Newton 插值的算法实现

Lagrange 插值公式结构紧凑，便于理论分析。利用插值基函数也容易到插值多项式的值。Lagrange 插值公式的缺点是，当插值节点增加，或其位置变化时，全部插值基函数均要随之变化，从而整个插值公式的结构也发生变化，这在实际计算中是非常不利的。下面引入的 Newton 插值公式可以克服这个缺点。

1、问题描述

当 n=1 时，由点斜式直线方程知，过两点 $(x_0, f(x_0))$ 和 $(x_1, f(x_1))$ 的直线方程为

$$N_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0).$$

若记 $f[x_0, x_1] = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$，则可把 $N_1(x)$ 写成

$$N_1(x) = f(x_0) + f[x_0, x_1](x - x_0).$$

显然，$N_1(x)$ 就是一次 Lagrange 插值多项式 $L_1(x)$。由于 $y = N_1(x)$ 表示通过两点 $(x_0, f(x_0))$ 和 $(x_1, f(x_1))$ 的直线，因此一次插值亦称为**线性插值**。

当 n=2 时，进而记

$$f[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1},\ f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_1}$$

类似地，构造不超过二次的多项式

$$N_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

容易检验，这样的 $N_2(x)$ 满足插值条件

$$N_2(x_0) = f(x_0),\ N_2(x_1) = f(x_1),\ N_2(x_2) = f(x_2).$$

因此，$N_2(x)$ 就是二次 Lagrange 插值多项式 $L_2(x)$。二次插值的几何解释是，用通过三点 $(x_0, f(x_0))$，$(x_1, f(x_1))$，$(x_2, f(x_2))$ 的抛物线 $y = N_2(x)$ 来近似所考察的曲线 $y = f(x)$，因此这类插值亦称为**抛物线插值**。

从上述公式可见，

$$N_2(x) = N_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

这种递推性，对于数值计算是很有效的。下面给出一般的计算公式。

2、Newton 插值多项式

为进一步构造更一般的插值多项式，记

$$\omega_k(x) = \prod_{i=0}^{k-1}(x - x_i)$$

由此构造

$$N_n(x) = f(x_0) + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k]\omega_k(x) \tag{1.1}$$

其中

$$f[x_0, x_1, \cdots, x_k] = \frac{f[x_1, x_2, \cdots, x_k] - f[x_0, x_1, \cdots, x_{k-1}]}{x_k - x_0} \tag{1.2}$$

称之为 $f(x)$ 在 $x_0, x_1, \cdots, x_k$ 上的 k **阶均差（或差商）**。

显然，$N_n(x)$ 是次数不超过 n 次的多项式，并由式（1.1）可知，它满足插值条件

$$N_n(x_i)=f(x_i),\quad i=0,1,\cdots,n.$$

因此把式(1.1)称作 **n 次 Newton 插值多项式**。根据插值多项式的惟一性，$N_n(x)$ 就是 $L_n(x)$ 。

进一步地,有插值余项的**均差型余项**

$$R_n(x)=f(x)-N_n(x)=f[x,x_0,x_1,\cdots,x_n]\omega_{n+1}(x).\qquad(1.3)$$

在实际计算中，经常利用由式（1.2）构造出的均差表 1.1 计算均差，即由表 1.1 中加下划横线的均差值直接构造插值多项式（1.1）。

表 1.1 均差表

x_k	$f(x_k)$	一阶均差	二阶均差	三阶均差	四阶均差
x_0	$\underline{f(x_0)}$				
x_1	$f(x_1)$	$\underline{f[x_0,x_1]}$			
x_2	$f(x_2)$	$f[x_1,x_2]$	$\underline{f[x_0,x_1,x_2]}$		
x_3	$f(x_3)$	$f[x_2,x_3]$	$f[x_1,x_2,x_3]$	$\underline{f[x_0,x_1,x_2,x_3]}$	
x_4	$f(x_4)$	$f[x_3,x_4]$	$f[x_2,x_3,x_4]$	$f[x_1,x_2,x_3,x_4]$	$\underline{f[x_0,x_1,x_2,x_3,x_4]}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

3、算法描述

Newton 插值多项式可以灵活地增加插值节点进行递推计算。该公式形式对称，结构紧凑，因而容易编写计算程序。

由于 Matlab 支持矩阵运算，且下标从 1 开始，故在没有特别说明的情况下，所有算法实现部分，下标都从 1 开始。

首先输入节点 (x,y)，插值点 PX 。其中 x 和 y 都为 $n\times 1$ 矩阵， PX 为 $m\times 1$ 矩阵（表示可以计算多个插值）。

