MOTION COMMAND GENERATION FOR MULTI-AXIS MACHINING

By

ROBERT V FLEISIG. BASc. MEng, PEng

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

MOTION COMMAND GENERATION FOR MULTI-AXIS MACHINING

DOCTOR OF PHILOSOPHY (2000) McMaster University
(Mechanical Engineering) Hamilton. Ontario

TITLE: Motion Command Generation for Multi-axis Machining

AUTHOR: Robert V Fleisig, BASc (University of Waterloo).
MEng (McMaster University). PEng

SUPERVISOR: Associate Professor Allan D Spence

NUMBER OF PAGES: xxviii. 198

Abstract

In current industrial practice, the transformation of Computer-Aided Geometric Design surfaces into Computer Numerical Control (CNC) machine tool axis commands is performed with the aid of Computer-Aided Manufacturing software and a closed CNC machine tool controller. The advent of new technologies such as Open Architecture Control has enabled the rethinking of motion command generation.

This thesis describes a five-axis motion command generation architecture and algorithms in which a parameterized tool-path is interpolated off-line and the inverse kinematics mapping is performed in real-time, on the CNC controller. This architecture eliminates the need for time consuming repost-processing of the tool-path in the event of kinematic changes and additionally introduces the benefits of parametric splines with controlled feedrate.

To deterministically attain a near constant feedrate tool-path, near arc-length parameterized splines are prepared off-line. The C^2 position spline which is near arc-length parameterized improves on the previously reported research. The orientation unit vectors are interpolated with a C^2 spherical Bézier spline. These two splines are then synchronized by means of a monotonic reparameterization spline. This results in reduced effective feedrate oscillation. The interpolated tool-path and axis commands

are demonstrated to be smooth and C^2 continuous. To cope with actuator saturation, a feedrate interpolation algorithm is developed which ensures C^2 continuity but allows the feedrate to be adjusted as needed. The developed algorithms were simulated for two tool-paths and two five-axis machines tools. A test part was cut to demonstrate geometric correctness.

To
Barbara
and
Jacqueline

Acknowledgements

I am grateful to the many individuals who have contributed to my university education in mechanical and manufacturing engineering over the past eleven years. To the faculty, staff and students at the University of Waterloo's Automation and Control Group in the Department of Mechanical Engineering, Professor Sanjeev Bedi, Paul Hoskins, Professor Fathy M Ismail, and Robert Wagner, I am indebted for the opportunity and help they freely gave in using their five-axis CNC machine tool affectionately known as the *Rambaudi*. Enough thanks cannot be given for the unfailing support and patience of the departmental administrative and technical staff, Betty-Anne Bedell-Ryc, Rebecca Clifford, Marsha Duncan, Ron Lodewyks, Jane Mah, Jim McLaren, Dave Schick, Joe Verhaeghe and Allyson Wenzowski. I would like to acknowledge the support and valuable criticism of both my PhD supervisory and examination committees: Professor Sanjeev Bedi, Professor David W Capson, Professor Ahmed Ghobarah, Assistant Professor Emil Sekerinski, Associate Professor John C Wilson, Professor Mohamed A Elbestawi and my thesis supervisor Associate Professor Allan D Spence. In particular, Dr Elbestawi has taught me by example what it is to be a professional engineer as well as a engineering scientist and teacher with a passion for his work. Dr Spence has been both encouraging and supportive in my

choice of research topic. His guidance both as a thesis supervisor and friend cannot be overvalued.

For the tremendous support of my studies from the people at e-Manufacturing Networks Inc., especially Tom Gaasenbeeck and Dr Stephen Lane-Smith, I shall ever be thankful.

I wish to acknowledge the generous financial support made by the Natural Sciences and Engineering Research Council of Canada (NSERC) for the NSERC Postgraduate Scholarship that has helped make this research possible. As well, I would like to thank the Ontario Ministry of Education and Training for an Ontario Graduate Scholarship and McMaster University for an Ontario Graduate Scholarship Science & Engineering.

Research and study is always more stimulating when in the company of friends and colleagues. Among my friends and colleagues I would like to acknowledge the help and support of Dr Farid Abrari, Sarmad Aziz, Trevor Bailey, Dr Nael Barakat, Fadi Benjamin, Alex Chan, Paul Clayton, Derek Jenkins, Peter Saturley, Dr Richard Teltz and Dr Stephen Veldhuis.

To my wife, Barbara, I owe more than can be written on this page. She is not only my partner but also the mother of our daughter, Jacqueline, who has brought great joy into our lives.

Toronto, Ontario ROBERT V FLEISIG

December 2000

Kolo sreće se okreće.
Tko bi doli sad je gori.
Tko bi gori sad je doli

As the wheel of fortune turns.
Who is down now is up.
Who is up now is down

Ivan Gundulić, *Osman*

Contents

ABSTRACT v

ACKNOWLEDGEMENTS ix

LIST OF FIGURES xvii

LIST OF TABLES xxiii

NOMENCLATURE xxv

CHAPTER 1 INTRODUCTION 1

1–1 Product development cycle 3

1–1–1 Design 3

1–1–2 Process planning 5

1–1–3 Process automation 7

1–2 Thesis outline 8

CHAPTER 2 LITERATURE REVIEW 11

2-1 Mathematical sculptured surface representation 12

2-2 Metal cutting mechanics 13

2-3 Tool-path generation 17

2-4 Open architecture control 20

2-5 Command generation 21

2-6 Summary 29

CHAPTER 3 MOTION COMMAND GENERATION ARCHITECTURE 31

3-1 Objective 31

3-2 Constraints 33

3-3 Kinematics of five-axis machines tools 34

3-4 Architectures 38

3-5 Summary 40

CHAPTER 4 TOOL-PART GEOMETRY AND KINEMATICS 41

4-1 Tool-path representation 41

4-2 Tool displacement representation 44

4-2-1 Homogeneous transformation matrices 45

4-2-2 Euler angles 47

4-2-3 Taït-Bryan/Cardanic angles 50

4-2-4 Cayley-Klein parameters 50

4-2-5 Rodriguez-Hamilton parameters 51

4-2-6 Vector-parameters/angle-axis/Euler angle and axis 52

4-2-7 Euler parameters/unit quaternions 53

4-2-8 Dual unit quaternions 55

4-2-9 CL-data discrete five-axis tool-path representation 57

4-3 Summary 57

CHAPTER 5 TOOL-PATH INTERPOLATION 59

5-1 Parametric interpolants 59

5-1-1 Smoothness 59

5-1-2 Continuity 60

5-1-3 Parameterization 61

5-2 Selection of tool displacement vectors 63

5-3 Position interpolation 65

5-3-1 The near arc-length parameterized quintic spline 65

5-3-2 The cubic spline 67

5-3-3 Obtaining the tangent and curvatures for the quintic spline . . 71

5-3-4 Determination of segment range 72

5-3-5 Interpolation algorithm 74

5-3-6 Improvement of arc-length parameterization 74

5-4 Orientation interpolation 79

5-4-1 Spherical Bézier spline 79

5-4-2 Interpolating the orientation spline 82

5-4-3 Fitting the quintic spline 82

5-4-4 Parameterization improvement 84

5-5 Orientation reparameterization 85

5-6 Feedrate interpolation 90

5-7 Summary 94

CHAPTER 6 IMPLEMENTATION **95**

6–1 Off-line implementation 95

6–2 Real-time implementation 98

6–2–1 Inverse kinematics 99

6–3 Execution time 102

CHAPTER 7 SIMULATIONS AND EXPERIMENTS **105**

7–1 Side milling example 106

7–2 Ballnose milling example 109

7–3 Summary 110

CHAPTER 8 CONCLUSIONS **141**

APPENDICES **145**

A Derivatives of the Spherical Bézier Curve and Spline 145

B Derivatives of the Feedrate Spline 151

C Example Tool-path Data 153

D Derivatives of Interpolated Tool-path and Axis Commands 157

E Derivatives of Machine Tool Inverse Kinematics 159

F Source Code for Real-time Tool-path Sampling 163

REFERENCES 183

List of Figures

1–1 Typical product development cycle for sculptured surface parts. 4

1–2 Modelling operations in a B-rep modeller. 5

1–3 Feature-based process planning. 6

1–4 Geometry information processing in tool-path planning. 7

2–1 Example of a Bézier curve with control points and control polygons. 12

2–2 Orthogonal metal cutting. 14

2–3 End milling operation. 15

2–4 Downmilling process geometry. 16

2–5 Effective feedrate in five-axis milling. 18

2–6 Linear interpolation of a general spatial curve. 23

2–7 Conventional five-axis machining motion command generation architecture. 27

2–8 Five-axis machining motion command generation architecture proposed by Lin and Koren [61]. 28

3–1 A tilting-rotary table five-axis vertical machining centre. 35

3–2 General machine tool kinematics..... 36
3–3 Example of axis command interpolation mapped back into tool-path space..... 39

4–1 Drive and part surface..... 42
4–2 Cutter-Contact point..... 43
4–3 Tool. tool tip position vector and tool axis unit vector..... 45
4–4 The rotations defining the Euler angles..... 49
4–5 Cartesian and spherical Bézier curves..... 55

5–1 Demonstration of the real-time arc-length parameterization algorithm... 62
5–2 Proposed command generation architecture..... 64
5–3 Chordal deviation of linear interpolation..... 64
5–4 Position spline conventions..... 67
5–5 Comparison of the parameterization error of the proposed position spline algorithm and the algorithm proposed by Wright *et al*..... 76
5–6 Position spline interpolation algorithm..... 77
5–7 Comparison of curve speed for curves with and without auxiliary points. 78
5–8 Orientation spline conventions..... 81
5–9 Orientation spline interpolation algorithm..... 88
5–10 Comparison of angular velocity and acceleration of naturally and near arc-length parameterized spherical C^2 splines..... 89

6–1 Side milling example..... 96
6–2 Ballnose milling example..... 97
6–3 A tilting-rotary table five-axis vertical machining centre..... 99

7-1 Plot of simulated (a) position and (b) orientation curve components interpolating the side milling tool-path. 111

7-2 Plot of parameterization error of the (a) position and (b) orientation curves for the side milling tool-path..... 112

7-3 Plot of (a) reparameterization curve, (b) first and (c) second derivatives interpolating the side milling tool-path. 113

7-4 Plot of simulated (a) feedrate, (b) derivative and (c) FFT for side milling tool-path..... 114

7-5 Plot of simulated (a) angular speed, (b) derivative and (c) FFT for side milling tool-path. 115

7-6 Plot of simulated (a) position curve components, (b) first and (c) second derivatives with respect to time interpolating the side milling tool-path at the feedrate 400 mm min^{-1}..... 116

7-7 Plot of simulated (a) orientation curve components, (b) first and (c) second derivatives interpolating the side milling tool-path at the feedrate 400 mm min^{-1}..... 117

7-8 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and constant feedrate..... 118

7-9 Plot of simulated (a) rotary axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and constant feedrate..... 119

7-10 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the side milling tool-path and constant feedrate..... 120

7-11 Plot of simulated (a) rotary axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the side milling tool-path and constant feedrate. 121
7-12 Plot of simulated (a) feedrate and (b) angular velocity for side milling tool-path with limited axis velocity. 122
7-13 Plot of simulated (a) linear axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and limited axis velocity. 123
7-14 Plot of simulated (a) rotary axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and limited axis velocity. 124
7-15 Photographs of side milling example. 125
7-16 Photograph of a gantry type CMM. 126
7-17 Photograph of a side milling tool-path part being measure with a scanning laser probe on a CMM. 127
7-18 Colour plot of dimensional error. 128
7-19 Plot of simulated (a) position and (b) orientation curve components interpolating the ballnose milling tool-path. 129
7-20 Plot of (a) reparameterization curve, (b) first and (c) second derivatives interpolating the ballnose milling tool-path. 130
7-21 Plot of parameterization error of the (a) position and (b) orientation curves for the ballnose milling tool-path. 131
7-22 Plot of simulated (a) feedrate, (b) derivative and (c) FFT for ballnose milling tool-path. 132

7-23 Plot of simulated (a) angular speed. (b) derivative and (c) FFT for ballnose milling tool-path..... 133

7-24 Plot of simulated (a) position curve components. (b) first and (c) second derivatives interpolating the ballnose milling tool-path at the feedrate 400 mm min^{-1}..... 134

7-25 Plot of simulated (a) orientation curve components. (b) first and (c) second derivatives interpolating the ballnose milling tool-path at the feedrate 400 mm min^{-1}..... 135

7-26 Plot of simulated (a) linear axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the ballnose milling tool-path and constant feedrate..... 136

7-27 Plot of simulated (a) rotary axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the ballnose milling tool-path and constant feedrate..... 137

7-28 Plot of simulated (a) linear axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the ballnose milling tool-path and constant feedrate..... 138

7-29 Plot of simulated (a) rotary axes. (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the ballnose milling tool-path and constant feedrate..... 139

List of Tables

3–1 Possible arrangements of rotary and translational axes for five-axis machine tools with three translational and two rotary axes [83]. 37

5–1 Curve definition of Fig. 5–1..... 63

5–2 Interpolatory points for Fig. 5–7..... 79

C–1 Interpolatory data for the example peripheral milling tool-path..... 153

C–2 Interpolatory data for the example ballnose milling tool-path..... 154

Nomenclature

A	Tool transformation
$\hat{a}_0$, $\hat{a}_1$	Dual quaternion coefficients
a, b, c	Rotary axes of a five-axis machine tool
$\mathbf{B}$	Spherical interpolation curve
B	Tool-path chain
$\mathbf{c}$	Position polynomial coefficient
c_x, c_y, c_z	Intersection point of rotary axes of a tilting-rotary table five-axis machine tool
c_0, c_1, c_2, c_3	Quaternion components
$\mathbf{d}$	Spherical Bézier control point
d_c	Axial depth of cut
d_x, d_y, d_z	Difference between tool tip and intersection point of rotary axes of a tilting-rotary table five-axis machine tool
$\mathbf{e}$	Vector of Euler parameters
e	Distance along cutter axis
e_0, e_1, e_2, e_3	Euler parameters
F	Feedrate spline
f	Feedrate

f_e	Effective feedrate
f_t	Feed per tooth
G	Fixed of machine coordinates system
g	Coefficient of the feedrate spline
$\mathbf{h}$	Vector of Rodriguez-Hamilton parameters
h	First derivative of the reparameterization curve
h_u	Uncut chip thickness
$\mathbf{I}$	Identity matrix
i	Segment index
L	Translation axis coordinate system
l	Length of a segment of the position spline
M	Moving coordinate system
m	Curve sampling index
n	Number of segments in a spline
n_t	Number of teeth
N	Machine tool bed coordinate system
$\mathbf{P}$	Position spline
$\mathbf{p}$	Cartesian position vector
P	Part coordinate system
p	Position vector in vector quaternion form
p_x, p_y, p_z, p_w	Components of the Cartesian position curve vector
$\mathbf{Q}$	Orientation spline
$\mathbf{q}$	Orientation curve spherical unit vector
$\hat{q}$	Dual quaternion
q	Quaternion

q_x, q_y, q_z	Components of the orientation unit vector
$\mathbf{R}$	A 3×3 orthogonal rotation matrix
R	Rotary axis coordinate system
$\mathbf{r}$	Angle-axis vector
r_c	Cutter radius
r_x, r_y, r_z	Components of the angle-axis vector
$\mathbf{S}$	Spherical interpolation curve
$\mathbf{s}$	Axis of rotation
s_1, s_2, s_3	Rotation axis parameters
$\mathbf{T}$	A 4×4 homogenous transformation matrix
$\hat{T}$	Dual quaternion transformation
T	Tool coordinate system
t	Time
U	Feedrate spline as a function of time
u	Position curve parameter
V	Orientation reparameterization spline
v	Orientation curve parameter
v_c	Cutting speed
$\hat{w}$, $\hat{w}_0$, $\hat{w}_1$	Dual quaternion normalizing factor and components
w_c	Width of cut
$\hat{x}$	Normalized dual quaternion
x, y, z	Translational axes of a five-axis machine tool
α	Machine tool axis
Δt	Servo update period
ζ_1, ζ_2, ζ_3	Euler angles

θ	Rotation angle about axis
$\kappa_1, \kappa_2, \kappa_3$	Rodriguez-Hamilton parameters
λ	Range of spherical Bézier segment
μ	Accumulated arc-length of an orienation curve
ϕ_1, ϕ_2, ϕ_3	Taït-Bryan/Cardanic angles
$\psi_1, \psi_2, \psi_3, \psi_4$	Cayley-Klein parameters
ω	Spindle speed

CHAPTER 1
Introduction

Many products are designed with sculptured surfaces to meet either functional or aesthetic requirements. Common applications include

1. aerospace parts such as compressor and turbine blades [25, 28, 29],
2. die and mould tooling,
3. medical implants and
4. optical equipment.

These products require high accuracy parts with complex surfaces which cannot be defined by simple primitives such as planes, spheres, cylinders, etc. Instead, such complex or sculptured surfaces are often defined by Non-Uniform Rational B-Spline (NURBS) curves and surfaces [21, 76] or similar free-form surface representations. Given these requirements, machining is the most common mode of manufacture for die and mould tooling and for small batches of discrete parts.

The focus of this thesis is motion command generation for five-axis milling. The common usage of the term five-axis is apt to create ambiguity. It most commonly

refers to a tool-path for which the tool translates and rotates with respect to the workpiece during a machining operation and also to any machine tool which can perform this motion. In other words, the tool-path has five degrees of freedom: three translation and two rotation. The third orientation degree of freedom of the cutter is redundant because it revolves about its spindle axis. However, not all five-axis machine tools have five degrees of freedom. A hexapod or Stewart platform, a six degree of freedom parallel link mechanism, is one example. Similarly, three-axis refers only to tool-paths and machine tools which allow translation of the tool with respect to the workpiece. Multi-axis denotes three-axis, four-axis or five-axis.

In industrial practice, three-axis Computer Numerical Control (CNC) machining centres are most often employed in the manufacture of sculptured surface parts. Alternatively, five-axis machine tools may be used. Research has demonstrated increased metal removal rate, improved repeatability of one setup, and improved surface finish using five-axis machine tools in comparison to three-axis machines [55]. Despite these advantages, five-axis machines are not common in sculptured surface machining. The primary obstacle in the adoption of five-axis machine tools is the complication in axis command generation due to machine tool kinematics [42]. Three-axis machine tools have a linear correspondence between the tool-path and axis motions whereas this intuitive relationship does not exist with five-axis machine tools. As a result, CNC machine tool operators cannot visualize the axis motions as they would occur for a given tool-path. Consequently, errors in the tool-path or setup are more difficult to discover and therefore must be carefully verified. Furthermore, any change in the cutter length or location of the workpiece requires the time consuming repost-processing of the tool-path.

The complexities of five-axis machine tool kinematics has also led to a lag in the adoption of motion command generation technologies known as *parametric interpolators* for five-axis machining. This thesis combines the advantages of an architecture which simplifies some of the complexities of five-axis machining with the benefits of parametric interpolation.

1–1 PRODUCT DEVELOPMENT CYCLE

The following sections, describing the product development cycle for sculptured surface parts, furnish a context for the remaining chapters. An emphasis is placed on topics relevant to this thesis: the role of computing technology, part and tool-path geometry, and process automation. Fig. 1–1 depicts the typical product development cycle for sculptured surface parts.

1–1–1 DESIGN

Every product begins as a concept intended as an improvement on existing products. At some point, the idea is made more real to the engineer with a drawing or model. In recent years, three dimensional modelling software has become the choice of most mechanical design engineers for communicating their ideas to others. Several powerful Computer-Aided Design (CAD) packages exist on the commercial market for this purpose. They are all built on a common geometric modelling theory: Boundary Representation (B-rep) solid modelling [41]. Within B-rep modellers, volumes are enclosed by bounding surfaces. Several surfaces may be stitched together to bound a given volume. In turn, each surface is bounded by one or more curves, where the curves are bound by two points. This connectivity aspect of B-rep modellers is known as *topology*. Lines, conics, Bézier curves, Basis Spline (B-spline) curves and NURBS

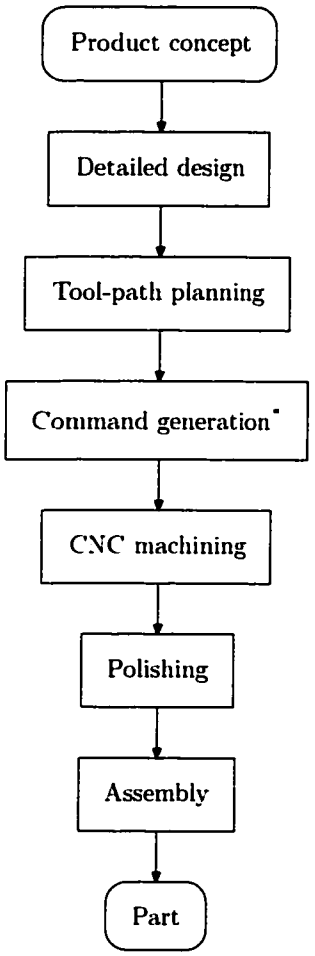


FIGURE 1–1 Typical product development cycle for sculptured surface parts.

curves [21, 76] make up the geometry of the curves and surfaces. Figure 1-2 depicts the relationships between geometric entities found in a B-rep modeller. Models are constructed from any number of either Boolean operations (i.e., union, intersection, subtraction) or surface operations (i.e., lofting, sweeping, interpolation, etc.). Using these operations the engineer lays out a three dimensional geometric model of the part. As an alternative, handmade moulds are sometimes first designed. The mould is then digitized using one of the various scanning technologies to reverse engineer the

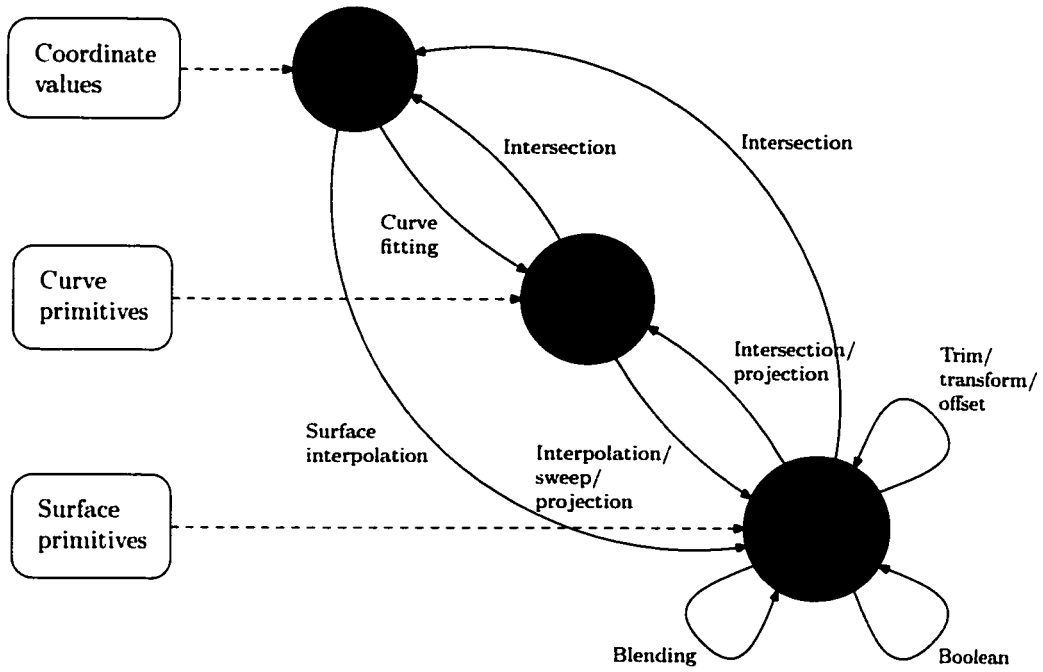


FIGURE 1–2 Modelling operations in a B-rep modeller.

design surfaces. Once digitized, a mathematical surface representation is interpolated through the data and the resultant surface is bound to a solid and can therefore be used in a CAD system.

The design process is very much engineer driven. Automation is limited to only highly mathematical and repetitive tasks. The B-rep model of the part resulting from the design process provides the basis for planning a machining process.

1-1-2 PROCESS PLANNING

In general, process planning for machining is the generation of a set of tool-paths and feedrate schedules that efficiently remove stock material to produce the designed part. With the aid of computer software, the machining process plan is computed off-line either with generative or variant process planning [91]. The complexity of

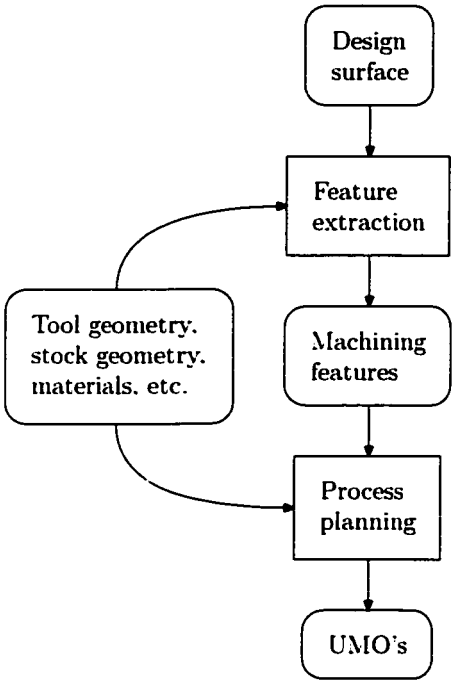


FIGURE 1–3 Feature-based process planning.

the analysis and final process plan are only limited by the sophistication of the user. Computer-Aided Manufacturing (CAM) software and computing resources available. Variant process planning generates plans by modifying extant plans. A more flexible approach is feature-based process planning combined with generative machining where plans are developed as a series of Unit Machining Operations (UMOs). The engineer instructs the software as to which features are to be machined with a given a strategy and tool (see Fig. 1–3). In this way the engineer defines the Unit Machining Operation (UMO)s of the process plan. The UMO is the basic unit of machining specifying a continuous tool-path for a particular area of the part surface using one specific cutting tool. UMOs are often grouped into roughing, semi-finishing, or finishing groups to aid the engineer. Following tool-path generation, the tool-path is verified to ensure it meets the part geometry and metal cutting mechanics constraints

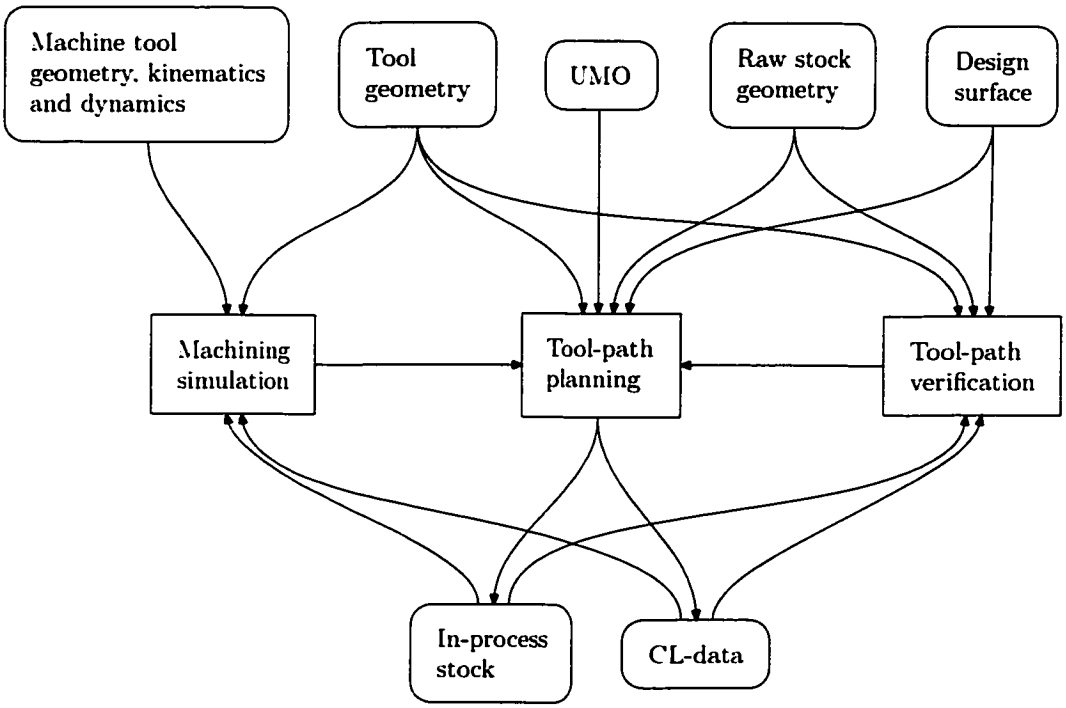


FIGURE 1–4 Geometry information processing in tool-path planning.

to which it is subject (see Fig. 1–4). The end result is Cutter Location Data (CL-data) which is written in the Automatically Programmed Tooling (APT) language [3]. CL-data describes the tool-path and feedrate schedule with respect to the part. That is, the CL-data file is not specific to any machine tool.

1–1–3 PROCESS AUTOMATION

Whereas design and process planning are chiefly human directed and only aided by computing technology, the real-time machining process automation hardware is directed to execute pre-planned part programs with only minor human intervention. The part program is planned off-line where computational time has no bearing on the machining process. However, to obtain high quality surface finish, the part programs

must be executed in *real-time*.

The part program is derived from the process plan using a *post-processor*. The off-line post-processor converts machine tool independent CL-data to a machine specific part program [19, 20]. For five-axis machining, post-processing must additionally solve a set of kinematic equations specific to the targeted machine tool for each point along the tool-path. The part program primarily commands the position in time of the machine tool axes which is computed from the tool-path in the CL-data. The real-time CNC machine tool controllers may have additional functionality to adaptively control the feedrate for more efficient machining.

