y(k) = polyval(C, z);
end
R = M*q1/c1;
return
```

【例 3-7】 已知 $\sin30^\circ=0.5$，$\sin45^\circ=0.7071$，$\sin60^\circ=0.8660$，用牛顿插值法求 $\sin40^\circ$ 的近似值，并且估计其误差。

代码如下：

```
%chapter3/test9.m
x=2*pi/9;
M=1;
X=[pi/6 ,pi/4, pi/3];
Y=[0.5,0.7071,0.8660];
[y,R]= newcz(X,Y,x,M)
```

运行结果如下：

```
y =
    0.6434
R =
    8.8610e-04
```

3）牛顿插值法的 MATLAB 综合程序

输入参数：$n+1$ 个节点$(x_i,y_i)(i=1,2,\cdots,n+1)$的横坐标向量 $\boldsymbol{X}$ 和纵坐标向量$\boldsymbol{Y}$，x 是以向量形式输入的 m 个插值点，M 在$[a,b]$上满足$|f^{(n+1)}(x)|\leqslant M$。

输出参数：向量 $\boldsymbol{x}$ 的插值向量 $\boldsymbol{y}$ 及其误差限 R，n 次牛顿插值多项式 L 及其系数向量 $\boldsymbol{C}$，差商的矩阵 $\boldsymbol{A}$。

代码如下：

```
%chapter3/test10.m
function [y,R,A,C,L]=newdscg(X,Y,x,M)
n=length(X); m=length(x);
for t=1:m
    z=x(t); A=zeros(n,n);A(:,1)=Y';
s=0.0; p=1.0; q1=1.0; c1=1.0;
          for  j=2:n
            for i=j:n
              A(i,j)=(A(i,j-1)- A(i-1,j-1))/(X(i)-X(i-j+1));
            end
             q1=abs(q1*(z-X(j-1)));c1=c1*j;
          end
           C=A(n,n);q1=abs(q1*(z-X(n)));
for k=(n-1):-1:1
C=conv(C,poly(X(k)));
d=length(C);C(d)=C(d)+A(k,k);
end
        y(k)= polyval(C, z);
end
R=M*q1/c1;L(k,:)=poly2sym(C);
return
```

【例 3-8】 给出节点数据 $f(-4.00)=27.00$，$f(0.00)=1.00$，$f(1.00)=2.00$，

$f(2.00)=17.00$ 作三阶牛顿插值多项式，计算 $f(-2.345)$，并估计误差。

代码如下：

```
%chapter3/test10.m
X=[-4,0,1,2];
Y =[27,1,2,17];
x= -2.345;
[y,R,A,C,P]=newdscg(X,Y,x,1)
```

运行结果如下：

