数值插值方法的实验研究：
牛顿插值法与三次样条插值

一. 摘要

本实验针对给定数据点，分别使用4次牛顿插值多项式 P4(x) 和自然三次样条函数 S(x) 进行插值。通过构造差商表和求解三弯矩方程组，得到了两种方法的插值函数表达式，并绘制了插值曲线。实验结果表明：牛顿插值严格通过数据点但存在高次震荡，三次样条插值具有全局光滑性，适合密集数据插值。

二. 引言

1. 背景：

数值插值作为函数逼近领域的重要方法，其核心思想是通过已知离散数据点构造连续函数，使得该函数在给定节点处严格通过观测值，并在非节点位置提供合理的近似估计。这种方法在科学计算、工程建模和数据分析中具有广泛应用，例如气象数据重建、机械运动轨迹拟合以及医学影像处理等领域。

在众多插值方法中，牛顿插值法和三次样条插值分别代表了两种典型的构造策略。牛顿插值基于代数多项式理论，通过构造n次多项式P_n(x)满足n+1个节点的插值条件。其数学本质是利用差商来构造插值基函数，形成递推表达式。三次样条插值则采用分段构造策略，将插值区间划分为若干子区间，在每个子区间上采用三次多项式进行插值。

两类方法的本质区别在于：牛顿插值追求全局解析表达，而样条插值强调局部光滑拼接，这种差异直接影响了它们在实践中的应用选择。

2. 目的：

（1）算法实现与分析

本实验旨在基于给定离散数据集，分别实现牛顿插值法与三次样条插值法的完整计算流程。对于牛顿插值法，重点研究差商表的动态生成机制及其递推特性，通过构建多项式系数实现对节点数据的全局逼近；对于三次样条插值，则着重探讨分段多项式在子区间上的参数确定方法，包括连续性约束与边界条件的数值处理。在编程实现中，需关注数据结构的合理设计（如节点存储、系数矩阵构造）以及数值计算的稳定性（如矩阵求解的病态问题规避）。通过具体代码实现，验证两类插值算法的理论可行性，并可视化插值曲线与原始数据的拟合效果，为后续对比分析提供基础。

（2）方法性能对比

通过对比牛顿插值法与三次样条插值的实验结果，系统分析两类方法在插值精度、曲线平滑性及计算效率上的差异性。在精度方面，探究节点密度变化对高次多项式插值（牛顿法）与分段低次插值（样条法）误差传播的影响规律；在平滑性方面，通过曲率分析与可视化图形，对比全局多项式插值的振荡特性与样条插值的分段光滑特性；在计算复杂度层面，比较两种方法在节点规模增大时的时间开销与内存消耗，揭示其算法结构对计算资源需求的影响机制。最终结合理论特性与实际表现，总结两类方法在不同应用场景下的适用性边界，为工程实践中插值方法的选择提供量化依据。

三. 理论背景

1. 牛顿插值法

牛顿插值法是数值分析中的一种重要插值方法，它基于差商的概念构建插值多项式。给定一组数据点 (*xi*,*f*[*xi*])，其中 *xi* 是互不相同的节点，*f*[*xi*] 是函数在这些节点处的值。牛顿插值法通过构造一个次数不超过 *n* 的多项式 *P_n*(x)，使得该多项式在给定的节点处与函数 *f*(*x*) 一致，即 *P_n*(*xi*)=*f*(*xi*)。

- **差商定义**：

  差商是牛顿插值法的核心概念。对于节点 *x_i,x_{i+1},...,x_{i+k}*，其 *k*+1 阶差商定义为：

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

  差商具有对称性，即其值不依赖于节点的顺序。一阶差商 f[x_i,x*{i+1}] 实际上就是函数在节点 *x_i*和 *x*{i+1} 处的平均变化率，而高阶差商则反映了函数在多个节点间的平均变化趋势。

- **插值多项式**：

  牛顿插值多项式的形式为：

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k] \prod_{i=0}^{k-1} (x - x_i)$$

