Approximation of CNC Part Program within a Given Tolerance Band

Min Qiao

Department of Mathematics

Justus-Liebig University Giessen

A thesis submitted for the degree of

Dr. rer. nat.

Erlangen, Oct. 2010

1. Reviewer: Prof. Dr. Tomas Sauer

2. Reviewer: Prof. Dr. Johannes Wallner

3. Reviewer: Prof. Dr. Wolfgang Papiernik

Abstract

Freeform surfaces are widely used in various fields of engineering design such as automotive, aerospace and mold and die industries. With the geometric model designed by freeform surfaces in the CAD system, the tool path will be next constructed and sampled by the CAM system according to the predefined chordal deviation. The sequentially sampled short linear blocks are stored in the file called the part program, which is later interpreted by the CNC system to control the machine tool and mill the workpiece.

In order to deliver the desired motion smoothly and rapidly, the short linear blocks in the part program will be approximated with smooth spline curves within a specified tolerance band. With spline curves, which consist of longer polynomial pieces, the attainable feed rate of the machine tool can be greatly enhanced. In addition, with minimization of curvature variation, velocity and acceleration jumps in the relevant machine axes and undesired mechanical oscillation of the machine can be reduced significantly. This allows higher traversing speeds and reduces the vibration of the machine, thus improves the surface quality.

Acknowledgements

First, I want to express my sincere thanks to Prof. Tomas Sauer for his patient guidance throughout my study. His profound knowledge and enthusiasm in research motivate me to solve the problems and the discussion with him is always full of inspirations and ideas.

My deepest gratitude also goes to Prof. Wolfgang Papiernik. He offers invaluable advice and suggestions whenever I need them. His expertise has broadened my perspective on the practical aspects in the industry.

In particular, I want to convey my thanks to Dr. wolfgang Speth for his generous support and constructive comments. Moreover, I would also like to thank my colleagues and friends especially Marco, Paul, Loay, Dominik, Florian and Philippe at Siemens for their advice and help.

Last but not least, I would like to thank my family who has supported me from the very beginning.

Contents

List of Figures **vii**

1 Introduction **1**

2 Theoretical background **7**

2.1 Fundamental concepts 7

2.2 Spline curves 11

2.2.1 Bézier curves 11

2.2.2 Basics about spline curves 12

2.2.3 Derivatives of spline curves 15

2.2.4 Integral of spline curves 17

2.2.5 Knot insertion 19

2.2.6 De Boor's algorithm 20

2.2.7 Knot removal 21

2.2.8 Degree elevation 22

2.2.9 Application of spline curves 25

2.2.9.1 Parameter and knot sequence selection 25

2.2.9.2 Interpolation with splines 27

2.2.9.3 Least squares approximation with splines 28

2.3 Equality-constrained optimization scheme 30

3 Basic strategy **33**

3.1 The problem 33

3.2 Curve segmentation 34

3.3 Knot selection 36

3.3.1 Knot distribution based on curvature characteristics 36

3.3.2 Curvature jump detection and multiple knots 40
3.3.3 Number of knots 42
3.4 Approximation 45
3.4.1 Comparison between two splines 46
3.4.2 Least squares approximation 49
3.4.3 Smoothness 49
3.4.4 Optimization problem 52
3.5 Summary of the strategy 53

4 Extended strategy **55**
4.1 Preprocessing 55
4.1.1 Cluster modifications 55
4.1.2 Remove redundant points on a straight line 58
4.2 Localization 60
4.2.1 Local knot modification 60
4.2.2 Local smoothness 64
4.3 Soft edge detection and quasi multiple knots 65
4.4 Curvature estimation methods 67
4.4.1 Divided difference 68
4.4.2 Area invariant and Connolly function for planar curves 70
4.4.3 Area invariant method vs circumcircle method 72
4.5 Reference curve and smoothness 77
4.6 Complete methods 79

5 Experimental results **81**
5.1 Example 1: the influence of knot's multiplicity 81
5.2 Example 2: the influence of knot distribution 83
5.3 Example 3: the influence of the smoothness term 84
5.4 The typical test workpieces 84

6 Conclusions and remarks **93**

Bibliography **97**

List of Figures

1.1 Process chain of CNC machining (Siemens) 2
1.2 Key components in numerical control kernel (NCK) (Siemens) 2
1.3 Compressor: approximate the linear blocks with smooth spline curves within a tolerance band (Siemens) 4
1.4 Spoiled surface caused by short linear blocks (left: ideal; middle: beveled pattern; right: vibration) (Siemens) 5

2.1 C^n continuity at the joint $\mathbf{p}$ 8
2.2 The relation between derivatives with respect to s and u 10
2.3 A straight line which is G^1 but not C^1 continuous 10
2.4 A curve which is G^1 but not C^1 continuous 10
2.5 The shape of a curve is greatly influenced by parameter selection methods 27
2.6 Spline interpolation vs least squares approximation 29

3.1 Approximation within the tolerance band with a smooth spline curve . . 33
3.2 Oscillation or bulges caused by undetected sharp edges 34
3.3 Edge detection by angles between neighboring points 35
3.4 The relation between the block length and the curvature radius 37
3.5 Discrete curvature estimation by circumcircle through three neighboring points 39
3.6 Integral of square root of curvature for knot placement 39
3.7 Comparison between the approximating results with multiple knots and simple knots 41
3.8 Integral of square root of curvature for knot placement with multiple knots 42
3.9 Data points on the part program 'daimler' 43

LIST OF FIGURES

3.10 Curvature plot using the circumcirle method 43
3.11 Integral of square root of curvature 44
3.12 The approximating spline curve and the corresponding polynomial pieces 44
3.13 Determine the initial number q_k based on N_k 45
3.14 Control points before knot insertion and degree elevation 48
3.15 Control points after knot insertion and degree elevation 48
3.16 The deviation to arc length parametrization 51

4.1 Critical points (close clusters) in the part program 'beetle' 56
4.2 A cluster of close points 56
4.3 Flowchart of the cluster modification method 59
4.4 Detection of redundant points on a straight line 59
4.5 Map the violating intervals from T^f to T^m 61
4.6 Overlapped violating intervals in T^m 61
4.7 Local modification scheme 63
4.8 Top: global knot placement; Bottom: local knot placement 64
4.9 Knot placement for a soft edge decision 66
4.10 Knot placement with quasi multiple knots 67
4.11 The relation between the knot distance d and opening angle α 68
4.12 Significant error when estimating curvature of noisy data using the circumcircle method 69
4.13 Area invariant method for curvature estimation of planar curves 71
4.14 Connolly function 72
4.15 Area invariant method for a non-smooth curve 73
4.16 Example 1(a): area invariant vs circumcircle method 74
4.17 Example 1(b): area invariant vs circumcircle method on noisy data . . . 75
4.18 Example 2: area invariant vs circumcircle method for curvature jump detection 75
4.19 Example 3: area invariant vs circumcircle method on workpiece 'beetle' 76

5.1 Spline approximation with correct edge detection 82
5.2 Spline approximation with 4-fold knots at edges 82
5.3 Spline approximation with simple knots at edges 83
5.4 Error plot with different knot's multiplicity at edges 86

5.5 One test path from the workpiece 'daimler' 87
5.6 The approximation error with uniformly-distributed simple knots 87
5.7 The approximation error with knots based on curvature 88
5.8 The error to arc length parametrization 88
5.9 The milling result of the workpiece 'turm' 89
5.10 Plot of discrete curvature 89
5.11 Case 1: no smoothing (top); Case 2: $\min \|\ddot{\mathbf{f}}(s)\|^2$ (middle); Case 3: $\min \|\dddot{\mathbf{f}}(s)\|^2$ (down) 90
5.12 The part program of the workpiece 'daimler' 91
5.13 The spline approximation of the workpiece 'daimler' 91
5.14 The curvature visualization of the workpiece 'daimler' 92
5.15 The milling result of the workpiece 'daimler' 92

LIST OF FIGURES

1

Introduction

Freeform surfaces are widely used to describe the surface of complex 3D geometric elements such as turbine blades and car bodies in various fields of engineering design from automotive, aerospace to mold and die industries. In modern industry, the process starting from freeform surface modeling to surface finishing is highly automated, benefiting from the integration of CAD/CAM (*Computer Aided Design/Manufacturing*) system and CNC (*Computerized Numerical Control*) machine tools.

In this process as shown in Fig. 1.1, the geometric model of the workpiece will be firstly designed by freeform surfaces of certain types within a CAD system. Given a user specified cutting strategy, the tool path that the center point of the milling tool has to follow will next be constructed with cutter radius compensation. The tool path usually cannot be expressed in a closed mathematical form, therefore it will be sampled by the CAM system according to the predefined chordal deviation. The sequentially sampled short linear blocks together with their block number and other commands are stored in the file called the *part program*. The CNC system interprets the part program and then controls the machine tool to mill the workpiece.

The purpose of the NC machine is to generate workpieces of complex shapes rapidly and precisely. The key functional components and task flow of the *numerical control kernel* (NCK) are shown in Fig. 1.2. First, the interpreter parses the blocks in the part program and stores the interpreted data in the block buffer. The compressor carries out the task of converting the short linear blocks to C^2 continuous (spline) curves. Next the look ahead function "looks ahead" for critical points such as singular points or regions with extreme curvature and calculates maximal feasible feed rates within the maximal

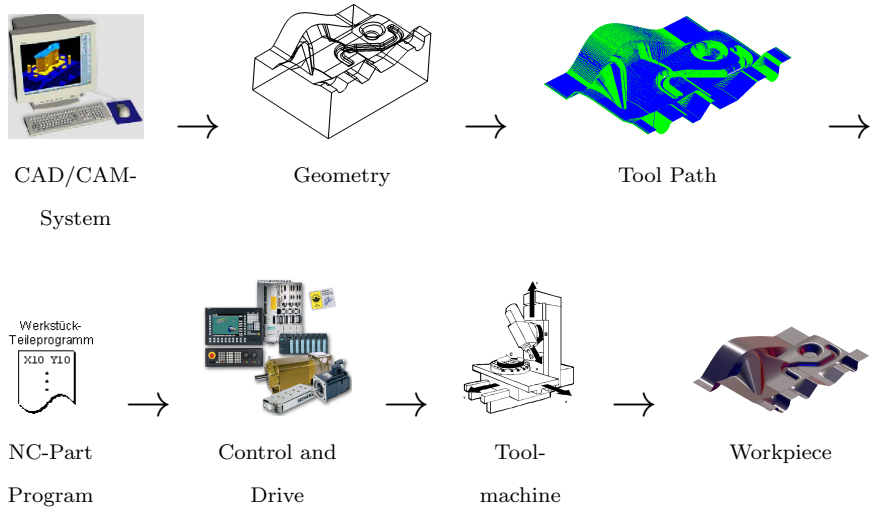


Figure 1.1: Process chain of CNC machining (Siemens)

allowable ranges for velocity, acceleration and jerk. Based on the preparation and look ahead function, the motion control unit generates the profiles for path position, path velocity and path acceleration, which are afterwards interpolated according to the interpolation sampling grid to generate the setpoints for the path position, velocity and acceleration. Subsequently, kinematic transformations are performed to map the setpoints from the Cartesian coordinate to the machine coordinate system for position control. The position controller issues velocity commands to the motor driving system in order to minimize the position difference between the set position and the actual position.

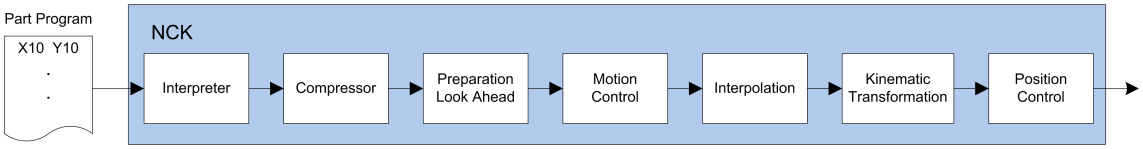


Figure 1.2: Key components in numerical control kernel (NCK) (Siemens)

Among the key components of the NCK, the *compressor* is the main subject of the thesis. The task of a compressor is to approximate the short linear blocks by smooth spline curves consisting of longer polynomial pieces within a specified tolerance band, as illustrated in Fig. 1.3. The motivation to convert the linear blocks to smooth splines

is based on the following arguments:

- When the tool path is sampled in the CAM system, a small chordal error is chosen to guarantee high approximation accuracy, which often results in short linear blocks and thus large volumes of data. A part program that is larger than 10MB is quite common. The large volume of data demands larger storage space and consumes longer transmission and processing power.

- On the other hand, the short linear blocks restrict the maximum feed rate. Since the linear blocks are processed by the CNC system at a certain interpolation rate, the cycle T of which is typically 1 to 10ms, the maximum feed rate for a block of length l is given by
$$F_{\max} = \frac{l}{T}.$$
For example, if the block length l is 10 μm, the interpolation cycle T is 4 ms, then the maximum feed rate is $F_{\max} = \frac{l}{T} = 150\frac{\text{mm}}{\text{min}}$, which is quite slow compared to the typical range of values of the feed rate: $5000 - 15000$ mm/min.

- The short linear blocks can also lead to acceleration jumps in the machine axes at the block transitions. If the machine tool travels the path at a constant velocity, it results in Dirac pulses in the acceleration at the block transitions. Therefore the path velocity must be reduced significantly around the block transitions in order to avoid overloading the machine tool. Furthermore it can cause resonance in the machine axes which will spoil the surface of the workpiece by chamfered edges or by vibration as shown in Fig. 1.4.

Therefore, it is of great importance to approximate the linear blocks by smooth spline curves in order to deliver the desired motion smoothly and rapidly. With spline curves, which consist of longer polynomial pieces, the attainable feed rate of the machine tool can be greatly enhanced. In addition, with smooth C^2 continuous curves, velocity and acceleration jumps in the relevant machine axes and undesired mechanical oscillation of the machine can be reduced significantly. This allows higher traversing speeds and reduces the vibration of the machine, thus improves the surface quality.

The basic strategies to perform spline approximation within tolerance band and least squares optimization are discussed and investigated in Chapter 3. Two principal

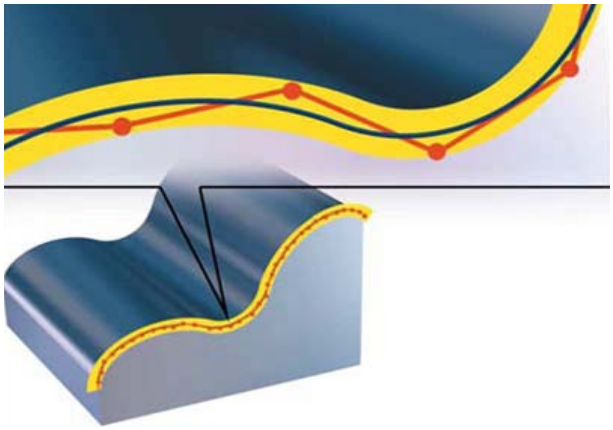


Figure 1.3: Compressor: approximate the linear blocks with smooth spline curves within a tolerance band (Siemens)

issues are the knot placement based on curvature characteristics and minimization of curvature variation for smoother spline curves. The extended strategies such as preprocessing, localization and soft edge detection methods are investigated in Chapter 4 to deal with some critical problems in the part program. The experimental results, concerning the compression rate, curvature variation of the spline curve and the surface quality are illustrated and evaluated in Chapter 5.

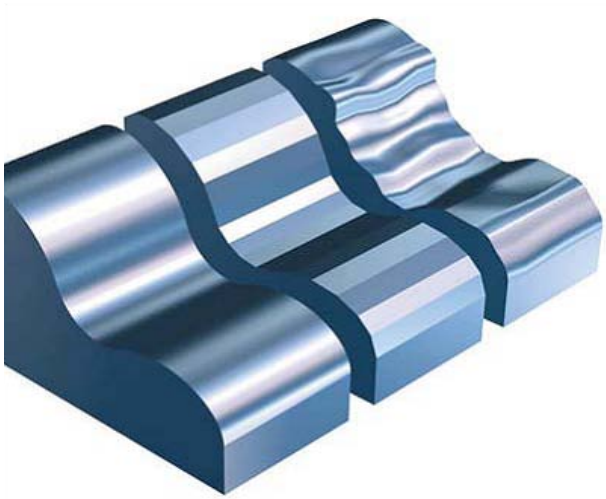


Figure 1.4: Spoiled surface caused by short linear blocks (left: ideal; middle: beveled pattern; right: vibration) (Siemens)

2

Theoretical background

2.1 Fundamental concepts

Parametric curves

Curves and surfaces can have *explicit*, *implicit*, and *parametric* representations, among which parametric representations are most commonly used in computer graphics and CAD.

An implicit function is a function in which the dependent variables have not been given explicitly in terms of the independent variables. To give a function f explicitly is to provide a prescription for determining the output value of the function y in terms of the input value x:

$$y = f(x).$$

In contrast, the function is implicit if the value of y is defined relative to x by solving an equation of the form:

$$R(x, y) = 0.$$

A parametric curve in space $\mathbb{R}^3$ has the following form:

$$\mathbf{f}(u) = (f(u), g(u), h(u)), \qquad u \in [a, b] \subset \mathbb{R},$$

where f, g and h are real-valued functions. Thus, $\mathbf{f}(u)$ maps a real value u in the closed interval $[a, b]$ to a point in space $\mathbb{R}^3$.

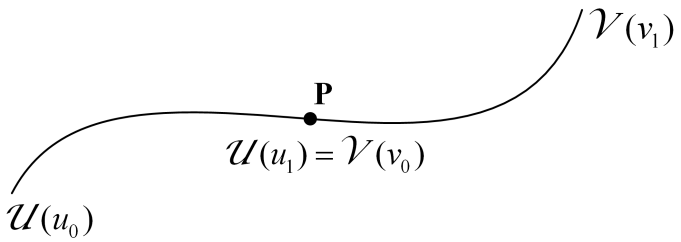


Figure 2.1: C^n continuity at the joint **p**

Geometric and parametric continuity

Definition 1. A parametric curve $\mathcal{C}(u)$ is said to be C^n *continuous*, if it is n-times continuously differentiable with respect to parameter u, which means the first to n-th derivatives of the curve with respect to the parameter u are continuous.

For a piecewise curve consisting of two segments $\mathcal{U}(u), u \in [u_0, u_1]$ and $\mathcal{V}(v), v \in [v_0, v_1]$ (as shown in Fig. 2.1) connected with C^n continuity at the joint point **p**, this requires

$$\left.\frac{d^k \mathcal{U}}{du^k}\right|_{u_1-} = \left.\frac{d^k \mathcal{V}}{dv^k}\right|_{v_0+}, \qquad k = 0, \ldots, n, \tag{2.1}$$

where $\left.\frac{d^k \mathcal{U}}{du^k}\right|_{u_1-}$ denotes the kth left-hand derivative at u_1 and $\left.\frac{d^k \mathcal{V}}{dv^k}\right|_{v_0+}$ denotes the kth right-hand derivative at v_0.

As the name implies, parametric continuity is dependent on the underlying parameter and as we know, the parametrization for a curve is not unique. Distinguished from parametric continuity, another measure for continuity based solely on geometric properties is referred to as geometric continuity, denoted as G^n *continuity*. In contrast to parametric continuity, geometric continuity is *intrinsic* and *parameter independent*.

Definition 2. The curve is called G^n continuous if the curve is C^n continuous when reparametrized with respect to the arc length s. For a curve $\mathcal{C}(u), u \in [a, b]$, the arc length $s(u)$ is defined as

$$s = \int_a^u \|\mathcal{C}'(u)\| du. \tag{2.2}$$

With specific n, G^n continuity is as follows:

- G^0: The curve is continuous.
- G^1: The curve is continuous and the curve has continuously varying unit tangent, however left and right limit of the magnitude of the tangent vector may jump.

- G^2: The curve is at first G^1 continuous and the curve has continuously varying signed curvature.

The derivatives with respect to parameter u and arc length s are shown in Fig. 2.2.
Here we will illustrate the relation between geometric and parametric continuity by means of two examples. As illustrated in Fig. 2.3, three points $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ are collinear and the straight line consisting of two segments $\mathcal{U}$ and $\mathcal{V}$ is parametrized in the following form

$$\mathcal{U} = \mathbf{p}_1 + (\mathbf{p}_2 - \mathbf{p}_1)u \quad \rightarrow \quad \frac{d\mathcal{U}}{du} = \mathbf{p}_2 - \mathbf{p}_1 \qquad u \in [0,1] \tag{2.3}$$

$$\mathcal{V} = \mathbf{p}_2 + (\mathbf{p}_3 - \mathbf{p}_2)v, \quad \rightarrow \quad \frac{d\mathcal{V}}{dv} = \mathbf{p}_3 - \mathbf{p}_2 \qquad v \in [0,1] \tag{2.4}$$

If $\mathbf{p}_3 - \mathbf{p}_2 \neq \mathbf{p}_2 - \mathbf{p}_1$, the straight line is not C^1 continuous. Therefore whether it is C^1 or not depends on the parametrization, however it is G^1 continuous, since $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ are collinear.
Another example is shown in Fig. 2.4 where the curve consists of two quarter arcs joined at point $\mathbf{p}$. The curve segments $\mathcal{U}$ and $\mathcal{V}$ are parametrized with respect to θ in the following way:

$$\begin{aligned} \mathcal{U} &= \begin{bmatrix} r_2\cos(\theta) \\ r_2\sin(\theta) + r_1 - r_2 \end{bmatrix}, && \theta \in [\tfrac{\pi}{2}, \pi] \\ \mathcal{V} &= \begin{bmatrix} r_1\cos(\theta) \\ r_1\sin(\theta) \end{bmatrix}, && \theta \in [0, \tfrac{\pi}{2}]. \end{aligned} \tag{2.5}$$

Since $\frac{d\mathcal{U}}{d\theta}\big|_{\theta=\frac{\pi}{2}} = [-r_2 \ 0]^T$ is not equal to $\frac{d\mathcal{V}}{d\theta}\big|_{\theta=\frac{\pi}{2}} = [-r_1 \ 0]^T$, the curve is not C^1 continuous at the joint $\mathbf{p}$. Then we reparametrize the curve with respect to the arc length s as follows:

$$\begin{aligned} \mathcal{U} &= \begin{bmatrix} r_2\cos(\frac{s}{r_2} + \frac{\pi}{2}) \\ r_2\sin(\frac{s}{r_2} + \frac{\pi}{2}) + r_1 - r_2 \end{bmatrix}, && s \in [0, \tfrac{\pi r_2}{2}] \\ \mathcal{V} &= \begin{bmatrix} r_1\cos(\frac{s}{r_1}) \\ r_1\sin(\frac{s}{r_1}) \end{bmatrix}, && s \in [0, \tfrac{\pi r_1}{2}]. \end{aligned} \tag{2.6}$$

With the arc length parametrization it is easy to get that $\frac{d\mathcal{U}}{ds}\big|_{s=0} = \frac{d\mathcal{V}}{ds}\big|_{s=\frac{\pi r_1}{2}} = [-1 \ 0]^T$, therefore the curve is G^1 continuous.

Geometric continuity can be taken as parametric continuity with respect to a very special parametrization, the arc length parametrization. Parametric continuity of order

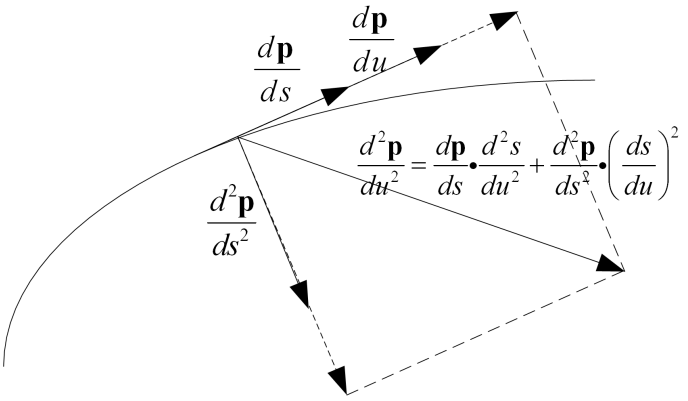


Figure 2.2: The relation between derivatives with respect to s and u

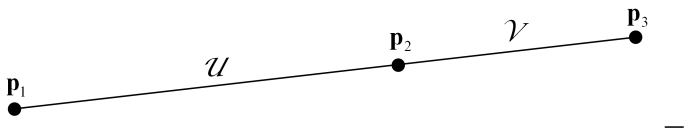


Figure 2.3: A straight line which is G^1 but not C^1 continuous

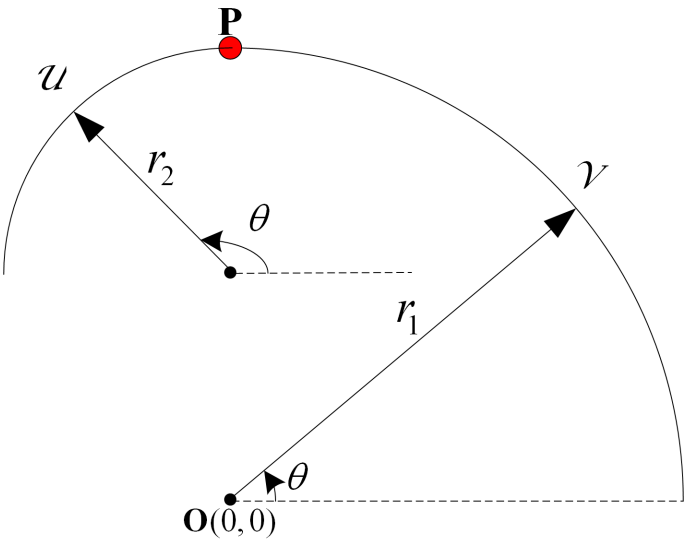


Figure 2.4: A curve which is G^1 but not C^1 continuous

n implies geometric continuity of order n, but not vice-versa, therefore parametric continuity is the stronger condition. In the field of CAGD or computer graphics, geometric continuity is usually adopted to evaluate the geometry or shape, since it it intrinsic and parameter independent. While in some other applications, for instance considering motion along a curve, it is not enough for the path to be smooth, parametric continuity is often investigated to guarantee the continuity of the velocity and acceleration vector.

2.2 Spline curves

The word *spline* originally comes from the ship building industry and describes a useful tool used by draftsmen in laying out curved ship hull contours. Nowadays spline curves are widely used in the fields of computer-aided geometric design, due to the following advantages:

- Flexibility and ease to manipulate the shape.
- A spline curve can be broken into segments and modified locally.
- There are efficient and numerically stable algorithms to evaluate the spline curve.

2.2.1 Bézier curves

We first start with the *Bézier curve*, which is a particular case of the spline curve.

Definition 3. A *Bézier curve* of degree n is defined by $n+1$ control points $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$ as

$$\mathbf{f}(u) = \sum_{i=0}^{n} \mathbf{d}_i B_i^n(u), \tag{2.7}$$

in which $B_i^n(u)$ is referred to as *Bézier basis function* or *Bernstein basis polynomial*:

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}, \tag{2.8}$$

satisfying the recursive formula:

$$B_i^{n+1}(u) = uB_{i-1}^n(u) + (1-u)B_i^n(u), \tag{2.9}$$

with $B_{-1}^n = B_{n+1}^n = 0$ and $B_0^0 = 1$.

The Bernstein basis polynomials have the following property:

- Non-negativity: all Bernstein basis polynomials $B_i^n(u)$ are non-negative.
- Partition of unity: the sum of the Bernstein basis polynomials at any u is 1.

Derived from these properties of the Bernstein basis polynomials, we can get the following properties of a Bézier curve:

- Affine invariance: any point $\mathbf{f}(u)$ is an *affine combination* of the control points since the Bernstein polynomials sum to 1. As a consequence, the Bézier curve is *affinely invariant*.
- Convex hull property: the Bézier curve lies completely in the convex hull of the control points since the Bernstein polynomials are non-negative.

When designing complex shapes, more control points are needed to increase the degree of freedom. In consequence, the degree of the Bézier curve also gets higher, which is undesirable since, as the degree increases, the computation complexity increases and curves of high degree are more sensitive to round-off errors. An alternative solution is to connect Bézier curve segments of lower degree smoothly to form a *piecewise Bézier curve*. To combine the curves segments, the control points must be chosen appropriately to satisfy a certain C^n continuity at the junctions. However, maintaining this C^n (even C^1) continuity condition may be tedious and difficult. This disadvantage can be overcome by working with spline curves in a better representation. In addition, a spline curve requires much fewer control points than a piecewise Bézier curve.

2.2.2 Basics about spline curves

Spline curves are piecewise polynomials which are joined together in a smooth fashion. There are various representations of spline curves. Similar to the Bézier representation of polynomial curves, it is desirable to write a spline curve as a linear combination of *basis functions* called *B-splines*, in which the coefficients of B-splines are also known as *control points*. Before specifying spline curves, we begin with the introduction of *knot* sequences.

