TOOL PATH GENERATION AND 3D TOLERANCE ANALYSIS FOR FREE-FORM SURFACES

A Dissertation

by

YOUNG KEUN CHOI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2004

Major Subject: Industrial Engineering

TOOL PATH GENERATION AND 3D TOLERANCE ANALYSIS FOR FREE-FORM SURFACES

A Dissertation

by

YOUNG KEUN CHOI

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Amarnath Banerjee
(Chair of Committee)

César O. Malavé
(Member)

Sheng-Jen "Tony" Hsieh
(Member)

John C. Keyser
(Member)

Mark L. Spearman
(Head of Department)

May 2004

Major Subject: Industrial Engineering

ABSTRACT

Tool Path Generation and 3D Tolerance Analysis for
Free-Form Surfaces. (May 2004)
Young Keun Choi, B.S., Konkuk University;
M.S., Konkuk University
Chair of Advisory Committee: Dr. Amarnath Banerjee

This dissertation focuses on developing algorithms that generate tool paths for free-form surfaces based on the accuracy of a desired manufactured part. A manufacturing part is represented by mathematical curves and surfaces. Using the mathematical representation of the manufacturing part, we generate reliable and near optimal tool paths as well as cutter location (CL) data file for postprocessing. This algorithm includes two components. First is the forward-step function which determines the maximum distance, called forward step, between two cutter contact (CC) points with a given tolerance. This function is independent of the surface type and is applicable to all continuous parametric surfaces that are twice differentiable. The second component is the side-step function which determines the maximum distance, called side-step, between two adjacent tool paths with a given scallop height. This algorithm reduces manufacturing and computing time as well as the CC points while keeping the given tolerance and scallop height in the tool paths. Several parts, for which the CC points are generated using the proposed algorithm, are machined using a three axes milling machine. As part of the validation process, the tool paths generated during machining are analyzed to compare the machined part and the desired part.

ACKNOWLEDGMENTS

I would like to wholeheartedly thank my graduate advisor Dr. Amarnath Banerjee for supporting and guiding me throughout this research. He has been very generous with his time and has helped me focus on the research at hand. He was always encouraging me through the good and bad phases of my research.

My dissertation committee members, Dr. César O. Malavé, Dr. Sheng-Jen “Tony” Hsieh, and Dr. John C. Keyser all deserve a special thank for devoting the time to give me words of advice during my research and for their effort in reading my dissertation.

I would like to thank Mr. Shivcharan Venkat Kamaraju and Mr. Chang-ho Chin for the number of hours they spent with me machining and measuring several real parts.

I appreciate the help of the Department of Industrial Engineering and Dr. Sheng-Jen “Tony” Hsieh for providing financial assistance as a graduate assistant lecturer and teaching assistant, which helped me in establishing a teaching career. In particular, I would like to thank Mrs. Judy Meeks for her help.

My parents, Mr. Byoung-Hong Choi and Mrs. Sinja Cho, deserve a big thanks for always encouraging me to do my best and for never losing faith in my abilities. I would like to thank my wife, Yeonkung, my daughter, Heewon, and my son, William (Yongjun) for their love and devotion during this endeavor. It would not have been possible without their love and patience.

There are many others who helped me during this research and though I am not mentioning any names here, I am grateful to all of them.

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION 1

- A. Introduction 1
  - 1. Introduction 1
  - 2. Computer-Aided Part Programming 2
- B. Related Work and Literature Review 6
  - 1. Tool Path Distribution Strategy 7
  - 2. Tool Path Calculation Strategy 10
  - 3. Literature Review 14
- C. Research Motivation 15
- D. Dissertation Outline 19

II MATHEMATICAL REPRESENTATION 20

- A. The De Casteljau Algorithm 20
- B. Blossom of Polynomial 21
- C. Bézier Curve 24
  - 1. Mathematical Representation of Bézier Curve 24
  - 2. The Derivative of a Bézier Curve 26
- D. Bézier Surface Patch 29
  - 1. Mathematical Representation of Bézier Surface 29
  - 2. The Derivative of a Bézier Surface 30
- E. Differential Geometry 31
  - 1. Differential Geometry of Curves 32
  - 2. Differential Geometry of Surfaces 34

III PROPOSED APPROACH 36

- A. Tool Characteristics 36
  - 1. Cutter Contact Point and Cutter Location Point . . . 37
  - 2. Conversion of CC to CL 37
- B. Overall Conceptual Approach 38
- C. Calculation of Forward-Step 41
  - 1. Theory 41
  - 2. Forward-Step Function 44
- D. Calculation of Side-Step 47

CHAPTER Page

        1. Part Geometry 49
        2. Side-Step Function 51
    E. Combination of Forward and Side-Step Error 55
    F. Conversion of the Interval to the Parametric Domain . . . 55
        1. Conversion of Forward and Side-Step Size 55
            a. Direction of Forward-Step 57
            b. Direction of Side-Step 57
            c. Conversion 58
        2. Approximation of Functions 60

IV IMPLEMENTATION 62

    A. Experiment and Results 63
    B. 3D Tolerance Analysis 78
    C. Integrate to CAD/CAM System 82

V RESEARCH SUMMARY AND CONCLUSIONS 84

    A. Research Contribution and Conclusion 84
    B. Future Direction 85

REFERENCES 87

APPENDIX A 92

VITA 104

LIST OF FIGURES

FIGURE Page

1 Process planning modules and databases 3

2 Tasks in computer-aided part programming 7

3 Distribution strategies 9

4 Surfaces in APT 11

5 Boolean intersection 13

6 CSG operation 13

7 Forward-step and side-step 17

8 The de Casteljau algorithm 22

9 Bézier curve over interval $[a, b]$ 25

10 Frenet frame 33

11 Curve frame 34

12 CC and CL point 38

13 Overall procedure 39

14 Cubic Bézier curve generated by the de Casteljau algorithm . . . 48

15 Cubic Bézier curve and CC points 48

16 Linear approximation of CC points 49

17 Sign convention of radius of the curvature 52

18 Tool position on flat surface 52

19 Tool position on convex surface 54

FIGURE Page

20 Tool position on concave surface 54

21 Practical tool position on convex surface 56

22 Practical tool position on concave surface 56

23 Angular differences between the parametric direction and forward-step 58

24 Angular differences between the parametric direction and side-step 59

25 Tool path of example 1 64

26 Comparison between desired part and machined part for example 1 with tolerance 0.05 inches 65

27 Machined part of example 1 with tolerance 0.05 inches 65

28 Scanned surface of example 1 with tolerance 0.05 inches 66

29 Comparison between desired part and machined part for example 1 with tolerance 0.01 inches 67

30 Machined part of example 1 with tolerance 0.01 inches 67

31 Scanned surface of example 1 with tolerance 0.01 inches 68

32 Tool path of example 2 72

33 Comparison between desired part and machined part for example 2 with tolerance 0.05 inches 73

34 Machined part of example 2 with tolerance 0.05 inches 74

35 Scanned surface of example 2 with tolerance 0.05 inches 75

36 Comparison between desired part and machined part for example 2 with tolerance 0.01 inches 76

37 Machined part of example 2 with tolerance 0.01 inches 76

FIGURE Page

38 Scanned surface of example 2 with tolerance 0.01 inches 77

39 Annotation on the top view 79

40 CAD/CAM-integrated tool path generation 83

41 Cross section through X and Y axis 92

42 Cross section analysis through X and Y axis 93

43 Cross section through Y and Z axis 94

44 Cross section analysis through Y and Z axis 95

45 Cross section through X and Z axis 96

46 Cross section analysis through X and Z axis 97

47 Back view 98

48 Left view 99

49 Right view 100

50 Top view 101

51 Bottom view 102

52 Isometric view 103

LIST OF TABLES

TABLE		Page
I	Value of s	45
II	Computational results	49
III	Result of example 1	69
IV	Result of example 2	71
V	Tolerance analysis 1	80
VI	Tolerance analysis 2	81

CHAPTER I

INTRODUCTION

A. Introduction

1. Introduction

Process planning is the function within a manufacturing facility that establishes which processes and parameters are to be used to convert a part from its initial form to a final form predetermined in an engineering drawing. Alternatively, process planning could be defined as the act of preparing detailed work instructions to produce a part. Initial material can take a number of forms, the most common of which are bar stock, plate, casting, forging, or maybe just a slab of metal. With these raw material as a base, the process planner must prepare a list of processes to convert this normally predetermined material into a predetermined final shape.

Figure 1 represents the structure of a complete computer-aided process planning system. Although no existing turnkey system integrates all of the functions shown in the figure 1, it illustrates the functional dependencies of a complete process planning system. In figure 1, the modules are not necessarily arranged based on importance or decision sequence. The system monitor controls the execution sequences of the individual modules. Each module may require execution several times in order to obtain an ”optimum” process plan.

The input to the system will be a three-dimensional model from a computer-aided design (CAD) data base. The model contains not only the shape and dimensioning information, but also the tolerance and special features. The process plan can be

---

The journal model is *IEEE Transactions on Automatic Control.*

routed directly to the production planning system and production control system. Time estimates and resource requirement can be sent to the production planning system for scheduling. The part program, cutter location (CL) file, and material handing control program can also be sent to the control system.

Process planning is the critical bridge between design and manufacturing. Design information can be translated into manufacturing language only through process planning. Today both computer-aided design (CAD) and manufacturing (CAM) have been implemented. Integrating, or bridging, these functions requires automated process planning[6].

2. Computer-Aided Part Programming

In figure 1, cutter path generation module is the focus of this dissertation. In this module, numerical control (NC) plays an very important role to transform the raw material into a finished part specified on an engineering design that is either design on paper or in a CAD model. NC system consists of three basic components:

1. A program of instructions
2. A machine control unit
3. Processing equipment

The program of instructions is the detailed step-by-step command that direct the actions of the processing equipment. In machine tool application, the program of instructions is called a part program. In this application, the individual commands refer to positions of a cutting tool relative to the work table on which the work-part is held. The program is coded on a suitable medium for submission to the machine control unit.

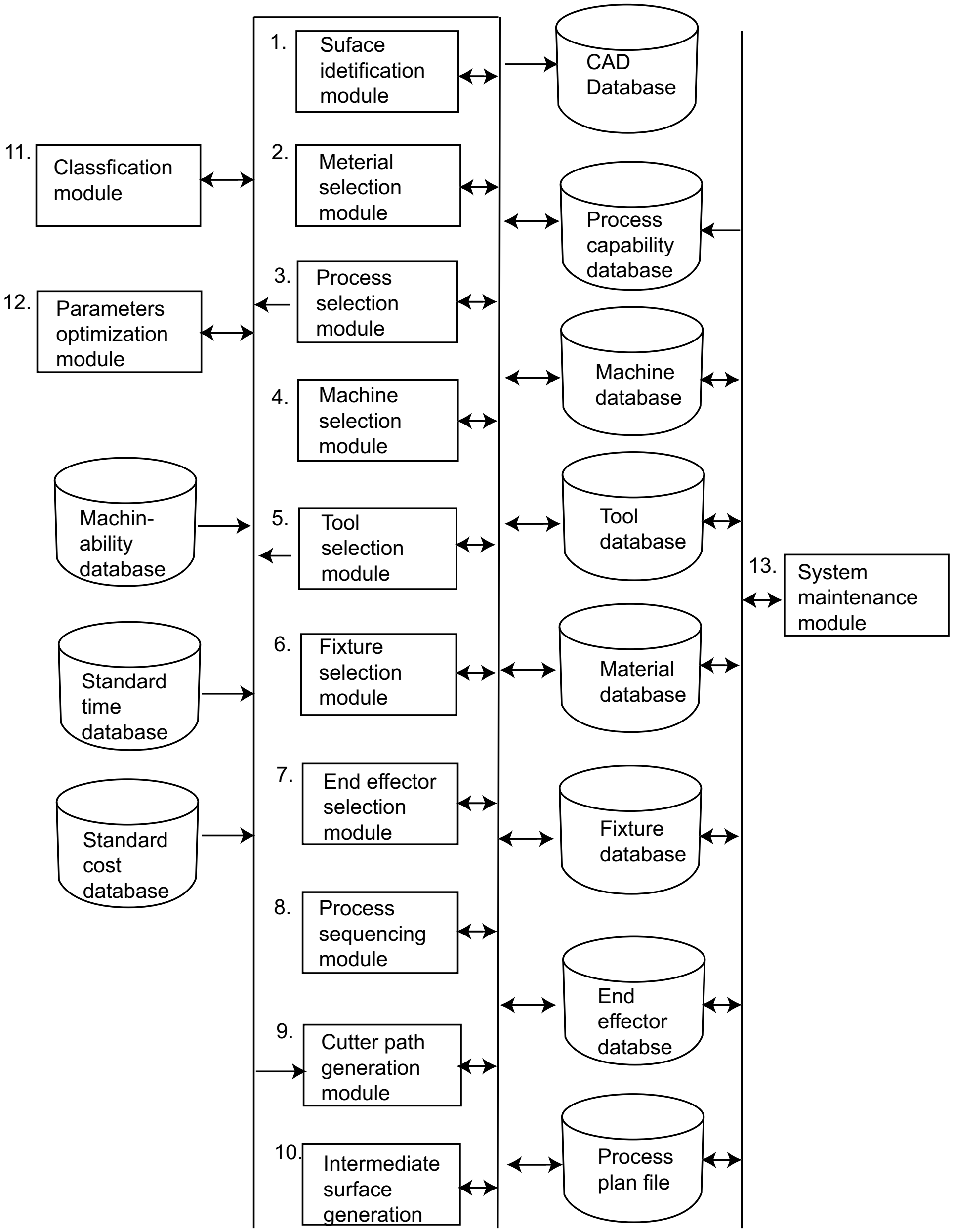

1. Suface idetification module
2. Meterial selection module
3. Process selection module
4. Machine selection module
5. Tool selection module
6. Fixture selection module
7. End effector selection module
8. Process sequencing module
9. Cutter path generation module
10. Intermediate surface generation
11. Classfication module
12. Parameters optimization module
13. System maintenance module
CAD Database
Process capability database
Machine database
Tool database
Material database
Fixture database
End effector databse
Process plan file
Machin-ability database
Standard time database
Standard cost database

Fig. 1. Process planning modules and databases

The machine control unit (MCU) consists of a micro computer and related control hardware that stores the program of instructions and executes it by converting each command into mechanical actions of the processing equipment, one command at a time. The MCU includes control system software, calculation algorithm, and translation software to convert the NC part program into a usable format for the MCU. Today, all MCUs are based on computer technology, hence computer numerical control (CNC) is referred to NC system.

The last component of NC system is the processing equipment. It accomplishes the processing steps to transform the starting work-part into a finished part. Its operation is controlled directly by the MCU, which in turn is driven by instructions contained in the part programming. In NC machines, the processing equipment consists of the worktable and spindle as well as the motors and controls to drive them.

There are different types of movement accomplished by MCU whose features are explained below. MCU systems for NC can be divided into two types.

1. point-to-point
2. continuous path

Point-to-point systems, also called positioning system, move the work table to a programmed location without considering the path taken to get to that location. Once the move has been completed, some processing action is accomplished by the work head at the location such as drilling or punching a hole.

Continuous path systems generally refer to systems that are capable of continuous simultaneous control of two or more axes. This provide control of the tool trajectory relative to work-part. In this case, the tool performs the process while the worktable is moving, thus enabling the system to generate angular surface, two dimensional curves, or three dimensional contours in the work-part. This control mode is required in milling operations. The term contouring is used when continuous path control is used for simultaneous control of two or more axes in machining operations. One of the important aspects of contouring is interpolation. The paths that a contouring-type NC system is require to generate often consists of circular arcs and other smooth nonlinear shapes. To cut along a curved path, the curve must be divided into a series of straight line segments that approximate the curve. The tool is commanded to machine each line segment in succession so that the machine surface closely matches the desired shape. The maximum error between the desired surface and the finished surface can be controlled by the length of the individual line segments which is one of the main tasks of this dissertation. We will discuss more in detail in following sections.

The computer's role in computer-aided part programming consists of the following tasks

1. input translation
2. arithmetic and cutter offset computations
3. editing
4. postprocessing

The first three tasks are carried out under the supervision of the language processing program. The fourth task, postprocessing, requires a separate computer program. The sequence and relationship of the tasks of the part programmer and the computer are portrayed in figure 2.

The input translation module converts the coded instructions contained in the program into computer-usable form, preparatory to further processing.

The arithmetic module consists of a set of subroutines to perform the mathematical computations required to define the part surface and generate the tool path, including compensation for cutter offset. The individual subroutines are called by the various statements used in the part programming language. The arithmetic computations are performed on the PROFIL file. The arithmetic module frees the programmer from the time consuming and error-prone geometry and trigonometry calculations to concentrate on issues related to work-part processing. The output of this module is a file called CLFILE, which stands for "cutter location(CL) file". This file consists mainly of tool path data.

In the editing phase, the CLFILE is edited, and a new file is generated called CLDATA. CLDATA provides readable data on cutter locations and machine tool operating commands. The machine tool commands can be converted to specific instructions during postprocessing. Some of the editing of CLFILE involves processing of special functions associated with the part programming language. The output of the editing phase is a part program in a format that can be postprocessed for the given machine tool on which the job will be accomplished.

The final task is postprocessing, in which the cutter location data and machining commands in the CLDATA file are converted into low-level code that can be interpreted by the NC controller for a specific machine tool. The output of postprocessing is a part program consisting of G-codes, x-, y-, and z-coordinates, S, F, M, and other functions in word address format[17].

B. Related Work and Literature Review

The goal of tool path generation is to approximate the part being processed with a number of curve that can be approximated by line segments. Ideally, every point on

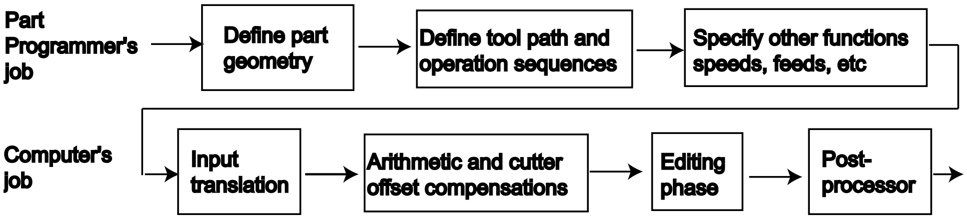

Part Programmer's job
Define part geometry
Define tool path and operation sequences
Specify other functions speeds, feeds, etc
Computer's job
Input translation
Arithmetic and cutter offset compensations
Editing phase
Post-processor

Fig. 2. Tasks in computer-aided part programming

the designed part should be a CC point so as to minimize machining error. However, it is very expensive and time consuming. In the process of tool path generation for a designed part, we distinguish between a tool path distribution strategy and a tool path calculation strategy[30]. We introduce each strategy in the following sections.

1. Tool Path Distribution Strategy

There are several strategies of distributing the tool path in the domain of the designed part. The goal of tool path distribution strategies is to span the entire designed part. The commonly used tool path distribution strategies are

1. zig-zag or raster curves
2. contour curves
3. spiral curves
4. space filling curves
5. sequential generated curves

In this section, we outline commonly used tool path distribution strategies.

1. Zig-zag Curves:

   The most commonly used tool path distribution strategies is the zig-zag strategy, due to the simple algorithm involved in calculating the spanning elements. This strategy involves filling the domain with parallel rays which are trimmed at the boundaries.

2. Contour Curves:

   The contour strategy is advantageous when the boundary contours are included as spanning elements, since a uniform boundary results. This strategy involves shrinking/expanding contours till the entire domain is spanned.

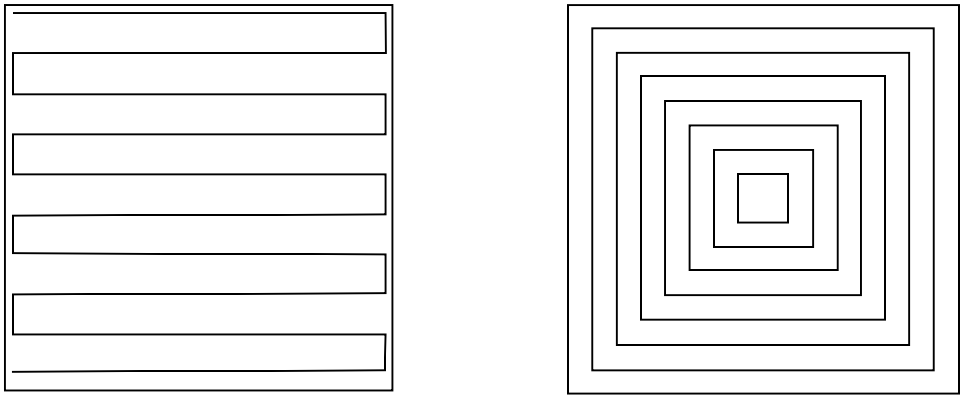
3. Space Filling Curves:

   The previous strategies give a directionality or lay to the surface finish on the manufactured part. The space filling strategy avoids the directionality by frequent changes in orientation of the spanning elements by means of recursive algorithms. The disadvantage of using the space filling strategy is that the overall length of the spanning elements, in general, is large, increasing manufacturing time. In addition, the number of short spanning elements is disadvantageous to the NC machine, since the tool is unable to accelerate to the specified feedrate.

4. Sequential Curves:

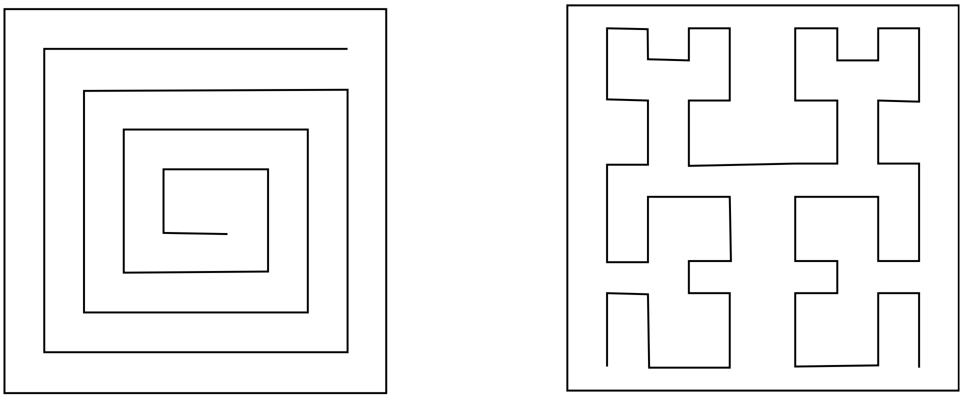
   This strategies involves sequentially generating spanning elements starting with a given initial spanning element. The sequential distribution strategy is advantageous due to the flexibility of generating various geometries of spanning elements. The disadvantage of this strategy is the complexity involved in calculating spanning elements.

Figure 3 schematically shows the above strategies of tool path distribution on a unit square by means of spanning elements. These spanning elements on a unit square are then calculated, such that they lie on the designed part, by means of tool path calculation methods discussed in the next section. It is in the tool path



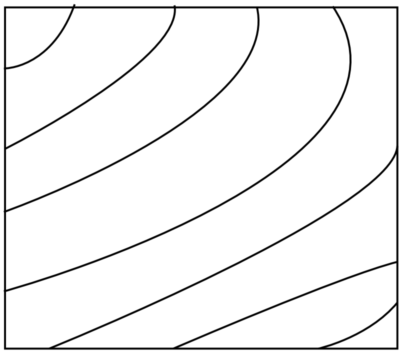
(i) Zig-Zag

(ii) Contour



(iii) Spiral

(iv) Space Filling



(v) Sequential

Fig. 3. Distribution strategies

calculation methods that the spacing between the tool paths and geometric accuracy is determined.

2. Tool Path Calculation Strategy

Tool path calculation is the method by which the trajectories of the tool is constrained to lie on the designed part. Commonly used tool path generations are as follows:

1. planar section curve
2. iso-parametric curve
3. offset curve
4. projection curve
5. constructive solid geometry (CSG)

Each method is discussed in this section.

1. Planar section curve:

   One of the earliest method of generating tool paths was driving the tool along curves which are intersections of user specified surfaces with the designed part. Gouging or undercutting of the manufactured part was prevented by defining check surfaces[3]. This was the basis of APT(automatically programmed tools) which was developed in the early sixties. It is shown in figure 4. Even today, generating tool paths along planar intersection curves is a very common method and improved planar section tool path generation methods that take into consideration maximum inaccuracy of the manufactured part have been developed[20]. Some of the disadvantages of using plane sections as tool path

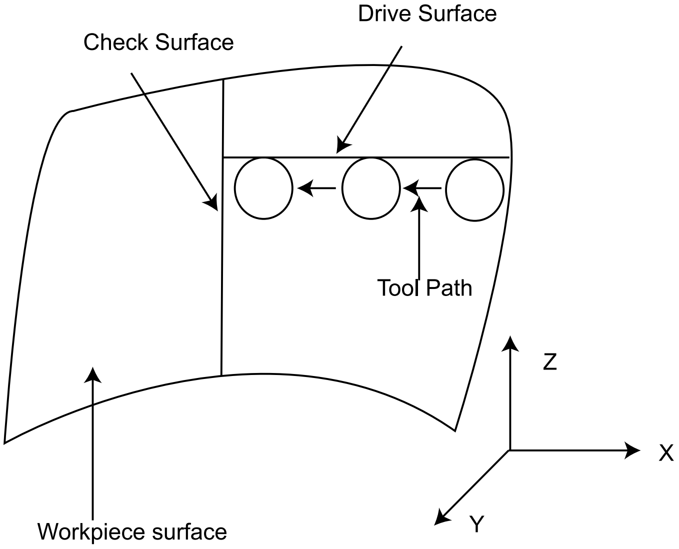


Fig. 4. Surfaces in APT

are (a) the overall length of tool path is very large, (b) the tool paths do not take into account the geometry of the designed part.