1–2 THESIS OUTLINE

Motion command generation maps the tool-path given in CL-data form into part program axis motion commands. The key aspect of motion command generation is the computation of the tool-path as a set of curves. The process of computing these curves is known as *interpolation* in Computer-Aided Geometric Design (CAGD). A similar term, *interpolator*, is used in manufacturing to mean sampling of the curves in real-time. The former meaning will be adhered to for the remainder of this thesis.

Although the focus of this thesis is five-axis motion command generation for milling, the results are equally applicable to all multi-axis commanded motion. To this end, Chapter 2 reviews the research relevant to motion command generation. The geometry of the tool-path and therefore axis commands is extracted from the design surface representation. The most common design surface representations, Bézier, B-spline and NURBS are reviewed. Tool-path planning maps design surface geometry into tool-paths and plans feedrate schedules. Therefore, tool-path planning has direct bearing on the information available at the motion command generation stage

and is also covered. Similarly, real-time process automation places strict constraints on the timing and form of the axis commands. In addition, research in metal cutting mechanics is reviewed as it is of direct consequence to both process planning and automation related decisions.

Chapter 3 investigates an alternative command generation architecture to the conventional practice of off-line post-processing. This architecture entails mathematical methods of tool-path and feedrate interpolation. Chapter 4 covers concepts of tool-path kinematics and geometry followed by tool-path and feedrate interpolation algorithms in Chapter 5. Details of implementing the proposed algorithms in real-time are described in Chapter 6. Experiments and simulations are presented in Chapter 7 to verify the proposed architecture and algorithms. Finally, Chapter 8 discusses the results and conclusions.

CHAPTER 2
Literature Review

The objective of motion command generation is to reproduce the planned tool-path on the CNC machine tool constrained to the capabilities of the machine. Motion command generation thus plays the role of bridging the gap between the desired tool-path and the physical reality of motion control. In this chapter research in the product development cycle relevant to motion command generation is reviewed. The origin of geometric information for the axis commands, the design surface, and its mathematical representation is covered in Section 2–1. Section 2–2 reviews work on metal cutting mechanics that is of importance to both tool-path planning and command generation. Open Architecture Control (OAC), briefly reviewed in section 2–4, provides the enabling technology which makes advanced interpolation, process monitoring and process control possible. Tool-path planning determines the form of the tool-path geometry and feedrate schedule. This is reviewed in section 2–3. Next, research directly related to the thesis topic is covered in section 2–5 in some detail, providing the basis for the following chapters.

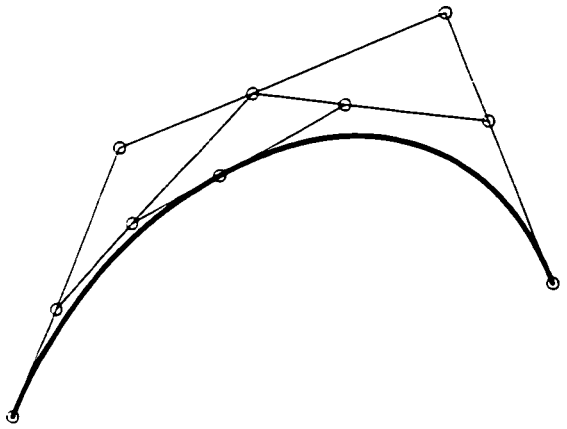


FIGURE 2–1 Example of a Bézier curve with control points and control polygons.

2–1 MATHEMATICAL SCULPTURED SURFACE REPRESENTATION

In the 1950's. CNC machine tools first became available. Although the CNC machine tools were capable of machining sculptured surfaces, the lack of advanced CAM software limited their use. To machine the complex surfaces a method of free-form surface representation is necessary. The most promising avenue was identified as parametric surfaces. This was the origin of the study of Computer-Aided Geometric Design (CAGD). The most significant advances came in the form of Bézier curves and Coons patches. By recursively applying linear interpolation to a set of points a Bézier curve is obtained (see Fig. 2–1). The concept is easily extended to splines and surfaces. Coons surfaces interpolate three or four bounding curves to form a surface. Since its introduction, the Bézier representation has become the foundation for piece-wise polynomial curves and surfaces. NURBS has become the *de facto* standard for representation of design geometry and the data exchange of geometric information is CAD/CAM applications. This success is due to:

1. a unified mathematical basis for the non-rational and rational forms of both the Bézier and B-spline curves and surfaces as well as conic sections. quadric surfaces and free-form entities:

2. algorithms that are fast and numerically stable:

3. an intuitive geometric representation and manipulation for design: and

4. invariance with respect to common transformations (i.e., translation. rotation. scaling. projection. etc.).

Recent research has furthered the generalization of Bézier entities by NURBS with dual Kriging [58]. CAGD concepts and algorithms are the basis of sculptured surface representation and many motion command generation interpolation algorithms.

2–2 METAL CUTTING MECHANICS

In the idealized orthogonal metal cutting model, the passage of the cutter tooth just below the part surface. at the cutting speed v_c. removes a thin slice of metal [87]. The tooth is immersed in the part a distance h_u known as the uncut chip thickness. As the tooth progresses in a direction parallel to the part surface it causes the metal to deform along the shear plane as shown in Fig. 2–2. As a result shear and friction forces are imparted on the rake and flank faces of the cutter tooth.

The orthogonal model shown in Fig. 2–2 is a cross-section taken perpendicular to the tool axis. The geometry of the flat end-mill is shown in Fig. 2–3. A volume of cross-sectional area $d_c w_c$ is removed as the cutter travels at the feedrate f. The cutter teeth revolve about the tool axis at the angular velocity known as the spindle speed ω. The spindle speed is related to the cutting speed by the following approximation

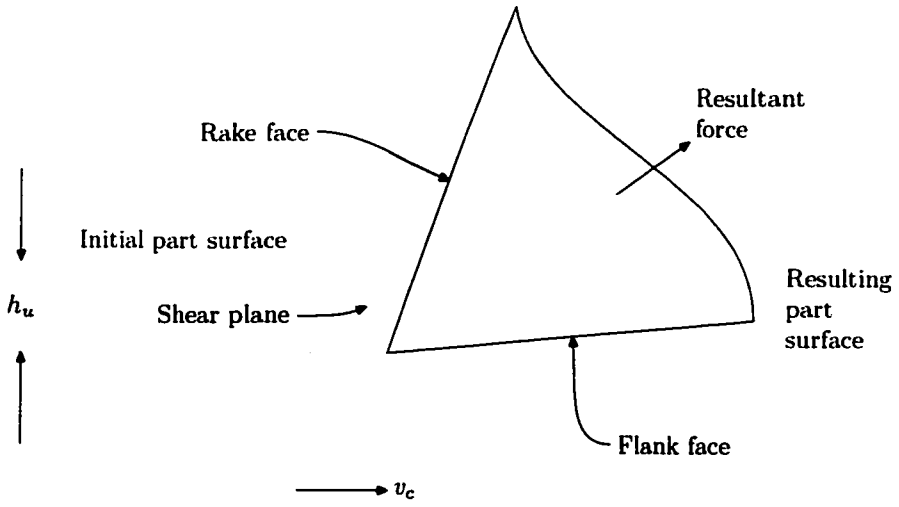


FIGURE 2–2 Orthogonal metal cutting.

[74]:

$$\omega = \frac{v_c}{r_c} \tag{2-1}$$

where r_c is the cutter radius.

Martellotti [67, 68] noted that as the cutter moves in a straight line in a direction normal to the tool axis at the feedrate f the cutter teeth trace a trochoidal motion. However, because the cutting speed is generally much greater than the feedrate, this motion may be reasonably approximated with a circular arc. Hence the simplified Equations 2–1 and

$$f = \frac{f_t n_t \omega}{2\pi} \tag{2-2}$$

are possible where n_t is the number of flutes on the cutter and f_t is the feed per tooth.

Figure 2–4 depicts the circular motion of a four tooth cutter. With every quarter revolution, the cutter advances a distance f_t and a tooth removes the area shown in darker grey. This effectively removes a width of cut w_c as the cutter progresses across

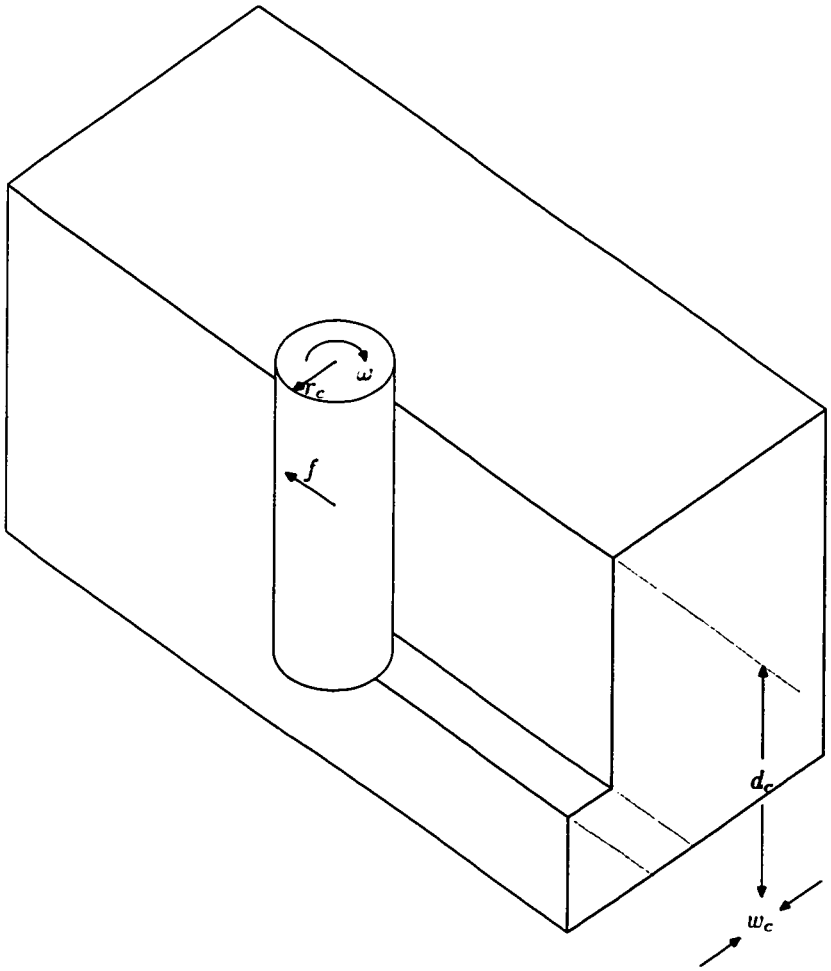


FIGURE 2–3 End milling operation.

the part surface. Assuming that the entire sweep of the tooth is immersed in the workpiece, the maximum uncut chip thickness must be found where the line of tool travel intersects the tooth motion. At the point at which the uncut chip thickness is greatest it is denoted as the feed per tooth f_t. This situation is altered should the tool change its orientation during cutting such as in five-axis milling. In that case the maximum uncut chip thickness will vary with the angular velocity of the cutter axis and along the axis of the tool.

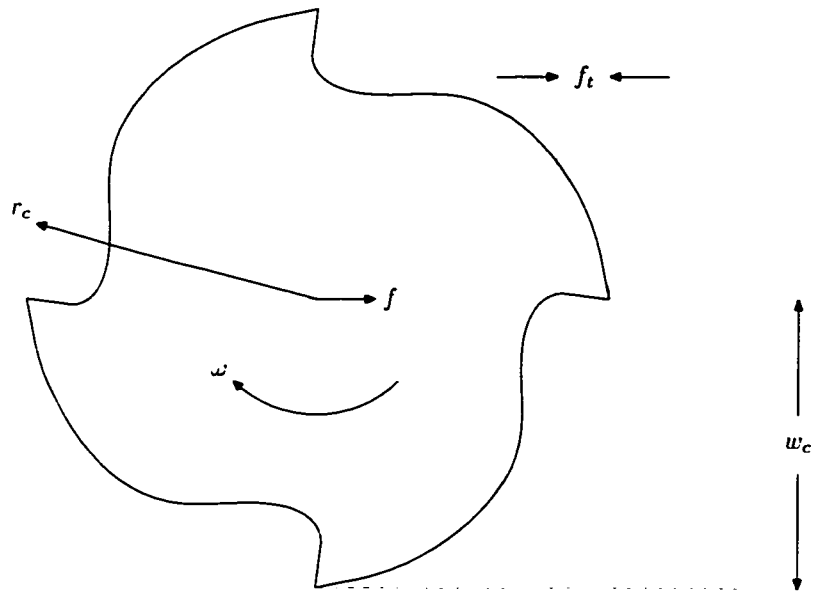


FIGURE 2–4 Downmilling process geometry.

In three-axis milling under conditions of full cutter immersion, the maximum uncut chip thickness is equivalent to the feed per tooth. Furthermore, the maximum uncut chip thickness is measured in the direction of the velocity vector of the cutter $\dot{\mathbf{p}}$ for small feeds. This velocity vector is normal to the cutter axis $\mathbf{q}$ in three-axis machining. It follows that the feedrate is given by

$$f = \|\dot{\mathbf{p}}\|. \tag{2-3}$$

However, when the cutter is tilted forward without changing the direction of motion for example, the direction of motion is no longer normal to the cutter axis. The feedrate which is based on maximum chip thickness is measured in a plane normal

to the cutter axis according to the orthogonal cutting model. Therefore, the feedrate must take into account the change in orientation of the cutter axis. Consequently, the velocity of the cutter is projected into the plane normal to the cutter axis and passing through the tool tip with

$$f_e = \|\dot{\mathbf{p}} - (\dot{\mathbf{p}} \cdot \mathbf{q})\mathbf{q}\| \tag{2-4}$$

where f_e is called the *effective feedrate.*

Suppose that the orientation of the cutter is changing during machining. The effective feedrate is then in a plane normal to the cutter axis and a distance e from the tool tip and the angular velocity $\dot{\mathbf{q}}$ of the cutter must be taken into account. By replacing $\dot{\mathbf{p}}$ in Eq. 2-4 with $\dot{\mathbf{p}} + e\dot{\mathbf{q}}$, and taking into account the angular velocity of the cutter axis, the following equation is obtained taking into account the angular velocity of the cutter axis

$$f_e = \|\dot{\mathbf{p}} + e\dot{\mathbf{q}} - ((\dot{\mathbf{p}} + e\dot{\mathbf{q}}) \cdot \mathbf{q})\mathbf{q}\|. \tag{2-5}$$

See Fig. 2-5.

It is clear from this equation that a larger angular velocity and an increase in the distance between the Cutter-Contact (CC) point and the tool tip may result in an effective feedrate well in excess of the effective feedrate at the cutter tip.

2-3 TOOL-PATH GENERATION

Traditionally, the tool-paths required to machine sculptured surfaces have been programmed using the CL-data form of APT based on geometric data and machining parameters provided by CAD/CAM software systems. The CL-data format specifies a

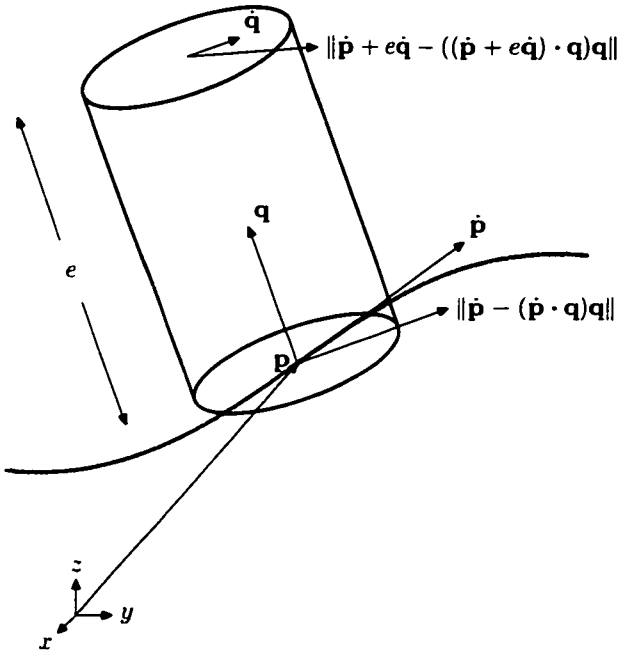


FIGURE 2–5 Effective feedrate in five-axis milling.

tool-path with respect to the part. A further step is required, called *post-processing*, to convert the tool-paths into axis motion commands that the CNC machine tool controller understands. The CNC instructions are generated first by reducing the two dimensional surfaces to one dimensional curves representing a tool-path. Most commercial software still relies on a further step of reducing the data to a set of points which are linearly interpolated by the CNC controller. The use of the spline interpolators allows this step to be circumvented so that the CNC controller works either directly from the surface data or from a tool-path.

The programming of tool-paths for sculptured surfaces typically consists of [10]: roughing, finishing and clean-up. The roughing pass removes most of the unwanted raw stock material while keeping the tool at a safe distance from the part surface. As long as the roughing pass does not contact the design surface, it should not contribute

to the final shape of the part [48]. Finish machining is of salient interest as it is where five-axis machining can be particularly efficient [102]. In finish machining the cutter contacts the design surface at a single point, multiple points, or along some one dimensional geometry. For sculptured surfaces, ballnose milling is the common method of single point machining a surface. More efficient methods using bullnose cutters or flat end mill cutters have been proposed [47, 62, 82, 106]. In five-axis milling, a cutter axis of a flat end mill in side milling traces a *ruled surface*. For design surfaces for which the principal curvatures are very small the ruled surface can be used as an approximation [80].

The strategy routinely employed for planning a single pass, should it be roughing, finishing or clean-up, begins with selection of the cutter. Incumbent in this selection is the choice of tool geometry, number of flutes, material, etc. In many cases the choice of cutter geometry directly depends on the surface geometry. With the cutter chosen, a strategy for removing the excess metal is chosen. Again, this may depend on geometry. It is often efficacious to choose to follow one of the two parametric directions of a parametric surface. These are called *isoparametric* tool-paths. There are many ways in which a sculptured surface can be machined [18].

Every tool-path generated by a CAM system must then be verified to avoid common problems like collision and gouging [9, 45, 46, 73, 114]. Common methods of correction are local adjustments to the tool-path and/or a following clean-up pass.

While a geometric tool-path plan provides an useful nominal guide to commanding the axes of the CNC machine tool, in reality it is strictly an idealization which disregards machine tool dynamics, part dynamics, and cutter dynamics and optimization through feedrate scheduling and spindle speed variation. A tool-path which is geometrically correct is not necessarily physically possible or practicable. Every tool-

path is subject to constraints [92] such as tool chipping, shank breakage, chatter and material-related limitations. This in turn depends on the dynamic properties of the machine, cutter and part. The metal removal process is simulated to both test and adjust for these limitations. Spence [92] simulated pocket milling in order to optimize the feedrate and thus create a non-constant feedrate schedule. Abrari [1] simulated the five-axis machining of a flexible part. Other work includes the dynamic modelling of the machine tool and cutting process for the correction of the axis commands [12]. Correction for tool deflection, geometric error or thermal error are other necessary adjustments of either the tool-path or the axis commands [98, 101, 112]. The corrections directly involved in the geometry of the tool-path take place in real-time on the CNC controller, particularly if they correct the axis commands directly. However, others such as flexible part simulation adjust the tool-path and perform complex operations using a B-rep modeller and Finite Element Analysis (FEA) software off-line.

2–4 OPEN ARCHITECTURE CONTROL

Recent research into OAC for CNC machine tool controllers has been motivated by the inflexible closed nature of commercial machine tool controllers. Often these provide only a rudimentary interface via the primitive G-code language. Researchers [86, 99, 100, 107] have demonstrated that OAC is an enabling technology which provides a framework for implementing advanced manufacturing technologies. These technologies include process control and process monitoring and have been demonstrated to increase productivity and improve competitiveness. Such a framework is necessary for implementing advanced interpolator technologies.

2–5 COMMAND GENERATION

Command generation for CNC machine tools is the transformation of tool-path information generated off-line into axis motion commands in real-time.

In the vast majority of cases, command generation begins with a three-axis, four-axis or five-axis tool-path represented discretely by a set of points. For pure translational motion, $n+1$ Cartesian position vectors

$$\mathbf{p}_i = \begin{pmatrix} p_{x_i} \\ p_{y_i} \\ p_{z_i} \end{pmatrix}, \quad i = 1, \ldots, n+1 \tag{2-6}$$

in E^3 are given.

The earliest interpolators were severely limited by the computer hardware. The primary considerations were transmission bandwidth and computation time. The deterministic demands of real-time control require an uninterrupted flow of reference inputs for each axis control loop. The reference inputs must be supplied at a servo update period of Δt which is typically in the range of 10 ms to 0.1 ms. The only viable solutions are to either transfer the fully computed reference data into real-time from storage or to transfer a condensed form of the reference data to the real-time control loops. In early CNC controllers the first option was impossible due to the very small capacity for data transfer. The only option was to employ a condensed form of the data which for the small cost of real-time computation time could be expanded into the necessary information.

Linear interpolation fulfilled these requirements in the form of a Digital Differential Analyzer (DDA) or reference-pulse interpolator [54]. This hardware based

interpolator linearly interpolates $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ with

$$\mathbf{p} = \mathbf{P}_i(u) = \mathbf{p}_i(1-u) + \mathbf{p}_{i+1}u \tag{2-7}$$

where u is incremented at a constant feedrate f and servo update period Δt:

$$u_m = u_{m-1} + f\Delta t. \tag{2-8}$$

Another scheme, the sampled-data interpolator [54], is more complex and required substantially more memory but is not limited to a maximum velocity determined by the computer like the reference-pulse technique.

Researchers recognized the frequent use of circular arcs in tool-paths. Various circular interpolators based on the reference-pulse and sampled-data interpolator were developed [6, 69, 70, 75] to obviate the need to use many short linear segments to approximate a circular arc and thereby alleviate transmission problems and reproduce the arcs more accurately. Similarly, Koren developed a parabolic interpolator [53].

More recently, detailed analyses of the linear interpolation and its drawbacks have been made by Koren, Shpitalni, Lo, Yang and Kong [56, 89, 110]. These researchers pointed out that approximation of general spatial curves with linear segments involves two competing criteria. The first is the requirement of accuracy or minimum tolerance. With an increase in curvature, an increasing number of linear segments is required. As a consequence, a large number of linear segments requires greater transmission, storage and computational capability. Furthermore, linear interpolation has the following disadvantages (Fig. 2-6):

1. Discontinuities of the first and higher derivatives at the junctions to segments. This leads to large accelerations of the machine tool axes and therefore poor

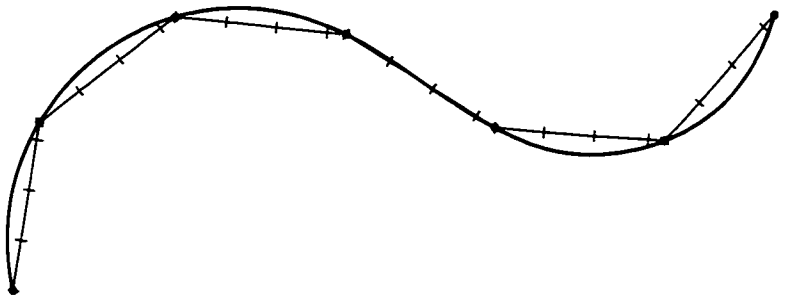


FIGURE 2–6 Linear interpolation of a general spatial curve. The dots denote points $\mathbf{p}_i$ on the tool-path. The points are linearly interpolated with the hash marks spaced apart a distance $f\Delta t$ or less. The cutter travels from left to right.

machining quality.

2. Each linear segment is sampled every servo update period a distance $f\Delta t$ from the previous sample. As a result, the last sample along a given linear segment will likely be a distance less than $f\Delta t$ from the previous sample. The axis motor is forced to decelerate at this last sample and then must re-accelerate for the next segment. The greater the number of segments the greater this problem. For regions of curves with large curvature the average feedrate will be lower than the specified feedrate and therefore the machining time is increased. The additional accelerations on the motors contribute to poor machining quality and wear of the machine tool.

It is therefore surmised that even if computer hardware limitations were non-existent, linear interpolation performs poorly with respect to general spatial curves.

Many earlier researchers recognized some of these limitations and set out to improve on the linear interpolator. Makino [65, 66] developed a special clothoidal curve interpolator for high speed machines. Stadelmann [97] proposed a method of calculating the next sample point along a curve subject to a feedrate profile, constant

acceleration and tangential continuity when the surface is defined by a set of Bézier curves. However, this interpolator does not guarantee that all interpolatory points lie on the curve. Sata [85] proposed a method of cubic Bézier curve generation. The significant advantage is that the amount of data required to represent a curve is greatly reduced in comparison to linear interpolation. A further advantage is first derivative continuity along the entire interpolatory curve. Yeung and Walton [113] proposed an interpolator using only arc-splines and making use of existing work on circular interpolators. Another B-spline interpolator for both curves and surfaces was proposed by Bedi *et al.* [5]. Chen *et al.* [8] also developed a B-spline surface interpolator. The surface interpolators have the advantage of perfect accuracy matching the part surface at each servo update. Interpolated curves inherently approximate the part surface between interpolatory points. Other work has been done in interpolating incrementally, which is useful for real-time interpolation [16, 52] and requires no off-line computation, but cannot obtain optimal smoothness.

This literature on CNC machine tool interpolators contains many specialized solutions. However, a general solution has certain requirements not met by any of the aforementioned work. Some of the interpolators posit a need for curve continuity. It is not clear what level of continuity is necessary but most researchers recognize C^2 continuity as a minimum requirement. This ensures the acceleration is C^0 continuous and therefore jerk is finite although discontinuous. From the analysis of linear interpolation it is clear that the feedrate must not be allowed to vary indiscriminately. A constant feedrate, that is common in many machining operations, will reduce machining time. If a variable feedrate is desired it too must be smooth and at least C^1 continuous to maintain the minimum C^2 continuity of the axis motion commands.

The most significant recent advancement in generalized interpolator research is

based on the machine tool model proposed by Chou and Yang [11. 12. 108. 109]. Huang and Yang [43. 44] developed a general parametric curve interpolator from this theory. Suppose a general parametric curve $\mathbf{P}(u)$. parameterized by the single variable u and interpolated through the set of $\mathbf{p}_i$. is given. A C^2 or greater continuity curve interpolated through the $\mathbf{p}_i$ is readily obtained using the mathematical methods of CAGD [2. 21. 76]. The requirement of constant or smoothly varying feedrate remains to be met. To do so. the static geometric data of the CAD model representing the part surface must be related to the dynamic information requirements of the CNC machine tool control system. Proceed with the derivative of the interpolatory curve with respect to time t

$$\frac{d}{dt}\mathbf{P}(u) = \dot{\mathbf{p}} = \frac{d\mathbf{p}}{du}\frac{du}{dt} = \begin{pmatrix} \frac{dp_x}{du} \\ \frac{dp_y}{du} \\ \frac{dp_z}{du} \end{pmatrix} \frac{du}{dt}. \tag{2-9}$$

The factor $\frac{du}{dt}$ defines the relationship between geometric and time information. To obtain $\frac{du}{dt}$. consider the Euclidean norm of Eq. 2–9

$$f = F(u) = \left\| \frac{d\mathbf{p}}{du} \right\| \frac{du}{dt} = \sqrt{\left(\frac{dp_x}{du}\right)^2 + \left(\frac{dp_y}{du}\right)^2 + \left(\frac{dp_z}{du}\right)^2}\frac{du}{dt} \tag{2-10}$$

where the function $F(u)$ is the feedrate parameterized with the geometric parameter u. Rewriting Eq. 2–10 the following is obtained

$$\frac{du}{dt} = \frac{F(u)}{\left\| \frac{d\mathbf{p}}{du} \right\|}. \tag{2-11}$$

Equation 2–11 is a first order differential equation for which the first order approxi-

mation is

$$u_m = u_{m-1} + \frac{F(u_{m-1})\Delta t}{\sqrt{\left(\frac{dp_x}{du}\right)^2_{m-1} + \left(\frac{dp_y}{du}\right)^2_{m-1} + \left(\frac{dp_z}{du}\right)^2_{m-1}}}. \qquad (2\text{-}12)$$

If a constant feedrate is desired then $F(u) = f$. With Eq. 2–12 a general parametric curve may be sampled in real-time with a constant or variable feedrate. Even if the $F(u) = f$ is chosen the feedrate will not be exactly constant because Eq. 2–12 is an approximation. Additional higher order terms can be employed to improve the accuracy of the prediction of u_m which will come at a penalty of computation time.

Equation 2–12 has been extended by Lin and Koren [60] to free-form surfaces. Lo [63] also developed this research to implicit curves.

An alternative solution is based on the work of Renner [81], and Wang *et al.* [86, 103, 104, 105, 111]. The set of $\mathbf{p}_i$ are interpolated by a curve which has the property of near unit speed or near arc-length parameterization. Hence the curve is sampled with

$$u_m = u_{m-1} + F(u_{m-1})\Delta t. \qquad (2\text{-}13)$$

The advantage is a reduction in the computation time required in real-time but arbitrary parametric curves cannot be used in this method without sampling points from the curve and interpolating them with the near arc-length parameterized curve.

