基于欧拉法的光纤传感器曲线重构问题研究

摘要

随着光纤通信技术在人们日常生活中的进一步普及，光纤传感技术作为一种新型的传感器技术也正逐渐发展起来。光纤传感器主要依靠光波作为传输信号、光纤作为传输载体来感知外部环境信号。由于**光纤传感器**具有质地轻、体积小、易弯曲、灵敏度高、抗电磁能力强的优点，其在医学领域上已经得到了较为广泛的运用。因此通过传感器返回数据，获得结构实时应变信息，再通过解调出来的应变参数来重构得到结构的形变或位移对于光纤传感器的广泛运用至关重要。

本文主要研究了依据曲率进行**平面曲线重构**的问题。根据 Frenet 框架在二维条件下的简化模型，利用欧拉法、龙格-库塔法等算法实现了在已知离散点曲率的条件下完成了曲线的重构，并分析了模型在多方面上的误差。

针对问题一光纤上各点曲率的估算，本文首先论证了在给定条件下曲率与反射波长间的线性关系，并根据题目所给关系式，通过传感器读数估计曲线上特定传感点曲率依次为 ***2.219***、***2.217***、***2.233***、***2.230***、***2.236***、***2.222***；***2.986***、***2.978***、***2.973***、***2.981***、***2.984***、***2.975***，然后分析得到了样条插值法的优越性，借助**三次样条插值法**重新构建了光纤曲线，再通过该重构曲线求解特定横坐标下曲线的估计曲率值，如文中图 1 所示。

针对问题二根据离散点处的曲率进行曲线的重构，本文首先介绍了在三维曲线重构中广泛应用的 **Frenet 框架**，并推导出其退化至二维的一些基础算法。再利用其衍生得到的**小步长欧拉法**构造重构曲线。借助局部放大分析曲线类圆的特点，并绘制重构曲线的曲率图像验证该曲线类似螺旋线的走势，且保持总长相等的物理性质。得出曲线特点后本文又借助**多项式拟合法**再次进行曲线拟合，并将两种算法得到的曲线及曲率放在同一坐标系中对比，并无太大差异，从而证实了模型的可靠性。

针对问题三对已知三次多项式的曲率采样与曲线重构，本文首先给出了**等间距弧长采样**的优点和实现过程。对已知三次曲线进行采样得到图 7 之后又分别采取**前向欧拉法**、**四阶龙格-库塔法**以及**有限差分法**完成了平面曲线的重构，并从不同类型误差的角度综合分析了这三种不同算法重构曲线的效果。以此为启发本文又探究了模拟退火法在本题目中的应用及其拟合结果，其与目标曲线高度重合的结果验证了该算法的优越性。

关键字：光纤传感器　Frenet 框架　欧拉法　龙格-库塔法　平面曲线重构

目录

一、 问题重述

1.1 问题背景

通信技术的大规模普及，使得人们愈发注重更加高效稳定的光纤通信技术。光纤技术需要实时获得结构的应变信息，再根据所得数据得到光纤形变或位移进而对曲线进行重构。为解决这个问题，题目给出了两组不同初始条件下受力前后的光纤传感器返回的波长值，并且提供了传感器信号波长和曲线曲率的近似关系式。

1.2 问题提出

问题一：请根据表中给出的波长数据估计每个传感点的曲率。并将其置于坐标系中规定原始切线方向，估计特定横坐标对应点处的曲率。

问题二：针对受到外力的光纤重构一条平面曲线，使其满足问题一中得到的波长测量数据以及曲率，并对得到的平面曲线形状特点进行分析。

问题三：请根据给定的三次项平面曲线进行等间距弧长采样，计算采样点处曲率，并以此为根据重构一条平面曲线，分析得到曲线与给定的原始曲线之间出现的误差。

二、 问题分析

问题一：这一问根据题目给出波长 λ 和曲线曲率 κ 之间的近似关系

$$\kappa = \frac{\mathrm{c}\,(\lambda - \lambda_0)}{\lambda_0} \tag{1}$$

可将表 1 中各个传感器测得的波长数据转化为平面光栅各传感点的曲线曲率。针对第二小问，在给定起始点位置、方向及切线方向的情况下，本文将依据第一小问得到的各传感点曲线斜率，通过三次样条曲率插值法拟合曲线，并根据此拟合曲线估算出题目要求特定横坐标 x 轴相应位置曲率。

问题二：这一问需要综合考虑题目所给的各传感点波长以及第一问求得的传感点曲线曲率，构建数学模型获得光栅的平面重构曲线。本文将使用欧拉法对第一问获得的离散点曲率进行曲线重构，借助 Matlab 将测试 1 和测试 2 的两组拟合曲线展示在图表中，并从平滑度、趋势度、拟合度等多方面分析曲线特点。

问题三：这一问需首先对已知曲线 $y=x^3+x$ 进行适当的等间距弧长采样，获取多个离散曲率数据后，再基于这些数据，通过龙格-库塔法或其他算法重构一条拟合曲线。借助 Matlab 工具将拟合曲线与给定曲线展示在一张图表上，观察重构曲线产生的误差并分析其产生的原因.

三、 模型的假设

(1) 本文假设波长 λ 和曲线曲率 κ 在题目考查范围内始终满足线性关系式 (1)，且关系式 (1) 中的 c 视作常数 4200。

(2) 本文假设在对获得的离散的传感点曲率进行曲线拟合时，目标曲线应始终保持较高的平滑度，从而可以使用三次样条差分法进行基本的曲线拟合。

(3) 本文假设光纤上的传感器在水平状态和受到外力作用时都能保持稳定的等间距，且始终能保持返回该位置的波长值。

(4) 本文假设在对取样点曲率进行三次样条插值时，函数可在采样点之间用三次多项式近似表示，

四、 符号说明

符号	意义
λ_0	水平光纤初始测量波长（nm）
κ	曲线曲率
$\delta\mathbf{r}(s_n)$	曲线绝对位置误差
RMSE	曲线均方根误差
MAE	曲线平均绝对误差误差
$\lvert\delta\mathbf{r}_{\text{end}}\rvert$	曲线末端绝对误差
EPE	末端误差
FEM	前向欧拉法
RK4	四阶龙格-库塔法
FDM	有限差分法

五、 模型建立与求解

5.1 问题一平面各传感点曲率估算模型

光纤布拉格光栅（FBG）的反射波长和折射率周期的关系为:

$$\lambda_{\mathrm{C}} = 2 \cdot \Lambda \cdot n_{\mathrm{eff}} \tag{2}$$

其中 Λ 为光栅的栅格周期，n_{eff} 为等效折射率 [2]。

FBG 易受应力作用导致折射率变化而改变有效弹光系数，或因温度变化导致其热光系数和热膨胀系数的改变。FBG 反射波长与应变以及温度的关系如下:

$$\frac{\Delta\lambda_{\mathrm{C}}}{\lambda_{\mathrm{C}}} = (1 - p_{\mathrm{e}})\,\varepsilon + \left(\alpha + \frac{1}{n_{\mathrm{eff}}}\xi\right)\Delta T \tag{3}$$

其中 ε 为 FBG 的轴向应变，p_{e} 为有效弹光系数，ξ 为热光系数。

本题不讨论环境中的温度的变化，故可将 (3) 式简化为

$$\frac{\Delta\lambda_{\mathrm{C}}}{\lambda_{\mathrm{C}}} = (1 - p_{\mathrm{e}})\,\varepsilon \tag{4}$$

据此我们可以理解题目中指出曲率与反射波长满足的线性关系式 (1) 。

5.1.1 模型建立思路

首先，我们需要从给定的光纤传感器数据中计算曲率。根据题目描述和光纤传感器的工作原理，曲率 κ 可以通过已知关系式 (1) 与波长变化联系起来:

$$\kappa = \frac{c(\lambda - \lambda_0)}{\lambda_0}$$

其中，c 是一个已知常数（题目中给定为 4200），λ 是受力后的波长，而 λ_0 是初始波长。这一公式基于波长变化对光纤传感器产生的影响，将物理变化量转化为曲率的估计，为后续的曲线重构提供必要的数据。

从题目提供的表格数据中，可以获取各个传感点在不同初始状态下受力前后的波长值。首先，对于每个传感器位置，应用上述公式计算出曲率。给定的数据是六个传感点在两种不同初始状态下的波长，我们将得到两组曲率数据。

表 1　Curvature of target point

	FBG1	FBG2	FBG3	FBG4	FBG5	FBG6
Test1 曲率 κ	2.219	2.217	2.233	2.230	2.236	2.222
Test2 曲率 κ	2.986	2.978	2.973	2.981	2.984	2.975

由于传感器是间隔分布的，我们需要在传感器之间的位置估算曲率。可以使用多种插值方法，如线性插值、三次样条插值等，来在连续的区间内估算曲率。这一步骤对于平滑曲率分布并提供更精确的曲线重构非常关键。

线性插值假设传感器之间的曲率变化是线性的，这是最简单的插值方法，计算速度快，但可能无法准确反映曲率的真实变化，尤其是在曲率变化剧烈的区域。与线性插值相比，样条插值通常能提供更加平滑且接近真实曲率变化的结果。

因此我们选择采用三次样条插值法进行曲线重构，使用 Matlab 内置的 ‘fit‘函数，以 ‘’cubicinterp’‘选项执行插值，并对重构结果加以分析。

5.1.2 曲率数据的插值比较

我们使用 MATLAB 实现了上述三次样条插值方法，并基于实验数据计算了曲率值。以下为实现结果。

表 2　Estimated curvature of target point

横坐标 x（米）	0.3	0.4	0.5	0.6	0.7
测试一曲率 κ	2.2120	2.2126	2.2143	2.2167	2.2198
测试二曲率 κ	2.9832	2.9816	2.9799	2.9782	2.9765