2. Iso-parametric curve:

   With the introduction of parametric patches such as B$\acute{e}$zier and B-spline surface in the late sixties and seventies, machining along iso-parametric curves was seen as an alternative to APT. Using iso-parametric curves, costly surface-surface intersection computations were avoided and it became easier for the user to specify tool paths[23][4].

3. Offset curve:

   Offset curves on the designed part was proposed to obtain a desired accuracy on the manufactured part. Of all the tool path generation techniques available, this technique has the potential of offering the user a direct control over the accuracy of the manufactured part[35][22].

4. Constructive solid geometry:

   In CAD/CAM, a variety of useful operations can be performed before the object is manufactured. We may wish to determine whether two objects interfere with each other, for example, whether a cutting tool will cut only the material it is intended to remove. There are several techniques to represent object as a solid[15]. In constructive solid geometry (CSG), simple primitives are combined by means of regularized Boolean set operators. Before we discuss the CSG, regularized Boolean set operators is explained. Regularized boolean set operation is one of the most intuitive technique to represent object using boolean set operation such as union, difference, and intersection. However, boolean set operations generate a solid, a plane, a line, a point as well as null object. It generates “dangling” boundary points, lines, or surfaces. In contrast, regularized boolean set operations can contain no “dangling” points, lines, as well as surfaces. As shown in figure 5 (a), the ordinary boolean intersection of two objects contains the intersection of the interior and boundary of each object with the interior and boundary of the other as shown in 5 (b). The regularized boolean intersection of two objects contains the intersection of their interior and the intersection of the interior of each with the boundary of the other, but only a subset of the intersection of their boundaries as shown in 5 (c). In CSG, an object is stored as a tree with operators at the internal nodes and simple primitives at the leaves. The general processing strategy is a depth-first tree walk to combine nodes from the leaves on up the tree. The complexity of this task depends on the representation in which the leaf objects at the tree’s root must actually be produced. As shown in figure 6, there are two objects, A and B. The figure is shown A $\cup$ B.

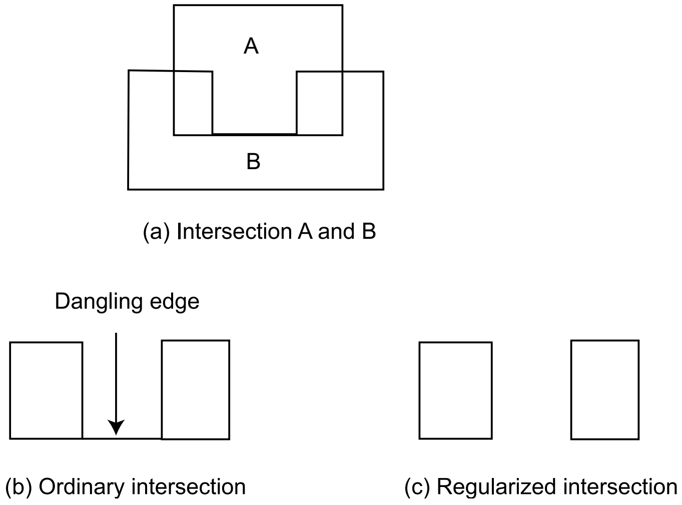


Fig. 5. Boolean intersection

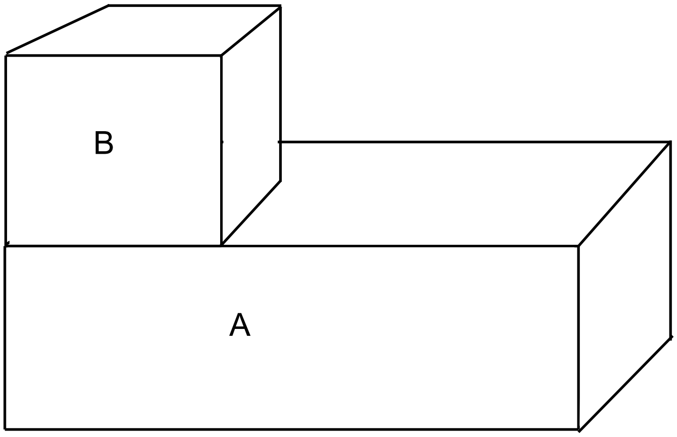


Fig. 6. CSG operation

3. Literature Review

The tool paths generation can be classified into 3 methods, iso-parameric, iso-planning, and iso-scallop height.

In iso-parametric method, Loney and Ozoy[23] and Broomhead and Edkins[4] have studied the tool path generation using iso-parametirc curve. They approximated tool path into surface using iso-parametric curve on the surfaces and the distance between adjacent tool paths, side-step (g), is calculated based on the worst case scallop height. The forward-step was calculated by “quick and dirty” method which is iterative and lengthy. Although this approach is the most widely used for the tool paths generation, it is computationally expensive because a search strategy was employed. In order to improve the computational efficiency, a quick estimate approach may be more desirable.

In iso-planar machining, the tool paths are along with the series of planes on the part. Borrow [3] and Huang and Oliver [20] have developed iso-planar NC tool-paths generation. A part consists of CSG is sectioned by a series of planes to obtain cutter contact points. Since the calculations are very tedious and sectioning plane is a non-trivial problem, this methods are not efficient to generate tool paths for free-form surfaces. Huang and Oliver[20] have also developed iso-planar tool path generation methods based on non-constant scallop height on the manufactured part. They implemented iso-planar machining on parametric surfaces and determine a machining error using a computational approach. However, calculations are also iterative and lengthy. A search strategy is also used to determine the forward-step, as compared to our method for the forward-step.

The last approach is iso-scallop machining. Suresh and Yang[35]and Lin and Koren[22] have studied scallop height machining. It was proposed to obtain a desired accuracy on the manufactured part. Using this method, the user can control the accuracy of the manufactured part. In these methods of tool path generation, the objective is to generate shortest overall length of tool paths with predetermined accuracy of the manufacturing part, tolerance and scallop height. Suh and Lee[34] have focused on spiral machining, "iso-offset zigzag machining". The spiral tool path as constant offset in the Euclidean space is also calculated by considering the worst case along the boundary profile.

Our method for tool path generation is similar to iso-scallop machining in that we generate tool paths based on predetermined tolerance and scallop height by using offset of an iso-parametric curve on the designed part. However, we propose a new and accurate method to generate tool paths on the designed part. There are problems to be solved in generation of tool paths. The first problem is cutting efficiency and cutting accuracy in milling operation. The second problem is computing efficiency, since surface calculations are generally iterative and lengthy. Last, true machining error will be verified. In order to obtain good accuracies we have used exact mathematical representation of the surface.

C. Research Motivation

A manufactured part is produced by a NC program containing a series of coded instruction called NC code which directly affect accuracy and cost of manufactured part, because accuracy and cost are proportional to the machining time. The coded instruction (specific command and numerical value) makes specific trajectories on the part being processed called tool path. There are two main tasks (main subjects of this research) in NC part programming. The first task is tool path generation. The second task is defining geometry of the part to be machined.

Milling operation is the primary machining process in the manufacturing of a part and divided by two stages. The first stage is rough stage, the second stage is finish stage. In rough stage, the part is machined in incremental layers and cutter removes most of the material on the surface so as to avoid damage of tool or/and machine. In finish stage, the surface is machined smoothly by approximating surface using line segments to get desired part with predetermined accuracy and shape. The tool paths in finish stage are important, since the tool path directly affect the accuracy and manufacturing time of the manufactured part.

In milling operation, a rotating spindle touches the surface at cutter contact point (CC) and moves to next CC point linearly, that is a curved path is approximated by a straight line segment as shown in figure 7(a). The accuracy of this linear approximation controlled by deviation is called tolerance. There is also an un-machined region between adjacent tool paths called scallop or cusp [figure 7(b)]. After the machining, a grinding operation is needed to remove scallop. However, the grinding operation to remove scollop between adjacent tool paths is very expensive and time consuming. The large scallops increase amount of machining time to smooth the machined surface. Therefore, appropriate tool path in finish stage is very important to reduce the amount of secondary processes such as grinding and/or polishing.

It is also important to generate tool path with less cutter contact point with given tolerance and scallop height that affect accuracy of work-part being processed since we assume that the more line segments, the more is the machining time. Each discrete line segment quantity is called a forward-step denoted as “s” in figure 7(a) and the maximum allowable deviation is referred to as the tolerance denoted as “e” in figure 7(a). Further, the distance between two adjacent tool paths called the side-step denoted as “g” in figure 7(b). The maximum allowable height of this scallop is called the scallop-height denoted as “h” in figure 7(b). The value of “e” and “h” are

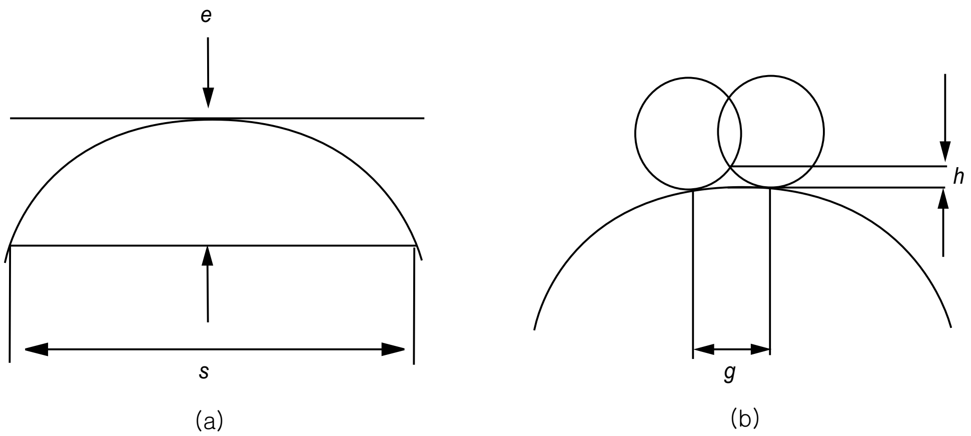


Fig. 7. Forward-step and side-step

determined in advance and then “s” and “g” are calculated from the value of “e” and “h”, respectively[35].

A ball-end mill cutter on a 3-axis milling machine is used to generate tool paths. When we design the NC part program, one of the main tasks is defining geometry of the part. A surface is the image of a sufficiently regular mapping of a set of points in a domain into a 3D space and expressed as

$$r(u, v) = (x(u, v), y(u, v), z(u, v)) \tag{1.1}$$

where u and v are the parameters of the surface. If the surface is defined on a bounded domain, generally $u = 1, v = 1$, it is called a surface patch. A composite surface may consist of single patches with predetermined continuity condition between patches. When the domain of a surface is the xy-plane of the given Cartesian coordinate system, the parametric surface equation given by equation (1.1) reduces to

$$Z = f(x, y) \tag{1.2}$$

It is in an explicit function form and is called a nonparametric surface equation. Now consider an analytic function

$$g(x, y, z) = 0 \tag{1.3}$$

which gives an implicit surface equation. If $g(x, y, z)$ is a linear function it becomes a plane equation; if $g(x, y, z)$ is a polynomial of degree 2 then it gives a quadratic surface. A surface represented by equation (1.3) is called an 'analytic surface', and one represented by equation (1.1) is called a 'sculptured surface'. A surface that consist of analytic surface elements (equation (1.2) or equation (1.3)) is called 'analytic compound surface', and one that consists of sculptured surface elements (equation (1.1)) is called a 'parametric compound surface'[8].

In this research, we develop a efficient approach to generate tool paths for NC machining of free-form surfaces. As such the primary goals of this research are

1. developing a new method for tool path generation in milling operations
2. verifying true machining error in milling operations

We use mathematical representation of tool and manufactured parts to make our algorithm efficient and reliable. Using this mathematical representation, we are able to determine reliable forward-step size. From there we develop a method for side-step size by studying the geometry of the tool and the differential geometry of the designed part. In this step, we reduce not only the size of the CL (Cutter Location) data file and machining time but also manufacturing data generated from machining, that is we reduce cost of data manipulation as well as storage. We then verify true machining errors by comparing the machined and designed surfaces using the point cloud method. As a result of this algorithm, the part can be machined in the least machining time while keeping up with predetermined tolerance and scallop height in the tool path.

D. Dissertation Outline

The remainder of this dissertation is as follows. Chapter II introduces mathematical background of curves and surfaces by which a designed manufactured part can be represented. This chapter includes mathematical formulation of curves and surfaces, their derivatives, and differential geometry of curve and surface. In Chapter III, we propose algorithms to generate tool paths using mathematical representation described in Chapter II. In this chapter, we develop new methods that calculate forward and side-step in milling operation for free-form surface. It also includes the methods that convert forward and side-step quantity from physical domain to parametric domain. In Chapter IV, the proposed approach is implemented; CL points are generated using the proposed approach and are subsequently machined using a 3-axis milling machine. This chapter also includes results of experiment. We verify tolerance and scallop height after machining by combining the designed and machined surface. We conclude with Chapter V, that summarizes this dissertation. In this chapter includes important contribution of this dissertation as well as recommendation for future research.

CHAPTER II

MATHEMATICAL REPRESENTATION

In this chapter, we introduce the mathematical preliminaries which help in developing a new method for tool path generation. The designed part can be represented by parametric curves and surfaces such as B*é*zier curves and surfaces. We generate B*é*zier curve using de Casteljau algorithm as described in section A. In sections B and C, blossoms and mathematical representation of B*é*zier curve are introduced. The B*é*zier surface patch is explained in section D. In last section (section E), we introduce the differential geometry of curves and surfaces on the designed part. The mathematical representation in this chapter is based on [12], [14], [1], [29], [10], and [15]. In the following sections, bold letters will represent vectorial quantities and unit vectors will be shown by a hat.

A. The De Casteljau Algorithm

de Casteljau algorithm:

Given: $\mathbf{b_0}, \mathbf{b_1}, \cdots, \mathbf{b_n} \in \mathbb{E}^3$ and $t \in \mathbb{R}$,

set

$$\mathbf{b_i^r}(t) = (1-t)\mathbf{b_i^{r-1}}(t) + t\mathbf{b_{i+1}^{r-1}}(t) \qquad \begin{cases} r = 1, \cdots, n \\ i = 0, \cdots, n-r \end{cases}$$

and $\mathbf{b_i^0}(t) = \mathbf{b_i}$. Then $\mathbf{b_0^n}(t)$ is the point with parameter value t on the B*é*zier Curve $\mathbf{b^n}$, hence $\mathbf{b^n}(t) = \mathbf{b_0^n}(t)$. The polygon $\mathbf{P}$ formed by $\mathbf{b_0}, \cdots, \mathbf{b_n}$ is called the B*é*zier polygon or control polygon of the curve $\mathbf{b^n}$. Similarly, the polygon vertices $\mathbf{b_i}$ are called control points. A cubic B*é*zier curve is illustrated in figure 8. The point, $\mathbf{b_0^3}$ can be obtained from iterative linear interpolation of control points. For example,

$\mathbf{b_0^1}(\frac{1}{2}) = \frac{1}{2}\mathbf{b^1}(\frac{1}{2}) + \frac{1}{2}\mathbf{b_1}(\frac{1}{2})$. In this example, $t = \frac{1}{2}$. This curve is the Bernstein-B$\acute{e}$zier approximation to the control polygon. The intermediate coefficient $\mathbf{b_i^r}(t)$ can be written in triangular array, the de Casteljau scheme. The cubic case can be expressed as

$$\begin{array}{llll} \mathbf{b_0} & & & \\ \mathbf{b_1} & \mathbf{b_0^1} & & \\ \mathbf{b_2} & \mathbf{b_1^1} & \mathbf{b_0^2} & \\ \mathbf{b_3} & \mathbf{b_2^1} & \mathbf{b_1^2} & \mathbf{b_0^3} \end{array} \tag{2.1}$$

In equation 2.1, $\mathbf{b_0}, \mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}$ are control points for the curve as shown in figure 8. $\mathbf{b_0^1}$ can be obtained linear interpolation between $\mathbf{b_0}$ and $\mathbf{b_1}$. If parameter $t = \frac{1}{2}$, the point $\mathbf{b_0^1}$ should be placed a half way between $\mathbf{b_0}$ and $\mathbf{b_1}$ as shown in figure 8. If parameter $t = 0$ and $t = 1$, the point, $\mathbf{b_0^1}$, can be $\mathbf{b_0}$ and $\mathbf{b_1}$, respectively. In similar manner, the control points $\mathbf{b_1^1}$ is linear interpolant between control points, $\mathbf{b_1}$ and $\mathbf{b_2}$. The last interpolant $\mathbf{b_0^3}$ can be obtained linear interpolation between $\mathbf{b_0^2}$ and $\mathbf{b_1^2}$.

B. Blossom of Polynomial

Blossoming is an elegant tool for studying polynomial curves expressed in the B$\acute{e}$zier form. It provides a simply way of determining the control polygon of the polynomial curve over an interval. The fundamental idea of this approach is that for any polynomial function $F(x)$ of degree n, there exists a unique function $f(u_1, u_2, \ldots, u_n)$ which is $n - affine$ (i.e.,$f(u_1, u_2, \ldots, u_n) = f(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(n)})$) for any permutation σ on $1, 2, ..., n$) and satisfies $f(x, x, \ldots, x) = F(x)$ for every $x \in R$. Function f is called the blossom of polar form of F.

If the polynomial F is expressed in the Bernstein basis, $B_i^n(x)$, over an interval

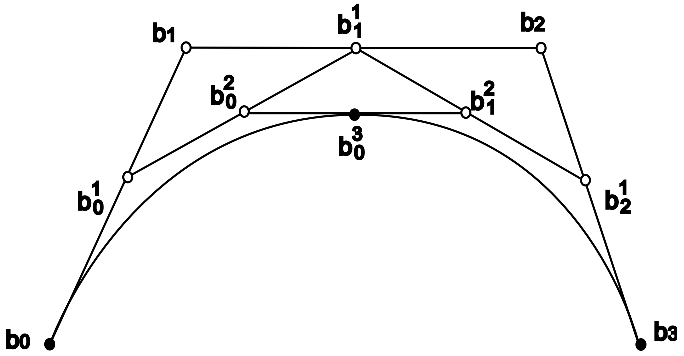


Fig. 8. The de Casteljau algorithm

$[a, b]$, i.e.,

$$F(x) = \sum_{i=0}^{n} p_i B_i^n(x)$$

where $B_i^n(x), i = 0, \ldots, n$, are the Bernstein polynomials defined by

$$B_i^n(x) = C_n^i \alpha(x)^{n-i} \beta(x)^i$$

and $\alpha(x), \beta(x)$ the barycentric coordinates of x with respect to a and b, i.e.,

$$\alpha(x) = (b - x)/(b - a), \beta(x) = (x - a)/(x - b)$$

then the value of p_i is equal to $f(\underbrace{a, \ldots, a}_{n-i}, \underbrace{b, \ldots, b}_{i})$, where f is the blossom of the polynomial F. The points $(a + \frac{i(b-a)}{n}, p_i), i = 0, 1, \ldots, n$, are called the B$\acute{e}$zier points of F with respect to the interval $[a, b]$. The linear interpolant of the B$\acute{e}$zier points is also called the control polygon of the polynomial F with respect to the interval $[a, b]$. In this dissertation, we also called the sequence of real values $(p_0, p_1, \ldots, p_n)$ the control polygon of the polynomial F with respect to interval [a,b]

Since the blossoms are closely related to the de Casteljau algorithm, the blossoms are also represented in triangular array.

$$\begin{array}{llll}
\mathbf{b_0} & & & \\
\mathbf{b_1} & \mathbf{b_0^1}[t_1] & & \\
\mathbf{b_2} & \mathbf{b_1^1}[t_1] & \mathbf{b_0^2}[t_1, t_2] & \\
\mathbf{b_3} & \mathbf{b_2^1}[t_1] & \mathbf{b_1^2}[t_1, t_2] & \mathbf{b_0^3}[t_1, t_2, t_3]
\end{array} \tag{2.2}$$

If we set all three values equal, $t = t_1 = t_2 = t_3$, the original curve can be recovered. The first and last points of curve can be expressed using blossoms that is $\mathbf{b}[0, 0, 0] = \mathbf{b_0}$ and $\mathbf{b}[1, 1, 1] = \mathbf{b_3}$. If $[t_1, t_2, t_3] = [0, 1, 1]$, the triangular array equation 2.2 can be simplified to the following triangular array.

$$\begin{array}{llll}
\mathbf{b_0} & & & \\
\mathbf{b_1} & \mathbf{b_0} & & \\
\mathbf{b_2} & \mathbf{b_1} & \mathbf{b_1} & \\
\mathbf{b_3} & \mathbf{b_2} & \mathbf{b_2} & \mathbf{b_2}[0, 1, 1]
\end{array} \tag{2.3}$$

Thus we can find original Bézier points using the blossoms at arguments consisting only $0's$ and $1's$. Therefore, the de Casteljau algorithm can be expressed as following triangular array

$$\begin{array}{llll}
\mathbf{b_0} = \mathbf{b}[0, 0, 0] & & & \\
\mathbf{b_1} = \mathbf{b}[0, 0, 1] & \mathbf{b_0^1}[0, 0, t] & & \\
\mathbf{b_2} = \mathbf{b}[0, 1, 1] & \mathbf{b_1^1}[0, t, 1] & \mathbf{b_0^2}[0, t, t] & \\
\mathbf{b_3} = \mathbf{b}[1, 1, 1] & \mathbf{b_2^1}[t, 1, 1] & \mathbf{b_1^2}[t, t, 1] & \mathbf{b_0^3}[t, t, t]
\end{array} \tag{2.4}$$

Thus, we can express the Bézier points using these blossom value:

$$\mathbf{b_i} = \mathbf{b}[0^{<n-i>}, 1^{<i>}] \tag{2.5}$$

where $0^{<n-i>}$ means that 0 appears $n - i$ times in argument. So, the de Casteljau algorithm can be expressed as

$$\mathbf{b}[0^{<n-r-i>}, t^{<r>}, 1^{<i>}] =$$

$$(1-t)\mathbf{b}[0^{<n-r-i+1>}, t^{<r-1>}, 1^{<i>}] + t\mathbf{b}[0^{<n-r-i>}, t^{<r-1>}, 1^{<i>}] \tag{2.6}$$

The point on the curve is $\mathbf{b}[t^{<n>}]$.

We represented B$\acute{e}$zier curve defined by interval $[0, 1]$ so far. However,we can also represented B$\acute{e}$zier defined by interval over $[a, b]$. In this interval, B$\acute{e}$zier points are expressed by the following equation:

$$\mathbf{b_i} = \mathbf{b}[a^{<n-i>}, b^{<i>}] \tag{2.7}$$

Figure 9 shows the B$\acute{e}$zier curve defined over interval $[a, b]$.

C. B$\acute{e}$zier Curve

1. Mathematical Representation of B$\acute{e}$zier Curve

B$\acute{e}$zier curves can be defined by a recursive algorithm, which is how de Casteljau first developed them. We will express B$\acute{e}$zier curves in terms of Bernstein polynomials

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

One of their important properties is that they satisfy the following recursion:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

with

$$B_0^0(t) \equiv 1$$

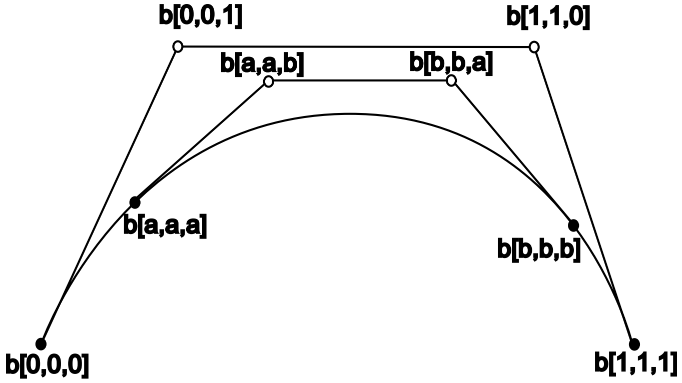


Fig. 9. Bézier curve over interval $[a, b]$

and

$$B_j^n(t) \equiv 0 \qquad for \quad j \notin 0, 1, \ldots, n$$

The proof is simple:

$$\begin{aligned} B_i^n(t) &= \tbinom{n}{i} t^i (1-t)^{n-i} \\ &= \tbinom{n-1}{i} t^i (1-t)^{n-i} + \tbinom{n-1}{i-1} t^i (1-t)^{n-i} \\ &= (1-t) B_i^{n-1}(t) + t B_{i-1}^{n-1}(t) \end{aligned}$$

Another important property is that Bernstein polynomials form a partition of unity:

$$\sum_{j=0}^{n} B_j^n(t) \equiv 1$$

A Bézier curve may be written as $\mathbf{b}[t^{<n>}]$ in blossom form. Since $t = (1-t)\cdot 0 + t \cdot 1$, the blossom may be expressed as $\mathbf{b}[((1-t)\cdot 0 + t \cdot 1)^{<n>}]$, and now the Leibniz formula directly yields,

$$\mathbf{b}(t) = \mathbf{b}[t^{<n>}] = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) \tag{2.8}$$

since $\mathbf{b}_i = \mathbf{b}[0^{<n-i>}, 1^{<i>}]$.

