Fast and Robust Approach to Find the Gouge-free Tool Position of the Toroidal Cutter for the Bézier Surface in Five Axis Machining

by

Mukhmeet Singh

A thesis
presented to the University of Waterloo
in the fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

One of the approach used for tool path generation for Bézier surfaces is the Multipoint machining (MPM) approach, in which the toroidal cutter touches the machined surface at two points of contact. Multipoint machining helps in reducing the machining time by providing the tool path data that machines the surface in wider strips positioning the tool in the close proximity to the surface. The tool path generation using MPM is computationally expensive and time consuming, as it involves the solving of non-linear transcendental equations that require numerical methods. Numerical method such as Newton's method are a time consuming and iterative process, and are not always able to give a solution. In this work, two methods, the 'Drop, Rotate and Drop (DRD) method' and the 'Vertical and Circular Ray Firing (VCRF) method', are developed, implemented and tested on bi-cubic Bézier surfaces using a Hi-Dyn tilt-rotary simultaneous five axis machining center. These methods follow the Multipoint machining approach. The DRD method limits the use of Newton's method for convergence to the solution of two unknowns or variables. Whereas, the VCRF eliminates the use of Newton's method for obtaining the solution and instead uses the implicit equations for firing the rays vertical or circular from the surface towards the toroidal cutter surface. Hence, the methods developed in this work give a fast and robust approach for generating tool path data for the Bézier surfaces.

Acknowledgement

I would like to express my sincere gratitude to my mentors; Professor Sanjeev Bedi and Professor Stephen Mann, for the support they have provided me. I am highly obliged for the guidance they have given to me, without which this thesis would not have been possible. For all their help throughout the years, I am indebted to them.

I am also thankful to Jason Benninger, for his support in machining the surfaces and obtaining the CMM data of the surfaces.

Table of Contents

List of Figures

List of Tables

Complexity is Good but Simple is Genius

CHAPTER 1 Introduction

Machining of curved surfaces in the dies and molds industries is a time consuming task. A number of researcher have made efforts to reduce the machining time for these curved surfaces and its preprocessing without losing the desired surface finish. Surface finishing and machining time are inversely proportional to each other. Higher finishing requires higher machining time. Moreover, to get better surface finish, the toolpath for machining is required to be fine and closely packed with a smaller side and forward step. One way of reducing the machining time is to remove unwanted material in wider strips in close proximity to the finished surface. Wider strips give the flexibility to increase the side step for the toolpath generation. In the dies and molds industries tensor product surfaces are used, which can be machined using wider strips.

Two types of machines, 3-axis and 5-axis machines, are common in manufacturing industries. 5-axis machining with their additional degrees of freedom in rotation and tilting about the z and x axis, respectively, gives an edge in the form of flexibility in machine kinematics as compared to 3-axis machines which only have three degrees of freedom with linear motions in the x, y and z axes. This enhanced flexibility in 5-axis machines allows the tool to be positioned in close proximity to the desired surface, which leads to machining with wider strips and fewer passes[1], [2].

To maintain the required surface finish while machining with wider strips the tool geometry should resemble the surface geometry at the point of machining. For this match, a number of methods were developed to place the tool geometry as close as possible to the surface geometry. The popular methods for placing the tool to the close proximity of surface geometry are the Principal Axis Methods (PAM) and the Multi-Point Machining method (MPM) [3-11]. The radiused end mill, also known as toroidal cutter, is used for the toolpath generation with these methods. A Toroidal cutter provides suitable variations in curvature that can be matched with the surface geometry, and also gives a generalized tool profile from which other tools can be derived.

The principal axis method gives tool positions with single point of contact but can lead to gouging on the surface if not checked properly. Multipoint method gives gouge free

tool positions with two point of contact but leads to solving complex, non-linear, transcendental equations for tool path generation. Duvedi et al. [6,15] gave a numerical method for solving these higher order, non-linear, transcendental equations using Newton's method.

1.1 Objective

Newton's method helps to make calculations easy, but it leads to higher computational time, with a proneness to not converge and giving incorrect solutions at times. Hence the objective of this work is to reduce or eliminate the use of Newton's method for finding the gouge free position of the tool over a Bézier surface.

Newton's method helps to solve the higher order non-linear transcendental equations numerically, but gouge checking is still required for making sure that the tool position found is not overcutting at the points of contact. This additional checking of tool positions for gouging adds to the computational time along with the time taken by the Newton's method for converging and giving the solution. The purpose of my work is to develop a method that is purely based on gouge checking without using Newton's method to position the tool over the Bézier surface.

1.2 Organization

This thesis is laid out in 6 chapters. Chapter 1 introduces the work and its objectives. Chapter 2 gives the background of the topic and the prior research that has been done so far and is related to this work, concluding with a gap in that literature. Chapter 3 describes the working of the Drop, Rotate and Drop (DRD) method and gives its algorithm. Chapter 4 describes the working of Vertical and Circular Ray Firing (VCRF) method and gives the algorithm of the same. Chapter 5 gives the implementation of both the methods and discusses the results obtained from the implementation. The thesis is concluded in Chapter 6.

CHAPTER 2 Background

2.1 Toolpath Generation

A toolpath consists of sequential tool positions that describe the location and orientation of the tool over the surface, and that a tool follows to machine a gouge free desired surface. The motion of the tool between the two successive tool positions is linear.

Three elements are required for generating the tool path a) definition or type of surface to be machined, b) geometry of the tool used and c) footprint that the tool should follow.

2.1.1 Surface Definition

A number of CAD data structures can be used for the definition of the surface or the geometry to be machined. The most common format, for toolpath generation, is the STL format. In STL format the surface is defined as a set of triangles. A large number of the triangles results in higher accuracy and finishing of the surface but the computational time for the toolpath generation increases.

In the dies and mold industry, sculptured surfaces are commonly defined using parametric surfaces such as Bézier Surfaces, B-Spline Surfaces and Non-uniform rational B-spline surfaces (NURBS). In this work *bi-cubic Bézier surfaces* are used and the mathematical model for a Bézier surface is given by

$$\vec{S}_{(u,v)} = \sum_{i=0}^{n}\sum_{j=0}^{n} \vec{P}_{(i,j)}\, B_{(i,n)}(u)\, B_{(j,n)}(v) \tag{1}$$

for $0 \leq (u,v) \leq 1$; where $\vec{P}_{(i,j)}$ are the *control points* of the surface, n is the degree of the surface, e.g., for cubic $n = 3$ or for quadratic $n = 2$, and $B_{(i,n)}(x)$ are the Bernstein functions given by

$$B_{(i,n)}(x) = \binom{n}{i} x^i (1-x)^{(n-i)}. \tag{2}$$

2.1.2 Tool definition

The most common types of tools used in the industry are a) Ball nose end mill, b) Flat end mill and c) Radiused end mill. In this work, the tool path is generated for the radiused end mill. Geometrically, the radiused end mill can be represented as a torus, as shown in *Figure 2.1*.

The toroidal cutter is defined with the two radii, R_i – radius of the minor circle and R_o – radius of the major circle. The minor circle with radius R_i represents the *pseudo-insert* that corresponds to the cutting edge on the physical cutter. The pseudo-insert sweeps a torus around the circle with radius R_o known as the major circle.

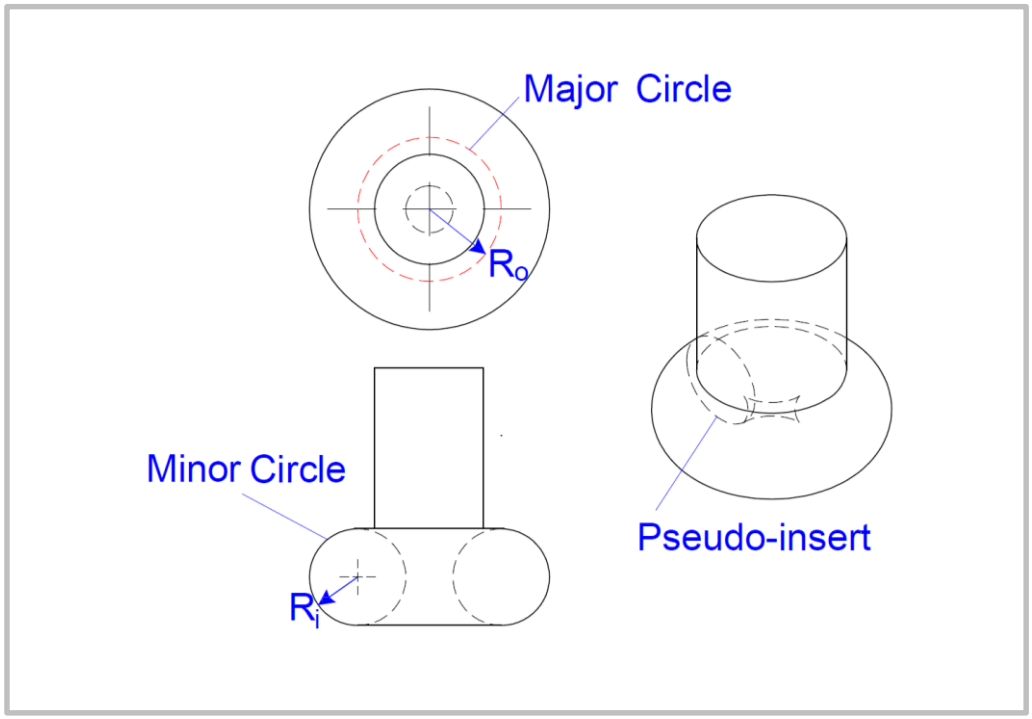


Figure 2.1 Geometry of Toroidal Cutter used for Toolpath Generation

Toroidal cutters can be represented as a general end mill tool. The flat end mill and ball nose end mill are special cases of the toroidal cutter. For a model of flat end mill, $R_i = 0$ and for ball nose end mill, $R_o = 0$. This is shown in *Figure 2.2*.

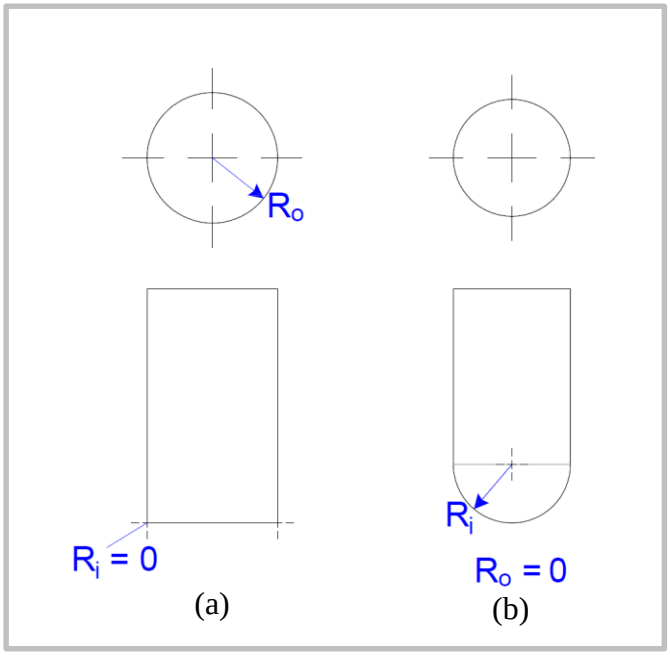


Figure 2.2 (a) Flat end mill Cutter, (b) Ball nose end mill Cutter

2.1.3 Toolpath Footprint

The *Toolpath Footprint* is the sequence or pattern that the tool follows moving over raw stock during machining. The sequence for the tool positions in a toolpath is guided by the footprint laid on (the XY) plane perpendicular to the tool axis, as shown in *Figure 2.3*. The concept of parallel footprint is used in this work, with the X- axis as the side step direction and the Y- axis as the feed forward direction. The passes in the Side step direction are separated by $Side_{step}$ and the tool position in the feed forward direction are separated by $Forward_{step}$ interval.

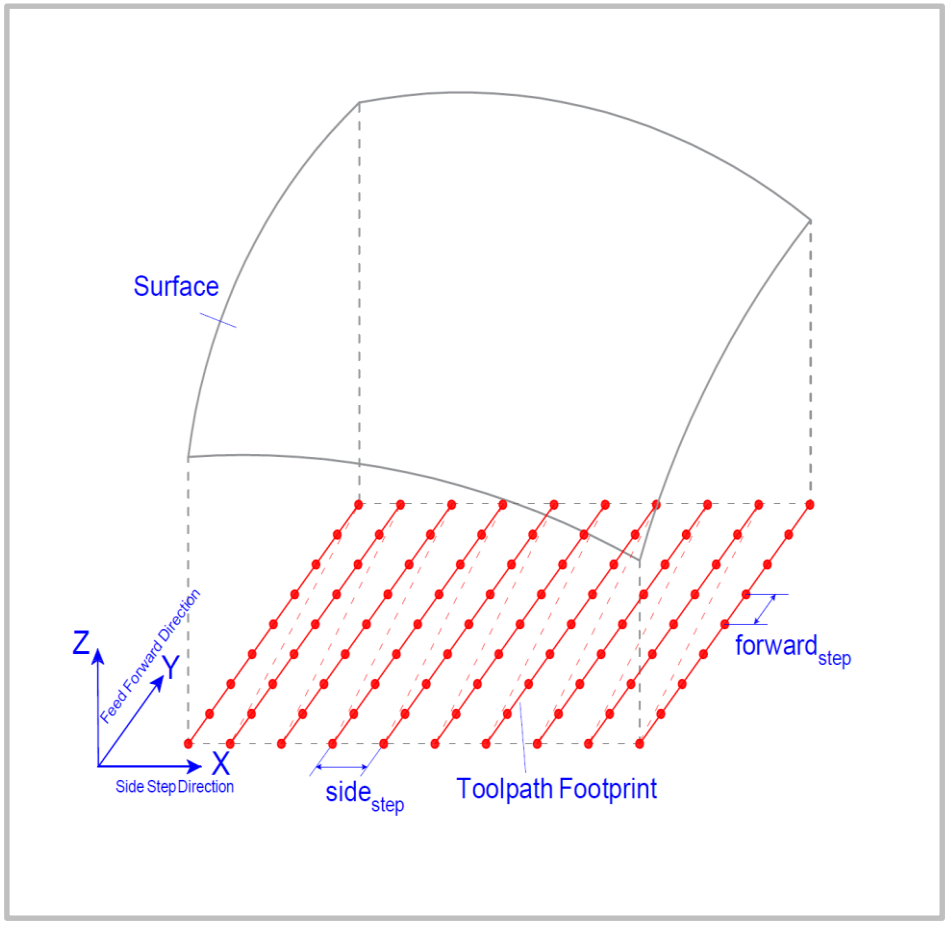


Figure 2.3 XY Parallel Toolpath Footprint used for toolpath planning

2.2 Prior work

Positioning the tool in close proximity to the surface can be achieved by two methods. In the first method, the curvature of the tool is matched with the curvature of the surface. This method is known as the Principle Axis Method (PAM). After matching the curvature of tool and surface, the tool may still gouge the desired surface. Each tool position is checked for gouging before inclusion in the tool path[3]–[5]. The second method is based on the idea that an appropriately inclined tool will make tangential contact at least at two different points on the surface at the same time. The tool

positioning method based on this attribute is known as the MultiPoint Method (MPM)[6]–[11]. Both methods are based on cutting with the radius end mill cutter.

The MPM method was proposed by Warkentin [11], [12], in which an iterative process was used to find the point of contacts algorithmically, and an algebraic library was used to obtain the solutions. The solution was chosen using an optimization method. The procedure was lacking in robustness and was slow. To increase the speed and robustness of the tool path planning, the machining of STL surface was proposed[6]–[8], [13]-[14]. Duvedi et al. [6], [7] presented the Drop and Tilt method to obtain two points of contact for any triangulated surface. In their method the tool was first dropped to find the first point of contact with the triangular mesh and then tilted until the tool touches at a second point of contact on the triangulated surface. The algorithm is run on all the triangles falling under the shadow of the tool. This method of using STL surfaces gives the exact solutions for both points of contact. As linear equations are used to obtain the solutions the method is both robust and fast method. The accuracy of the solution depends upon the accuracy of triangulated mesh. If the number of triangles used to approximate the surface are few then the accuracy will be poor and facets will be seen on the machined surface. Increasing the number of triangles for better surface finish and accuracy increases the computational time drastically.

Duvedi et al. [15]–[17] extended the Drop and tilt method to Bézier surfaces, which removes the dependency of surface finishing on the triangulated mesh. The DTM approach was numerically implemented and tested on bi-quadratic and bi-cubic Bézier surfaces. The implementation involves simultaneously solving a number of higher order non-linear transcendental equations, which were solved using Newton's method. After solving, gouge checking was done for each tool position. Hence, although the technique is robust and efficient for finding the two points of contact, it took more computation time. Although MPM has been applied to triangulated surfaces and to tensor product surfaces a number of deficiencies exist. Triangulated surfaces by definition are approximate and Newton's method based solutions for tensor product are prone to identifying the incorrect solutions at times. Thus, a better method for the Multi Point tool positioning is still desired for tensor product surfaces.

In this work, new techniques are proposed that eliminate the use of Newton's method to solve the higher order non-linear transcendental equations [16]. The proposed methods are implemented for bi-cubic Bézier surfaces using a console application built in C++. The algorithms are tested by simulating three parts having concave and convex regions using the ToolSim machining simulator tool and by machining successfully on a DMU-80P five axis machine.

CHAPTER 3 Drop Rotate and Drop

In Multi-Point Machining the objective is to determining a gouge-free tool position on a Bézier surface such that the toroidal tool makes tangential contact with the surface at least at two different points. The tool orientation and location are determined in two steps, as proposed by Duvedi et al. [15-16]. The tool is first dropped on the surface and then the surface is tilted until a second point of contact is found. The key algorithm used in both the steps emulates the dropping of a tool along a specified tool axis. This algorithm is referred to as the *tool drop algorithm*. Algorithms varying from the direct solution of algebraic equations to algorithms based on bi-section and Newton's methods have been presented in the literature. The algorithm proposed in this work is based on firing rays from the tool surface and determining the distance to the intersection with surface. The two step algorithm and the drop and tilt processes are detailed below.