```
y =

    22.3211
R =
2.3503
A =

    27.0000         0        0        0
     1.0000   -6.5000        0        0
     2.0000    1.0000   1.5000        0
    17.0000   15.0000   7.0000   0.9167
C =
0.9167    4.2500   -4.1667    1.0000
P =
(11*x^3)/12 + (17*x^2)/4 - (25*x)/6 + 1
```

3.2.3　分段插值法

许多实际问题希望插值函数具有较高阶的整体光滑性。分段高次拉格朗日插值和牛顿插值等是做不到的，在插值结点上它们只能保证插值函数连续。用多项式作为插值函数来逼近某一函数 $f(x)$ 是最简单易行的一种插值方法，但是插值多项式的次数是随着插值节点的数目而增加的，且次数高的插值多项式往往插值效果并不理想，会出现所谓的 Runge 现象，即在插值函数 $p_n(x)$ 的两端会发生激烈振荡，如图 3-4 所示。

增加拉格朗日插值多项式的阶数，并不能显著地减少插值的误差，因此，牛顿插值在五阶以上就很少使用了。为此，在实际应用中常采用分段插值方法。所谓分段插值法就是将被插值函数逐段多项式化，构造一个分段多项式作为插值函数。

首先，将插值区间划分为若干小段，在每一小段上使用低阶插值，然后将各小段上的插值多项式拼接在一起作为整个区间上的插值函数。如果使用的低阶插值为线性插值（两点插值），则将拼接成一条折线，用它来逼近函数 $f(x)$。

1. 分段线性插值

将插值区间 $[a,b]$ 分成

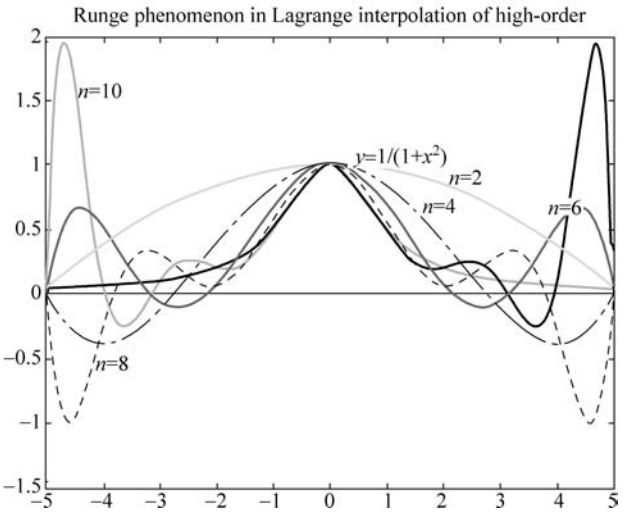

图 3-4 Runge 现象

$$a = x_0, x_1, \cdots, x_n = b \tag{3-42}$$

n 个小段，在每个小段$(x_i y_i)(i=1,2,\cdots,n+1)$上，其分段线性插值的公式为

$$s(x) = y_i + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_i) \tag{3-43}$$

根据

$$i = \begin{cases} 1, & x \leqslant x_0 \\ k, & x_{k-1} < x \leqslant x_k (1 \leqslant k \leqslant n) \\ n, & x > x_n \end{cases} \tag{3-44}$$

选择插值节点，即当插值节点为 $\boldsymbol{J}_{\mathrm{W}} = [\boldsymbol{A\theta} - \boldsymbol{y}]^{\mathrm{T}} \boldsymbol{W} [\boldsymbol{A\theta} - \boldsymbol{y}]$时，依次从左至右取出各节点。如果插值点 x 不超过节点 x_1（在 $[x_0, x_1]$），则取节点 x_0 和 x_1 进行线性插值，否则，再检查 x 是否超过 x_2，……，依次逐步检查。一旦发现 x 不超过某个节点 X_n，则取它与前面一个节点 x_{n-1} 进行线性插值。如果 x 已超过 x_{n-1}，则不论是否超过 x_n，插值节点均取 x_n 和 x_{n-1}（一律当成在 $[x_{n-1}, x_n]$）范围内取插值点。

在小段 $[x_{n-1}, x_n]$ 上，分段线性插值的误差为

$$|R(x)| = |f(x) - s(x)| \leqslant \frac{|f^{(2)}(\xi)|}{8}(x_n - x_{n-1})^2, \quad \xi \in [x_{n-1}, x_n] \tag{3-45}$$

可见，当 $f^{(2)}$ 有界时，小段 $[x_{n-1}, x_n]$ 越小，分段线性插值的误差就越小。用分段线性插值方法提高插值精度是有效的。

2. 分段线性插值图形的 MATLAB 程序

一维插值函数 interp1，格式如下：

```
yi = interp1(x,y,xi,'method')
```

yi：xi 处的插值结果
x,y：插值节点
xi：被插值点
method：插值方法
nearest：最邻近插值
linear：线性插值(缺省)
spline：三次样条插值
cubic：立方插值
缺省时分段线性插值。

【例 3-9】 用三次样条插值选取 11 个基点计算 $g(x)=\dfrac{1}{1+x^2}$，$-5\leqslant x\leqslant 5$ 的插值。

代码如下：