Definition 4. A knot sequence $T = T_{m,n} = \{t_1, \ldots, t_{m+n+1}\}$ of a spline curve of degree m with n control points is a finite set of nondecreasing values, i.e. knots, satisfying $t_1 \leq \cdots \leq t_{n+m+1}$ and $t_j < t_{j+m+1}$, for $j = 1, \ldots, n$.

Remark 1. Some comments on the knot sequence T:

- In order to facilitate the implementation in Matlab, we adopt indexing from 1.
- To be strictly formal, a knot sequence is really a multiset, which is a set with repeated items. We do not really need that formalization here, so we use the knot sequence, which is more intuitive.
- If $t_{j-1} < t_j = \ldots = t_{j+k-1} < t_{j+k}$, for $k > 0$, the *multiplicity* of the knot(s) $t_j = \ldots = t_{j+k-1}$ is k and the knot t_j is also called *k-fold* knot.
- The notation $T^*_{m,n}$ is used in the thesis for the knot sequence $T_{m,n}$ with $m+1$-fold end knots, i.e. $t_1 = \cdots = t_{m+1}$ and $t_{n+1} = \cdots = t_{n+m+1}$.
- If T is written as $T = \{b_1^{m_1}, \ldots b_s^{m_s}\}$, where b_j, $j = 1, \ldots, s$, are the distinct elements in T sorted in ascending order and m_j, $j = 1, \ldots, s$, are the corresponding multiplicities, then b_j, $j = 1, \ldots, s$, are called the *break points* of the knot sequence and the spline curve consists of $s - 1$ polynomial pieces.

Definition 5. For $u \in [t_{m+1}, t_{n+1})$, the B-spline basis functions $N_i^k(u)$ are given by the following de Boor recursive formula:

- For $k = 0$:

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad t_i \le u < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n+m \tag{2.10}$$

- For $k = 1, \ldots, m$:

$$N_i^k(u) = \frac{u - t_i}{t_{i+k} - t_i} N_i^{k-1}(u) + \frac{t_{i+k+1} - u}{t_{i+k+1} - t_{i+1}} N_{i+1}^{k-1}(u), \qquad i = 1, \ldots, n+m-k. \tag{2.11}$$

Given the control points $\mathbf{d}_i \in \mathbb{R}^d, i = 1, \ldots, n$, (usually $d = 2, 3$) and the knot sequence $T = T_{m,n}$, the spline curve $S_m\mathbf{d}$ of degree m in B-spline representation is defined as:

$$S_m\mathbf{d}(u) = \sum_{i=1}^{n} \mathbf{d}_i N_i^m(u|T), \tag{2.12}$$

in which $N_i^m(u)$ is the i-th B-spline basis function of degree m.

The B-spline basis is well conditioned and has many nice properties which usually lead to stable and simple algorithms, see Schoenberg [28] and de Boor [5]. The important properties are listed as follows:

- Local support: $N_i^m(u) = 0,\ u \notin [t_i, t_{i+m+1})$.
- Non-negativity: $N_i^m(u) \geq 0,\ u \in [t_1, t_{m+n+1}]$.
- Partition of unity: $\sum_{i=\mu-m}^{\mu} N_i^m(u) = 1,\ u \in [t_\mu, t_{\mu+1})$.

As a consequence, the spline curve has the following important properties:

- Strong convex hull property: a spline curve is contained in the convex hull of its control polygon. More specifically, if u lies in the knot interval $[t_\mu, t_{\mu+1})$, then the curve $S_m\mathbf{d}(u)$ is contained in the convex hull of the control points $\mathbf{d}_{\mu-m}, \mathbf{d}_{\mu-m+1}, \ldots, \mathbf{d}_\mu$.
- Local modification property: the control point $\mathbf{d}_i$ only affects the curve $S_m\mathbf{d}$ on the interval $[t_i, t_{i+m+1})$.
- Differentiability and smoothness: the curve $S_m\mathbf{d}(u)$ is *at least* C^{m-k} continuous at a knot of multiplicity k.
- End interpolation: with $m+1$-fold knots at both ends, the spline curve $S_m\mathbf{d}(u)$ passes through the end control points $\mathbf{d}_1$ and $\mathbf{d}_n$. A curve with end points interpolation is often called a *clamped spline curve*.
- A Bézier curve of degree m can be taken as a special case of a spline curve of degree m with the knot sequence $T = \{t_1 = \cdots = t_m = t_{m+1} < t_{m+2} = \cdots = t_{2m+2}\}$.

The matrix representation of B-splines are described in the following theorem:

Theorem 1. *[18] Let $T = T_{m,n}$ be a knot sequence of degree m, let μ be an integer such that $t_\mu < t_{\mu+1}$ and $m+1 \leq \mu \leq n$. For each positive integer k with $k \leq m$, define the matrix $\boldsymbol{R}_k^\mu(x) = \boldsymbol{R}_k(x) \in \mathbb{R}^{k\times(k+1)}$ by*

$$\boldsymbol{R}_k(x) = \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu+1-k}} & \frac{x-t_{\mu+1-k}}{t_{\mu+1}-t_{\mu+1-k}} & 0 & \cdots & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_{\mu+2-k}} & \frac{x-t_{\mu+2-k}}{t_{\mu+2}-t_{\mu+2-k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{t_{\mu+k}-x}{t_{\mu+k}-t_\mu} & \frac{x-t_\mu}{t_{\mu+k}-t_\mu} \end{pmatrix} \tag{2.13}$$

Then for u in the interval $[t_\mu, t_{\mu+1})$, the $m+1$ B-splines $\{N_j^m(u|T)\}_{j=\mu-m}^{\mu}$ of degree m that are nonzero on this interval can be written as

$$\mathbf{N}^m = (N_{\mu-m}^m, N_{\mu-m+1}^m, \ldots, N_\mu^m) = \mathbf{R}_1(u)\mathbf{R}_2(u)\cdots\mathbf{R}_m(u). \tag{2.14}$$

If u is restricted to the interval $[t_\mu, t_{\mu+1})$, then $S_m\mathbf{d}(u) = \sum_{j=1}^{n} \mathbf{d}_j N_j^m$ is given by

$$S_m\mathbf{d}(u) = \mathbf{R}_1(u)\mathbf{R}_2(u)\cdots\mathbf{R}_m(u)\mathbf{d}^m, \tag{2.15}$$

where the matrix $\mathbf{d}^m \in \mathbb{R}^{(m+1)\times d}$ is given by $\mathbf{d}^m = (\mathbf{d}_{\mu-m}, \mathbf{d}_{\mu-m+1}, \ldots, \mathbf{d}_\mu)^T$.

2.2.3 Derivatives of spline curves

To compute the derivatives of a spline curve, we first consider the derivatives of the B-spline basis functions:

Lemma 2. *Let $T = T_{m,n}$ be a knot sequence and for $u \in [t_\mu, t_{\mu+1})$ satisfying $t_\mu < t_{\mu+1}$ and $m+1 \le \mu \le n$, we have*

$$\frac{d}{du}N_j^m(u) = \frac{m}{t_{j+m}-t_j}N_j^{m-1}(u) - \frac{m}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}(u), \qquad j = 1, \ldots, n. \tag{2.16}$$

By inserting (2.16) into the definition of a spline curve (2.12), we get the derivative of a spline curve:

$$\begin{aligned}
\frac{dS_m\mathbf{d}(u)}{du} &= \sum_{j=1}^{n}\mathbf{d}_j\left[\frac{m}{t_{j+m}-t_j}N_j^{m-1}(u) - \frac{m}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}(u)\right] \\
&= m\sum_{j=1}^{n}\frac{\mathbf{d}_j}{t_{j+m}-t_j}N_j^{m-1}(u) - m\sum_{j=2}^{n+1}\frac{\mathbf{d}_{j-1}}{t_{j+m}-t_j}N_j^{m-1}(u) \\
&= \underbrace{m\frac{\mathbf{d}_1}{t_{m+1}-t_1}N_1^{m-1}(u)}_{=0} + m\sum_{j=2}^{n}\frac{(\mathbf{d}_j-\mathbf{d}_{j-1})}{t_{j+m}-t_j}N_j^{m-1}(u) \\
&\quad \underbrace{- m\frac{\mathbf{d}_n}{t_{n+m+1}-t_{n+1}}N_{n+1}^{m-1}(u)}_{=0} \\
&= \sum_{j=1}^{n-1}\frac{m(\mathbf{d}_{j+1}-\mathbf{d}_j)}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}(u).
\end{aligned} \tag{2.17}$$

Theorem 3. *Let $T = T_{m,n}$ be a knot sequence and for $u \in [t_\mu, t_{\mu+1})$ satisfying $t_\mu < t_{\mu+1}$ and $m+1 \le \mu \le n$, we have*

$$\frac{dS_m \boldsymbol{d}(u)}{du} = \sum_{j=1}^{n-1} \boldsymbol{d}'_j N_{j+1}^{m-1}(u|T), \tag{2.18}$$

where

$$\boldsymbol{d}'_j = \frac{m}{t_{j+m+1} - t_{j+1}}(\boldsymbol{d}_{j+1} - \boldsymbol{d}_j). \tag{2.19}$$

If $T = T^*_{m,n}$, then for $u \in [t_{m+1}, t_{n+1})$, $N_{j+1}^{m-1}(u|T)$ evaluated on the original knot sequence $T = T^*_{m,n}$ is equal to $N_j^{m-1}(u|T')$ on the new knot sequence $T' = T^*_{m-1,n-1}$, i.e. $N_{j+1}^{m-1}(u|T) = N_j^{m-1}(u|T')$. Therefore, on the new knot sequence T' the derivative of a B-spline curve can be written as the following:

$$\frac{dS_m \mathbf{d}(u|T)}{du} = \sum_{j=1}^{n-1} \mathbf{d}'_j N_j^{m-1}(u|T') = S_{m-1}\mathbf{d}'(u|T'). \tag{2.20}$$

Therefore, the derivative of a spline curve is another spline curve of degree $m-1$ with a new set of control points $\{\mathbf{d}'_j\}_{j=1}^{n-1}$ and a knot sequence that discards one copy of the first and last knot. In the following theorem, the r-th derivative of a spline curve in matrix form is given.

Theorem 4. *[18] Let $T = T_{m,n}$ be a knot sequence for a spline curve of degree m and let μ be an integer such that $t_\mu < t_{\mu+1}$ and $m+1 \le \mu \le n$. For $u \in (t_\mu, t_{\mu+1})$, the r-th ($r \le m$) derivative of the vector of B-splines $\boldsymbol{N}^m(u) = (N^m_{\mu-m}(u), \ldots, N^m_\mu(u))$ is given by*

$$\frac{d^r}{du^r}\boldsymbol{N}^m(u) = \frac{m!}{(m-r)!}\boldsymbol{N}^{m-r}(u)\boldsymbol{D}_{m-r+1}\cdots\boldsymbol{D}_m. \tag{2.21}$$

Then the r-th derivative of $S_m\boldsymbol{d}(u) = \sum_{i=1}^{n} \boldsymbol{d}_i N_i^m(u)$ is given by

$$\frac{d^r}{du^r} S_m \boldsymbol{d}(u) = \frac{m!}{(m-r)!}\boldsymbol{R}_1(u)\cdots\boldsymbol{R}_{m-r}(u)\boldsymbol{D}_{m-r+1}\cdots\boldsymbol{D}_m\boldsymbol{d}^m, \tag{2.22}$$

where $\boldsymbol{D}_k$ denotes the matrix obtained by differentiating each entry in $\boldsymbol{R}_k(x)$ with respect to x,

$$\boldsymbol{D}_k = \frac{d\boldsymbol{R}_k(x)}{dx} = \begin{pmatrix} \frac{-1}{t_{\mu+1}-t_{\mu+1-k}} & \frac{1}{t_{\mu+1}-t_{\mu+1-k}} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \frac{-1}{t_{\mu+k}-t_\mu} & \frac{1}{t_{\mu+k}-t_\mu} \end{pmatrix}. \tag{2.23}$$

2.2.4 Integral of spline curves

Derived from the differentiation formula in Lemma 2 and Theorem 3, the indefinite integral of B-splines and spline curves can be given in explicit form, see [6] and [27]. In our application, we often need to compute integrals of the type:

$$\int_{\mathbb{R}} \left| \frac{d^r}{du^r} S_m \mathbf{d}(u|T) \right|^2 du = \mathbf{d}^T \left(\int_{\mathbb{R}} \left(\mathbf{N}^{(r)} \right)^T \mathbf{N}^{(r)} \, du \right) \mathbf{d}, \tag{2.24}$$

$$\text{in which} \quad \mathbf{N}^{(r)} = \left(\frac{d^r}{du^r} N_1^m(u|T), \cdots, \frac{d^r}{du^r} N_n^m(u|T) \right) \qquad (r = 1, 2, 3),$$
$$\text{and} \quad \mathbf{d} = (\mathbf{d}_1, \cdots, \mathbf{d}_n)^T \in \mathbb{R}^{n \times d},$$

in the context of least squares approximation. Since the integrand in (2.24) still consists of piecewise polynomials, the integration problem can be solved numerically by *Gaussian quadrature*.

Gaussian quadrature

An l-node Gaussian quadrature rule allows to integrate polynomials of order up to $2l-1$ exactly. The quadrature rule has the form

$$\int_{a_1}^{a_2} f(x) w(x) \, dx = \sum_{k=1}^{l} w_k f(x_k).$$

If $w(x) = 1$ and $[a_1, a_2] = [-1, 1]$, the quadrature rule is called *Gauss-Legendre Quadrature*, where the nodes are the roots of *Legendre polynomials*. The approximate nodes and weights up to $l = 4$ are listed in the table below.

l	x_k	w_k
1	0	2
2	± 0.577350269	1
3	0	0.888888889
	±0.774596669	0.555555556
4	±0.861136312	0.347854845
	±0.339981043	0.652145155

We then extend the quadrature for $f(x)$ over $[a_1, a_2]$. With $x = \frac{a_2 - a_1}{2} t + \frac{a_2 + a_1}{2}$, we get

$$\int_{a_1}^{a_2} f(x) \, dx = \int_{-1}^{1} f(\frac{a_2 - a_1}{2} t + \frac{a_2 + a_1}{2}) \frac{a_2 - a_1}{2} \, dt,$$

therefore the nodes $\hat{x}_k$ and weights $\hat{w}_k$ over $[a_1, a_2]$ are given as:

$$\begin{aligned}\hat{x}_k &= \frac{a_2 - a_1}{2} x_k + \frac{a_2 + a_1}{2}, \\ \hat{w}_k &= \frac{a_2 - a_1}{2} w_k.\end{aligned}$$

Back to solving the problem (2.24), the product of two B-splines of degree $m - r$ is a piecewise polynomial function of degree $2(m - r)$. Thus the number of nodes needed to compute (2.24) exactly is determined by $2l - 1 \geq 2(m - r)$. Therefore $l = m - r + 1$ is sufficient. The functional (2.24) can be written as:

$$\begin{aligned}& \mathbf{d}^T \left(\sum_{i=1}^{s-1} \int_{b_i}^{b_{i+1}} \mathbf{N}^{(r)}(u)^T \mathbf{N}^{(r)}(u)\, du \right) \mathbf{d} \\ = \;& \mathbf{d}^T \left(\sum_{i=1}^{s-1} \sum_{k=1}^{l} \mathbf{N}^{(r)}(\hat{x}_{ik})^T \hat{w}_{ik} \mathbf{N}^{(r)}(\hat{x}_{ik}) \right) \mathbf{d} \\ =: \;& \mathbf{d}^T \mathbf{M} \mathbf{d},\end{aligned}$$

where b_i are the break points of the knots sequence, $\hat{x_{ik}} = \frac{b_{i+1} - b_i}{2} x_k + \frac{b_{i+1} + b_i}{2}$ and $\hat{w}_{ik} = \frac{b_{i+1} - b_i}{2} w_k$ ($i = 1, \ldots, s-1$, $k = 1, \ldots, l$), in which x_k, w_k are the nodes and weights of l-node Gaussian quadrature over $[-1, 1]$.

We can get $\mathbf{M} = \mathbf{C}^T \mathbf{W} \mathbf{C}$, where $\mathbf{C}$ is the collocation matrix of the r-th derivatives:

$$\mathbf{C} = \left(\begin{array}{cccc} N_1^{(r)}(\hat{x}_{11}) & N_2^{(r)}(\hat{x}_{11}) & \cdots & N_n^{(r)}(\hat{x}_{11}) \\ N_1^{(r)}(\hat{x}_{12}) & N_2^{(r)}(\hat{x}_{12}) & \cdots & N_n^{(r)}(\hat{x}_{12}) \\ & \vdots & & \\ N_1^{(r)}(\hat{x}_{1l}) & N_2^{(r)}(\hat{x}_{1l}) & \cdots & N_n^{(r)}(\hat{x}_{1l}) \\ \hline N_1^{(r)}(\hat{x}_{21}) & N_2^{(r)}(\hat{x}_{21}) & \cdots & N_n^{(r)}(\hat{x}_{21}) \\ N_1^{(r)}(\hat{x}_{22}) & N_2^{(r)}(\hat{x}_{22}) & \cdots & N_n^{(r)}(\hat{x}_{22}) \\ & \vdots & & \\ N_1^{(r)}(\hat{x}_{2l}) & N_2^{(r)}(\hat{x}_{2l}) & \cdots & N_n^{(r)}(\hat{x}_{2l}) \\ \hline & \vdots & & \\ & \vdots & & \\ \hline N_1^{(r)}(\hat{x}_{s'1}) & N_2^{(r)}(\hat{x}_{s'1}) & \cdots & N_n^{(r)}(\hat{x}_{s'1}) \\ N_1^{(r)}(\hat{x}_{s'2}) & N_2^{(r)}(\hat{x}_{s'2}) & \cdots & N_n^{(r)}(\hat{x}_{s'2}) \\ & \vdots & & \\ N_1^{(r)}(\hat{x}_{s'l}) & N_2^{(r)}(\hat{x}_{s'l}) & \cdots & N_n^{(r)}(\hat{x}_{s'l}) \end{array} \right), \quad \text{in which } s' = s - 1,$$

and $\mathbf{W} = \text{diag}(\hat{w}_{11}, \hat{w}_{12}, \ldots, \hat{w}_{1l}, \hat{w}_{21}, \hat{w}_{22}, \ldots, \hat{w}_{2l}, \ldots, \hat{w}_{s'1}, \hat{w}_{s'2}, \ldots, \hat{w}_{s'l})$.

With Gaussian quadrature, the integral of spline curves can be reduced to the problem of evaluating the spline curves at certain $\hat{x}_{ik}$.

2.2.5 Knot insertion

Knot insertion refers to the process of adding new knots into the existing knot sequence without changing the curve's shape.We first consider the case of inserting one knot at a time and we can update the B-spline coefficients after each insertion via implementing *Boehm's algorithm* [2].

Algorithm 1. (Boehm's algorithm) Let $T = T_{m,n}$ be a given knot sequence and $\widetilde{T} = \{t_1, \ldots, t_\mu, \tilde{t}, t_{\mu+1}, \ldots, t_{n+m+1}\}$ be the knot sequence obtained by inserting a knot $\tilde{t}$ in T in the interval $[t_\mu, t_{\mu+1})$.
If

$$S_m\mathbf{d}(u|T) = \sum_{j=1}^{n} \mathbf{d}_j N_j^m(u|T) = \sum_{i=1}^{n+1} \widetilde{\mathbf{d}}_i N_i^m(u|\widetilde{T}) = S_m\widetilde{\mathbf{d}}(u|\widetilde{T}), \tag{2.25}$$

then $(\widetilde{\mathbf{d}}_i)_{i=1}^{n+1}$ can be expressed in terms of $(\mathbf{d}_j)_{j=1}^{n}$ through the formulas:

$$\widetilde{\mathbf{d}}_i = \begin{cases} \mathbf{d}_i & \text{if} \quad 1 \le i \le \mu - m, \\ \frac{\tilde{t}-t_i}{t_{i+m}-t_i}\mathbf{d}_i + \frac{t_{i+m}-\tilde{t}}{t_{i+m}-t_i}\mathbf{d}_{i-1} & \text{if} \quad \mu - m + 1 \le i \le \mu, \\ \mathbf{d}_{i-1} & \text{if} \quad \mu + 1 \le i \le n + 1. \end{cases} \tag{2.26}$$

By using linear algebra, the refined control points $\widetilde{\mathbf{d}}$ can be written as a product of the knot insertion matrix $\mathbf{A}$ and the original control points $\mathbf{d}$:

$$\widetilde{\mathbf{d}} = \mathbf{A}(\tilde{t})\mathbf{d}, \qquad \mathbf{A}(\tilde{t}) = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & 1-\alpha_{\mu-m+1} & \alpha_{\mu-m+1} & & & & \\ & & & \ddots & \ddots & & & \\ & & & & & 1-\alpha_\mu & \alpha_\mu & \\ & & & & & & 1 & \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix}, \tag{2.27}$$

where $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_n)^T \in \mathbb{R}^{n\times d}$, $\widetilde{\mathbf{d}} = (\widetilde{\mathbf{d}}_1, \ldots, \widetilde{\mathbf{d}}_{n+1})^T \in \mathbb{R}^{(n+1)\times d}$ and

$$\alpha_i = \frac{t_{i+m} - \tilde{t}}{t_{i+m} - t_i}, \qquad i = \mu - m + 1, \ldots, \mu.$$

By applying Boehm's algorithm repeatedly, we can compute the refined control points after inserting a sequence of knots $\tilde{t}_1, \ldots, \tilde{t}_k$:

$$\tilde{\mathbf{d}} = \mathbf{A}(\tilde{t}_1, \ldots, \tilde{t}_k)\mathbf{d}, \qquad \mathbf{A}(\tilde{t}_1, \ldots, \tilde{t}_k) = \mathbf{A}(\tilde{t}_k) \cdots \mathbf{A}(\tilde{t}_1) \in \mathbb{R}^{(n+k)\times n}. \tag{2.28}$$

Instead of inserting knots sequentially using Boehm's algorithm, we can insert knots simultaneously using the *Oslo algorithm*, which is based on the recurrence of *discrete B-splines* [9, 18]. Especially when we insert a large number of knots, the Oslo algorithm is more efficient.

Algorithm 2. [18] (Oslo algorithm I) Assume that the two knot sequences $T = \{t_1, \ldots, t_{n_1+m+1}\}$ and $\widetilde{T} = \{\tilde{t}_1, \ldots, \tilde{t}_{n_2+m+1}\}$ with common knots at the ends are given and $T \subset \widetilde{T}$. To compute the knot insertion matrix $\mathbf{A}^m(T, \widetilde{T}) = (a_{j,m}(i))_{i,j=1}^{n2,n1} \in \mathbb{R}^{n_2 \times n_1}$ from T to $\widetilde{T}$ such that $\widetilde{\mathbf{d}} = \mathbf{A}^m(T, \widetilde{T})\mathbf{d}$, we perform the following steps:
For $i = 1, \ldots, n_2$,

1. Determine μ such that $t_\mu \leq \tilde{t}_i < t_{\mu+1}$
2. Compute entries $\mu - m, \ldots, \mu$ of row i by evaluating

$$\mathbf{a}_m(i) = (a_{\mu-m,m}(i), \ldots, a_{\mu,m}(i)) = \mathbf{R}_1(\tilde{t}_{i+1}) \cdots \mathbf{R}_m(\tilde{t}_{i+m})$$

All other entries in row i are zero. The entry $\mathbf{R}_k(x)$ is defined in (2.13).

Knot insertion is a basic and important strategy for spline curves which can be applied to decompose the spline curve to enable local control and to refine the knot sequence to enhance the flexibility and increase the degrees of freedom. Knot insertion can also be used to compare two spline curves with different knot sequences, which will be further discussed in Section 3.4.1.

2.2.6 De Boor's algorithm

De Boor's algorithm provides a fast and numerically stable way to evaluate a spline curve, which can also be realized via artificial knot insertion. If we want to evaluate the spline curve $S_m\mathbf{d}$ at the parameter value u, we can insert the knot u several times to make its multiplicity m and the last generated new control point is the point on the curve that corresponds to the parameter u.

Algorithm 3. (De Boor's algorithm):

- Input: the parameter u
- Output: the point in the curve $S_m\mathbf{d}(u)$
- If u lies in $[t_\mu, t_{\mu+1})$ and $u \neq t_\mu$, then u is inserted m times. If $u = t_\mu$ and the multiplicity of t_μ is m_μ, then u needs to be inserted $m - m_\mu$ times. The affected control points $\mathbf{d}_{\mu-m_\mu}, \ldots, \mathbf{d}_{\mu-m}$ are renamed as $\mathbf{d}^0_{\mu-m_\mu}, \ldots, \mathbf{d}^0_{\mu-m}$.

  for $r = 1$ to $m - m_\mu$ **do**
  for $i = \mu - m + r$ to $\mu - m_\mu$ **do**
  Let

$$\beta_i^r = \frac{u - t_i}{t_{i+m-r+1} - t_i},$$

$$\mathbf{d}_i^r = (1 - \beta_i^r)\mathbf{d}_{i-1}^{r-1} + \beta_i^r \mathbf{d}_i^{r-1}.$$

  end for
  end for

$\mathbf{d}^{m-m_\mu}_{\mu-m_\mu}$ is $S_m\mathbf{d}(u)$.

De Boor's algorithm can also be used to subdivide a spline curve. Notice that the running time of the algorithm depends only on the degree m rather than on the number n of control points. After inserting the knots several times, making each of the interior knot's multiplicity m, the spline curve becomes a piecewise Bézier curve, which will be further discussed in Section 2.2.8.

2.2.7 Knot removal

Knot removal is the inverse process of knot insertion. As we know, knot insertion is a precise procedure, i.e. the spline curve after knot insertion is precisely the same as the original one without shape change. However, knot removal usually produces only an approximation of the original curve. The only exception occurs if the knot has been inserted before, in other words, if the curve is of higher order of continuity than C^{m-k} at a knot of multiplicity k. The strategies of knot removal to produce a good approximation are discussed in a number of papers [16, 17, 29].

Given the spline curve $S_m\widetilde{\mathbf{d}}(u|\widetilde{T})$, we consider the problem to determine the control points $\mathbf{d}$ of the spline curve $S_m\mathbf{d}(u|T)$ after removing the knot $\tilde{t}$ from the knot sequence

$\widetilde{T}$. To do so, we need minimize the error between the spline curves to determine the control points $\mathbf{d}$:

$$\min_{\mathbf{d}} \|S_m\mathbf{d}(\cdot|T) - S_m\widetilde{\mathbf{d}}(\cdot|\widetilde{T})\|. \tag{2.29}$$

Since the minimization functional (2.29) is continuous and hard to evaluate, it is preferable to minimize the error in a discrete form. There are various ways to choose the minimization functional and the norm of the error function, but a simple choice is to minimize the error between the control points $\widetilde{\mathbf{d}}$ and $\mathbf{A}(\tilde{t})\mathbf{d}$ in L^2-norm:

$$\min_{\mathbf{d}} \|\mathbf{A}(\tilde{t})\mathbf{d} - \widetilde{\mathbf{d}}\|^2 = \min_{\mathbf{d}} \mathbf{d}^T\mathbf{A}^T\mathbf{A}\mathbf{d} - 2\widetilde{\mathbf{d}}^T\mathbf{A}\mathbf{d} + \widetilde{\mathbf{d}}^T\widetilde{\mathbf{d}}, \tag{2.30}$$

which can be reduced to solving the normal equation:

$$\mathbf{A}^T\mathbf{A}\mathbf{d} = \mathbf{A}^T\widetilde{\mathbf{d}} \tag{2.31}$$

The implementation of knot removal in this thesis is restricted to remove the knots inserted previously in the context of degree elevation. This can be performed precisely and will be discussed in the next section.

2.2.8 Degree elevation

In some applications we require two spline curves to have the same degree to facilitate combining curve segments or comparison between curves. *Degree elevation* of a spline curve refers to the process of representing a spline curve using basis functions of a higher degree. The spline curve after degree elevation has the same parametrization and geometry as the original one. There are various methods to raise the degree of the spline curve [24, 25, 4, 15] and here we discuss the algorithm introduced by Piegl and Tiller [22].

Degree elevation of Bézier curves

We will first introduce the degree elevation of Bézier curves, since it is involved in raising the degree of spline curves.