3.1 Dropping the Tool

In the first step of the proposed method, the cutting surface of torus is first discretized into an array of points. The points on the torus are given by

$$\overrightarrow{Tool}_{(l,k)} = \overrightarrow{T_1} + \hat{t}\, d_{(l,k)} + \overrightarrow{Tor}(\theta_l, \emptyset_k) \tag{3}$$

where $\hat{t}$ is the unit vector given as $[0,0,1]^T$ and

$$\overrightarrow{Tor}(\theta_l, \emptyset_k) = \{(R_o + R_i \cos\theta_l) \cos\emptyset_k, (R_o + R_i \cos\theta_l) \sin\emptyset_k, R_i \sin\theta_l\} \tag{4}$$

$\theta_l, \emptyset_k \in [0, 2\pi]$, R_i is the radius of the minor circle; and R_o is the major radius, as shown in *Figure 3.2*. The tool axis and the minor circle are coplanar in three-dimensional space.

In the second step, rays parallel to the tool axis are fired towards the surface. The Bézier surface is represented by equation 1.

Figure 3.1 shows a set of rays emanating from points $Tool_{(l,k)}$ in a direction opposite to $\hat{t}$. These rays intersect the surface at certain points on the surface denoted by S with

a subscript, such that $S_{(l,k)}$ corresponds to $Tool_{(l,k)}$, and hence projecting the curved surface of the cutter onto the Bézier surface patch. The projected

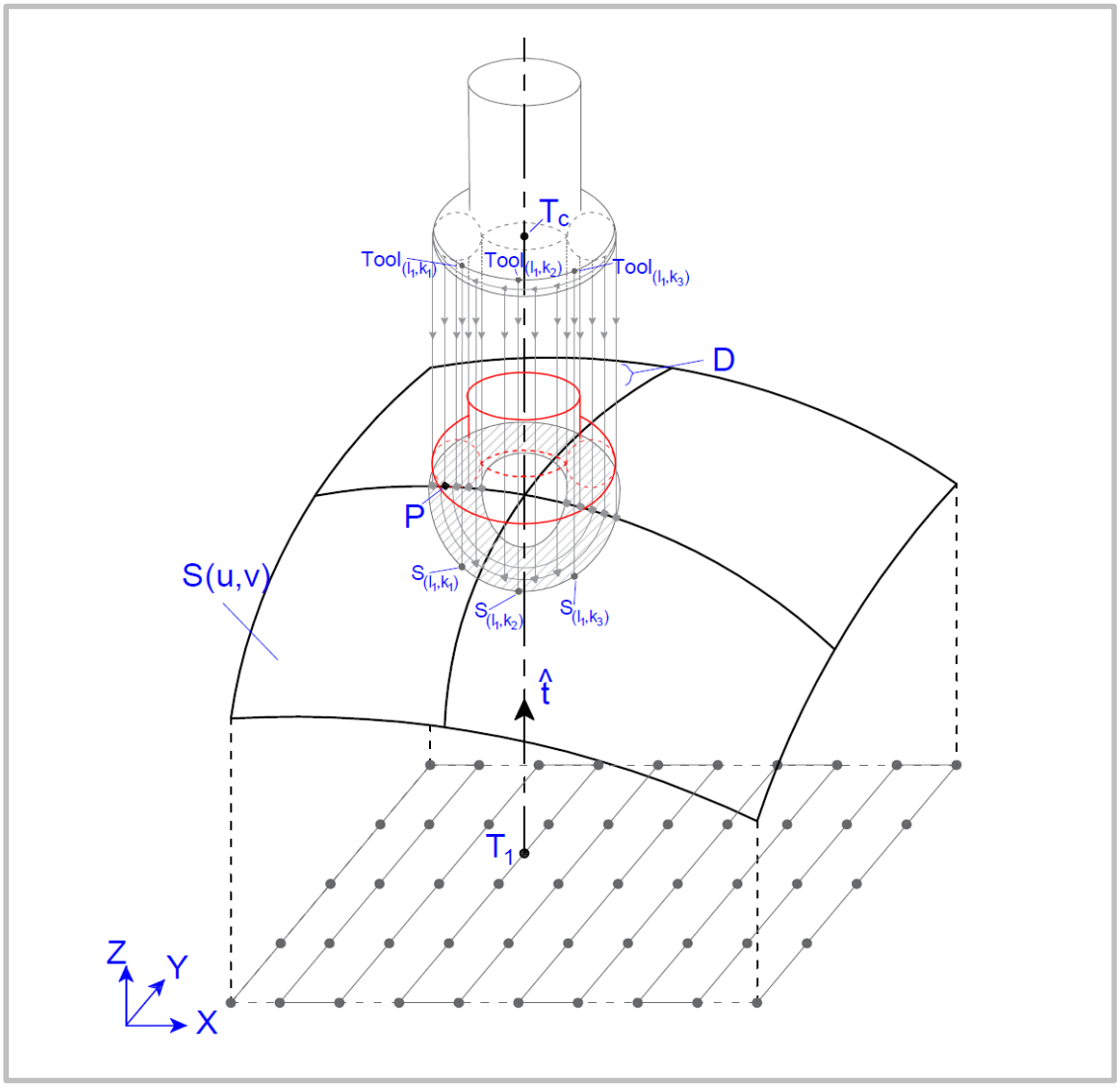


Figure 3.1 : Dropping the tool over the Bézier surface along tool axis $\hat{t}$. A patch formed under the shadow of the tool and drop distance set D obtained by intersection of fired rays from the tool surface onto Bézier surface

patch of the toroidal cutter is shown as hatched area on the surface. The intersection point is the solution of

$$Tool_{(l,k)} = S_{(u,v)} \tag{5}$$

The length of the ray, from the surface to the intersection of the rays with the surface, represents the distance the tool must be dropped to bring that specific surface point in contact with the tool. At each point of the surface this drop distance is different. The drop distance for all the points in the surface array shown in *Figure 3.1* form a drop distance set. The drop distance set D is

$$D = \{d_{(l,k)} \; \forall \; (l = 1 \; to \; n, k \; = 1 \; to \; m)\} \tag{6}$$

where $d_{(l,k)}$ is a solution of obtained from equations (3) and (4) is given by

$$\hat{t} \; d_{(l,k)} = \vec{S}_{(u,v)} - (\overrightarrow{T_1} + \overrightarrow{Tor}(\theta_l, \emptyset_k)) \tag{7}$$

The idea is that at the point of intersection, the surface point and the point on the tool surface are same. The difference of both the points gives the deviation of the points and form three set equations, derived from equation (5) and is a vector equation represented as

$$\begin{bmatrix} Sol_x \\ Sol_y \\ Sol_z \end{bmatrix} = \begin{bmatrix} PointOnSurface_x - Tool_x \\ PointOnSurface_y - Tool_y \\ PointOnSurface_z - Tool_z \end{bmatrix} \tag{8}$$

that must be solved simultaneously for three unknowns $[u, v, d_{(l,k)}]$, where $u \; and \; v$ are the parameters for the surface point and $d_{(l,k)}$ is the drop distance.

The value of $\hat{t}$ is $[0,0,1]^T$ and hence equation (7) can be represented as

$$\begin{bmatrix} 0 \\ 0 \\ d_{(l,k)} \end{bmatrix} = \begin{bmatrix} S_{x(u,v)} - (T_{1x} + Tor_x(\theta_l, \emptyset_k)) \\ S_{y(u,v)} - (T_{1y} + Tor_y(\theta_l, \emptyset_k)) \\ S_{z(u,v)} - (T_{1z} + Tor_z(\theta_l, \emptyset_k)) \end{bmatrix} \tag{9}$$

In equation (9), only the third equation corresponding to the z-direction is a function of the drop distance, $d_{(l,k)}$. Based on this observation the solution of (9) is broken into two steps. In the first step, Newton's method is used to find the values of parameters u and v using the first and second equation of (9). Given a set of non-linear equation Newton's method reduces to:

$$x_i = x_{i-1} - [J]^{-1} F(x_{i-1}) \tag{10}$$

where $x_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$, $F_i = \begin{bmatrix} Sol_x(u_i, v_i) \\ Sol_y(u_i, v_i) \end{bmatrix}$, and

$$J = \begin{bmatrix} \frac{\partial Sol_x}{\partial u} & \frac{\partial Sol_x}{\partial v} \\ \frac{\partial Sol_y}{\partial u} & \frac{\partial Sol_y}{\partial v} \end{bmatrix}$$

J is the Jacobian matrix of these equations and x_i is the vector of unknowns, u, v. The initial value of x_0 is assumed and the subsequent values of x_i are calculated iteratively until they are within a user specified tolerance. In the second step, the values of parameters u and v are used to solve for drop distance $d_{(l,k)}$ in the z direction.

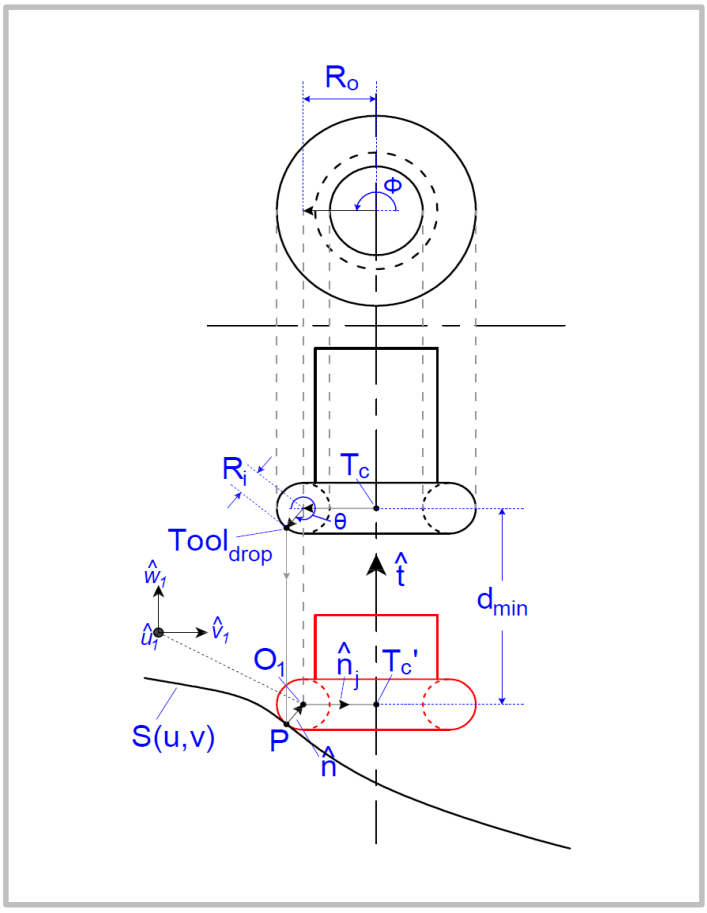


Figure 3.2 Cross-sectional view of tool touching the Bézier surface at P co-planar to the pseudo-insert.

The minimum of the drop distance set $\min\{D\}$, i.e., d_{min}, gives the distance the tool should be dropped to just touch the surface without any gouging. Any distance greater than d_{min} will result in overcutting and any distance less than the d_{min} will result in undercutting. The surface point $S_{(u_1,v_1)}$ corresponding to the d_{min} gives the first point of contact P.

The accuracy with which this shaded area is represented depends on the discretization of the torus surface. Adaptive discretization has been used in this work to obtain greater accuracy and quick convergence.

After dropping the tool, the first point of contact P lies on one of the pseudo-inserts, as shown in *Figure 3.2*. This minor circle is the same size as the insert in a physical tool and as the tool rotates the physical insert coincides with the minor circle as it touches the surface. The center of the pseudo-insert O_1 is given by

$$O_1 = P + R_i\,\hat{n}. \tag{11}$$

$\hat{n}$ is the surface unit normal at the point of contact $S(u,v)$ and is given by

$$\hat{n} = \frac{\frac{\partial S_{(u,v)}}{\partial u} \times \frac{\partial S_{(u,v)}}{\partial v}}{\left| \frac{\partial S_{(u,v)}}{\partial u} \times \frac{\partial S_{(u,v)}}{\partial v} \right|}. \tag{12}$$

The axis of the pseudo-insert passes through O_1 and is perpendicular to the plane containing the pseudo-insert. A rotation of the tool about the pseudo-insert axis will result in tilting the tool but will ensure that the rotated tool still touches the first point of contact $S_{(u_1,v_1)}$ tangentially although at a different point on the pseudo-insert. The ability to tilt the tool while maintaining contact at the first point of contact is used to find the second point of contact. The surface is tilted algorithmically until second point of contact is found.

3.2 Tilting the Surface

To find the second point of contact, the surface is tilted around the pseudo-insert of contact until the surface comes into contact with the tool. A new coordinate frame $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ is created at O_1, the center of the pseudo-insert at the first point of contact, where

$$\hat{w}_1 = \hat{t} \tag{13}$$

$$\hat{u}_1 = \frac{\overrightarrow{(Tc - P)} \times \hat{w}_1}{\left|\overrightarrow{(Tc - P)} \times \hat{w}_1\right|} \tag{14}$$

$$\hat{v}_1 = \hat{w}_1 \times \hat{u}_1. \tag{15}$$

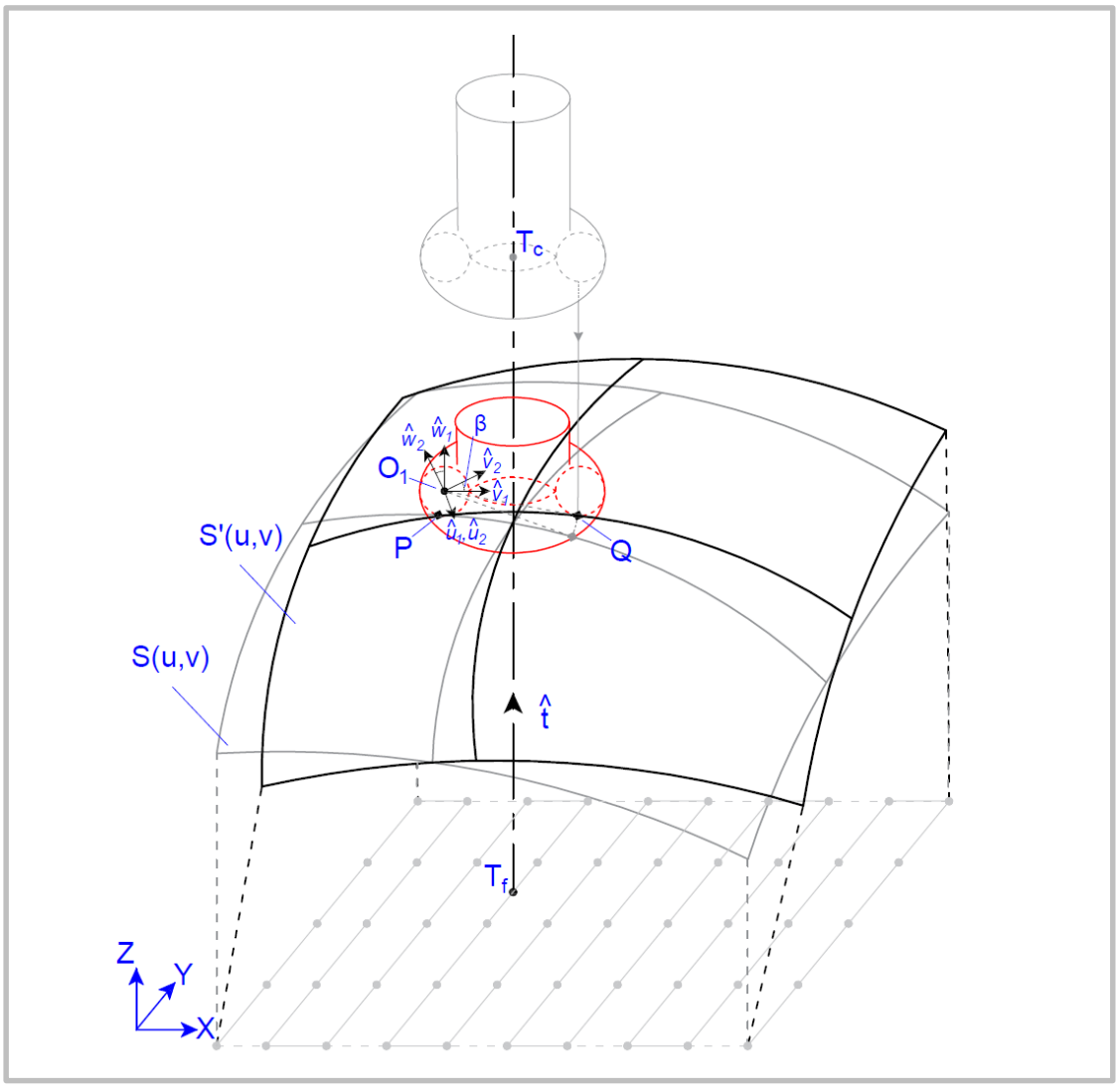


Figure 3.3 Tilting the surface with β around O_1 about axis $\hat{u}_1$

The surface is then defined in the new coordinate frame and is then rotated about $\hat{u}_1$ with an angle β to form a new rotated surface $S'_{(u,v)}$. This rotation leads to the formation of third rotated coordinate frame $\{\hat{u}_2, \hat{v}_2, \hat{w}_2\}$ as shown in *Figure 3.3*, where,

$$\hat{u}_2 = \hat{u}_1 \tag{16}$$

$$\hat{v}_2 = \cos\beta\ \hat{v}_1 + \sin\beta\ \hat{w}_1 \tag{17}$$

$$\hat{w}_2 = -\sin\beta\ \hat{v}_1 + \cos\beta\ \hat{w}_1 \tag{18}$$

To rotate a tensor product surface, only the control points needed to be rotated. The surface is rotated initially with a minimum rotation angle β and the rotation is increased incrementally until the second gouge free point of contact Q is found. This results in two points of contact P and Q, at which the surface $S_{(u,v)}$ and toroidal surface of the tool are tangent to each other.

