Founded in 1905

FIVE-AXIS TOOL PATH GENERATION USING PIECEWISE RATIONAL BEZIER MOTIONS OF A FLAT-END CUTTER

BY

ZHANG Wei

(B.Eng., M.Eng.)

DEPARTMENT OF MECHANICAL ENGINEERING

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

NATIONAL UNIVERSITY OF SINGPAORE

2003

ACKNOWLEDGEMENT

The author would like to express her sincere appreciation to her supervisor, A/Prof. Zhang Yunfeng, from the Department of Mechanical Engineering at the National University of Singapore, together with Dr. Q. Jeffrey Ge, Associate Professor from the State University of New York at Stony Brook, USA, for their invaluable guidance, advice and discussion in the entire duration of the project. It has been a rewarding research experience under their supervision.

She would also like to acknowledge the financial support, the research scholarship from the National University of Singapore.

Special thanks are given to A/Prof. Fuh Ying Hsi, for his kind assistance. The author also wishes to thank her fellow graduate students Mr. Wu Yifeng, Mr. Fan Liqing, Mr. Wang Zhigang, Ms. Li Lingling and Ms. Wang Binfang, for their encouragement and support.

Finally, the author thanks her family for their kindness and love. Without their deep love and constant support, she cannot smoothly complete the project.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This thesis studies the automatic tool path generation for 5-axis machining of sculptured surfaces. An efficient approach that uses piecewise rational Bézier motion to generate 5-axis tool path for sculptured surface machining (finish cut) with a flat-end cutter is presented.

A method is proposed in which dual quaternion is used to represent spatial displacements of an object. The representation of kinematic motions for the cutter bottom circle of the flat-end cutter is then formulated. Based on that, a new approach for tool path generation using piecewise rational Bézier cutter motions is described, in which key issues such as gouging and collision avoidance and surface accuracy requirement are addressed. First, a set of cutter contact points on an iso-parametric curve of the designed surface are obtained based on a given fitting tolerance. The associated cutter locations (CLs) are then obtained by finding the suitable cutter orientations that avoid any interference. Based on these CLs, the rational Bézier dual quaternion curve for cutter motion is generated. The entire tool path is therefore established based on the cutter undergoing the rational Bézier motion. Second, the whole tool path is checked to find (1) if there is any interference between the cutter bottom and the designed surface, and (2) whether the deviation between the surface generated by the cutter motion and the designed surface is larger than the given surface error tolerance. The problematic CLs, which cause gouging, collision or accuracy problem, are then modified and the tool path is updated accordingly. The process of *tool path checking → CLs modification → tool path regeneration* continues until the whole tool path is gouging-free and collision-free and meets the accuracy requirement.

After that, the effective cutting shape is represented accurately by intersecting the swept surface generated by the cutter undergoing the rational Bézier motion and the cutting plane. With this representation of the effective cutting shape, an iterative process to generate the adjacent tool path has been conducted. The candidate next tool path is generated with an estimated step size, and the scallop height between the current and this candidate next tool path is consequently calculated. If the scallop height is out of tolerance, the candidate next tool path is modified and the scallop height is recalculated. This process continues until we find the suitable scallop height between the current and candidate next tool path.

Finally, computer implementation and illustrative example are presented to demonstrate the efficacy of the approach.

CHAPTER 1

INTRODUCTION

1.1 Sculptured Surface

With the development of modern technology, the demand for complicated components such as dies, moulds, rotor and impellers has risen rapidly in recent years. The original design concepts of these products are often embodied in physical models, perhaps sculptured from the clay by skilled artisans or from which measurement data is scanned. After that, sculptured surfaces are fitted to the scanned data, and mathematically precise descriptions are then available for subsequent steps in the product-design process. A sculptured surface, also called a free form surface, is generally defined as a surface with variable curvature. Its representation consists of the mathematics and computational aspects of geometry. Currently, the sculptured surface models are one of the main fields in computer-aided geometric design and manufacturing. Many systems have been developed for designing sculptured surface, and most of them are based on various mathematical expressions such as Coons, Bézier, B-spline, or recently NURBS (Faux and Pratt 1981, Piegl and Tiller 1995). Among these expressions, NURBS is the most powerful description for sculptured surface. In this expression, sculptured parts are represented by free-form surface patches, and each of these surface patches is made by a number of free-form curves. Each curve is controlled by a number of control points. Nowadays, sculptured surfaces begin to be used in a wide variety of applications in the automotive, aerospace and ship building industries.

Sculptured Surface Machining (SSM) plays a vital role in the process of bringing new products to the market place. A great variety of products, from automotive body-panels to mobile phones, rely on this technology for the machining of their dies and moulds. In general, to machine a finished die surface starting from a raw stock, the following sequences of metal removal operations are usually required:

(1) Rough cutting, to remove most material of the initial cavity on a sequence of cutting planes.

(2) Semi-roughing, to remove the shoulders left on the part surface after roughing.

(3) Finishing, to finish the sculptured part surface

(4) Scraping, polishing or grinding, to smooth the surface.

However, since sculptured surfaces usually have free-formed geometry of complex shapes and irregular curvature distributions, machining sculptured surface is a challenging issue. With growing industrial demand for design and manufacturing of free-form surface cavities, the more complex, able and accurate metal-cutting technology for sculptured surfaces is in great need. Traditionally, 3-axis Numerical Control (NC) machine tool with ball end mill is used to machine sculptured surfaces. Ball end mills are easy to position relative to the surface and generate simple machining programs. Also, the NC programmer has a relatively easy time to select a ball end mill for a particular surface. However, the whole ball end mill machining process is inefficient and the finish surface quality is inaccurate. To overcome these difficulties, 5-axis Computer Numerical Control machine tool with flat end mill is applied in the SSM.

1.2 Five-Axis Machining

Followings are a number of important criteria for ideal NC machining (Li and Jerard, 1994):

(1) Accuracy: the shape errors introduced by NC machining must be bounded, and machined surfaces must be interference-free.

(2) Efficiency: there are three important measures of efficiency: (a) increased programmer productivity with a resultant speedup in the product development process. (b) Algorithm efficiency in terms of both CPU time and memory space. (c) The machining time required producing the finished part.

(3) Robustness: a robust system is able to cope with the multiple surfaces, concavities and topological inconsistencies caused by gaps, overlapping surfaces and fillets.

In 3-axis machining, a tool is positioned with three degrees of freedom, i.e., a 3-axis NC machine tool can move a ball end tool with a fixed orientation to any point in its workspace. While in 5-axis machining, the tool axis can be arbitrarily oriented, and it is often oriented close to the surface normal. A flat end mill can be tipped at an angle so that the machined surface conforms closely to the designed surface. The effect of a ball end cutter with an increased effective cutter radius in 3-axis machining can be realized by tilting a flat end cutter in a 5-axis NC machine tool. In theory, the 5-axis machining of sculptured surfaces offers many advantages over 3-axis machining (You and Chu, 1997). First, with two additional degrees, it can be used to handle the complex and overlapped surfaces. Second, machining preparatory work such as set-up changes is reduced. In addition, the step-over between two adjacent tool paths is decreased, since the cutting end of the tool is able to match the shape of the machined surface. Therefore, the total manufacturing time from stock materials to finished part

can be greatly shortened in 5-axis machining. Vickers and Quan (1989) analysed the effective cutting edge of the fixed angle flat end milling and found a twenty-time higher materials removal rate in 5-axis machining than that in 3-axis machining using ball end-mills. As a result, faster material-removal rates, improved surface finish and the elimination of hand finishing in 5-axis machining are achieved. Recently, 5-axis machining has been used in more and more applications of the fields such as automotive, aerospace and tooling industries.

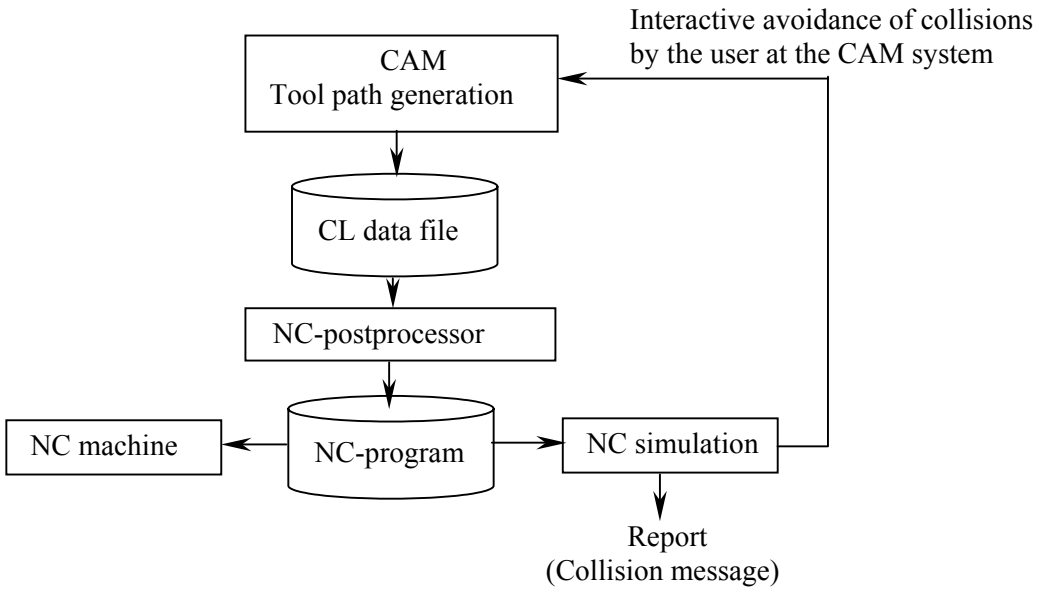


Fig. 1.1 The flowchart of 5-axis NC code generation

As shown in Fig 1.1, the basic procedure for 5-axis NC code generation is as follows (Choi et al., 1993):

(1) Cutter contact (CC) path generation. A point on the part surface at which the cutter is planned to make contact is called CC point, and a series of CC points can form a CC path.

(2) Cutter Location (CL) data generation. The location of a cutter is called CL data, which is completely specified by the cutter centre position and cutter axis vector. The CL data is generated from the CC data.

(3) Tool position correction. This step includes gouging avoidance in concave areas and global collision avoidance.

(4) NC code generation by post-processing the result CL data.

However, despite its advantages, 5-axis machining tool path generation remains a difficult task due to the complicated tool movements and the irregular curvature distributions of sculpture surfaces. In 5-axis machining, while the orientation of the tool is adjusted by the two additional degrees of freedom so as to obtain efficient machining compared to 3-axis machining, it is often computationally expensive when specifying tool orientation for machining. Moreover, global tool interference and local cutter gouging are prone to occur during the machining process. Other problems also exist in 5-axis machining, such as expensive machinery, insufficient support by conventional CAD and CAM systems, highly complex algorithms for gouging avoidance and collision detection between the tool and the non-machined portion of the workpiece. To summarise, 5-axis machining has brought advantages and added flexibility as well as new problems.

1.3 Literature Survey of 5-Axis Machining

Five-axis machining is to machine the workpiece using three translation and two rotation degrees of freedom. In order to improve the efficacy and solve the problems in 5-axis machining, many algorithms for the tool path generation, verification simulation and optimisation have been developed in recent years. Following are some reviews on NC tool path generation: Dragomatz and Mann (1998) provided a classified bibliography of the literature on NC tool path generation including surveys, methods for tool path generation and verification. Choi and Jerard (1998) gave an extensive introduction of 5-axis machining, including the fundamental mathematics, the

machining process, simulation and verification of NC programs. Jensen and Anderson (1996) presented a mathematical review of methods and algorithms used to compute milling cutter placement for multi-axis finished surface milling.

The commonly used tool path generation methods can be classified as follows:

(1) Iso-parameter tool path

This kind of the tool path generation is to use lines of constant parameter. The tool path distribution is determined by calculating, at each path, the smallest tool path interval and using it as a constant offset in the next tool path. You and Chu (1997) presented a method for determination of the tool position and orientation for Iso parameter tool path generation. Elber and Cohen (1994) also developed an adaptive iso-curve extraction method for tool path generation of milling free form surface. Iso-parameter tool paths are computationally simple to generate, however, one serious problem of this method is the inefficient machining due to the non-predictable scallop remaining on the part surface.

(2) Iso-planar tool path

Another approach for tool path generation is to use intersection curves between the parametric surface and series of vertical planes. The path interval or the distance between the vertical planes is also determined based on the scallop height limitation. Rao et al. (1996) planned the tool path using the principal axis method. In his approach, the feed direction at the CC point is consistent to the direction of the principal curvatures of the surface. Huang and Oliver (1994) implemented iso-planar machining on the parametric surface. Iso-planar tool paths are not optimal in general and the choice of a good plane is not at all obvious.

(3) Iso-scallop tool path

In this approach for tool path generation, the scallop height between the two neighboring tool paths is approximately constant. Suresh and Yang (1994) generated a constant scallop height tool path in 3-axis NC machine tool with ball end mill. Lo (1999) proposed an efficient algorithm in searching the iso-scallop cutter paths and extended the algorithm to 5-axis machining with flat end cutter. Sarma and Dutta (1997, 1998) presented the various type of scallop height functions and gave the part programmer direct control over the scallop height of the manufacture surface, and then used a novel technique for grinding tool path generation based on tracking the crest curves of the milled surface so as to maximize material removal and keep the scallop height constant. Pi et al. (1998) generated a grind free tool path that avoids gouging and has scallop height between adjacent tool paths indistinguishable from surface roughness. Lee (1998a) calculated the machining strip widths between the adjacent tool paths according to the scallop height tolerance and generated non-iso-parametric and nearly constant scallop height tool path. Chiou and Lee (2002) furthered Lee's work and implemented global optimisation of tool path distribution.

Most of the work also focuses on finding the gouging and collision free tool path. Gouging, or local tool interference, is one of the most critical problems in 5-axis machining. It results when a high curvature surface is machined using too large of a cutter or by a cutter improperly oriented. The machining of objects, which are composed of multiple surfaces, can also cause gouging. Li and Jerard (1994) observed that tool movement affects only a small portion of the tessellated surface and suggested localized interference checking using a bucketing strategy. Once interference is detected, the tool is tilted away from the interference until it barely

touches the colliding triangle. Pi et al. (1998) and Jensen et al. (2002) proposed a gouging detection method, which uses polynomial resultants to calculate intersection conditions between the bottom of a cutter and the lower profile tolerance surface offset of the part. Cutter interference occurs if there exists intersection. Many other studies are using concepts of differential and analytic geometry such as local curvature properties to detect gouging. Lee (1997) found the admissible gouging free tool orientation by considering both local and global surface shapes. In his method, based on the local surface shape, a feasible tool orientation for gouging avoidance along two orthogonal cutting places is found firstly. Adjacent geometry is then taken into consideration for detecting possible rear gouging. Lee (1998b) presented a method for gouging avoidance by matching the effective cutting curvatures with the curvatures of the part surface at the normal and osculating planes. However, these papers used some rough approximations, such as the 'effective cutting shape' to determine a locally optimal cutter position. Sarma (2000) showed that the exact effective cutting shape, which is the intersection between the cutting plane and the swept surface of the base of the cutter, could be significantly different from the approximated effective cutting shape. This approximation may lead to unwanted collisions and has to be improved for machining high quality surfaces. In order to solve these problems, Rao and Sarma (2000) detected and avoided local gouging by matching the effective cutting curvature of the tool swept surface with the normal curvature of the part surface at the CC points. Yoon et al. (2003) furthered Rao's work, but he did not compute a parameterisation of the swept surface of the moving cutter to derive its second order behaviour at the contact point of the cutter. This can be done in a simpler geometric way using concepts of classical constructive differential geometry. His work overcomes the weakness of

effective cutting shape methods and fully exploits the possibility of finding the locally optimal cutting positions for sculptured surface machining.

Besides gouging, interference between the non-cutting portions of the tool and the surface is usually referred to as a collision or global gouging. The existence of collision problem would lead to not only the bad surface quality but also the damage of cutter and machine tool. Many researchers have studied collision avoidance. Some of them tried to find a collision-free tool path based on a trial and error process, where the provisional determination of tool posture is repeated until collision does not occur. Li and Jerard (1994) presented a method to generate the tool path in Cartesian space by triangulating the surface and finding the collision-free cutter locations by rotating the cutters until the cutter has no intersection with the triangulation of the surface. Lee and Chang (1995) used a two-phase approach for global tool interference avoidance. In his method, the tool position is checked for possible interference with the convex hull of the designed surface. If interference between the tool and the convex hull is detected, further calculation for checking interference between the tool and the designed surface is performed and the tool orientation is corrected if needed. The advantage of finding the collision-free tool path by gradually adjusting tool orientation is the computational efficiency. However, this method cannot achieve the optimal tool orientation and can cause the irregularity of the surface appearance. Recently, some researchers began to use the global automatic strategy to find the collision-free tool orientation. Morishige et al. (1997, 1999) used a C space, adapted from robot motion planning, to represent the tool orientation in an appropriate space in which the obstacles are mapped. Jun et al. (2002) further developed this work and applied the C space method to find the optimal tool orientation by considering the local gouging, rear gouging and global collision in 5-axis machining, and minimizing the scallop height between the adjacent

tool paths. Unfortunately, although intuitively and intellectually appealing, the C space approach has an obvious problem: mapping obstacles to the C space is often a computationally intractable task. Woo (1994) first demonstrated the use of visibility cones, which are an alternative representation of the C space, in collision detection and avoidance. A point on an object is visible from a point at infinity if the straight-line segment connecting these two points does not intersect with the object. Yang and Xiong (1999) developed a method of computing a visibility cone to analyse the machinability of the milling direction. Suh and Kang (1995) proposed an application based on visibility cones to aid the process planning for manufacturing a free form surface. Spitz and Requicha (1990) proposed an interesting algorithm for computing the access cones of a uniform diameter tool at a point in objects by computing the visibility of the point in the object. Balasubramaniam et al. 2003) used a discretized approach to check visibility, and took advantages of the rapid performance of graphics hardware to generate the visibility information. Visibility is a useful precursor for the more expensive accessibility computation. Although visibility approaches provided some simplification, they also tend to be computationally expensive in practice. Some approaches for collision avoidance also focus on possible collision between machine and part, machine and tool or between moving machining components (Lauwers et al. 2002). Liu (1995) described tool interference avoidance using the side mill in 5-axis machining.

Many other efforts focused on obtaining the optimal cutter orientations to improve the efficiency of the tool path generation process. Pure analytical methods described by Kruth and Klewais (1994), Lee (1997, 1998b) and Jensen and Anderson (1993) optimised the tool orientation based on the curvature information on the CC point. These methods do not take the surface anomalies in the neighbourhood for the

CC point into account. This may result in the occurrence of gouging. Other tool optimisation methods described by Redonnet et al. (1998) and Rao and Sarma (2000) fit the tool as close as possible to the part surface. These optimisation techniques use the entire surface definition to avoid the above problems. In contradiction to the pure analytical methods, most of these algorithms determine the optimal tool posture iteratively. Jensen et al. (2002) also used both 5-axis orientation and positioning algorithms in conjunction with tool selection procedures to provide a more efficient and accurate machining solution for complex surfaces. Some researchers developed various methods of predicting the real scallop height to generate optimal CL data in a multi-axis machine tool. Kim and Chu (1994) provided the effect cutter marks on the surface roughness and examined the scallop height in the milling process. Lee (1996b) presented an error analysis method for 5-axis machining which applied differential geometry technique to evaluate the scallop height between adjacent cutter locations. Choi et al. (1993) presented a method of generating optimal CL data for 5-axis NC contour milling by finding minimal scallop height distance given a fixed path interval. Other works concentrate on finding the relationship between the part surface geometry and the tool path machining efficiency. Wang and Tang (1999) suggested that the optimal tool paths are normally parallel to the longest boundary. Marciniak (1987, 1991) and Kruth and Klewais (1994) analysed the cutting direction and the part surface geometry property. They concluded that the optimal cutting direction encompasses the largest cutting width when the tool path matches the smaller principal curvature direction of the part surface.

One of the other challenging tasks for 5-axis machining is the automatic generation of tool path without depending on human interaction to machine the sculpture surface. Lee and Chang (1991) develop a methodology, which automatically

decides the machining procedure, selects the best possible cutters for machining a cavity with islands bounded by sculptured surfaces, and then generates gouging-free cutter paths for roughing and finishing steps. Choi and Jerard (1998) also presented a framework for developing sculptured surface machining software.

In general, most of the reported tool path generation methods are numerical and discrete in nature. They basically follow a two-step approach:

(1) Given a surface description (either in NURBS representation or triangular polyhedral meshes), a set of CC points are generated based on a machining strategy and the given surface error tolerance.

(2) For each CC point, CL is determined that avoids gouging and collision and is within the machine's axis limits.

In order to satisfy the surface error tolerance, the number of CC points is generally very large. At the same time, algorithms that search for a feasible CL from a CC point are iterative in nature, which normally leads to extremely long computation time. A further drawback of this kind of approach is that the complete elimination of gouging or collision between the neighboring CLs is not guaranteed.

Instead of focusing on a particular instant of the tool motion and studying local geometric issues at the instant, tool path can be generated as envelopes of moving cutter. Wang and Joe (1997) presented that surfaces can be generated by sweeping a profile curve along a given spline curve. Juttler and Wagner (1996, 1999) proposed a method to generate rational motion-based surface emphasizing the special cases of a moving cylinder of cone of revolution. Ge an Srinivasan (1998) presented two algorithms for fine-tuning rational B-spline motions suitable for computer-aided design. Xia and Ge (1999,2001a, 2001b) provided the representation of the boundary surfaces of the swept surface undergoing rational Bézier and B-spline motions and

proposed a method for 5-axis tool path generation using rational Bézier and B-spline motions. This method can generate the tool path efficiently and, at the same time, allow an accurate representation of the swept surface generated by the cutter. However, their work has not yet explicitly dealt with the issue related to gouging and collision detection and avoidance. Their approach for generating the entire tool paths for the sculpture surface is also incomplete. Hence, their work needs to be extended.

1.4 Objective of the Project

The objective of this work is to develop a method for tool path generation in 5-axis machining of the sculptured surface. The errors introduced by tool path generation algorithms must be bounded. Specifically, the tool path must be gouging-free and collision-free, and scallop height between two paths must be controlled with allowable tolerance. The algorithm must also be efficient in terms of both CPU time and memory space, and robust capable coping with the multiple surfaces including concave and convex surface with irregular curvatures.

1.5 Organization of the Thesis

This thesis contains six chapters, and the organization of this thesis is as follows:

In chapter 1, the methods for the sculptured surface machining are introduced first. The previous researches on 5-axis tool path generation are then reviewed. After that, the objective and the organization of the thesis are introduced.

In chapter 2, the fundamental mathematics required in developing the thesis is presented. The basic geometric modelling methods in computer aided geometric design are reviewed and the concepts of kinematic driven geometric modelling are introduced.

In chapter 3, an efficient approach to generate a single gouging-free and collision-free tool path for 5-axis sculptured surface machining using rational Bézier motion of the flat-end cutter is presented.

In chapter 4, an iterative method to generate the adjacent tool path so that the scallop height between two neighboring tool paths is within the allowable tolerance is presented.

In chapter 5, the examples to illustrate the efficiency of the developed algorithm for tool path generation using the piecewise rational Bézier cutter motion is presented.

Finally, in chapter 6, the conclusions are drawn and the recommendations for future works are discussed.

CHAPTER 2

MATHEMATIC FUNDAMENTALS

Computer-Aided Geometric Design (CAGD) deals with the problem of representation and manipulation of geometric shapes in a manner suitable for computer processing. Much of the existing work in CAGD for geometric shapes design is based on point geometry. In recent years, geometric shape design techniques in CAGD such as Bézier and B-spline methods have been extended from pure geometric domain to kinematic domain (Ge and Ravani, 1991, 1993, 1994; Srinivasan and Ge, 1996, 1997 and 1998b; Juttler and Wagner, 1996). Kinematics-Driven Geometric Modelling, once developed, would provide a new methodology for designing kinematically generated free-form surfaces. In this chapter, the basic geometric modelling methods in CAGD are reviewed and the concepts of kinematic driven geometric modelling are introduced. Comprehensive study of the subject can be found in CAGD texts such as Farin (1996), Faux (1981), Piegl and Tiller (1995).

2.1 Geometric Modelling Based on Point Geometry

2.1.1 Bézier curve and surface

2.1.1.1 Bézier curve

The Bézier curve representation is one that is utilized most frequently in computer graphics and geometric modelling. The curve is defined geometrically, which indicates that its parameters have geometric meaning.

Given the set of control points, $\{\mathbf{P}_0, \mathbf{P}_1, \ldots \mathbf{P}_n\}$, we can define a Bézier curve of degree n by either of the following two definitions: an analytic definition specifying the blending of the control points, and a geometric definition specifying a recursive generation procedure that calculates successive points on line segments developed from the control point sequence.

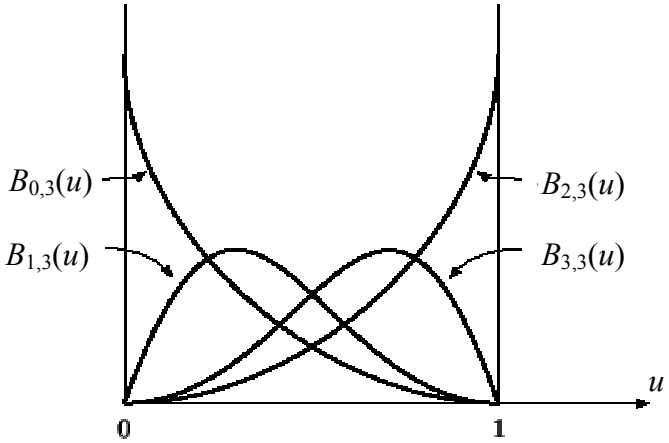


Fig. 2.1 The cubic Bernstein polynomials

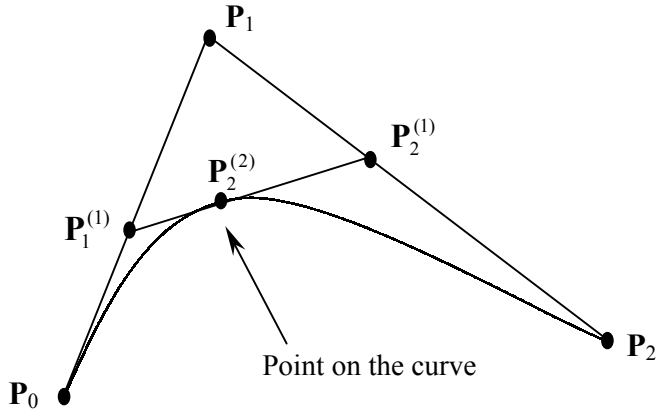


Fig. 2.2 Quadratic Bézier curve generated by de Casteljau method

The Analytic Definition

$$\mathbf{P}(u) = \sum_{i=0}^{n} B_{i,n}(u)\mathbf{P}_i \tag{2.1}$$

where $B_{i,n}(u) = \dfrac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$ are the Bernstein polynomials of degree n, and $0 \le u \le 1$. For example, the Bernstein polynomials of degree 3 are

$$B_{0,3} = (1-u)^3 \;\; B_{1,3} = 3u(1-u)^2 \;\; B_{2,3} = 3u^2(1-u)^2 \;\; B_{3,3} = u^3$$

and can be plotted as in Fig. 2.1.

Geometric Definition

$$\mathbf{P}(u) = \mathbf{P}_n^{(n)}(u) \tag{2.2}$$

where $\mathbf{P}_i^{(j)}(u) = \begin{cases} (1-u)\mathbf{P}_{i-1}^{(j-1)}(u) + u\mathbf{P}_i^{(j-1)}(u) & \text{if } j > 0 \\ \mathbf{P}_i & \textit{otherwise} \end{cases}$ and u ranges between zero and one, i.e., $0 \le u \le 1$. The algorithm of the generation of Bézier curves based on repeated linear interpolation as in Eq. (2.2) is called the de Casteljau algorithm. Fig. 2.2 shows the quadratic Bézier curves constructed based on de Casteljau method.

From Eq. (2.1) and Fig. 2.2, we can know that the tangent vector to the curve at the point $\mathbf{P}_0$ is the line $\overline{\mathbf{P}_0\mathbf{P}_1}$ and $\dot{\mathbf{P}}(0)=3(\mathbf{P}_1-\mathbf{P}_0)$. The tangent to the curve at the point $\mathbf{P}_n$ is the line $\overline{\mathbf{P}_{n-1}\mathbf{P}_n}$ and $\dot{\mathbf{P}}(1)=3(\mathbf{P}_n-\mathbf{P}_{n-1})$.