```
% chapter3/test11.m
x0 = linspace( - 5,5,11);
y0 = 1./(1 + x0.^2);
x = linspace( - 5,5,100);
y = interp1(x0,y0,x,'spline');
x1 = linspace( - 5,5,100);
y1 = 1./(1 + x1.^2);
plot(x1,y1,'k',x0,y0,' + ',x,y,'r')
```

运行结果如图 3-5 所示。可以看出插值函数得到的函数图像与原函数图像很接近。分段线性插值收敛性较好，但在节点处光滑不足，为使节点处光滑，可采用样条插值。

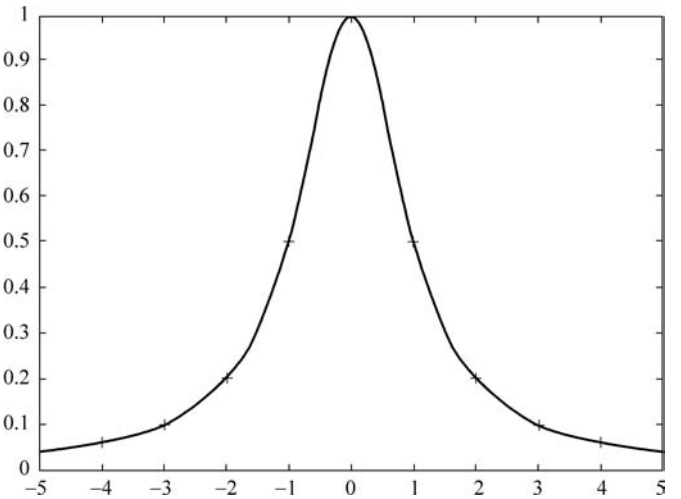

图 3-5　三次样条插值图形

3.2.4 样条插值

样条插值是用分段低次多项式去逼近被逼近函数，并且能满足对光滑性的要求，又无须给出每个节点处的导数值。它除了要求给出各个节点处的函数值外，只需提供两条边界节点处的导数信息。三次样条又是其中用得较为广泛的一种，这里重点讲解三次样条插值。

三次样条函数的定义：

设在区间$[a,b]$上取$n+1$个节点$a=x_0<x_1<\cdots<x_n=b$，函数$y=f(x)$在各个节点处的函数值为$y_i=f(x_i)$，$i=0,1,\cdots,n$若$S(x)$满足：

(1) $S(x_i)=y_i$，$i=0,1,\cdots,n$。

(2) 在区间$[a,b]$上，$S(x)$具有连续的二阶导数。

(3) 在区间$[x_i,x_{i-1}]$($i=0,1,\cdots,n-1$)上，$S(x)$是x三次的多项式。

则称$S(x)$是函数$y=f(x)$在区间$[a,b]$上的三次样条插值函数。

由以上定义可以看出，虽然每个子区间上的多项式可以各不相同，但在相邻子区间的连接处却是光滑的，因此，样条插值也称为分段光滑插值。

从定义可以求出$S(x)$，在每个小区间$[x_i,x_{i-1}]$($i=0,1,\cdots,n-1$)上确定4个待定系数，共有n个小区间，故应有$4n$个参数。

根据$S(x)$在$[a,b]$上二阶导数连续，在节点$i=1,2,\cdots,n-1$处满足连续性条件

$$\begin{cases} S(x_i-0)=S(x_i+0) \\ S'(x_i-0)=S'(x_i+0) \\ S''(x_i-0)=S''(x_i+0) \end{cases} \tag{3-46}$$

共有$3n-3$个条件，再加上$S(x)$满足插值条件$S(x_i)=y_i$，$i=0,1,\cdots,n$共有$4n-2$个条件，因此还需要2个条件才能确定$S(x)$。

通常可在区间端点上各加一个条件(称为边界条件)，可根据实际问题的要求给定，通常有以下3种：

(1) 已知端点的一阶导数值，即

$$S'(x_0)=f'_0,\quad S'(x_n)=f'_n \tag{3-47}$$

(2) 两个端点的二阶导数已知，即

$$S''(x_0)=f''_0,\quad S''(x_n)=f''_n \tag{3-48}$$

其特殊情况$S''(x_0)=S''(x_n)=0$，称为自然边界条件。

(3) 当$f(x)$是以x_n-x_0为周期的函数时，则要求$S(x)$也是周期函数。这时边界条件应满足

$$\begin{cases} S(x_i-0)=S(x_i+0) \\ S'(x_i-0)=S'(x_i+0) \\ S''(x_i-0)=S''(x_i+0) \end{cases} \tag{3-49}$$

而此时$y_0=y_n$。这样确定的样条函数$S(x)$称为周期函数。

MATLAB 函数库中有现成的三次样条函数 spline 及 interp1，格式如下：