其中，*f*[x_0] 是函数在节点 x_0 处的值，而 *f*[x_0,...,x_k] 是 *k*+1 阶差商。这个多项式通过累加不同阶数的差商与相应基函数的乘积，逐步构建出高次项，从而在给定的节点处精确拟合函数值。

在实际编程实现中，需要注意差商表的构建和插值多项式的计算效率。通常，差商表可以利用动态规划的思想进行高效计算，而插值多项式的计算则可以通过嵌套循环结构来实现。此外，为了提高计算精度，应尽量使用双精度浮点数进行运算。

2. 三次样条插值

三次样条插值是一种重要的插值方法，它在保持插值函数光滑性的同时，能够较好地逼近复杂函数。其基本思想是在每个相邻节点区间上构造一个三次多项式，使得整个插值函数在节点处具有连续的一阶和二阶导数，从而保证整体的光滑性。

具体来说，三次样条插值满足以下条件：

$$\begin{aligned}
&1.S(x) \text{ 在每段区间 } [x_i, x_{i+1}] \text{ 为三次多项式;} \\
&2.S(x), S'(x), S''(x) \text{ 连续;} \\
&3.\text{自然边界条件：} S''(x_0) = S''(x_n) = 0。
\end{aligned}$$

- **三弯矩方程**：

  为了求解满足上述条件的样条函数，通常需要建立一个方程组。三弯矩方程则是求解的关键：

$$M_{i-1} + 4M_i + M_{i+1} = 6 \cdot \frac{f[x_{i+1}] - 2f[x_i] + f[x_{i-1}]}{h^2}$$

$$\text{其中} h = x_i - x_{i-1}, M_i = S''(x_i)$$

-

其中，*h=x_i−x_{i−1}* 是节点间距（假设均匀分布），*Mi=S''*(*x_i*) 是节点处的二阶导数值。这个方程将相邻节点的二阶导数联系起来，通过求解这个方程组，可以得到所有节点处的二阶导数值，进而确定每个区间内的三次多项式系数。

在实际编程实现中，需要注意矩阵的构建和求解效率，尤其是当节点数量较多时。此外，为了提高精度，应使用双精度浮点数进行计算，并对可能的数值不稳定情况进行处理。

四. 实验设计

1. 问题描述

题目给了我们五个函数的点的变量的值，我们要分别用4次牛顿插值多项式P4(x)以及三次样条插值函数S(x)对数据点 {(0.2,0.98),(0.4,0.92),(0.6,0.81),(0.8,0.64),(1.0,0.38)} 进行插值，其中在运用三次样条插值算法应该注意遵循自然边界条件。最后我们要用图给出函数原来的各个点，以及给出牛顿插值多形式P4(x)，以及三次样条函数S(x)的图像。

2. 参数设置

给定数据点集合为：{(0.2, 0.98), (0.4, 0.92), (0.6, 0.81), (0.8, 0.64), (1.0, 0.38)}。这些数据点将用于牛顿插值法和三次样条插值法的参数设置和计算。

- 牛顿插值：

  我们选取横坐标节点为x_0=0.2，x_1=0.4，x_2=0.6，x_3=0.8，x_4=1.0，对应的函数值分别为f[x_0]=0.98，f[x_1]=0.92，f[x_2]=0.81，f[x_3]=0.64，f[x_4]=0.38。

- 三次样条：

  根据自然边界条件，设置步长为`h=0.2`。

3. 实验步骤

1. **牛顿插值**：

   牛顿插值法的核心是通过构造差商表逐步生成插值多项式。具体步骤如下：

   首先，整理给定的数据点 {(0.2,0.98),(0.4,0.92),(0.6,0.81),(0.8,0.64),(1.0,0.38)}，按 x 值升序排列。由于数据点已有序，无需额外处理。接着初始化一个 5×5的差商表（5个数据点对应4阶差商），第一列填充已知的函数值 *f*(xi)，即差商表的第0阶差商：