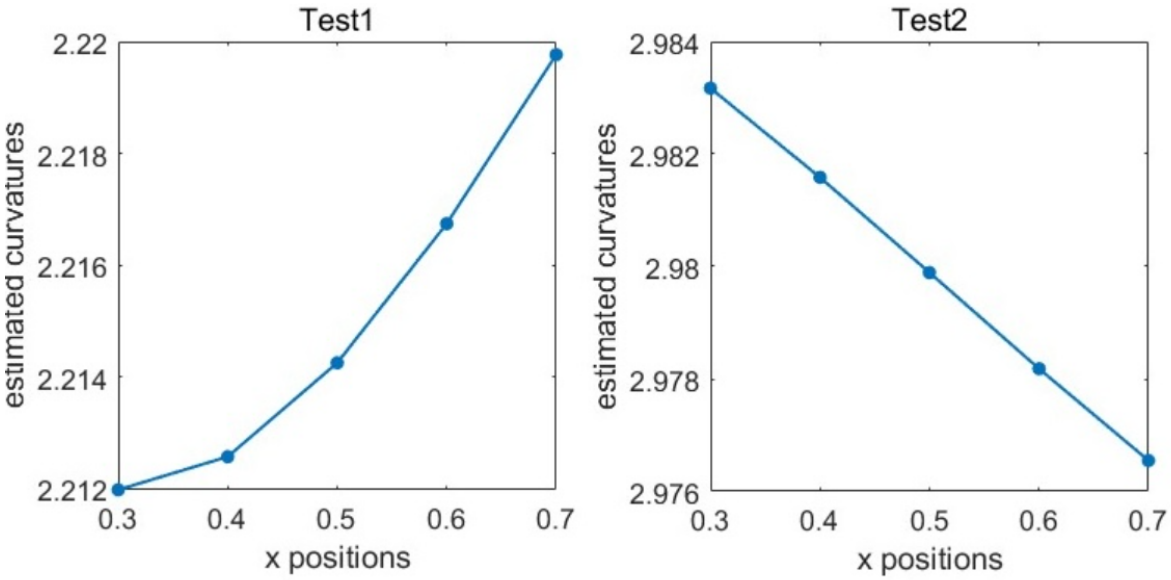

图 1　curvatureCalculation

5.2 问题二曲线重构模型

5.2.1 模型建立思路

问题一中，我们借助光纤传感器得到了相距 0.6m 的多个传感点处曲率值，但仅通过这些曲率值进行曲线坐标点拟合会因为曲线中间缺失较多曲率信息，从而导致坐标点拟合精度很低。故求得若干离散检测点的曲率后，需将离散点间的曲线段进行细分，再通过曲率插值的方法得到每一个微段的曲率值，从而提升坐标点拟合精度。这种方法可以看作是 Frenet 框架从三维退化至二维的简化应用。

5.2.2 Frenet 框架到二维的降维

Frenet-Serret 公式是描述曲线在空间中的位置和方向的一组微分方程。这些公式基于曲率（curvature）和扭率（torsion）来描述曲线。[4] 在三维空间中，对于任意一点上的曲线，Frenet-Serret 公式使用法向量（normal）、切向量（tangent）、和副法向量（binormal）来定义一个随曲线移动的局部坐标系。坐标移动的平移量为

$$\boldsymbol{p}_{i+1} = \begin{bmatrix} -\left(1-\cos\left(\theta_i\right)\right)/k_i & 0 & \sin\left(\theta_i\right)/k_i \end{bmatrix}^{\mathrm{T}} \tag{5}$$

再将新坐标系沿不同轴旋转 θ_i 和 $\Delta\varphi_{i+1}$ 得到两个旋转矩阵

$$\boldsymbol{T}_{B_i}^{\theta_i} = \begin{bmatrix} \cos\left(\theta_i\right) & 0 & -\sin\left(\theta_i\right) & 0 \\ 0 & 1 & 0 & 0 \\ \sin\left(\theta_i\right) & 0 & \cos\left(\theta_i\right) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$\boldsymbol{T}_{T_i}^{\Delta\varphi_{\omega 1}} = \begin{bmatrix} \cos\left(\Delta\varphi_{i+1}\right) & -\sin\left(\Delta\varphi_{i+1}\right) & 0 & 0 \\ \sin\left(\Delta\varphi_{i+1}\right) & \cos\left(\Delta\varphi_{i+1}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

据此不断变换矩阵可以不断建立新的运动坐标系，将每个微段末端修正后连接起来得到拟合曲线 [1]

但本题要求我们探讨的是基于曲率的平面曲线重构，既不存在扭率，也不存在副法向量，仅需我们通过连续调整切线角度来更新位置。

1. 位置更新

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + h \cdot \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} \tag{8}$$

这里，h 是步长，θ_i 是当前切线与 x 轴的夹角。

2. 角度更新

$$\theta_{i+1} = \theta_i + h \cdot k_i \tag{9}$$

这里，k_i 是在点 i 的曲率。

3. 曲率更新如果曲率沿曲线变化，可以用插值或者其他方式在每一步计算 k_i。

据此我们可以通过简化的 Frenet-Serret 公式完成平面曲线的重构。

5.2.3 小步长欧拉法法实现曲线重构

步长设置：设置了一个小的步长 ds，这是用于数值积分的步长，控制积分过程的精度。

重构函数：reconstruct_curve 函数是核心部分，它使用传入的曲率和步长从给定的初始条件开始，重构曲线。

初始条件：起始点 (x_0, y_0) 和起始角度 θ_0

积分方法：函数通过遍历每个传感器位置间的步骤，使用简单的欧拉积分来逐步计算曲线的新点。对于每一小步，更新角度 θ_0，然后通过将当前点的坐标加上由当前角度和步长决定的增量计算新的 x 和 y 坐标。

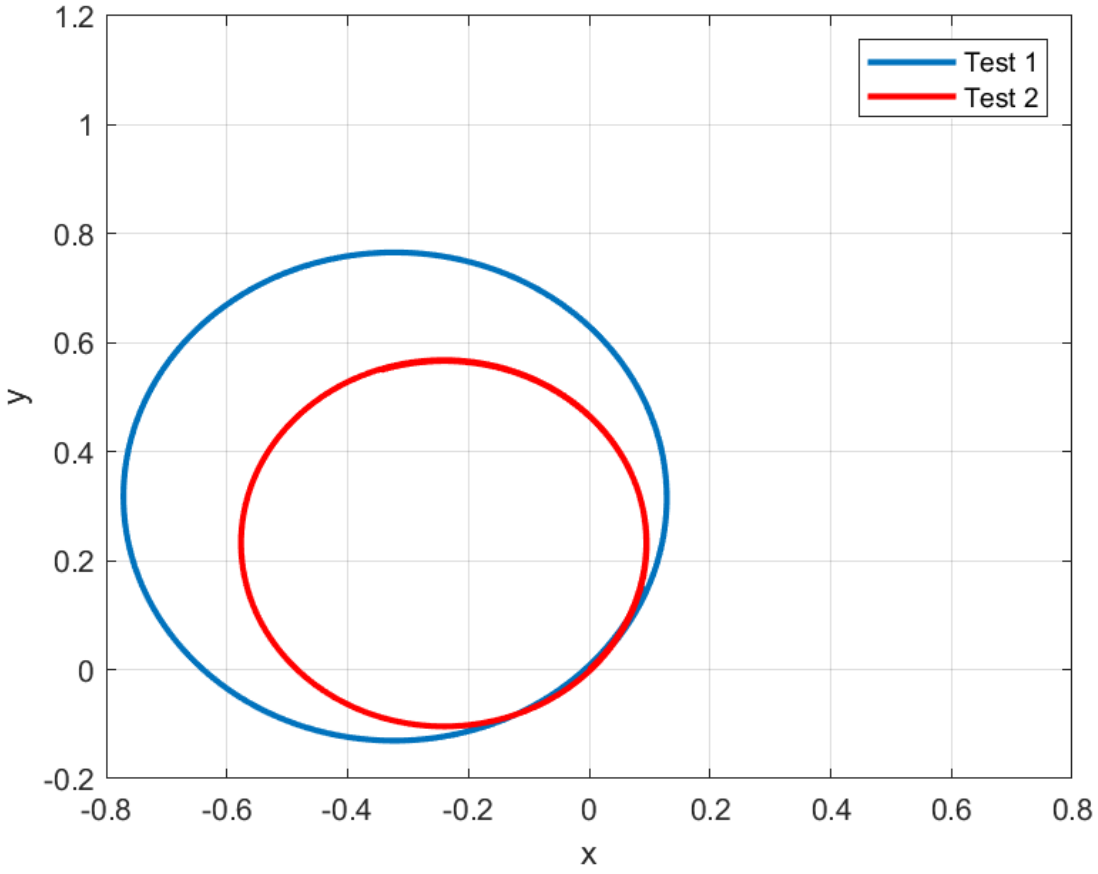

图 2　Reconstructed Plane Curve

5.2.4 重构曲线特点分析

通过重构的两平面曲线图像，我们能够很自然的猜测两次测试分别得到的是两个半径不同的圆形。但这种猜测明显存在一些自相矛盾：一条光纤不可能在两次测试中呈现直径不同的两个恰好完整的圆形。