```
YI = spline (X,Y,XI)
YI = spline (X,[df0,Y,dfn],XI)
YI = spline (X,Y)
yy = ppval(YI,xx)
```

在上面的函数中，(X,Y)表示样点数据，xx 则是插值的数据点系列。下面将利用简单的例子来讲解 spline 函数的使用方法。

【例 3-10】 对基础数据进行样条插值运算，分别使用样条函数和一维插值命令，并对插值结果进行比较。

代码如下：

```
% chapter3/test12.m
x = -4:4
y = [0 0.15 1.12 2.36 2.36 1.46 0.49 0.06 0]
plot(x,y)
hold on
cs = spline(x,[0 y 0])
xx = linspace(-4,4,101)
yy = ppval(cs,xx)
yyt = interp1(x,y,xx,'spline')
plot(xx,yy,'k.')
hold on
plot(xx,yyt,'m')
grid on
```

运行结果如图 3-6 所示。

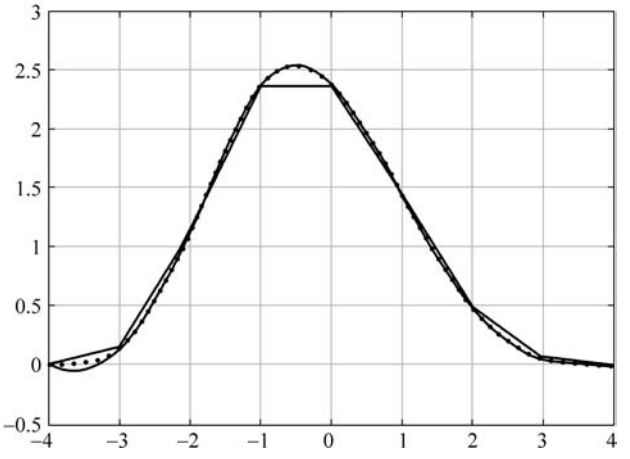

图 3-6　基础数据图形和样条插值图形

【例 3-11】 设函数 $f(x)=\dfrac{1}{1+25x^2}$，定义在区间[−1,1]上，取 $n=10$，按等距节点构造分段三次样条函数，试用 MATLAB 程序计算在各区间中点处分段三次样条插值及其相对误差。

代码如下：

```
% chapter3/test13.m
h = 0.2;
x0 = -1:h:1;
y0 = 1./(1 + 25. * x0.^2);
xi = -0.9:h:0.9;
fi = 1./(1 + 25. * xi.^2);
yi = spline (x0,y0,xi)
plot(x0,y0,xi,yi)
Ri = abs((fi - yi)./fi);
xi,fi,yi,Ri,
i = [xi',fi',yi',Ri']
```

运行结果如下，函数图形如图 3-7 所示。

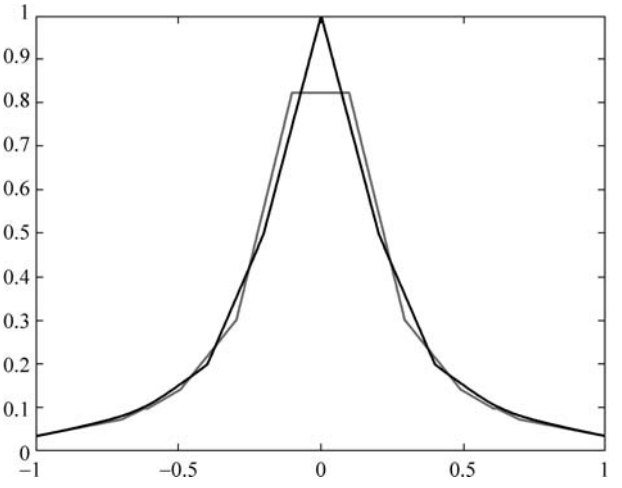

图 3-7 基础数据图形和样条插值图形

```
xi =
   -0.9000   -0.7000   -0.5000   -0.3000   -0.1000    0.1000    0.3000
0.5000    0.7000    0.9000
fi =
    0.0471    0.0755    0.1379    0.3077    0.8000    0.8000    0.3077
0.1379    0.0755    0.0471
yi =
    0.0484    0.0745    0.1401    0.2973    0.8205    0.8205    0.2973
```