Although Bézier curve offers many advantages, there exist a number of important curves such as circles, ellipses, etc, that cannot be represented precisely using Bézier curve. In order to solve this problem, rational Bézier curve is developed. The basic idea of rational Bézier curve is to define a curve in one higher dimension space and project it down on the homogenizing variable. For implementation, rational curve design assigns every control point of Bézier curve a weight to provide additional control over the curve shape. An *n*th-degree rational Bézier curve is given by:

$$\mathbf{P}(t) = \sum_{i=0}^{n} R_{i,n}(u)\mathbf{P}_i \tag{2.3}$$

Where $R_{i,n}(u) = \dfrac{B_{i,n}(u)w_i}{\sum_{j=0}^{n} B_{j,n}(u)w_j}$, $B_{i,n}(u)$ are the Bernstein polynomials; $\mathbf{P}_i$ are the control points of the rational Bézier curve; the w_i are scalars, called the *weights*.

The weights are typically used as shape parameters. If we increase w_i, the curve is pulled toward the corresponding $\mathbf{P}_i$. Fig. 2.3 shows that the rational cubic Bézier curve is pulled toward $\mathbf{P}_1$ when w_1 is increased.

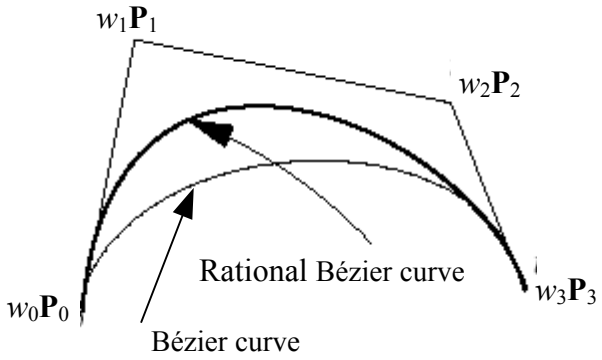


Fig. 2.3 Rational cubic Bézier curve

2.1.1.2 C^1 and C^2 continuity between two cubic Bézier curves

In this section, we summarize the relationships between the control points of the two cubic Bézier curves in order to get $C^1$and C^2 continuity at the junction of these two curves (Kang, 1997).

Given two cubic Bézier curves $\mathbf{s}_0$ and $\mathbf{s}_1$ with control points [$\mathbf{P}_{-3}$, $\mathbf{P}_{-2}$, $\mathbf{P}_{-1}$, $\mathbf{P}_0$] and [$\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$], we can combine these two curves into one composite curve, defined as the map of the interval $[u_0, u_2]$ into E^3. The left segment $\mathbf{s}_0$ is defined over an interval $[u_0, u_1]$, while the right segment $\mathbf{s}_1$ is defined over $[u_1, u_2]$.

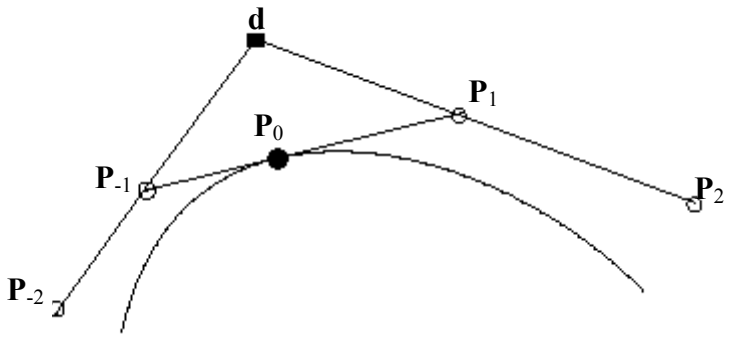


Fig. 2.4 C^2 continuity of two Bézier curve segments

The two Bézier curves are C^1 continuous at $u = u_1$ if

$$\frac{d\mathbf{s}_0}{du}|_{u=u_1} = \frac{d\mathbf{s}_1}{du}|_{u=u_1} \tag{2.4}$$

Set $\Delta_0 = u_1 - u_0$ and $\Delta_1 = u_2 - u_1$, and according to the properties of the Bézier curves, we have the simpler formula of Eq. (2.4) as:

$$\frac{1}{\Delta_0}(\mathbf{P}_0 - \mathbf{P}_{-1}) = \frac{1}{\Delta_1}(\mathbf{P}_1 - \mathbf{P}_0) \tag{2.5}$$

This means that the three points $\mathbf{P}_{-1}$, $\mathbf{P}_0$, $\mathbf{P}_1$ must be collinear and also be in the ratio $(u_1 - u_0) : (u_2 - u_1) = \Delta_0 : \Delta_1$ so that the composite curve is C^1 continuous at the junction point. The two Bézier curves are C^2 continuous at $u = u_1$ if in addition

$$\frac{d^2\mathbf{s}_0}{du^2}|_{u=u_1} = \frac{d^2\mathbf{s}_1}{du^2}|_{u=u_1} \tag{2.6}$$

After the substitution of the second order derivatives, we obtain:

$$\mathbf{P}_{-1} + \frac{\Delta_1}{\Delta_0}(\mathbf{P}_{-1} - \mathbf{P}_{-2}) = \mathbf{P}_1 + \frac{\Delta_0}{\Delta_1}(\mathbf{P}_1 - \mathbf{P}_2) = \mathbf{d} \tag{2.7}$$

Eq. (2.7) indicates that line $\overline{\mathbf{P}_{-2}\mathbf{P}_{-1}}$ and $\overline{\mathbf{P}_2\mathbf{P}_1}$ must intersect at a point **d**. Rearranging Eq. (2.7), we obtain

$$\mathbf{P}_{-1} = (1 - t_1)\mathbf{P}_{-2} + t_1\mathbf{d} \quad \mathbf{P}_1 = (1 - t_1)\mathbf{d} + t_1\mathbf{P}_2 \tag{2.8}$$

where $t_1 = \Delta_0/(\Delta_0 + \Delta_1)$ and **d** is called the *deBoor control point* (Fig. 2.4).

2.1.1.3 Tensor product Bézier surface

Methods for generating Bézier curves can be extended to two dimensions to obtain tensor product Bézier surfaces. The tensor product method that uses basis functions and geometric coefficients is basically a bi-directional curve scheme. The basis functions are bivariate functions of u and v, which are constructed as products of univariate basis functions. The geometric coefficients are arranged in a bi-directional,

$n \times m$ net. For tensor product Bézier surfaces, the univariate basis functions are Bernstein polynomials. Thus, a tensor product Bézier surface has the form:

$$\mathbf{S}(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\mathbf{P}_{i,j} \quad \text{where } 0 \le u,\ v \le 1 \tag{2.9}$$

Where the net of the $\mathbf{P}_{i,j}$ is called the *Bézier net* or *control net* of the Bézier surface. The $\mathbf{P}_{i,j}$ are called *control points* or *Bézier points*. The surface can also be treated as the locus of a Bézier curve $\mathbf{S}_i(v)= \sum_{j=0}^{m} B_{j,m}(v)\mathbf{P}_{i,j}$ moving along u-direction and thereby changing its shape on its way.

Tensor product Bézier surfaces can also be obtained by repeated application of bilinear interpolation according to the deCasteljau algorithm. Given a control net $\{\mathbf{P}_{i,j}\}$ with $0 \le i \le n$ and $0 \le j \le m$ and parameter u and v, the following algorithm generates a point on a surface according to deCasteljau algorithm:

$$\mathbf{S}_{i,j}^{r,s}(u,v) = \begin{cases} \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} \mathbf{S}_{i,j}^{r-1,s-1} & \mathbf{S}_{i,j+1}^{r-1,s-1} \\ \mathbf{S}_{i+1,j}^{r-1,s-1} & \mathbf{S}_{i+1,j+1}^{r-1,s-1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} & \text{if } r,s > 0 \\ \mathbf{P}_{i,j} & \text{if } r,s = 0 \end{cases} \tag{2.10}$$

where $0 \le r \le n$; $0 \le s \le m$; $0 \le i \le n\text{-}r$; $0 \le j \le m\text{-}s$.

The rational Bézier surface is defined to be perspective projection of a four-dimensional polynomial Bézier surface:

$$\mathbf{S}^{w}(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\mathbf{P}_{i,j}^{w} \tag{2.11}$$

Where $\mathbf{P}_{i,j}^{w} = \{\mathbf{P}_{i,j}, w_{i,j}\}$ and $w_{i,j}$ are the *weights* of control point $\mathbf{P}_{i,j}$. The corresponding three-dimensional rational Bézier surface is:

$$\mathbf{S}(u,v)= \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)w_{i,j}} \tag{2.12}$$

$\mathbf{S}(u,v)$is not a tensor product surface, but $\mathbf{S}^w(u,v)$ is.

2.1.2 B-spline curve and surface

The main advantage of B-spline curve is its performance in interactive shape design. Using B-spline curves, we can utilize both control point movement and weight modification to attain local shape control.

A pth-degree B-spline curve is defined by:

$$\mathbf{P}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \qquad 0 \le u \le 1 \tag{2.13}$$

where the $\{\mathbf{P}_i\}$are the control points, and the $\{N_{i,p}(u)\}$ are the pth degree B-spline basis functions defined on the non-uniform knot vector

$$U=\{u_0, \ldots, u_m\}=\{\underbrace{0,\cdots 0}_{p+1}, u_{p+1}, \ldots u_{m-p-1}, \underbrace{1,\cdots 1}_{p+1}\}$$

Where $m=n+p+1$. The ith B-spline basis function of p-degree, denoted by $N_{i,p}(u)$ is defined as:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & otherwise \end{cases}$$

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u) \tag{2.14}$$

The B-spline basis function has the following properties:

- $N_{i,p}(u)$ is a step function, equal to zero everywhere except on the half-open interval $u \in [u_i, u_{i+1})$
- For $p>0$, $N_{i,p}(u)$ is a linear combination of two $(p-1)$th degree basis functions.
- The $N_{i,p}(u)$ are piecewise polynomials, defined on the entire real line, generally only the interval $[u_0, u_m]$ is of interest.

- The computation of the *p*th-degree functions generates a truncated triangular table:

$$\begin{array}{ccccc} N_{0,0} & & & & \\ & N_{0,1} & & & \\ N_{1,0} & & N_{0,2} & & \\ & N_{1,1} & & N_{0,3} & \\ N_{2,0} & & N_{1,2} & \vdots & \\ & N_{1,2} & \vdots & & \\ N_{3,0} & \vdots & & & \\ \vdots & & & & \end{array}$$

The Non-Uniform Rational B-Spline (NURBS) curve can be represented as the perspective projection of a four-dimensional polynomial B-spline curve as:

$$\mathbf{P}^{\mathbf{w}}(u)=\sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i^{w} \tag{2.15}$$

Where $\mathbf{P}_i^{w}=\{\mathbf{P}_i, w_i\}$.

A B-spline surface of degree p in the u direction and degree q in the v direction is a bivariate vector-valued piecewise rational function of the form:

$$\mathbf{S}(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j} \tag{2.16}$$

where the $\{\mathbf{P}_{i,j}\}$are the control points, the $\{w_{i,j}\}$are the weights, and the $\{N_{i,p}(u)\}$ are the *p*th degree B-spline basis functions defined on the non-uniform knot vector

$$U=\{u_0, \ldots, u_s\}=\{\underbrace{0,\cdots 0}_{p+1}, u_{p+1}, \ldots u_{r-p-1}, \underbrace{1,\cdots 1}_{p+1}\}$$

Where r=*n*+*p*+1; The $\{N_{j,q}(v)\}$ are the *q*th degree B-spline basis functions defined on the non-uniform knot vector

$$V=\{v_0, \ldots, v_s\}=\{\underbrace{0,\cdots 0}_{q+1}, v_{q+1}, \ldots v_{s-q-1}, \underbrace{1,\cdots 1}_{q+1}\}$$

Where $s=m+q+1$. The Non-Uniform Rational B-Spline (NURBS) surface can be represented as the perspective projection of a four-dimensional polynomial B-spline surface as:

$$\mathbf{S^w}(u,v)= \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}^{w} \tag{2.17}$$

Where $\mathbf{P}_{i,j}^{w} = \{\mathbf{P}_{i,j}, w_{i,j}\}$.

2.1.3 B-spline curve fitting

Given a set of point $\{\mathbf{Q}_k\}$, $k=0, \ldots n$, we can interpolate these points with a pth degree non-rational B-spline curve. If we assign a parameter value, $\tilde{u}_k$, to each $\mathbf{Q}_k$, and select an appropriate knot vector $U=\{u_0, \ldots u_m\}$, we can set up the $(n+1)\times(n+1)$ system of linear equations:

$$\mathbf{Q}_k=\mathbf{P}(\tilde{u}_k)= \sum_{i=0}^{n} N_{i,p}(\tilde{u}_k)\mathbf{P}_i \tag{2.18}$$

The control points, $\mathbf{P}_i$, are the $n+1$ unknowns. Note that the equation is independent of the number of the coordinates in $\mathbf{Q}_k$. With $n+1$ equations, $\mathbf{P}_i$ can be solved out.

The problems of choosing the $\tilde{u}_k$ and U remain, and their choice affects the shape and parameterisation of the curve. Basically, there are three methods of choosing the $\tilde{u}_k$: equally spaced, chord length and centripetal method. In these methods, the chord length is most widely used and it is generally adequate. In this application, we adopt the chord length to calculate the parameter $\tilde{u}_k$. Assume that the parameter lies in the range $u \in [0,1]$ and let d be the total chord length

$$d=\sum_{k=1}^{n}\left|\mathbf{Q}_k - \mathbf{Q}_{k-1}\right| \tag{2.18}$$

then

$$\tilde{u}_0=0$$

$$\tilde{u}_n=1$$

$$\tilde{u}_k=\tilde{u}_k+\frac{\sqrt{|\mathbf{Q}_k-\mathbf{Q}_{k-1}|}}{d} \quad k=1,\ldots n\text{-}1 \tag{2.19}$$

The following technique of finding knots is recommended:

$$u_0=\ldots=u_p=0$$

$$u_{m-p}=\ldots=u_m=0$$

$$u_{j+p}=\frac{1}{p}\sum_{i=j}^{j+p-1}\tilde{u}_i \text{ and } j=1,\ldots n\text{-}p \tag{2.20}$$

2.1.4 Changing from cubic B-spline curve to piecewise Bézier curve

Let us consider a C^2 cubic B-spline curve $\mathbf{s}(u)$ defined over L intervals $u_0<\ldots<u_L$ with deBoor control points $\mathbf{d}_{-1},\mathbf{d}_0,\ldots,\mathbf{d}_{L+1}$. We will compute the inner Bézier points $\mathbf{b}_{3i+1}$, $\mathbf{b}_{3i+2}$ according to the C^2 condition and the junction points $\mathbf{b}_{3i}$ with the C^1 condition. The $\mathbf{d}_{-1},\mathbf{d}_0,\ldots,\mathbf{d}_{L+1}$ is called the B-spline polygons. The C^1 and C^2 continuity of Bézier curve have been introduced in above section.

Denoting the steps of knot sequence $\Delta_i=u_{i+1}-u_i$, we can compute the inner Bézier and junction points as follows:

(1) At the start segment, we set

$$\mathbf{b}_0=\mathbf{d}_{-1},\ \mathbf{b}_1=\mathbf{d}_0,\ \ \mathbf{b}_2=\frac{\Delta_1}{\Delta_0+\Delta_1}\mathbf{d}_0+\frac{\Delta_0}{\Delta_0+\Delta_1}\mathbf{d}_1 \tag{2.21}$$

(2) At the end segment, we set

$$\mathbf{b}_{3L}=\mathbf{d}_{L+1},\ \mathbf{b}_{3L-1}=\mathbf{d}_L,\ \ \mathbf{b}_{3L-2}=\frac{\Delta_{L-1}}{\Delta_{L-2}+\Delta_{L-1}}\mathbf{d}_{L-1}+\frac{\Delta_{L-2}}{\Delta_{L-2}+\Delta_{L-1}}\mathbf{d}_L \tag{2.22}$$

(3) Computing the inner Bézier points on the leg $\overline{\mathbf{d}_{i-1}\mathbf{d}_i}$ according to the C^2 condition

$$\mathbf{b}_{3i-2}=\frac{\Delta_{i-1}+\Delta_{i+1}}{\Delta}\mathbf{d}_{i-1}+\frac{\Delta_{i-2}}{\Delta}\mathbf{d}_i$$

$$\mathbf{b}_{3i-1} = \frac{\Delta_i}{\Delta}\mathbf{d}_{i-1} + \frac{\Delta_{i-2} + \Delta_{i-1}}{\Delta}\mathbf{d}_i \tag{2.23}$$

where i=2,…L-1, and $\Delta=\Delta_{i-2}+\Delta_{i-1}+\Delta_i$.

(4) Computing the junction points $\mathbf{b}_{3i}$ on the leg of $\overline{\mathbf{b}_{3i-1}\mathbf{b}_{3i}}$ according to C^2 condition

$$\mathbf{b}_{3i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i}\mathbf{b}_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i}\mathbf{b}_{3i+1} \tag{2.24}$$

where i=1,…L-1.

In this application, the number of the result composite Bézier curves is L+1.

2.2 Geometric Modelling Based on Kinematics

2.2.1 Dual number and dual vector

According to Bottema and Roth (1873), a dual number is defined as:

$$\hat{a} = a + \varepsilon a^0 \tag{2.25}$$

where a, a^0 are real numbers known as the real part and the dual part respectively. The symbol ε represents the dual unit which has the property ε^2=0.

Let $\hat{a}_1 = a_1 + \varepsilon a_1^0$ and $\hat{a}_2 = a_2 + \varepsilon a_2^0$ be two dual numbers. The dual numbers have the following properties:

- Addition and subtraction: $\hat{a}_1 \pm \hat{a}_2 = (a_1 \pm a_2) + \varepsilon(a_1^0 \pm a_2^0)$
- Multiplication: $\hat{a}_1\hat{a}_2 = (a_1 a_2) + \varepsilon(a_1 a_2^0 + a_2 a_1^0)$
- Division: $\hat{a}_1/\hat{a}_2 = (a_1/a_2) + \varepsilon(a_1 a_2^0 - a_2 a_1^0)/(a_2)^2$

A dual vector $\hat{\mathbf{u}}$ is defined as a vector whose components are dual numbers:

$$\hat{\mathbf{u}} = \mathbf{u} + \varepsilon \mathbf{u}^0 = (\hat{u}_1, \hat{u}_2, \hat{u}_3) \tag{2.26}$$

where $\hat{u}_l = u_l + \varepsilon u_l^0$, l=1,2,3, u_l and u_l^0 are real numbers, and $\mathbf{u}$=(u_1, u_2, u_3), $\mathbf{u}^0$=(u_1^0, u_2^0, u_3^0) are vectors.

Let $\hat{\mathbf{u}} = \mathbf{u} + \varepsilon\mathbf{u}^0 = (\hat{u}_1, \hat{u}_2, \hat{u}_3)$ and $\hat{\mathbf{v}} = \mathbf{v} + \varepsilon\mathbf{v}^0 = (\hat{v}_1, \hat{v}_2, \hat{v}_3)$ be two dual vectors. The dual vectors have the following properties:

- Dot product: $\hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = u \cdot v + \varepsilon(u \cdot v^0 + u^0 \cdot v) = \hat{u}_1\hat{v}_1 + \hat{u}_2\hat{v}_2 + \hat{u}_3\hat{v}_3$
- Cross product:

$$\hat{\mathbf{u}} \times \hat{\mathbf{v}} = u \times v + \varepsilon(u \times v^0 + u^0 \times v) = ((\hat{u}_2\hat{v}_3 - \hat{u}_3\hat{v}_2), (\hat{u}_3\hat{v}_1 - \hat{u}_1\hat{v}_3), (\hat{u}_1\hat{v}_2 - \hat{u}_2\hat{v}_1))$$

- Wedge product:

$$\hat{\mathbf{u}} \wedge \hat{\mathbf{v}} = u \wedge v + \varepsilon(u \wedge v^0 + u^0 \wedge v) = ((\hat{u}_2\hat{v}_3 - \hat{u}_3\hat{v}_2), (\hat{u}_3\hat{v}_1 - \hat{u}_1\hat{v}_3), (\hat{u}_1\hat{v}_2 - \hat{u}_2\hat{v}_1))$$

2.2.2 Quaternion and dual quaternion

According to Hamilton (1969), a quaternion is a hypercomplex number consisting of a real part and three imaginery parts:

$$\mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} + q_4 \tag{2.27}$$

where q_i (i=1, 2, 3, 4) are real numbers, called the components of **q**, and four quaternion units 1, **i**, **j**, **k** satisfy the relations $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$ and $\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$. A unit quaternion is a quaternion with $\left\|\mathbf{q}^2\right\| = \sum q_i^2 = 1$. Let **P** and **Q** be two quaternions. The quaternion has the following properties:

- Addition: $\mathbf{p}+\mathbf{q} = (p_1 + q_1)\mathbf{i} + (p_2 + q_2)\mathbf{j} + (p_2 + q_2)\mathbf{k} + (p_4 + q_4)$
- Multiplication:

$$\mathbf{pq} = (p_4q_1 + p_1q_4 + p_2q_3 - p_3q_2)\mathbf{i} + (p_4q_2 + p_2q_4 + p_3q_1 - p_1q_3)\mathbf{j} + (p_4q_3 + p_3q_4 + p_1q_2 - p_2q_1)\mathbf{k}$$
$$+ (p_4q_4 - p_1q_1 - p_2q_2 - p_3q_3)$$

The conjugate quaternion $\mathbf{q}^*$ of the quaternion **q** is defined by: $\mathbf{q}^* = -q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} + q_4$. From quaternion multiplication, it follows that $\mathbf{qq}^* = q_1^2 + q_2^2 + q_3^2 + q_4^2$

A dual quaternion is the combination of dual number and quaternion, which has the form of:

$$\hat{\mathbf{q}} = \mathbf{q} + \varepsilon\mathbf{q}^0 = \hat{q}_1\mathbf{i} + \hat{q}_2\mathbf{j} + \hat{q}_3\mathbf{k} + \hat{q}_4 \tag{2.28}$$

where the real part $\mathbf{q}$ and the dual part $\mathbf{q^0}$ are both quaternions, and $\hat{q}_l = q_l + \varepsilon q_l^0$, l=1,2,3,4, q_l and q_l^0 are real numbers. The multiplication of two dual quaternions $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$ is as follows:

$$\hat{\mathbf{p}}\hat{\mathbf{q}} = pq + \varepsilon(pq^0 + qp^0)$$

2.2.3 Representing a spatial displacement with a dual quaternion

One basic problem in computer animation is to interpolate a given set of the positions of a rigid body so that the resulting animated motion looks smooth and natural. In general, there are two basic issues in design of motion interpolations. The first one is very basic to kinematics, which concerns with the representation of displacements. The second basic issue is computational geometric in nature and is related to parametrization and piecing of motion interpolations.

The traditional approach for computer animation of 3D objects treats the interpolations of translations and rotations separately. The translation is represented by a vector **d** (point in Euclidean space) and the rotation is represented by an orthogonal matrix [**A**]. Thus, in a traditional approach, a spatial displacement in Euclidean three-space E^3 is expressed by [**A**] and **d** as:

$$\tilde{\mathbf{P}} = \begin{pmatrix} & [\mathbf{A}] & & \mathbf{d} \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{P} \tag{2.29}$$

where $\tilde{\mathbf{P}}$ and $\mathbf{P}$ are homogeneous coordinates of a point measured in the fixed and moving reference frames.

Dual quaternion $\hat{\mathbf{q}} = \mathbf{q} + \varepsilon\mathbf{q}^0$ can also be used to represent spatial displacement and be an elegant tool to represent the interpolation of the rotations. The real part **q** is a

unit quaternion and the four components of **q** can be expressed by the homogeneous Euler parameters of rotation:

$$\mathbf{q} = (q_1, q_2, q_3, q_4) = (s_1 \sin(\theta/2), s_2 \sin(\theta/2), s_3 \sin(\theta/2), \cos(\theta/2)) \quad (2.30)$$

where $\left\|\mathbf{q}^2\right\| = \sum q_i^2 = 1$, and the parameters ($s_1$, s_2, s_3) define the unit vector **s** along the axis of rotation and θ denotes the angle of rotation. The rotation matrix [**A**] of conventional spatial displacement can be expressed in terms of **q** as:

$$A = \begin{bmatrix} q_4^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_4q_3) & 2(q_1q_3 + q_4q_2) \\ 2(q_2q_1 + q_4q_3) & (q_4^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_4q_1) \\ 2(q_3q_2 - q_4q_2) & 2(q_3q_2 + q_4q_1) & (q_4^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \quad (2.31)$$

Eq. (2.31) contains the information of rotational component of a spatial displacement.

The four components of the dual part $\mathbf{q}^0$ form another quaternion whose components are defined as:

$$\begin{bmatrix} q_1^0 \\ q_2^0 \\ q_3^0 \\ q_4^0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -d_z & d_y & d_x \\ d_z & 0 & -d_x & d_y \\ -d_y & d_x & 0 & d_z \\ -d_x & -d_y & -d_z & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad (2.32)$$

where $\mathbf{d}=(d_x, d_y, d_z)$ is the translation vector. $\mathbf{q^0}$ includes the information of the translation of a spatial displacement. The translation vector **d** can be recovered from ($\mathbf{q}$, $\mathbf{q}^0$) as:

$$\mathbf{d}=2\begin{bmatrix} q_4^0q_1 - q_1^0q_4 + q_2^0q_3 - q_3^0q_2 \\ q_4^0q_2 - q_2^0q_4 + q_3^0q_1 - q_1^0q_3 \\ q_4^0q_3 - q_3^0q_4 + q_1^0q_2 - q_2^0q_1 \end{bmatrix} \quad (2.33)$$

Since the real part of the dual quaternion $\hat{\mathbf{q}}$ represents the rotation of a spatial displacement and the dual part of $\hat{\mathbf{q}}$ represents the translation of a spatial displacement, dual quaternion is capable of representing transformation. Therefore, a spatial displacement in Euclidean three-space E^3 can be expressed by dual quaternion as:

$$\widetilde{\mathbf{P}} = \begin{pmatrix} [\mathbf{A}(\mathbf{q})] & \mathbf{d}(\mathbf{q},\mathbf{q}^0) \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \mathbf{P} \tag{2.34}$$

The matrix transformation of point coordinates as given above can be put in a compact form using quaternion algebra:

$$\widetilde{\mathbf{P}} = \mathbf{q}\mathbf{P}\mathbf{q}^* + p_4(\mathbf{q}^0\mathbf{q}^* - \mathbf{q}\mathbf{q}^{0*})\mathbf{P} \tag{2.35}$$

where"*"denotes the conjugate of a quaternion.

To study the point trajectory of a rational motion defined by a dual quaternion curve, it is more convenient to use the dual-quaternion representation of point coordinate transformation. The dual quaternion representation of spatial displacement offers many advantages over matrix representation. The representation of rotation using quaternion is more compact and faster than the rotation matrix. Moreover, dual quaternion representation can lead to coordinate-frame invariant formulation of motion synthesis problems. By representing a spatial displacement with a dual quaternion, we can also transform the motion design problem into a curve design problem in the space of dual quaternions. This makes it possible for applying curve design techniques in CAGD to the problem of synthesizing parametric motions.

2.2.4 Representing point trajectory using piecewise rational Bézier dual quaternion curve

Given a moving frame O_M - $X_MY_MZ_M$, a fix frame O_F -$X_FY_FZ_F$, and a point **P** in the moving frame, one can get the point trajectory of **P** through the motion of the moving frame relative to the fix frame, as shown in Fig. 2.5. Points $\mathbf{P}_A$ and $\mathbf{P}_B$ are on the point trajectory and represent the intermediate position for point motion. Denoting $\widetilde{\mathbf{P}}$ and **P** as homogeneous coordinates of a point measured in the fixed and moving frames at instant position, one can obtain the transformation between $\widetilde{\mathbf{P}}$ and **P** using dual

quaternion $\hat{\mathbf{q}}$ according to section 2.2.3. Therefore, when point **P** moves to point $\mathbf{P}_A$, transformation between $\tilde{\mathbf{P}}_A$ and $\mathbf{P}_A$ can be represented by $\hat{\mathbf{q}}_A$; similarly, the transformation between $\tilde{\mathbf{P}}_B$ and $\mathbf{P}_B$ can be represented by $\hat{\mathbf{q}}_B$.

