รายงานวิจัยฉบับสมบูรณ์

โครงการ

The Algorithms for Simulation of Inverse Kinematics of a Five-Axis Milling Machine

แบบจำลองการทำงานของเครื่องจักร CNC 5 แกน โดยใช้คอมพิวเตอร์กราฟฟิก 3 มิติ

โดย ผศ.ดร. หมัดอามีน หมันหลิน

31 กรกฎาคม 2546

สัญญาเลขที่ PDF/93/2544

รายงานวิจัยฉบับสมบูรณ์

โครงการ

The Algorithms for Simulation of Inverse Kinematics of a Five-Axis Milling Machine

แบบจำลองการทำงานของเครื่องจักร CNC 5 แกน โดยใช้คอมพิวเตอร์กราฟฟิก 3 มิติ

โดย ผศ.ดร. หมัดอามีน หมันหลิน

Information Technology Program

Sirindhorn International Institute of Technology

Thammasat University, Rangsit Campus,

Tel: 9869101 Fax: 9869113

email: amin@siit.tu.ac.th

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณสำนักงานกองทุนสนับสนุนการวิจัย (สกว) ที่ได้เอื้อเฟื้อเงินทุนในการทำงานวิจัยนี้จนสำเร็จลุล่วงไปด้วยดี

และขอขอบคุณ Assoc. Prof. Dr. S. S. Makhanov ที่เป็นผู้ให้คำแนะนำ รวมไปถึงข้อท้วงติงต่าง เพื่อปรับปรุงงานวิจัยนี้ให้มีความถูกต้อง และสามารถตีพิมพ์ในวารสารนานาชาติได้

Table of Contents

1. Abstract (บทคัดย่อ)

Project Code: PDF/93/2544

Project Title: The Algorithms for Simulation of Inverse Kinematics of a Five-Axis Milling Machine

ชื่อโครงการ: แบบจำลองการทำงานของเครื่องจักร CNC 5 แกน โดยใช้คอมพิวเตอร์กราฟฟิก 3 มิติ

Investigator: ผศ.ดร. หมัดอามีน หมันหลิน

Information Technology Program

Sirindhorn International Institute of Technology

Thammasat University, Rangsit Campus,

Tel: 9869101 Fax: 9869113

Email Address: amin@siit.tu.ac.th

Project Period: 1 สิงหาคม 2544 – 31 กรกฎาคม 2546

We present the **algorithms to simulate non-linear kinematics** of the five-axis CNC milling machine. The simulator is based on **3D representation** and employing the **inverse kinematics** approach to derive the corresponding rotational and translation movement of the mechanism. The simulator makes it possible to analyze an accuracy of a 3D tool path based on a prescribed set of the cutter location (CL) points as well as a set of the cutter contact (CC) points. The resulting trajectory of the tool is not unique and depends on the initial set up of the machine. Furthermore, the simulator can be used to **simulate the milling process, verify the final machining and estimate the errors** of the actual tool path before the real workpiece is actually tested with the real machine. Thus, it reduces the cost of iterative trial and errors.

Next, we present a series of cutting experiments performed by means of the proposed software and evaluate the accuracy of milling. We demonstrate that the proposed graphical 3D software presents an efficient interactive approach to the interactive modification of a tool path based on an appropriate set of transformations as well as to **verification of the tool path optimization algorithms.** The result of the simulation is tested using the Maho600E Five-Axis CNC Milling Machine at Computer Integrated Manufacturing Laboratory in Asian Institute of Technology.

Keywords: Five-axis CNC, Inverse kinematics, Tool path optimization

Abstract (บทคัดย่อ)

งานวิจัยนี้นำเสนอแบบจำลองการทำงานของเครื่องจักร CNC 5 แกน โดยใช้คอมพิวเตอร์กราฟฟิก 3 มิติ ซึ่งสามารถจำลองแนวการเคลื่อนที่ของ cutter ได้ทั้งแบบใช้จุดกึ่งกลาง (CL) หรือจุดสัมผัส (CC) ของ cutter กับชิ้นงานได้ เนื่องจากการเคลื่อนที่ของ cutter มีได้หลายแบบ ขึ้นอยู่กับโปรแกรม G-Code หรือ Inverse Kinematics ของเครื่องจักร CNC นั้น ๆ การใช้แบบจำลองคอมพิวเตอร์ จำลองการเคลื่อนที่หลาย ๆแบบทำให้เราสามารตรวจสอบและคำนวณข้อผิดพลาดเบื้องต้นได้ และเลือกวิธีที่ดีที่สุดในการกัดชิ้นงานที่มีข้อผิดพลาดน้อยที่สุด จากนั้นจึงส่งโปรแกรม G-Code ไปยังเครื่องจักร CNC 5 แกนจริง ๆต่อไป

วิธีการจำลองการกัดชิ้นงาน ที่นำเสนอโดยงานวิจัยนี้ ได้ผ่านการทดสอบกับตัวอย่างชิ้นงานหลาย ๆรูปแบบ โดยใช้เครื่องจักร CNC 5 แกน MAHO 600E จากห้องปฏิบัติการที่สถาบันเทคโนโลยีแห่งเอเชีย ซึ่งพบว่าสามารถลดข้อผิพลาดจากการกัดแบบปกติได้ประมาณ 5 – 65 % นอกจากนั้น การกัดชิ้นงานโดยใช้วิธีการนี้สามารถทำได้ภายในครั้งเดียว และไม่มีปัญหาเรื่องการแตกหักของ cutter แม้แต่ครั้งเดียว เนื่องจากได้ผ่านการตรวจสอบโดยใช้แบบจำลองคอมพิวเตอร์มาแล้ว

2. Objectives

1. Develop the **algorithms** and **a new software** prototype for simulation, optimization, and verification of cutting operations of a five-axis milling machine.
2. Develop **methodological principles** and recommendations for practical applications.
3. **Publish the results** in internationally recognized journals

3. Research Plan

Research and Programming	Year 1				Year 2			
	1-3	4-6	7-9	10-12	1-3	4-6	7-9	10-12
State of art review in Solid Modeling, and Simulators of Robots.								
Pilot programming, tool path Generation, Inverse kinematics Practical machining.								
Kinematics error analysis, Virtual Environment.								
Virtual Milling Machine ver. 1 Error Estimator ver. 1.								
Solid Model of Five-Axis Machine, CL and CC points computation Practical machining.								
Gouging, undercut/overcut, Practical machining, Error analysis.								
Virtual Milling Machine ver. 2 Error Simulator ver. 2.								
Practical machining, Error analysis, Debugging.								
Verification of the system.								

4. Actual Output

1) **Publications** published in recognized international journals and conferences.

2) **Software Prototype of the Virtual Five-Axis Milling Machine (VMM)** consists of the following algorithms:
   - **Inverse Kinematics Algorithms (Interactive Post-Processing Software)** capable of generating a tool path in the five dimensional spaces based on the inverse kinematics of the five-axis milling machine. The software evaluates the correct orientation of the tool for an arbitrary sculptured surface, including saddle type regions, multiple extreme, steep regions having a high curvature etc. The efficient preprocessing includes a user-friendly graphic interface and some preliminary optimization tools such as the angle adjustment and switching.
   - **The Error Estimator Algorithms.** We present the Error Estimator based on the inverse kinematics of the five-axis Milling Machine. The Error Estimator makes it possible to predict and visualize the milling errors associated with the trajectory of the tool tip as well as the scallops between the consecutive tool tracks.
   - **Algorithms for optimization of rotations angle near stationary points (Errors Minimization Algorithms).** We consider a new algorithm designed for five-axis milling to minimize the kinematics error near the stationary points of the machined surface. Given the tool orientations, the algorithm optimizes the required rotations on the set of the solutions of the corresponding inverse kinematics equations. We solve the problem by means of the shortest path scheme based on minimization of the kinematics errors.

3) **Series of cutting experiments.** It is plain that debugging the professional software requires a large series of cutting experiments including elementary surfaces as well as sophisticated composite surfaces. We present such a series of experiments and outline the modifications of our software prototype.

The software prototype can be used in the CAD/CAM industries to produce and verify the NC program (G-Code) of the workpiece before the real cutting operation. In addition, the proposed software could constitute a supplementary tool for industrial companies to transfer and develop the new NC technologies (instead of using them). Besides, Thai academic or corporate research group could encourage industrial companies to:

- facilitate renovation of the production schemes by replacing the 3-axis machining by five-axis machining,
- examine possibilities of five-axis NC machining as applied to new industrial prototypes,
- employ the new NC code generation techniques without involving international software vendors.

Finally, the software may be used in academic institution for teaching and training on five-axis machine without having to buy an expensive five-axis machine.

Remark: By the "software prototype" we understand a code capable of performing the required task, but not endowed with a commercial interface. Given the high computational complexity of the problem, this ponderous task can only be achieved by an extended software engineering team.

Summary of the Actual Output according to a Research Plan

Year	Month	Actual Output
1	1-6	1. Solid Modeling, Simulators of Robots. State of art survey. – **Done** 2. Inverse kinematics algorithms. – **Done** 3. Post processing programs for G-Code generation. – **Done** 4. Results: Practical Machining. – **Done**
	7-12	1. Virtual Milling Machine ver. 1 (without machine movement). – **Done** 2. Error Estimator ver. 1. – **Done** 3. Results: Practical Machining. – **Done** 4. Technical Report. – **Done** 5. Paper: International conference on Production Research. – **Done** 6. Paper: IEEE International Conference on Industrial Technology. – **Done**
2	1-6	1. Interactive Virtual Milling Machine with 3D simulation of machine movement and runtime G-Code modification. – **Done** 2. Results: Error analysis of CL-points and CC-Points – **Done** 3. Results: Error Minimization – **Done** 4. Results: Practical Machining – **Done**
	7-12	1. Results: Gouging, undercut/overcut – **Done** 2. Results: simulation compared with practical machining and other commercial CAD/CAM systems – **Done** 3. Virtual Milling Machine and Error Simulator ver. 2 (with machine movement). – **Done** 4. Debugging and verification of the system – **Done** 5. Technical report. – **Done** 6. Paper: 19th International conference on CAD/CAM and Robotics (CAR&FOF) 2003. – **Done** 7. Paper: Computer Aided Design (CAD) Journal – **Done** 8. Paper: Mathematics and Computer Simulation Journal – **Done**

5. Publications

Accepted International Journal (Article in Press):

[1]. M. Munlin, S. S. Makhanov and E.L. J. Bohez, "Optimization of Rotations of a Five-Axis Milling Machine Near Stationary Points", accepted for publication in Computer Aided Design (CAD) Journal.

[2]. S.S. Makhanov, M. Munlin and S.A. Ivanenko. "New Numerical Algorithms to Optimize Cutting Operation of a Five-Axis milling machine", accepted for publication in Mathematics and Computer Simulation (IMACS Journal).

Published International Conferences:

[3]. M. Munlin and S. S. Makhanov, "A Software for Simulation of Inverse Kinematics of a Five-Axis Milling Machine", International Conference on Production Research, Prague, 29 July – 3 August, 2001.

[4]. M. Munlin, A Constraint-Based Virtual 5-Axis Milling Machine Simulator, International Conference on Production Research, Prague 29 July-3 August 2001.

[5]. M. Munlin, "Tool Path Simulation Using a Virtual Five-Axis Milling Machine" Proc. Of 2002 IEEE International Conference on Industrial Technology, Bangkok, 11-14 December 2002.

[6]. M. Munlin, "Errors Estimation and Minimization for a Five-Axis Milling Machine", Proc. Of 2002 IEEE International Conference on Industrial Technology, Bangkok, 11-14 December 2002.

[7]. M. Munlin, S. S. Makhanov and E.L. J. Bohez, "Optimization of Rotations Near Stationary Points of A Five-Axis Milling Machine", 19th CAR-FOF 2003, Malaysia, 22-24 July 2003.

6. Simulation of Non-Linear Inverse Kinematics and Error Estimators of a Five-Axis Milling Machine

6.1 Introduction

The errors introduced during five-axis machining are common. Several physical phenomena, such as machine kinematics, thermal effects, static and dynamic loading and common-cause failures are errors that often affect the quality of the surfaces produced by five-axis machining. There are a number of error components such as thermal errors, dynamic errors, spindle errors, reference point errors, toolpath generation errors, and inverse kinematics errors [1,2,3,4,5]. However, the particular effect of machine kinematics and geometric errors seems to be the most significant [6,7,8]. These errors, however, may be detected and minimized in advance before the real workpiece is being machined. In order to compute and estimate such errors, the simulation of five-axis milling is a must. The software must include tool-path tracing, dynamic display of all moving elements combined with realistic solid modeling. This paper introduces software for calculating and estimating the inverse kinematics errors and constructing a G-Code from a given toolpath (or CL-points) and simulating the milling operations of a five-axis milling machine. The G-Code is constructed based on the inverse kinematics of the machine. The inverse kinematics of the machine combined with the geometric constraints constitutes a basis to develop a solid modeling system for simulation, verification and optimization of the cutting operations. In particular, the system allows for an efficient simulation of inverse kinematics of multi–axis-milling machines. Moreover, it makes it possible to build a virtual environment that enables the user to interactively evaluate the kinematics of the mechanism and estimate the geometric and kinematics errors.

Consider a typical configuration of the five-axis milling machine with the rotary axis on the table (Fig.1). The machine is guided by axial commands carrying the 3 spatial coordinates (X, Y, Z) of the tool-tip in the machine coordinate system M and the two rotation angles (A, B). The supporting CAM software generates a successive set of coordinates (X, Y, Z, I, J, K, called the cutter location points or CL-points) in the workpiece coordinate system W. Typically, the CAM software distributes the CL-points along a set of curves which constitutes the so-called zigzag or spiral pattern (Fig.2). An appropriate transformation into the M-system generates a set of the machine axial commands which provides the reference inputs for the servo-controllers of the milling robot.