Assume that a Bézier curve of degree k defined by $k+1$ control points $\mathbf{d}_1^{[k]}, \ldots, \mathbf{d}_{k+1}^{[k]}$ and we want to increase the degree of this curve to $k+1$. Since a Bézier curve of degree $k+1$ is defined by $k+2$ control points, we need to find such a new set of control

points $\mathbf{d}_1^{[k+1]}, \ldots, \mathbf{d}_{k+2}^{[k+1]}$. The new control points after degree elevation are computed as follows:

$$\mathbf{d}^{[k+1]} = \begin{pmatrix} 1 & & & \\ \frac{1}{k+1} & \frac{k}{k+1} & & \\ & \ddots & \ddots & \\ & & \frac{k}{k+1} & \frac{1}{k+1} \\ & & & 1 \end{pmatrix} \mathbf{d}^{[k]} =: \mathbf{E}_k \mathbf{d}^{[k]}, \tag{2.32}$$

in which $\mathbf{d}^{[k+1]} = \left(\mathbf{d}_1^{[k+1]}, \ldots, \mathbf{d}_{k+2}^{[k+1]}\right)^T \in \mathbb{R}^{(k+2)\times d}$, $\mathbf{d}^{[k]} = \left(\mathbf{d}_1^{[k]}, \ldots, \mathbf{d}_{k+1}^{[k]}\right)^T \in \mathbb{R}^{(k+1)\times d}$ and $\mathbf{E}_k \in \mathbb{R}^{(k+2)\times(k+1)}$.

With (2.32), we can derive that the control points $\overline{\mathbf{d}}^{[k]} = \left(\overline{\mathbf{d}}_1^{[k]}, \ldots, \overline{\mathbf{d}}_{lk+1}^{[k]}\right)^T \in \mathbb{R}^{(lk+1)\times d}$ for a piecewise Bézier curve consisting of l pieces of Bézier segments of degree k are updated as follows:

$$\overline{\mathbf{d}}^{[k+1]} = \begin{pmatrix} 1 & & & & & & & \\ \frac{1}{k+1} & \frac{k}{k+1} & & & & & & \\ & \ddots & \ddots & & & & & \\ & & \frac{k}{k+1} & \frac{1}{k+1} & & & & \\ & & & 1 & & & & \\ & & & \frac{1}{k+1} & \frac{k}{k+1} & & & \\ & & & & \ddots & \ddots & & \\ & & & & & \frac{k}{k+1} & \frac{1}{k+1} & \\ & & & & & & 1 & \\ & & & & & & & \ddots \end{pmatrix} \overline{\mathbf{d}}^{[k]} =: \overline{\mathbf{E}}_k \overline{\mathbf{d}}^{[k]}, \tag{2.33}$$

where $\overline{\mathbf{d}}^{[k+1]} = \left(\overline{\mathbf{d}}_1^{[k+1]}, \ldots, \overline{\mathbf{d}}_{lk+l+1}^{[k+1]}\right)^T \in \mathbb{R}^{(lk+l+1)\times d}$, $\overline{\mathbf{E}}_k \in \mathbb{R}^{(lk+l+1)\times(lk+1)}$ and the number of the repeated pattern in $\overline{\mathbf{E}}_k$ is equal to the number of Bézier segments l.

Degree elevation of spline curves

Given a spline curve of degree m,

$$S_m \mathbf{d}(u|T) = \sum_{i=1}^{n} \mathbf{d}_i N_i^m(u|T). \tag{2.34}$$

To illustrate the variation of the knot sequence more clearly during degree elevation, the knot sequence T is written in the following form:

$$T = \{\underbrace{b_1, \ldots, b_1}_{m_1=m+1}, \underbrace{b_2, \ldots, b_2}_{m_2}, \ldots, \underbrace{b_{s-1}, \ldots, b_{s-1}}_{m_{s-1}}, \underbrace{b_s, \ldots, b_s}_{m_s=m+1}\}, \tag{2.35}$$

in which the end knots have multiplicity $m+1$ and interior break points b_i have multiplicity m_i. We raise the degree from m to $m+r$ and the curve after degree elevation has the form

$$S_{m+r}\widetilde{\mathbf{d}}(u|\widetilde{T}) = \sum_{i=1}^{\widetilde{n}} \widetilde{\mathbf{d}}_i N_i^{m+r}(u|\widetilde{T}), \tag{2.36}$$

with knot sequence

$$\widetilde{T} = \{\underbrace{b_1,\ldots,b_1}_{m+r+1}, \underbrace{b_2,\ldots,b_2}_{m_2+r}, \ldots, \underbrace{b_{s-1},\ldots,b_{s-1}}_{m_{s-1}+r}, \underbrace{b_s,\ldots,b_s}_{m+r+1}\} \tag{2.37}$$

and $\widetilde{n} = n + sr$.

Degree elevation can be performed in the following steps:

1. Subdivide the spline curve into piecewise Bézier curves at the break points using the Oslo or Boehm's algorithm to increase the multiplicity of interior knots to m. The piecewise Bézier curve of degree m after knot insertion can be written as

   $$S_m\widehat{\mathbf{d}}(u|\widehat{T}) = \sum_{i=1}^{\widehat{n}} \widehat{\mathbf{d}}_i N_i^m(u|\widehat{T}), \tag{2.38}$$

   with knot sequence

   $$\widehat{T} = \{\underbrace{b_1,\ldots,b_1}_{m+1}, \underbrace{b_2,\ldots,b_2}_{m}, \ldots, \underbrace{b_{s-1},\ldots,b_{s-1}}_{m}, \underbrace{b_s,\ldots,b_s}_{m+1}\}. \tag{2.39}$$

   The control points of the piecewise Bézier curve are computed as

   $$\widehat{\mathbf{d}} = \widehat{\mathbf{M}}\mathbf{d}, \tag{2.40}$$

   in which $\widehat{\mathbf{M}} = \mathbf{A}^m(T,\widehat{T})$ is the knot insertion matrix from T to $\widehat{T}$.

2. Elevate the degree of the piecewise Bézier curve .

   If we raise the degree of the piecewise Bézier curve $S_m\widehat{\mathbf{d}}(u|\widehat{T})$ from m to $m+r$, we obtain a spline curve

   $$S_{m+r}\overline{\mathbf{d}}(u|\overline{T}) = \sum_{i=1}^{\overline{n}} \overline{\mathbf{d}}_i N_i^{m+r}(u|\overline{T}), \tag{2.41}$$

   with knot sequence

   $$\overline{T} = \{\underbrace{b_1,\ldots,b_1}_{m+r+1}, \underbrace{b_2,\ldots,b_2}_{m+r}, \ldots, \underbrace{b_{s-1},\ldots,b_{s-1}}_{m+r}, \underbrace{b_s,\ldots,b_s}_{m+r+1}\}. \tag{2.42}$$

The control points are updated by

$$\overline{\mathbf{d}} = \overline{\mathbf{M}}\widehat{\mathbf{d}}, \tag{2.43}$$

in which $\overline{\mathbf{M}} = \overline{\mathbf{E}}_{m+r-1} \cdots \overline{\mathbf{E}}_m$.

3. Represent the piecewise Bézier curve as a spline curve with respect to the initial knot distribution, which can be carried out via knot removal.

   As mentioned before, a knot inserted previously can be removed precisely. From the spline curve $S_{m+r}\overline{\mathbf{d}}(u|\overline{T})$, the target curve $S_{m+r}\widetilde{\mathbf{d}}(u|\widetilde{T})$ is obtained via knot removal. Inversely we can get $S_{m+r}\overline{\mathbf{d}}(u|\overline{T})$ from the target curve $S_{m+r}\widetilde{\mathbf{d}}(u|\widetilde{T})$ by artificial knot insertion, so we have:

$$\overline{\mathbf{d}} = \mathbf{A}^{m+r}(\widetilde{T}, \overline{T})\widetilde{\mathbf{d}}. \tag{2.44}$$

   For the sake of convenience, we simplify the knot insertion matrix $\mathbf{A}^{m+r}(\widetilde{T}, \overline{T})$ as $\widetilde{\mathbf{A}}$. Since $\widetilde{\mathbf{A}}$ has full rank and $\widetilde{\mathbf{A}}^T\widetilde{\mathbf{A}}$ is non-singular, we get

$$\widetilde{\mathbf{d}} = (\widetilde{\mathbf{A}}^T\widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{A}}^T\overline{\mathbf{d}} =: \widetilde{\mathbf{M}}\overline{\mathbf{d}}. \tag{2.45}$$

If we summarize the above steps, the control points of the final spline curve are computed as

$$\widetilde{\mathbf{d}} = \widetilde{\mathbf{M}}\overline{\mathbf{M}}\overline{\mathbf{M}}\widehat{\mathbf{M}}\mathbf{d}. \tag{2.46}$$

2.2.9 Application of spline curves

A typical application of splines is to fit a curve to a given set of data points. *Curve fitting* using spline curves consists of two main categories: *spline interpolation* and *spline approximation*, which will be discussed in this section.

2.2.9.1 Parameter and knot sequence selection

Before considering spline interpolation and spline approximation, we first need to find the corresponding parameters $u_1, \ldots, u_n$ that can be assigned to the respective data points $\mathbf{p}_1, \ldots, \mathbf{p}_n$. By choosing different parameters, the shape of the curve can be influenced considerably, see [7]. Most commonly used parameter selection schemes are listed below. For simplicity, we consider the parameters on a normalized interval $[0, 1]$.

- Uniformly spaced:
  $u_k = \dfrac{k-1}{n-1},\ k = 1, \ldots, n.$
- Chordal length:
  Let $L = \sum_{i=2}^{n} \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$ and $L_k = \dfrac{\sum_{i=2}^{k} \|\mathbf{p}_i - \mathbf{p}_{i-1}\|}{L}$,
  then $u_1 = 0$ and $u_k = L_k,\ k = 2, \ldots, n.$
- Centripetal:
  Let $L = \sum_{i=2}^{n} \sqrt{\|\mathbf{p}_i - \mathbf{p}_{i-1}\|}$ and $L_k = \dfrac{\sum_{i=2}^{k} \sqrt{\|\mathbf{p}_i - \mathbf{p}_{i-1}\|}}{L}$,
  then $u_1 = 0$ and $u_k = L_k,\ k = 2, \ldots, n.$

Although the uniformly spaced method is the simplest one to implement, it totally ignores the geometry of the data points. The chordal length method is widely used and usually performs well since it represents the distribution of the data points. For motion control, it is convenient to specify the motion constraints if the curve is arc length parametrized. However a lot of parametric curves, for instance the polynomial curves (proved by R. Farouki [8]) cannot be parametrized to have unit speed, but the chordal length method can be taken as an approximation to arc length parametrization. Fig. 2.5 illustrates how the shape of the curve is affected by different methods. For the chordal length method, a longer chord may sometimes cause its curve segment to have a bulge bigger than necessary. This phenomenon can be reduced by the centripetal method introduced by Lee [14], since the impact of a longer chord on the length of the data polygon is reduced, meanwhile the impact of a shorter chord on the length of the data polygon is increased. Because of this characteristic, the centripetal method can handle the sharp turn in Fig. 2.5 better than the chord length method.

Knot selection is a key issue for spline interpolation and approximation and worth to be investigated in depth. If the knots are located at the data points, the positions of the knots can be chosen in the same way as the parameter selection, among which uniformly spaced, chordal length and centripetal methods are the most basic and most commonly used ones. However, how to choose optimal knots in general is difficult and there are probably no 'optimal' knots, which can fit all data sets well. Therefore a multitude of heuristic knot selection schemes is introduced for specific applications. To deal with our problem, we propose a knot placement scheme based on curvature characteristics, which will be described in detail in Section 3.3.1.

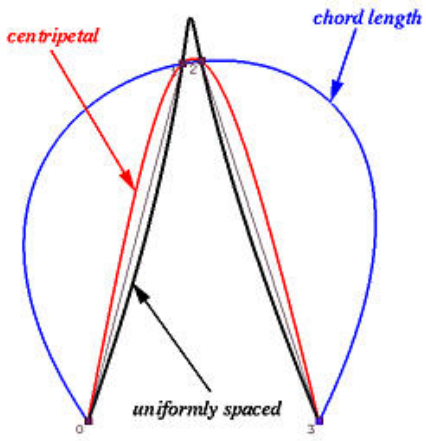


Figure 2.5: The shape of a curve is greatly influenced by parameter selection methods

2.2.9.2 Interpolation with splines

With the techniques for parameter selection and knot generation, we can obtain the set of parameter values and the knot sequence. In this section, we will discuss a general interpolation problem to find a spline curve of degree m defined by n control points that passes all n data points in the given order.

Problem 1. Given the data points $\mathbf{p}_1, \ldots, \mathbf{p}_n$, we need to find a spline curve $S_m\mathbf{d}(u|T)$ of degree m with knot sequence $T = T_{m,n}$ to interpolate the data points at the corresponding parameters $u_1, \ldots, u_n$ which are strictly increasing.

$$S_m\mathbf{d}(u_j|T) = \sum_{i=1}^{n} \mathbf{d}_i N_i^m(u_j|T) = \mathbf{p}_j, \qquad \text{for } j = 1, \ldots, n. \tag{2.47}$$

The equations in (2.47) form a system of n equations and n unknowns. The linear system of equations can be written in matrix form as:

$$\mathbf{N}\mathbf{d} = \mathbf{p}, \tag{2.48}$$

where

$$\mathbf{N} = \begin{pmatrix} N_1^m(u_1) & \cdots & N_n^m(u_1) \\ \vdots & \ddots & \vdots \\ N_1^m(u_n) & \cdots & N_n^m(u_n) \end{pmatrix} \in \mathbb{R}^{n\times n}, \ \mathbf{d} = \begin{pmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_n \end{pmatrix} \in \mathbb{R}^{n\times d}, \ \mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_n \end{pmatrix} \in \mathbb{R}^{n\times d}.$$

The matrix $\mathbf{N}$ is often referred to as *B-spline collocation matrix*. The necessary and sufficient condition for the collocation matrix to be *nonsigular* is stated in the following theorem.

Theorem 5. *The collocation matrix* $\boldsymbol{N}$ *with entries* $(N_j^m(u_i))_{i,j=1}^n$ *is nonsingular if and only if its diagonal elements are nonzero, i.e.*

$$N_i^m(u_i) \neq 0, \qquad \text{for } i = 1, \ldots, n. \tag{2.49}$$

The condition that the diagonal elements of $\mathbf{N}$ should be nonzero is known as *Schoenberg-Whitney condition*, i.e.

$$t_i < u_i < t_{i+m+1}, \qquad i = 1, \ldots, n. \tag{2.50}$$

It is important to point out the fact that the collocation matrix $\mathbf{N}$ is totally positive and is a banded matrix, with semi-bandwidth less than $m+1$ (i.e. $N_i^m(u_k) = 0$, if $|i-k| > m$), which means in any row of the matrix $\mathbf{N}$, at most $2m+1$ consecutive entries are nonzero. Thus the linear equation system (2.48) can be solved safely by Gaussian elimination *without* pivoting, see [5, Chap. XIII].

2.2.9.3 Least squares approximation with splines

In some applications, it may be sufficient and even better to find a curve which is only close enough to the given data points rather than passing through all of them. The curves obtained this way are referred to as *approximating curves*. As shown in Fig. 2.6, the interpolating curve may wiggle strongly through all data points instead of following the data polygon closely. The approximation technique can handle this wiggling problem better with properly chosen knots by relaxing the strict condition that the curve must pass through all the data points. Furthermore, approximation is more flexible than interpolation and fewer polynomial pieces are needed for approximation curves.

A measure of discrepancy which is mainly chosen to be minimized is the sum of the squares of the distances between the data points and the curve at corresponding parameters. This kind of approximation is called approximation in the sense of *least squares*.

Problem 2. Given the data points $\mathbf{p}_1, \ldots, \mathbf{p}_n$, we need to find a spline curve $S_m\mathbf{d}(u|T)$ of degree m with control points $\mathbf{d}_1, \ldots, \mathbf{d}_l$, $l \leq n$, and an appropriate knot sequence $T = T_{m,l}$ to minimize the sum of the squares of the distances between the data points and the curve at the corresponding parameters $u_1, \ldots, u_n$:

$$\min_{\mathbf{d}} \sum_{j=1}^{n} (S_m\mathbf{d}(u_j|T) - \mathbf{p}_j)^2 = \min_{\mathbf{d}} \sum_{j=1}^{n} (\sum_{i=1}^{l} \mathbf{d}_i N_i^m(u_j) - \mathbf{p}_j)^2. \tag{2.51}$$

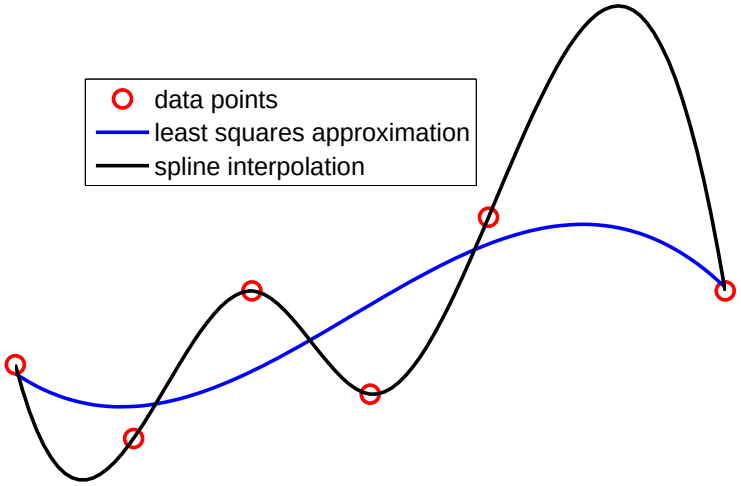


Figure 2.6: Spline interpolation vs least squares approximation

Problem 2 is equivalent to the linear least squares problem:

$$\min_{\mathbf{d}} \|\mathbf{N}\mathbf{d} - \mathbf{p}\|^2, \tag{2.52}$$

where

$$\mathbf{N} = \begin{pmatrix} N_1^m(u_1) & \cdots & N_l^m(u_1) \\ \vdots & \ddots & \vdots \\ N_1^m(u_n) & \cdots & N_l^m(u_n) \end{pmatrix} \in \mathbb{R}^{n\times l},\ \mathbf{d} = \begin{pmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_l \end{pmatrix} \in \mathbb{R}^{l\times d},\ \mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_n \end{pmatrix} \in \mathbb{R}^{n\times d}.$$

Lemma 6. *The linear least squares problem* (2.52) *always has a solution* $\boldsymbol{d}^*$ *which can be found by solving the linear set of equations, known as normal equations*

$$\boldsymbol{N}^T\boldsymbol{N}\boldsymbol{d}^* = \boldsymbol{N}^T\boldsymbol{p},$$

where the matrix $\boldsymbol{N}^T\boldsymbol{N}$ *is symmetric and positive semidefinite.*

An advantage of least squares approximation is the desired smoothing effect and less oscillation compared to interpolation methods or other approximation schemes such as the sup-norm. It is important to note that how much a curve is allowed to oscillate also depends strongly on the knot sequence. The measure of discrepancy in (2.51) only considers the errors at the data points, but the accuracy of the approximating curve to

the entire data polygon cannot be guaranteed. In our application to approximate the part programs, the spline curve is restricted to stay in the tolerance band around the data polygon. Therefore, a measure of discrepancy which only considers the errors at the data points is not sufficient. The fundamental problem to choose an appropriate minimization function and a proper knot sequence for our specific application will be discussed in detail in Chapter 3.

2.3 Equality-constrained optimization scheme

In this thesis, we will consider the equality-constrained optimization problem

$$\min f(x), \text{ subject to } c(x) = 0, \tag{2.53}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are smooth functions.
The *Lagrangian function* for problem (2.53) is defined as $\Lambda(x, \mu) = f(x) - \mu^T c(x)$ and $\mu = [\mu_1, \mu_2, \ldots, \mu_m]$ is called *Lagrange multiplier*. We can search for solutions of the equality-constrained problem by seeking stationary points of the Lagrangian function. The first-order necessary condition for x^* to be a local minimizer for the problem (2.53) is defined in the following theorem.

Theorem 7. *Suppose that x^* is the local solution of (2.53), that the functions f and c are continuously differentiable, and that the LICQ [1] holds at x^*. Then there is a Lagrange multiplier vector μ^*, such that the following conditions are satisfied at (x^*, μ^*)*

$$\nabla_x \Lambda(x^*, \mu^*) = 0, \tag{2.54}$$
$$c(x^*) = 0. \tag{2.55}$$

The conditions are often known as the *Karush-Kuhn-Tucker* conditions [21], or the *KKT* conditions for short. By using *KKT* conditions, we obtain a system of $n + m$ equations with $n + m$ unknowns x and μ:

$$F(x, \mu) = \begin{pmatrix} \nabla_x \Lambda(x, \mu) \\ c(x) \end{pmatrix} = 0, \tag{2.56}$$
$$\Rightarrow \begin{pmatrix} \nabla f(x) - \mathbf{J}(x)^T \mu \\ c(x) \end{pmatrix} = 0, \tag{2.57}$$

[1] LICQ stands for linear independence constraint qualification which means the gradients of the equality constraints are linearly independent at x^*.

where $\mathbf{J}(x)$ is the *Jacobian matrix* of the constraints, that is,

$$\mathbf{J}(x)^T = (\nabla c_1(x), \nabla c_2(x), \ldots, \nabla c_m(x)).$$

To solve the nonlinear equations (2.57), *Newton's method* [10] is usually employed. The Jacobian of $F(x, \mu)$ is given by

$$\begin{pmatrix} \nabla^2_{xx}\Lambda(x,\mu) & \nabla^2_{x\mu}\Lambda(x,\mu) \\ \nabla_x c(x) & \nabla_\mu c(x) \end{pmatrix} = \begin{pmatrix} \mathbf{H}(x,\mu) & -\mathbf{J}(x)^T \\ \mathbf{J}(x) & 0 \end{pmatrix}, \tag{2.58}$$

where $\mathbf{H}$ denotes the *Hessian matrix* of the Lagrangian function with respect to x. The Newton iteration step (x_k, μ_k) is given by

$$\begin{pmatrix} x_{k+1} \\ \mu_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \mu_k \end{pmatrix} + \begin{pmatrix} p_k \\ p_\mu \end{pmatrix}, \tag{2.59}$$

where p_k and p_μ are computed by solving the KKT system:

$$\begin{pmatrix} \mathbf{H}_k & -\mathbf{J}_k^T \\ \mathbf{J}_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ p_\mu \end{pmatrix} = \begin{pmatrix} -\nabla f_k + \mathbf{J}_k^T \mu_k \\ -c_k \end{pmatrix}. \tag{2.60}$$

3

Basic strategy

3.1 The problem

A brief description of our problem is that we have to approximate the densely non-uniformly sampled discrete points by smooth spline curves within a specified tolerance band τ as shown in Fig. 3.1. The following aspects must be taken into consideration:

- In order to achieve higher attainable feed rate and to reduce the storage of the data, a smaller number of polynomials and therefore longer polynomial pieces are desired to approximate the data with the specified accuracy.
- When the approximation accuracy is satisfied, the curve should be as smooth as possible. How to evaluate the smoothness (fairness) of a curve is discussed in detail in Section 3.4.3. With smoothing, undesired resonance of the machine tool

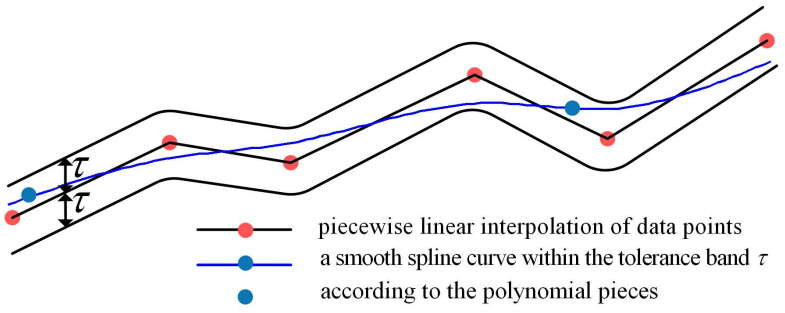


Figure 3.1: Approximation within the tolerance band with a smooth spline curve

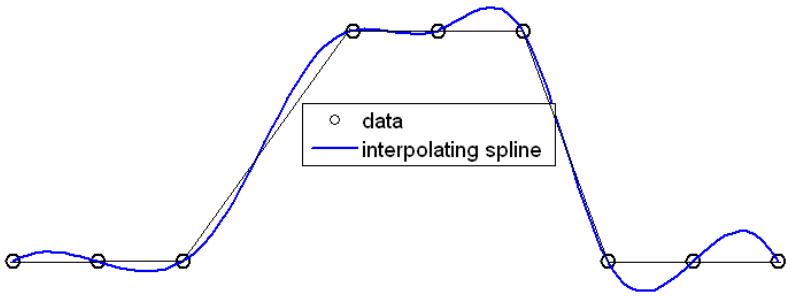


Figure 3.2: Oscillation or bulges caused by undetected sharp edges

can be avoided and velocity, acceleration and jerk capability can be made more use of.

- During the approximation and smoothing, sharp edges represented by large opening angles should be preserved.
- It is important to maintain consistency between the neighboring paths and enhance the milling result in consequence.

3.2 Curve segmentation

Since the part program consists of a large amount of discrete data, in some cases up to millions of points, it is necessary to subdivide the data into small sets to reduce the computation complexity for an *online* compressor. Therefore, breaking the curve at edges is a natural and reasonable way to do the partitioning. In addition, when approximating the discrete data points in the part program, we need to preserve and reproduce the desired edges in order to meet better approximation precision and to avoid undesired bulges or oscillations as shown in Fig. 3.2. However it is a difficult task to identify if the edge is desired or not, in particular when the data points are noisy or in some cases the edges are not so manifest.

In this section, we only discuss what we call *hard edge detection*, which means only one threshold τ_α is employed for the angle α between successive chords $\overline{\mathbf{p}_{i-1}\mathbf{p}_i}$ and $\overline{\mathbf{p}_i\mathbf{p}_{i+1}}$ to determine between edge and non-edge.

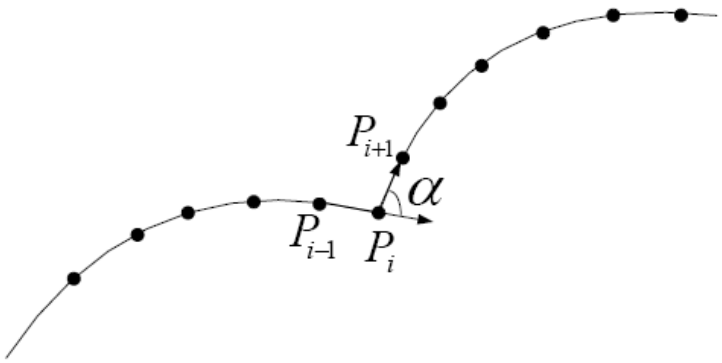


Figure 3.3: Edge detection by angles between neighboring points

As shown in Fig. 3.3 , if

$$\alpha = \arccos \frac{(\mathbf{p}_i - \mathbf{p}_{i-1}) \cdot (\mathbf{p}_{i+1} - \mathbf{p}_i)}{\|\mathbf{p}_i - \mathbf{p}_{i-1}\| \, \|\mathbf{p}_{i+1} - \mathbf{p}_i\|} > \tau_\alpha,$$

the curve is broken into two segments at the point $\mathbf{p}_i$. Afterwards, we consider the problem to approximate each of the curve segments with interpolating end point conditions.

Given the discrete data points $\mathbf{p}_1, \ldots, \mathbf{p}_N$, which can be seen as a linear spline with chordal length parametrization:

$$\mathbf{s}_l = \sum_{j=1}^{N} \mathbf{p}_j N_j^1(\ u \mid T_1\ ), \quad u \in [a_1, a_2],$$

where $T_1 = \{b_1^2, b_2^1, \ldots, b_N^2\}$ in which $a_1 = b_1 = 0$, $b_{j+1} = b_j + \|\mathbf{p}_{j+1} - \mathbf{p}_j\|$, $j = 1, \ldots, N-1$ and $a_2 = \sum_{j=1}^{N-1} \|\mathbf{p}_{j+1} - \mathbf{p}_j\|$.

We want to find a smooth spline curve of degree m within the tolerance band τ around the chords $\overline{\mathbf{p}_j \mathbf{p}_{j+1}}$, $j = 1, \ldots, N-1$.

$$\begin{cases} \mathbf{f}(u) &= \sum_{j=1}^{n} \mathbf{d}_j N_j^m(\ u \mid T_m\ ), \quad u \in [a_1, a_2], \\ \mathbf{f}(a_1) &= \mathbf{p}_1, \\ \mathbf{f}(a_2) &= \mathbf{p}_N, \end{cases} \tag{3.1}$$

where the variables to be determined are the knot sequence $T_m = \{t_1^{m+1}, T^*, t_e^{m+1}\}$ and the control points $\mathbf{d}_j$, $j = 1, \ldots, n$. The end knots in T_m have the same value as in T_1 except the multiplicity is $m + 1$, i.e. $t_1 = b_1 = a_1$ and $t_e = b_N = a_2$. The problem how to determine the interior knots T^* will be discussed in the next section.