Similarly, the intermediate de Casteljau point $\mathbf{b}_i^r$ can be expressed in terms of Bernstein polynomials of degree r:

$$\mathbf{b}_i^r(t) = \sum_{j=0}^{r} \mathbf{b}_{i+j} B_j^r(t) \tag{2.9}$$

this follows directly from

$$\mathbf{b}_i^r(t) = \mathbf{b}[0^{n-r-i}, t^r, 1^i]$$

and the Leibniz formula.

With the intermediate points $\mathbf{b}_i^r$ at hand, we can write a Bézier curve in the form

$$\mathbf{b}^n(t) = \sum_{i=0}^{n-r} \mathbf{b}_i^r(t) B_i^{n-r}(t) \tag{2.10}$$

This is to be interpreted as follows: first, compute r levels of the de Casteljau algorithm with respect to t. Then, interpret the resulting points $\mathbf{b}_i^r(t)$ as control points of a Bézier curve of degree $n - r$ and evaluate it at t.

2. The Derivative of a Bézier Curve

We start with an identity, closely resembling Leibniz's formula for derivatives. Let t be on the real line, and let $\vec{v}$ be a vector in the associated 1D linear space. Then

$$\mathbf{b}[(t + \vec{v})^{<n>}] = \sum_{i=0}^{n} \binom{n}{i} \mathbf{b}[t^{n-i}, \vec{v}^{<i>}] \tag{2.11}$$

This is an immediate consequence of the Leibniz formula.

The derivative of a curve $\mathbf{x}(t)$ is typically defined as

$$\frac{d\mathbf{x}(t)}{dt} = \lim_{h \to 0} \frac{1}{h} [\mathbf{x}(t + h) - \mathbf{x}(t)]$$

We will be a little more precise and observe that t is a 1D point, whereas h is a 1D vector. We thus denoted it by $\vec{h}$ and obtain

$$\frac{d\mathbf{x}(t)}{dt} = \lim_{\vec{h}\to\vec{0}} \frac{1}{\left|\vec{h}\right|}[\mathbf{x}(t+\vec{h}) - \mathbf{x}(t)]$$

Then, we have

$$\frac{d\mathbf{x}(t)}{dt} = \lim_{\vec{h}\to\vec{0}} \frac{1}{\left|\vec{h}\right|}\left[\sum_{i=0}^{n}\binom{n}{i}\mathbf{b}[t^{<n-i>}, \vec{h}^{<i>}] - \mathbf{b}[t^{<n>}]\right] \tag{2.12}$$

For $i = 0$, two terms $\mathbf{b}[t^n]$ cancel. We expand the rest and factor in the term $\left|\vec{h}\right|$:

$$\frac{d\mathbf{x}(t)}{dt} = \lim_{\vec{h}\to\vec{0}}\left(n\mathbf{b}\left[t^{n-1}, \frac{\vec{h}}{\left|\vec{h}\right|}\right] + \binom{n}{2}\mathbf{b}\left[t^{<n-2>}, \frac{\vec{h}}{\left|\vec{h}\right|}, \vec{h}\right] + \cdots\right)$$

We observe that $\frac{\vec{h}}{|\vec{h}|} = \vec{1}$. Taking the limit annihilates all other terms containing $\vec{h}$, and we thus have

$$\frac{d\mathbf{x}(t)}{dt} = n\mathbf{b}[t^{<n-1>}, \vec{1}] \tag{2.13}$$

From now on, we use the expression $\dot{\mathbf{x}}(t)$ for the first derivative. This has two possible interpretations. For the first one, we perform a de Castejau step with respect to $\vec{1}$, and then n-1 steps with respect to t; as an equation:

$$\dot{\mathbf{x}}(t) = n\sum_{j=0}^{n-1}(\mathbf{b}_{j+1} - \mathbf{b}_j)B_j^{n-1}(t) \tag{2.14}$$

This can be simplified somewhat by the introduction of the forward difference operator Δ:

$$\Delta\mathbf{b}_j = \mathbf{b}_{j+1} - \mathbf{b}_j \tag{2.15}$$

We now have for the derivative of a B*é*zier curve:

$$\dot{\mathbf{x}}(t) = n\sum_{j=0}^{n-1} \Delta\mathbf{b}_j B_j^{n-1}(t); \quad \Delta\mathbf{b}_j \in \mathbb{R}^3 \tag{2.16}$$

For a second interpretation of (2.13), we first perform n-1 steps of the de Casteljau algorithm, resulting in the two points $\mathbf{b}_1^{n-1}(t)$ and $\mathbf{b}_0^{n-1}(t)$. Now performing one step with respect to $\vec{1}$ yields (after multiplication by n):

$$\dot{\mathbf{x}}(t) = n\big(\mathbf{b}_1^{n-1}(t) - \mathbf{b}_0^{n-1}(t)\big) \tag{2.17}$$

Higher derivatives follows the same pattern:

$$\frac{d^r\mathbf{x}(t)}{dt^r} = \frac{n}{(n-r)!}\mathbf{b}[t^{<n-r>}, \vec{1}^{<r>}] \tag{2.18}$$

To compute the derivatives from the B*é*zier points, we first generalize the forward difference operator(2.15): the iterated forward difference operator Δ^r is defined by

$$\Delta^r\mathbf{b}_j = \Delta^{r-1}\mathbf{b}_{j+1} - \Delta^{r-1}\mathbf{b}_j \tag{2.19}$$

We list a few examples:

$$\begin{aligned}
\Delta^0\mathbf{b}_i &= \mathbf{b}_i \\
\Delta^1\mathbf{b}_i &= \mathbf{b}_{i+1} - \mathbf{b}_i \\
\Delta^2\mathbf{b}_i &= \mathbf{b}_{i+2} - 2\mathbf{b}_{i+1} + \mathbf{b}_i \\
\Delta^3\mathbf{b}_i &= \mathbf{b}_{i+3} - 3\mathbf{b}_{i+2} + 3\mathbf{b}_{i+1} - \mathbf{b}_i
\end{aligned}$$

The factors on the right-hand sides are binomial coefficients, forming a Pascal like triangle. This pattern holds in general:

$$\Delta^r\mathbf{b}_i = \sum_{j=0}^{r}\binom{r}{j}(-1)^{r-j}\mathbf{b}_{i+j} \tag{2.20}$$

The r^{th} derivative of a Bézier curve is now given by

$$\frac{d^r}{dt^r}\mathbf{b}^n(t) = \frac{n!}{(n-r)!}\sum_{j=0}^{n-r} \Delta^r \mathbf{b}_j B_j^{n-r}(t) \tag{2.21}$$

Two important special cases of (2.21) are given by $t = 0$ and $t = 1$.

$$\frac{d^r}{dt^r}\mathbf{b}^n(0) = \frac{n!}{(n-r)!}\Delta^r \mathbf{b}_0 \tag{2.22}$$

and

$$\frac{d^r}{dt^r}\mathbf{b}^n(1) = \frac{n!}{(n-r)!}\Delta^r \mathbf{b}_{n-r} \tag{2.23}$$

Thus the r^{th} derivative of a Bézier curve at an end point depends only on the $r + 1$ Bézier points near (and including) that end point. For $r = 0$, we get the already established property of endpoint interpolation. The case $r = 1$ states that $\mathbf{b}_0$ and $\mathbf{b}_1$ define the tangent at $t = 0$, provided they are distinct. Similarly, $\mathbf{b}_{n-1}$ and $\mathbf{b}_n$ determine the tangent at $t = 1$.

D. Bézier Surface Patch

1. Mathematical Representation of Bézier Surface

The definition of a surface is "a surface is the locus of a curve that is moving through space and thereby changing its shape". We now formalize this intuitive concept in order to arrive at a mathematical description of a surface. First, we assume that the moving curve is a Bézier curve of constant degree m. At any time, the moving curve is then determined by a set of control points. Each original control point moves through space on a curve. Our next assumption is that this curve is also a Bézier curve, and that the curve on which the control points move are all of the same degree. This can be formalized as follows: Let the initial curve be a Bézier curve of degree m:

$$\mathbf{b}^m(u) = \sum_{i=0}^{m} \mathbf{b}_i B_i^m(u).$$

Let each $\mathbf{b}_i$ traverse a B$\acute{e}$zier curve of degree n:

$$\mathbf{b}_i = \mathbf{b}_i(v) = \sum_{j=0}^{n} \mathbf{b}_{i,j} B_j^n(v).$$

We can now combine these two equations and obtain the point $\mathbf{b}^{m,n}(u,v)$ on the surface $\mathbf{b}^{m,n}$ as

$$\mathbf{b}^{m,n}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{i,j} B_i^m(u) B_j^n(v). \tag{2.24}$$

With this notation, the original curve $\mathbf{b}^m(u)$ now has B$\acute{e}$zier points $\mathbf{b}_{i,0}; i = 0, \ldots, m$.

An arbitrary iso-parametric curve $\hat{v} = const$ of a B$\acute{e}$zier surface $\mathbf{b}^{m,n}$ is a B$\acute{e}$zier curve of degree m in u, and its $m+1$ B$\acute{e}$zier points are obtained by evaluating all column of the control net at $\hat{v} = const$. As a formula:

$$\mathbf{b}_{i,0}^{0,n}(\hat{v}) = \sum_{j=0}^{n} \mathbf{b}_{ij} B_j^n(\hat{v}); \qquad i = 0, \ldots, m.$$

Iso-parametric curves $\hat{u} = const$ are treated analogously.

2. The Derivative of a B$\acute{e}$zier Surface

In the curve case, taking derivatives was accomplished by differencing the control points. A partial derivative is the tangent vector of an iso-parametric curve. It can be found by a straightforward calculation:

The derivatives are partial derivatives $\frac{\partial}{\partial u}$ or $\frac{\partial}{\partial v}$. This partial derivative is tangent vector of iso-parametric curve. It can be calculated by following Equation.

$$\frac{\partial}{\partial u}\mathbf{b}^{m,n}(u,v) = \sum_{j=0}^{n} \left[\frac{\partial}{\partial u} \sum_{i=0}^{m} \mathbf{b}_{i,j} B_i^m(u) \right] B_j^n(v). \tag{2.25}$$

where m is degree of B$\acute{e}$zier curve in u, and n is degree of B$\acute{e}$zier curve in v.

We can rewrite Equation 2.25 in terms of forward difference operator Δ.

$$\frac{\partial}{\partial u}\mathbf{b}^{m,n}(u,v) = m\sum_{j=0}^{n}\sum_{i=0}^{m-1}\Delta^{1,0}\mathbf{b}_{i,j}B_i^{m-1}(u)B_j^{n}(v). \tag{2.26}$$

The superscript $(1,0)$ of difference operator means that differencing is performed only on the first subscript. $\Delta^{1,0}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}$. If we take partial derivative in terms of v, the difference operator act only on the second subscript. $\Delta^{0,1}\mathbf{b}_{i,j} = \mathbf{b}_{i,j+1} - \mathbf{b}_{i,j}$. Therefore, partial derivative in v can be written by following equation.

$$\frac{\partial}{\partial v}\mathbf{b}^{m,n}(u,v) = n\sum_{i=0}^{m}\sum_{j=0}^{n-1}\Delta^{0,1}\mathbf{b}_{i,j}B_j^{n-1}(v)B_i^{m}(u). \tag{2.27}$$

Therefore, we can write the formulas for higher-order partial:

$$\frac{\partial^r}{\partial u^r}\mathbf{b}^{m,n}(u,v) = \frac{m!}{(m-r)!}\sum_{j=0}^{n}\sum_{i=0}^{m-r}\Delta^{r,0}\mathbf{b}_{i,j}B_i^{m-r}(u)B_j^{n}(v). \tag{2.28}$$

Where the difference operator $\Delta^{r,0}\mathbf{b}_{i,j} = \Delta^{r-1,0}\mathbf{b}_{i+1,j} - \Delta^{r-1,0}\mathbf{b}_{i,j}$ and

$$\frac{\partial^s}{\partial v^s}\mathbf{b}^{m,n}(u,v) = \frac{n!}{(n-s)!}\sum_{i=0}^{m}\sum_{j=0}^{n-s}\Delta^{0,s}\mathbf{b}_{i,j}B_j^{n-s}(v)B_i^{m}(u). \tag{2.29}$$

Where the difference operator $\Delta^{0,s}\mathbf{b}_{i,j} = \Delta^{0,s-1}\mathbf{b}_{i,j+1} - \Delta^{0,s-1}\mathbf{b}_{i,j}$

We can write down the mixed partial of arbitrary order:

$$\frac{\partial^{r+s}}{\partial u^r \partial v^s}\mathbf{b}^{m,n}(u,v) = \frac{m!n!}{(m-r)!(n-s)!}\sum_{i=0}^{m-r}\sum_{j=0}^{n-s}\Delta^{r,s}\mathbf{b}_{i,j}B_i^{m-r}(u)B_j^{n-s}(v). \tag{2.30}$$

E. Differential Geometry

Differential geometry is the description of local curve and surface properties (such as curvature). Specifically, we discuss differential geometry preliminaries which help in quantifying the local geometry and curves and surfaces of the designed part.

1. Differential Geometry of Curves

A curve in $\mathbb{E}^3$ is given by the parametric representation

$$\mathbf{x} = \mathbf{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \quad t \in [a, b] \subset \mathbb{R} \tag{2.31}$$

Any point on a curve is obtained by specifying a value for the parameter t. Where its cartesian coordinates x, y, and z are differentiable functions of t. To avoid potential problems concerning the parametrization of the curve, we shall assume that

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} \neq 0, \quad t \in [a, b] \tag{2.32}$$

Where dots denote derivatives with respect to t. Such a parametrization is called regular. Any point on a space curve x is obtained by specifying a value for the parameter t. The length of the curve, estimated between any two values of the parameter t is referred to as the arc length. The arc length can be estimated as follows:

$$s(t) = \int_a^b \sqrt{\left(\frac{dx^2}{dt} + \frac{dy^2}{dt} + \frac{dz^2}{dt}\right) du} \tag{2.33}$$

Alternatively, the square of the differential arc length can be represented as follows:

$$ds^2 = dx \cdot dx = dx^2 + dy^2 + dz^2 \tag{2.34}$$

We can obtain a local cartesian (orthogonal) system with origin x and axes $\mathbf{t}, \mathbf{m}, \mathbf{b}$ as shown in figure 10.

$$\mathbf{t} = \frac{\dot{\mathbf{x}}}{|\dot{\mathbf{x}}|}, \qquad \mathbf{m} = \mathbf{b} \wedge \mathbf{t}, \qquad \mathbf{b} = \frac{\dot{\mathbf{x}} \wedge \ddot{\mathbf{x}}}{|\dot{\mathbf{x}} \wedge \ddot{\mathbf{x}}|} \tag{2.35}$$

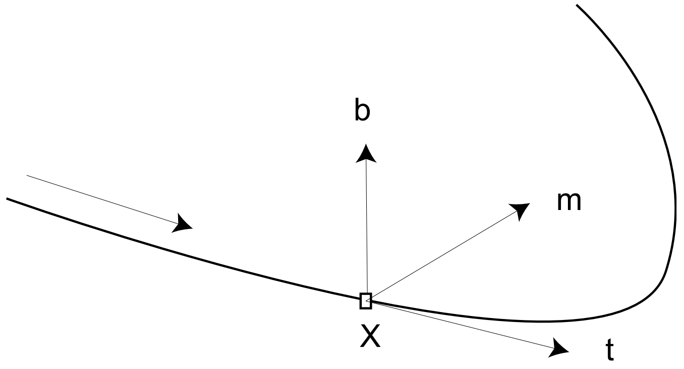


Fig. 10. Frenet frame

where $\wedge$ denotes the cross product.

The vectors,$\mathbf{t}, \mathbf{m}, \mathbf{b}$ are the tangent, normal, binormal vectors. The frame(or trihedron) $\mathbf{t}, \mathbf{m}, \mathbf{b}$ is called the Frenet frame; it varies its orientation as t traces out the curve.

The tangent can be geometrically visualized as a line that passes through two infinitesimally close points on the curve. The plane containing the tangent and the normal is referred to as the osculating plane and can be visualized as the plane containing three infinitesimally close points on the curve $\mathbf{x}(t)$. The plane containing the normal and the binormal is called the normal plane and the plane containing the binormal and tangent vectors is called the rectifying plane. The scalar quantity $k(s)$ is the curvature of the curve $\mathbf{x}(t)$ at the point of evaluation. The curvature of the curve measures the rate of change of the tangent vector along the curve. The radius of curvature (inverse of curvature) is the radius of a circle that passes through three infinitesimally close points on the curve. Figure 11 schematically shows the above mentioned planes.

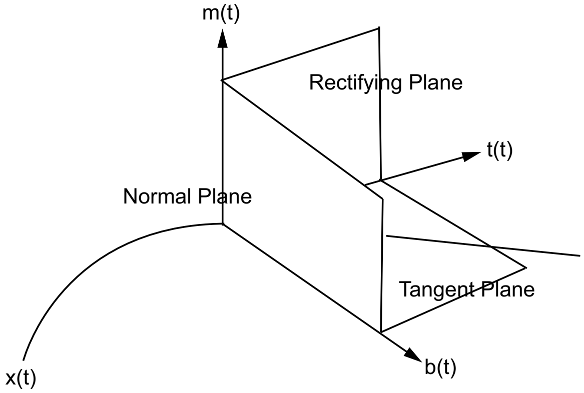


Fig. 11. Curve frame

2. Differential Geometry of Surfaces

A surface may be given by an implicit form $f(x, y, z) = 0$ or by its parametric form. The cartesian coordinates of a parametric surface x in terms of two parameters as shown below.

$$\mathbf{r}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} \in [a,b] \subset \mathbb{R}^2 \tag{2.36}$$

Points on the surface $\mathbf{S}$ are obtained by varying the parameters u and v. Where the cartesian coordinates x, y, z of a surface point are differentiable functions of parameters u and v and $[a, b]$ denotes a rectangle in the $u - v$ plane. As in the case of a curve, the surface $\mathbf{S}$ also has a frame consisting of three orthogonal unit vectors. This frame is referred to as the surface frame and is represented as follows:

$$\mathbf{F}(u,v) = \begin{bmatrix} \hat{t}_1(u,v) \\ \hat{t}_2(u,v) \\ \hat{n}(u,v) \end{bmatrix} \tag{2.37}$$

The vectors $\hat{\mathbf{t}}_{\mathbf{1}}$ and $\hat{\mathbf{t}}_{\mathbf{2}}$ represent the unit vectors and the vector $\hat{\mathbf{n}}$ represents the unit normal vector of the surface S at given values of the parameters u and v as shown below. The surface normal $\hat{\mathbf{n}}$ at an arbitrary point $\mathbf{P}(u, v)$ is expressed as

$$\hat{\mathbf{n}} = \frac{\mathbf{X}_u \times \mathbf{X}_v}{|\mathbf{X}_u \times \mathbf{X}_v|} \tag{2.38}$$

Given an embedded curve $\mathbf{P}(u(t), v(t))$ on the designed surface $\mathbf{S}$ passing through a point $\mathbf{P_p}$ represents a parametric curve on the designed surface. The square of the differential arc length at $\mathbf{P_p}$, along curve $\mathbf{P}$, on the surface can be expressed as follows.

$$I = P^t \cdot P^t = E\frac{du}{dt}\frac{du}{dt} + 2F\frac{du}{dt}\frac{dv}{dt} + G\frac{dv}{dt}\frac{dv}{dt} \tag{2.39}$$

is referred to as the first fundamental-form, where "t" is the independent variable along the path and

$$E = P^u \cdot P^u; \quad F = P^u \cdot P^v; \quad G = P^v \cdot P^v \tag{2.40}$$

are the coefficients of the first fundamental form. The first fundamental form provides us with information of metric properties of the surface such as measurement of lengths, areas and angles. The quadratic

$$II = L\frac{du}{dt}\frac{du}{dt} + 2M\frac{du}{dt}\frac{dv}{dt} + N\frac{dv}{dt}\frac{dv}{dt} \tag{2.41}$$

is referred to as the second fundamental-form where

$$L = P^{uu} \cdot n, \quad M = P^{uv} \cdot n, \quad N = P^{vv} \cdot n \tag{2.42}$$

are the coefficients of the second fundamental form. The second fundamental form II is of interest in deriving expression for curvature of a surface. The second fundamental form provides measurement of change in the unit normal along the given curve on the surface.

CHAPTER III

PROPOSED APPROACH

In this chapter, we develop new methods for generating tool paths for free-form surfaces that can be represented by parametric curves and surfaces. In section A, we discuss how the CC point is on a tool and how the CL point is a reference point by which the tool moves along a surface in removing material. In section B, we discuss the conceptual approach through which we outline each step for generating tool path. Sections C, D, and F are devoted to the new methods for tool paths generation. In section C, we introduce the algorithm used to calculate forward-step size. In section D, the algorithm used to calculate side-step size is introduced. In section F, we introduce a method that converts forward and side-step size from the physical domain to parametric domain.

A. Tool Characteristics

A proper tool must be used in machining that covers all the chip-making processes (such as milling, drilling, turning, and boring). When the degree of complexity for machining is increased, the number of tool selections and work-part materials also increases. There are several factors to be considered in machining: first, selecting tool material and geometry involves material, shape, size of the work-part, design requirement, and operation type (roughing or finishing); second, deciding on machining conditions such as feed rate, spindle speed, and depth of cut. The metal cutting process is the removal of work piece materials to obtain a designed part. In this dissertation, we assume that the operation is milling with a ball-end tool. To reduce machining error we have considered two kinds of points on the tool as shown in figure

12. One is CC the point and the other is the CL point.

1. Cutter Contact Point and Cutter Location Point

The cutter contact(CC) point can be any point on the tool path where there is instantaneous contact between the tool and the manufactured part. The cutter location (CL) point is a fixed point which the machine drive tool references in moving along the tool path. Ideally, the CC point should lie on the designed part so as to minimize manufacturing errors. However, as shown in figure 12, the CC point is not always located at the center point of the tool nose when working on a free-form surface. In case of down-hill or up-hill machining, CC points would be to the left or right side of the tip of the the tool nose. To reduce machining errors, CC points can be converted to CL points in order to compensate. The CL point is always placed along the normal direction of the point on the surface (as shown in figure 12). Thus, a CL point can be obtained from a CC point and the surface normal of the point.

2. Conversion of CC to CL

The surface normal $\mathbf{n}$ at an arbitrary point $\mathbf{P}(u, v)$ is expressed as

$$\mathbf{n} = \frac{\mathbf{S}_u \times \mathbf{S}_v}{|\mathbf{S}_u \times \mathbf{S}_v|} \tag{3.1}$$

where $\mathbf{S}_u$ and $\mathbf{S}_v$ are the derivatives along the u and v directions on surface $\mathbf{S}$ at point $\mathbf{P}$. Assuming that the cutter axis $\mathbf{T}$ is parallel to the Z-axis, let $\mathbf{T}$ = (0,0,1). If the surface normal vector $\mathbf{n}$ points to the -Z direction, the surface cannot be machined with a 3-axis milling machine. As such, it is assumed in this research that all the surface normal vectors point to the +Z direction, i.e $(\mathbf{T} \cdot \mathbf{n}) > 0$.

For a ball-end mill, the CL point is the point on the offset surface of the workpiece,

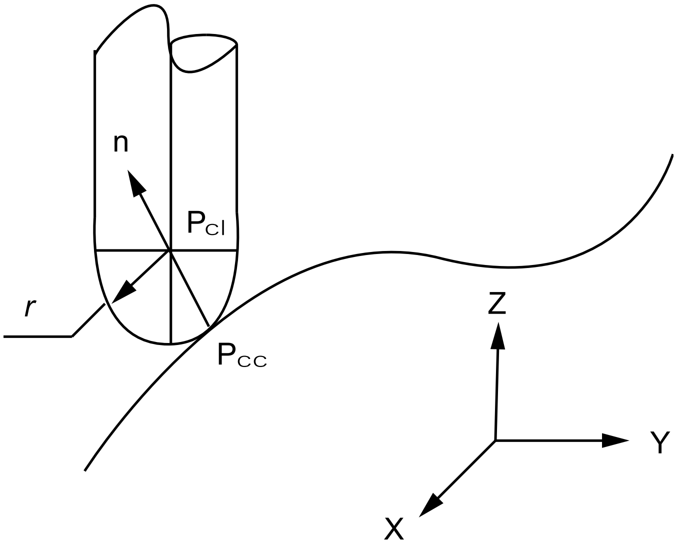


Fig. 12. CC and CL point

while the CC point is the point on the cutter that contacts the work piece surface($\mathbf{P}_{cc}$) (figure 12). Let ($\mathbf{P}_{cc}$) be the CC point, r be the radius of the ball-end mill, the CL point ($\mathbf{P}_{cl}$) is given by

$$\mathbf{P}_{cl} = \mathbf{P}_{cc} + r\mathbf{n} \tag{3.2}$$

B. Overall Conceptual Approach

In this section, we summarize the approaches that can be used to generate tool paths. The proposed approach in this research is derived from an overall conceptual approach as explained briefly in this section. The overall conceptual approach is also summarized in the flowchart shown in figure 13.