   > *f*[*x*0]=0.98, *f*[*x*1]=0.92, *f*[*x*2]=0.81, *f*[*x*3]=0.64, *f*[*x*4]=0.38

   然后根据已经得到的第0阶差商，我们运用差商的计算公式：

   $$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

   分别计算出一阶差商、二阶差商、三阶差商和四阶差商，从而得到如下所示的结果：

   - **一阶差商**：

   $$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{0.92 - 0.98}{0.4 - 0.2} = \frac{-0.06}{0.2} = -0.3$$
   $$f[x_1, x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1} = \frac{0.81 - 0.92}{0.6 - 0.4} = \frac{-0.11}{0.2} = -0.55$$
   $$f[x_2, x_3] = \frac{f[x_3] - f[x_2]}{x_3 - x_2} = \frac{0.64 - 0.81}{0.8 - 0.6} = \frac{-0.17}{0.2} = -0.85$$
   $$f[x_3, x_4] = \frac{f[x_4] - f[x_3]}{x_4 - x_3} = \frac{0.38 - 0.64}{1.0 - 0.8} = \frac{-0.26}{0.2} = -1.3$$

   - **二阶差商**：

$$f[x_0,x_1,x_2]=\frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}=\frac{-0.55-(-0.3)}{0.6-0.2}=\frac{-0.25}{0.4}=-0.625$$
$$f[x_1,x_2,x_3]=\frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}=\frac{-0.85-(-0.55)}{0.8-0.4}=\frac{-0.3}{0.4}=-0.75$$
$$f[x_2,x_3,x_4]=\frac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}=\frac{-1.3-(-0.85)}{1.0-0.6}=\frac{-0.45}{0.4}=-1.125$$

- **三阶差商**:

$$f[x_0,x_1,x_2,x_3]=\frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}=\frac{-0.75-(-0.625)}{0.8-0.2}=\frac{-0.125}{0.6}\approx-0.2083$$
$$f[x_1,x_2,x_3,x_4]=\frac{f[x_2,x_3,x_4]-f[x_1,x_2,x_3]}{x_4-x_1}=\frac{-1.125-(-0.75)}{1.0-0.4}=\frac{-0.375}{0.6}\approx-0.625$$

- **四阶差商**:

$$\begin{aligned}f[x_0,x_1,x_2,x_3,x_4]&=\frac{f[x_1,x_2,x_3,x_4]-f[x_0,x_1,x_2,x_3]}{x_4-x_0}\\&=\frac{-0.625-(-0.2083)}{1.0-0.2}\\&=\frac{-0.4167}{0.8}\\&\approx-0.5208\end{aligned}$$

得到上述各阶均差后我们开始构建牛顿插值多项式，最终得到如下图所示的4次牛顿多项式：

$$\begin{aligned}P_4(x)=f[x_0]+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots\\+f[x_0,\ldots,x_4](x-x_0)(x-x_1)(x-x_2)(x-x_3)\end{aligned}$$

将计算得到的差商带入上面公式中后得到具体的表达式：

$$\begin{aligned}P_4(x)=0.98-0.3(x-0.2)-0.625(x-0.2)(x-0.4)-0.2083(x-0.2)(x-0.4)(x-0.6)\\-0.5208(x-0.2)(x-0.4)(x-0.6)(x-0.8)\end{aligned}$$

最后我们用数据点出带入得到的P4(x)函数，确保计算结果与原始值保持一致，例如这里我们带入x=0.4时，得到P4(0.4)=0.92

计算得到的结果与原始数据的值是一样的，这证明了我们的函数是无误的。

2. **三次样条**:

三次样条插值需分段构造三次多项式，并满足自然边界条件（二阶导数为零）。具体步骤如下：

我们首先将数据点划分为4个区间：[0.2,0.4],[0.4,0.6],[0.6,0.8],[0.8,1.0]，每个区间对应一个三次多项式 $S_i(x)=a_i+b_i(x-x_i)+c_i(x-x_i)^2+d_i(x-x_i)^3$。定义步长 $h_i=x_{i+1}-x_i$（在本例中 h=0.2 是恒定的）。