3.3 Knot selection

For spline approximation, a key and fundamental task is to select the proper knot sequence in order to achieve better approximation accuracy with a smaller number of polynomial pieces. Other than the conventional methods for knot generation, we come up with a novel scheme to construct the knot sequence based on curvature characteristics. In the following, we will explain how to generate the knots from three aspects: the positions of the knots, the multiplicity of the knots and the number of the knots.

3.3.1 Knot distribution based on curvature characteristics

Here we will discuss how to choose the appropriate knot positions. How well a curve can be approximated by splines of a certain degree with a fixed number of knots depends considerably on where the knots are placed. First we consider the distribution of simple knots for an ideal curve without cusps such as edges or curvature jumps.

Curvature plays an extremely important role in capturing the essence of a curve and it can be even taken as the signature of a curve. Therefore, the basic idea is to distribute the knots according to *curvature characteristics*, which means that at regions of higher curvature knots are placed more densely in order to achieve better approximation accuracy. In addition, we want to find a knot distribution which can well represent the distribution of the data points in the part program.

Since the tool path is discretized (sampled) by the CAM system according to a specified chordal error, the block length between the neighboring sampled points is determined by the curvature of the tool path. The relation between the chordal error e, block length l and curvature $\kappa = \dfrac{1}{r}$ is illustrated in Fig. 3.4.

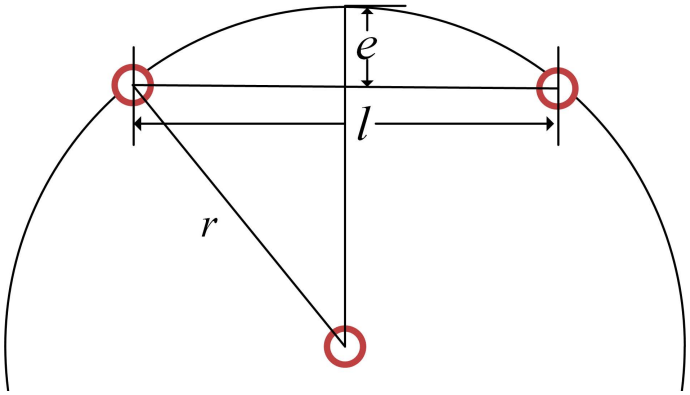


Figure 3.4: The relation between the block length and the curvature radius

$$r^2 = \left(\frac{l}{2}\right)^2 + (r-e)^2, \tag{3.2}$$

$$l = \sqrt{8re - 4e^2}. \tag{3.3}$$

Since usually $e << l$, we get $l \approx \sqrt{8e} \cdot \sqrt{r} = \frac{\sqrt{8e}}{\sqrt{\kappa}}$. Therefore, approximately the block length is reciprocally proportional to the square root of curvature.

In order to well represent the distribution of the data points, the criterion we choose is to distribute the knots to make the integral of the square root of the curvature in each knot interval to be a constant C:

$$\int_{b_i}^{b_{i+1}} \sqrt{\kappa}\, du = C, \qquad b_i : \text{interior break point.} \tag{3.4}$$

Furthermore, by distributing the knots according to the square root of the curvature, we can achieve better approximation accuracy, see the following theorem:

Theorem 8. *[5, Chap. XII] Assume that the function f is continuous on $[a_1, a_2]$ and is k times continuously differentiable at all but finitely many points in $[a_1, a_2]$ near which $D^k f$ is monotone and the $k-th$ root of $D^k f$ is integrable, i.e. $\int_{a_1}^{a_2} \left|D^k f(x)\right|^{1/k} dx < \infty$. If the break points $b_1, \ldots, b_s$ are chosen to make*

$$\int_{b_i}^{b_{i+1}} \left|D^k f(x)\right|^{1/k} dx = \frac{1}{s-1} \int_{a_1}^{a_2} \left|D^k f(x)\right|^{1/k} dx, \qquad i = 1, \ldots, s-1, \tag{3.5}$$

we have that

$$dist(f, Af) \leq C_k (s-1)^{-k} \left(\int_{a_1}^{a_2} \left|D^k f(x)\right|^{1/k} dx \right)^k, \tag{3.6}$$

where Af is an approximating spline curve to f, see the details in [5, Chap. XII].

If k is chosen to be 2 and the curve is approximately arc length parametrized, we get from (3.5) that

$$\int_{b_i}^{b_{i+1}} \sqrt{\kappa}\, du = C. \tag{3.7}$$

Discrete curvature estimation

In order to distribute the knots based on curvature, we need first estimate the curvature. Since the points $\mathbf{p}_j,\ j = 1, \ldots, N$, stored in a part program, are discrete data, the curvature κ_j ($j = 2, \ldots, N-1$) is also estimated at discrete parameter values u_j, $u_1 = 0$ and $u_{j+1} - u_j = \left\|\mathbf{p}_{j+1} - \mathbf{p}_j\right\|$, $j = 1, \ldots, N-1$. There are various ways to estimate the discrete curvature [3, 11, 12], but in this section we only discuss a basic method called *circumcircle method*.

The discrete curvature $\kappa_j(u_j)$ at the point $\mathbf{p}_j$, $j = 2, \ldots, N-1$, is determined by the reciprocal of the radius R_j of the circle through the three neighboring points $\mathbf{p}_{j-1}, \mathbf{p}_j, \mathbf{p}_{j+1}$ as illustrated in Fig. 3.5. The circumcircle of a triangle $(\mathbf{p}_{j-1}, \mathbf{p}_j, \mathbf{p}_{j+1})$ with side of length $a = \|\mathbf{p}_{j+1} - \mathbf{p}_{j-1}\|$ and opposing angle α has radius

$$R = \frac{a/2}{\sin(\beta)} = \frac{a}{2\sin(\pi - \alpha)}, \tag{3.8}$$

where

$$\pi - \alpha = \arccos \frac{(\mathbf{p}_{j+1} - \mathbf{p}_j) \cdot (\mathbf{p}_j - \mathbf{p}_{j-1})}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\| \|\mathbf{p}_j - \mathbf{p}_{j-1}\|}. \tag{3.9}$$

Since the circumcircle method is sensitive to noise, especially for dense points, some other schemes to estimate the curvature will be considered in detail in Section 4.4. For example, a locally interpolating quadratic polynomial [13] or Akima spline [1] can be used to estimate the discrete curvature. A more sophisticated scheme based on area invariant method, using the concept of integral instead of differentiation, is more robust to noise.

To explain the knot distribution more intuitively, we plot $f(x) = \int_0^x \sqrt{\kappa}(u)\, du, x \in [u_1, u_N]$, see Fig. 3.6. If we place q interior knots, then the axis of $\int \sqrt{\kappa}\, du$ is first divided into $q+1$ segments. Horizontal lines are drawn from the axis of $\int \sqrt{\kappa}\, du$ to intersect the plot of the integral of the square root of curvature, then the intersections are projected to the axis of parameter u to find the positions of the interior knots

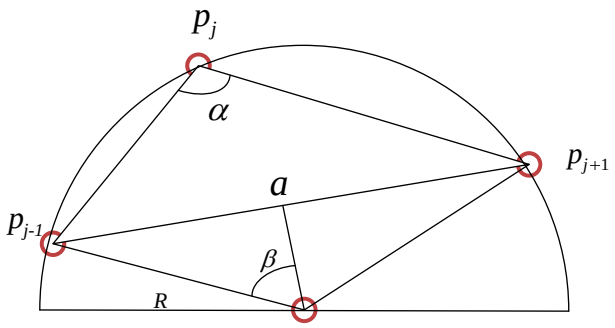


Figure 3.5: Discrete curvature estimation by circumcircle through three neighboring points

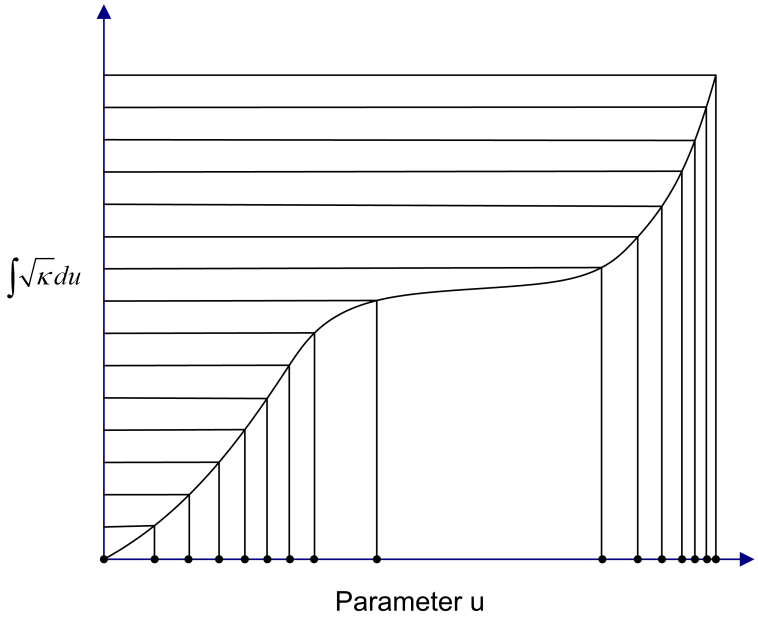


Figure 3.6: Integral of square root of curvature for knot placement

t_i. Since the curvature $\kappa(u_j)$ is given at discrete parameters $u_j, j = 1, \ldots, N$, we can approximate the integral of the square root of curvature by computing the cumulative integral via the trapezoidal method or some other quadrature methods.

3.3.2 Curvature jump detection and multiple knots

In the previous section, we consider the distribution of simple knots for an ideal curve without cusps. However, when approximating the part program which contains some *feature points* such as sharp edges or curvature jumps, simple knots may be not sufficient. Instead, by using multiple knots, we can represent the feature points more precisely and can achieve better approximation accuracy as well. The rule to choose the knot's multiplicity is based on the property that the spline curve of degree m is at least C^{m-k} continuous at the knots of multiplicity k. As we know, at the curvature jumps, the curve can only be expected to be C^1 continuous. According to the property of the spline curve, the C^1 continuity can be realized by a spline curve of degree m with $m-1$-fold knots at the appropriate positions. Instead, if only simple knots are used, the side effect of undesired undulations may be visible although the order of continuity is higher, see Fig. 3.7. In the figure, the data points are extracted from the part program 'daimler'. For both approximating spline curves, the break points of the knot sequence are chosen the same. The difference is that for the 'red' spline, the multiplicity of the knots at the curvature jumps is $m-1$ and the 'blue' spline has simple knots. With the multiple knots, we can achieve much better approximation precision (approximation error: $7\mu m$) with the same number of polynomials, compared to the result with simple knots (approximation error $50\mu m$). In order to set the multiplicity of the knots accordingly, we need to first detect the jumps of curvature. With the curvature estimation method described before, a jump of curvature is detected if the curvature difference between neighboring points is larger than a specified threshold τ_κ. The indices J of the curvature jumps are

$$J = \{j \mid |\kappa_j - \kappa_{j-1}| > \tau_\kappa\}.$$

After detecting the curvature jumps, $m-1$-fold knots should be set correspondingly at the points $\mathbf{p}_j, j \in J$, and meanwhile the $m-1$-fold knots u_j,

$$\{u_j^{m-1}\} \subset T^*, \quad j \in J,$$

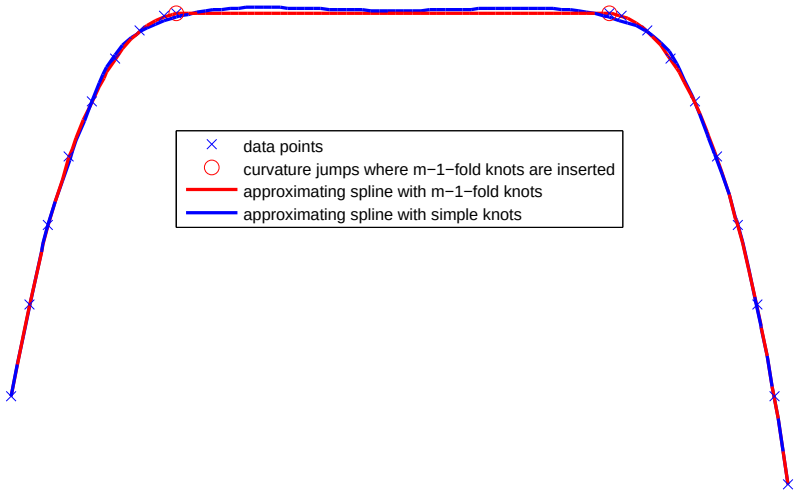


Figure 3.7: Comparison between the approximating results with multiple knots and simple knots

are also determined.

Remark 2. Similar to curvature jumps, another way to handle the sharp edges is to use a spline of degree m with m-fold knots at the edges. Considering the computation complexity, we choose to break the curve into segments instead of using m-fold knots, which is described in Section 3.2.

When the $m-1$-fold knots are fixed, the curve is divided into several curve segments $\mathcal{C}_k$, $k = 1, \ldots, \#(J)+1$ separated by the curvature jumps. In each curve segment, the simple knots are distributed in the same way as described in (3.4). As shown in Fig. 3.8, a curve is subdivided into three segments A-P, P-Q and Q-B separated by curvature jumps at P and Q. For instance, the respective number of simple knots is $q_k = 7, 2$ and 5 for the curve segments A-P, P-Q and Q-B. Then for each curve segment, the axis of $\int \sqrt{\kappa}$ is divided into $q_k + 1$ segments. In the same way, we can get the positions of the simple knots. And the simple and multiple knots are shown as black and red dots respectively along the axis of parameter u.

We extract a path from the part program 'daimler' to illustrate how to place the knots based on curvature characteristics. The data points, the curvature and the integral of the square root of curvature are shown in Fig. 3.9, Fig. 3.10 and Fig. 3.11

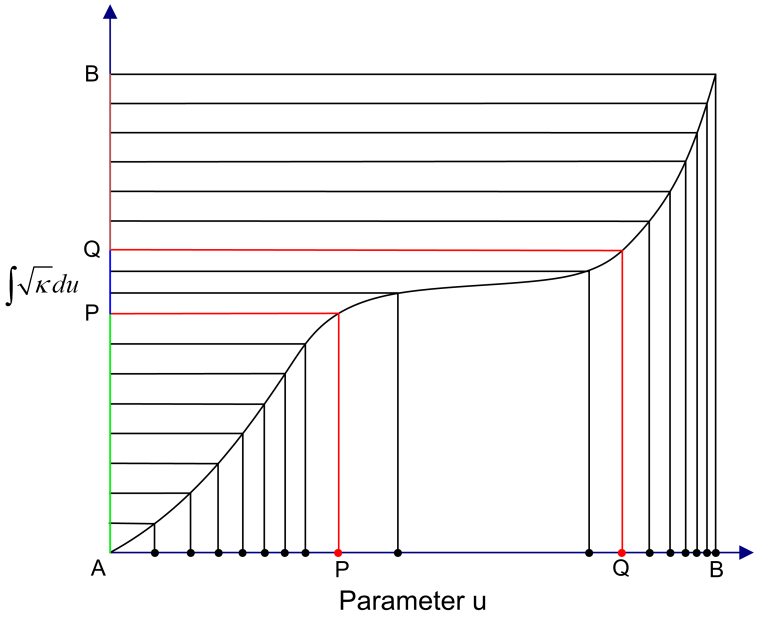


Figure 3.8: Integral of square root of curvature for knot placement with multiple knots

respectively. The approximating spline curve and the corresponding polynomial pieces are illustrated in Fig. 3.12. From the figures we can see that the curvature is high in the regions where the data points are densely sampled and dense knots are also placed accordingly in the areas of high curvature.

3.3.3 Number of knots

Another important aspect for placing the knots is concerning the number of the knots. On the one hand, we want to get a spline curve with fewer polynomial pieces and on the other hand we still need to preserve some more degree of freedom for the smoothing phase. In the implementation, the number of knots are updated iteratively until the tolerance requirement is satisfied. In order to accelerate the iteration process, the initial number of knots is chosen in a heuristic way instead of starting from 1.

The *initial* number q_k of simple knots in each curve segment $\mathcal{C}_k$ is determined by the number of data points N_k of the respective curve segment heuristically. Assume that $q = g(N)$, where the function g is a "compression function". Since we want to get a higher "compression rate" for a larger number of data points, the function g should

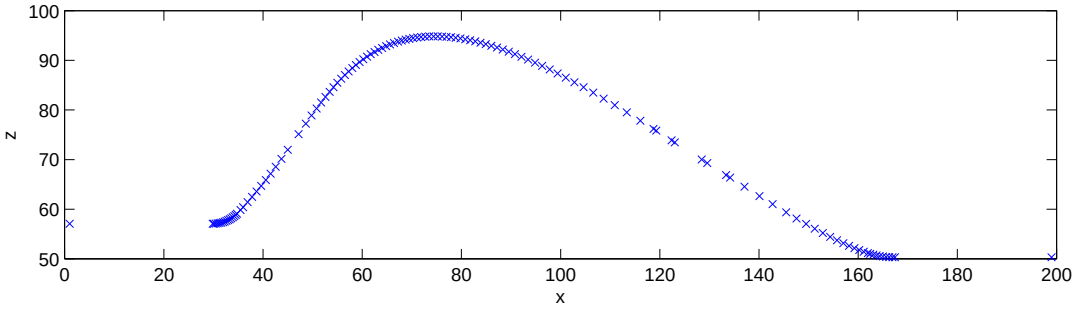


Figure 3.9: Data points on the part program 'daimler'

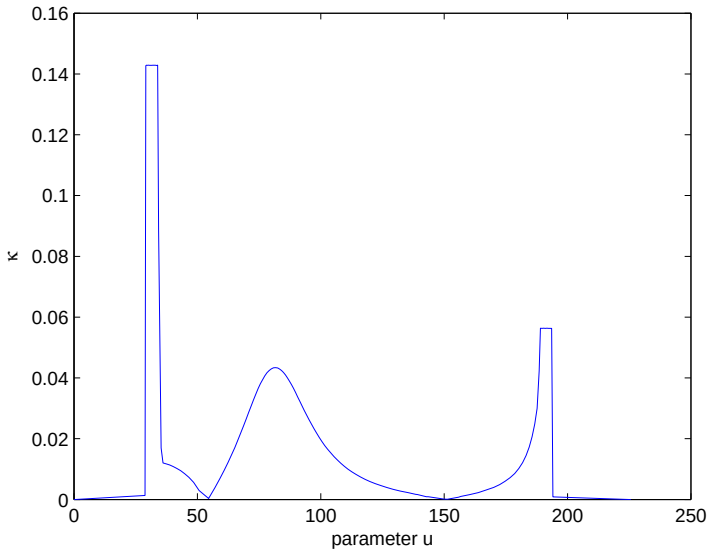


Figure 3.10: Curvature plot using the circumcirle method

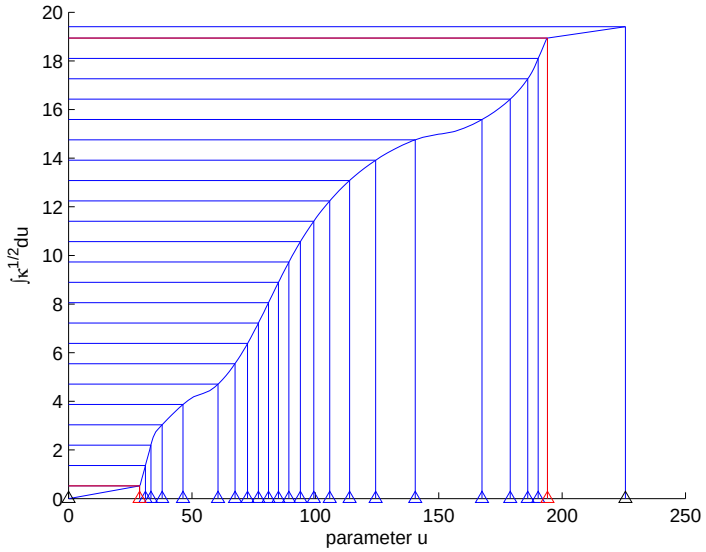


Figure 3.11: Integral of square root of curvature

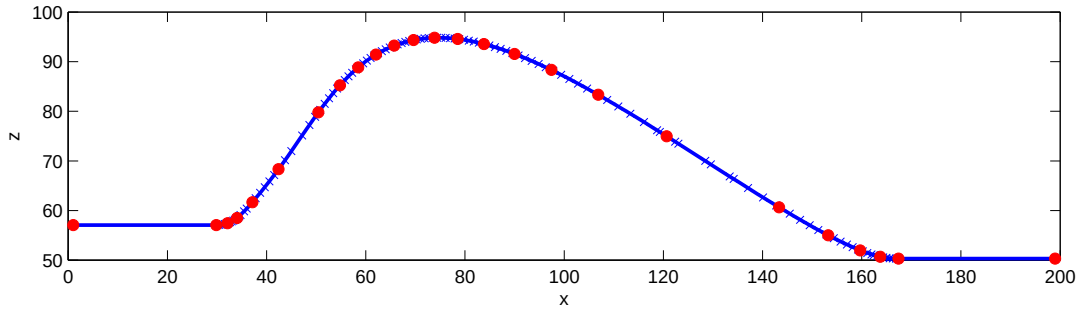


Figure 3.12: The approximating spline curve and the corresponding polynomial pieces

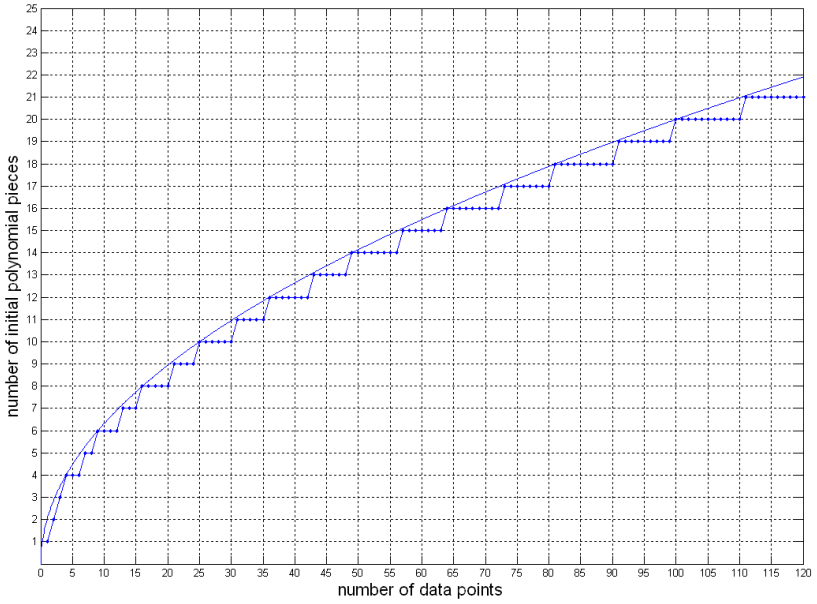


Figure 3.13: Determine the initial number q_k based on N_k

be chosen to be monotonically increasing with decreasing derivatives. The square root of N is a good candidate:

$$q_k = \min\left(\left\lfloor \omega\sqrt{N_k} \right\rfloor, N_k\right), \tag{3.10}$$

in which ω is an integer and $k = 1, \ldots, \#(J) + 1$. For $\omega = 2$, the relation between the the initial number of knots and the data points is presented in Fig. 3.13. To determine an appropriate integer ω, we should take into account the trade-off between the final number of polynomials pieces and the iteration steps needed to satisfy the approximation accuracy.

3.4 Approximation

In this section, we will discuss another key issue of the thesis: how to choose the proper measure of discrepancy and the appropriate minimization functional.

3.4.1 Comparison between two splines

In order to evaluate the distance between two splines and to control the approximation error, we need to make the two splines comparable. This requires the two splines to be defined on the same knot sequence and to be of the same degree, which means that they have the same basis functions. If this condition is satisfied, the maximum error between two splines $\mathbf{s}_1$ and $\mathbf{s}_2$ of the same basis functions is bounded by the maximum error between the control points:

$$\|\mathbf{s}_1(u) - \mathbf{s}_2(u)\| = \|\sum_{j=1}^{L} \mathbf{e}_j N_j^m(u|T)\| \leq \sum_{j=1}^{L} \|\mathbf{e}_j\| N_j^m(u|T) \leq \max_{j=1}^{L} \|\mathbf{e}_j\|, \tag{3.11}$$

where $\mathbf{e}_j$ is the difference of control points between splines $\mathbf{s}_1(u)$ and $\mathbf{s}_2(u)$.

It is easy to prove (3.11) by using the property that the basis function $N_j^m(\ u \mid T\ )$ is non-negative and the sum of all non-zero basis functions of degree m on span $[u_i, u_{i+1})$ is 1.

However, in our case the two splines

$$\mathbf{s}_l = \sum_{j=1}^{N} \mathbf{p}_j N_j^1(\ u \mid T_1\ )$$

and

$$\mathbf{f} = \sum_{j=1}^{n} \mathbf{d}_j N_j^m(\ u \mid T_m\ )$$

which we want to compare are not of the same degree and have different knot sequences as well. Therefore, for the linear spline, we need to first elevate its degree to m: $\mathbf{s}_m = \sum_{j=1}^{n'} \mathbf{p}_j^m N_j^m(\ u \mid T_2\ )$, then insert additional knots $T_1^* = \{x | x \in T_m \text{ and } \notin T_2\}$ in succession. For the target spline curve $\mathbf{f}$ that we work for, we have to insert knot sequence $T_2^* = \{x | x \in T_2 \text{ and } \notin T_m\}$.

$$\begin{array}{lll}
\mathbf{s}_l = \sum_{j=1}^{N} \mathbf{p}_j N_j^1(\ u \mid T_1\ ) & \xrightarrow{\text{degree elevation}} & \mathbf{s}_m = \sum_{j=1}^{n'} \mathbf{p}_j^m N_j^m(\ u \mid T_2\ ) \\
 & \xrightarrow{\text{insert } T_1^*} & \mathbf{s}_f = \sum_{j=1}^{n''} \mathbf{p}_j^f N_j^m(\ u \mid T_f\ ) \\
\mathbf{f} = \sum_{j=1}^{n} \mathbf{d}_j N_j^m(\ u \mid T_m\ ) & \xrightarrow{\text{insert } T_2^*} & \mathbf{f}_f = \sum_{j=1}^{n''} \mathbf{d}_j^f N_j^m(\ u \mid T_f\ )
\end{array}$$

Degree elevation

Since a linear spline $\mathbf{s}_l = \sum_{j=1}^{N} \mathbf{p}_j N_j^1(\ u \mid T_1\ )$ is a piecewise Bézier curve, we can directly use the degree elevation formula from (2.33), giving

$$\mathbf{p}^{k+1} = \overline{\mathbf{E}}_k \mathbf{p}^k,\ k = 1, \ldots, m-1,$$

starting with $\mathbf{p}^1 = \mathbf{p}$ and the number of the repeated pattern in the degree raising matrix $\overline{\mathbf{E}}_k$ is the number of Bézier curve segments, which in our case is $N-1$. After the degree elevation, the linear spline becomes a spline curve of degree m: $\mathbf{s}_m = \sum_{j=1}^{n'} \mathbf{p}_j^m N_j^m(\ u \mid T_2\ )$, where $n' = (N-1)m$, $\mathbf{p}^m = \overline{\mathbf{E}}_{m-1} \cdots \overline{\mathbf{E}}_1 \mathbf{p} =: \mathbf{E}\mathbf{p}$ and $T_2 = \{b_1^{m+1}, b_2^m, \ldots, b_N^{m+1}\}$.