The literature so far cited is strictly limited to three-axis or translational tool-paths. The complications of cutter orientation and machine tools kinematics are not considered. Among the first work to consider these added complications is the five-axis surface interpolator of Lin and Koren [55, 59, 61]. Their work begins by analyzing the conventional practice in five-axis post-processing and machining. Five-axis machining necessitates mapping tool-path points into the joint space of the machine tool. The conventional command generation architecture depicted in Fig. 2–7 calls

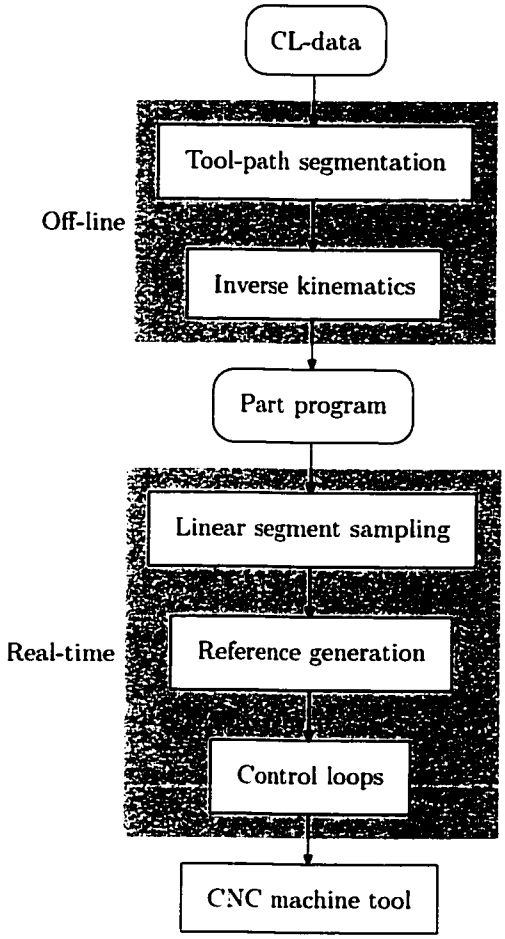


FIGURE 2–7 Conventional five-axis machining motion command generation architecture.

for the linear interpolation of the resultant axes commands. This particular scheme has the following disadvantages:

1. All the disadvantages of linear interpolators previously listed are found in conventional five-axis machining practice.

2. The tool orientation is inaccurately interpolated with respect to the part coordinate system causing position errors and dimensional surface errors.

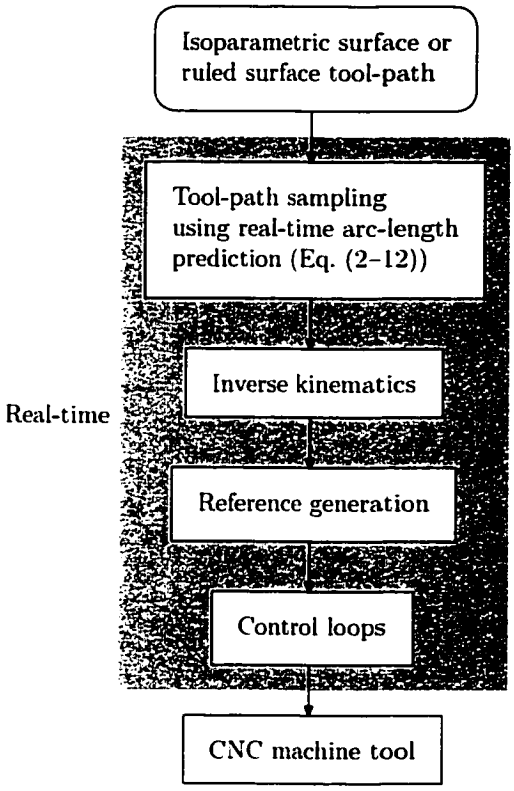


FIGURE 2–8 Five-axis machining motion command generation architecture proposed by Lin and Koren [61].

An alternative architecture shown in Fig. 2-8, utilizing Eq. 2-12 in the real-time control, was proposed to remedy the situation. Thereby a parametric interpolant of the tool-path could be used to accurately and smoothly command the axes. The architecture was demonstrated for ruled surface tool-paths. These are a small set of the possible five-axis tool-paths, and no demonstration or suggestion of a general five-axis parametric interpolatory curve was made.

Most research assumes a constant feedrate, which is often practical in three-axis machining, but becomes less so in five-axis machining where the tool orientation has a direct effect on the force exerted on the cutter. Huang proposed a variable

feedrate $f = F(u)$ in Eq. 2-12 but suggested no way of computing a C^1 continuous feedrate curve. Yang and Chou [108] developed a model for the smooth transition of feedrates from one region of constant feedrate to another. Spence [92] and Lo [64] have demonstrated a need for variable feedrate to obtain maximum feedrate and constant feed at the CC point, respectively.

2-6 SUMMARY

Section 2-5 reviews recent advancements in command generation technology. Three-axis, constant feed interpolation has been the main focus of the research. Some work with respect to five-axis machining in the form of a ruled surface interpolator is the only research related to command generation for five-axis machining. A general solution which translates a tool-path and feedrate schedule into axis commands for multi-axis machining is therefore needed.

CHAPTER 3

Motion Command Generation Architecture

This chapter lays the groundwork for a general motion generation solution. The objective is an architecture which allows arbitrary tool-path generation to be transformed into commands for position control of multi-axis machining subject to the constraints of machine tool dynamics, metal cutting mechanics, machine tool kinematics and information technology.

3–1 OBJECTIVE

The starting point of command generation is the tool-path and feedrate schedule furnished by the process planner. The geometry of the tool-path may be specified in an innumerable variety of forms as is illustrated in Section 2–3. However, all forms are derived from a B-rep model and therefore have a basis in an explicit or implicit continuous mathematical representation. The tool-path is also likely to traverse multiple surfaces of varying types. The feedrate is usually specified as a constant over a section of tool-path. The change in feedrate from one constant region to another is

commanded to be instantaneous.

Transforming the tool-path and feedrate schedule onto CNC machine tool axis commands is subject to practical requirements. CNC machine tools most commonly employ either Direct Current (DC) rotary servo motors, linear motors or stepper motors. Position and velocity of the motors is controlled using closed loop discrete controllers to compensate for the dynamics of the machine tool and metal cutting process. The motors require commands or reference targets to be generated in a quantized, discrete form. The actual command is given as an integer, in units of the Basic Length Unit (BLU) for the machine tool axis. Decimal numbers must therefore be discretized to meet this requirement. The quantization requirement predicates three consequences. Firstly, the axis commands must be delivered to the controller at a fixed interval. This fixed interval, known as the *servo update period* Δt is typically around 1 ms. The second consequence of quantization is the limit on computation time. In the typical period of 1 ms the position controller must either retrieve or compute the reference command and control law. Safety checks, position correction [94], process monitoring and adaptive control are additional burdens on the computation time available during each update. Modern computer technology has radically altered what a controller is capable of in the period of 1 ms since the original DDA circuits. For example, using an Intel Architecture 1 GHz Pentium® III processor [13, 14, 15] makes it possible to perform approximately 20 000 floating point square root operations in the space of one update period[1]. Thirdly, although feedrate and tool-path are separately specified by the tool-path planner, time is intrinsic in the axis command. Therefore the axes must be commanded taking into account not only the geometry of the tool-path but also the feedrate to determine position in time.

[1] $\left(\frac{10^9 \text{ clock cycles}}{\text{second}}\right)\left(\frac{1 \text{ square root operation}}{50 \text{ clock cycles}}\right)\Delta t = 20\,000$ square root operations

The second practical requirement is one of machine tool kinematics. For three-axis machines this is a linear transformation while for five-axis machines a set of non-linear equations must be solved. These equations are specific to each kinematics configuration which includes the machine tool kinematics. tool length and part location in the workspace. Should any of these variables change, the result becomes invalid.

The first interpolators. such as the linear and circular DDA. for three-axis machine tools. were designed to meet the requirements of discretization. quantization and constant feedrate. However. the advent of high speed and five-axis machining has imposed new constraints and made the previously suitable solutions inadequate.

3–2 CONSTRAINTS

Early efforts in machining interpolators aimed to balance real-time computation time. data transmission bandwidth, data storage capability and accuracy. The restrictions of these computational resources meant that accuracy in the commanded tool-path had to be sacrificed. Modern information technology has expanded the computing resources but they are not unlimited. For example, if a long machining job of 24 h is necessary it would require approximately 1648 MB of storage given five axes and a 32 bit integer command per axis[2]. Although by modern standards this number is not large it does create practical difficulties. Transmission of a large quantity of data even over a 100 Mbps network is onerous. Furthermore, should this data be the subject of verification or physical simulation, the task becomes intractable. Consequently, any solution to the command generation problem must consider constraints due to information technology.

More recent efforts studying command generation have recognized the need for

[2] $\left(\frac{5\text{ axes}}{\text{update}}\right)\left(\frac{4\text{ bytes}}{\text{axis}}\right)\left(\frac{1000\text{ updates}}{\text{s}}\right) 24 \times 60 \times 60\text{ s} = 1648\text{ MB}$

"smoothness" in the generated commands. No universal definition or meaning to smoothness is agreed upon. It is however agreed that smoothness in the axis command is necessary to reduce commanded acceleration. This in turn has influence on the contour error and consequently increases accuracy of the part, reduces wear on the machine tool, and may even help reduce chatter [104]. This constraint may in particular circumstances supersede the specified tool-path. For example, in the corner of a pocket being milled a circular section and linear section meet which differ in curvature. To command this exact path requires infinite jerk at the junction of the two sections. Clearly, the machine tool cannot comply, resulting in larger than normal contour error. Smoothing the tool-path in the area of the junction will change the tool-path in the vicinity. However, the final part may demonstrate less error.

Similar to the limitations due to dynamics is the issue of machine tool kinematics. The elements of the Jacobian transformation of the tool-path onto axis commands tend toward infinity near kinematic singularities. Therefore, an axis under certain circumstances may be commanded with excessive velocity. When these velocities exceed the capabilities of an axis large contour errors will result. Maintaining axes velocities within their capabilities is necessary to manufacture acceptable parts.

3–3 KINEMATICS OF FIVE-AXIS MACHINES TOOLS

The advantages and disadvantages of the available motion command generation architectures are direct consequences of machine tool kinematics. In order to more closely examine machine tool kinematics the following discussion will concentrate on the most common kinematic configurations for five-axis machine tools. Figure 3–1 illustrates a typical five-axis machine tool.

Consider the machine tool kinematics illustrated in Fig. 3–2. Using 4×4 homo-

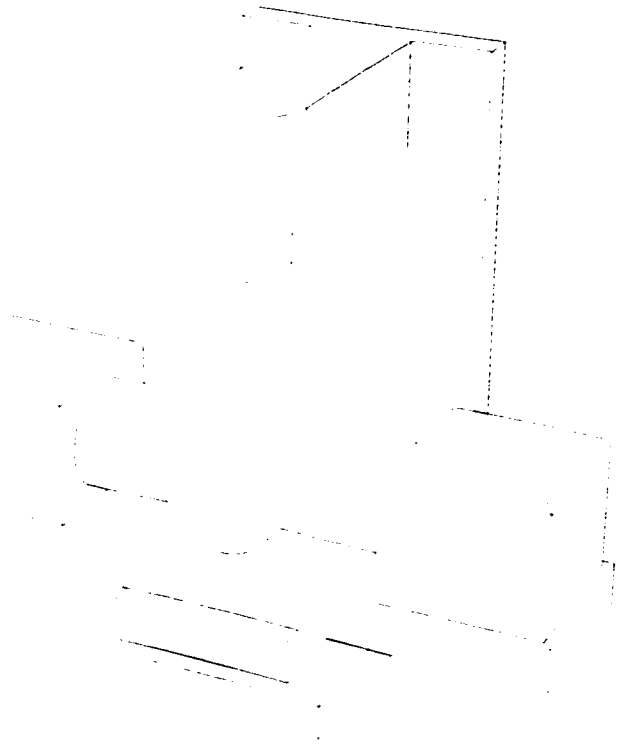


FIGURE 3–1 A tilting-rotary table five-axis vertical machining centre.

geneous transformation matrices the mechanism is described by two open chains of structure equations. The first transformation $A : G \rightarrow T$ relates the machine coordinate system G to the coordinate system T coincident with the tool. The tool chain will encompass zero or more axes plus the spindle which includes the tool length.

The other transformation, $B : G \rightarrow P$, maps the machine coordinate system onto the part P. The part chain includes zero or more axes, the fixture and part. The total number of axes will be between three and six. To ensure the tool is placed correctly on the part

$$A = B \tag{3-1}$$

must be true.

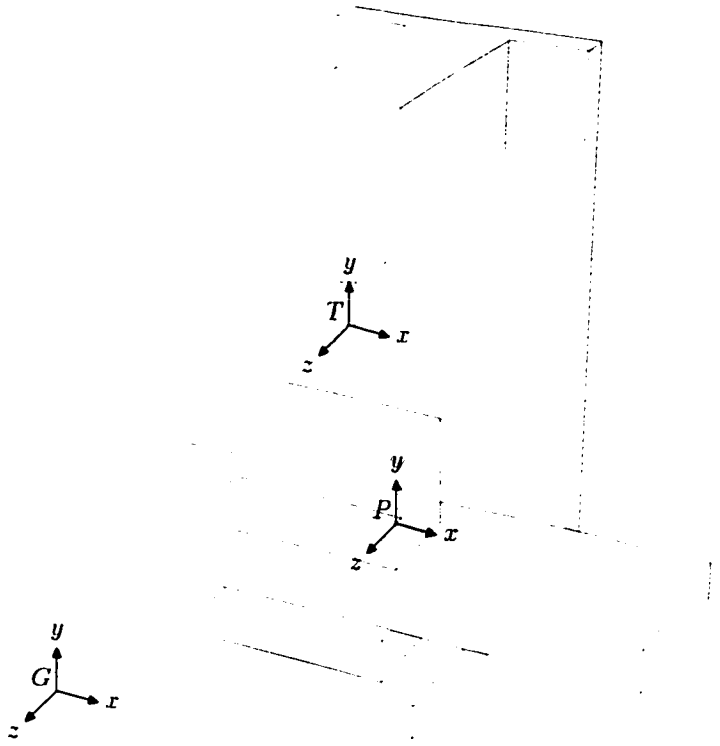


FIGURE 3–2 General machine tool kinematics.

For three-axis machine tools the mapping of the translational axis variables x, y, z in terms of the tool position $\mathbf{p}$ is straightforward. The two mappings A and B are rearranged to give the mapping $A^{-1}B : T \rightarrow P$ [90]. The displacement column of the resulting homogeneous matrix corresponds to the tool tip position vector, $\mathbf{p}$. This relation is strictly linear.

For five-axis machine tools the kinematics involve three translational and two rotary axes. Rüegg [83] suggests a generalized five-axis machine comprising of three translational and four rotary axes. Only two of the four rotary joints are active at any one time. The possible combinations for the arrangement of the rotary and translational axes are listed in Table 3–1. Each row represents the mapping $A^{-1}B : T \rightarrow P$. There are three possible arrangements of the rotary and four for the translational

Tool	Rotary Axis	Rotary Axis	Machine Bed	Translational Axis	Machine Bed	Translational Axis	Machine Bed	Translational Axis	Machine Bed	Basis	Rotary Axis	Rotary Axis	Part
T	R_1	R_2								G	—	—	P
T	—	R_2								G	R_3	—	P
T	—	—								G	R_3	R_4	P
T			N_1	L_1	—	L_2	—	L_3	—	G			P
T			—	L_1	N_2	L_2	—	L_3	—	G			P
T			—	L_1	—	L_2	N_3	L_3	—	G			P
T			—	L_1	—	L_2	—	L_3	N_4	G			P

TABLE 3–1 Possible arrangements of rotary and translational axes for five-axis machine tools with three translational and two rotary axes [83].

axes, which makes a total of twelve. This is the basis for the generalized five-axis machine tool. As an example consider the five-machine tool configuration shown in Fig. 3–1. This illustrated machine tool corresponds to the third and fifth lines of Table 3–1.

Similar to the three-axis case the product of the coordinate system homogeneous matrices gives a matrix in which the displacement column is the tool tip position, $\mathbf{p}$, and the third column is the orientation unit vector, $\mathbf{q}$. The inverse kinematics is obtained by solving for the rotary axis variables followed by the translational axis variables. The rotary variables are embedded inside transcendental functions, hence for the inverse kinematics, inverse transcendental functions must be computed. If there is a small change in the kinematics of the machine tool such as a change in the location of the part or tool length, it becomes necessary to recalculate the entire set of axis commands. Since these calculations must be performed off-line, it causes delays on the machine shop floor.

3–4 ARCHITECTURES

Conventional practice and the available research may be categorized into two command generation architectures whose salient difference is in the order of the computations. The first, used in conventional practice due to its simplicity, performs inverse kinematic calculations on tool-path points followed by interpolation and real-time sampling. The second, proposed by Koren [61], inverts the order of the inverse kinematics and interpolation.

Fig. 2–7 illustrates the command generation architecture used in conventional practice. This architecture takes points from the tool-path and then performs the inverse kinematic calculations on them. The resulting points are then linearly interpolated. These computations are performed off-line line using a post-processor. The resulting part program is transmitted to the real-time controller and is sampled in real-time. The difference revolves around machine tool kinematics.

In conventional industrial practice, the axis commands are interpolated rather than the tool-path. Consequently, any change in the kinematics results in a different tool-path being followed between interpolatory points. This consequence is not suffered when only translation of the tool is required. In translation, $\mathbf{p}$ and α_i are related linearly. However, when the rotary axes must change during machining, the relationship becomes non-linear. As an example, two pairs of orientation vectors are interpolated in Fig. 3–3. In each case the position remains constant, $\mathbf{p} = (0\,0\,0)^T$. The tool orientation is interpolated from $\mathbf{q}_1 = (1\,0\,0)^T$ to $\mathbf{q}_2 = (3^{-1/2}\,3^{-1/2}\,3^{-1/2})^T$ and from $\mathbf{q}_3 = (0\,0\,1)^T$ to $\mathbf{q}_2 = (-3^{-1/2}\,-3^{-1/2}\,-3^{-1/2})^T$. Each unit vector pair is separated by 54.7 ° in part space. The traces marked "spherical interpolation" interpolate the pairs with a geodesic curve on the unit sphere and therefore differ only by a rotation. The traces marked "axis interpolation" interpolate the vectors

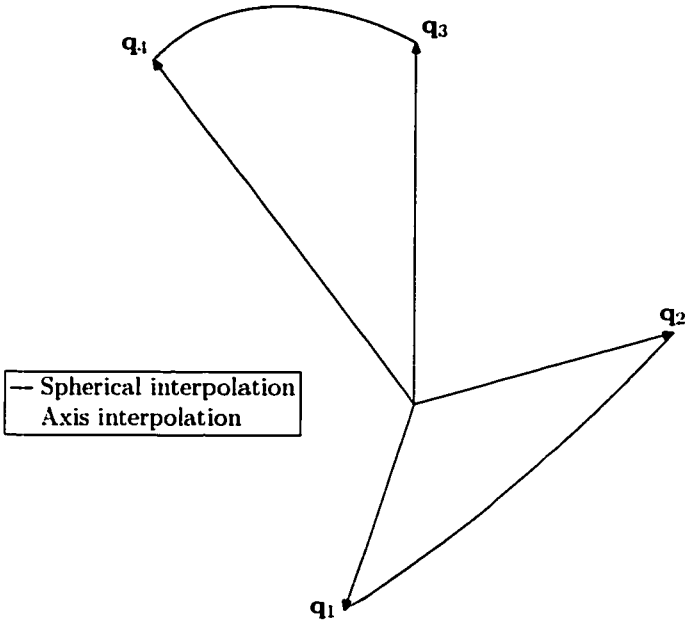


FIGURE 3–3 Example of axis command interpolation mapped back into tool-path space.

pairs after being mapped into joint space. The kinematics of a tilting-rotary five-axis vertical machining centre were used. The traces shown are therefore mapped from joint space back into part space to be compared with the spherical interpolation. The difference is that the two traces interpolated in the joint space do not follow the same path. The two curves are not different by only a rotation. In fact, the maximum separation between the spherical interpolation and axis interpolation curves is 2.7 ° and 38.1 °, respectively. This demonstrates that a pair of tool orientations differing by only a rotation will not follow the same tool-path if interpolated in joint space. This is significant if a feature on a part is repeated on a design at different orientations. Furthermore, this means that a change in tool and therefore tool length, a change in location of the part or a different kinematic configuration will dictate the path when

linearly interpolating in joint space. Clearly, this is reduced if the distance between the unit vectors is smaller but is not eliminated. This is a significant drawback of the conventional method of command generation.

The architecture proposed by Lin and Koren uses parametric curves and performs kinematic computations following the tool-path sampling. This offers significant advantages over the traditional approach. However, it cannot be considered a general interpolator since it is limited to simple parametric surfaces and curves which are followed exactly by the cutter.

3–5 SUMMARY

In this chapter, the foundation for an architecture in which the kinematics computations are performed in real-time was laid. This architecture requires either a part surface from which the tool position and orientation maybe obtained in real-time, or an interpolated tool-path to be sampled by the CNC controller.

CHAPTER 4

Tool-Part Geometry and Kinematics

In order to use the motion command generation architecture proposed in the previous chapter with any tool-path, a tool-path interpolation technique and tool-path point representation are necessary.

4–1 TOOL-PATH REPRESENTATION

The tool-path is represented in one of four ways each of which is a combination of discrete or continuous, and implicit or explicit as shown in this table

	Discrete	Continuous
Implicit	1	3
Explicit	2	4

Implicit means that the tool-path is implied in the mathematics or geometry given to represent it. For example, the CL-data standard utilizes a combination of design and part surfaces illustrated in Fig. 4–1 to specify the trajectory of the tool. The advantage of this representation is correctness. Whereas discrete point representations

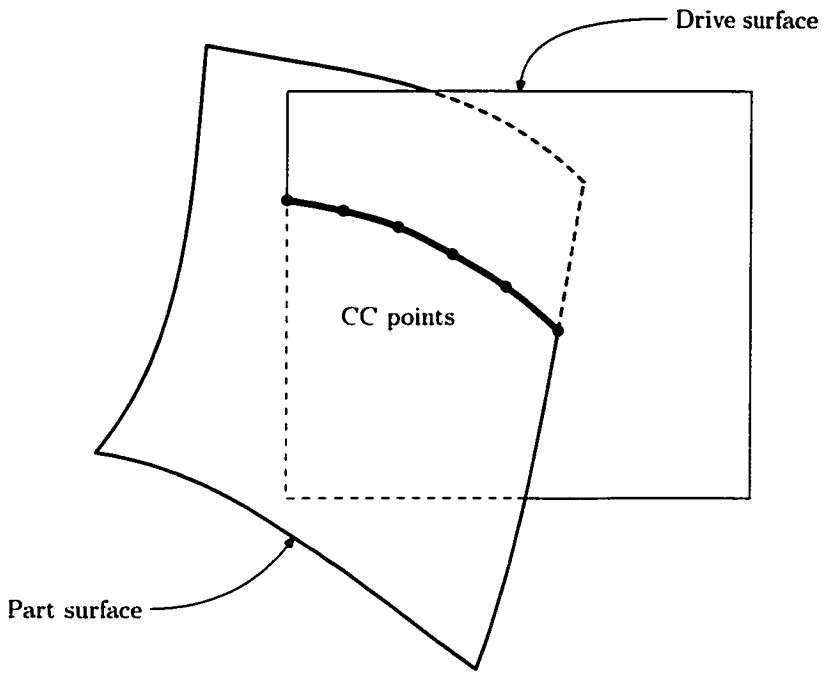


FIGURE 4–1 Drive and part surface.

do not specify the tool-path between interpolatory points this implicit method represents a complete tool-path including derivatives. The difficulty is that the calculation time cannot necessarily be bounded making the real-time implementation uncertain. Another common CL-data format is the CC format indicating where the tool should make contact with the part [30]. The location of the tool tip and orientation of the tool axis are implicit given the design surface and tool shape (see Fig. 4–2).

Ideally, an explicit and continuous tool-path representation would follow the tool-path at all points and would be readily implemented in real-time. However, an explicit tool-path, especially a five-axis tool-path, is rarely available. The one notable exception is a ruled surface tool-path. The command generation architecture proposed by Lin and Koren [59] was originally developed for this type of tool-path.

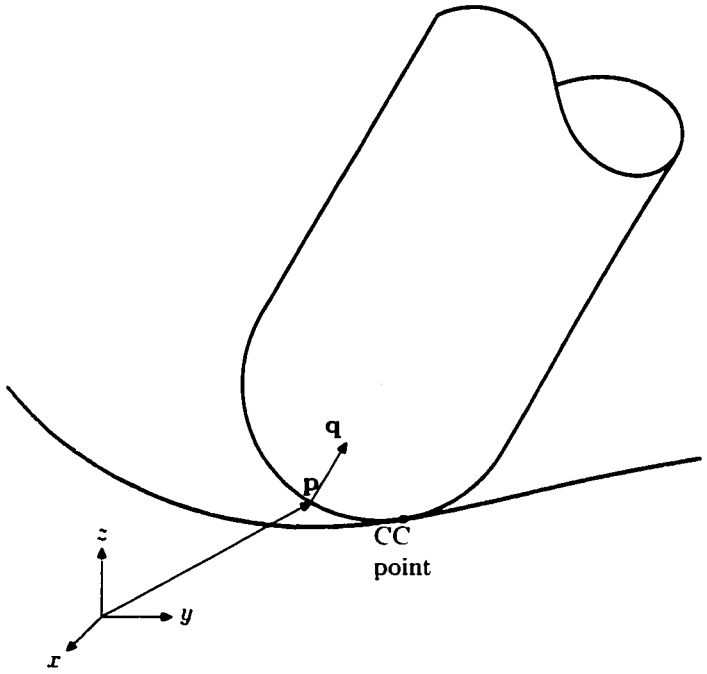


FIGURE 4–2 Cutter-Contact point.

The discrete, explicit representation is the most general of the four above options. Any tool-path, whether three-axis, four-axis or five-axis, can be reduced to this form. Its main disadvantage is the loss of information between the discrete tool displacements. By interpolation new information is inferred between the discrete displacements but it cannot be guaranteed to be exact. Some deviation from the nominal tool-path must result. Linear interpolation is the most straightforward to implement but it performs poorly. For three-axis machining a plethora of solutions exist to overcome the drawback of linear interpolation using explicit and smooth curves whereas for five-axis tool-path interpolation no comprehensive solution exists.

The remainder of this chapter concentrates on finding a discrete and explicit parametric representation of a five degree of freedom tool-path which is suitable for interpolation. This representation is then amenable to all tool-paths.

4–2 TOOL DISPLACEMENT REPRESENTATION

The milling cutter has five degrees of freedom of which three are translational and two rotational. The most obvious method of representing the displacement is separating the position and orientation. Then position is simply represented using Cartesian coordinates whereas one of several possibilities exist for orientation. This is closely examined below in addition to other methods which combine both position and orientation in a single vector-like quantity.

The goal is to obtain a parametric representation of the tool-path from a set of discrete tool-path points given as

$$\mathbf{p} = (p_x\, p_y\, p_z)^T. \quad \mathbf{q} = (q_x\, q_y\, q_z)^T. \quad ||\mathbf{q}|| = 1 \tag{4-1}$$

where $\mathbf{p}$ is the position of the tool tip and $\mathbf{q}$ is the unit direction vector of the tool axis (see Fig. 4–3). To investigate interpolating the position-orientation vector pairs each of the following alternative representations will be considered.

It is in the interpolation of the position and orientation that theoretical kinematics meets the field of CAGD. Whereas the only constraint on choice of tool-path point representation is the ease of analysis and computation from the point of view of theoretical kinematicians, interpolation of these sometimes complex quantities adds additional difficulties. The primary requirement of an interpolatory curve is one of smoothness or continuity. Linear interpolation of Cartesian vectors yields G^0 continuity at least. However, the discontinuity of the derivatives is highly undesirable when commanding machine tool axes. Hence higher levels of continuity are desired which leads to the computation and solution of curves with multiple constraints. Furthermore, machining demands controlled or at least constant feedrate for optimal results.

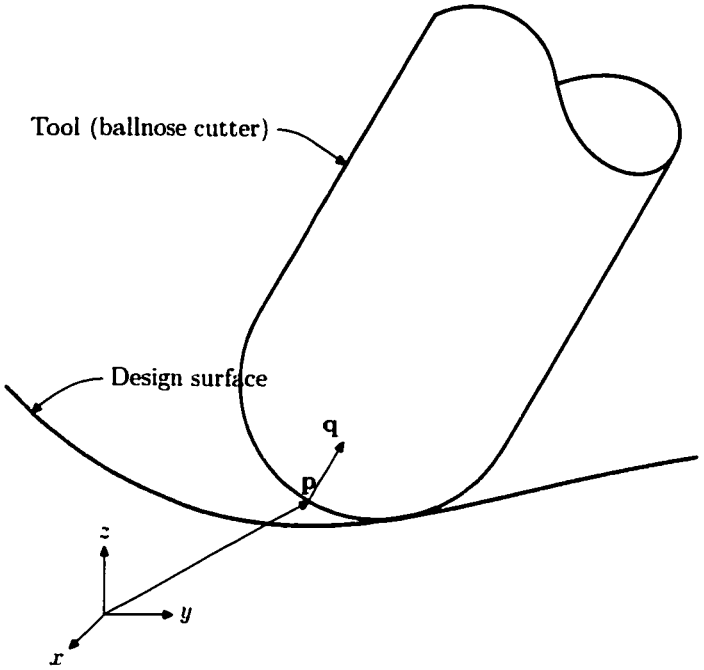


FIGURE 4–3 Tool. tool tip position vector and tool axis unit vector.

This topic will be further developed in Chapter 5. Thus. the following section begins with the most common and intuitive representations and carries onto the more esoteric.

