



廣東工業大學

课 程 设 计

题目名称 面向非圆曲线的 CAM 软件开发

学生学院 机电工程学院

专业班级

姓 名

学 号

指导老师

年 月 日

广东工业大学“数控技术”课程设计任务书

题目名称 面向非圆曲线的 CAM 软件开发

学生学院 机电工程学院

专业班级

姓 名

学 号

一、课程设计的内容

用计算机高级编程语言（如 VB, VC++, C#, Python 等）来实现非圆曲线的计算机辅助制造(CAM)软件的开发，针对不同的非圆曲线，可任选（1）直线逼近（如等间距法、等弦长法、等误差法等）、或（2）圆弧逼近的方法产生节点。要求在满足允许误差的前提下，使得逼近的直线段或圆弧段的数量最少（即最优解），根据加工曲线轮廓自动生成刀具中心轨迹，自动生成加工 NC 代码，并能模拟实际加工走刀过程。

二、课程设计的要求与数据

针对非圆曲线的逼近处理，要求：

- （1）列出一般的直线或圆弧逼近的算法（流程图）。
- （2）列出改进的直线或圆弧逼近的算法（流程图）——即优化算法。比较改进前与改进后的两种算法结果。

刀具补偿功能的实现：

- （1）针对给定的某一由非圆曲线所构成的平面轮廓，根据指定的走刀方向、起刀点，自动生成 CNC 代码。
- （2）有刀具自动补偿功能（左右刀补均要有实现），根据给定的补偿量和进给方向自动计算刀具中心轨迹，有过切报警功能。

对逼近处理后的轮廓进行仿真加工，要求：

- （1）列出要采用的插补算法的四个象限的流程图，设置进给轴的脉冲当量以及插补运算周期，分别实现脉冲增量插补和数字增量插补算法，对比两个插补算法的不同。
- （2）在屏幕上显示该非圆曲线所构成的平面轮廓。根据给定的进给速度能模拟加工过程，并在屏幕上留下刀具所走中心轨迹。

非圆曲线选择:

- 0: 抛物线
- 1: 椭圆曲线
- 2: 阿基米德螺线/等速螺线 (900 °)
- 3: 幂函数曲线。 $f(x)=x^a$ 的函数, $a=4$ 或 8
- 4: 幂函数曲线。 $f(x)=x^a$ 的函数, $a=1/4$ 或 $1/8$
- 5: 焦点 Y 轴上的双曲线
- 6: 心脏线
- 7: 匀加减速盘形凸轮廓线
- 8: 余弦加减速盘形凸轮廓线 (四段: 升程-远休-回程-近休)
- 9: 正弦加减速盘形凸轮廓线 (四段: 升程-远休-回程-近休)

三、课程设计应完成的工作

每组学生应在规定时间内, 独立完成所选题目。运用 VC 或其它编程语言, 编写计算机软件在 WINDOWS 实现数控装置的计算机仿真。要求清楚地分析问题、提出算法、确定人机界面、列出流程图, 最后用程序验证, 完成软件测试, 并且提交程序说明书。

要求用编写计算机软件的方法解决典型非圆曲线的 CAM 问题。可以任选用自己熟悉的一种编程语言, 要求清楚地分析问题、提出算法、列出流程图, 最后用程序验证, 并且提交程序说明书。

四、课程设计进程安排

序号	设计各阶段内容	地点	起止日期
1	1、布置任务, 领取课程设计任务书, 了解课程设计的目的、内容和要求; 了解课程设计的步骤; 2、理解本课程设计题目的具体内容要求, 根据各自不同情况选择题目;	学 生 宿舍	13 周一~周二
	1: 了解和掌握有关软件开发的知识, 如 VB 编程、VC 编程、软件工程、软件开发的常用技巧及注意事项; 2: 调查研究, 收集资料, 查阅文献。学生对所选题目进行算法设计及方案设计。		13 周三~周五
2	1: 掌握数控结构设计要求, 具体技术指标和计算要求; 进行软件设计; 2: 进行数控系统和算法的软件编程与开发, 初步实现系统的基本功能	学 生 宿舍	14 周一~周二
3	1: 通过多个实例来验证和改进系统功能, 完善软件界面 2: 对所开发的软件程序进行标识和说明 3: 按要求的格式编制课程设计说明书 4: 课程设计答辩	学 生 宿舍	14 周三~周五

五、应收集的资料及主要参考文献

- 1 廖效果. 数控技术. 武汉:湖北科学技术出版社. 2000. 7
- 2 刘又午. 数字控制机床. 北京:机械工业出版社
- 3 杨林, 李继良. Visual Basic 编程高手. 北京:北京大学出版社. 2000
- 4 一组专用凸轮的计算机辅助设计. 机械工程师 1998, (4): p58-59
- 5 平面凸轮机构 CAD 系统的研究与开发. 机械设计与制造 2000, (5):p12-13
- 6 圆柱非圆曲线槽凸轮的数控加工. 制造技术与机床 2000, (8):p34
- 7 圆柱凸轮的参数化设计及数控加工. 精密制造及自动化 2001, 11: p28
- 8 参数化凸轮轮廓转换及 NC 代码自动生成. 机床与液压 2001, 6, p29~31
- 9 非圆曲线的等误差拟合数控节点算法研究. 数字技术与应用. 2010.07
- 10 椭圆弧等误差直线拟合的算法研究与轨迹仿真[J]. 山东理工大学学报(自然科学版) 2010.01
- 11 赵佩凤. 刀具补偿控制系统的研究与开发[D]. 大连交通大学. 2010
- 12 孙海洋, 范大鹏. 一种新的刀具圆弧过渡补偿算法[J]. 中国机械工程. 2007.03

发出任务书日期：2024 年 4 月 24 日

计划完成日期：2024 年 5 月 31 日

指导教师签名：

基层教学单位责任人签章：

主管院长签章：

目录

一、课设选题	6
1.1 非圆曲线的选择	6
1.2 课程设计内容	6
1.3 具体要求	6
二、直线逼近的原理	7
2.1 等间距法	7
2.1.1 基本原理	7
2.1.2 算法实现步骤	8
2.2 等误差法	9
2.2.1 基本原理	9
2.2.2 算法实现步骤	11
三、算法优化前后分析	12
3.1 等间距法（优化前算法）	12
3.2 等误差法（优化后算法）	12
3.3 优化前后算法效果对比	12
四、软件系统设计及功能说明	14
4.1 开发工具选择	14
4.2 PyQt6 第三方软件库介绍	14
4.3 人机交互页面介绍	14
4.4 生成 NC 代码原理	17
五、刀具半径补偿	20
5.1 刀具半径补偿原理	20
5.2 刀具半径补偿的转接方式	20
5.3 刀具半径补偿算法实现	22
5.3.1 直线方向矢量的计算	22
5.3.2 方向矢量与刀具半径矢量的关系	23
5.4 刀具半径补偿过切的判断	24
5.5 刀具半径补偿转接点的计算流程	25
六、仿真加工效果演示	26
6.1 心脏线刀补显示	26
6.2 过切报警	26
6.3 正弦加减速盘形凸轮廓线	27
6.4 过切报警	27
七、设计总结	28
参考文献	29
附录	29

一、课设选题

1.1 非圆曲线的选择

- (1) 6: 心脏线
- (2) 9: 正弦加减速盘形凸轮廓线（四段：升程-远休-回程-近休）

1.2 课程设计内容

使用直线逼近（如等间距法、等误差法等）的方法产生节点。要求在满足允许误差的前提下，使得逼近的直线段的数量最少（即最优解），根据加工曲线轮廓自动生成刀具中心轨迹，自动生成加工 NC 代码，并能模拟实际加工走刀过程。

1.3 具体要求

- (1) 列出一般的直线逼近的算法（流程图）。
- (2) 列出改进的直线逼近的算法（流程图）——即优化算法。比较改进前与改进后的两种算法结果。
- (3) 针对给定的某一由非圆曲线所构成的平面轮廓，根据指定的走刀方向、起刀点，自动生成 CNC 代码。
- (4) 有刀具自动补偿功能（左右刀补均要有实现），根据给定的补偿量和进给方向自动计算刀具中心轨迹，有过切报警功能。
- (5) 在屏幕上显示该非圆曲线所构成的平面轮廓。根据给定的进给速度能模拟加工过程，并在屏幕上留下刀具所走中心轨迹。

二、直线逼近的原理

2.1 等间距法

2.1.1 基本原理

等间距法是在非圆曲线与逼近直线的最大偏差小于允许偏差 ($\delta_{\text{最大}} \leq \delta_{\text{允}}$) 的条件下, 将某一坐标轴或曲线方程中的某一参变量划分成相等的间距。具体来说, 假设沿 X 轴方向取 Δx 为等间距长度, 根据已知曲线的方程可得 $y_i = f(x_i)$, 之后沿 X 轴减小 (或增加) 一个间距 Δx , 即 $x_{i+1} = x_i \pm \Delta x$, $y_i = f(x_{i+1})$, 循环求得一系列等间距节点的坐标值。

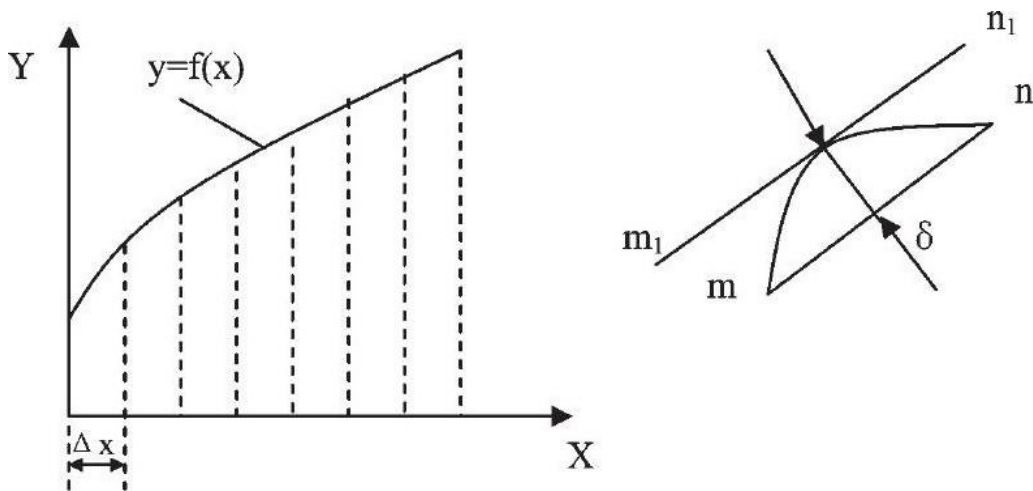


图 1: 等间距法原理图

由于要求曲线 $y=f(x)$ 与相邻两节点连线间的法向距离小于允许的程序编制误差 $\delta_{\text{允}}$, Δx 值不能任意设定。一般先取 $\Delta x = 0.1$ 或 100 倍左右 $\delta_{\text{允}}$ 进行试算。实际处理时, 并非任意相邻两点间的误差都要验算, 对于曲线曲率半径变化较小处, 只需验算两节点间距离最长处的误差, 而对曲线曲率半径变化较大处, 应验算曲率半径最小处的误差, 具体计算方法如下:

(1) 使用曲率半径公式 $\rho = \frac{(1+y'^2)^{\frac{3}{2}}}{y''}$ 求出轮廓图形的曲率半径表达式。

(2) 对曲率半径表达式求导, 并令 $\frac{d\rho}{dx} = 0$, 结合边界点求出曲率半径最小处的坐标 (x_ρ, y_ρ) 。

(3) 根据点 (x_ρ, y_ρ) 的位置 $x_{\text{起点}} + n \times \Delta x \leq x_\rho \leq x_{\text{起点}} + (n+1) \times \Delta x$ 判

断需要校验的曲线段(假设为图 1 中的 mn 段, 即 $x_m \leq x_p \leq x_n$), 并求出实际最大误差 $\delta_{\text{实}}$ 。

则 m、n 两点的直线方程为: $\frac{x-x_n}{y-y_n} = \frac{x_m-x_n}{y_m-y_n}$, 即: $(y_m - y_n)x + (x_n - x_m)y = x_n y_m - x_m y_n$

令 $A = (y_m - y_n)$, $B = (x_n - x_m)$, $C = x_n y_m - x_m y_n$

则 $Ax + By = C$, 即为过 m、n 两点的直线方程。

距 mn 直线为 δ 的等距线 $m_1 n_1$ 的直线方程为: $Ax + By = C \pm \delta \sqrt{A^2 + B^2}$

式中, 当所求直线 $m_1 n_1$ 在 mn 上边时取 “+” 号, 在 mn 下边时取 “-” 号。 δ 为与两直线间的距离。

解联立上述方程组, 由此方程组解出的 δ 即为实际最大误差 $\delta_{\text{实}}$ 。

2.1.2 算法实现步骤

S1: 输入曲线方程 $y=f(x)$ 各项系数, 起点坐标为 (x_a, y_a) , 终点坐标 (x_b, y_b) , 取 $\Delta x = 0.1$, 刀具走刀次数为 n;

S2: 计算任意一点的坐标为 $(x_0 + n\Delta x, f(x_0 + n\Delta x))$;

S3: 判断是否到达终点:

当 $x_0 + n\Delta x < x_b$ 时, 进行下一步 S4;

当 $x_0 + n\Delta x \geq x_b$ 时, 则直接进行 S5;

S4: 计算 $\delta_{\text{实}}$:

当 $\delta_{\text{实}} < \delta_{\text{允}}$ 时, 则 $n = n + 1$, 接着继续 S2;

当 $\delta_{\text{实}} \geq \delta_{\text{允}}$ 时, 则 $\Delta x = \Delta x - 0.01$, 接着继续 S2;

S5: 输出终点坐标结束

算法实现步骤的流程图如图 2 所示:

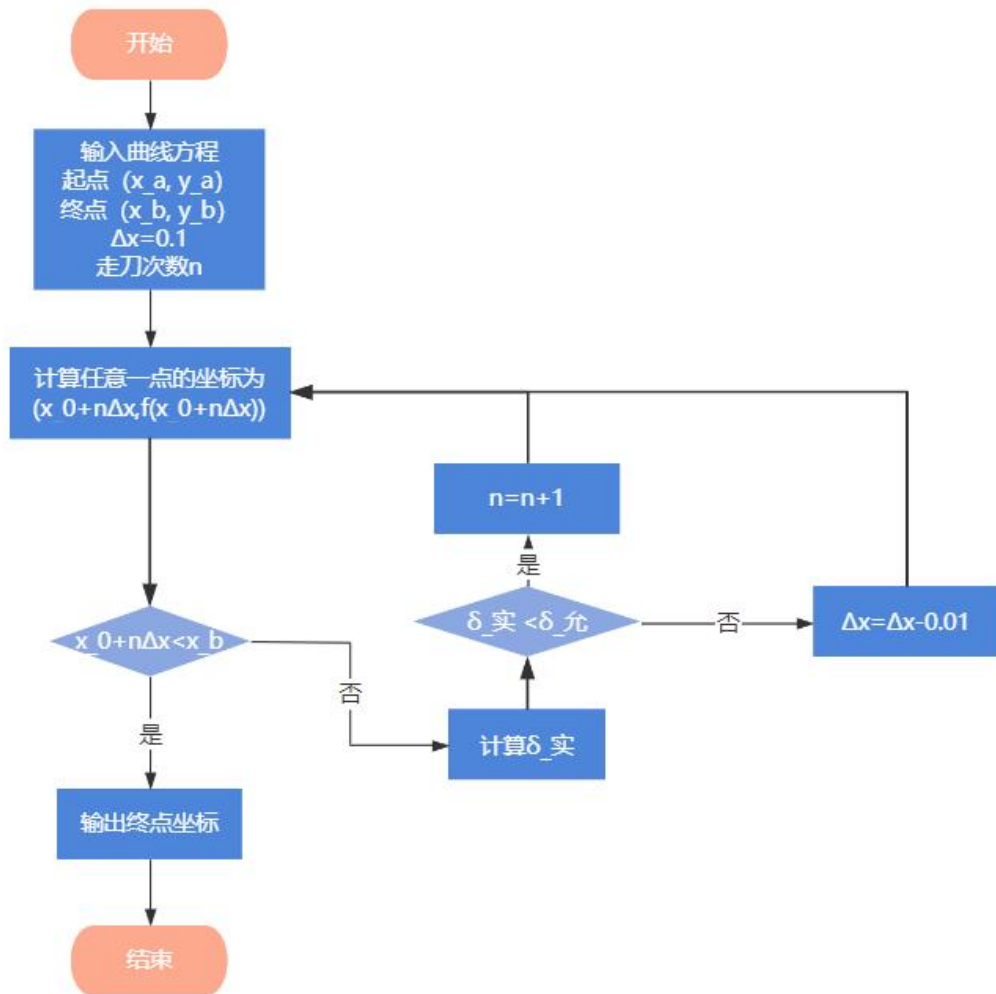


图 2：等间距法算法实现步骤的流程图

2.2 等误差法

2.2.1 基本原理

等误差直线逼近算法是平行线法，所谓平行线法，就是用相距为定值的平行线去相割与相切被逼近的曲线，其中一条直线与曲线相割，其交点称为节点；另一条直线与曲线相切，即与曲线相割的直线与相切的直线的误差 δ 相等。等误差的原理图如图 3 所示：