Knot insertion

After the degree elevation, the splines $\mathbf{s}_m$ and $\mathbf{f}$ are of the same degree. Since their knot sequences are still different, we need to insert their complementary knots to each of them to make them have the same basis functions.

$$\mathbf{s}_m = \sum_{j=1}^{n'} \mathbf{p}_j^m N_j^m(\ u \mid T_2\ ) \quad \overset{\text{insert } T_1^*}{\longrightarrow} \quad \mathbf{s}_f = \sum_{j=1}^{n''} \mathbf{p}_j^f N_j^m(\ u \mid T_f\ ),$$
$$\mathbf{f} = \sum_{j=1}^{n} \mathbf{d}_j N_j^m(\ u \mid T_m\ ) \quad \overset{\text{insert } T_2^*}{\longrightarrow} \quad \mathbf{f}_f = \sum_{j=1}^{n''} \mathbf{d}_j^f N_j^m(\ u \mid T_f\ ).$$

After knot insertion is performed, the two splines $\mathbf{s}_f$ and $\mathbf{f}_f$ we got have the same knot sequence, the same basis functions and the same number of control points. Their control points are determined by

$$\mathbf{p}^f = \mathbf{A}_1^m(T_2, T_f)\mathbf{p}^m = \mathbf{A}_1^m(T_2, T_f)\mathbf{E}\mathbf{p}$$

and

$$\mathbf{d}^f = \mathbf{A}_2^m(T_m, T_f)\mathbf{d}.$$

An example to show the change of the control points before and after the degree elevation and knot insertion is illustrated in Fig. 3.14 and Fig. 3.15. From the figures,

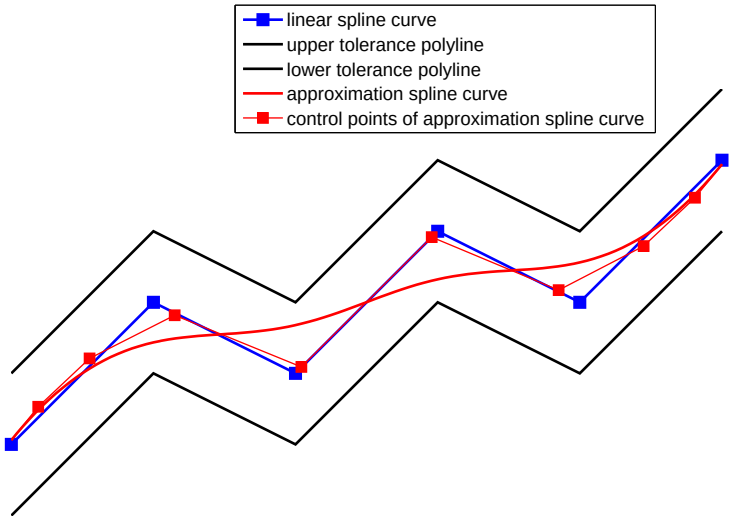


Figure 3.14: Control points before knot insertion and degree elevation

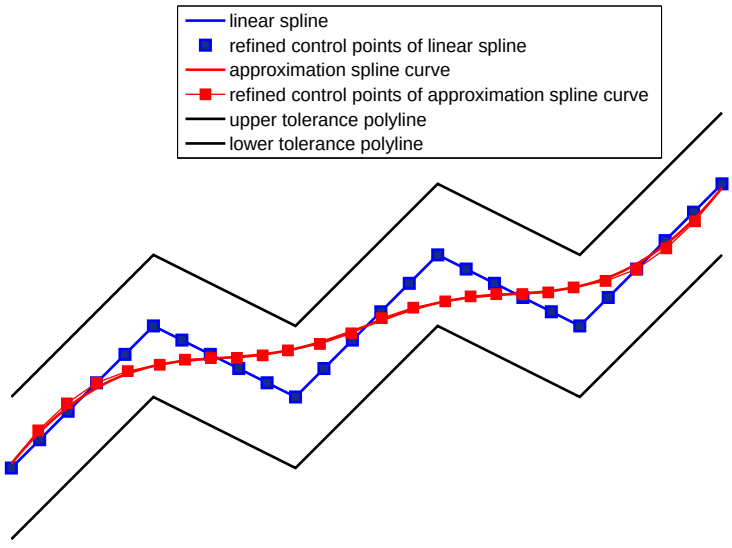


Figure 3.15: Control points after knot insertion and degree elevation

we can see that the number of the refined control points is increased considerably and the refined control points move closer to the spline curve.

3.4.2 Least squares approximation

As mentioned in Section 2.2.9.3, the minimization of the discrete errors between the data points and the curve at corresponding parameter values cannot guarantee global accuracy to the data polygon. A better solution is to minimize the least squares error between the refined control points $\mathbf{p}^f$ and $\mathbf{d}^f$, since the refined control points are closer to their spline curves and we get much denser points to compare.

$$\sum_{j=1}^{n''} \|(\mathbf{A}_2^m(T_m, T_f)\mathbf{d})_j - (\mathbf{A}_1^m(T_2, T_f)\mathbf{E}\mathbf{p})_j\|^2 \to \min. \tag{3.12}$$

Since this minimization problem is a quadratic one, the control points $\mathbf{d}$ can be determined by solving the normal equation:

$$\mathbf{A}_2^T\mathbf{A}_2\mathbf{d} = \mathbf{A}_2^T\mathbf{A}_1\mathbf{E}\mathbf{p}.$$

And the maximum error between two splines with the same basis functions is bounded by the maximum error between the control points.

$$\begin{aligned} \delta &:= \max_j \|(\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\| \\ &\geq \max_u \|\sum_{j=1}^{n} \mathbf{d}_j N_j^m(\ u \mid T_m\ ) - \sum_{j=1}^{N} \mathbf{p}_j N_j^1(\ u \mid T_1\ )\|. \end{aligned} \tag{3.13}$$

Since the precise error between two spline curves is continuous and hard to compute, (3.13) gives us a conservative bound of the maximum error by evaluating only the discrete control points, which is a linear computation and more efficient. If $\delta \leq \tau$, the spline $\mathbf{f}$ can be guaranteed to lie within the tolerance band τ around the linear spline $\mathbf{s}_l$. Otherwise, the number of interior knots q_k is increased by a factor $\gamma > 1$ iteratively until the tolerance condition is satisfied.

3.4.3 Smoothness

After the approximating spline satisfying the tolerance band condition is found, the next step is to make the approximating spline based on the same knot sequence as

smooth as possible by regulating the control points. Intuitively, a curve possessing less undesirable undulations such as bulges and wrinkles is smoother. The concept of fairness is typically associated with the curvature characteristics of a curve, see [20, 26]. Here we will discuss two smoothness criteria based on curvature:

Strain Energy: The functional to be minimized is the integral of the square of the curvature. Since the integral measures the strain energy accumulated in the bending rod, this criterion is also referred to as strain energy criterion (SE).

$$\int |\ddot{\mathbf{f}}(s)|^2 \, ds \to \min. \tag{3.14}$$

In (3.14), s is the arc length and $\ddot{\mathbf{f}}(s)$ is equal to the curvature. For unified notation, we use $\dot{\mathbf{f}}(s)$ to denote the derivative with respect to arc length, while $\mathbf{f}'(u)$ denotes the derivative with respect to the underlying parameter.

Curvature Variation: The functional to be minimized is the integral of the square of the curvature derivative. Physically, the criterion can be taken as minimization of the shear force of the beam. With this criterion (CV), the curve has gradual varying curvature with less curvature extrema and inflections.

$$\int |\dddot{\mathbf{f}}(s)|^2 \, ds \to \min. \tag{3.15}$$

In the context of our application, the criterion CV is superior to the criterion SE due to the following aspects:

- CV requires one order more of continuity and thus yields "smoother" curves.
- CV demands to keep the curvature as constant as possible instead of as small as possible to reduce undesired inflections and sign changes in curvature.
- From the point of view of dynamics and the concept of 'look ahead', SE is relevant to the acceleration $\mathbf{f}''(t)$ while CV leads to a reduced jerk $\mathbf{f}'''(t)$ of the machine, as shown in the following equations:

$$\begin{aligned}
\mathbf{f}'(t) &= \dot{\mathbf{f}}(s)\frac{ds}{dt}, \\
\mathbf{f}''(t) &= \dot{\mathbf{f}}(s)\frac{d^2s}{dt^2} + \ddot{\mathbf{f}}(s)(\frac{ds}{dt})^2, \\
\mathbf{f}'''(t) &= \dot{\mathbf{f}}(s)\frac{d^3s}{dt^3} + 3\ddot{\mathbf{f}}(s)\frac{ds}{dt}\frac{d^2s}{dt^2} + \dddot{\mathbf{f}}(s)(\frac{ds}{dt})^3.
\end{aligned}$$

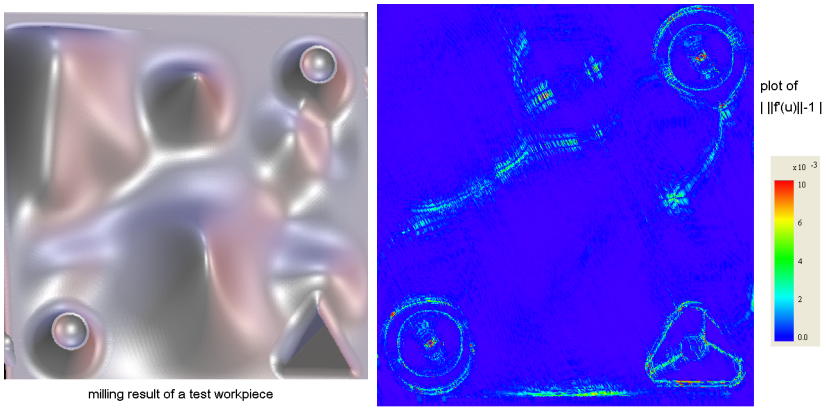


Figure 3.16: The deviation to arc length parametrization

In the following discussion, we will adopt the criterion to minimize the curvature variation. The functional to be minimized is:

$$\int |\dot{\kappa}(s)|^2 \, ds = \int \frac{|\kappa'(u)|^2}{\|\mathbf{f}'(u)\|} \, du \to \min, \qquad \kappa(u) = \ddot{\mathbf{f}}(s) = \frac{\mathbf{f}'(u) \times \mathbf{f}''(u)}{\|\mathbf{f}'(u)\|^3},$$

which is complicated and highly nonlinear. If the curve is assumed to be approximately parametrized by its arc length, the CV functional can be simplified as:

$$\int |\mathbf{f}'''(u)|^2 \, du \to \min, \tag{3.16}$$

where u is an approximation to the arc length s.

The assumption that the spline curve can be taken as approximately arc length parametrized is well justified, if $\|\mathbf{f}'(u)\| \approx 1$. As shown in Fig. 3.16, $|\|\mathbf{f}'(u)\| - 1|$ is evaluated on the approximating spline of a test workpiece and we found that the deviation to the arc length parametrization is quite small.

If we substitute the B-spline representation $\mathbf{f}(u) = \sum_{j=1}^{n} \mathbf{d}_j N_j^m(u|T^m)$ into the function (3.16), it can be reduced to quadratic form:

$$V_1(\mathbf{d}) := \int \left| \sum_{j=1}^{n} \mathbf{d}_j (N_j^m)'''(u|T^m) \right|^2 du =: \mathbf{d}^T \mathbf{M} \mathbf{d}, \tag{3.17}$$

where

$$\mathbf{M} = \left[\int (N_j^m)''' (N_k^m)''' : j, k \right],$$

which can be computed by Gaussian quadrature, see Section 2.2.4.

3.4.4 Optimization problem

To integrate smoothness into the approximation term, we derive the final optimization problem:

- Approximation term:

$$V_0(\mathbf{d}) := \sum_{j=1}^{n''} \|(\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\|^2.$$

- Smoothness term:

$$\begin{aligned} V_1(\mathbf{d}) &:= \int \left| \sum_{j=1}^{n} \mathbf{d}_j (N_j^m)'''(u|T_m) \right|^2 du \\ &=: \mathbf{d}^T\mathbf{M}\mathbf{d}. \end{aligned}$$

- Optimization problem:

$$\min V(\mathbf{d}), \qquad V(\mathbf{d}) := V_0(\mathbf{d}) + \lambda V_1(\mathbf{d}),$$

  where λ is used to regulate the weight of the smoothness term.

- Equality constraint (endpoint interpolation at the transitions of successive path segments):

$$\mathbf{Q}_{eq}\mathbf{d} = \mathbf{p}_{eq},$$

  with

$$\mathbf{Q}_{eq} = \begin{bmatrix} N_1(t_1|T_m) & \cdots & N_n(t_1|T_m) \\ N_1(t_e|T_m) & \cdots & N_n(t_e|T_m) \end{bmatrix}$$

  and

$$\mathbf{p}_{eq} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_N \end{bmatrix}$$

The optimization problem is a quadratic one:

$$\begin{aligned} V(\mathbf{d}) &= (\mathbf{A}_2\mathbf{d} - \mathbf{A}_1\mathbf{E}\mathbf{p})^T(\mathbf{A}_2\mathbf{d} - \mathbf{A}_1\mathbf{E}\mathbf{p}) + \lambda\mathbf{d}^T\mathbf{M}\mathbf{d} \\ &= \mathbf{d}^T(\mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{M})\mathbf{d} - 2(\mathbf{A}_1\mathbf{E}\mathbf{p})^T\mathbf{A}_2\mathbf{d} + (\mathbf{A}_1\mathbf{E}\mathbf{p})^T(\mathbf{A}_1\mathbf{E}\mathbf{p}), \end{aligned}$$

hence of the form

$$\min_x \frac{1}{2}x^TGx + p^Tx + b, \qquad x \in \mathbb{R}^3, \tag{3.18}$$

with the equality constraint $\mathbf{Q}_{eq}\mathbf{d} - \mathbf{p}_{eq} = 0$.
By means of *Lagrange multipliers* μ, we need to solve the following system of equations:

$$\begin{aligned} 2(\mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{M})\mathbf{d} - 2\mathbf{A}_2^T(\mathbf{A}_1\mathbf{E}\mathbf{p}) + \mathbf{Q}_{eq}^T\mu &= 0, \\ \mathbf{Q}_{eq}\mathbf{d} - \mathbf{p}_{eq} &= 0, \end{aligned}$$

that can be written in matrix form as:

$$\begin{bmatrix} 2(\mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{M}) & \mathbf{Q}_{eq}^T \\ \mathbf{Q}_{eq} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ \mu \end{bmatrix} = \begin{bmatrix} 2(\mathbf{A}_1\mathbf{E}\mathbf{p})^T\mathbf{A}_2 \\ \mathbf{p}_{eq} \end{bmatrix}. \tag{3.19}$$

The control points of the final smooth spline curve are determined by solving the equation. To find the smoothest curve within the tolerance band, the smoothing factor λ is updated iteratively and a *bisection* method is employed to find the largest possible smoothing factor.

3.5 Summary of the strategy

Approximation to the part program has two principal issues: to control the approximation error within the specified tolerance τ and to achieve a curve as smooth as possible. It will be solved in two basic steps: first find a spline curve within the tolerance band, which consists of a small number of polynomials and then improve the smoothness of the curve as long as the tolerance requirement is satisfied. The general strategy to find the approximating spline curve with smoothing is stated as follows:

1. Find a spline curve staying within the tolerance band τ *without* smoothing. The essential idea in this step is to find a proper knot sequence T_m, which consists of a small number of knots and meanwhile better approximation precision can be achieved with an appropriate knot distribution based on curvature characteristics.

   (a) Determine the initial knot sequence which involves

      i. curvature jump detection;

      ii. determination of the initial number of simple knots;

      iii. distribution of the simple knots based on curvature characteristics.

   (b) Determine the control points $\mathbf{d}$ by least squares approximation, which is reduced to solve (3.19) with $\lambda = 0$.

(c) Estimate the maximum error δ (3.13).

(d) **While** $\delta \geq \tau$

i. Update the knot sequence by increasing the number of simple knots;

ii. Determine the control points by solving (3.19) with the updated knots and the updated knot insertion matrix.

2. Find the *smoothest* spline based on the knot sequence obtained in step 1 without violating the tolerance band condition.
The essential task in this step is to adjust the control points by updating the weighting factor λ until we find the largest possible value for λ.

(a) Initialize λ to be a sufficiently large number.

(b) Optimization problem to solve (3.19).

(c) Estimate the maximum error δ (3.13).

(d) Use *bisection* method to update λ,
repeat steps 2((b)-(c)) until $0 < \tau - \delta < \xi$ is satisfied (ξ is a very small number).

In the next chapter we will proceed to discuss some extended strategies to cope with the specific problems in the part programs.

4

Extended strategy

4.1 Preprocessing

When the tool path is sampled in the CAM system with a certain tolerance, the discrete data points in the part program are usually not ideally distributed on the tool path according to the curvature characteristics and in some cases, the sampled points are even very critical. For example, as shown in Fig. 4.1 which is a part extracted from the workpiece 'beetle', there exist clusters of very close points where the distances between the neighboring points are even less than the tolerance of the compressor. Besides, there may exist a lot of redundant data points in the part program, which are of no use and can create problems. To solve such problems, a preprocessing step is necessary in order to deal with the artifacts in the part program.

4.1.1 Cluster modifications

The first part of the preprocessing step is to treat the critical points where the distances between the neighboring points are smaller than the compressor tolerance ϵ. In addition, the close points are even more critical when they are noisy, since these close and noisy data can result in incorrect estimation of discrete curvature and consequently, significant overestimation of curvature jumps.

Before describing the strategy to remove the critical points, some notations are first defined and illustrated as follows:

Note 3. 1. The close points usually appear as clusters in the part program. Here the cluster of critical points is defined as the set of points where the distances

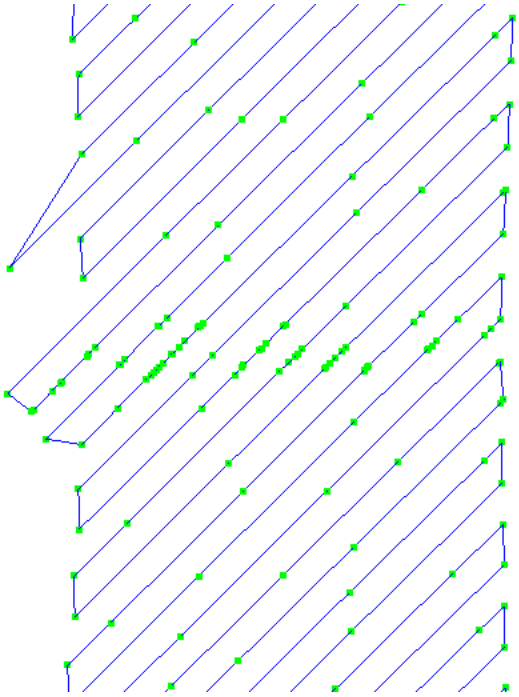


Figure 4.1: Critical points (close clusters) in the part program 'beetle'

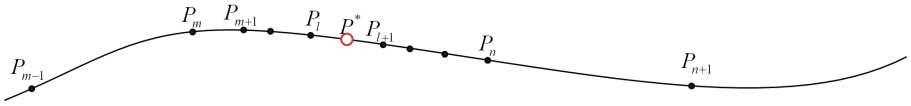


Figure 4.2: A cluster of close points

between two successive points are not larger than ϵ. As shown in Fig. 4.2, the set of points $[\mathbf{p}_m, \ldots, \mathbf{p}_n]$ is recognized as a cluster of critical points if

$$\begin{cases} \|\mathbf{p}_{i+1} - \mathbf{p}_i\| \leq \epsilon, & i = m, \ldots, n-1, \\ \|\mathbf{p}_m - \mathbf{p}_{m-1}\| > \epsilon, \\ \|\mathbf{p}_{n+1} - \mathbf{p}_n\| > \epsilon. \end{cases}$$

2. $\mathcal{L}(\mathbf{p}_j, \mathbf{p}_k)$ is defined as the accumulated distance between the points $\mathbf{p}_j$ and $\mathbf{p}_k$.

$$\mathcal{L}(\mathbf{p}_j, \mathbf{p}_k) = \sum_{i=j}^{k-1} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|$$

3. The point $\mathbf{p}^*$ on the piecewise linear curve from $\mathbf{p}_m$ to $\mathbf{p}_n$ is said to bisect the cluster if $\mathcal{L}(\mathbf{p}_m, \mathbf{p}^*) = \mathcal{L}(\mathbf{p}^*, \mathbf{p}_n)$.

4. A cluster C can be subdivided into two bisecting sub-clusters S_1 and S_2 separated by $\mathbf{p}^*$.
   a) If $\mathbf{p}^*$ is one of the critical points, i.e. $\mathbf{p}^* = \mathbf{p}_l$,
   $\Longrightarrow$ $S_1 = [\mathbf{p}_m, \ldots, \mathbf{p}_l]$ and $S_2 = [\mathbf{p}_l, \ldots, \mathbf{p}_n]$.
   b) If $\mathbf{p}^*$ is between two critical points $\mathbf{p}_l$ and $\mathbf{p}_{l+1}$,
   $\Longrightarrow$ $S_1 = [\mathbf{p}_m, \ldots, \mathbf{p}_l]$ and $S_2 = [\mathbf{p}_{l+1}, \ldots, \mathbf{p}_n]$.

The idea of the strategy is to extract the critical clusters and then recursively bisect the clusters until the sub-clusters could be replaced by their midpoints, if the distance between the end points of the cluster is smaller than the tolerance ϵ or all the points in the cluster are contained in the circle of radius ϵ around the midpoint. The method is direction neutral, which means the results are the same whether we approach the paths from one direction or the other.

Algorithm 4. Remove the cluster of close points:

1. Extract the critical clusters.

2. Process the clusters:

   - a) If $\mathcal{L}(\mathbf{p}_s, \mathbf{p}_e) \leq \epsilon$, where $\mathbf{p}_s, \mathbf{p}_e$ are the start and end points of the cluster respectively, the cluster is replaced by its midpoint $\mathbf{p}_a = \dfrac{1}{e-s+1}(\mathbf{p}_s + \cdots + \mathbf{p}_e)$.
   - b) If all the points are contained in a circle of radius ϵ around the midpoint, then the cluster is replaced by its midpoint.

- c) Otherwise, the cluster is broken into two bisecting sub-clusters separated by $\mathbf{p}^*$ and the procedure is applied recursively.

The flowchart of the strategy is illustrated in Fig. 4.3.

4.1.2 Remove redundant points on a straight line

Only two points are necessary to define a unique straight line. However in the CAM system, more than two points are often sampled along a straight line. These redundant points will result in an overestimation of the initial number of polynomial pieces for the approximating spline. In this section, a strategy to remove the redundant points along a straight line will be discussed. The basic idea is that if $l\,(l \geq 3)$ points are detected to be collinear, we go on checking $l+1$ points until *C2* in the following assumption is not satisfied.

Definition 6. We say that $l+1$ points $\mathbf{p}_i, \ldots, \mathbf{p}_{i+l}\,(l \geq 2)$ are approximately on a straight line if

- (*C1*) l points $\mathbf{p}_i, \ldots, \mathbf{p}_{i+l-1}$ are on a straight line,
- (*C2*) the distances d_{il}^j of the interior points $\mathbf{p}_j$, $j = i+1, \ldots, i+l-1$ to the chord $\overline{\mathbf{p}_i\mathbf{p}_{i+l}}$ is smaller than the compressor tolerance ϵ.

Algorithm 5. Remove collinear points:
Given are the updated points $\mathbf{p}_1, \ldots, \mathbf{p}_n$ after clusters modification.

```
Initialize i = 1 and l = l_0 = 2
while i < n − 2 do
  Compute d_{il_0}^{i+1}
  if d_{il_0}^{i+1} ≥ ε then
    i = i + 1
  else
    while d_{il}^j < ε (j = i + 1, ..., i + l − 1) & i < n − l do
      l = l + 1
      Compute updated d_{il}^j (j = i + 1, ..., i + l − 1)
    end while
    Remove the interior redundant points p_{i+1}, ..., p_{i+l−1}
    Let i = i + l and l = 2
  end if
end while
```

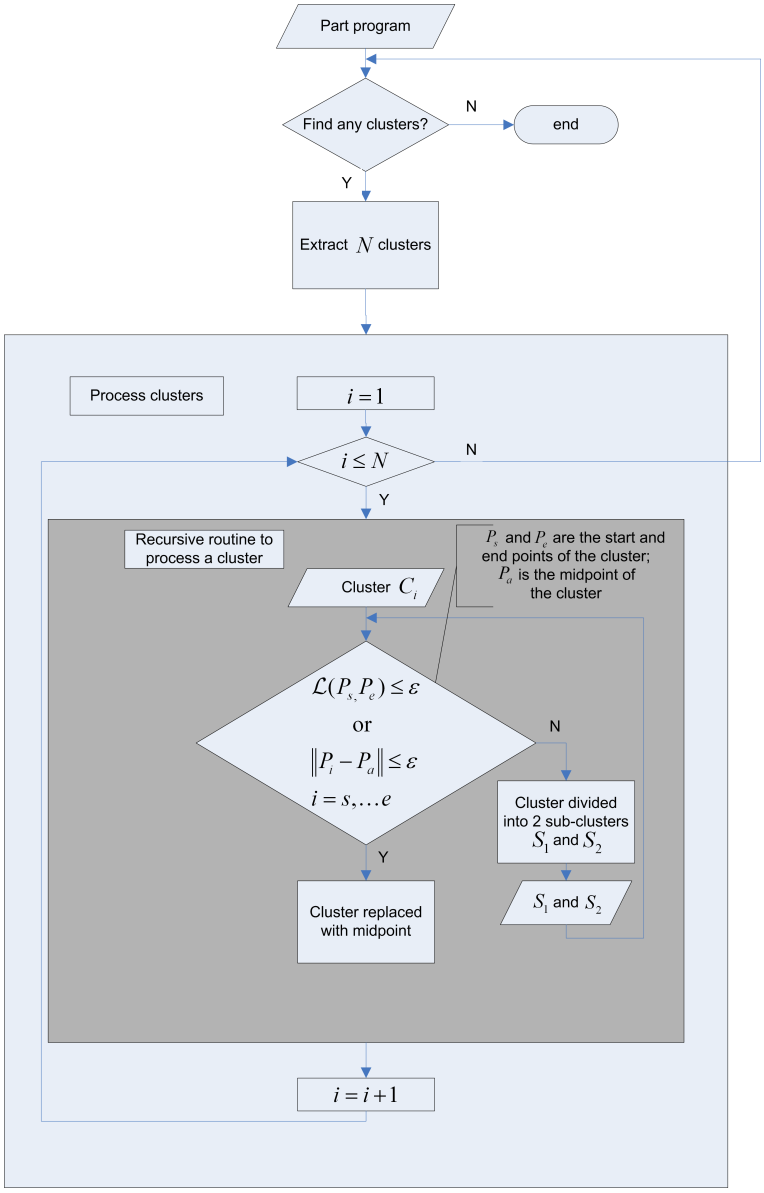


Figure 4.3: Flowchart of the cluster modification method

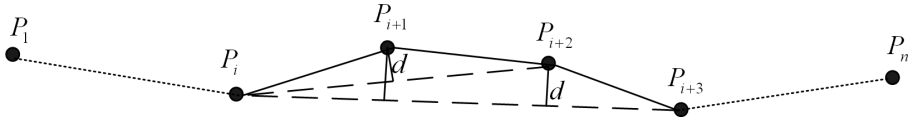


Figure 4.4: Detection of redundant points on a straight line

4.2 Localization

4.2.1 Local knot modification

In Section 3.3, a global knot placement strategy based on curvature is discussed. When the tolerance condition is not satisfied, i.e. the error of the refined control points between the degree elevated linear spline and the knot-inserted approximating spline $\delta := \max_j \|(\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\|$ is larger than the tolerance ϵ, the interior knots are updated globally with an increased number of knots. The weak point of the global strategy is that lots of redundant knots are placed at the regions where the tolerance requirement is already satisfied and the positions of the simple knots are changed completely. To overcome the weakness of the global strategy, a local knot modification scheme is employed to insert additional knots only at the regions where the tolerance condition is violated.

The local modification scheme is based on the local support property of spline curves. Due to this property, spline curves can be broken into segments and investigated and modified locally. The idea of our scheme is to modify the knots locally instead of really splitting the curve into segments. The specific scheme is as follows:

We estimate the errors between the linear spline curve and the *target spline curve* by evaluating the errors between the refined control points $e_j = \|((\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\|$, $j = 1, \ldots, n''$ and the corresponding *violating knot interval* is constructed from the refined knots $T_f = \{t_1^f, \ldots, t_{n''+m+1}^f\}$. We first find the indices

$$J = \{j : \ \|(\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\| > \epsilon\},$$

where the errors between the refined control points exceed the tolerance. We call J the set of *violating indices* of the refined control points. Now only the curve region affected by the corresponding violating control points should be modified to satisfy the tolerance band condition. For example, if e_j is larger than the tolerance ϵ, only the curve region between the knots t_j^f and t_{j+m+1}^f should be modified, which means that the knot density of the target spline $\mathbf{f}$ in the corresponding violating knot interval $I_j^f := [t_j^f, t_{j+m+1}^f]$, $j \in J$, should be increased. In the implementation, the following points must be taken into consideration:

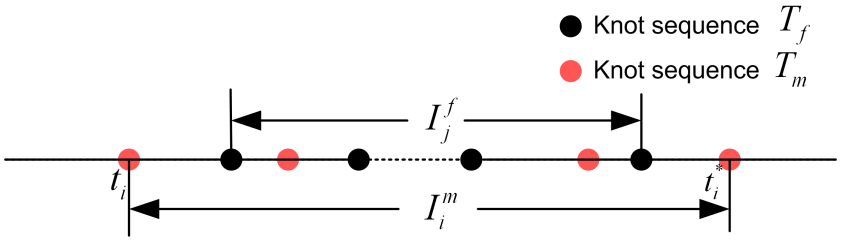


Figure 4.5: Map the violating intervals from T^f to T^m

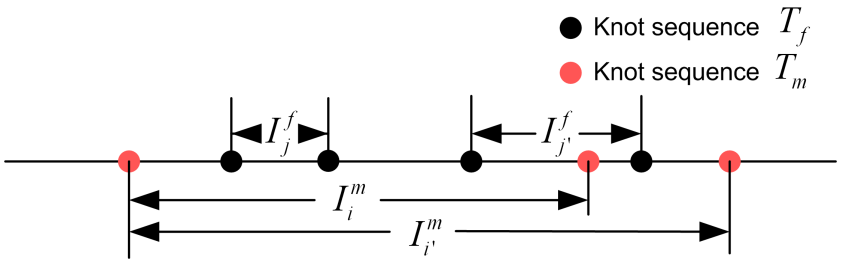


Figure 4.6: Overlapped violating intervals in T^m

1. Given the violating indices $j \in J$ of the refined control points, we first find all the corresponding violating knot intervals I_j^f in the refined knot sequence T_f influenced by the violating control points. If neighboring elements j and $j' \in J$ are too close to each other, say

$$j' - j \leq m + 1, \qquad \text{i.e.} \qquad I_j^f \cap I_{j'}^f \neq \emptyset,$$

   we connect the overlapped violating knot intervals by replacing I_j^f by $(I_j^f \cup I_{j'}^f) = [t_j^f, t_{j'+m+1}^f]$ and removing j' from the violating indices J.