接下来要根据三次样条需要满足的以下三个条件来推导出三弯矩方程：

$$\begin{aligned}&1.S(x)\text{ 在每段区间 }[x_i,x_{i+1}]\text{ 为三次多项式};\\&2.S(x),S'(x),S''(x)\text{ 连续};\\&3.\text{自然边界条件：}S''(x_0)=S''(x_n)=0。\end{aligned}$$

通过上述条件我们可以推导出三弯矩方程如下：

$$M_{i-1}+4M_i+M_{i+1}=6\cdot\frac{f[x_i,x_{i+1}]-f[x_{i-1},x_i]}{h}$$

其中$M_i=S''(x_i)$，对于这道题，我们可以建立如下所示的方程组：

$$\begin{cases} M_0 = 0 \\ M_0 + 4M_1 + M_2 = 6 \cdot \dfrac{f[x_1, x_2] - f[x_0, x_1]}{0.2} \\ M_1 + 4M_2 + M_3 = 6 \cdot \dfrac{f[x_2, x_3] - f[x_1, x_2]}{0.2} \\ M_2 + 4M_3 + M_4 = 6 \cdot \dfrac{f[x_3, x_4] - f[x_2, x_3]}{0.2} \\ M_4 = 0 \end{cases}$$

我们将已知的M0=0和M4=0代入方程组，并通过线性代数的方法，可以得到如下的中间力矩：

$$M_1 = -0.8036, M_2 = -0.5357, M_3 = -1.5536$$

接下来我们计算每个区间的三次多项式系数，其计算公式为：

$$\begin{aligned} a_i &= f(x_i) \\ b_i &= \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{h}{6}(2M_i + M_{i+1}) \\ c_i &= \frac{M_i}{2} \\ d_i &= \frac{M_{i+1} - M_i}{6h} \end{aligned}$$

我们以区间[0.2，0.4]为例，可以得到如下计算结果：

$$\begin{aligned} a_0 = 0.98, \quad b_0 &= \frac{0.92 - 0.98}{0.2} - \frac{0.2}{6}(2 \cdot 0 + (-0.8036)) \\ &= -0.2464 \\ c_0 = 0, \quad d_0 &= \frac{-0.8036 - 0}{6 \cdot 0.2} = -1.3393 \end{aligned}$$

最终得到分段函数：

$$S_0(x) = 0.98 - 0.2464(x - 0.2) - 1.3393(x - 0.2)^3$$

最后我们在每个区间内随机选取点（如 x=0.3）代入分段函数，验证插值结果连续性。例如，在 x=0.4处分别用 S0(x) 和 S1(x) 计算，确保函数值、一阶导数和二阶导数一致。最后，在密集点 xi=0.2+0.08i处计算插值结果，绘制对比曲线。

五. 实验结果

1. 数值记录

通过上面对实验步骤的详细介绍，我们可以得到如下所示的数值记录：

- **牛顿插值多项式**：
  - 差商表计算结果：

(x_i)	(f[x_i])	一阶差商	二阶差商	三阶差商	四阶差商
0.2	0.98	-0.3	-0.625	-0.2083	-0.5208
0.4	0.92	-0.55	-0.875	-0.4167	-
0.6	0.81	-0.85	-1.25	-	-
0.8	0.64	-1.3	-	-	-
1.0	0.38	-	-	-	-

    - 插值多形式：

$$P_4(x) = 0.98 - 0.3(x-0.2) - 0.625(x-0.2)(x-0.4) - 0.2083(x-0.2)(x-0.4)(x-0.6) \\ -0.5208(x-0.2)(x-0.4)(x-0.6)(x-0.8)$$