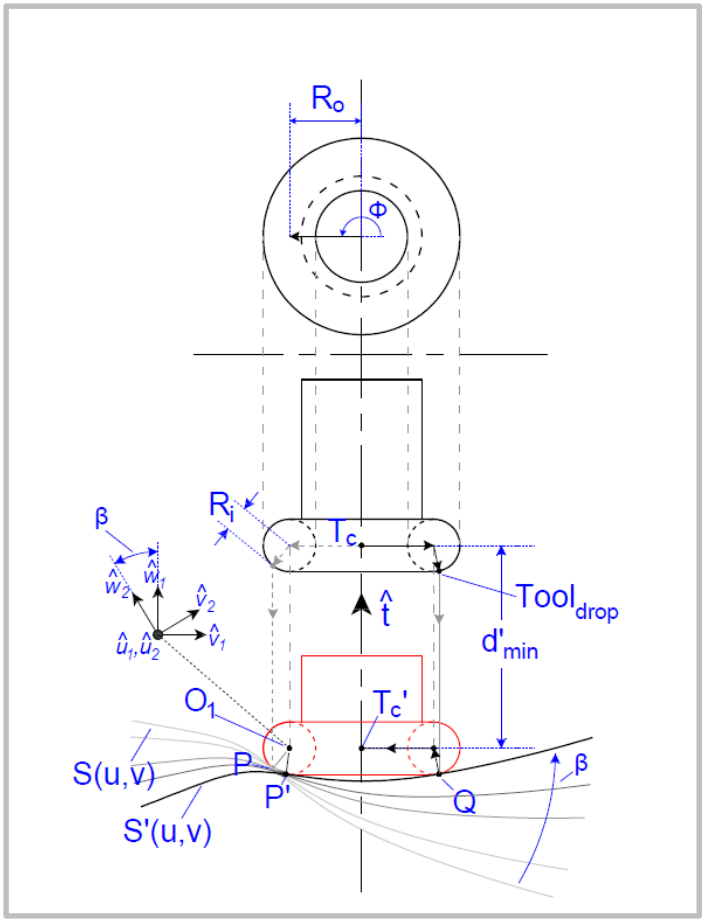


Figure 3.4 Cross sectional view co-planar to the pseudo-insert after tilting the surface giving the first and second point of contact, P and Q

As the surface is tilted in $\{\hat{u}_2, \hat{v}_2, \hat{w}_2\}$, the pseudo-insert maintains its contact with the surface; however, the position of first point of contact on the pseudo-insert will change sliding along the pseudo-insert from *P* to *P'*, as shown in *Figure 3.4*. The surface $S_{(u,v)}$ will stay tangent to the toroidal surface of the tool, as the rotation is about the center of the pseudo-insert O_1.

The bisection method is used to determine the rotation angle that gives the second point of contact Q. In the bisection method the surface is rotated by angle β and the tool is dropped over the rotated surface S' in the same manner as the first point of contact. If the point of contact of the dropping tool is the same as the first point of contact, then the surface is tilted further by increasing β otherwise the rotation angle is halved to $\beta/2$ and the process is continued until the second point of contact is found.

The condition for the second point of contact Q to be gouge free is checked by comparing the two heights of drop, i.e., d_{min} and d'_{min} at every stage of rotation. The rotation angle where both the heights are equal gives the second point of contact.

3.3 DRD Algorithm

The pseudo-code presented in *Figure 3.5* gives the algorithm for the proposed DRD (Drop, Rotate and Drop) concept. It starts with generating the toolpath footprint (2.0), which, in this case, are parallel lines in the u-v plane. These parallel lines map to lines in the XY domain that extends between x_{min} to x_{max} and from y_{min} to y_{max}, as shown in *Figure 2.3*. The toolpath footprint is discretized with a spacing of $Side_{step}$ in X direction and $forward_{step}$ in Y direction.

At each of the discretized points on the tool path footprint (4.0), the toroidal tool is positioned (4.1) with the center of the tool $\vec{T}_c$ at a specified height above the Bézier surface such that the surface patch is sandwiched between the toolpath footprint plane and the initial tool position $\vec{T}_c$, as shown in *Figure 3.1*. Tool drop algorithm is applied (4.2) for the tool position $\vec{T}_c$ which gives the position parameters $[u_1, v_1]$ and drop distance d_{min} for the first point of contact $\vec{P}$. The algorithm for dropping the tool is explained in section 3.3.1.

After obtaining the first point of contact $\vec{P}$, center $\vec{O}_1$ of pseudo-insert is found (4.4) and the Bézier surface is rotated by angle β about an axis pointing in direction of $\hat{u}_1$ of

1.0 Define array of Control Points of surface: $P[n][n]$

2.0 Generate Toolpath Footprint:
$FootPrint[k][2] = GenerateFootprint(side_{step}, forward_{step}, x_{max}, y_{max})$

3.0 Set
$z_{max} = maximum\_Z(P[n][n]) + 50, Origin = [0,0,0], \hat{t} = [0,0,1], R_i = 6.0$

4.0 $For\ i\ =\ 1;\ i\ \leq\ k$

4.1 $T\ =\ [FootPrint[i][1],\quad FootPrint[i][2],\quad z_{max}]$

4.2 $[u_1, v_1, d_{min_i}] = ToolDrop(T, P[n][n], \hat{t})$

4.3 Get Normal and first surface contact point:
$[P_i, \hat{n}_{P_i}] = S(u_1, v_1, P[n][n])$

4.4 $O_i\ =\ P_i + R_i\hat{n}_i$

4.5 Set coordinate frame $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$:
$\hat{w}_1\ =\ \hat{t}, \hat{u}_1\ =\ \frac{\overrightarrow{(T-P_i)} \times \hat{w}_1}{|\overrightarrow{(T-P_i)} \times \hat{w}_1|}, \hat{v}_1\ =\ \hat{w}_1\ \times\ \hat{u}_1$

4.6 Set $\beta_{min} = 0, \beta_{max}\ =\ \pi/4\ , \beta\ =\ \beta_{max}, d'_{min}\ =\ d_{min} + 50, j = 1, counter = 1$

4.7 $while\ counter\ <\ 10$

4.7.1 $\beta = (\beta_{max} + \beta_{min})/2$

4.7.2 Rotate Bézier Surface around $\hat{u}_1$ axis at O_1 with angle of rotation β :
$P'[n][n]\ =\ rotate(P[n][n], \beta, \hat{u}_1, O_1)$

4.7.3 $[u_2, v_2, d'_{min_j}] = ToolDrop(T, P'[n][n], \hat{t})$

4.7.4 $\in = d_{min_i} - d'_{min_j}$

4.7.5 $If\ (\in > 0.01)\ then$
$\beta_{max}\ =\ \beta$
$Else\ If\ (\in \leq 0.01)\ then$
$\beta_{min}\ =\ \beta$

4.7.6 $Q_j = S(u_2, v_2, P[n][n])$

4.7.7 $If\ (Q_j\ =\ Q_{j-1})\ then$
$counter = counter + 1$
$Else$
$counter = 1$

4.7.8 $Set\ j = j + 1$

4.8 Get Normal and second surface contact point:
$[Q_i, \hat{n}_{Q_i}] = S(u_2, v_2, P[n][n])$

4.9 Rotate tool axis around global x axis at $Origin$ with angle of rotation β:
$\hat{t}' = rotate(\hat{t}, \beta, x, Origin)$

4.10 Print Tool position data:
$print(P_i, \hat{n}_{P_i}, Q_i, \hat{n}_{Q_i}, \hat{t}')$

Figure 3.5 Pseudo-code for the Algorithm for getting gouge free tool position over the Bézier surface

newly formed coordinate frame, (4.5). Rotation of the surface about $\hat{u}_1$ with pseudo-insert $\vec{O}_1$ as the center of rotation ensures the tangency of the toroidal cutter with the surface to be machined at the first point of contact, even though the first point of contact $\vec{P}$ moves along the pseudo-insert. To obtain the required value of rotation for which the tool touches the surface at the second point of contact without any gouging and at the same time maintains the condition of tangency with the first point of contact, the bisection method is used, (4.7). The minimum and maximum value of rotation angle (4.6), i.e., β_{min} and β_{max} is set to 0 and $\pi/4$, respectively. Rotation beyond this defined limit will not yield any viable solutions, as β_{min} represents a 3-axis machine and β_{max} represents the physical limit of a 5-axis machine tilt table. The rotation angle β_{min} or close to β_{min} will yield the first point of contact as the contact point. Whereas, with β_{max} angle of rotation, the tool will touch the surface at a second point of contact but will require a smaller drop distance than the first point of contact. Thus the rotation angle giving two points of contact lies between β_{min} and β_{max}.

The surface is rotated with the average of minimum and maximum rotation angle (4.7.1). After rotating the surface, tool is dropped on the rotated surface and the drop distance d'_{min} for the shortest intersected ray from the tool to the rotated surface is obtained, which is then compared with the previously obtained drop distance d_{min} from dropping of the tool over the surface without any rotation (the first point of contact). The comparison of the drop distance will decide the new values of β_{min} and β_{max} for the next iteration. Now the maximum and minimum limits of rotation angle, i.e., β_{min} and β_{max} are reset according to ϵ, which is the difference of drop distance at first point of contact and drop distance after the rotation of surface, (4.7.4) and (4.7.5). A tolerance of $\epsilon = 10^{-2}$ was used to accommodate numerical errors in computation. Resetting the β_{min} and β_{max} after every iteration brings the algorithm closer to the converged solution quickly.

As β_{min} and β_{max} converge, the rotation angle obtained at every iteration will yield a second point of contact close to one from the previous iteration but numerically different from each other. The difference is small but numerically sufficient to keep the loop running for the bisection method. A hard limit is set to break the loop after achieving

the result in the zone of the required accuracy defined by the user. For this, the point of contact obtained in the current loop iteration after rotation $\vec{Q}_j$ is compared with the point of contact $\vec{Q}_{j-1}$ from the previous loop iteration (4.7.7); and if both points of contact are physically the same a counter is increased or if the points of contact differ then the counter is reset. The loop exits after the counter exceeds the limit defined by the user, which in this work was 10 (4.7). As the loop breaks, the rotation angle at that stage is the smallest possible rotation needed to produce the second gouge free point of contact. At this stage the two point of contacts $\vec{P}$ and $\vec{Q}$ needed to define the tool position for Five axis machining are obtained.

This method described above is applied to all points in the tool path footprint to create a 5-axis tool path for machining the desired surface.

3.3.1 Tool Drop Algorithm

The tool is dropped over the surface using ray firing, discussed in the previous section, to find the first point of contact. The algorithm for dropping the tool is given in the pseudocode shown in *Figure 3.6*. The tool drop algorithm starts with the discretization of the toroidal surface of the tool (1.0 and 2.0). Theoretically, the toroidal surface of the tool can be discretized into an infinite number of points from which rays could be fired toward the surface; but this will lead to higher computational time and a formation of large drop distance set {D}. To keep the computational time to a minimum, the range is user specified. In this work, a tool with a small hollow in the bottom plane, as shown in *Figure 3.4*, is used. Furthermore, the target parts for 5-axis machines are dies and molds that can be accessed from above, thus the range for θ is set within $3\pi/2$ to 2π. There are no such restrictions on $\emptyset$ and it can range from 0 to 2π. This defined range for θ and $\emptyset$ is where the tool will make contact with the surface without any gouging. In the tool drop algorithm the range is divided in 10 equal parts for both θ and $\emptyset$. At the onset, for the first point of contact, these 121 distinct points over the toroidal surface are considered.

$ToolDrop(T, P[n][n], \hat{t})$

1.0 Set $\theta_{max} = 2\pi, \theta_{min} = 3\pi/2\,, \Delta\theta = (\theta_{max} - \theta_{min})/10$
2.0 Set $\emptyset_{max} = 2\pi, \emptyset_{min} = 0, \Delta\emptyset = (\emptyset_{max} - \emptyset_{min})/10$
3.0 Set d_{min} larger than the Z coordinate of T vector
4.0 Initialize $u = 0.1, v = 0.1, Sol = [0,0,0]$
5.0 $For\ iterations = 1, iterations \leq 5$
  5.1 $For\ \theta = \theta_{min}, \theta \leq \theta_{max}$
    5.1.1 $For\ \emptyset = \emptyset_{min}, \alpha \leq \emptyset_{max}$
      5.1.1.1 Define '$Tool_{(l,k)}$' from equation (3)
        $Tool_{(l,k)} = T + Tor(\theta, \emptyset)$
      5.1.1.2 $PointOnSurface = S(u, v, P[n][n])$
      5.1.1.3 Solution for equation (5)
        $Sol = PointOnSurface - Tool_{(l,k)}$
      5.1.1.4 $While\ (Sol[1]\ or\ Sol[2] > 0)$
        Assign $function = [Sol[1], Sol[2]]$
        Get derivative of surface point at (u, v)
        $$\begin{bmatrix} \partial S/\partial u \\ \partial S/\partial v \end{bmatrix} = derivative(u, v, P[n][n])$$
        Get Jacobian of surface point:
        $$J[2][2] = \begin{bmatrix} \partial S[1]/\partial u & \partial S[2]/\partial u \\ \partial S[1]/\partial v & \partial S[2]/\partial v \end{bmatrix}$$
        Solve
        $$\begin{bmatrix} temp_u \\ temp_v \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} - ([J]^{-1} \times [function])$$
        Reassign u, v as:
        $u = temp_u, v = temp_v$
        Repeat Steps 5.1.1.2 and 5.1.1.3
        Store minimum Drop Distance and associated surface and tool parameters
        $If\ (d_{min} > (-Sol[3]))\ then$
          $d_{min} = -Sol[3], u_{sol} = u, v_{sol} = v, temp_\theta = \theta, temp_\emptyset = \emptyset$
  5.2 Reset
    $\theta_{max} = temp_\theta + \Delta\theta, \theta_{min} = temp_\theta - \Delta\theta, \Delta\theta = (\theta_{max} - \theta_{min})/10$
  5.3 Reset
    $\emptyset_{max} = temp_\emptyset + \Delta\emptyset, \emptyset_{min} = temp_\emptyset - \Delta\emptyset, \Delta\emptyset = (\emptyset_{max} - \emptyset_{min})/10$

Return $[u_{sol}, v_{sol}, d_{min}]$

Figure 3.6 Pseudo-code for the Tool Drop (firing of rays) Algorithm

For initialization, surface parameters $u\ and\ v$ are set to any arbitrary value ranging for 0 to 1; in this work, both $u\ and\ v$ values are set to 0.1 initially (4.0). Now for the given $\theta\ and\ \emptyset$ the point on the toroidal surface of the tool $Tool_{(l,k)}$ is obtained as given in Step (5.1.1.1). After getting the point on the toroidal surface of the tool, the point on the surface is also obtained from the initial seed of the surface parameters $u\ and\ v$ (5.1.1.2) and the solution of equation (5) given by equation (8) is obtained (5.1.1.3).

Step (5.1.1.4) of the pseudo-code given in *Figure 3.6* is used to find the intersection point on the surface and the shortest distance of the fired rays. $[u_{sol}, v_{sol}]$ are used to keep track of the parameters used to obtain the surface point for corresponding shortest fired ray having drop distance stored as d_{min}.

If $Sol_z(i)$ is less than the $Sol_z(i-1)$ then it is stored as d_{min}, which is used to keep the track of minimum drop distance of set {D}. The parameters corresponding to the d_{min} are also stored (u_{sol} and v_{sol}), as they give the location of first point of contact on the Bézier surface.

The drop distance is calculated at all these points and stored in the drop distance set {D}. The first point of contact is near the minimum member of this set. To find this point with accuracy, the minimum and the maximum values, i.e., $[\theta_{min}, \theta_{max}]$ and $[\emptyset_{min}, \emptyset_{max}]$ are redefined and finely discretized around the minimum drop distance point and the iteration process is repeated. Redefining of ranges for θ and $\emptyset$ is given in Step (5.2) and (5.3) of the pseudocode in *Figure 3.6*. The range of toroidal angles is condensed five times or until the first gouge free point of contact is found within the user specified tolerance.

CHAPTER 4 Vertical and Circular Ray Firing

The DRD method explained in previous chapter limits the use of Newton's method for getting the solutions of two unknowns for non-linear equation instead of three unknowns as explained by Duvedi et al [15]. The reduced dependency on Newton's method and rotation of the surface instead of the tool geometry for solving the non-linear equation is less prone to not giving the solution, but it is still computationally slow. Even though the computational effort is reduced by limiting the use of Newton's method for two unknowns, the computational time is still too high as the surface control points are rotated again and again in the bisection method for the calculation of rotation to obtain the second point of contact. To overcome this, a method is developed that completely eliminates the use of Newton's method and does not require any repetitive rotations of either the tool or the surface. The tool position giving the tool orientation and location is calculated in two steps. An implicit equation in Cartesian coordinates is used for the toroidal cutter definition, which is radially symmetric about the z-axis.

4.1 Motivation

In MultiPoint machining (MPM) the toroidal cutter touches the surface at two points of contact. Finding the points on the surface making contact with the toroidal cutter, resulting in machining, is a sequential process. *Figure 4.1* shows the stage wise process of finding the two points of contact for the vertical and circular ray firing method. The toroidal tool is placed over the Bézier surface at T_c' as the center of the tool and both the points of contact P and Q over the Bézier surface where the tool is supposed to touch the surface without any gouging are shown in the *Figure 4.1(A)*. Now for getting the first point of contact P in this method, the vertical rays are fired from the surface towards the tool; as opposite to the DRD method, in which, the rays are fired from the toroidal surface towards the Bézier surface. The complete process of vertical ray firing is given in the next section. After getting the required drop distance d, the tool is dropped down vertical with the drop distance and touches the Bézier surface at point P giving the location and position of the pseudo-insert.