2. Since the knot sequence to be modified is $T_m = \{t_1, \ldots, t_{n+m+1}\}$ of the target spline curve $\mathbf{f}$, the violating knot intervals in the refined knot sequence must be mapped to those in the knot sequence T_m of the target spline curve. For each $j \in J$, we need to find a *smallest* violating knot interval $I_j^m := [t, t^*]$ in T_m satisfying $I_j^f \subseteq I_j^m$, in other words, t and t^* are the largest possible knot and smallest possible knot respectively in T_m such that $I_j^f \subseteq [t, t^*]$. If the neighboring violating knot intervals I_j^m and $I_{j'}^m$ get overlapped as shown in Fig. 4.6, one union violating interval $I_j^m \cup I_{j'}^m$ is again used to replace these two overlapped blocks.

The knots in the violating knot intervals I_j^m are distributed according to the same rule as in Section 3.3 with increased number of interior knots. The other knots lying outside the violating blocks are kept the same. In each iteration step, the violating knot intervals are updated and the knots are locally modified in the corresponding spans until the tolerance requirement is satisfied for the whole curve.

Algorithm 6. Local knot modification scheme: modify T_m locally to satisfy the tolerance condition.

1. Evaluate the errors between the refined control points $e_j = \|((\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\|,\ j = 1, \ldots, n''$ to find the violating indices

$$J = \{j :\ \|(\mathbf{A}_2\mathbf{d})_j - (\mathbf{A}_1\mathbf{E}\mathbf{p})_j\| > \epsilon\}.$$

   If J is empty, then *break*.

2. Generate the violating knot intervals I_j^f, $j \in J$ in T^f.

3. Map the violating knot intervals I_j^f from the refined knot sequence T_f to the violating intervals I_j^m of the target knot sequence T_m.

4. Update the knots locally in the violating knot intervals I_j^m by placing an increased number of interior knots according to the same knot distribution rule based on curvature characteristics.

5. Apply the procedure iteratively until the tolerance condition is satisfied.

The Fig. 4.7 gives an example to illustrate the local knot modification scheme. The data points are extracted from the workpiece 'turm' and tolerance is chosen to be 3μm. The curve segments between the markers 'square' is the region where the tolerance condition is violated. In each iteration step, an additional knot is inserted locally in the corresponding region until the tolerance requirement is satisfied and the rest of the knots are kept the same.

The comparison between global knot placement and local knot placement is illustrated in Fig. 4.8. Compared to the global knot placement strategy, we can reduce the number of polynomial pieces significantly by local knot placement while maintaining the same quality of approximation.

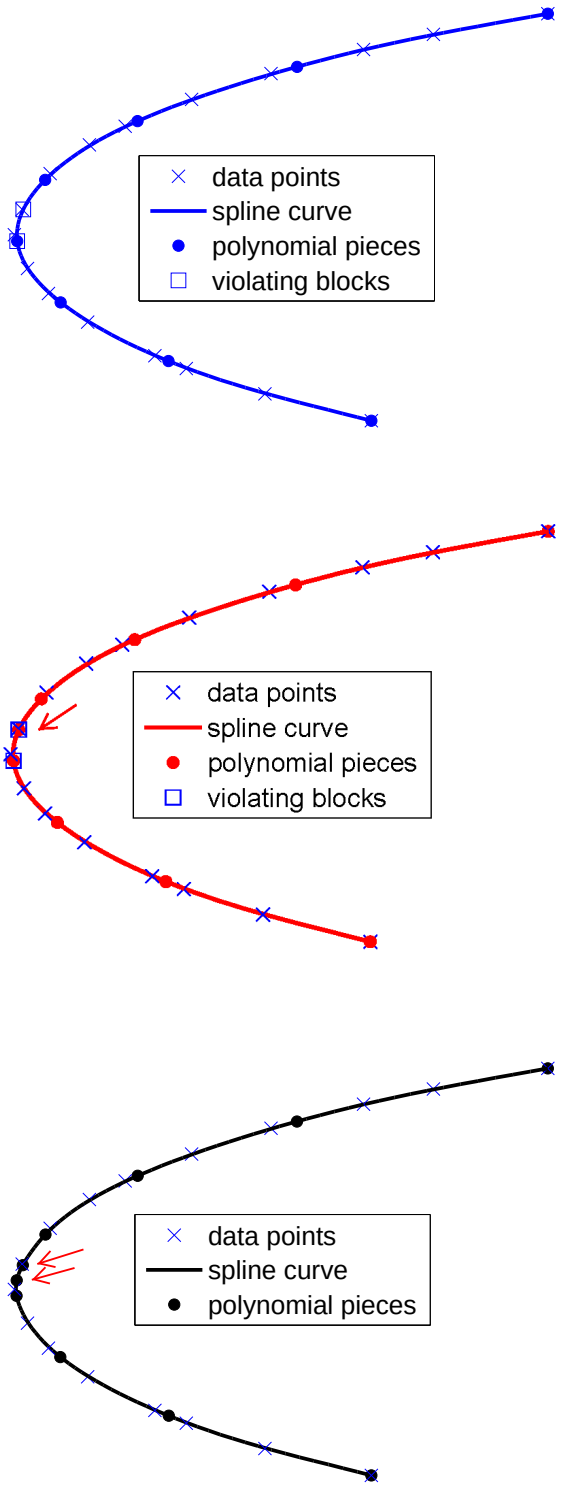


Figure 4.7: Local modification scheme

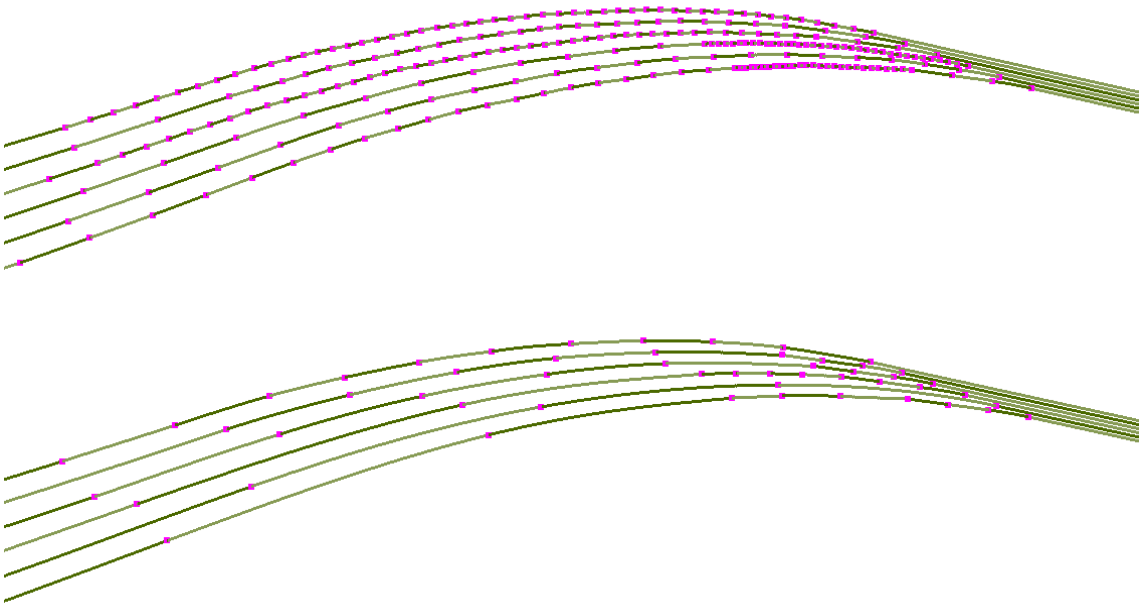


Figure 4.8: Top: global knot placement; Bottom: local knot placement

4.2.2 Local smoothness

In order to achieve more flexibility in the smoothness of the spline curve, we try setting the smoothing weights locally according to the respective knot intervals of the spline curve instead of a global smoothing weight for the whole spline curve. For instance, we may want to obtain stronger smoothing effect in the regions with higher/lower curvature or higher/lower curvature variation. The local smoothness term to be minimized is as follows:

$$\lambda \sum_{i=1}^{n} \int_{b_i}^{b_{i+1}} c_i |\mathbf{f}''(u)|^2 \, du, \tag{4.1}$$

where b_i, $i = 1, \ldots, n+1$ are the breaks of the knot sequence and we tried setting the weight c_i in three different ways:

1. $c_i = 1/(b_{i+1} - b_i)$,
   (c_i is reciprocal to the knot intervals, which can be taken as approximately proportional to the curvature.)

2. $c_i = \|\mathbf{f}'''(u)\|$, $u = (b_i + b_{i+1})/2$,
   (c_i can be taken as approximately proportional to the curvature variation.)

3. $c_i = 1/(C + \|\mathbf{f}'''(u)\|)$, $u = (b_i + b_{i+1})/2$.
   (c_i can be taken as approximately reciprocal to the curvature variation.)

The functional (4.1) can be written as:

$$\begin{aligned} & \mathbf{d}^T \left(\sum_{i=1}^{n} c_i \int_{b_i}^{b_{i+1}} \mathbf{N}'''(u)^T \mathbf{N}'''(u)\, du \right) \mathbf{d} \\ = \; & \mathbf{d}^T \left(\sum_{i=1}^{n} c_i \sum_{k=1}^{l} \mathbf{N}'''(\hat{x_{ik}})^T \hat{w_{ik}} \mathbf{N}'''(\hat{x_{ik}}) \right) \mathbf{d} \\ =: \; & \mathbf{d}^T \mathbf{M} \mathbf{d}, \end{aligned}$$

which can be computed the same way as in Section 2.2.4 except with modified $\mathbf{W} = \mathrm{diag}(c_1\hat{w_{11}}, c_1\hat{w_{12}}, \ldots, c_1\hat{w_{1l}}, c_2\hat{w_{21}}, c_2\hat{w_{22}}, \ldots, c_2\hat{w_{2l}}, \ldots, c_n\hat{w_{n1}}, c_n\hat{w_{n2}}, \ldots, c_n\hat{w_{nl}})$.

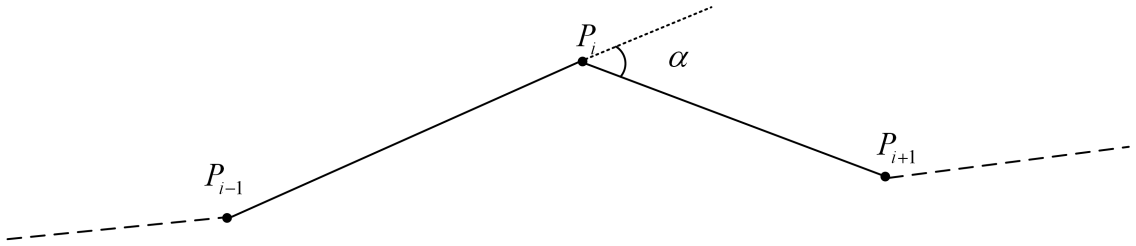
When we compare these three local smoothness methods and the global smoothness method, we cannot find obvious difference in the curvature plot of the resulting spline curves, given a specified tolerance condition. The difference is the value of the weight λ (with normalized c_i) needed to obtain the smoothest spline curve within the tolerance band. The phenomenon $\lambda_2 < \lambda_1 < \lambda_{\text{global}} < \lambda_3$ can imply that the second local smoothness method may have stronger smoothing effect than the others.

4.3 Soft edge detection and quasi multiple knots

In Section 3.2, we discussed the scheme of hard edge detection, where only one threshold is set for the opening angle to determine whether it is an edge or not. The hard decision scheme will run into problems, especially when the opening angles from path to path swing back and forth around the threshold. Even though the opening angles of two neighboring paths are quite close to each other, i.e. one is slightly above the threshold and the other is slightly below the threshold, there exists a clear cut between edges and non-edges, which will result in non-smooth and inconsistent transitions from path to path and thus spoil the milling result. That is why soft edge detection method is considered.

The basic idea is to use two thresholds to obtain a soft decision:

- If the opening angle is larger than the upper threshold τ_1, we call it a definite edge or a sharp edge. Then we do curve segmentation similar to hard edge detection and break the path here as separate curve segments.



- If the opening angle is smaller than the lower threshold τ_0, then there is no edge here. Therefore we need not place additional knots at this position.

- If the opening angle is between τ_0 and τ_1, it is ambiguous to identify if it as an edge or not. Accordingly we place the so-called *quasi m-fold knots* as shown in Fig. 4.9, which are $m-2$-fold knots in the middle and two simple knots at two sides. Since the side knots are quite close to the middle knots, they can generate a fake edge which is in effect a short curve segment with high curvature to imitate a definite edge. When the knot interval d gets smaller, then the quasi m-fold knots converge to the m-fold knots and there appears a definite edge. In addition, the quasi m-fold knots are symmetric, therefore the scheme is independent of the processing direction of the path.

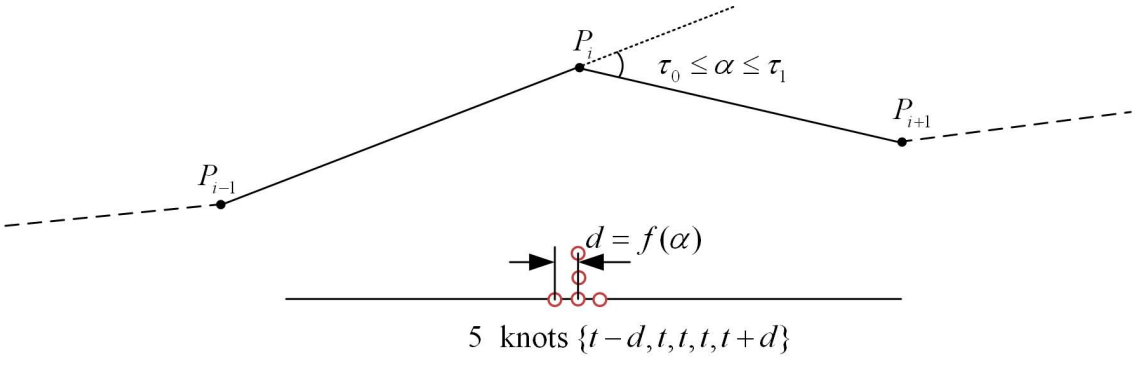


Figure 4.9: Knot placement for a soft edge decision

Fig. 4.10 illustrates the knot placement scheme when both $m-1$-fold knots and quasi m-fold knots are present and the simple knots are distributed in the same way based on curvature characteristics, see Section 3.3. If the opening angle is between two thresholds and meanwhile a curvature jump is also detected at the same position, then we choose placing quasi m-fold knots instead of $m-1$-fold knots.

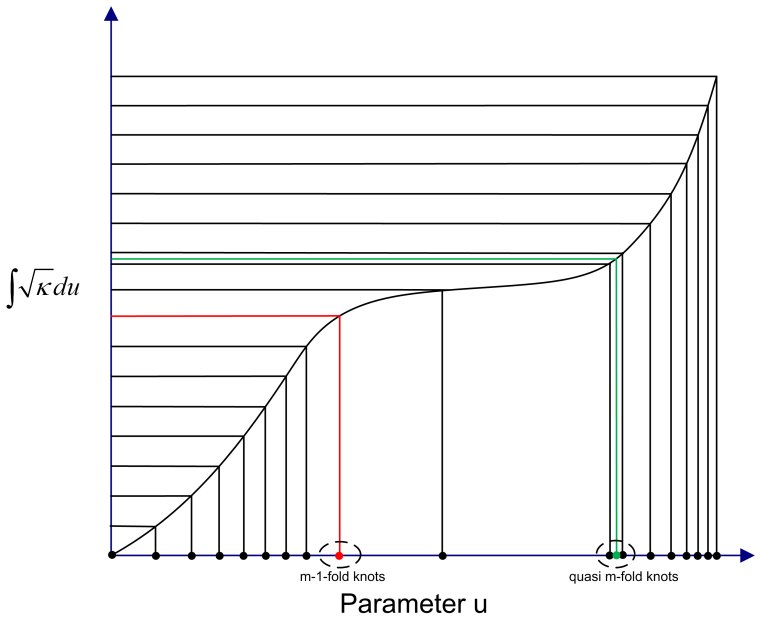


Figure 4.10: Knot placement with quasi multiple knots

The knot interval d is not a fixed value but defined as a decreasing function of the opening angle α. The simplest choice is a linear function as illustrated in Fig. 4.11:

$$f(\alpha) = \frac{\epsilon_1 - \epsilon_0}{\tau_1 - \tau_0}(\alpha - \tau_0) + \epsilon_0.$$

If ϵ_1 is set to be zero, then the fake edge can converge to a definite edge. However in the implementation, ϵ_1 is often chosen as the smallest tolerable length rather than zero, since too tiny pieces limit the feed rate significantly, which is explained in Chapter 1.

4.4 Curvature estimation methods

There are various ways to estimate the discrete curvature. In Section 3.3.1, the basic method called *circumcircle* is described and in this section we will discuss some alternative methods.

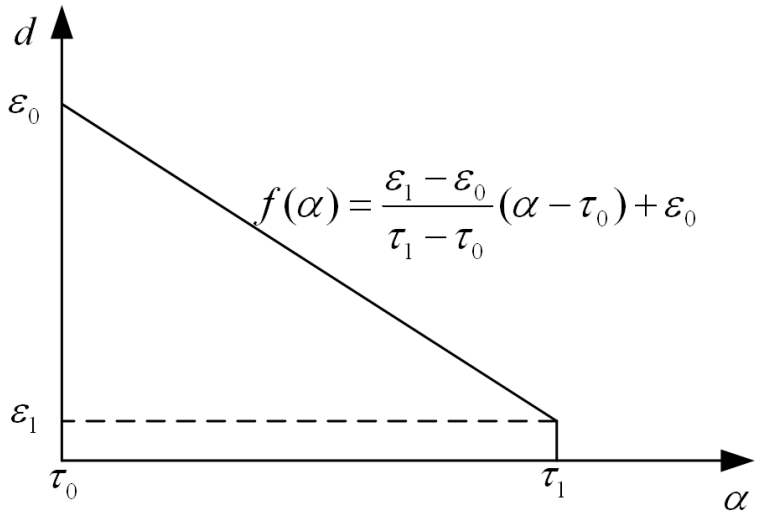


Figure 4.11: The relation between the knot distance d and opening angle α

4.4.1 Divided difference

For a smooth curve $\mathbf{f}$, which is at least C^2 continuous, the curvature is defined as the second derivative of $\mathbf{f}$ with respect to the arc length s

$$\kappa = \ddot{\mathbf{f}}(s).$$

In the discrete case, the unit tangent vector is calculated approximately as

$$\mathbf{q}_i \approx \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{\left\|\mathbf{p}_i - \mathbf{p}_{i-1}\right\|} \qquad \text{and} \qquad \mathbf{q}_{i+1} \approx \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\left\|\mathbf{p}_{i+1} - \mathbf{p}_i\right\|}.$$

For a coarse estimation, the curvature is approximated by a second divided difference:

$$\kappa \approx \frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{\frac{1}{2}(\left\|\mathbf{p}_i - \mathbf{p}_{i-1}\right\| + \left\|\mathbf{p}_{i+1} - \mathbf{p}_i\right\|)}$$

Both the circumcircle method and the divided difference method utilize particular concepts of discrete differential geometry that are very sensitive to noise. A minor perturbation in the discrete data can result in great error in the curvature estimation. As shown in Fig. 4.12, the original data points $\mathbf{p}_o$ are sampled on a circle of radius 5 and the noisy points $\mathbf{p}_n$ are generated randomly on circles of radius 0.05 around the respective original points

$$\mathbf{p}_n = \mathbf{p}_o + 0.05 \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix},$$

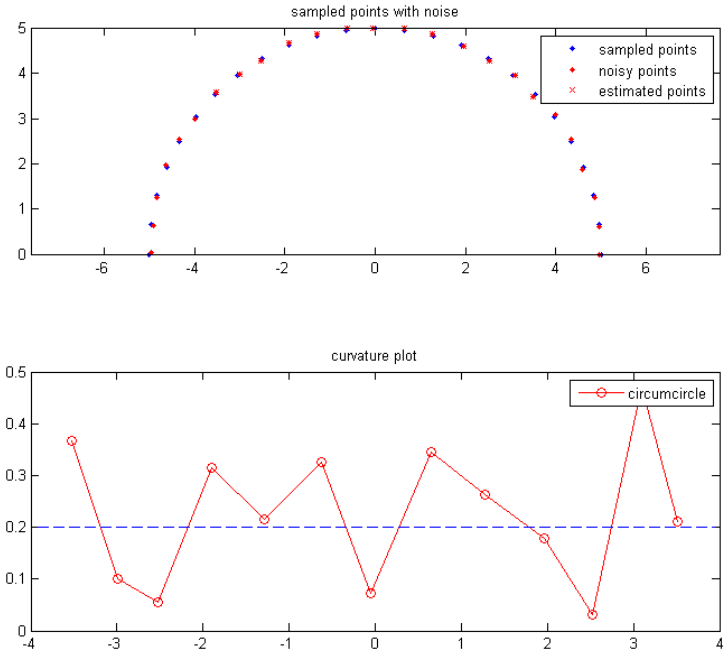


Figure 4.12: Significant error when estimating curvature of noisy data using the circumcircle method

where θ is chosen randomly between 0 and 2π. From the figure, we can see that even with minor errors added to the original data, the curvature estimation deviates significantly from the accurate curvature value 0.2.

Now we will investigate how the curvature estimation using the circumcircle method is affected by a perturbation of the data. The curvature estimated by the circumcircle method is

$$\kappa = \frac{2\sqrt{1-x^2}}{\|\mathbf{p}_{j+1} - \mathbf{p}_{j-1}\|}, \qquad \text{where} \qquad x = \frac{(\mathbf{p}_{j+1} - \mathbf{p}_j) \cdot (\mathbf{p}_j - \mathbf{p}_{j-1})}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\| \|\mathbf{p}_j - \mathbf{p}_{j-1}\|}. \tag{4.2}$$

Let $(\mathbf{p}_{j+1} - \mathbf{p}_j) \cdot (\mathbf{p}_j - \mathbf{p}_{j-1}) =: a$ and $\|\mathbf{p}_{j+1} - \mathbf{p}_j\| \|\mathbf{p}_j - \mathbf{p}_{j-1}\| =: c$. Even if we consider the simple case of a perturbation ς added only to the middle point $\mathbf{p}_j$, we get

$$\begin{aligned} \hat{a} &= (\mathbf{p}_{j+1} - (\mathbf{p}_j + \zeta)) \cdot (\mathbf{p}_j + \zeta - \mathbf{p}_{j-1}) \\ &= (\mathbf{p}_{j+1} - \mathbf{p}_j) \cdot (\mathbf{p}_j - \mathbf{p}_{j-1}) + \zeta(\mathbf{p}_{j+1} - 2\mathbf{p}_j + \mathbf{p}_{j-1}) - \|\zeta\|^2 \end{aligned}$$

and

$$\begin{aligned}\hat{c} &= \|\mathbf{p}_{j+1} - (\mathbf{p}_j + \zeta)\|\|(\mathbf{p}_j + \zeta) - \mathbf{p}_{j-1}\| \\ &\geq (\|\mathbf{p}_{j+1} - \mathbf{p}_j\| - \|\zeta\|)(\|\mathbf{p}_j - \mathbf{p}_{j-1}\| - \|\zeta\|) \\ &= \|\mathbf{p}_{j+1} - \mathbf{p}_j\|\|\mathbf{p}_j - \mathbf{p}_{j-1}\|(1 - \frac{\|\zeta\|}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\|})(1 - \frac{\|\zeta\|}{\|\mathbf{p}_j - \mathbf{p}_{j-1}\|}).\end{aligned}$$

Therefore

$$e = \hat{x} - x = \frac{\hat{a}}{\hat{c}} - x$$

$$\leq \frac{(\mathbf{p}_{j+1} - \mathbf{p}_j) \cdot (\mathbf{p}_j - \mathbf{p}_{j-1}) + \zeta \cdot (\mathbf{p}_{j+1} - 2\mathbf{p}_j + \mathbf{p}_{j-1}) - \|\zeta\|^2}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\|\|\mathbf{p}_j - \mathbf{p}_{j-1}\|(1 - \frac{\|\zeta\|}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\|})(1 - \frac{\|\zeta\|}{\|\mathbf{p}_j - \mathbf{p}_{j-1}\|})} - \frac{(\mathbf{p}_{j+1} - \mathbf{p}_j) \cdot (\mathbf{p}_j - \mathbf{p}_{j-1})}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\|\|\mathbf{p}_j - \mathbf{p}_{j-1}\|}. \tag{4.3}$$

From (4.3) we can see that if the noise ζ is significantly smaller than the distances between the points and especially when the noise vector is perpendicular to the vector $(\mathbf{p}_{j+1} - 2\mathbf{p}_j + \mathbf{p}_{j-1})$, the additional term in $\hat{x}$ compared to x is neglectable and the circumcirle method can give a relatively precise estimation. In contrary, if the data points are so dense that the distances between them are almost comparable with the noise, the term $(1 - \frac{\|\zeta\|}{\|\mathbf{p}_{j+1} - \mathbf{p}_j\|})(1 - \frac{\|\zeta\|}{\|\mathbf{p}_j - \mathbf{p}_{j-1}\|})$ can become close to zero and the error e may deviate from zero considerably.

Since curvature estimation based on discrete differential geometry is very sensitive to noise, a different approach to use integral rather than differentiation to estimate curvature will be taken into consideration in the next subsection.

4.4.2 Area invariant and Connolly function for planar curves

The *area invariant* method is a special case of *integral invariant* methods, which was first introduced by Manay et al. in [19] and further investigated in [23]. The idea is to estimate the curvature of a curve $\mathcal{C}$ at a point $\mathbf{p}$ via computing the area $A_r(\mathbf{p})$, defined as the intersection between the circle of radius r and the curve $\mathcal{C}$, see Fig. 4.13. The relation between $A_r(\mathbf{p})$ and the curvature is

$$A_r = \frac{\pi}{2}r^2 + \frac{\kappa}{3}r^3 + O(r^4), \tag{4.4}$$

where r is the radius of the *checking circle*, κ is the estimated curvature and A_r is the area of intersection of the curve and the checking circle.

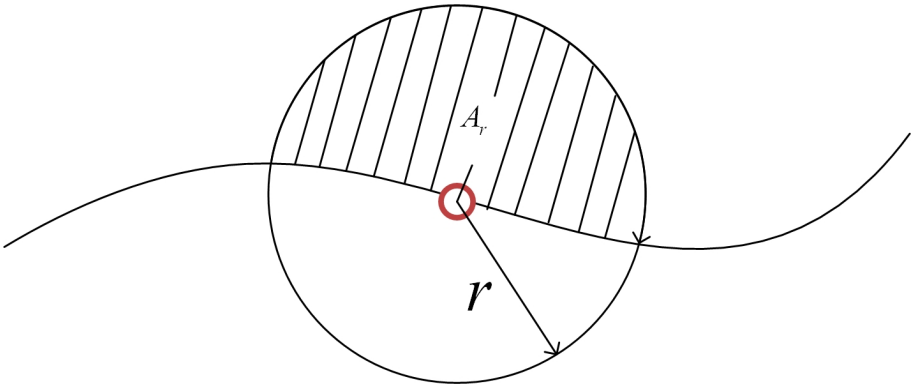


Figure 4.13: Area invariant method for curvature estimation of planar curves

In order to derive (4.4), the so-called Connolly function will be first introduced. Let $\mathcal{C}$ be a smooth arc length s parametrized curve in $\mathbb{R}^2$. As shown in Fig. 4.14, the circle of radius η, centered at point $\gamma(s)$ intersects the curve $\mathcal{C}$ at two points $\gamma(t^-(s))$ and $\gamma(t^+(s))$ with $t^-(s) < s < t^+(s)$. The angles between the tangent vector $\gamma'(s)$ and the vector $\overline{\gamma(s)\gamma(t^+(s))}$, $\overline{\gamma(t^-(s))\gamma(s)}$ are denoted by $\theta^+(s)$ and $\theta^-(s)$ respectively.