1. Define the designed surface in the $u, v - plane$.

   A designed part can be represented using Bézier curves and surfaces as described in the previous chapter. In parametric surface, holding one parameter

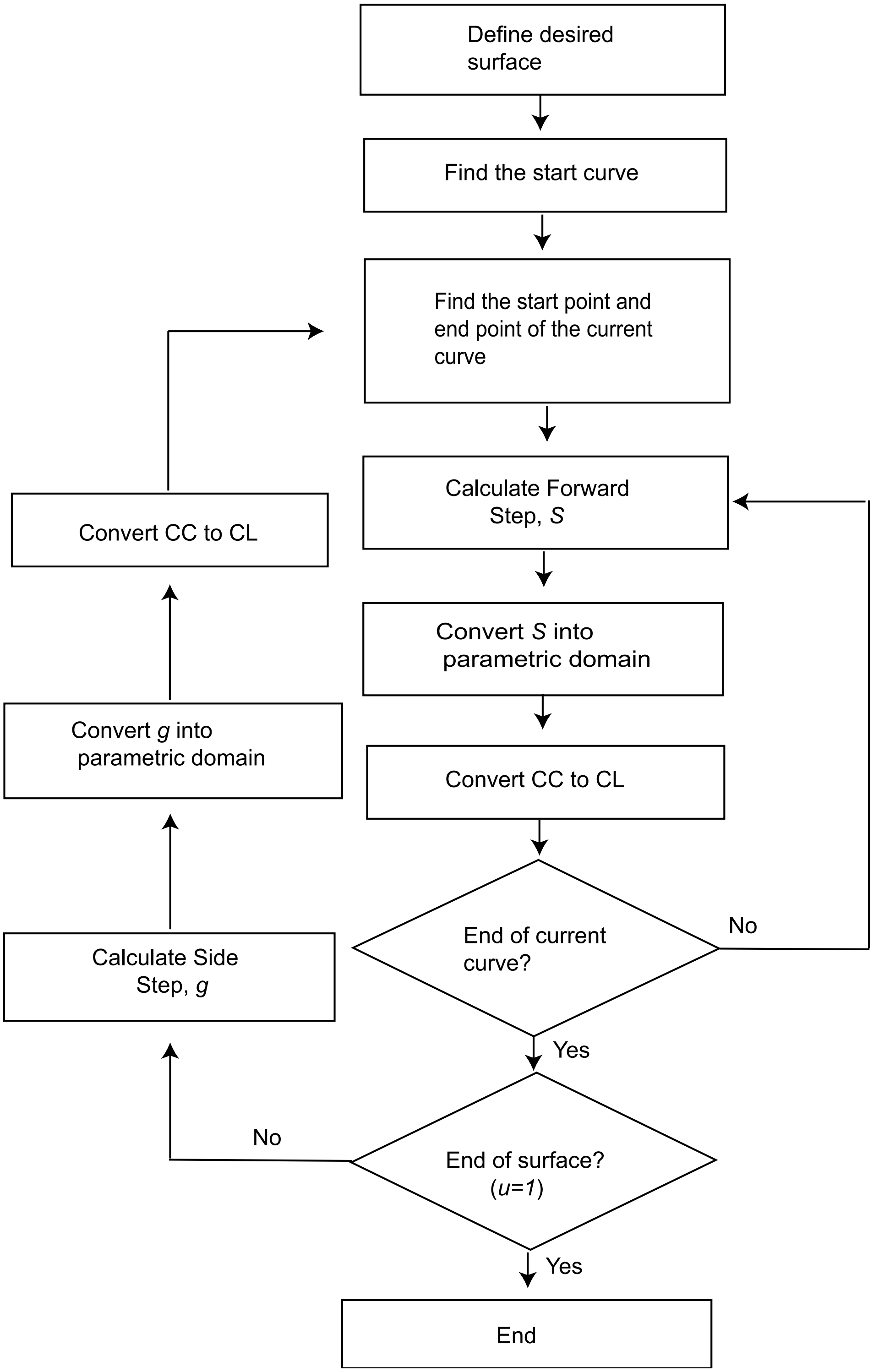


Fig. 13. Overall procedure

constant defines an iso-parametric curve. If one of the parameters reaches zero or one, the curve becomes exactly one of the boundary curves surrounding the surface. If both parameters are held constant, a point is specified on the surface patch.

2. Calculate the forward-step size with given tolerance, e.

   Each tool path on the surface is approximated by discrete points that make errors in tolerance. In this step, we approximate the tool path by linear interpolation. The forward-step size, s can be calculated using derivatives of B$\acute{e}$zier curves while keeping up with predetermined tolerance. The forward-step size is the maximum distance between CC points on the current tool path in which deviation does not exceed given tolerance, e.

3. Convert the forward-step size form the physical domain into the parametric domain

   The calculated forward-step size is in the physical domain instead of the parametric domain, u, v. Thus, to calculate the next CC point, the forward-step size has to be converted because work-part being processed is represented by B$\acute{e}$zier curves and surfaces using parameter values (u, v).

4. Convert CC points to CL points

   The result of calculating the forward-step is a CC point that can be any point on the tool. However, to reduce machining errors, it has to be converted to a CL point using equation 3.2.

5. Calculate side-step, g with given scallop height, h

   A manufactured part can be approximated by a series of tool paths. Unmachined regions between adjacent tool paths are called scallop. When the parameter value of an iso-parametric curve reaches one at the end of current curve, side-step (g), should be calculated for the next tool path. The side-step size is the maximum distance between two adjacent tool paths in which maximum scallop height is expressed.

6. Convert side-step from the physical domain into the parametric domain

   The calculated side-step size is also in physical domain and it does need to be converted in order to generate the next tool path.

7. Convert CC points to CL point points

   The CC points are also converted to CL points to store the points as a CL data file.

C. Calculation of Forward-Step

Each tool path is approximated by a series of line segments whose accuracy of tool path is controlled by deviation [figure 7(a)]. Each segment amount is a forward-step and the maximum deviation is called tolerance. To calculate forward-step, we use first and second derivatives; therefore, this function is independent of surface types and is applicable to all continuous parametric surfaces that are twice differentiable.

1. Theory

The mathematical formula for forward-step can be derived from iso-parametric curve on the surface using some theorems and lemmas as shown below. First, we consider Lipschitz condition.

Theorem 1. *If $f(x, y)$ is continuous at all points (x, y) in some rectangle*

$$\mathbf{R} : |x - x_0| < a, \quad |y - y_0| < b$$

and bounded in $\mathbf{R}$*, say,*

$$|f(x,y)| \leq K \qquad \forall (x,y) \in \mathbf{R} \tag{3.3}$$

then the problem, $y' = f(x,y), \quad y(x_0) = y_0$*, has at least one solution* $y(x)$*, which is defined at least for all* x *in the interval* $|x - x_0| < \alpha$ *where* α *is the smaller of the two numbers* a *and* b/K

Theorem 2. *if* $f(x,y)$ *and* $\frac{\partial f}{\partial y}$ *are continuous for all* (x,y) *in that rectangle* $\mathbf{R}$ *and bounded, say,*

$$(a)\, |f| \leq K, \quad (b) \left|\frac{\partial f}{\partial y}\right| \leq M \qquad \forall (x,y) \in \mathbf{R} \tag{3.4}$$

then the problem, $y' = f(x,y), \quad y(x_0) = y_0$*, has only one solution* $y(x)$*, which is defined at least for all* x *in that interval* $|x - x_0| < \alpha$

According to the theorems 1 and 2, since $y' = f(x,y)$, condition 3.3 implies that $|f(x,y)| \leq K$; that is, the slope of any solution curve $y(x)$ in $\mathbf{R}$ is at least $-K$ and at most K. Hence the solution curve which passes through the point must lie in the region bounded by lines whose slopes are $-K$ and K, respectively. The conditions in the two theorems above are sufficient conditions rather than necessary ones and can be lessened. By the mean value theorem of differential calculus we have

$$f(x,y_2) - f(x,y_1) = (y_2 - y_1)\frac{\partial f}{\partial y}|_{y=\tilde{y}} \tag{3.5}$$

where $\tilde{y}$ is a suitable value between y_1 and y_2. From this equation it follows that

$$|f(x,y_2) - f(x,y_1)| \leq M\,|y_2 - y_1| \tag{3.6}$$

and the condition 3.4(b) may be replaced by the above equation 3.6 which is known as a Lipschitz condition.

Lemma 1. *Let (i)* Ω *be the set of all functions* $f(x)$ *satisfying the Lipschitz condition* $|f(x_3) - f(x_4)| \leq K_1 |x_3 - x_4|$ *with a finite* K_1 *in the interval* (x_1, x_2)*, and* $f(x_1) = y_1, f(x_2) = y_2$*; (ii)* $f_1(x)$ *be the function such that* $f_1(x_1) = y_1, f_1(x_2) = y_2, f_1'(x) = K_1$ *for all* $x \in (x_1, x_3), f_1'(x) = -K_1$ *for all* $x \in (x_3, x_2), x_1 \leq x_3 \leq x_2$*. Then for any* $f(x) \in \Omega, f(x) \leq f_1(x)$*for all* $x \in (x_1, x_2)$

Lemma 2. *Let (i)* $f_1(x)$ *and* Ω *be as defined above; (ii)* $L(x)$ *be the function whose graph is the straight line joining two points* (x_1, y_1) *and* (x_2, y_2)*. then*

$$\max_{f(x)\in\Omega}\{\max_{x\in(x_1,x_2)} |f(x) - L(x)|\} = \max_{x\in(x_1,x_2)} |f(x) - L(x)| = \frac{K_1\delta}{2}$$

where $\delta = (x_2 - x_1)$.

According to this lemma, the maximum deviation $|f(x) - L(x)|$ for $f(x) \in \Omega$ is obtained by considering the $f_1(x)$ of Ω. The magnitude of this deviation equals $\left|(\frac{\delta}{2K_1})(K_1^2 - m^2)\right|$ where $m = \dot{L}(x)$, and its maximum is attained for $m = 0$ giving us $\frac{K_1\delta}{2}$.

Since a function with a bounded derivative satisfied Lipschitz condition, using the above lemmas we have the following theorem

Theorem 3. *Let* $f(x)$ *be a differentiable function with* $\left|\dot{f}(x)\right| \leq K_1$ *for all* $x \in (a, b)$*. If* $\hat{f}(x)$*is the piecewise linear approximation of* $f(x)$ *in* (a, b) *with subdivision interval* s*, then* $\max_{x\in(a,b)} \left|f(x) - \hat{f}(x)\right| \leq K_1\frac{s}{2}$.

Since twice differentiability of a function $f(x)$ implies a Lipschitz condition on $\dot{f}$, we have the following theorem

Theorem 4. *Let* $f(x)$ *be a twice differentiable function with* $\left|\ddot{f}\right| \leq K_2$ *for all* $x \in (a, b)$*. If* $\hat{f}(x)$*is the piecewise linear approximation of* $f(x)$ *in* (a, b) *with subdivision interval* s*, then* $\max_{x\in(a,b)} \left|f(x) - \hat{f}(x)\right| \leq K_2\frac{s^2}{8}$.

Now the case in which both $\left|\dot{f}\right|$ and $\left|\ddot{f}\right|$ have known bounds K_1 and K_2 respectively, is handled in the following theorem.

Theorem 5. *Let* $f(x)$ *be a twice differentiable function passing through the points* $(x_1, y_1)(x_2, y_2)$ *with* $\left|\dot{f}\right| \leq K_1, \left|\ddot{f}\right| \leq K_2$. *Let* $x_2 \geq x_1, (x_2 - x_1) = s, m = \frac{(y_2-y_1)}{(x_2-x_1)}(|m| \leq K_1)$. *Then depending on the values of* δ, K_1, K_2 *we have the information of Table I.*

The distance between two CC points can be determined by using maximum deviation (e), K_1, K_2. Using the above theorems, the forward-step, s, can be found for any $\max\left|f(x) - \hat{f}(x)\right|(e)$ in the interval (x_1, x_2)[36]. They can then be used to find the maximum value of s by using $\left|f(x) - \hat{f}(x)\right|, x \in (a, b)$, where $\hat{f}$ is the piecewise linear approximation of $f(x)$ in (a, b). It should be noted that Theorem 3, 4, 5 imply the existence of the derivatives since they are stated in terms of the first, second, or both the derivatives. Again this method is independent of surface types and is applicable to all continuous parametric surface that are twice differentiable.

2. Forward-Step Function

The derivatives are partial derivatives $\frac{\partial}{\partial u}$ or $\frac{\partial}{\partial v}$. This partial derivative is a tangent vector of an iso-parametric curve. It can be calculated by

$$\frac{\partial}{\partial u}\mathbf{b}^{m,n}(u,v) = \sum_{j=0}^{n}\left[\frac{\partial}{\partial u}\sum_{i=0}^{m}\mathbf{b}_{i,j}B_i^m(u)\right]B_j^n(v) \tag{3.7}$$

where m is a degree of Bézier curve in u, and n is a degree of Bézier curve in v. We can rewrite equation 3.7 in terms of forward difference operator Δ from the chapter II.

$$\frac{\partial}{\partial u}\mathbf{b}^{m,n}(u,v) = m\sum_{j=0}^{n}\sum_{i=0}^{m-1}\Delta^{1,0}\mathbf{b}_{i,j}B_i^{m-1}(u)B_j^n(v). \tag{3.8}$$

The superscript $(1, 0)$ of the difference operator means that difference is per-

Table I. Value of s

Value of s	Error:max $\left\| f(x) - \hat{f}(x) \right\|$
$s < \frac{2(K_1 - \|m\|)}{K_2}$	$E_1 = \frac{K_2 s^2}{8}$
$s > \frac{2(K_1 + \|m\|)}{K_2}$	$E_2 = \frac{1}{2}(K_1^2 - \|m\|^2)(\frac{s}{K_1} - \frac{1}{K_2})$
$\frac{2(K_1 - \|m\|)}{K_2} < s < \frac{2(K_1 + \|m\|)}{K_2}$	$E_3 = (K_1 + \|m\|)\{\sqrt{\frac{2s(K_1 + \|m\|)}{K_2}}\frac{K_1 + \|m\|}{2K_2} - s\}$
$s = \frac{2(K_1 - \|m\|)}{K_2}$	$E_4 = \max\{E_1, E_3\}$
$s = \frac{2(K_1 + \|m\|)}{K_2}$	$E_5 = \max\{E_2, E_3\}$

formed only on the first subscript. $\Delta^{1,0}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}$. If we take $v - partial$, the difference operator acts only on the second subscript. $\Delta^{0,1}\mathbf{b}_{i,j} = \mathbf{b}_{i,j+1} - \mathbf{b}_{i,j}$. Therefore, partial derivative in v can be written

$$\frac{\partial}{\partial v}\mathbf{b}^{m,n}(u,v) = n\sum_{i=0}^{m}\sum_{j=0}^{n-1}\Delta^{0,1}\mathbf{b}_{i,j}B_j^{n-1}(v)B_i^m(u). \tag{3.9}$$

However, higher-order partial can be expressed as

$$\frac{\partial^r}{\partial u^r}\mathbf{b}^{m,n}(u,v) = \frac{m!}{(m-r)!}\sum_{j=0}^{n}\sum_{i=0}^{m-r}\Delta^{r,0}\mathbf{b}_{i,j}B_i^{m-r}(u)B_j^n(v). \tag{3.10}$$

where the difference operator is $\Delta^{r,0}\mathbf{b}_{i,j} = \Delta^{r-1,0}\mathbf{b}_{i+1}j - \Delta^{r-1,0}\mathbf{b}_{i,j}$

and

$$\frac{\partial^s}{\partial v^s}\mathbf{b}^{m,n}(u,v) = \frac{n!}{(n-s)!}\sum_{i=0}^{m}\sum_{j=0}^{n-s}\Delta^{0,s}\mathbf{b}_{i,j}B_j^{n-s}(v)B_i^m(u). \tag{3.11}$$

where the difference operator is $\Delta^{0,s}\mathbf{b}_{i,j} = \Delta^{0,s-1}\mathbf{b}_ij + 1 - \Delta^{0,s-1}\mathbf{b}_{i,j}$

We can write down the mixed partial of arbitrary order

$$\frac{\partial^{r+s}}{\partial u^r \partial v^s}\mathbf{b}^{m,n}(u,v) = \frac{m!n!}{(m-r)!(n-s)!}\sum_{i=0}^{m-r}\sum_{j=0}^{n-s}\Delta^{r,s}\mathbf{b}_{i,j}B_i^{m-r}(u)B_j^{n-s}(v). \quad (3.12)$$

We can also determine partial and mixed derivatives of a point on a surface using the above equations. From these, we are interested in four boundary curves such as $\frac{\partial}{\partial u}\mid_{u=o}, \frac{\partial}{\partial u}\mid_{u=1}, \frac{\partial}{\partial v}\mid_{v=o}, and \frac{\partial}{\partial v}\mid_{v=1}$. We can obtain one of the boundary curves, $\frac{\partial}{\partial u}\mid_{u=0}$, using the following equation.

$$\frac{\partial^{r}}{\partial u^r}\mathbf{b}^{m,n}(0,v) = \frac{m!}{(m-r)!}\sum_{j=0}^{n}\Delta^{r,0}\mathbf{b}_{0,j}B_j^{n}(v). \quad (3.13)$$

Similar patterns hold for the other three boundary curves. First and second derivatives, $K1$ and $K2$, can be calculated by the above equation at each point, $\mathbf{p}$, on the surface. The maximum forward-step with given tolerance can then be determined.

Let K_2 be the maximum second derivative of current curve. The forward-step size can be calculated using

$$s^2 = \frac{K_2}{e * 8} \quad (3.14)$$

where e is given tolerance.

The cubic B*é*zier curve can be represented by four control points. Figure 14 shows a cubic B*é*zier curve with four control points generated by the de Casteljau algorithm using MATLAB. In this figure, each circle (vertex of the control polygon) is a control point of the cubic B*é*zier curve. Figure 15 shows the cubic B*é*zier curve and CC points calculated by equation 3.14. In figure 15, each circle on the curve represents CC points that will be converted to CL points used to generate the NC-code for machining. Figure 16 shows linear interpolation of the CC points. In this example, the maximum distance between two CC points is 0.0816 inches and a total of twenty six CC points were generated. Maximum and minimum deviation between the designed curve and the linear interpolation of the CC points are 0.000694 inches and 0 inch with a given tolerance of 0.005 inches. The result is summarized in Table II. The second column represents maximum and minimum measured tolerance. Distance (third column), is the maximum step size between two CC points in the physical domain; in the parametric domain, the maximum distance is 0.0401 if the curve is bounded by parameter value $u = 1$. (The method that converts step size in physical domain to that in parametric domain is introduced in section F.) According to the result, there is no point at which the deviation between linear interpolation of CC points and the curved surface that exceeds given tolerance, e. Thus, the algorithm for calculating forward-step is working well for the parametric curve.

The forward-step function reduced CC points by 74%. The designed curve was represented by one-hundred points, where the parameter value of the distance between two points is 0.01 inches. However, we approximated the curve by twenty six points with a given tolerance of 0.005 inches.

D. Calculation of Side-Step

For the purpose of machining, the designed part is approximated by a series of parametric curves and the distance between two adjacent tool-paths is a finite distance called the side-step. The side-step, in general, may vary along the machined surface and the un-machined region between two adjacent tool paths (the scallop or cusp). The upper limit on the height of this scallop is called the scallop-height-allowance. Typically, the desired value of the scallop height is given, from which the side-step,

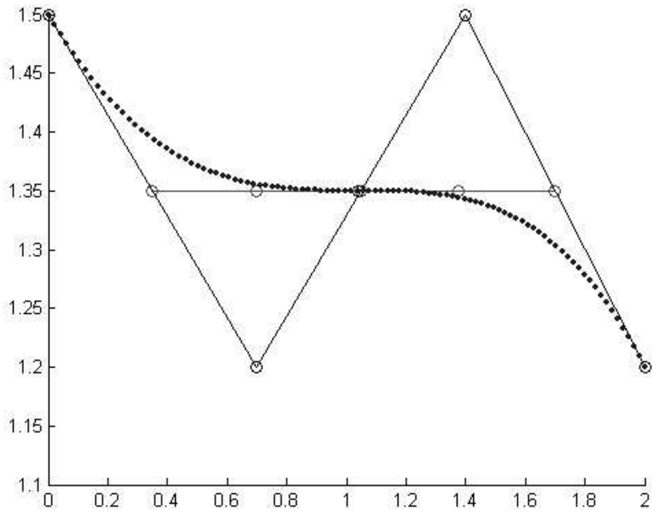


Fig. 14. Cubic Bézier curve generated by the de Castelljau algorithm

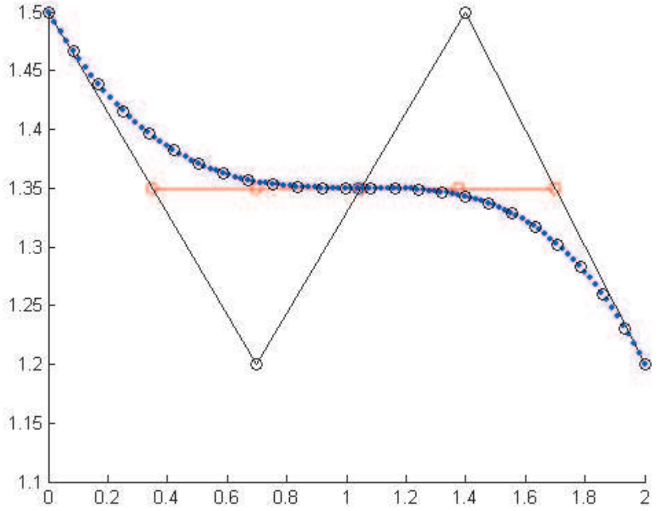


Fig. 15. Cubic Bézier curve and CC points

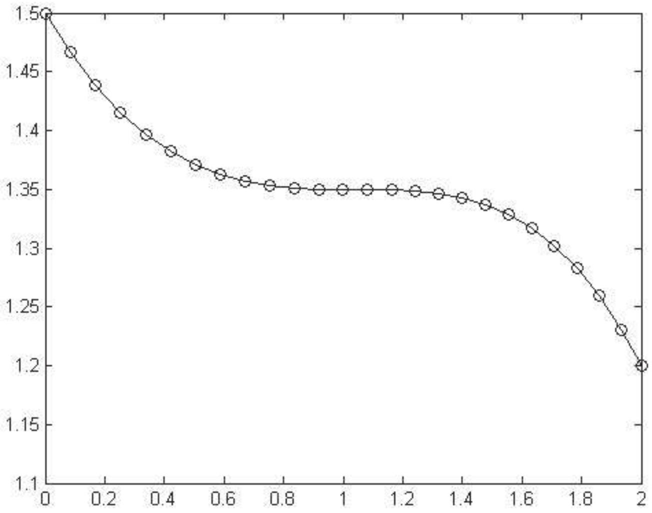


Fig. 16. Linear approximation of CC points

g, is determined.

1. Part Geometry

Consider a designed part $\mathbf{S} : \mathbf{r} = \mathbf{r}(\mathbf{u}, \mathbf{v})$ which has unique normal $\hat{\mathbf{n}}(\mathbf{u}, \mathbf{v})$ defined on its parametric domain. The method for calculating side-step used to calculate the maximum side-step distance between two adjacent tool paths while observing the

Table II. Computational results

Given	Measured		Distance	Number of
tolerance	Max	Min	s	CL points
0.005	0.0007	0.0	0.0816	26

unit=inches

given scallop height, h. The designed part $\mathbf{S}$ in figure 17, is approximated by the circle of curvature. Since we are interested in the path perpendicular to the tool path, it is denoted $*$. The curvature of the surface in the direction of the given curve is called the normal curvature and is calculated from

$$k_n = \frac{II}{I} \tag{3.15}$$

where I and II are first and second fundamental form introduced in section II.E.

The sign of the normal curvature indicates the direction of the center of the curvature with respect to the surface normal. As shown in figure 17, the sign of the curvature determines surface type (such as flat, concave, and convex). We are interested in the radius of the curvature(R^*) along the path perpendicular to the tool path at point, $\mathbf{p}$ on the surface.