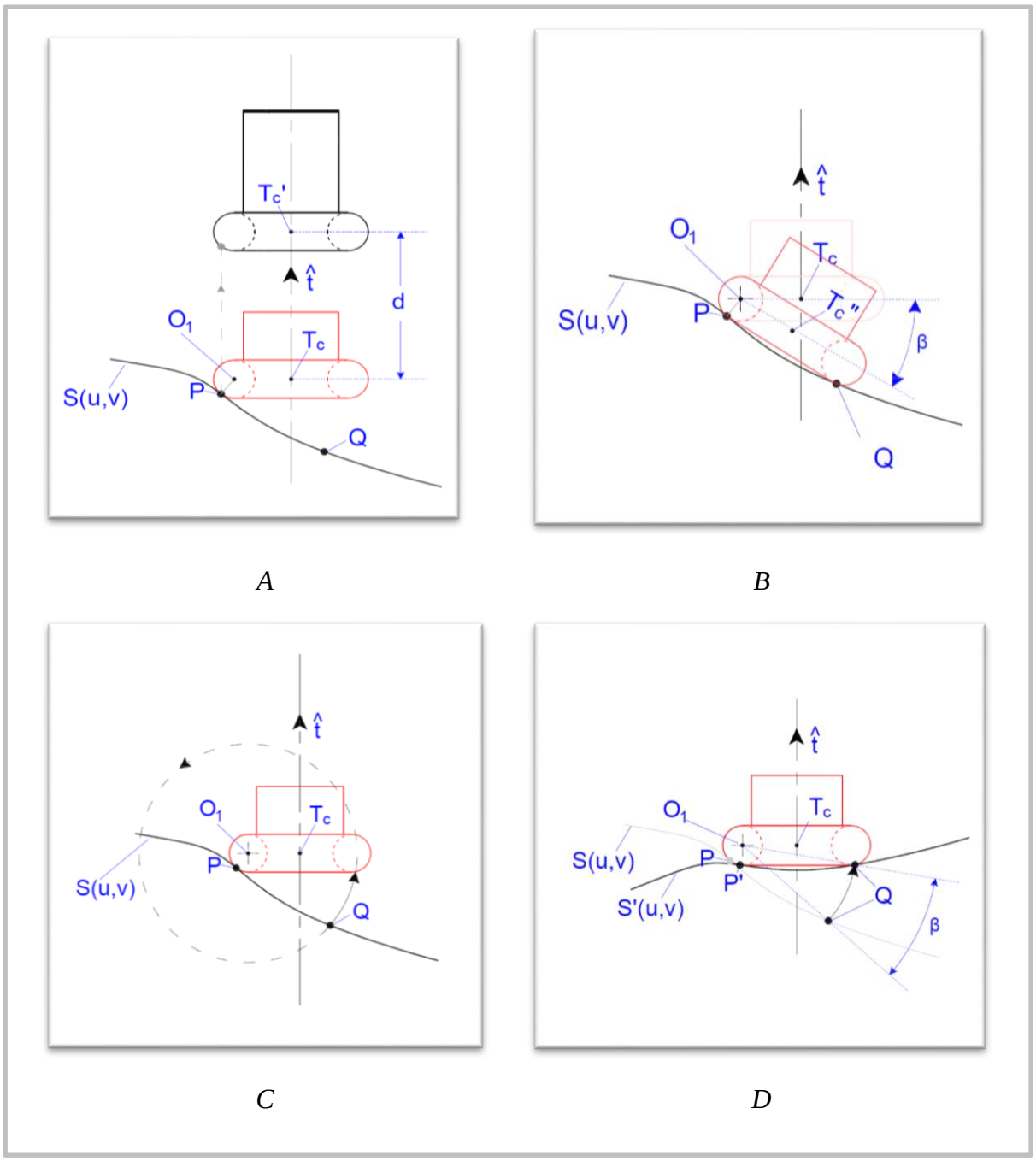


Figure 4.1: Different stages shown for getting the points of contact over the Bézier surface using the vertical and circular ray firing method in comparison with tilting the tool

Now for getting the second point of contact, Duvedi et al [15] gave a method of titling the tool about the axis of pseudo insert till the tool touches the surface at Q, as shown in *Figure 4.1(B)* which involves the solution of complex non-linear transcendental

equations and in DRD method the Bézier surface is rotated about the axis of the pseudo insert till the surface touches the toroidal cutter at Q, which is also an iterative process, as shown in *Figure 3.4*. Rotating the surface, till it the touches the toroidal cutter, also results in the similar second point of contact as obtained in tilting the tool.

Now if tool is kept stationary and a circular ray is fired from the so called second point of contact Q on the un-rotated Bézier surface with the axis of pseudo-insert as its center, the ray will intersect with the torus and form an angular arc segment, as shown in *Figure 4.1(C)*. The angular arc segment of the circular ray fired from the surface intersecting with the tool gives the tilt angle. The rotation of the surface with the tilt angle obtained from the angular arc segment emulates the calculation of the second point of contact Q, as shown in *Figure 4.1(D)*.

The process of firing the circular ray is given in section 4.3. With the circular ray firing method, the required tilt angle is obtained and hence gives the second point of contact Q. *Figure 4.1(D)* shows the arc formed by the circular ray and the associated tilt angle β, and also shows the Bézier surface $S'(u,v)$ touching the toroidal tool surface at two points of contact P' and Q without any gouging. As the surface is rotated with the tilt angle β about the axis of the pseudo-insert, the first point of contact slides from P to P' around the pseudo-insert.

Both the steps of vertical and circular ray firing involve the solution of implicit equations of toroidal cutter defined in Cartesian coordinate system and eliminates the use of iterative process involving complex non-linear transcendental equations. Hence making the method simpler, robust and fast. It is to be noted that, while the *Figure 4.1* makes it appear that O_1, P, Q, and the circular ray from Q lie in a common plane, that is not the case; instead, Q and the circular ray from Q lie in a plane parallel to the plane of the pseudo-insert centered at O_1.

4.2 Ray Firing in Tool Axis Direction for Drop

The algorithm in the proposed method is based on rays fired from the tensor product surface towards the direction of the tool and the ray that makes the intersection with the tool having the least distance traveled gives the first point of contact. The surface is

discretized finely and rays are fired in the tool axis direction towards the toroidal surface. In this work a Bézier surface $S(u, v)$ is given by equation (1) is used

The parameter u and v are used to finely discretize the surface. With the parameters defined $\vec{S}_{(u,v)}$ will give the position of a point on the surface in the Cartesian coordinate system as

$$\vec{S}_{(u,v)} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} \tag{19}$$

The toroidal surface is defined implicitly using the Cartesian coordinate system symmetric to z-axis and is given by the solution of $Torus(x, y, z) = 0$ at any position above the surface, where

$$Torus(x, y, z) = \left(\sqrt{{Tor_x}^2 + {Tor_y}^2} - R_o\right)^2 + (Tor_z + h)^2 - {R_i}^2 \tag{20}$$

(Tor_x, Tor_y, Tor_z) are the coordinates of any point $\overrightarrow{Tor}$ on the toroidal surface in (x, y, z) direction, respectively. $'h'$ is the height at which the torus is positioned from origin in z-direction over the Bézier surface as shown in *Figure 4.2*.

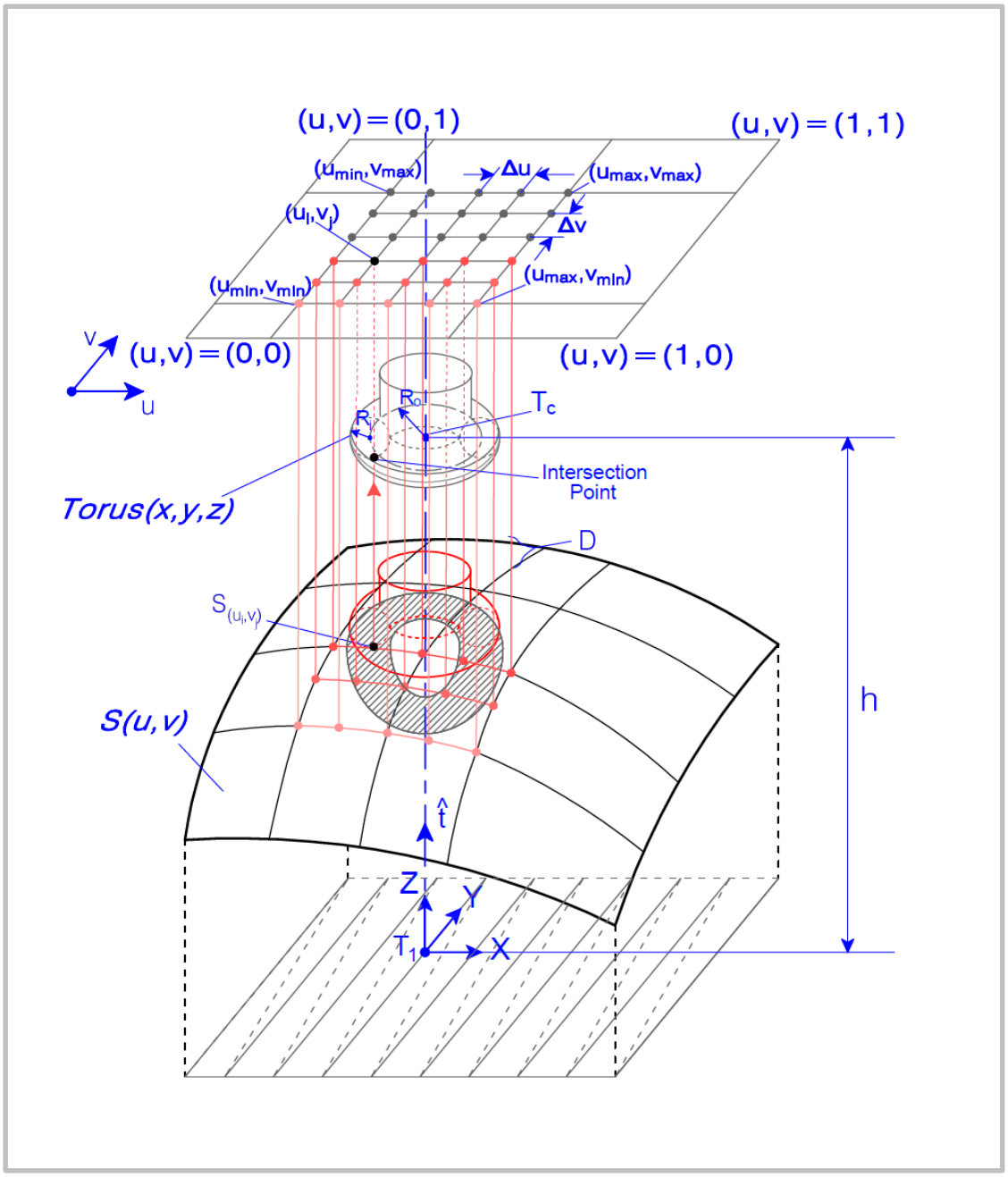


Figure 4.2: Firing the rays from the Bézier surface along tool axis $\hat{t}$. A patch formed under the shadow of the tool and drop distance set D obtained by intersection of fired rays from the Bézier surface

The tool could be positioned anywhere over the surface and hence firing the rays from some points $\vec{S}_{(u,v)}$ will not intersect with the tool surface. Rays yielding no intersection with the toroidal surface will give a complex solution and hence can be ignored. To limit the number of rays that do not intersect the tool, the surface patch under the shadow of the tool is found and then rays are cast from the discretized points of that particular patch instead of casting rays from the whole surface. Over the shadow of the tool a parametric grid with the limits $(u_{min}, u_{max})\ and\ (v_{min}, v_{max})$ is created which is then discretized with $\Delta u\ and\ \Delta v$, respectively, as shown in *Figure 4.2*.

The ray is cast from the Bézier surface in tool direction, $\hat{t} = \{0,0,1\}$ at each parameter (u_i, v_j) within the defined grid range, $(u_{min} \leq u_i \leq u_{max})$ and $(v_{min} \leq v_j \leq v_{max})$. The corresponding intersection point on the toroidal surface with the ray can be obtained by comparing $\vec{S}_{(u_i,v_j)}$ with $\overrightarrow{Tor}$. As the ray fired is in the tool direction and both the points, the point on surface and the intersection point, are collinear in the z-direction, as shown in *Figure 4.3*, so the comparison of $\vec{S}_{(u_i,v_j)}$ and $\overrightarrow{Tor}$ is given by

$$\vec{S}_{(u_i,v_j)} = \begin{bmatrix} S_{x1} \\ S_{y1} \\ S_{z1} \end{bmatrix} = \begin{bmatrix} Tor_x \\ Tor_y \\ Tor_z - d_{(u_i,v_j)} \end{bmatrix}. \tag{21}$$

Every point within the defined range of the grid on the surface will form a drop distance set $\{D\}$ with $d_{(u,v)}$ as drop distance, which is given by (from equation (23))

$$d_{(u_i,v_j)} = Tor_z - S_{z1} \tag{22}$$

where Tor_z can be obtained from equation (22) and equation (24) as

$$Tor_z = h - \sqrt{{R_i}^2 - \left(\sqrt{{S_{x1}}^2 + {S_{y1}}^2} - R_o\right)^2}. \tag{23}$$

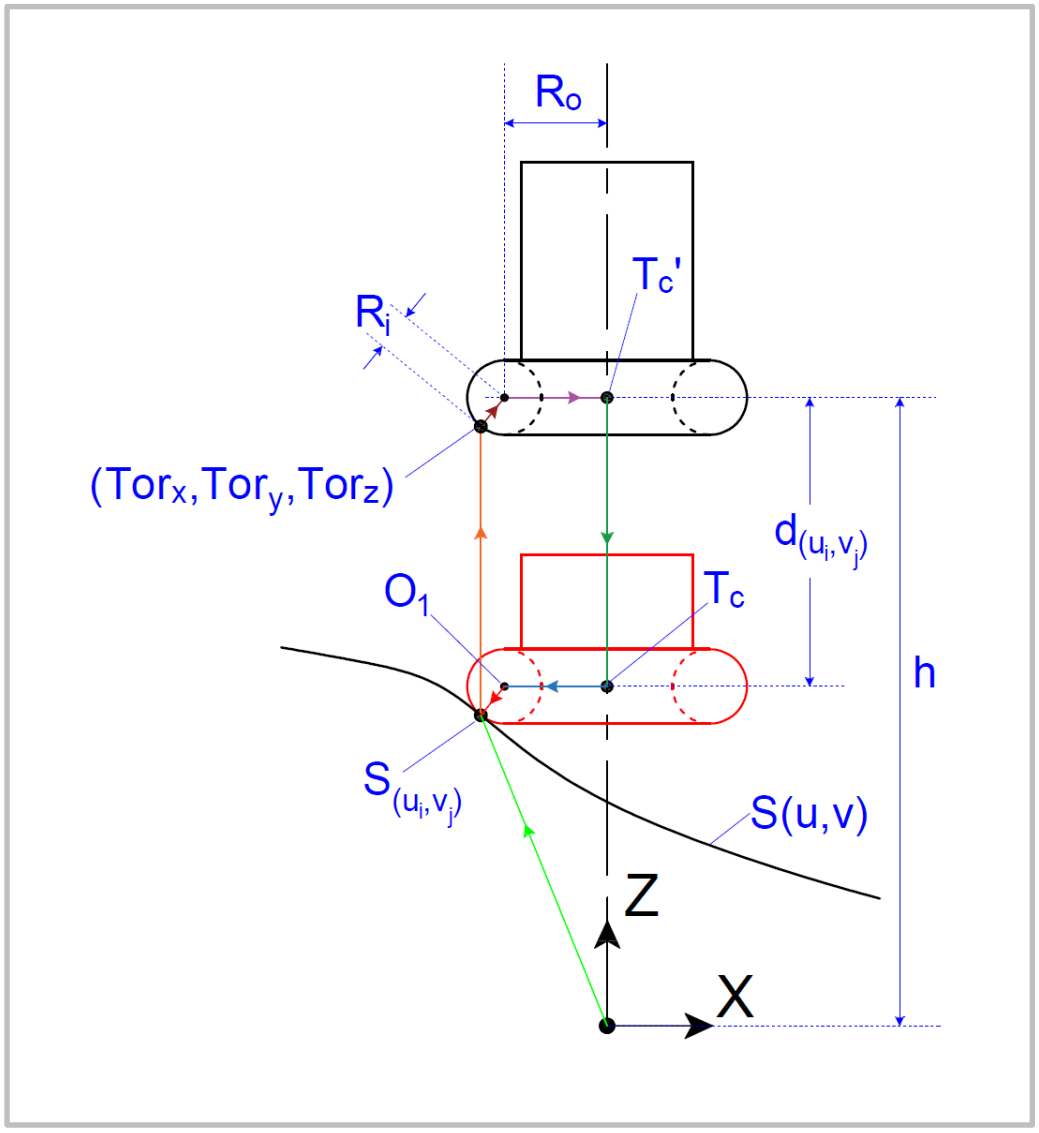


Figure 4.3: Ray fired from Bézier surface at $\vec{S}_{(u_i,v_j)}$ and the intersection point on torus (Tor_x, Tor_y, Tor_z) showing the collinearity in the XZ plane

The quadratic nature of the equation (25) clearly indicates that the rays will intersect the toroidal surface at two different positions, one with the upper surface of the torus and second with the lower surface of the torus. As the torus is symmetric about the z-axis and the xy plane, the lower surface of the torus will make the contact with Bézier surface. Hence the drop distance set obtained by the intersection points of fired rays with the lower surface of torus are stored to form the drop distance set $\{D\}$. The surface point corresponding to the minimum value of the drop distance d_{min} of drop distance

set $\{D\}$ yields the first point of contact, that the toroidal cutter will make with the Bézier surface.