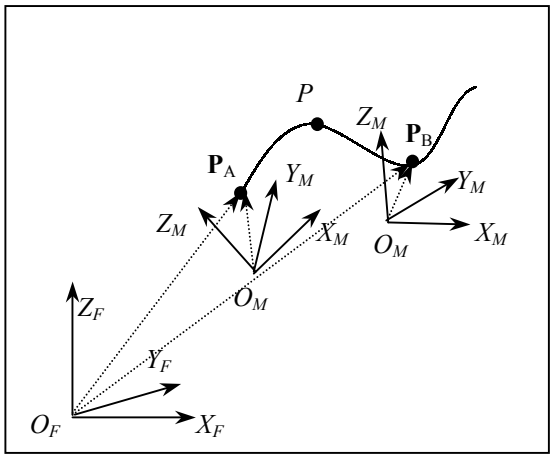


Fig. 2.5 Point trajectory generated by the motion of frame

Given a set of dual quaternion $\hat{\mathbf{q}}_i$ that represents transformation between point $\tilde{\mathbf{P}}$ and **P** at different point positions as point **P** is undergoing motions, one can treat these dual quaternions as the point in CAGD and construct a piecewise rational cubic Bézier dual quaternion curve that represents the motions of point according to sections 2.1.3 and 2.1.4. Therefore one can interpolate the transformation between point $\tilde{\mathbf{P}}$ and **P** at arbitrary position. As a result, according to Eq. (2.35) one can determine the coordinate of point **P** in the fix frame at arbitrary position.

In order to construct the piecewise rational cubic Bézier dual quaternion curve that passes through a set of quaternions $\hat{\mathbf{q}}_i$ (i = 0,…n), where $\hat{\mathbf{q}}_i$ is the dual quaternion representation of transformation at ith position chosen for constructing dual quaternion curve, a set of control dual quaternions $\hat{\mathbf{b}}_j$ (j = 0,…3(n-2)) need to be obtained first according to section 2.1.3 and 2.1.4. Denote $\hat{\mathbf{p}}_i(c) = \hat{\mathbf{b}}_{3c+i}$ (i=0,1,2,3 and c=0,…n-3),

then *c*th segment of the piecewise cubic rational Bézier dual quaternion curve is given as:

$$\hat{\mathbf{q}}(t) = \sum_{j=0}^{3} B_j^3(t)\hat{\mathbf{p}}_j(c) \tag{2.36}$$

where $B_j^3(t)$ denotes the *Bernstein* basis functions. The dual quaternion curve $\hat{\mathbf{q}}(t)$ is:

$$\hat{\mathbf{q}}(t)=\mathbf{q}(t) + \varepsilon\mathbf{q}^0(t)$$

where $\mathbf{q}(t) = \sum_{j=0}^{3} B_j^3(t)\mathbf{p}_j(c)$ and $\mathbf{q}^0(t) = \sum_{j=0}^{3} B_j^3(t)\mathbf{p}_j^0(c)$. Substitute the above into Eq. (2.35) and rearrange it, we can obtain the coordinate of point **P** in the fix frame as:

$$\widetilde{\mathbf{P}}(t) = [H^{2n}(t)]\mathbf{P} \tag{2.37}$$

where $H^{2n}(t) = \sum_{k=0}^{2n} B_k^{2n}(t)[H_k]$

with $$[H_k] = \frac{1}{C_k^{2n}} \sum_{i+j=k} C_i^n C_j^n ([H_i^+][H_j^-] + [H_j^-][H_i^{0+}] - [H_i^+][H_j^{0-}]$$

$$[H_i^+] = \begin{bmatrix} p_{i,4}(c) & -p_{i,3}(c) & p_{i,2}(c) & p_{i,1}(c) \\ p_{i,3}(c) & p_{i,4}(c) & -p_{i,1}(c) & p_{i,2}(c) \\ -p_{i,2}(c) & p_{i,1}(c) & p_{i,4}(c) & p_{i,3}(c) \\ -p_{i,1}(c) & -p_{i,2}(c) & -p_{i,3}(c) & p_{i,4}(c) \end{bmatrix} \quad [H_j^-] = \begin{bmatrix} p_{j,4}(c) & -p_{j,3}(c) & p_{j,2}(c) & p_{j,1}(c) \\ p_{j,3}(c) & p_{j,4}(c) & -p_{j,1}(c) & p_{j,2}(c) \\ -p_{j,2}(c) & p_{j,1}(c) & p_{j,4}(c) & p_{j,3}(c) \\ -p_{j,1}(c) & -p_{j,2}(c) & -p_{j,3}(c) & p_{j,4}(c) \end{bmatrix}$$

$$[H_i^{0+}] = \begin{bmatrix} 0 & 0 & 0 & p_{i,1}^0(c) \\ 0 & 0 & 0 & p_{i,2}^0(c) \\ 0 & 0 & 0 & p_{i,3}^0(c) \\ 0 & 0 & 0 & p_{i,4}^0(c) \end{bmatrix} \qquad [H_j^{0-}] = \begin{bmatrix} 0 & 0 & 0 & -p_{j,1}^0(c) \\ 0 & 0 & 0 & -p_{j,2}^0(c) \\ 0 & 0 & 0 & -p_{j,3}^0(c) \\ 0 & 0 & 0 & p_{j,4}^0(c) \end{bmatrix}$$

CHAPTER 3

SINGLE ISO-PARAMETRIC TOOL PATH GENERATION USING RATIONAL BÉZIER MOTION

In this chapter, an efficient approach to generate a single tool path for 5-axis sculptured surface machining using piecewise rational Bézier motion of a flat-end cutter is presented. Since gouging and collision are the main problems for the 5-axis tool path generation, the algorithms for gouging and collision detection and avoidance are also developed here. The problems of generating multi tool paths that deal with keeping the scallop height in tolerance will be discussed in next chapter.

3.1 The Geometry of 5-axis Machining

Generally, the sculptured surfaces are represented by free-form surface patches, such as Coons, Bézier, B-spline, or recently NURBS (Faux and Pratt, 1981, Piegl and Tiller, 1997). In our application, the designed surface to be machined is a B-spline surface. The basic concepts of B-spline curves and surfaces are introduced in chapter 2. Also, a flat-end cutter is used for sculptured surface machining in our application.

Fig. 3.1 shows the geometry of the designed surface and the cutter in 5-axis machining. O_G-$X_GY_GZ_G$ is the global coordinate system, which is fixed on the workpiece. O_L-$X_LY_LZ_L$ is the local coordinate system that is centred at the CC point.

The CL point is the centre of the bottom of the cutter, while a CL includes the CL point as well as the orientation of the cutter. X_L is in the direction of the tangent vector at the CC point along current cutting direction. Z_L is in the direction of the normal of the surface at the CC point. O_T-$X_TY_TZ_T$ is the cutter coordinate system that is obtained by first rotating angle λ_L around Y_L and second angle -ω_L around Z_L, and then translating $-r$ along X_L, where r is the cutter radius.

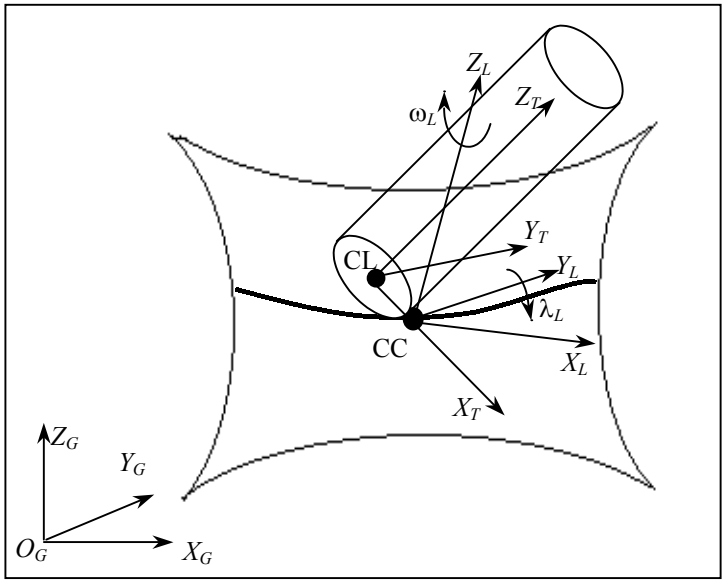


Fig. 3.1 The geometry of 5-axis machining

A point (x_T, y_T, z_T) in the cutter frame can be expressed as point (x_G, y_G, z_G) in the global frame as:

$$\begin{pmatrix} x_G \\ y_G \\ z_G \end{pmatrix} = Rot(G \to L)Rot_y(\lambda_L)Rot_z(-\omega_L)\begin{pmatrix} x_T - r \\ y_T \\ z_T \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{X}_L \cdot \mathbf{X}_G & \mathbf{Y}_L \cdot \mathbf{X}_G & \mathbf{Z}_L \cdot \mathbf{X}_G \\ \mathbf{X}_L \cdot \mathbf{Y}_G & \mathbf{Y}_L \cdot \mathbf{Y}_G & \mathbf{Z}_L \cdot \mathbf{Y}_G \\ \mathbf{X}_L \cdot \mathbf{Z}_G & \mathbf{Y}_L \cdot \mathbf{Z}_G & \mathbf{Z}_L \cdot \mathbf{Z}_G \end{pmatrix} \begin{pmatrix} \cos\lambda_L & 0 & \sin\lambda_L \\ 0 & 1 & 0 \\ -\sin\lambda_L & 0 & \cos\lambda_L \end{pmatrix} \begin{pmatrix} \cos\omega_L & \sin\omega_L & 0 \\ -\sin\omega_L & \cos\omega_L & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_T - r \\ y_T \\ z_T \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \tag{3.1}$$

where $(x_c, y_c z_c)$ is the global coordinate of the *CC* point; ($\mathbf{X}_L$, $\mathbf{Y}_L$, $\mathbf{Z}_L$) and ($\mathbf{X}_G$, $\mathbf{Y}_G$, $\mathbf{Z}_G$) are three unit coordinate axis vectors of the local and global coordinate system respectively.

The transformation between the cutter frame and the global frame can also be represented by dual quaternion $\hat{\mathbf{q}}$ as:

$$\hat{\mathbf{q}} = \hat{\mathbf{q}}_{trLG} \cdot \hat{\mathbf{q}}_{r\lambda} \cdot \hat{\mathbf{q}}_{-r\omega} \cdot \hat{\mathbf{q}}_{-tr} \tag{3.2}$$

where $\hat{\mathbf{q}}_{trLG}$ is the dual quaternion that represents the transformation between the global frame and the local frame; $\hat{\mathbf{q}}_{r\lambda}$ represents rotation of angle λ_L around Y_L; $\hat{\mathbf{q}}_{-r\omega}$ is the dual quaternion that represents rotation of angle $-\omega_L$ around Z_L; $\hat{\mathbf{q}}_{-tr}$ represents the translation of $-r$ along X_L. Therefore, a point (x_T, y_T, z_T) in the cutter frame can be expressed as point (x_G, y_G, z_G) according to Eq. (3.2) and Eq. (2.35).

3.2 Representation of Cutter Bottom Circle Undergoing Rational Bézier Motion

The point trajectory undergoing piecewise rational Bézier motion is given by Eq. (2.37). We can determine the representation of the bottom circle of a flat-end cutter undergoing a piecewise rational Bézier motion by applying the similar principle.

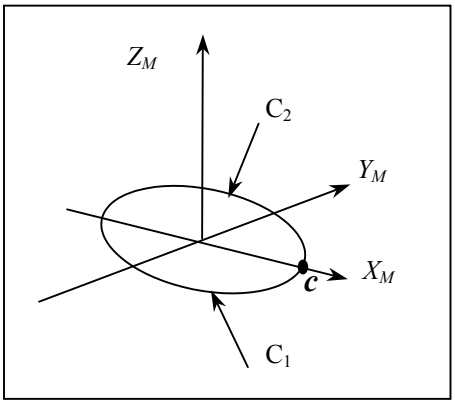


Fig. 3.2 Position of cutter bottom circle in the moving frame

We firstly construct the cutter bottom circle in the moving frame and assume that the axis of the cutter is along the *Z*-axis, as shown in Fig. 3.2. Then, we use the rational Bézier curve to construct this cutter bottom circle. Generally, although non-

rational curves can be used to represent most curves and offer many advantages, there exist a number of important curve and surface types, such as circles, ellipses and spheres, which cannot be represented precisely using the polynomials. Therefore, in our case, rational Bézier curve is applied to represent the circle (Xia, 2001). The idea is to use homogenous coordinates to represent a rational curve in 3-dimensional space as a polynomial curve in 4-dimension space. Denoting r as the radius of the cutter bottom circle and taking $\mathbf{P}_0$=(r,0,0,1), $\mathbf{P}_1$=(0,-r,0,0), $\mathbf{P}_2$=(-r,0,0,1) as the homogeneous coordinates of three Bézier control points, we can obtain the expression of the half circle C_1 below X_M axis in X_MY_M plane as:

$$\mathbf{P}(s) = \sum_{i=0}^{2} B_i^2(s)\mathbf{P}_i \tag{3.3}$$

Similarly, the circular arc C_2 above X_M axis in X_MY_M plane can be represented by the same formula except $\mathbf{P}_1$=(0,r,0,0). According to Eq. (2.37), the swept surface of circular arc of cutter bottom circle undergoing the rational cubic Bézier motion can be expressed as:

$$\mathbf{P}(s,t) = [H^6(t)]\sum_{i=0}^{2} B_i^2(s)\mathbf{P}_i = \sum_{k=0}^{6}\sum_{i=0}^{2} B_k^6(t)B_i^2(s)[H_k]\mathbf{P}_i \tag{3.4}$$

From Eq. (3.4), we know that the swept surface of cutter motion is a tensor product Bézier surface.

3.3 A Single Iso-parametric Tool Path Generation Using Rational Bézier Cutter Motion

Traditional methods for tool path generation suffer several disadvantages. First, because of the lack of compact representation of tool path, a huge number of discrete *CC* points need to be generated. Second, the possible gouging and collision need to be checked for each of these *CC* points. All of these would result in heavy computational

load. In order to overcome these disadvantages, we present a method for iso-parametric tool path generation using rational Bézier cutter motion. The rational Bézier dual quaternion curve is constructed to obtain the mathematic forms of the cutter motion along the surface curve; thus, the computation of large number of discrete CC points is avoided. Since the swept surface of cutter bottom circle undergoing rational Bézier motion can be expressed in Eq. (3.4), gouging checking can be considered as finding interference between the swept surface of cutter bottom undergoing rational Bézier motion and the designed surface, while gouging avoidance can be considered as modifying the swept surface of cutter bottom motion so that the interference between the swept surface and the designed surface no longer exists. Collision checking and detection can also be performed based on the swept surface of cutter bottom undergoing rational Bézier motion.

The algorithm of tool path generation using rational Bézier cutter motion is implemented in four steps.

(1) Given an iso-parametric curve on the design surface, the first step is to find several discrete CC points on the curve based on a fitting tolerance. The corresponding CLs that do not cause any gouging are then determined.

(2) Secondly, we convert these CLs to the dual quaternion representation so that the rational Bézier dual quaternion curve can be constructed.

(3) In the third step, the rational Bézier dual quaternion curve for the cuter motion is constructed and the swept surface generated by the rational Bézier motion of cutter bottom is obtained.

(4) Finally, fitness and interference checking between the swept surface and the designed surface is performed and correction of the rational Bézier dual

quaternion curve is carried out if the tool path is unsatisfactory. Thus, a satisfactory tool path is obtained.

3.3.1 Determining the cutter contact (CC) points

In this section, the aim is to find a set of CC points along an iso-parametric curve on a sculptured surface such that the deviation between the tool path and the underlying curve is within a given fitting tolerance. Since the rational Bézier cutter motion yields the tool path, the more closed match between this tool path with the underlying curve on the designed surface, the more accurate the 5-axis machining is. Apparently, the selected CC points affect the fitting error between the tool path and the underlying curve on the designed surface. Therefore, the locations of these CC points, which are on the underlying curve, should represent the feature points of this curve, i.e., the number and distributions of these CC points depend on the curvature of the surface curve. In general, points should be dense for the part of the curve with higher curvature, while sparse with lower curvature.

Given a sculptured surface $\mathbf{S}(u,v)$, a surface curve $\mathbf{S}(u_0,v)$ can be obtained with $u = u_0$. The first CC point is at $v = 0$. Then the other CC points on this curve need to be determined adaptively by considering the fitting error between the linearly linked CC points and $\mathbf{S}(u_0,v)$. Two steps are involved in determining the next CC point. First, a search for the adaptive step size L_i is conducted, which is based on the local curvature of surface curve at the ith CC point and the pre-defined fitting tolerance τ. After that, the conversion from the step size L_i to the increment of parameter Δv_i is performed and the next CC point is obtained.

The fitting tolerance τ should be chosen based on the given surface error tolerance. However, if we use the surface error tolerance directly as τ, we will end up

with a large number of CC points. In our approach, these CC points only serve as a starting set and the final surface error will only be accurately checked and ensured after the whole tool path is generated. Therefore, we use a rather relaxed bound as the fitting tolerance. Based on our simulation experiments, the fitting tolerance is set between 5 to 10 times of the surface error tolerance. In this way, we found that the total number of the final CC points is normally much less than the number of CC points obtained using the surface error tolerance as the fitting tolerance.

First, denote $\mathbf{S}_u=\partial\mathbf{S}(u,v)/\partial u$, $\mathbf{S}_v=\partial\mathbf{S}(u,v)/\partial v$, $\mathbf{n}=(\mathbf{S}_u\times\mathbf{S}_v)/|\mathbf{S}_u\times\mathbf{S}_v)|$, $\mathbf{S}_{uu}=\partial\mathbf{S}(u,v)/\partial u^2$, $\mathbf{S}_{vv}=\partial\mathbf{S}(u,v)/\partial v^2$, $\mathbf{S}_{uv}=\partial\mathbf{S}(u,v)/\partial u\partial v$. According to Faux and Pratt (1981) the first fundamental matrix G and the second fundamental matrix D of the designed surface are given as follows:

$$G=\begin{bmatrix}\mathbf{S}_u\cdot\mathbf{S}_u & \mathbf{S}_u\cdot\mathbf{S}_v\\ \mathbf{S}_v\cdot\mathbf{S}_u & \mathbf{S}_v\cdot\mathbf{S}_v\end{bmatrix}=\begin{bmatrix}g_{11} & g_{12}\\ g_{21} & g_{22}\end{bmatrix}\qquad D=\begin{bmatrix}\mathbf{n}\cdot\mathbf{S}_{uu} & \mathbf{n}\cdot\mathbf{S}_{uv}\\ \mathbf{n}\cdot\mathbf{S}_{vu} & \mathbf{n}\cdot\mathbf{S}_{vv}\end{bmatrix}=\begin{bmatrix}d_{11} & d_{12}\\ d_{21} & d_{22}\end{bmatrix} \quad (3.5)$$

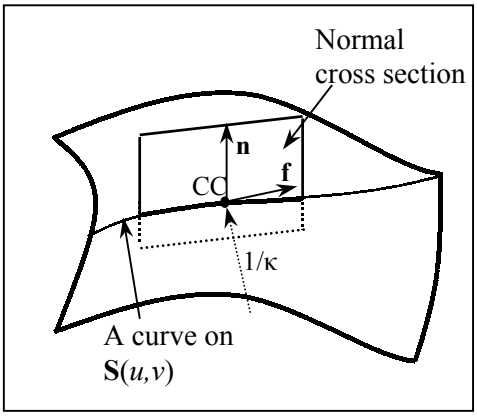


Fig. 3.3 Local surface curvature

As shown in Fig. 3.3, the normal curvature κ_n of the surface in the direction of the tangent vector **f** is the curvature of the intersection curve between the surface and the plane containing the surface normal **n** and **f**. The definition of the normal curvature is given as:

$$\kappa_n=\frac{F^{\mathrm{T}}\mathrm{D}F}{F^{\mathrm{T}}\mathrm{G}F} \quad (3.6)$$

where $F = [\mathbf{S}_u \cdot \mathbf{f}\ ,\ \mathbf{S}_v \cdot \mathbf{f}\ ]^T$. Expanding Eq. (3.6), define $\sigma = \mathbf{S}_u \cdot \mathbf{f} / \mathbf{S}_v \cdot \mathbf{f}$, we get

$$\kappa_n = \frac{d_{11}\sigma^2 + 2d_{12}\sigma + d_{22}}{g_{11}\sigma^2 + 2g_{12}\sigma + g_{22}} \tag{3.7}$$

With this definition, the curvature κ_n is positive when the curve is turning towards the positive direction of the surface normal. Otherwise, κ_n is negative.

Given the current CC point $\mathbf{C}_i$, one needs to find the next CC point $\mathbf{C}_{i+1}$ on $\mathbf{S}(u_0,v)$ and to ensure that the fitting error between two neighboring CC points is within the fitting tolerance τ. The step forward size L_i can be calculated based on the local surface curvature κ_n at $\mathbf{C}_i$ in the direction of cutting direction. The geometry of surface curve $\mathbf{S}(u_0,v)$ at the vicinity of $\mathbf{C}_i$ can be considered as a circular curve, as shown in Fig. 3.4a, and the radius of the circle is the radius of curvature κ_n at $\mathbf{C}_i$ in the direction of cutter direction. Then the step forward size L_i is given by:

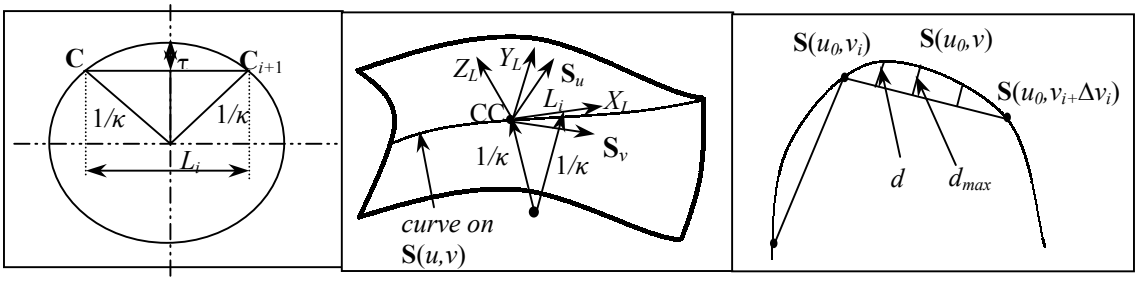
$$L_i = \sqrt{\frac{8\tau - 4\tau^2\kappa_n}{\kappa_n}} \tag{3.8}$$

The step forward size L_i is then converted to the parameter increments Δv_i, so that we can get the next CC point $\mathbf{S}(u_0,v_i+\Delta v_i)$. As shown in Fig. 3.4b, the conversion is determined by solving the following equation:

$$\Delta u_i(S_u \cdot \mathbf{X}_L) + \Delta v_i(S_v \cdot \mathbf{X}_L) = L_i \text{ and } \Delta u_i = 0 \tag{3.9}$$

Then

$$\Delta v_i = L_i / (S_v \cdot \mathbf{X}_L) \tag{3.10}$$



(a) (b) (c)

Fig. 3.4 The geometry of surface curve $\mathbf{S}(u_0,v)$ at the vicinity of $\mathbf{C}_i$

The parameter increment Δv_i is only an approximated value since the surface curve $S(u_0,v)$ at the vicinity of C_i is considered as a circular curve. Therefore, we need to calculate the maximum deviation d_{max} between the points on surface curve $S(u_0,v)$, where $v_i \le v \le v_i + \Delta v_i$, and the linear segment connected by $S(u_0, v_i)$ and $S(u_0, v_i + \Delta v_i)$, as shown in Fig. 3.4c. If d_{max} is larger than τ, decrease the parameter increment Δv_i, and recalculate d_{max} again until it is less than τ. The overall procedure for finding the CC points can be summarized as follows:

(a) Add $\mathbf{C}_0$ ($\mathbf{S}(u_0, 0)$) to the CC point set $\mathbf{C}$. Set the current CC point $\mathbf{C}_i = \mathbf{C}_0$.

(b) Calculate the curvature κ_i of $\mathbf{C}_i$ according to Eq. (3.7).

(c) Obtain the step forward size L_i according to Eq. (3.8).

(d) Convert L_i to the parameter increment of Δv_i using Eq. (3.10).

(e) Calculate the maximum deviation d_{max} between $\mathbf{S}(u_0,v)$, where $v_i \le v \le v_i + \Delta v_i$, and linear segment $\overline{\mathbf{S}(u_0, v_i)\mathbf{S}(u_0, v_i + \Delta v_i)}$. If $d_{max} > \tau$, reduce Δv_i while ensuring d_{max} is no larger than τ.

(f) If $v_i + \Delta v_i < 1$, add point $\mathbf{C}_{i+1} = \mathbf{S}(u_0, v_i + \Delta v_i)$ into the CC point set, and $i = i + 1$, then go back to step (b). Otherwise, add point $\mathbf{C}_{i+1} = \mathbf{S}(u_0, 1)$ into the CC-point set, and stop.

3.3.2 Obtaining the associated gouging–free and collision-free cutter locations (CLs)

Given a set of CC points calculated in section 3.3.1, one can obtain the CLs if the orientations (the inclination angle λ_L and tilting angle ω_L) of the cutter are given at each CC point. However, interference has to be avoided when we determine the

orientations of the cutter. There are three kinds of interference between the cutter and the designed surface as shown in Fig. 3.5: local gouging, rear gouging and collision. Generally, local gouging refers to the removal of excess material in the vicinity of the cutter contact point due to the mismatch in curvatures between the cutter and the designed surface at the CC point. Rear gouging occurs at a CC point when cutter bottom plane interferes with the designed surface at positions other than the CC point. To avoid local gouging, we need to adjust the inclination and tilt angles of the cutter to make sure that the curvature of the cutter is larger than the surface curvature in the cutting plane. To avoid rear gouging, we need to guarantee that the cutter bottom plane does not interfere with the designed surface. Collision is regarded as the global gouging that the cylindrical portion of cutter interferes with the part surface or the machine tool.

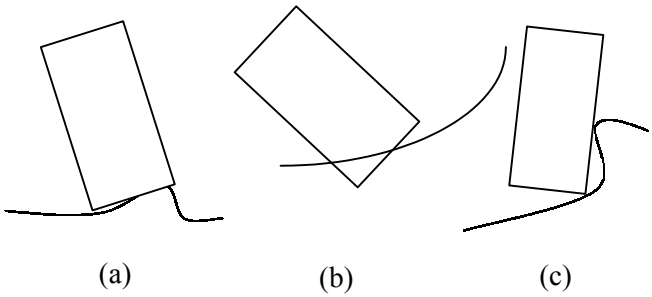
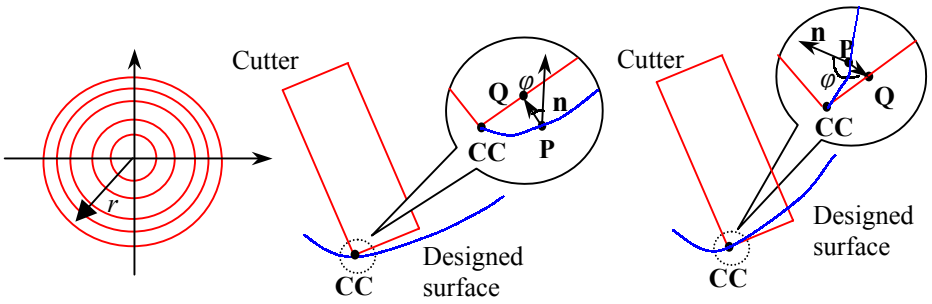


Fig. 3.5 Three kinds of interference in 5-axis machining

(a) Rear gouging (b) Local gouging (c) Collision

The proposed method to obtain the gouging-free and collision-free CLs follows a checking-correction approach. First, the default inclining and tilting angles are assigned. This is followed by a checking procedure to find whether there is any interference between the cutter (including the cutter bottom plane and the cylindrical surface of the cutter) and the designed surface. If interference exists, a correction

procedure is applied to change these two angles. The checking and correction are repeated until no interference is found.



