# 第三次上机作业

学号: 15220212202842　姓名: 陈子睿

October $18^{th}$ 2023

## 1 数值微分

### 1.1 问题描述

| $h$ | 二点格式数值微分值 | 误差 | 三点格式数值微分值 | 误差 |
|---|---|---|---|---|
| 0.1 | | | | |
| 0.01 | | | | |
| 0.001 | | | | |
| 0.0001 | | | | |
| 0.00001 | | | | |
| 0.000001 | | | | |
| 0.0000001 | | | | |

表 1: 数值微分的误差

a) 分析表格中的误差与 h 的关系。考虑浮点数的有效数字位对误差的影响，误差阶是否与理论一致? 特别地，当 h 的值减小时，舍入误差会变得更为显著。假定单精度浮点数具有 7 位有效数字，而双精度浮点数具有 16 位有效数字。请推断在上述两点格式中，分别使用单精度和双精度计算时，哪个 h 值会导致最小的误差。

b) 设计一个更高精度的数值微分格式，使其的截断误差为 $O(h^4)$。然后进行数值实验验证你的格式，并将结果填写在与表格一相似的表中。

c) 推导一种非均匀网格设定下的三点数值微分格式:

$$f'(x_i) \approx c_1 f(x_{i-1} + c_2 f(x_i) + c_3 f(x_{i+1}) \tag{1}$$

## 1.2 数学描述

### 1.2.1 舍入误差分析

我们首先使用二点和三点公式计算数值微分:

给定函数 $f(x) = e^x$, (在这里我们仍暂时用 $f(x)$ 指代, 后续将带入数值进行具体计算), 我们已知:

二点前向公式为:

$$f'(x_0) \approx \varphi_1'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} \tag{2}$$

三点公式为:

$$f'(x_0) \approx \varphi_2'(x_0) = \frac{-3f(x_0) + 4f(x_0+h) - f(x_0+2h)}{2h} \tag{3}$$

当我们持续减少 $h$ 的值时, 尽管我们的期望误差会减小, 但是由于计算机浮点运算的舍入误差, 我们的实际误差可能会增大. 对于单精度和双精度, 我们估计舍入误差的界限分别为 $10^{-7}$ 和 $10^{-16}$.

对于二点公式, 舍入误差的主要来源是 $f(x_0+h)$ 与 $f(x_0)$ 的差分中. 当 h 接近单精度的舍入误差界限 (约为 $10^{-7}$ 时), 这时候的差分将会受到严重的舍入误差的影响. 对于双精度而言, 这个界限是 $10^{-16}$.

### 1.2.2 更高精度数值微分格式

为了得到一个截断误差为 $O(h^4)$ 的数值微分格式, 我们可以考虑使用五点中心差分公式:

$$f'(x_0) \approx \frac{f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)}{12h} \tag{4}$$

### 1.2.3 非均匀网格的三点数值微分格式

对于非均匀网络, 给定 $x_0$, $x_1$,$x_2$ 三个节点, 我们假设:

$f(x) \in C^2_{[a,b]}$, 其中 [a,b] 是包含点 $x_0$, $x_1$,$x_2$ 的任一区间.

显然我们可以使用泰勒级数来完成题目的要求. 首先, 我们可以通过泰勒级数将 $f(x)$ 在 $x_1$ 附近展开:

$$f(x) = f(x_1) + (x-x_1)f'(x_1) + \frac{(x-x_1)^2}{2!}f''(x_1) + R_2(x) \tag{5}$$

然后我们可以在 $x_0$ 与 $x_2$ 处应用上述公式, 得到:

$$f(x_0) = f(x_1) + (x_0-x_1)f'(x_1) + \frac{(x_0-x_1)^2}{2!}f''(x_1) + R_2(x_0) \tag{6}$$

$$f(x_2) = f(x_1) + (x_2-x_1)f'(x_1) + \frac{(x_2-x_1)^2}{2!}f''(x_1) + R_2(x_2) \tag{7}$$

倘若忽略余项 $R_2(x_i)(i = 0, 2)$, 我们可以将上述二式视作一个线性方程组:

$$\begin{cases} f(x_0) = f(x_1) + (x_0 - x_1)f'(x_1) + \frac{(x_0-x_1)^2}{2!}f''(x_1) \\ f(x_2) = f(x_1) + (x_2 - x_1)f'(x_1) + \frac{(x_2-x_1)^2}{2!}f''(x_1) \end{cases} \tag{8}$$

我们可以解出 $f'(x_1)$:

$$f'(x_1) \approx \frac{h_0 f(x_1 + h_1)}{(h_0 + h_1)h_1} - \frac{h_1 f(x_1 - h_0)}{(h_0 + h_1)h_0} + \frac{(h_1 - h_0)f(x)}{h_0 h_1} \tag{9}$$

其中 $h_1 = x_2 - x_1$,$h_0 = x_1 - x_0$.

从而我们便推导出了非均匀网格下的三点数值微分格式.

## 1.3 数值结果与图像

### 1.3.1 二点与三点数值微分

通过编写的程序, 我们可以将得到的数据填入表中. 通过观察我们发现, 误差阶与理论上是一致的 (见表 2).

| $h$ | 二点格式数值微分值 | 误差 | 三点格式数值微分值 | 误差 |
|---|---|---|---|---|
| 0.1 | 2.858841955 | 0.140560126 | 2.708508438 | 0.00977339 |
| 0.01 | 2.731918656 | 0.013636827 | 2.718190536 | 9.13E-05 |
| 0.001 | 2.719641423 | 0.001359594 | 2.718280922 | 9.07E-07 |
| 0.0001 | 2.718417747 | 0.000135919 | 2.718281819 | 9.06E-09 |
| 0.00001 | 2.71829542 | 1.36E-05 | 2.718281828 | 3.02E-11 |
| 0.000001 | 2.718283187 | 1.36E-06 | 2.718281828 | 8.30E-10 |
| 0.0000001 | 2.718281968 | 1.40E-07 | 2.718281835 | 6.72E-09 |

表 2: 二点与三点数值微分的误差

### 1.3.2 五点中心差分

运用前文提到的五点中心差分公式 (方程 (4)), 我们可以得到如下结果:

| $h$ | 五点中心差分微分值 | 误差 |
|---|---|---|
| 1 | 2.616232609 | 0.102049219 |
| 0.5 | 2.712447716 | 0.005834112 |
| 0.1 | 2.718272757 | 9.07E-06 |
| 0.01 | 2.718281828 | 9.06E-10 |