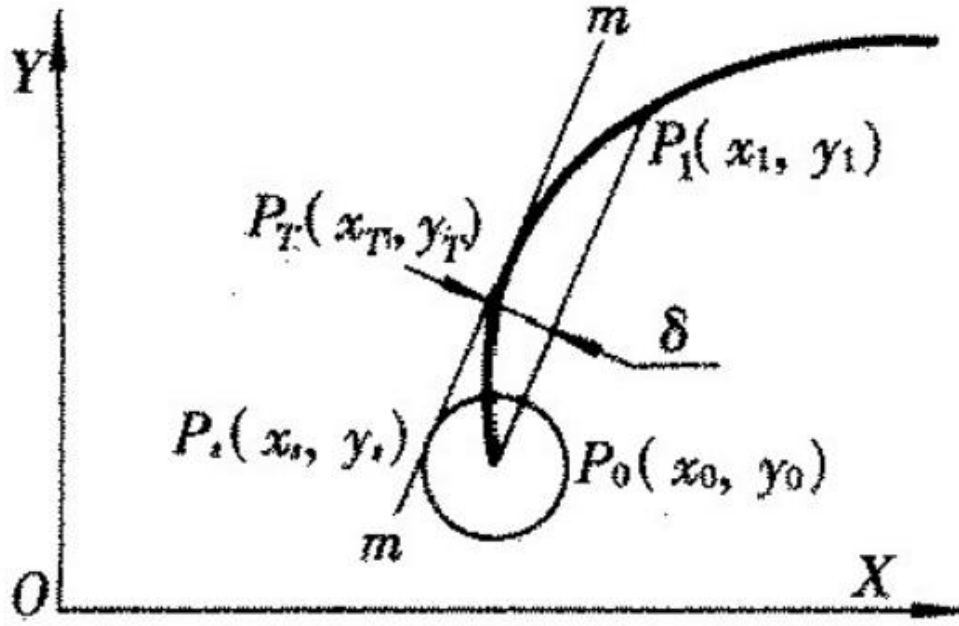


图 3：等误差法原理图

节点坐标具体求解步骤如下：

(1) 求逼近直线的斜率(如图 3 所示) 设线性非圆曲线 $y = f(x)$, $P_0(x_0, y_0)$ 为曲线的起点, δ 为允许误差。现以 P_0 为圆心、 δ 为半径作一个误差圆, 然后作一条与误差圆、非圆曲线 $y = f(x)$ 均相切的直线, 即公切线 m-m, 切点分别为: $P_s(x_s, y_s)$ 、 $P_T(x_T, y_T)$, 由此可得:

$$\text{误差圆方程: } (x_s - x_0)^2 + (y_s - y_0)^2 = \delta^2 \quad (1)$$

$$\text{曲线方程: } y_T = f(x_T) \quad (2)$$

$$\text{公切线斜率: } k = -\frac{x_s - x_0}{y_s - y_0} \quad (3)$$

$$\text{公切线方程: } y_T - y_s = k(x_T - x_s) \quad (4)$$

$$\text{起点到公切线的距离: } \delta = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}} \quad (5)$$

式中, $A = y_T - y_s$; $B = x_T - x_s$; $C = y_s x_T - x_s y_T$

联立式(1)~(5), 求解可得斜率 k 。

2) 求节点坐标, 第一个节点用下列方程组求得:

$$\begin{cases} y_1 - y_0 = k(x_1 - x_0) \\ y_1 = f(x_1) \end{cases}$$

对其它节点, 可用已求出的节点作为起点, 依次重复使用式(1)~(5)获取对应

直线的斜率 k , 并用节点方程组求取节点 $P_i(x_i, y_i)$ 的坐标,

$$\begin{cases} y_i - y_{i-1} = k(x_i - x_{i-1}) \\ y_i = f(x_i) \end{cases} \quad (6)$$

由此, 所有节点坐标均可通过联立式(1)~(6)解得。

2.2.2 算法实现步骤

S1: 输入曲线 $y = f(x)$ 函数, 起点坐标为 (x_0, y_0) , 终点坐标为 (x_i, y_i) ;

S2: 以节点 (x_n, y_n) 为圆心、 $\delta_{允}$ 为半径作误差圆;

S3: 求出误差圆与曲线的公切线斜率 k ;

S4: 以节点 (x_n, y_n) 为起点, 公切线斜率 k 为斜率作逼近直线

S5: 判断 $x_n < x_i$ 是否成立:

若成立, 则 $n = n + 1$, 重复 S2、S3、S4、S5;

若不成立, 则输出曲线轮廓线的终点坐标, 结束。

算法实现步骤的流程图如图 4 所示:

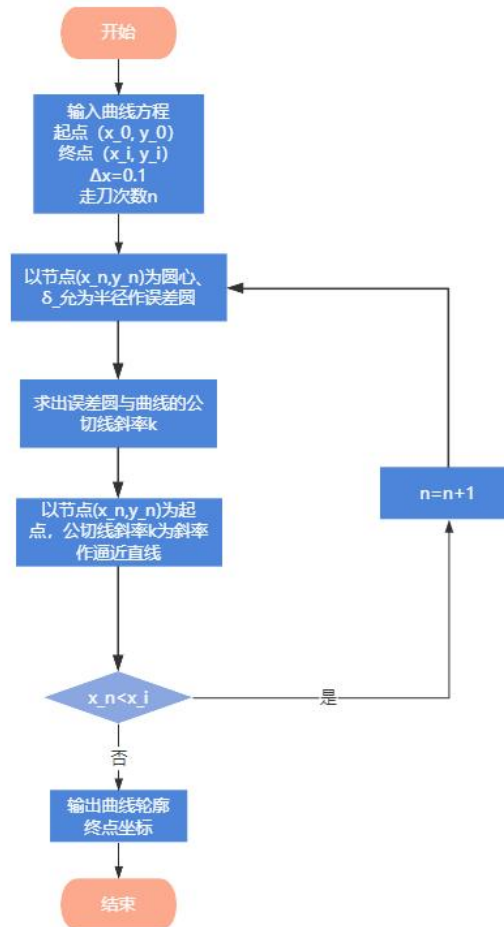


图 4: 等误差法算法实现流程图

三、算法优化前后分析

非圆曲线的逼近原则是在保证逼近精度的前提下，使节点数目尽量少，以缩短程序长度，提高加工效率。由于圆弧逼近计算相当复杂，使得手工编程实现几乎无法做到，与其相比，直线逼近法数学处理简单，因此在加工精度不太高时被广泛采用，目前有等间距、等误差等几种节点计算方法。

3.1 等间距法（优化前算法）

等间距法就是将某一坐标轴划分成相等的间距，将相邻节点连成直线，用这些直线段组成的折线代替原来的曲线，间距 Δx 取得越大，产生的实际误差就越大，并且 Δx 是我们给定的定值，因此这个方法的程序段较多，编程会比较费时。经过程序验证和结果输出，等间距法适用于曲率变化小的非圆曲线。而我们小组选择的曲线是心脏线和正弦加减速凸轮轮廓线，这两个曲线的曲率变化都比较大，所以在同等逼近精度条件下，使用等间距法产生的节点会比较多。

3.2 等误差法（优化后算法）

等误差法是通过使各节点之间曲线与直线的误差 δ 相等来确定节点坐标。由于曲率半径的变化，逼近的直线段步长会渐增大，但每段的逼近误差保持基本相等。确定允许误差的圆方程，以起点为圆心，以允许误差为半径作圆。求解该圆与加工曲线的公切线，以及公切线与曲线的交点，这些交点即为逼近的节点。将相邻的节点连成直线，这些线段组成的折线代替原来的曲线。在相同的条件下，等误差逼近法实现了误差可控，同时所需计算的节点数最少，简化了编程前的数值处理。而且加工出的零件表面精度和尺寸精度高。等误差直线逼近法在数控加工中广泛应用于非圆曲线（如渐开线、抛物线等）的高精度加工。而我们小组选择的曲线是心脏线和正弦加减速凸轮轮廓线，这两个曲线的曲率变化都比较大，所以在同等逼近精度条件下，使用等误差法产生的节点会少很多，等误差法比等间距法要好一些。

3.3 优化前后算法效果对比

分别用优化前的等间距法和优化后的等误差法对心脏线和正弦加减速凸轮轮廓线的曲线进行直线逼近，并且取允许误差为 0.1，利用 python 编写程序对算法优化前后处理的数据进行比较分析，下表为分析结果，其中心脏线的起始坐标为(0.0020, 5.0511)，正弦加减速凸轮轮廓线的起始坐标为(8.0000, 0.0000)，

迭代计算 20 步，得出下表优化前后的结果，即两种曲线优化前后直线逼近节点坐标数据显示。同样是迭代 20 步，心脏线优化前等间距法到达终点在 $x=0.0140$ 的位置，优化后的等误差算法到达终点在 $x=0.2149$ 的位置，而正弦加减速凸轮轮廓线优化前的等间距法到达终点在 $x=7.9686$ 的位置，优化后的等误差算法到达终点在 $x=7.8842$ 的位置，由此可知，优化后的等误差算法结果明显好于优化前的等间距算法结果。

曲线方程	心脏线方程		正弦加减速凸轮轮廓线	
步数	等间距法 (优化前)	等误差法 (优化后)	等间距法 (优化前)	等误差法 (优化后)
1	0.0020	0.0020	8.0000	8.0000
2	0.0022	0.0034	7.9999	7.9996
3	0.0025	0.0055	7.9996	7.9984
4	0.0029	0.0082	7.9991	7.9965
5	0.0033	0.0116	7.9985	7.9939
6	0.0037	0.0159	7.9977	7.9905
7	0.0042	0.0212	7.9966	7.9865
8	0.0047	0.0274	7.9954	7.9819
9	0.0052	0.0349	7.9941	7.9766
10	0.0058	0.0435	7.9926	7.9707
11	0.0064	0.0534	7.9909	7.9643
12	0.0070	0.0647	7.9890	7.9573
13	0.0077	0.0775	7.9870	7.9497
14	0.0085	0.0918	7.9848	7.9417
15	0.0093	0.1078	7.9825	7.9332
16	0.0101	0.1255	7.9800	7.9243
17	0.0110	0.1449	7.9774	7.9149
18	0.0119	0.1663	7.9746	7.9050
19	0.0129	0.1896	7.9717	7.8948
20	0.0140	0.2149	7.9686	7.8842

四、软件系统设计及功能说明

4.1 开发工具选择

这次数控技术课程设计经过小组讨论，选择基于 Python 语言的 PyQt6 库来设计软件的 UI 界面。

4.2 PyQt6 第三方软件库介绍

PyQt6 是 Python 编程语言的一个绑定库，它无缝集成了 Qt6 应用框架的功能，旨在帮助开发者高效地创建跨平台的图形界面应用程序。Qt6 是一个广泛采用的 C++ 图形库，提供了强大的 UI 设计工具、丰富的控件集合、2D/3D 图形渲染能力、网络支持、数据库集成、多媒体处理等高级特性。PyQt6 作为其面向 Python 的接口，不仅封装了 Qt6 的大量功能，还充分利用 Python 的简洁语法和广泛的生态系统，使得开发者可以更加灵活、高效地开发桌面应用、移动应用乃至嵌入式系统界面。

4.3 人机交互页面介绍

我们通过对本次课程设计内容的分析，做出以下的人机交互界面，本次设计的软件为一个基于 Windows 系统的 CAM 软件。如图 5 软件程序流程图所示。

如图 6 人机交互页面所示，在页面窗口的左边为曲线轮廓的参数设置。该软件提供心脏线和正弦加减速盘形凸轮廓线两种非圆曲线可供选择，并可通过输入可变参数来获得目标的曲线图形，如图 7 所示。同时，该软件也提供等间距插补算法和等误差插补算法来实现加工曲线轮廓的生成，并可输入允许的误差来调整插补的精度要求，如图 8 所示。在默认的情况下，软件会预先选择心脏线并给定一个默认参数 a ，同时默认选择等间距的插补算法，允许误差为 0.1。在页面窗口的右边是加工轮廓的显示区域，该区域能够将左边设置的非圆曲线实时地绘画呈现，同时我们也采用了不同的画笔颜色（红色代表等间距插补算法，蓝色代表等误差插补算法）来区分不同插补算法绘制出的轮廓曲线。在页面的下方有四个黄色的按钮，当修改完曲线参数后，可点击下方的“生成函数”按钮，则在右边的绘图区域将能快速生成目标轮廓曲线。点击“关于”按钮则能显示我们小组成员以及指导老师的信息。点击“关闭程序”按钮则程序将自动停止，窗口消失。

当设置完参数后，点击“仿真加工”按钮，则软件跳转到仿真加工页面，如图 9 所示。在该页面的左边加工参数设置区域可以选择“走刀方向”、“刀补方

式”等，同时也可在下方设置刀具半径和进给速度的参数。当加工参数设置完毕后，点击下方的“生成仿真动画按钮”，则在页面的右边区域将显示出仿真加工的动画，红色代表编程轨迹，蓝色代表刀具中心轨迹。通过修改刀具半径的大小，则刀具中心轨迹也能够进行修正改变。而更改进给速度的大小则能够改变加工仿真动画运动的速度。如果想返回曲线设置页面，我们提供一个“返回曲线设置”按钮，点击后软件将跳转回第一个页面进而方便用户重新设置曲线参数。在第二个页面中，软件默认设置走到方向为顺时针，刀补方式为左刀补，并自动设置默认的加工参数，用户可根据自己的需要进行调参。

设置完加工参数后，点击“生成 NC 代码”按钮，则软件跳转到第三个页面，在该页面的左侧将显示出数控机床控制所需的代码，如图 10 所示。若要重新设置加工参数，可点击“返回仿真加工设置”按钮，方便用户快速调整加工参数。

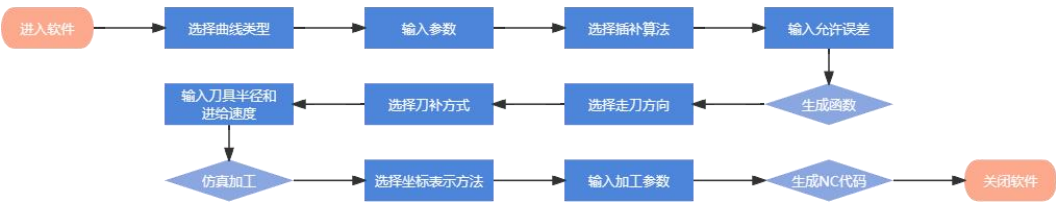


图 5：软件程序流程图

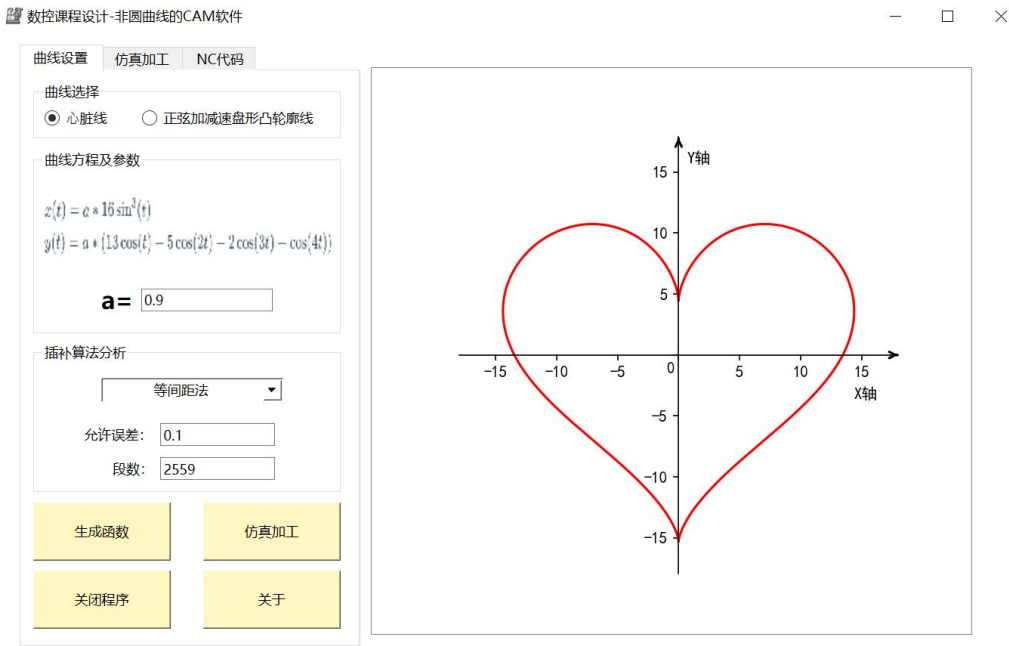


图 6：人机交互页面

曲线选择

☒ 心脏线
☐ 正弦加减速盘形凸轮廓线

曲线方程及参数

$$x(t) = a * 16 \sin^3(t)$$

$$y(t) = a * (13 \cos(t) - 5 \cos(2t) - 2 \cos(3t) - \cos(4t))$$

a=

曲线选择

☐ 心脏线
☒ 正弦加减速盘形凸轮廓线

曲线方程及参数

基圆半径:

行程:

远休止角:

近休止角:

图 7：曲线参数页面

插补算法分析

等间距法

等间距法

等误差法

允许误差

段数:

图 8：算法选择页面

数控课程设计-非圆曲线的CAM软件

— □ ×

曲线设置

仿真加工

NC代码

走刀方向

☒ 顺时针
☐ 逆时针

刀补方式

☒ 左刀补
☐ 右刀补

加工参数

☒ 绝对坐标
☐ 相对坐标

刀具半径:

进给速度:

主轴转速:

安全高度:

加工深度:

起刀点

X=

Y=

Z=

生成仿真加工动画

生成NC代码

返回曲线设置

仿真加工

图 9：仿真加工页面

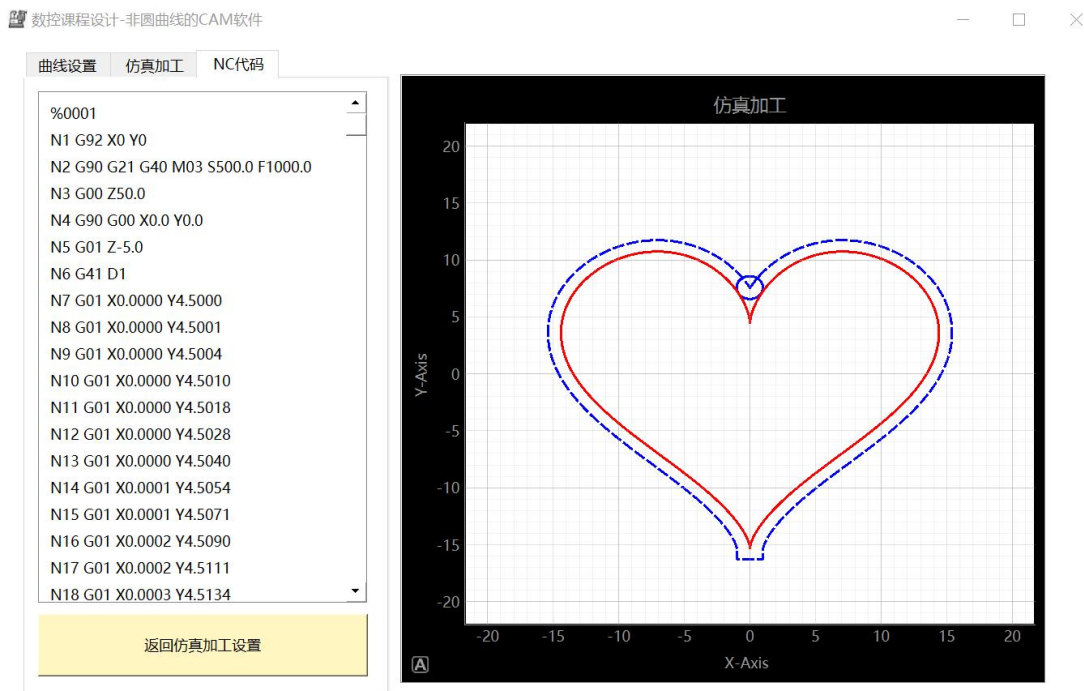


图 10: NC 代码生成页面

4.4 生成 NC 代码原理

如图 11 程序转换代码所示，首先利用 PyQt6 软件获得插补后的轮廓曲线基础数据，再利用 Python 软件编程对该数据进行处理转换。通过设置 G、M、F、S 的符号数组，用 if 条件判断在 NC 代码的开头输出各种指令代码，然后中间调用算法计算出的各个节点的 x、y 坐标，结尾再输出各种结束指令代码。