(a) (b) (c)

Fig. 3.6 Interference checking of cutter bottom plane and designed surface

Intersection checking between the cutter bottom plane and the designed surface is not a trivial task. Here, we simplify this problem by converting the cutter bottom plane into a limited number of circles (see Fig. 3.6a), and then checking the intersection between the circles and the designed surface. As shown in Fig. 3.2, the cutter bottom plane can be expressed in cutter frame as: $x_T = tr\cos\theta$, $y_T = tr\sin\theta$, $z_T = 0$, where t varies from 0 to 1 and θ from 0 to 2π. If the increment of t between any two neighboring circles is sufficiently small, this expression can be said to be sufficiently accurate. This expression is then transformed to the global frame according to Eq. (3.1) as (assuming X_G=(1,0,0), Y_G=(1,0,0), Z_G=(1,0,0), X_L=(x_1, x_2, x_3), Y_L=(y_1, y_2, y_3), Z_L=(y_1, y_2, y_3)):

$$x_G=(x_1\cos\omega\cos\lambda - y_1\sin\omega - z_1\sin\lambda\cos\omega)(tr\cos\theta - tr)+ (x_1\cos\lambda\sin\omega + y_1\cos\omega - z_1\sin\lambda\cos\omega)tr\sin\theta + x_c$$

$$y_G=(x_2\cos\omega\cos\lambda - y_2\sin\omega - z_2\sin\lambda\cos\omega)(tr\cos\theta - tr)+ (x_2\cos\lambda\sin\omega + y_2\cos\omega- z_2\sin\lambda \cos\omega)tr\sin\theta + y_c$$

$$z_G=(x_3\cos\omega\cos\lambda - y_3\sin\omega - z_3\sin\lambda\cos\omega)(tr\cos\theta - tr)+ (x_3\cos\lambda\sin\omega + y_3\cos\omega- z_3\sin\lambda \cos\omega)tr\sin\theta + z_c$$

(3.11)

where $0 \le \lambda_L \le \pi/2$, $0 \le \omega_L \le 2\pi$, (x_c, y_c, z_c) is the global coordinate of the CC point.

We can also simplify the intersection checking between the cylindrical surface of the cutter and the designed surface by converting the cylindrical surface into a limited number of circles (see Fig. 3.7a), and then checking the intersection between the circles and the designed surface. According to Fig. 3.2, the cylindrical surface of the cutter can be expressed in cutter frame as: $x_T = r\cos\theta$, $y_T = r\sin\theta$, $z_T = h$, where θ from 0 to 2π and h from 0 to the height of cutter. This expression is then transformed to the global frame according to Eq. (3.1) as:

$$x_G=(x_1\cos\omega\cos\lambda - y_1\sin\omega - z_1\sin\lambda\cos\omega)(r\cos\theta - r)+ (x_1\cos\lambda\sin\omega + y_1\cos\omega- z_1\sin\lambda \cos\omega)r\sin\theta +(x_1\sin\lambda+ z_1\cos\lambda)h+ x_c$$

$$y_G=(x_2\cos\omega\cos\lambda - y_2\sin\omega - z_2\sin\lambda\cos\omega)(r\cos\theta - r)+ (x_2\cos\lambda\sin\omega + y_2\cos\omega- z_2\sin\lambda \cos\omega)r\sin\theta +(x_2\sin\lambda+ z_2\cos\lambda)h + y_c$$

$$z_G=(x_3\cos\omega\cos\lambda - y_3\sin\omega - z_3\sin\lambda\cos\omega)(r\cos\theta - r)+ (x_3\cos\lambda\sin\omega + y_3\cos\omega- z_3\sin\lambda \cos\omega)r\sin\theta +(x_3\sin\lambda+ z_3\cos\lambda)h+ z_c$$

(3.12)

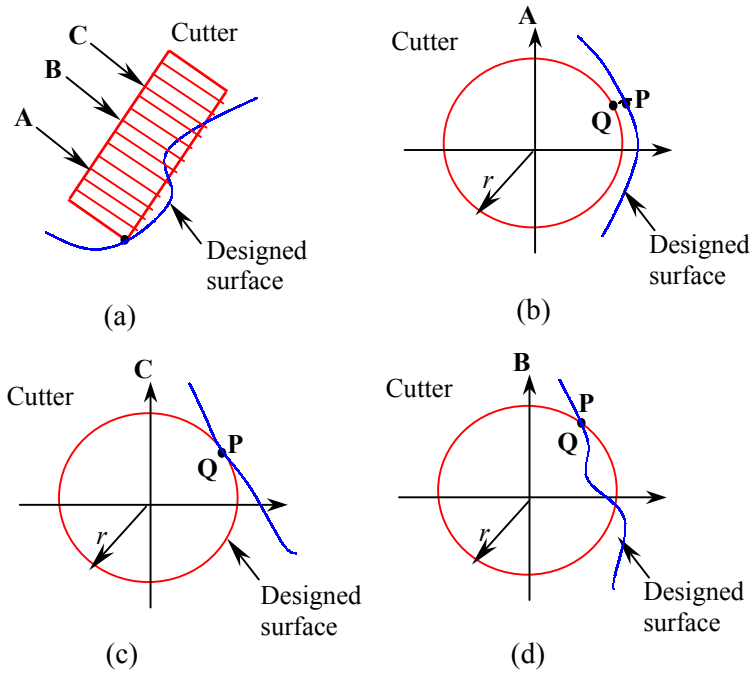


Fig. 3.7 Interference checking of designed surface and cutter cylindrical surface

To check the possible intersection between a circle and the designed surface, we calculate the minimal distance between the circle and the surface. This problem is formulated as:

$$\text{Minimise } \textit{distance}[\mathbf{P}(x_G, y_G, z_G), \mathbf{S}(u, v)] = f(\theta, u, v)$$

$$\text{Subject to: } x_G, y_G, z_G \text{ satisfies Eq. (3.11) or Eq. (3.18)}$$

$$0 \le u \le 1 \text{ and } 0 \le v \le 1$$

Where $\mathbf{P}(x_G, y_G, z_G)$ is a point on the circle, and $\mathbf{S}(u,v)$ is a point on the designed surface.

Different search algorithms are available to solve this optimisation problem, such as Levenberg-Marquardt Algorithm, Downhill Simplex Algorithm, Simulated Annealing and Grid Search. Downhill Simplex method (Nelder and Mead, 1965) differs from the other methods in that it does not use derivatives, which confers safer convergence properties to the Simplex method since it is much less prone to finding

false minima. One of the more remarkable features of the Downhill Simplex algorithm is that no divisions are required. It uses linear adjustment of the parameters until some convergence criterion is met. We therefore apply Downhill Simplex algorithm to find the minimum distance:

Denote the identified point on the cutter bottom as **Q** and the point on the designed surface as **P**, the existence of gouging is determined in the following scenarios:

(1) If the minimum distance is closed to zero ($|\mathbf{P} - \mathbf{Q}| < \rho$, where ρ is a very small value) and **Q** is outside the vicinity of the CC point, gouging is said to occur.

(2) If $|\mathbf{P} - \mathbf{Q}| \geq \rho$, there are two possibilities. The first is that no interference exists (see Fig. 3.6b) and the second is that the cutter bottom is completely below the designed surface (see Fig. 3.6c). To distinguish these two scenarios, we calculate the angle, φ, between vector **PQ** and the normal vector at **P**, i.e., **n**. If $\varphi \leq 90^{\circ}$, as shown in Fig. 3.6b, there will be no gouging. If, however, $\varphi > 90^{\circ}$, gouging will occur.

Denote the identified point on the cutter cylindrical surface as **Q** and the point on the designed surface as **P**, the existence of collision is determined in the following scenarios (see Fig. 3.7b,c,d):

(1) If the minimum distance is closed to zero ($|\mathbf{P} - \mathbf{Q}| < \rho$, where ρ is a very small value) collision is said to occur.

(2) If $|\mathbf{P} - \mathbf{Q}| \geq \rho$, we transform the point **P** on the designed surface in the global frame to the cutter frame. If the transformed $\mathbf{P}_T$ is within the volume of the cutter, collision occurs. Otherwise, there is no collision.

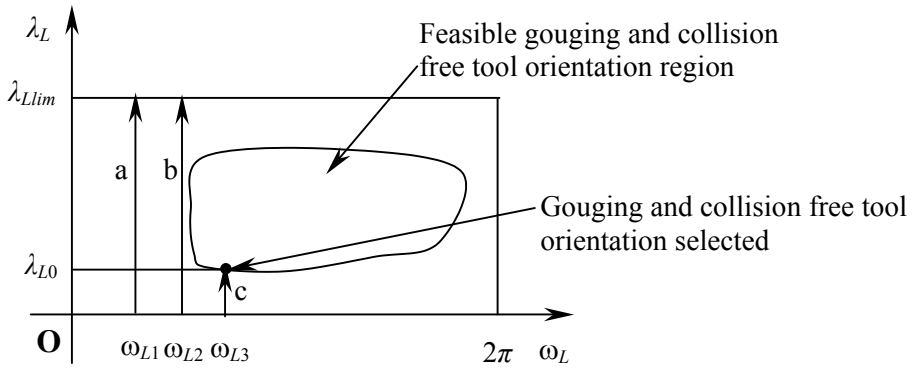


Fig. 3.8 Finding the gouging-free and collision-free tool orientation

If gouging or collision exists, the corresponding CL needs to be adjusted. The strategy for adjusting the inclining and tilting angles is given as follows (see Fig. 3.8 for reference):

(1) Increase λ_L by a small amount and keep ω_L unchanged. Check the existence of gouging and collision.

(2) If no gouging and collision exists, output λ_L and ω_L, stop. Otherwise, check if the machine limit for λ_L is reached. If so, go to step (3). If not, go back to step (1).

(3) Increase ω_L by a small amount and keep λ_L at its default value. Go back to step (1).

It is worth mentioning that in the above procedure, there is no checking for the machine limit for ω_L. This is based on an assumption that for a given CC point, a gouging-free and collision-free pair of λ_L and ω_L always exists.

3.3.3 Constructing the dual quaternion curve of cutter motion for a single tool path

Given a set of CLs, their dual quaternion representations $\hat{\mathbf{q}}_i$ can be obtained using Eq. (3.2). In this section, we focus on constructing a piecewise rational cubic Bézier

motion that interpolates or approximates the arbitrary cutter location on one tool path using the dual quaternion representation $\hat{\mathbf{q}}_i$ of these CLs. The problem can be described as follows: Given a set of dual quaternion $\hat{\mathbf{q}}_i (0 \le i \le n)$ generated from the CLs for one tool path, and corresponding knot sequence $\bar{u}_i$ $(0 \le i \le n)$, find a piecewise rational cubic Bézier dual quaternion curve **Q** determined by the knots sequence and a set of control dual quaternions $\hat{\mathbf{b}}_j$ (j = 0,…3(n-2)) such that $\mathbf{Q}(\bar{u}_i) = \hat{\mathbf{q}}_i$. In section 2.2.4 of chapter 2, we have already presented the algorithm to solve this problem. Therefore, the swept surface of cutter bottom circle undergoing the piecewise rational cubic Bézier motion can be represented as a set of tensor product Bézier surface and each of these Bézier surfaces can be expressed in Eq. (3.4) with different [H_k].

3.3.4 Tool path verification and modification

The piecewise rational Bézier dual quaternion curve for the motion of the cutter in section 3.3.3 represents a complete tool path. Over this tool path, however, the cutter positions between the neighboring seed CLs may cause out-of-bound surface error and/or gouging and collision. Therefore, verification of the complete tool path on surface error, gouging and collision must be carried out. Here, we use the swept surface generated by the motion of cutter bottom and the underlying surface to check the existence of surface error and gouging.

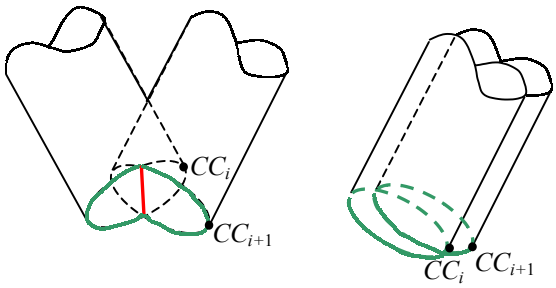


Fig. 3.9 Two types of swept surfaces generated by rational motion of cutter bottom

The swept surface of the bottom of a cylindrical cutter undergoing rational motion is considered to be formed by two types of surfaces: one is generated by the cutter bottom surface within the bottom circle (type-I) and the other by the rational motion of cutter bottom circle (type-II). Fig. 3.9a shows the type-I surface. The red line is part of swept surface generated by the motion of cutter bottom surface. Fig. 3.9b shows the type-II surface in which the green line is generated by the motion of cutter bottom circle.

To check whether there is any gouging over the tool path, we need to check whether there is any interference between the swept surface (both type-I and type-II) and the designed surface. For surface error checking, however, we only need to check the deviation between type- II surface and the designed surface. This is because the final generated surface must be the swept surface formed by the motion of the bottom circle if the tool path is gouging-free.

The surface fitness, gouging and collision checking are carried out on a number of selected CL points. The deviation between type-II surface and the designed surface is checked firstly. If the deviation is out of tolerance, the corresponding tolerance-violation CLs are recorded. Gouging checking is subsequently carried out over the same set of CL points and the corresponding gouging CLs are also recorded. After that, collision checking is subsequently carried out over the same set of CL points and

the corresponding collision CLs are also recorded. These problematic CLs are then modified and the quaternion motion curve is reconstructed. Subsequently, fitness, gouging and collision checking are carried out again. This process goes on until a gouging-free and collision-free tool path with a satisfactory fitness is achieved. The algorithms are described in the following sections.

3.3.4.1 Fitness checking

To achieve a good fitness, the deviation between the swept surface generated by the motion of cutter bottom and the underlying surface should be within the user specified surface tolerance. The fitness checking is to calculate such deviation and find whether it is larger than the surface tolerance. If true, modification of piecewise rational cubic dual quaternion curve of cutter bottom motion is needed. Modification of dual quaternion curve is an iterative process. Some crucial CL points that yield violation of fitness requirements are added to the original set of cutter location points, and then the dual quaternion motion curve is reconstructed according to the algorithm in section 3.3.3. Subsequently, the swept surface generated by the motion of cutter bottom is updated until a good fitness is achieved.

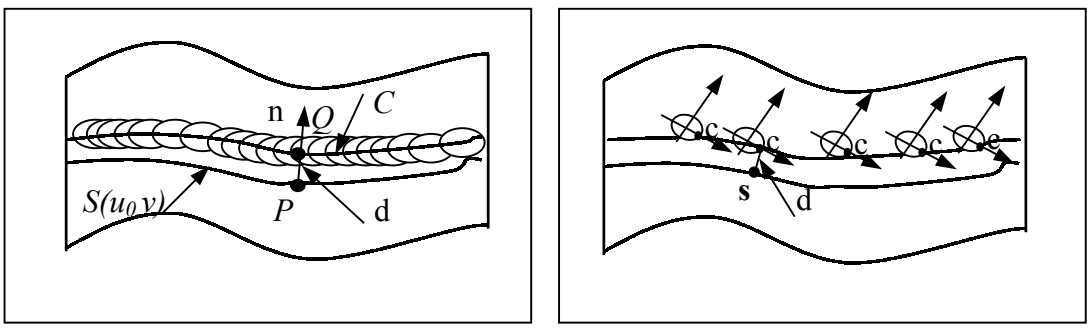


Fig. 3.10 Two kinds of deviation estimation between swept and designed surface

The deviation between the swept surface and designed surface is defined as follows:

(1) The projection of the surface curve $\mathbf{S}(u_0, v)$ onto the swept surface generated by the motion of cutter bottom in the direction of the normal vectors of all the surface points on $\mathbf{S}(u_0,v)$ yields a curve C on the swept surface. Then the deviation between curve C and the surface curve $\mathbf{S}(u_0,v)$ can be an estimation as the deviation between the swept surface and the designed surface.

(2) For any point **P** on the surface curve $\mathbf{S}(u_0,v)$, the deviation between $\mathbf{S}(u_0,v)$ and curve C is obtained by calculating the distance between point **P** and its projection point **Q** on curve C according to its normal direction. After the calculation of distances for all the surface points on curve $\mathbf{S}(u_0,v)$, the largest distance is defined as the deviation between surface curve $\mathbf{S}(u_0,v)$ and curve C. It is also the deviation between the swept surface and the designed surface curve (see Fig. 3.10a).

Obviously, the calculation of deviation according to this definition is time-consuming and complicated. A simplified method is needed to estimate the deviation between the swept surface generated by the motion of cutter bottom and the surface $\mathbf{S}(u, v)$. Instead of using the projection curve C, we can use another curve to estimate deviation between swept surface and the designed surface. As shown in Fig. 3.10b and Fig. 3.2, point **c** in cutter bottom circle is always the CC point corresponding to any location of surface $\mathbf{S}(u_0,v)$. We can then use the curve formed by point **c** undergoing the rational Bézier motion to approximate curve C. After that, at any location of cutter undergoing the Bézier motion, we can obtain the minimal distance between point **c** at this location and the surface curve $\mathbf{S}(u_0,v)$. If this minimal distance is larger than the surface tolerance, the deviation between the swept surface and designed surface must be out of

tolerance, and thus the modification of the dual quaternion motion curve needs to be performed. The detail implement procedure is as follows:

(a) Find the curve generated by point ***c*** undergoing rational Bézier motion

As shown in Fig. 3.2, since point ***c*** (CC point) is the start point of two circular arcs expressed in Eq. (3.3), the parameter $s = 0$. Thus, according to Eq. (3.4), the curve generated by point ***c*** undergoing the B-spline motion is given by $\mathbf{P}(0,t)$ as follows:

$$\mathbf{P}(0,t) = [H^6(t)]\sum_{i=0}^{2} B_i^2(0)\mathbf{P}_i = \sum_{k=0}^{6}\sum_{i=0}^{2} B_k^6(\mathrm{t})B_i^2(0)[H_k]\mathbf{P}_i \tag{3.13}$$

(b) Discrete the curve $\mathbf{P}(0, t)$ to find a set of instant points $\boldsymbol{c}_i$ on the curve.

This step is to generate a set of CC points from $\mathbf{P}(0, t)$. For implementation, the method for CC point generation described in section 3.3.1 is adopted, in which the designed surface $\mathbf{S}(u,v)$ and surface curve $\mathbf{S}(u_0,v)$ are replaced by the swept surface $\mathbf{P}(s, t)$ and the curve $\mathbf{P}(0, t)$ respectively. The given surface error tolerance is used as the fitting tolerance for CC point generation. The generated CC point set corresponds to a set of parameters $\{t_i\ i = 1, 2, \ldots, K\}$ which also correspond to a set of CLs whose bottom circles are $\{\mathrm{P}(s, t_i),\ i = 1, 2, \ldots, K\}$(see Fig. 3.11 for reference).

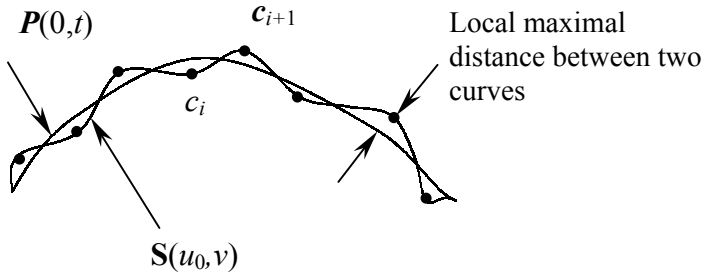


Fig. 3.11 Finding a set of instant points $\boldsymbol{c}_i$ on the curve $\mathrm{P}(0,t)$

(c) Calculate the distance between the point c_i and the surface curve $\mathrm{S}(u_0,v)$.

The calculation of the minimum distance between the CC point at t_i and the underlying curve on the designed surface can, therefore, be defined as:

Minimise $distance[\mathbf{P}(0, t_i), \mathbf{S}(u_0, v)] = g(v)$

Subject to: $0 \leq v \leq 1$

This ensures that the surface error between the machined surface and the designed surface is within the surface error tolerance on the tool path curve $\mathbf{S}(u_0, v)$.

The Downhill Simplex algorithm is used here to solve this optimisation problem. The output gives a set of K points on $\mathbf{S}(u_0,v)$, $\{\mathbf{S}(u_0, v_i), i = 1, 2, \ldots, K\}$, that corresponds to the minimum distances at the K CLs. If the minimal distance at a CL is found to be larger than the surface error tolerance, the corresponding point, which belongs to $\{\mathbf{S}(u_0, v_i), i = 1, 2, \ldots, K\}$, that yields this distance is recorded into a *supplementary CC-point set*. The CLs that satisfy the tolerance are to be used for gouging and collision checking.

3.3.4.2 Gouging and collision checking

As shown in Fig. 3.12, the interference detection needs to be carried out at instant cutter locations. For simplicity, in our application, the instant cutter locations for interference detection are the same set of cutter locations determined in the process of checking the fitness. Therefore, interference detection for one tool path is decomposed to the detection of interference between cutter and the designed surface at all such instant cutter locations. This method is similar to the gouging and collision detection method introduced in section 3.3.2. However, there is still a little difference between them. The adjustment of the incline angle of cutter at instant location is no longer needed at this stage, since the adjustment will be done later by modifying the rational Bézier dual quaternion curve.

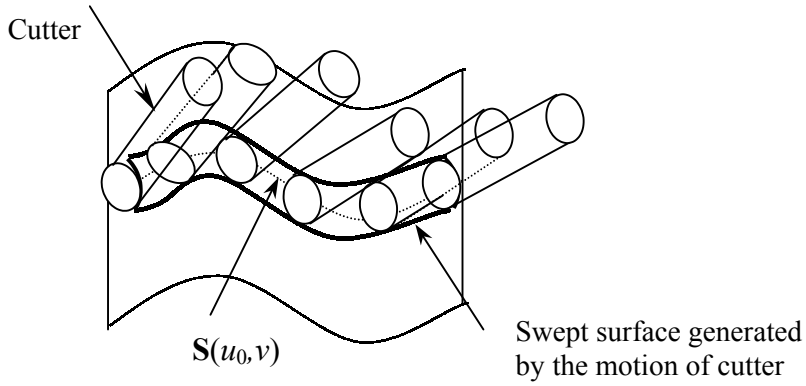


Fig. 3.12 Interference checking for one tool path

Note that the minimal distance calculated for interference detection is between the cutter and the designed surface as a whole. If interference occurs at a CL (t_i), the corresponding point on $\mathbf{S}(u_0,v)$ should be found and added to the supplementary CC-point set. The requirement for this point is that by using it as a CC point, a feasible CL can be found to avoid interference. Therefore, there should be more than one solution to this point. In our approach, we use the point on $\mathbf{S}(u_0,v)$ that corresponds to the minimal distance between $\mathbf{P}(0, t_i)$ and $\mathbf{S}(u_0,v)$, i.e., from $\{\mathbf{S}(u_0, v_i), i = 1, 2, \ldots, K\}$, which is obtained during the above fitting checking procedure.

Finally, the supplementary CC-point set is completed, which will be used to modify the curve that defines the cutter motion.

3.3.4.3 Modification of the rational Bézier dual quaternion curve

Having found the problematic CLs (on the dual quaternion curve), a direct modification of this approach is to use the points in the supplementary CC-point set to determine which of their CLs are interference-free and have satisfactory fitting error. These CLs, together with the existing CLs, are then used to re-construct the dual quaternion curve.

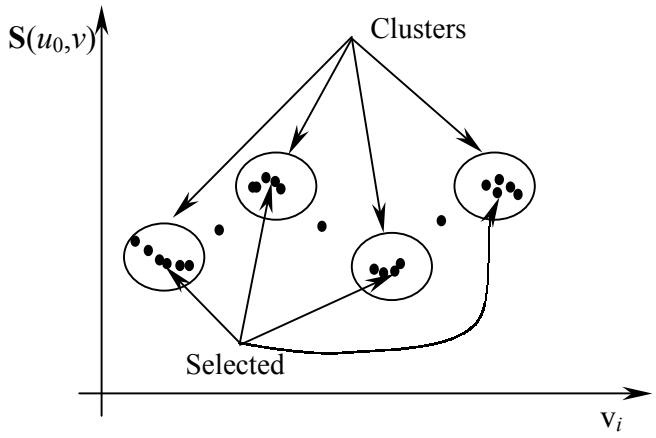


Fig. 3.13 Points Reducing in the supplementary CC point set

We have already recorded points on $\mathbf{S}(u_0,v)$ corresponding to the problematic CLs. However, the number of these points is very large and these points may be clustered at specified locations (see Fig. 3.13). Moreover, it is possible that a point repeats to occur in the supplementary CC-point set. This can result in heavy computational load and inaccuracy for the construction of rational Bézier motion curve. In order to solve this problem, the number of the recorded points on $\mathbf{S}(u_0,v)$ needs to be reduced. In our case, we firstly find the cluster that the points are located in, and then find a point which is nearest to the centre of the cluster to represent all the points in this cluster, and delete all other points in the cluster. In this way, our modification process becomes more efficient and effective.

After the reduction, the points left in the supplementary CC-point set are considered as new CC points, and the corresponding cutter postures (CLs) at these positions are then obtained based on gouging and collision avoidance described in section 3.3.2. The newly generated CLs are converted to dual quaternion representation. Thus the control quaternions are modified and a new quaternion motion curve is generated from these modified quaternions. The swept surface of cutter undergoing the new quaternion motion will have more contact points with the

underlying curve. It is therefore expected that the new tool path will have less problems in terms of fitting error, gouging and collision. Fitness, gouging and collision checking will be carried out on the new tool path. This checking-modification-checking process is repeated until fitting test, gouging and collision test are satisfactory. However, if the number of iteration is larger than a pre-specified threshold value, the checking-modification-checking process will stop and give user an alert message.

3.3.5 The Summary of the whole algorithm

The procedure of generating a single gouging-free and collision-free tool path using the piecewise rational Bézier motion of the cutter is summarized as follows:

(1) Get an iso-parametric curve $\mathbf{S}(u_0,v)$ from the designed surface $\mathbf{S}(u,v)$ by fixing $u=u_0$.

(2) Obtain a set of cutter contact points $\mathbf{S}(u_0,v_j)$ on the surface curve $\mathbf{S}(u_0,v)$, where $j=0,..n,$ and find the corresponding set of centres of the cutter bottom circle CL_j using the method mentioned in section 3.3.1 and 3.3.2 respectively.

(3) For each j, convert the cutter location CL_j into the dual quaternion representation using Eq. (3.2).

(4) Find the control quaternions and construct the rational Bézier quaternion curve $\hat{\mathbf{q}}(t)$ using the method mentioned in section 3.3.3.

(5) Get the swept surface $\mathbf{P}(s,t)$ of the cutter undergoing rational Bézier quaternion motion using Eq. (3.4)

(6) Fitness checking and interference checking using the method mentioned in section 3.3.4. If the fitness meets the surface tolerance and the interference does not exist, stop. Otherwise, modify the control quaternions and go back to (4).

CHAPTER 4

MULTI TOOL PATHS GENERATON

In 5-axis machining, the generated tool path must be gouging-free and collision-free, and scallop height between the adjacent tool paths must be controlled in a pre-defined tolerance. In chapter 3, we have already presented an algorithm to obtain a single gouging-free and collision-free iso-parameter tool path, the main focus in this chapter is to generate the adjacent tool path such that the scallop height between two neighboring tool paths is within the allowable tolerance.

4.1 Scallop Height and Effective Cutting Shape

Machining error occurs when an excess of material is left between adjacent overlapping cutter paths, as shown in Fig. 4.1. The volume of unremoved material is referred to as a scallop. It can be exactly described by subtracting the designed surface from the machined surface generated by the cutter. Scallop curve is a ridge protrusion formed at the intersection of the swept surface of the adjacent cutting paths. The distance of the scallop curve to the designed surface represents a local maximum of the unremoved materials extending above the designed surface. This distance is referred as scallop height. Controlling scallop height is a significant factor in 5-axis NC machining since scallop height influences the manufacturing efficiency and finish surface quality. The machined surface with small scallop height significantly reduces the manual grinding and smoothing required by the specified surface roughness design.

Some researchers introduced the local geometry of the surface and the cutter to estimate the scallop height. Choi (1993) presented a method that evaluates the scallop height by finding the intersection between two curves generated by projecting cutter bottom onto the cutting planes at the adjacent CC points on the current and the next tool paths. Lee (1996b) developed an error analysis method for 5-axis machining which applied differential geometry technique to evaluate the scallop height between adjacent cutter locations. Generally, in these algorithms, effective cutting shape is applied to evaluate the scallop height. However, their approximation of effective cutting shape is not highly accurate.