4–2–1 HOMOGENEOUS TRANSFORMATION MATRICES

Consider two coinciding real Euclidean three-spaces. the fixed space G and the moving space M. Both spaces are associated with right-handed Cartesian coordinate frames. The space M is assumed to perform an Euclidean motion with respect to G. Each displacement of G can be described with the help of a real 4×4 Homogeneous

Transformation Matrix (HTM)

$$\mathbf{T} = \left(\begin{array}{ccc|c} & & & p_x \\ & \mathbf{R} & & p_y \\ & & & p_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{4-2}$$

where the 3×3 sub-matrix $\mathbf{R}$ satisfies the orthogonality conditions

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \quad \text{and} \quad \det \mathbf{R} = 1. \tag{4-3}$$

The variable $\mathbf{I}$ denotes the identity matrix. The displacement $\mathbf{p}_2 \in E^3$ of a vector $\mathbf{p}_1$ of the moving space G results from the linear displacement

$$\mathbf{p}_1 \rightarrow \mathbf{p}_2 = \mathbf{T}\mathbf{p}_1. \tag{4-4}$$

The matrix $\mathbf{T}$ represents a spatial transformation.

A continuous one-parameter set of displacements of G defines a tool-path $\mathbf{T} = \mathbf{T}(t)$. If it is assumed that the undisplaced tool resides at the origin of F and the tool axis is aligned with its z-axis then

$$\mathbf{T}(t) = \left(\begin{array}{ccc|c} & & & \\ \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q} & \mathbf{p} \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{4-5}$$

where $\mathbf{p}$ is the location of the tool tip and $\mathbf{q}$ is the orientation of the tool axis.

Jüttler and Wagner [50] developed a rational piecewise interpolant for HTMs

$$\mathbf{T}(t) = \left(\begin{array}{ccc|c} & & & p_x \\ & p_w\mathbf{R} & & p_y \\ & & & p_z \\ \hline 0 & 0 & 0 & p_w\mathbf{e}^T\mathbf{e} \end{array}\right) \tag{4-6}$$

where $\mathbf{e} = (e_0\, e_1\, e_2\, e_3)^T$ are the Euler parameters

$$e_0 = ||\mathbf{e}|| \cos\frac{\theta}{2}, \quad (e_1\, e_2\, e_3)^T = ||\mathbf{e}|| \sin\frac{\theta}{2}\mathbf{s}. \quad \mathbf{s}^T\mathbf{s} = 1. \tag{4-7}$$

where $\mathbf{s}$ is the axis of rotation and θ the angle of rotation. The vector $\mathbf{e}$ formed by Euler's parameters can be identified with the unit quaternion which corresponds to rotation

$$\mathbf{R} = \begin{pmatrix} e_0^2 + e_1^2 - e_2^2 - e_3^2 & 2(e_1e_2 - e_0e_3) & 2(e_1e_3 + e_0e_2) \\ 2(e_1e_2 + e_0e_3) & e_0^2 - e_1^2 + e_2^2 - e_3^2 & 2(e_2e_3 + e_0e_1) \\ 2(e_1e_3 - e_0e_2) & 2(e_2e_3 + e_0e_1) & e_0^2 - e_1^2 - e_2^2 + e_3^2 \end{pmatrix}. \tag{4-8}$$

The eight parameters $e_0(t)$, $e_1(t)$.....$e_3(t)$ and p_x, p_y, p_z, p_w are piecewise polynomials.

This representation has the disadvantage that the orientation and displacement are interpolated simultaneously and therefore the angular velocity of the cutter axis cannot be controlled.

4-2-2 EULER ANGLES

The Euler angles are the parameters of a sequence of three rotations used to define the final orientation of the coordinate system [39]. The three orientations are to some

extent arbitrary. The initial rotation can be taken about any of the coordinate axes. If $\mathbf{R}_1$ is the initial or rotation about z then

$$\mathbf{R}_1 = \begin{pmatrix} \cos\zeta_1 & \sin\zeta_1 & 0 \\ -\sin\zeta_1 & \cos\zeta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4-9}$$

In the subsequent two rotations, the only limitation is that no two successive rotations can be about the same axis. For example,

$$\mathbf{R}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\zeta_2 & \sin\zeta_2 \\ 0 & -\sin\zeta_2 & \cos\zeta_2 \end{pmatrix} \tag{4-10}$$

and

$$\mathbf{R}_3 = \begin{pmatrix} \cos\zeta_3 & \sin\zeta_3 & 0 \\ -\sin\zeta_3 & \cos\zeta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4-11}$$

Fig. 4-4 illustrates the final rotation is defined by

$$\mathbf{R} = \mathbf{R}_3\mathbf{R}_2\mathbf{R}_1. \tag{4-12}$$

A total of twelve conventions is therefore possible in defining Euler angles in a right-handed coordinate system ($\det \mathbf{R} = 1$). The mapping is then

$$\mathbf{q}_1 \rightarrow \mathbf{q}_2(\zeta_1, \zeta_2, \zeta_3) = \mathbf{R}(\zeta_1, \zeta_2, \zeta_3)\mathbf{q}_1. \tag{4-13}$$

Euler angles are the most common method of orientation representation. Three

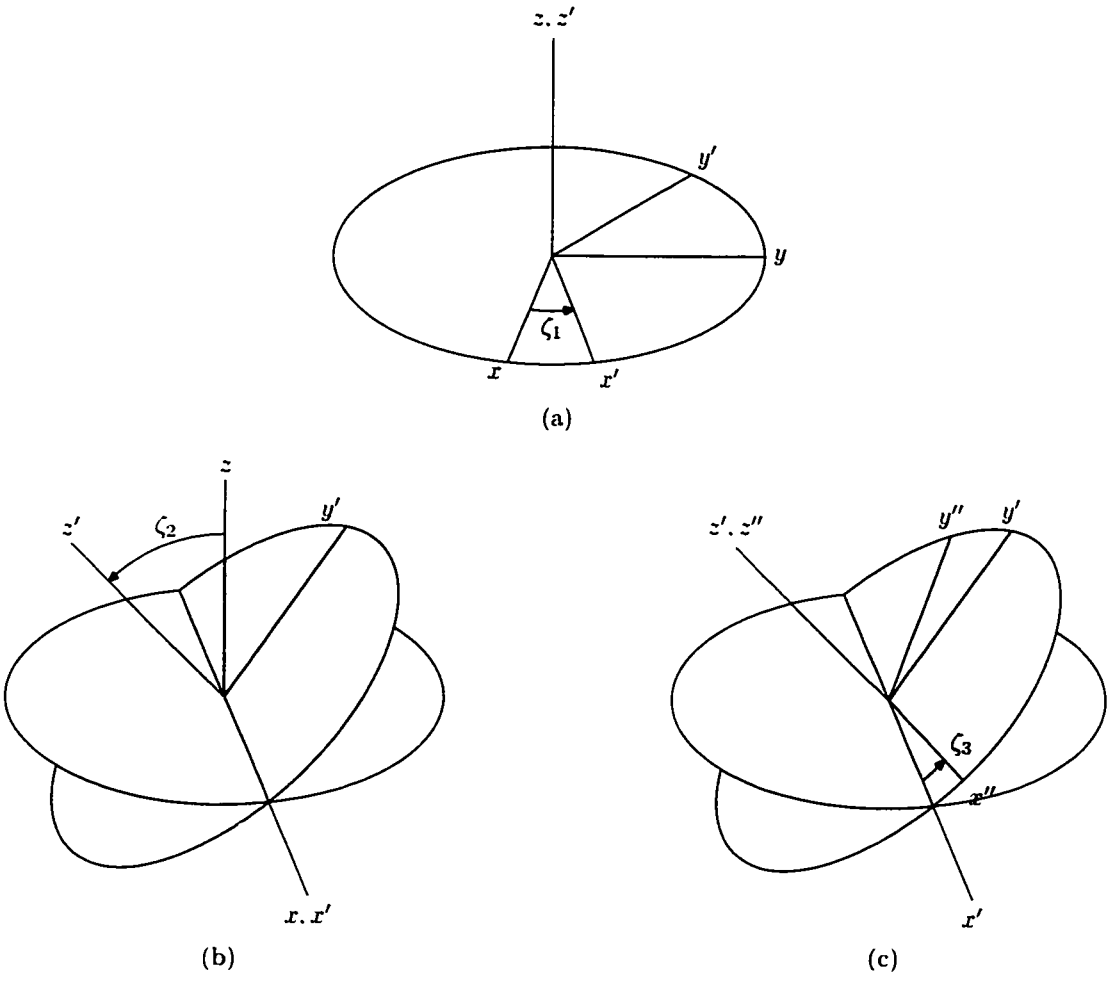


FIGURE 4–4 The rotations defining the Euler angles.

angles represent a given rigid body orientation. For a milling cutter only two are necessary resulting in the equivalent of the azimuth and declination angles used to mark locations on a sphere. Euler angles must be applied in a particular order because rotation transformations do not commute [77]. Euler angles also suffer from non-uniformity. A fixed change in Euler angles does not always give the same amount of rotation change. This means that Euler angles do not describe orientation in a rotation independent way. Euler angles also suffer from *gimbal lock* [57]. At certain

orientations a rotational degree of freedom is lost. Gimbal lock is due to a singularity in the rotations at $\zeta_2 = \pm\pi$. [57].

Although, Euler angles are a simple and intuitive way to represent the orientation its non-uniformity and incongruity makes them unsuitable.

4-2-3 TAÏT-BRYAN/CARDANIC ANGLES

Similar to Euler angles are Taït-Bryan/Cardanic angles. An orientation is carried out by successive rotations [57].

An example rotation matrix is

$$\mathbf{R} = \begin{pmatrix} \mathrm{c}\,\phi_2\,\mathrm{c}\,\phi_3 & -\mathrm{c}\,\phi_2\,\mathrm{s}\,\phi_3 & \mathrm{s}\,\phi_2 \\ \mathrm{s}\,\phi_1\,\mathrm{s}\,\phi_2\,\mathrm{c}\,\phi_3 + \mathrm{c}\,\phi_1\,\mathrm{s}\,\phi_3 & -\mathrm{s}\,\phi_1\,\mathrm{s}\,\phi_2\,\mathrm{s}\,\phi_3 + \mathrm{c}\,\phi_1\,\mathrm{c}\,\phi_3 & -\mathrm{s}\,\phi_1\,\mathrm{c}\,\phi_2 \\ -\mathrm{c}\,\phi_1\,\mathrm{s}\,\phi_2\,\mathrm{c}\,\phi_3 + \mathrm{s}\,\phi_1\,\mathrm{s}\,\phi_3 & \mathrm{c}\,\phi_1\,\mathrm{s}\,\phi_2\,\mathrm{s}\,\phi_3 + \mathrm{s}\,\phi_1\,\mathrm{c}\,\phi_3 & \mathrm{c}\,\phi_1\,\mathrm{c}\,\phi_2 \end{pmatrix} \tag{4-14}$$

where s and c denote the sine and cosine functions, respectively.

As with Euler angles, a singularity can arise. It takes place for $\phi_2 = \pi/2 + k\pi$.

4-2-4 CAYLEY-KLEIN PARAMETERS

The so-called Cayley-Klein parameters are ψ_1, ψ_2, ψ_3, ψ_4. The Euler parameters can be replaced by the Cayley-Klein parameters by substituting,

$$e_0 = \frac{\psi_3 + \psi_2}{2i}, \quad e_1 = \frac{\psi_4 + \psi_1}{2}, \quad e_2 = \frac{\psi_4 - \psi_1}{2i}, \quad e_3 = \frac{\psi_3 - \psi_2}{2}. \tag{4-15}$$

where $i = \sqrt{-1}$, and ψ_1, ψ_2, ψ_3, ψ_4 are restricted so that $\psi_1\psi_4 - \psi_2\psi_3 = 1$. The result is

$$\mathbf{R} = \begin{pmatrix} \dfrac{\psi_1^2 - \psi_3^2 + \psi_4^2 - \psi_2^2}{2} & \dfrac{i(\psi_1^2 + \psi_3^2 - \psi_2^2 - \psi_3^2)}{2} & \psi_2\psi_4 - \psi_1\psi_3 \\ \dfrac{i(\psi_3^2 - \psi_1^2 + \psi_4^2 - \psi_2^2)}{2} & \dfrac{\psi_1^2 + \psi_3^2 + \psi_2^2 + \psi_4^2}{2} & i(\psi_1\psi_3 + \psi_2\psi_4) \\ \psi_3\psi_4 - \psi_1\psi_2 & -i(\psi_1\psi_2 + \psi_3\psi_4) & \psi_1\psi_4 + \psi_2\psi_3 \end{pmatrix}. \quad (4\text{-}16)$$

Cayley-Klein parameters do not exhibit the difficulties related to gimbal lock but it is not clear how they could be interpolated.

4-2-5 RODRIGUEZ-HAMILTON PARAMETERS

Yet another representation are the Rodriguez-Hamilton parameters defined by [57]

$$\kappa_1 = s_1 \tan\frac{\theta}{2}, \quad \kappa_2 = s_2 \tan\frac{\theta}{2}, \quad \kappa_3 = s_3 \tan\frac{\theta}{2} \quad (4\text{-}17)$$

or as a function of Euler parameters:

$$e_0 = \sqrt{1 + \kappa_1^2 + \kappa_2^2 + \kappa_3^2}, \quad e_1 = e_0\kappa_1, \quad e_2 = e_0\kappa_2, \quad e_3 = e_0\kappa_3. \quad (4\text{-}18)$$

Therefore they do not form a set of redundant variables, but present a singularity for $\theta = \pm\pi$. They have the advantage of bringing about rotation in a simple way using the vector $\mathbf{h} = \mathbf{s}\tan(\theta/2)$ where $\mathbf{h}$ is the vector with components κ_1, κ_2 and κ_3 in F.

The rotation matrix can be expressed as a function of these parameters

$$\mathbf{R} = (1+\kappa_1^2+\kappa_2^2+\kappa_3^2)\begin{pmatrix} 1+\kappa_1^2-\kappa_2^2-\kappa_3^2 & 2(\kappa_1\kappa_2-\kappa_3) & 2(\kappa_1\kappa_3+\kappa_2) \\ 2(\kappa_1\kappa_2+\kappa_3) & 1-\kappa_1^2+\kappa_2^2-\kappa_3^2 & 2(\kappa_2\kappa_3-\kappa_1) \\ 2(\kappa_1\kappa_3-\kappa_2) & 2(\kappa_2\kappa_3+\kappa_1) & 1-\kappa_1^2-\kappa_2^2+\kappa_3^2 \end{pmatrix}. \tag{4-19}$$

4-2-6 VECTOR-PARAMETERS/ANGLE-AXIS/EULER ANGLE AND AXIS

Another non-redundant and representative method of defining the rotation is the Euler angle and axis which gives the vector [57]

$$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} s_1\theta \\ s_2\theta \\ s_3\theta \end{pmatrix} \tag{4-20}$$

where the unit vector $(s_1\ s_2\ s_3)^T$ is in the direction of the axis of rotation and θ is the angle of rotation. From this and the axis direction the following can be deduced

$$\theta = \pm\,(r_x^2+r_y^2+r_z^2)^{\frac{1}{2}} \tag{4-21}$$

$$s_1 = \frac{r_x}{\theta} \tag{4-22}$$

$$s_2 = \frac{r_y}{\theta} \tag{4-23}$$

$$s_3 = \frac{r_z}{\theta}. \tag{4-24}$$

Clearly at $\theta = 0$ this definition is indeterminate. This particular method offers no obvious way of interpolation.

4-2-7 EULER PARAMETERS/UNIT QUATERNIONS

A quaternion is a method by which rotations or orientations can be treated in an elegant way [7, 72]. The quaternion q is a complex number

$$q = c_0 + c_1 i + c_2 j + c_3 k \tag{4-25}$$

where i, j, k are complex units following the rules

$$\begin{gathered} 1i = i1 = i, \quad 1j = j1 = j, \quad 1k = k1 = k, \\ i^2 = j^2 = k^2 = -1, \\ jk = -kj = i, \quad ki = -ik = j, \quad ij = -ji = k. \end{gathered} \tag{4-26}$$

Quaternion addition is identical to vector addition but due to Eq. 4-25 quaternion multiplication of q and q' is as follows

$$\begin{aligned} qq' =& (c_0c_0' - c_1c_1' - c_2c_2' - c_3c_3') + (c_0c_1' + c_1c_0' + c_2c_3' - c_3c_2')i \\ &+ (c_0c_2' + c_2c_0' + c_3c_1' - c_1c_3')j + (c_0c_3' + c_3c_0' + c_1c_2' - c_2c_1')k. \end{aligned} \tag{4-27}$$

In the application to kinematics, the unit quaternion $c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1$ conveniently represents a rotation. Written in terms of the rotation angle θ and rotation axis $\mathbf{s} = (s_1\ s_2\ s_3)^T$ this becomes

$$q = \cos\frac{\theta}{2} + s_1 \sin\frac{\theta}{2} i + s_2 \sin\frac{\theta}{2} j + s_3 \sin\frac{\theta}{2} k. \tag{4-28}$$

Similar to the HTM a vector quaternion $\mathbf{p}_1$ is rotated into a vector quaternion $\mathbf{p}_2$

$(\mathbf{p} = p_x i + p_y j + p_z k)$ by

$$\mathbf{p}_2 = q\mathbf{p}_1 q^* \tag{4-29}$$

where $q^* = c_0 - c_1 i - c_2 j - c_3 k$ is the conjugate of q.

Various researchers [4, 32, 34, 36, 38, 51, 77, 79, 88, 95] have dealt successfully with the problem of interpolating quaternion curves. Due to the unit quaternion constraint $c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1$ the interpolated curve must lie on the unit hypersphere. Thus it is easily seen that the equivalent of a linear interpolation on the unit hypersphere is the geodesic curve between the rotations following a great circle. Thus using either quaternion calculus

$$q(t) = q_1(q_1 q_2^*)^t \tag{4-30}$$

or Cartesian calculus

$$q(t) = \frac{q_1 \sin(\theta t) + q_2 \sin(\theta(1-t))}{\sin(\theta)}, \quad \theta = \arccos(q_1 \cdot q_2) \tag{4-31}$$

q_1 and q_2 are interpolated and parameterized with t.

The research has demonstrated interpolated unit quaternion curves using the de Casteljau algorithm [21, 76] with either Eq. 4-30 or Eq. 4-31 to form a spherical Bézier curve as illustrated in Fig. 4-5. The central difficulty with this technique is in the non-linear nature of Eq. 4-30 and Eq. 4-31. They require either transcendental or exponential functions and therefore the common and powerful techniques of NURBS curves is unavailable for interpolating and adjusting the interpolatory curves.

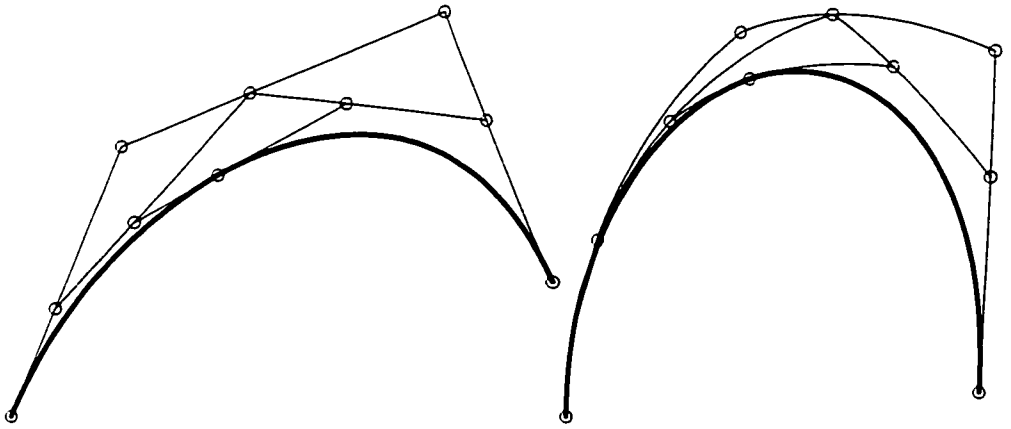


FIGURE 4–5 Cartesian and spherical Bézier curves. The curve (a) is a cubic Cartesian curve and curve (b) is a cubic spherical Bézier curve.

4-2-8 DUAL UNIT QUATERNIONS

The displacement of a coordinate frame M with respect to fixed frame G can be represented by a dual quaternion

$$\hat{q} = q + \epsilon q_0 \tag{4-32}$$

where the dual unit ϵ is defined by $\epsilon^2 = 0$ and the following rules are valid for the dual number $a + \epsilon b$

$$\begin{gathered} 0\epsilon = \epsilon 0 = 0, \quad a\epsilon = \epsilon a, \quad (a_1 + \epsilon b_1) + (a_2 + \epsilon b_2) = a_1 + a_2 + \epsilon(b_1 + b_2), \\ (a_1 + \epsilon b_1)(a_2 + \epsilon b_2) = a_1 a_2 + \epsilon(a_1 b_2 + a_2 b_1), \\ \text{if } a_1 + \epsilon b_1 = a_2 + \epsilon b_2 \text{ then } a_1 = a_2,\ b_1 = b_2. \end{gathered} \tag{4-33}$$

The real part $q = c_0 + c_1 i + c_2 j + c_3 k$ of Eq. 4–32 is defined by the Euler parameters

of the rotation **R** (see Eq. 4–28). The dual part, q_0, is given by the formula:

$$q_0 = \frac{1}{2}pq \tag{4-34}$$

where $p = p_x i + p_y j + p_z k$ is the vector quaternion formed from the translation vector **p**.

If $\hat{q}_1$ defines a rigid body in the moving reference frame M, when $\hat{q}_1$ is measured in the fixed reference frame G its coordinates are given by $\hat{q}_2$ defined by the transformation

$$\hat{q}_1 \rightarrow \hat{q}_2 = \hat{T}\hat{q}_1\hat{T}^* \tag{4-35}$$

where $\hat{T}^*$ is the conjugate of the transformation $\hat{T}$.

Ge and Ravani [37] show that it is possible to linearly interpolate two displacements represented by dual quaternions with

$$q(t) = \hat{w}(t)\hat{x}(t) = \hat{a}_0 + \hat{a}_1 t \tag{4-36}$$

where $\hat{q}(t)$, $\hat{x}(t)$ denote the general and unit-normalized coordinates, respectively, and $\hat{w}(t)$ denotes the normalizing factor. The coefficient vectors $\hat{a}_0$, $\hat{a}_1$ are sought such that

$$\hat{x}(0) = \hat{x}_0, \quad \hat{x}(1) = \hat{x}_1. \tag{4-37}$$

The result is the following unifold linear interpolant:

$$\hat{q}(t) = \hat{w}(t)\hat{x}(t) = (1-t)\hat{w}_0\hat{x}_0 + t\hat{w}_1\hat{x}_1, \quad t \in [0,1] \tag{4-38}$$

where $\hat{w}_0 = \hat{w}(0)$ and $\hat{w}_1 = \hat{w}(1)$ can be arbitrary dual numbers. In practice $\hat{w}_0$ and $\hat{w}_1$ must have positive real parts so that all displacements of the linear motion are

similarly oriented. The normalizing factor $\hat{w}(t)$ is obtained from Eq. 4–38 as

$$\hat{w}(t) = \sqrt{(1-t)^2\hat{w}_0^2 + t^2\hat{w}_1^2 + 2t(1-t)\hat{w}_0\hat{w}_1(\hat{x}_0 \cdot \hat{x}_1)}. \tag{4–39}$$

It is noted that in this linear interpolation the pitch or angular velocity remain constant throughout the screw motion [49. 37].

4–2–9 CL-DATA DISCRETE FIVE-AXIS TOOL-PATH REPRESENTATION

The CL-data standard utilizes a simple five degree of freedom representation of a tool-path. The E^3 vectors $\mathbf{p}$ and $\mathbf{q}$ represent the tool tip position and direction. respectively. where $\|\mathbf{q}\| = 1$. Interpolating the orientation is the same as the problem solved by Eq. 4–30 and Eq. 4–31 for unit quaternion interpolation.

4–3 SUMMARY

Several of the definitions of orientations above are widely employed. However, many of them suffer from singularities such as gimbal lock rendering them less useful for interpolation. Research has shown that the unit quaternion and dual quaternion representations are relatively easily interpolated. These then need to be adapted for the purpose of tool-path interpolation. Dual quaternion curves however offer no way of separately controlling the feedrate of the tool tip and the effective feedrate. If a dual quaternion curve interpolating a tool-path is parameterized for constant feedrate the angular velocity of the tool axis is intrinsic to the interpolated curve. By separately considering the tool tip and tool axis it is possible to control the feedrate of the tool tip while reducing the angular acceleration of the tool axis.

CHAPTER 5
Tool-path Interpolation

In chapter 4 it was shown that promising representations for cutter orientation are the unit quaternion and unit direction vector. Developing a continuous tool-path interpolated through a set of paired position and unit direction vectors is the focus of this chapter [31].

5–1 PARAMETRIC INTERPOLANTS

The aim of tool-path interpolation is to fit a single parameter function in u to the set of paired vectors $\mathbf{p}_i$ and $\mathbf{q}_i$ where $i = 1, \ldots, n+1$. The resultant function must be smooth, at least C^2 continuous, with constant feedrate or smoothly variable feedrate as the application requires. Furthermore, the effective feedrate should also be smooth with minimal oscillation.

5-1-1 SMOOTHNESS

One impetus to using curves to represent tool-paths is the need to reduce acceleration and jerk which leads to improved accuracy, better surface finish and less machine tool wear. The measure of the smoothness of a curve determines its effective-

ness. However, the smoothness of a curve has no strict or accepted meaning. Wang and Yang [105] likened interpolating splines through data points to the behaviour of an elastic beam which forms a smooth curve with minimum energy when small displacements are exerted at points along its length. Srinivasan and Ge [96] extended this concept by measuring the kinetic energy of the curve to determine smoothness. A further and very common measure is the visual or aesthetic metric of smoothness. "Visually pleasing" often appears in research papers as a measure of smoothness for a given example (see [33]).

When solving for a set of curve parameters such as the coefficients of polynomial curves or control points of NURBS curves, the number of unknowns is normally equal to the number of equations. This leaves no need to measure the kinetic energy of the fitted curve since there are no further parameters to adjust. Such linear systems of equations are easily solved and are well known to produce smooth curves with little oscillation as long as the degree of the curve is kept small, as for example a cubic B-spline. Higher orders of polynomial curves are more likely to produce unwanted oscillations or loops.

5-1-2 CONTINUITY

With polynomial curves, curve smoothness is inherent in the curve. However, when extending the curve to a piecewise spline the continuity must be enforced to maintain smoothness to the desired derivative. Continuity is studied in terms of either geometric or parametric [71, 95] hence the use of the capital letters G and C to represent them. Geometric continuity measures the continuity of the curve and its derivatives with respect to the arc length parameter s, whereas parametric continuity is taken with respect to an arbitrary parameter which is chosen to be u. Consequently,

a curve that is parametrically continuous to a given degree must also be geometrically continuous to the same degree. For example, the junction of two curves is considered G^1 continuous when their derivative vectors are collinear but do not necessarily have the same magnitude. If they both have the same magnitude then the curves at the junction are said to be C^1 continuous.

The chosen level of continuity of the curves used to generate axis motion commands has a direct effect on the dynamics of the position control and the resulting part surface. Clearly, C^0 continuity of the axis motion commands is necessary. C^1 continuity adds continuous velocity which makes acceleration discontinuous but finite. In terms of the rigid body dynamics the third derivative, known as jerk, influences vibration. As a minimum therefore the axis motion commands should be C^2 continuous to reduce acceleration and bound the jerk.