然后构造一个 $n\times n$ 均差矩阵 FN 并赋初值 0。由均差表（表 1.1）可知， FN 的第一列为 y ，根据均差公式（1.2），第 i 行和第 j 列 $(2\le j\le i\le n)$ 交叉处元素值计算公式如下

$$FN(i,j)=\frac{FN(i,j-1)-FN(i-1,j-1)}{x(i)-x(i-j+1)}.\qquad(1.4)$$

根据公式（1.4）得到均差矩阵 FN 。

最后，对插值点 PX 的每一个分量，用式（1.1）计算插值值，结果保存在 $m\times 1$ 矩阵 PY 对应分量中。在用 Matlab 编程时，我们对式（1.1）作如下调整。

构造更一般的插值多项式，记

$$N_{n-1}(x)=f(x_1)+\sum_{k=2}^{n}f[x_1,x_2,\cdots,x_k]\omega_k(x)\qquad(1.5)$$

其中

$$f[x_1,x_2,\cdots,x_k]=FN(k,k)\qquad(1.6)$$

$$\omega_k(x)=\prod_{i=1}^{k-1}(x-x_i)\qquad(1.7)$$

基于上述表示式，给出算法框图见图 1.1。

按照算法框图 1.1，易于编制计算机程序进行算法实现。下面给出在 Matlab 上可执行的 Newton 插值方法**函数 PY=newton(x,y,PX)源代码**如下。

```
% 程序文件，newton.m
function PY=newton(x,y,PX)
```