To determine the radius of the curvature perpendicular to a parametric curve, first we consider the angle γ between two curve directions.

$$\dot{\mathbf{P}} = \mathbf{P}_u\dot{u} + \mathbf{P}_v\dot{v}, \quad \dot{\mathbf{P}}^* = \mathbf{P}_u\dot{u}^* + \mathbf{P}_v\dot{v}^*$$

Then γ is given by

$$cos\gamma = \frac{\dot{\mathbf{P}} \cdot \dot{\mathbf{P}}^*}{\left|\dot{\mathbf{P}}\right| \cdot \left|\dot{\mathbf{P}}^*\right|} = \frac{E\dot{u}\dot{u}^* + F(\dot{u}\dot{v}^* + \dot{v}\dot{u}^*) + G\dot{v}\dot{v}^*}{\sqrt{E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2}\sqrt{E\dot{u}{*}^2 + 2F\dot{u}{*}\dot{v}{*} + G\dot{v}{*}^2}}$$

where E, F, and G are coefficients of the first and second fundamental forms. If $\dot{\mathbf{P}}^*$ is orthogonal to $\dot{\mathbf{P}}(\gamma = \frac{\pi}{2})$, then it follows that

$$\frac{\dot{u}^*}{\dot{v}^*} = -\frac{F\dot{u} + G\dot{v}}{E\dot{u} + F\dot{v}} = \alpha \tag{3.16}$$

This leads to a simple expression of R^*.

$$R^* = \left|\frac{I}{II}\right| \tag{3.17}$$

where

$I = E\frac{du^*}{dt}\frac{du^*}{dt} + 2F\frac{du^*}{dt}\frac{dv^*}{dt} + G\frac{dv^*}{dt}\frac{dv^*}{dt}$,

and

$II = L\frac{du^*}{dt}\frac{du^*}{dt} + 2M\frac{du^*}{dt}\frac{dv^*}{dt} + N\frac{dv^*}{dt}\frac{dv^*}{dt}$

R^* [3.17] then reduces to the equation

$$R^* = \left|\frac{E + 2F\alpha + G\alpha^2}{L + 2M\alpha + N\alpha^2}\right| \tag{3.18}$$

2. Side-Step Function

The side-step, g, is a function of the scollop height (h), tool-radius(r) and the local radius of the curvature (R^*). In this section, we develop a new method to calculate side-step size with the given constraint of scallop height. The designed part's surface can be classified into convex, concave, and flat surface. The curvature of convex, concave, and flat surface are positive, negative, and zero, respectively. Therefore, we consider three different cases to calculate side-step size, g.

First, a flat surface as shown in figure 18, II=0

$$h = r - \sqrt{r^2 - (\frac{g}{2})^2}$$

$$g^2 = 4r^2 - 4(r-h)^2$$

$$g = 2\sqrt{r^2 - (r-h)^2} \tag{3.19}$$

Second, a convex curvature shown in figure 19, $II < 0$

To find the side-step for a convex surface we find δ, the difference between a designed

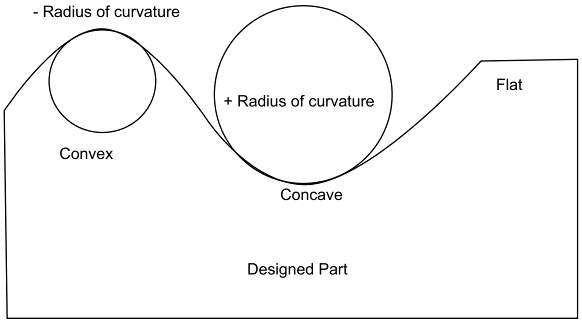


Fig. 17. Sign convention of radius of the curvature

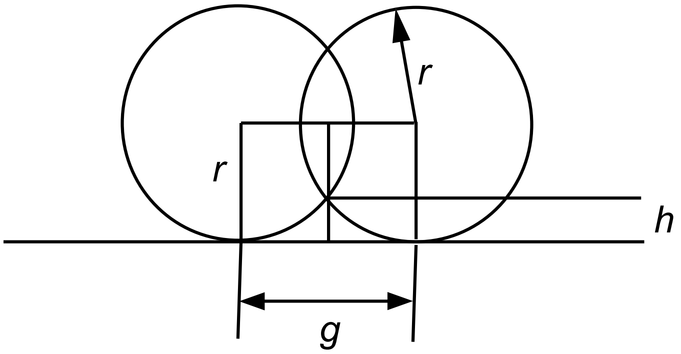


Fig. 18. Tool position on flat surface

curve and a linear tool path, as shown in figure 19

$$\delta = OB - OA$$

To find OA, we first calculate the side-step size, g ($\frac{g}{2} = p$) using equation 3.19. Because the side-step size is very small, we could use p ($\frac{g}{2}$) as an initial value. So

$$\begin{aligned} OA &= \sqrt{r^2 - p^2} \\ OB &= R^* \\ h &= r - \sqrt{r^2 - p^2} + \delta \\ g &= 2\sqrt{r^2 - (r + \delta - h)^2} \end{aligned} \tag{3.20}$$

where R^* is the local radius of the curvature of the convex surface.

Third, a concave curvature, $II > 0$

In a similar manner, we calculate the step size for a concave surface (figure 20).

$$\delta = OB - OA$$

$$\begin{aligned} OA &= \sqrt{r^2 - p^2} \\ OB &= R^* \\ h &= r - \sqrt{r^2 - p^2} - \delta \\ g &= 2\sqrt{r^2 - (r - \delta - h)^2} \end{aligned} \tag{3.21}$$

where R^* is the local radius of the curvature of the concave surface.

Using the above equations, g is calculated at each point on the current curve. Among these values, the minimum is taken as the optimum value for the path interval (g).

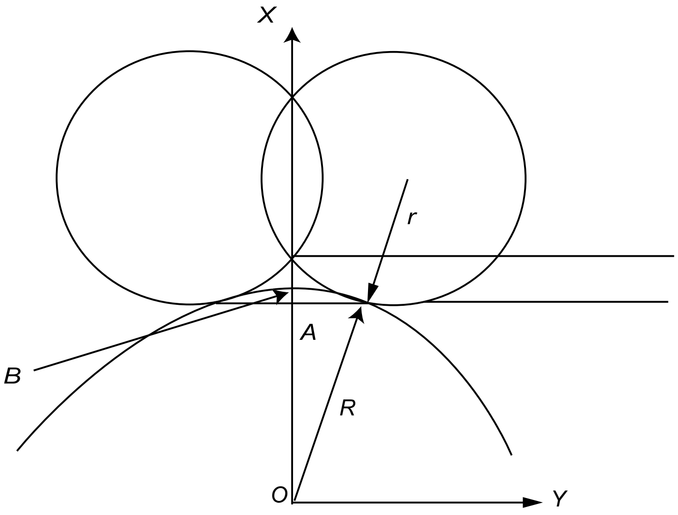


Fig. 19. Tool position on convex surface

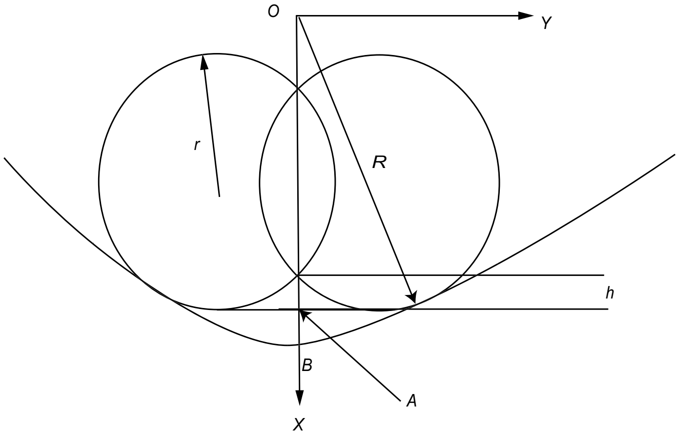


Fig. 20. Tool position on concave surface

E. Combination of Forward and Side-Step Error

The side-step function in the previous section is independent of forward-step function controlled by chordal deviation (tolerance, e). However, the side-step function can be improved by considering given tolerance, e. The location of cutter on the tool path is placed below (convex surface) or above (concave surface) the designed surface due to the given tolerance, e as shown in figures 21 and 22. Thus, the scallop height is overestimated and underestimated for the free-form surfaces. The maximum scallop height is twice of the given scallop height and the minimum scallop height is half of the given scallop height for free-form surfaces. Therefore, the scallop height is varying from $h/2$ to $2h$ $(= e + h)$ for free-form surfaces, although the part is machined with constant scallop height, h. In the following, the maximum error will be close to $2h$ for free-form surfaces because the tolerance and scallop height are set as the same value. If the scallop height is set as half of the given scallop height, h, all areas of machined surface will be good within the given tolerance and scallop height.

F. Conversion of the Interval to the Parametric Domain

The forward and side-step size calculated in the previous section is in the physical domain instead of the parametric domain (u, v). Since the part being processed is represented by a Bézier surface which is described by using u, v parameters, we have to convert the forward and side-step in the physical domain into the parametric domain in order to calculate the next CC point on the surface.

1. Conversion of Forward and Side-Step Size

The forward and side-step size calculated in the previous section can not be apply to the parametric domain directly because their directions of forward and side-step

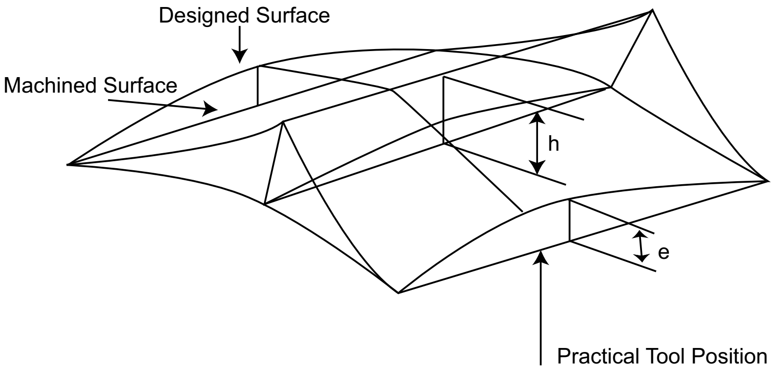


Fig. 21. Practical tool position on convex surface

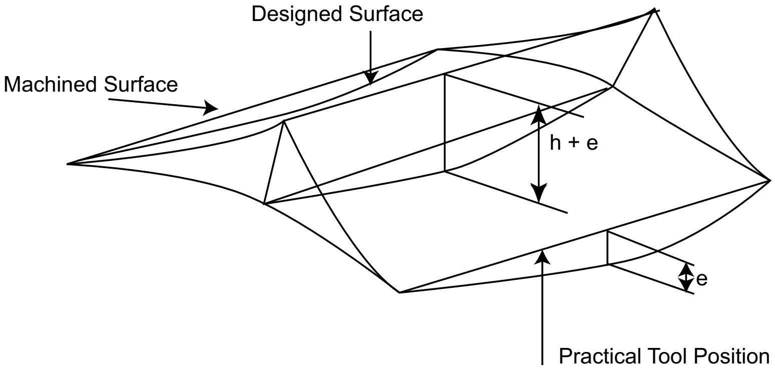


Fig. 22. Practical tool position on concave surface

in the physical domain are not same as the direction of the parametric curve and is calculated with physical unit (inch) instead of parameter value (u, v). Therefore, we have to convert them in order to calculate the next CC point and tool path on the surface as it is represented by a Bézier surface.

First, we consider the direction of the forward-step that is parallel to the x axis and not in the direction of the iso-parametric curve on the surface to obtain u, v parameter values corresponding to the forward-step size in the physical domain.

Second, we consider the direction of the side-step that is not an orthogonal direction of the tool path.

a. Direction of Forward-Step

The forward-step size that is parallel to the x axis is transferred to the direction of iso-parametric curve by

$$cos\theta = \frac{s}{s_p} \tag{3.22}$$

where s_p is the forward-step in the current parametric curve direction and g is the forward-step in the direction of the physical domain as shown in figure 23. θ is the angle between the tangent vector of the forward-step in the physical domain, $\mathbf{T}$, and the tangent vector of the parametric curve which can be calculated by

$$\theta = cos^{-1}\left(\frac{\frac{\partial P}{\partial v} \cdot \mathbf{T}}{\frac{\partial P}{\partial v} \cdot \mathbf{T}}\right) \tag{3.23}$$

b. Direction of Side-Step

The side-step size calculated in the previous section is also in the physical domain instead of parametric domain,(u, v). Thus, the side-step size, in orthogonal direction of the tool path, is needed to be transferred to the direction of the iso-parametric curve by

$$g_p = \frac{g}{sin\theta} \tag{3.24}$$

where g_p is the side-step in the current parametric curve direction and g is the side-step in the orthogonal direction of the current tool path (figure 24). θ is the angle between the tangent vector of the tool path and the tangent vector of the parametric

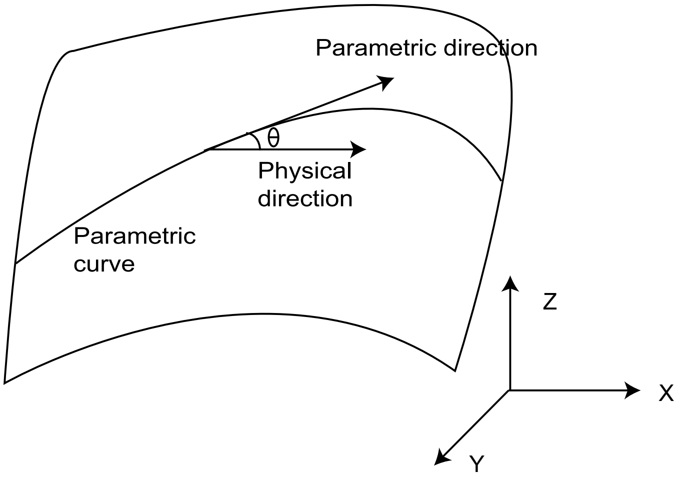


Fig. 23. Angular differences between the parametric direction and forward-step

curve which is calculated by

$$\theta = cos^{-1}\left(\frac{\frac{\partial P}{\partial u}\cdot \mathbf{T}}{\frac{\partial P}{\partial u}\cdot \mathbf{T}}\right) \tag{3.25}$$

c. Conversion

The forward, s, and side-step, g, quantity in the physical domain is converted into the parametric domain Δv, Δu in order to calculate the next tool path or CC point on the iso-parametric curve as shown in figures 23 and 24. We can convert the forward and side-step in physical domain into the parametric domain using the Taylor expansion and an error compensation technique shown below.

Given a parametric curve $P(v), 0 \leq v \leq 1$, which is the current curve on the surface, the Taylor series expansion of this parametric curve is

$$P(v) = P(v_0) + \dot{P}(v_0)\Delta v + \frac{1}{2}\ddot{P}(v_0)\Delta v^2 + \frac{1}{3!}\dddot{P}(v_0)\Delta v^3 + \ldots \tag{3.26}$$

where$\Delta v = v - v_0$

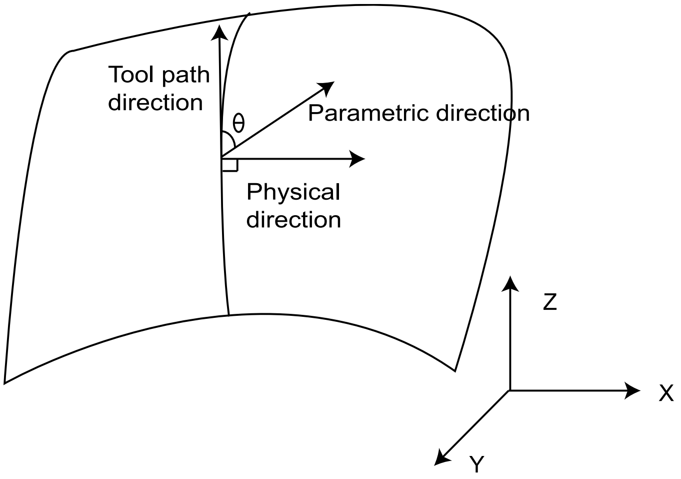


Fig. 24. Angular differences between the parametric direction and side-step

For side-step quantity, g, the equation is

$$P(u) = P(u_0) + \dot{P}(u_0)\Delta u + \frac{1}{2}\ddot{P}(u_0)\Delta u^2 + \frac{1}{3!}\dddot{P}(u_0)\Delta u^3 + \dots \tag{3.27}$$

where$\Delta u = u - u_0$

Δv and Δu are the parametric quantities between two CC points on the iso-parametric curve and the distance between two tool paths that correspond to the forward-step size and the side-step size in physical domain, respectively. v_0 and u_0 are the current CC point in the parametric domain and the term $|P(v_0) - P(v)|$ and $|P(u_0) - P(u)|$ are actually the forward and side-step in the parametric direction, s_p and g_p. If we neglect higher order terms, the forward-step can be derived as

$$s_p = |P(v_0) - P(v)| = |\dot{P}(v_0)\Delta v - \frac{1}{2}\ddot{P}(v_0)\Delta v^2| \tag{3.28}$$

$$g_p = |P(u_0) - P(u)| = |\dot{P}(u_0)\Delta u - \frac{1}{2}\ddot{P}(u_0)\Delta u^2| \tag{3.29}$$

Therefore,

$$s_p^2 = \dot{P}(v_0)^2\Delta v^2 + \dot{P}(v_0)\ddot{P}(v_0)\Delta v^3 + \frac{1}{4}\ddot{P}(v_0)^2\Delta v^4 \tag{3.30}$$

$$g_p^2 = \dot{P}(u_0)^2\Delta u^2 + \dot{P}(u_0)\ddot{P}(u_0)\Delta u^3 + \frac{1}{4}\ddot{P}(u_0)^2\Delta u^4 \tag{3.31}$$

Equation 3.30 and 3.31 can be written as

$$\begin{aligned} s_p^2 \;=\; & \left[\left(\frac{dx}{dv}\right)^2 + \left(\frac{dy}{dv}\right)^2 + \left(\frac{dz}{dv}\right)^2\right]\Delta v^2 + \\ & \left[\frac{dx}{dv}\cdot\frac{d^2x}{dv^2} + \frac{dy}{dv}\cdot\frac{d^2y}{dv^2} + \frac{dz}{dv}\cdot\frac{d^2z}{dv^2}\right]\Delta v^3 + \\ & \frac{1}{4}\left[\left(\frac{d^2x}{dv^2}\right)^2 + \left(\frac{d^2y}{dv^2}\right)^2 + \left(\frac{d^2z}{dv^2}\right)^2\right]\Delta v^4 \end{aligned} \tag{3.32}$$

$$\begin{aligned} g_p^2 \;=\; & \left[\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2\right]\Delta u^2 + \\ & \left[\frac{dx}{du}\cdot\frac{d^2x}{du^2} + \frac{dy}{du}\cdot\frac{d^2y}{du^2} + \frac{dz}{du}\cdot\frac{d^2z}{du^2}\right]\Delta u^3 + \\ & \frac{1}{4}\left[\left(\frac{d^2x}{du^2}\right)^2 + \left(\frac{d^2y}{du^2}\right)^2 + \left(\frac{d^2z}{du^2}\right)^2\right]\Delta u^4 \end{aligned} \tag{3.33}$$

However, the above equations are iterative. In order to find a parameter value, an initial value is needed such as Newton's method. The approximation of these functions is introduced in the following section.

2. Approximation of Functions

Equation 3.32 which represents the forward-step s_p requires a tedious iterative solution process and a proper initial value. To calculate Δv, we apply an error-compensation method introduced by Lin and Koren[22] to solve Δv and Δu in equation 3.32 and 3.33. In this section, only an approximation of the equation 3.32 is introduced. We can approximate the equation 3.33 using a similar manner.

The first order approximation of equation 3.32 is

$$\Delta v_a = \frac{s_p}{\sqrt{\left[\left(\frac{dx}{dv}\right)^2 + \left(\frac{dy}{dv}\right)^2 + \left(\frac{dz}{dv}\right)^2\right]}} \tag{3.34}$$

We can define error term ϵ, for the last two terms in equation 3.32.

That is,

$$\begin{aligned} \epsilon \;=\; & \left[\frac{dx}{dv} \cdot \frac{d^2x}{dv^2} + \frac{dy}{dv} \cdot \frac{d^2y}{dv^2} + \frac{dz}{dv} \cdot \frac{d^2z}{dv^2}\right] \Delta v^3 + \\ & \frac{1}{4}\left[\left(\frac{d^2x}{dv^2}\right)^2 + \left(\frac{d^2y}{dv^2}\right)^2 + \left(\frac{d^2z}{dv^2}\right)^2\right] \Delta v^4 \end{aligned} \tag{3.35}$$

The error is calculated by using $\Delta v = \Delta v_a$ from equation 3.34. Therefore, we can calculate Δv using the following equation.

$$\Delta v = \frac{\sqrt{s_p^2 - \epsilon}}{\sqrt{\left[\left(\frac{dx}{dv}\right)^2 + \left(\frac{dy}{dv}\right)^2 + \left(\frac{dz}{dv}\right)^2\right]}} \tag{3.36}$$

By using this error-compensation method, the accuracy is of the same order of magnitude as Newton's method, and it only needs one iteration.

CHAPTER IV

IMPLEMENTATION

The proposed solution approach was developed and implemented. There are several parts for which the CC points were generated using the proposed algorithm and were subsequently machined using a 3-axis milling machine with different tolerance and scallop height. The hardware and software used to implement proposed algorithms are:

1. Hardware:
    - Milling machine: proLIGHT 1000 Machining Center
    - 3D Laser scanner: LDI RPS 150
2. Software:
    - MATLAB 6.5
    - AUTOCAD 2002
    - Surveyor Scan Control
    - GEOMAGIC QUALIFY 5
    - WPLM1000 Control Software

The proposed algorithms were coded in MATLAB on a personal computer (Pentium III, 1.0 GHhz CPU, 256 Mb of physical memory) operating under Microsoft XP Professional. We machined a free-form shaped part using a block of wax and measured tolerance between the machined surface and the desired surface. After machining, a 3D laser machine (LDI RPS 150) with Surveyor Scan Control software (point cloud method) was used to scan the machined surface. GEOMAGIC QUALIFY 5 was used to compare surfaces. The desired surface was generated by AUTOCAD 2002 using X, Y, and Z coordinates generated by MATLAB 6.5. The WPLM1000 control software was used to control the milling machine described above. Several parts were machined with the following machining parameters:

Ball-end tool radius : 0.125 inches.
Maximum tolerance : 0.05 and 0.01 inches.
Maximum Scallop height: 0.05 and 0.01 inches.

A. Experiment and Results

A bi-cubic *Bézier* surface defined by 4 X 4 control points matrix as shown in below.

Example 1

(0.0 0.0 1.5)	(0.0 1.0 1.2)	(0.0 2.0 1.2)	(0.0 3.0 1.5)
(0.7 0.0 1.2)	(0.7 1.0 0.9)	(0.7 2.0 0.9)	(0.7 3.0 1.2)
(1.4 0.0 1.5)	(1.4 1.0 1.2)	(1.4 2.0 1.2)	(1.4 3.0 1.5)
(2.0 0.0 1.2)	(2.0 1.0 0.9)	(2.0 2.0 0.9)	(2.0 3.0 1.2)

Figure 25 refers to the tool path for the surface defined by the single patch Bézier surface in example 1 with the maximum allowed tolerance and scallop height are 0.01 inches (0.25 *mm*). One of the harder tasks was to combine the surfaces by finding reference points. GEOMAGIC QUALIFY 5 was used to compare the designed and finished surfaces. In GEOMAGIC QUALIFY 5, best fit alignment option was used to combine surfaces in which the software sampled 300 to 1500 points as a

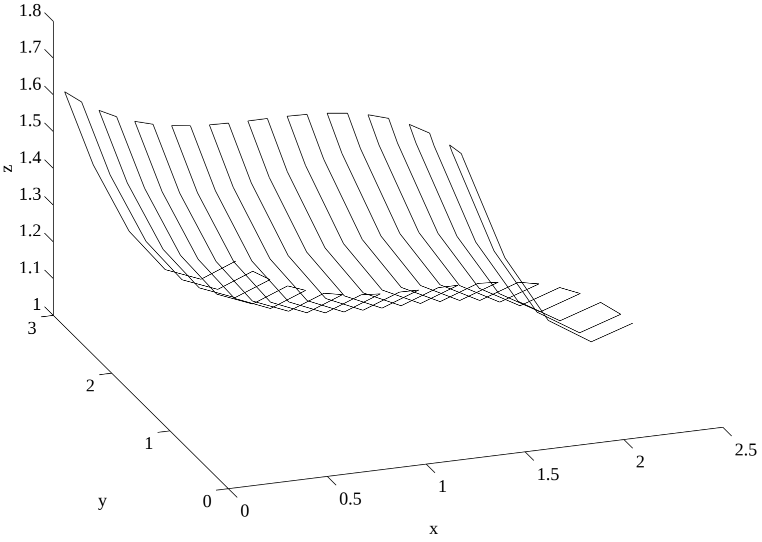


Fig. 25. Tool path of example 1

reference. Figures 26 and 29 refer to the results of the comparison between desired and machined surface with 0.05 inches and 0.01 inches tolerance and scallop height, respectively. Figures 27 and 28 show the machined part and scanned surface. Figures 30 and 31 show the real machined part and the scanned surface of the machined part with 0.01 inches tolerance and scallop height. The points on the surface were shaded using their normal vector to get a better view of the surface.