5-1-3 PARAMETERIZATION

Obtaining a smooth C^2 interpolant through the data points does not ensure that the feedrate

$$F(u) = \left\| \frac{d}{dt}\mathbf{P}(u) \right\| \tag{5-1}$$

will be constant if the curve parameter u is followed with a constant increment. In fact, it has already been noted that a polynomial curve cannot be unit speed [22]—the curve $\mathbf{P}(u)$ is *unit speed* if $\|\frac{d}{dt}\mathbf{P}(u)\| = 1$ for the entire range of u. This leaves the two alternatives detailed earlier: near arc-length interpolated curves and real-time prediction of the arc-length. The advantage of the real-time prediction of arc-length is its simplicity. Any curve whose first or higher derivatives can be easily calculated in a real-time control loop are candidates for use. However, as is illustrated by Fig. 5-1 the error in the algorithm proposed by Lin and Koren depends on the particular curve

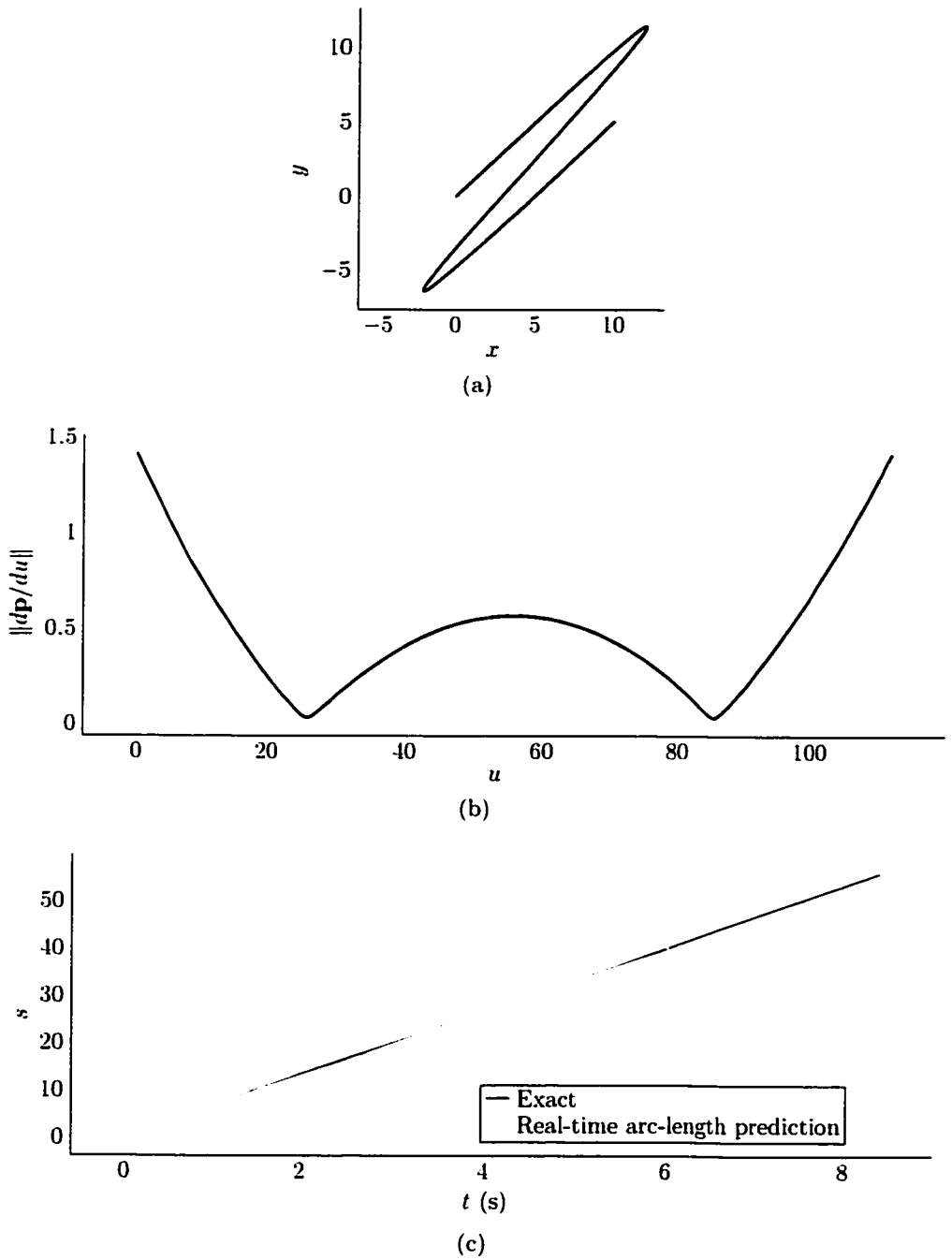


FIGURE 5–1 Demonstration of the real-time arc-length parameterization algorithm. See Table 5–1 for curve coefficients.

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_1'$	$\mathbf{p}_2'$
x	0	10	1	1
y	0	5	1	1

TABLE 5–1 Curve definition of Fig. 5–1 with $F(u) = 400\,\mathrm{s}^{-1}$, $\Delta t = 0.1\,\mathrm{s}$ and $l = 10\|\mathbf{p}_1 - \mathbf{p}_2\|$.

and choice of servo update period. For arbitrary curves the degree of error due to this algorithms is not known *a priori*. As compensation, higher order terms may be added to Eq. 2–12 to reduce the error but the lack of determinism is not eliminated. This is the primary advantage of the near arc-length method. Its deviation from unit speed is known and if it is too great can be corrected for, as illustrated later in this chapter. The near arc-length parameterization technique is therefore chosen as the basis for the tool-path interpolation algorithm to follow. Figure 5–2 depicts the proposed command generation architecture based on the tool-path and feedrate interpolation algorithms presented in this chapter.

5–2 SELECTION OF TOOL DISPLACEMENT VECTORS

A critical step in the interpolation of the tool-path is the selection of the interpolatory data. For linear interpolation of three-axis tool-paths this is accomplished by choosing an arc-length between linear segment end points such that the chordal deviation between the linear segment and the actual tool-path curve is no greater than a specified value (see Fig. 5–3). Sarma and Rao [84] extended this approach to five-axis machining. The distance between points for a specified arc-length is calculated for both the position and orientation, and choosing the smaller of the two. This method has some merit in choosing data points for the interpolatory spline too. However, splines better mimic the desired tool-path and therefore require fewer points. There

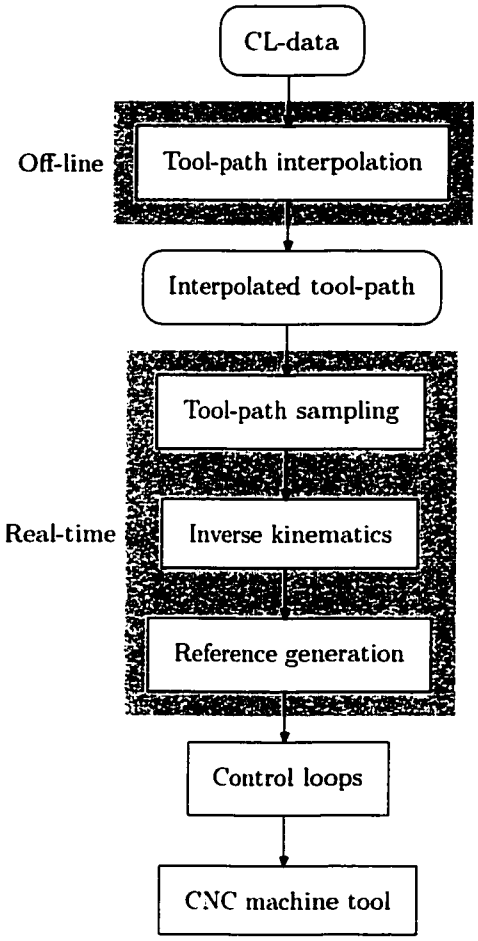


FIGURE 5–2 Proposed command generation architecture.

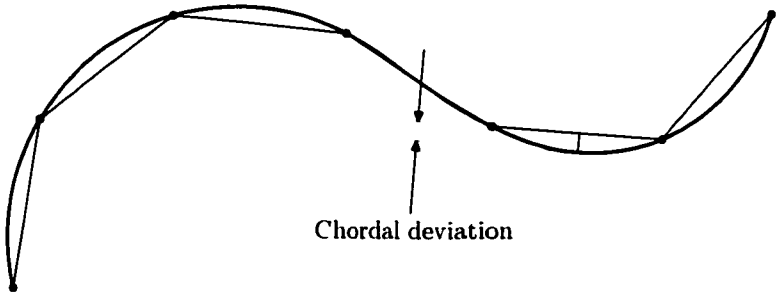


FIGURE 5–3 Chordal deviation of linear interpolation.

is little literature available on this subject. The following interpolation algorithms must therefore depend on the data provided as is in CL-data files.

5–3 POSITION INTERPOLATION

5–3–1 THE NEAR ARC-LENGTH PARAMETERIZED QUINTIC SPLINE

The position spline is based on the successive work of Renner, Wang, Yang, Schofield and Wright [81, 86, 103, 104, 105, 111] in near-arc length parameterized splines. The later work develops the method for C^2 continuous quintic polynomial splines

$$\mathbf{P}_i(u) = \mathbf{c}_{1_i} + \mathbf{c}_{2_i} u + \mathbf{c}_{3_i} u^2 + \mathbf{c}_{4_i} u^3 + \mathbf{c}_{5_i} u^4 + \mathbf{c}_{6_i} u^5, \quad u \in [0, l_i] \tag{5–2}$$

where $\mathbf{c}_{1_i}$.....$\mathbf{c}_{6_i}$ are the vector spline coefficients and l_i is the range of u. Thus the problem is interpolating a set of n+1 Cartesian position vectors $\mathbf{p}_i$ where $i = 1, ... n$+1 with Eq. 5–2.

Clearly, only a cubic polynomial is necessary to create a C^2 continuous spline. However, the additional degrees of freedom including the parameter range are necessary to imbue the spline with a near arc-length parameterization. Wang and Yang [105, 111] noticed that higher order splines tended to exhibit unwanted oscillations. To combat this problem, a C^2 cubic spline will be interpolated from which the tangent and curvature vectors at the data points will be extracted and employed in constructing the quintic spline.

A quintic spline segment has six coefficients and one range parameter. Therefore it has seven degrees of freedom. The following parameters

1. the position vectors $\mathbf{p}_i$ at the ends of each segment,

2. the first derivative vectors $\mathbf{p}'_i$ at the ends of each segment,

3. the second derivative vectors $\mathbf{p}_i''$ at the ends of each segment, and

4. the range of each segment l_i,

enumerate the seven degrees of freedom of each quintic spline segment. Written mathematically the first three constraints are

$$\mathbf{P}_i(u)\Big|_{i=0} = \mathbf{p}_i, \tag{5-3}$$

$$\mathbf{P}_i(u)\Big|_{i=l_i} = \mathbf{p}_{i+1}, \tag{5-4}$$

$$\frac{d}{du}\mathbf{P}_i(u)\Big|_{i=0} = \mathbf{p}_i', \tag{5-5}$$

$$\frac{d}{du}\mathbf{P}_i(u)\Big|_{i=l_i} = \mathbf{p}_{i+1}', \tag{5-6}$$

$$\frac{d^2}{du^2}\mathbf{P}_i(u)\Big|_{i=0} = \mathbf{p}_i'', \tag{5-7}$$

$$\frac{d^2}{du^2}\mathbf{P}_i(u)\Big|_{i=l_i} = \mathbf{p}_{i+1}''. \tag{5-8}$$

These equations are sufficient to determine the spline segment coefficients:

$$\mathbf{c}_{1_i} = \mathbf{p}_i, \tag{5-9}$$

$$\mathbf{c}_{2_i} = \mathbf{p}_i', \tag{5-10}$$

$$\mathbf{c}_{3_i} = \frac{\mathbf{p}_i''}{2}, \tag{5-11}$$

$$\mathbf{c}_{4_i} = \frac{10(\mathbf{p}_{i+1} - \mathbf{p}_i)}{l_i^3} - \frac{2(2\mathbf{p}_{i+1}' + 3\mathbf{p}_i')}{l_i^2} + \frac{\mathbf{p}_{i+1}'' - 3\mathbf{p}_i''}{2l_i}, \tag{5-12}$$

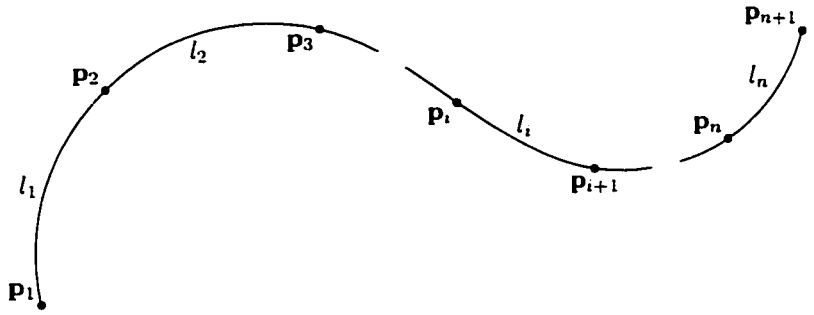


FIGURE 5–4 Position spline conventions.

$$\mathbf{c}_{5_i} = -\frac{15(\mathbf{p}_{i+1} - \mathbf{p}_i)}{l_i^4} + \frac{7\mathbf{p}'_{i+1} + 8\mathbf{p}'_i}{l_i^3} - \frac{2\mathbf{p}''_{i+1} - 3\mathbf{p}''_i}{2l_i^2}. \tag{5–13}$$

$$\mathbf{c}_{6_i} = \frac{6(\mathbf{p}_{i+1} - \mathbf{p}_i)}{l_i^5} - \frac{3(\mathbf{p}'_{i+1} + \mathbf{p}'_i)}{l_i^4} - \frac{\mathbf{p}''_{i+1} - \mathbf{p}''_i}{2l_i^3}. \tag{5–14}$$

The spline coefficients are clearly functions of the six variables $\mathbf{p}_i$, $\mathbf{p}_{i+1}$, $\mathbf{p}'_i$, $\mathbf{p}'_{i+1}$, $\mathbf{p}''_i$, $\mathbf{p}''_{i+1}$. This data is derived from a C^2 cubic polynomial spline interpolated through the same $\mathbf{p}_i$. See Fig. 5–4.

5-3-2 THE CUBIC SPLINE

The cubic polynomial spline similar to the quintic spline of Eq. 5–2

$$\mathbf{P}_i(u) = \mathbf{c}_{1_i} + \mathbf{c}_{2_i}u + \mathbf{c}_{3_i}u^2 + \mathbf{c}_{4_i}u^3, \quad u \in [0, l_i] \tag{5–15}$$

is fitted to the data points $\mathbf{p}_i$ first. This curve is initially chord length parameterized and both C^2 and G^2 continuous. Ideally, l_i should be the arc-length of the spline segment between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. Since this arc-length is unknown, the chord length is

used as an initial approximation:

$$l_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|, \quad i = 1, \ldots, n-1. \tag{5-16}$$

The four unknowns $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, $\mathbf{c}_4$ of n spline segments making a total of $4n$ unknowns must be constrained to obtain C^2 continuity. Firstly, C^0 and G^0 continuity of the cubic spline is imposed with

$$\mathbf{P}_i(u)\Big|_{u=0} = \mathbf{p}_i, \quad i = 1, \ldots, n. \tag{5-17}$$

$$\mathbf{P}_i(u)\Big|_{u=l_i} = \mathbf{p}_{i+1}, \quad i = 1, \ldots, n. \tag{5-18}$$

This eliminates $2n$ unknowns. C^1 and G^1 continuity eliminates a further $n-1$ degrees of freedom

$$\frac{d}{du}\mathbf{P}_i(u)\Big|_{u=l_i} = \frac{d}{du}\mathbf{P}_{i+1}(u)\Big|_{u=0}, \quad i = 1, \ldots, n-1. \tag{5-19}$$

Finally, C^2 and G^2 continuity is guaranteed by

$$\frac{d^2}{du^2}\mathbf{P}_i(u)\Big|_{u=l_i} = \frac{d^2}{du^2}\mathbf{P}_{i+1}(u)\Big|_{u=0}, \quad i = 1, \ldots, n-1. \tag{5-20}$$

This eliminates a further $n-1$ degrees of freedom.

These equations combine for a total of $4n-2$ constraints. However, to fully specify a cubic spline of n segments requires $4n$ constraints. An additional two constraints

are derived by specifying the first derivatives at the ends of the spline

$$\frac{d}{du}\mathbf{P}_1(u)\Big|_{u=0} = \mathbf{p}'_1. \tag{5-21}$$

$$\frac{d}{du}\mathbf{P}_n(u)\Big|_{u=l_{n-1}} = \mathbf{p}'_{n+1}. \tag{5-22}$$

These derivatives $\mathbf{p}'_1$, $\mathbf{p}'_{n+1}$ are readily obtained by estimation using a quadratic curve interpolated through the first and last three data points, respectively.

To find the vector coefficients for the entire cubic spline begin by evaluating [2. 23]

$$\mathbf{P}_i(u)\Big|_{i=0} = \mathbf{p}_i, \tag{5-23}$$

$$\mathbf{P}_i(u)\Big|_{i=l_i} = \mathbf{p}_{i+1}. \tag{5-24}$$

$$\frac{d^2}{du^2}\mathbf{P}_i(u)\Big|_{i=0} = \mathbf{p}''_i. \tag{5-25}$$

$$\frac{d^2}{du^2}\mathbf{P}_i(u)\Big|_{i=l_i} = \mathbf{p}''_{i+1}, \tag{5-26}$$

and then by rearranging, the vector coefficients are obtained in terms of the interpolatory data $\mathbf{p}_i$ and second derivatives at the interpolatory points $\mathbf{p}''_i$

$$\mathbf{c}_1 = \mathbf{p}_i, \tag{5-27}$$

$$\mathbf{c}_2 = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{l_i} - \frac{l_i}{6}(\mathbf{p}''_{i+1} + 2\mathbf{p}''_i), \tag{5-28}$$

$$\mathbf{c}_3 = \frac{\mathbf{p}''_i}{2}, \tag{5-29}$$

$$\mathbf{c}_4 = \frac{\mathbf{p}''_{i+1} - \mathbf{p}''_i}{6l_i}. \tag{5-30}$$

To find the $\mathbf{p}''$ evaluate

$$\left.\frac{d}{du}\mathbf{P}_i(u)\right|_{i=l_i} = \left.\frac{d}{du}\mathbf{P}_{i+1}(u)\right|_{i=0} \tag{5-31}$$

and substitute Eq. 5-27. 5-28. 5-29 and 5-30 to obtain

$$\frac{\mathbf{p}''_i l_i + 2\mathbf{p}''_{i+1}(l_i + l_{i+1}) + \mathbf{p}''_{i+2}}{6} = \frac{\mathbf{p}_{i+2} - \mathbf{p}_{i+1}}{l_{i+1}} - \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{l_i}. \tag{5-32}$$

Rearranging into matrix form and adding Eq. 5-21 and 5-22 yields

$$\begin{pmatrix} \frac{l_1}{3} & \frac{l_1}{6} & & & & & 0 \\ \frac{l_1}{6} & \frac{l_1+l_2}{3} & \frac{l_2}{6} & & & & \\ & & \ddots & & & & \\ & & \frac{l_{i-1}}{6} & \frac{l_{i-1}+l_i}{3} & \frac{l_i}{6} & & \\ & & & & \ddots & & \\ & & & & \frac{l_n}{6} & \frac{l_n+l_{n+1}}{3} & \frac{l_{n-1}}{6} \\ 0 & & & & & \frac{l_{n+1}}{6} & \frac{l_{n+1}}{3} \end{pmatrix} \begin{pmatrix} (\mathbf{p}''_1)^T \\ \vdots \\ (\mathbf{p}''_i)^T \\ \vdots \\ (\mathbf{p}''_{n+1})^T \end{pmatrix} =$$

$$\begin{pmatrix} \left(\frac{\mathbf{p}_2-\mathbf{p}_1}{l_1} - \mathbf{p}'_1\right)^T \\ \left(\frac{\mathbf{p}_3-\mathbf{p}_2}{l_2} - \frac{\mathbf{p}_2-\mathbf{p}_1}{l_1}\right)^T \\ \vdots \\ \left(\frac{\mathbf{p}_{i+1}-\mathbf{p}_i}{l_i} - \frac{\mathbf{p}_i-\mathbf{p}_{i-1}}{l_{i-1}}\right)^T \\ \vdots \\ \left(\frac{\mathbf{p}_{n+1}-\mathbf{p}_n}{l_n} - \frac{\mathbf{p}_n-\mathbf{p}_{n-1}}{l_{n-1}}\right)^T \\ \left(\mathbf{p}'_{n+1} - \frac{\mathbf{p}_{n+1}-\mathbf{p}_n}{l_n}\right)^T \end{pmatrix}. \tag{5-33}$$

This is a simple set of linear equations. The square matrix is tridiagonal and is therefore readily solved by LU decomposition [78].

5-3-3 OBTAINING THE TANGENT AND CURVATURES FOR THE QUINTIC SPLINE

When Eq. 5–33 is solved the $\mathbf{p}_i$ and $\mathbf{p}_i''$ for $i = 1.....n+1$ of the cubic spline are known. The $\mathbf{p}_i'$ are obtained from Eq. 5–21 and 5–22 for $i = 1$ and $i = n+1$, respectively. For the remainder the following equation is obtained

$$\mathbf{p}_i' = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{l_i} - \frac{l_i(\mathbf{p}_{i+1}'' + 2\mathbf{p}_i'')}{6}. \quad i = 2.....n. \tag{5–34}$$

There is sufficient information to compute the quintic spline coefficients of Eqs. 5–9, 5–10, 5–11, 5–12, 5–13 and 5–14 using the first and second derivatives from the cubic spline. However, the quintic spline is made closer to arc-length parameterization by using the tangent and curvature vectors in place of the first and second derivatives from the cubic spline.

The tangent of the curve $\mathbf{P}(u)$ is found by taking the derivative of it with respect to the arc-length parameter s

$$\frac{d}{ds}\mathbf{P}(u) = \frac{d\mathbf{p}}{ds} = \frac{d\mathbf{p}}{du}\frac{du}{ds}. \tag{5–35}$$

By finding the length of the tangent vector $d\mathbf{p}/ds$ in Eq. 5–35 and noting that the length of the tangent vector must be unity the following result is obtained

$$\frac{du}{ds} = \frac{1}{\left\|\frac{d\mathbf{p}}{du}\right\|}. \tag{5–36}$$

Combining Eqs. 5–35 and 5–36 results in the well known equation for calculating the tangent of a single parameter curve

$$\frac{d}{ds}\mathbf{P}(u) = \frac{d\mathbf{p}}{ds} = \frac{\frac{d\mathbf{p}}{du}}{\left\|\frac{d\mathbf{p}}{du}\right\|}. \tag{5–37}$$

The curvature is obtained by differentiating Eq. 5–37. This yields

$$\frac{d^2}{ds^2}\mathbf{P}(u) = \frac{d^2\mathbf{p}}{ds^2} = \frac{\frac{d^2\mathbf{p}}{du^2}\frac{du}{ds}}{\left\|\frac{d\mathbf{p}}{du}\right\|} - \frac{\left(\frac{d\mathbf{p}}{du}\cdot\frac{d^2\mathbf{p}}{du^2}\right)\frac{du}{ds}\frac{d\mathbf{p}}{du}}{\left\|\frac{d\mathbf{p}}{du}\right\|^3}. \tag{5–38}$$

Combining Eq. 5–38 with Eq. 5–36 and 5–37 results in a relation for curvature depending only on the first and second derivative of curve $\mathbf{P}(u)$

$$\frac{d^2\mathbf{p}}{ds^2} = \frac{\left(\frac{d\mathbf{p}}{du}\cdot\frac{d\mathbf{p}}{du}\right)\frac{d^2\mathbf{p}}{du^2} - \left(\frac{d\mathbf{p}}{du}\cdot\frac{d^2\mathbf{p}}{du^2}\right)\frac{d\mathbf{p}}{du}}{\left(\frac{d\mathbf{p}}{du}\cdot\frac{d\mathbf{p}}{du}\right)^2}. \tag{5–39}$$

Wright *et al.* [86, 103, 104] compute the quintic spline with C^3 continuity. This disallows the possibility of specifying the curvature at the data points. Hence at the price of a greater degree of continuity comes greater error in the parameterization. Figure 5–5 illustrates the results from a sample tool-path. The average parameterization error for the C^3 continuous spline is 0.0091% and the average error for the C^2 continuous spline is 0.0048%, or approximately half as large.

5–3–4 DETERMINATION OF SEGMENT RANGE

Farouki and Sakkalis [22] proved that real rational curves cannot be unit speed. Hence there is only one alternative—making the curves unit speed at discrete points spaced closely enough so that the deviation from arc-length parameterization is within acceptable limits between the points. Besides forcing arc-length parameterization at

the end points of each spline segment, the parameterization error is further reduced by doing so at a point in between ($u = l_i/2$)

$$\left\| \frac{d}{du}\mathbf{P}_i(u)\Big|_{u=\frac{l_i}{2}} \right\| = 1. \tag{5-40}$$

The variable l_i must be found. From Eqs. 5–40 and 5–2, the following is obtained

$$\left\| \mathbf{c}_{2_i} + \mathbf{c}_{3_i} l_i + \frac{\mathbf{c}_{4_i}}{2} + 2\mathbf{c}_{5_i} l_i^3 + \frac{5\mathbf{c}_{6_i} l_i^4}{2} \right\| = 1. \tag{5-41}$$

After the substitution of Eqs. 5-9, 5-10, 5-11, 5-12, 5-13 and 5-14

$$a + bl_i + cl_i^2 + dl_i^3 + el_i^4 = 0 \tag{5-42}$$

where

$$a = 3600(\mathbf{p}_{i+1} - \mathbf{p}_i)^2, \tag{5-43}$$

$$b = -1680(\mathbf{p}_{i+1} - \mathbf{p}_i)(\mathbf{p}'_{i+1} + \mathbf{p}'_i), \tag{5-44}$$

$$c = -632 + 392(\mathbf{p}'_i\mathbf{p}'_{i+1}) + 120(\mathbf{p}_{i+1} - \mathbf{p}_i)(\mathbf{p}''_{i+1} - \mathbf{p}''_i), \tag{5-45}$$

$$d = -28(\mathbf{p}''_{i+1} - \mathbf{p}''_i)(\mathbf{p}'_{i+1} + \mathbf{p}'_i), \tag{5-46}$$

$$e = (\mathbf{p}''_{i+1} - \mathbf{p}''_i)^2. \tag{5-47}$$

This equation can be easily solved by the Newton-Raphson iteration method [78] for l_i.

5–3–5 INTERPOLATION ALGORITHM

The near arc-length parameterized C^2 interpolatory quintic spline is computed iteratively. The algorithm iteration scheme is shown in Fig. 5–6. Initially, the l_i are set to chord length. It is then used to compute the C^2 cubic interpolatory spline. From the cubic spline are extracted the first derivative and second derivatives corresponding to the interpolatory points. The tangent and curvature vectors are computed starting from the first and second derivative vectors and employed in computing the quintic spline coefficients. The parameterization is then improved by calculating new l_i such that the middle of the spline segments are unit speed in addition to the end-points. This new parameterization violates the Eq. 5–12, Eq. 5–13 and Eq. 5–14 and therefore the algorithm must be repeated from the calculation of the cubic spline. Iteration is terminated when the length of the curve ($\sum l_i$) changes by less than a specified threshold.

5–3–6 IMPROVEMENT OF ARC-LENGTH PARAMETERIZATION

The essence of the above algorithm is the forcing of the quintic spline to be unit speed at three discrete points per segment: the start point, middle point and end point. Consequently, the curve deviates from arc-length parameterization between these three points of each spline segment. From experience it is observed that the error in the parameterization is greatest under the following conditions:

1. the curvature of the spline segment is large; and

2. an inflection point occurs in the spline segment.

Wang, Schofield and Wright [103, 104] propose a simple and effective remedy. An auxiliary point is inserted in the spline segment where the error in parameterization

is excessive. The procedure for inserting the auxiliary points into the quintic spline is:

1. Locate the largest error in parameterization along the curve. Naturally this will occur farthest from the points on the curve which are forced to unit speed. Hence at each spline segment check at $u = l_i/4$ and $u = 3l_i/4$.

2. If the error is outside acceptable limits between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ evaluate Eq. 5–2 at $u = l_i/2$ as $\mathbf{p}_{i+1/2}$.

3. Insert $\mathbf{p}_{i+1/2}$ between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and reconstruct the quintic spline.

4. Repeat steps 1–3 until the arc-length parameterization is satisfied or the distance between two points reaches a preset minimum.

This method proves to be efficient in improving the arc-length parameterization of the curve. The speed of convergence of the iterations to the satisfactory arc-length parameterization is reasonably fast. As described above, this is due to the fact that the worst parameterization occurs at $u = l_i/4$ and $u = 3l_i/4$. The auxiliary point causes the parameterization to be clamped to unit speed at exactly these locations.

Figure 5–7 applies this algorithm to the data of Table 5–2. An initial five points were interpolated, yielding curve speeds between 0.96 and 1.08 approximately. A further eight points were inserted. The general nature of the curve does not change (no new oscillations or loops were introduced) but the speed of the curve is reduced to the range 0.997 to 1.001 approximately. The range of the error was reduced by approximately a factor of 30.

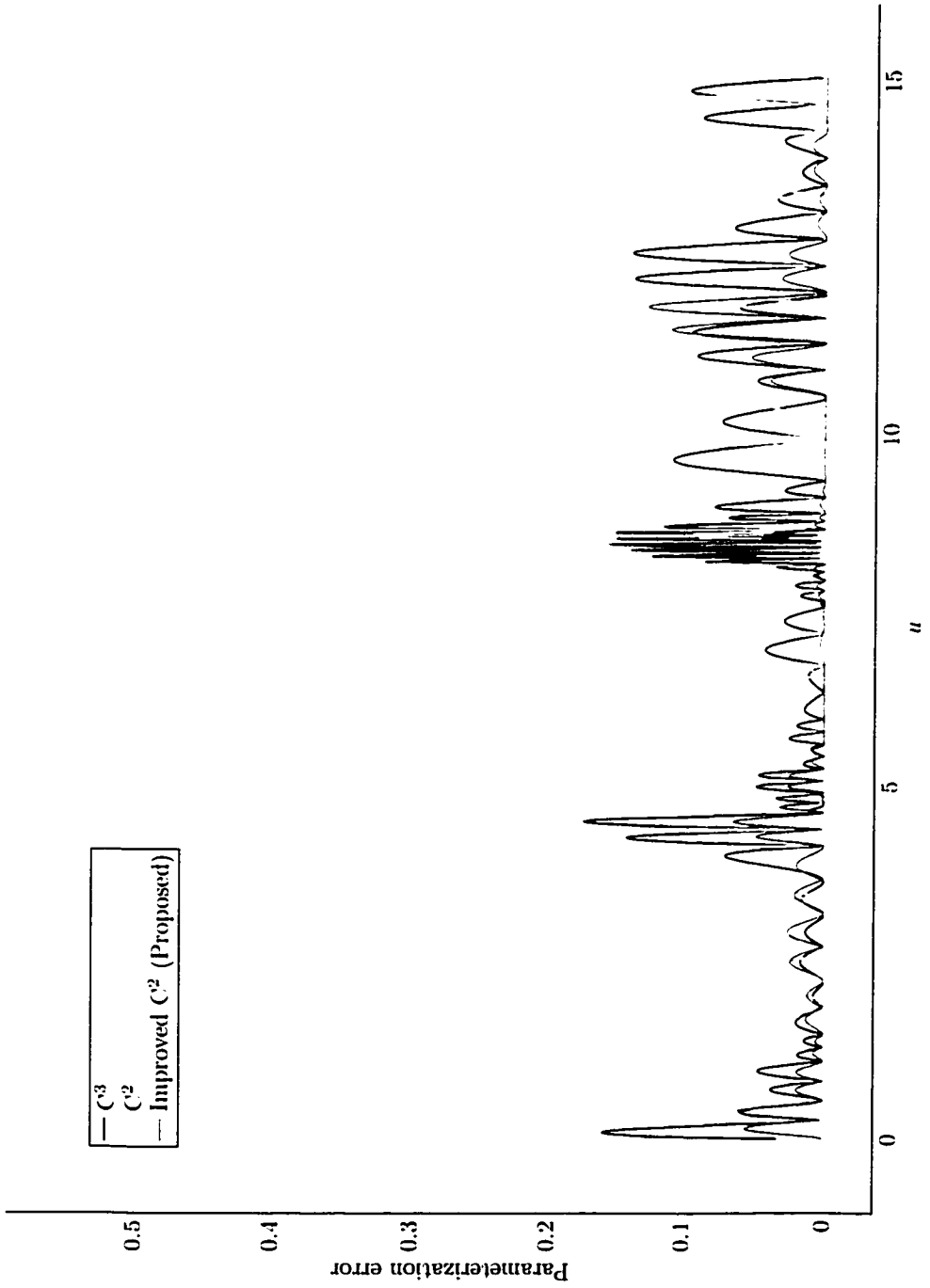


FIGURE 5–5 Comparison of the parameterization error of the proposed position spline algorithm and the algorithm proposed by Wright *et al.*. The parameterization error is measured as $|1 - \|d\mathbf{p}/du\|| \times 100\%$.