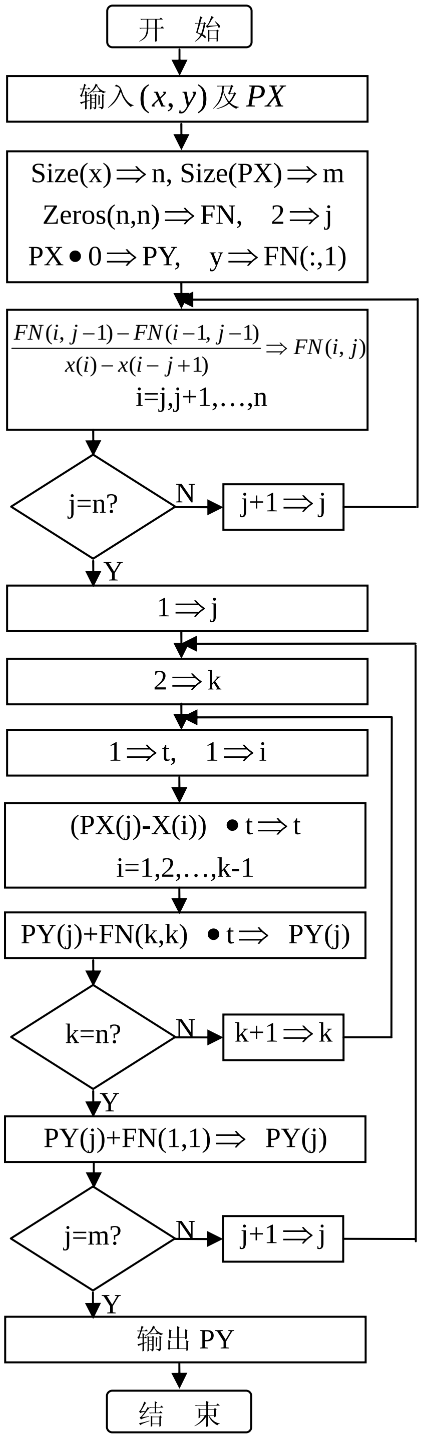


图 1.1 Newton 插值算法框图

```
% 根据已知节点(xi,yi)  (i=1,2,...,n) 计算插值点 PXj (j=1,2,...,m)对应的近似值 PYj
% (j=1,2,...,m),
x=x(:);
y=y(:);
% x,y 都为列向量
n=length(x);          % 已知节点个数
m=length(PX);         % 插值点个数
FN=zeros(n,n);        % 均差表二维矩阵，对角线上元素对应
                      %  FN(1,1)=f(x1), FN(2,2)=f[x1,x2],...,FN(n,n)=f[x1,x2,...,xn]
```

```
                    % 递推计算公式
                    % f[x1,x2,...,xk]=( f[x2,x3,...,xk] -f[x1,x2,...,x(k-1)] )/( xk-x1 )
PY=0*PX;            % 插值点对应的近似值
FN(:,1)=y;          % 均差表第一列为已知节点函数值

% 计算均差表
for j=2:n           % 从第二列开始计算
    for i=j:n
        FN(i,j)=( FN(i,j-1)-FN(i-1,j-1) )/( x(i)-x(i-j+1) );   % 根据均差表特点计算公式
    end
end

% 用 n-1 次 Newton 插值多项式计算 PX 中每个元素的近似值
for j=1:m
    for k=2:n
        t=1;
        for i=1:k-1
            t=( PX(j)-x(i) )*t;
        end
        PY(j)=PY(j)+FN(k,k)*t;
    end
    PY(j)=PY(j)+FN(1,1);
End
```

4、实例分析

先给出一个 Newton 插值效果好的例子。

例 1.1 设 $f(x)=2e^x+\sin x$，讨论 $f(x)$ 在区间 $x\in[0,2]$ 上等距节点 $x_i\ (i=1,2,...,n)$ 处的 Newton 插值多项式。

解：在 $x\in[0,2]$ 上，取步长 $h=0.01$，作 $f(x)$ 的图形（图 1.2），Matlab 命令如下：

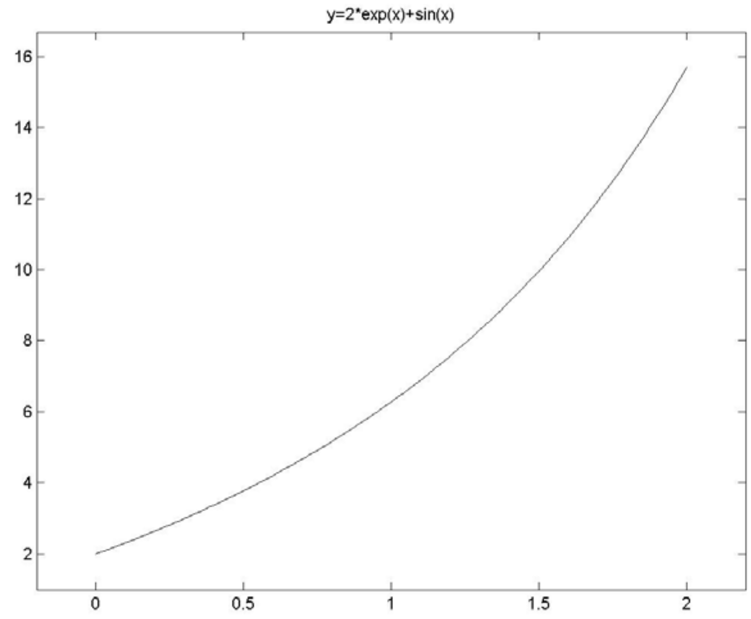


图 1.2 函数 $f(x)=2e^x+\sin x$ 在 $x\in[0,2]$ 上图形

```
>>x=[0:0.01:2];
```

```
>>y=2*exp(x)+sin(x);
>>plot(x,y,'k');
>>title('y=2*exp(x)+sin(x)');

>>% 控制图形显示位置
>>i=0.20;
>>xma=max(x)+i;、
>>xmi=min(x)-i;
>>yma=max(y)+5*i;
>>ymi=min(y)-5*i;
>>axis([xmi xma ymi yma]);
```