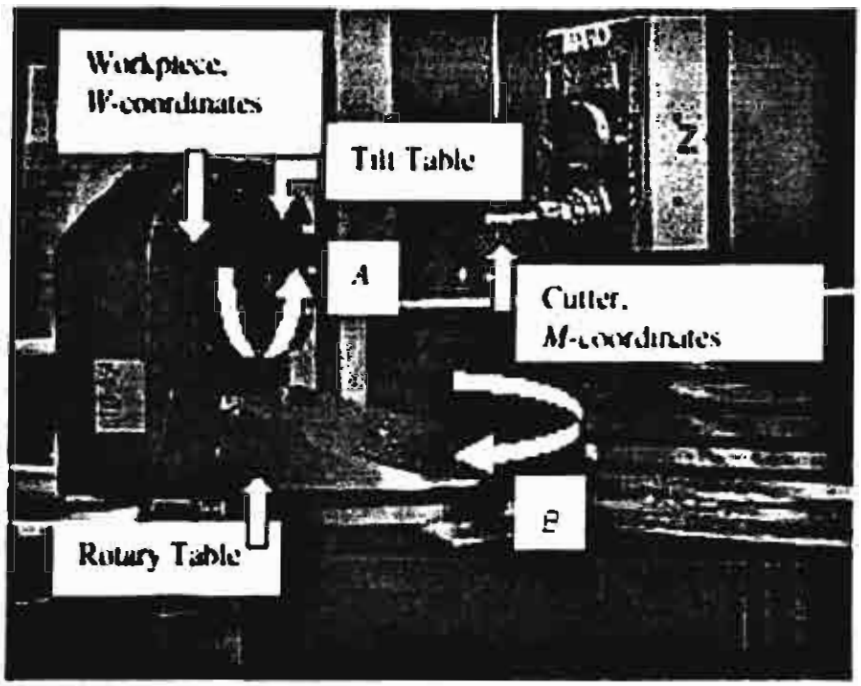

Figure 1. Five-axis milling machine

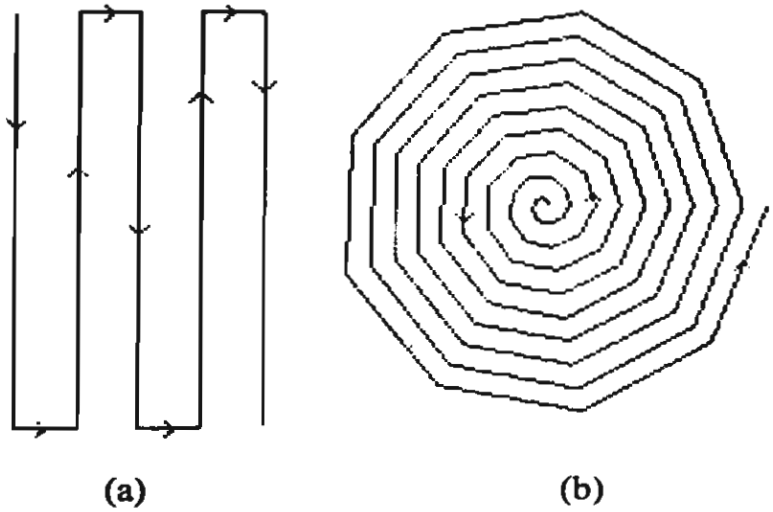

Figure 2. Zigzag (a) and spiral (b) tool path

In order to ensure a prescribed tolerance, the standard CAM software "estimates" the local errors and incorporates additional cuts (if applicable) into a single G-Code block. However, such a strategy invokes a substantial increase of the CL-points and consequently a substantial increase of the machining time [7]. Therefore, recent papers have displayed a number of sophisticated methods to optimize a zigzag or spiral pattern combined with techniques dealing with the geometric complexity of the workpiece [8,9,10]. Besides, there exist a variety of off-line methods to generate a suitable non-uniform tool-path, for instance, the neural network modeling approach [11] and the Voronoi diagram technique [10]. Verification of this method requires an up to date software involving appropriate non-linear kinematics as well as a solid model of the milling machine.

6.2 Non-Linear Inverse Kinematics of a Five-Axis Milling Machine

A five-axis machine designed for high quality manufacturing offers the highest degree of freedom with regard to the orientation of the workpiece and the cutting tool. However the current CAD/CAM commercial software is not capable of optimizing the removal of the material. We outline the following open problems: (i) Optimization of the tool angle at each cutter contact points (CC points with tool inclination) so that undercut/overcut are avoided and the geometric errors are minimized, (ii) Detection of the errors and collisions including side milling effect. The tool path between two consecutive points on a five-axis machine is not a straight line. The real CC-point path is a space curve, which needs to be compared with the reference surface. Only this operation produces a real geometric error.

Furthermore, the reference surface must be presented in the IGES format compatible with professional CAD/CAM systems. The tool geometry is a cylinder (flat end mill). Since we employ a parametric model of the surface, the corresponding equations produce the point coordinates and the normal vector. The CL point is positioned at the centerline of the tool along the surface normal. Such a representation must be changed to a CC point positioned in the tool tip plane and on the circle produced by intersection of the tool cylinder and the tool tip plane. This is shown in Fig. 3. However the input to the five-axis CNC machine is the sequence of CL-points transformed to the machine
coordinate system by means of the inverse kinematics. The algorithm to compute this location must be integrated with tool path optimization software and with a solid model of the material removal.

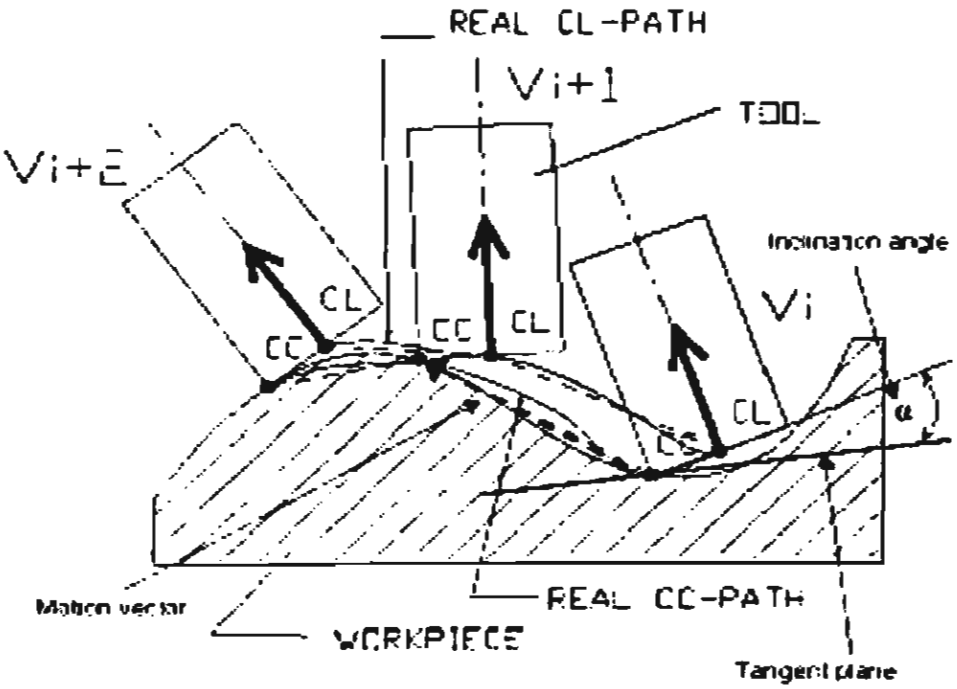

Figure 3. Tool Workpiece Contact Zone

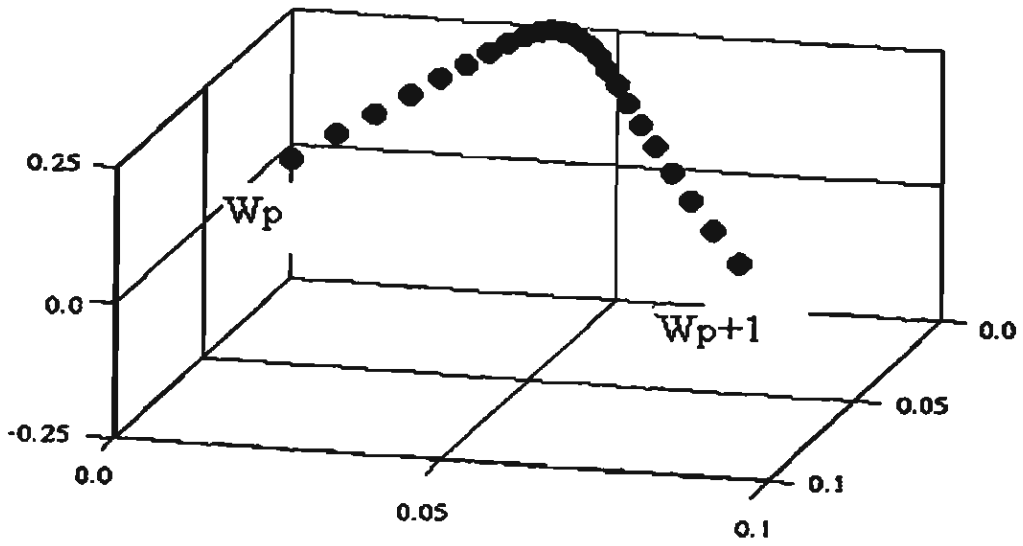

Figure 4. Simulation of non-linear tool path

Consider how the axial command translates the centers of rotation and simultaneously rotates the W-coordinates (Fig.4). Let $W_p \equiv (x_p, y_p, z_p)$ and $W_{p+1} \equiv (x_{p+1}, y_{p+1}, z_{p+1})$ be two successive spatial positions of the tool path and (a_p, b_p), (a_{p+1}, b_{p+1}) the corresponding rotation angles given by $a_p = \arctan(j_p/i_p)$, $b_p = -\arcsin(k_p)$, where (i_p, j_p, k_p) denotes the normal to the required surface at W_p. In order to calculate the tool trajectory between W_p and W_{p+1}, we first, invoke the inverse kinematics [7] to transform the part-surface coordinates into the machine coordinates $M_p \equiv (X_p, Y_p, Z_p)$ and $M_{p+1} \equiv (X_{p+1}, Y_{p+1}, Z_{p+1})$. Second, the rotation angles $a(t), b(t)$ and the machine coordinates $M \equiv M(t) \equiv (X(t), Y(t), Z(t))$ are assumed to change linearly between the prescribed points, namely,

$$\begin{aligned} M(t) &= tM_{p+1} + (1-t)M_p, \\ a(t) &= ta_{p+1} + (1-t)a_p, \\ b(t) &= tb_{p+1}t + (1-t)b_p, \end{aligned} \qquad (1)$$

where t is the fictitious time coordinate ($0 \le t \le 1$). Finally, invoking the transformation from M back to W yields $W(t) \equiv (x(t), y(t), z(t))$.

The kinematics are represented by the functions $A \equiv A(a(t)), B \equiv B(b(t))$ associated with the rotations around the primary (the rotary table) and the secondary (tilt table) axes respectively. They are specified by the structure of the machine. For the five-axis machine in Fig 1, the kinematics involving two rotations and three translations are given by

$$M(t) = B(t)(A(t)(W(t)+R)+T)+C, \qquad (2)$$

where R, T, and C are respectively the coordinates of the origin of the workpiece in the rotary table coordinates, coordinates of the origin of the rotary table coordinates in the tilt table coordinates and the origin of the tilt table coordinates in the cutter center coordinates. The general inverse kinematics are given by

$$W = (A^{-1}(B^{-1}(M-C)-T)) - R. \qquad (3)$$

In this work, we use the following sequence of transformations to calculate the inverse kinematics of the five-axis machine.

1. Transform a CL-point P_1 (x, y, z, i, j, k) into the local coordinate system (LCS) of the rotary table by means of the rotation matrix R_1 and the translation vector T_1. The new position is given by $P_2 = R_1(P_1 + T_1)$.
2. Transform P_2 into the LCS of the tilt table by translation vector T_2 and rotation R_2 and R_3. The new position given by is $P_3 = R_3 R_2 (P_2 + T_2)$.
3. Transform P_3 until it is coincident with the LCS of the tool tip by translation T_3. The final position is given by $P_4 = P_3 + T_3$.

The translation vectors are given by $T_1 = [t_{1x}, t_{1y}, t_{1z}]$, $T_2 = [t_{2x}, t_{2y}, t_{2z}]$, $T_3 = [t_{3x}, t_{3y}, t_{3z}]$, where t_{ix}, t_{iy}, t_{iz} are the corresponding offsets which depend on the reference position of the machine (Fig 1).

The rotation matrices are given by

$$R_1 = \begin{pmatrix} \cos(A) & \sin(A) & 0 \\ -\sin(A) & \cos(A) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (4)$$

$$R_2 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad (5)$$

$$R_3 = \begin{pmatrix} \cos(B) & 0 & -\sin(B) \\ 0 & 1 & 0 \\ \sin(B) & 0 & \cos(B) \end{pmatrix} \quad (6)$$

The rotation angles are

$A = \tan^{-1}(j/i)$, $0 \le A \le 2\pi$. (7)

$B = -\sin^{-1}(k)$, $0 \le B \le \pi/2$. (8)

Furthermore, given the M-coordinates of two consecutive CL-points $P_1=(x_1, y_1, z_1)$ and $P_2=(x_2, y_2, z_2)$ and the two rotation angles representing an orientation of the tool, the virtual machine will perform the required motion by means of the constrained-based algorithm [12]. At this stage, it may seem that the virtual machine performs nothing (does not yet show the movements of each machine's axis) more than the inverse transformation given by $R_1^{-1}(R_2^{-1}R_3^{-1}(P_4 - T_3) - T_2) - T_1$. Nevertheless, the results of the inverse kinematics can be visualized and used to inspect whether or not the virtual machine performs the correct actions. According to our experiment, all surfaces have been machining with satisfactory result in a single cut. Finally, it is worth noting that the results surprised the technician who had been running the machine for many years. He rarely witnessed machining in a single cut by means of other post-processing software.