```
0.1401    0.0745    0.0484
Ri =
    0.0279    0.0131    0.0160    0.0337    0.0257    0.0257    0.0337
0.0160    0.0131    0.0279
```

3.3　曲线拟合

在科学和工程领域，曲线拟合的主要目的是寻求平滑的曲线来最好地表现带有噪声的测量数据，从这些测量数据中寻求两个函数变量之间的关系或者变化趋势，最后得到曲线拟合的函数表达式。从“插值”一节中可以看出，使用多项式进行数据拟合会出现数据振荡或者 Runge 的现象，而 spline 插值方法可以得到很好的平滑效果，但是关于该插值方法有太多的参数，不适合进行曲线拟合。

同时，由于在进行曲线拟合的时候，认为所有测量数据中已经包含了噪声，因此，最后的拟合曲线并不要求通过每个已知数据点，衡量拟合数据的标准则是整体数据拟合的误差最小。一般情况下，MATLAB 的曲线拟合方法使用的是“最小方差”函数进行曲线拟合，其中方差的数值是拟合曲线和已知数据之间的垂直距离。和插值相同，对于曲线拟合也存在着各种不同的标准，为了满足不同的拟合要求，同样需要自行编写对应的 M 文件。在本节中，将分别介绍几种比较常见的曲线拟合方法，最后，还将介绍 MATLAB 提供的曲线拟合界面操作方法。

曲线拟合(Curve Fitting)的数学定义是指用连续曲线近似地刻画或比拟平面上一组离散点所表示的坐标之间的函数关系，是一种用解析表达式逼近离散数据的方法。曲线拟合通俗的说法就是“拉曲线”，也就是将现有数据通过数学方法来代入一条数学方程式的表示方法。在科学和工程中遇到的很多问题，往往只能通过诸如采样、实验等方法获得若干离散的数据，根据这些数据，如果能够找到一个连续的函数(曲线)或者更加密集的离散方程，使实验数据与方程的曲线能够在最大程度上近似吻合，就可以根据曲线方程对数据进行数学计算，对实验结果进行理论分析，甚至对某些不具备测量条件的位置的结果进行估算。

3.3.1　多项式拟合

在 MATLAB 中，提供了 polyfit 函数来计算多项式拟合系数，其设定曲线拟合的目标是最小方差(Least Squares)或者被称为最小二乘法，polyfit 函数的调用格式如下：

```
[p,S,mu] = polyfit(x,y,n)
[y,delta] = polyval(p,x,S,mu)
```

其中，x 和 y 表示的是已知测量数据，n 是多项式拟合的阶数。同时参数满足下面的等式 $\hat{x}=\frac{x-\mu_1}{\mu_2}$，其中 $\mu_1=\mathrm{mean}(x)$，$\mu_2=\mathrm{std}(x)$，而且 $\mu=[\mu_1,\mu_2]$。

通过上面的命令，最后可以得到的拟合曲线的多项式为

$$y = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1} \tag{3-50}$$

【例 3-12】 使用 polyfit 命令进行多项式的数值拟合，并分析曲线拟合的误差情况。

代码如下：

```
%chapter3/test14.m
x=(0:0.1:5)'
y=erf(x)
[p,s]=polyfit(x,y,6)
[yp,delta]=polyval(p,x,s)
plot(x,y,'+',x,yp,'g-')
legend('original','fitting')
```

运行结果如图 3-8 所示。

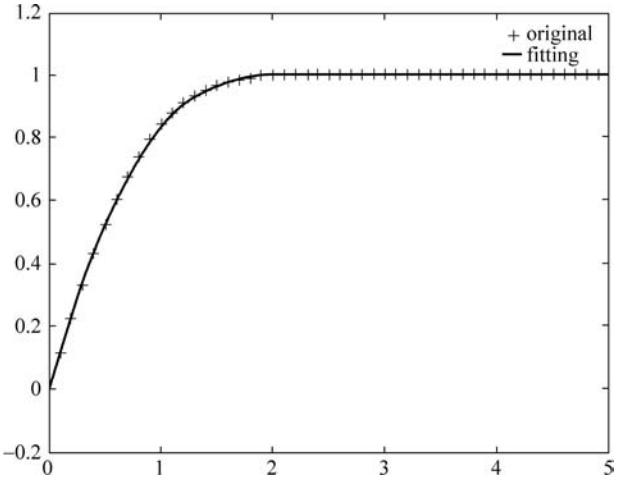

图 3-8 拟合曲线

【例 3-13】 已知 $x=[0.5,1.0,1.5,2.0,2.5,3.0]$，$y=[1.75,2.45,3.81,4.80,7.00,8.60]$。

代码如下：