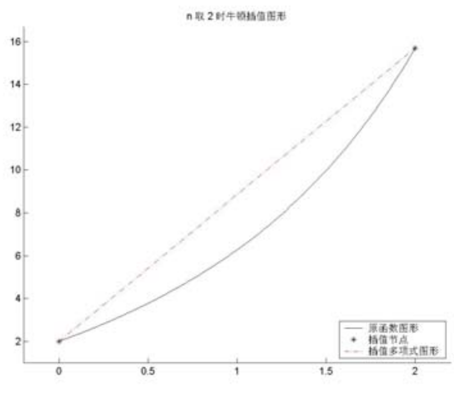
调用上面的 Newton 插值算法的 Matlab 函数文件，可得插值计算结果。Matlab 命令如下，对于不同的 n，只需要在程序中修改 n 值即可。

```
%   程序文件，examp_001.m
x=[0:0.01:2];                %(x,y)用于画原函数图
y=2*exp(x)+sin(x);
n=2;                         %已知节点个数
xx=linspace(0,2,n);          % (xx,yy)为已知节点
yy=2*exp(xx)+sin(xx);
px=[0:0.01:2];               % (px,py)待插值节点
py=newton(xx,yy,px); % 用牛顿法插值
hold on;
plot(x,y,'k',xx,yy,'k*',px,py,'r-.');
legend('原函数图形','插值节点','插值多项式图形',4);
txt=['n 取 ' num2str(n) ' 时 Newton 插值图形'];
title(txt);
% 控制图形显示位置
i=0.20;
xma=max(x)+i;
xmi=min(x)-i;
yma=max(y)+5*i;
ymi=min(y)-5*i;
axis([xmi xma ymi yma]);
```

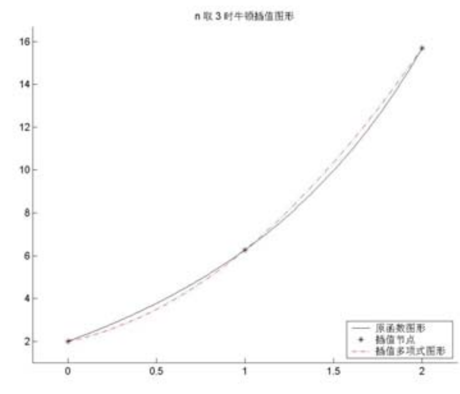
当 $n=2$ 时为线性插值，从图 1.3（a）可以看出插值效果不理想。

当 $n=3,6,10$ 时，效果很好，且从图中可以看出，当 $n\geq 6$ 时，插值图形与原函数图形几乎吻合。

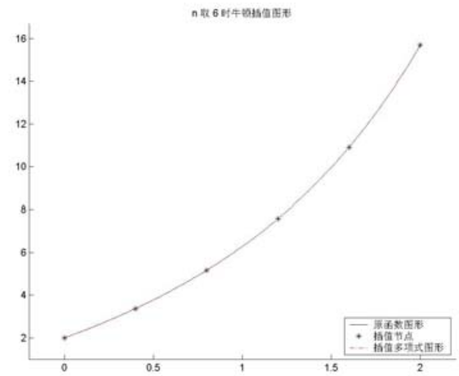
$n\geq 20$ 已经很难看出被插函数与 Newton 插值多项式的区别了。



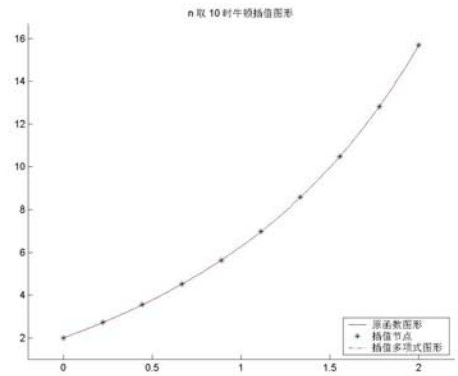
(a) n=2



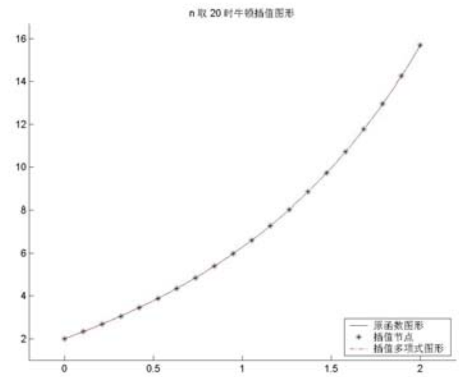
(b) n=3



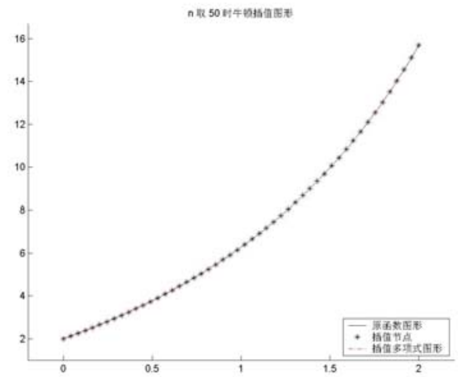
(c) n=6



(d) n=10



(e) n=20



(f) n=50

图 1.3　$n=2,3,6,10,20,50$ 时，Newton 插值效果对比

再给出一个 Newton 插值效果不好的例子。

从上例可以看出，用插值多项式近似被插函数 $f(x)=2e^x+\sin x$ 时，插值多项式的次数越高，效果越好。那么，是否对于所有的被插函数，都是这样的吗？答案是否定的。下面是说明这种现象的一个典型例子。