6.3 Errors Estimation and Minimization

The CAD software generates the CL points in the workpiece coordinate system whereas the CAM software generates the G Code in the machine coordinate system from the prescribed CL points.

The G-Code guides the cutting tool of the five-axis machine to travel along a nonlinear trajectory to reach the required CL point. The nonlinear trajectories constitute a trajectory surface, which is slightly different from the actual surface. The error surface is estimated by computing the difference between the actual and trajectory surfaces.

Let four points $W1(i,j)$, $W2(i+1,j)$, $W3(i+1,j+1)$ and $W4(i,j+1)$ in Figure 5 be the grid $\{(u,v)_{ij}\}$ which represents the actual surface $S(u,v)$. First, we apply the inverse kinematics (see section 6.2) to transform each point into the corresponding machine coordinate $M1$, $M2$, $M3$, and $M4$ respectively. Then, we perform the linear interpolation procedure on the machine coordinate M and invoke the inverse transformation from M back to W (for every t) yields the tool path $W(t)=(x(t),y(t),z(t))$. Next, the calculated toolpath is mapped on the grid $\{(u,v)_{ij}\}$. Finally, using the bilinear interpolation procedure on the grid $\{(u,v)_{ij}\}$ yields an approximation of the machined surface $T(u,v)$. The error surface is then calculated by

$$\omega_s(u,v) = |S(u,v)-T(u,v)|, \qquad (9)$$

where $S(u,v) = (x(u,v),y(u,v),z(u,v))$ is the actual surface and $T(u,v) = (x_T(u,v),y_T(u,v),z_T(u,v))$, is the trajectory surface as shown in Figure 5. These errors are estimated and visualized graphically by the virtual machine. We propose two algorithms, namely "angle adjustment" and "angle switching" to minimize such errors.

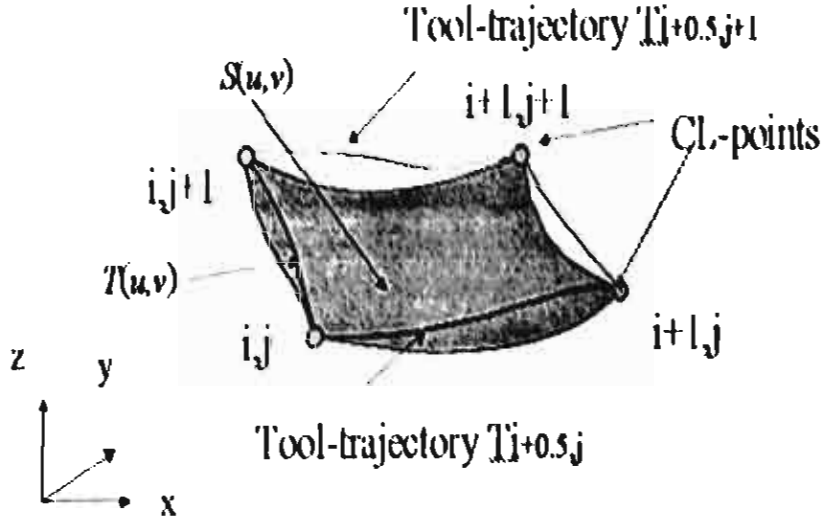

Figure 5. Error surface computation

6.3.1 Angle Adjustment

If the a-angle jumps unexpectedly from minimum to maximum (e.g. from 5 degree to 355 degree) or vice versa, this may cause an expected collision among each machine's axis. Therefore, the angle adjustment algorithm is introduced to produce the continuous variation of the angles. The sequence $\{a_p\}$ should be adjusted in order to minimize the difference between the successive values. For instance if *a angle* changes from 355 to 5 the algorithm must replace 5 by 365, etc. The following C++ program illustrates the angle adjustment algorithm.

```
for(i=0;i<N;i++) {
   // get vertex
   a1 = a(i); a2 = a(i+1); ak = a2;
   // compare a(i) and a(i+1)
   if(a2 - a1  > PI ) {
        // adjust the angle
        for(k=i+1; k<N; k++)
              ak = ak - 2*PI;
   }
   else if(a2 - a1 < -PI) {
        // adjust the angle
        for(k=i+1; k<N; k++)
              ak = ak + 2*PI;
   }
```

6.3.2 Angle Switching

According to our initial setup of the five-axis machine, the *a-angle* is given by a = tan $^{-1}$ (*j*/*i*), 0 ≤ a ≤ 2π. However, the normal vector *i and j* can be in any of the four quadrants. Therefore, we can define

$$a_{base}=\begin{cases}\tan^{-1}(\frac{j}{i}),\text{if } i\geq 0 \text{and } j\geq 0,\\ \pi+\tan^{-1}(\frac{j}{i}),\text{if } (i<0 \text{ and } j>0) \text{ or} (i<0 \text{ and } j<0),\\ 2\pi+\tan^{-1}(\frac{j}{i}),\text{otherwise},\end{cases} \quad (10)$$

Furthermore, there are four sets of *a-angle* and *b-angles* within the range [0,2π] that can rotate the tool vector into the required orientation. The set of the *a-angles* is defined by $\{a_{base}, a_{base}-2\pi, a_{base}-\pi, a_{base}+\pi\}$. Note that, after the rotation by a_{base} or $a_{base}-2\pi$ the tool is positioned in the same quadrant with the original orientation whereas the rotation by $a_{base}-\pi, a_{base}+\pi$ corresponds to the tool being in opposite direction. The set of the feasible *b-angle* (b_{base},- π-b_{base}) is required to transform the tool vector into the require orientation. It is not hard to demonstrate that a_{base} or $a_{base}-2\pi$ requires the rotation $b=b_{base}=-\sin^{-1}(k)$ whereas the rotation $a_{base}-\pi, a_{base}+\pi$ corresponds to another solution $b=-\pi-b_{base}$. Therefore, we have four paths represented by four pairs of feasible *(a, b)* to transform the tool vector p_i into the required location p_{i+1}. That is,

$$p_{i+1} = \begin{cases} x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1}, b_{i+1} \\ x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1} - 2\pi, b_{i+1} \\ x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1} - \pi, -b_{i+1} - \pi \\ x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1} + \pi, -b_{i+1} - \pi \end{cases} \tag{11}$$

We can determine the path by examining the minimum errors at each point along the tool path. According to our experiment, every line on the tool path near or cross a stationary point such as minimum or maximum or saddle generates the loop trajectories. These loops represent the larger errors than any other points due to the longer distance the tool has to travel to cross the stationary point. Therefore, we may perform the optimization with regards to this set. We will minimize the distance traveled by the tool in the space *(a, b)* with the Euclidean distance:

$$\text{distance}(p+1, p) = \frac{(a_{p+1} - a_p)^2 + (b_{p+1} - b_p)^2}{(x_{p+1} - x_p)^2 + (y_{p+1} - y_p)^2 + (z_{p+1} - z_p)^2} \tag{12}$$

The total distance traveled by the tool within the vicinity of the large errors is given by

$$\sum_{p=1}^{n} \frac{(a_{p+1} - a_p)^2 + (b_{p+1} - b_p)^2}{(x_{p+1} - x_p)^2 + (y_{p+1} - y_p)^2 + (z_{p+1} - z_p)^2} \tag{13}$$

Note that, we consider only the distant between *(a,b)* because the position *(x,y,z)* of p_i and p_{i+1} does not change. However, we have four set of feasible *(a,b)* between p_i and p_{i+1} and only one set of *(a,b)* with the smallest distant is selected to optimize the errors. The actual errors, however, are computed using equation (9) which takes into account both *(x,y,z)* and *(a,b)*.

When the tool enter the zone of large milling errors, we apply the angle-switching algorithm to select the shortest distance for the tool to travel from the current point to the next. Figure 6 illustrates the graphical result of this algorithm in which errors are clearly decreased. The errors displayed in Figure 6 enable the user to locate the "problem areas" where the loops exist or the errors are greater than the tolerance.

Note that a procedure to compute the rotation angles may become ill-conditioned in the regions $|i + j| < \varepsilon$ (ε is a small number). For instance for "flat" surfaces or near the points of maximum or minimum. As a matter of fact, small variations of the normal vector in the above mentioned regions produce sharp variations of the rotation angles and numerical instabilities. However, the algorithm from section 6.3.1 only eliminates unwanted variations of more than 180 degree. We still face the problem of large variation of the angle between 0 to 180-degree producing the loops due to the non-linearity of the inverse kinematics of the machine. This phenomenon may cause an unexpected motion of the machine and even collisions with the tilt table. We, therefore, introduce an algorithm called "Angle Switching" to select the shortest path from the feasible sequences $\{a_{i+1}\}$ in such a way that $\Sigma|a_{p+1} - a_p| + \Sigma|b_{p+1} - b_p|$ is

minimized using possible equivalent angles. For instance, a position $\{a_p,b_p\}$ could be followed by either $\{a_{p+1},b_{p+1}\}$, calculated by means of the formula above, or $\{a_{p+1}+\pi,-b_{p+1}-\pi\}$ or $(a_{p+1}\ \pi,-b_{p+1}-\pi)$. The loop detection and elimination method is based on the following two ideas. 1) Since the loops usually occur near a minimum or a maximum of the concave or convex surface, the Angle Switching algorithm must be applied only in the vicinity of an extrema. 2) In order to obtain the optimal path, the graph based shortest path algorithm must be employed.

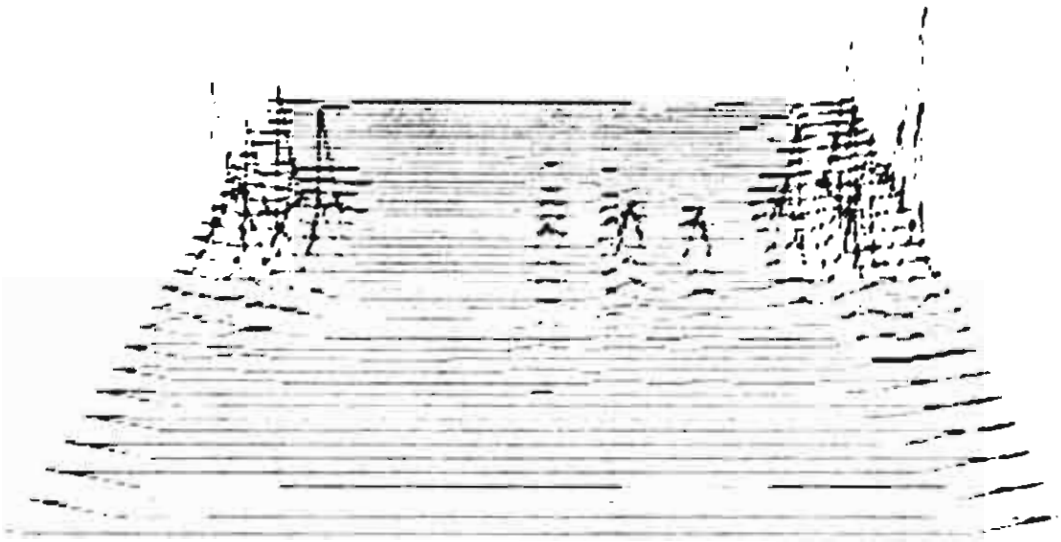

Figure 6a. Error surface without angle switching

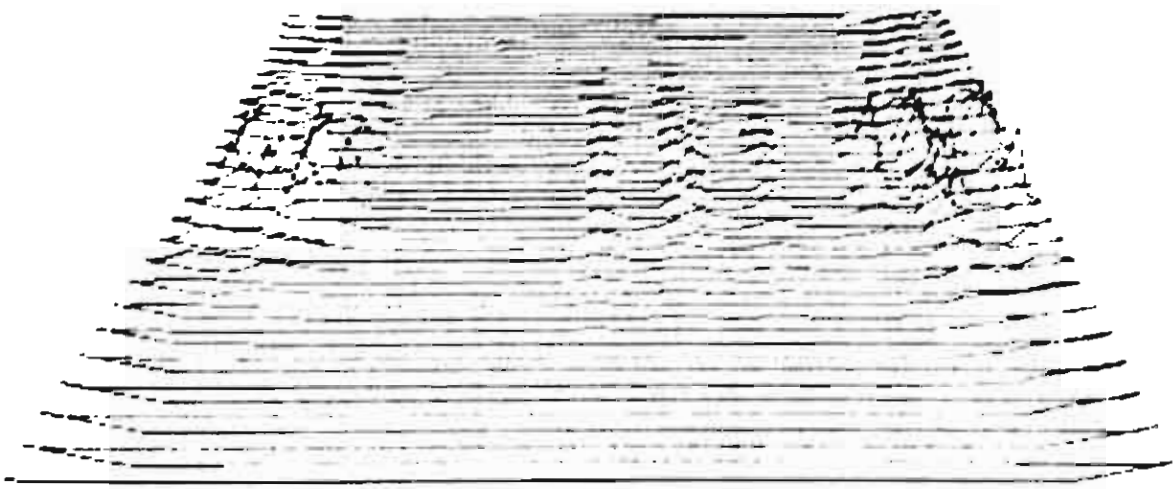

Figure 6b. Error surface with angle switching

The "Angle Switching" algorithms for optimizing the tool path by computing the smallest distance within the vicinity of the errors are described by the following steps:

1. Locate the vicinity of the large milling errors to determine the source (s) and the destination (t) vertices. In this work, we demonstrate the tool path optimization based on the distant optimization using the saddle surface that contains both convex (maximum) and concave (minimum) areas. We use our virtual five-axis simulator to graphically visualize the errors near the maximum and minimum of the saddle surface and determine the source and the destination vertices as well as the area of vicinity for

optimization. The large circle or loop (Fig 10a) represents the maximum error of the saddle surface.