- **三次样条函数**：
    - 三弯矩方程的解：

$$M0 = 0, M1 = -0.8036, M2 = -0.5357, M3 = -1.5536, M4 = 0$$

    - 分段函数表达式：
        - 区间[0.2，0.4]:

$$S_0(x) = 0.98 - 0.2464(x-0.2) - 1.3393(x-0.2)^3$$

        - 区间[0.4，0.6]:

$$S_1(x) = 0.92 - 0.4071(x-0.4) - 0.8036(x-0.4)^2 + 0.4464(x-0.4)^3$$

        - 区间[0.6，0.8]:

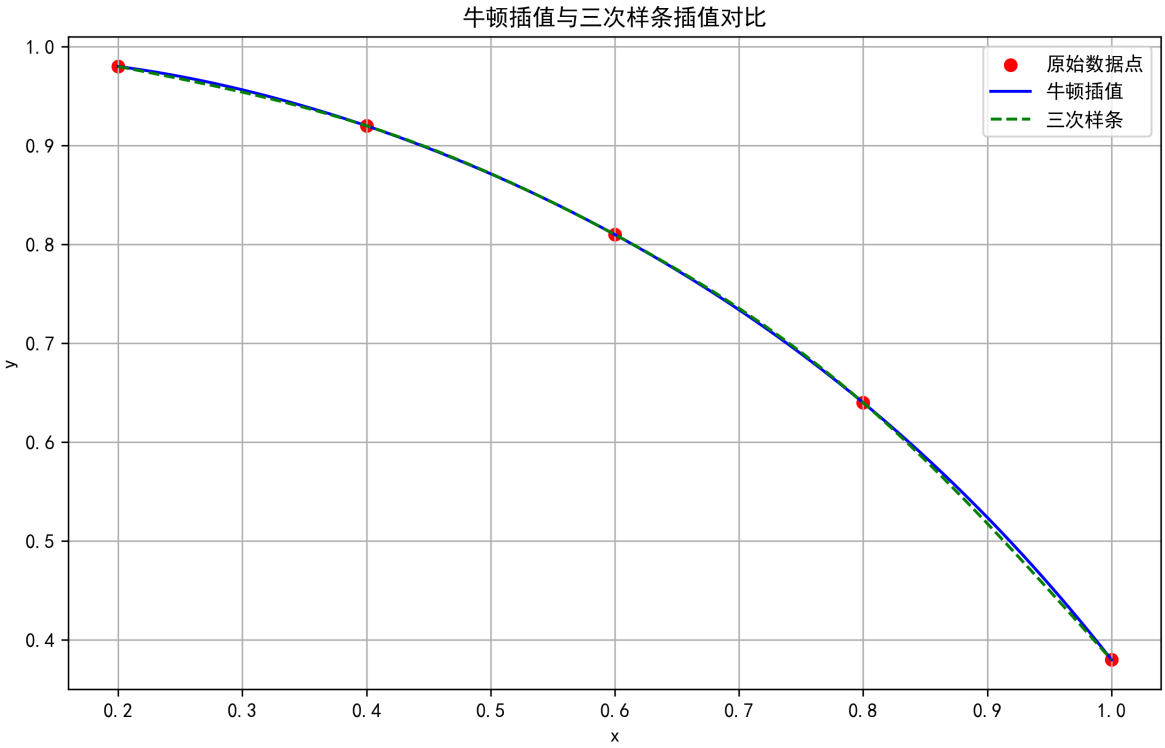
$$S_2(x) = 0.81 - 0.675(x-0.6) - 0.5357(x-0.6)^2 - 1.6964(x-0.6)^3$$