因此我们将构建的平面曲线进行局部放大，寻找其中不符合圆形的曲线特征。

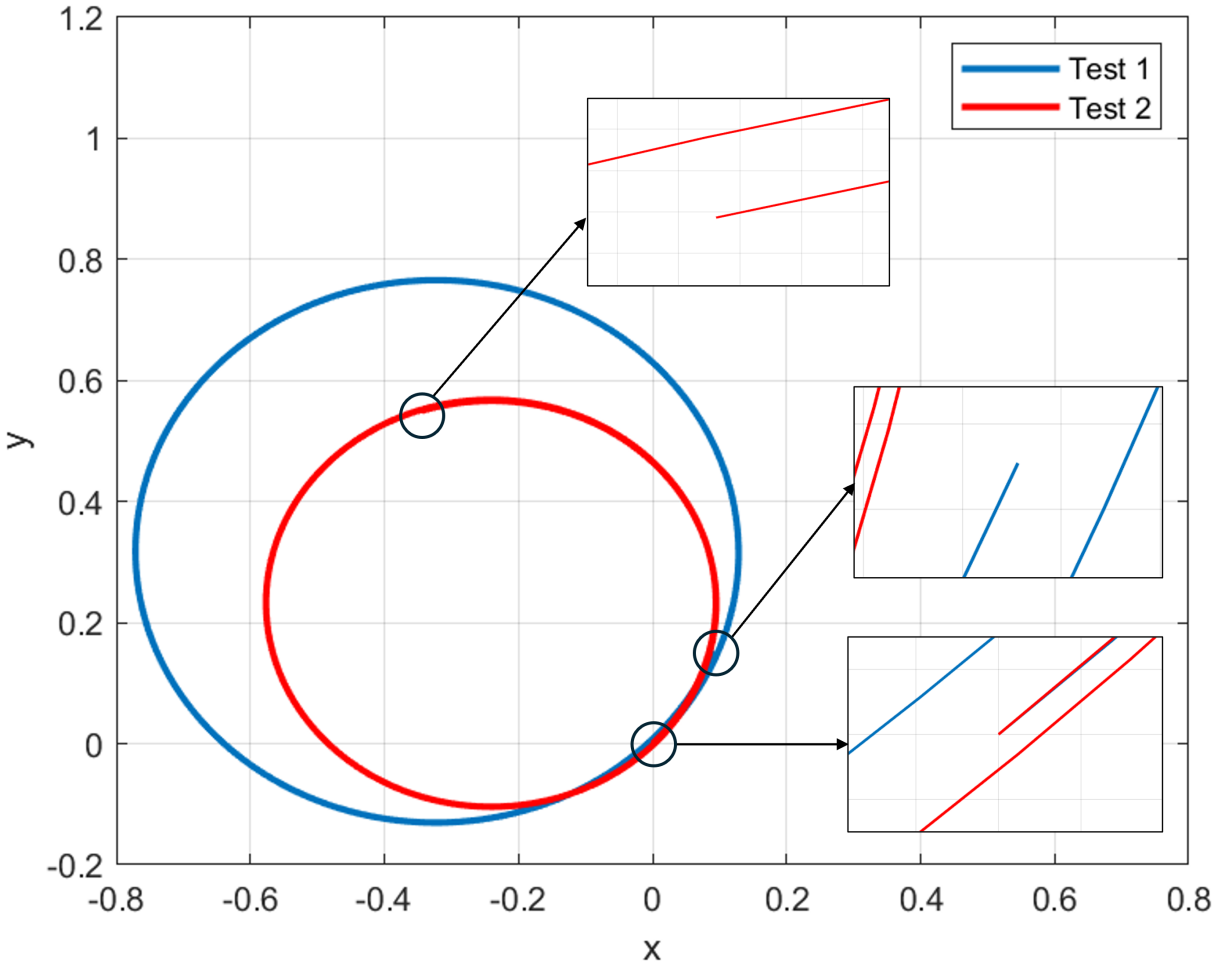

图 3　Reconstructed Plane Curve with Furture

果然我们发现了许多并不属于圆形的曲线特征，两条曲线并不是完美的圆形。

对于圆而言，圆上每个点相对固定坐标轴的曲率都是恒定不变的。为进一步验证该结论，我们希望从曲率入手，通过探讨该曲线上各点曲率是否保持一致来判断重构曲线是否为完整的圆形，并研究重构曲线曲率的变化趋势。

借助 Matlab，我们以两曲线各自在对应 x 坐标处的曲率为纵坐标，绘制出曲率图像，展示在坐标系中。

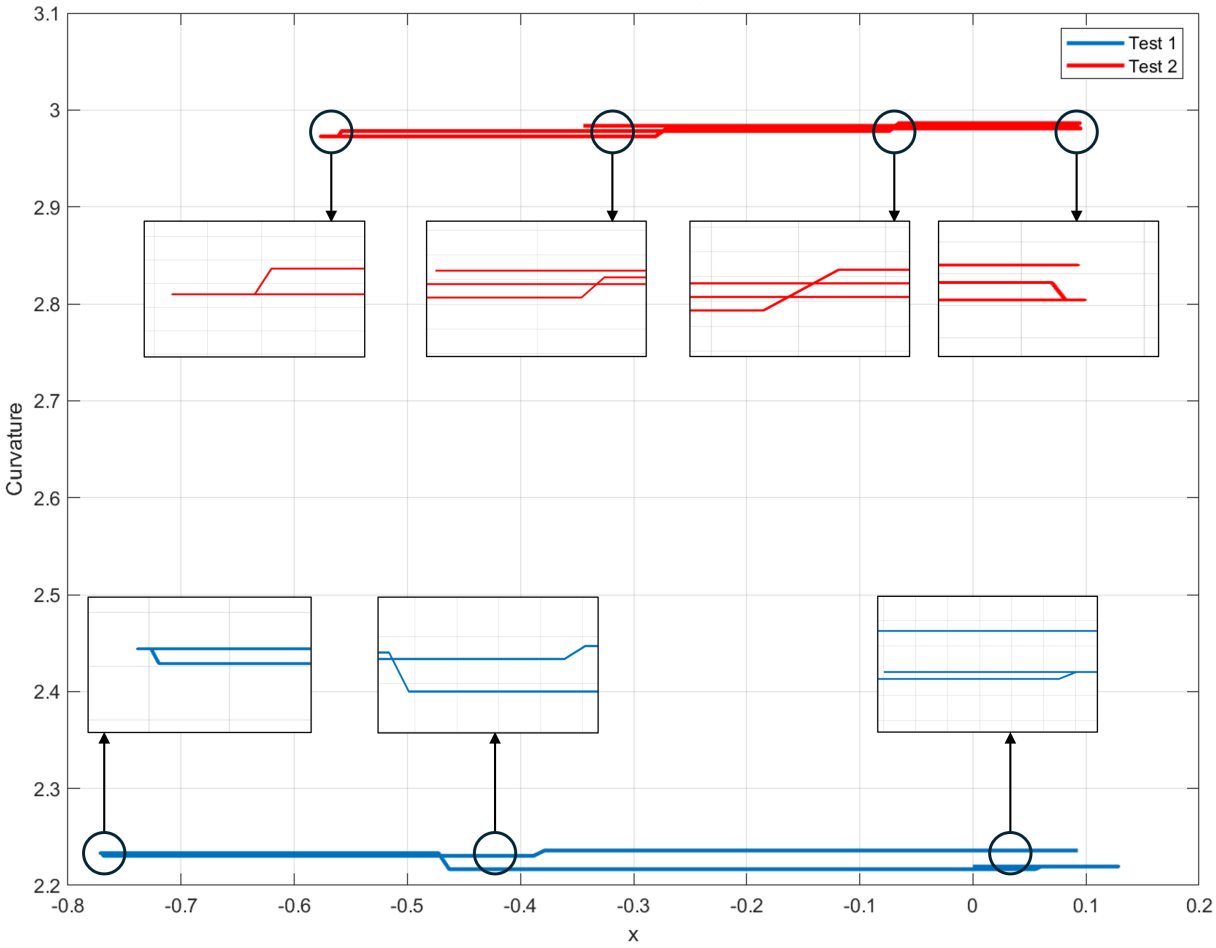

图 4　Reconstructed Plane Curve Curvature with Feature

从图像中我们可以发现，测试一中从原点开始沿逆时针方向旋转，曲线曲率先降低再增大，在左顶点附近再次降低，最后又增大至比起点还大的曲率值；测试二也基本满足此变化趋势，但最终并未回升至起点处的曲率值。

综上，两条重构曲线的特点是：（1）在整体上类似于圆（2）在走势上类似螺旋线（3）在细节上，每点的曲率呈现出并不规则的姿态（4）曲线的总长度都仍为 3，对应的光纤的总长度在受外力作用下并不会发生变化。

5.2.5 五次多项式曲线拟合（5-OPCF）

将曲率数据对应于各个传感器位用多项式进行曲率曲线的拟合。使用多项式拟合的目的是为了能在更加连续的位置上估算曲率，而不仅仅是在离散的传感器位置。

我们选择多项式的阶数为 5。使用拟合的多项式和初始角度，通过数值积分方法重构出曲线。核心思路是从某一起始点（通常是原点），根据计算得到的连续曲率变化逐步构建曲线。数学上，可以通过以下步骤进行：