2. Construct the graph to represent four feasible paths from *s* to *t* using the adjacency list. The graph nodes represent the vertices and the arcs represent the distance between two adjacent nodes of all four paths as illustrated in Figure 7. We do not create the error graph because error computation is much more expensive than the distance computation. However, the average error from *s* to *t* is proportional to the average distance from *s* to *t*, such that distant optimization gives the result proportional to error optimization.

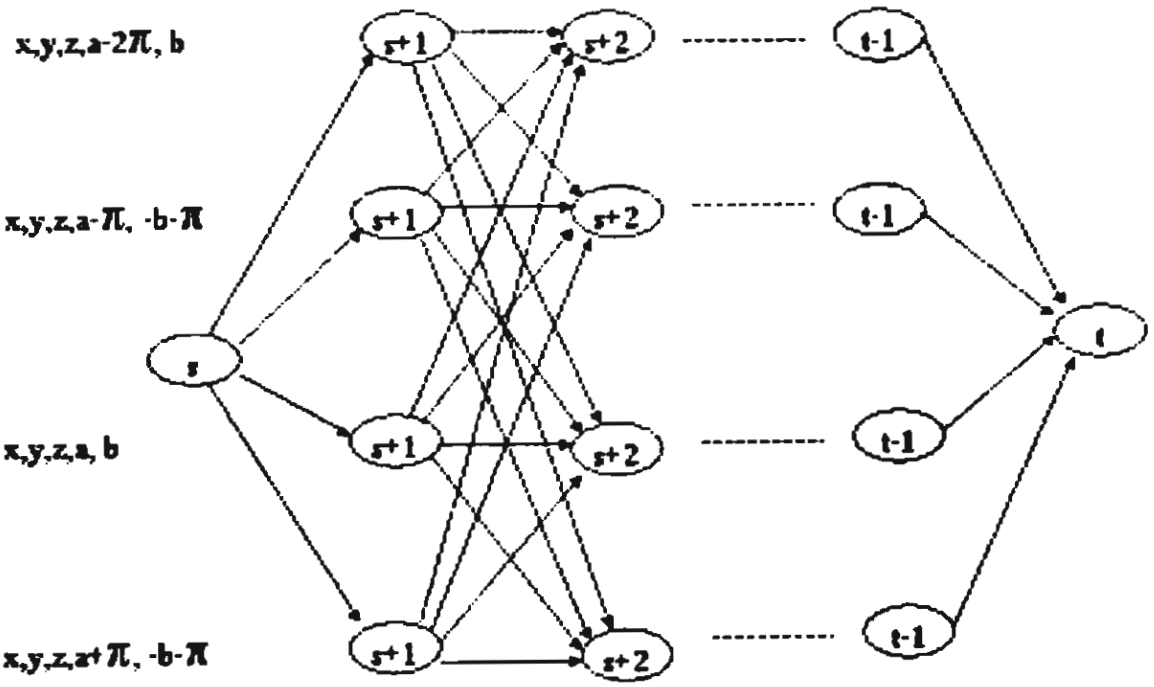

Figure 7. Distant graph represents four feasible paths from s to t

3. Apply the Dijkstra's shortest path algorithm [13] to compute the smallest distant from *s* to *t*, from the distant graph in Figure 7.

4. Update all vertices from *s* to *t* using the output shortest path from the previous step.

5. Increase the number of vertices from *s* to *t* until the total errors are minimized. Note that the position of *s* and *t* on the surface is the same, although their new vertex indices are different. The new vertex indices are computed using the following equation:

$$new\_id = old\_id * \frac{(new\_size - 1)}{(old\_size - 1)} \tag{14}$$

6.3.3 Example

The objective of the pilot cutting experiments is the calibration of the parameters involved in the inverse kinematics. The inverse kinematics transforms the tool reference vector (x, y, z, i, j, k) fixed to the workpiece into the machine coordinates X, Y, Z, A, B fixed to the machine frame. A parametric saddle surface designed for a telephone set, which contains both convex (maximum) and concave (minimum) regions is used as a case study to demonstrate the error estimation and minimization. The experiment constitutes the basic test of how our graphic simulation software would detect such kinematics errors, locate the problem areas, as well as minimize the errors. The parametric phone surface is given below.

$$P(u,v) \begin{vmatrix} & & 20u & 10 \\ & & 20v & 10 \\ 20\left|(u-0.4)^2+(v-0.3)^2\right| & & 20\left|(u-0.7)^2+(v-0.7)^2\right| & \\ e & +e & & \end{vmatrix} \tag{15}$$

The normal to the surface is computed using the derivative with regard to u and v.

$$N(u,v) = \frac{\partial P/\partial u \times \partial P/\partial v}{\left|\partial P/\partial u \times \partial P/\partial v\right|} \tag{16}$$

We then construct a zigzag tool path of 40x40 points.

$$n_i = 40, i \quad 0..39$$

$$n_j = 40, j \quad 0..39$$

The original CL points (X, Y, Z, I, J, K) are calculated from:

$$X_{i,j} = P(u_i, v_j)_0 \qquad I_{i,j} = N(u_i, v_j)_0$$

$$Y_{i,j} = P(u_i, v_j)_1 \qquad J_{i,j} = N(u_i, v_j)_1$$

$$Z_{i,j} = P(u_i, v_j)_2 \qquad K_{i,j} = N(u_i, v_j)_2 \tag{17}$$

The *a-angle* and *b-angle* are computed using machine inverse kinematics described in section 6.2. A and B axis are rotating simultaneously ranging from 0 to 360 and from 0 to -90 respectively. Fig 8 shows the required saddle surface. Fig 9 shows the corresponding graphs of the *a angles* and the angle adjustment. Note that the angle adjustment is required to eliminate sharp variations of the rotation angles near minimum or the maximum of the surface. Note that although the general case requires

an adjustment of B as well, the case of the saddle surface implies that B $\in$[-90,0]. Therefore the adjustment is not required.

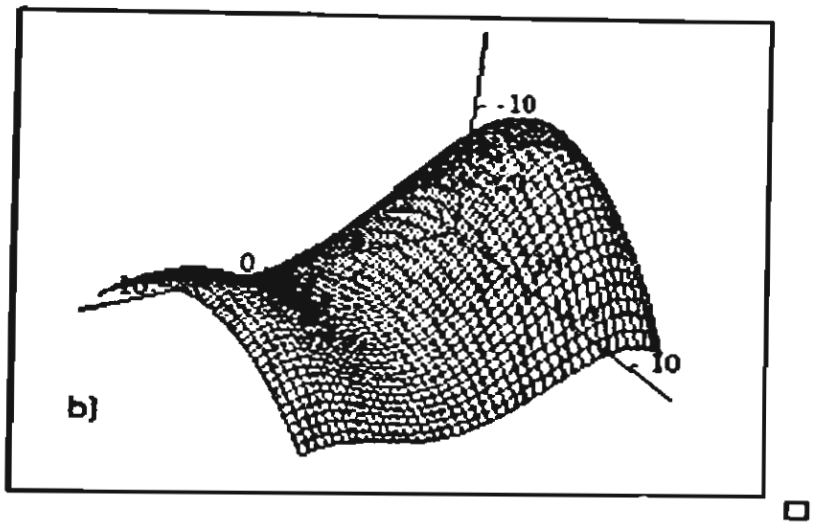

Figure 8: The saddle surface

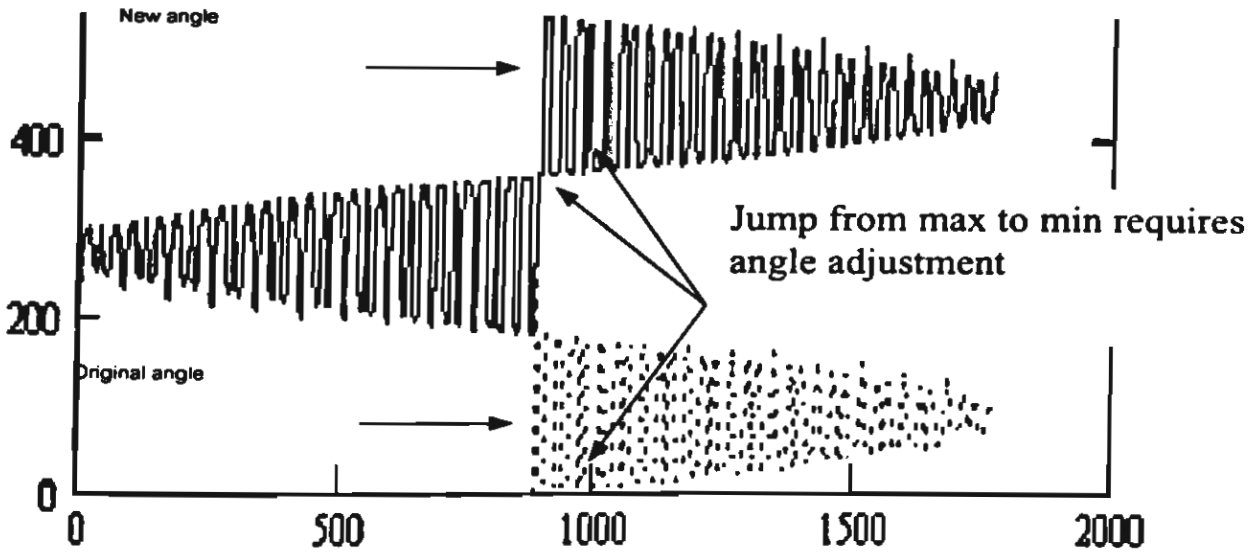

Figure 9: Angle adjustment for the saddle surface

Figure 10a illustrates the saddle surface produced by the virtual machine without angle switching. Figure 10b shows that the maximum error is at vertex number 272 and the virtual machine automatically locate the vicinity of the large milling errors to determine the source vertex as 268 and the destination vertices as 285. Note that, the large circle or loop represents the maximum error of the saddle surface at vertex number 272.