4.3 Circular Ray Intersection for Tilt

The first point of contact $\vec{P}$ lies on one of the pseudo-insert, as shown in *Figure 4.4*. The insert in the physical tool is of the same size of the minor circle with radius R_i. Any rotation about the axis of this pseudo-insert maintains the contact between tool and surface at $\vec{P}$. To achieve the rotation about the axis of the pseudo-insert, a coordinate frame $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ is needed to define at the center of the pseudo-insert $\vec{O}_1$ and $\overrightarrow{Tc}$ as the tool center, where

$$\hat{w}_1 = \hat{t} \tag{24}$$

$$\hat{u}_1 = \frac{\hat{w}_1 \times \overrightarrow{(P - Tc)}}{\left| \hat{w}_1 \times \overrightarrow{(P - Tc)} \right|} \tag{25}$$

$$\hat{v}_1 = \hat{w}_1 \times \hat{u}_1 \tag{26}$$

The center of pseudo-insert $\vec{O}_1$ can be obtained by

$$\vec{O}_1 = \overrightarrow{Tc} - R_o \hat{v}_1 \tag{27}$$

The Bézier surface is redefined in the new coordinate frame $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ with the pseudo-insert center as its origin.

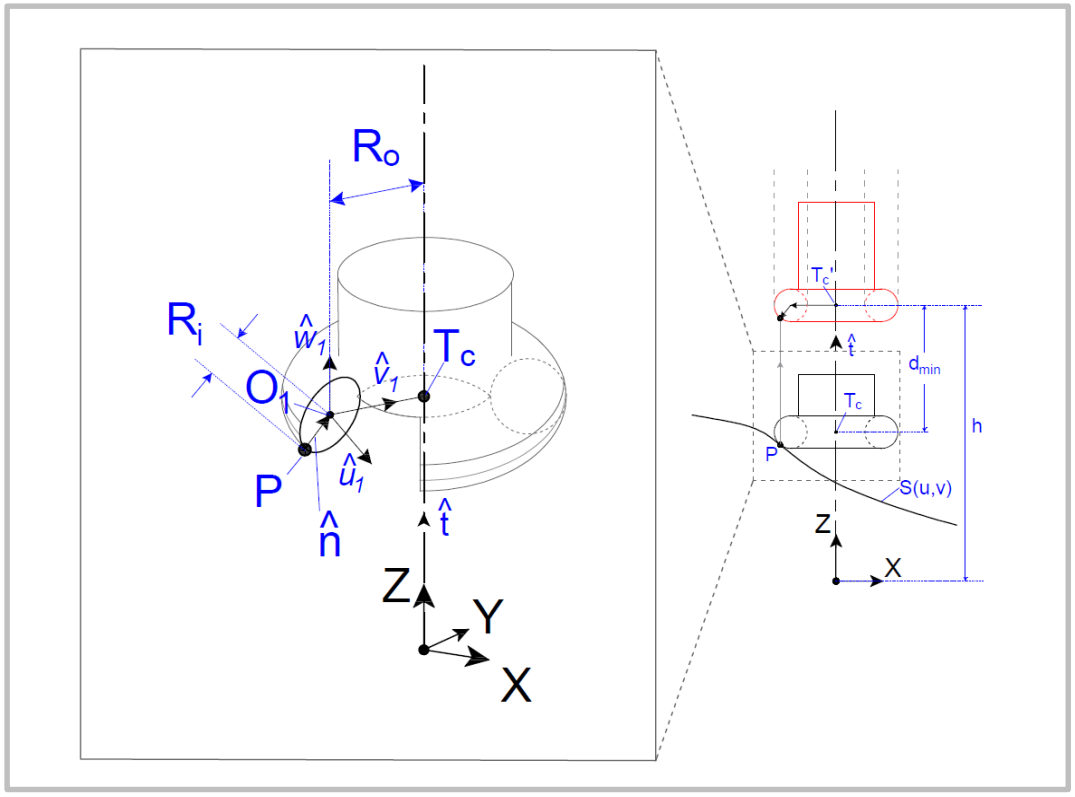


Figure 4.4: A new coordinate frame is generated at the center of the pseudo-insert O_1

To obtain the second point of the contact and tilt angle β, circular rays with the circle center on the axis of the pseudo-insert are fired from the surface defined in the new coordinate system and the intersection of each ray with the toroidal surface is found. The ray that intersects the torus with the least angle traveled gives the second point of contact and the corresponding angle of arc formed by the circular rays gives the tilt angle required to tilt the tool for machining the surface with tool touching the surface at multiple points without any gouging.

A plane is created at a distance of S_{u_1} in $\hat{u}_1$ direction from the origin. *Figure 4.5* depicts the casting of a circular ray from a point on surface from a plane parallel to the plane $\hat{v}_1\hat{w}_1$. The surface point $\vec{S}_{(u,v)}$ in $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ coordinate frame with its origin at O_1 is given by

$$\vec{S}_{(u,v)} = \begin{bmatrix} S_{u_1} \\ S_{v_1} \\ S_{w_1} \end{bmatrix}. \tag{28}$$

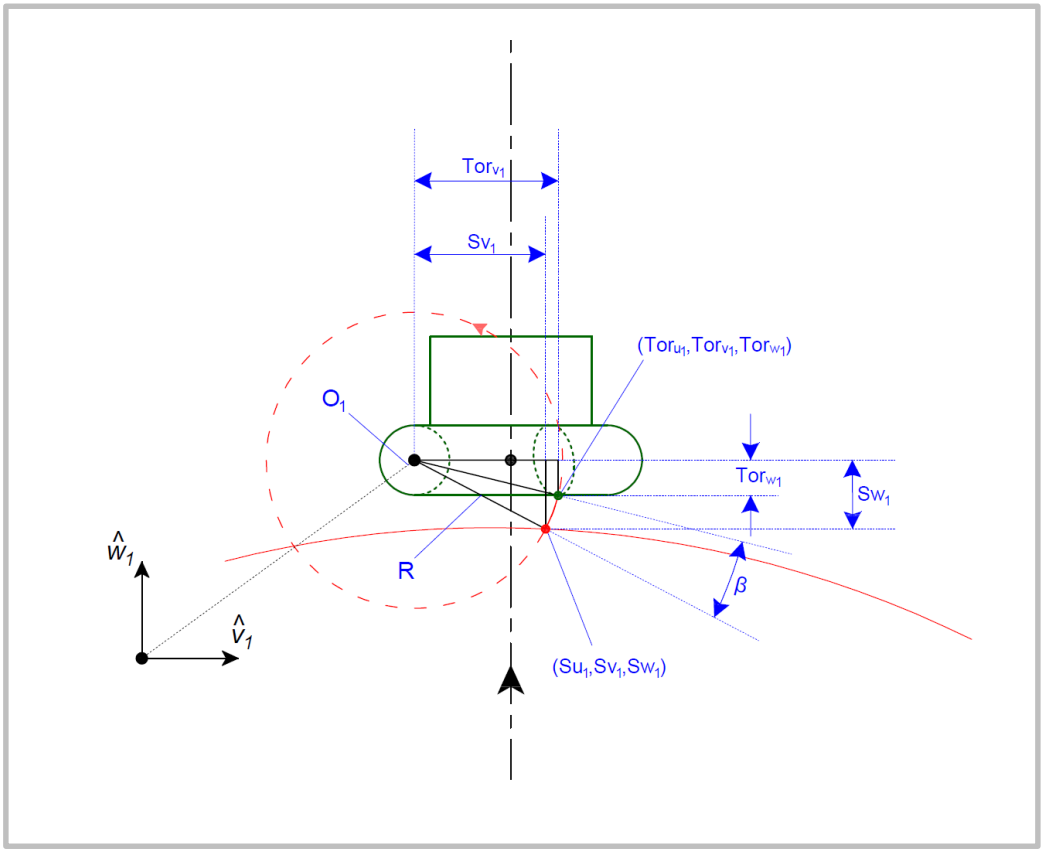


Figure 4.5: Planar view of the circular ray fired and giving the required tilt angle β

The torus is also redefined implicitly in the $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ coordinate frame with an offset of R_o in $\hat{v}_1$ direction and is given by

$$Torus(u_1, v_1, w_1) = \left(\sqrt{{Tor_{u_1}}^2 + (Tor_{v_1} - R_o)^2} - R_o\right)^2 + {Tor_{w_1}}^2 - {R_i}^2 \tag{29}$$

After finding the surface point, a plane parallel to the $\hat{v}_1\hat{w}_1$ plane is created at that point, which intersects the axis in the $\hat{u}_1$ direction at S_{u_1}. In that plane, as shown in *Figure 4.5*, a circular ray is created with radius R and its center lying on the axis, i.e., $(S_{u_1}, 0, 0)$, which intersects with both the surface and the torus. The equation of the circle at the torus intersection is given by

$$R^2 = {Tor_{v_1}}^2 + {Tor_{w_1}}^2. \quad (30)$$

The value of radius R can be computed from the equation of the circle at the point of intersection with the surface $\vec{S}_{(u,v)}$, i.e.,

$$R^2 = {S_{v_1}}^2 + {S_{w_1}}^2. \quad (31)$$

As the circular ray and both the point of intersection lie in a plane parallel to the vw plane at S_{u_1}, the coordinate of the torus at the point of intersection in the $\hat{u}_1$ direction is same as the S_{u_1}, i.e.,

$$Tor_{u_1} = S_{u_1}. \quad (32)$$

Now from Equations (29), (30) and (31), the values of unknown coordinates $[Tor_{v_1}, Tor_{w_1}]$ are found, which gives the intersection point on the toroidal surface with the circular ray. The fired circular ray forms an arc segment from intersection point on surface to the intersection point on the torus. The angle of the formed arc segment, as shown in *Figure 4.5*, is calculated and stored as

$$\beta = \tan^{-1}\left(\frac{Tor_{w_1}}{Tor_{v_1}}\right) - \tan^{-1}\left(\frac{S_{w_1}}{S_{v_1}}\right). \quad (33)$$

Figure 4.6 shows the three dimensional view of casting of circular ray from the surface point $(S_{u_1}, S_{v_1}, S_{w_1})$ that intersects the torus at $(Tor_{u_1}, Tor_{v_1}, Tor_{w_1})$. The plane v_1w_1 parallel to the plane $\hat{v}_1\hat{w}_1$ is at a distance of S_{u_1} from the center of the pseudo-insert.

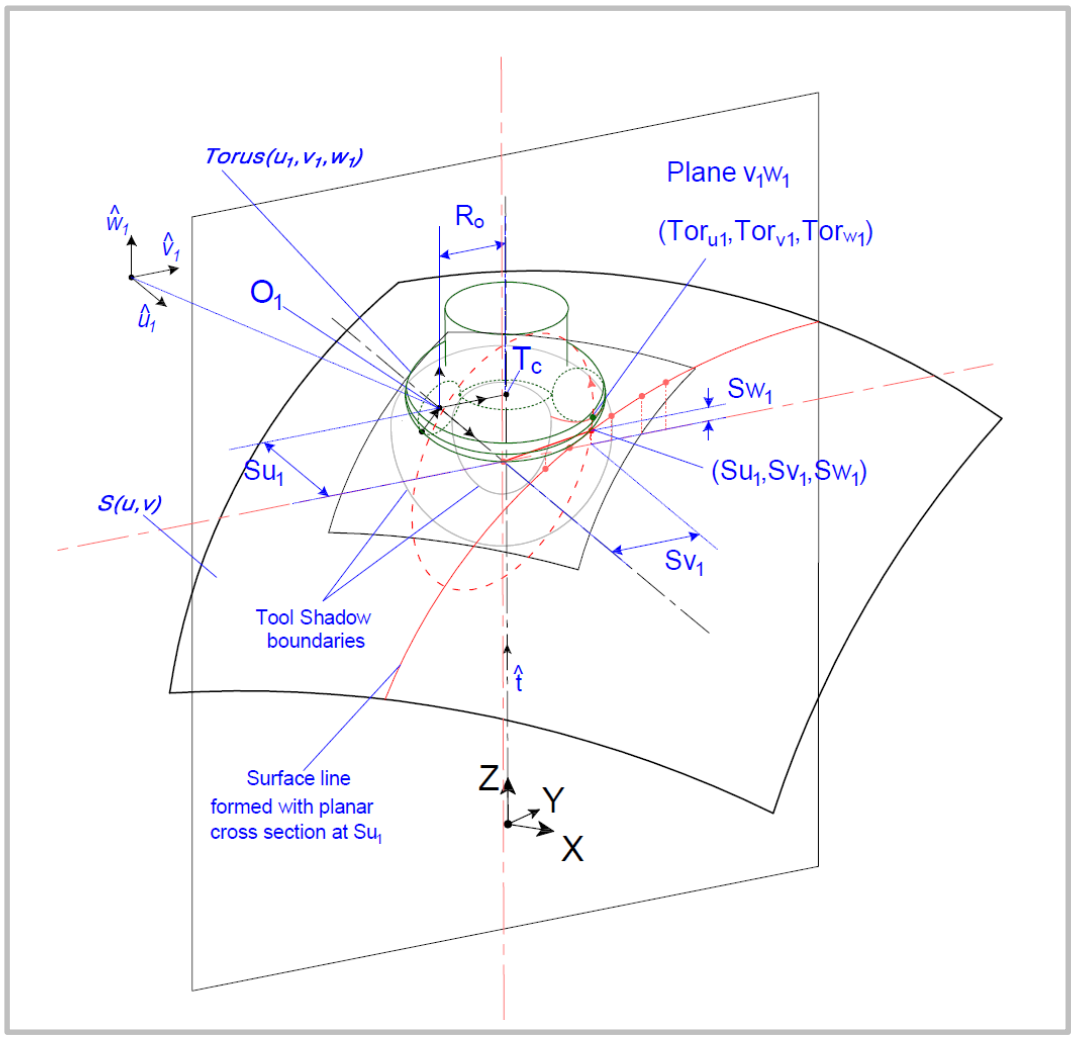


Figure 4.6: Circular ray fired from the surface points in the v_1w_1 plane positioned at S_{u_1}, intersecting the torus defined in $\{u_1, v_1, w_1\}$ coordinate frame

A grid of surface points is formed beneath the shadow of the tool, similar to the grid formed in the previous section with the limits (u_{min}, u_{max}) *and* (v_{min}, v_{max}) and discretized by Δu *and* Δv, respectively. Firing of circular rays from all the surface points on the defined grid forms a set of Angular arcs $\{B\}$. The minimum angle β_{min} of the set $\{B\}$ gives the required angle for tilting the tool for the machining to be gouge free with multi points of contact; and the corresponding values of parameter $[u, v]$ gives the location of the second point of contact. It is to be noted that the associated circular ray fired with the minimum tilt angle β_{min} intersects the torus perpendicularly with the tangent formed on the toroidal surface at the point of intersection.

4.4 Vertical and Circular Ray Firing (VCRF) Algorithm

The pseudo-code presented in *Figure 4.7* gives the algorithm for the proposed Vertical and Circular Ray Firing concept. It starts with (1.0) defining the surface from the given control points and generates the toolpath footprint (2.0), which, in this case, are parallel lines in the u-v plane. These parallel lines map to lines in the XY domain that extends between x_{min} to x_{max} and from y_{min} to y_{max}, as shown in *Figure 2.3*. The toolpath footprint is discretized with a spacing of $Side_{step}$ in the x direction and $forward_{step}$ in the y direction.

At each of the discretized points on the tool path footprint (4.0), the global coordinate system is set (4.1) and the toroidal tool is defined in that coordinate frame with the implicit equation given by equation (22) with the center of the tool $\vec{T}_c$, as shown in *Figure 4.2* at a specified height given by h in the equation. In this method, the tool is made stationary by setting the global coordinate frame at every footprint point. Therefore, the Bézier surface is made to move along with the changing position of the global coordinate frame with every footprint point; and this is achieved by translating the Bézier control points with the negative footprint point on the XY plane (4.2). For getting the minimum drop distance and the first point of contact, the rays are cast from the surface by dividing the parameters (u, v) used to define the surface, into the gird with user defined ∂u and ∂v. To reduce the number of unnecessary rays yielding no solution, only the surface parameters (u_{shadow}, v_{shadow}) under the shadow of the tool are used (4.3). After getting the surface parameters under the shadow, the vertical ray firing algorithm is applied (4.4) for the current Footprint position, which gives the surface parameters $[u_{drop}, v_{drop}]$ and the drop distance d_{min} for the first point of contact $\vec{P}$, which is then calculated (4.5) using surface parameters obtained from vertical firing of rays. The algorithm for vertical ray firing is explained later in Section 4.3.2.