初始化曲线的起始点 (x, y) 和起始角度 θ。在较细的步长 ds 上，用多项式拟合的曲

率值来更新角度 θ 和坐标 (x, y)：

1. 角度更新：

$$\theta_{new} = \theta + \kappa \cdot ds$$

2. 坐标更新：

$$x_{new} = x + \cos(\theta) \cdot ds$$

$$y_{new} = y + \sin(\theta) \cdot ds$$

这里，每一步的角度更新取决于当前位置的曲率和步长，而位置的更新则是基于当前角度和步长计算出的位移。

5.2.6 多项式拟合与欧拉法拟合对比

我们分别将两种算法得到的重构曲线以及其曲率图像置于同一坐标系中

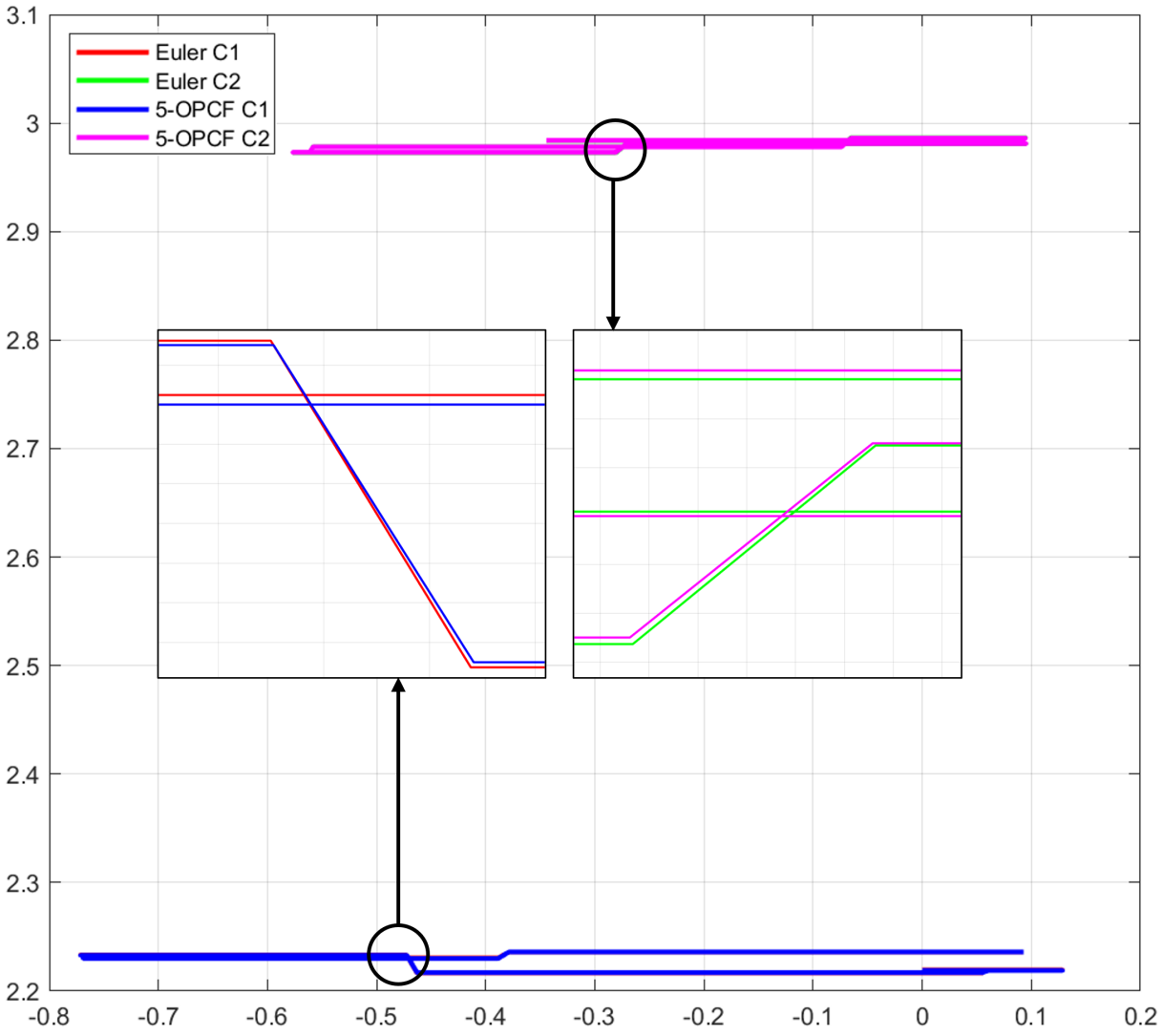

图 5　5-OPCT & Euler Curvature with Feature

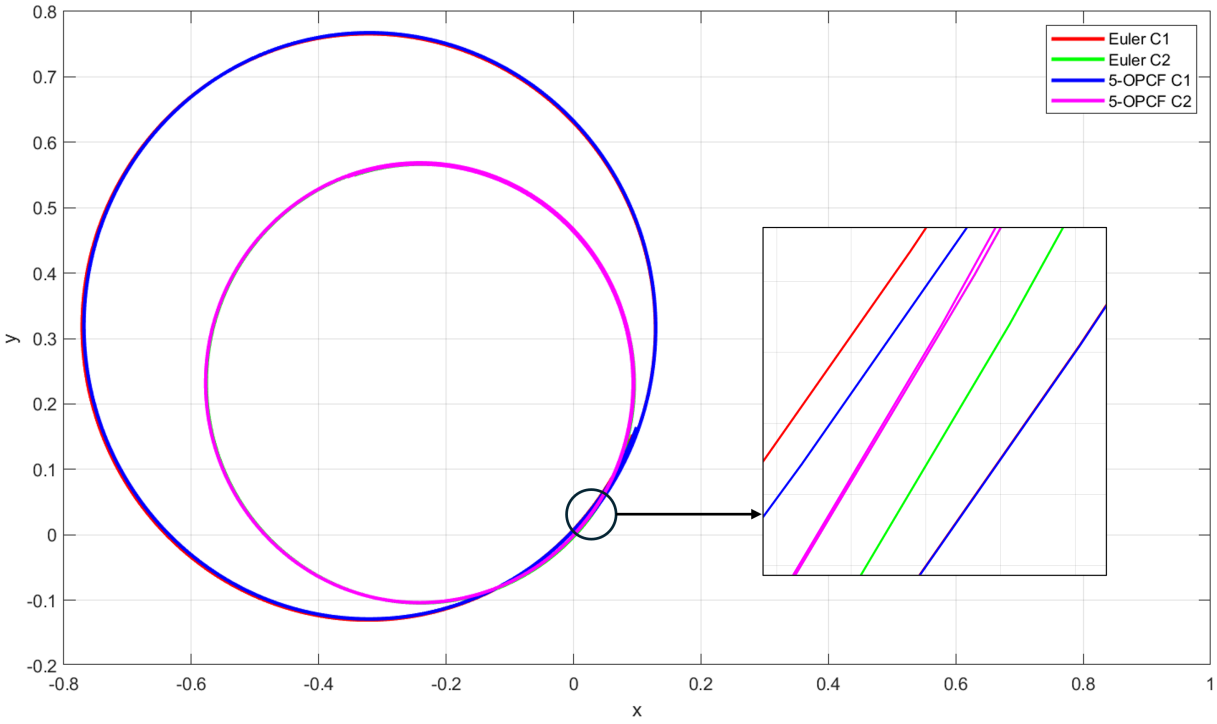

图 6　5-OPCT & Euler with Feature

可以发现，两条重构曲线的图像以及其曲率图像基本一致，局部放大后观察能够发现细节仍有差别。这是一个强有力的指示，我们找到的解决方案可能非常接近全局的最优解，可以证明我们建立的模型是十分稳定可靠的。

5.3 问题三对三次多项式曲率取样及重构曲线

5.3.1 等间距弧长取样

等间距弧长采样是一种在计算几何和计算机图形学中常用的技术，特别适用于处理和分析曲线和曲面。这种方法的主要目的是从曲线上均匀地按照弧长采样点，这样在曲线上的每个点之间的距离都相等。这对于许多应用来说非常重要，它能够保证处理的均匀性和公平性，避免在曲线的某些部分出现点过密或过疏的情况。题目要求我们采用适当的间距对目标曲线进行等弧长采样。

我们的主要目标是对曲线 $y = x^3 + x$ 进行等间距弧长采样，并尝试使用采样点重构曲线。

1. 定义曲线及其导数

2. 计算曲率: 曲率的计算依赖于一阶和二阶导数，使用公式：

$$\kappa(x) = \frac{|f''(x)|}{(1 + (f'(x))^2)^{3/2}} \tag{10}$$

3. 求解采样点的 x 值和对应的 y 值：

4. 计算曲线的弧长：使用累积积分的方法来估算从曲线起点到每个样本点的弧长。

5. 等间距弧长采样：根据计算出的总弧长将曲线分成若干等间隔的段。然后使用插值找到每个段的 x 值，这些 x 值对应于曲线上的等间距弧长点。

通过 cumtrapz 方法，代码计算了从曲线的起始点到每一个 x 点的累积弧长。这为后续步骤提供了能够根据弧长找到对应 x 值的基础数据。通过将总弧长均匀分割，并使用 'interp1' 在已知的弧长与 x 值之间进行插值，代码有效地找到了应该在曲线上取样的 x 值，确保了这些 x 值对应的是等间距的弧长。