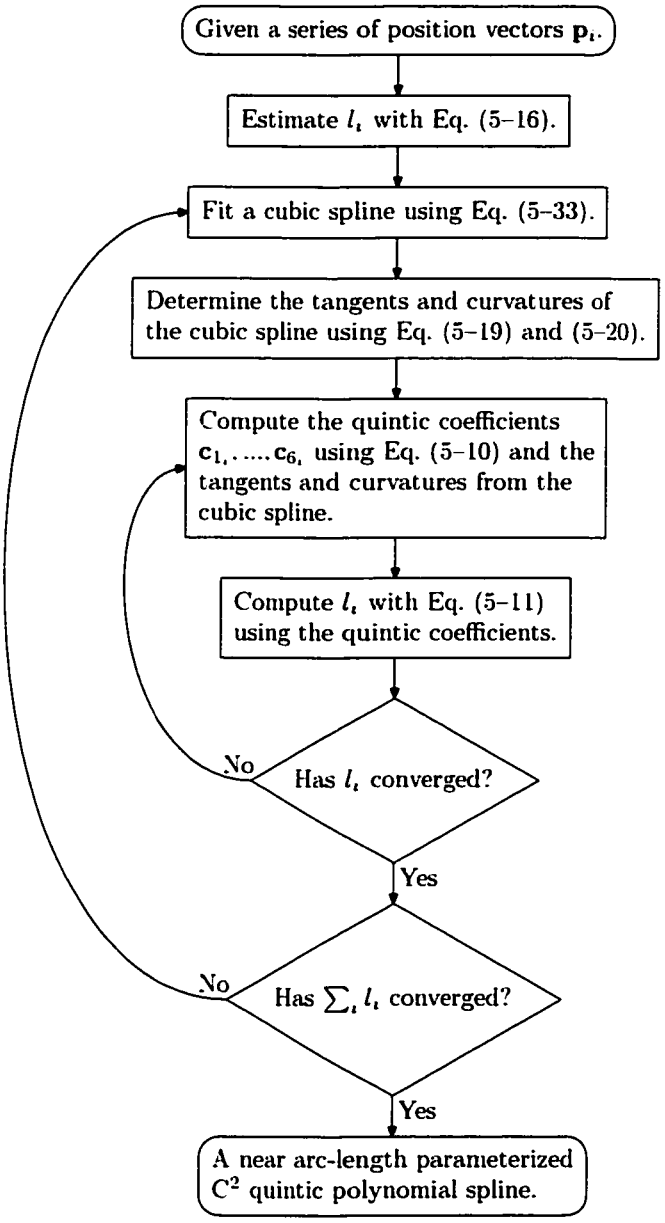


FIGURE 5–6 Position spline interpolation algorithm.

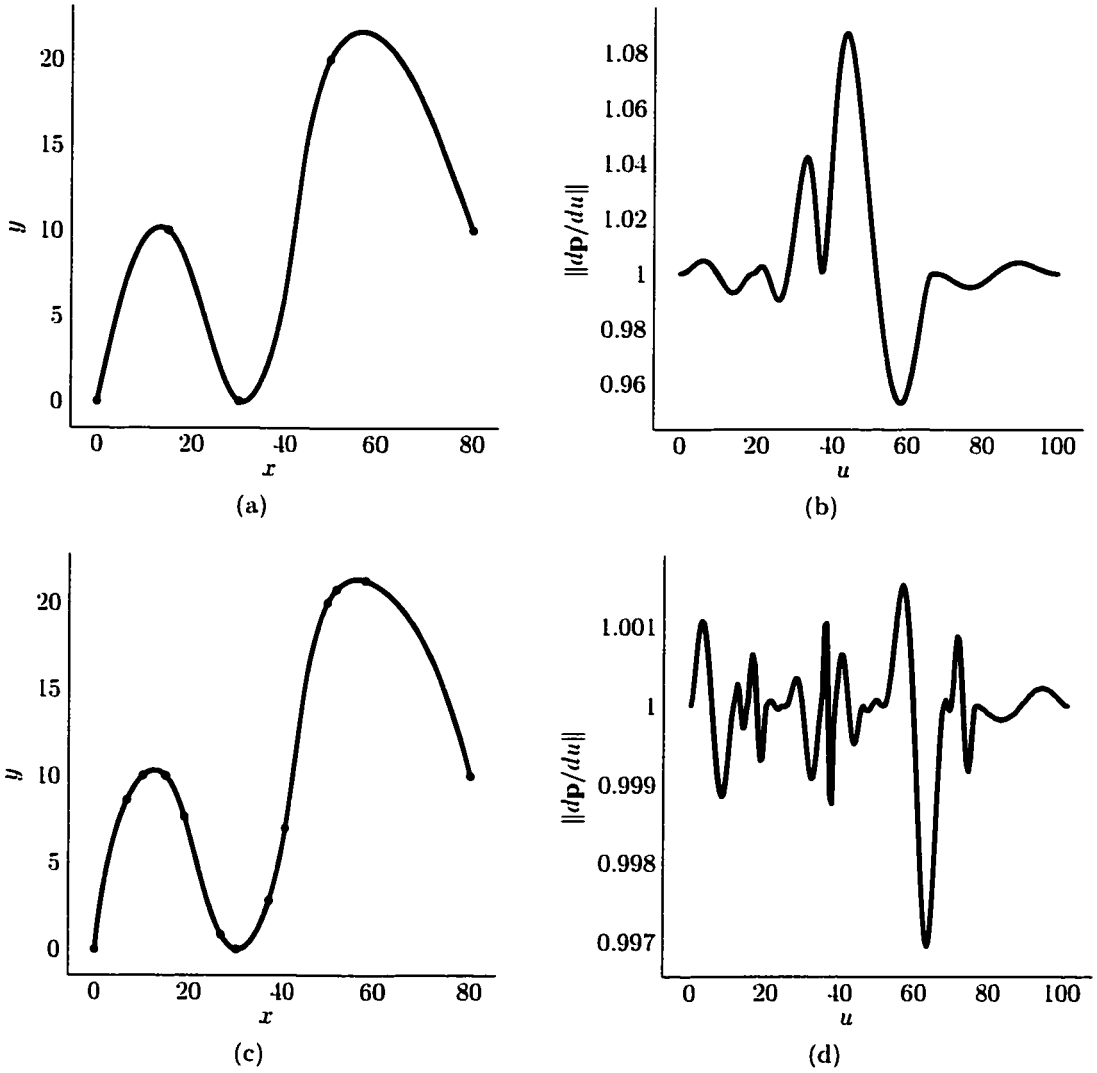


FIGURE 5–7 Comparison of curve speed for curves with and without auxiliary points. Plots (a) and (b) illustrate the original curve, and plots (c) and (d) illustrate the curve interpolated through the refined point set. See Table 5–2 for interpolatory point data.

	Original Points		Refined Points	
	x	y	x	y
1	0	0	0	0
2	15	10	6.837317	8.610378
3	30	0	10.251177	10.016348
4	50	20	15	10
5	80	10	18.930453	7.640185
6			26.694403	0.856462
7			30	0
8			36.918713	2.828018
9			40.412620	6.990993
10			50	20
11			51.958650	20.755156
12			58.201171	21.263857
13			80	10

TABLE 5–2 Interpolatory points for Fig. 5–7.

5–4 ORIENTATION INTERPOLATION

The orientation spline interpolates the orientation portion of the tool-path. Since the orientation of the tool is defined by a unit vector, the orientation spline must lie on the surface of the unit sphere. If instead the orientation spline interpolates unit quaternions then the interpolatory curve lies on a unit hypersphere. Construction of a near arc-length parameterized orientation spline is discussed in two parts below. First the spherical Bézier spline, which lies on a unit sphere, is introduced. Interpolating a set of unit vectors with a near arc-length parameterized spherical Bézier spline is then discussed.

5–4–1 SPHERICAL BÉZIER SPLINE

To construct a curve which lies on the unit sphere using the de Casteljau form of the Bézier curve [21, 76], replace the linear interpolation with a spherical interpolation.

The spherical interpolation curve, $\mathbf{S}(v)$, interpolates the two unit vectors or two unit quaternions, $\mathbf{q}_1$ and $\mathbf{q}_2$ as follows [32]

$$\mathbf{S}(v) = \frac{\mathbf{q}_1 \sin(\theta(1-v)) + \mathbf{q}_2 \sin(\theta v)}{\sin(\theta)} \tag{5-48}$$

where

$$\theta = \arccos(\mathbf{q}_1 \cdot \mathbf{q}_2). \tag{5-49}$$

$$\|\mathbf{q}_1\| = \|\mathbf{q}_2\| = 1. \tag{5-50}$$

It can be shown that

$$\|\mathbf{S}(v)\| = 1, \quad v \in [0, 1] \tag{5-51}$$

and

$$\mathbf{S}(v)\Big|_{v=0} = \mathbf{q}_1. \tag{5-52}$$

$$\mathbf{S}(v)\Big|_{v=1} = \mathbf{q}_2. \tag{5-53}$$

Given $j+1$ control points $\mathbf{d}_k$ a spherical Bézier curve, $\mathbf{B}_k^j(v)$, of degree j is given by

$$\mathbf{B}_k^j(v) = \begin{cases} \mathbf{d}_k, & \text{if } j = 0 \\ \dfrac{\mathbf{B}_k^{j-1}(v)\sin(\theta(1-v)) + \mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v)}{\sin(\theta)}, & \text{if } j > 0 \end{cases} \tag{5-54}$$

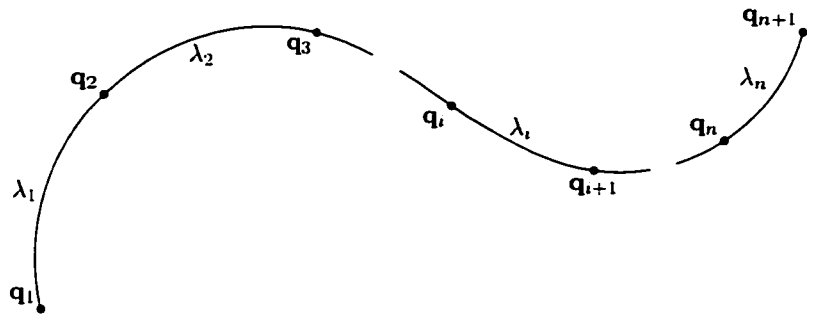


FIGURE 5–8 Orientation spline conventions.

where

$$\|\mathbf{d}_k\| = 1, \tag{5–55}$$

$$\theta = \arccos(\mathbf{B}_k^{j-1}(v) \cdot \mathbf{B}_{k+1}^{j-1}(v)). \tag{5–56}$$

The ith segment of a spherical Bézier spline is given by

$$\mathbf{Q}_i^j(v) = \mathbf{B}_{1_i}^j\left(\frac{v}{\lambda_i}\right), \quad v \in [0, \lambda_i], \quad i = 1, \ldots, n \tag{5–57}$$

where v is the orientation parameter, and λ_i is the range of v. This equation is the orientation analogue of the position spline, Eq. 5–2. It has six unknown vectors $\mathbf{d}_{1_i}, \ldots, \mathbf{d}_{6_i}$ and a segment length λ_i. See Fig. 5–8. Similar to the Bézier curve in Euclidean space, the spherical Bézier spline has the following properties:

1. Eq. 5–52 ensures endpoints of a segment (at $v = 0$ and $v = \lambda$) correspond to the first and last control point, $\mathbf{d}_1$ and $\mathbf{d}_{j+1}$, respectively.

2. The first derivatives at the endpoints of a segment depend on only the nearest two control points, $\mathbf{d}_1$ and $\mathbf{d}_2$ at $v = 0$, and $\mathbf{d}_j$ and $\mathbf{d}_{j+1}$ at $v = \lambda$. To ensure

C^1 continuity at the junctions. the four local control points $\mathbf{d}_{j_i}$. $\mathbf{d}_{(j+1)_i} = \mathbf{d}_{1_{i+1}}$ and $\mathbf{d}_{2_{i+1}}$ must lie on a great circle of the unit sphere.

3. Similarly, the second derivatives at the endpoints of a segment depend only on the nearest three control points.

5-4-2 INTERPOLATING THE ORIENTATION SPLINE

The object of the orientation spline is to interpolate the set of $n + 1$ orientation vectors. $\mathbf{q}_1,....\mathbf{q}_{n+1}$. with the quintic ($j = 5$) spherical Bézier spline of Eq. 5–57. As previously discussed. the interpolated spline must have both the near arc-length parameterization property and C^2 continuity. The same algorithm used for the position spline is applied to the orientation spline. with adjustment for the curve type.

The algorithm for computing the orientation spline is summarized in Fig. 5–9. It combines two distinct procedures. The first is the fitting of a C^2 quintic spline. The second is the improvement of the parameterization of the spline such that the parameterization approaches arc-length.

5-4-3 FITTING THE QUINTIC SPLINE

Fitting the C^2 quintic spline involves two steps: computing a C^2 cubic spline and then using information from the cubic spline to construct the quintic spline.

The cubic spline is computed by solving the set of non-linear equations

$$\mathbf{Q}_i^j(v)\Big|_{v=0} = \mathbf{q}_i, \qquad i = 1, ..., n \tag{5–58}$$

$$\mathbf{Q}_i^j(v)\Big|_{v=\lambda_i} = \mathbf{q}_{i+1}, \qquad i = 1, ..., n \tag{5–59}$$

$$\left.\frac{d}{dv}\mathbf{Q}_i^j(v)\right|_{v=\lambda_i} = \left.\frac{d}{dv}\mathbf{Q}_{i+1}^j(v)\right|_{v=0}, \qquad i = 1, \ldots, n-1 \tag{5-60}$$

$$\left.\frac{d^2}{dv^2}\mathbf{Q}_i^j(v)\right|_{v=\lambda_i} = \left.\frac{d^2}{dv^2}\mathbf{Q}_{i+1}^j(v)\right|_{v=0}, \qquad i = 1, \ldots, n-1 \tag{5-61}$$

for the unknown control points $\mathbf{d}_{2_i}$ and $\mathbf{d}_{3_i}$. The control points $\mathbf{d}_{1_i}$ and $\mathbf{d}_{4_i}$ correspond to the data points $\mathbf{q}_i$ and are therefore known. Broyden's method [78] has been successfully used to solve this set of equations, but it requires the first derivatives at the ends of the spline and initial estimates of the control points. If estimates of the tangents, $\mathbf{q}'_i$, can be obtained first then the control points can be determined from Eq. A–7 and A–8.

Estimates for these first derivatives are obtained by interpolating successive sets of three data points with a quadratic spherical Bézier curve. The quadratic Bézier curve has three control points. The first and third control points correspond respectively to the first and third data points. The middle control point is computed using Broyden's method with the middle data point as the initial estimate. After each iteration of this method the middle control point is normalized to a unit vector. From these quadratic curves the tangents necessary to compute the cubic spline are extracted.

Once a cubic spline has been fitted, the first and second derivatives of the cubic spline at the data points are extracted. These are used to compute the tangents and curvatures in the same way as for the position spline using Eq. 5–37 and 5–39. Using the tangents and curvatures from the cubic spline the unknown quintic control points $\mathbf{d}_{2_i}$, $\mathbf{d}_{3_i}$, $\mathbf{d}_{4_i}$ and $\mathbf{d}_{5_i}$ are found.

5–4–4 PARAMETERIZATION IMPROVEMENT

An initial estimate of λ_i is obtained in a similar manner to the chord length for position splines

$$\lambda_i = \arccos(\mathbf{q}_i \cdot \mathbf{q}_{i+1}) \tag{5–62}$$

This equation computes the geodesic distance on a unit sphere between the two unit vectors $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$.

The first improvement to the parameterization is made in the selection of the tangents and curvatures of the quintic spline by modifying first and second derivatives extracted from the cubic spline using Eq. 5–37 and 5–39.

The second improvement to the parameterization is made once the quintic spline control points are known. By forcing the unit tangency property at the middle of each segment, the segment lengths are obtained and parameterization of the spline is improved:

$$\left\| \frac{d}{dv} \mathbf{Q}_i^5(v) \Big|_{v=\frac{\lambda_i}{2}} \right\| = 1 \tag{5–63}$$

This equation can be solved efficiently for λ_i with Brent's method [78].

However, the quintic control points depend on λ_i. Once λ_i is recomputed with Eq. 5–63 the control points are no longer valid and Eq. 5–60 and Eq. 5–61 are violated. Therefore it is necessary to iterate through these two steps until the change in λ_i is less than a specified threshold value.

Similarly, the cubic and quintic splines are refitted until $\sum_i \lambda_i$ changes by less than a specified threshold.

In computing the cubic and quintic spherical Bézier spline it is necessary to calculate the first and second derivatives at the ends of each segment of the spline. It is also necessary to calculate the control points in the vicinity of the end points of each

segment given the tangent and curvature at the end points. See Appendix A.

5–5 ORIENTATION REPARAMETERIZATION

The orientation reparameterization spline ensures coordinated motion by the position and orientation splines. In three-axis machining, near arc-length parameterized splines are used to achieve constant feedrate coordinated motion. The length of the spline therefore determines the distance the tool travels, and machining time required. In multi-axis machining, maintaining a constant feedrate along the position spline and a constant angular velocity along the orientation spline would result in uncoordinated motion. This is because the two splines have different lengths. Merely scaling the length of the orientation or position spline is insufficient since the spacing between matching points on the two splines is unlikely to be proportional. Consequently, the orientation spline must be reparameterized to meet the coordinated motion requirement.[1]

Coordinated motion is achieved by reparameterizing the orientation spline with an orientation reparameterization spline. The reparameterization spline relates the position spline parameter u to the orientation spline parameter v for each segment:

$$v = V_i(u). \tag{5-64}$$

For each segment of the spline coordinated motion is ensured with

$$V_i(0) = 0, \tag{5-65}$$

$$V_i(l_i) = \lambda_i. \tag{5-66}$$

[1]Alternatively the position spline could be reparameterized should a constant angular velocity be desired.

The reparameterization spline is formulated as a C^2 monotonic rational quadratic spline based on the work of Fritsch and Carlson [33], and Delbourgo and Gregory [17. 40]. The spline is interpolated through the knots of the position and orientation splines and is given as

$$V_i(u) = \frac{\mu_{i+1}u^2 + l_i\lambda_i^{-1}(\mu_{i+1}h_i + \mu_i h_{i+1})u(l_i - u) + \mu_i(l_i - u)^2}{u^2 + l_i\lambda_i^{-1}(h_i + h_{i+1})u(l_i - u) + (l_i - u)^2} - \mu_i. \quad i = 1. \ldots . n \tag{5-67}$$

where

$$\mu_1 = 0. \quad \mu_i = \sum_{j=1}^{i-1} \lambda_j. \quad i = 2. \ldots . n + 1.$$

The h_i are the first derivatives of the spline at the segment junctions and are unknown. They are solved for in the following procedure. Let

$$a_i = \frac{1}{\lambda_i}. \qquad i = 1. \ldots . n. \tag{5-68}$$

$$b_i = \frac{\lambda_{i-1}}{l_{i-1}^2} + \frac{\lambda_i}{l_i^2}. \quad i = 2. \ldots . n. \tag{5-69}$$

$$c_i = \frac{1}{l_{i-1}} + \frac{1}{l_i}. \quad i = 2, \ldots , n. \tag{5-70}$$

Initialize the h_i with

$$h_1 = \frac{\lambda_1 l_2(l_1 + l_2 + 1) - \lambda_2 l_1}{l_1 l_2(l_1 + l_2)}, \tag{5-71}$$

$$h_i = \left(\frac{b_i}{a_{i-1} + a_i}\right)^{\frac{1}{2}}, \quad i = 2, \ldots , n, \tag{5-72}$$

$$h_{n+1} = \frac{\lambda_n l_{n-1}(l_n + l_{n-1} + 1) - \lambda_{n-1} l_n}{l_n l_{n-1}(l_n + l_{n-1})}. \tag{5-73}$$

Using Gauss-Seidel [78] iterate

$$h_i^{(w+1)} = \frac{c_i - a_{i-1}h_{i-1}^{(w+1)} - a_i h_{i+1}^{(w)} + [(c_i - a_{i-1}h_{i-1}^{(w+1)} - a_i h_{i+1}^{(w)})^2 + 4(a_{i-1} + a_i)b_i]^{\frac{1}{2}}}{2(a_{i-1} + a_i)}.$$

$$i = 2, \ldots, n \tag{5-74}$$

until

$$\max_i \left| h_i^{(w+1)} - h_i^{(w)} \right| \leq \varepsilon. \tag{5-75}$$

The reparameterization spline overwhelms the parameterization of the orientation spline making the angular velocity non-constant. However, in Fig. 5–10 the proposed algorithm is compared to the work of Ge and Srinivasan [35, 95] which essentially assumes $\lambda_i = l_i$. In the figure, the natural parameterization traces correspond to the work of Ge and Srinivasan, while the near arc-length parameterization traces refer to the proposed method. The former is reproduced using a cubic spherical Bézier spline with parameterization derived from the position spline. The orientation spline based on this assumption therefore need only be a cubic spherical spline and consequently is not near arc-length parameterized. Hence fluctuations between interpolatory points appear in the angular velocity. If instead a near arc-length parameterized orientation spline is used along with a reparameterization spline, the result is an improvement in the smoothness of the angular velocity and on average a reduction in angular acceleration of 11.6% and a reduction of 39.3% at the peak near 12 s. The improvement is a result of the near arc-length parameterization technique. The technique reduces speed fluctuation between interpolatory points in the orientation spline.

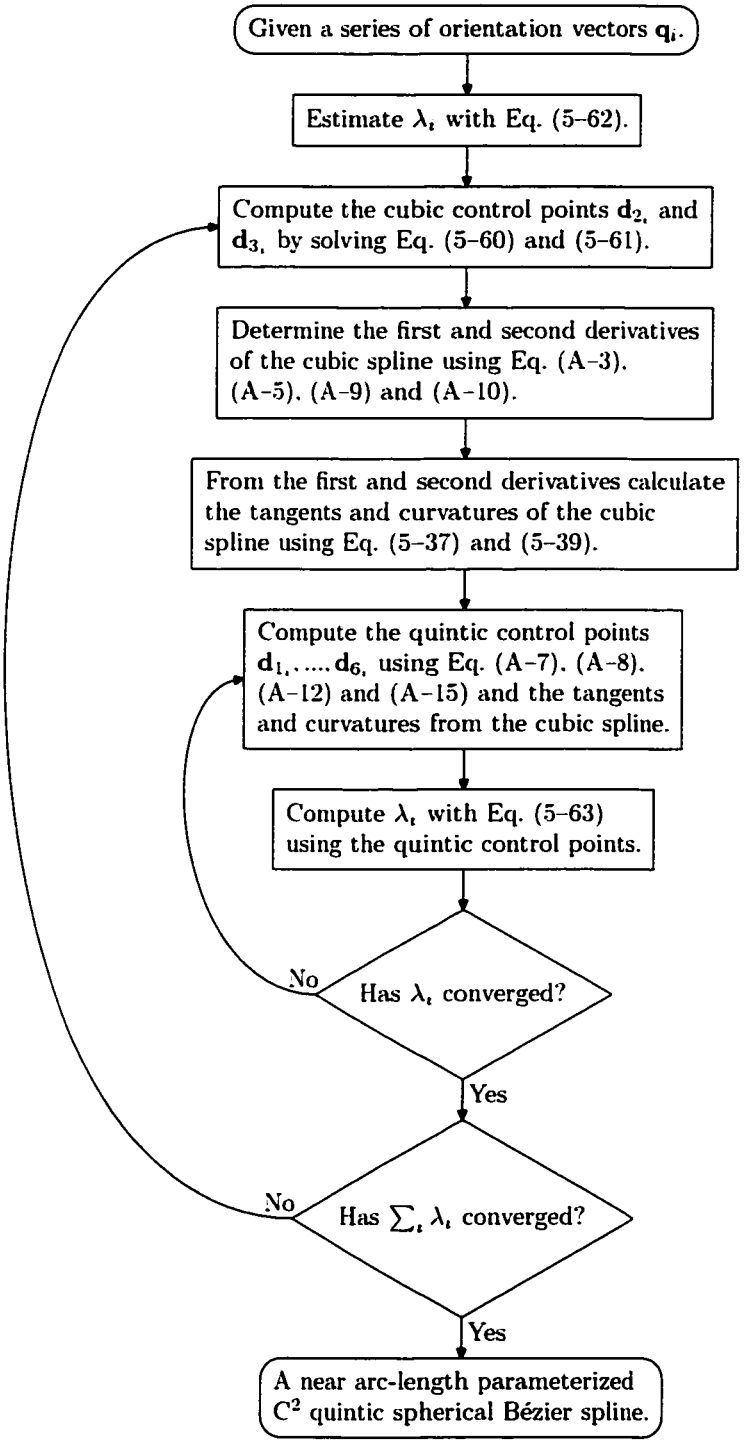


FIGURE 5–9 Orientation spline interpolation algorithm.

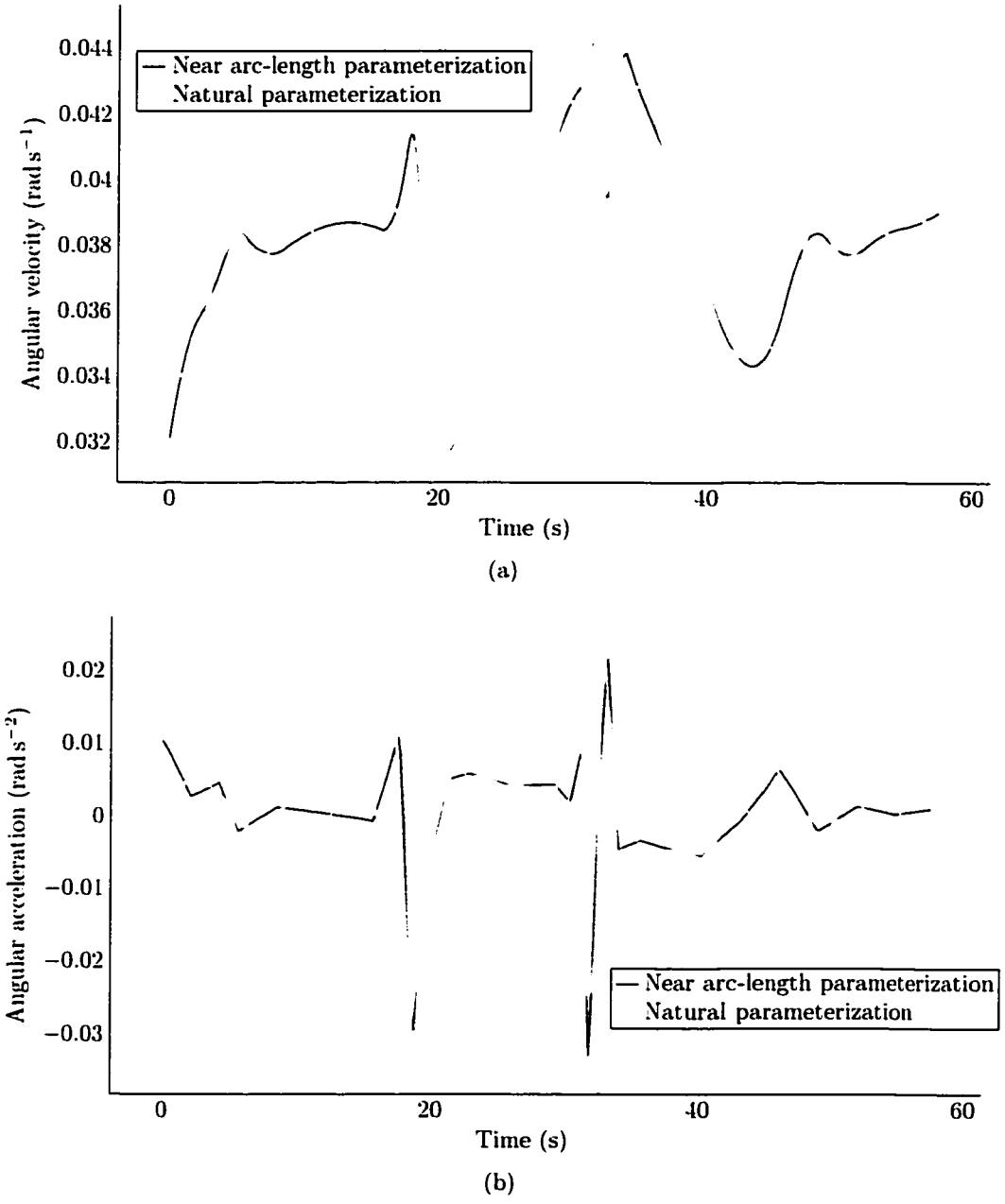


FIGURE 5–10 Comparison of angular velocity and acceleration of naturally and near arc-length parameterized spherical C^2 splines.