Definition 7. The *Connolly function* is defined as the arc length between two intersections $t^-(s)$ and $t^+(s)$ normalized by η, which is equivalent to the angle

$$\Phi(s) = \pi + \theta^+(s) + \theta^-(s). \tag{4.5}$$

Lemma 9. *For a fixed arc length s, one has:*

$$t(s,\eta) = s + \epsilon\eta + o(\eta^2), \qquad \epsilon = +1 \ or \ -1 \tag{4.6}$$

$$\theta'(s,\eta) = \frac{\partial\theta}{\partial s}(s,\eta) = \frac{1}{2}\frac{d\kappa}{ds}(s)\eta + O(\eta^2). \tag{4.7}$$

According to Lemma 9, we can get the Connolly function: $\Phi = \pi + \kappa\eta + O(\eta^2)$. Given the Connolly function, the arc length between the intersections is $\Phi\eta$:

$$L_\eta = \pi\eta + \kappa\eta^2 + O(\eta^3). \tag{4.8}$$

The area is computed by integrating (4.8) over η:

$$A_r = \int_0^r L_\eta \, d\eta = \frac{\pi}{2}r^2 + \frac{\kappa}{3}r^3 + O(r^4). \tag{4.9}$$

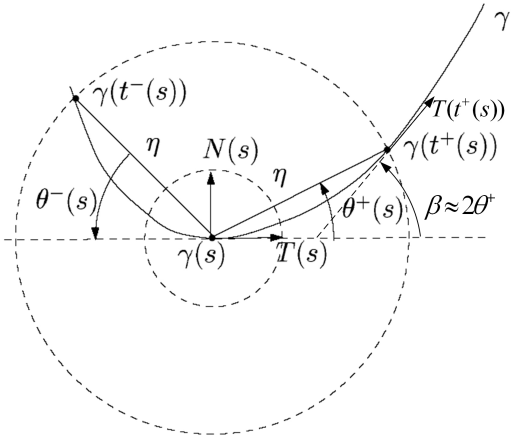


Figure 4.14: Connolly function

The function (4.4) is only suitable for estimating curvature for smooth curve, which means tangentially continuous here. If the curve $\mathcal{C}$ is not smooth but consists of two curvature continuous pieces joined at $\mathbf{p}$ as shown in Fig. 4.15, the left and right limit curvature κ_- and κ_+ can be computed by a 'semicircle method':

$$A_r{}^+ = \frac{\pi}{4}r^2 + \frac{\kappa^+}{6}r^3 + O(r^4),$$

$$A_r{}^- = \frac{\pi}{4}r^2 + \frac{\kappa^-}{6}r^3 + O(r^4).$$

Then,

$$\begin{aligned} A_r &= A_r{}^+ + A_r{}^- + \tfrac{\beta}{2}r^2 \\ &= \tfrac{\pi}{2}r^2 + \tfrac{\pi-\alpha}{2}r^2 + \tfrac{\kappa^+ + \kappa^-}{6}r^3 + O(r^4) \\ &= \tfrac{2\pi-\alpha}{2}r^2 + \tfrac{\kappa^+ + \kappa^-}{6}r^3 + O(r^4) \end{aligned} \tag{4.10}$$

When the curve is not smooth, (4.10) instead of (4.9) is used to estimate the mean curvature at the curvature discontinuous point.

4.4.3 Area invariant method vs circumcircle method

One important property of the *integral invariant* method is its robustness against noise, since the area invariant method, being principally based on integration, has a smoothing

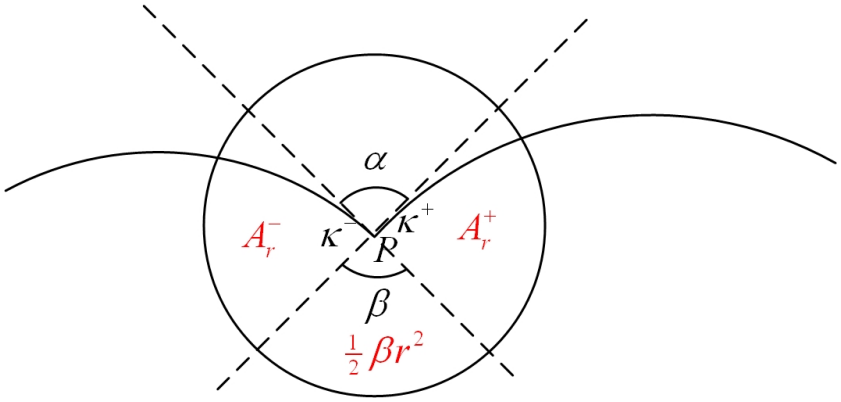


Figure 4.15: Area invariant method for a non-smooth curve

effect. Another advantage is that we can get to know the multi-scale behavior when we choose different checking radii for the estimation.

(4.4) gives an estimate of the curvature at scale r. With various radii r, we can exploit the behavior of the curve at different scales. However, how to choose appropriate checking radii is not an easy task. With larger checking circles, the curvature estimate is smoother and more robust to noise, while with smaller checking circles, the curvature estimate is relatively more accurate but more sensitive to noise. In our implementation, three checking circles are chosen heuristically to estimate the curvature. The smallest radius r_1 is chosen according to the smallest distance between the sampled points. And the checking radii r_2 and r_3 are twice and three times the smallest radius r_1 respectively. The least squares result with respect to the three different radii can then be taken as the final estimate of the curvature.

$$\begin{bmatrix} r_1^3 \\ r_2^3 \\ r_3^3 \end{bmatrix} (\frac{\kappa}{3}) = \begin{bmatrix} A_{r_1} - \frac{\pi}{2} r_1^2 \\ A_{r_2} - \frac{\pi}{2} r_2^2 \\ A_{r_3} - \frac{\pi}{2} r_3^2 \end{bmatrix}. \tag{4.11}$$

The multi-scale property of the area invariant method and a comparison with circumcirle method are illustrated in the following examples. As shown in Fig. 4.16, 25 points were sampled uniformly on a semicircle of radius 5. Both the circumcircle method and least squares solution of the area invariant method can give a precise estimate of the curvature.

In Fig. 4.17 minor noise of magnitude 0.05 was imposed to the points in Fig. 4.16 and we can see that least squares solution of the area invariant method is more robust

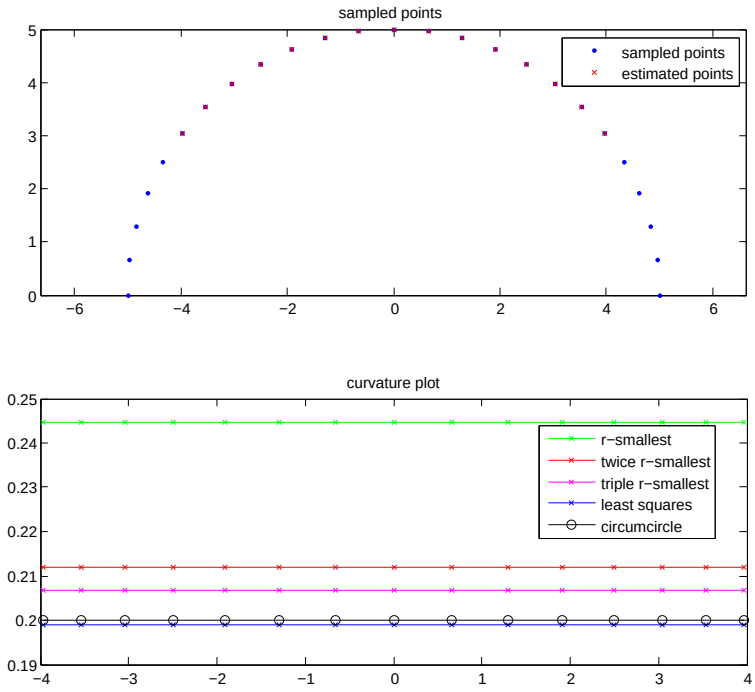


Figure 4.16: Example 1(a): area invariant vs circumcircle method

to noise than the circumcircle method. For the area invariant method at different scales r, the curvature estimate becomes more robust to noise with increasing checking radii, since the smoothing effect becomes stronger with larger r.

The next example in Fig. 4.18 is to show if the area invariant method can detect the curvature jumps at the junction of the arc and straight line correctly, since the integral method may encounter underdetection with large checking radii. In the Fig. 4.18, a sharp jump can be detected with a relatively small radius. With increasing radii, the curvature gets smoothed out, which may result in underdetection of the curvature jumps, while the circumcircle method can estimate the jumps more precisely in such a case.

In the third example, the curvature estimate is performed on real world data (workpiece of 'beetle'). From Fig. 4.19, we can see that the curvature plot with the area invariant method is much smoother than the one obtained by the circumcircle method.

In summary, the area invariant method exhibits two important properties:

- Robustness with respect to noise with appropriate checking radii,

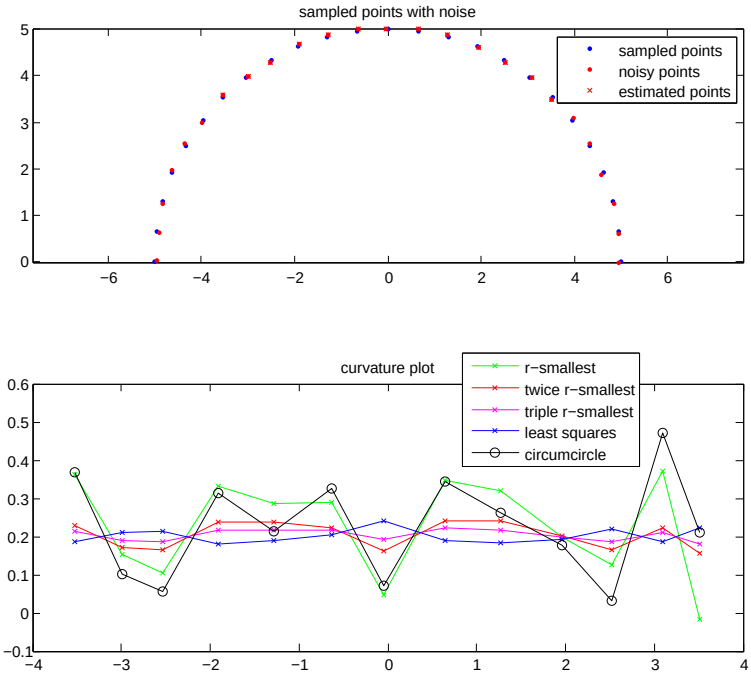


Figure 4.17: Example 1(b): area invariant vs circumcircle method on noisy data

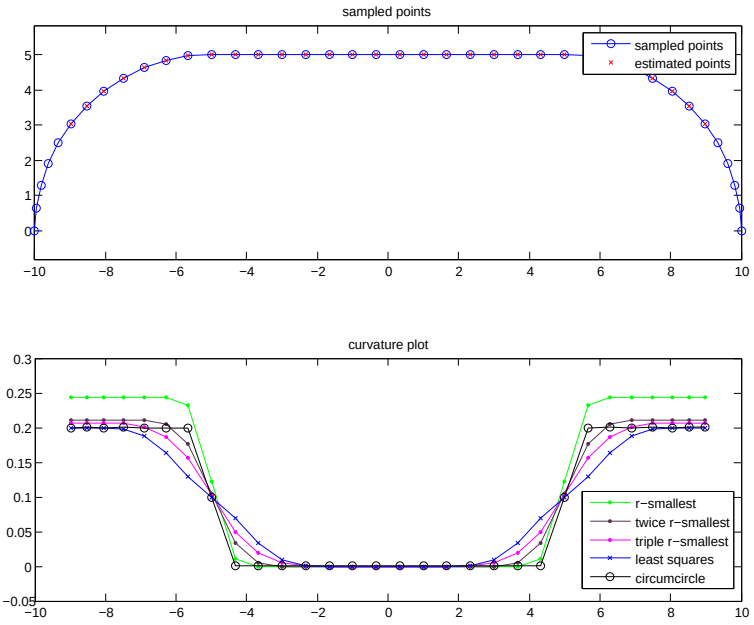


Figure 4.18: Example 2: area invariant vs circumcircle method for curvature jump detection

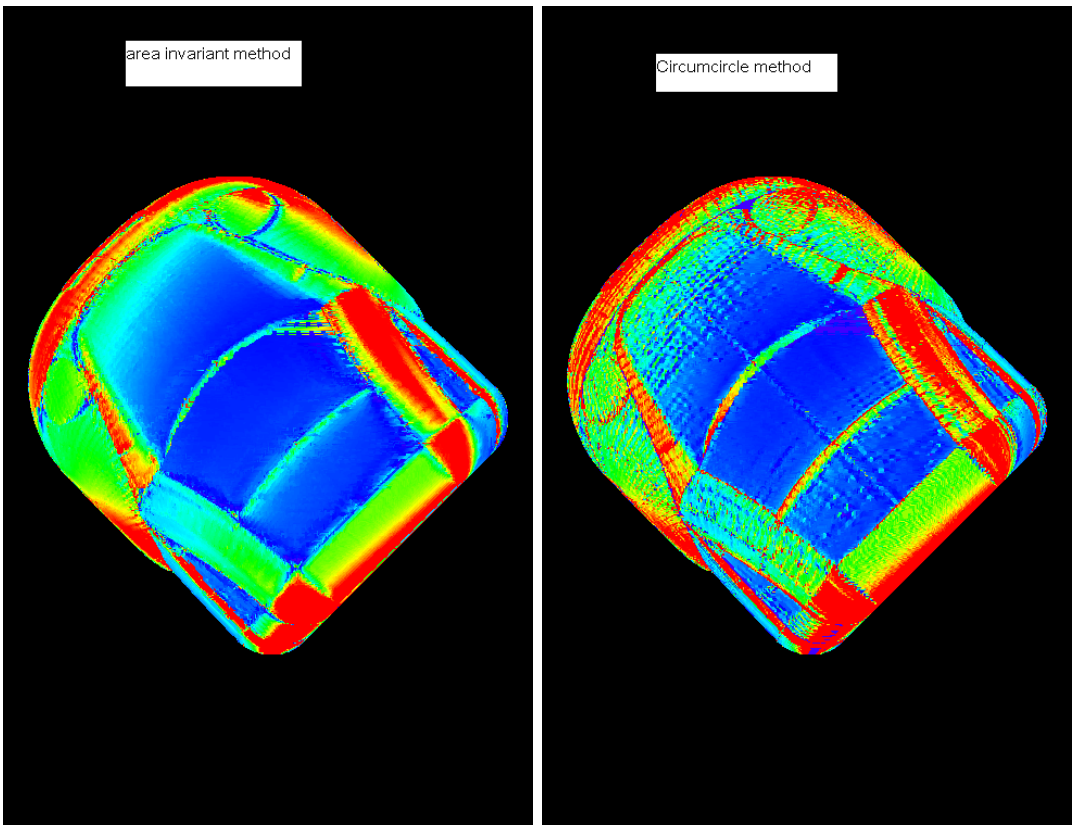


Figure 4.19: Example 3: area invariant vs circumcircle method on workpiece 'beetle'

circumcircle method	area invariant method
- sensitive to noise	+ robust to noise
- overdetection of curvature jumps	- underdetection of curvature jumps
+ lower computational complexity	- higher computational complexity

Table 4.1: Comparison between circumcircle and area invariant method

- Multi-scale behavior.

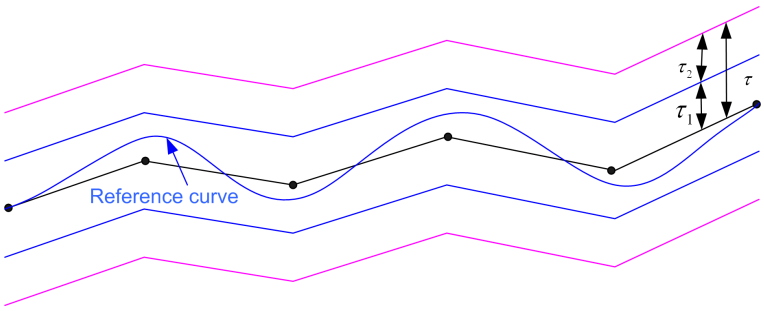
Meanwhile we cannot ignore the weakness of it:

- Higher computational effort,
- Underdetection of curvature jumps with too large checking radius,
- Wrong estimation if edges (tangent jumps) are not detected correctly in advance,
- Difficulty to choose appropriate checking radii.

The positive and negative points of the circumcircle and area invariant method are listed in Table 4.1. Due to the promising advantages, the area invariant method is worthwhile being investigated, while the circumcircle method is chosen as standard routine for curvature estimation because of efficiency.

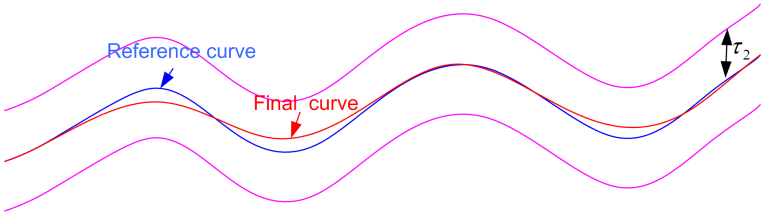
4.5 Reference curve and smoothness

One inherent problem is how to get a better balance between approximation and smoothness, since over-smoothing will result in inconsistency between the neighboring paths and visible artifacts in the milling result. To achieve a better compromise between smoothness and consistency between the neighboring paths, we divide the tolerance band τ into two 'sub-bands' τ_1 and τ_2 with $\tau_1 + \tau_2 = \tau$.



1. Find the *reference curve* $\mathbf{s}_r = \sum_{j=1}^{n} \mathbf{d}_j^r N_j^m(u|T_m)$ without a smoothness term which lies in the band τ_1 around the linear spline curve.

2. Find a *target curve* $\mathbf{f}$ as smooth as possible that lies in the band τ_2 around the reference curve. And it is obvious that the target curve is restricted in the tolerance band τ around the linear spline curve.



We keep the knots of the target curve $\mathbf{f} = \sum_{j=1}^{n} \mathbf{d}_j^f N_j^m(u|T_m)$ the same as the knots of the reference curve and minimize the functional:

$$\int \|\mathbf{s}_r - \mathbf{f}\|^2 \, du + \lambda \int |\mathbf{f}'''|^2 \, du. \tag{4.12}$$

Since we get a reference curve to compare with, we can minimize the continuous error between the target spline curve and the reference spline curve instead of the discrete errors between the refined control points as discussed in (3.12). The functional (4.12) can be written in matrix form as:

$$\begin{aligned}
& \int (\mathbf{N}\mathbf{d}_r - \mathbf{N}\mathbf{d}_f)^T (\mathbf{N}\mathbf{d}_r - \mathbf{N}\mathbf{d}_f) \, du + \lambda \int (\mathbf{N}'''\mathbf{d}_f)^2 \, du \\
= & \int \left(\mathbf{d}_f^T \mathbf{N}^T \mathbf{N} \mathbf{d}_f - 2\mathbf{d}_r^T \mathbf{N}^T \mathbf{N} \mathbf{d}_f + \lambda \mathbf{d}_f^T \mathbf{N}'''^T \mathbf{N}''' \mathbf{d}_f \right) du + \mathrm{C} \\
= & \mathbf{d}_f^T \left(\left(\int \mathbf{N}^T \mathbf{N} \, du \right) + \lambda \left(\int \mathbf{N}'''^T \mathbf{N}''' \, du \right) \right) \mathbf{d}_f - 2\mathbf{d}_r^T \left(\int \mathbf{N}^T \mathbf{N} \, du \right) \mathbf{d}_f + \mathrm{C} \\
=: & \mathbf{d}_f^T \left(\mathbf{M} + \lambda \mathbf{M}''' \right) \mathbf{d}_f - 2\mathbf{d}_r^T \mathbf{M} \mathbf{d}_f + \mathrm{C},
\end{aligned}$$

where $\mathbf{M} = \int \mathbf{N}^T \mathbf{N} \, du$ and $\mathbf{M}''' = \int \mathbf{N}'''^T \mathbf{N}''' \, du$ can be calculated by Gaussian quadrature.

The minimization problem:

$$\min_{\mathbf{d}_f} \mathbf{d}_f^T (\mathbf{M} + \lambda \mathbf{M}''') \mathbf{d}_f - 2\mathbf{d}_r^T \mathbf{M} \mathbf{d}_f$$

can again be reduced to solving the normal equation:

$$(\mathbf{M} + \lambda \mathbf{M}^{'''})\mathbf{d}_f = \mathbf{M}\mathbf{d}_r.$$

4.6 Complete methods

As the specific schemes are discussed in the previous sections, the complete methods are summarized in this section.

- Preprocessing, see Section 4.1.
- Curve segmentation, see Section 3.2.
- Process each curve segment.
  - Find the approximating spline in the tolerance band without smoothness term.
    - Knot generation.
      - Curvature estimation, see Section 3.3.1 and 4.4.
      - Knot distribution based on curvature characteristics (global/local), see Section 3.3, 4.2.1 and 4.3.
    - Least squares approximation to compute the control points, see Section 3.4.2.
    - Error evaluation, see Section 3.4.1.
  - Find the smoothest curve within the tolerance band.
    - Least squares approximation with smoothness term, see Section 3.4.3 and 3.4.4.
    - Error evaluation.

5

Experimental results

In this chapter, some experimental results will be presented to illustrate the influence of three important factors: the distribution of knots; the knot's multiplicity and the smoothness term. The spline approximation routine has been implemented in *Matlab*. As inputs it needs a standard CNC part program loaded as ASCII data, a threshold for edge detection, a threshold for curvature jump detection and a tolerance specification. The output spline curves written in Sinumerik 840D native format can be visualized and analyzed by *Visutool*.

5.1 Example 1: the influence of knot's multiplicity

In the first experiment, we consider one path extracted from the workpiece 'daimler', where hard edges are present. The purpose of this experiment is to illustrate how significantly the knot's multiplicity will affect the approximation precision.

In the first case as shown in Fig. 5.1, the hard edges are detected correctly when an appropriate threshold is set for edge detection. We can achieve very accurate approximation at a high compression rate, for instance, we can obtain an axis error less than 6μm by using only 26 polynomial pieces to approximate 220 linear segments.

In the second case as shown in Fig. 5.2, We assume the hard edges are not detected due to a falsely set threshold for edge detection. Instead, the edges are detected as curvature jumps, if we set the threshold for curvature jump detection properly. Therefore 4-fold knots instead of 5-fold knots are placed at curvature jumps and meanwhile other knots are kept the same as in the first case. With 4-fold knots, the spline curve is at

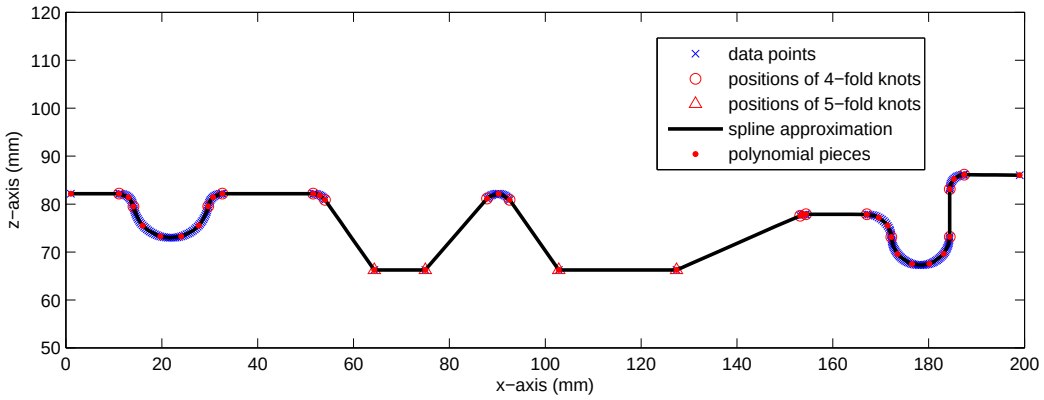


Figure 5.1: Spline approximation with correct edge detection

least C^1 continuous. However, the approximation error goes from 6μm up to 500μm as shown in Fig. 5.4.

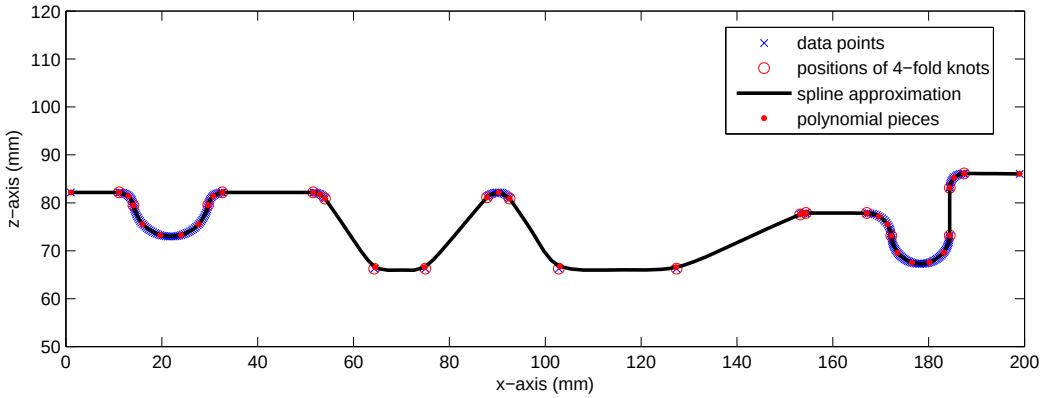


Figure 5.2: Spline approximation with 4-fold knots at edges

In the worst case as shown in Fig. 5.3, only simple knots are placed at the edges. The spline curve is now C^4 continuous at the edges, but it deviates radically from the linear segments and the approximation error is up to 1.5mm.

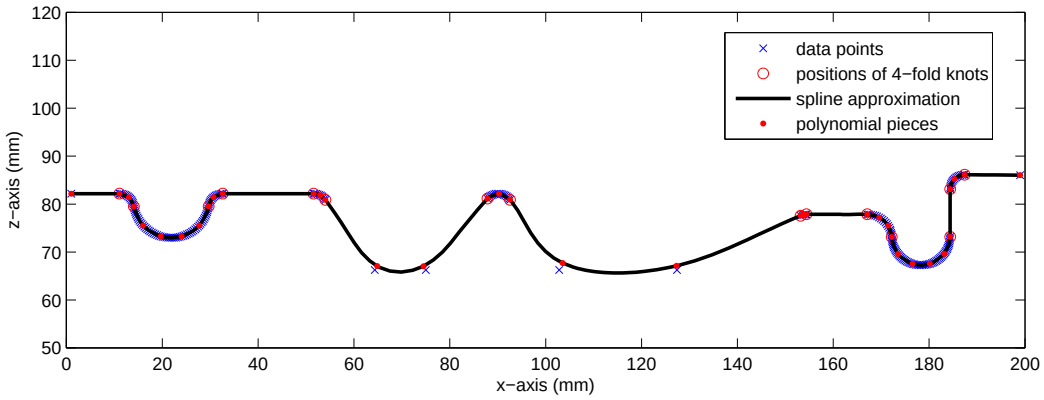


Figure 5.3: Spline approximation with simple knots at edges

5.2 Example 2: the influence of knot distribution

In this experiment, we use another path from workpiece 'daimler' for testing spline approximation with different knot distribution methods. From Fig. 5.5, we can observe that the sampled data points are distributed relevant to the curvature characteristics rather than equally spaced. One approach is to use a knot sequence with uniform simple knots for approximation, as illustrated in Fig. 5.6. For comparison, we employ a knot sequence (see Fig. 5.7) for spline approximation with the same multiple knots and the same number of polynomial pieces, but the breaks are more reasonably distributed cohering with the curvature of the underlying curve. As is evident from the results shown in Fig. 5.6 and Fig. 5.7, we can achieve much better approximation accuracy with the maximum approximation error being 10μm by using knots based on curvature, in contrast to that being 16μm by using uniform knots. Particularly in the linear region A, the approximation error is also greatly reduced with the denser knots in region B due to the compact support property of splines.