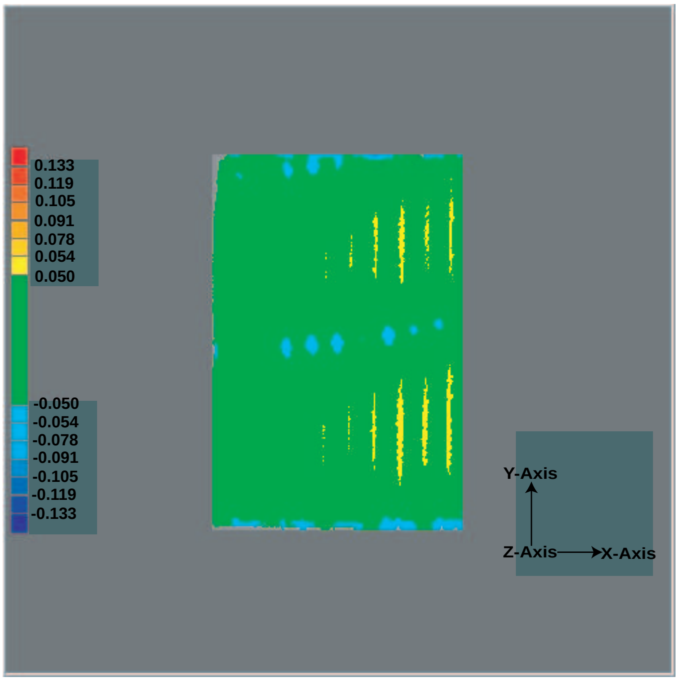


Fig. 26. Comparison between desired part and machined part for example 1 with tolerance 0.05 inches

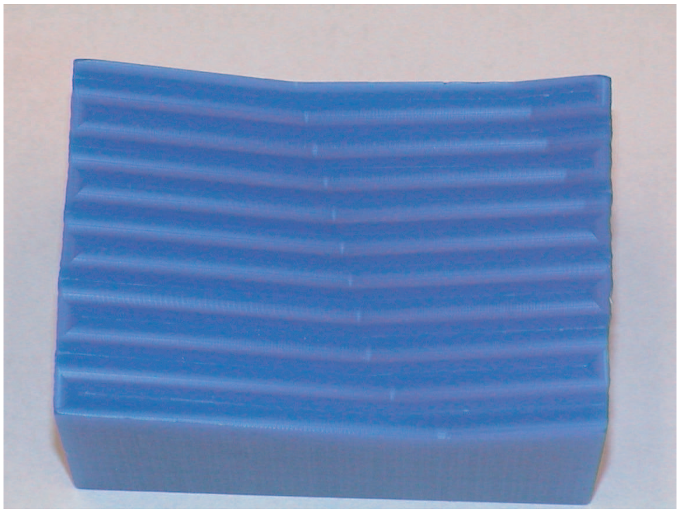


Fig. 27. Machined part of example 1 with tolerance 0.05 inches

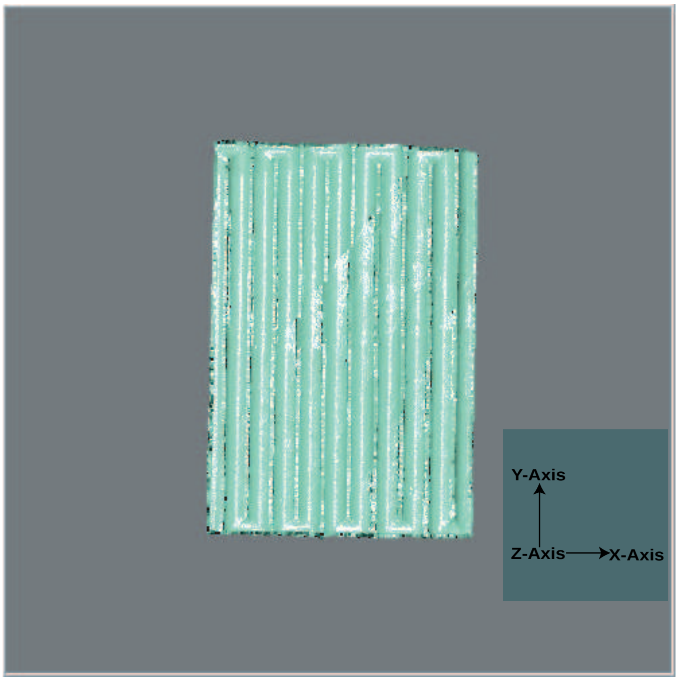


Fig. 28. Scanned surface of example 1 with tolerance 0.05 inches

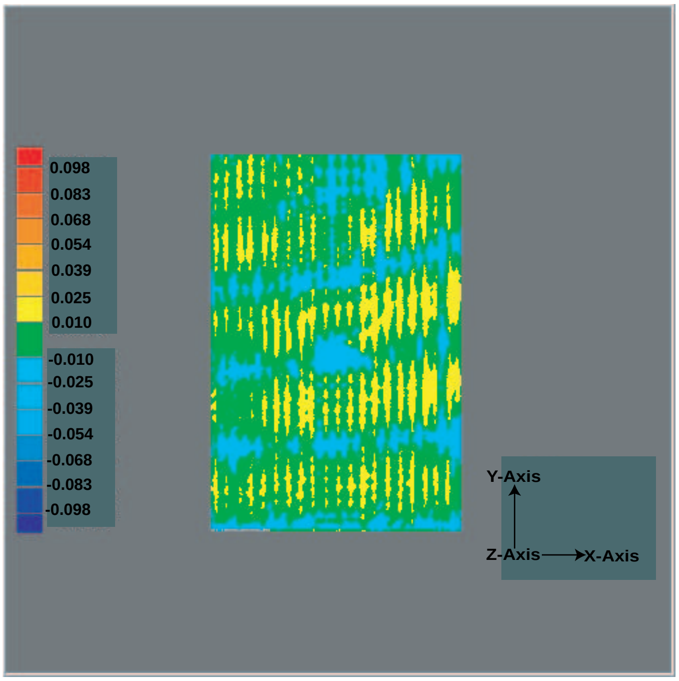


Fig. 29. Comparison between desired part and machined part for example 1 with tolerance 0.01 inches

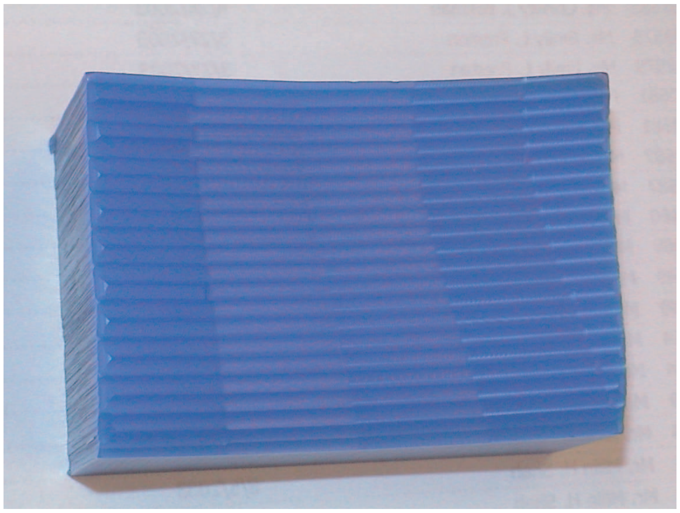


Fig. 30. Machined part of example 1 with tolerance 0.01 inches

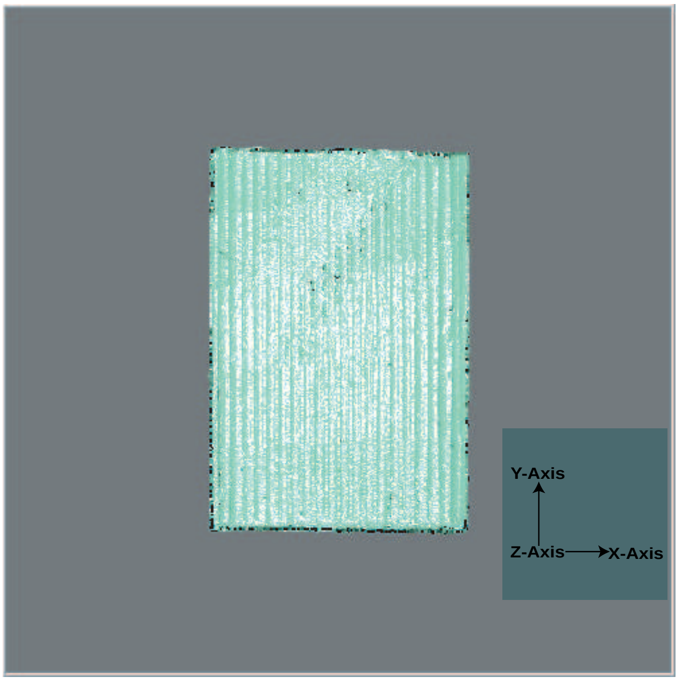


Fig. 31. Scanned surface of example 1 with tolerance 0.01 inches

Table III. Result of example 1

Scallop and Tolerance	SE	TE	Average	Number of CL points	Total Length	
					Proposed	Iso-scallop
0.01	0.019	0.018	0.0046	132	67.41	67.62
0.05	0.067	0.06	0.017	42	37.6	38.47

unit=inches

The results of example 1 are summarized in Table III. In the first column, “Scallop” and “Tolerance” represent maximum allowed tolerance and scallop height. The second column, SE, represents the maximum scallop height on the machined surface. TE stands for maximum error of tolerance. “Average” in the fourth column represents the average tolerance and scallop height over the surface. The fifth column, “Number of CL points”, is for CL points used to create the NC-code by which the machined surface was generated. The last column, “Total Length” shows the total length of tool paths. The total length of tool paths generated by proposed approach was compared with total length of tool paths of efficient iso-scallop approach proposed by Lin and Koren[22]. They also showed the efficient iso-scallop approach is efficient machining compared with iso-parametric machining. The total length of the tool paths generated by proposed approach is shorter than the total length of the tool paths for the efficient iso-scallop approach. Therefore, the proposed approach is efficient machining compared with the iso-scallop and iso-parametric approach. The total lengths of tool paths of the proposed approach are 37.6 and 67.41 inches and they are 38.47 and 67.62 inches of the iso-scallop approach with tolerance and scallop height 0.05 and 0.01 inches, respectively. According to the Table III, the proposed algorithm reduced CL points significantly and almost all areas tolerance and scallop height were within the given maximum allowed tolerance and scallop height for the free-form surface. However, the tolerance around a few CC points is a little higher than the given tolerance and some scallop heights are a little higher as well. In Table III, the maximum error of scallop height and tolerance are not greater than 0.017 inches ($0.43mm$) and 0.01 inches ($0.25mm$).

We conclude that the proposed approach is efficient machining compared with the iso-scallop approach and the maximum machining error is from 0.01 inches to 0.017 inches for the tolerance and scallop height for this example.

Example 2:

The control points shown below makes a convex shape surface.

(0.0 0.0 1.5)	(0.0 1.0 1.2)	(0.0 2.0 1.2)	(0.0 3.0 1.5)
(0.7 0.0 1.2)	(0.7 1.0 0.7)	(0.7 2.0 0.7)	(0.7 3.0 1.2)
(1.4 0.0 1.2)	(1.4 1.0 0.7)	(1.4 2.0 0.7)	(1.4 3.0 1.2)
(2.0 0.0 1.5)	(2.0 1.0 1.2)	(2.0 2.0 1.2)	(2.0 3.0 1.5)

Figure 32 refers to the tool path for the surface defined by the single patch Bézier surface in example 2 with the maximum allowed tolerance and scallop height are 0.01 inches (0.25 *mm*). Figures 33 and 36 refer to the result of comparison between the desired surface and the machined surface with 0.05 inches and 0.01 inches of tolerance and scallop height. Figures 34 and 35 show the machined part and scanned surface. Figures 37 and 38 show the machined part and scanned surface of machined part with 0.01 inches tolerance and scallop height. The points on the surface were shaded using their normal vector.

Table IV. Result of example 2

Scallop and				Number of	Total Length	
Tolerance	SE	TE	Average	CL points	Proposed	Iso-scallop
0.01	0.02	0.02	0.0045	154	70.63	70.76
0.05	0.063	0.054	0.016	40	37.67	38.13

unit=inches

The results of example 2 is summarized in Table IV. According to Table IV, our algorithm also reduced CL points significantly and almost all areas of tolerance and scallop height were within the given maximum allowed tolerance and scallop

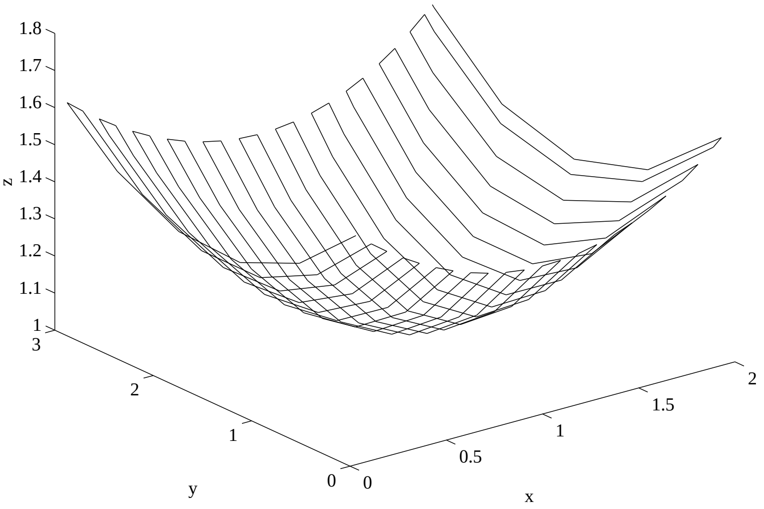


Fig. 32. Tool path of example 2

height for the convex shape surface. Again, some scallop height is a little higher than allowed scallop height and tolerance around a few CC points is a little higher than the given parameters. From Table IV, the maximum error of scallop height and tolerance was less than 0.013 inches (0.33mm) and 0.01 inches (0.2mm). There is also an approximated constant machining error for this case. The total lengths of tool paths of the proposed approach are 37.67 and 70.63 inches and they are 38.13 and 70.76 inches of the iso-scallop approach with tolerance and scallop height 0.05 and 0.01 inches, respectively. Therefore, we can say that the proposed approach is more efficient machining than the iso-scallop as well as iso-parametric machining and the maximum machining error is from 0.01 inches to 0.013 inches for the scallop height and tolerance.

According to the Tables III and IV, our algorithm reduced CL points significantly and is efficient machining compared with the iso-scallop approach. Almost all areas

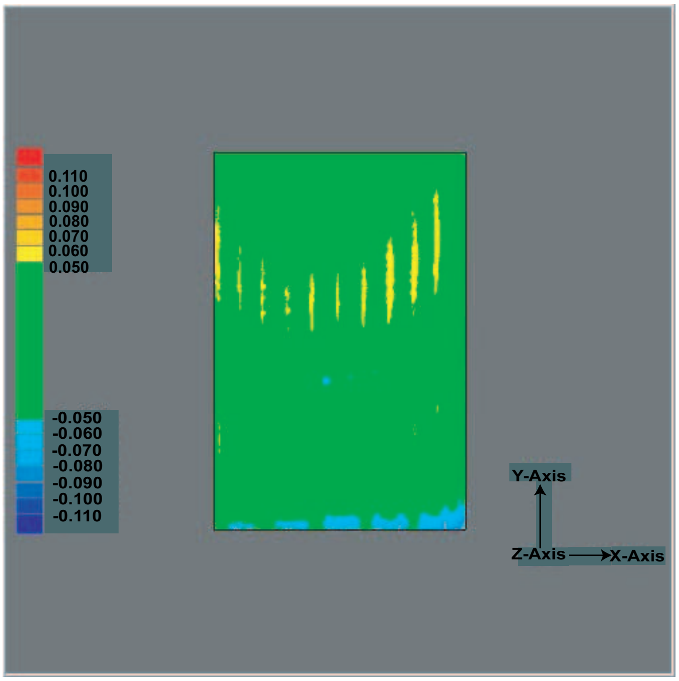


Fig. 33. Comparison between desired part and machined part for example 2 with tolerance 0.05 inches

of tolerance and scallop height were given maximum allowed tolerance and scallop height. According to the results in chapter III, there is no point whose deviation exceeds the given tolerance. However, tolerance around a few CC points is a little higher and can be attributed to machining errors such as vibration and cutting force as well as measuring (point cloud method) errors. For instance, we have to set up the origin of the part to be machined manually and that can make serious errors for the machining with very small allowed scallop height and tolerance. Analysis of these errors is beyond the scope of this work. In Tables III and IV, the maximum error of scallop height and tolerance were not greater than 0.017 inches ($0.43mm$) and 0.01 inches ($0.25mm$). There is an approximated constant machining error for both cases; therefore, we conclude that the proposed approach is more efficient machining than

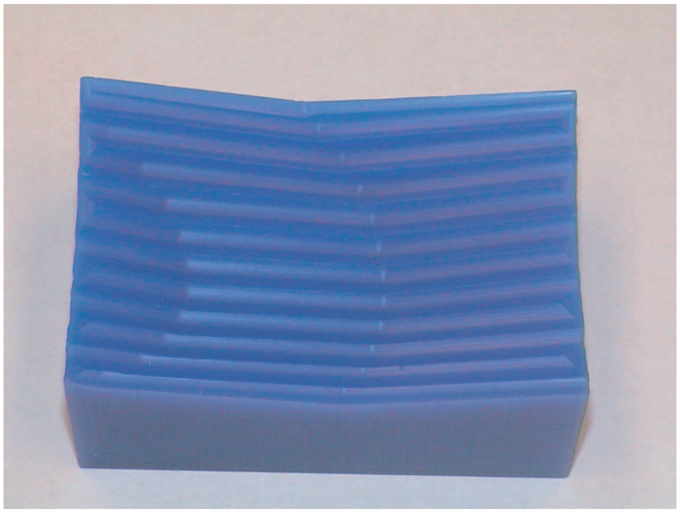


Fig. 34. Machined part of example 2 with tolerance 0.05 inches

the iso-scallop approach as well as the iso-parametric approach and the machining error is from 0.01 inches to 0.017 inches for the machining.

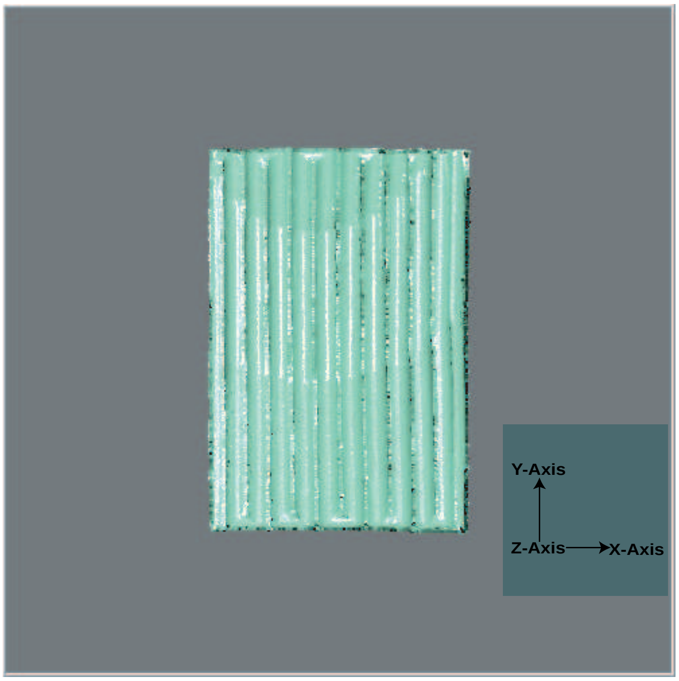


Fig. 35. Scanned surface of example 2 with tolerance 0.05 inches

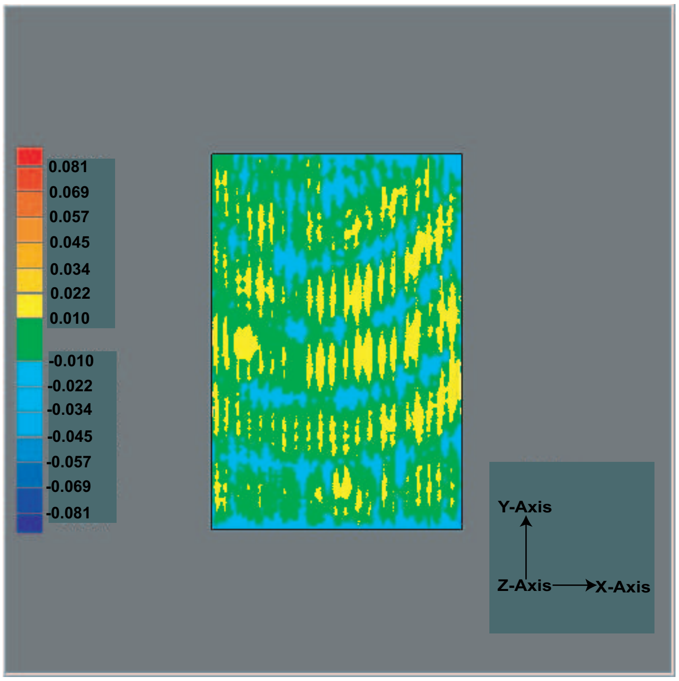


Fig. 36. Comparison between desired part and machined part for example 2 with tolerance 0.01 inches

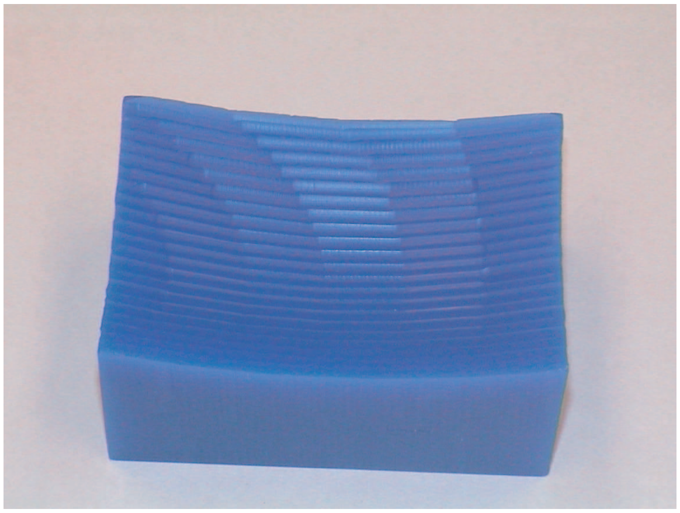


Fig. 37. Machined part of example 2 with tolerance 0.01 inches

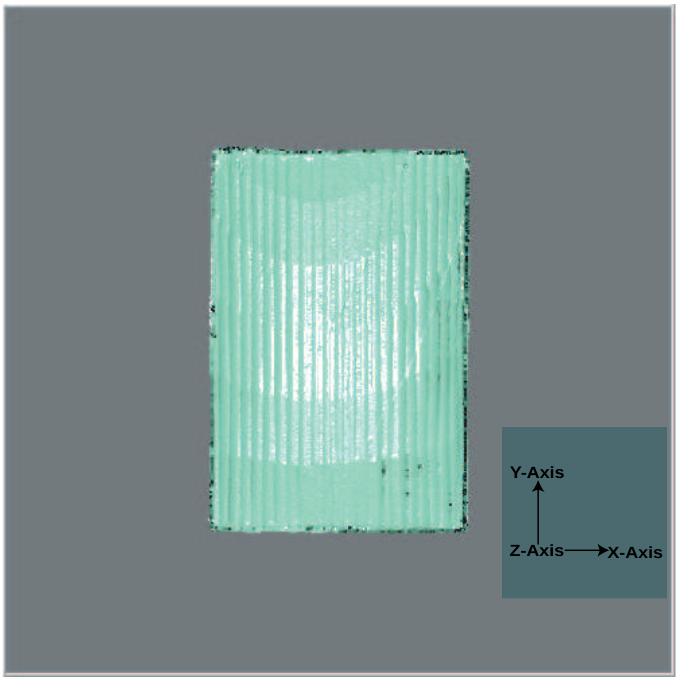


Fig. 38. Scanned surface of example 2 with tolerance 0.01 inches

B. 3D Tolerance Analysis

In this section, we analyze tolerance on the machined surface. Figure 39 shows X, Y, and Z coordinates of the designed surface and machined surface where the two surfaces are combined with tolerance and maximum scallop height of 0.05 inches. We sampled eight points on the surface to perform this analysis and the results of 3D tolerance analysis are summarized in Table V and VI. In the first column, "Name" represents the x, y, and z coordinates of a point. The second column, "Designed part", represents the x, y, and z coordinate at the point on the designed surface. The third column, "Manufactured part", represents the x, y, and z coordinate at the point on the machined surface. Deviation in the fourth column represents the difference at the point between the designed and manufactured surfaces.