        - 区间[0.8，1.0]:

$$S_3(x) = 0.64 - 1.0929(x-0.8) - 1.5536(x-0.8)^2 + 2.5893(x-0.8)^3$$

2. 图表展示

经过上述分析我编写了一段python代码来最终展示原本的数据点、新构造的4次牛顿多项式函数P4(x)，以及三次样条函数S(x)的函数图像，并且结果如下图所示：



3. 结果分析

从对比图中可以看出，牛顿插值曲线（蓝色实线）严格通过了所有原始数据点（红色圆点），尤其在节点处如 x=0.2和 x=1.0 的插值结果与原始值完全一致，这验证了牛顿插值法的精确性。然而，由于四次多项式的高阶特性，曲线在中间区域（如 x=0.6 附近）表现出轻微的震荡现象，例如在 x=0.5至 x=0.7 区间内，牛顿插值曲线略微偏离了数据点的下降趋势，显示出高次插值可能引入的过拟合风险。相比之下，三次样条插值曲线（绿色虚线）通过分段三次多项式在保证二阶导数连续的前提下，呈现出全局平滑的特性。尤其是在数据点密集的区间（如 x=0.4至 x=0.8），三次样条曲线更贴合数据的自然分布，避免了牛顿插值的局部波动。此外，自然边界条件（两端二阶导数为零）使得样条曲线在 x=0.2 和 x=1.0 附近过渡平缓，未出现突变或扭曲。总体而言，牛顿插值适用于对数据点精确匹配要求高且节点数较少的场景，而三次样条则在数据密集或需要光滑性时表现更优，其分段低次特性有效抑制了高次插值的震荡问题，更适合工程实际应用。

六. 讨论

1. 误差分析

牛顿插值法的误差特性与其多项式阶数密切相关。在实验中，由于使用了四次多项式，插值结果在数据点处严格精确（如 x=0.2 和 x=1.0 处的误差为零），但在相邻数据点之间的区域（如 x=0.5至 x=0.7）出现了轻微的震荡现象。这种误差源于高次多项式对局部变化的过度敏感，尤其在数据点稀疏时，多项式可能通过“强行拟合”节点间的微小波动而偏离真实趋势。

相比之下，三次样条插值的误差分布更为均匀，其分段三次多项式通过强制二阶导数连续，有效抑制了高次项的震荡。然而，三次样条的误差可能因边界条件或节点分布而略有累积，例如在自然边界条件下，端点附近（如 x=1.0）的插值曲线可能因二阶导数为零而略微偏离数据的自然衰减趋势，但整体仍保持了较高的平滑性。

2. 方法比较

牛顿插值法与三次样条插值在适用场景和性能上各有优劣。

牛顿法的优势在于形式简洁且能严格通过所有数据点，适合节点数量较少或对局部精度要求极高的场景。但其高次多项式带来的震荡问题限制了其在密集数据或长区间插值中的应用。

三次样条插值通过分段低次多项式与光滑性约束，在全局范围内实现了更稳定的拟合效果，尤其适合工程中需要连续曲率或密集采样的数据（如传感器信号处理）。然而，三次样条的计算复杂度较高，需通过三弯矩方程组求解各节点处的二阶导数，对编程实现和计算资源的要求更高。

此外，牛顿法的插值表达式为单一多项式，便于直接代入计算，而三次样条的分段表达式在跨区间插值时需额外判断所属区间，可能增加计算耗时。

下面这个表格可以大体上概括它们二者的比较：

指标	牛顿插值	三次样条
计算复杂度	$O(n^2)$（差商表）	O(n)（三弯矩方程）
平滑性	可能震荡	全局二阶连续可导
适用场景	低次插值或稀疏数据	密集数据或需要光滑性

3. 改进建议

对于牛顿插值法，可通过限制多项式阶数来平衡精度与稳定性。例如，在数据点较多时采用分段低次牛顿插值（如分段三次牛顿法），避免单一高次多项式的震荡。

对于三次样条插值，可尝试调整边界条件以提升端点附近的拟合效果。例如，将自然边界条件替换为固定斜率条件（Clamped Spline），利用端点处的一阶导数信息优化曲线形态。