表 3: 五点中心数值微分的误差

#### 1.3.3 非均匀网格下的三点数值微分

运用前文提到的非均匀网络下的三点数值微分格式 (方程 (9)), 我们可以得到如下结果:

| $h_0$ | $h_1$ | 非均匀网格下的三点数值微分值 | 误差 |
|---|---|---|---|
| 0.5 | 0.2 | 2.76058851256760 | 0.0423066841085595 |
| 0.1 | 0.2 | 2.7275832724764 | 0.00930144401742750 |
| 0.01 | 0.02 | 2.71837266573971 | 9.08372806622815e-05 |
| 0.001 | 0.002 | 2.71828273477968 | 9.06320632321211e-07 |

表 4: 五点中心数值微分的误差

## 1.4 结论

#### 1.4.1 问题背景

Hermite 插值是数值分析中的一种插值技术, 它不仅考虑在插值节点上的函数值, 而且还考虑在这些节点上的导数值. 给定两个点和它们在这两个点上的导数值, Hermite 插值旨在找到一个三次多项式与给定的函数值和导数值相符.

#### 1.4.2 算法概述

二点数值微分格式: 二点格式是一种基于函数在给定点和该点前进一定距离后的值来近似导数的方法. 它是基于斜率的最基本定义得出的.

三点数值微分格式: 三点格式可以采用中心差分方法, 它使用函数在给定点前后的值来近似该点的导数. 由于它考虑了函数在该点前后的行为, 所以通常比二点格式更精确.

中心差分数值微分格式: 中心差分格式可以通过将 $f(x)$ 在 $x$ 点附近的泰勒级数展开, 然后消去高阶项来得到.

非均匀网络下的三点数值微分格式: 非均匀网格下的三点数值微分格式是求解函数导数的一种方法. 通过泰勒级数展开并构造线性方程系统来求得函数在某点的导数近似值.

### 1.4.3 总结

二点与三点数值微分格式基于最基本的斜率定义, 并且只需要函数在一个额外的点上的值, 由于只需要两个函数值, 所以计算开销较小. 但是它是一个前向差分, 因此仅具有 $O(h)$ 的准确性, 这意味着误差与步长成正比. 我们可以通过使用更小的步长可以减少误差, 但有一个下限, 超过这个下限可能会增加由于舍入误差引起的误差。

五点中心差分方法具有高阶截断误差 $O(h^4)$. 这意味着随着步长 $h$ 的减小, 误差会以 $h$ 的四次幂的速度减小. 这种方法精度较高, 由于具有 $O(h^4)$ 的截断误差, 这种方法比具有较低截断误差阶的数值微分方法 (如前向差分和后向差分) 提供了更高的精度. 但是其计算成本较高, 我们需要在多个点上评估函数, 这可能会增加计算成本, 特别是当函数评估成本较高或步长较小时. 并且其依赖于函数的光滑性, 如果函数在某些点上不光滑或有噪声, 这种方法可能会产生较大的误差. 我们可以设计一个算法, 该算法能够根据函数的局部行为自适应地选择步长 . 例如, 如果函数在某个区域内变化较快, 则可能需要选择较小的步长以获得更准确的结果. 如果函数包含噪声, 可以考虑应用一些噪声滤波技术, 如滑动平均或其他滤波方法, 以减少噪声对数值微分的影响. 这种高阶中心差分方法在需要高精度数值微分的场合中非常有用, 但也需要注意其可能的计算成本和对函数光滑性的依赖, 通过适当的改进措施, 可以进一步提高其效率和准确性.

非均匀网格下的三点数值微分格式是求解函数导数的一种方法. 通过泰勒级数展开并构造线性方程系统来求得函数在某点的导数近似值. 该方法可以应用于非均匀网格, 这使得它具有一定的灵活性和适应性, 尤其是在处理具有不同尺度或非均匀采样的问题时. 且其推导过程以及应用相对简单. 但是非均匀网格可能会降低数值微分的准确性, 尤其是在网格非常不均匀或函数变化非常剧烈的情况下, 可能会得到较大的误差. 我们可以通过优化网格的选择, 例如使用更加均匀的网格或在需要的区域增加网格密度, 可能会提高数值微分的准确性和稳定性. 也可以考虑使用更高阶的数值微分格式, 例如五点或七点差分格式, 可能会提高数值微分的准确性. 同时也可以通过引入误差估计和自适应网格策略, 可以在保证准确性的同时, 尽可能地减少计算量. 综上, 非均匀网格下的三点数值微分格式提供了一种求解函数导数的方法, 但可能会受到准确性和稳定性问题的影响. 通过改进网格选择和考虑高阶差分格式, 以及引入误差估计和自适应方法, 可以在一定程度上改善这些问题.

# 2 Hermite 插值

## 2.1 问题描述

请编写实现 Hermite 插值的函数，函数名、输入输出数据格式可参考如下形式：

```
function [yy] = interp_hermite3(x, y, yp, xx)
% yy = interp_Hermite3(x, y, yp, xx)
% 使用 3 次 Hermite 插值计算基于点 x(1),x(2),y(1),y(2),yp(1),yp(2) 的插值多项式,
% 函数返回该多项式在估值点 xx 的值, 返回值保存在 yy 中
% xx 可为一个数或向量, 相应的 yy 也为一个数或向量
```