From Tables V and VI, the point (A8) has the largest tolerance at 0.11 inches. More detailed representations are shown in figures 41 - 50, which are included in Appendix A.

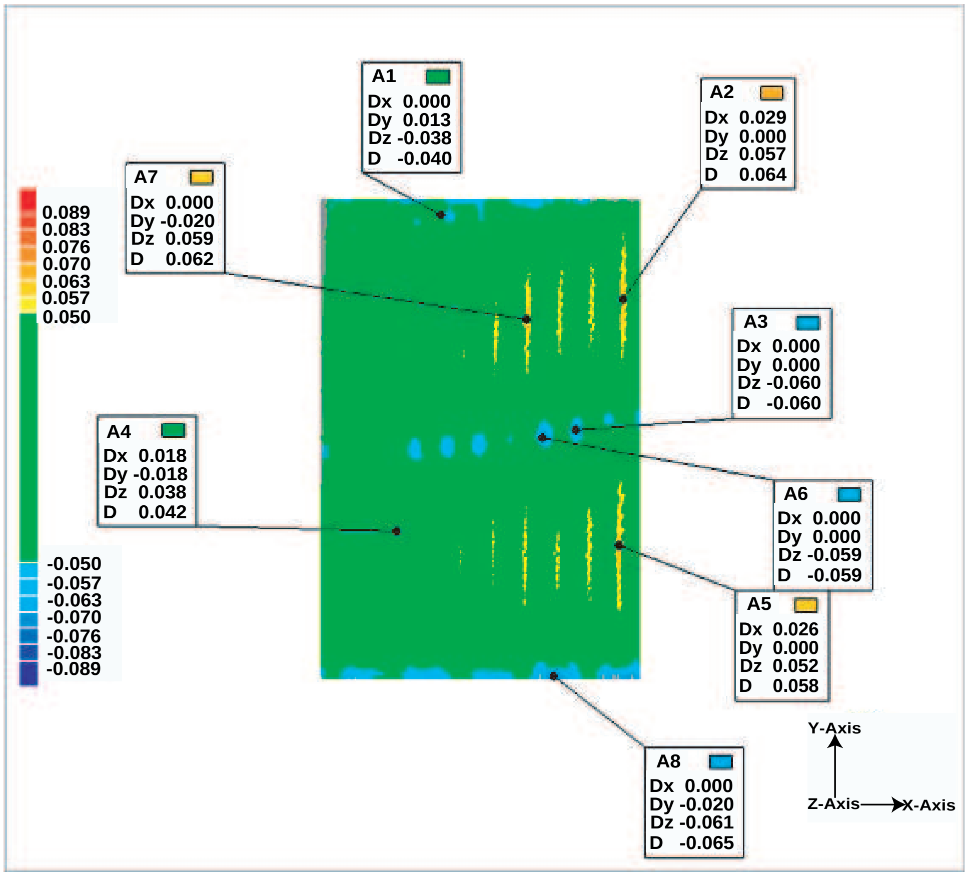


Fig. 39. Annotation on the top view

Table V. Tolerance analysis 1

Name	Designed part	Manufactured part	Deviation
A1			-0.040
X	0.753	0.753	0.000
Y	2.900	2.1913	0.013
Z	1.327	1.289	-0.038
A2			0.064
X	1.890	1.919	0.029
Y	2.372	2.372	0.000
Z	1.100	1.157	0.057
A3			-0.060
X	1.595	1.595	0.000
Y	1.556	1.556	0.000
Z	1.100	1.040	-0.060
A4			0.042
X	0.477	0.495	0.018
Y	0.922	0.922	-0.000
Z	1.183	1.220	0.038

Table VI. Tolerance analysis 2

Name	Designed part	Manufactured part	Deviation
A5			0.058
X	1.865	1.891	0.026
Y	0.834	0.834	0.000
Z	1.073	1.124	0.052
A6			-0.059
X	1.387	1.387	0.000
Y	1.506	1.506	0.000
Z	1.120	1.061	-0.059
A7			0.062
X	1.287	1.287	0.000
Y	2.247	2.227	-0.020
Z	1.179	1.238	0.059
A8			-0.065
X	1.457	1.457	0.000
Y	0.017	-0.003	-0.020
Z	1.334	1.273	-0.061

C. Integrate to CAD/CAM System

Process planning is the function in a manufacturing system that determines which processes and parameters are to be used to convert a part from its initial form to a final form designed in an engineering drawing. The input of process planning will be a three-dimensional model from a CAD data base. The model contains not only geometric information such as shape and dimensioning, but also the tolerance and special features. The basis of the process plan consists of geometric and technical information about raw materials and the finished product. The geometric information is taken from the engineering drawing generated by a CAD system in which free form curves and surfaces are available. The most popular and common type in CAD systems is Bézier curve that can be designed directly on a computer using control points. Commonly used surfaces in CAD systems are bicubic surfaces such as Bézier surface, B-spline surfaces, and nonuniform rational B-spline surface (NURB).

In this section, the method for integration of tool path generation module to CAD system is introduced. Figure 40 shows the logical process used to integrate tool path generation module to CAD/CAM system. The CAD/CAM system contains geometric and engineering information of raw material and finished product. This information can be represented in an engineering BOM (bill of material). In this dissertation, since designed and machined parts are represented by Bézier curves and surfaces, control points of the Bézier surfaces should be included in the engineering BOM. Using this information, a tool path generation module generates CL data file and generates machine independent NC-code. (These modules are subject of this dissertation.) Then the commercial CAD/CAM system generates NC-code using their postprocesser to machine the part.

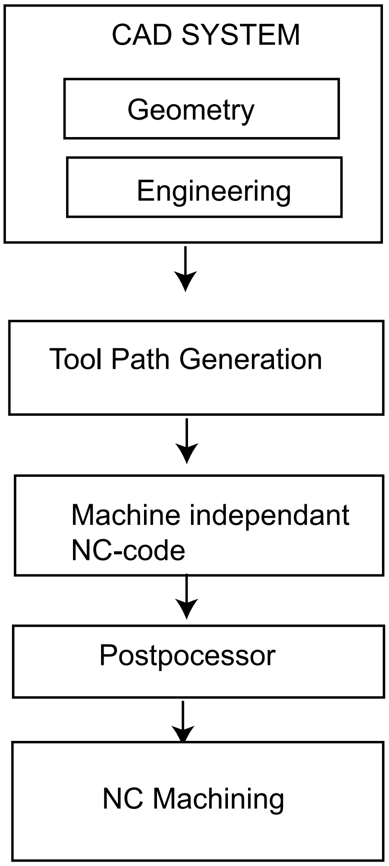


Fig. 40. CAD/CAM-integrated tool path generation

CHAPTER V

RESEARCH SUMMARY AND CONCLUSIONS

A. Research Contribution and Conclusion

The proposed algorithm for tool path generation was developed and implemented successfully through the integration of mathematical modelling used for calculating forward and side-step size into the core of the our algorithms. As such the primary contribution of this dissertation are

1. developing a new method for tool path generation in milling operations
2. verifying true machining error in milling operations

We used mathematical representation of tool and manufactured parts to make our algorithm efficient and reliable. Using this mathematical representation, we were able to determine forward-step size. From there we have developed a method for side-step size by studying the geometry of the tool and the differential geometry of the designed part. We then verified true machining errors by comparing machined and designed surfaces using the point cloud method. The implementation of this algorithm shows that it is very efficient for finish machining and the algorithm involved one iteration compared to existing methods.

Additional contribution is related to mathematical representation of manufactured parts through the use of parametric curves and surfaces.

As a conclusion, we list some advantages of this dissertation.

1. We reduced CL points significantly (by which NC code was generated) as shown in Tables III and IV in chapter IV. For example, the designed part was generated by 100×100 points. However, we generated machined surface by less than

160 CL points with predetermined scallop height and tolerance for small test problems. As a result of this, the manufacturing data generated from machining also decreased significantly, that is we reduce cost of data manipulation as well as storage.

2. We verified true machining errors by comparing designed surface and machined surface using the point cloud method. Previous efforts have relied upon computational approach (such as Huang and Oliver[20]) and our work provides superior verification via true machining.

3. Our method is independent of surface types and is applicable to all continuous parametric surfaces that are twice differentiable. Therefore,this approach is well suited for sculptured and analytic surfaces.

B. Future Direction

While this dissertation gives us an increased understanding of NC tool path generation, it also gives us several ideas for future research. Some of these directions described below are extensions of this dissertation.

1. Virtual Prototype: In this dissertation, we machined a real part and compared to the surface of the desired part and manufactured part to verify tool paths and true machining errors. However, there are several ways to verify tool paths. One is dry cut on the machine without a work-part but a dry cut can not detect detailed geometric errors. Other method is actually machine a prototype. However, the cost of machining is very expensive and time consuming in some cases. For instance, the work-part to be machined is very expensive or machining cost is very high. Thus, a virtual prototype will be needed to verify tool paths and

true machining error without real machining by considering appropriate machining conditions such as feed rate, spindle speed, and tool radius, as well as work piece material characteristics.

2. Tool path generation using flat ended tool: Although we used a ball-end tool in the milling operation, flat ended tool is also commonly used in milling operations. For a flat ended tool, swept section changes are needed depending on the inclination of the tool with the surface normal.

3. Integration: Tool path generation is part of the process planning in manufacturing systems as described in chapter I. Therefore, the module of tool path generation has to be integrated into other processes of manufacturing. The integration of data generated from the different module in process planning would be a direct extension of this dissertation.

REFERENCES

[1] R. Ait-Haddou and W. Herzog, “Convex subdivision of a Bezier curve,” *Computer Aided Geometric Design*, vol. 19, pp. 663-671, 2002.

[2] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming*, New York: John Wiley & Sons, 1993.

[3] J. E. Bobrow, “NC machine tool path generation from CSG part representations,” *Computer Aided Geometric Design*, vol. 17, no. 2, pp. 69-76, 1985.

[4] P. Broomhead and M. Edkins, “Generating NC data at the machine tool for the manufacture of free form surface ,” *International Journal of Production Research*, vol. 24, no. 1, pp. 1-14, 1986.

[5] G. Catania, “A computer-aided prototype system for NC rough milling of free-form shaped mechanical part-pieces,” *Computers in Industry*, vol. 20, pp. 275-293, 1992.

[6] T. C. Chang, R. A. Wysk, and H.P. Wang, *Computer-aided Manufacturing*, Upper Saddle River, NJ: Prentice-Hall, 1998.

[7] Z. C. Chen, Z. Dong, and G. W. Vickers, “Automated surface subdivision and tool path generation for $3\frac{1}{2}\frac{1}{2}$ axis CNC machining of sculptured parts,” *Computers in Industry*, vol. 50, pp. 319-331, 2003.

[8] B. K. Choi, C. S. Lee, J. S. Hwang, and C. S. Jun, “Compound surface modeling and machining ,” *Computer Aided Design* , vol. 20, no. 3, pp. 127-136, 1988.

[9] B. K. Choi and C. S. Jun, “Ball-end cutter interference avoidance in NC machining of sculptured surfaces ,” *Computer Aided Design*, vol. 21, no. 6, pp. 371-378.

[10] H. S. M. Coxeter, *Introduction to Geometry*, New York: John Wiley & Sons, 1966.

[11] M. A. Duchaineau, “Using general polar value as control points for polynomial curves,” *Computer Aided Geometric Design*, vol. 11, pp. 411-423, 1994.

[12] G. Farin, *Curves and Surfaces for CAGD*, San Diego, CA: Academic Press, 5th edition, 2002.

[13] R. T. Farouki, Y. Tsai, and G. Yuan, “Contour machining of free-form surfaces with real time PH curve CNC interpolation,” *Computer Aided Geometric Design*, vol. 16, pp. 61-76, 1999.

[14] I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture*, Chichester, West Sussex, England: Ellis Horwood Limited, 1979.

[15] J. D. Foley and A. van Dam, *Computer Graphics: Principles and Practice*, New York: Addison-Wesley Publishing Company, 2nd edition, 1997.

[16] K. K. George and N. R. Babu, “On the effective tool path planning algorithms for sculptured surface manufacture,” *Computers & Industrial Engineering*, vol. 28, no. 4, pp. 823 - 836, 1995.

[17] M. P. Groover, *Automation, Production Systems, and computer Integrated Manufacturing*, Upper Saddle River, NJ: Prentice-Hall, 1987.

[18] D. Hanselman and B. Littlefield, *Mastering MATLAB 6*, Upper Saddle River, NJ: Prentice-Hall, 2001.

[19] J. Hoschek, D. Lasser, *Fundamentals of Computer Aided Geometric Design*, Wellesley, MA: A K Peters, 1993.

[20] Y. Huang and J. Oliver, "Non-constant papameter NC tool path generation on sculptured surfaces," *International Journal of Advanced Manufacturing Technology*, vol. 9, no. 2, pp. 281-290, 1994.

[21] C. G. Jensen and D. C. Anderson, "A review of numerically controlled methods for finish-sculptured-surface machining," *IIE Transactions*, vol. 28, pp. 30-39, 1996.

[22] R.-S. Lin, and Y. Koren, "Efficient tool-path planning for machining free-from surfaces ," *Transactions of the ASME* , vol. 118, pp. 20-28, 1994.

[23] G. C. Loney and T. M. Ozsoy, "Machining of free form surface," *Computer Aided Design*, vol. 19, no. 2, pp. 85-89, 1987.

[24] R. R. Meyer, "Computational aspects of two-segment separable programming," *Mathematical Programming*, vol. 26, pp. 21-39, 1983.

[25] M. E. Mortenson, *Geometric Modeling*, New York: John Wiley & Sons, 1985.

[26] M. E. Mortenson, *Mathematics for Computer Graphics Application*, New York: Industrial Press Inc., 1999.

[27] G. Omura, *AutoCAD 2000*, Alameda, CA: SYBEX Inc., 1999.

[28] S. C. Park, "Tool path generation for Z-constant contour machining," *Computer Aided Design*, vol. 35, pp. 27-36, 2003.

[29] L. Ramshaw, "Blossoms are polar forms," *Computer Aided Geometric Design*, vol. 6, pp. 323-358, 1989.

[30] R. Sarma, " NC tool path synthesis," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1996.

[31] R. Sarma and D. Dutta, " An integrated system for NC machining of multi-patch surface," *Computer Aided Design*, vol. 29, no. 11,pp 741-749, 1997.

[32] A. W. Scheer, *Business Process Engineering*, Berlin: Springer-Verlag, 1998.

[33] S. Smith and J. Tlusty, "An overview of modeling and simulation of the milling process," *Journal of Engineering for Industry*, vol. 113, pp. 169-175, 1991.

[34] Y. S. Suh, and K. Lee, "NC milling tool path generation for arbitrary pockets defined by sculptured surfaces," *Computer Aided Design*, vol. 22, no. 5, pp. 273-284, 1990.

[35] K. suresh and D. C. H. Yang, "Constant scallop-height machining of free-form surfaces," *Journal of Engineering for Industry*, vol. 116, pp. 253 - 259, 1994.

[36] L. S. Thakur, "Error analysis for convex separable programs: The piecewise linear approximation and the bounds on the optimal objective value," *SIAM Journal of Applied Mathematics*, vol. 43, no. 4, pp. 704 - 714, 1978.

[37] G. W. Vickers and K. W. Quan, "Ball mills versus end-mills for curved surface machining," *Transaction of the ASME*, vol. 111, pp. 22-26, 1989.

[38] D. C. H. Yang, J. J. Chung, and T. H. Oulee "Boundary-conformed toolpath generation for teimmed free-form surfaces," *Computer Aided Design*, vol. 35, pp. 127-139, 2003.

[39] G. M. Zhang and S. G. Kapoor, "Dynamic generation of machined surfaces, part 1: Discription of a random excitation system," *Journal of Engineering for Industry*, vol. 113, pp. 137-144, 1991.

[40] G. M. Zhang and S. G. Kapoor, “Dynamic generation of machined surfaces, part 2: Construction of surface topography,” *Journal of Engineering for Industry*, vol. 113, pp. 145-153, 1991.

[41] L. P. Zhang, J. Y. H. Fuh, and A. Y. C. Lee, “Tool path generation for mold design modification,” *Computer Aided Design*, vol. 35, no. 4, pp. 813-824, 2003.