As discussed in Section 3.4.3, if a spline curve is nearly arc length parametrized, the variation of curvature can be simplified as the third derivative of the curve. As a by-product, Fig. 5.8 shows the deviation to arc length parametrization of both methods, which is evaluated by $\|f'(u)\| - 1$. The comparison indicates that with knots based on curvature, the resultant spline curve is much closer to being arc length parametrized, especially in the area of higher curvature.

5.3 Example 3: the influence of the smoothness term

In Section 3.4.3, two smoothness criteria *strain energy* and *curvature variation* have been discussed. In this section, we will give the experimental results to present and analyze the performance of these two methods. The test data we use is a workpiece called 'turm', shown in Fig. 5.9. Fig. 5.10 illustrates the discrete curvature of the part program, evaluated with circumcircle method. The curvature and 'torsion' plots of the output spline curves in three different cases such as approximation without smoothing term, with SE minimization and with CV minimization are presented and compared in Fig. 5.11. Notice that the 'torsion' plot in Visutool visualizes, in effect, the variation of curvature rather than the actual torsion as defined in differential geometry.

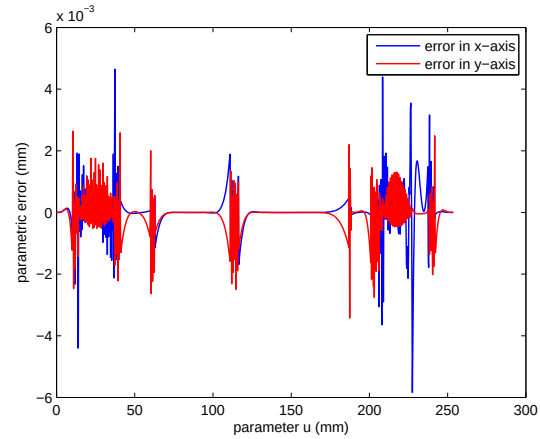
Compared to the discrete curvature of the part program, the curvature plot of the spline curve even without smoothing (Case 1) is smoother and contains less curvature extrema since the spline curve itself has the built-in smoothing effect. From Fig. 5.11 we can clearly see that the curvature plots of the spline curves with both SE and CV methods are considerably smoothed out in contrast to that without smoothness term. Comparing the curvature between case 2 and case 3, we can observe that with CV minimization, the curvature varies even more gradually and homogeneously and the 'torsion' plot contains less extrema which indicates that the jerk of the machine can be further reduced. The obvious difference between the marked area in Fig. 5.11 well represents and agrees with the argument that CV demands to keep the curvature as constant as possible while SE requires to keep the curvature as small as possible.

5.4 The typical test workpieces

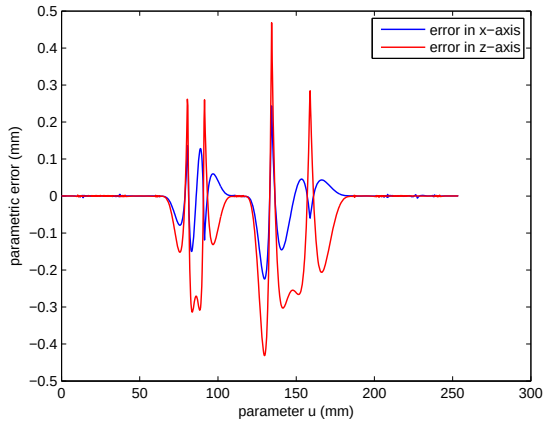
We have tested our spline approximation routine on the test suites and the simulations exhibit satisfactory results. Here we only represent the performance on the typical workpiece 'daimler' and investigate the results in three aspects: the surface quality, compression rate and the curvature plot of the resulting spline curves.

The original part program is shown in Fig. 5.12 and the discrete data points are displayed as green dots. Fig. 5.13 illustrates the compressor output given a tolerance of 10μm. The transitions between the polynomial pieces are highlighted with green dots and it is obvious that the number of polynomial pieces is reduced significantly, especially in the translational surface. Another strength is that we can obtain a consistent

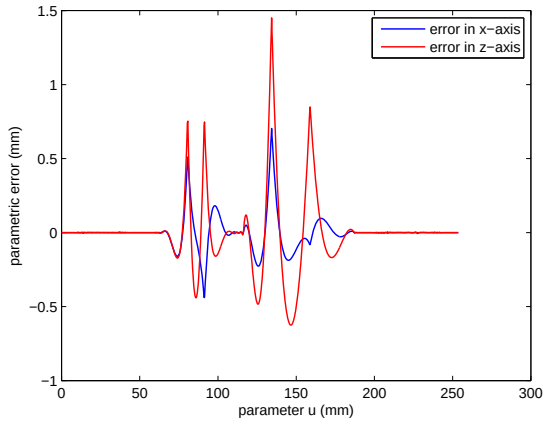
distribution of knots (polynomial pieces) in the neighboring paths of the translational surface even with irregularly distributed data points in the part program. The curvature plot of the spline curves in Fig. 5.14 is smooth and contains no distinct curvature extrema which enables high cutting speed of the machine tool. The milling result of the compressor output shown in Fig. 5.15 is also of satisfactory quality without visible artifacts or roughness.



5-fold knots at edges



4-fold knots at edges



simple knots at edges

Figure 5.4: Error plot with different knot's multiplicity at edges

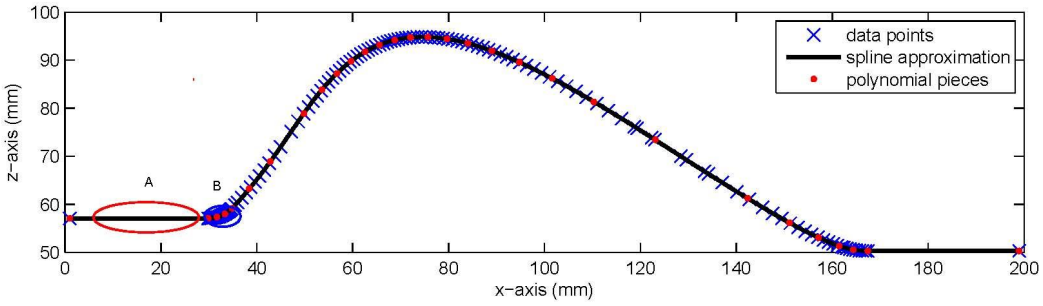


Figure 5.5: One test path from the workpiece 'daimler'

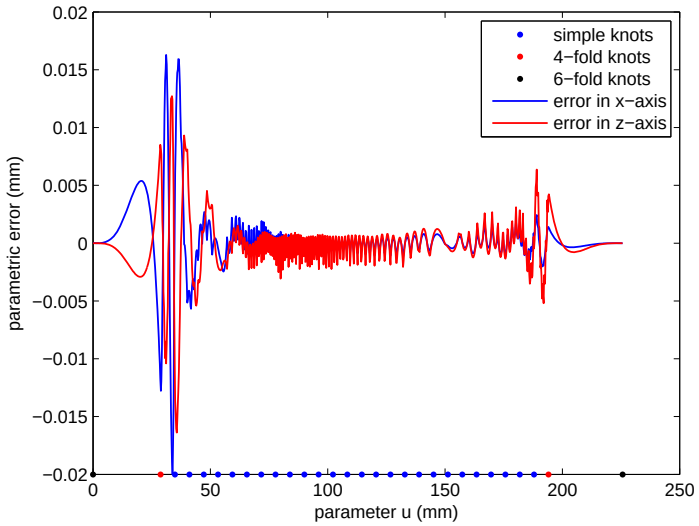


Figure 5.6: The approximation error with uniformly-distributed simple knots

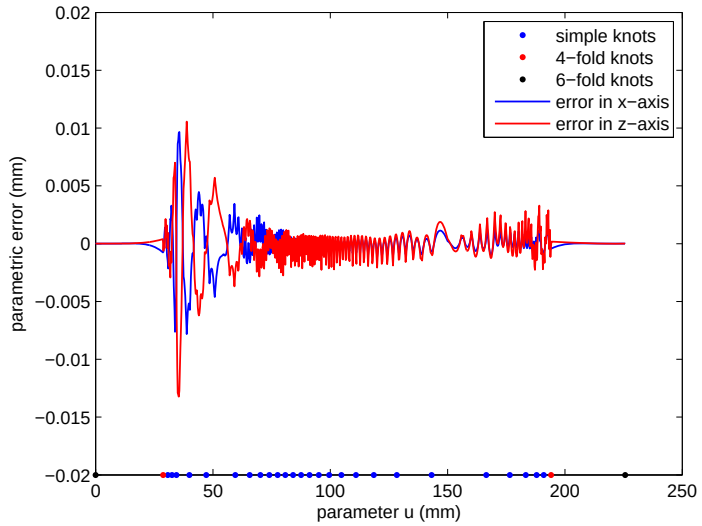


Figure 5.7: The approximation error with knots based on curvature

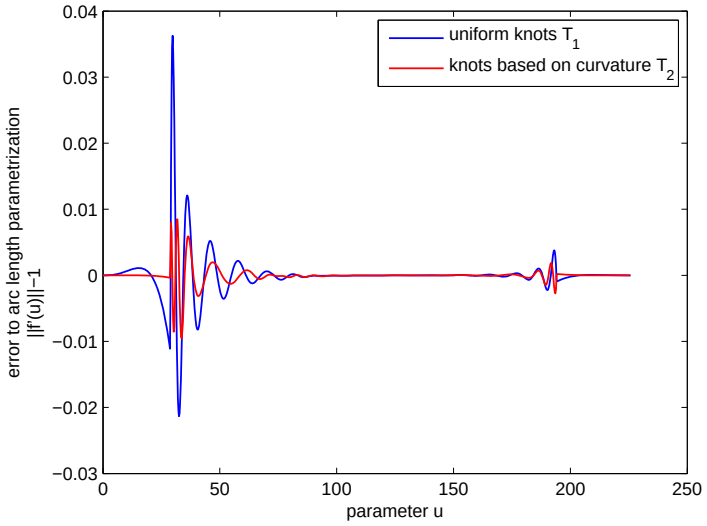


Figure 5.8: The error to arc length parametrization

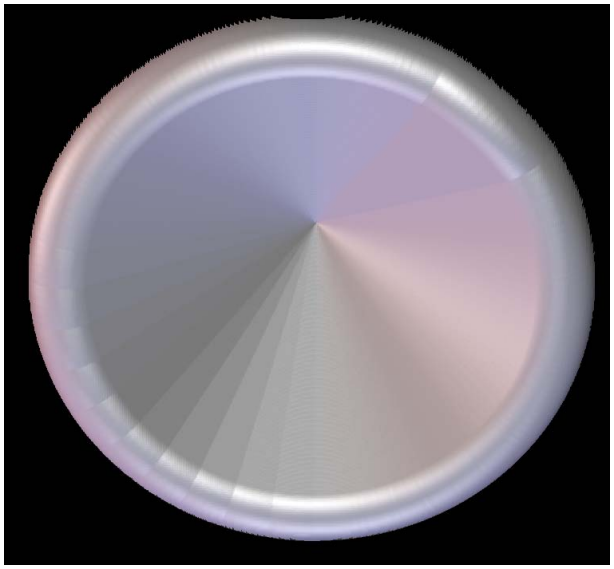


Figure 5.9: The milling result of the workpiece 'turm'

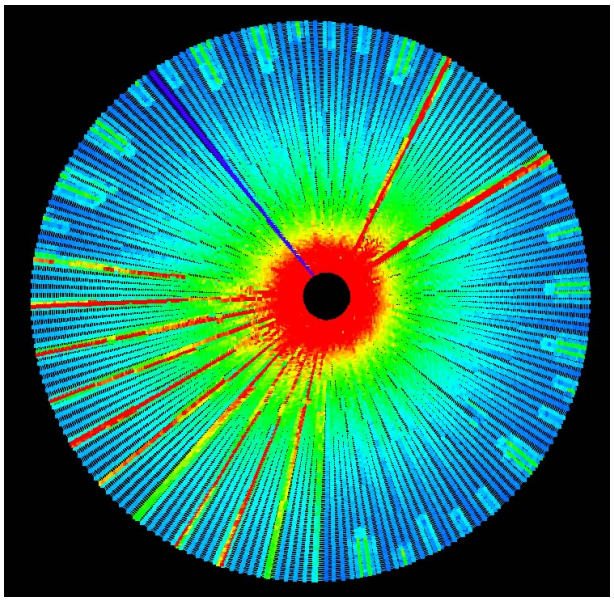
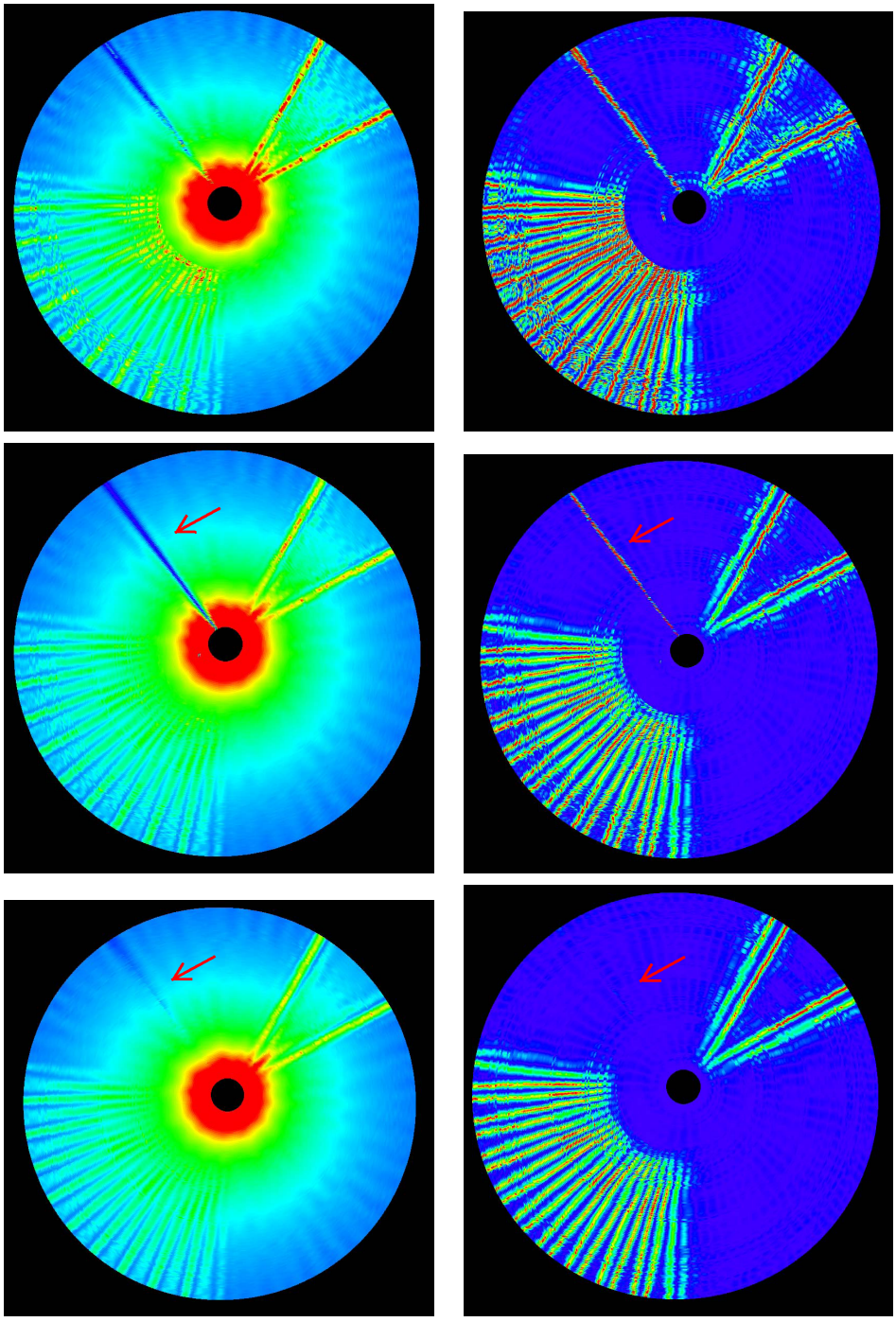


Figure 5.10: Plot of discrete curvature



curvature plot 'torsion' plot

Figure 5.11: Case 1: no smoothing (top); Case 2: $\min \|\ddot{\mathbf{f}}(s)\|^2$ (middle); Case 3: $\min \|\dddot{\mathbf{f}}(s)\|^2$ (down)

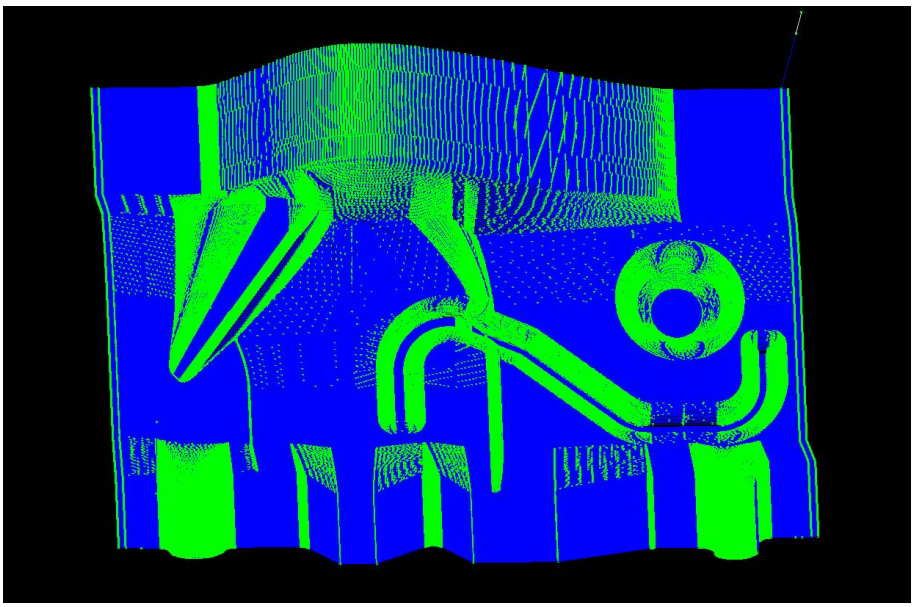


Figure 5.12: The part program of the workpiece 'daimler'

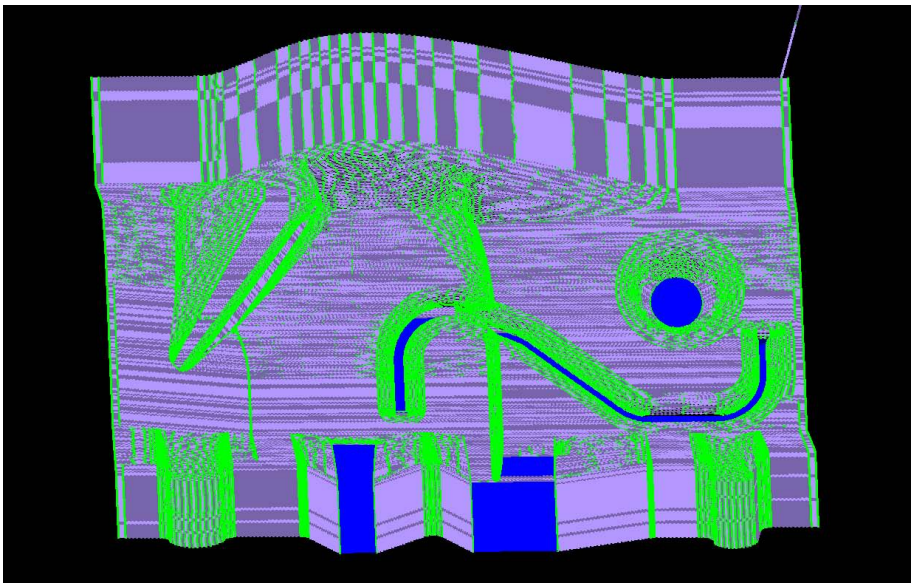


Figure 5.13: The spline approximation of the workpiece 'daimler'

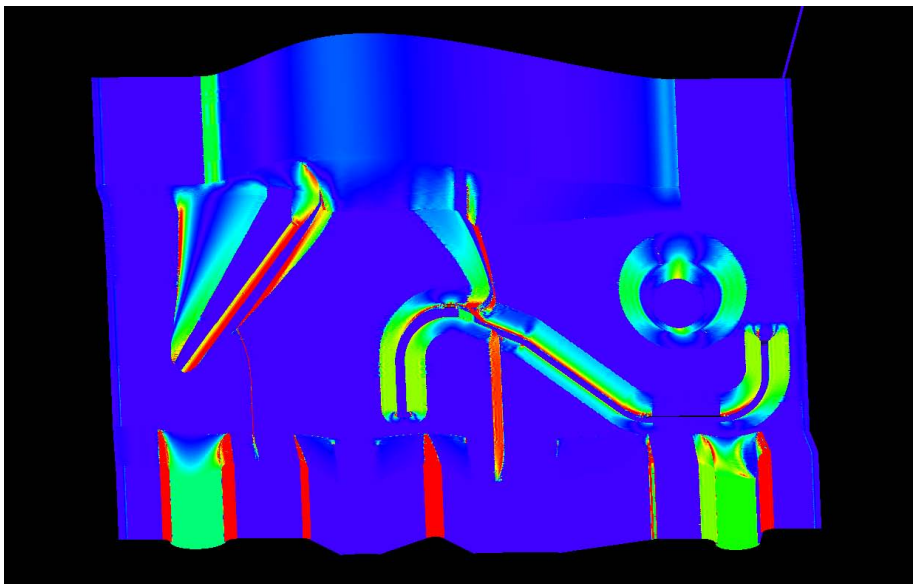


Figure 5.14: The curvature visualization of the workpiece 'daimler'

Figure 5.15: The milling result of the workpiece 'daimler'

6

Conclusions and remarks

The thesis is concerning a key component in the NC kernel called compressor, the essential task of which is to approximate the short linear blocks in the CNC part program using smooth spline curves within a specified tolerance band. The approximating spline curves should consist of minimal number of polynomials and achieve maximal smoothness while satisfying the tolerance condition. Two fundamental issues are the placement of the knots and the choice of appropriate smoothness measures.

From the study, we found the distribution of the knots plays a key role in spline approximation. In the thesis, a knot placement strategy based on curvature characteristics is considered and implemented. With this knot placement strategy, better approximation accuracy can be achieved with fewer number of polynomials. Another advantage is that we can achieve very consistent distribution of the polynomial pieces for the neighboring paths in the translational surface. In addition, a local knot placement scheme is implemented to modify the knot sequence locally only at the regions where the tolerance is violated. To obtain accurate and robust estimation of the curvature, various discrete curvature estimation schemes are also investigated and compared.

Concerning the smoothness of the curve, we have discussed the fairness measures based on minimizing the variation of curvature that can produce spline curves of superior quality and hence lead to reduced acceleration and jerk of the machine tool. In order to dramatically reduce the computational complexity, the minimization of the highly non-linear functional is simplified to a quadratic optimization problem, based on the reasonable assumption that the spline curve is approximately arc length parametrized.

6. CONCLUSIONS AND REMARKS

Together with some extended strategies to deal with some critical problems in the part program, we can achieve satisfying compressor output concerning the compression rate, curvature variation of the spline curve and the surface quality.

List of symbols

Parametric curves

s	arc length
κ	curvture
$\dot{\mathbf{f}}(s)$	tangent vector
$\ddot{\mathbf{f}}(s)$	curvature vector
C^n	n-th order parametric continuity
G^n	n-th order geometric continuity

Spline curves

m	the degree of a spline curve
$\mathbf{d}_i$	the i-th control point of a spline curve
$T_{m,n}$	a knot sequence of a spline curve of degree m with n control points
B_i^n	the Bézier basis function of degree n
N_i^m	the spline basis function of degree m
$S_m\mathbf{d}$	a spline curve of degree m with control points $\mathbf{d}$
$\mathbf{A}^m(T_1, T_2)$	knot insertion matrix from knot sequence T_1 to T_2 of spline curves of degree m
$\mathbf{E}_k$	degree elevation matrix from degree k to $k+1$

Spline approximation and optimization

τ	the specified tolerance for spline approximation
Λ	Lagrangian function
μ	Lagrange multiplier
$\mathbf{J}$	Jacobian matrix
$\mathbf{H}$	Hessian matrix
λ	the smoothing factor

Bibliography

[1] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM (JACM)*, 17(4):589 – 602, 1970. 38

[2] Wolfgang Boehm. Inserting new knots into B-spline curves. *Computer Aided Design*, 12:199–201, 1980. 19

[3] David Coeurjolly, Serge Miguet, and Laure Tougne. Discrete curvature based on osculating circle estimation. In *IWVF-4: Proceedings of the 4th International Workshop on Visual Form*, pages 303–312, London, UK, 2001. Springer-Verlag. 38

[4] Elaine Cohen, Tom Lyche, and Larry L. Schumaker. Algorithms for degree-raising of splines. *ACM Trans. Graph.*, 4(3):171–181, 1985. 22

[5] Carl de Boor. *A Practical Guide to Splines.* Springer-Verlag, 1978. 13, 28, 37, 38

[6] Paul Dierckx. *Curve and Surface Fitting with Splines*, page 9. Clarendon Press, Oxford, 1995. 17

[7] Gerald Farin. *Curves and Surfaces for CAGD.* Morgan Kaufmann, 2001. 25

[8] Rida T. Farouki and Takis Sakkalis. Real rational curves are not "unit speed". *Comput. Aided Geom. Des.*, 8(2):151–157, 1991. 26

[9] Ron Goldman and Tom Lyche. *Knot Insertion and Deletion Algorithms for B-spline Curves and Surfaces.* Philadelphia: Society for Industrial and Applied Mathematics, 1993. 20

[10] Eugene Isaacson and Herbert Bishop Keller. *Analysis of Numerical Methods.* Dover Publications, 1994. 31

[11] B. Kerautret and J. Lachaud. Curvature estimation along noisy digital contours by approximate global optimization. *Pattern Recognition*, 42(10):2265–2278, 2009. 38

[12] B. Kerautret, J. Lachaud, and B. Naegel. Comparison of discrete curvature estimators and application to corner detection. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, pages 710 – 719, Las Vegas, NV, 2008. Springer-Verlag, Berlin, Heidelberg. 38

[13] Hyoungsoek Kim and Jarek Rossignac. Parabola-based discrete curvature estimation. technical report, 2005. 38

[14] E. T. Y. Lee. Choosing nodes in parametric curve interpolation. *Comput. Aided Des.*, 21(6):363–370, 1989. 26

[15] Wayne Liu. A simple, efficient degree raising algorithm for B-spline curves. *Comput. Aided Geom. Des.*, 14(7):693–698, 1997. 22

[16] Tom Lyche and Knut Mørken. Knot removal for parametric B-spline curves and surfaces. *CAGD*, 4:217–230, 1987. 21

[17] Tom Lyche and Knut Mørken. A data-reduction strategy for splines with applications to the approximation of functions and data. *Journal of Numerical Analysis*, 8:185–208, 1988. 21

[18] Tom Lyche and Knut Mørken. *Spline Methods*, chapter 2. 2008. 14, 16, 20

[19] S. Manay, A. J. Yezzi, B. W. Hong, and S. Soatto. Integral invariant signatures. In *Proc. of the Eur. Conf. on Comp. Vision*, 2004. 70

[20] Henry P. Moreton. Functional optimization for fair surface design. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, volume 26, pages 167–176, 1992. 50

[21] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 1999. 30

[22] L. Piegl and W. Tiller. *The NURBS Book(2. Edition)*. Springer-Verlag, 1997. 22

[23] Helmut Pottmann, Johannes Wallner, Qixing Huang, and Yong-Liang Yang. Integral invariants for robust geometry processing. *Comput. Aided Geom. Design*, 26:37–60, 2009. 70

[24] Hartmut Prautzsch. Degree elevation of B-spline curves. *Computer Aided Geometric Design*, 1(2):193–198, 1984. 22

[25] Hartmut Prautzsch and Bruce Piper. A fast algorithm to raise the degree of spline curves. *Comput. Aided Geom. Des.*, 8(4):253–265, 1991. 22

[26] Nickolas S. Sapidis. *Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design*. ASIM, 1987. 50

[27] Tomas Sauer. Splinekurven und -flaechen in CAGD. Vorlesung, 2003. 17

[28] I. J. Schoenberg. On spline functions. In O. Shisha, editor, *"Inequalities" Symposium at Wright-Patterson Air Force Base*, pages 255–291, 1967. 13

[29] Larry L. Schumaker and Sonya Stanley. Shape-preserving knot removal. *Comput. Aided Geom. Des.*, 13(9):851–872, 1996. 21

Declaration

I hereby declare that I have written this paper without the prohibited assistance of third parties and without making use of assistance other than those specified. This paper has not been presented previously in identical or similar form to any other German or foreign examination boards.

Erlangen, Oct. 2010