## 2.2 数学描述

### 2.2.1 Hermite 插值函数

给定 $x_0$,$x_1$ 和相应的函数值 $y_0$,$y_1$ 以及微商值 $y_0'$,$y_1'$，构造插值函数 $H(x)$，要求满足条件:
(1) $H(x)$ 是不超过三次的多项式.
(2) $H(x_0)=y_0$,$H(x_1)=y_1$,$H'(x_0)=y_0'$,$H'(x_1)=y_1'$.
我们考虑构造插值基函数 $h_0(x)$:

$$h_0(x)=(1+2\frac{x-x_0}{x_1-x_0})(\frac{x-x_1}{x_0-x_1})^2 \tag{10}$$

以及

$$h_1(x)=(1+2\frac{x-x_1}{x_0-x_1})(\frac{x-x_0}{x_1-x_0})^2 \tag{11}$$

下面我们构造 $H_0(x)$:

$$H_0(x)=(x-x_0)(\frac{x-x_1}{x_0-x_1})^2 \tag{12}$$

以及

$$H_1(x)=(x-x_1)(\frac{x-x_0}{x_1-x_0})^2 \tag{13}$$

利用这四个插值基函数我们可以直接写出 Hermite 插值函数:

$$H(x)=y_0h_0(x)+y_1h_1(x)+y_0'H_0(x)+y_1'H_1(x) \tag{14}$$

## 2.3 数值结果与图像

最终得到的 Hermite 插值函数为:

$$H(x)=-x^3+\frac{3}{2}x^2+\frac{1}{2}x+1 \tag{15}$$

如下图所示，我们可以观察到我们所构造的 Hermite 插值函数是一个较为光滑的函数:

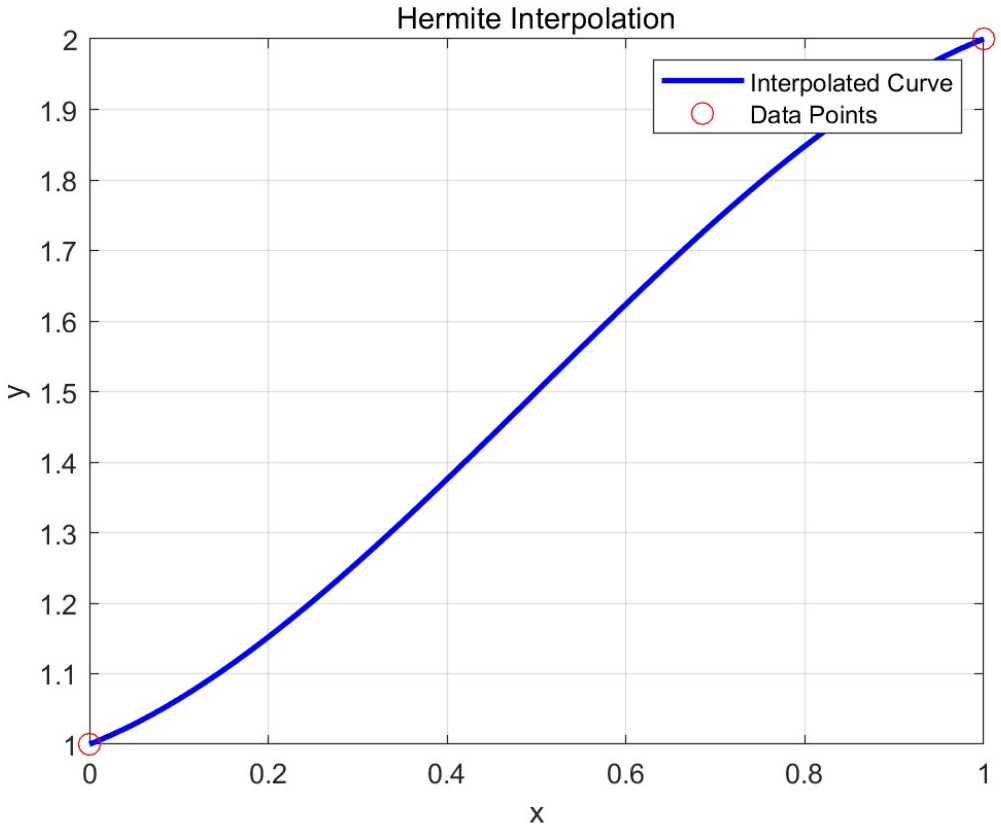



图 1: Hermite 插值函数图像

## 2.4 结论

### 2.4.1 问题背景

Hermite 插值是数值分析中的一种插值技术, 它不仅考虑在插值节点上的函数值, 而且还考虑在这些节点上的导数值. 给定两个点和它们在这两个点上的导数值, Hermite 插值旨在找到一个三次多项式与给定的函数值和导数值相符.

### 2.4.2 算法概述

Lagrange 基函数: Hermite 插值首先构建了 Lagrange 基函数, 这是由给定的插值节点派生出来的.

Hermite 多项式: 基于上述基函数和它们的导数, 我们构建了一个 Hermite 插值多项式. 这个多项式不仅考虑了函数在给定点的值, 还考虑了函数在这些点上的导数值.

### 2.4.3 符号计算的应用

为了获取插值函数的各项系数, 我们采用了 MATLAB 中的符号计算功能. 通过符号计算, 我们可以直观地看到插值多项式的表达式和它的各项系数.

#### 2.4.4 测试与验证

我们为这个插值函数设计了一个简单的测试程序, 其中选取了特定的点和导数值进行测试. 然后, 我们在一个范围内评估了 Hermite 插值多项式, 并使用图形显示了结果.

#### 2.4.5 挑战与解决方法

函数定义的复杂性: 由于 Hermite 插值需要同时考虑函数值和导数值, 所以它的定义比简单的 Lagrange 或 Newton 插值要复杂, 但通过系统地构建 Lagrange 基函数及其导数, 我们可以逐步构建 Hermite 插值函数.

代码错误和修正: 在给定的原始代码中, 存在一些定义错误和遗漏. 经过仔细分析和测试, 我们已经对代码进行了修正, 使其能够正确运行.

#### 2.4.6 总结

Hermite 插值提供了一种强大的工具, 允许我们根据已知的函数值和导数值进行插值. 虽然其定义比其他插值技术更为复杂, 但它为我们提供了更高的准确性, 尤其是当我们具有关于函数导数的额外信息时. MATLAB 为我们提供了一组强大的工具, 使得实现和测试 Hermite 插值变得相对简单.