For 循环：主要是为了把每个节点的坐标及其各种指令代码输出到 NC 代码中。

if 条件判断：逐步输出 N 序号，G 指令。

```

1. def generate_nc_code(t_values, x_values, y_values, tool_radius, feed_rate, spindle_speed,
2.                      safety_height, cut_depth, start_point, compensation_direction,
3.                      use_relative_coordinates):
4.     nc_code = []
5.     # 初始化 NC 代码
6.     nc_code.append("%0001")
7.     # 设置坐标系和起刀点
8.     if use_relative_coordinates:
9.         # nc_code.append("G91") # 使用相对坐标
10.        nc_code.append(f"N1 G91 G21 G40 M03 S{spindle_speed} F{feed_rate}") # 在相对坐标下移动到起刀点（需要确保是相对于初始位置的偏移）

```

```

11.         nc_code.append(f"N2 G00 Z{safety_height}")
12.         start_x, start_y = start_point
13.         nc_code.append(f"N3 G91 G00 X{start_x} Y{start_y}")
14.         nc_code.append(f"N4 G01 Z-{cut_depth + safety_height}") # 下刀到加工深度
15.
16.         # 设置刀具补偿方向
17.         if compensation_direction == 'left':
18.             nc_code.append("N5 G41 D1") # 左刀补
19.         elif compensation_direction == 'right':
20.             nc_code.append("N6 G42 D1") # 右刀补
21.         # 遍历离散点并生成直线插补指令
22.         for i in range(len(t_values) - 1):
23.             xi, yi = x_values[i], y_values[i]
24.             xi_next, yi_next = x_values[i + 1], y_values[i + 1]
25.             if use_relative_coordinates:
26.                 dx = xi_next - xi
27.                 dy = yi_next - yi
28.                 nc_code.append(f"N{i + 5} G01 X{dx:.4f} Y{dy:.4f}") # 使用相对坐标进行直线插补
29.             else:
30.                 nc_code.append(f"N{i + 5} G01 X{xi_next:.4f} Y{yi_next:.4f}") # 使用绝对坐标进行直线插补
31.             if use_relative_coordinates:
32.                 nc_code.append(f"N{len(t_values) + 5} G40 G00 Z{cut_depth + safety_height}")
33.             else:
34.                 nc_code.append(f"N{len(t_values) + 5} G40 G00 Z{safety_height}") # 快速定位到安全高度
35.         nc_code.append(f"N{len(t_values) + 6} M30") # 程序结束
36.         return nc_code
37.     else:
38.         # nc_code.append("G90") # 使用绝对坐标
39.         nc_code.append(f"N1 G92 X0 Y0")
40.         nc_code.append(f"N2 G90 G21 G40 M03 S{spindle_speed} F{feed_rate}") # 在相对坐标下移动到起刀点（需要确保是相对
        于初始位置的偏移）
41.         nc_code.append(f"N3 G00 Z{safety_height}")
42.         start_x, start_y = start_point
43.         nc_code.append(f"N4 G90 G00 X{start_x} Y{start_y}")
44.         nc_code.append(f"N5 G01 Z-{cut_depth}") # 下刀到加工深度
45.         # 设置刀具补偿方向
46.         if compensation_direction == 'left':
47.             nc_code.append("N6 G41 D1") # 左刀补
48.         elif compensation_direction == 'right':
49.             nc_code.append("N7 G42 D1") # 右刀补
50.         # 遍历离散点并生成直线插补指令
51.         for i in range(len(t_values) - 1):
52.             xi, yi = x_values[i], y_values[i]
53.             xi_next, yi_next = x_values[i + 1], y_values[i + 1]

```

```

54.         if use_relative_coordinates:
55.             dx = xi_next - xi
56.             dy = yi_next - yi
57.             nc_code.append(f"N{i + 7} G01 X{dx:.4f} Y{dy:.4f}") # 使用相对坐标进行直线插补
58.         else:
59.             nc_code.append(f"N{i + 7} G01 X{xi_next:.4f} Y{yi_next:.4f}") # 使用绝对坐标进行直线插补
60.         if use_relative_coordinates:
61.             nc_code.append(f"N{len(t_values) + 6} G40 G00 Z{cut_depth + safety_height}")
62.         else:
63.             nc_code.append(f"N{len(t_values) + 6} G40 G00 Z{safety_height}") # 快速定位到安全高度
64.         nc_code.append(f"N{len(t_values) + 7} M30") # 程序结束
65.     return nc_code

```

图 11: 程序转换代码图

五、刀具半径补偿

5.1 刀具半径补偿原理

为了方便零件加工程序编制, 编程轨迹为零件轮廓轨迹, 而 CNC 系统控制刀具移动的轨迹为刀具中心轨迹。由于刀具半径的存在, 零件轮廓轨迹与刀具中心轨迹不相重合。为了加工出符合图纸要求的零件轮廓, 必须进行刀具半径偏移。加工外轮廓时, 应向轮廓外偏移一个刀具半径; 加工内轮廓时, 应向轮廓内偏移一个刀具半径。

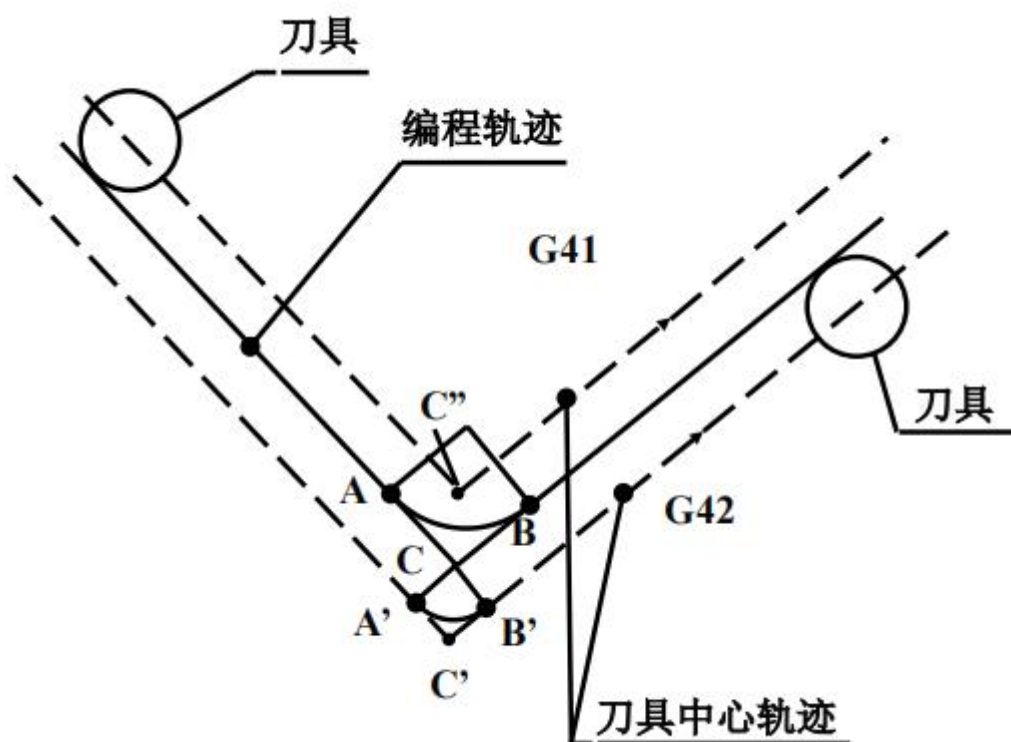


图 12: 刀补原理示意图

5.2 刀具半径补偿的转接方式

根据两段编程轨迹的矢量夹角和刀补方向的不同 (两段程序段轨迹在工件侧的夹角 (转接角) 和刀具补偿 (G41 或 G42) 的不同), 刀具中心轨迹从一个编程段到另一个编程段的段间连接方式, 即转接方式可分为缩短型、伸长型和插入型三种, 这三种情况如下图 13 所示。下图 14 为不同转接形式的刀补建立撤销示意图, 图 15 为不同转接形式的刀补进行示意图。

缩短型: 刀具中心轨迹短于编程轨迹的过渡方式。 ($\alpha \geq 180^\circ$)

伸长型：刀具中心轨迹长于编程轨迹的过渡方式。（ $90^{\circ} \leq \alpha < 180^{\circ}$ ）

插入型：两段刀具中心轨迹间插入一段直线的过渡方式。（ $\alpha < 90^{\circ}$ ）

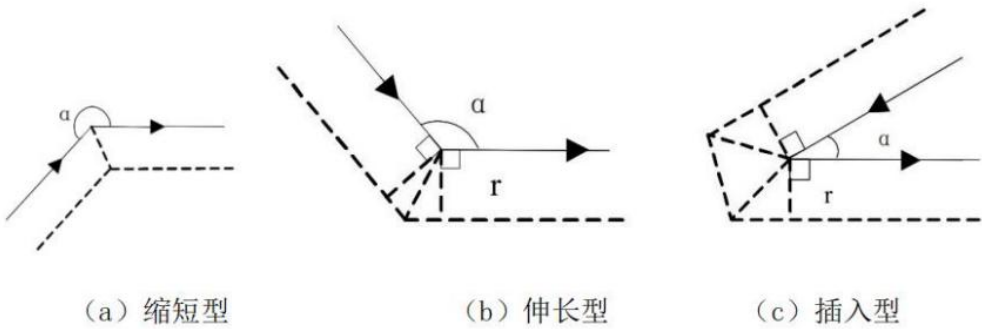


图 13：刀补转接形式

转接 形式 夹角	刀补建立 (G42)		刀补撤销 (G42)		过渡 方式
	直线 ——— 直线	直线 ——— 圆弧	直线 ——— 直线	圆弧 ——— 直线	
$\alpha \geq 180^{\circ}$					缩 短 型
$90^{\circ} \leq \alpha < 180^{\circ}$					伸 长 型
$\alpha < 90^{\circ}$					插 入 型

图 14：不同转接形式的刀补建立撤销示意图

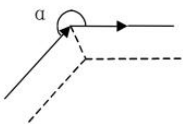
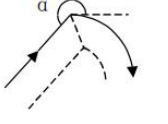
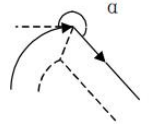
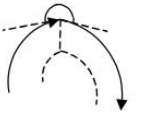
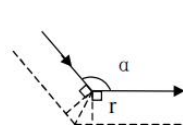
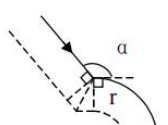
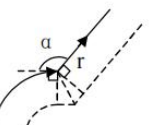
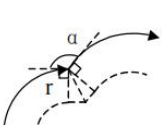
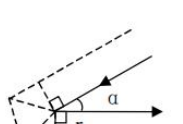
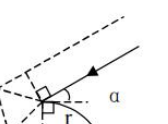
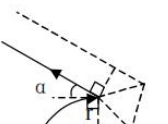
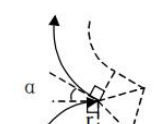
	刀 补 进 行 (G42)				过渡方式
	直线 ——— 直线	直线 ——— 圆弧	圆弧 ——— 直线	圆弧 ——— 圆弧	
$\alpha \geq 180^\circ$					缩短型
$90^\circ \leq \alpha < 180^\circ$					伸长型
$\alpha < 90^\circ$					插入型

图 15：不同转接形式的刀补进行示意图

5.3 刀具半径补偿算法实现

刀具中心轨迹是刀具半径补偿根据工件轮廓和刀具中心偏移量计算得出，在研究直线-直线、直线-圆弧、圆弧-直线和圆弧-圆弧四种转接的过程中，我们使用一种基于单位矢量的刀具半径补偿算法，通过计算加工轨迹的方向矢量和刀具半径矢量，从而计算出转接点的坐标值。此算法降低了计算的复杂性，同时避免了多组解取舍判别的问题。首先需要判断四种转接类型的矢量夹角的大小。为了算法的编程简单，在判断矢量夹角之前，先自定义方向矢量、刀具半径矢量、转接点等。

所谓方向矢量，是指在零件轮廓上与编程方向一致的单位矢量。零件轮廓上任意一点的方向矢量就是该点的单位切线矢量，其计算方法与零件轮廓的线型有关，下面讨论直线和圆弧的方向矢量。

5.3.1 直线方向矢量的计算

如图 16 所示，可得：

$$\begin{cases} \vec{l} = X_l \vec{i} + Y_l \vec{j} \\ X_l = (X_2 - X_1)/R \\ Y_l = (Y_2 - Y_1)/R \\ R = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \end{cases}$$

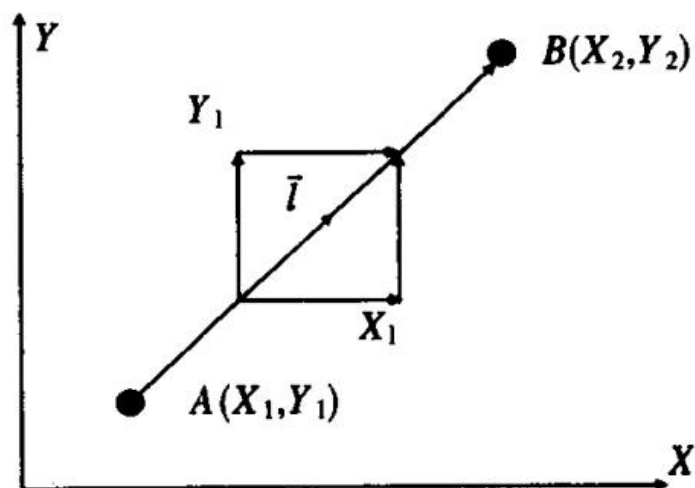


图 16：直线方向矢量示意图

5.3.2 方向矢量与刀具半径矢量的关系

在加工过程中始终垂直于编程轨迹、指向刀具中心且大小等于刀具半径的矢量称为刀具半径矢量。且有：

$$r = \begin{cases} |r|(\text{左刀补}) \\ -|r|(\text{右刀补}) \end{cases}$$

如图 17 所示，可得：

$$\begin{cases} \vec{r} = X_r \vec{i} + Y_r \vec{j} \\ X_r = -rY_l \\ Y_r = rX_l \end{cases}$$

所以：

$$\vec{r} = -rY_l \vec{i} + rX_l \vec{j}$$

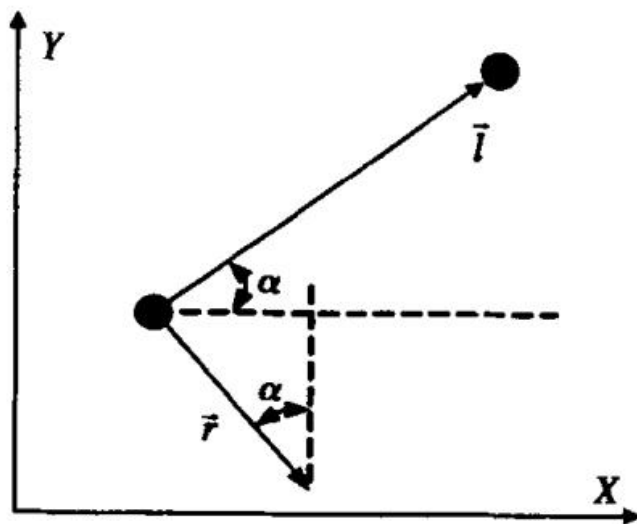


图 17：方向矢量与刀具半径矢量关系示意图

5.4 刀具半径补偿过切的判断

刀具半径补偿使用中出现的过切（即干涉）指的是在零件的加工过程中，刀具按照程序设定的轨迹进行运动，由于使用了刀具补偿功能，在执行某些指令时，出现或可能出现刀具过度切削零件的现象，过切现象如下图 18 所示。

根据本课设所选的曲线，对于可能出现的过切现象进行分析计算。所以针对缩短型—缩短型组合判断过切的方法，被铣削槽底宽小于刀具直径，当起用刀具半径补偿功能时，刀心轨迹从 P_1 点移动到 O 点，此时刀具把 A 区切除；由于刀具半径补偿功能使用，刀心轨迹从 O 点移动到 O' 点，此时刀具把 B 区切除。这就是被铣削槽底宽小于刀具直径，造成的过切现象。

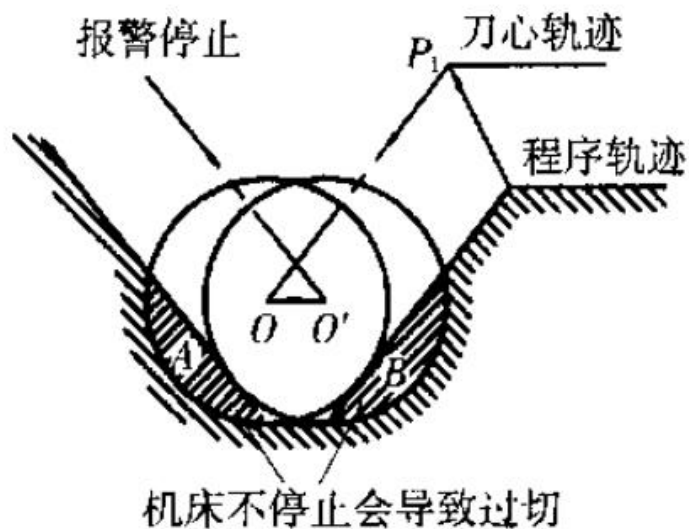


图 18：刀补过切示意图

5.5 刀具半径补偿转接点的计算流程

根据相关的刀具转接点计算方法，可以得到刀补转接点的计算流程如图所示，首先读入三个插补点（两段连续的线段），判断这两段数据是否有刀具半径补偿要求。如果有，判断其刀补状态，根据刀补状态和两相交线段的两个端点，计算转接角并判断刀补类型，最后根据相关公式即可得到这两段的刀补转接点。具体的程序流程如下图 19 所示。