```
%chapter3/test15.m
x=[0.5,1.0,1.5,2.0,2.5,3.0];
y=[1.75,2.45,3.81,4.80,7.00,8.60];
p=polyfit(x,y,2)   %多项式拟合
x1=0.5:0.5:3.0;
y1=polyval(p,x1);
plot(x,y,'*r',x1,y1,'-b')
```

运行结果如图 3-9 所示。

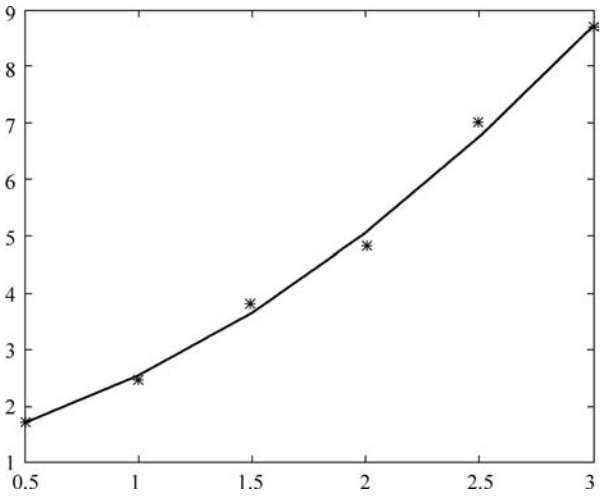

图 3-9　拟合曲线

3.3.2　加权最小方差拟合

所谓加权最小方差(Weighted Least Squares)，就是根据基础数据本身各自的准确度(或者被称为可靠性)的不同，在拟合的时候给每个数据以不同的加权数值。这种方法比前面所介绍的单纯最小方差方法要更加符合拟合的初衷。

对于 N 阶多项式的拟合公式，所要求解的拟合系数需要求解线性方程组，其中线性方程组的系数矩阵和需要求解的拟合系数(变量)矩阵分别为

$$\boldsymbol{A}=\begin{bmatrix} x_1^N & \cdots & x_1 & 1 \\ x_2^N & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_M^N & \cdots & x_M & 1 \end{bmatrix},\quad \boldsymbol{\theta}=\begin{bmatrix} \theta_N \\ \theta_{N-1} \\ \vdots \\ \theta_1 \end{bmatrix} \tag{3-51}$$

使用加权最小方差(WLS)方法求解得到的拟合系数为

$$\boldsymbol{\theta}_{\xi}^{o}=\begin{bmatrix} \theta_{gN}^{o} \\ \theta_{wN-1}^{o} \\ \vdots \\ \theta_1^{o} \end{bmatrix}=[\boldsymbol{A}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{A}]^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{W}_{Y} \tag{3-52}$$

其对应的加权最小方差为表达式 $\boldsymbol{J}_{\mathrm{W}}=[\boldsymbol{A}\theta-\boldsymbol{y}]^{\mathrm{T}}\boldsymbol{W}[\boldsymbol{A}\theta-\boldsymbol{y}]$最小。

3.3.3　数据拟合——适用加权最小方差 WLS 方法

【例 3-14】 根据 WLS 数据拟合方法，自行编写使用 WLS 方法拟合数据的 M 函数，然

后使用 WLS 方法进行数据拟合。

代码如下：

```
% chapter3/test16.m
function [th,err,yi] = wlspoly(x,y,N,xi,r)
      % x、y:数据点系列
      % N:多项式拟合的系统
      % r:加权系数的逆矩阵

      M = length(x);
      x = x(:);
      y = y(:);

      % 判断调用函数的格式
      if nargin == 4
      % 当调用的格式为 (x,y,N,r)
               if length(xi) == M
                        r = xi;
                        xi = x;
      % 当调用的格式为(x,y,N,xi)
               else r = 1;
               end;
      % 当调用格式为(x,y,N)
      elseif nargin == 3
               xi = x;
               r = 1;
      end
      % 求解系数矩阵
      A(:,N+1) = ones(M,1);
      for n = N:-1:1
              A(:,n) = A(:,n+1).*x;
      end
      if length(r) == M
               for m =1:M
                        A(m,:) = A(m,:)/r(m);
                        y(m) = y(m)/r(m);
               end
      end
      % 计算拟合系数
      th = (A\y)';
      ye = polyval(th,x);
      err = norm(y-ye)/norm(y);
      yi = polyval(th,xi);
```

WLS 数据拟合的代码如下：