# 3 样条插值函数

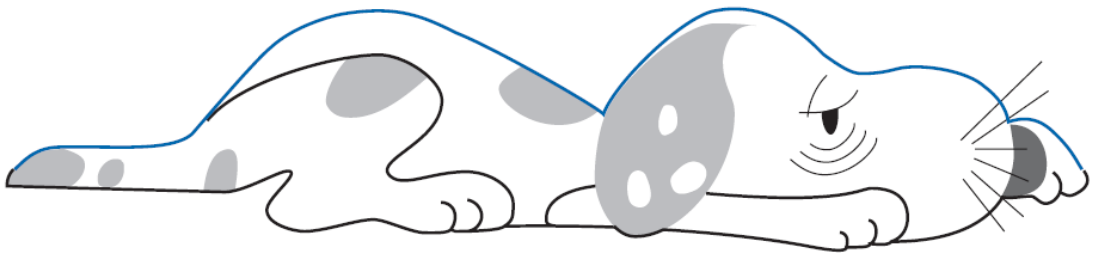
## 3.1 问题描述

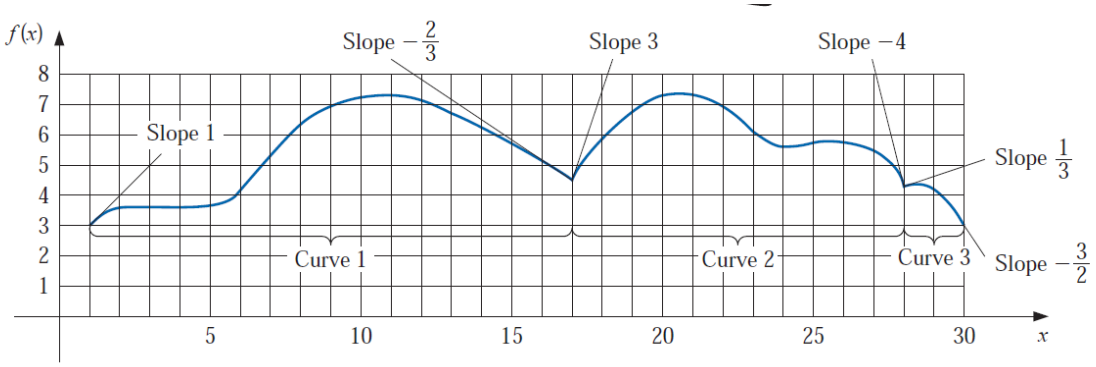
请完善程序 $build-spline3.m$, 实现 “EndSlope” 边界条件 (即书中的第一种边界条件):

a) 用课本 P45, Ex.10 (3) 中的例子验证原有程序中第二类边界条件的程序实现是否正确? 在得到一阶导数值之后, 可用附录二中的 $build-hermite3.m$ 构造课后答案形式的多项式.

b) 实现 “EndSlope” 边界条件后用 P46, Ex.12 检验程序实现是否正确.

c) 样条经常用于计算机辅助设计 (CAD) 中线条或曲面的描述, 请用样条表示下面卡通动物的上半部曲线, 分成三段曲线, 需要的数据如下 (来源于 Numerical Analysis, Richard. L. Burden, J. Douglas Faires, 9ed, Brooks, 2011);






| Curve 1 | | | | Curve 2 | | | | Curve 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | $x_i$ | $f(x_i)$ | $f'(x_i)$ | $i$ | $x_i$ | $f(x_i)$ | $f'(x_i)$ | $i$ | $x_i$ | $f(x_i)$ | $f'(x_i)$ |
| 0 | 1 | 3.0 | 1.0 | 0 | 17 | 4.5 | 3.0 | 0 | 27.7 | 4.1 | 0.33 |
| 1 | 2 | 3.7 | | 1 | 20 | 7.0 | | 1 | 28 | 4.3 | |
| 2 | 5 | 3.9 | | 2 | 23 | 6.1 | | 2 | 29 | 4.1 | |
| 3 | 6 | 4.2 | | 3 | 24 | 5.6 | | 3 | 30 | 3.0 | −1.5 |
| 4 | 7 | 5.7 | | 4 | 25 | 5.8 | | | | | |
| 5 | 8 | 6.6 | | 5 | 27 | 5.2 | | | | | |
| 6 | 10 | 7.1 | | 6 | 27.7 | 4.1 | −4.0 | | | | |
| 7 | 13 | 6.7 | | | | | | | | | |
| 8 | 17 | 4.5 | −0.67 | | | | | | | | |

## 3.2 数学描述

回顾三次样条函数的定义:

给定一组数据点 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$，三次样条插值函数 $S$ 在每个子区间 $[x_i, x_{i+1}]$ 上都是一个三次多项式，它可以写为:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{16}$$

其中，$i = 0, 1, \ldots, n-1$，并且 $a_i, b_i, c_i,$ 和 $d_i$ 是待定的系数。

为了确定这些系数，我们需要满足以下条件:

1. 函数 $S$ 在每个数据点上的值与给定的值相等。
2. 函数 $S$ 在每个子区间的两端的一阶导数是连续的。
3. 函数 $S$ 在每个子区间的两端的二阶导数是连续的。

我们已经知道, 在子区间 $[x_{i-1}, x_i]$ 上有:

$$s_i = M_{i-1}\frac{(x_i - x)^3}{6h_i} + M_i\frac{(x - x_{i-1})^3}{6h_i} + \frac{1}{h_i}(y_{i-1} - \frac{1}{6}M_{i-1}h_i^2)(x_i - x) + \frac{1}{h_i}(y_i - \frac{1}{6}M_i h_i^2)(x - x_{i-1}) \tag{17}$$

三次样条插值通常考虑以下三种边界条件:

1. **自然边界条件** (Natural Spline): 当 $S''(x_0) = 0$ 和 $S''(x_n) = 0$ 时，称为自然边界条件。

2. **固定边界条件** (Clamped or Fixed-End Spline): 当 $S'(x_0) = f'(x_0)$ 和 $S'(x_n) = f'(x_n)$ 时，称为固定边界条件。这里的 $f'$ 是已知的导数值。

3. **周期边界条件** (Periodic Spline): 当 $S'(x_0) = S'(x_n)$ 和 $S''(x_0) = S''(x_n)$ 时，称为周期边界条件。

具体的计算过程在此不再赘述, 直接给出三种边界条件的线性方程.
对于**自然边界条件**, 我们可以得到线性方程组如下:

$$\begin{bmatrix} 2 & \alpha_1 & & & \\ \alpha_2 & 2 & 1-\alpha_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{n-2} & 2 & 1-\alpha_{n-2} \\ & & & \alpha_{n-1} & 2 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-2} \\ m_{n-1} \end{bmatrix} = \begin{bmatrix} \beta_1 - \alpha_1 y_0'' \\ \beta_2 \\ \vdots \\ \beta_{n-2} \\ \beta_{n-1} - (1-\alpha_{n-1}) y_n'' \end{bmatrix}$$