1.0 Define array of Control Points of surface: $P[n][n]$
2.0 Generate Toolpath Footprint:

$FootPrint[k][2] = GenerateFootprint(side_{step}, forward_{step}, x_{max}, y_{max})$

3.0 Set

$Origin = [0,0,0], \hat{t} = [0,0,1], R_i = 6.0, R_o = 6.7, numberOfDivisions = 16$

4.0 $For\ i\ =\ 1;\ i\ \leq\ k; i++$

4.1 $T\ =\ [FootPrint[i][1],\quad FootPrint[i][2],\quad 0]$

4.2 Translate Bézier Surface with T to bring the Origin at Footprint point :
$P'[n][n] = translate(P[n][n], -T)$

4.3 Get any surface point under the shadow of the tool
$[u_{shadow}, v_{shadow}] = GetSurfacePointUnderToolShadow(P'[n][n], R_o, R_i)$

4.4 Get the minimum drop distance and associated surface parameters
$[d_{min}, u_{drop}, v_{drop}] = VerticalRayFiring(Torus(x, y, z), P'[n][n], u_{shadow}, v_{shadow}, R_o, R_i)$

4.5 Get the First point of contact and its normal on surface
$[P_i, \hat{n}_{P_i}] = S(u_{drop}, v_{drop}, P[n][n])$

4.6 Get the Tool Centre: $T_c = T + (h - d_{min}) * \hat{t}$

4.7 Set coordinate frame $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$:
$\hat{w}_1 = \hat{t}, \hat{u}_1 = \frac{\hat{w}_1 \times \overrightarrow{(P_i - T_c)}}{|\hat{w}_1 \times \overrightarrow{(P_i - T_c)}|}, \hat{v}_1 = \hat{w}_1 \times \hat{u}_1$

4.8 Get center of pseudo-insert
$O_1 = (-R_o * \hat{v}_1) + (T_c[2] * \hat{w}_1)$

4.9 Redefine Surface in$\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ setup at O_1

$P"[n][n]\ =\ RedefineSurface(\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}, O_1, P'[n][n])$

4.10 Get the minimum angle of arc and associated surface parameters
$[\beta_{min}, u_{tilt}, v_{tilt}] = CircularRayFiring(Torus(u_1, v_1, w_1), P"[n][n], u_{drop}, v_{drop}, R_o, R_i)$

4.11 Get the Second point of contact and its normal on surface
$[Q_i, \hat{n}_{Q_i}] = S(u_{tilt}, v_{tilt}, P[n][n])$

4.12 Rotate tool axis around global x axis at $Origin$ with angle of rotation β_{min}:
$\hat{t}' = rotate(\hat{t}, \beta_{min}, x, Origin)$

4.13 Print Tool position data:
$print(P_i, \hat{n}_{P_i}, Q_i, \hat{n}_{Q_i}, \hat{t}')$

Figure 4.7: Pseudo-Code for the VCRF Algorithm for computing gouge free tool position over the Bézier surface

After obtaining the first point of contact $\vec{P}$, the tool center T_c is obtained using the difference of height h and the minimum drop distance d_{min} (4.6) obtained from vertical ray firing. Positioning the toroidal cutter with its tool center at T_c, the tool touches the Bézier surface at a single point without any gouging. The new coordinate frame $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ is setup with its origin at the center $\vec{O}_1$ of the pseudo-insert (4.7 and 4.8). The translated Bézier surface and the toroidal tool surface are redefined in the new

coordinate frame (4.9). The definition of the torus in the new coordinate frame is given by equation (30), in which the torus is offset in $\hat{v}_1$with a distance of outer radius R_o of the torus. After redefining the surface and torus in the $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ coordinate frame, the circular ray firing algorithm is applied (4.10) for the current footprint position with the seed of surface parameters obtained in vertical ray firing $[u_{drop}, v_{drop}]$ for initialization of the parametric grid. The circular ray firing module computes the surface parameters $[u_{tilit}, v_{tilt}]$ and minimum tilt angle β_{min} for the second point of contact $\vec{Q}$, which is then calculated (4.11) using surface parameters obtained from circular firing of rays. The algorithm for circular ray firing is explained later in section 4.3.3. At this stage the two point of contacts $\vec{P}$ and $\vec{Q}$ needed to define the tool position for five axis machining are obtained.

4.4.1 Surface Parameters under Tool Shadow

The pseudo-code presented in *Figure 4.8* gives the algorithm for finding the tool shadow. For getting the surface parameters (u_{shadow}, v_{shadow}) under the shadow of the tool for initialization, the surface is scanned by firing the rays from the surface to obtain any intersection with the toroidal surface. For firing the rays from the surface without leaving any patch on the surface un-scanned, a grid of uv parameters is formed with minimum and maximum values for both parameters as 0 and 1, respectively; and the grid is divided into a parametric mesh with difference $\partial u \; and \; \partial v$, defined by the user (1.0).

The surface point for every parametric node is calculated and the magnitude of the point on surface, which ranges from $0 \; to \; \infty$, in the XY plane is obtained. As the global coordinate system is set at the tool position in the XY plane and the tool is made stationary, only the value of magnitude of the $x \; and \; y$ coordinates of the surface points decides whether or not the surface point lies under the shadow of the tool. If the minimum value of magnitude is less than $((R_o + R_i) * 0.5)$ then that point lies under the shadow of the tool (2.1.2) and corresponding parametric values (u, v) are stored as well otherwise the number of divisions are doubled which reduces the size of intervals

$\partial u \; and \; \partial v$ and makes the mesh denser; the surface is rescanned with the new grid size values until the surface point lying under the shadow of the tool is obtained.

$GetSurfacePointUnderToolShadow(P'[n][n], R_o, R_i)$

1.0 Set Parameter limits:

$u_{min} = 0, u_{max} = 1, \partial u = (u_{max} - u_{min})/numberOfDivisions$

$v_{min} = 0, v_{max} = 1, \partial v = (v_{max} - v_{min})/numberOfDivisions$

2.0 $while(!\, RayIntersectWithTorus)$

Initialize min_mag_{xy} = 1e24

2.1 $For\ u = u_{min}, u \le u_{max},\ u \mathrel{+}= \partial u$

2.1.1 $For\ v = v_{min}, v \le v_{max}, v \mathrel{+}= \partial v$

Get surface point at $[u, v]$:

$[Sx, Sy, Sz] = S(u, v, P'[n][n])$

Get the Magnitude of surface point in XY plane

$mag_{xy} = Sx^2 + Sy^2$

Store the $[u, v]$ parameters and mag_{xy} of surface point closest to origin in XY plane

$if\ \left(mag_{xy} < \text{min\_}mag_{xy}\right)\ then$

$\left[\text{min\_}mag_{xy}, u_{shadow}, v_{shadow}\right] = \{mag_{xy}, u, v\}$

2.1.2 $if\ \left(\text{min\_}mag_{xy} < ((R_o + R_i) * 0.5)\right)\ then$

$RayIntersectWithTorus$

$Else$

$numberOfDivisions \mathrel{*}= 2,$

$Reset(\partial u, \partial v)$

3.0 Return $[u_{shadow}, v_{shadow}]$

Figure 4.8: Pseudo-code for the getting the surface parameters under the shadow of the tool Algorithm

4.4.2 Vertical Ray Firing Algorithm

The pseudo-code presented in *Figure 4.9* gives the vertical ray firing algorithm for finding minimum drop distance d_{min} and associated surface parameters giving the first point of contact. After computing the surface parameters under the shadow of the tool, another parametric grid is formed around the shadow of the tool, which is slightly bigger than the shadow of the tool (1.0). Now the surface point for every u, v node of that grid is calculated. From equation (23) it is known that the x and y direction coordinates in the XY plane for both the surface point and intersection point on torus are the same.

Hence the z coordinate for the intersecting point on the torus can be obtained using equation (22) and is stored as Tor_z. Then the drop distance is given by the difference of the z coordinates of a surface point and the associated intersecting point on the toroidal surface, for each uv node. The minimum drop distance d_{min} is stored and the corresponding surface parameters are also stored. Then the grid parameters are again reset according to the newly obtained u, v parameters corresponding to the minimum drop distance, in such a way that the newly formed grid creates a minuscule patch around the current iteration u and v parameters with a tolerance of $\pm\epsilon$ in both parametric directions. This process is repeated four times for the better accuracy of the results and minuscule patch formation around the u, v parameters in every iteration helps the algorithm to converge fast.

$VerticalRayFiring(Torus(x, y, z), P'[n][n], u_{shadow}, v_{shadow}, R_o, R_i)$

1.0 Set the Grid parameters over the shadow of tool i.e. $[u_{shadow}, v_{shadow}]$
$SetupGridParameters(u_{min}, u_{max}, \partial u, v_{min}, v_{max}, \partial v), numberOfDivisions = 32$
2.0 Initialize minimum drop distance: $d_{min} = 1e24$
3.0 $For\ iteration = 0, iteration < 4, iteration++$
3.1 $For\ u = u_{min}, u \leq u_{max},\ u+= \partial u$
3.1.1 $For\ v = v_{min}, v \leq v_{max}, v+= \partial v$
Get surface point at $[u, v]$:
$[Sx, Sy, Sz] = S(u, v, P'[n][n])$
Get the Tor_z for corresponding surface point in XY plane
$Tor_z = getTorusZ(Sx, Sy)$
Calculate drop distance $d_{u,v}$ corresponding surface point in XY plane
$d_{u,v} = Tor_z - Sz$
$if\ (d_{u,v} < d_{min})\ then$
$[d_{min}, u_{drop}, v_{drop}] = \{d_{u,v}, u, v\}$
3.1.2 $Reset(u_{min}, u_{max}, \partial u, v_{min}, v_{max}, \partial v, \epsilon)$
4.0 Return $[d_{min}, u_{drop}, v_{drop}]$

Figure 4.9: Pseudo-code for vertical firing of rays in global coordinate frame algorithm

4.4.3 Circular Ray Firing Algorithm

The pseudo-code presented in *Figure 4.10* gives the circular ray firing algorithm for finding the minimum tilt angle β_{min} and associated surface parameters giving the second point of contact. The surface parameters of the first point of contact $\vec{P}$ which lies under the shadow of the tool, are fed to this algorithm as the seed to initialize and setting up the grid parameters (1.0). For each u, v node the corresponding surface point, defined in the $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ coordinate frame, is obtained. As the circular ray is fired in the plane parallel to the $\hat{v}_1\hat{w}_1$ plane created at a distance of S_{u_1} with its center on the axis passing through the pseudo-insert in the $\hat{u}_1$ direction, so the $\hat{u}_1$ coordinate of the torus defined in the same frame, i.e., Tor_{u_1} is similar to S_{u_1} and the radius of the circle R only depends upon the coordinates in the $\hat{v}_1, \hat{w}_1$ directions and is given by equations (32 and 33). Now Tor_{v_1} and Tor_{w_1} are calculated by solving equations (30, 32 and 33) to get the position of the point at which the circular ray intersects with the toroidal surface starting from (S_{v_1}, S_{w_1}) at a plane at S_{u_1}. After getting both points, the surface point and the intersection point on torus, the difference of angles made in the Euclidean plane between the $\hat{v}_1$ axis and the points $\left(Tor_{v_1}, Tor_{w_1}\right)$ and (S_{v_1}, S_{w_1}), as shown in *Figure 4.6*, and named as β. The surface point parameters giving the minimum angle is stored as $[u_{tilt}, v_{tilti}]$ and the associated angle is stored as β_{min}, which gives the minimum possible rotation required to tilt the tool around the pseudo-insert axis touching the surface at second point of contact $\vec{Q}$ along with $\vec{P}$ without gouging the surface.

$CircularRayFiring(Torus(u_1, v_1, w_1), P"[n][n], u_{drop}, v_{drop}, R_o, R_i)$

1.0 Set the Grid parameters with the seed from drop i.e. $[u_{drop}, v_{drop}]$
$SetupGridParameters(u_{min}, u_{max}, \partial u, v_{min}, v_{max}, \partial v), numberOfDivisions = 32$
2.0 Initialize angle of arc: $\beta_{min} = 1e24$
3.0 $For\ iteration = 0, iteration < 4, iteration++$
3.1 $For\ u = u_{min}, u \leq u_{max},\ u += \partial u$
3.1.1 $For\ v = v_{min}, v \leq v_{max}, v += \partial v$
Get surface point at $[u, v]$:

$$[S_{u_1}, S_{v_1}, S_{w_1}] = S(u, v, P"[n][n])$$

Calculate radius of circular arc in $\hat{v}_1\hat{w}_1$ plane

$$R = {S_{v_1}}^2 + {S_{w_1}}^2$$

Get the unknowns i.e. $[Tor_{w_1}, Tor_{v_1}]$ for corresponding surface point in $\hat{v}_1\hat{w}_1$ plane

$$[Tor_{w_1}, Tor_{v_1}] = solve(Torus(u_1, v_1, w_1), R)$$

Calculate angle of arc β corresponding surface point in $\hat{v}_1\hat{w}_1$ plane

$$\beta = atan2(Tor_{w_1}, Tor_{v_1}) - atan2(S_{w_1}, S_{v_1})$$

$if\ (\beta < \beta_{min})\ then$

$$[\beta_{min}, u_{tilt}, v_{tilt}] = \{\beta, u, v\}$$

3.2 $Reset(u_{min}, u_{max}, \partial u, v_{min}, v_{max}, \partial v)$
4.0 Return $[\beta_{min}, u_{tilt}, v_{tilt}]$

Figure 4.10: *Pseudo-code for Circular Firing of rays in $\{\hat{u}_1, \hat{v}_1, \hat{w}_1\}$ coordinate frame Algorithm*

4.4.4 Setting Up the Grid Parameters

The purpose of setting the grid parameters is to reduce unnecessary computation by firing rays from a small patch over the surface that encloses the shadow of the tool instead of firing the rays from the whole surface, as shown in *Figure 4.2*. The minimum and maximum values of both the parameters need to be set according to the surface parameters obtained under the shadow of the tool and the tool parameters such as R_o, R_i. To ensure the grid encloses the whole tool, a square grid of three times the size of radius of the outside circle of the torus is used. The parameters of the grid are given by

For the parameter in the u direction

$$u_{min} = u_{shadow} - \ Tool\ Parameter\ Factor$$

$$u_{max} = u_{shadow} + \ Tool\ Parameter\ Factor$$

$$\partial u = \frac{u_{max} - u_{min}}{number\ of\ divisions}$$

For the parameter in the v direction

$$v_{min} = v_{shadow} - \ Tool\ Parameter\ Factor$$

$$v_{max} = v_{shadow} + \ Tool\ Parameter\ Factor$$

$$\partial v = \frac{v_{max} - v_{min}}{number\ of\ divisions}$$

where, u_{shadow}, v_{shadow} are the surface parameter of any point obtained under the shadow of the tool and $'Tool\ Parameter\ Factor'$ is given by

$$Tool\ Parameter\ Factor = \left(\frac{R_o + R_i}{Max(X\ or\ Y)}\right) \times 1.5$$

where $R_o\ and\ R_i$ are the major and minor radii of the toroidal cutter and $Max(X\ or\ Y)$ is the maximum value of the control points defining the surface either in the x direction – if u direction parameters are set or in the y direction – if v direction parameters are set. $'number\ of\ divisions'$ is user defined and can be set according to the accuracy required by the user. Increasing the $number\ of\ Divisions$ makes the grid size finer and increases the number of uv nodes in the grid, hence leads to higher computation time.

4.4.5 Resetting the Grid through Iterations

Going through the iterations in the vertical ray firing and the circular ray firing algorithms, the grid parameters are reset at the end of every iteration according to the surface parameters u, v associated with the minimum drop distance or tilt angle for the current iteration. In this, instead of covering the whole shadow of the tool, a small grid is formed around the surface parameters u, v with a factor of ϵ, which is user defined. The parameters of grid are given by

For the parameter in the u direction

$$u_{min} = u - \epsilon$$

$$u_{max} = u + \epsilon$$

$$\partial u = \frac{u_{max} - u_{min}}{number\ of\ divisions}$$

For the parameter in the v direction

$$v_{min} = v - \epsilon$$

$$v_{max} = v + \epsilon$$

$$\partial v = \frac{v_{max} - v_{min}}{number\ of\ divisions}$$

CHAPTER 5 Results and Discussion

5.1 Implementation

The given algorithms were implemented using C++. The methods developed in this work uses the surface points for the calculation of tool position and are not dependent on the type of the surface. So, the Bézier surfaces are used for testing the algorithms. Moreover, B-Spline surfaces are considered as generalization of Bézier surfaces and share a lot of similarities, as each piecewise polynomials of B-Spline surfaces, defined by the knot vectors, can be considered as a Bézier surface. Hence, the methods developed in this work can be further implemented onto B-Spline surfaces giving the desired results.

The algorithms were tested on three Bi-cubic Bézier surfaces that included a convex surface, a concave surface and a saddle surface having both concave and convex regions. All three surfaces have a span of $150 \times 150\ mm$ in the XY plane. The x and y coordinates of the control points are uniformly distributed with a span interval of 50 starting from the origin. The z coordinates for the three bi-cubic Bézier surfaces are given in *Table 1*. The three test surfaces are shown in *Figure 5.1*. The surfaces were machined using an XY parallel toolpath as shown in *Figure 2.3* having 10 passes in total with the first 9 passes separated by $Side_{step} = 18.0\ mm$ and 10th pass at $x = 150\ mm$. Each pass contains 78 tool positions separated by $Forward_{step} = 2.0\ mm$, out of which the first and last tool position of the pass is used to lift the tool to avoid gouging on the surface as the tool shifts from one pass to the next. The algorithm runs for 760 effective tool positions that physically making contact with the surface to be machined. The toroidal cutter used for machining had a major radius $R_o = 6.7\ mm$ and minor radius $R_i = 6.0\ mm$.

Table 1: Z Coordinates (in mm) for the three Bi-cubic Bézier Surfaces used for testing the Algorithms

Surfaces	**P_{00}**	**P_{01}**	**P_{02}**	**P_{03}**	**P_{10}**	**P_{11}**	**P_{12}**	**P_{13}**	**P_{20}**	**P_{21}**	**P_{22}**	**P_{23}**	**P_{30}**	**P_{31}**	**P_{32}**	**P_{33}**
Convex	80	90	90	80	90	105	105	90	90	105	105	90	80	90	90	80
Concave	80	70	70	80	70	55	55	70	70	55	55	70	80	70	70	80
Saddle	80	65	90	85	80	85	90	95	100	105	95	100	90	100	100	85

The convex and concave surfaces from *Figure 5.1* were used to check the accuracy of both algorithms and the saddle surface (that includes both the convex and concave regions) was used to check the behavior of the algorithms as the tool transits from concave to convex and vice versa, and also gives the robustness of the algorithms.