5–6 FEEDRATE INTERPOLATION

It has been established that a constant feedrate for five-axis machining is unrealistic. With the conventional practice of commanding step changes in feedrate the C^2 continuity of the above algorithms is lost at the point of the change in feedrate. To maintain the axis motion command continuity for a non-constant feedrate, the feedrate must be specified with the equivalent continuity. Simply stated, an interpolated curve is necessary to interpolate feedrates f_i corresponding to the $\mathbf{p}_i$ where $i = 1, \ldots, n+1$. The distances between the feedrates are given by l_i from the position spline. To take into account machining constraints, the interpolated feedrate may not be negative and should never exceed a maximum feedrate.

Two solutions to this problem are possible considering Eq. 2–13. The feedrate could be specified as a function of the position spline parameter u. This method, while trivial, makes it impossible to accurately calculate the machining time *a priori* and the derivatives with respect to time cannot be extracted for the axis commands. The alternative solution is to construct the interpolated spline $U(t)$. This solution solves the problems of the former. It need only be C^1 continuous as will become apparent in the following derivation. Furthermore, the $U(t)$ solution leads to a parameterized solution for the tool-path which is invaluable tool in simulation and verification of tool-paths.

Given a set of f_i and l_i suppose a function $F(u)$ of the feedrate parameterized with the position spline parameter u is obtained. The feedrate itself is the change in the distance along the position spline with respect to time:

$$F(u) = \frac{du}{dt}. \tag{5–76}$$

To rewrite this equation as $u = U(t)$ proceed as follows. Integrate

$$t = \int_0^t dt = \int_0^u \frac{du}{F(u)} \tag{5-77}$$

If Eq. 5–77 can be inverted then $U(t)$ results.

The reciprocal of the quadratic equation, or parabola, can be integrated and then inverted to yield an analytic solution to Eq. 5–77. Since the quadratic equation

$$F_i(u) = g_{1_i} + g_{2_i} u + g_{3_i} u^2, \quad u \in [0, l_i] \tag{5-78}$$

has only three degrees of freedom, the end-points of a quadratic segment and only one tangent can be specified. Since the tangent at one end-point cannot be controlled, a common situation in which a smooth transition between regions of constant feedrate is necessary is not possible without violating the C^2 axis motion command constraint. However, by using two concatenated quadratic segments for each position spline segment a good solution is obtained.

Between two quadratic segments there are a total of six degrees of freedom. To maintain C^1 continuity at the junction two degrees of freedom are spent. Then the four remaining degrees of freedom allow both the position and tangent to be specified at either end of segment. This gives

$$F_{1_i}(u) = g_{1_i} + g_{2_i} u + g_{3_i} u^2, \quad u \in \left[0, \frac{l_i}{2}\right] \tag{5-79}$$

$$F_{2_i}(u) = g_{4_i} + g_{5_i} u + g_{6_i} u^2, \quad u \in \left[0, \frac{l_i}{2}\right] \tag{5-80}$$

constrained to

$$F_{1_i}(u)\Big|_{u=\frac{l_i}{2}} = F_{1_i}(u)\Big|_{u=0} \tag{5-81}$$

$$\frac{d}{du}F_{1_i}(u)\Big|_{u=\frac{l_i}{2}} = \frac{d}{du}F_{1_i}(u)\Big|_{u=0}. \tag{5-82}$$

Given f_i. f_{i+1}. f'_i. f'_{i+1} and l_i the coefficients g_{1_i}. g_{2_i}. g_{3_i}. g_{4_i}. g_{5_i} and g_{6_i} are obtained with

$$g_{1_i} = \frac{4(f_{i+1} - f_i) - l_i f'_{i+1} - 3l_i p'_i}{2l_i^2} \tag{5-83}$$

$$g_{2_i} = f'_i \tag{5-84}$$

$$g_{3_i} = f_i \tag{5-85}$$

$$g_{4_i} = \frac{f'_{i+1} - f'_i}{l_i} - g_{1_i} \tag{5-86}$$

$$g_{5_i} = l_i g_{1_i} + f'_i \tag{5-87}$$

$$g_{6_i} = \frac{l_i^2 g_{1_i}}{4} + \frac{l_i f'_i}{2} + f_i. \tag{5-88}$$

After reciprocating and integrating Eq. 5–79 the following results. Note that g_1. g_2 and g_3 respectively correspond to to either g_{1_i}, g_{2_i} and g_{3_i}. or g_{4_i}, g_{5_i} and g_{6_i}. For

clarity, the integration constant has been discarded.

$$U(t) = \begin{cases} g_1 t. & \text{if } g_2 = g_3 = 0 \\ \dfrac{g_1(\exp(g_2 t) - 1)}{g_2}. & \text{if } g_3 = 0 \\ \operatorname{sgn}(g_1 g_3)\sqrt{\dfrac{g_1}{g_3}}\tan(\sqrt{g_1 g_3}\,t). & \text{if } g_2 = 0 \\ g_2\left(\dfrac{1}{g_3(2 - g_2 t)} - \dfrac{1}{2g_3}\right). & \text{if } g_2^2 = 4g_1 g_3 \\ \dfrac{\sigma(\varrho + g_2) - g_2 + \varrho}{2g_3(1 - \sigma)}. & \text{if } g_2^2 > 4g_1 g_3 \\ \dfrac{\rho\tan(\frac{\varrho t}{2} + \arctan(g_2, \rho)) - g_2}{2g_3}. & \text{otherwise} \end{cases} \tag{5-89}$$

where

$$\varrho = \sqrt{g_2^2 - 4g_1 g_3},$$
$$\sigma = \exp\left(\varrho t + \ln\frac{g_2 - \varrho}{g_2 + \varrho}\right),$$
$$\rho = \sqrt{4g_1 g_3 - g_2^2}.$$

The first and second derivatives of these equations are given in Appendix B.

Due to the integration, the C^1 continuity of Eq. 5-79 and Eq. 5-80 becomes C^2 upon integration. Thus C^2 axis motion commands are assured.

In constructing the spline, the derivatives of the feedrate f_i' must be specified. These can estimated by interpolating a quadratic polynomial curve through three adjacent points and extracting the derivative of the curve at the middle point. The dynamic capability of the machine tool should be taken into account by not exceeding the maximum axis accelerations. Where there a region of variable feedrate abuts a region of constant feedrate a slope of zero should be specified.

By using a quadratic one-dimensional curve, to interpolate the feedrates, it is possible to ensure that between any two f_i, f_{i+1} the interpolated cannot exceed either f_i or f_{i+1} as long as the corresponding f_i', f_{i+1}' are both of the same sign, or one or both are zero [40].

The central drawback of this technique is that the f_i may never be zero. This also serves to ensure the feedrate can never become negative. However, since acceleration to and from a zero feedrate is never executed while in cut, a step input for starting and stopping will have no affect on the part.

5–7 SUMMARY

This chapter introduced a new scheme for tool-path and feedrate interpolation. The tool-path is interpolated by three curves. The position curve is based on the near arc-length parameterization method for three-axis machining but improves on the literature. The cutter axis orientation is interpolated by a spherical Bézier spline which is also near arc-length parameterized. Synchronous motion is achieved by reparameterizing the orientation spline with a monotonic rational quadratic spline. All three formulations are C^2 continuous and therefore the resulting axis motion commands must be C^2 continuous. A variable feedrate which maintains this level of continuity is obtained with a polynomial quadratic spline.

CHAPTER 6
Implementation

6–1 OFF-LINE IMPLEMENTATION

The off-line algorithms of Chapter 5 have been implemented in MATLAB®. Whereas for performance reasons, the real-time algorithms were implemented in ISO C++ (see Appendix F).

To verify the proposed algorithms two sets of representative data were chosen. A side milling tool-path shown in Fig. 6–1 based on a fan profile [104, 105] is the first example. The original fan profile was scaled by three, and then offset towards the centre of the fan profile by the tool radius 12.7 mm. The resulting spline was duplicated, translated in the positive z direction by 25.4 mm and rotated by 0.1745 rad (10 °). These two splines form a ruled surface. The tool-path was generated by offsetting points in a direction normal to the design surface to a distance equal to the tool radius. Because of symmetry, only one quarter of the fan profile was used. The resulting ruled surface tool-path is shown in Fig. 6–1 and data points in Table C–1.

The second example is typical of ballnose milling. Here a ballnose cutter of 10 mm

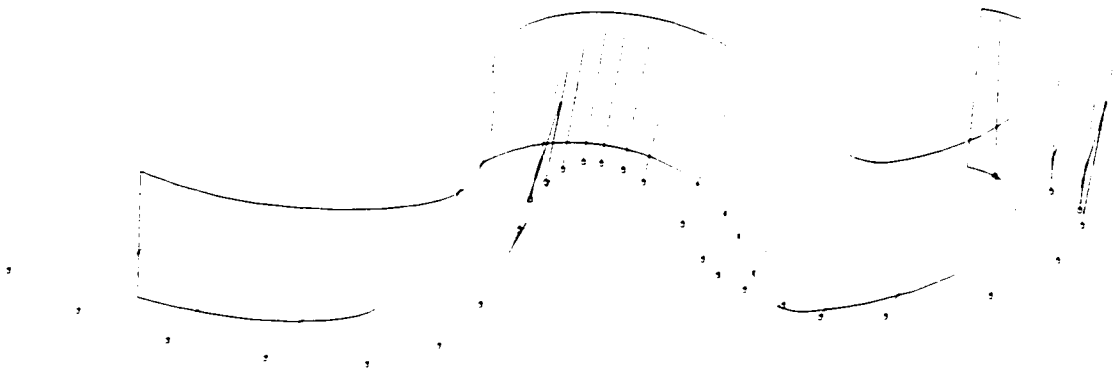


FIGURE 6–1 Side milling example. The closed wireframe is the portion of the design surface being milled. The line segments indicate tool orientation and circle at the ends of the line segments indicate tool position.

radius follows the CC point at an inclination of 20 ° to the sculptured surface

$$z = (100 \text{ mm}) \cos\left(2\pi \frac{x}{1000 \text{ mm}}\right) \cos\left(2\pi \frac{y}{1000 \text{ mm}}\right) \tag{6–1}$$

where

$$x \in [0, 1000 \text{ mm}]$$

and

$$y \in [0, 1000 \text{ mm}]$$

illustrated in Fig. 6–2. The 85 data points for the tool-path are given in Table C–2.

For the side and ballnose milling examples approximate calculation times on a 333 MHz Intel Architecture Pentium II® computer were 1 min and 10 min, respectively. Given the quantity of data, 25 and 88 points respectively, the execution times

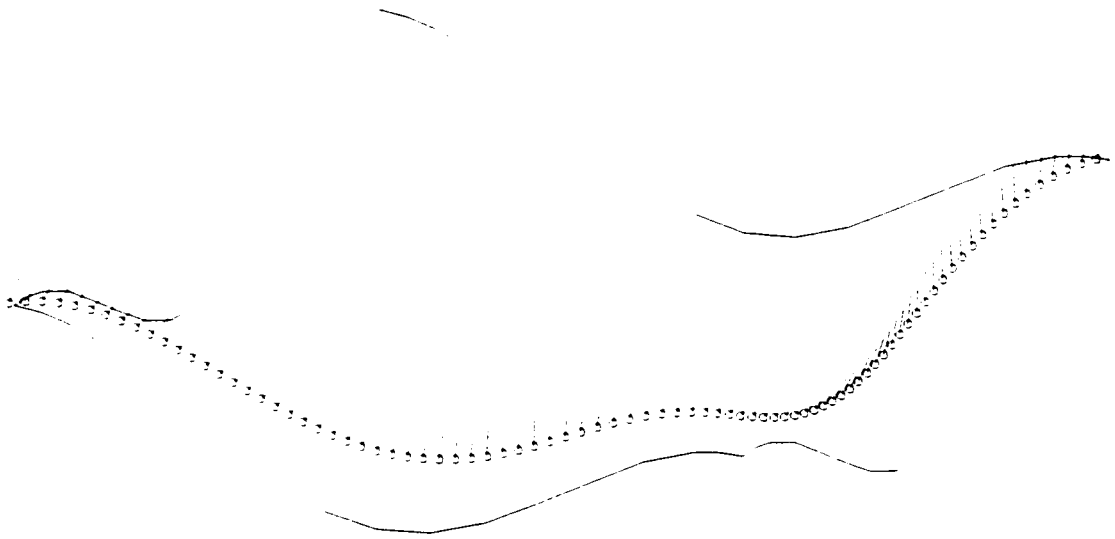


FIGURE 6–2 Ballnose milling example. The closed wireframe is the portion of the design surface being milled. The line segments indicate tool orientation and circle at the ends of the line segments indicate tool position.

would be impractical in a production environment. For long tool-paths thousands of points is not unusual and the above calculation times would therefore be inflated by orders of magnitude. Consequently, the algorithms would have to reimplemented in a more efficient manner such as the C programming language. Experience has shown that mathematical computations written in C will reduce calculation time by a factor or two or three orders of magnitude. For example, the example side milling tool-path was evaluated with the feedrate schedule shown in Fig. 7.12(a) at a servo update period of 885.4 ms. Using a MATLAB implementation the total execution time was 2 847 s while 2.422 s was recorded as the total execution time for the ISO C++ implementation of Appendix F. The compiled program executed at better than three orders of magnitude faster than the interpreted MATLAB scripts.

6–2 REAL-TIME IMPLEMENTATION

The real-time implementation of the proposed algorithms requires sampling of the tool-path, machine tool inverse kinematics and reference generation. To evaluate axis commands at a given instant in time, the interpolated tool-path parameter u must be related to time for constant feedrate with

$$U(t) = ft \tag{6–2}$$

and or for variable feedrate $U(t)$ is given by Eq. 5–89.

Then the equations for the motion of the cutter relative to the part reference frame at a constant feedrate are given by the following equations:

$$\mathbf{p} = \mathbf{P}_i(u) = \mathbf{P}_i(U(t)) \tag{6–3}$$

$$\mathbf{q} = \mathbf{Q}_i(V_i(u)) = \mathbf{Q}_i(V_i(U(t))). \tag{6–4}$$

Derivatives of these equations which are useful for simulations are given in Appendix D.

Given $\mathbf{p}$ and $\mathbf{q}$, the axis commands are calculated by solving the inverse kinematics of the particular machine tool configuration. This will include the workpiece displacement and cutter length constants. The problem of reference generation is one of unit conversion from millimetres or radians of axis displacement into encoder pulses.

FIGURE 6–3 A tilting-rotary table five-axis vertical machining centre.

6–2–1 INVERSE KINEMATICS

To demonstrate the proposed algorithms two machine tool kinematics configurations were chosen. The two are small variations of a tilting-rotary table five-axis vertical machining centre. One such machine is shown in the photograph Fig. 6–3 with the table tilted forward approximately 45 °.

The first variation on the tilting-rotary table theme is the situation when the table

is in the horizontal position. The equations are given by

$$d_x = p_x - c_x \tag{6-5}$$

$$d_y = p_y - c_y \tag{6-6}$$

$$d_z = p_z - c_z \tag{6-7}$$

where c_x, c_y and c_z are the distance of the part origin from the intersection of the axes of rotation of the two rotary axes. Then the axes commands are

$$a = \pm \arccos q_z \tag{6-8}$$

$$c = \begin{cases} \arctan(q_x, q_y) \\ \arctan(-q_x, -q_y) \end{cases} \tag{6-9}$$

$$x = d_x \cos c - d_y \sin c + c_x \tag{6-10}$$

$$y = d_x \cos a \sin c + d_y \cos a \cos c - d_z \sin a + c_y \tag{6-11}$$

$$z = d_x \sin a \sin c + d_y \sin a \cos c + d_z \cos a + c_z \tag{6-12}$$

where a and c are the rotary axes and x, y and z are the translational axes. These equations are easily obtained by performing a kinematics analysis using HTMs. To reduce computation time a and c are substituted into Eq. 6-10, 6-11 and 6-12 to

obtain

$$x = \frac{d_y q_x - d_x q_y}{(1 - q_z^2)^{\frac{1}{2}}} + c_x \tag{6-13}$$

$$y = \frac{d_z(1 - q_z^2) - q_z(d_x q_x + d_y q_y)}{(1 - q_z^2)^{\frac{1}{2}}} + c_y \tag{6-14}$$

$$z = d_x q_x + d_y q_y + d_z q_z + c_z \tag{6-15}$$

for

$$a = -\arccos q_z \tag{6-16}$$

$$c = \arctan(-q_x, -q_y). \tag{6-17}$$

See Appendix E for derivatives of these equations.

The centre of rotation $(c_x\, c_y\, c_z)^T$ is $(0\,0\,0)^T$. For the simulations to follow the tool-path is offset by the vector $(0\,0\,140.8417\text{ mm})^T$ from the centre of rotation.

When the table tilted to be vertical a different set equations is obtained

$$a = \arctan(-q_y, q_z) \tag{6-18}$$

$$b = \arcsin q_x \tag{6-19}$$

$$x = d_x \cos b - d_z \sin b + c_x \tag{6-20}$$

$$y = d_x \sin a \sin b + d_y \cos a + d_z \sin a \cos b + c_y \tag{6-21}$$

$$z = d_x \cos a \sin b - d_y \sin a + d_z \cos a \cos b + c_z \tag{6-22}$$

where a and b are the rotary axes and x, y and z are the translational axes. See Appendix E for derivatives of these equations.

The centre of rotation $(c_x\, c_y\, c_z)^T$ is $(0\,0\,0)^T$. For the simulations to follow the tool-path is offset by the vector $(0\,0\,304.8\text{ mm})^T$ from the centre of rotation.

6–3 EXECUTION TIME

The deterministic requirement of completing the above computation in a fixed time less than Δt must be considered. To do so, these algorithms were implemented on a Delta-Tau PMAC OAC controller residing in the PCI slot of an Intel Architecture Pentium 120 MHz computer. The computations were performed by the hardware Floating Point Unit (FPU) of the Pentium processor every servo update period of 885.4 ms. It was found that the particular configuration of the computer was sufficient to complete the necessary computations for each servo update period.

A further benchmark was run with an Intel Architecture Pentium II 333 MHz processor using the source code in Appendix F. The tool-path interpolation algorithm with constant feedrate and the tilting-rotary table five axis machine tool with the table in the horizontal position were implemented. The operating system timer granularity was limited to 1 ms which was too coarse for measuring the Central Processing Unit (CPU) time needed to compute the axis commands for a single servo update. Instead, all the computations at each servo update period were repeated 1000 times to obtain an average time for each servo update. It was found that on average over all samples, the computations required 3.357 μs with a worst computation time of 5.1 μs for any given sample. This worst time is only 5.76% of the servo update period. This leaves ample time for the other computations necessary to close the control loop during each sample.

At the time of the writing of this dissertation CPUs in excess of 1 GHz have become available with improved bus architectures and FPUs. It can be surmised that the computation time available for real-time control will only increase.

CHAPTER 7
Simulations and Experiments

To test and verify the proposed algorithms, two sample tool-paths have been chosen. Introduced in the previous chapter, the side milling example (Fig. 6–1 and Table C–1) and ballnose milling example (Fig. 6–2 and Table C–2), are representative of common five-axis milling situations.

In Chapter 5 the near arc-length parameterization and real-time arc-length prediction algorithms were compared (see Fig. 5–1). For reasons of determinism it was shown that the near arc-length method is more suitable for the automation of CNC machine tool controls. Then, an improvement in the near arc-length parameterization methods was demonstrated over existing algorithms (see Fig. 5–5). Angular acceleration is reduced by the proposed algorithm as demonstrated in Fig. 5–10 which has the effect of reducing oscillation of the effective feedrate of Eq. 2–5 when $e \neq 0$.

In this chapter, the claims of curve smoothness will be demonstrated as well as the utility of the smoothly variable feedrate for the side milling and ballnose milling tool-paths and for the two kinematic configurations developed in the last chapter.

7–1 SIDE MILLING EXAMPLE

Figure 7–1 depicts the interpolated position and orientation components of the side milling tool-path. Note the dots on the curves indicate the interpolatory data of Table C–1. The interpolatory points are most dense where the curvature of the component curves is greatest. Since the deviation of the curve from unit speed is greatest in areas of largest curvature, more points were chosen in these regions to reduce parameterization error. It is likely that under some conditions the position data and orientation data may require higher densities of points in different areas of the tool-path. It does not follow that the additional points need be added to both the position curve and orientation curve. If additional data is added to only one curve then these points are ignored when constructing the reparameterization spline and their parameter ranges are summed. Adding points to only one curve conserves transmission bandwidth and storage capacity. Note that the parameter range of u is approximately 380 mm which is also an approximation of its arc-length. Similarly, the orientation curve has a total length of approximately 2.16639.

Parameterization error of both curves is shown in Fig. 7–2. Parameterization error is measured as $|\|\mathbf{p}'\| - 1|$ and $|\|\mathbf{q}'\| - 1|$ for position and orientation, respectively. Parameterization errors of less than 0.15% and 0.04% respectively have been achieved.

Figure 7–3 illustrates the parameterization curve generated to synchronize the position and orientation splines of the side milling tool-path. The dots in Fig. 7.3(a) are the interpolatory data based on the segment lengths of the position and orientation curves. In fact, the curve is very nearly a straight line with only small variation in some small deviations clearly seen in Fig. 7.3(b) and Fig. 7.3(c). Note that the curve appears C^1 continuous in Fig. 7.3(b) and C^0 continuous in Fig. 7.3(c).

The element of time is introduced with a constant feedrate of 400 mm min^{-1} in

Fig. 7-4 and Fig. 7-5. The near arc-length parameterization algorithm demonstrates its effectiveness with a range of feedrate within 2 mm min^{-1}. However, the angular speed is not constant due to the effect of the reparameterization curve and therefore mimics the shape of the first derivative of the reparameterization curve in Fig. 7.3(b). Again, both the feedrate and angular speed curves appear C^1 continuous. For both feedrate and angular speed a Fast Fourier Transform (FFT) was performed. It is critical to the sound operation of the machine and good surface finish that the tool-path does not excite the natural frequencies of the machine tool. As can be seen in Fig. 7.4(c) and Fig. 7.23(b) the majority of the frequency components are well below 10 Hz which is acceptable since the natural frequency of the average machine tool is in the range of 200 Hz to 300 Hz [93]. In actual practice the limited digital to analogue conversion resolution, and bandwidth of the axes servo systems, will further attenuate vibrations.

Figure 7-6 and Fig. 7-7 illustrate the tool-path components and first and second derivatives. This demonstrates the C^1 continuity at the first derivative and C^0 continuity at the second derivative which infers C^2 continuity of the position and orientation splines. Note that the position acceleration is very small and corresponds well with the results from previous researchers.

With inverse kinematics these components are transformed into the data plotted in Fig. 7-8, Fig. 7-9, Fig. 7-10 and Fig. 7-11. These pairs of plots correspond to the inverse kinematics equations compiled in the last chapter in sequence: tilting-rotary table machine tool with table in the horizontal position and the tilting-rotary table machine tool with the table in the vertical position. From the plots it is ascertained that the C^2 continuity of the axes commands is maintained. However, there is a significant variation in the velocities and accelerations of the various kinematics con-

figurations. The tilting-rotary five-axis machine tool with its table in the horizontal position has a particularly large acceleration of the c axis in comparison to the b axis when the home position of the table is vertical. This must be due to the kinematics singularity of this particular configuration. The kinematic singularity occurs when the c axis is aligned with the spindle. Hence the c axis is made redundant and a singularity occurs at this point. Suppose that the cutter is reoriented in a path which travels the geodesic curve between two unit vectors on the sphere and passes through $\mathbf{q} = (0\,0\,1)^T$ which is the point of singularity. According to the equations, the machine tool would then be instructed to tilt the a axis until the singular point was reached. At the singular point the c axis would be commanded to rotate 180 ° instantaneously. This is impossible in terms of machine tool dynamics and can only result in very large geometric error and poor surface finish of the workpiece. If this same tool-path was adjusted such that it passed near but not through $\mathbf{q} = (0\,0\,1)^T$ the initial large velocities and accelerations of c axis would be reduced proportional to the proximity of the tool-path to the singularity. These simulations are a valuable tool in determining tool-path feasibility due to actuator saturation.

To eliminate actuator saturation the feedrate can be reduced. For example, in Fig. 7–9(b) the c axis can be limited to 0.1 $\mathrm{rad\,s^{-1}}$ by lowering the feedrate in the region of the excess velocity. This effect is shown in Fig. 7–12, Fig. 7–13 and Fig. 7–14. A side effect of the reduction in the feedrate is commensurate increase in the acceleration. The feedrate must vary over a larger portion of the tool-path to excite smaller accelerations. Nevertheless, by employing the proposed variable feedrate interpolation algorithm, the smoothness of the axes commands is preserved.

Finally, a machining test of the side milling tool-path was performed with a 25.4 mm diameter High Speed Steel (HSS) flat end mill cutter. The stock was a

plate of 25.4 mm thick aluminum. Results are displayed in Fig. 7-15. To test the geometric veracity of the part a Coordinate Measuring Machine (CMM) as depicted in Fig. 7-16 was used to measure the machined workpiece (Fig. 7-17). An HS-45C laser scanning camera from Hymarc Inc.[1] measured the part and the assembled data was compared to the design surface. Figure 7-18 shows a plot of the dimensional error in millimetres, compared to the CAD design surface, and generated using SDRC Verdict v10.0. During fitting this software averages the error, producing the ±0.5 mm display shown. If the error scale is shifted by +0.5 mm, then nearly all of the machined surface will have a very small dimensional error. There remains only the +1.0 mm error concentrated near 36 s (Fig. 7-9). This is attributed to the inability of the machine tool rotary axes to follow the high commanded acceleration at this tool-path location. It is expected that, by using the proposed varying feedrate (Fig. 7-12), this error would be substantially reduced.

7-2 BALLNOSE MILLING EXAMPLE

For clarity the ballnose milling example simulations are arranged in the same order as the side milling simulation plots. The results and conclusions drawn from the ballnose milling example are virtually identical to that of the side milling tool-path. The ballnose tool-path demonstrates a large rotation of the cutter but does so over a greater distance: approximately 1.58 m.

Compare Fig. 7-1 and Fig. 7-19. The larger number of points and larger distance of the ballnose tool-path means the parameterization error is an order of magnitude less than in the previous examples. However, it appears the second derivative of the orientation components in Fig. 7.29(c) oscillate as if the interpolation algorithm was

[1]Hymarc 3-D Systems, Nepean, Ontario, Canada

unable to solve the non-linear equations. Nevertheless, the graph demonstrates that C^2 continuity is preserved.

The ballnose tool-path feedrate error of approximately ± 0.5 μm min^{-1} is insignificant. Consider that a typical axis BLU is 2.54 μm and servo update period is 0.001 s. From Fig. 7.22(a), the feedrate of 400 mm min^{-1} results in a travel of 6.67 μm by the axis during one servo update. If there is an error of 0.5 μm min^{-1} in the feedrate this could translate into a distance error as great as 8.33 pm. This number is considerably less than the BLU of the axis and therefore insignificant.

In terms of the machine tool kinematics, the tilting-rotary table in the horizontal position exhibits significantly greater velocities and accelerations than with the table in the vertical position. It must be noted that the table in the vertical position creates a fixturing problem in that the workpiece must be mounted such that its z direction is normal to the c axis of the table.

7–3 SUMMARY

This chapter demonstrated by machining experiment and simulation the proposed algorithms. Two different machine tool kinematics configurations were employed for peripheral milling and ballnose milling tool-paths. The machining test demonstrated the correctness of the algorithm. C^2 continuity is maintained in all the simulation results. Velocities and accelerations compared well with previous work in three-axis command generation and were significantly less than those obtained by linear interpolators.