对于**固定边界条件**, 我们可以得到线性方程组如下:

$$\begin{bmatrix} 2 & 1 & & & \\ \alpha_1 & 2 & 1-\alpha_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{n-1} & 2 & 1-\alpha_{n-1} \\ & & & 1 & 2 \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{bmatrix}$$

## 3.3 数值结果及图像

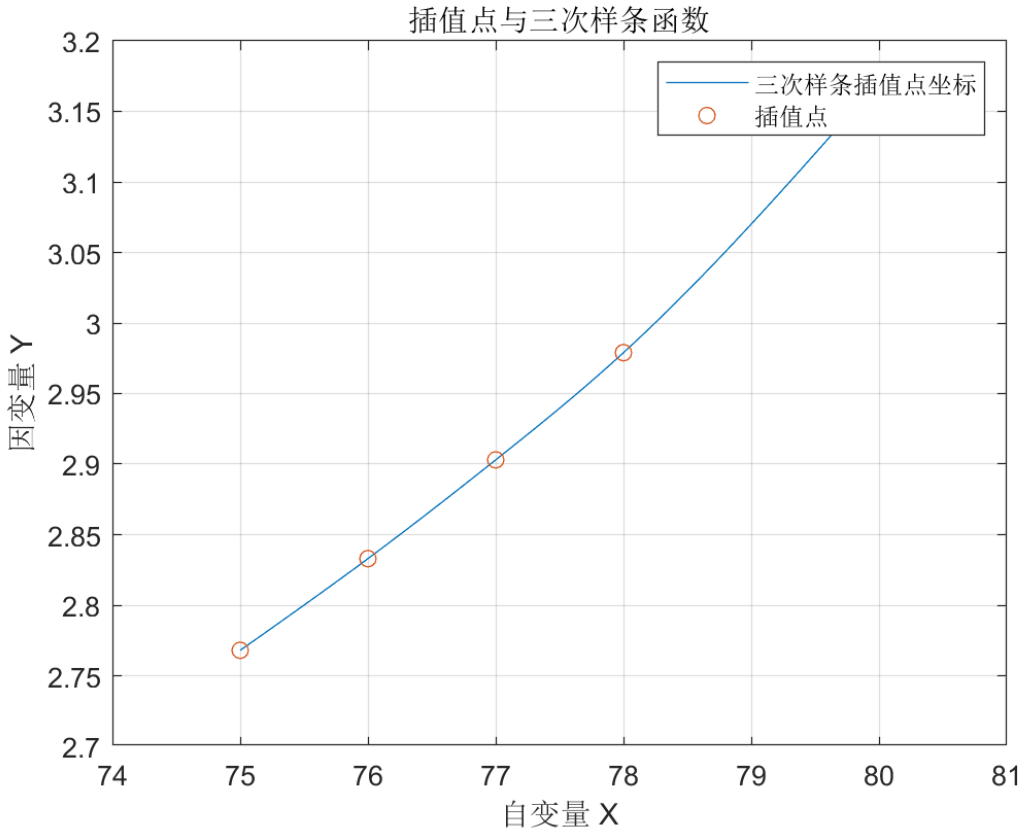
第一小问例题如下:

10. 给定数组

| $x$ | 75 | 76 | 77 | 78 | 79 | 80 |
|---|---|---|---|---|---|---|
| $y$ | 2.768 | 2.833 | 2.903 | 2.979 | 3.062 | 3.153 |

(1) 作一分段线性插值函数.
(2) 取第二类边界条件,作三次样条插值多项式.
(3) 用两种插值函数分别计算 $x=75.5$ 和 $x=78.3$ 的函数值.

属于第二边界值 (自然边界) 问题, 我们将数值写入, 调用写好的 matlab 代码 (见附录), 可以得到运行后的结果如下:




第二小问例题如下:

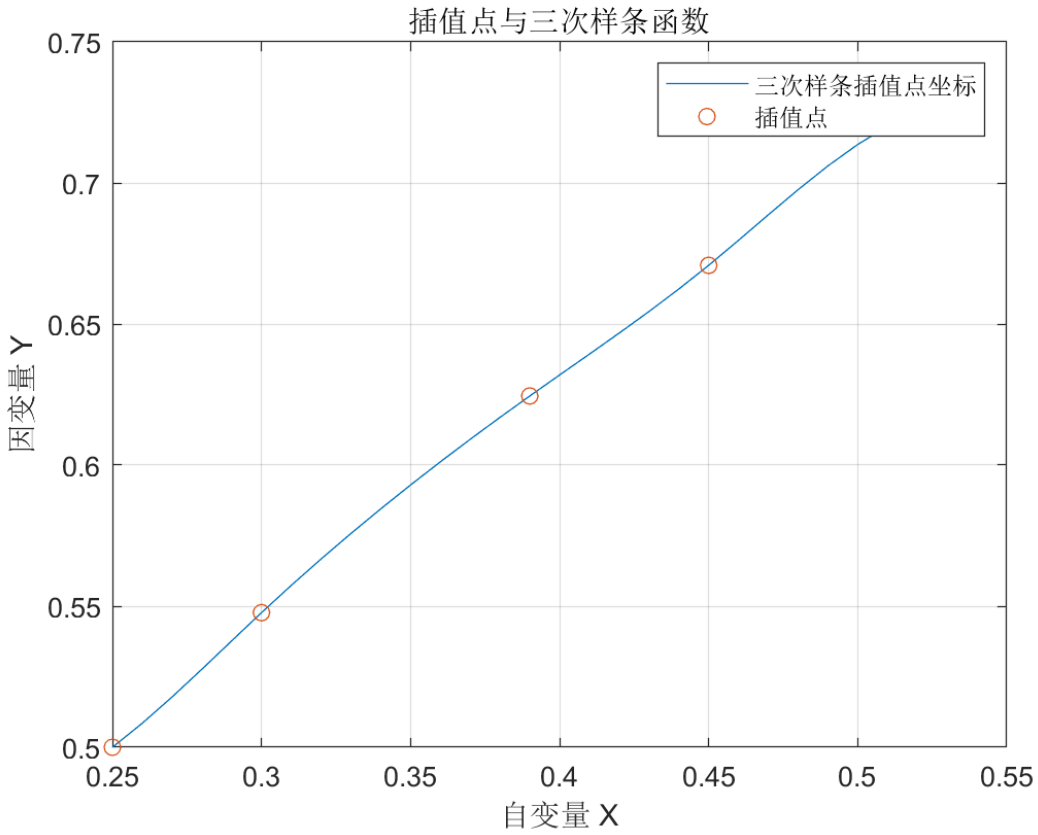
12. 求三次样条函数 $s(x)$,已知

| $x_i$ | 0.25 | 0.30 | 0.39 | 0.45 | 0.53 |
|---|---|---|---|---|---|
| $y_i$ | 0.500 0 | 0.547 7 | 0.624 5 | 0.670 8 | 0.728 0 |