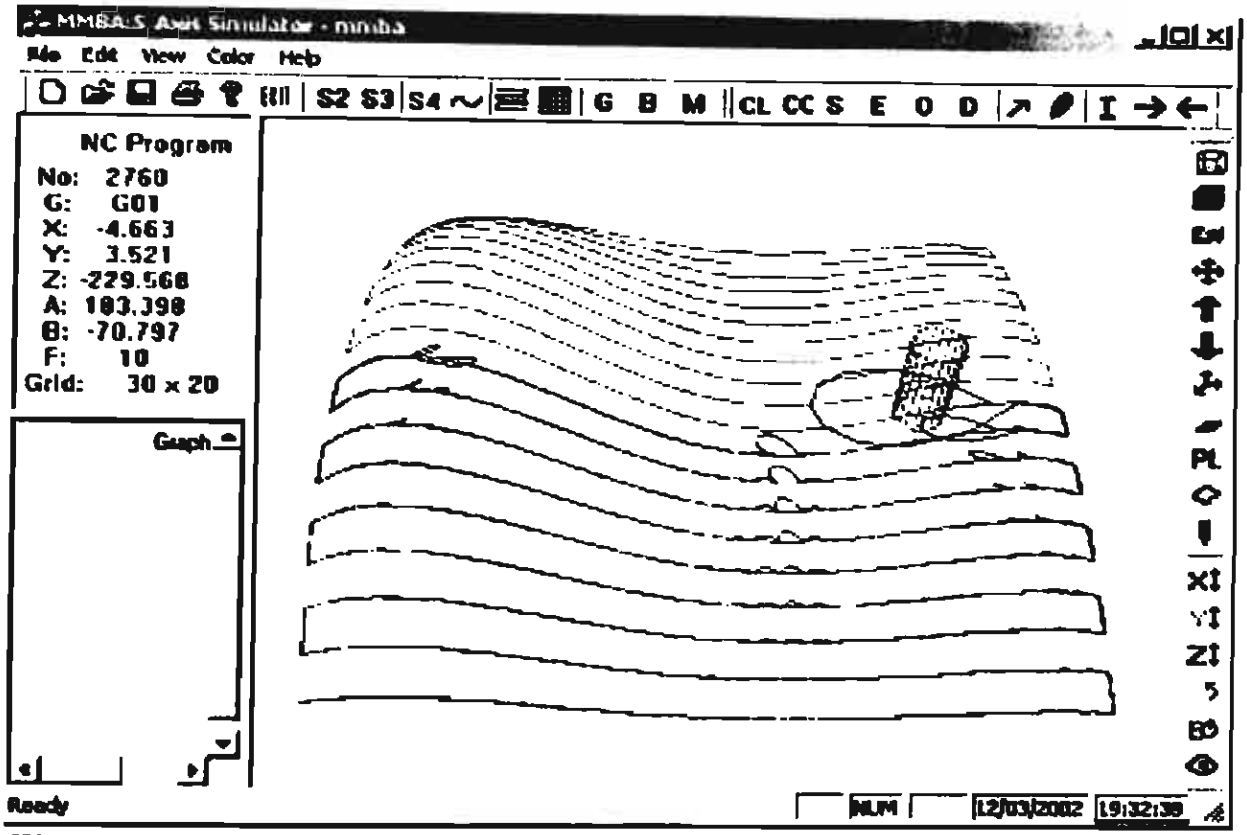

Figure 10a. Error identified by the virtual machine for the saddle surface

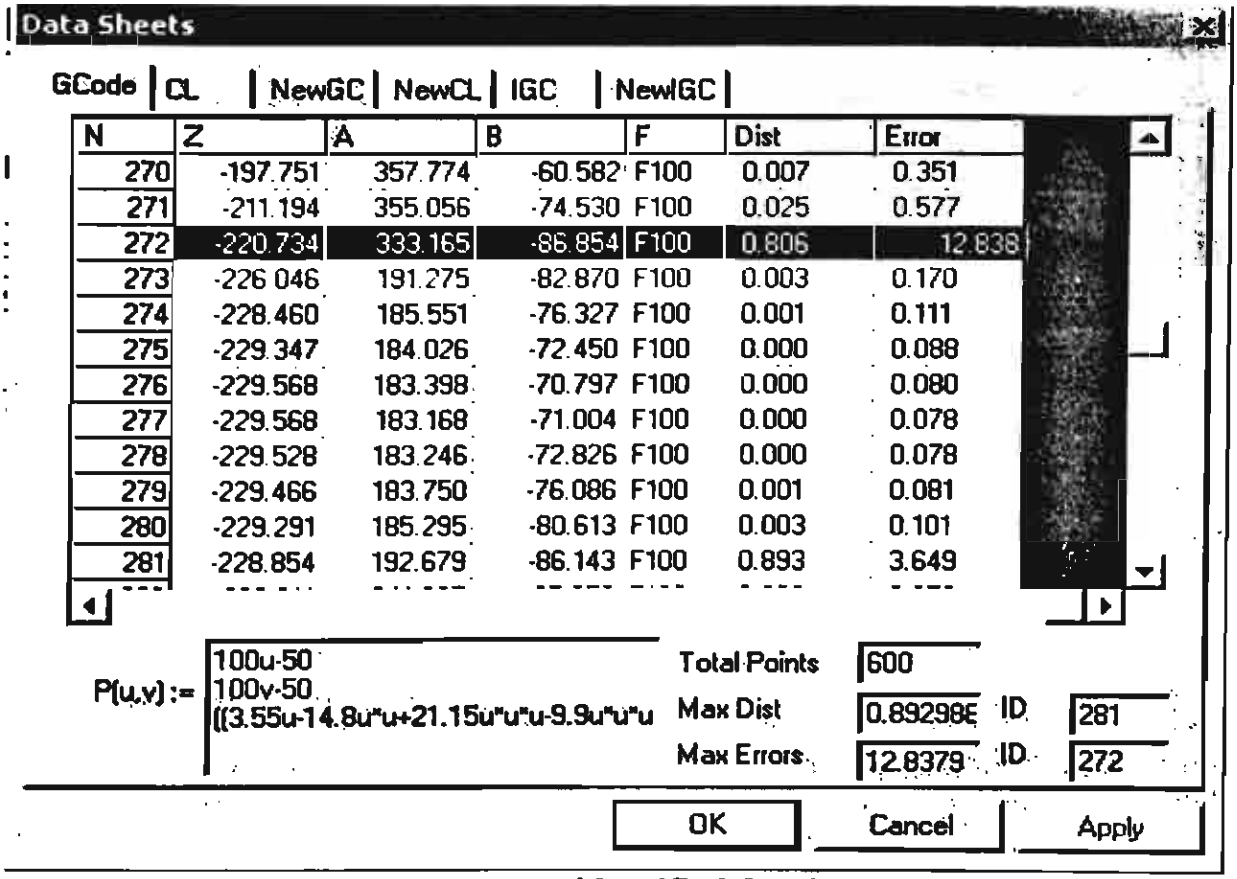

Figure 10b. Maximum error identified for the saddle surface

A non optimized tool path characterized by the loop-like trajectories induced by large gradients of the rotation angles near the stationary points is shown in Figure 10a. The loops produce a considerable error. However, the optimization makes it possible to substantially increase the accuracy.

In order to minimize such errors, First, we apply the angle switching algorithm to the above mentioned regions by constructing the graph to represent four feasible paths from vertex 268 to 285 and apply the Dijkstra's shortest path algorithm to compute the smallest distant for this region. The result is given in Figure 11 and Table 1. The large circle that makes up maximum errors has disappeared. We can apply angle switching to other regions in a similar manner.

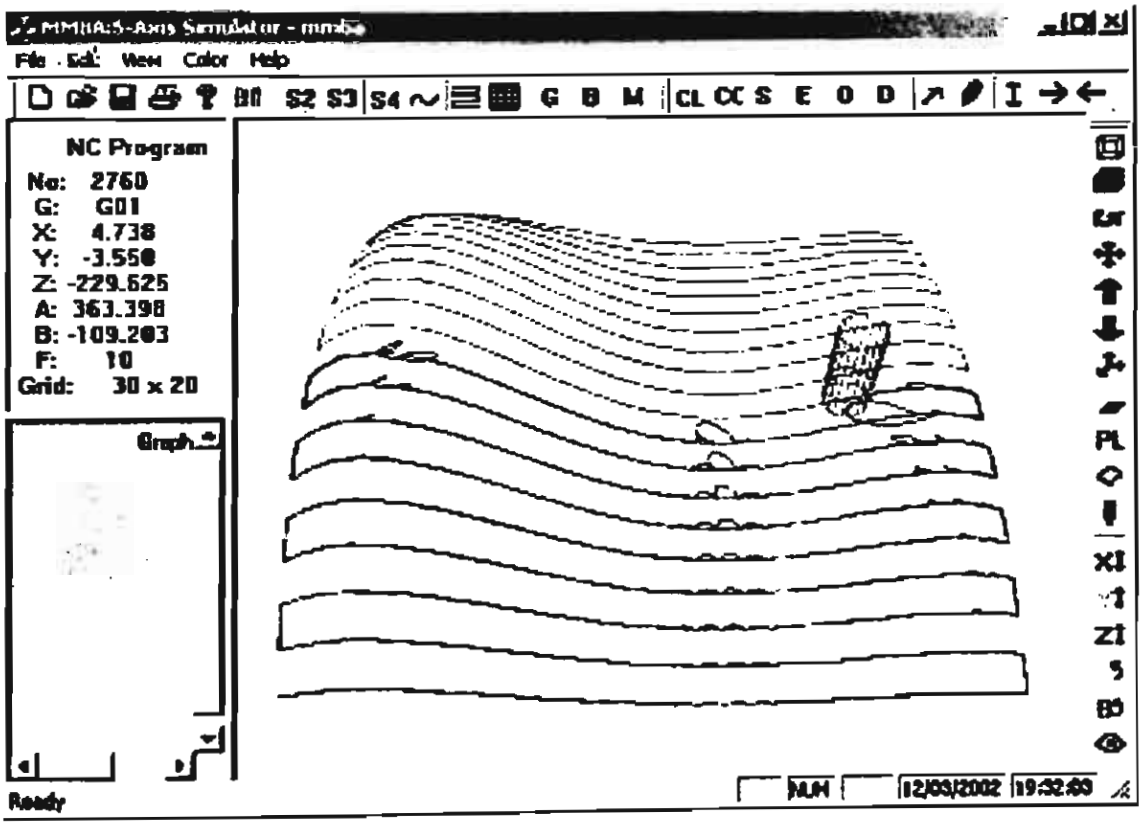

Figure 11. Optimized tool path with minimum error of the saddle surface

Consider Table 1, which displays some results of optimization. For instance, optimization of the tool paths consisting of 400 and 800 points leads to an error reduction of about 65% and 20 % respectively.

It should be noted that the optimization may make sense only for the so-called rough cutting or for cuts characterized by sharp gradients. For instance, Table 1 presents the case of a rough cut when the number of the required cutter location points is not very large. Increasing the number of points along the cutting direction (see Table 1) shows that small angular steps make the optimization superfluous (see the path 130 x 20). When the angular step is small, switching between the feasible trajectories increases the step and therefore amplifies the error.

Finally, it should be noted that we consider only smooth surfaces. However, in the case of sharp corners of the required surface (e.g. when one or both of the first derivatives are discontinuous) inserting additional CL points does not decrease the jump in the rotation angles. In this case the proposed method must be combined with either smoothing the angles or the surface itself near the singular points.

Figure 12 shows the finished part of the saddle surface using 100x100-mm workpiece and 6-mm tool size. The result is a good surface quality, with an average undercut.

Number of the CL-points	No optimization error (mm) avg/max	Optimization error (mm) avg/max	The max error decrease (%)	Path length Non-opt/opt(mm)
20 x 20	0.245/15.838	0.185/5.541	65.01	2508.88/2241.74
30 x 20	0.125/12.841	0.103/5.405	57.91	2330.73/2180.67
40 x 20	0.079/6.997	0.073/5.555	20.61	2217.96/2168.54
50 x 20	0.060/9.287	0.053/4.105	55.80	2191.15/2108.12
60 x 20	0.049/10.144	0.043/4.341	57.21	2166.09/2078.01
70 x 20	0.042/8.673	0.038/5.449	31.17	2131.56/2070.28
100 x 20	0.032/5.969	0.031/5.725	4.09	2068.25/2063.25
130 x 20	0.028/3.254	0.028/3.254	0.00	2029.41/2029.41

Table 1 Kinematics error for the optimized and non-optimized tool path.

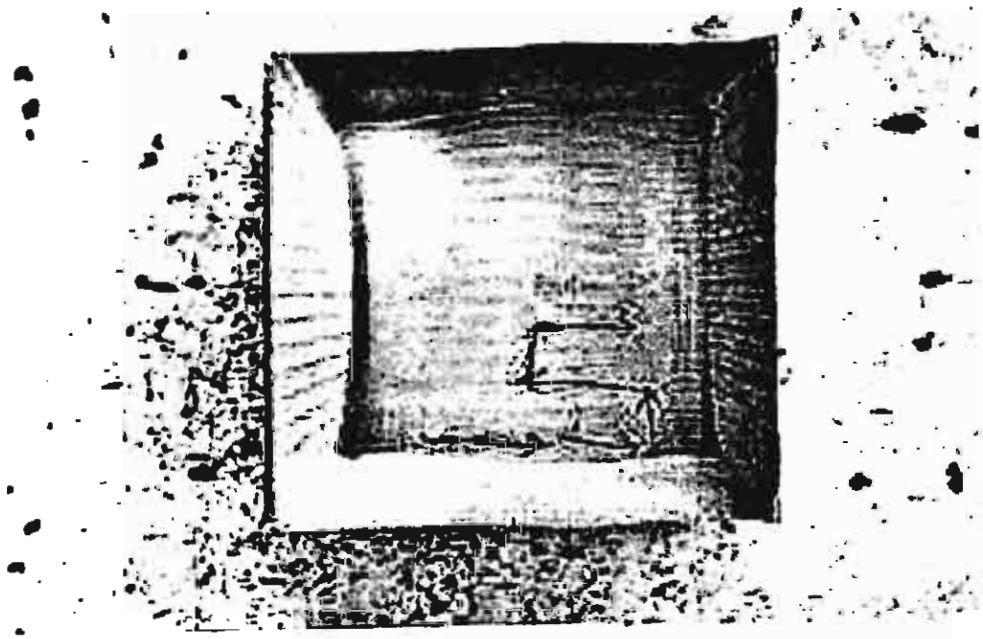

Figure 12. The final cut of the phone surface

6.4 Computation of CC Points and Tool Inclination

The CAM software usually generates a successive set of CL points, which are converted into the G-code to represent the machine motion in which the tool tip coincides with the CL point. In the case of a non-planar surface cutting with CL points usually generate undercut and overcut. Therefore optimization of the tool angle position at each cutter contact point (CC point) is required to minimize the undercut and overcut. The tool geometry is a cylinder (flat end mill). Since we employ the parametric model of the surface $\{x(u,v),\ y(u,v),\ z(u,v)\}$, the parametric equations produce the point coordinates and the normal vector. The CL point is positioned at the centerline of the tool along the surface normal. Such a representation must be changed to a CC point positioned in the tool tip plane and on the circle produced by intersection of the tool cylinder and the tool tip plane as shown in Fig 3.

Therefore, In practice, five-axis machining requires that the tool is slightly inclined with regard to the surface normal. The inclination improves the quality of the surface and makes it possible to eliminate the so-called surface and curvature interference. If the surface is convex then a surface interference is eliminated by shifting the tool in the direction opposite to the motion vector by the tool radius (Figure 3). In the convex case the tool may be or may not be inclined. It depends on the technological characteristics of the cutting process. Usually a small lead angle of about 5^0-10^0 is recommended [14] . However, a concave surface requires an inclination to eliminate the undercut (see Figure A15 in Appendix). In this case the region characterized by large kinematics errors is shifted with regard to the stationary point in the direction of the tool movement. However, if the curvature of the machined surface is small then the inclination angle required to avoid the undercut is also small. Therefore, the optimization is required only in the neighborhood of the stationary points. However, in the case of large inclinations the algorithm must deal with regions where k (the z-component of the orientation vector) is close to 1. We outline the algorithm for constructing a CC point as the following:

1. Construct the tangent plane perpendicular to the tool vector (normal vector)

$$\tau(x, y) = -\frac{i(x - x_1)}{k} - \frac{j(y - y_1)}{k} + z_1, \qquad (18)$$

where $CC = (x_1, y_1, z_1)$ represents the CC point and (i, j, k) is the tool orientation.