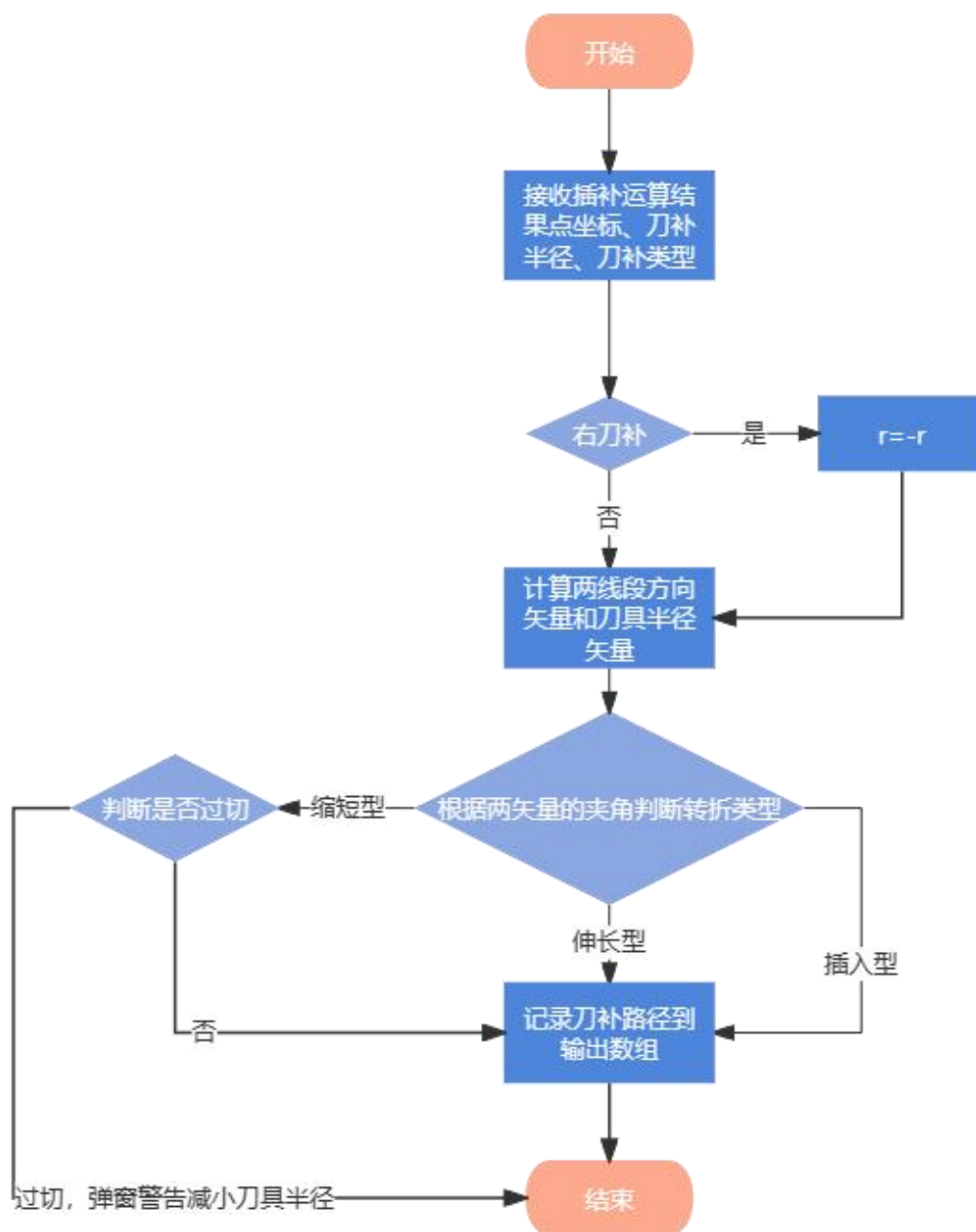


图 19：刀补算法流程图

六、仿真加工效果演示

6.1 心脏线刀补显示

如图 20 所示，当在曲线设置页面选择心脏线并设置完曲线参数后，进入仿真加工页面后默认走刀方向为顺时针方向，刀补方式为左刀补，并给定默认加工参数，点击“生成仿真加工动画”按钮在页面右边绘图区域会呈现出仿真加工的动画。红色为编程轨迹，蓝色为刀具中心轨迹。

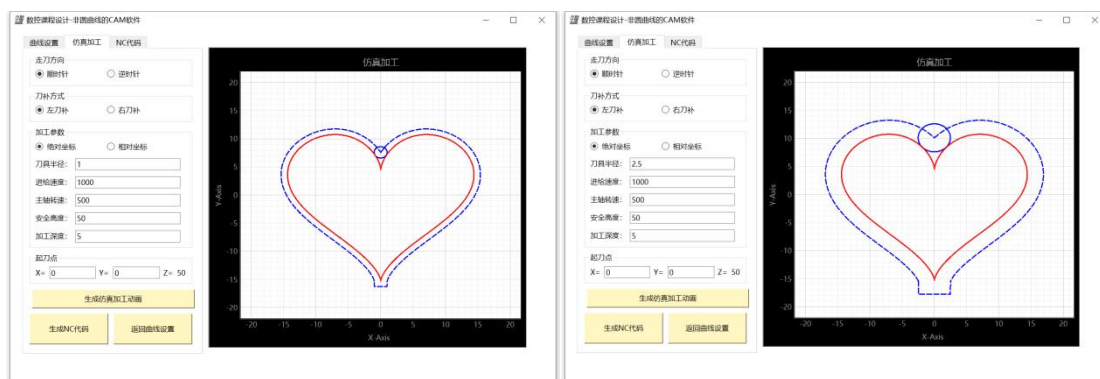


图 20：心脏线左刀补动画显示



图 21：心脏线右刀补动画显示

6.2 过切报警

如图 22 心脏线过切警告显示所示，由于心脏线的形状较为复杂和特殊，其左刀补和右刀补在有形的刀具下都无法加工出来，即都有过切问题，因此该软件会提醒用户过切问题，并自动对加工曲线进行近似处理。



图 22：心脏线过切警告显示

6.3 正弦加减速盘形凸轮廓线

如图 23 所示，当在曲线设置页面选择正弦加减速盘形凸轮廓线并设置完曲线参数后，进入仿真加工页面后默认走刀方向为顺时针方向，刀补方式为左刀补，并给定默认加工参数，点击“生成仿真加工动画”按钮在页面右边绘图区域会呈现出仿真加工的动画。红色为编程轨迹，蓝色为刀具中心轨迹。

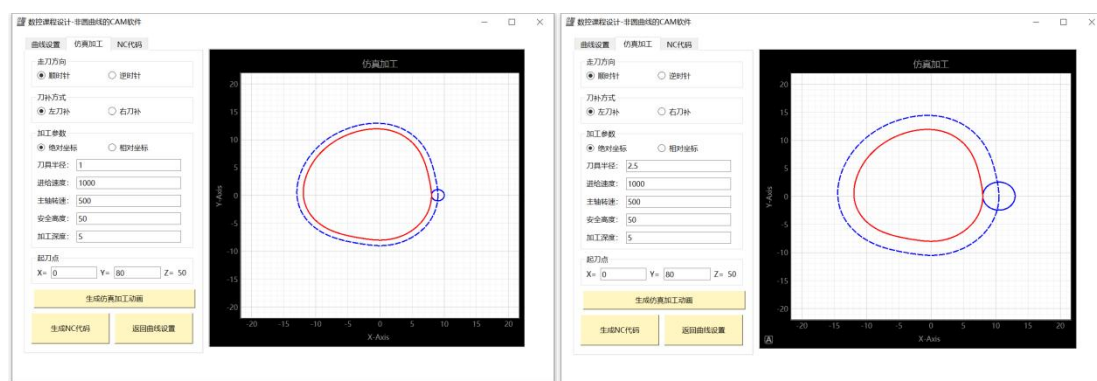


图 23：正弦加减速盘形凸轮廓线左刀补动画显示

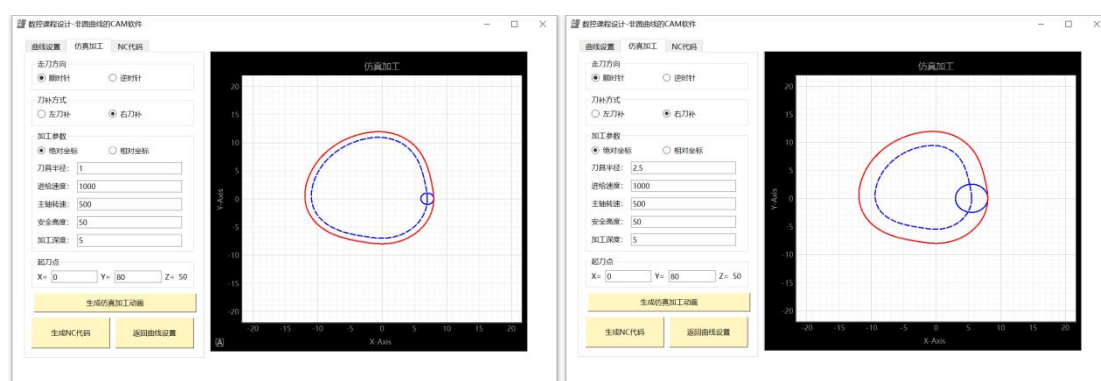


图 24：正弦加减速盘形凸轮廓线右刀补动画显示

6.4 过切报警

如图 25 正弦加减速盘形凸轮廓线过切警告显示所示，当输入的刀补半径过大而导致过切时就会弹出过切警告窗口，提示减小刀具半径。

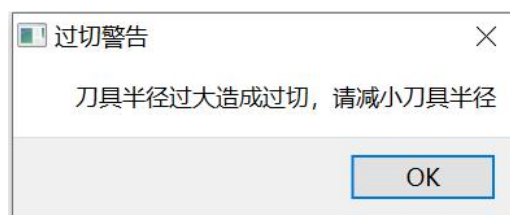


图 25：正弦加减速盘形凸轮廓线过切警告显示

七、设计总结

经过本次数控技术课程设计的动手实践，我们对于数控技术有了更深层次地了解和掌握。在完成本次课程设计的时候，我们选择用 PyQt6 这个基于 Python 的 GUI 库来完成我们的非圆曲线 CAM 软件设计，在完成了本次项目之后，我们对于开发 GUI 软件界面有了更丰富的经验。起初我们面对这个课题的时候是有些手足无措的，但我们小组不断快速学习新知识，面对困难不断探索解决方法，经过了两周的摸索我们尽全力地完成了指定的任务，并不断优化软件的可用性和易用性。回望这次项目经历，我们感觉非常充实并且硕果累累，我们每天都能取得新的进展和收获新的知识。这次课程设计包括前期相关资料地收集、软件设计与编程、后期程序调试以及项目报告的撰写，每一个步骤都与我们数控技术的理论课紧密相连，我们真正领悟到我们所学的理论知识是转化为落地实践的基石。在这个过程中，我们小组分工明确，以求最高效率来完成任务。

程序编写过程中难免会出现一些问题，有时候一些看似简单的 BUG 却将我们小组的进度卡住许久。不过得益于我们小组分工合作，集思广益，最终完成的软件实现了课题要求的功能：1. 可选择心脏线或正弦加减速盘形凸轮廓线并自由设置对应参数同时也能够选择不同的插补算法来完成轮廓曲线的可视化；2. 实现了仿真加工动画功能，并可根据走刀方向、刀补方式、刀具半径、进给速度等来调试仿真加工动画，即生成刀具中心轨迹，如果生成的刀具中心轨迹有过切问题，则会自动进行过切预警；3. 根据设置的加工参数可生成标准格式的 NC 代码。

总的来说，我们小组基本完成了本次课程设计要求的任务。同时，我们认为该软件仍有较大的改进空间，未来我们可以为我们的软件提供更多样化的加工曲线选择。另一方面，我们目前的加工仿真功能是 2D 的动画，我们考虑到实际加工中可能会遇到许多 3D 的情况，因此可以开发一个 3D 仿真加工动画的功能，为用户提供更直观地可视化功能。这次课程设计我们学习到非常多的动手实践知识，除此之外更为弥足珍贵的是培养了我们的团队协作能力和合作意识，在实际工作中，团队作战是一个不可避免的事情，我们这次非常棒的合作经历将为我们未来的发展奠定一个良好的基础。

参考文献

- [1]陶维利.等间距法直线逼近非圆曲线的误差控制[J].泰州职业技术学院学报,2007,(04):5-7.
- [2]周慧.非圆曲线的等误差直线逼近[J].现代制造工程,2002,(03):29-31.
- [3]郝树萌,王士军,崔林.圆管相贯线接缝优化的等间距直线逼近算法的研究[J].制造技术与机床,2017,(12):68-72.
- [4]倪春杰,姚振强,张立文.用等间距法直线逼近非圆曲线[J].机械设计与研究,2010,26(05):17-19.
- [5]周冠兴,王子牛.基于单位矢量的刀具半径补偿算法研究[J].贵州大学学报(自然科学版),2012,29(6):82-85,90.
- [6]唐和业,梅江平.由"刀具半径补偿"引起的过切现象[J].机械工程师,2005(12):76-77.

附录

1. Project_Main.py

```
1. import pyqtgraph as pg
2. from PyQt6.QtWidgets import (
3.     QApplication, QMainWindow, QMessageBox, QGraphicsView, QGraphicsScene, QGraphicsProxyWidget,
4.     QWidget, QVBoxLayout, QLabel
5. )
6. from PyQt6.QtGui import QIcon, QPixmap
7. from PyQt6.QtCore import Qt, QTimer
8.
9. from Project import Ui_NC_Project
10. import sys
11. import warnings
12. import math
13. import numpy as np
14. from matplotlib.figure import Figure
15. from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
16. from matplotlib import rcParams
17.
18. from daobu_chabu_dengjianjv import plot_heart_curve_with_tool_path
19. from daobu_chabu_dengjianjv_sin import offset_curve
20. from generate_nccode_v4 import generate_nc_code
21.
22. warnings.filterwarnings("ignore",category=DeprecationWarning)
23.
24. # 在创建图形之前设置字体
25. rcParams['font.family'] = 'SimHei' # 例如, 使用 SimHei 字体支持中文
26. rcParams['axes.unicode_minus'] = False # 解决负号显示问题
27.
28. class CustomGraphicsView(QGraphicsView):
29.     def __init__(self, parent=None):
30.         super().__init__(parent)
31.         self.scene = QGraphicsScene(self)
32.         self.setScene(self.scene)
33.         self.setHorizontalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOff)
34.         self.setVerticalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOff)
35.
36. class My_NC_Project(Ui_NC_Project, QMainWindow):
37.     def __init__(self):
38.         super().__init__()
39.         self.setupUi(self)
40.         self.setWindowIcon(QIcon('LOGO.JPG'))
41.         self.radioButton_xinzang.toggled.connect(self.xinzang)
```

```

42.         self.radioButton_tulun.toggled.connect(self.tulun)
43.         self.radioButton_clockwise.toggled.connect(self.clockwise)
44.         self.radioButton_anticlockwise.toggled.connect(self.anticlockwise)
45.         self.radioButton_zuodaobu.toggled.connect(self.zuodaobu)
46.         self.radioButton_youdaobu.toggled.connect(self.youdaobu)
47.         self.pushButton_start.clicked.connect(self.start)
48.         self.pushButton_close.clicked.connect(self.close)
49.         self.pushButton_about.clicked.connect(self.about)
50.         self.pushButton_fangzhen.clicked.connect(self.fangzhen)
51.         self.pushButton_back.clicked.connect(self.back)
52.         self.pushButton_NC_code.clicked.connect(self.display_NC_code)
53.         self.pushButton_back2.clicked.connect(self.back2)
54.         self.pushButton_jiagongdonghua.clicked.connect(self.jiagongdonghua)
55.         self.customGraphicsView = CustomGraphicsView(self.centralwidget)
56.         self.customGraphicsView.setGeometry(self.graphicsView.geometry())
57.         self.comboBox_chabu.currentIndexChanged.connect(self.start)
58.         self.scene = self.customGraphicsView.scene
59.         self.graphicsView.deleteLater()
60.         self.anquan.textChanged.connect(self.anquan_Z)
61.         #初始化
62.         self.radioButton_xinzang.setChecked(True)
63.         self.tabWidget.setCurrentIndex(0)
64.         self.NC_code.setWidgetResizable(True)
65.         self.scroll_content = QWidget()
66.         self.layout = QVBoxLayout(self.scroll_content)
67.         self.NC_code.setWidget(self.scroll_content)
68.         self.show()
69.
70.     def xinzang(self):
71.         if self.radioButton_xinzang.isChecked():
72.             self.groupBox_xinzang.setVisible(True)
73.             self.groupBox_tulun.setVisible(False)
74.         else:
75.             return
76.
77.         pixmap = QPixmap("formula1.jpg") # 将 "path/to/your/image.png" 替换为你的图片路径
78.         scaled_pixmap = pixmap.scaled(pixmap.width() // 2.6 , pixmap.height() // 1.5 )
79.         self.Function_Equation.setPixmap(scaled_pixmap)
80.         self.Function_Equation.resize(scaled_pixmap.width(), scaled_pixmap.height())# 调整 QLabel 的大小以适应图片
81.         if not self.Parameter_Function.text():
82.             self.Parameter_Function.setText("0.9")
83.         if not self.lineEdit_wucha.text():
84.             self.lineEdit_wucha.setText("0.1")
85.         self.a = float(self.Parameter_Function.text()) # 获取参数值