和边界条件

$$s'(0.25)=1.000\ 0, s'(0.53)=0.686\ 8$$

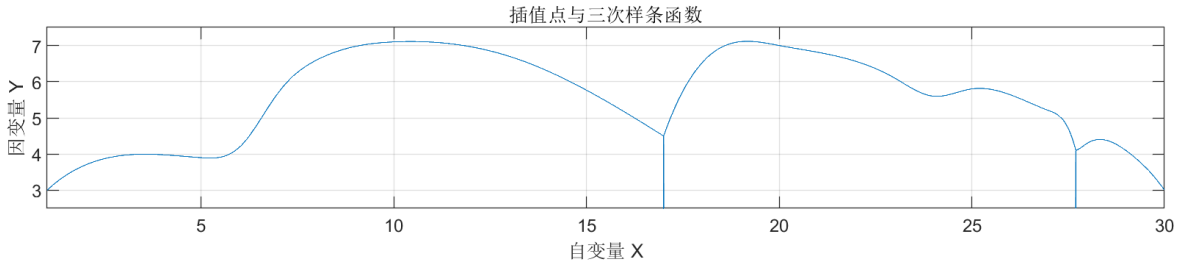
属于第一边界值 (固定边界) 问题, 我们将数值写入, 调用写好的 matlab 代码 (见附录), 可以得到运行后的结果如下:




以及所要求的三次样条函数 $s(x)$:

$$
\begin{aligned}
s(x) = &\left(-\frac{6869378826224293}{140737488355328}x^3 + \frac{5928974525113843}{140737488355328}x^2 - \frac{6255554967822417}{562949953421312}x + \frac{1584912528053635}{1125899906842624}\right) \\
&+ \left(\frac{8186227545664405}{2251799813685248}x^3 - \frac{5711533743453145}{1125899906842624}x^2 + \frac{857972261916083}{281474976710656}x - \frac{18774740554563}{2251799813685248}\right) \\
&+ \left(\frac{1381845895029699}{70368744177664}x^3 - \frac{3348843223024429}{140737488355328}x^2 + \frac{5826390871776913}{562949953421312}x - \frac{2156206841485829}{2251799813685248}\right) \\
&+ \left(-\frac{251589506805309}{4398046511104}x^3 + \frac{2812701846886277}{35184372088832}x^2 - \frac{5113245056812063}{140737488355328}x + \frac{3402853954482317}{562949953421312}\right).
\end{aligned}
\tag{18}
$$

对于第三小问, 我们只需要反复将三个曲线的数据代入我们曾写好的 matlab 代码中, 并将所得到的向量合并, 并利用 plot 函数输出图像即可:



## 3.4 结论

### 3.4.1 算法概述

我们以固定边界问题的代码为例, 对算法进行概述:
(1) **three1** 函数, 计算三次样条插值中使用的参数.

- **输入**:
  - $X$ 和 $Y$: 给定的插值点的 x 和 y 值.
  - $y_0$ 和 $y_n$: 分别是第一个和最后一个插值点的导数值（自然边界条件.
- **输出**:
  - $D$: 系数矩阵.
  - $h$: 每两个相邻插值点之间的差值.
  - $A$: 差商表.
  - $g$: 辅助向量, 用于求解线性系统.
  - $M$: 三次样条插值的二阶导数值.
- **算法概述**:
  1. 初始化 $A$, $D$, $g$ 和其他所需变量.
  2. 计算差商表 $A$.
  3. 计算每两个相邻插值点之间的差值 $h$.
  4. 构建三对角系数矩阵 $D$ 和辅助向量 $g$.
  5. 使用 $\backslash$ 运算符求解线性系统得到 $M$.

(2)**threesimple1** 函数使用上述的参数来为给定的 x 值计算插值.

- **输入**:
  - $X$ 和 $Y$: 给定的插值点的 x 和 y 值.
  - $x$: 要插值的 x 值.
  - $y_0$ 和 $y_n$: 分别是第一个和最后一个插值点的导数值.
- **输出**:

- $s$: 对应于输入 $x$ 的插值 y 值.

- **算法概述**:

  1. 使用 three1 函数计算 $D$, $h$, $A$, $g$ 和 $M$.
  2. 对于每个要插值的 $x$ 值:
     - 确定它属于哪个插值区间.
     - 使用三次样条插值公式计算 $s$ 值.

#### 3.4.2 总结

三次样条插值是一种插值方法, 它使用三次多项式来拟合数据点, 并确保相邻的多项式在接缝处具有连续的一阶、二阶导数. 这种方法在许多应用中都很受欢迎, 因为它可以产生光滑且自然的曲线.