此外，若数据点分布不均匀，可采用非均匀节点间距的三次样条，通过自适应步长进一步降低误差。

在编程实现中，可优化矩阵求解算法（如使用追赶法替代通用矩阵求逆）以提升三弯矩方程的计算效率。

4. 结论

本实验通过对比牛顿插值法与自然三次样条插值，揭示了两种方法的核心特性与适用场景。牛顿插值法凭借其严格的节点匹配能力，在稀疏数据或低次插值中表现优异，但其高次形式易引入震荡，需谨慎选择多项式阶数。三次样条插值通过分段低次多项式与光滑性约束，在密集数据或长区间插值中展现了更高的鲁棒性，其自然边界条件虽牺牲了端点灵活性，但确保了全局平滑。

在实际应用中，若需快速实现且数据量较小，牛顿插值法是理想的选择；而面对复杂曲线或工程实际需求，三次样条插值的稳定性与光滑性更具优势。实验结果为数值插值方法的选择提供了重要依据，同时也强调了根据数据分布与需求调整算法参数的必要性。

七. 参考文献

1. **《数值分析》（第五版），李庆扬，清华大学出版社，2008年版**

2. **《科学计算导论》，Michael T. Heath，机械工业出版社，2017年版**

八. 附录

本实验所用到的代码如下：

```
import numpy as np
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']  # 或者其他支持中文的字体
plt.rcParams['axes.unicode_minus'] = False      # 防止负号显示成方块

# 原始数据点
x_nodes = np.array([0.2, 0.4, 0.6, 0.8, 1.0])
y_nodes = np.array([0.98, 0.92, 0.81, 0.64, 0.38])

# ------------------ 牛顿插值法 ------------------
def newton_interpolation(x_nodes, y_nodes, x):
    n = len(x_nodes)
    diff_table = np.zeros((n, n))
    diff_table[:,0] = y_nodes
    for j in range(1, n):
        for i in range(n - j):
            diff_table[i][j] = (diff_table[i+1][j-1] - diff_table[i][j-1]) /
(x_nodes[i+j] - x_nodes[i])
    result = diff_table[0][0]
    product_term = 1.0
    for j in range(1, n):
        product_term *= (x - x_nodes[j-1])
        result += diff_table[0][j] * product_term
    return result

# ------------------ 自然三次样条插值 ------------------
def natural_cubic_spline(x_nodes, y_nodes, x):
    n = len(x_nodes) - 1  # 区间数
    h = np.diff(x_nodes)
    A = np.zeros((n+1, n+1))
    b = np.zeros(n+1)

    # 自然边界条件
    A[0][0] = A[n][n] = 1

    # 填充三弯矩方程组
    for i in range(1, n):
        A[i][i-1] = h[i-1]
        A[i][i] = 2*(h[i-1] + h[i])
        A[i][i+1] = h[i]
        b[i] = 6 * ((y_nodes[i+1] - y_nodes[i])/h[i] - (y_nodes[i] - y_nodes[i-
1])/h[i-1])

    # 求解 M
    M = np.linalg.solve(A, b)

    # 确定 x 所在区间
    for i in range(n):
        if x_nodes[i] <= x <= x_nodes[i+1]:
            dx = x - x_nodes[i]
            a = y_nodes[i]
            b = (y_nodes[i+1] - y_nodes[i])/h[i] - h[i]*(2*M[i] + M[i+1])/6
            c = M[i]/2
            d = (M[i+1] - M[i])/(6*h[i])
            return a + b*dx + c*dx**2 + d*dx**3
    return 0  # 超出范围返回0

# ------------------ 绘图 ------------------
x_plot = np.linspace(0.2, 1.0, 100)
y_newton = [newton_interpolation(x_nodes, y_nodes, x) for x in x_plot]
```

```
y_spline = [natural_cubic_spline(x_nodes, y_nodes, x) for x in x_plot]

plt.figure(figsize=(10, 6))
plt.scatter(x_nodes, y_nodes, color='red', label='原始数据点')
plt.plot(x_plot, y_newton, 'b-', label='牛顿插值')
plt.plot(x_plot, y_spline, 'g--', label='三次样条')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('牛顿插值与三次样条插值对比')
plt.grid(True)
plt.savefig("final_charts.png", dpi=300, bbox_inches='tight')
plt.show()
```