```

```

86.         error = float(self.lineEdit_wucha.text())
87.         scale = 10
88.         if self.comboBox_chabu.currentIndex() == 0:
89.             # 设定参数范围和初始插补点数、最大距离阈值
90.             t_start, t_end = 0, 2 * np.pi
91.             initial_segments = 5 # 初始插补点数, 不固定
92.             max_distance = error # 最大距离阈值, 根据实际需求设定
93.             # 进行等间距插补
94.             self.t_values, self.x_values, self.y_values, self.num_segments = self.adaptive_equal_spacing_interpolation\
            (t_start, t_end, initial_segments, max_distance)
95.             self.plot_figure(self.x_values, self.y_values, scale)
96.             self.lineEdit_duanshu.setText(f"{self.num_segments}")
97.             if self.comboBox_chabu.currentIndex() == 1:
98.                 # 设定参数范围和误差阈值
99.                 t_start, t_end = 0, 2 * np.pi
100.                 error_threshold = float(self.lineEdit_wucha.text()) # 误差阈值, 根据实际需求设定
101.                 # 进行等误差插补
102.                 self.t_values, self.x_values, self.y_values, self.num_segments = self.equal_error_interpolation(t_start, t_end, error_threshold)
103.                 self.plot_figure(self.x_values, self.y_values, scale)
104.                 self.lineEdit_duanshu.setText(f"{self.num_segments}")
105.
106.
107.
108.     def tulun(self):
109.         if self.radioButton_tulun.isChecked():
110.             self.groupBox_xinzang.setVisible(False)
111.             self.groupBox_tulun.setVisible(True)
112.         else:
113.             return
114.         if not self.lineEdit_base_radius.text():
115.             self.lineEdit_base_radius.setText("8")
116.             self.lineEdit_hight.setText("4")
117.             self.lineEdit_far_angle.setText("60")
118.             self.lineEdit_near_angle.setText("90")
119.         if not self.lineEdit_wucha.text():
120.             self.lineEdit_wucha.setText("0.1")
121.
122.         #####起刀点设置
123.         self.label_X.setText("0")
124.         self.label_Y.setText("80")
125.
126.         self.base_radius = float(self.lineEdit_base_radius.text()) # 获取参数值
127.         self.hight = float(self.lineEdit_hight.text())

```



```

128.         self.far_angle = float(self.lineEdit_far_angle.text())
129.         self.near_angle = float(self.lineEdit_near_angle.text())
130.         self.far_angle = self.far_angle * math.pi / 180.0
131.         self.near_angle = self.near_angle * math.pi / 180.0
132.         self.to_angle = (math.pi * 2 - self.near_angle - self.far_angle) / 2
133.         self.return_angle = self.to_angle
134.
135.         error = float(self.lineEdit_wucha.text())
136.         scale = 10
137.         if self.comboBox_chabu.currentIndex() == 0:
138.             # 设定参数范围和初始插补点数、最大距离阈值
139.             t_start, t_end = 0, 2 * np.pi
140.             initial_segments = 10 # 初始插补点数, 不固定
141.             max_distance = error # 最大距离阈值, 根据实际需求设定
142.             # 进行等间距插补
143.             self.t_values, self.x_values, self.y_values, self.num_segments = self.adaptive_equal_spacing_interpolation \
144.                 (t_start, t_end, initial_segments, max_distance)
145.             self.plot_figure(self.x_values, self.y_values, scale)
146.             self.lineEdit_duanshu.setText(f"{self.num_segments}")
147.         if self.comboBox_chabu.currentIndex() == 1:
148.             # 设定参数范围和误差阈值
149.             t_start, t_end = 0, 2 * np.pi
150.             error_threshold = float(self.lineEdit_wucha.text()) # 误差阈值, 根据实际需求设定
151.             # 进行等误差插补
152.             self.t_values, self.x_values, self.y_values, self.num_segments = self.equal_error_interpolation(t_start, t_end, error_threshold)
153.             self.plot_figure(self.x_values, self.y_values, scale)
154.             self.lineEdit_duanshu.setText(f"{self.num_segments}")
155.         def sinline_function(self, angle):
156.             if angle < self.to_angle:
157.                 return self.hight * ((angle / self.to_angle) -
158.                     math.sin(2.0 * math.pi * angle / self.to_angle) / 2.0 / math.pi) + self.base_radius
159.             if self.to_angle <= angle < self.to_angle + self.far_angle:
160.                 return self.base_radius + self.hight
161.             if self.to_angle + self.far_angle <= angle < self.to_angle + self.far_angle + self.return_angle:
162.                 angle = angle - self.to_angle - self.far_angle
163.                 return self.hight * (1 - (angle / self.return_angle) + math.sin(
164.                     2.0 * math.pi * angle / self.to_angle) / 2.0 / math.pi) + self.base_radius
165.             else:
166.                 return self.base_radius
167.
168.         def sinline(self, angle):
169.             x = self.sinline_function(angle) * math.cos(angle)

```

```

170.         y = self.sinline_function(angle) * math.sin(angle)
171.         return x, y
172.
173.     def fangzhen(self):
174.         self.tabWidget.setCurrentIndex(1)
175.         if not self.radioButton_clockwise.isChecked():
176.             self.radioButton_clockwise.setChecked(True)
177.             self.radioButton_zuodaobu.setChecked(True)
178.         if not self.radioButton_juedui.isChecked():
179.             self.daoju.setText("1")
180.             self.jingei.setText("1000")
181.             self.zhuzhou.setText("500")
182.             self.anquan.setText("50")
183.             self.shendu.setText("5")
184.             #####起刀点设置
185.             self.label_X.setText("0")
186.             self.label_Y.setText("0")
187.             self.radioButton_juedui.setChecked(True)
188.             self.jiagongdonghua()
189.
190.     def back(self):
191.         self.tabWidget.setCurrentIndex(0)
192.
193.     def back2(self):
194.         self.tabWidget.setCurrentIndex(1)
195.
196.     def start(self):
197.         if self.radioButton_xinzang.isChecked():
198.             self.a = float(self.Parameter_Function.text()) # 获取参数值
199.             scale = 10
200.             if self.comboBox_chabu.currentIndex() == 0:
201.                 # 设定参数范围和初始插补点数、最大距离阈值
202.                 t_start, t_end = 0, 2 * np.pi
203.                 initial_segments = 10 # 初始插补点数, 不固定
204.                 error = float(self.lineEdit_wucha.text())
205.                 max_distance = error # 最大距离阈值, 根据实际需求设定
206.                 # 进行等间距插补
207.                 self.t_values, self.x_values, self.y_values, self.num_segments = self.adaptive_equal_spacing_interpolation \
208.                     (t_start, t_end, initial_segments, max_distance)
209.                 self.plot_figure(self.x_values, self.y_values, scale)
210.                 self.lineEdit_duanshu.setText(f"{self.num_segments}")
211.             else:
212.                 # 设定参数范围和误差阈值

```

```

213.         t_start, t_end = 0, 2 * np.pi
214.         error_threshold = float(self.lineEdit_wucha.text()) # 误差阈值, 根据实际需求设定
215.
216.         # 进行等误差插补
217.         self.t_values, self.x_values, self.y_values, self.num_segments = self.equal_error_interpolation(t_
            start, t_end, error_threshold)
218.         self.plot_figure(self.x_values, self.y_values, scale)
219.         self.lineEdit_duanshu.setText(f"{self.num_segments}")
220.     else:
221.         self.base_radius = float(self.lineEdit_base_radius.text()) # 获取参数值
222.         self.hight = float(self.lineEdit_hight.text())
223.         self.far_angle = float(self.lineEdit_far_angle.text())
224.         self.near_angle = float(self.lineEdit_near_angle.text())
225.         self.far_angle = self.far_angle * math.pi / 180.0
226.         self.near_angle = self.near_angle * math.pi / 180.0
227.         self.to_angle = (math.pi * 2 - self.near_angle - self.far_angle) / 2
228.         self.return_angle = self.to_angle
229.
230.         error = float(self.lineEdit_wucha.text())
231.         scale = 10
232.
233.         if self.comboBox_chabu.currentIndex() == 0:
234.             # 设定参数范围和初始插补点数、最大距离阈值
235.             t_start, t_end = 0, 2 * np.pi
236.             initial_segments = 10 # 初始插补点数, 不固定
237.             max_distance = error # 最大距离阈值, 根据实际需求设定
238.             # 进行等间距插补
239.             self.t_values, self.x_values, self.y_values, self.num_segments = self.adaptive_equal_spacing_inter
                polation \
                (t_start, t_end, initial_segments, max_distance)
240.             self.plot_figure(self.x_values, self.y_values, scale)
241.             self.lineEdit_duanshu.setText(f"{self.num_segments}")
242.
243.         if self.comboBox_chabu.currentIndex() == 1:
244.             # 设定参数范围和误差阈值
245.             t_start, t_end = 0, 2 * np.pi
246.             error_threshold = float(self.lineEdit_wucha.text()) # 误差阈值, 根据实际需求设定
247.             # 进行等误差插补
248.             self.t_values, self.x_values, self.y_values, self.num_segments = self.equal_error_interpolation(t_
                start, t_end, error_threshold,)
249.             self.plot_figure(self.x_values, self.y_values, scale)
250.             self.lineEdit_duanshu.setText(f"{self.num_segments}")
251.
252.
253.     def about(self):

```

```

254.         msg_box = QMessageBox()
255.         msg_box.setWindowTitle("小组信息")
256.         msg_box.setText("小组成员: \nxxx 3xxx\nxxx 3xxx\nxxx 3xxx\n 指导老师: xxx")
257.         msg_box.exec()
258.
259.     def close(self):
260.         QApplication.quit() # 退出应用程序
261.
262.     def display_NC_code(self):
263.         self.tabWidget.setCurrentIndex(2)
264.         self.generate_NC_code()
265.
266.     def generate_NC_code(self):
267.         if self.layout is not None:
268.             while self.layout.count():
269.                 child = self.layout.takeAt(0)
270.                 if child.widget() is not None:
271.                     child.widget().deleteLater()
272.                 tool_radius = float(self.daoju.text())
273.                 feed_rate = float(self.jingei.text())
274.                 spindle_speed = float(self.zhuzhou.text())
275.                 safety_height = float(self.anquan.text())
276.                 shendu = float(self.shendu.text())
277.                 start_point = (float(self.label_X.text()),float(self.label_Y.text()))
278.                 if self.radioButton_zuodaobu.isChecked():
279.                     compensation_direction = 'left'
280.                 else:
281.                     compensation_direction = 'right'
282.                 if self.radioButton_juedui.isChecked():
283.                     use_relative_coordinates = False
284.                 else:
285.                     use_relative_coordinates = True
286.                 nc_code = generate_nc_code(self.t_values, self.x_values, self.y_values, tool_radius=tool_radius, feed_rate
=feed_rate, spindle_speed=spindle_speed,
287.                 safety_height=safety_height, cut_depth=shendu, start_point=start_point, compensation_directio
n=compensation_direction,
288.                 use_relative_coordinates=use_relative_coordinates)
289.                 # 将数控代码添加到内容部件
290.                 for line in nc_code:
291.                     label = QLabel(line)
292.                     self.layout.addWidget(label)
293.
294.     def plot_figure(self,x_vals, y_vals,scale):
295.         # 清空场景

```

```

296.         self.customGraphicsView.scene.clear()
297.         # 创建图形
298.         fig = Figure(figsize=(5, 5), dpi=100)
299.         ax = fig.add_subplot(111)
300.         if self.comboBox_chabu.currentIndex()==0:
301.             ax.plot(x_vals, y_vals, color='red')
302.         else:
303.             ax.plot(x_vals, y_vals, color='blue')
304.         ax.axis('equal')
305.         # 将原点设置在中心, 绘制十字形坐标轴
306.         ax.spines['left'].set_position('center')
307.         ax.spines['bottom'].set_position('center')
308.         ax.spines['right'].set_color('none')
309.         ax.spines['top'].set_color('none')
310.         ax.xaxis.set_ticks_position('bottom')
311.         ax.yaxis.set_ticks_position('left')
312.         # 手动设置 x 轴和 y 轴的刻度
313.         x_ticks = list(range(-15, 16, 5)) # 设置 x 轴刻度
314.         y_ticks = list(range(-15, 16, 5)) # 设置 y 轴刻度
315.         ax.set_xticks(x_ticks)
316.         ax.set_yticks(y_ticks)
317.         # 设置每个刻度都有标签
318.         ax.set_xticklabels([str(tick) for tick in x_ticks])
319.         ax.set_yticklabels([str(tick) for tick in y_ticks])
320.         # 设置 x 轴和 y 轴的范围
321.         ax.set_xlim(-scale * 1.8, scale * 1.8)
322.         ax.set_ylim(-scale * 1.8, scale * 1.8)
323.         # 隐藏 Y 轴上的 "0" 标签
324.         yticks = ax.get_yticks()
325.         ytick_labels = ['' if y == 0 else str(y) for y in yticks]
326.         ax.set_yticks(yticks)
327.         ax.set_yticklabels(ytick_labels)
328.         # 隐藏 X 轴上的 "0" 标签
329.         xticks = ax.get_xticks()
330.         xtick_labels = ['' if x == 0 else str(x) for x in xticks]
331.         ax.set_xticks(xticks)
332.         ax.set_xticklabels(xtick_labels)
333.         # 设置 x 轴上的 0 与坐标轴错开
334.         ax.annotate('0', xy=(0, 0), xytext=(-1, -1.5))
335.         # 添加箭头
336.         ax.annotate('', xy=(18, 0), xytext=(17.9, 0),
337.                     arrowprops=dict(arrowstyle="->", lw=1.5, color='black'))
338.         ax.annotate('', xy=(0, 17.8), xytext=(0, 17.7),
339.                     arrowprops=dict(arrowstyle="->", lw=1.5, color='black'))

```

```

340.         # 添加坐标轴标签
341.         ax.text(0.9, 0.40, 'X 轴', transform=ax.transAxes)
342.         ax.text(0.52, 0.94, 'Y 轴', transform=ax.transAxes)
343.         # 将图形添加到 QGraphicsView 中
344.         canvas = FigureCanvas(fig)
345.         proxy_widget = QGraphicsProxyWidget()
346.         proxy_widget.setWidget(canvas)
347.         self.customGraphicsView.scene.addItem(proxy_widget)
348.
349.     def plot_xinzang(self, t):
350.         x = self.a * (16 * math.sin(t) ** 3)
351.         y = self.a * (13 * math.cos(t) - 5 * math.cos(2 * t) - 2 * math.cos(3 * t) - math.cos(4 * t))
352.         return x,y
353.
354.     # 等间距插补法
355.     def adaptive_equal_spacing_interpolation(self,t_start, t_end, initial_segments, max_distance,):
356.         segments = initial_segments
357.         while True:
358.             # 在参数 t 上进行等间距插补
359.             t_values = np.linspace(t_start, t_end, segments)
360.             if self.radioButton_xinzang.isChecked():
361.                 x_values, y_values = zip(*[self.plot_xinzang(t) for t in t_values])
362.             else:
363.                 x_values, y_values = zip(*[self.sinline(t) for t in t_values])
364.             # 计算相邻点之间的距离
365.             distances = np.sqrt(np.diff(x_values) ** 2 + np.diff(y_values) ** 2)
366.             # 检查最大距离是否满足条件
367.             if np.max(distances) <= max_distance or segments >= 3000: # 假设最大插补点数为 1000
368.                 break
369.             # 如果不满足条件, 则增加插补点的数量
370.             segments *= 2
371.             # 添加起始点到插补点
372.             t_values = np.concatenate(([t_start], t_values))
373.             if self.radioButton_xinzang.isChecked():
374.                 x_values = np.concatenate([self.plot_xinzang(t_start)[0]], x_values)
375.                 y_values = np.concatenate([self.plot_xinzang(t_start)[1]], y_values)
376.             else:
377.                 x_values = np.concatenate([self.sinline(t_start)[0]], x_values)
378.                 y_values = np.concatenate([self.sinline(t_start)[1]], y_values)
379.             return t_values, x_values, y_values, segments - 1 # 返回线段数 (即插补点数量减一)
380.
381.     # 等误差插补法
382.     def equal_error_interpolation(self, t_start, t_end, error_threshold):
383.         if self.radioButton_xinzang.isChecked():

```

```

384.         points = [(t_start, self.plot_xinzang(t_start))]
385.     else:
386.         points = [(t_start, self.sinline(t_start))]
387.     t = t_start
388.     while t < t_end:
389.         h = 0.01 # 初始步长, 可以根据需要调整
390.         while True:
391.             t_next = t + h
392.             if t_next > t_end:
393.                 t_next = t_end
394.             if self.radioButton_xinzang.isChecked():
395.                 x_next, y_next = self.plot_xinzang(t_next)
396.             else:
397.                 x_next, y_next = self.sinline(t_next)
398.             chord = self.chord_length(t, t_next)
399.             # 如果弦长误差小于阈值, 则接受这个步长
400.             if chord <= error_threshold:
401.                 break
402.             # 否则减小步长并重新尝试
403.             h /= 2
404.             # 添加新点到插补结果中
405.             points.append((t_next, (x_next, y_next)))
406.             t = t_next
407.             # 提取插补点的参数和坐标
408.             t_values = [p[0] for p in points]
409.             x_values, y_values = zip(*[p[1] for p in points])
410.             # 返回插补结果和线段数 (即插补点数量减一)
411.             return t_values, x_values, y_values, len(points) - 1
412.
413. # 计算两点之间的弦长
414. def chord_length(self, t1, t2, n=100):
415.     t_values = np.linspace(t1, t2, n)
416.     if self.radioButton_xinzang.isChecked():
417.         x_values, y_values = zip(*[self.plot_xinzang(t=t) for t in t_values])
418.     else:
419.         x_values, y_values = zip(*[self.sinline(t) for t in t_values])
420.     dx = np.diff(x_values)
421.     dy = np.diff(y_values)
422.     return np.sum(np.sqrt(dx ** 2 + dy ** 2))
423.
424. def plot_animation(self):
425.     # 清空场景
426.     self.customGraphicsView.scene.clear()
427.