# 4 附录: MATLAB 程序

## 4.1 数值微分二点与三点格式

### 4.1.1 主函数

```
    function [two_point_values, three_point_values] = Numerical_differentiation_a(f, true
% f is the function handle
% true_derivative is the actual derivative value at x=1

h_values = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001];
x = 1;

two_point_values = zeros(length(h_values), 2); % [numerical derivative, error]
three_point_values = zeros(length(h_values), 2); % [numerical derivative, error]

for i = 1:length(h_values)
    h = h_values(i);

    % Two point formula
    two_point_derivative = (f(x+h) - f(x)) / h;
    two_point_values(i, 1) = two_point_derivative;
    two_point_values(i, 2) = abs(two_point_derivative - true_derivative);
```

```
        % Three point formula
        three_point_derivative = (-3*f(x) + 4*f(x+h) - f(x+2*h)) / (2*h);
        three_point_values(i, 1) = three_point_derivative;
        three_point_values(i, 2) = abs(three_point_derivative - true_derivative);
    end
end
```

#### 4.1.2 测试程序

```
    % Test the function
f = @(x) exp(x);
true_derivative_at_1 = exp(1);

[two_point, three_point] = Numerical_differentiation_a(f, true_derivative_at_1);
disp('Two Point Formula:');
disp(two_point);
disp('Three Point Formula:');
disp(three_point);
```

### 4.2 五点中心差分格式

#### 4.2.1 主函数

```
    function [five_point_center_differentiation] = Numerical_differentiation_b(f, true_deriva
    % f is the function handle
    % true_derivative is the actual derivative value at x=1

    h_values = [1, 0.5, 0.1, 0.01];
    x = 1;

    five_point_center_differentiation = zeros(length(h_values), 2); % [numerical derivative,
    for i = 1:length(h_values)
        h = h_values(i);
```

```
        % Five point center differentiation formula
        five_point_center_derivative = (- 8*f(x-h) + 8*f(x+h) - f(x+2*h) + f(x-2*h)) / (12*h);
        five_point_center_differentiation(i, 1) = five_point_center_derivative;
        five_point_center_differentiation(i, 2) = abs(five_point_center_derivative - true_deri
    end
end
```

#### 4.2.2 测试函数

```
    % Test the function
f = @(x) exp(x);
true_derivative_at_1 = exp(1);

results = Numerical_differentiation_b(f, true_derivative_at_1);

disp(results);
```

### 4.3 非均匀网格下的三点格式

#### 4.3.1 主函数

```
    function [Inhomogenous_differentiation] = Numerical_differentiation_c(f, true_derivative)
    % f is the function handle
    % true_derivative is the actual derivative value at x=1

    h_0 = 0.001;
    h_1 = 0.002;
    x = 1;

    Inhomogenous_differentiation = zeros(1, 2); % [numerical derivative, error]

    % Five point center differentiation formula
    Inhomogenous_derivative =  (h_0*f(x+h_1))/((h_0+h_1)*h_1) - (h_1*f(x-h_0))/((h_0+h_1)*h_0)
        (f(x)*(h_1-h_0))/(h_0*h_1);
    Inhomogenous_differentiation(1, 1) = Inhomogenous_derivative;
```

```
    Inhomogenous_differentiation(1, 2) = abs(Inhomogenous_derivative - true_derivative);
end
```

#### 4.3.2 测试函数

```
    % Test the function
f = @(x) exp(x);
true_derivative_at_1 = exp(1);

results = Numerical_differentiation_c(f, true_derivative_at_1);

disp(results);
```

### 4.4 Hermite 插值公式

#### 4.4.1 主函数

```
    function [H, coefficients] = Hermite_interpolation(x, y, yp)
    % 输入: x = [x1, x2], y = [y1, y2], yp = [y'1, y'2]
    % 输出: H = Hermite 多项式 (作为一个函数句柄)
    %       coefficients = Hermite 多项式的各项系数

    % 使用符号计算
    syms x_val;

    % 定义 Lagrange 基函数
    h_1 = (1 + 2 * (x_val - x(1))/(x(2) - x(1))) * ((x_val - x(2))/(x(1) - x(2))) *((x_val - x
    h_2 = (1 + 2 * (x_val - x(2))/(x(1) - x(2))) * ((x_val - x(1))/(x(2) - x(1))) *((x_val - x

    % 定义 Lagrange 基函数的导数
    H_1 = (x_val - x(1)) * ((x_val - x(2))/(x(1) - x(2))) *((x_val - x(2))/(x(1) - x(2)));
    H_2 = (x_val - x(2)) * ((x_val - x(1))/(x(2) - x(1))) *((x_val - x(1))/(x(2) - x(1)));

    % 定义 Hermite 插值多项式
    H_sym = y(1) * h_1 + y(2) * h_2 + yp(1) * H_1 + yp(2) * H_2;
```

```
    % 获取 Hermite 插值多项式的各项系数
    coefficients = coeffs(H_sym, x_val);

    % 将 Hermite 多项式转换为函数句柄
    H = matlabFunction(H_sym);
end
```

### 4.4.2 测试函数

```
    % 定义测试数据
x = [0, 1];
y = [1, 2];
yp = [0.5, 0.5];

% 获取 Hermite 插值多项式和其系数
[H, coefficients] = Hermite_interpolation(x, y, yp);

% 在某些点上评估 Hermite 插值多项式
x_test = 0;
y_test = H(x_test);
fprintf('Hermite interpolated value at x = %.2f is y = %.2f\n', x_test, y_test);

% 绘制图形
x_range = linspace(x(1), x(2), 100);
y_range = arrayfun(H, x_range);
figure;
plot(x_range, y_range, 'b-', 'LineWidth', 2); % 插值多项式
hold on;
plot(x, y, 'ro', 'MarkerSize', 8); % 已知数据点
xlabel('x');
ylabel('y');
title('Hermite Interpolation');
legend('Interpolated Curve', 'Data Points');
grid on;
```