2. Compute two coplanar vectors v_1, v_2 at the tangent plane given by

$$v_1 = (1,0,-i/k), v_2 = (0,1,-j/k). \qquad (19)$$

3. Compute a projection of the motion vector M onto τ..

$$P = (M \bullet v_1)v_1 + (M \bullet v_2)v_2, \qquad (20)$$

where $M = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$ and where (x_2, y_2, z_2) the next successive point of the tool path.

4. Translate the CC point by r in the direction opposite to the positive direction of the projection vector to get the new CL point, where r denotes the tool radius.

$$CL = CC - rP. \qquad (21)$$

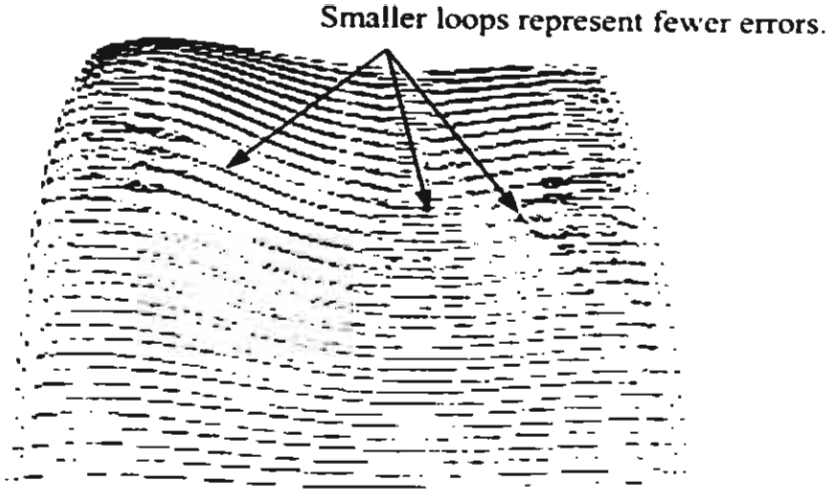

Figure 13. Tool path simulation of CC points

Fig 13 illustrates graphical view of the CC points of the saddle surface.

In case of the concave surface, the translation of the tool along the tangent plane may not eliminate the undercut. We tackle the problem by changing the tool inclination with regard to the tangent plane (see Fig 3).

It should be noted that advanced algorithms [15,16] are required to compute the curvature interference between the tool and the surface to determine the optimal tool inclination angle α. However at this stage we use an heuristic α-degree inclination and compare the errors with the case without inclination. An algorithm for constructing the CC point and tool inclination given a CC point is presented below.

1. Compute vector O orthogonal to both projection vector P and normal vector N.
   $O = N \times P$
2. Rotate the tool about O by α degree using transformation matrix T

$$T = \begin{pmatrix} P_x & O_x & i \\ P_y & O_x & j \\ P_z & O_y & k \end{pmatrix} \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \quad \textbf{(22)}$$

3. Translate the CC point by the distance of the tool radius r in the opposite direction of the projection vector to get the new CL point

$$CL = \begin{pmatrix} P_x & O_x & i \\ P_y & O_x & j \\ P_z & O_y & k \end{pmatrix} \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \begin{pmatrix} -r \\ 0 \\ 0 \end{pmatrix} + CC \quad \textbf{(23)}$$

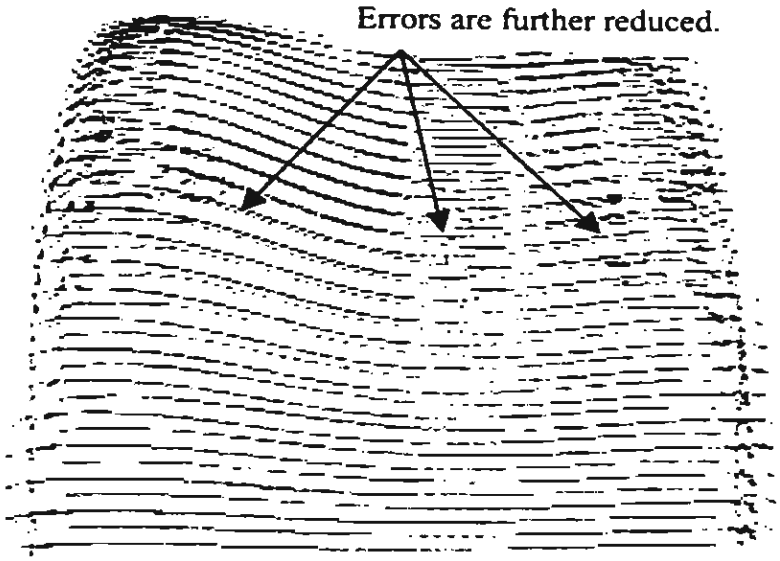

Figure 14. Tool path simulation of CC points with 10^0 tool inclination

Fig 14 illustrates graphical view of the CC points with 15^0-tool inclination.

Note that, generation of tool path using CC points reduces the size of loops at the surface stationary points. This implies that the errors have also been reduced. The CC points with tool inclination further reduce the errors as illustrated in Fig 14.

Table 2 and 3 show the error before and after optimization performed for the saddle surface when the tool is inclined by 5^0 and 15^0 lead angle respectively. Although the error decrease is no longer monotone it is of the same order. The tables clearly show that our optimization scheme is applicable to the case of an inclined tool as well. Finally, it is clear that with the kinematics error that large, errors due to curvature interference, local gouging as well as the thermal errors are negligible.

Number of the CL-points	No optimization error (mm) avg/max	Optimization error, (mm) avg/max	error decrease (%)	Path length Non-opt/opt (mm)
20 x 20	0.236/13.986	0.180/5.199	62.83	2485.82/2224.49
30 x 20	0.128/7.871	0.098/4.299	45.38	2368.56/2160.61
40 x 20	0.084/7.981	0.065/3.652	54.24	2288.54/2117.87
50 x 20	0.060/5.655	0.053/3.629	35.83	2192.93/2113.41
60 x 20	0.049/6.117	0.043/2.678	56.22	2165.54/2084.19
70 x 20	0.041/5.235	0.037/2.276	56.52	2135.49/2070.56
100 x 20	0.031/3.062	0.031/3.062	0.00	2070.54/2070.54
130 x 20	0.028/1.886	0.028/1.886	0.00	2034.90/2034.90

Table 2 Kinematics errors for the optimized and non-optimized tool path with the inclination angle 5^0.

Number of the CL-points	No optimization error (mm) avg/max	Optimization error (mm) avg/max	error decrease (%)	Path length non-opt/opt (mm)
20 x 20	0.249/11.174	0.194/4.569	59.11	2555.35/2286.40
30 x 20	0.129/6.293	0.121/4.374	30.49	2326.21/2265.82
40 x 20	0.091/8.515	0.084/4.471	47.49	2293.42/2218.14
50 x 20	0.071/7.906	0.065/3.430	56.62	2256.84/2179.24
60 x 20	0.057/5.402	0.056/2.840	47.43	2199.05/2174.77
70 x 20	0.050/6.537	0.048/3.383	48.29	2182.04/2149.31
100 x 20	0.039/5.051	0.038/3.247	35.72	2136.00/2123.82
130 x 20	0.034/3.941	0.034/3.941	0.00	2106.55/2106.55

Table 3 Kinematics errors for the optimized and non-optimized tool path. The inclination angle 15^0.

7. A Prototype Virtual Five-Axis Milling Machine (VMM)

The virtual five-axis milling machine has been developed based on the machine kinematics described in section 6. It consists of five components: **Tool Path Optimizer, Inverse Kinematics, Modelling Kernel, Virtual environment** and **Error Estimator** as shown in Figure 15.

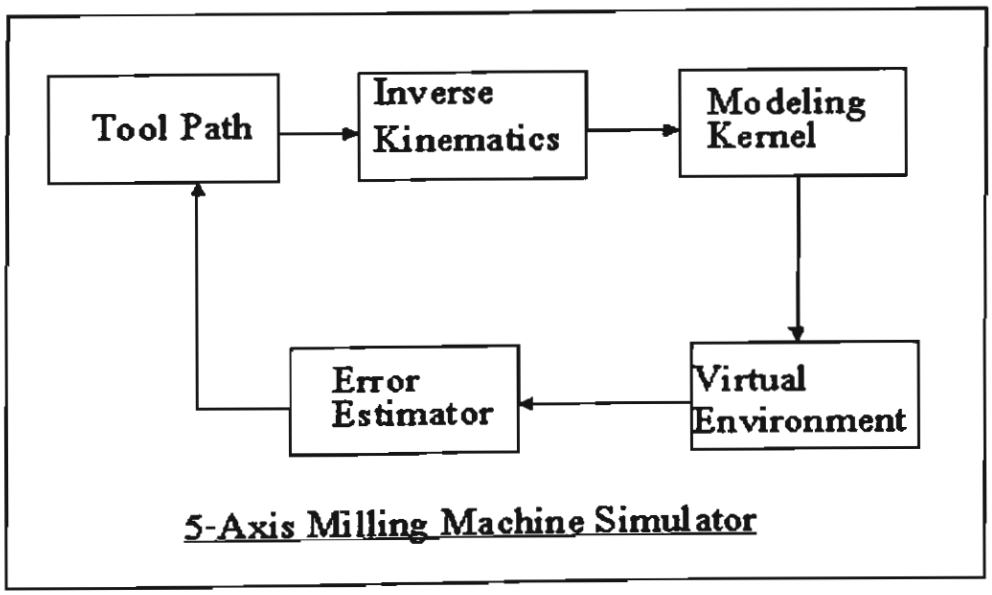

Figure 15: Virtual Five-Axis Milling Machine

7.1 Tool Path Optimizer: This is the output from the tool path generation program (such as Grid Generation) and will be the input to the simulator. However, In the early stage of the research, we first need to solve the problems of geometric (e.g. the tool tip position and orientation, tool size, undercut, overcut and the cutting step) and kinematics errors (e.g. nonlinear movement between two successive CL points). It may not worth to make use of an optimized tool path while we still have lots of geometric errors. Therefore, we will employ a parametric surface as simple as plane to the complex surface like parabolic, exponential and NURBS surfaces as the input

to the simulator which in turn produce the G-Code for the five-axis milling machine. A number of practical machining will be carried out using these surfaces. A number of examples are given in Appendix.

7.2 Inverse Kinematics: This component is already described in section 6.2.

7.3 Error Estimator: This consists of error computation and optimization algorithm to reduce the geometric and kinematics errors by modifying the existing tool path in the vicinity of large milling errors such as the surface stationary point. It will link with the tool-path generation program so that a further optimized tool-path can be regenerated. This component is already described in section 6.3.

7.4 Modelling Kernel: This represents the underlying data structure including the 3D model of the five-axis milling machine and the workpiece for constructing a 3D virtual five-axis milling machine. The simulation of the workpiece is accomplished by incorporating the trajectories generated by the VMM with the Unigraphics solid modeling system. The VMM (Figure 16) was developed based on the aforementioned machine kinematics. At present, VMM is capable of transforming any set of CL-points into the machine G-Code that can be visualized and modified using a 3D graphical user interface (GUI). It also allows the user to modify the machine parameters (such as the workpiece and tool offset) interactively. Thus, the preliminary result can be verified using the VMM before actually milling with the real machine. Therefore, the kinematics errors may be reduced beforehand. Moreover, the existing tool path can also be modified (e.g. increase/decrease the number of points and the cutting step) by the VMM. A number of surfaces will be post-processing by the VMM and then feed to the real five-axis machine for machining. Our experiments (see Appendix) have shown that the results simulated by the VMM are exactly the same as the one produced by the real machine.

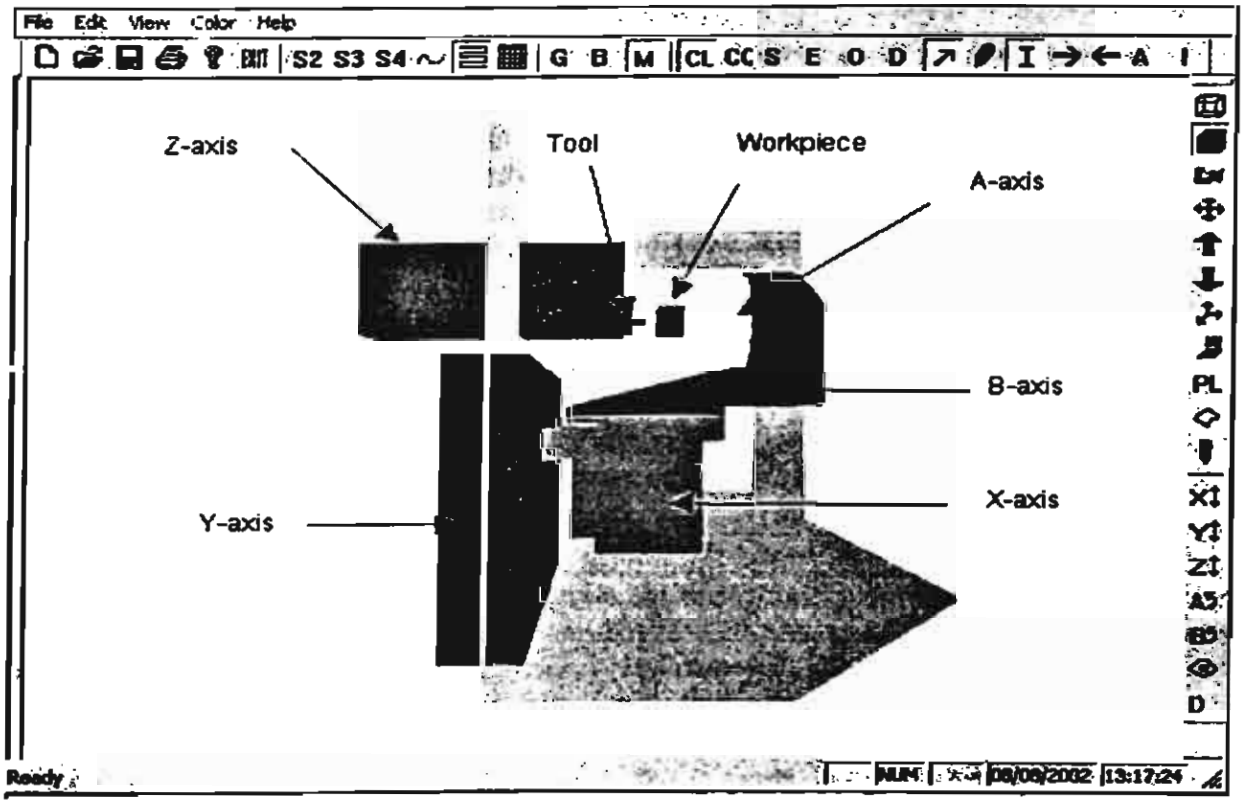

Figure 16: A solid model of the VMM

7.5 Virtual Environment: This component provides a 3D graphical environment for displaying the visual representation of the machine and the workpiece and the interaction between the user and the system (Figure 16). The OpenGL is used to build the 3D environment. Visual C++ is used to build all GUI interfaces (menus, buttons, toolbars, status bar, and dialogs). The 3D five-axis machine model is represented using polygon data structure. Simultaneous simulation of each machine's axis is done by applying the appropriate transformation from a given NC program or G-Code. The 3D interaction within the virtual environment is implemented using a quaternion-based trackball interface to allow 2D input devices such as mouse to behave like a 3D trackball so that the user can manipulate (translate, zoom, and rotate) the virtual machine freely in 3D space in any position and orientation.