得到的等间距弧长采样点如下图

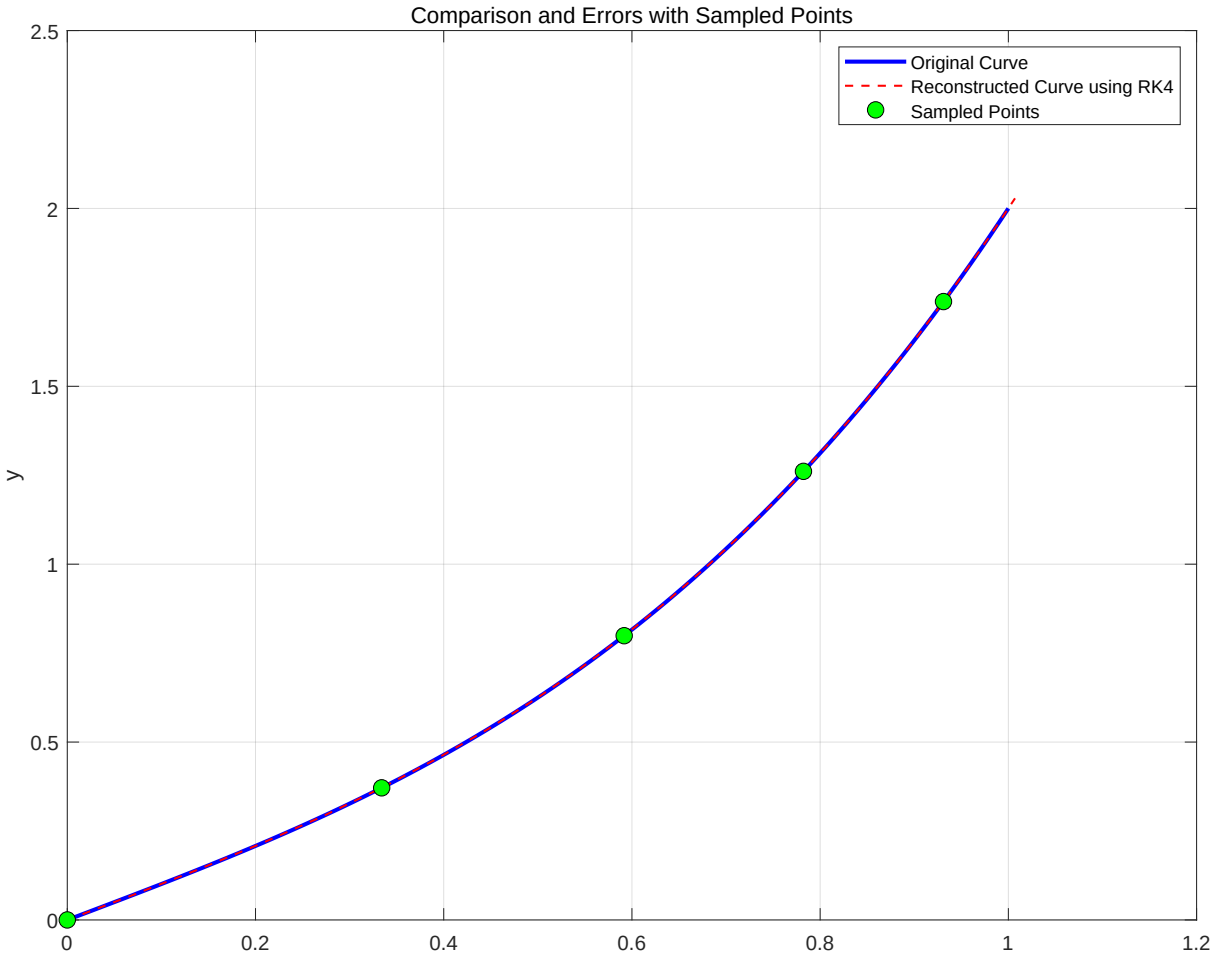

图 7　Enter Caption

5.3.2 平面曲线重构方法

1. 前向欧拉法（FEM）

(a) 基本原理：前向欧拉方法通过直接利用微分方程的切线斜率来逼近曲线的下一点。在曲线重构的上下文中，它通过当前点的曲率和方向（即切线角度）来预测曲线的下一位置。

(b) 实现方式：重构过程基于给定的起始角度 θ，通过 θ 的连续更新 $\theta = \theta + \kappa \cdot ds$ 来模拟曲线。这里，κ 是给定点的曲率，而 ds 是小的步长，用于模拟前进。新的 x

和 y 坐标通过 $x = x + \cos(\theta) \cdot ds$ 和 $y = y + \sin(\theta) \cdot ds$ 计算得出。

(c) 方法的适用性: 这种方法对于曲线重构特别有效，当你有关于曲线形状的局部信息（如曲率）并希望逐点重建整条曲线时。它尤其适合在动态系统模拟、计算机图形和其他需要准确曲线追踪的场合。

(d) 方法的局限：尽管前向欧拉方法实现简单，它可能在步长选择不当时引入较大的数值误差，尤其是在曲率变化较大的区域。这种方法在实际应用中可以调整以提高其精度和适应性，通过调整步长或采用更复杂的数值方法来适应不同的应用需求。

2. 四阶龙格-库塔法（RK4）

(a) RK4 算法详解: 初始条件设为 $x = 0, y = 0, \theta = \tan^{-1}(f'(0))$。

RK4 步骤被用于迭代更新 x, y, θ：

$k_{1x}, k_{1y}, k_{1\theta}$ 基于当前位置和曲率。

$k_{2x}, k_{2y}, k_{2\theta}$ 在半步后预测的状态。

$k_{3x}, k_{3y}, k_{3\theta}$ 再次在半步后预测的状态。

$k_{4x}, k_{4y}, k_{4\theta}$ 在整步后预测的状态。

组合这些 k 值来得到下一个 x, y, θ 的最优预测。

(b) 重构曲线: 在整个定义域 $[0, 1]$ 上应用 RK4，通过逐步增加 x 并相应地更新 y 和 θ 来构建曲线。

3. 有限差分法（FDM）

(a) 有限差分法原理有限差分法是一种数值技术，用于近似微分方程中的导数。它通过替代无限小的变化（导数的定义）来估计有限小的变化。

(b) 一阶导数近似对于函数 $f(x)$，其一阶导数 $f'(x)$ 的中心差分公式是：

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

这个公式是根据泰勒展开推导出来的，其中 h 是一个小的步长。

(c) 二阶导数近似类似地，函数的二阶导数 $f''(x)$ 可以通过有限差分法近似为：

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

(d) 泰勒级数扩展原理泰勒级数可以将一个光滑的函数在某一点的邻域内展开为它在该点的导数的级数。对于足够小的 h，函数 $f(x)$ 在 $x + h$ 处的值可以近似为：

$$f(x+h) \approx f(x) + f'(x)h + \frac{f''(x)h^2}{2}$$

(e) 模型建立详细过程

原始函数：你的目标函数是 $y = f(x) = x^3 + x$。

导数的有限差分近似：使用有限差分法来估计函数在每个 x 点的一阶和二阶导数。

泰勒级数近似下一个 y 值：在已知 x 和 y 的情况下，使用泰勒级数扩展来估计下一个 y 的值。

这个方法的准确性依赖于步长 h 的大小。步长越小，近似的准确性越高，但是会增加计算量。而且，有限差分法在处理高曲率变化的函数时可能不够准确，因为它忽略了更高阶的项。

5.3.3 重构曲线与目标曲线间的误差比较

曲线重构系统核心性能指标为曲线重构长度、精度等，因此常用绝对位置误差、均方根误差以及末端误差等来评价曲线的重构精度。

曲线的绝对位置误差表征的是曲线上某点重构坐标与理论坐标间的偏差，是反映曲线重构精度高低的基本特征，其具体表达式为:

$$\begin{aligned}\delta\mathbf{r}\left(s_n\right) &= \mathbf{r}'\left(s_n\right) - \mathbf{r}\left(s_n\right) \\ &= \begin{bmatrix} x'_n & y'_n & z'_n \end{bmatrix}^T - \begin{bmatrix} x_n & y_n & z_n \end{bmatrix}^T \\ &= \begin{bmatrix} \delta x_n & \delta y_n & \delta z_n \end{bmatrix}^T\end{aligned} \tag{11}$$

其中 $\mathbf{r}'\left(s_n\right)$ 代表重构曲线上各点对应的空间向量，其坐标为 $\begin{bmatrix} x'_n & y'_n & z'_n \end{bmatrix}^T$,$\mathbf{r}\left(s_n\right)$ 代表实际曲线上各点对应的空间向量,其坐标为 $\begin{bmatrix} x_n & y_n & z_n \end{bmatrix}^T$,两者作差得到 $\begin{bmatrix} \delta x_n & \delta y_n & \delta z_n \end{bmatrix}^T$ 则代表拟合曲线各点和理论坐标的位置差异。

但绝对位置误差仅适合分析曲线上某点的误差，并不能直观表现曲线的整体拟合精度，为对获得的重构曲线进行更整体的误差分析，我们引入均方根误差:

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{N}\left|\delta\mathbf{r}\left(s_n\right)\right|^2}{N}} \tag{12}$$

其中 N 为测量的总点数。

而在医疗健康等应用领域中，曲线末端的定位要求极高 [3]，故我们在对光纤曲线重构时也引入末端误差来表征曲线在其末端处的绝对误差:

$$\left|\delta\mathbf{r}_{\text{end}}\right| = \sqrt{\delta x_{\text{end}}{}^2 + \delta y_{\text{end}}{}^2 + \delta z_{\text{end}}{}^2} \tag{13}$$

其中 δx_{end} 、δy_{end} 、δz_{end} 分别表示重构曲线的末端绝对位置误差在 X、Y、Z 方向上的分量。基于二维平面，我们引入末端误差

$$\left|\delta\mathbf{r}_{\text{end}}\right| = \sqrt{\delta y_{\text{end}}{}^2} \tag{14}$$

5.3.4 曲线重构模型求解

初始设置：从 $x = 0, y = 0$ 开始，初始角度 θ 为 $\tan^{-1}(f'(0))$，基于曲线在 $x = 0$ 时的切线。

迭代计算：对于每个等间距的 x 点，使用起始曲率 κ 和较小的步长 ds 进行迭代，利用 θ 的更新和简单的三角关系 $dx = \cos(\theta) \times ds$ 和 $dy = \sin(\theta) \times ds$ 来计算新的 x 和 y 位置。

作图比较：使用 MATLAB 的绘图功能，将前向欧拉法、四阶龙格-库塔法、有限差分法三种方法得到的重构曲线和原始曲线放在一起进行视觉比较。

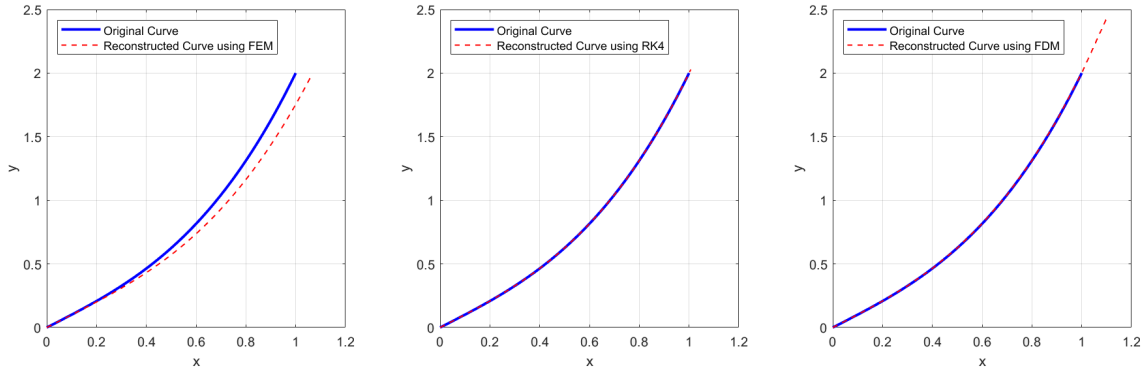

图 8　3 Reconstructed Curves

5.3.5 曲线误差分析

前向欧拉法、四阶龙格-库塔法（RK4）和有限差分法都是数值解法。我们将三种方法得到的理论重构曲线与实际曲线对比，在使用图形进行直观的重构结果对比的同时，也可以将这三种方法产生的各类误差进行量化，具体包括均方根误差，平均绝对误差，最大误差，末端误差。