例 1.2　给定函数

$$f(x)=\frac{1}{1+x^2},-5\le x\le 5$$

取等距插值节点 $x_k=-5+10k/n, k=1,2,...,n$，构造 $n-1$ 次 Newton 插值多项式

$$N_{n-1}(x)=f(x_1)+\sum_{k=2}^{n}f[x_1,x_2,\cdots,x_k]\omega_k(x).$$

当 $n=11$ 时，10 次插值多项式 $N_{10}(x)$ 以及函数 $f(x)$ 的图形如图 1.4。由此可见，$N_{10}(x)$ 的截断误差 $R_{10}(x)=f(x)-N_{10}(x)$ 在区间 $[-5,\ 5]$ 的两端非常大。例如，$N_{10}(4.8)=1.80438545612784$，而 $f(4.8)=0.04159733777038$。这种现象称为 **Runge 现象**。不管 n 取多大，Runge 现象依然存在。

因此，对函数做插值多项式时，必须小心处理，不能认为插值节点取得越多，插值余项就越小。此外，当节点增多是，舍入误差的影响不能低估。为了克服高次插值的不足，采用分段插值将是理论和实际应用的一个良好的插值方法。

方案一、将 11 个已知插值节点分为两段，对两段分别用 5 次 Newton 多项式插值，再把两段拼接，如图 1.5。从图中可以看出，两段效果明显高于 10 次 Newton 多项式插值，不足的是两段拼接处出现**尖角**。