Toolpath data for all three surfaces were generated using both the algorithms purposed in this work for every tool position of all the passes of the toolpath footprint. The generated toolpath data was verified by first simulating the toolpath data using ToolSim, and then physically machining the three surface on aluminum stock using the same tool path with a DMU-80P Hi-Dyn tilt-rotary simultaneous five axis machining center.

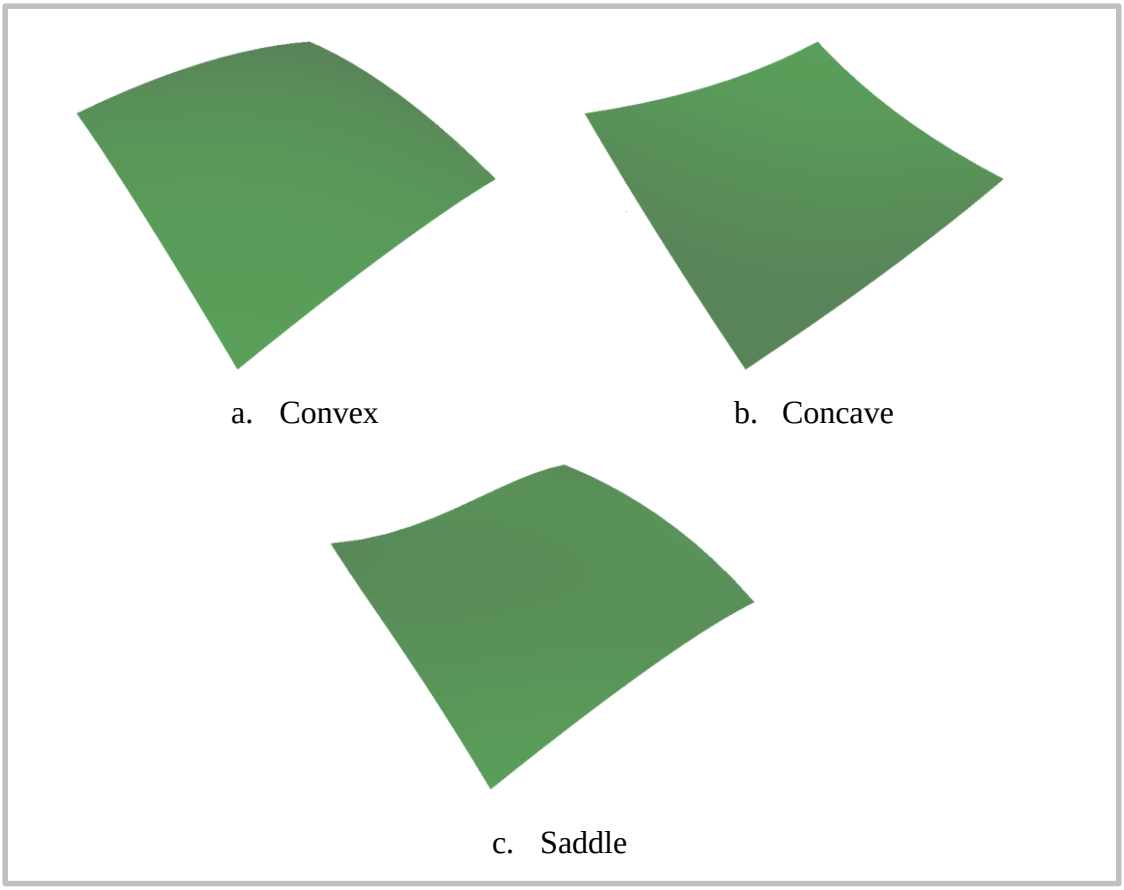


Figure 5.1: Cubic Surface used to test the Algorithms

5.2 DRD Method Results

The results of physical machining and simulator emulates the anticipated part surfaces as shown in *Figure 5.2*, which gives the side-by-side representations of the simulated part surface and machined part surface. The scallops on

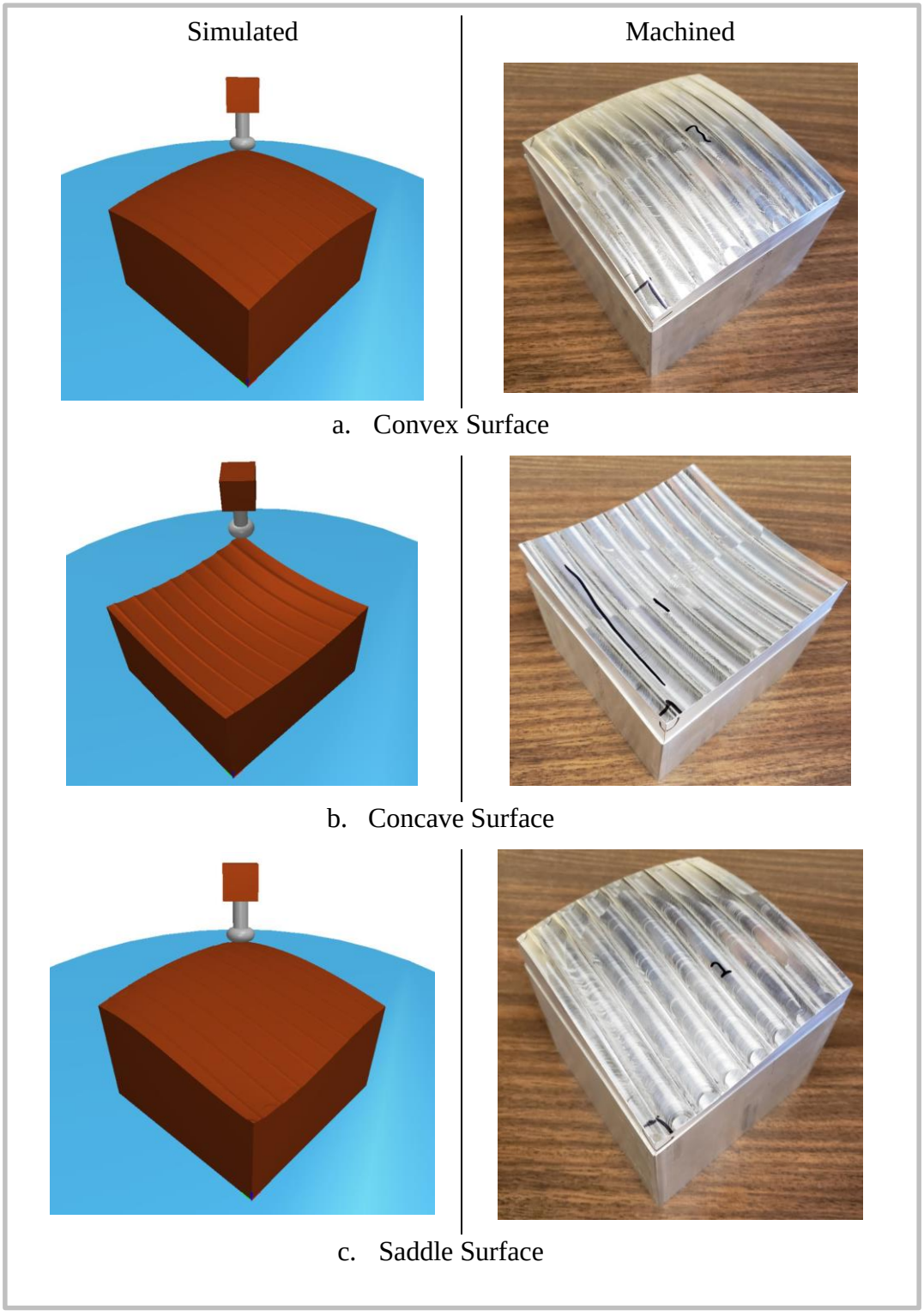
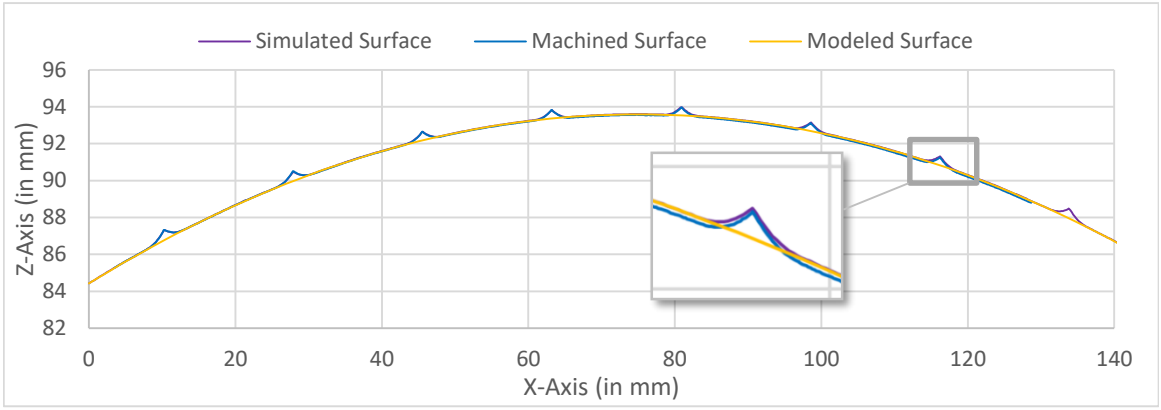


Figure 5.2: Three surfaces tested and verified by simulator under the section simulated surface; and by machining with DMU-80P Hi-Dyn tilt-rotary simultaneous five axis machining center under the section machined surfaces for DRD algorithm

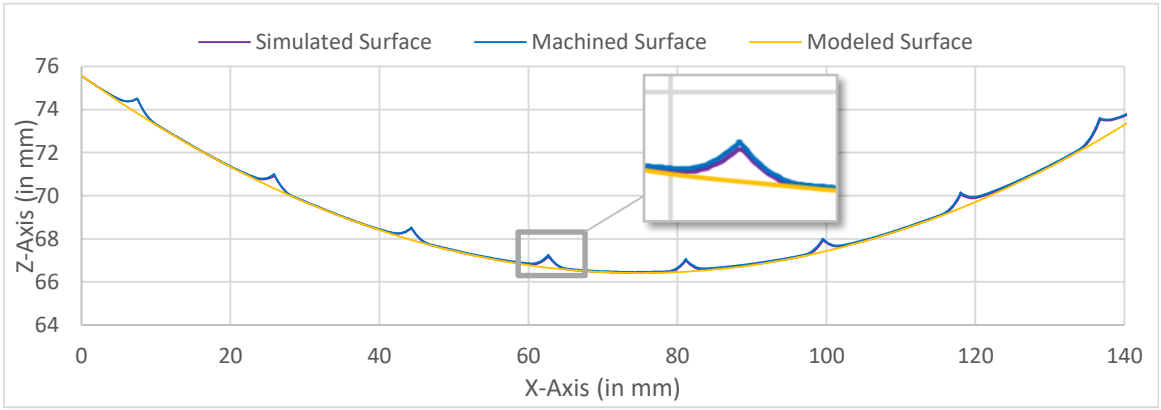
both the machined and simulated surfaces can be seen as the effect of larger side step taken for the machining.

After machining, the three surfaces were geometrically measured using a coordinate measuring machine (CMM) at a cross section taken in the side-step direction at Y = 27.00 mm in the XY plane with a probe of diameter 4 mm. Another set of geometrical data points was taken from the machined stock produced in ToolSim. ToolSim has an option to save the machine stock as an OBJ file, which stores the triangulated mesh of the stock surface [18] generated after the simulation of the toolpath in the form of vertices, their normal and connectivity information. The vertices for the cross section at Y= 27.00 mm in the XY plane were found and stored separately for the comparison. The obtained geometrical data from the machined geometry and the simulated surface were compared with the cross section taken from the modeled surface. *Figure 5.3* shows the graphical comparison of all the three surfaces, convex, concave and saddle, for the three sets of geometrical data obtained from simulated surface, machined surface and modeled surface.

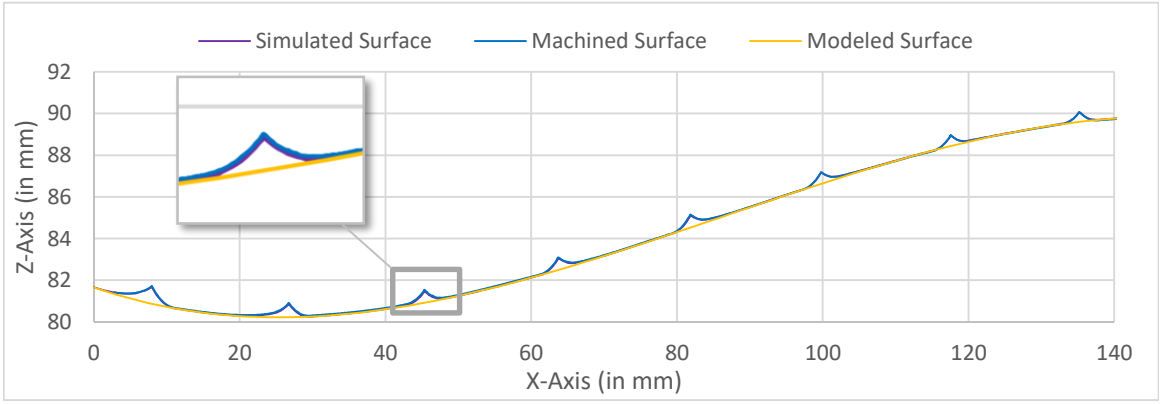
It can be seen from the graphical representation of the surfaces in *Figure 5.3* that the surface data obtained from machined and simulated geometries for the three parts is similar and overlaps with each other, whereas deviation of the machined and simulated surfaces from the modeled surface in the form of scallops can be seen. The larger value of side-step taken is the reason behind the formation of these scallops, and that can be controlled by reducing the side-step value as per required surface finish.



(A)



(B)



(C)

Figure 5.3: Graph Showing the comparison of machined and simulated data with the modeled Surface at a cross-section taken in XY plane at Y=27.0 mm for (A) convex surface, (B) concave surface and (C) saddle surface for toolpath generated using DRD algorithm

Table 2: Minimum and maximum deviation (in mm) of the machined and simulated geometries from the modeled geometry for DRD algorithm

Surface	Machined		Simulated	
	Min	**Max**	**Min**	**Max**
Convex	8.8E-05	0.57	2.9E-05	0.54
Concave	1.8E-02	1.00	1.4E-03	0.92
Saddle	7.6E-05	0.86	2.4E-05	0.79

Table 2 gives the minimum and maximum values of deviation on the machined and simulated geometries from the modeled geometry for all the three test surfaces. Maximum deviation gives the value of maximum scallop height which is higher in case of the concave surface as compared to the convex and saddle surfaces. Whereas, there is not any significant difference that can be found on comparison of the machined surface with the simulated surface, depicting the accuracy of the simulator in predicting the machining surface. This can also be verified from the graphical representation in *Figure 5.3*, as the machined and simulated surface shown overlaps.

5.3 VCRF Method Results

The physical and simulated results of machining emulates the anticipated part surfaces as shown in *Figure 5.4*, given by side-by-side representation of the simulated part surface and machined part surface. The scallops can also be seen on both the machined and simulated surfaces due to the effect of the larger side step.

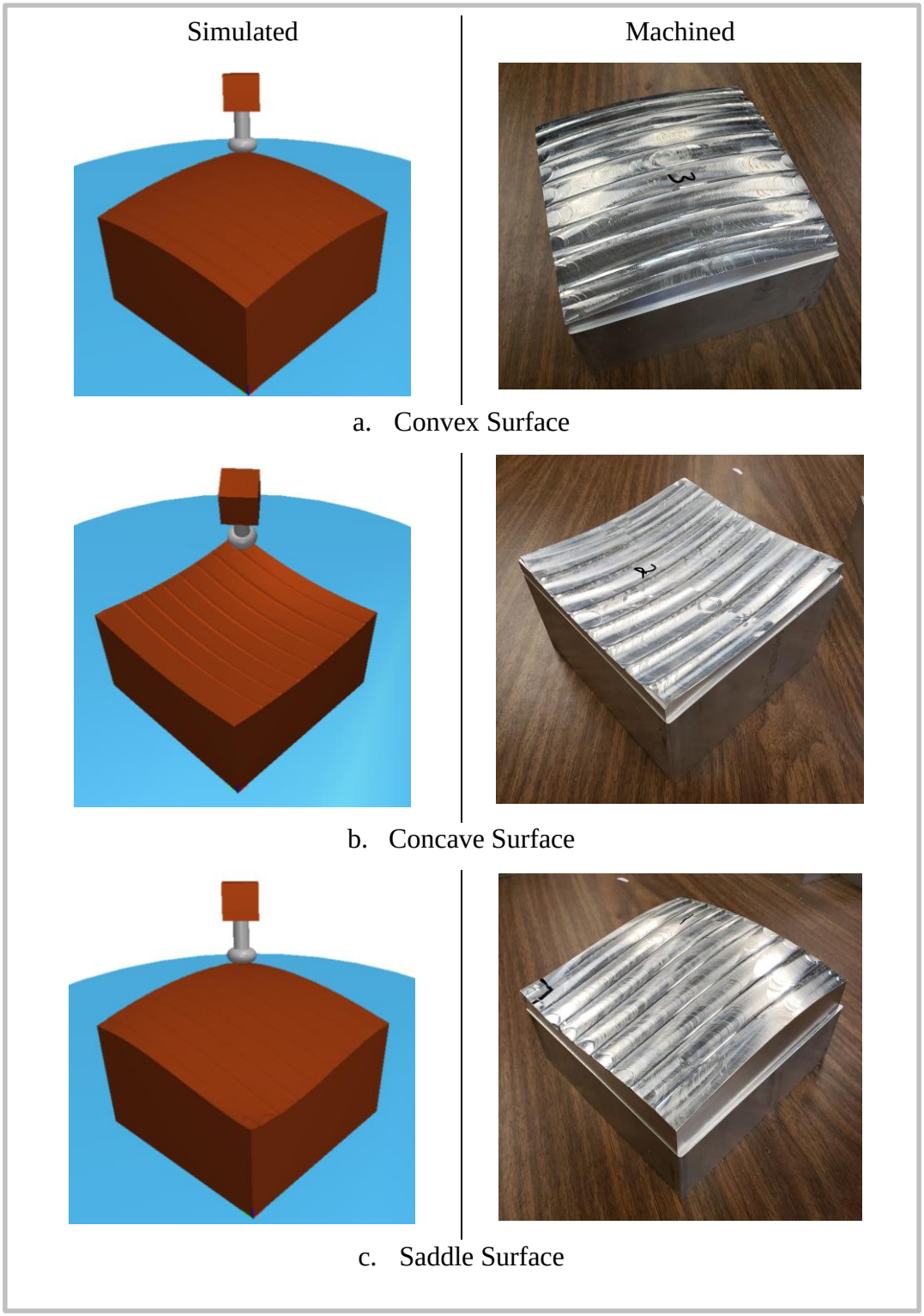
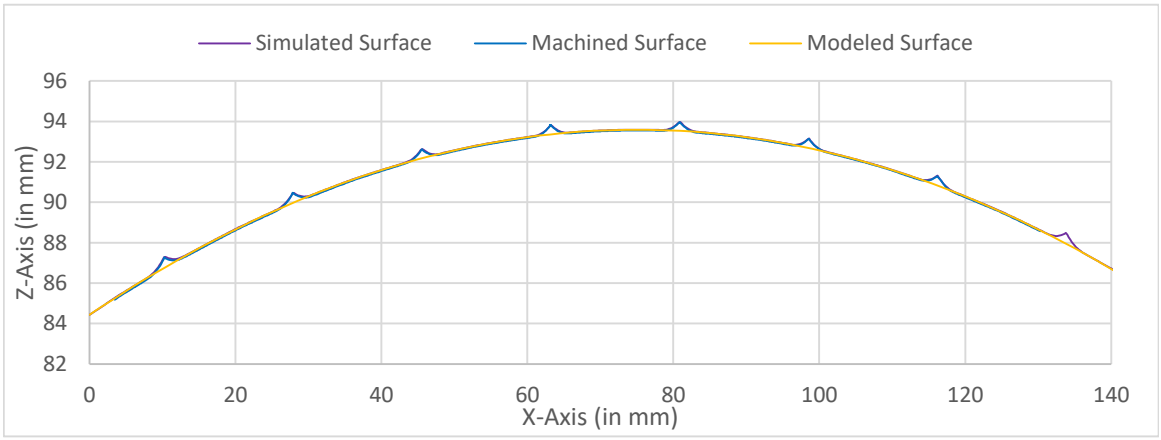