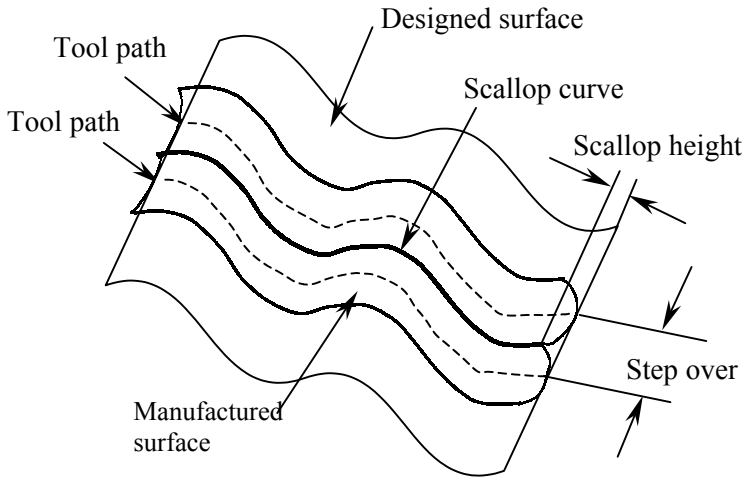
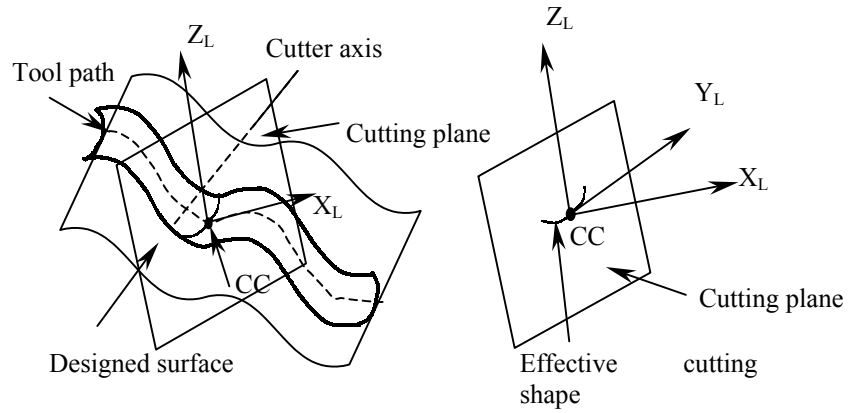


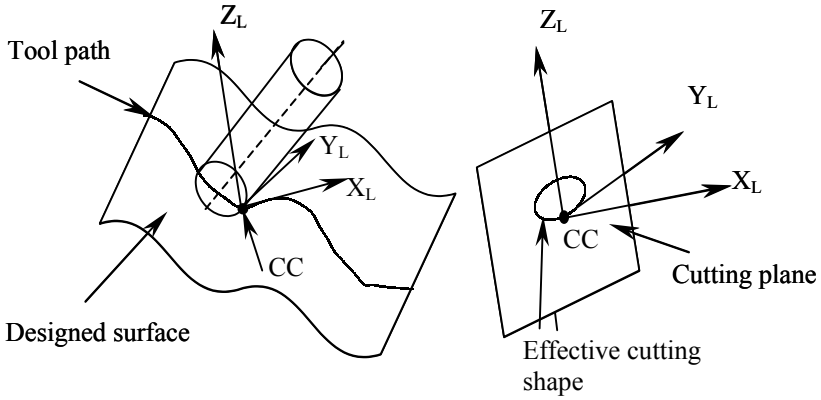
Fig. 4.1 The illustration of scallop height

The effective cutting shape is defined as the intersection between the cutting plane at the CC point under consideration and the swept surface formed by sweeping cutter bottom along the tool path (see Fig. 4.2a). However, in most of the researchers' reports, the effective cutting shape is approximated by projecting the cutter bottom to the cutting plane at the CC point under consideration (see Fig. 4.2b). From Sarma's work (2000), we can see that this approximation can be significantly different from the accurate effective cutting shape. The scallop height calculated using this approximation is consequently inaccurate. Since the scallop height determines the

finish surface quality, we need to seek a more accurate method to evaluate it. In our approach, since the swept surface generated by the rational Bézier motion of the cutter bottom circle can be determined analytically, the effective cutting shape can be represented accurately. Hence, the corresponding scallop height can also be calculated accurately.



(a)



(b)

Fig. 4.2 Effective cutting shape

(a) The exact definition of effective cutting shape

(b) Traditional method to estimate the effective cutting shape

4.2 Evaluating the Effective Cutting Shape

In this section, our aim is to obtain the analytical expression of the effective cutting shape by intersecting the cutting plane at the CC point under consideration and the swept surface generated by the rational Bézier motion of the cutter bottom circle.

We firstly represent the cutting plane mathematically. For the iso-parameter tool path, the cutting plane at CC point $\mathbf{C}_i$ is perpendicular to the cutting direction $\mathbf{S}_v(u_0,,v_i)$ and passes through $\mathbf{C}_i$. Denote the cutting plane as **CP**, we can obtain the homogeneous representation of the cutting plane as:

$$\mathbf{CP}=(\mathbf{n}, -d) \tag{4.1}$$

where $\mathbf{n}=(n_1, n_2, n_3)=\dfrac{\mathbf{S}_v(u_0,v_i)}{\|\mathbf{S}_v(u_0,v_i)\|}$ is the unit normal vector of the plane **CP**; $d= -(n_1c_1+ n_2c_2+ n_3c_3)$ is the distance from the origin to the plane **CP**; $(c_1, c_2\ c_3)$ is the coordinate of $\mathbf{C}_i$.

The analytic expression of swept surface generated by the rational cubic Bézier motion of the cutter bottom circle is shown in Eq. (3.4). Therefore, the intersection of **CP** and the swept surface can be expressed as:

$$\mathrm{G}(s,t)= \mathbf{P}(s,t)\cdot\mathbf{CP} = \sum_{k=0}^{6}\sum_{i=0}^{2} B_k^6(t)B_i^2(s)[H_k]\mathbf{P}_i \cdot\mathbf{CP} =0 \tag{4.2}$$

This yields a curve on the swept surface, which is the effective cutting shape. Eq. (4.2) is an implicit function of s and t. To get the explicit function, we can express the Eq. (4.2) as:

$$\sum_{i=0}^{2} B_i^2(s)g_i = 0 \tag{4.3}$$

where $g_i = \sum_{k=0}^{6} B_k^6(t)[H_k]\mathbf{P}_i \cdot\mathbf{CP}$. Expend Eq. (4.3), we can obtain:

$$\mathrm{G}(s,t)=(1-s)^2g_0+2s(1-s)g_1+s^2g_2$$

$$=(g_0+g_2-2g_1)s^2+2(g_1-g_0)s+g_0=0$$

Solving the above quadratic equation, we can obtain the explicit function of parameter *s* and *t* on the effective cutting shape as:

$$s(t)=\frac{g_0-g_1\pm\sqrt{g_1^2-g_0g_2}}{g_0+g_2-2g_1} \tag{4.4}$$

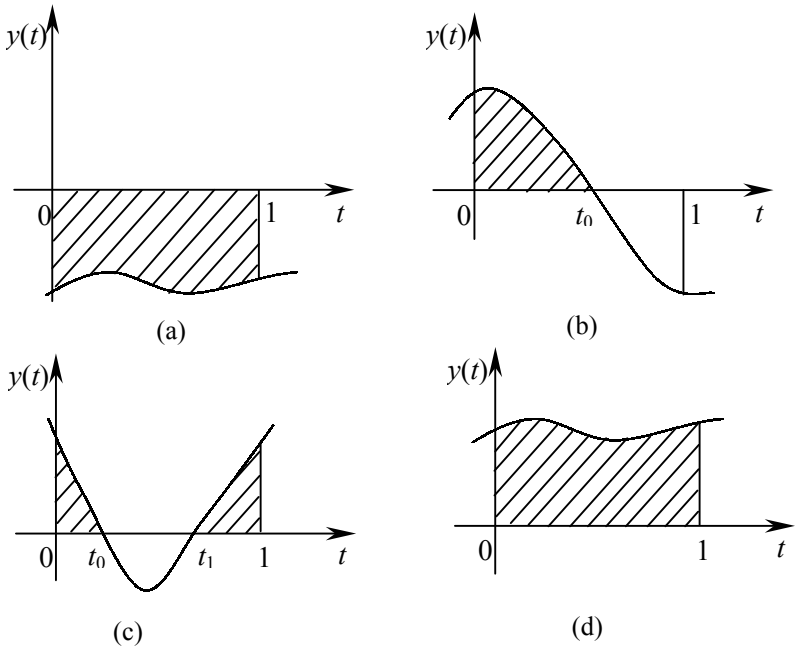


Fig. 4.3 The geometry of function $y(t)$

If the intersection between the cutting plane **CP** and the swept surface **P**(*s*,*t*) exists, Eq. (4.4) should satisfy the following three conditions:

$$\begin{cases} 0\le t\le 1 \\ 0\le s\le 1 \\ g_1^2-g_0g_2\ge 0 \end{cases} \tag{4.5}$$

From Eq. (4.3), we can know that $y(t)=g_1^2-g_0g_2$ is the polynomial function of *t* with degree 4*n*. Thus, there are 4*n* solutions to the function $y(t)$=0. To find the intervals of *t* that yield $y(t)$>0, we need to investigate the geometry of the function $y(t)$ when *t* varies from 0 and 1. There are four cases for function $y(t)$ as follows:

(1) $y(t)<0$ for any t within 0 and 1, as shown in Fig. 4.3a;

(2) One solution to the function $y(t)=0$ is within 0 and 1, as shown in Fig. 4.3b;

(3) Two or more solutions to the function $y(t)=0$ are within 0 and 1, as shown in Fig. 4.3c;

(4) $y(t)\geq 0$ for any t in the interval of 0 and 1, as shown in Fig. 4.3d.

If case (1) occurs, there is no solution of parameter s according to Eq. (4.4). As a result, the intersection between the swept surface and the cutting plane at $\mathbf{C}_i$ does not exist. If cases (2)-(4) occur, we can always find some t, where $0\leq t\leq 1$, that yield $y(t)\geq 0$. Therefore, the solution s to Eq. (4.4) exists.

Given a function $y(t)$, if any cases in (2)-(4) occurs, we can obtain the ranges of t that yield $y(t)\geq 0$ and $t\in[0,1]$. Combine these ranges of t, we can get:

$$\mathbf{R}=\{[t_1, t_2], \ldots [t_i, t_{i+1}], \ldots [t_{m-1}, t_m]\}$$

Where $t_1<\ldots<t_i \ldots<t_m$. Any ranges of t in **R** can yield the existence of solution s of the Eq. (4.4). However, according to Eq. (4.5), if the intersection between the swept surface and the cutting plane exists, we still need to make sure the parameter s lies within the range [0,1]. Since the feasible range **R** of t for the solution of Eq. (4.4) has been obtained, we can further use **R** to find the corresponding range of parameter s. Solve the following equations:

$$\frac{ds(t)}{dt}=0, s(t)=0, s(t)=1, \text{ where } 0\leq t\leq 1 \tag{4.6}$$

We can obtain a set of solutions $t_j^{'}$ within the range [0,1] of Eq. (4.6) , as shown Fig. 4.4. Subdivide the range [0,1] into smaller ones, and take each of the $t_j^{'}$ as the subdivision point, we obtain a set $\mathbf{R}_1$:

$$\mathbf{R}_1=\{[0, t_1^{'}], \ldots [t_j^{'}, t_{j+1}^{'}], \ldots [t_n^{'}, 1]\};$$

Where $t_1^{'}<\ldots<t_j^{'}\ \ldots<t_n^{'}$. Intersect the range **R** and the range $\mathbf{R}_1$, and we can get a new subdivision range $\mathbf{R}_2$ as:

$$\mathbf{R}_2=\{[\bar{t}_1,\ \bar{t}_2],\ \ldots\ [\bar{t}_i,\bar{t}_{i+1}],\ \ldots\ [\bar{t}_{k-1},\bar{t}_k]\}$$

Where $\bar{t}_1<\ldots<\bar{t}_i\ \ldots<\bar{t}_k$.

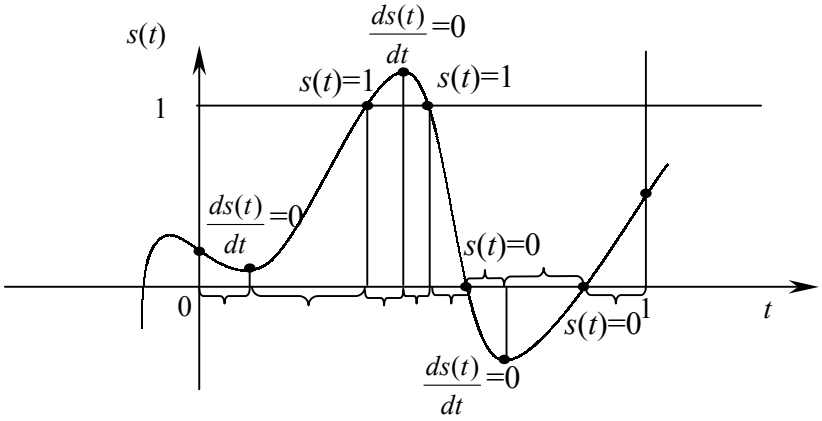


Fig. 4.4 Finding the range R_1

In each of the subdivision range $[\bar{t}_i,\bar{t}_{i+1}]$, $s(t)$ solely increases or decreases with the parameter $\bar{t}$ increases from $\bar{t}_i$ to $\bar{t}_{i+1}$, as shown in Fig. 4.4. Thus, if $0\le s(\bar{t}_i)\le 1$ and $0\le s(\bar{t}_{i+1})\le 1$, the intersection between the swept surface and the cutting plane at $\mathbf{C}_i$, i.e. effective cutting shape, exists. We can then get the feasible range of s as $[\min(s(\bar{t}_i),\ s(\bar{t}_{i+1})),\ \max(s(\bar{t}_i),\ s(\bar{t}_{i+1}))]$. Unite all of the ranges of s that yield $0\le s(\bar{t}_i)\le 1$ and $0\le s(\bar{t}_{i+1})\le 1$, we can get a new range $\mathbf{R}_s$ as:

$$\mathbf{R}_s=\{[s_1,\ s_2],\ \ldots\ [s_i,\ s_{i+1}],\ \ldots\ [s_{h-1},\ s_h]\}$$

Where $s_1<\ldots<s_i\ \ldots<s_h$. and the corresponding range $\mathbf{R}_t$ that yields these range of s as:

$$\mathbf{R}_t=\{[t_1,\ t_2],\ \ldots\ [t_i,\ t_{i+1}],\ \ldots\ \ [t_{h-1},\ t_h]\};$$

If $\mathbf{R}_s$ and $\mathbf{R}_t$ are not empty, the intersection between the swept surface and the cutting plane at $\mathbf{C}_i$, i.e. effective cutting shape, exists. Then $\mathbf{P}(s(t),t)$ is the exact representation of the effective cutting shape at the $\mathbf{C}_i$. Otherwise, there is no intersection.

4.3 Constructing the Adjacent Tool Path

Based on surface curve $\mathbf{S}(u_i,v)$, we can use the algorithm in chapter 3 to obtain a iso-parameter tool path and the swept surface generated by the rational Bézier motion of cutter bottom circle along this tool path. Similarly, the swept surface of the motion of cutter bottom circle and the candidate next iso-parameter tool path can be obtained based on $\mathbf{S}(u_i+\Delta u,v)$, where Δu is the side step size. The intersection curve between these two swept surfaces yields the scallop curve $h(v)$. For each point, $h(v_i)$, on the scallop curve $h(v)$, if the distance between $h(v_i)$ and the surface $\mathbf{S}(u,v)$ is within the pre-defined tolerance δ, this candidate next tool path can be said satisfactory. Otherwise, the step size Δu for the candidate next tool path needs to be modified. With this modified Δu, a new candidate next tool path is generated and subsequently, the scallop height is re-checked. This process goes on until scallop height is within the pre-defined tolerance. Therefore, a suitable next tool path is found. Repeatly applying the above process, we can obtain the entire tool paths on the designed surface.

The key issue in the above process is to find the intersection curve between the adjacent swept surfaces. This is a surface/surface intersection problem (SSI), which is always encountered in computer graphics. Many researchers tried different methods to solve this problem. Among these methods, the recursive subdivision method and the incremental tracing method are commonly used (Nadim et al., 1990). However, the disadvantage of these methods is the extremely heavy computational load. In computer graphic fields, most of the complex curve and surface intersection problems can also be solved by the numerical method. We adopt the numerical method to evaluate the scallop height. First, according to the curvature of the surface curve $\mathbf{S}(u_0,v)$, we discretise the current tool path to a set of CC points. Second, for each of these CC points, find the intersection point h_i between $\mathbf{P}_1(s(t),t)$ and $\mathbf{P}_2(s(t),t)$, where $\mathbf{P}_1(s(t),t)$

and $\mathbf{P}_2(s(t),t)$, are the intersection curves between the cutting plane at this CC point and the two swept surfaces generated by the motion of cutter bottom on adjacent tool paths respectively. h_i is on scallop height curve $h(v)$, as shown in Fig. 4.5. All the intersection points $\{h_i\}$ are used to represent the scallop height curve $h(v)$.

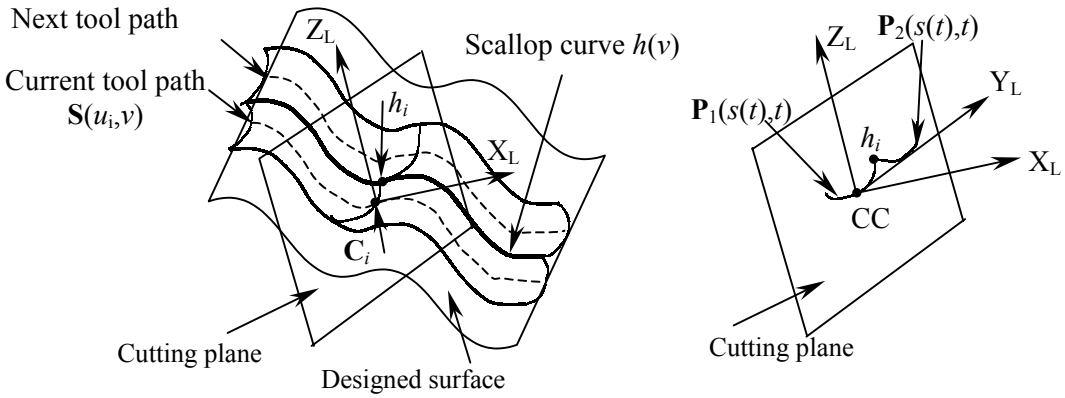


Fig. 4.5 Calculation of scallop height

Based on the above descriptions, the process of generating the adjacent tool path can be divided into the following steps: (1) estimate the initial step over L_i and the step size Δu for the candidate next tool path, (2) generate the collision-free and gouging-free candidate next tool path, (3) discretise the surface curve $\mathbf{S}(u_0,v)$ into a set of surface points $\{\mathbf{C}_i\}$ according to its curvature, (4) for each of these surface points $\mathbf{C}_i$, find the intersections curves $\mathbf{P}_1(s(t),t)$ and $\mathbf{P}_2(s(t),t)$ between the cutting plane at $\mathbf{C}_i$ and the two neighboring swept surfaces, (5) obtain the intersection point h_i of $\mathbf{P}_1(s(t),t)$ and $\mathbf{P}_2(s(t),t)$, and (6) find the distance between h_i and the designed surface $\mathbf{S}(u,v)$.

As described in chapter 3, the swept surface generated by the piecewise rational Bézier motion of the cutter bottom circle is constituted by several Bézier surface patches. Because of this, some tough problems will be encountered, such as determination of the existence of intersection curve between the swept surface patches

and the cutting plane, and determination of positions that contribute to the generation of scallop height. We then identify and solve these problems in the following section.

4.3.1 Generating the candidate next tool path

In order to generate the candidate next tool path, the initial step over distance L and the step size Δu need to be determined first. The step over distance L is very important to generate the next iso-parameter tool path. If L is too small, the result scallop height is small and good finish surface quality can be achieved, but the number of tool paths is very large and thus the machining efficiency is very low, and vice versa for a large L. Lin and Koren (1996) investigated this issue and estimated the initial step over L_i for a specified CC point $\mathbf{C}_i$ as (see Fig. 4.6):

$$L_i = \begin{cases} \sqrt{\dfrac{8 r_s r_e h_{\text{tol}}}{r_s + r_e \cdot sign}} & \text{(concave or convex designed surface)} \\ \sqrt{8 r_s r_e h_{\text{tol}}} & \text{(plane designed surface)} \end{cases} \tag{4.7}$$

Where r_s is the radius of surface curvature in the cutting plane at $\mathbf{C}_i$. r_e is the radius of effective cutting curvature on the same cutting plane. h_{tol} is the scallop height tolerance. If a convex surface is applied, *sign* is equal to 1; if a concave surface is applied, *sign* is equal to –1. Note that r_e is calculated according to the effective cutting shape generated in section 4.1.

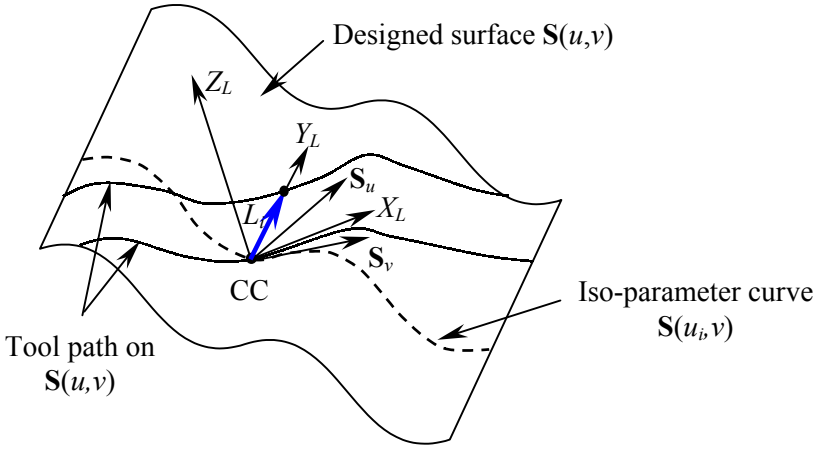


Fig. 4.6 Calculating the step over and the step size

After obtaining the step over distance L_i at $\mathbf{C}_i$, we need to calculate the step size Δu_i. As shown in Fig. 4.6, the step size Δu_i is determined using the following equation:

$$\Delta u_i(\mathbf{S}_u \cdot \mathbf{Y}_L) + \Delta v_i(\mathbf{S}_v \cdot \mathbf{Y}_L) = L_i \text{ and } \Delta v_i = 0$$

Therefore, we can get the step size Δu_i as:

$$\Delta u_i = L_i / (\mathbf{S}_u \cdot \mathbf{X}_L) \tag{4.8}$$

where $\mathbf{Y}_L$ is the cross product of cutting direction $\mathbf{X}_L$ and the surface normal $\mathbf{Z}_L$ at $\mathbf{C}_i$; $\mathbf{S}_v = \partial \mathbf{S}(u,v)/\partial v$ and $\mathbf{S}_u = \partial \mathbf{S}(u,v)/\partial u$.

For each of the CC points on $\mathbf{S}(u_0,v_i)$, the corresponding step over distance L_i and the step size Δu_i can be calculated according to Eq. (4.7) and Eq. (4.8). The smallest step size in the set $\{\Delta u_i\}$ can be assigned as the step size Δu of the tool path, and the candidate next tool path is generated based on the surface curve $\mathbf{S}(u_0+\Delta u,v)$.

Given a surface curve $\mathbf{S}(u_0+\Delta u,v)$, the candidate next tool path without gouging and collision can be generated using the method presented in chapter 3.

4.3.2 Discretizing surface curve $\mathbf{S}(u_0,v)$

We then discretize the surface curve $\mathbf{S}(u_0,v)$ into a set of surface points , so that the scallop height can be estimated at each of these positions. The key issue here is to find the distribution of these surface points. Since the distribution of the surface points greatly depends on the geometry of $\mathbf{S}(u_0,v)$, we can discrete $\mathbf{S}(u_0,v)$ according to its curvature. We face the same problem as that in one tool path generation, where the sample points on the surface curve $\mathbf{S}(u_0,v)$ need to be found so that the rational Bézier motion of the cutter bottom circle could be constructed, as mentioned chapter 3. Similar method is adopted here, but the fitting tolerance τ is chosen more strictly. The fitting tolerance τ is 0.5 to 1 times of surface error.

4.3.3 Finding intersection curve between the cutting plane and the swept surfaces

In our approach, the swept surface is generated by the piecewise rational Bézier motion of the cutter bottom circle, and therefore it is constituted of a set of rational Bézier surface patches, as described earlier. Consequently, finding the intersection between the swept surface and the cutting plane at the CC point is equivalent to finding the intersection between all of the swept surface patches and the cutting plane. However, the problem becomes a little tougher because of the existence of these swept surface patches. First, for each of the swept surface patches, the existence of intersection curve with the cutting plane needs to be checked. Second, sometimes, the intersection of the swept surface with the cutting plane may result in two or more intersection positions, and we need determine which position contributes to the generation of scallop height. In this section, our main focus is to obtain the intersection curve between the swept surface and the cutting plane and solve the above problems.

Since the swept surface is the combination of the rational Bézier surface patches, the intersection curve between the swept surface and the cutting plane are assigned as the combination of the intersection curves between the swept surface patchs and the cutting plane. In section 4.1, we have presented the method to check the existence of intersection between the rational Bézier swept surface patches and the cutting plane at the CC point. There, we assert that, if $\mathbf{R}_s$ and $\mathbf{R}_t$ are not empty, the intersection exists, and the corresponding ranges of parameter s and t can also be obtained. Then $\mathbf{P}(s(t),t)$ is the intersection curve at the CC point. For each of the swept surface patches, this checking process must be performed, and the intersection curve between these swept surface patches and the cutting plane at the CC point $\mathbf{P}_{ai}(s_i(t_i),t_i)$, where a indicates the ath tool path and i represents the ith swept surface patch, can be

obtained. Therefore, the combination of the intersection curves can then be expressed as follows:

$$\mathbf{P}_a(s(t),t)= \mathbf{P}_{a1}(s_1(t_1),t_1)\cup \mathbf{P}_{a2}(s_2(t_2),t_2)\cup \ldots \mathbf{P}_{ai}(s_i(t_i),t_i) \cup \ldots \mathbf{P}_{ak}(s_k(t_k),t_k) \quad (4.9)$$

Where s_i and t_i are the feasible parameters that contribute to the intersection curves, and k is the number of swept surface patches that have intersection with the cutting plane at the CC point.

Thus, $\mathbf{P}_a(s(t),t)$ is the intersection curve between the swept surface and the cutting plane. For the example, in Fig. 4.7, there are two swept surface patches $\mathbf{P}_{a1}(s_1,t_1)$ and $\mathbf{P}_{a2}(s_2,t_2)$ that intersect with the cutting plane **CP** at **CC**, therefore, the intersection curve between the whole swept surface and **CP** constitutes of two intersection curves, $\mathbf{P}_{a1}(s_1(t_1),t_1)$ and $\mathbf{P}_{a2}(s_2(t_2),t_2)$.