```

```

428.         # 创建 pyqtgraph 的 GraphicsLayoutWidget
429.         self.plot_widget = pg.GraphicsLayoutWidget()
430.
431.         # 创建一个 plot
432.         self.plot = self.plot_widget.addPlot(title="仿真加工")
433.         self.plot.setXRange(-20,20)
434.         self.plot.setYRange(-20, 20)
435.         self.plot.showGrid(x=True, y=True)
436.
437.         # 设置坐标轴标签
438.         self.plot.setLabel('left', 'Y-Axis')
439.         self.plot.setLabel('bottom', 'X-Axis')
440.
441.         # 设置坐标轴标签和刻度
442.         self.plot.getAxis('left').setStyle(showValues=True)
443.         self.plot.getAxis('bottom').setStyle(showValues=True)
444.         self.plot.setLabel('left', 'Y-Axis')
445.         self.plot.setLabel('bottom', 'X-Axis')
446.
447.         # 将 GraphicsLayoutWidget 包装成 QGraphicsProxyWidget 并添加到场景
448.         self.proxy = QGraphicsProxyWidget()
449.         self.proxy.setWidget(self.plot_widget)
450.         self.customGraphicsView.scene.addItem(self.proxy)
451.
452.         # 设置背景颜色为白色
453.         self.plot.getViewBox().setBackgroundColor('w')
454.
455.         # 调整 plot_widget 的大小和位置，使其与 QGraphicsView 的场景匹配
456.         self.proxy.resize(self.customGraphicsView.width(), self.customGraphicsView.height())
457.
458.         # 将 self.proxy 的位置设定为 self.customGraphicsView 的中心位置
459.         view_center = self.customGraphicsView.viewport().rect().center()
460.         self.proxy.setPos(view_center.x() - self.proxy.boundingRect().width() / 2,
461.                             view_center.y() - self.proxy.boundingRect().height() / 2)
462.
463.         ## 初始化数据
464.         # 初始化动态数据线（已存在）
465.         self.data_line_moving = self.plot.plot([], [], pen=pg.mkPen(color='b', width=3, style=Qt.PenStyle.DashLine)
466.         )
467.         self.data_line_cycle = self.plot.plot([], [], pen=pg.mkPen(color='b', width=3))
468.         # 新增初始化静态数据线
469.         self.data_line_static = self.plot.plot([], [], pen=pg.mkPen(color='r', width=3)) # 红色代表静态曲线
470.         self.current_index = 0
471.         self.continue_updates = True

```



```

471.
472.     # 创建一个定时器
473.     self.timer = QTimer(self)
474.     self.timer.timeout.connect(self.update_plot)
475.     if self.jingei.text() == '':
476.         time = 1000
477.     else:
478.         time = float(self.jingei.text())
479.     self.timer.start(1000/time) # 每 xx 毫秒更新一次
480.     # 心脏线过切报警
481.     if self.radioButton_xinzang.isChecked():
482.         if not (self.daoju.text() == ''):
483.             self.guoqieyujing()
484.
485.     def update_plot(self):
486.         if self.current_index >= len(self.x_values):
487.             self.timer.stop()
488.         else:
489.             tool_radius = float(self.daoju.text())
490.             angle = np.linspace(0, 2 * np.pi, 100)
491.             if self.radioButton_clockwise.isChecked():
492.                 if self.radioButton_xinzang.isChecked():
493.                     x_data = self.offset_x_array[self.current_index + 1]
494.                     y_data = self.offset_y_array[self.current_index + 1]
495.                 else:
496.                     x_data = self.offset_x_array[-(self.current_index + 1)][::-1]
497.                     y_data = self.offset_y_array[-(self.current_index + 1)][::-1]
498.             else:
499.                 if self.radioButton_xinzang.isChecked():
500.                     x_data = self.offset_x_array[-(self.current_index + 1)][::-1]
501.                     y_data = self.offset_y_array[-(self.current_index + 1)][::-1]
502.                 else:
503.                     x_data = self.offset_x_array[self.current_index + 1]
504.                     y_data = self.offset_y_array[self.current_index + 1]
505.             self.data_line_static.setData(self.x_values, self.y_values)
506.             self.data_line_moving.setData(x_data, y_data)
507.             # 刀具圆
508.             if self.current_index < len(self.offset_x_array) and self.current_index < len(self.offset_y_array):
509.                 tool_radius = float(self.daoju.text())
510.                 angle = np.linspace(0, 2 * np.pi, 100)
511.                 if self.radioButton_clockwise.isChecked():
512.                     if self.radioButton_xinzang.isChecked():
513.                         x_circle = self.offset_x_array[self.current_index] + tool_radius * np.cos(angle)
514.                         y_circle = self.offset_y_array[self.current_index] + tool_radius * np.sin(angle)

```

```

515.         else:
516.             x_circle = self.offset_x_array[-(self.current_index)] + tool_radius * np.cos(angle)
517.             y_circle = self.offset_y_array[-(self.current_index)] + tool_radius * np.sin(angle)
518.         else:
519.             if self.radioButton_xinzang.isChecked():
520.                 x_circle = self.offset_x_array[-(self.current_index)] + tool_radius * np.cos(angle)
521.                 y_circle = self.offset_y_array[-(self.current_index)] + tool_radius * np.sin(angle)
522.             else:
523.                 x_circle = self.offset_x_array[self.current_index] + tool_radius * np.cos(angle)
524.                 y_circle = self.offset_y_array[self.current_index] + tool_radius * np.sin(angle)
525.             self.data_line_cycle.setData(x_circle, y_circle)
526.             self.current_index += 1
527.
528.     def clockwise(self):
529.         if not self.radioButton_clockwise.isChecked():
530.             return
531.         if (self.daoju.text()==''):
532.             return
533.         else:
534.             if self.daoju.text() == '':
535.                 tool_radius = 1
536.             else:
537.                 tool_radius = float(self.daoju.text())
538.             if self.radioButton_xinzang.isChecked():
539.                 if self.radioButton_zuodaobu.isChecked():
540.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
y_values, tool_radius, 'left')
541.                 else:
542.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
y_values, tool_radius, 'right')
543.             else:
544.                 base_radius = float(self.lineEdit_base_radius.text())
545.                 if self.radioButton_zuodaobu.isChecked():
546.                     self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
ius, base_radius, 'right')
547.                 if (self.offset_x_array == self.offset_y_array) :
548.                     return
549.                 else:
550.                     self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
ius, base_radius, 'left')
551.                 if (self.offset_x_array == self.offset_y_array) :
552.                     return
553.                 self.plot_animation()
554.

```

```

555.     def anticlockwise(self):
556.         if not self.radioButton_anticlockwise.isChecked():
557.             return
558.         else:
559.             if self.daoju.text() == '':
560.                 tool_radius = 1
561.             else:
562.                 tool_radius = float(self.daoju.text())
563.             if self.radioButton_xinzang.isChecked():
564.                 if self.radioButton_zuodaobu.isChecked():
565.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
y_values, tool_radius, 'right')
566.                 else:
567.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
y_values, tool_radius, 'left')
568.             else:
569.                 base_radius = float(self.lineEdit_base_radius.text())
570.                 if self.radioButton_zuodaobu.isChecked():
571.                     self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
ius, base_radius, 'left')
572.                 if (self.offset_x_array == self.offset_y_array) :
573.                     return
574.                 else:
575.                     self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
ius, base_radius, 'right')
576.                 if (self.offset_x_array == self.offset_y_array) :
577.                     return
578.                 self.plot_animation()
579.
580.     def anquan_Z(self):
581.         Z = self.anquan.text()
582.         self.label_Z.setText(f"{Z}")
583.
584.     def zuodaobu(self):
585.         if not self.radioButton_zuodaobu.isChecked():
586.             return
587.         if (self.daoju.text()==''):
588.             return
589.         if self.daoju.text() == '':
590.             tool_radius = 1
591.         else:
592.             tool_radius = float(self.daoju.text())
593.         if self.radioButton_xinzang.isChecked():
594.             if self.radioButton_clockwise.isChecked():

```

```

595.         self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.y_v
        alues, tool_radius, 'left')
596.         else:
597.             self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.y_v
        alues, tool_radius, 'right')
598.         else:
599.             base_radius = float(self.lineEdit_base_radius.text())
600.             if self.radioButton_clockwise.isChecked():
601.                 self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_radius,
        base_radius, 'right')
602.             if (self.offset_x_array == self.offset_y_array) :
603.                 return
604.             else:
605.                 self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_radius,
        base_radius, 'left')
606.             if (self.offset_x_array == self.offset_y_array) :
607.                 return
608.             self.plot_animation()
609.
610.     def youdaobu(self):
611.         if not self.radioButton_youdaobu.isChecked():
612.             return
613.         if self.daoju.text() == '':
614.             tool_radius = 0.5
615.         else:
616.             tool_radius = float(self.daoju.text())
617.         if self.radioButton_xinzang.isChecked():
618.             if self.radioButton_clockwise.isChecked():
619.                 self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.y_v
        alues, tool_radius, 'right')
620.             else:
621.                 self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.y_v
        alues, tool_radius, 'left')
622.             self.plot_animation()
623.         else:
624.             base_radius = float(self.lineEdit_base_radius.text())
625.             if self.radioButton_clockwise.isChecked():
626.                 self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_radius,
        base_radius, 'left')
627.             if (self.offset_x_array == self.offset_y_array) :
628.                 return
629.             else:
630.                 self.plot_animation()
631.         else:

```

```

632.         self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_radius,
        base_radius, 'right')
633.         if (self.offset_x_array == self.offset_y_array) :
634.             return
635.         else:
636.             self.plot_animation()
637.
638.     def jiagongdonghua(self):
639.         tool_radius = float(self.daoju.text())
640.         if self.radioButton_xinzang.isChecked():
641.             if self.radioButton_clockwise.isChecked():
642.                 if self.radioButton_youdaobu.isChecked():
643.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
        y_values, tool_radius, 'right')
644.                 if self.radioButton_zuodaobu.isChecked():
645.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
        y_values, tool_radius, 'left')
646.             else:
647.                 if self.radioButton_youdaobu.isChecked():
648.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
        y_values, tool_radius, 'left')
649.                 if self.radioButton_zuodaobu.isChecked():
650.                     self.offset_x_array, self.offset_y_array = plot_heart_curve_with_tool_path(self.x_values, self.
        y_values, tool_radius, 'right')
651.             else:
652.                 base_radius = float(self.lineEdit_base_radius.text())
653.                 if self.radioButton_clockwise.isChecked():
654.                     if self.radioButton_youdaobu.isChecked():
655.                         self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
        ius, base_radius, 'left')
656.                     if (self.offset_x_array == self.offset_y_array) :
657.                         return
658.                     if self.radioButton_zuodaobu.isChecked():
659.                         self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
        ius, base_radius, 'right')
660.                     if (self.offset_x_array == self.offset_y_array) :
661.                         return
662.                     else:
663.                         if self.radioButton_youdaobu.isChecked():
664.                             self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
        ius, base_radius, 'right')
665.                         if (self.offset_x_array == self.offset_y_array) :
666.                             return
667.                         if self.radioButton_zuodaobu.isChecked():

```

```

668.             self.offset_x_array, self.offset_y_array = offset_curve(self.x_values, self.y_values, tool_rad
            ius, base_radius, 'left')
669.             if (self.offset_x_array == self.offset_y_array) :
670.                 return
671.             self.plot_animation()
672.
673.     def guoqieyujing(self):
674.         msg_box = QMessageBox()
675.         msg_box.setWindowTitle("过切警告")
676.         msg_box.setText("心脏线加工会有过切问题，已做近似处理")
677.         msg_box.exec()
678.
679. if __name__ == "__main__":
680.     app = QApplication(sys.argv)
681.     my_NC_Project = My_NC_Project()
682.     sys.exit(app.exec())

```

2. Project.py

```
1.     from PyQt6 import QtCore, QtGui, QtWidgets
2.
3.
4.     class Ui_NC_Project(object):
5.         def setupUi(self, NC_Project):
6.             NC_Project.setObjectName("NC_Project")
7.             NC_Project.resize(909, 591)
8.             NC_Project.setStyleSheet("background:rgb(255, 255, 255)")
9.             self.centralwidget = QtWidgets.QWidget(parent=NC_Project)
10.            self.centralwidget.setObjectName("centralwidget")
11.            self.graphicsView = QtWidgets.QGraphicsView(parent=self.centralwidget)
12.            self.graphicsView.setGeometry(QtCore.QRect(330, 30, 531, 501))
13.            self.graphicsView.setStyleSheet("border-color: rgb(0, 0, 0);")
14.            self.graphicsView.setObjectName("graphicsView")
15.            self.tabWidget = QtWidgets.QTabWidget(parent=self.centralwidget)
16.            self.tabWidget.setGeometry(QtCore.QRect(20, 10, 301, 531))
17.            self.tabWidget.setObjectName("tabWidget")
18.            self.tab = QtWidgets.QWidget()
19.            self.tab.setObjectName("tab")
20.            self.groupBox = QtWidgets.QGroupBox(parent=self.tab)
21.            self.groupBox.setGeometry(QtCore.QRect(10, 10, 271, 49))
22.            self.groupBox.setObjectName("groupBox")
23.            self.layoutWidget = QtWidgets.QWidget(parent=self.groupBox)
24.            self.layoutWidget.setGeometry(QtCore.QRect(10, 20, 241, 22))
25.            self.layoutWidget.setObjectName("layoutWidget")
26.            self.horizontalLayout_5 = QtWidgets.QHBoxLayout(self.layoutWidget)
27.            self.horizontalLayout_5.setContentsMargins(0, 0, 0, 0)
28.            self.horizontalLayout_5.setObjectName("horizontalLayout_5")
29.            self.radioButton_xinzang = QtWidgets.QRadioButton(parent=self.layoutWidget)
30.            self.radioButton_xinzang.setObjectName("radioButton_xinzang")
31.            self.horizontalLayout_5.addWidget(self.radioButton_xinzang)
32.            self.radioButton_tulun = QtWidgets.QRadioButton(parent=self.layoutWidget)
33.            self.radioButton_tulun.setObjectName("radioButton_tulun")
34.            self.horizontalLayout_5.addWidget(self.radioButton_tulun)
35.            self.groupBox_xinzang = QtWidgets.QGroupBox(parent=self.tab)
36.            self.groupBox_xinzang.setGeometry(QtCore.QRect(10, 70, 271, 161))
37.            self.groupBox_xinzang.setObjectName("groupBox_xinzang")
38.            self.layoutWidget1 = QtWidgets.QWidget(parent=self.groupBox_xinzang)
39.            self.layoutWidget1.setGeometry(QtCore.QRect(60, 110, 151, 43))
40.            self.layoutWidget1.setObjectName("layoutWidget1")
41.            self.horizontalLayout_8 = QtWidgets.QHBoxLayout(self.layoutWidget1)
42.            self.horizontalLayout_8.setContentsMargins(0, 0, 0, 0)
43.            self.horizontalLayout_8.setObjectName("horizontalLayout_8")
```

```

44.         self.label = QtWidgets.QLabel(parent=self.layoutWidget1)
45.         font = QtGui.QFont()
46.         font.setPointSize(16)
47.         font.setBold(True)
48.         self.label.setFont(font)
49.         self.label.setObjectName("label")
50.         self.horizontalLayout_8.addWidget(self.label)
51.         self.Parameter_Function = QtWidgets.QLineEdit(parent=self.layoutWidget1)
52.         self.Parameter_Function.setObjectName("Parameter_Function")
53.         self.horizontalLayout_8.addWidget(self.Parameter_Function)
54.         self.Function_Equation = QtWidgets.QLabel(parent=self.groupBox_xinzang)
55.         self.Function_Equation.setGeometry(QtCore.QRect(10, 40, 251, 41))
56.         self.Function_Equation.setText("")
57.         self.Function_Equation.setObjectName("Function_Equation")
58.         self.groupBox_5 = QtWidgets.QGroupBox(parent=self.tab)
59.         self.groupBox_5.setGeometry(QtCore.QRect(10, 240, 271, 131))
60.         self.groupBox_5.setObjectName("groupBox_5")
61.         self.comboBox_chabu = QtWidgets.QComboBox(parent=self.groupBox_5)
62.         self.comboBox_chabu.setGeometry(QtCore.QRect(60, 30, 161, 22))
63.         self.comboBox_chabu.setObjectName("comboBox_chabu")
64.         self.comboBox_chabu.addItem("")
65.         self.comboBox_chabu.addItem("")
66.         self.label_9 = QtWidgets.QLabel(parent=self.groupBox_5)
67.         self.label_9.setGeometry(QtCore.QRect(70, 100, 36, 20))
68.         self.label_9.setObjectName("label_9")
69.         self.lineEdit_wucha = QtWidgets.QLineEdit(parent=self.groupBox_5)
70.         self.lineEdit_wucha.setGeometry(QtCore.QRect(112, 70, 101, 20))
71.         self.lineEdit_wucha.setObjectName("lineEdit_wucha")
72.         self.lineEdit_duanshu = QtWidgets.QLineEdit(parent=self.groupBox_5)
73.         self.lineEdit_duanshu.setGeometry(QtCore.QRect(112, 100, 101, 20))
74.         self.lineEdit_duanshu.setObjectName("lineEdit_duanshu")
75.         self.label_10 = QtWidgets.QLabel(parent=self.groupBox_5)
76.         self.label_10.setGeometry(QtCore.QRect(45, 70, 61, 20))
77.         self.label_10.setObjectName("label_10")
78.         self.pushButton_start = QtWidgets.QPushButton(parent=self.tab)
79.         self.pushButton_start.setGeometry(QtCore.QRect(10, 380, 121, 51))
80.         self.pushButton_start.setStyleSheet("background-color: rgb(255, 246, 193);\n"
81.         "")
82.         self.pushButton_start.setObjectName("pushButton_start")
83.         self.pushButton_close = QtWidgets.QPushButton(parent=self.tab)
84.         self.pushButton_close.setGeometry(QtCore.QRect(10, 440, 121, 51))
85.         self.pushButton_close.setStyleSheet("background-color: rgb(255, 246, 193);\n"
86.         "")
87.         self.pushButton_close.setObjectName("pushButton_close")