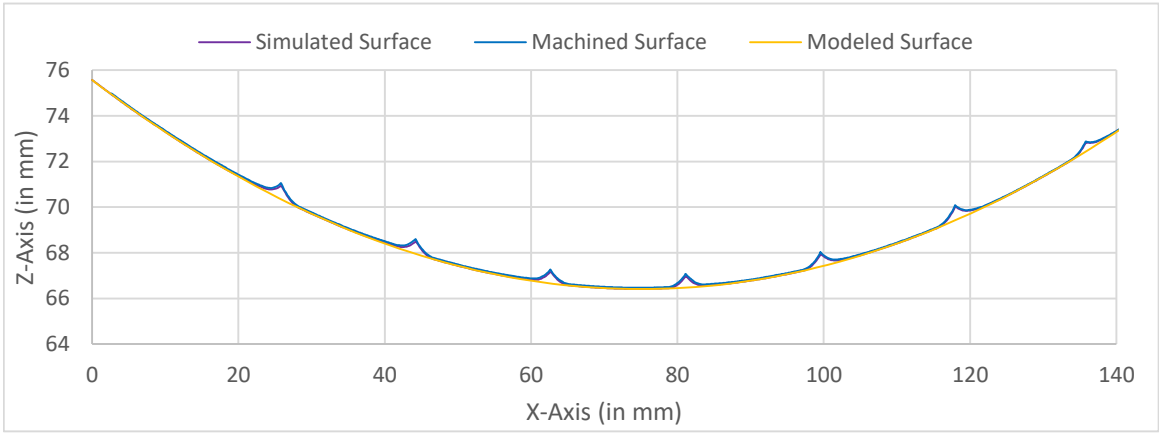
Figure 5.4: *Three surfaces tested and verified by simulator under the section simulated surface; and by machining with DMU-80P Hi-Dyn tilt-rotary simultaneous five axis machining center under the section machined surfaces for VCRF algorithm*

After machining, the three surfaces were also compared graphically in the similar manner as done in the previous section of the Results of testing of DRD Algorithm. The data obtained from CMM at a cross section taken in the side-step direction at Y = 27.00 mm in XY plane and the data obtained from OBJ file produced by the ToolSim is compared and shown in *Figure 5.5*.

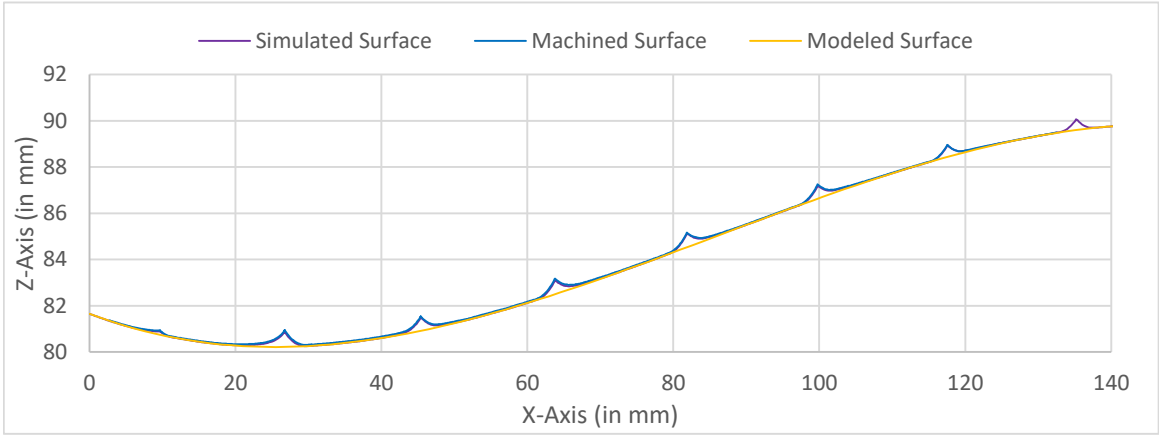
It can be seen from the graphical representation of the surfaces in *Figure 5.5* that the results obtained from the testing for VCRF Algorithm are similar to the results obtained from the testing of DRD Algorithm. Surface data obtained from machined and simulated geometries for the three parts is similar and overlaps with each other, whereas deviation of the machined and simulated surfaces from the modeled surface in the form of scallops can be seen.



(A)



(B)



(C)

Figure 5.5: Graph Showing the comparison of machined and simulated data with the modeled surface at a cross-section taken in XY plane at Y=27.0 mm for (A) convex surface, (B) concave surface and (C) saddle surface for toolpath generated using VCRF algorithm

Table 3: Minimum and Maximum deviation (in mm) of the machined and simulated geometries from the modeled geometry for VCRF algorithm

Surface	Machined		Simulated	
	Min	Max	Min	Max
Convex	3.4E-04	0.51	1.1E-06	0.53
Concave	1.7E-02	0.70	1.3E-06	0.60
Saddle	6.2E-03	0.72	9.0E-07	0.62

Table 3 gives the minimum and maximum values of deviation on the machined and simulated geometries from the modeled geometry for all the three test surfaces. Maximum deviation gives the value of maximum scallop height which is higher in case of the Saddle and Concave surfaces as compared to the Convex surface.

5.4 Time Comparison

The algorithms were run on a computer with an *Intel(R) Core(TM) i7-770HQ* processor with running frequency *2.80GHz* and using *16.0 GB RAM* running *64-bit Windows 10* operating system. Since both the algorithms, the DRD Algorithm and the VCRF Algorithm, are implemented using C++ on the same platform; hence a time comparison is done for both the algorithms, which is shown in *Figure 5.6*. The comparison gives the time taken in seconds by both the algorithms for computing the tool path data for all the three surfaces, Convex, Concave and Saddle.

It can be seen from the graph that the DRD algorithm took more time to compute the tool path data as compared to the VCRF algorithm. In the DRD method, even though the equation model is simpler but the DRD method depends upon the use of Newton's method for convergence. Moreover, the bisection method is used for tilting the surface to obtain the second point of contact and tilt angle. The bisection method is also an iterative method and takes number of iterations to converge and give a solution.

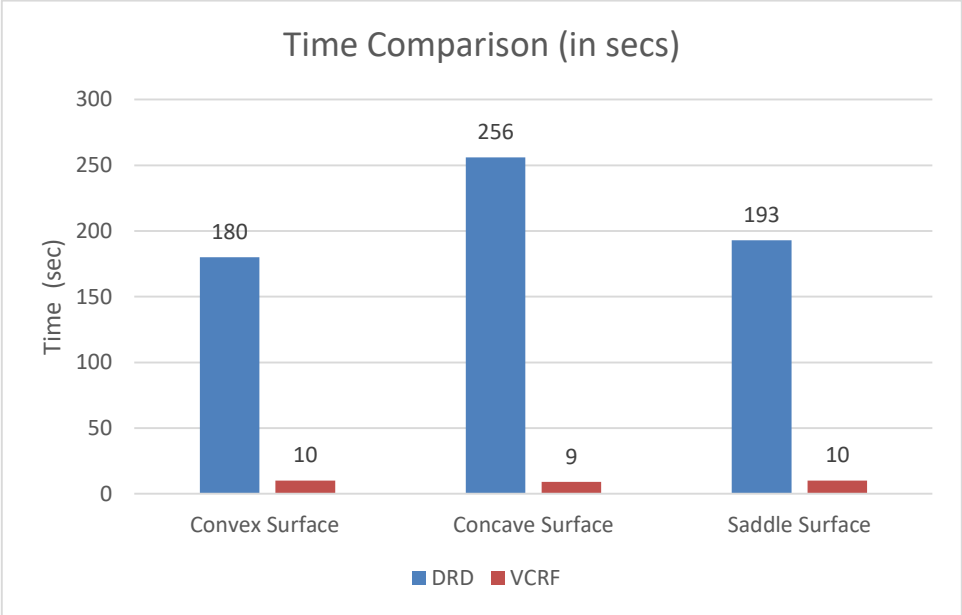


Figure 5.6: Comparison of time taken by algorithms (in seconds) for computing the tool path data for all the three surfaces

Whereas, the VCRF method eliminates the dependency on the Newton's method for obtaining the solution. No iterative process is used in the VCRF method. The solution from the implicit equations is quick. Hence, the time taken for the VCRF algorithm to compute the tool path data is less as compared to the time taken by the DRD algorithm.

It can also be seen from *Figure 5.6*, that the time taken by DRD algorithm to compute the tool path data is also dependent upon the shape of surface. For computing the tool path data for the concave surface, the DRD algorithm took more time as compared to the time taken to compute tool path data for saddle and convex surfaces. Whereas, the VCRF algorithm took a similar amount of time to compute the tool path data for all the three surfaces.

CHAPTER 6 Conclusion and Future Scope

In this work two methods, 'Drop Rotate and Drop (DRD) method' and 'Vertical and Circular Ray Firing (VCRF) method', for tool path generation were developed, implemented and tested. Three bi-cubic Bézier surfaces, consisting of concave, convex and saddle regions, are used for the toolpath generation and machined on DMU-80P Hi-Dyn tilt-rotary simultaneous five axis machining center using toroidal cutter. Multipoint machining approach was used for the tool path generation, in which the toroidal cutter touches the surface at two points of contact without any gouging.

The DRD method fires rays from the tool towards the surface and the first ray that intersects the surface travelling the shortest distance gives the first point of contact. After obtaining the first point of contact the surface is tilted iteratively around the axis of the pseudo-insert until the surface touches the tool at the second point of contact maintaining the tangency at the first point of contact without any gouging. This method reduces the dependency on the Newton's method for convergence of the solution. Even with the reduced dependency on the Newton's method the tool path generation is time consuming as an iterative process is used for getting the second point of contact.

The VCRF method eliminates the use of Newton's method for the convergence of the solution. The VCRF method uses the implicit equations of the toroidal surface defined in the Cartesian coordinate frame. In this method, vertical rays are fired from the surface towards the toroidal cutter and the ray intersecting the toroidal surface of the tool travelling the shortest distance gives the location of the first point of contact. After getting the first point of contact, then the circular rays are fired from the surface. The circular ray that intersects with the torus travelling the shortest angle gives the required tilt angle and the associated second point of contact.

Tool paths were generated by implementing both algorithms in C++, generating tool paths for three surfaces, and both simulating and machining parts from these tool paths. The machined surfaces, using both the methods emulates the simulated and anticipated part surfaces, giving the accuracy of machining using tool path generation with both the

methods without any gouging. However, the VCRF method is an order of magnitude faster than the DRD method.

6.1 Future Scope

Even though both the methods were successfully implemented and tested, there is still scope for the future work to be done. Both the methods were successfully tested for the uniform bi-cubic Bézier surfaces, but the testing of methods for the Bézier surface defined non-uniformly is still need to be done.

Moreover, the methods work fine for the singular patch of the Bézier surface but both methods need to be tested on higher order Bézier surfaces, as well as piecewise polynomials surfaces such as B-spline surfaces.

References

[1] D. Roth, F. Ismail, and S. Bedi, “Mechanistic modelling of the milling process using complex tool geometry,” *Int. J. Adv. Manuf. Technol.*, vol. 25, no. 1–2, pp. 140–144, 2005.

[2] S. Bedi, F. Ismail, M. J. Mahjoob, and Y. Chen, “Toroidal versus ball nose and flat bottom end mills,” *Int. J. Adv. Manuf. Technol.*, vol. 13, no. 5, pp. 326–332, 1997.

[3] S. Bedi, S. Gravelle, and Y. H. Chen, “Principal curvature alignment technique for machining complex surfaces,” *J. Manuf. Sci. Eng. Trans. ASME*, vol. 119, no. 4B, pp. 756–765, 1997.

[4] N. Rao, S. Bedi, and R. Buchal, “Implementation of the principal-axis method for machining of complex surfaces,” *Int. J. Adv. Manuf. Technol.*, vol. 11, no. 4, pp. 249–257, 1996.

[5] N. Rao, F. Ismail, and S. Bedi, “Tool path planning for five-axis machining using the principal axis method,” *Int. J. Mach. Tools Manuf.*, vol. 37, no. 7, pp. 1025–1040, 1997.

[6] R. K. Duvedi, S. Bedi, A. Batish, and S. Mann, “A multipoint method for 5-axis machining of triangulated surface models,” *CAD Comput. Aided Des.*, vol. 52, pp. 17–26, 2014.

[7] R. K. Duvedi, S. Bedi, A. Batish, and S. Mann, “Numeric implementation of drop and tilt method of 5-axis tool positioning for machining of triangulated surfaces,” *Int. J. Mach. Tools Manuf.*, vol. 78, pp. 1677–1690, 2015.

[8] R. K. Duvedi, S. Bedi, A. Batish, and S. Mann, “The edge–torus tangency problem in multipoint machining of triangulated surface models,” *Int. J. Adv. Manuf. Technol.*, vol. 82, no. 9–12, pp. 1959–1972, 2016.

[9] Y. He and Z. Chen, “Optimising tool positioning for achieving multi-point contact based on symmetrical error distribution curve in sculptured surface machining,” *Int. J. Adv. Manuf. Technol.*, vol. 73, no. 5–8, pp. 707–714, 2014.

[10] A. Warkentin, F. Ismail, and S. Bedi, “Comparison between multi-point and other 5-axis tool positioning strategies,” *International Journal of Machine Tools and Manufacture*, vol. 40, no. 2. pp. 185–208, 2000.

[11] A. Warkentin, F. Ismail, and S. Bedi, “Multi-point tool positioning strategy for 5-axis machining of sculptured surfaces,” *Computer Aided Geometric Design*, vol. 17, no. 1. pp. 83–100, 2000.

[12] A. Warkentin, F. Ismail, and S. Bedi, “Intersection approach to multi-point machining of sculptured surfaces,” *Computer Aided Geometric Design*, vol. 15, no. 6. pp. 567–584, 1998.

[13] P. J. Gray, F. Ismail, and S. Bedi, “Graphics-assisted Rolling Ball Method for 5-axis

surface machining," *CAD Computer Aided Design*, vol. 36, no. 7. pp. 653–663, 2004.

[14] H. T. Yau, C. M. Chuang, and Y. S. Lee, "Numerical control machining of triangulated sculptured surfaces in a stereo lithography format with a generalized cutter," *Int. J. Prod. Res.*, vol. 42, no. 13, pp. 2573–2598, 2004.

[15] R. K. Duvedi, S. Bedi, and S. Mann, "Drop and tilt method of five-axis tool positioning for tensor product surfaces," *Int. J. Adv. Manuf. Technol.*, vol. 93, pp. 617–622, 2017.

[16] R. K. Duvedi, S. Bedi, and S. Mann, "Numerical implementation of drop and tilt method of five-axis tool positioning for tensor product surfaces," *Int. J. Adv. Manuf. Technol.*, vol. 95, no. 1–4, pp. 219–232, 2018.

[17] R. K. Duvedi, S. Bedi, and S. Mann, "An efficient multipoint 5-axis tool positioning method for tensor product surfaces," *Int. J. Adv. Manuf. Technol.*, vol. 97, pp. 279–295, 2018.

[18] G. Israeli, "Software Simulation of Numerically Controlled Machining," Master's Thesis, University of Waterloo, 2006.