The VMM is implemented using object-oriented programming on IBM compatible personal computers. The end result is the interactive simulation software, which can process any CL points and transform into the machine G-Code. The user can simulate, verify, and regenerate the G-code until the errors are minimized. The final G-Code will then be delivered to the real five-axis machine for the real machining in a single cut. Obviously, the result may be invaluable for the current CAD/CAM commercial software where they normally provide the CL-points as the input to the VMM.

8. Conclusions and Future Work

We have developed 3D interactive G-Code post-processing and tool path simulation software capable of generating and simulating a tool-path of a five-axis milling machine. The software provides 3D graphical environment to manipulate (translate, rotate, zoom) the tool path as well as the G-Code.

We also developed 3D interactive error estimator software for a five-axis milling machine. The software evaluates a correct position of the tool in the five-dimensional space, i.e. the location and the orientation of the tool and computes the kinematics errors for an arbitrary sculptured surface. The software is also capable of visualizing errors graphically and numerically. This makes it easy to locate the "problem areas" where the loops exist or the error is greater than the tolerance. Thus, large milling errors are detected and can be correct beforehand. Each CL-point may be analyzed and edited manually. It is also possible to reduce the error by modifying the CL-points manually. Besides, two algorithms called "Angle Adjustment" and "Angle Switching" is capable of minimizing of such errors during the runtime simulation. The software has been verified by surfaces having the saddle type regions, multiple extreme, steep regions with a high curvature etc.

Currently, the simulator provides preliminary results that can be used to estimate and minimize the errors graphically. The inverse kinematics combined with the 3D solid model of the machine and the error estimators will provide the realistic and powerful simulator for simulation, verification and optimization of the cutting operations.

Future work includes the development of more accurate numerical error computation method to simulate and estimate the kinematics errors. We are also still working on an algorithm called "Angle Smoothing" to minimize the shifting angles and eliminate all the loops for an arbitrary surface having an arbitrary number of extreme. Other constraints such as the scallop height, machining strip width and dynamic tool inclination will also be taken into account so that total errors are minimized. Finally, advanced data structure will be developed to represent the cutting simulation of the workpiece.

9. References

[1] Srivastava AK, Veldhuis SC, Elbestawit MA, *" Modelling geometric and thermal errors in a five-axis CNC machine tool"*, Int J Mach Tools Manufact, 1995, V35, N9, pp. 1321-1337.

[2] Mu YH, Ngoi KA, *"Dynamic error compensation of coordinate measuring machines for high-speed measurement"*, Int J Adv Manuf Technol, 1999, V15, pp. 810-814.

[3] Tu JF, Bossmanns B, Hung SCC, *"Modeling and error analysis for assessing spindle radial error motions"*, Precision engineering, V21, N2, 1997, pp. 90-101.

[4] Bhat V, De Meter EC, *"An analysis of the effect of datum establishment methods on the geometric errors of machined features"*, Int J Mach Tools Manufact, V40, N13, 2000, pp. 1951-1975.

[5] Aekambaram R, Raman S, *"Improved toolpath generation, error measures and analysis for sculptured surface machining"*, Int J Prod Research, V37, N2, 1999, pp. 413-431.

[6] Bohez ELJ, *"Compensating for systematic errors in five-axis NC machining"*, CAD, V34, 2002, pp. 391-403.

[7] Y. Koren, *"Five-Axis Interpolators"*, Annals of CIPR, V.44, N1, 1995, pp.379-382.

[8] R. Lin and Y. Koren, *"Real Time Interpolator for Machining Ruled Surfaces"*, Proc. ASME Annual Meeting, 1994, DSC-V.55-2, pp.951-960.

[9] T. Altan, B Lilly, J.P. Kruth, W. Konig, H.K. Tonshoff, C.A. VanLuttervelt and A.B. Khairt, *"Advanced Techniques for die and mold manufacturing"*, Annals of CIPR, 1993, V42, N2, pp.707-716.

[10] J.Jeong and K.Kim, *"Generation of Tool Paths for Machining Free-Form Pockets with Islands Using Distance Maps"*, Advanced Manufacturing Technology, 15(1999), pp.3111-316.

[11] S.-H. Suh and Y.-S. Shin, *"Neural Network Modeling for Tool Path Planning of the Rough Cut in Complex Pocket Milling"*, Journal of Manufacturing Systems, V15, N5, 1996, pp.295-324.

[12] Mud-Armeen Munlin, *"Interactive Constraint-Based Assembly Modeling"*, Thammasat International Journal of Science and Technology, V6, N1, 2001, pp 36-45.

[13] M. A. Weiss, Data structures and algorithm analysis in C, Addison Wesley, 1997.

[14] J.-P. Kruth, P. Klewais, optimization and dynamic adaptation of the cutter inclination during five-axis milling of sculptured surfaces, Annals of CIRP, 43(1) 443-448.

[15] D. Yu, J. Deng, Z. Duan and J. Liu, *"Generation of gouge-free cutter location path on freeform surfaces for non-spherical cutters"*, Computer in Industry V28, 1996, pp.81-94.

[16] N. Rao, F. Ismail and S. Bedi, *"Tool path planning for five-axis machining using the principal axis method"*, International Journal of Machine Tools Manufacturing, V37, N7, 1997, pp. 1025-1040.

10. Appendix

10.1 A SERIES OF CUTTING EXPERIMENTS

The objective of the pilot cutting experiments is calibration of the parameters involved in the inverse kinematics. The inverse kinematics transform the tool reference vector (x, y, z, i, j, k) fixed to the workpiece into the machine coordinates X, Y, Z, A, B fixed to the machine frame. Since the inverse transform contains specific machine dependent geometric parameters, cutting two inclined planes allows for calibration of the rotation angles A and B. Continuous variation of the angles is verified a convex parabolic surface. The three experiments result in the excellent surface quality without an undercut. Next we analyze effects of undercut/overcut in the case of a convex/concave and the saddle surface. We observe that the tool interferes with the workpiece and removes an excessive amount of the material. The experiment constitutes the basic test of how our graphic simulation software would detect and avoid the undercut/overcut by finding the optimal tool inclination.

EXPERIMENT 1. Cutting a plane.

The plane is characterized by inclination angles α 0, β 30 (β is the angle between the plane normal and the z-axis). The plane equation is then given by

$$Ix + Jy + Kz \quad 0$$
$$\sin(\beta)\, x + \cos(\beta)\, z \quad 0$$
$$z \quad -\tan(\beta)\, x$$

The corresponding zigzag tool path generated by means of the graphical software is shown in Figure A1. Next, the program performs post processing and generates a G-Code for the five-axis machine. Figure A2 displays a graphical representation of the G-Code. Figure A3 shows the final cut of the real workpiece.

Workpiece 40x40mm, Tool size 10mm
Surface quality: Excellent.
Overcut: NO. Undercut NO.

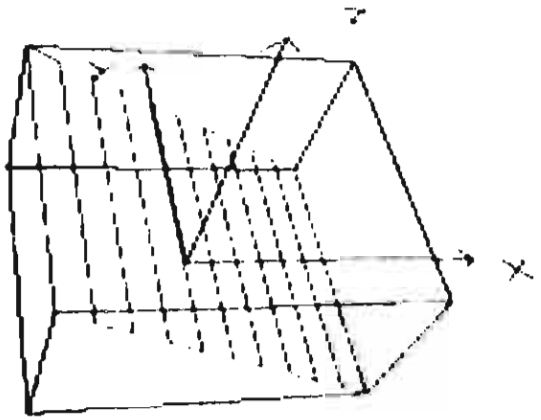

Figure A1. Plane graphical CL-Points

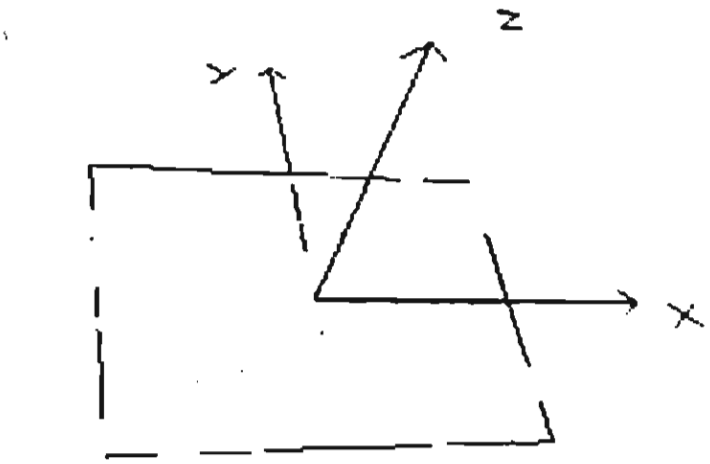

Figure A2. Plane graphical G-Code

Figure A3. Finished Plane

EXPERIMENT 2. Cutting a curvilinear tool path.

This experiment shows the capability of the virtual milling machine to generate a G-code corresponding to a curvilinear path. Observe that this feature is crucial since the software will be linked to a grid generator.

The plane equation is identical to that introduced by EXPERIMENT 1, namely

$$Ix + Jy + Kz = 0$$

$$\cos(\alpha)\sin(\beta)x + \sin(\alpha)\sin(\beta)y + \cos(\beta)z = 0$$

$$z = -\tan(\beta)\cos(\alpha)x - \tan(\beta)\sin(\alpha)y$$

However, the tool path in the (u,v) consists of parabolic curves.

Figure A4 displays the zigzag tool path for the plane produced by the virtual machine. Figure A5 displays a graphical representation of the G-Code. Figure A6 shows the final cut of the real workpiece.

Workpiece 40x40mm, Tool size 10mm
Surface quality: Excellent.
Overcut: NO, Undercut NO.

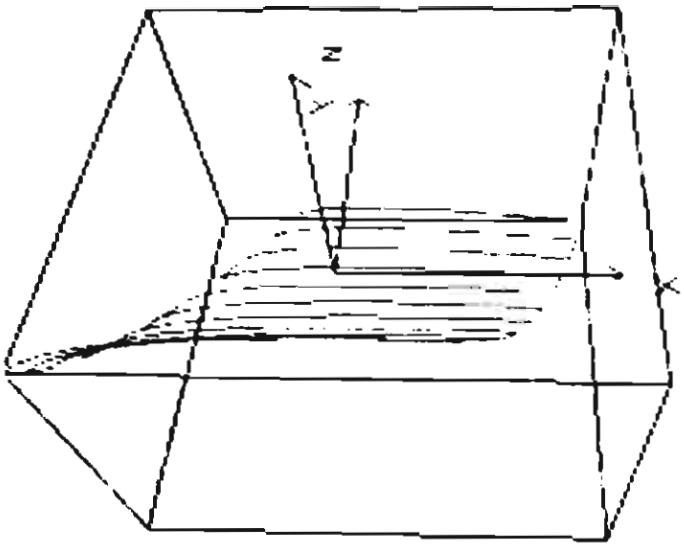

Figure A4. Curvilinear graphical CL-Points

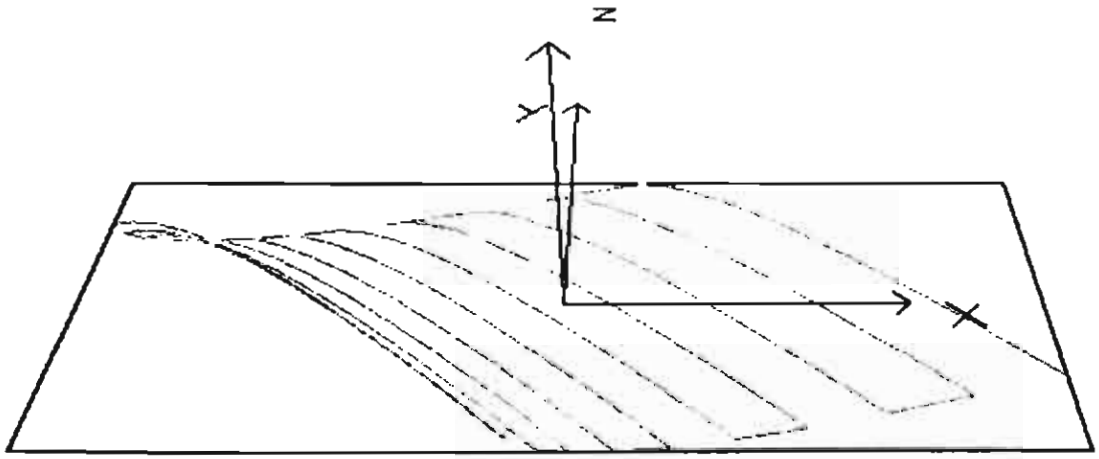

Figure A5. Curvilinear graphical G-Code

Figure A6. Finished Curvilinear

EXPERIMENT 3.Custom Tool path (Adaptation to a sinus- shaped zone of large milling errors, equivalent to 3 Axis Milling)

This experiment is performed to debug the virtual machine as applied to generate a curvilinear tool path produced by the grid generation procedure. In order to suppress an impact of the overcut or undercut we perform a 3-axis milling on the five-axis milling machine. The rotation angles are fixed, namely A=0, B=-90. Note that although we obtain a better surface quality in the zone of sharp variations, the total quality of then surface is average (due to 3-axis machining).