```



```

88.         self.pushButton_fangzhen = QtWidgets.QPushButton(parent=self.tab)
89.         self.pushButton_fangzhen.setGeometry(QtCore.QRect(160, 380, 121, 51))
90.         self.pushButton_fangzhen.setStyleSheet("background-color: rgb(255, 246, 193);\n"
91.         "")
92.         self.pushButton_fangzhen.setObjectName("pushButton_fangzhen")
93.         self.pushButton_about = QtWidgets.QPushButton(parent=self.tab)
94.         self.pushButton_about.setGeometry(QtCore.QRect(160, 440, 121, 51))
95.         self.pushButton_about.setStyleSheet("background-color: rgb(255, 246, 193);\n"
96.         "")
97.         self.pushButton_about.setObjectName("pushButton_about")
98.         self.groupBox_tulun = QtWidgets.QGroupBox(parent=self.tab)
99.         self.groupBox_tulun.setGeometry(QtCore.QRect(10, 70, 271, 161))
100.        self.groupBox_tulun.setObjectName("groupBox_tulun")
101.        self.layoutWidget2 = QtWidgets.QWidget(parent=self.groupBox_tulun)
102.        self.layoutWidget2.setGeometry(QtCore.QRect(41, 30, 171, 22))
103.        self.layoutWidget2.setObjectName("layoutWidget2")
104.        self.horizontalLayout_2 = QtWidgets.QHBoxLayout(self.layoutWidget2)
105.        self.horizontalLayout_2.setContentsMargins(0, 0, 0, 0)
106.        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
107.        self.label_31 = QtWidgets.QLabel(parent=self.layoutWidget2)
108.        self.label_31.setObjectName("label_31")
109.        self.horizontalLayout_2.addWidget(self.label_31)
110.        self.lineEdit_base_radius = QtWidgets.QLineEdit(parent=self.layoutWidget2)
111.        self.lineEdit_base_radius.setObjectName("lineEdit_base_radius")
112.        self.horizontalLayout_2.addWidget(self.lineEdit_base_radius)
113.        self.layoutWidget3 = QtWidgets.QWidget(parent=self.groupBox_tulun)
114.        self.layoutWidget3.setGeometry(QtCore.QRect(60, 60, 151, 22))
115.        self.layoutWidget3.setObjectName("layoutWidget3")
116.        self.horizontalLayout_4 = QtWidgets.QHBoxLayout(self.layoutWidget3)
117.        self.horizontalLayout_4.setContentsMargins(0, 0, 0, 0)
118.        self.horizontalLayout_4.setObjectName("horizontalLayout_4")
119.        self.label_34 = QtWidgets.QLabel(parent=self.layoutWidget3)
120.        self.label_34.setObjectName("label_34")
121.        self.horizontalLayout_4.addWidget(self.label_34)
122.        self.lineEdit_hight = QtWidgets.QLineEdit(parent=self.layoutWidget3)
123.        self.lineEdit_hight.setObjectName("lineEdit_hight")
124.        self.horizontalLayout_4.addWidget(self.lineEdit_hight)
125.        self.layoutWidget4 = QtWidgets.QWidget(parent=self.groupBox_tulun)
126.        self.layoutWidget4.setGeometry(QtCore.QRect(40, 100, 171, 22))
127.        self.layoutWidget4.setObjectName("layoutWidget4")
128.        self.horizontalLayout_7 = QtWidgets.QHBoxLayout(self.layoutWidget4)
129.        self.horizontalLayout_7.setContentsMargins(0, 0, 0, 0)
130.        self.horizontalLayout_7.setObjectName("horizontalLayout_7")
131.        self.label_35 = QtWidgets.QLabel(parent=self.layoutWidget4)

```

```

132.         self.label_35.setObjectName("label_35")
133.         self.horizontalLayout_7.addWidget(self.label_35)
134.         self.lineEdit_far_angle = QtWidgets.QLineEdit(parent=self.layoutWidget4)
135.         self.lineEdit_far_angle.setObjectName("lineEdit_far_angle")
136.         self.horizontalLayout_7.addWidget(self.lineEdit_far_angle)
137.         self.layoutWidget5 = QtWidgets.QWidget(parent=self.groupBox_tulun)
138.         self.layoutWidget5.setGeometry(QtCore.QRect(40, 130, 171, 22))
139.         self.layoutWidget5.setObjectName("layoutWidget5")
140.         self.horizontalLayout_13 = QtWidgets.QHBoxLayout(self.layoutWidget5)
141.         self.horizontalLayout_13.setContentsMargins(0, 0, 0, 0)
142.         self.horizontalLayout_13.setObjectName("horizontalLayout_13")
143.         self.label_37 = QtWidgets.QLabel(parent=self.layoutWidget5)
144.         self.label_37.setObjectName("label_37")
145.         self.horizontalLayout_13.addWidget(self.label_37)
146.         self.lineEdit_near_angle = QtWidgets.QLineEdit(parent=self.layoutWidget5)
147.         self.lineEdit_near_angle.setObjectName("lineEdit_near_angle")
148.         self.horizontalLayout_13.addWidget(self.lineEdit_near_angle)
149.         self.tabWidget.addTab(self.tab, "")
150.         self.tab_3 = QtWidgets.QWidget()
151.         self.tab_3.setObjectName("tab_3")
152.         self.groupBox_4 = QtWidgets.QGroupBox(parent=self.tab_3)
153.         self.groupBox_4.setGeometry(QtCore.QRect(10, 130, 271, 201))
154.         self.groupBox_4.setObjectName("groupBox_4")
155.         self.layoutWidget_2 = QtWidgets.QWidget(parent=self.groupBox_4)
156.         self.layoutWidget_2.setGeometry(QtCore.QRect(10, 80, 241, 22))
157.         self.layoutWidget_2.setObjectName("layoutWidget_2")
158.         self.horizontalLayout_3 = QtWidgets.QHBoxLayout(self.layoutWidget_2)
159.         self.horizontalLayout_3.setContentsMargins(0, 0, 0, 0)
160.         self.horizontalLayout_3.setObjectName("horizontalLayout_3")
161.         self.label_6 = QtWidgets.QLabel(parent=self.layoutWidget_2)
162.         self.label_6.setObjectName("label_6")
163.         self.horizontalLayout_3.addWidget(self.label_6)
164.         self.jingei = QtWidgets.QLineEdit(parent=self.layoutWidget_2)
165.         self.jingei.setObjectName("jingei")
166.         self.horizontalLayout_3.addWidget(self.jingei)
167.         self.layoutWidget6 = QtWidgets.QWidget(parent=self.groupBox_4)
168.         self.layoutWidget6.setGeometry(QtCore.QRect(10, 50, 241, 22))
169.         self.layoutWidget6.setObjectName("layoutWidget6")
170.         self.horizontalLayout = QtWidgets.QHBoxLayout(self.layoutWidget6)
171.         self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
172.         self.horizontalLayout.setObjectName("horizontalLayout")
173.         self.label_2 = QtWidgets.QLabel(parent=self.layoutWidget6)
174.         self.label_2.setObjectName("label_2")
175.         self.horizontalLayout.addWidget(self.label_2)

```

```

176.         self.daoju = QtWidgets.QLineEdit(parent=self.layoutWidget6)
177.         self.daoju.setObjectName("daoju")
178.         self.horizontalLayout.addWidget(self.daoju)
179.         self.layoutWidget_3 = QtWidgets.QWidget(parent=self.groupBox_4)
180.         self.layoutWidget_3.setGeometry(QtCore.QRect(10, 110, 241, 22))
181.         self.layoutWidget_3.setObjectName("layoutWidget_3")
182.         self.horizontalLayout_9 = QtWidgets.QHBoxLayout(self.layoutWidget_3)
183.         self.horizontalLayout_9.setContentsMargins(0, 0, 0, 0)
184.         self.horizontalLayout_9.setObjectName("horizontalLayout_9")
185.         self.label_7 = QtWidgets.QLabel(parent=self.layoutWidget_3)
186.         self.label_7.setObjectName("label_7")
187.         self.horizontalLayout_9.addWidget(self.label_7)
188.         self.zhuzhou = QtWidgets.QLineEdit(parent=self.layoutWidget_3)
189.         self.zhuzhou.setObjectName("zhuzhou")
190.         self.horizontalLayout_9.addWidget(self.zhuzhou)
191.         self.layoutWidget_4 = QtWidgets.QWidget(parent=self.groupBox_4)
192.         self.layoutWidget_4.setGeometry(QtCore.QRect(10, 140, 241, 22))
193.         self.layoutWidget_4.setObjectName("layoutWidget_4")
194.         self.horizontalLayout_11 = QtWidgets.QHBoxLayout(self.layoutWidget_4)
195.         self.horizontalLayout_11.setContentsMargins(0, 0, 0, 0)
196.         self.horizontalLayout_11.setObjectName("horizontalLayout_11")
197.         self.label_8 = QtWidgets.QLabel(parent=self.layoutWidget_4)
198.         self.label_8.setObjectName("label_8")
199.         self.horizontalLayout_11.addWidget(self.label_8)
200.         self.anquan = QtWidgets.QLineEdit(parent=self.layoutWidget_4)
201.         self.anquan.setObjectName("anquan")
202.         self.horizontalLayout_11.addWidget(self.anquan)
203.         self.layoutWidget_5 = QtWidgets.QWidget(parent=self.groupBox_4)
204.         self.layoutWidget_5.setGeometry(QtCore.QRect(10, 170, 241, 22))
205.         self.layoutWidget_5.setObjectName("layoutWidget_5")
206.         self.horizontalLayout_12 = QtWidgets.QHBoxLayout(self.layoutWidget_5)
207.         self.horizontalLayout_12.setContentsMargins(0, 0, 0, 0)
208.         self.horizontalLayout_12.setObjectName("horizontalLayout_12")
209.         self.label_11 = QtWidgets.QLabel(parent=self.layoutWidget_5)
210.         self.label_11.setObjectName("label_11")
211.         self.horizontalLayout_12.addWidget(self.label_11)
212.         self.shendu = QtWidgets.QLineEdit(parent=self.layoutWidget_5)
213.         self.shendu.setObjectName("shendu")
214.         self.horizontalLayout_12.addWidget(self.shendu)
215.         self.layoutWidget_6 = QtWidgets.QWidget(parent=self.groupBox_4)
216.         self.layoutWidget_6.setGeometry(QtCore.QRect(10, 20, 231, 24))
217.         self.layoutWidget_6.setObjectName("layoutWidget_6")
218.         self.horizontalLayout_14 = QtWidgets.QHBoxLayout(self.layoutWidget_6)
219.         self.horizontalLayout_14.setContentsMargins(0, 0, 0, 0)

```

```

220.         self.horizontalLayout_14.setObjectName("horizontalLayout_14")
221.         self.radioButton_juedui = QtWidgets.QRadioButton(parent=self.layoutWidget_6)
222.         self.radioButton_juedui.setObjectName("radioButton_juedui")
223.         self.horizontalLayout_14.addWidget(self.radioButton_juedui)
224.         self.radioButton_xiangdui = QtWidgets.QRadioButton(parent=self.layoutWidget_6)
225.         self.radioButton_xiangdui.setObjectName("radioButton_xiangdui")
226.         self.horizontalLayout_14.addWidget(self.radioButton_xiangdui)
227.         self.groupBox_2 = QtWidgets.QGroupBox(parent=self.tab_3)
228.         self.groupBox_2.setGeometry(QtCore.QRect(10, 10, 271, 51))
229.         self.groupBox_2.setObjectName("groupBox_2")
230.         self.layoutWidget7 = QtWidgets.QWidget(parent=self.groupBox_2)
231.         self.layoutWidget7.setGeometry(QtCore.QRect(10, 20, 231, 22))
232.         self.layoutWidget7.setObjectName("layoutWidget7")
233.         self.horizontalLayout_6 = QtWidgets.QHBoxLayout(self.layoutWidget7)
234.         self.horizontalLayout_6.setContentsMargins(0, 0, 0, 0)
235.         self.horizontalLayout_6.setObjectName("horizontalLayout_6")
236.         self.radioButton_clockwise = QtWidgets.QRadioButton(parent=self.layoutWidget7)
237.         self.radioButton_clockwise.setObjectName("radioButton_clockwise")
238.         self.horizontalLayout_6.addWidget(self.radioButton_clockwise)
239.         self.radioButton_anticlockwise = QtWidgets.QRadioButton(parent=self.layoutWidget7)
240.         self.radioButton_anticlockwise.setObjectName("radioButton_anticlockwise")
241.         self.horizontalLayout_6.addWidget(self.radioButton_anticlockwise)
242.         self.pushButton_back = QtWidgets.QPushButton(parent=self.tab_3)
243.         self.pushButton_back.setGeometry(QtCore.QRect(150, 440, 131, 51))
244.         self.pushButton_back.setStyleSheet("background-color: rgb(255, 246, 193);\n"
245.         "")
246.         self.pushButton_back.setObjectName("pushButton_back")
247.         self.pushButton_NC_code = QtWidgets.QPushButton(parent=self.tab_3)
248.         self.pushButton_NC_code.setGeometry(QtCore.QRect(10, 440, 131, 51))
249.         self.pushButton_NC_code.setStyleSheet("background-color: rgb(255, 246, 193);\n"
250.         "")
251.         self.pushButton_NC_code.setObjectName("pushButton_NC_code")
252.         self.groupBox_3 = QtWidgets.QGroupBox(parent=self.tab_3)
253.         self.groupBox_3.setGeometry(QtCore.QRect(10, 70, 271, 51))
254.         self.groupBox_3.setObjectName("groupBox_3")
255.         self.layoutWidget8 = QtWidgets.QWidget(parent=self.groupBox_3)
256.         self.layoutWidget8.setGeometry(QtCore.QRect(10, 20, 231, 22))
257.         self.layoutWidget8.setObjectName("layoutWidget8")
258.         self.horizontalLayout_10 = QtWidgets.QHBoxLayout(self.layoutWidget8)
259.         self.horizontalLayout_10.setContentsMargins(0, 0, 0, 0)
260.         self.horizontalLayout_10.setObjectName("horizontalLayout_10")
261.         self.radioButton_zuodaobu = QtWidgets.QRadioButton(parent=self.layoutWidget8)
262.         self.radioButton_zuodaobu.setObjectName("radioButton_zuodaobu")
263.         self.horizontalLayout_10.addWidget(self.radioButton_zuodaobu)

```

```

264.         self.radioButton_youdaobu = QtWidgets.QRadioButton(parent=self.layoutWidget8)
265.         self.radioButton_youdaobu.setObjectName("radioButton_youdaobu")
266.         self.horizontalLayout_10.addWidget(self.radioButton_youdaobu)
267.         self.groupBox_6 = QtWidgets.QGroupBox(parent=self.tab_3)
268.         self.groupBox_6.setGeometry(QtCore.QRect(10, 340, 271, 51))
269.         self.groupBox_6.setObjectName("groupBox_6")
270.         self.layoutWidget9 = QtWidgets.QWidget(parent=self.groupBox_6)
271.         self.layoutWidget9.setGeometry(QtCore.QRect(10, 20, 251, 24))
272.         self.layoutWidget9.setObjectName("layoutWidget9")
273.         self.horizontalLayout_18 = QtWidgets.QHBoxLayout(self.layoutWidget9)
274.         self.horizontalLayout_18.setContentsMargins(0, 0, 0, 0)
275.         self.horizontalLayout_18.setObjectName("horizontalLayout_18")
276.         self.horizontalLayout_15 = QtWidgets.QHBoxLayout()
277.         self.horizontalLayout_15.setObjectName("horizontalLayout_15")
278.         self.label_3 = QtWidgets.QLabel(parent=self.layoutWidget9)
279.         self.label_3.setObjectName("label_3")
280.         self.horizontalLayout_15.addWidget(self.label_3)
281.         self.label_X = QtWidgets.QLineEdit(parent=self.layoutWidget9)
282.         self.label_X.setObjectName("label_X")
283.         self.horizontalLayout_15.addWidget(self.label_X)
284.         self.horizontalLayout_18.addLayout(self.horizontalLayout_15)
285.         self.horizontalLayout_16 = QtWidgets.QHBoxLayout()
286.         self.horizontalLayout_16.setObjectName("horizontalLayout_16")
287.         self.label_4 = QtWidgets.QLabel(parent=self.layoutWidget9)
288.         self.label_4.setObjectName("label_4")
289.         self.horizontalLayout_16.addWidget(self.label_4)
290.         self.label_Y = QtWidgets.QLineEdit(parent=self.layoutWidget9)
291.         self.label_Y.setObjectName("label_Y")
292.         self.horizontalLayout_16.addWidget(self.label_Y)
293.         self.horizontalLayout_18.addLayout(self.horizontalLayout_16)
294.         self.horizontalLayout_17 = QtWidgets.QHBoxLayout()
295.         self.horizontalLayout_17.setObjectName("horizontalLayout_17")
296.         self.label_5 = QtWidgets.QLabel(parent=self.layoutWidget9)
297.         self.label_5.setObjectName("label_5")
298.         self.horizontalLayout_17.addWidget(self.label_5)
299.         self.label_Z = QtWidgets.QLabel(parent=self.layoutWidget9)
300.         self.label_Z.setObjectName("label_Z")
301.         self.horizontalLayout_17.addWidget(self.label_Z)
302.         self.horizontalLayout_18.addLayout(self.horizontalLayout_17)
303.         self.pushButton_jiagongdonghua = QtWidgets.QPushButton(parent=self.tab_3)
304.         self.pushButton_jiagongdonghua.setGeometry(QtCore.QRect(14, 400, 271, 31))
305.         self.pushButton_jiagongdonghua.setStyleSheet("background-color: rgb(255, 246, 193);\n"
306.         "")
307.         self.pushButton_jiagongdonghua.setObjectName("pushButton_jiagongdonghua")