均方误差 (MSE)：计算插值重构曲线和原始曲线在相同 x 值下的 y 值差异的平方的均值。

平均绝对误差 (MAE) 和最大误差：提供了重构精度的其他度量。

末端误差（EPE）：针对研究曲线末端位置的绝对误差。

我们利用 Matlab 计算相同步长下重构曲线与理论曲线之间的误差以及曲线重构所需要的运行时间，记录在表格中并提供详细的数据分析。

误差计算：

表 3　Error And Time

Step（h）	Method	RMSE	MAE	Max Error	EPE	Runtime
0.01	FEM	0.009883	0.007328	0.022032	0.002696	0.001324
	RK4	0.000005	0.000005	0.000009	0.008992	0.000473
	FDM	0.000029	0.000025	0.000044	0.000000	0.000223
0.02	FEM	0.020008	0.014787	0.044882	0.005564	0.001233
	RK4	0.000021	0.000018	0.000037	0.008992	0.000354
	FDM	0.000125	0.000099	0.000297	0.000000	0.000152
0.05	FEM	0.051460	0.037697	0.117202	0.014877	0.000695
	RK4	0.000130	0.000113	0.000235	0.028403	0.000217
	FDM	0.000786	0.000619	0.001764	0.000000	0.000104
0.1	FEM	0.105885	0.076547	0.246003	0.032792	0.000414
	RK4	0.000523	0.000455	0.000933	0.028405	0.000090
	FDM	0.003143	0.002475	0.007123	0.431000	0.000078
0.2	FEM	0.213243	0.150602	0.509364	0.078648	0.000418
	RK4	0.002150	0.001886	0.003799	0.125606	0.000118
	FDM	0.012526	0.009899	0.026987	0.000000	0.000072

分析表格内容，我们可以发现前向欧拉法与另两种重构方法相比有多个类型的较大误差. 对上述表格内容进行简单处理，我们将三种曲线重构方法对应的四种误差类型占比绘制成如下的扇形图，分别讨论三种算法各自不同类型误差的占比。

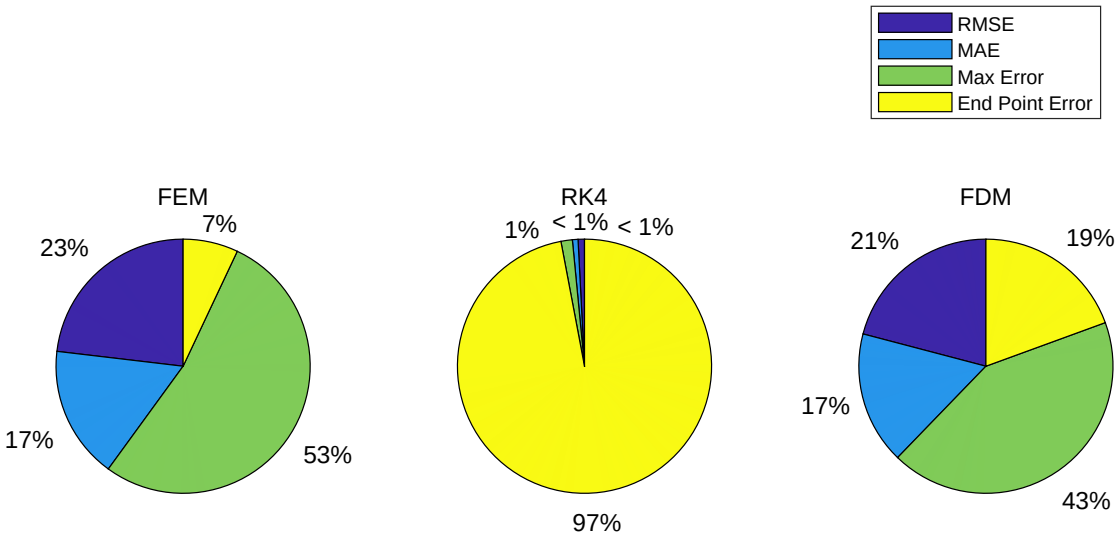

图 9　Error Statistic

前向欧拉法中，末端误差是其占比最小的一项，表明该模型在整个预测区间内的表现都较为一致。

四阶龙格-库塔法中，末端误差虽然占比极大，但其末端误差的实际数值其实很小，表明其其他各项误差值已经可以忽略不计。

有限差分法中，各类误差占比相差不大，表明该模型并没有非常突出的缺点，是一较为健壮全面的算法。

基于当今社会对于高精度产品末端误差的高要求，我们再重点讨论这三种重构算法中末端误差的大小。

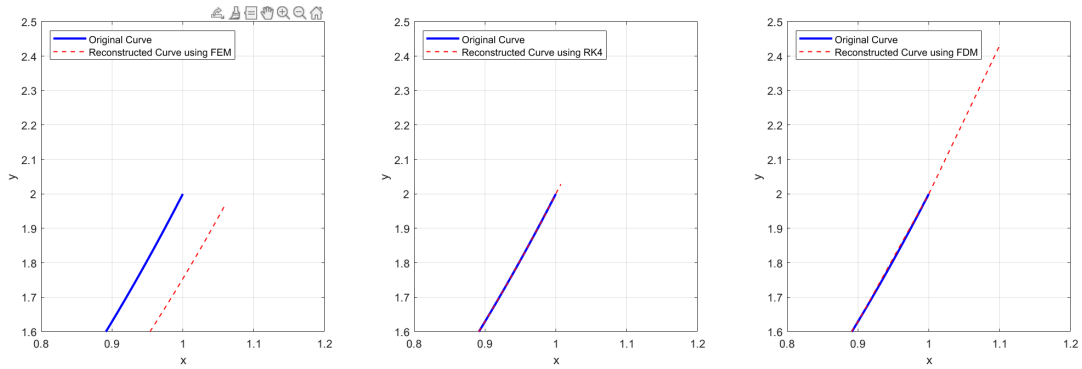

图 10　Reconstructed Curve End

值得注意的是，末端误差的大小与步长的选取并没有严格的相关关系，我们无法从所得数据中得到极为准确的末端误差评估。但从图像中我们可以大致猜测出三种拟合方

法中前向欧拉法和四阶龙格-库塔法在末端误差方面的优势.

5.3.6 三种曲线重构方法的误差来源

1. 欧拉法

   局部截断误差：由于欧拉法仅使用当前点的导数信息，它在每一步都进行线性逼近，忽略了导数在步长内的变化，导致一阶的局部截断误差。

   累积误差：由于局部误差在每一步迭代中都会产生，这些误差会在多步计算中累积，影响最终结果的准确性。

   步长大小：步长越大，每一步的误差越大，精度越低。然而，减小步长虽然可以减少误差，但会增加计算量。

2. 四阶龙格库塔法 (RK4)

   局部截断误差：RK4 使用四个不同点的斜率计算平均斜率，大大提高了逼近的准确性。尽管如此，它仍然会产生四阶的局部截断误差。

   圆滑误差：尽管 RK4 的局部截断误差较小，但在处理不连续或非常陡峭的函数时，可能无法精确模拟函数的动态变化。

   步长依赖性：与欧拉法类似，步长的选择在 RK4 中也非常关键。较小的步长可以减少误差，但会增加计算成本。

3. 有限差分法

   离散化误差：通过将连续的微分方程转换为离散点上的差分方程来近似求解，这种方法固有地引入了离散化误差。

   边界处理：在边界处的处理方式可能不够精确，尤其是当边界条件复杂或不规则时。

5.3.7 模拟退火法为代表的启发式算法在问题三中的应用

我们尝试通过启发式算法来进行曲线的重构。经过多组数据的测试，我们分别尝试了模拟退火算法、遗传算法、粒子群算法和蚁群算法，发现模拟退火算法是拟合效果及运行效率相对较好的一种。下面以其为例，来进行数据误差的分析：

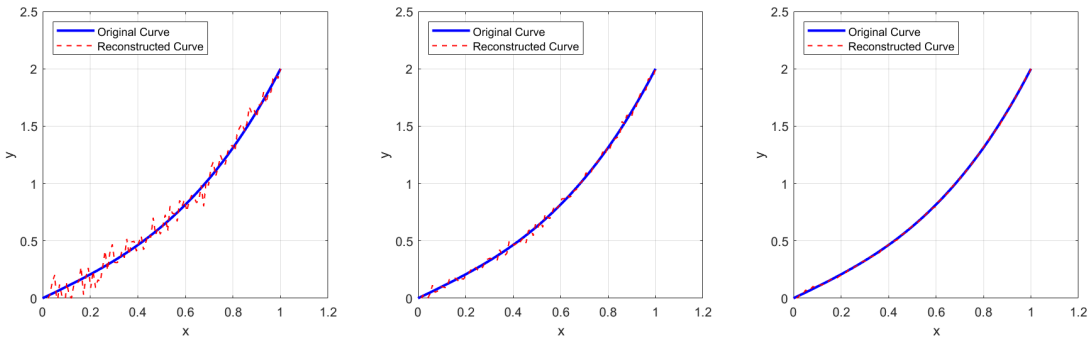

图 11　Original Curve and Reconstructed Curve using Simulated Annealing

模拟退火算法误差的一些主要来源：（1）初始温度设定（2）冷却计划（3）逃离最小值的随机性（4）接受准则.