However the experiment clearly shows that the virtual machine is applicable to any curvilinear zigzag pattern.

Figure A7 displays the corresponding zigzag tool path. Figure A8 displays a graphical representation of the G-Code. Figure A9 shows the final cut of the real workpiece.

Note that in the case of 3-axis milling the graphical CL-Points and G-Code must produce the same pattern. This can be used to verify the errors of the NC program before the real machining.

Workpiece 40x40mm, Tool size 10mm
Surface quality: Average.
Overcut: NO, Undercut NO.

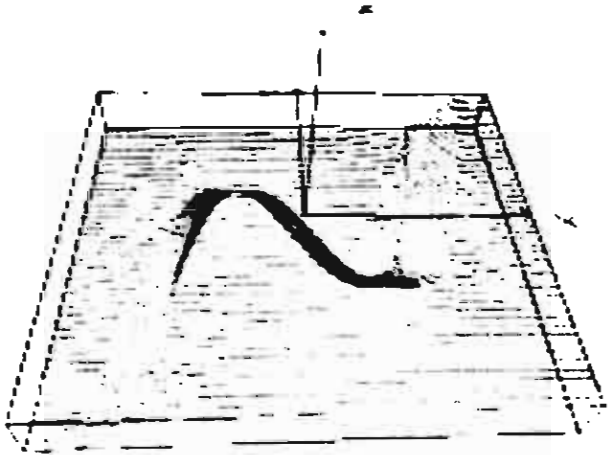

Figure A7. The graphical CL-Points of the custom tool path

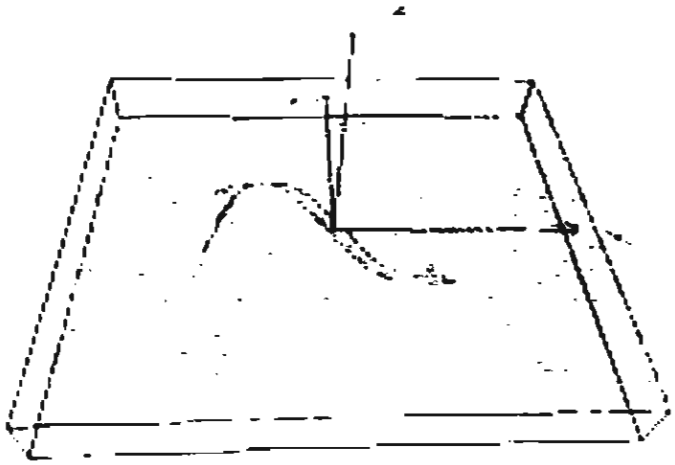

Figure A8. The graphical G-Code of the custom tool path

Figure A9. Finished custom tool path

EXPERIMENT 4. Calibration of simultaneous rotation of A and B axis. A bell shaped surface.

The experiment introduces a convex parabolic surface shown. A and B axis are rotating simultaneously ranging from 0 to 360 and from 0 to -90 respectively. Fig A10a shows the required surface, Fig A10b shows the corresponding graphs of the rotation angles A. The A-angle requires an adjustment shown in Fig A10c. Note that the angle adjustment is required to eliminate sharp variations of the rotation angles near 0 or 360^0. For instance if A changes from 5 to 355 the procedure replaces 355 by –5 etc. Note that although the general case requires an adjustment of B as well, the case of a bell shaped surface implies that B $\in$[-90,0]. Therefore the adjustment is not required (see Fig A11).

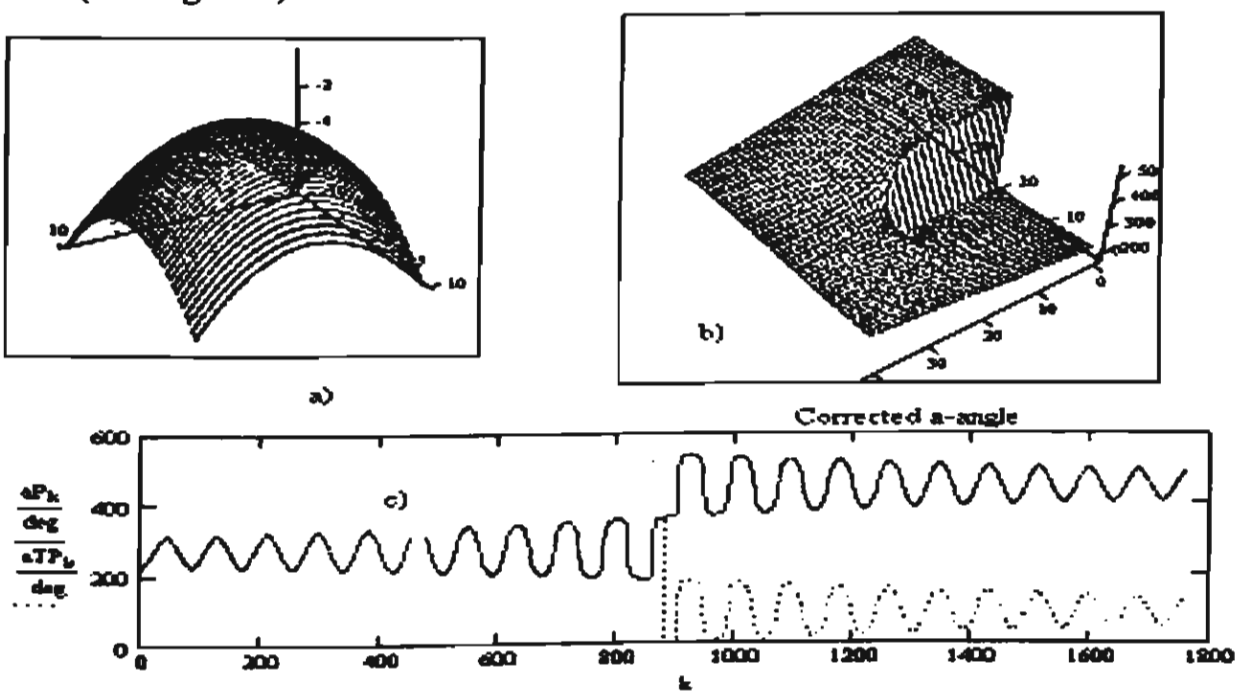

Figure A10: the bell surface

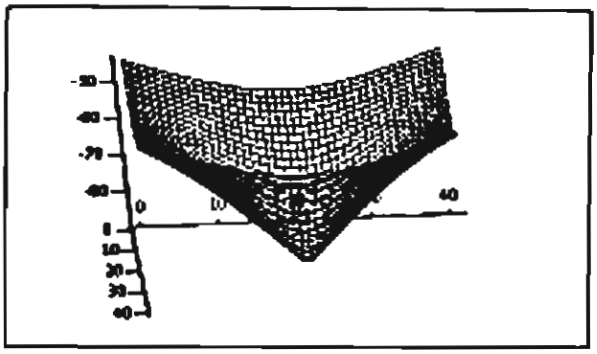

Figure A11. The B angle of the bell surface

Figure A12 displays the corresponding zigzag tool path produced by the virtual machine. Figure A13 displays graphical representation of the G-Code. Figure A14 shows the final cut of the bell surface.

Workpiece 60x60mm, Tool size 10mm
Surface quality: Excellent.
Overcut: NO, Undercut NO.

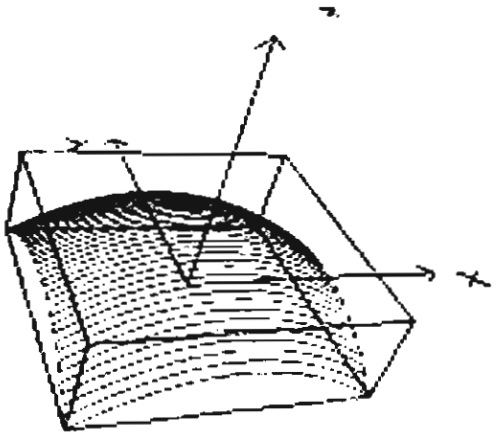

Figure A12. The graphical CL-Points of the bell surface

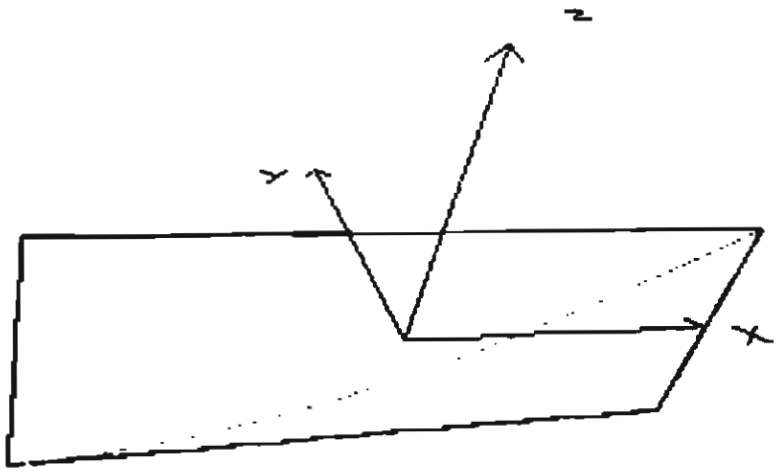

Figure A13. The graphical G-Code of the bell surface

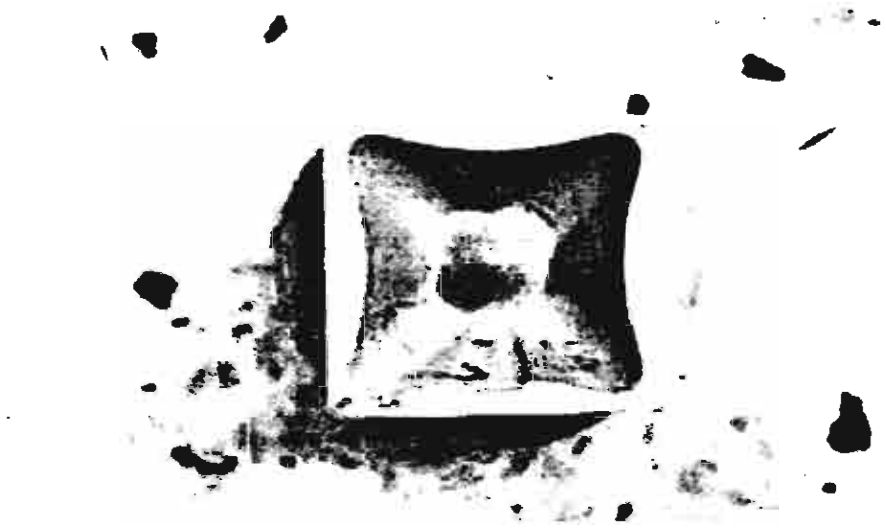

Figure A14. Finished bell surface

EXPERIMENT 5.Calibration of the tool size

Cutting a parametric two bell surface allows for an analysis of an overcut in the case of a convex-concave surface. The experiment displays saddle type regions may lead to substantial milling errors. The first type of the error is due to the large size of the tool. The second type is due to the removal of the material by a side of the tool. (The tool removes the material from the first bell while cutting the second bell (see Fig A15).

Note that the two-bell case requires a substantial adjustment of the angle shown in Fig A16. Figure A17 displays the zigzag tool path for the parametric surface produced by the virtual machine. Figure A18 displays a graphical representation of the G-Code that guides the machine. The real surface is shown in Figure A19.

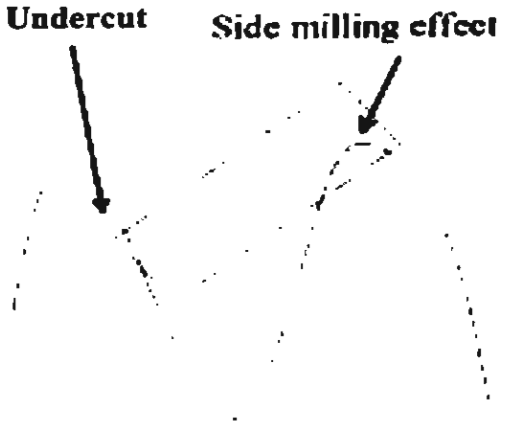

Figure A15. Undercut and the side milling effect