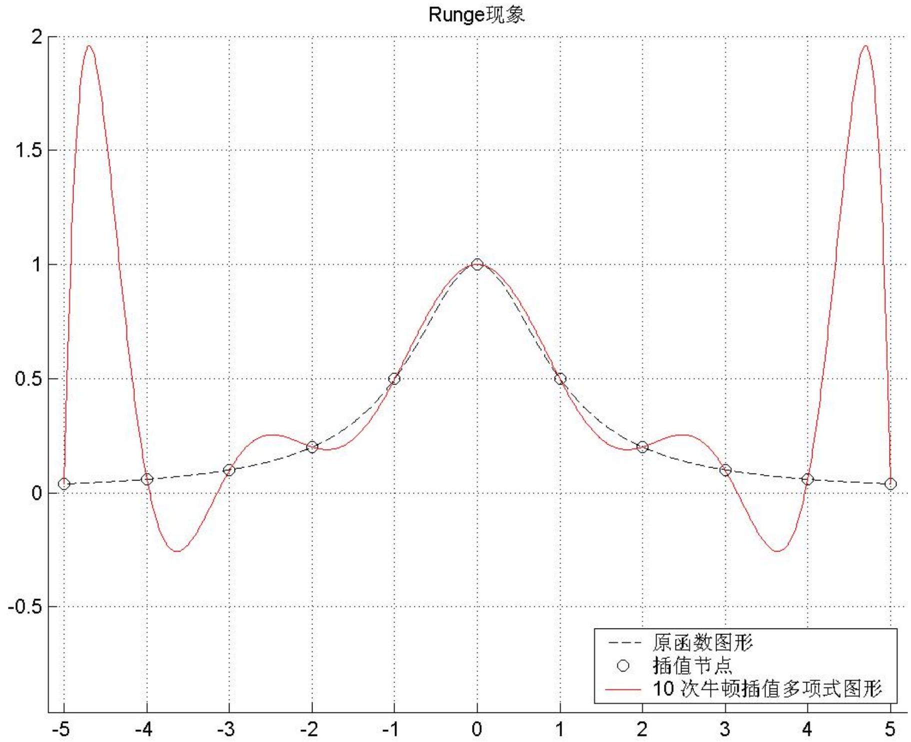


图 1.4　函数 $f(x)$ 与 $N_{10}(x)$ 的图形

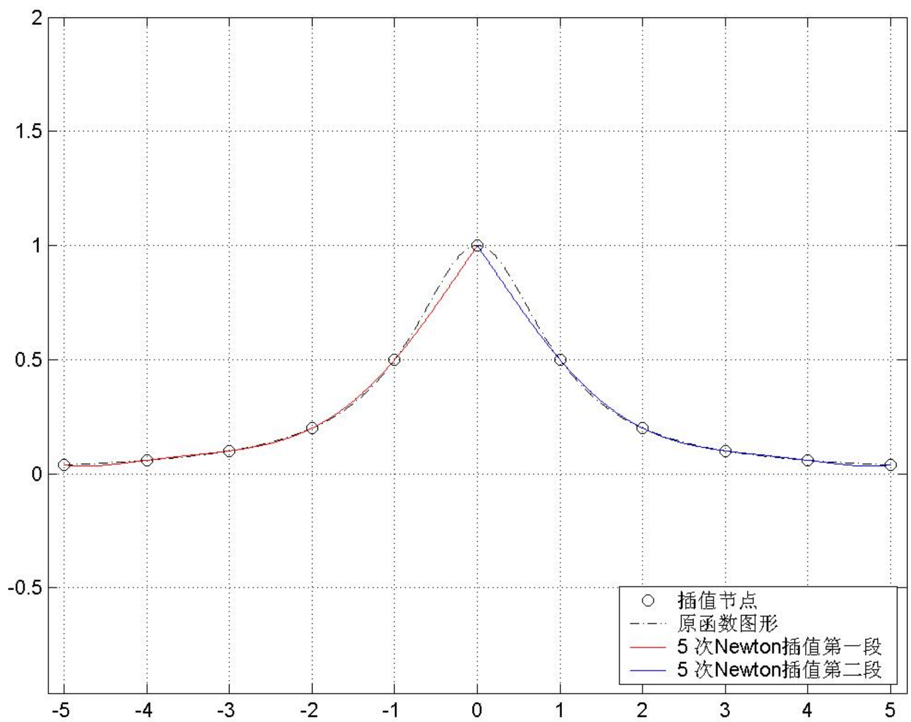
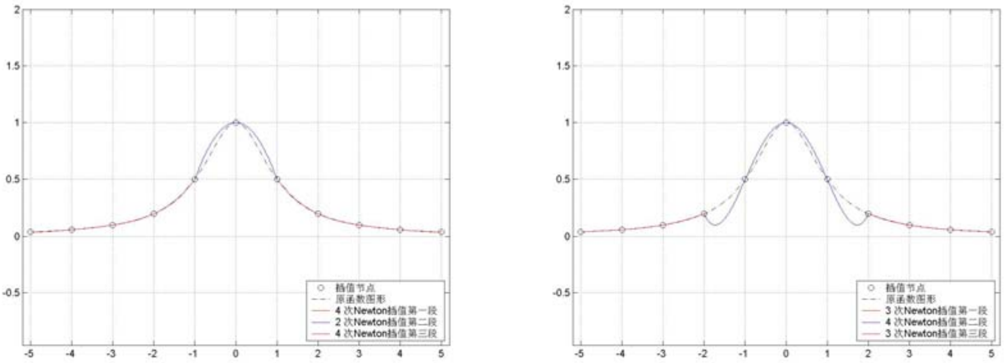


图 1.5　分两段 Newton 多项式插值效果

方案二、因为已知插值节点图形为一个轴对称图形，且开始五个点间线段斜率变化较缓，中间三个点间线段斜率变化较为剧烈，因此选择分为三段插值，插值点数分别为 5，3，5，依次得到 4 次，2 次，4 次 Newton 插值多项式，插值效果如图 1.6（a）。另外，若分段插值点数为 4，5，4，将依次得到 3 次，4 次，3 次 Newton 插值多项式，插值效果如图 1.6（b），出现了尖角。



(a)插值点数分别为 5,3,5　　　　(b) 插值点数分别为 4,5,4

图 1.6　分三段 Newton 多项式插值效果

从图形中可以看出，分段插值法能较好地逼近被插函数，只要节点间间距充分小，分段插值法总能获得所要求的精度，而不会像高次插值那样发生 Runge 现象。分段插值法另一个重要的特点是具有局部性质。但是，在两段拼接处如何处理或者避免尖角，达到光顺效果，这就要涉及到拼接处的导数问题，在后面我们将具体讨论带导数条件的插值方法。