## 4.5 样条插值函数公式 (固定边界条件)

### 4.5.1 主函数

```
    function [D,h,A,g,M]=three1(X,Y,y0,yn)
%       自然边界条件的三次样条函数 (第一种边界条件)
%       此函数为 M 值求值函数
%       D,h,A,g,M 输出量分别为系数矩阵 D, 插值宽度 h, 差商表 A, g 值,M 值
        n=length(X);
        A=zeros(n,n);A(:,1)=Y';D=zeros(n,n);g=zeros(n,1);
        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
        end

        for i=1:n-1
            h(i)=X(i+1)-X(i);
        end
        for i=1:n
            D(i,i)=2;
            D(1,2)=1;
            D(n,n-1)=1;
            if (i==1)
                g(i,1)=6/h(i)*(A(2,2)-y0);
            elseif (i==n)
                    g(i,1)=6/h(i-1)*(yn-A(i,2));
            else
                g(i,1)=(6/(h(i-1)+h(i)))*(A(i+1,2)-A(i,2));
            end

        end
        for i=1:n-2
            u(i)=h(i)/(h(i)+h(i+1));
            n(i)=1-u(i);
            D(i+1,i+2)=n(i);
```

```
            D(i+1,i)=u(i);              % 改到这里
        end
        M=D\g;
        %M=[0;M;0];
end

function f=dispf(a)   % 输出函数表达式
    syms x;
    f=a(:,1).*x^3+a(:,2).*x^2+a(:,3).*x+a(:,4);
end

function [s,s1,a]=threesimple1(X,Y,x,y0,yn)
%       三次样条插值函数第一类型代码
%
%       根据三次样条参数函数求出的 D,h,A,g,M
%       x 表示求解插值点函数点, X 为已知插值点

% output:
%       s1 表示三次样条插值函数插值点对应的函数值
%       a 为分段函数各项系数矩阵
%       s 为了画图生成丝滑曲线细化区间点对应的函数值
        [D,h,A,g,M]=three1(X,Y,y0,yn)
        n=length(X); m=length(x);
        for t=1:m
           for i=1:n-1
              if (x(t)<=X(i+1))&&(x(t)>=X(i))
                 p1=M(i,1)/(6*h(i));
                 p2=M(i+1,1)/(6*h(i));
                 p3=(A(i,1)-M(i,1)/6*(h(i))^2)/h(i);
                 p4=(A(i+1,1)-M(i+1,1)/6*(h(i))^2)/h(i);
                 p(i,:)=[p1,p2,p3,p4];
                 a1=p(i,2)-p(i,1);
                 a2=3*p(i,1)*X(i+1)-3*p(i,2)*X(i);
                 a3=3*p(i,2)*(X(i))^2-3*p(i,1)*(X(i+1))^2+p(i,4)-p(i,3);
                 a4=p(i,1)*(X(i+1))^3-p(i,2)*(X(i))^3+p(i,3)*X(i+1)-p(i,4)*X(i);
```

```
                a(i,:)=[a1,a2,a3,a4];
                s1(i)=3*a1*(X(i))^2+2*a2*(X(i))+a3;
                s(t)=p1*(X(i+1)-x(t))^3+p2*(x(t)-X(i))^3+p3*(X(i+1)-x(t))+p4*(x(t)-X(i));
                break;
            else
                s(t)=0;
            end
        end
    end
end
```

### 4.5.2 测试函数

```
    clc
clear all
format short
x=[0.25, 0.30, 0.39, 0.45, 0.53];
y=[0.500, 0.5477, 0.6245, 0.6708, 0.7280];
y0=0.8 ;          %  S'(x0)=f'(x0)=y0
yn=0.2;           %  S'(xn)=f'(xn)=yn
x0=0.25:0.01:0.53;
[s,s1,a]=threesimple1(x,y,x0,y0,yn)
figure
plot(x0,s)        % 绘制第一边界条件插值函数图像
hold on
grid on
plot(x,y,'o')
xlabel('自变量 X'), ylabel('因变量 Y')
title('插值点与三次样条函数')
legend('三次样条插值点坐标','插值点')
S=dispf(a)
```

## 4.6 样条插值函数公式 (自然边界条件)

### 4.6.1 主函数

```
function [D,h,A,g,M]=three2(X,Y,y0,yn)
%       第二边界条件的三次样条函数 (包含自然边界条件)
%       y0,yn 表示的是 S''(x0)=f''(x0)=y0, S''(xn)=f''(xn)=yn, 自然边界即条件值为 0
%       此函数为 M 值求值函数
%       D,h,A,g,M 输出量分别为系数矩阵 D, 插值宽度 h, 差商表 A, g 值,M 值
        n=length(X);
        A=zeros(n,n);A(:,1)=Y';D=zeros(n-2,n-2);g=zeros(n-2,1);
        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
        end

        for i=1:n-1
            h(i)=X(i+1)-X(i);
        end
        for i=1:n-2
            D(i,i)=2;
            if (i==1)
                g(i,1)=(6/(h(i+1)+h(i)))*(A(i+2,2)-A(i+1,2))-h(i)/(h(i)+h(i+1))*y0;
            elseif (i==(n-2))
                g(i,1)=(6/(h(i+1)+h(i)))*(A(i+2,2)-A(i+1,2))-(1-h(i)/(h(i)+h(i+1)))*yn;
            else
                g(i,1)=(6/(h(i+1)+h(i)))*(A(i+2,2)-A(i+1,2));
            end
        end
        for i=2:n-2
            u(i)=h(i)/(h(i)+h(i+1));
            n(i-1)=h(i)/(h(i-1)+h(i));
            D(i-1,i)=n(i-1);
            D(i,i-1)=u(i);
        end
        M=D\g;
        M=[y0;M;yn];
end
```

```
function s=threesimple2(X,Y,x,y0,yn)
%        第二边界条件函数
%        s 函数表示三次样条插值函数插值点对应的函数值
%        根据三次样条参数函数求出的 D,h,A,g,M
%        x 表示求解插值点函数点, X 为已知插值点
         [D,h,A,g,M]=three2(X,Y,y0,yn)
         n=length(X); m=length(x);
         for t=1:m
            for i=1:n-1
               if (x(t)<=X(i+1))&&(x(t)>=X(i))
                  p1=M(i,1)*(X(i+1)-x(t))^3/(6*h(i));
                  p2=M(i+1,1)*(x(t)-X(i))^3/(6*h(i));
                  p3=(A(i,1)-M(i,1)/6*(h(i))^2)*(X(i+1)-x(t))/h(i);
                  p4=(A(i+1,1)-M(i+1,1)/6*(h(i))^2)*(x(t)-X(i))/h(i);
                  s(t)=p1+p2+p3+p4;
                  break;
               else
                   s(t)=0;
               end
            end
         end
end
```

### 4.6.2 测试函数

```
    x=[75,76,77,78,89,80];
y=[2.768,2.833,2.903,2.979,3.062,3.153];
y0=0;          %  S''(x0)=f''(x0)=y0
yn=0;          %  S''(xn)=f''(xn)=yn
x0=75:0.1:80;
s=threesimple2(x,y,x0,y0,yn)
plot(x0,s)        % 绘制第二边界条件插值函数图像
hold on
grid on
```

```
plot(x,y,'o')
axis([74 81 2.7 3.2])
xlabel('自变量 X'), ylabel('因变量 Y')
title('插值点与三次样条函数')
legend('三次样条插值点坐标','插值点')
```