我们通过和上述三种重构算法相同的方法，在 Matlab 上获得模拟退火法的各项误差数据及运行时间，整理为下表：

表 4　Curvature of target point

	RMSE	MAE	Max Error	End Point Error	Runtime
1	0.07077	0.05620	0.17534	0	0.10264
2	0.02577	0.02021	0.06807	0	1.19580
3	0.00759	0.00601	0.01793	0	19.79669

从表格中我们可以得知模拟退火算法在初始条件不同的情况下，会通过牺牲算法自身的运行时间来实现更小的误差，从而达到更加精确的曲线重构的目的。

六、 模型的进一步讨论与评价

1. 问题一：光纤上各点曲率的估算

   在本部分中，我们首先基于给定条件论证了曲率与反射波长之间的线性关系，并通过传感器读数估计了光纤曲线上特定传感点的曲率。我们选择了三次样条插值法作为重构光纤曲线的方法，并且通过比较插值结果和原始数据的曲率，验证了该方法的优越性。进一步的分析可以集中在样条插值法的适用性和性能上，例如在不同噪声水平下的稳健性以及对于曲线局部特征的准确性。

2. 问题二：根据离散点处的曲率进行曲线的重构

   在这一部分，我们介绍了 Frenet 框架在二维条件下的基础算法，并利用小步长欧拉法构造了重构曲线。通过对重构曲线的局部放大分析和与多项式拟合法的对比，我们验证了算法的有效性和可靠性。未来的研究可以进一步探索在实际场景中应用该算法的可能性，例如在光纤传感器监测系统中的实时性和鲁棒性。

3. 问题三：对已知三次多项式的曲率采样与曲线重构

   在此部分，我们采用了前向欧拉法、四阶龙格-库塔法以及有限差分法等算法完成了平面曲线的重构，并进行了误差分析和算法效果的综合评估。我们还探讨了模拟退火法在曲线重构中的应用，验证了其与目标曲线高度重合的优越性。未来的研究可以进一步考虑更复杂曲线情况下各种算法的适用性，并且优化算法以提高重构曲线的准确性和稳定性。

4. 总结

通过对光纤传感技术在平面曲线重构中的应用进行了全面的模型分析，我们得出了一系列结论并提出了未来研究的方向。这些研究成果为光纤传感技术在工程实践中的应用提供了理论基础和方法指导，具有一定的理论和实践意义。

参考文献

[1] 刘卿卿, 邱东, 徐帅, and 王思语. 基于 frenet 坐标系的路径规划仿真. 计算机仿真, 41(01):166–170, 2024.

[2] 孙赓. 基于光纤光栅的形状传感自适应算法研究. Master's thesis, 哈尔滨工程大学, 2024.

[3] 田金容. 基于多芯光纤和光频域反射的三维曲线重构方法研究. Master's thesis, 华中科技大学, 2022.

[4] 程文胜. 基于超弱光纤光栅的曲线重构方法研究. Master's thesis, 三峡大学, 2022.