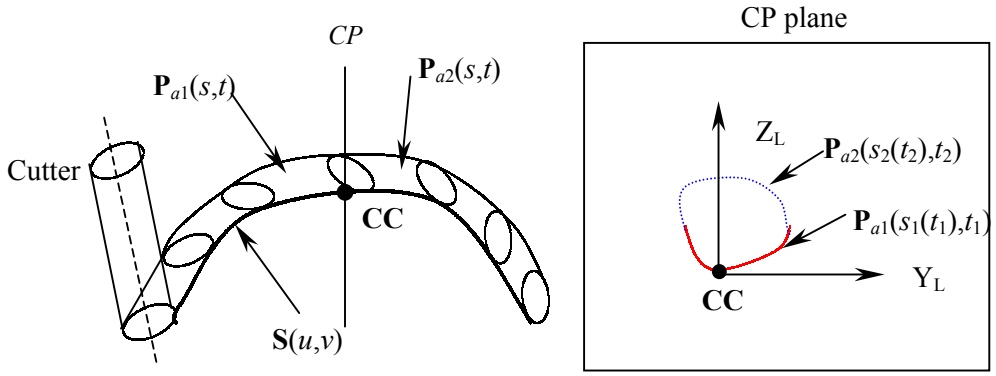


Fig. 4.7 Intersection curve between the swept surface and the cutting plane

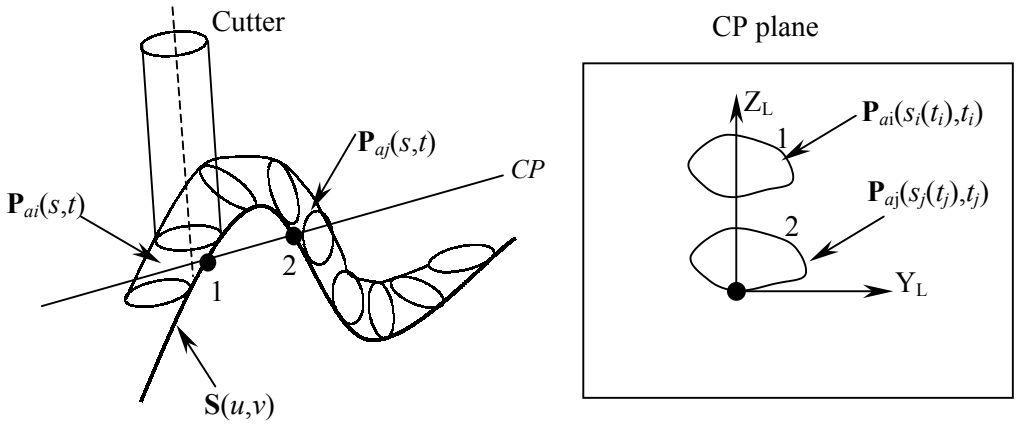


Fig. 4.8 Location of the intersection positions

However, in some applications, the intersection between the swept surface on one tool path and the cutting plane at the specified CC point may result in two or more intersection positions, as shown in Fig. 4.8. According to Eq. (4.9), since the combined intersection curve $\mathbf{P}_a(s(t),t)$ includes the intersection curves at these intersection positions, problems may occur when we calculate the scallop height. For example, when we calculate the scallop height at the cutting plane at CC point 2, there are two intersection curves $\mathbf{P}_{ai}(s_i(t_i),t_i)$ and $\mathbf{P}_{aj}(s_j(t_j),t_j)$ between the cutting plane **CP** and the swept surface in Fig. 4.8. Obviously, $\mathbf{P}_{aj}(s_j(t_j),t_j)$ is what we need to calculate the scallop height. But if the result scallop height is calculated according to $\mathbf{P}_{ai}(s_i(t_i),t_i)$, wrong decision (recalculating the step size or continuing to next checking point) would be made, hence it will lead to inefficiency and inaccuracy. The key issue to solve this problem is to find the proper intersection curve at the specified position.

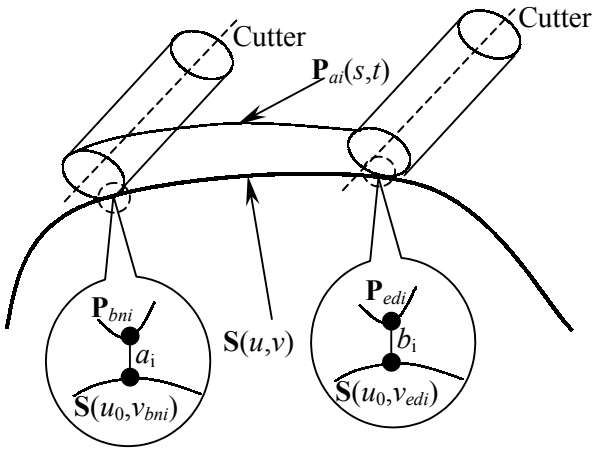


Fig. 4.9 Finding the range of v for a swept surface patch

In section 4.3.2, the surface curve $\mathbf{S}(u_0,v)$ is divided into a set of the surface points according to its curvature. This set of points are corresponding to a set of specified parameter v_i of the designed surface curve $\mathbf{S}(u_0,v)$. Since the cutting planes that are used to intersect with the swept surface are determined according to the geometry of the surface at each of these surface points, each of the cutting planes is

also corresponding to a specified parameter v_i. Since all the swept surface patches contribute to generating the tool path, each of these swept surface patches is also corresponding to a range of v. The calculation of the range of v is based on the calculation of minimal distances between the beginning and ending parts of the swept surface patches and the designed surface curve $\mathbf{S}(u_0,v)$. As shown in Fig. 4.9, the corresponding surface point that yields the minimal distance between the beginning part of the swept surface patches $\mathbf{P}_{ai}(s,t)$ and the designed surface curve $\mathbf{S}(u_0,v)$ is $\mathbf{S}(u_0,v_{bni})$, and the corresponding surface point that yields the minimal distance between the ending part of $\mathbf{P}_{ai}(s,t)$ and $\mathbf{S}(u_0,v)$ is $\mathbf{S}(u_0, v_{edi})$. Therefore the range $[v_{bni}, v_{edi}]$ is the corresponding range of v for a swept surface patch. After finding the range of v for each of the swept surface patch that intersects with the cutting plane, we can check whether the parameter v_i corresponding to the cutting plane is within this range. If so, the intersection curve of this patch with the cutting plane can be thought as part of effective cutting shape and is chosen to calculate the scallop height. Otherwise, we can discard this intersection curve. The detail algorithm can be described as follows:

From the first swept surface patch $\mathbf{P}_{ai}(s,t)$, where i=0, on a single tool path:

(1) Find the existence of the intersection curve between $\mathbf{P}_{ai}(s,t)$ and the cutting plane **CP** at the CC point under consideration.

(2) If the intersection does not exist, continue to the next swept surface patch and go back to step (1), until all the swept surface patchs are calculated. Otherwise, obtain the range of parameter v for this swept surface patch.

   (a) Obtain the corresponding surface point $\mathbf{S}(u_0,v_{bni})$ that yields the minimal distance between $\mathbf{S}(u_0,v)$ and the beginning part of $\mathbf{P}_{ai}(s,t)$.

   (b) Obtain the corresponding surface point $\mathbf{S}(u_0, v_{edi})$ that yields the minimal distance between $\mathbf{S}(u_0,v)$ and the ending part of $\mathbf{P}_{ai}(s,t)$.

(c) If the parameter v_j of CC point $\mathbf{S}(u_0,v_j)$ is within $[v_{bni}, v_{edi}]$, the intersection curve is on the effective cutting shape and is used to calculate the scallop height. Otherwise, go back to step (1), until all the swept surface patches are calculated.

4.3.4 Obtaining the intersection point between the cutting plane and the swept surfaces on the neighboring tool paths

From the definition of scallop height, we can know that the two swept surfaces on the current and its neighboring tool paths are needed to calculate the scallop height. The intersections between the cutting plane at the CC point and the two swept surfaces on the adjacent tool paths yield two intersection curves. According to section 4.3.3, the combined intersection curve $\mathbf{P}_r(s(t),t)$ between the cutting plane and the swept surface on the current tool path can be obtained, and similarly the combined intersection curve $\mathbf{P}_n(s(t),t)$ between the same cutting plane and the swept surface on next tool path can also be obtained. The designed surface and the intersection point of $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ can be used to estimate the scallop height. In this section, we focus on finding the intersection point between $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$.

Generally, $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ are ellipse-like shapes, as shown in Fig. 4.10, and thus there are two intersection points between these two curves. Obviously, the lower point (nearer to the designed surface) is what we need to calculate the scallop height. Hence, when calculating the intersection points between $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$, the lower intersection point need to be identified. We can then use the analytical expression of $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ according to Eq.(4.9) and Eq. (3.4) to find the intersection points. However, $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ are piecewise functions and very complicated, therefore the calculation of the intersection points is very difficult. In order to solve this problem, we can use two polygons to approximate the intersection

curves and find the lower intersection point between these two polygons (see Fig. 4.10).

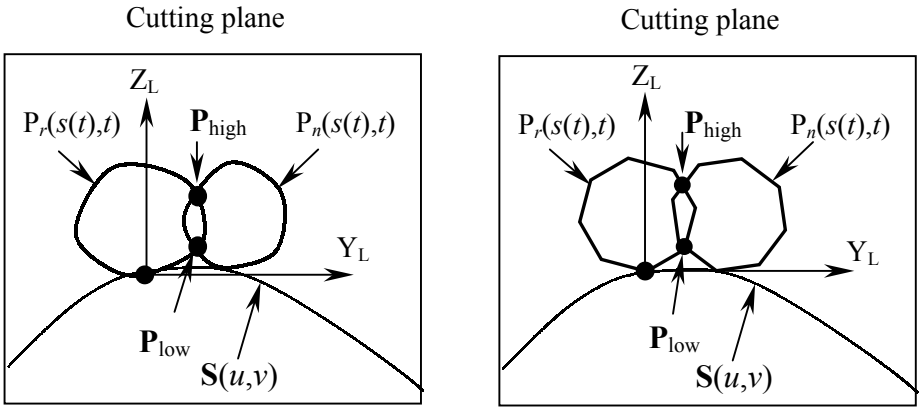


Fig. 4.10 Polygonization of the effective cutting shape

In our application, the method to polygonize the curve $\mathbf{P}(s(t),t)$ is similar to the method to discrete the surface curve $\mathbf{S}(u_0, v)$ according to the pre-defined tolerance mentioned earlier. First, choose a point on $\mathbf{P}(s(t),t)$ as the beginning point of the polygon. Second, approximate the vicinity of this point on $\mathbf{P}(s(t),t)$ as circular arc and find the next point on the polygon according to a pre-defined tolerance τ using the method mentioned in section 3.3.1. Repeat the second step to calculate the next point on the polygon until reaching the beginning point of $\mathbf{P}(s(t),t)$.

The intersection points between two curves $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ can then be approximated as the intersection points between the two polygons. Finding the intersection between the two polygons is not a tough problem and it has been encountered in most of the applications in computer science. After calculation, we obtain two intersection points $\mathbf{P}_{high}$ and $\mathbf{P}_{low.}$ We can then calculate the distances between each of the intersection points and the Y_L axis in cutting plane. The intersection point that yields to the smaller distance is the lower intersection point and this point is what we need.

However, sometimes, the intersection points between $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ when the cutting plane is at the beginning and ending part of surface curve are different with the regular ones. Generally, there exist the following cases:

(1) No intersection points exist between $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$.

(2) Only one intersection point exists between $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$.

If there is no intersection points between $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$, it means that the two neighboring swept surfaces of cutter motion have no intersection at this discrete testing position. Therefore, the scallop height at this position does not exist, and we can ignore the calculation of scallop height and continue to check the next position. If only one intersection point exists, we need to investigate the position of this intersection point carefully. If the intersection point locates in the upper intersection curve between the neighboring swept surfaces, the calculation of scallop height using this point is not necessary. However, if the intersection point locates in the lower intersection curve between the neighboring swept surfaces, this point should be used to execute the calculation of the scallop height. Generally, if the distances between this intersection point and the Y_L axis in cutting plane is larger than 0.2r, where r is the cutter radius, we can assume that this intersection point is on the upper intersection curve between the neighboring swept surface. Otherwise, this intersection point is in the lower intersection curve and the calculation of scallop height based on this intersection point is needed.

4.3.5 Calculation of scallop height

As we mentioned earlier, the scallop height is the minimal distance between the lower intersection point of the curve $\mathbf{P}_r(s(t),t)$ and $\mathbf{P}_n(s(t),t)$ and the designed surface $\mathbf{S}(u,v)$. This problem is formulated as:

Minimise $distance[\mathbf{P}_{low}, \mathbf{S}(u, v)] = f(u, v)$

Subject to: $0 \leq u \leq 1$ and $0 \leq v \leq 1$

Different search algorithms are available to solve this optimisation problem. In our approach, we apply Downhill Simplex algorithm to find the minimum distance. The advantages of Downhill Simplex algorithm have been described in chapter 3.

If the result scallop height is within the pre-defined tolerance, we can proceed to next CC point to check the scallop height. Otherwise, the candidate next tool path cannot meet the requirement and the step size Δu needs to be reduced to generate a new candidate next tool path. The whole process is revisited until a suitable next tool path is found.

CHAPTER 5

SOFTWARE SIMULATION RESULTS

In this thesis, our tasks are to solve the problems in 5-axis machining of the sculptured surface and exploit the full potential flexibility of 5-axis NC machining. In the previous chapters, we have presented an algorithm to find the gouging-free and collision-free iso-parameter tool path using the piecewise rational Bézier motion of the cutter, and keep the scallop height between the neighboring tool paths within the pre-defined tolerance. In order to show the advantages of this algorithm over traditional algorithms, in this chapter, we present the result of software simulation by employing the algorithm developed aforementioned. The proposed method has been implemented on PC using VC++ and OpenGL.

5.1 The Designed Surface

We choose several surfaces such as concave, convex surface with irregular curvatures as the surfaces to be machined by 5-axis NC machine tool.

The first designed surface shown in Fig. 5.1a is a concave Bézier surface patch described by $\mathbf{S}(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathrm{B}_i^3(u)\mathrm{B}_j^3(v)\mathbf{P}_{ij}$ where the control points $\mathbf{P}_{ij}$ are given as follows:

$$\mathbf{P}_{ij} = \begin{bmatrix} (-1,0,0) & (0,0,-2) & (1,0,2) & (2,0,0) \\ (-1,1,-2) & (0,1,-4) & (1,1,-4) & (2,1,-2) \\ (-1,2,-2) & (0,2,-4) & (1,2,-4) & (2,2,-2) \\ (-1,3,0) & (0,3,-2) & (1,3,-2) & (2,3,0) \end{bmatrix}$$

The second designed surface shown in Fig. 5.1b is a convex Bézier surface patch describe by $\mathbf{S}(u,v)=\sum_{i=0}^{2}\sum_{j=0}^{2}\mathrm{B}_i^2(u)\mathrm{B}_j^2(v)\mathbf{P}_{ij}$ where $\mathbf{P}_{ij}$ are given as follows:

$$\mathbf{P}_{ij}=\begin{bmatrix}(0,0,0) & (1,0,3) & (2,0,1)\\ (0,2,0) & (1,2,3) & (2,2,1)\\ (0,4,0) & (1,4,3) & (2,4,1)\end{bmatrix}$$

The third designed surface shown in Fig. 5.1c is a Bézier surface patch that is a wave-like surface. It can be described by $\mathbf{S}(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{4}\mathrm{B}_i^3(u)\mathrm{B}_j^4(v)\mathbf{P}_{ij}$, where $\mathbf{P}_{ij}$ are given as follows:

$$\mathbf{P}_{ij}=\begin{bmatrix}(1,0,0) & (2,0,4) & (2.5,0,0) & (3,0,-3) & (4.3,0,0)\\ (1,2,2) & (2,2,6) & (2.5,2,2) & (3,2,-1) & (4.3,2,0)\\ (1,3,1) & (2,3,5) & (2.5,3,2) & (3,3,-2) & (4.3,3,0)\\ (1,4,0) & (2,4,4) & (2.5,4,0) & (3,4,-3) & (4.3,4,0)\end{bmatrix}$$

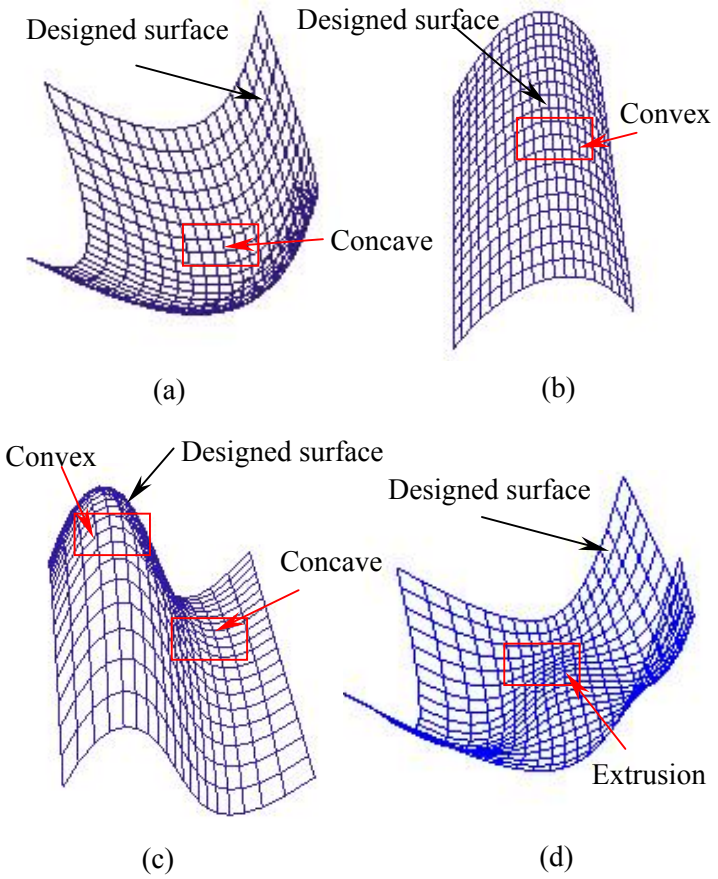


Fig. 5.1 The examples of designed surfaces to be machined

The fourth designed surface shown in Fig. 5.1d is a Bézier surface patch that is concave in general with an extrusion in the centre. This surface can be described by $\mathbf{S}(u,v) = \sum_{i=0}^{5} \sum_{j=0}^{4} B_i^5(u) B_j^4(v) \mathbf{P}_{ij}$, where $\mathbf{P}_{ij}$ are given as follows:

$$\mathbf{P}_{ij} = \begin{bmatrix} (-2,0,0) & (-0.5,0,-2) & (0,0,-2) & (0.5,0,-2) & (2,0,0) \\ (-2,1,-1) & (-0.5,1,-3) & (0,1,-2) & (0.5,1,-3) & (2,1,-1) \\ (-2,2,-2) & (-0.5,2,-4) & (0,2,2) & (0.5,2,-4) & (2,2,-2) \\ (-2,3,-2) & (-0.5,3,-4) & (0,3,2) & (0.5,3,-4) & (2,3,-2) \\ (-2,4,-1) & (-0.5,4,-3) & (0,4,-2) & (0.5,4,-3) & (2,4,-1) \\ (-2,5,0) & (-0.5,5,-2) & (0,5,-2) & (0.5,5,-2) & (2,5,0) \end{bmatrix}$$

5.2 Single Tool Path Generation

In our application, a flat-end cutter with the radius of 0.3 and the cutter length of 1 is used. The position of the single tool path is at u = 0.3, and the surface error is chosen to be 0.005.

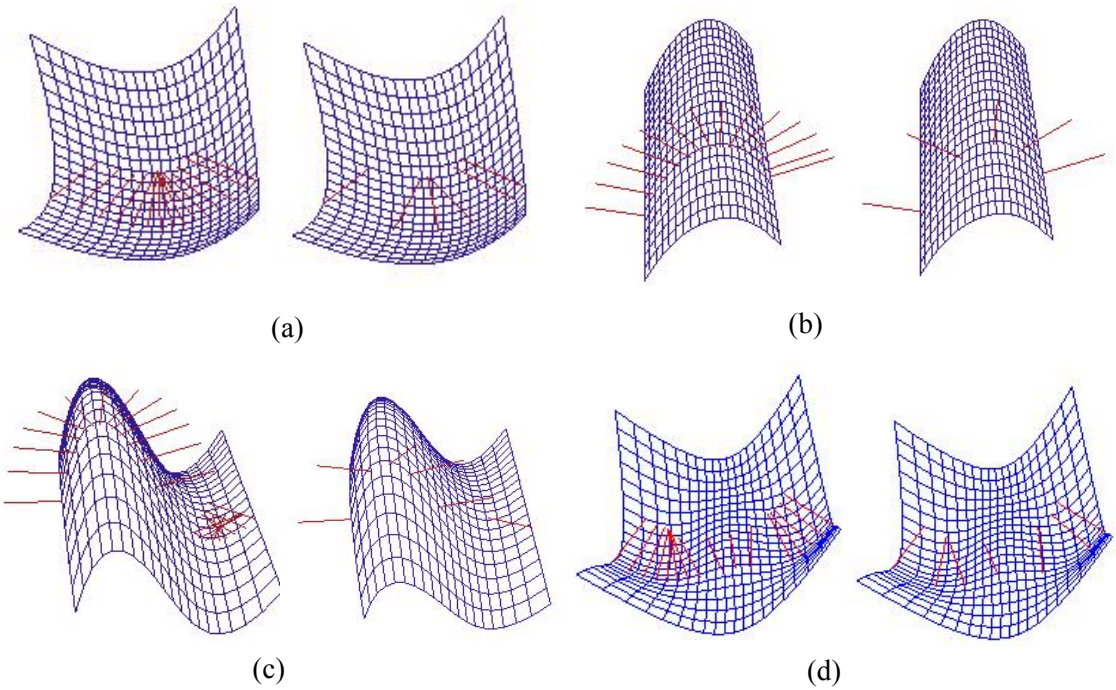


Fig. 5.2 The normal vectors of the CC points when τ =0.005 and τ =0.05

First, a set of CC points were generated on the surface curve **S**(0.3, v), assigning the searching tolerance τ as 0.005 and 0.05 respectively. Fig. 5.2 shows the

normal vectors of the CC points generated under the two surface error tolerances. We can see that the distribution of the CC points is according to the changes in curvature of the surface curve, i.e., CC points are dense on the surface part with larger curvature, while they are sparse where the curvature is smaller. Also, we can find that the larger the search tolerance, the less the CC points. For the sake of efficiency, the CC points with τ =0.05 is used here for the subsequent tool path generation.

Second, the associated CLs to the CC points are generated. The default inclining and tilting angles were chosen as λ_L=5° and ω_L=0°. Fig. 5.3 shows the CLs before gouging avoidance (the left image shows the cutter postures and the right one shows the view from the back of the surface patch). There are gouging problems between the cutter and the designed surface in Fig. 5.3a, c and d, while there is no interference in Fig. 5.3b. The reason is that gouging is prone to occurring at concave surfaces.

Gouging checking and avoidance is then performed and the result CLs after gouging avoidance can be shown in Fig. 5.4. We can see that the gouging no longer existed. However, the collision in Fig. 5.4c still exists because of the sharp change in curvature of this designed surface. Therefore, the algorithm for collision checking and avoidance need consequently be executed. Fig. 5.5 is the result CLs after collision avoidance for the third designed surface.

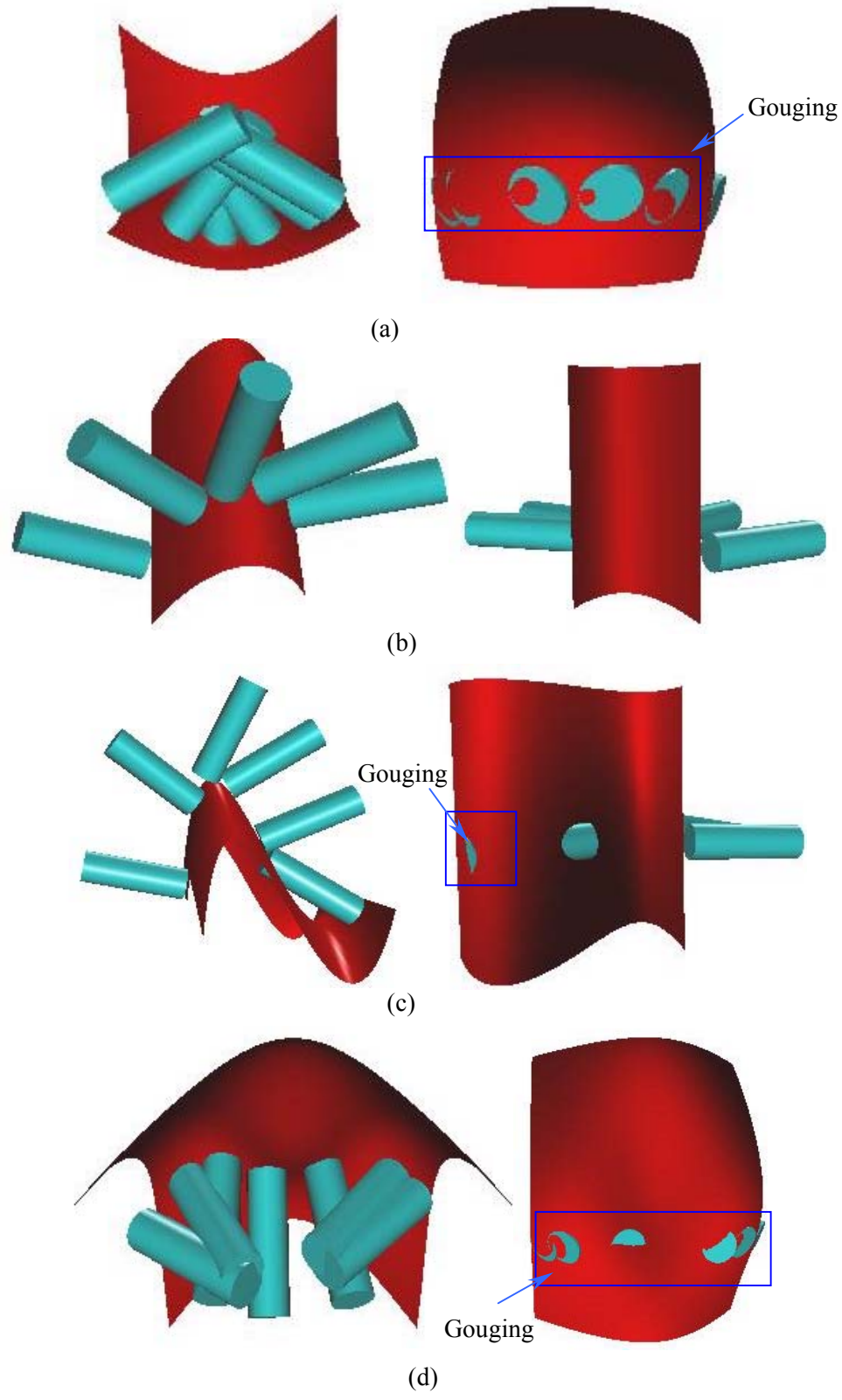


Fig. 5.3 The CLs before gouging avoidance

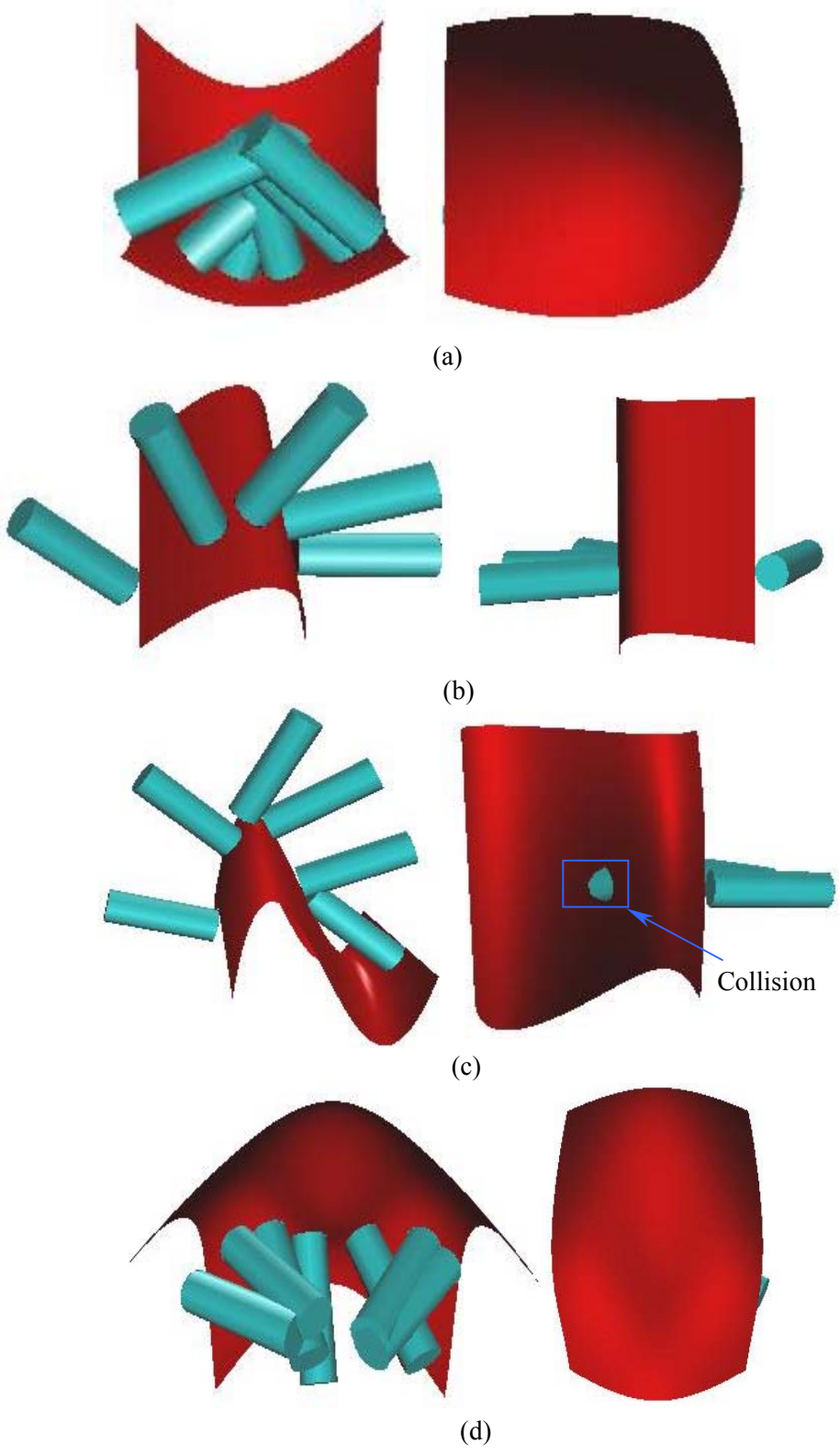


Fig. 5.4 The CLs after gouging avoidance

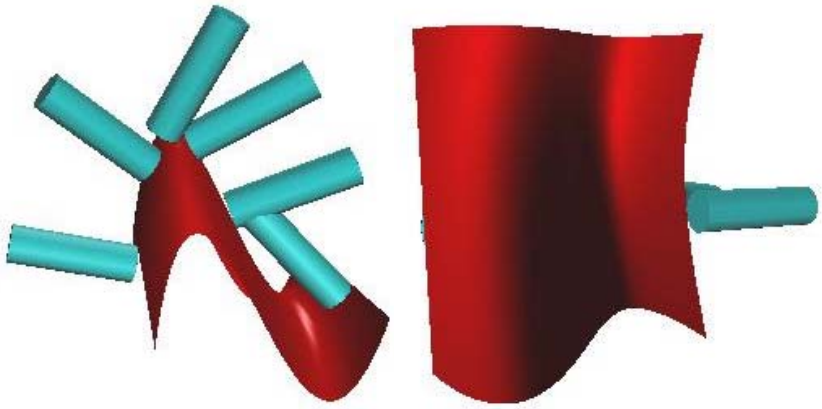


Fig. 5.5 The result CLs after collision avoidance for the third designed surface

After that, the piecewise rational Bézier dual quaternion curve of cutter motion is generated and then goes through the tool path verification and correction process. The cutter undergoing the initially generated piecewise rational Bézier motion for the first designed surface is shown in Fig. 5.6a and some interference exists. The cutters in *yellow* indicate the CLs where there is interference problem; while the cutters in *pink* indicate the CLs with fitting problem. Fig. 5.6b shows the resulted tool path after the first modification of the piecewise rational Bézier dual quaternion curve. The interference and fitting problem abated as opposed to that in Fig. 5.6a. Fig. 5.6c shows the resulted tool path after the second modification of the dual quaternion curve, from which we can see that neither interference nor fitting problems are detected and, therefore, the whole tool path is generated. The process for generating the interference-free tool path by the cutter undergoing the piecewise rational Bézier motion for the second, third and fourth designed surface is shown in Fig. 5.7, Fig. 5.8 and Fig. 5.9. Fig. 5.10 shows the resulted fitting error bound between the surface curve $\mathbf{S}(0.3, v)$ and the tool path generated by the piecewise rational Bézier motion of the cutter at the CC points (surface tolerance τ=0.005).