```

```

308.         self.tabWidget.addTab(self.tab_3, "")
309.         self.tab_2 = QtWidgets.QWidget()
310.         self.tab_2.setObjectName("tab_2")
311.         self.NC_code = QtWidgets.QScrollArea(parent=self.tab_2)
312.         self.NC_code.setGeometry(QtCore.QRect(10, 10, 271, 421))
313.         self.NC_code.setWidgetResizable(True)
314.         self.NC_code.setObjectName("NC_code")
315.         self.scrollAreaWidgetContents = QtWidgets.QWidget()
316.         self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 269, 419))
317.         self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
318.         self.verticalScrollBar = QtWidgets.QScrollBar(parent=self.scrollAreaWidgetContents)
319.         self.verticalScrollBar.setGeometry(QtCore.QRect(250, -1, 20, 421))
320.         self.verticalScrollBar.setOrientation(QtCore.Qt.Orientation.Vertical)
321.         self.verticalScrollBar.setObjectName("verticalScrollBar")
322.         self.horizontalScrollBar = QtWidgets.QScrollBar(parent=self.scrollAreaWidgetContents)
323.         self.horizontalScrollBar.setGeometry(QtCore.QRect(-1, 400, 251, 20))
324.         self.horizontalScrollBar.setOrientation(QtCore.Qt.Orientation.Horizontal)
325.         self.horizontalScrollBar.setObjectName("horizontalScrollBar")
326.         self.NC_code.setWidget(self.scrollAreaWidgetContents)
327.         self.pushButton_back2 = QtWidgets.QPushButton(parent=self.tab_2)
328.         self.pushButton_back2.setGeometry(QtCore.QRect(10, 440, 271, 51))
329.         self.pushButton_back2.setStyleSheet("background-color: rgb(255, 246, 193);\n"
330.         "")
331.         self.pushButton_back2.setObjectName("pushButton_back2")
332.         self.tabWidget.addTab(self.tab_2, "")
333.         NC_Project.setCentralWidget(self.centralwidget)
334.         self.menubar = QtWidgets.QMenuBar(parent=NC_Project)
335.         self.menubar.setGeometry(QtCore.QRect(0, 0, 909, 22))
336.         self.menubar.setObjectName("menubar")
337.         NC_Project.setMenuBar(self.menubar)
338.         self.statusbar = QtWidgets.QStatusBar(parent=NC_Project)
339.         self.statusbar.setObjectName("statusbar")
340.         NC_Project.setStatusBar(self.statusbar)
341.
342.         self.retranslateUi(NC_Project)
343.         self.tabWidget.setCurrentIndex(0)
344.         QtCore.QMetaObject.connectSlotsByName(NC_Project)
345.
346.     def retranslateUi(self, NC_Project):
347.         _translate = QtCore.QCoreApplication.translate
348.         NC_Project.setWindowTitle(_translate("NC_Project", "数控课程设计-非圆曲线的 CAM 软件"))
349.         self.groupBox.setTitle(_translate("NC_Project", "曲线选择"))
350.         self.radioButton_xinzang.setText(_translate("NC_Project", "心脏线"))
351.         self.radioButton_tulun.setText(_translate("NC_Project", "正弦加速盘形凸轮轮廓线"))

```

```

352.         self.groupBox_xinzang.setTitle(_translate("NC_Project", "曲线方程及参数"))
353.         self.label.setText(_translate("NC_Project", "a="))
354.         self.groupBox_5.setTitle(_translate("NC_Project", "插补算法分析"))
355.         self.comboBox_chabu.setCurrentText(_translate("NC_Project", "          等间距法"))
356.         self.comboBox_chabu.setItemText(0, _translate("NC_Project", "          等间距法"))
357.         self.comboBox_chabu.setItemText(1, _translate("NC_Project", "          等误差法"))
358.         self.label_9.setText(_translate("NC_Project", "段数: "))
359.         self.label_10.setText(_translate("NC_Project", "允许误差: "))
360.         self.pushButton_start.setText(_translate("NC_Project", "生成函数"))
361.         self.pushButton_close.setText(_translate("NC_Project", "关闭程序"))
362.         self.pushButton_fangzhen.setText(_translate("NC_Project", "仿真加工"))
363.         self.pushButton_about.setText(_translate("NC_Project", "关于"))
364.         self.groupBox_tulun.setTitle(_translate("NC_Project", "曲线方程及参数"))
365.         self.label_31.setText(_translate("NC_Project", "基圆半径: "))
366.         self.label_34.setText(_translate("NC_Project", "行程:  "))
367.         self.label_35.setText(_translate("NC_Project", "远休止角: "))
368.         self.label_37.setText(_translate("NC_Project", "近休止角: "))
369.         self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), _translate("NC_Project", "曲线设置"))
370.         self.groupBox_4.setTitle(_translate("NC_Project", "加工参数"))
371.         self.label_6.setText(_translate("NC_Project", "进给速度: "))
372.         self.label_2.setText(_translate("NC_Project", "刀具半径: "))
373.         self.label_7.setText(_translate("NC_Project", "主轴转速: "))
374.         self.label_8.setText(_translate("NC_Project", "安全高度: "))
375.         self.label_11.setText(_translate("NC_Project", "加工深度: "))
376.         self.radioButton_juedui.setText(_translate("NC_Project", "绝对坐标"))
377.         self.radioButton_xiangdui.setText(_translate("NC_Project", "相对坐标"))
378.         self.groupBox_2.setTitle(_translate("NC_Project", "走刀方向"))
379.         self.radioButton_clockwise.setText(_translate("NC_Project", "顺时针"))
380.         self.radioButton_anticlockwise.setText(_translate("NC_Project", "逆时针"))
381.         self.pushButton_back.setText(_translate("NC_Project", "返回曲线设置"))
382.         self.pushButton_NC_code.setText(_translate("NC_Project", "生成 NC 代码"))
383.         self.groupBox_3.setTitle(_translate("NC_Project", "刀补方式"))
384.         self.radioButton_zuodaobu.setText(_translate("NC_Project", "左刀补"))
385.         self.radioButton_youdaobu.setText(_translate("NC_Project", "右刀补"))
386.         self.groupBox_6.setTitle(_translate("NC_Project", "起刀点"))
387.         self.label_3.setText(_translate("NC_Project", "X="))
388.         self.label_4.setText(_translate("NC_Project", "Y="))
389.         self.label_5.setText(_translate("NC_Project", "Z="))
390.         self.label_Z.setText(_translate("NC_Project", "TextLabel"))
391.         self.pushButton_jiagongdonghua.setText(_translate("NC_Project", "生成仿真加工动画"))
392.         self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_3), _translate("NC_Project", "仿真加工"))
393.         self.pushButton_back2.setText(_translate("NC_Project", "返回仿真加工设置"))
394.         self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2), _translate("NC_Project", "NC 代码"))

```

3. daobu_chabu_dengjianv.py

```
1.     import math
2.     import numpy as np
3.     import matplotlib.pyplot as plt
4.     import cv2
5.     from PyQt6.QtWidgets import QMessageBox
6.     np.seterr(divide='ignore',invalid='ignore')
7.
8.     def offset_curve(x_values, y_values, tool_radius, direction):
9.         guoque = False
10.        offsets = []
11.        if direction == 'right':
12.            tool_radius = -tool_radius
13.            first_point_x = x_values[0]
14.            second_point_x = x_values[1]
15.            third_point_x = x_values[2]
16.            first_point_y = y_values[0]
17.            second_point_y = y_values[1]
18.            third_point_y = y_values[2]
19.            x = 0
20.            y = 0
21.            for i in range(3, len(x_values)):
22.                dist1 = float(math.sqrt(
23.                    math.pow(second_point_x - first_point_x, 2) + math.pow(second_point_y - first_point_y, 2)))
24.                dist2 = float(math.sqrt(
25.                    math.pow(third_point_x - second_point_x, 2) + math.pow(third_point_y - second_point_y, 2)))
26.                x_l1 = float((second_point_x - first_point_x) / dist1)
27.                y_l1 = float((second_point_y - first_point_y) / dist1)
28.                x_l2 = float((third_point_x - second_point_x) / dist2)
29.                y_l2 = float((third_point_y - second_point_y) / dist2)
30.                if tool_radius * (y_l2 * x_l1 - y_l1 * x_l2) >= 0:
31.                    if i == len(x_values):
32.                        x = second_point_x - tool_radius * y_l1
33.                        y = second_point_y + tool_radius * x_l1
34.                    else:
35.                        x = tool_radius * (x_l2 - x_l1) / (x_l1 * y_l2 - x_l2 * y_l1) + second_point_x
36.                        y = tool_radius * (y_l2 - y_l1) / (x_l1 * y_l2 - x_l2 * y_l1) + second_point_y
37.                    offsets.append((x, y))
38.                    if len(offsets) >= 2:
39.                        if (x - offsets[i - 4][0]) * (second_point_x - first_point_x) < 0:
40.                            guoque = True
41.                        elif tool_radius * (y_l2 * x_l1 - y_l1 * x_l2) < 0 <= y_l1 * y_l2 + x_l1 * x_l2:
42.                            if i == len(x_values):
43.                                x1 = tool_radius * (x_l2 - x_l1) / (x_l1 * y_l2 - x_l2 * y_l1) + second_point_x
```



```

44.         y1 = tool_radius * (y_l2 - y_l1) / (x_l1 * y_l2 - x_l2 * y_l1) + second_point_y
45.         x2 = second_point_x - tool_radius * y_l2
46.         y2 = second_point_y + tool_radius * x_l2
47.         offsets.append((x1, y1))
48.         offsets.append((x2, y2))
49.     else:
50.         x = tool_radius * (x_l2 - x_l1) / (x_l1 * y_l2 - x_l2 * y_l1) + second_point_x
51.         y = tool_radius * (y_l2 - y_l1) / (x_l1 * y_l2 - x_l2 * y_l1) + second_point_y
52.         offsets.append((x, y))
53.     elif tool_radius * (y_l2 * x_l1 - y_l1 * x_l2) < 0 < y_l1 * y_l2 + x_l1 * x_l2:
54.         x1 = second_point_x - tool_radius * y_l1
55.         y1 = second_point_y + tool_radius * x_l1
56.         x2 = math.fabs(tool_radius) * x_l1 + x1
57.         y2 = math.fabs(tool_radius) * y_l1 + y1
58.         x3 = second_point_x - tool_radius * y_l2 - math.fabs(tool_radius) * x_l2
59.         y3 = second_point_y + tool_radius * x_l2 - math.fabs(tool_radius) * y_l2
60.         offsets.append((x1, y1))
61.         offsets.append((x2, y2))
62.         offsets.append((x3, y3))
63.
64.         first_point_x = second_point_x
65.         first_point_y = second_point_y
66.         second_point_x = third_point_x
67.         second_point_y = third_point_y
68.         third_point_x = x_values[i]
69.         third_point_y = y_values[i]
70.
71.         offsets.append((offsets[0]))
72.         if guoqie and direction == 'left':
73.             left = int(len(offsets) * 3 / 8)
74.             right = int(len(offsets) * 5 / 8)
75.             while offsets[left][1] >= offsets[right][1]:
76.                 left += 1
77.             while offsets[right - 1][1] <= offsets[right][1]:
78.                 right -= 1
79.             while left + 1 != right:
80.                 del offsets[left + 1]
81.                 right -= 1
82.                 offsets.insert(left + 1, (offsets[left][0], offsets[left][1] - tool_radius))
83.                 offsets.insert(left + 2, (offsets[left + 2][0], offsets[left + 2][1] - tool_radius))
84.         return zip(*offsets)
85. def plot_heart_curve_with_tool_path(x_values, y_values, tool_radius, direction):
86.     offset_x, offset_y = offset_curve(x_values, y_values, tool_radius, direction)
87.     offset_x = list(offset_x)

```

```

88.     offset_y = list(offset_y)
89.     if direction == 'left':
90.         while offset_x[0] < 0:
91.             del offset_x[0]
92.             del offset_y[0]
93.         while True:
94.             num = len(offset_x) - 1
95.             if offset_x[num] > 0 or offset_y[num] < offset_y[0]:
96.                 del offset_x[num]
97.                 del offset_y[num]
98.             else:
99.                 break
100.        return offset_x, offset_y
101.
102.    elif direction == 'right':
103.        left = int(len(offset_x) / 4)
104.        right = int(len(offset_x) * 3 / 4)
105.        while left != right:
106.            if offset_x[left] > 0:
107.                left += 1
108.            else:
109.                break
110.        while left != right:
111.            if offset_x[right] < 0:
112.                right -= 1
113.            else:
114.                break
115.        del offset_x[left: right]
116.        del offset_y[left: right]
117.        while True:
118.            num = len(offset_x) - 1
119.            if num <= 0:
120.                msg_box = QMessageBox()
121.                msg_box.setWindowTitle("过切警告")
122.                msg_box.setText("刀具半径过大造成过切, 请减小刀具半径")
123.                msg_box.exec()
124.            break
125.        if offset_x[num] > 0 or offset_y[num] <= offset_y[0]:
126.            del offset_x[num]
127.            del offset_y[num]
128.        else:
129.            break
130.    return offset_x, offset_y

```

4. daobu_chabu_dengwucha_sin.py

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from sin_line_process.dengwuchachabu.dengwucha_suanfa import equal_error_interpolation
4. def offset_curve(x_values, y_values, tool_radius, direction='left'):
5.     offsets = []
6.     for i in range(len(x_values) - 1):
7.         dx = x_values[i + 1] - x_values[i]
8.         dy = y_values[i + 1] - y_values[i]
9.         dist = np.sqrt(dx ** 2 + dy ** 2)
10.        if dist == 0: # 防止除以零
11.            continue
12.        norm_dx = dx / dist
13.        norm_dy = dy / dist
14.        if direction == 'left':
15.            offset_x = x_values[i] - tool_radius * norm_dx
16.            offset_y = y_values[i] + tool_radius * norm_dy
17.        elif direction == 'right':
18.            offset_x = x_values[i] + tool_radius * norm_dx
19.            offset_y = y_values[i] - tool_radius * norm_dy
20.        else:
21.            raise ValueError("Direction must be 'left' or 'right'.")
22.        offsets.append((offset_x, offset_y))
23.    return zip(*offsets) # 返回 X 和 Y 的偏移坐标
24.
25.
26. def plot_heart_curve_with_tool_path(x_values, y_values, tool_radius, direction):
27.     offset_x, offset_y = offset_curve(x_values, y_values, tool_radius, direction)
28.     plt.figure(figsize=(10, 10))
29.     plt.plot(x_values, y_values, label='Heart Curve')
30.     plt.plot(list(offset_x), list(offset_y), label=f'Tool Path ({direction} offset)')
31.     plt.xlabel('X')
32.     plt.ylabel('Y')
33.     plt.title('Heart Curve with Tool Path')
34.     plt.legend()
35.     plt.grid(True)
36.     plt.show()
```

5. generate_nccode_v4.py

```
1.     def generate_nc_code(t_values, x_values, y_values, tool_radius, feed_rate, spindle_speed,
2.                           safety_height, cut_depth, start_point, compensation_direction,
3.                           use_relative_coordinates):
4.         nc_code = []
5.         # 初始化 NC 代码
6.         nc_code.append("%0001")
7.         # 设置坐标系和起刀点
8.         if use_relative_coordinates:
9.             nc_code.append(f"N1 G91 G21 G40 M03 S{spindle_speed} F{feed_rate}") # 在相对坐标下移动到起刀点（需要确保是相对
            于初始位置的偏移）
10.            nc_code.append(f"N2 G00 Z{safety_height}")
11.            start_x, start_y = start_point
12.            nc_code.append(f"N3 G91 G00 X{start_x} Y{start_y}")
13.            nc_code.append(f"N4 G01 Z-{cut_depth + safety_height}") # 下刀到加工深度
14.            # 设置刀具补偿方向
15.            if compensation_direction == 'left':
16.                nc_code.append("N5 G41 D1") # 左刀补
17.            elif compensation_direction == 'right':
18.                nc_code.append("N6 G42 D1") # 右刀补
19.            # 遍历离散点并生成直线插补指令
20.            for i in range(len(t_values) - 1):
21.                xi, yi = x_values[i], y_values[i]
22.                xi_next, yi_next = x_values[i + 1], y_values[i + 1]
23.                if use_relative_coordinates:
24.                    dx = xi_next - xi
25.                    dy = yi_next - yi
26.                    nc_code.append(f"N{i + 5} G01 X{dx:.4f} Y{dy:.4f}") # 使用相对坐标进行直线插补
27.                else:
28.                    nc_code.append(f"N{i + 5} G01 X{xi_next:.4f} Y{yi_next:.4f}") # 使用绝对坐标进行直线插补
29.                if use_relative_coordinates:
30.                    nc_code.append(f"N{len(t_values) + 5} G40 G00 Z{cut_depth + safety_height}")
31.                else:
32.                    nc_code.append(f"N{len(t_values) + 5} G40 G00 Z{safety_height}") # 快速定位到安全高度
33.            nc_code.append(f"N{len(t_values) + 6} M30") # 程序结束
34.            return nc_code
35.        else:
36.            nc_code.append(f"N1 G92 X0 Y0")
37.            nc_code.append(f"N2 G90 G21 G40 M03 S{spindle_speed} F{feed_rate}") # 在相对坐标下移动到起刀点（需要确保是相对
            于初始位置的偏移）
38.            nc_code.append(f"N3 G00 Z{safety_height}")
39.            start_x, start_y = start_point
40.            nc_code.append(f"N4 G90 G00 X{start_x} Y{start_y}")
41.            nc_code.append(f"N5 G01 Z-{cut_depth}") # 下刀到加工深度
```

```

42.         # 设置刀具补偿方向
43.         if compensation_direction == 'left':
44.             nc_code.append("N6 G41 D1") # 左刀补
45.         elif compensation_direction == 'right':
46.             nc_code.append("N7 G42 D1") # 右刀补
47.         # 遍历离散点并生成直线插补指令
48.         for i in range(len(t_values) - 1):
49.             xi, yi = x_values[i], y_values[i]
50.             xi_next, yi_next = x_values[i + 1], y_values[i + 1]
51.             if use_relative_coordinates:
52.                 dx = xi_next - xi
53.                 dy = yi_next - yi
54.                 nc_code.append(f"N{i + 7} G01 X{dx:.4f} Y{dy:.4f}") # 使用相对坐标进行直线插补
55.             else:
56.                 nc_code.append(f"N{i + 7} G01 X{xi_next:.4f} Y{yi_next:.4f}") # 使用绝对坐标进行直线插补
57.             if use_relative_coordinates:
58.                 nc_code.append(f"N{len(t_values) + 6} G40 G00 Z{cut_depth + safety_height}")
59.             else:
60.                 nc_code.append(f"N{len(t_values) + 6} G40 G00 Z{safety_height}") # 快速定位到安全高度
61.             nc_code.append(f"N{len(t_values) + 7} M30") # 程序结束
62.         return nc_code

```