附录

```
%q1.根据传感器读数返回曲率值
   %插值法求解特定点的曲率值

c = 4200;% 定义常数c

% 测试1的数据
lambda_0_test1 = [1529, 1529, 1529, 1529, 1529, 1529];
lambda_test1 = [1529.808, 1529.807, 1529.813, 1529.812, 1529.814, 1529.809];

% 测试2的数据
lambda_0_test2 = [1540, 1540, 1540, 1540, 1540, 1540];
lambda_test2 = [1541.095, 1541.092, 1541.090, 1541.093, 1541.094, 1541.091];

% 传感器位置
positions = [0, 0.6, 1.2, 1.8, 2.4, 3.0]; % 传感器位置

% 计算测试1的曲率
kappa_test1 = c * (lambda_test1 - lambda_0_test1) ./ lambda_0_test1;

% 创建插值函数（测试1）
curvature_interpolation_test1 = fit(positions', kappa_test1', 'cubicinterp');

% 计算测试2的曲率
kappa_test2 = c * (lambda_test2 - lambda_0_test2) ./ lambda_0_test2;

% 创建插值函数（测试2）
curvature_interpolation_test2 = fit(positions', kappa_test2', 'cubicinterp');

% 指定横坐标位置
x_positions = [0.3, 0.4, 0.5, 0.6, 0.7];

% 在这些位置估算曲率（测试1和测试2）
estimated_curvatures_test1 = curvature_interpolation_test1(x_positions);
estimated_curvatures_test2 = curvature_interpolation_test2(x_positions);

% 输出估算的曲率（测试1）
disp('测试1的估算曲率值为：');
for idx = 1:length(x_positions)
    fprintf('在 x = %.1f 米处的曲率：%.4f\n', x_positions(idx), estimated_curvatures_test1(idx));
end

% 输出估算的曲率（测试2）
disp('测试2的估算曲率值为：');
for idx = 1:length(x_positions)
```

```
    fprintf('在 x = %.1f 米处的曲率: %.4f\n', x_positions(idx), estimated_curvatures_test2(idx));
end
```

```
%q2.1 根据问题一求得结果重构完整曲线

c = 4200;% 定义常数c和传感器数据

% 测试1的数据
lambda_0_test1 = [1529, 1529, 1529, 1529, 1529, 1529];
lambda_test1 = [1529.808, 1529.807, 1529.813, 1529.812, 1529.814, 1529.809];
positions = [0, 0.6, 1.2, 1.8, 2.4, 3.0]; % 传感器位置

% 计算每个传感器点的曲率
kappa_test1 = c * (lambda_test1 - lambda_0_test1) ./ lambda_0_test1;

% 测试2的数据
lambda_0_test2 = [1540, 1540, 1540, 1540, 1540, 1540];
lambda_test2 = [1541.095, 1541.092, 1541.090, 1541.093, 1541.094, 1541.091];

% 计算每个传感器点的曲率
kappa_test2 = c * (lambda_test2 - lambda_0_test2) ./ lambda_0_test2;

% 设置步长
ds = 0.01;

% 重构测试1和测试2的曲线
[x_points_test1, y_points_test1] = reconstruct_curve(positions, kappa_test1, ds, pi/4);
[x_points_test2, y_points_test2] = reconstruct_curve(positions, kappa_test2, ds, pi/4);

% 绘制测试1和测试2的曲线
figure;
plot(x_points_test1, y_points_test1, 'LineWidth', 2);
hold on;
plot(x_points_test2, y_points_test2, 'r-', 'LineWidth', 2);
title('Reconstructed Curves for Test 1 and Test 2');
xlabel('x');
ylabel('y');
legend('Test 1', 'Test 2');
grid on;
hold off;

% 重构函数定义，增加初始角度参数
function [x_points, y_points] = reconstruct_curve(positions, kappa, ds, initial_angle)
    % 预分配数组
    total_steps = sum(round(diff(positions) / ds));
    x_points = zeros(1, total_steps + 1);
```

```
    y_points = zeros(1, total_steps + 1);

    % 初始条件，使用给定的初始角度
    theta = initial_angle; % 初始角度为 pi/4，对应斜率为 1
    x = 0;
    y = 0;
    index = 1;

    % 存储第一个点
    x_points(index) = x;
    y_points(index) = y;

    for i = 2:length(positions)
        steps = round((positions(i) - positions(i-1)) / ds);
        for j = 1:steps
            theta = theta + kappa(i-1) * ds;
            x = x + cos(theta) * ds;
            y = y + sin(theta) * ds;
            index = index + 1;
            x_points(index) = x;
            y_points(index) = y;
        end
    end
end
```

```
    %q2.2 利用多项式法进行曲线重构
c = 4200;% 定义常数c和传感器数据
% 传感器位置
positions = [0, 0.6, 1.2, 1.8, 2.4, 3.0];

% 测试1的数据和曲率计算
lambda_0_test1 = [1529, 1529, 1529, 1529, 1529, 1529];
lambda_test1 = [1529.808, 1529.807, 1529.813, 1529.812, 1529.814, 1529.809];
kappa_test1 = c * (lambda_test1 - lambda_0_test1) ./ lambda_0_test1;

% 测试2的数据和曲率计算
lambda_0_test2 = [1540, 1540, 1540, 1540, 1540, 1540];
lambda_test2 = [1541.095, 1541.092, 1541.090, 1541.093, 1541.094, 1541.091];
kappa_test2 = c * (lambda_test2 - lambda_0_test2) ./ lambda_0_test2;

% 拟合多项式曲率模型
poly_order =5; % 多项式的阶数可以根据需要调整
p_test1 = polyfit(positions, kappa_test1, poly_order);
p_test2 = polyfit(positions, kappa_test2, poly_order);
```

```
% 设置步长
ds = 0.01;

% 重构曲线
[x_points_test1, y_points_test1] = reconstruct_curve_polyfit(positions, p_test1, ds, pi/4);
[x_points_test2, y_points_test2] = reconstruct_curve_polyfit(positions, p_test2, ds, pi/4);

% 绘制曲线
figure;
plot(x_points_test1, y_points_test1, 'LineWidth', 2);
hold on;
plot(x_points_test2, y_points_test2, 'r-', 'LineWidth', 2);
title('Reconstructed Curves for Test 1 and Test 2 using Polynomial Fit');
xlabel('x');
ylabel('y');
legend('Test 1', 'Test 2');
grid on;
hold off;

% 重构函数定义，使用多项式拟合的曲率
function [x_points, y_points] = reconstruct_curve_polyfit(positions, p, ds, initial_angle)
    % 初始化
    x = 0;
    y = 0;
    theta = initial_angle;
    x_points = x;
    y_points = y;

    % 使用拟合多项式在更细的步长上计算曲率
    fine_positions = positions(1):ds:positions(end);
    kappa_fine = polyval(p, fine_positions);

    % 重构曲线
    for i = 1:length(kappa_fine)
        theta = theta + kappa_fine(i) * ds;
        x = x + cos(theta) * ds;
        y = y + sin(theta) * ds;
        x_points = [x_points, x];
        y_points = [y_points, y];
    end
end
```

```
        %q3.1等弧长间距对已知平面曲线采样
            function main
    % 定义原始函数及其导数
    f = @(x) x.^3 + x;
```

```
df = @(x) 3*x.^2 + 1;
ddf = @(x) 6*x;

% 计算曲率的函数
curvature = @(x) abs(ddf(x)) ./ (1 + df(x).^2).^(3/2);

% RK4 参数
h = 0.1; % 修改步长以更精确模拟
x = 0; % 初始条件
y = 0;
theta = atan(df(0)); % 初始角度

% 存储路径的数组
xs = [x];
ys = [y];
arc_length = [0]; % 初始化弧长数组

% 使用RK4求解
while x < 1
    k1x = h * cos(theta);
    k1y = h * sin(theta);
    k1t = h * curvature(x);

    k2x = h * cos(theta + 0.5 * k1t);
    k2y = h * sin(theta + 0.5 * k1t);
    k2t = h * curvature(x + 0.5 * k1x);

    k3x = h * cos(theta + 0.5 * k2t);
    k3y = h * sin(theta + 0.5 * k2t);
    k3t = h * curvature(x + 0.5 * k2x);

    k4x = h * cos(theta + k3t);
    k4y = h * sin(theta + k3t);
    k4t = h * curvature(x + k3x);

    x = x + (k1x + 2*k2x + 2*k3x + k4x) / 6;
    y = y + (k1y + 2*k2y + 2*k3y + k4y) / 6;
    theta = theta + (k1t + 2*k2t + 2*k3t + k4t) / 6;

    xs = [xs, x];
    ys = [ys, y];
    arc_length = [arc_length, arc_length(end) + sqrt(k4x^2 + k4y^2)]; % 计算弧长
end

% 生成原始曲线数据
x_original = linspace(0, 1, 100);
y_original = f(x_original);
```

```
    % 等间距采样点
    sample_interval = 0.5; % 采样间隔
    sample_points_x = [];
    sample_points_y = [];
    for s = 0:sample_interval:arc_length(end)
        [~, idx] = min(abs(arc_length - s));
        sample_points_x = [sample_points_x, xs(idx)];
        sample_points_y = [sample_points_y, ys(idx)];
    end

    % 绘制原始曲线和重构曲线，以及采样点
    figure;
    plot(x_original, y_original, 'b-', 'LineWidth', 2);
    hold on;
    plot(xs, ys, 'r--', 'LineWidth', 1);
    plot(sample_points_x, sample_points_y, 'ko', 'MarkerFaceColor', 'g'); % 绘制采样点
    title(sprintf('Comparison and Errors with Sampled Points'));
    xlabel('x');
    ylabel('y');
    legend('Original Curve', 'Reconstructed Curve using RK4', 'Sampled Points');
    grid on;
    hold off;
end
```

```
%q3.2。1前向欧拉法

f = @(x) x.^3 + x;
df = @(x) 3*x.^2 + 1;
ddf = @(x) 6*x; % 定义原始函数及其一阶和二阶导数

% 曲率计算函数
curvature = @(x) abs(ddf(x)) ./ (1 + df(x).^2).^(3/2);

% x 的范围和细分
x_samples = linspace(0, 1, 100);
y_samples = f(x_samples);

% 计算曲率
kappa_samples = curvature(x_samples);

% 插值以获得等间距的弧长采样
arc_length = cumtrapz(x_samples, sqrt(1 + df(x_samples).^2));
total_arc_length = arc_length(end);
h=0.05;%修改步长以更精确模拟
n_intervals = 1/h; % 间隔数
```

```
segment_length = total_arc_length / n_intervals;
arc_x_samples = interp1(arc_length, x_samples, linspace(0, total_arc_length, n_intervals+1));

% 重构曲线
theta = atan(df(0)); % 初始角度
x_recon = 0;
y_recon = 0;
x_recon_points = [x_recon];
y_recon_points = [y_recon];

for i = 1:length(arc_x_samples)-1
    x_start = arc_x_samples(i);
    x_end = arc_x_samples(i+1);
    kappa = curvature(x_start); % 使用起点的曲率进行简化处理
    ds = segment_length / 10; % 较小的步长用于模拟
    for j = 1:10
        theta = theta + kappa * ds;
        x_recon = x_recon + cos(theta) * ds;
        y_recon = y_recon + sin(theta) * ds;
        x_recon_points = [x_recon_points, x_recon];
        y_recon_points = [y_recon_points, y_recon];
    end
end

% 作图比较
figure;
hold on;
plot(x_samples, y_samples, 'b-', 'LineWidth', 2);
plot(x_recon_points, y_recon_points, 'r--', 'LineWidth', 1);
xlabel('x');
ylabel('y');
title('Comparison of Original and Reconstructed Curves');
legend('Original Curve', 'Reconstructed Curve');
grid on;
hold off;

% 计算误差
y_recon_interp = interp1(x_recon_points, y_recon_points, x_samples); % 在原始曲线的 x
    点处插值重构曲线的 y 值
errors = y_recon_interp - y_samples; % 误差向量
rmse = sqrt(mean(errors.^2)); % 均方根误差
mae = mean(abs(errors)); % 平均绝对误差
max_error = max(abs(errors)); % 最大误差
endpoint_error = abs(y_recon_points(end) - y_samples(end)); % 末端误差

% 显示误差
fprintf('均方根误差 (RMSE): %.9f\n', rmse);
```

```
fprintf('平均绝对误差 (MAE): %.9f\n', mae);
fprintf('最大误差: %.9f\n', max_error);
fprintf('末端误差: %.9f\n', endpoint_error);
```

```
          %q3.2.2四阶龙格-库塔法
          % 使用RK4求解
while x < 1
    k1x = h * cos(theta);
    k1y = h * sin(theta);
    k1t = h * curvature(x);

    k2x = h * cos(theta + 0.5 * k1t);
    k2y = h * sin(theta + 0.5 * k1t);
    k2t = h * curvature(x + 0.5 * k1x);

    k3x = h * cos(theta + 0.5 * k2t);
    k3y = h * sin(theta + 0.5 * k2t);
    k3t = h * curvature(x + 0.5 * k2x);

    k4x = h * cos(theta + k3t);
    k4y = h * sin(theta + k3t);
    k4t = h * curvature(x + k3x);

    x = x + (k1x + 2*k2x + 2*k3x + k4x) / 6;
    y = y + (k1y + 2*k2y + 2*k3y + k4y) / 6;
    theta = theta + (k1t + 2*k2t + 2*k3t + k4t) / 6;

    xs = [xs, x];
    ys = [ys, y];
end
```

```
          %q3.2.3有限差分法
          % 使用二阶有限差分法迭代
while x < 1
    x_next = x + h;
    df_x = df(x, h);
    ddf_x = ddf(x, h);
    y_next = y + df_x * h + 0.5 * ddf_x * h^2; % 使用泰勒展开预测下一个 y 值

    % 更新当前点
    x = x_next;
    y = y_next;

    % 存储新的点
    xs = [xs, x];
    ys = [ys, y];
```

```
end
```

```
        %q3.2.4模拟退火法
        function main
% 定义原始函数及其导数
f = @(x) x.^3 + x;
df = @(x) 3*x.^2 + 1;
ddf = @(x) 6*x;

% 计算曲率的函数
curvature = @(x) abs(ddf(x)) ./ (1 + df(x).^2).^(3/2);

% 初始曲线 - 直线 y = x
x = linspace(0, 1, 100);
y = x; % 初始猜测解

% 模拟退火参数
T = 0.5;    % 降低初始温度
T_min = 1e-6; % 更小的最小温度
alpha = 0.95; % 更低的温度衰减率
max_iter = 500; % 增加最大迭代次数

% 模拟退火迭代过程
while T > T_min
    for i = 1:max_iter
        % 生成新的解决方案
        new_y = y + (rand(size(y)) - 0.5) * T;
        new_y(1) = 0; % 确保始终从原点开始
        new_y(end) = f(1); % 确保末尾点正确

        % 计算能量差
        energy_diff = sum((f(x) - new_y).^2) - sum((f(x) - y).^2);

        % 概率接受准则
        if energy_diff < 0 || (exp(-energy_diff / T) > rand)
            y = new_y;
        end
    end
    % 降低温度
    T = T * alpha;
end

% 计算误差
y_actual = f(x); % 计算原始曲线上的实际 y 值
errors = y_actual - y; % 计算误差
rmse = sqrt(mean(errors.^2)); % 计算均方根误差
```

```
    mae = mean(abs(errors));  % 计算平均绝对误差
    max_error = max(abs(errors)); % 计算最大误差
    end_point_error = abs(y(end) - y_actual(end)); % 计算末端误差

    % 输出误差
    fprintf('均方根误差 (RMSE): %.9f\n', rmse);
    fprintf('平均绝对误差 (MAE): %.9f\n', mae);
    fprintf('最大误差: %.9f\n', max_error);
    fprintf('末端误差: %.9f\n', end_point_error);

    % 绘制原始曲线和重构曲线
    figure;
    plot(x, f(x), 'b-', 'LineWidth', 2);
    hold on;
    plot(x, y, 'r--', 'LineWidth', 1);
    title('Comparison between Original Curve and Reconstructed Curve using Simulated
        Annealing');
    xlabel('x');
    ylabel('y');
    legend('Original Curve', 'Reconstructed Curve');
    grid on;
    hold off;
end
```