```
% chapter3/test17.m
clear all
      clc
      x = [-3:1:3]';
      y = [1.1650 0.0751 -0.6965 0.0591 0.6268 0.3516 1.6961]';
      [x,i] = sort(x);
      y = y(i);
      xi = min(x) + [0:100]/100 * (max(x) - min(x));
      for i = 1:4
            N = 2 * i - 1;
            [th,err,yi] = wlspoly(x,y,N,xi,xi);
            subplot(2,2,i)
            plot(x,y,'o')
            hold on
            plot(xi,yi,'-')
            grid on
      end
```

程序运行的结果如图 3-10 所示。

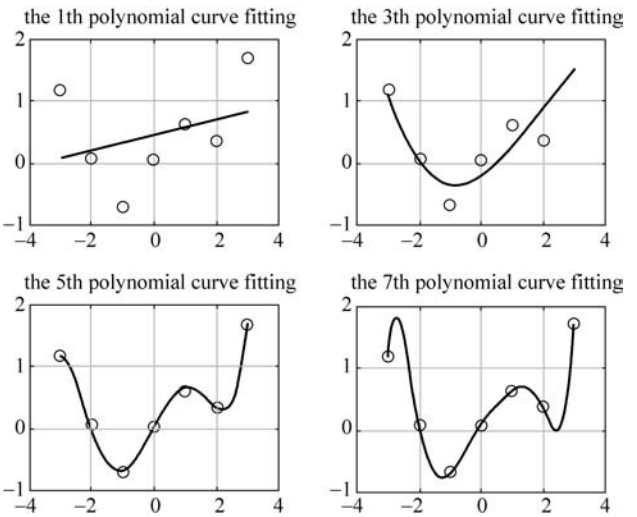

图 3-10　WLS 数据拟合曲线

LS 数据拟合的代码如下：

```
% chapter3/test18.m
  clear all
      clc
      x = [-3:1:3]';
      y = [1.1650 0.0751 -0.6965 0.0591 0.6268 0.3516 1.6961]';
```

```
for i = 1:4
  N = 2 * i - 1;
  [p,s] = polyfit(x,y,N);
  yi = polyval(p,x,s)
  subplot(2,2,i)
  plot(x,y,'o')
  hold on
  plot(x,yi,'-')
  title(['the',num2str(N),'th polynomial curve fitting'])
  grid on
end
```

运行结果如图 3-11 所示。

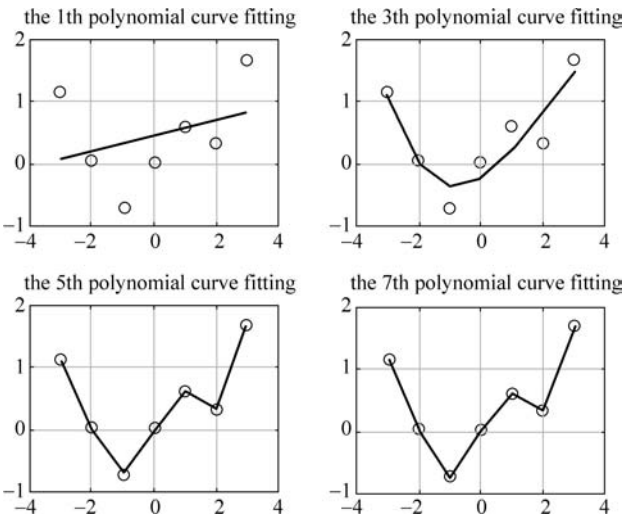

图 3-11 LS 数据拟合曲线

从上面的例子中可以看出，LS 方法其实是 WLS 方法的一种特例，相当于将每个基础数据的准确度都设为 1，但是，自行编写的 M 文件和默认的命令结果不同，读者可自行仔细比较。从图 3-11 可以看出，使用 WLS 得到的拟合结果比使用 LS 得到的拟合结果更加准确，只是参数比 LS 的参数更多。