APPENDIX A

3D TOLERANCE ANALYSIS

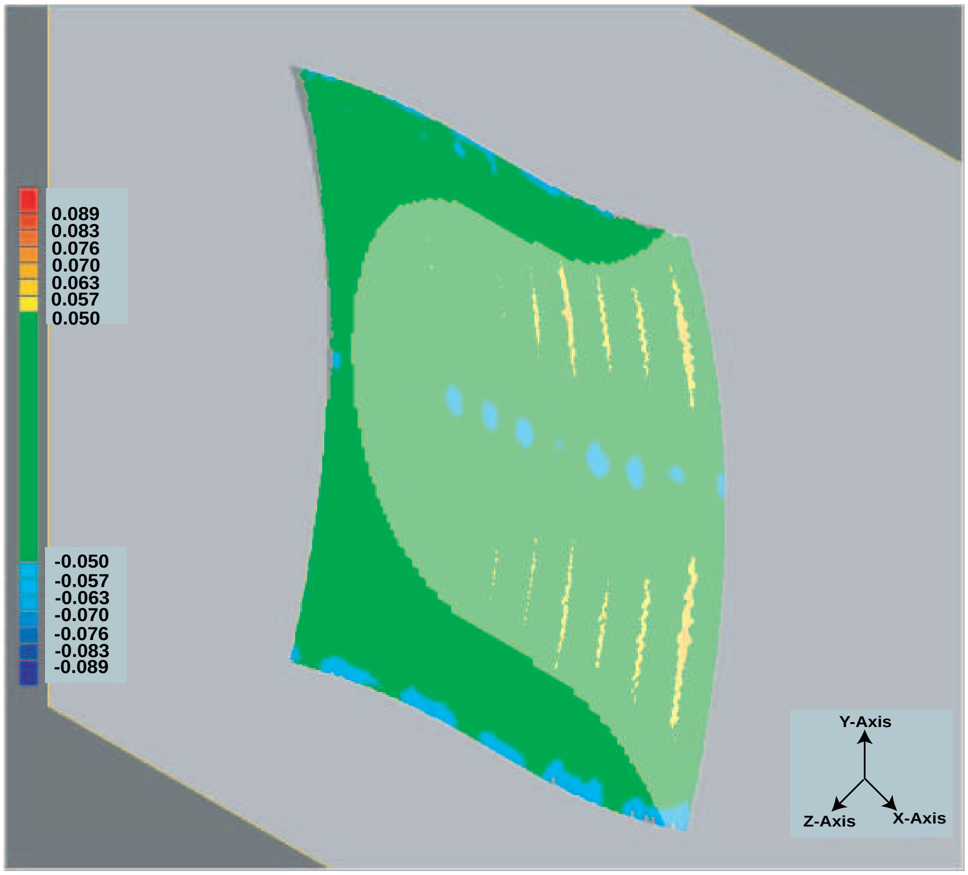


Fig. 41. Cross section through X and Y axis

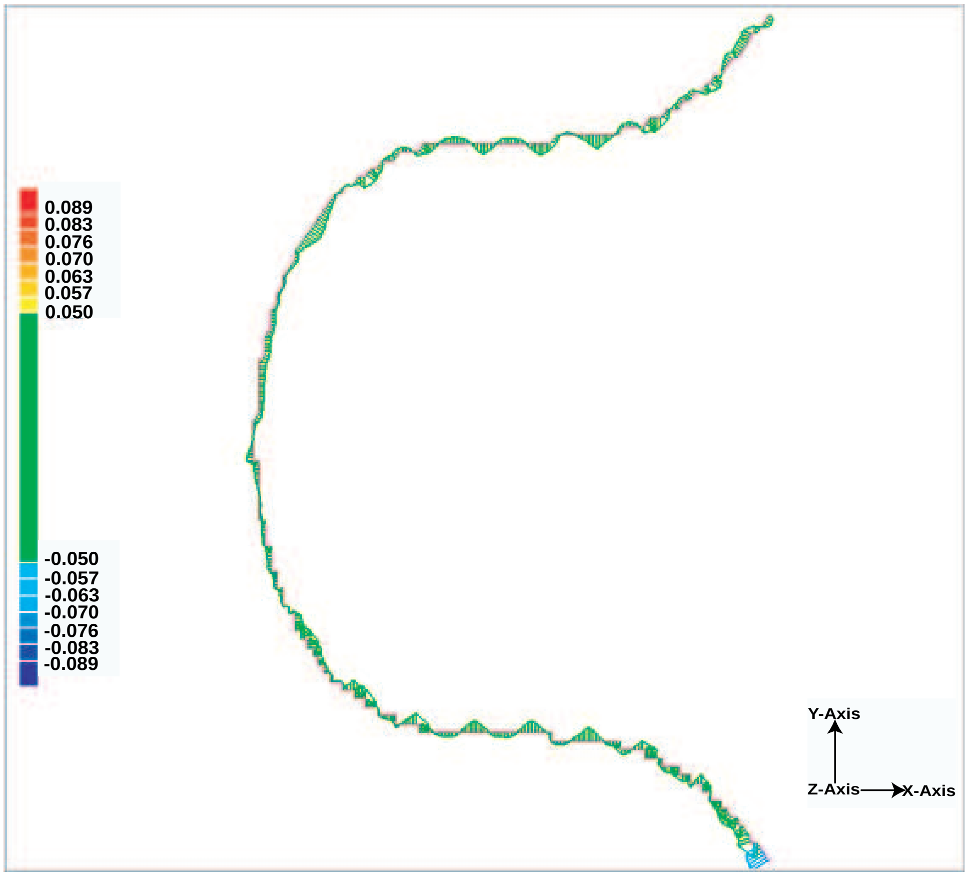


Fig. 42. Cross section analysis through X and Y axis

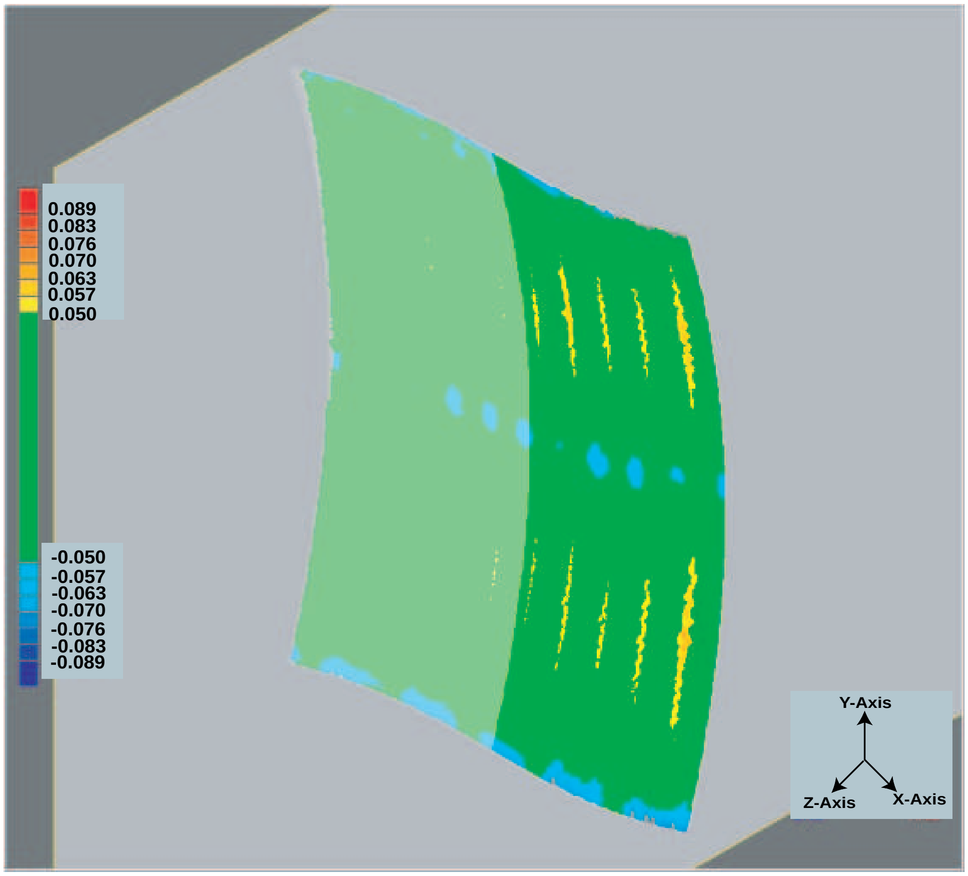


Fig. 43. Cross section through Y and Z axis

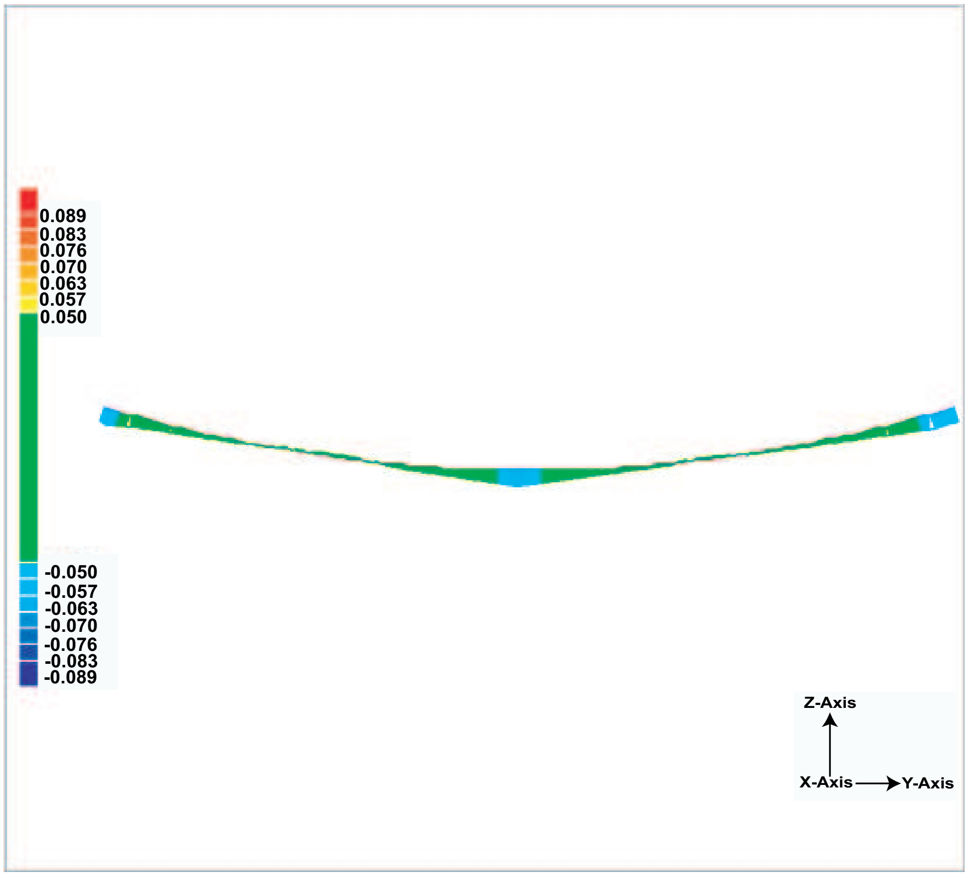


Fig. 44. Cross section analysis through Y and Z axis

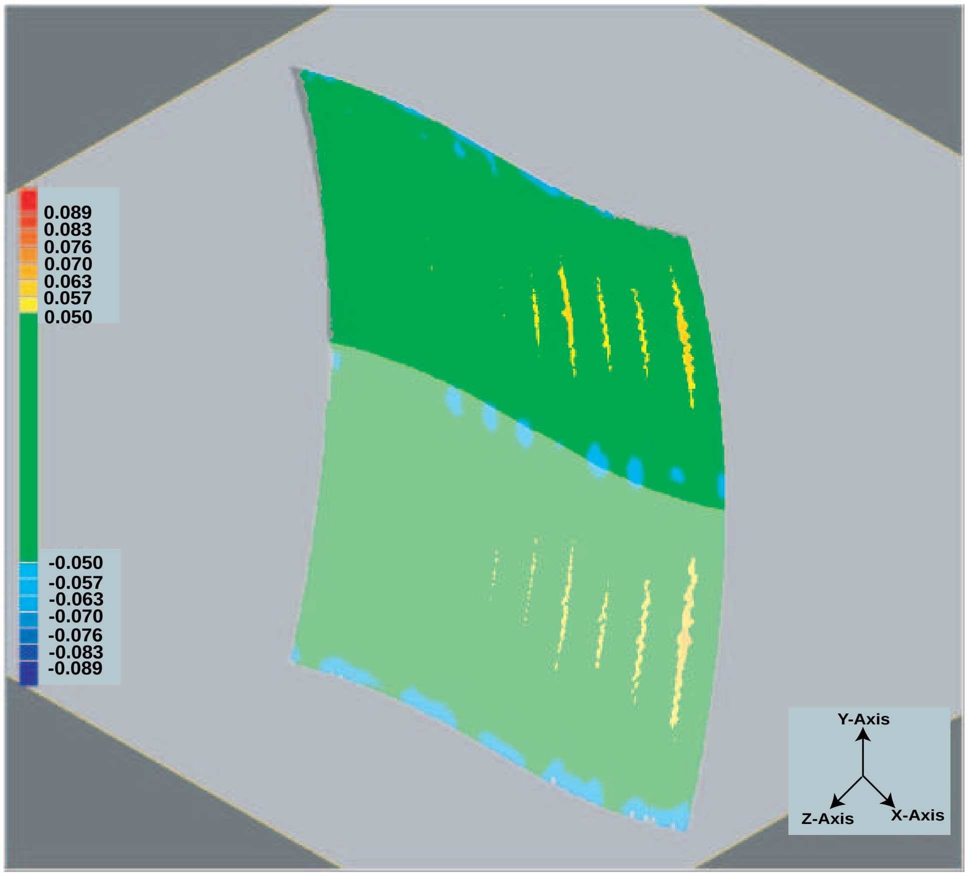


Fig. 45. Cross section through X and Z axis

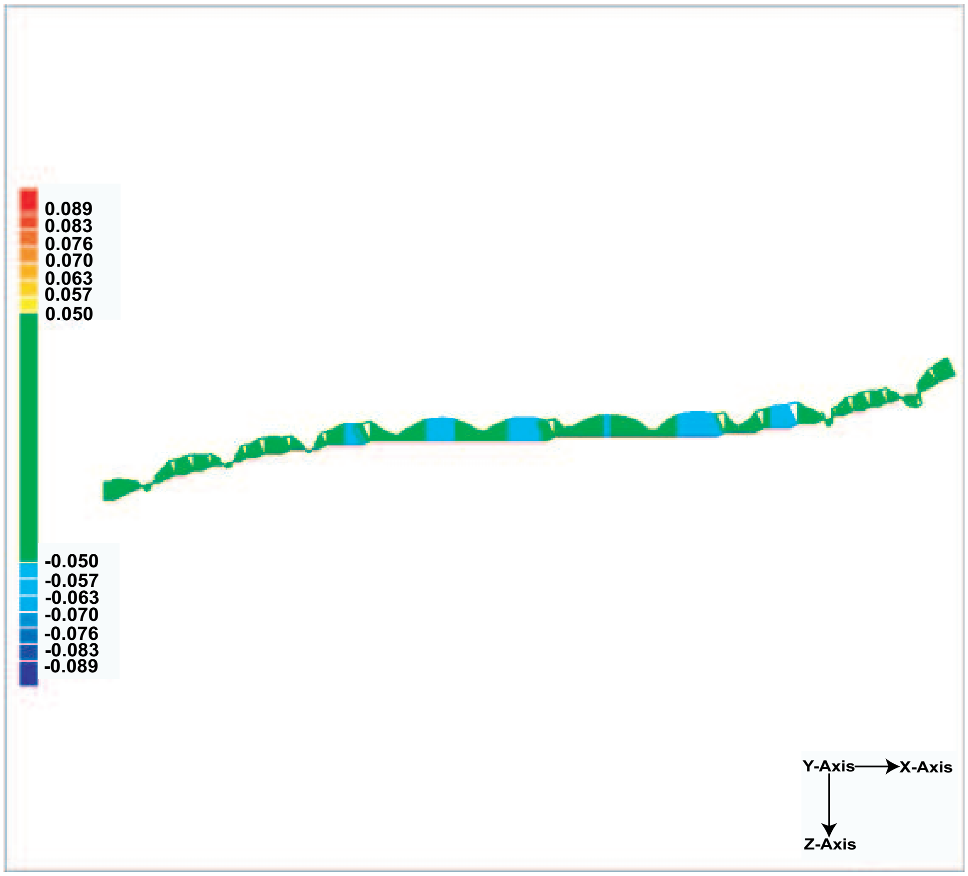


Fig. 46. Cross section analysis through X and Z axis

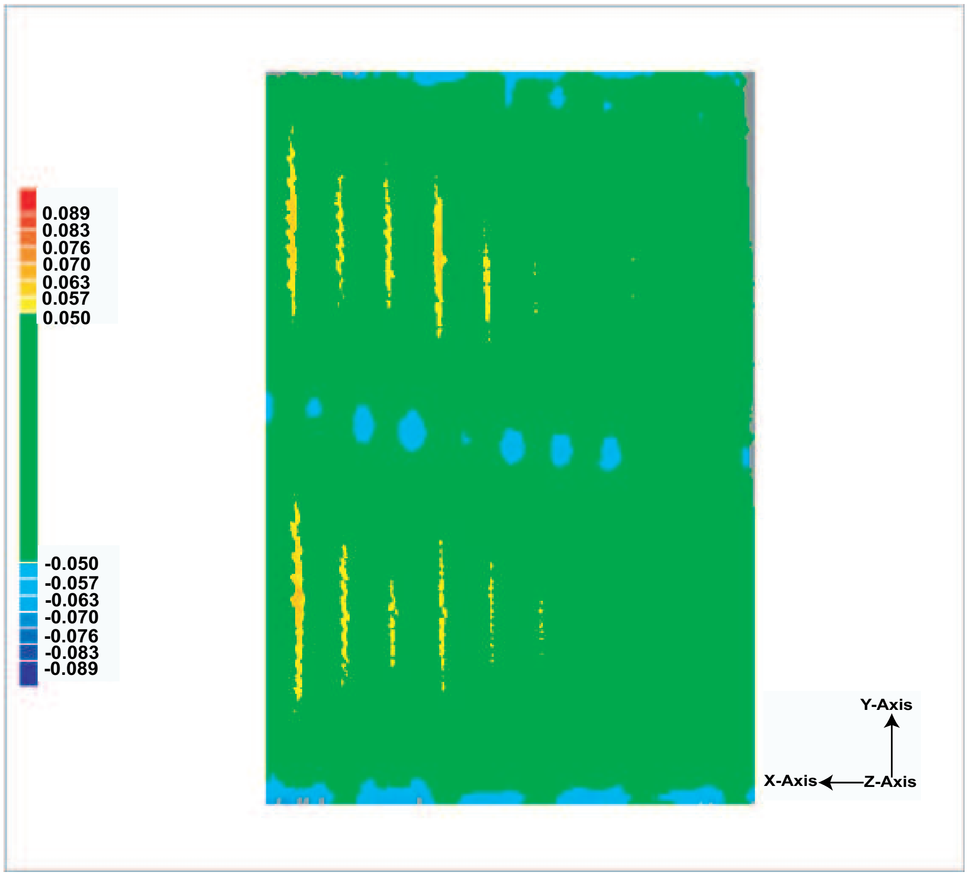


Fig. 47. Back view

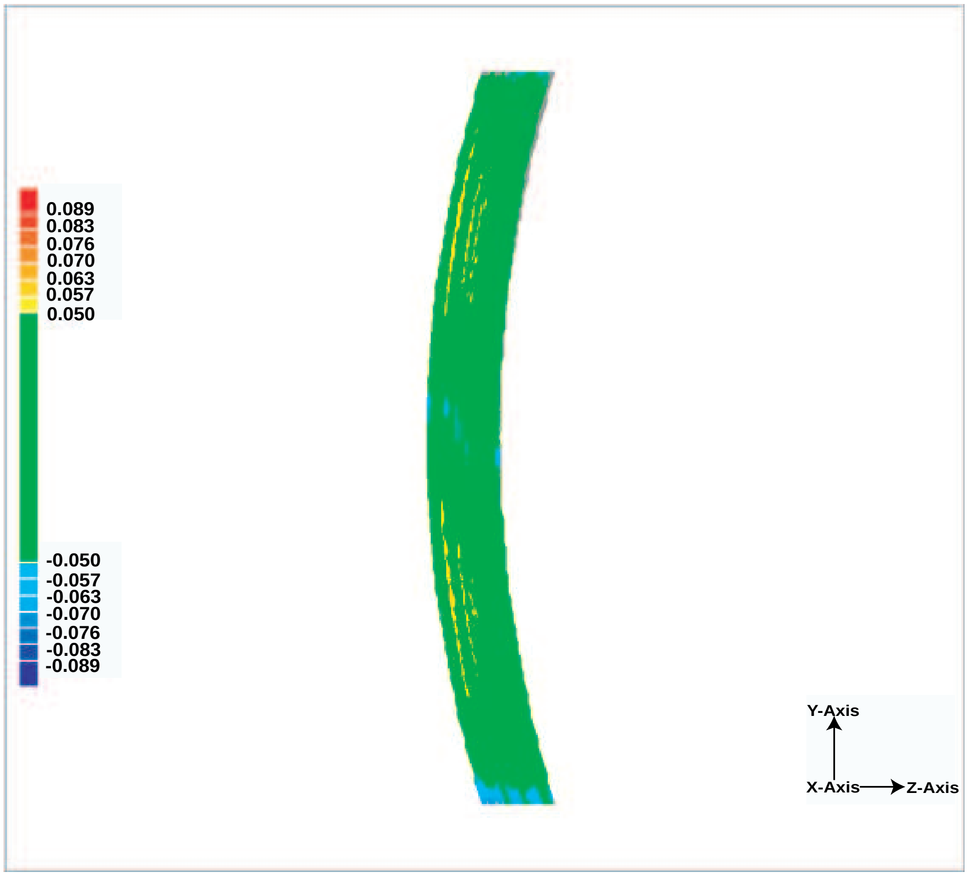


Fig. 48. Left view

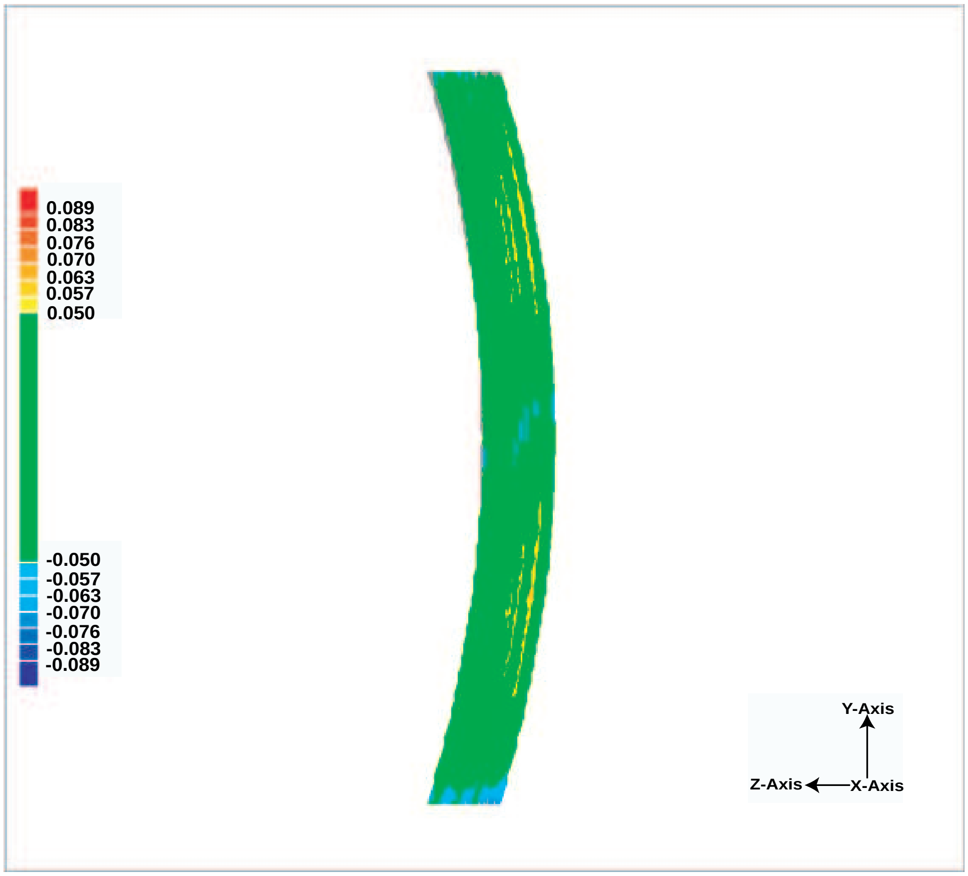


Fig. 49. Right view

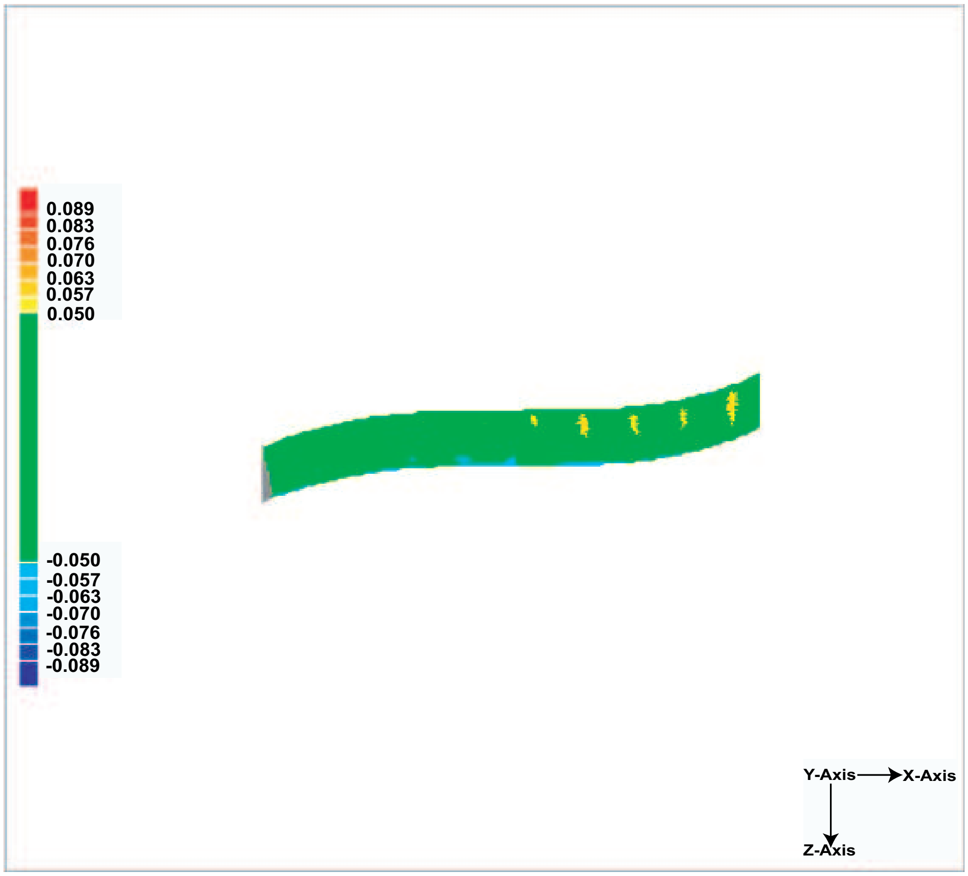


Fig. 50. Top view

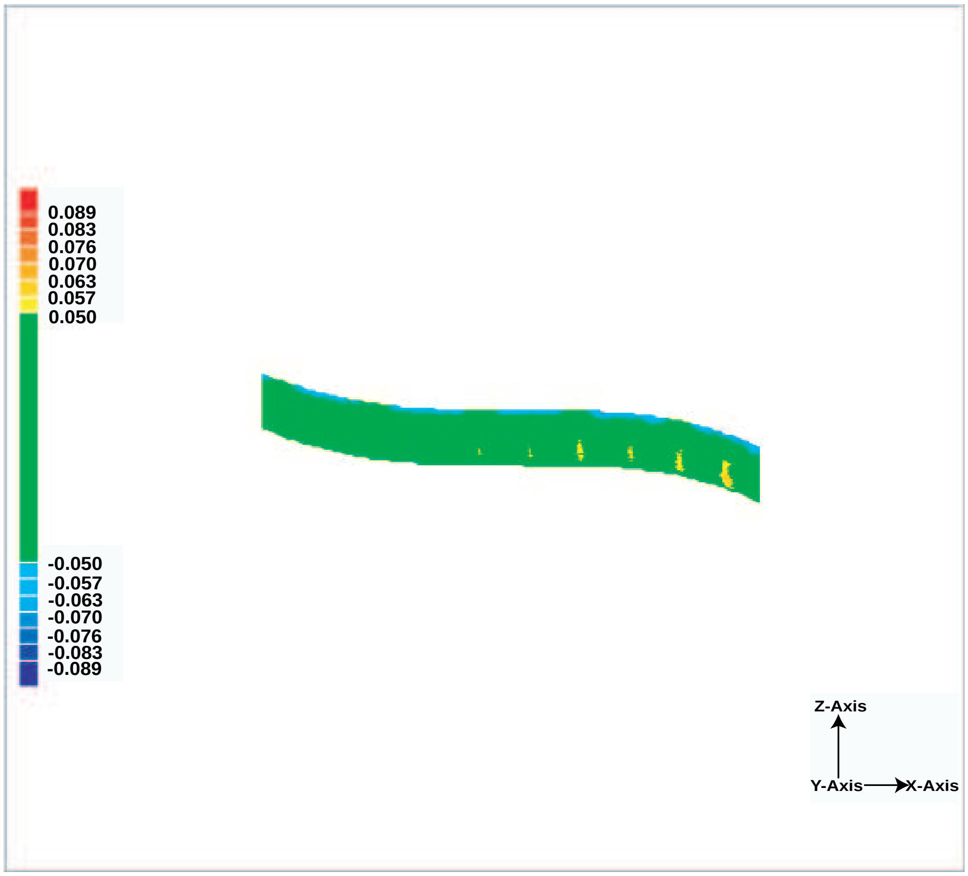


Fig. 51. Bottom view

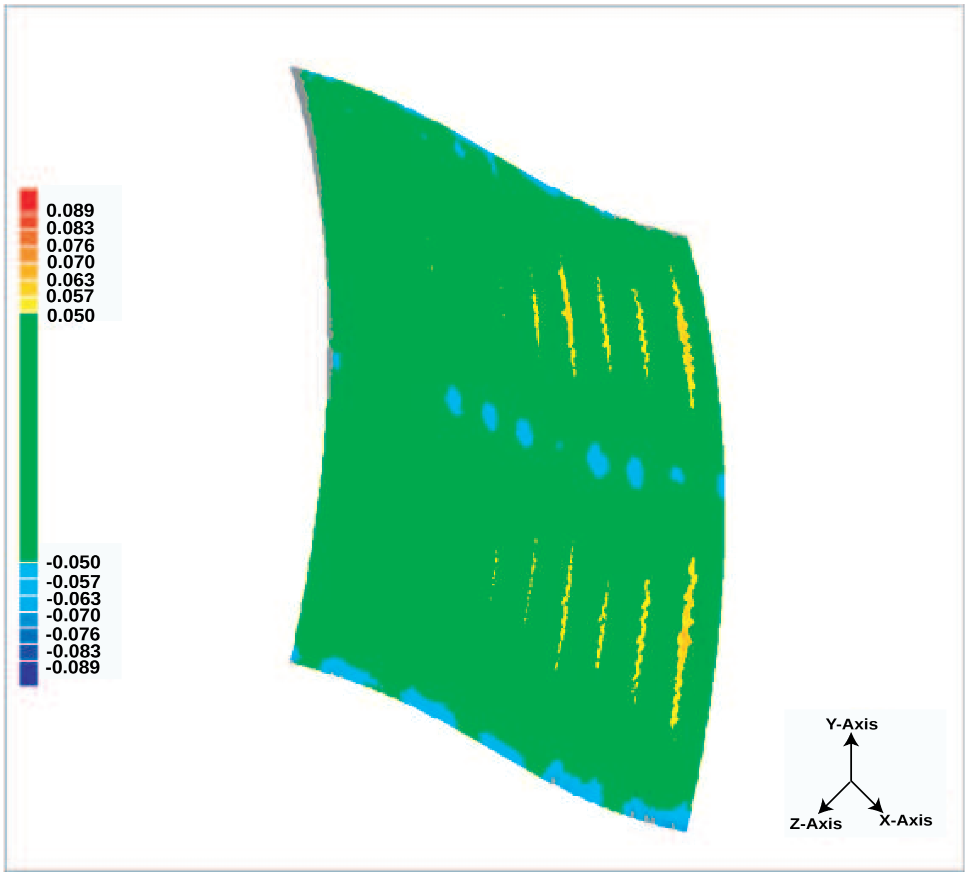


Fig. 52. Isometric view

VITA

Young Keun Choi received his bachelor's and master's degree in industrial engineering in 1993 and 1999 from Konkuk University in Seoul, Korea. He joined the industrial engineering graduate program at Texas A&M University in 1999 to pursue his academic goal. As a graduate assistant lecturer and a teaching assistant, he taught engineering economy and manufacturing automation and robot (Lab) at the undergraduate level. His research and teaching interests include CAD/CAM, computer integrated manufacturing, manufacturing system design and analysis, automation and robot, and virtual manufacturing. He received the Distinguished Graduate Student-Teaching Award for 2004 by the Association of Former Students of Texas A&M University. Young Keun is a member of the Institute of Industrial Engineers (IIE). His permanent address is 31-1 7/1, imdang-dong, Kangnung, Kangwon, South Korea.

The typist for this thesis was Young Keun Choi.