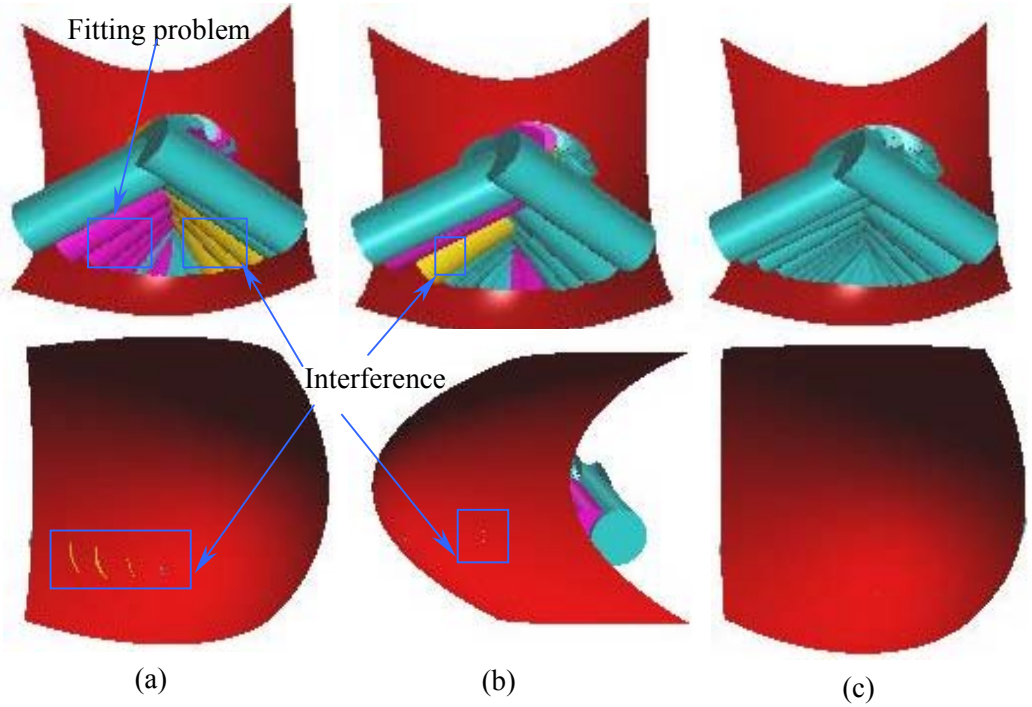


Fig. 5.6 The cutter undergoing the piecewise rational Bézier motion for 1st surface
(a) The initial tool path; (b) the 1st modified tool path; (c) the 2nd modified tool path;

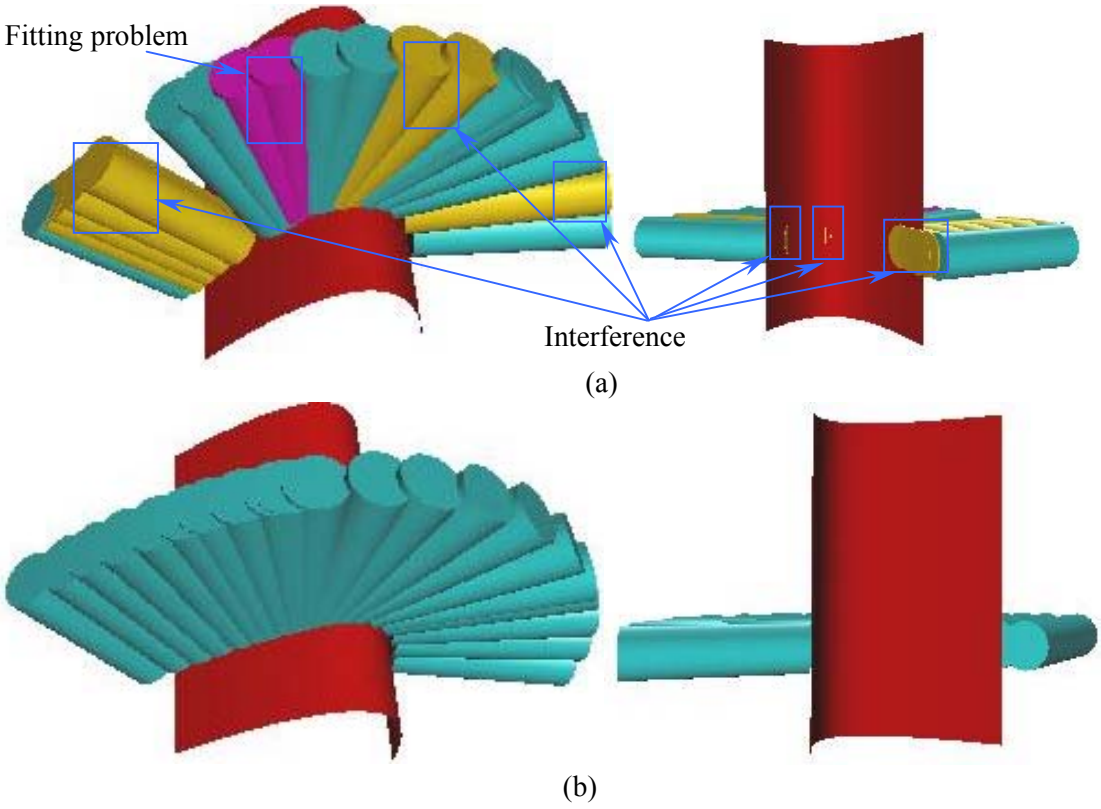


Fig. 5.7 The cutter undergoing the piecewise rational Bézier motion for 2nd surface
(a) The initial tool path; (b) the 1st modified tool path;

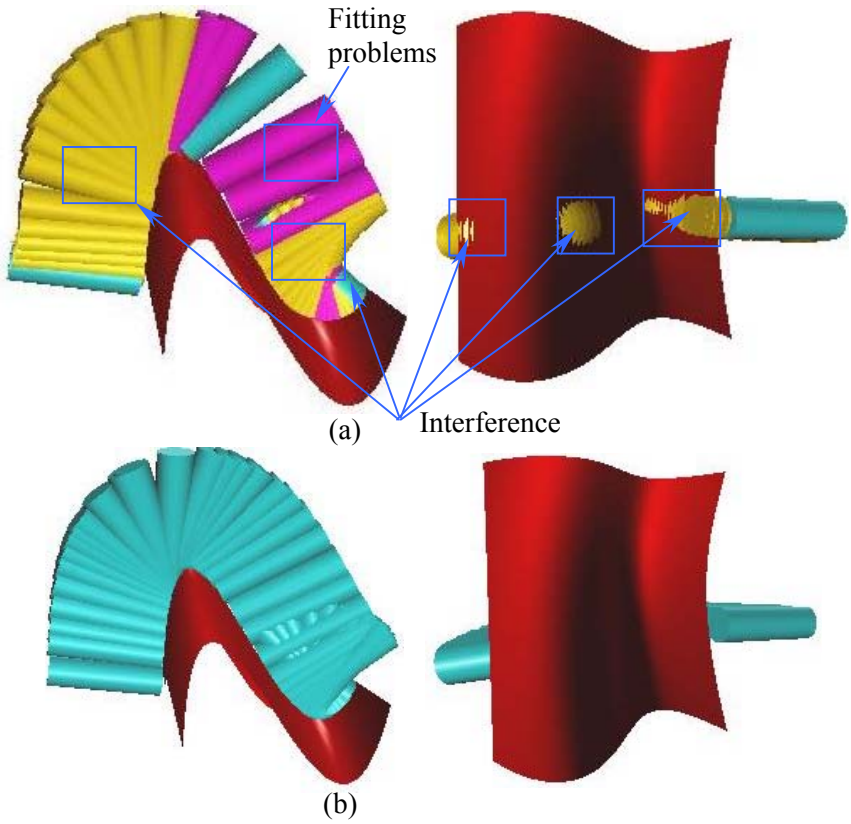


Fig. 5.8 The cutter undergoing the piecewise rational Bézier motion for 3rd surface (a) The initial tool path; (b) the 1st modified tool path;

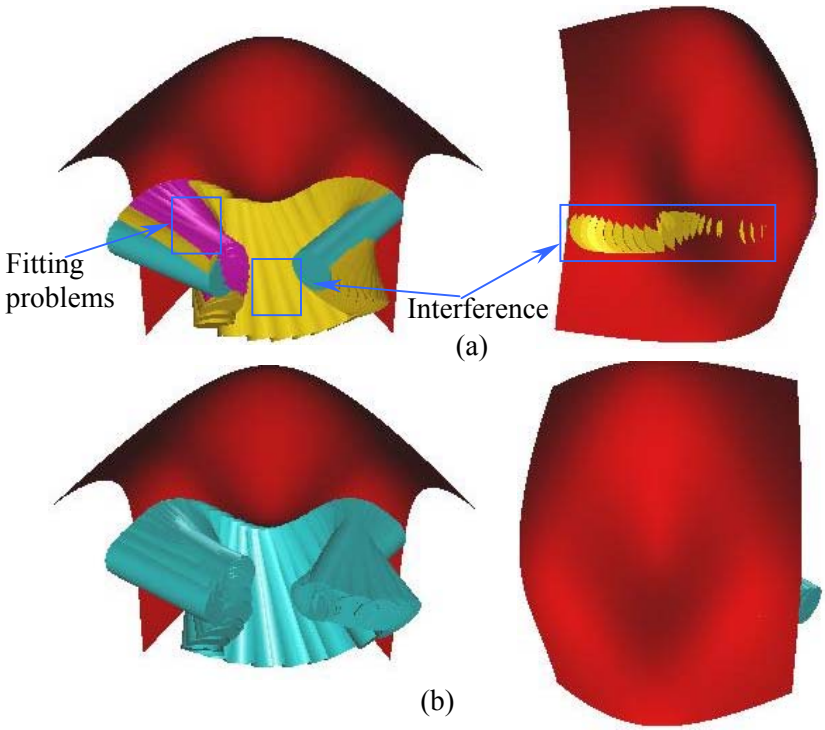


Fig. 5.9 The cutter undergoing the piecewise rational Bézier motion for 4th surface (a) The initial tool path; (b) the 1st modified tool path;

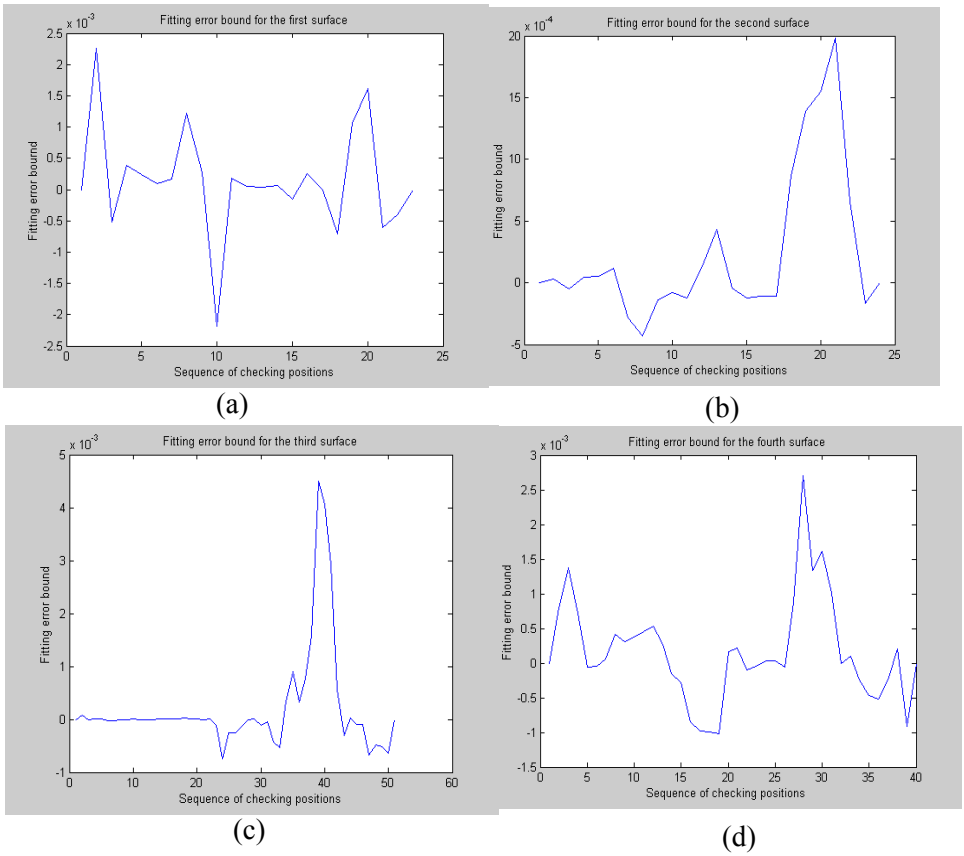
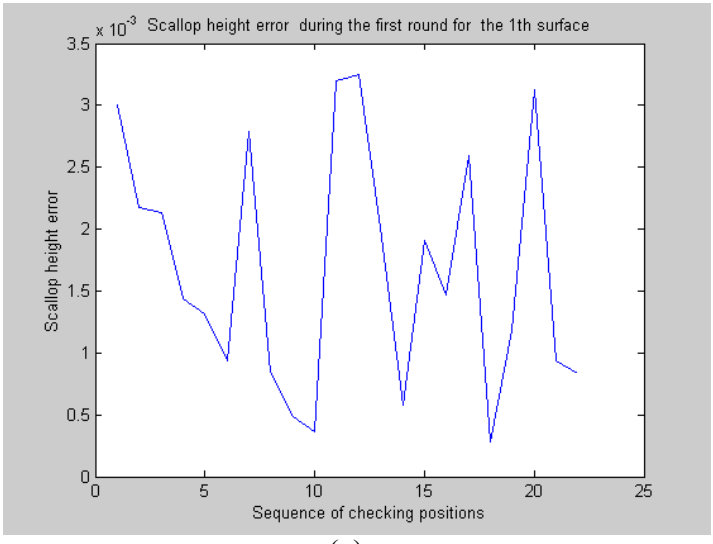


Fig. 5.10 Fitting error bound between **S**(0.3, *v*) and the tool path

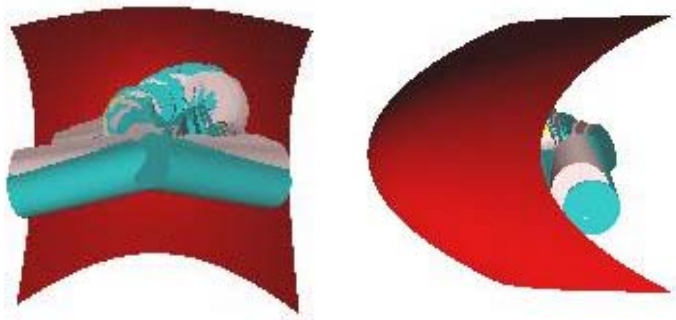
5.3 Multi Tool Paths Generation

Muti-tool path generation is an iterative process. According to chapter 4, the initial candidate next tool path is generated first, and the scallop height between the current and this candidate next tool path is consequently calculated. If the scallop height is out of tolerance (surface tolerance τ=0.005), the candidate next tool path is modified and the scallop height is recalculated. This process continues until we find the suitable scallop height between the current and candidate next tool path. This final candidate next tool path is then used as the next tool path. Fig. 5.11-Fig. 5.14 shows the process of finding the next tool path for above four surfaces defined in section 5.1. In Fig. 5.11, for the first surface, the scallop height of the first round estimation of the next tool path can meet the requirement. Thus, only one round is needed to obtain the next

tool path. In Fig. 5.12, for the second surface the scallop height in (c) is suitable and three rounds are needed to obtain the next tool path. For the third surface, four rounds are needed to generate next tool path, as shown in Fig. 5.13. From Fig. 5.14, we can see that for the fourth designed surface, the first round of the estimation of next tool path can meet our requirement.



(a)



(b)

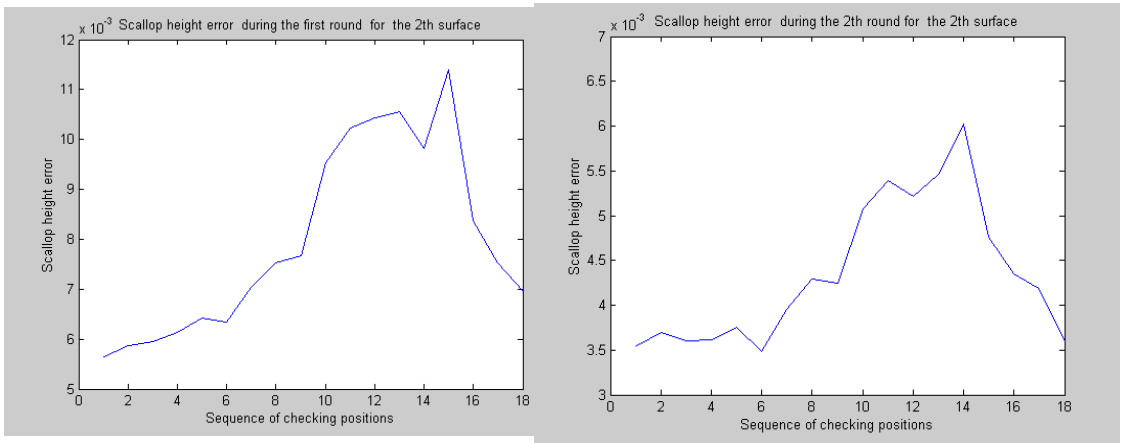
Fig. 5.11 The process of finding the next tool path for first designed surface

(a) Scallop height error when the parameter *v* for candidate next tool path is 0.361

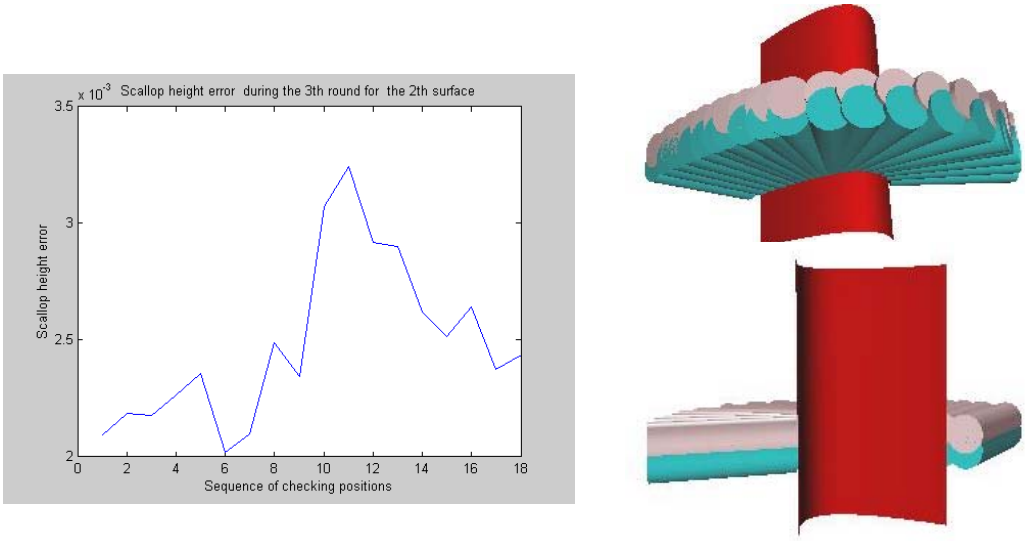
(b) The CLs of the neighboring tool paths

Using the similar method aforementioned, we can get the entire iso-parameter tool paths for these designed surfaces. Fig. 5.15 shows the entire CLs to generate the first designed surface, and 24 iso-parameter tool paths are needed to manufacture this

surface. The CLs of the tool paths for the second surface are shown in Fig. 5.16. 18 tool paths are needed to manufacture this surface. Fig. 5.17 shows the tool paths of the fourth surface. In this application, 37 tool paths are needed.



(a) (b)



(c) (d)

Fig. 5.12 The process of finding the next tool path for second designed surface

(a)-(c) Scallop height error when the parameter v for candidate next tool path are 0.394, 0.374, 0.359

(d) The CLs of the neighboring tool paths

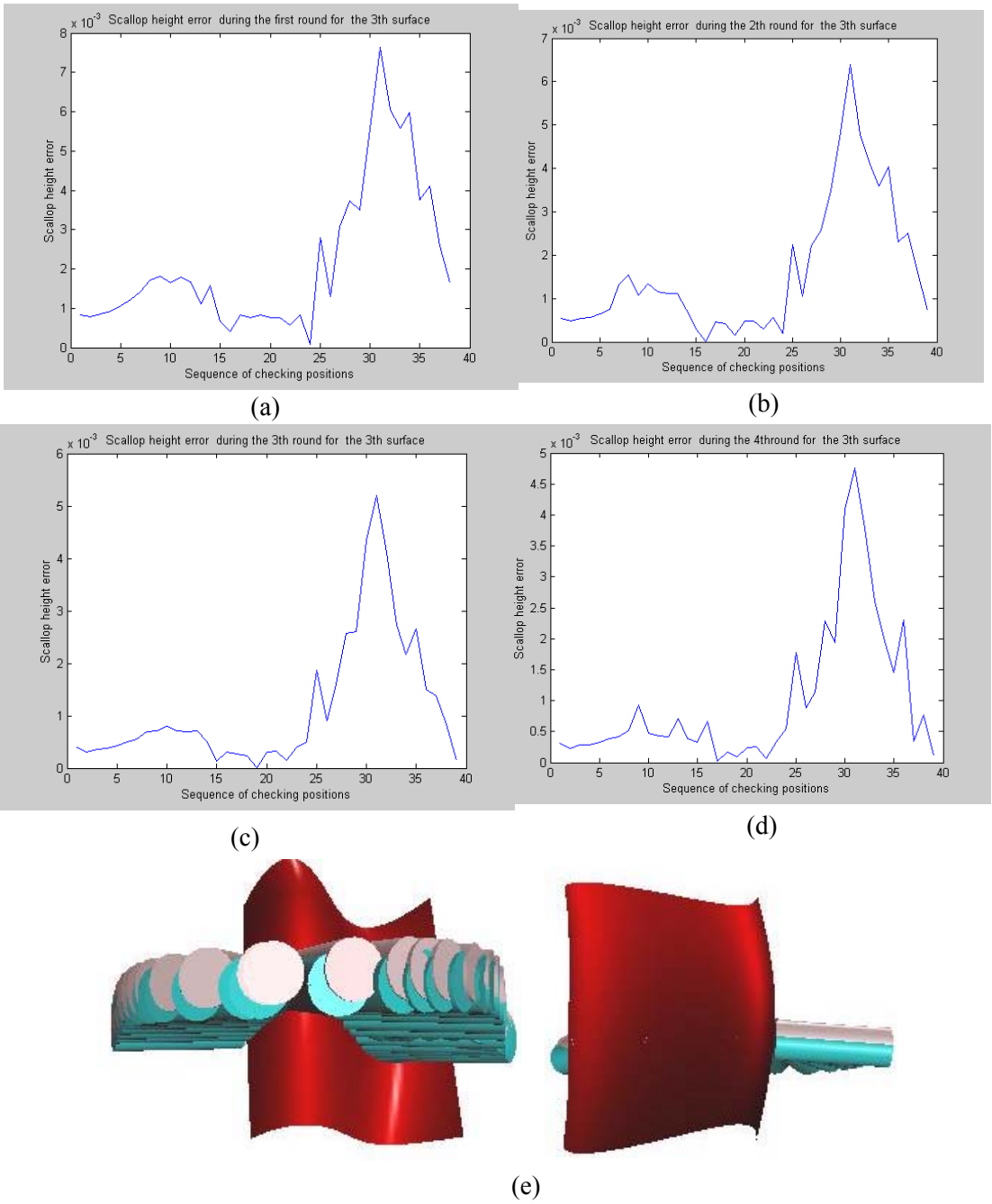
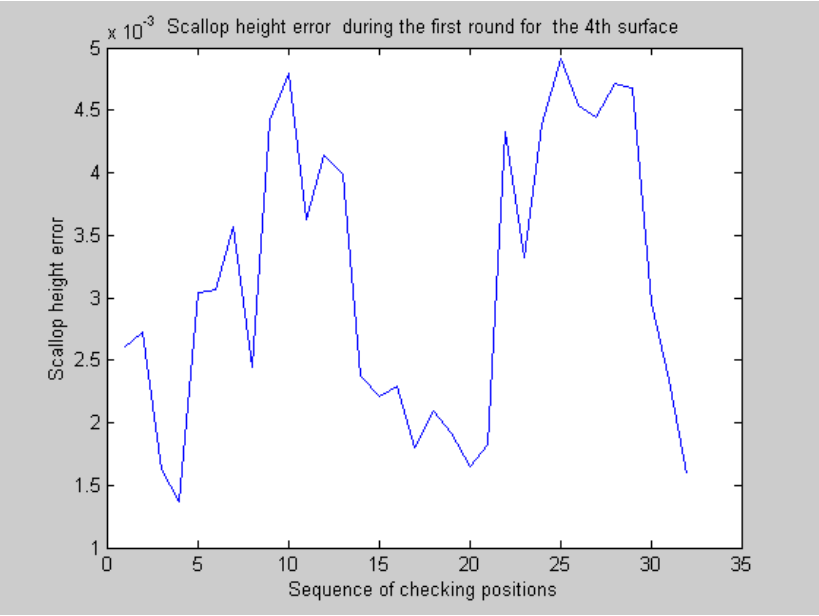


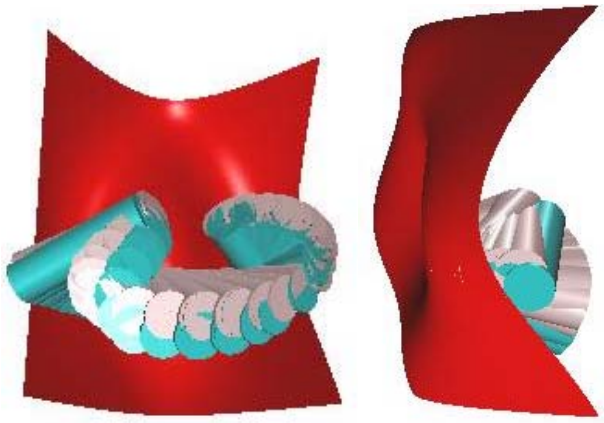
Fig. 5.13 The process of finding the next tool path for 3rd designed surface

(a)-(d) Scallop height error when the parameter v for candidate next tool path are 0.328, 0.322, 0.318,0.314

(e) The CLs of the neighboring tool paths



(a)

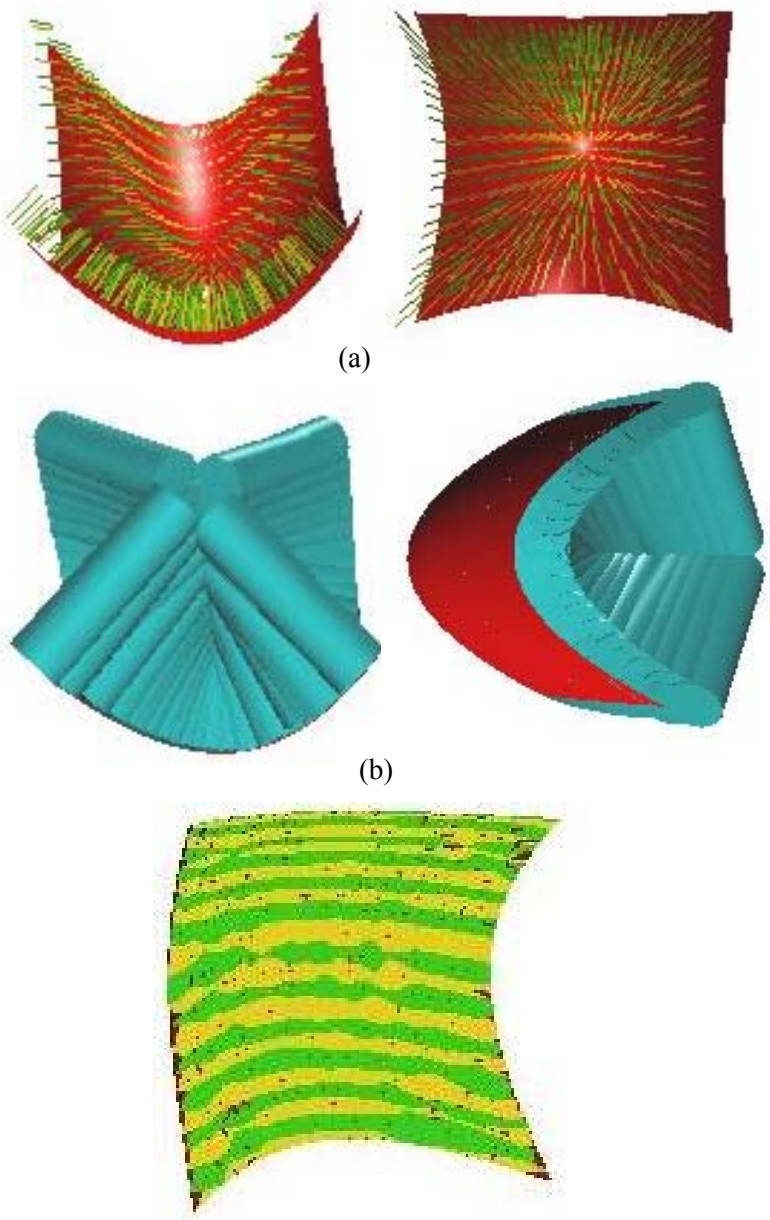


(b)

Fig. 5.14 The process of finding the next tool path for 4th designed surface

(a) Scallop height error when the parameter v for candidate next tool path is 0.340

(b) The CLs of the neighboring tool paths



(a)

(b)

(c)

Fig. 5.15 The entire tool paths generation for first designed surface

(a) The positions of the cutter axis

(b) The CLs of the entire tool paths

(c) Machined surface

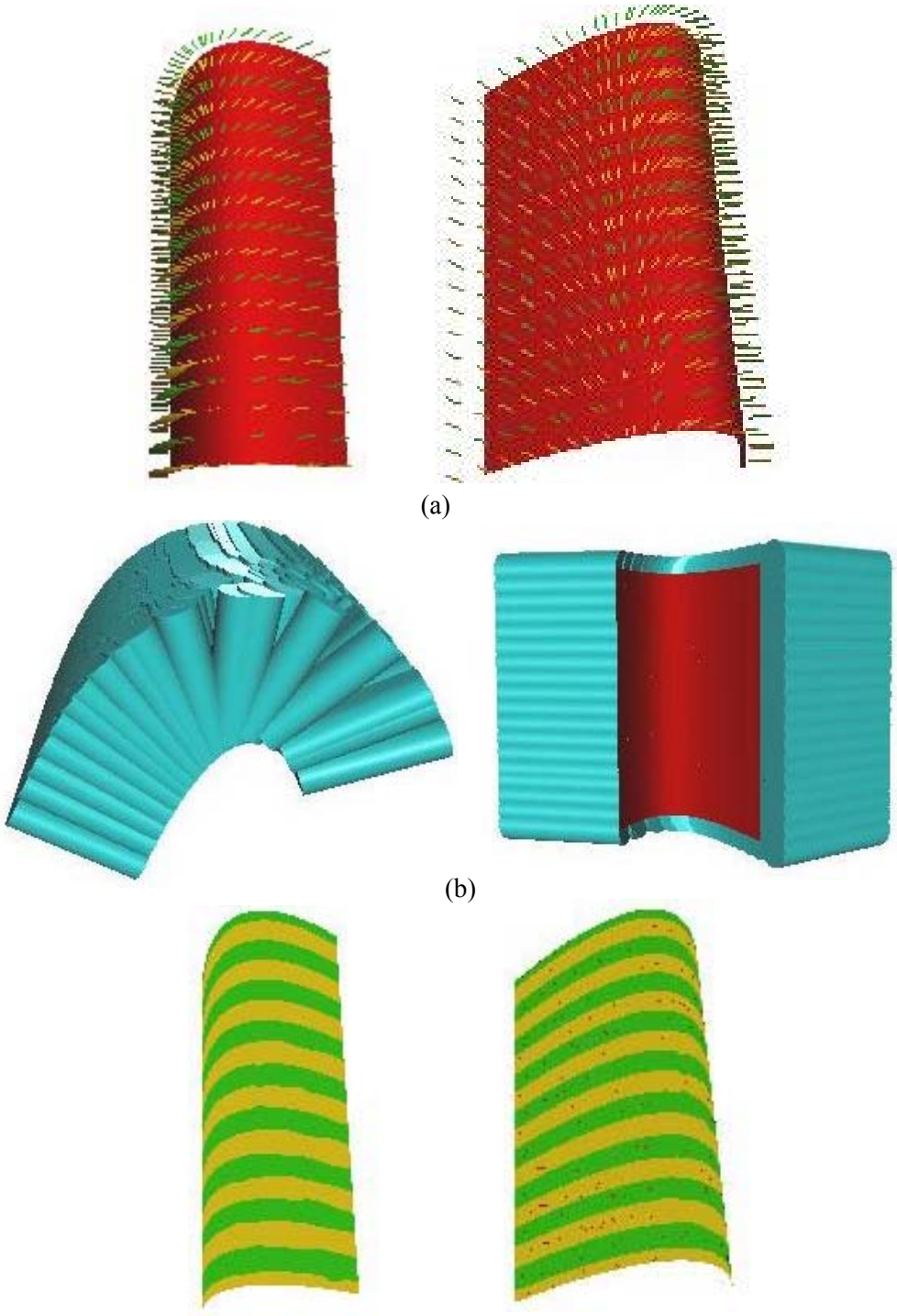
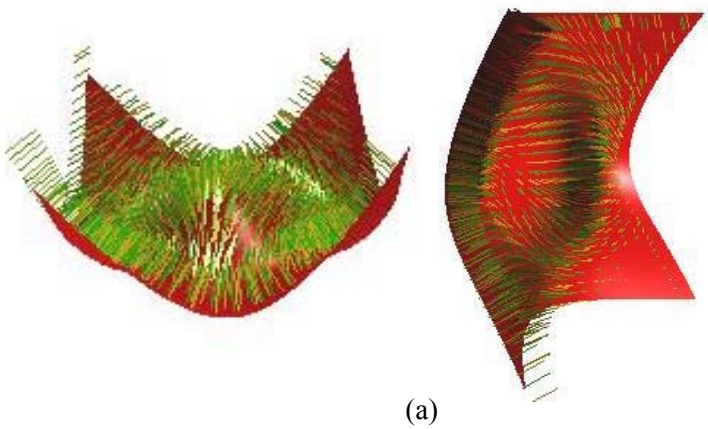


Fig. 5.16 Entire tool paths generation for second designed surface

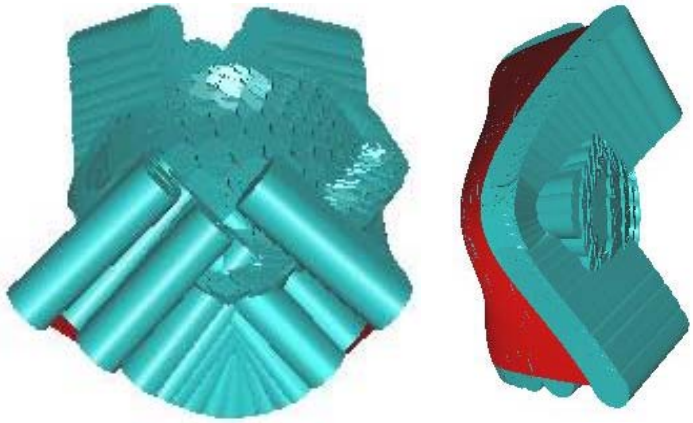
(a) The positions of the cutter axis

(b) The CLs of the entire tool paths

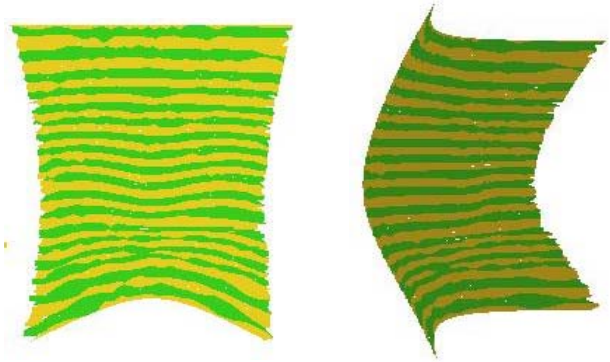
(c) Machined surface



(a)



(b)



(c)

Fig. 5.17 Entire tool paths generation for 4th designed surface

(a) The positions of the cutter axis

(b) The CLs of the entire tool paths

(c) Machined surface

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In modern manufacturing, 5-axis machining is commonly used in automotive, aerospace and tooling industries. Compared to 3-axis machining, 5-axis machining offers many advantages. However, for the moment, 5-axis machining tool path generation remains a difficult task. The problems include insufficient support by conventional CAD and CAM systems, highly complex algorithms for gouging avoidance and collision detection. In order to improve the efficacy and reduce the problems of 5-axis tool path generation, in this project, efforts are concentrated on finding a new method that uses the cutter undergoing piecewise rational Bézier motion to generate 5-axis tool path for sculptured surface. Following are the conclusion of the thesis:

(1) The state of art of 5-axis machining of sculptured surface has been studied. Traditional methods for tool path generation, verification simulation of gouging and collision, optimisation of cutter orientation and prediction the real scallop height have been stated. The general drawbacks of these methods have also been pointed out.

(2) The fundamental mathematics required in developing the thesis has been presented. The basic geometric modelling methods in CAGD such as Bézier, B-spline, and NURBS curve and surface are reviewed and the concepts of

kinematic Driven geometric modelling such as dual, quaternion and dual quaternion representation of a spatial displacement are introduced.

(3) An efficient approach to generate a single gouging-free and collision-free tool path for 5-axis sculptured surface machining using rational Bézier motion of the flat-end cutter is presented. The method deploys dual quaternion representation of a spatial displacement, and presents the mathematic form of cutter bottom circle undergoing the rational Bézier motion. According to the fundamental knowledge aforementioned, the procedure of 5-axis tool path generation is given in detail. First, a set of cutter contact points by considering local curvature of surface curve are obtained and the associated gouging-free and collision-free cutter locations are given. Second, all such cutter locations are converted to dual quaternion representation and then the rational Bézier dual quaternion motion curve that represents cutter motion is presented using the method for curve interpolation. Third, the swept surface of cutter undergoing rational Bézier motion is calculated and the interference checking is carried out based on this swept surface and the designed surface. Finally, the method of interference avoidance is presented and the feasible cutter locations on one tool path are obtained.

(4) Multi tool paths generation using the similar method as in (3) have been generated. First, the effective cutting shape has been represented accurately by intersecting the swept surface of the motion of the cutter and the cutting plane. Second, an iterative process to generate the adjacent tool path so that the scallop height between two neighboring tool paths is within the allowable tolerance has been conducted. In detail: (a) the candidate next tool path is obtained based on initial step size. (b) The scallop curve is calculated between

the two neighboring swept surface generated by the rational Bézier of cutter motion. (c) The scallop height checking is carried out by finding the distance between the scallop curve and designed surface. (d) If the scallop height is out of tolerance, the step size is reduced and the whole process is revisited until a suitable next tool path is found.

Compared with the traditional methods for tool path generation that mainly focus on a particular instant of the tool motion and studying local geometric issues at the instant, the main advantages of rational Bézier motions for tool path representation include:

(a) The entire tool path can be represented using a much more compact set of control positions of the motion as opposed to a huge data set of discrete cutter positions;

(b) Since the cutter motion representation is analytic, it provides an exact representation of effective cutting shape so that tool path verification can be carried out accurately.

(c) Fewer CC points are involved thus less computation required; Since the tool path is represented analytically, complete elimination of gouging and surface accuracy, to a large extent, can be guaranteed between neighboring CLs.

(d) Furthermore, in our method, the effective cutter shape used for the generation of the cutter locations in the next tool path can be represented exactly. This could lead to an accurate computation of scallop height between two neighboring tool paths.

6.2 Suggestions for Future Work

In this thesis, we have presented a new method that uses the cutter undergoing piecewise rational Bézier motion to generate 5-axis tool path for sculptured surface. The future improvement of the algorithm can be focused on the following aspects:

(1) The collision between tool holder and the designed surface can be investigated. In our algorithm, the collision between the designed surface and the cutter has been checked. However, the collision may possibly occur between the tool holder and the designed surface. Further study need take the tool holder into account.

(2) The swept surface of the cylindrical surface of the cutter undergoing the piecewise rational Bézier motion can be constructed and used to check collision. In our algorithm, the collision checking is still based on discrete points along the tool path. If we can generate the swept surface of the cylindrical surface of the cutter, the collision checking can be thought as finding the intersection between this swept surface and the designed surface. This representation is more compact.

(3) The constant scallop height tool paths can be generated. In our algorithm, the tool paths are the iso-parameter tool paths. Although iso-parameter tool paths are computationally simple to generate, one serious problem of this method is the inefficient machining due to the non-predictable scallop remaining on the machined surface. If the constant scallop height tool path generation method is applied, the number of the tool path will be much smaller and the efficiency of the machining is improved.

(4) The algorithm for modifying the dual quaternion curve when gouging or collision occur need to be improved. In our method, we first find out the surface points corresponding to the gouging or collision, and reconstruct the dual quaternion motion curve based on these points. It is better to modify rather than reconstruct this curve in order to achieve the efficiency of the algorithm.

REFERENCES

Balasubramaniam M, Sarma SE and Marciniak K, Collision-free finishing tool paths from visibility data, Computer Aided Design, Vol. 35, pp.359-374, 2003.

Bottema O and Roth B, Theoretical Kinematics, Amsterdam: North Holland Publishing Company, 1979.

Cheng MY, Tsai MC and Kuo JC, Real-time NURBS command generators for CNC servo controllers**,** International Journal of Machine Tools and Manufacture, Vol. 42, No.7, pp.801-813, 2002.

Chiou CJ and Lee YS, A machining potential field approach to tool path generation for multi-axis sculptured surface machining, Computer Aided Design, Vol. 34, pp. 357-371, 2002.

Choi BK,Park JW and Jun CS, ,Cutter-location data optimisation in 5-axis surface machining. Computer Aided Design, Vol. 25, No.6, pp.377-386, 1993.

Choi BK and Jerard RB, Sculptured surface machining, Dordrecht: Kluwer Academic, 1998.

Dragomatz D and Mann S, A classified bibliography of literature on NC milling path generation, Computer Aided Design, Vol. 29, No. 3, pp.239-247, 1998.

Elber G and Cohen E, Tool path generation for freeform surface models, Computer Aided Design, Vol. 26,No. 6, pp. 490-496, 1994.

Faux ID and Pratt MJ. Computational geometry for design and manufacturing, New York: Wiley, 1981.

Farin G, Curves and surfaces for computer-Aided geometric design: practical guide, 4th edition, Academic Press, 1996.

Ge QJ and Ravani B, Computer aided geometric design of motion interpolants, ASME Journal of Mechanical Design, Vol. 116, No.3, pp. 756-762, 1991.

Ge QJ and Ravani B, Computation of spatial displacements from geometric features, ASME Journal of Mechanical Design, Vol. 115,pp.95-102, 1993a.

Ge QJ and Ravani B, Computational geometry and motion approximation, In Computational Kinematics, Angeles J. et al. (eds.), pp. 229-238, Kluwer Academic Press, 1993b

Ge QJ and Ravani B, Geometric construction of Bézier motions. ASME Journal of Mechanical Design, Vol. 116, No.3, pp. 749-755, 1994a.

Ge QJ and Srinivasan LN, Fine tuning of rational B-Spline motions, ASME Journal of Mechanical Design, Vol. 120, No. 3, pp. 46-51,1998.

Hamilton WR, Elements of Quaternions, New York: Chelsea Pub, 1969.

Huang Y and Oliver JH, Non-constant parameter NC tool path generation of sculptured surface, International Journal of Advanced Manufacturing Technology, Vol. 9, pp. 281-290,1994

Jensen CG, Mullins SH and Anderson DC, Scallop elimination based on precise 5-axis tool placement, orientation and step over calculations, ASME Advanced Design Automation, Vol. 65, No.2, pp. 535-544,1993.

Jensen C and Anderson D, A review of numerically controlled methods for finish sculptured surface machining. IIE transactions, Vol. 28, No. 1, pp. 30-39,1996.

Jensen CG, Red WE, and Pi J, Tool selection for five-axis curvature matched machining, Computer Aided Design, Vol. 34, pp. 251-266, 2002.

Jun CS, Cha K and Lee YS, Optimizing tool orientations for 5-axis machining by configuration-space search method, computer aided design, Vol. 35, No.6, pp. 549-566, 2003.

Juttler B and Wagner MG, Computer-aided design with spatial rational B-spline motions, ASME Journal of Mechanical Design, Vol.118, No.2, pp.193-201, 1996.

Juttler B and Wagner MG, Rational motion-based surface generation, Computer aided design, Vol. 31, pp. 203-213, 1999.

Kang DL. Motion synthesis for computer aided geometric design. Ph.D Thesis. State University of New York,1997.

Kim BH and Chu CN, Effect of cutter mark on surface roughness and scallop height in sculptured surface machining, Computer Aided Design, Vol. 26, No. 3, pp. 179-188, 1994.

Kruth JP and Klewais P, Optimization and dynamic adaptation of the cutter inclination during five-axis milling of sculptured surfaces, Annals of the CIRP, Vol. 43, No. 1,pp. 443-448, 1994.

Lauwers B, Dejonghe P and Kruth JP, Optimal and collision free tool posture in five-axis machining through the tight integration of tool path generation and machine simulation, Vol. 35, No. 15, pp. 421-432, 2002.

Lee YS and Chang TC, CASCAM-An automated system for sculptured surface cavity machining, Computer in industry, Vol. 16, pp.321-342, 1991.

Lee YS, Chang TC. Two-phase approach to global tool interference avoidance in 5-axis machining. Computer-Aided Design, Vol. 27, No. 27, pp. 715-729, 1995.

Lee YS, Chang TC, Automatic cutter selection for 5-axis sculptured machining. International Journal of Production research, Vol. 34, pp.111-135, 1996a

Lee YS and Chang TC, Machined surface error analysis for 5-axis machining, International Journal of Production research, Vol34, pp.111-135, 1996b.

Lee YS, Admissible tool orientation control of gouging avoidance for 5-axis complex surface machining, Computer Aided Design, Vol. 29, No.7, pp. 507-521, 1997.

Lee YS, Non-isoparametric tool path planning by machining strip evaluation for 5-axis sculptured surface machining, computer aided design, Vol. 3, No.7, pp.559-570, 1998a.

Lee YS, Mathematical modelling using different end-mills and tool placement problems for 4- and 5-axis NC complex surface machining. International Journal of Production and Research, Vol. 36, pp.785-814, 1998b.

Li SX and Jerard RB, 5-axis machining of sculptured surfaces with a flat-end cutter, Vol. 26, pp. 165-178, 1994.

Liu XW, Five-axis NC cylindrical milling of sculptured surface, computer-aided design, Vol. 27, No.12, pp.887-894, 1995.

Lin R.S. and Koren Y, Efficient tool-path planning for machining freeform surfaces, ASME Journal of Engineering for Industry, Vol. 118, pp. 20-28, 1996.

Lo CC, Efficient cutter-path planning for five-axis surface machining with a flat end-cutter, Computer Aided Design, Vol. 31, pp. 557-566, 1999.

Marciniak K, Influence of surface shape on admissible tool positions in 5-axis face milling, Computer Aided Design, Vol. 19, No. 5, pp. 233-236, 1987.

Marciniak K, Geometric modelling for numerically controlled machining, Oxfond University Press, 1991.

Morishige K, Kase K and Takeuchi Y, Collision-free tool path generation using 2-dimensional C-space for 5-axis control machining, International Journal of Advance Manufacture Technology, Vol 13, No.6, pp.393-400, 1997.

Morishige K and Takeuchi Y, Tool path generation using C-Splace for 5-axis control machining, ASME Journal of Manufacturing Science and Engineering, Vol. 121, pp.144-149, 1999.

Nadim MA,Reda B and Sudarshan B, Bézier surface/surface intersection, IEEE Computer Graphics & Applications, Vol. 10, No. 1, pp. 50-58, 1990.

Nelder JA and Mead R, A Simplex Method for Function Minimization, Computer Journal, Vol. 7, pp. 308–313, 1965.

Pi JP, Red E and Jensen G, Grind-free tool path generation for five-axis surface machining, Computer Integrated Manufacturing Systems, Vol. 11, No. 4, pp. 337-350, 1998.

Piegl L and Tiller W, The NURBS book, Springer-Verlag, 1995.

Rao N, Bedi S and Buchal R, Implementation of the principal-axis method for machining of complex surface, International Journal of Advanced manufacture technology, Vol. 11, pp. 249-257, 1996.

Rao N, Ismail F. and Bedi S, Tool path planning for five-axis machining using the principal axis method, International machine tools manufacture, Vol 37, No.7, pp. 1025-1040, 1997.

Rao A and Sarma R, On local gouging in five-axis sculptured machining using flat-end tools, Computer Aided Design, Vol. 32, pp. 409-420, 2000.

Redonnet JM, Rubio W, Monies F and Dessein G, Optimising tool positioning for end-mill machining of free form surface on 5-axis machines for both semi finishing and finishing, International Journal of Advanced Manufacture Technology, Vol. 16, pp. 383-391, 1998.

Sarma R and Dutta D, The geometry and generation of NC tool path, Journal of Mechanical Design, Vol. 119, pp. 253-258, 1997.

Sarma R and Dutta D, Tool path generation for NC grinding, International Journal of Machine Tool Manufacture, Vol. 38, No. 3, pp.177-195, 1998.

Sarma R, Flat-ended Tool swept sections for five-axis NC machining of sculptured surface, ASME Journal of Manufacturing Science and Engineering, Vol. 22, No. 2, pp.158-165, 2000.

Spitz SN and Requicha AAG, Accessibility analysis for automatic inspection if mechanical parts by coordinate measuring machines, Proceedings of the IFFF international conference on robotics and automation, Cincinnati, OH, May 1990, pp. 1284-1289, 1990.

Suh SH and Kang JK, Process planning for multi-axis NC machining of free surfaces, International Journal Production and Research, Vol. 33, No. 10, pp. 2723-2738, 1995.

Suresh K and Yang DCH, Constant scallop-height machining of free-from surfaces, Journal of Engineering for Industry, Vol.116, No. 5, pp. 253-259, 1994.

Taylor RH, Planning and execution of straight line manipulator trajectories, IBM journal of research and development, Vol. 23, No. 4, pp. 424-436, 1979.

Vickers GW and Quan KW, Ball-Mills versus End-Mills for curved surface machining, Transactions of the ASME, Vol. 111, No. 2, pp. 22-26, 1989.

Wang H, Chang H, Wysk RA and Chandawarkar A, On the efficiency of NC tool path planning for face milling operations. Journal of Engineering for Industry, Vol. 109, No. 4, pp. 370-376, 1987.

Wang U and Tang XW, Five-axis machining of sculptured surfaces, International Journal of Advance Manufacture Technology, Vol. 15, pp. 7-14, 1999.

Wang W and Joe B, Robust computation of the RMF for swept surface modelling, Computer Aided Design, Vol. 29, pp. 379-391, 1997.

Woo TC, Visibility maps and spherical algorithms. Computer aided design, Vol. 26, No.1, pp.6-16, 1994.

Xia, J and Ge QJ, On the exact representation of the boundary surfaces of the swept surface undergoing rational Bézier and B-spline motions, Proc. 1999 ASME Design Automation Conference, 1999, Las Vegas, USA, paper No. DETC99/DAC-8607, 1999.

Xia J Motion based geometric modelling. Ph.D Thesis. State University of New York, 2001.

Xia J and Ge QJ, , Kinematic approximation of ruled surfaces usning NURBS motions of a cylindrical cutter, ASME Design Engineering Technical Conference, 2000, Maryland, USA, paper No. DETC2000/DAC-14280, 2000a.

Xia J and Ge QJ, On the exact computation of the swept surface of a cylindrical surface undergoing two-parameter rational Bézier motions, ASME Design Engineering Technical Conference, 2000 Maryland, USA, paper No. DETC2000/DFM-14039, 2000b.

Xia, J and Ge QJ, An exact representation of effective cutting shapes of 5 axis CNC machining using rational Bézier and B-spline tool motions. International conference on Robotics and automations, Korea ,2001, pp.253-258, 2001.

Yang W, Ding H and Xiong Y, Manufacturability analysis for a sculptured surface using visibility cone computation, International Journal of Advance Manufacture Technology, Vol 15, pp. 317-321, 1999.

Yoon JH, Pottmann H and Lee YS, Locally optimal cutting positions for 5-axis sculptured surface machining, Computer Aided Design, Vol. 35, pp. 69-81, 2003.

You CF and Chu CH, Tool-Path verification in Five-Axis machining of sculptured surfaces, international journal of advanced manufacturing technology, Vol. 13, pp.248-255, 1997.

Zhang QG and Greenway RB. Development and implementation of a NURBS curve motion command generator, Robotics and Computer-Integrated Manufacturing, Vol.14, pp.27–36, 1998.