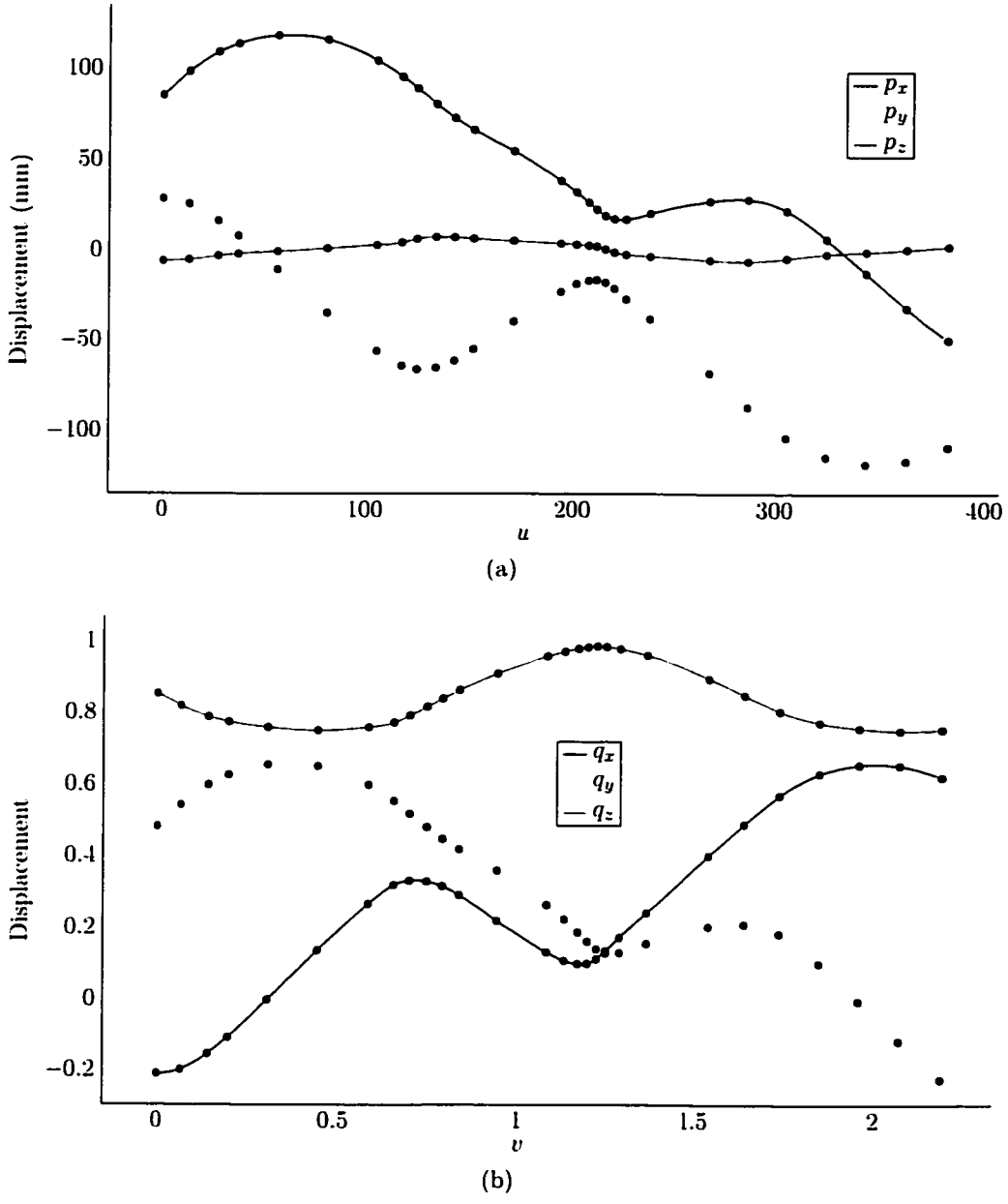


FIGURE 7–1 Plot of simulated (a) position and (b) orientation curve components interpolating the side milling tool-path.

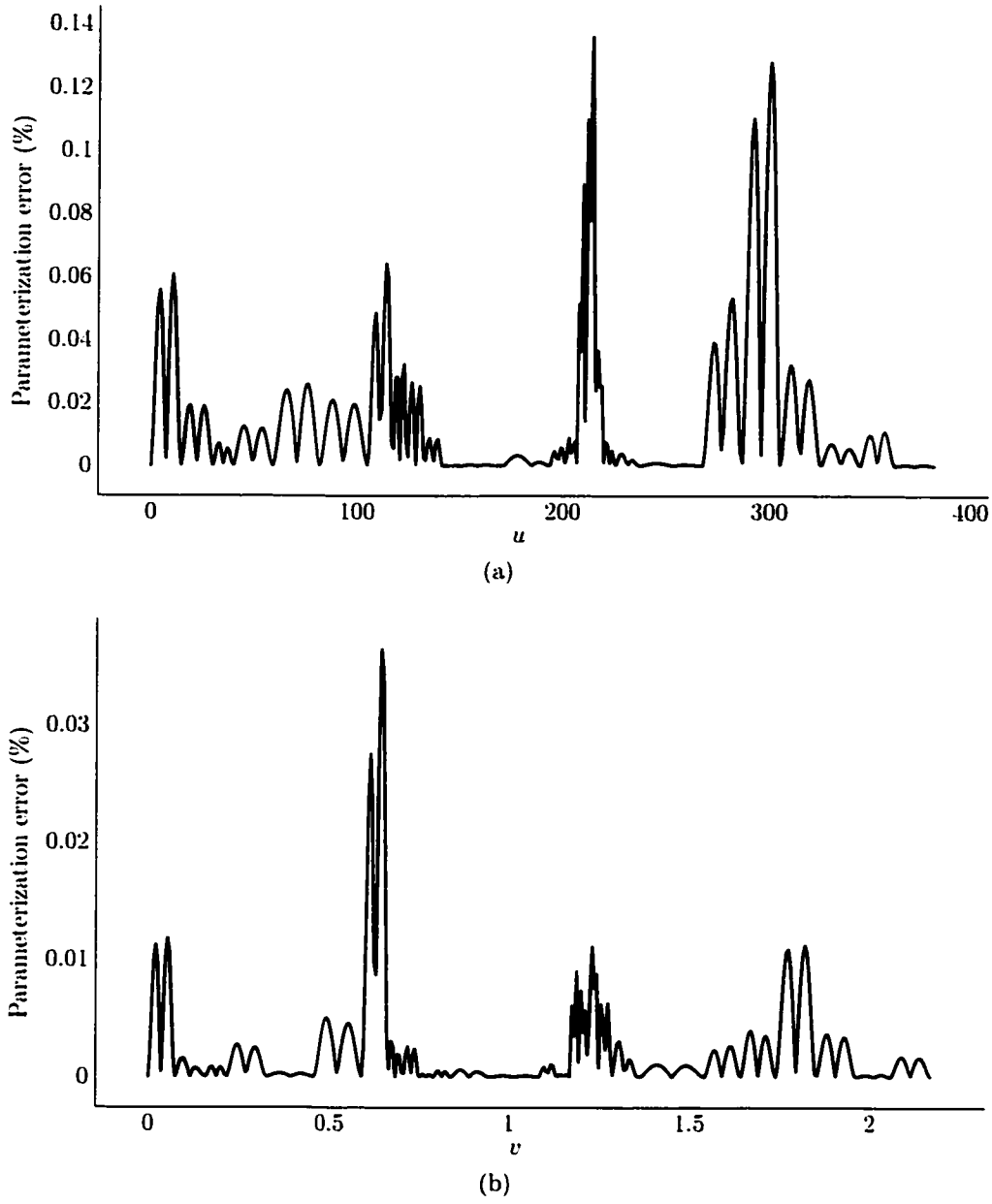


FIGURE 7–2 Plot of parameterization error of the (a) position and (b) orientation curves for the side milling tool-path.

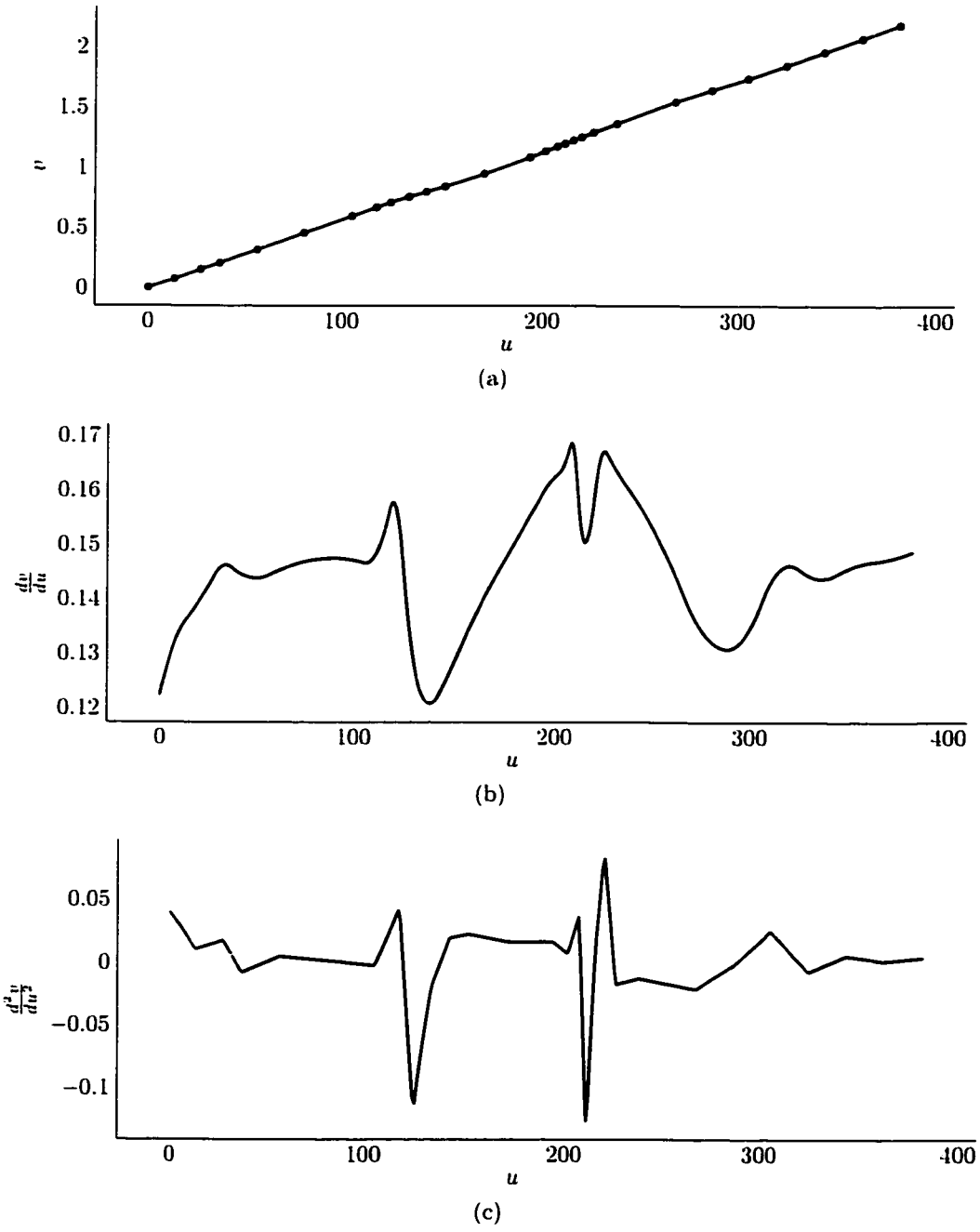


FIGURE 7–3 Plot of (a) reparameterization curve, (b) first and (c) second derivatives interpolating the side milling tool-path.

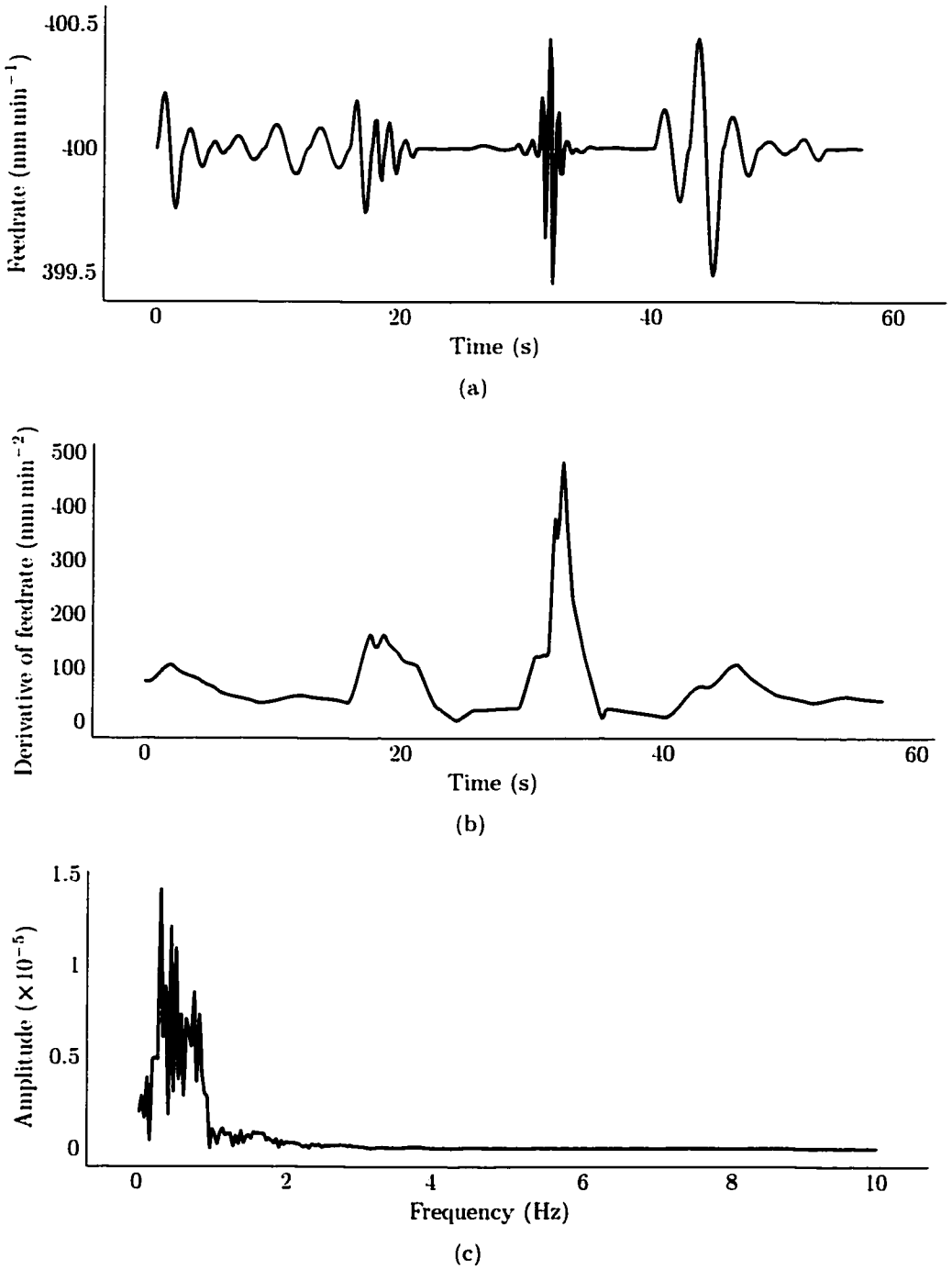


FIGURE 7–4 Plot of simulated (a) feedrate, (b) derivative and (c) FFT for side milling tool-path.

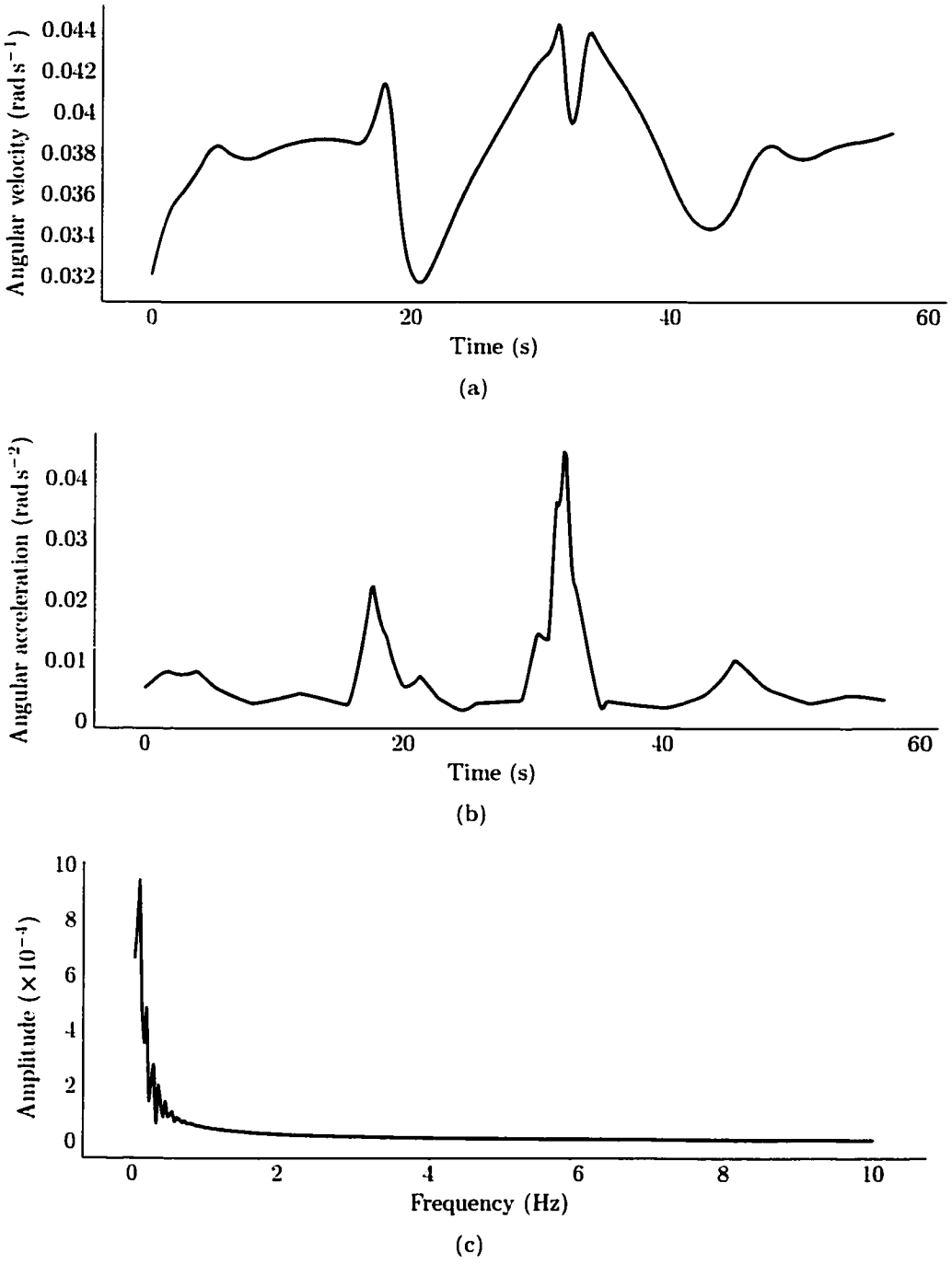


FIGURE 7–5 Plot of simulated (a) angular speed, (b) derivative and (c) FFT for side milling tool-path.

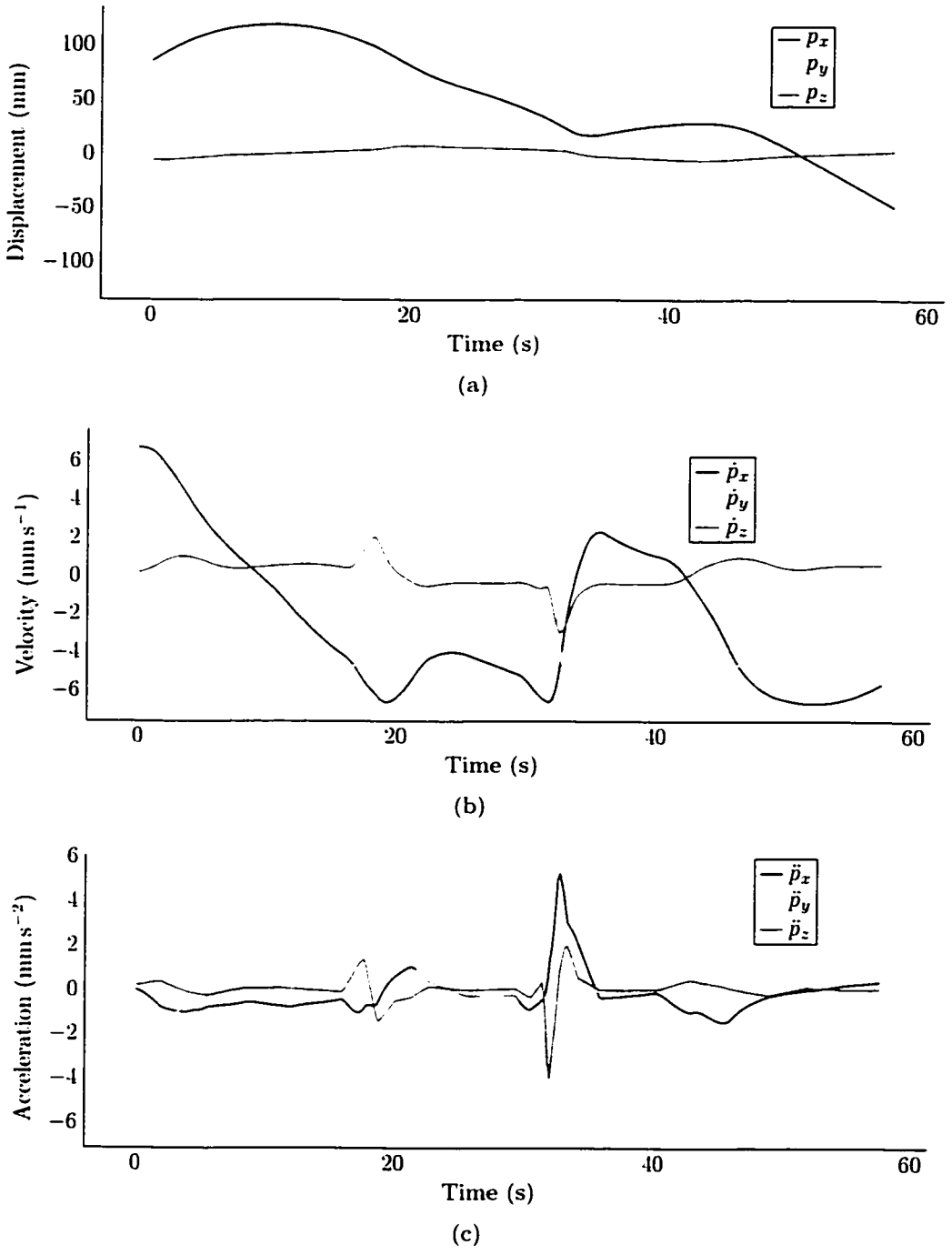


FIGURE 7–6 Plot of simulated (a) position curve components, (b) first and (c) second derivatives with respect to time interpolating the side milling tool-path at the feedrate 400 mm min^{-1}.

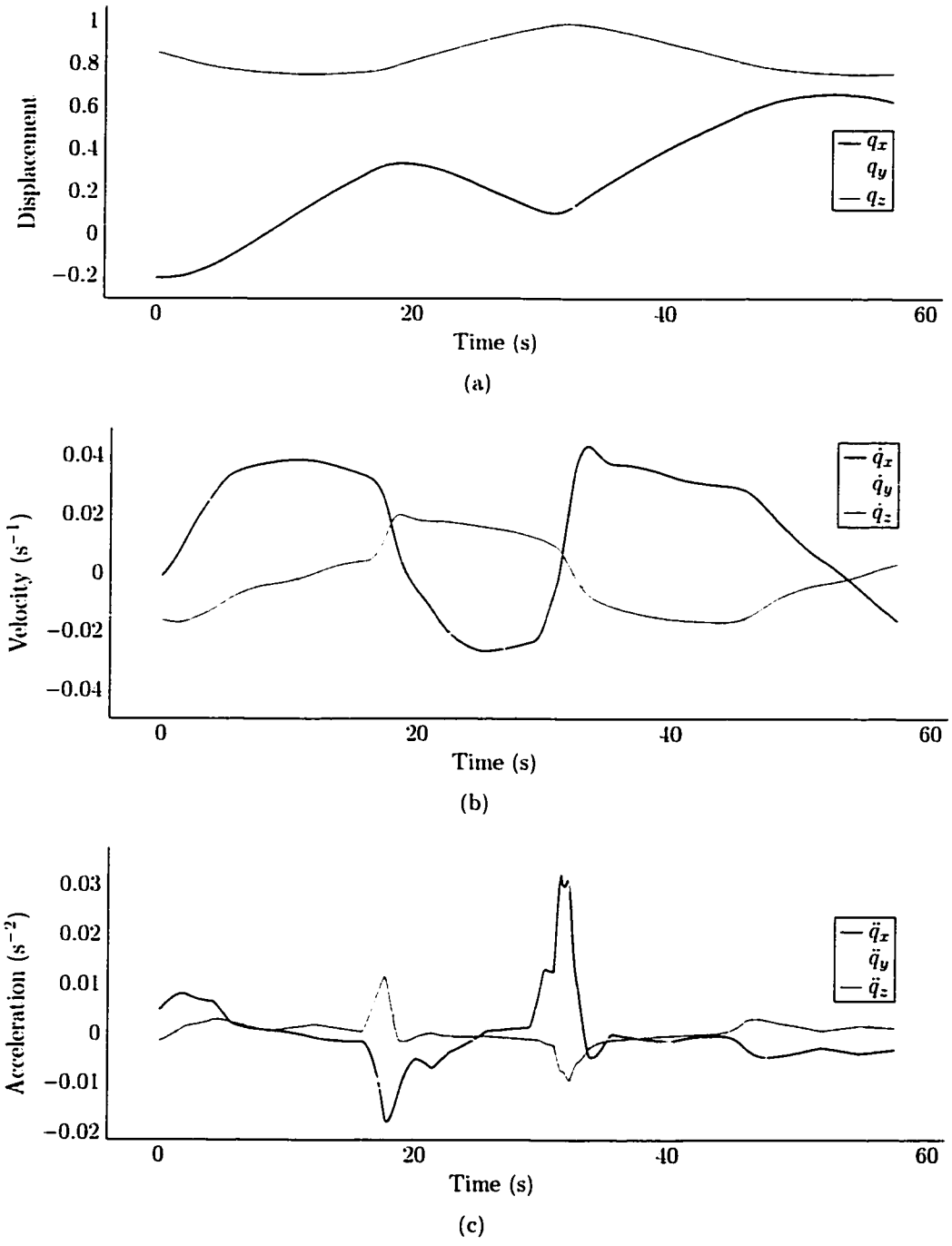


FIGURE 7–7 Plot of simulated (a) orientation curve components, (b) first and (c) second derivatives interpolating the side milling tool-path at the feedrate 400 mm min^{-1}.

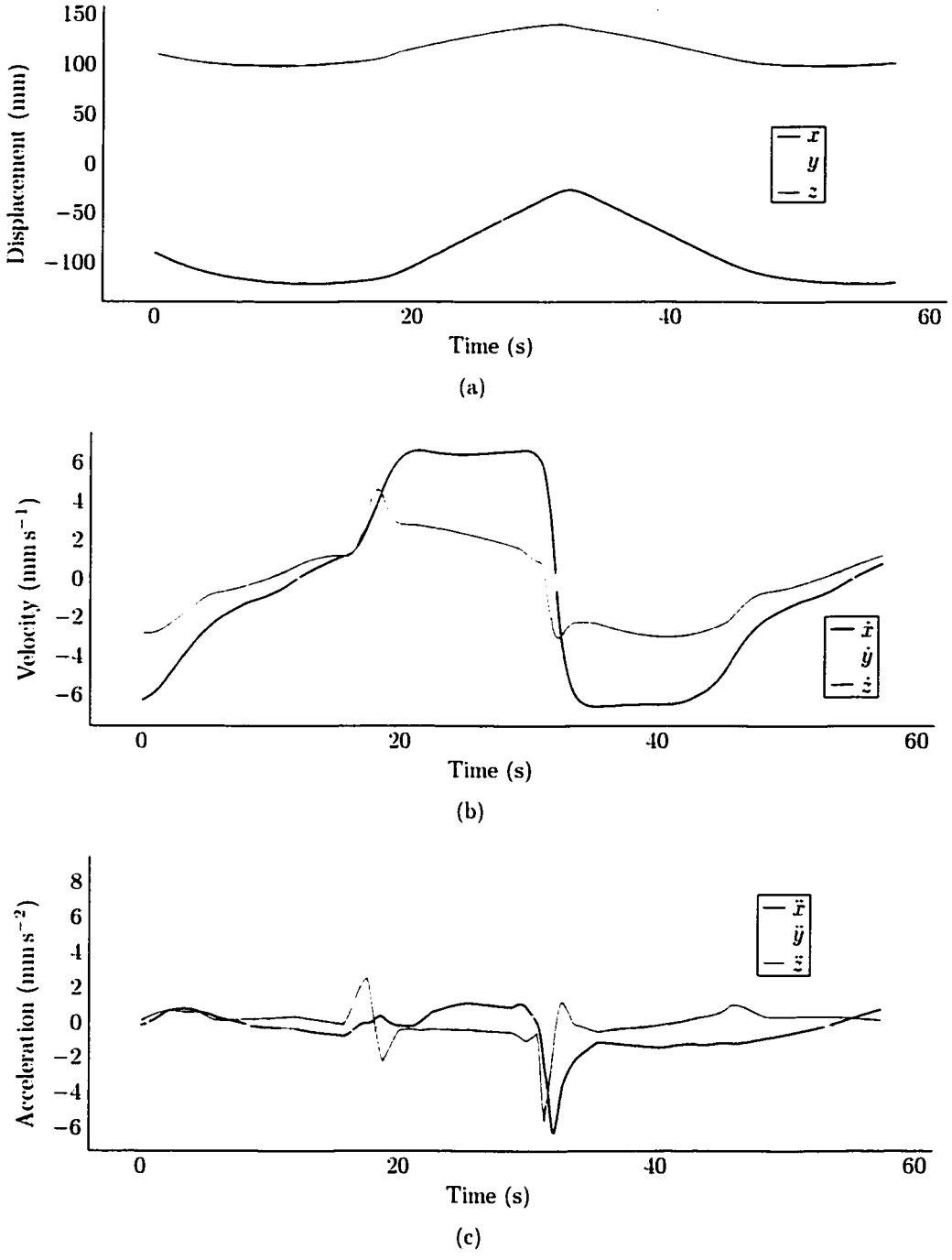


FIGURE 7–8 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and constant feedrate.

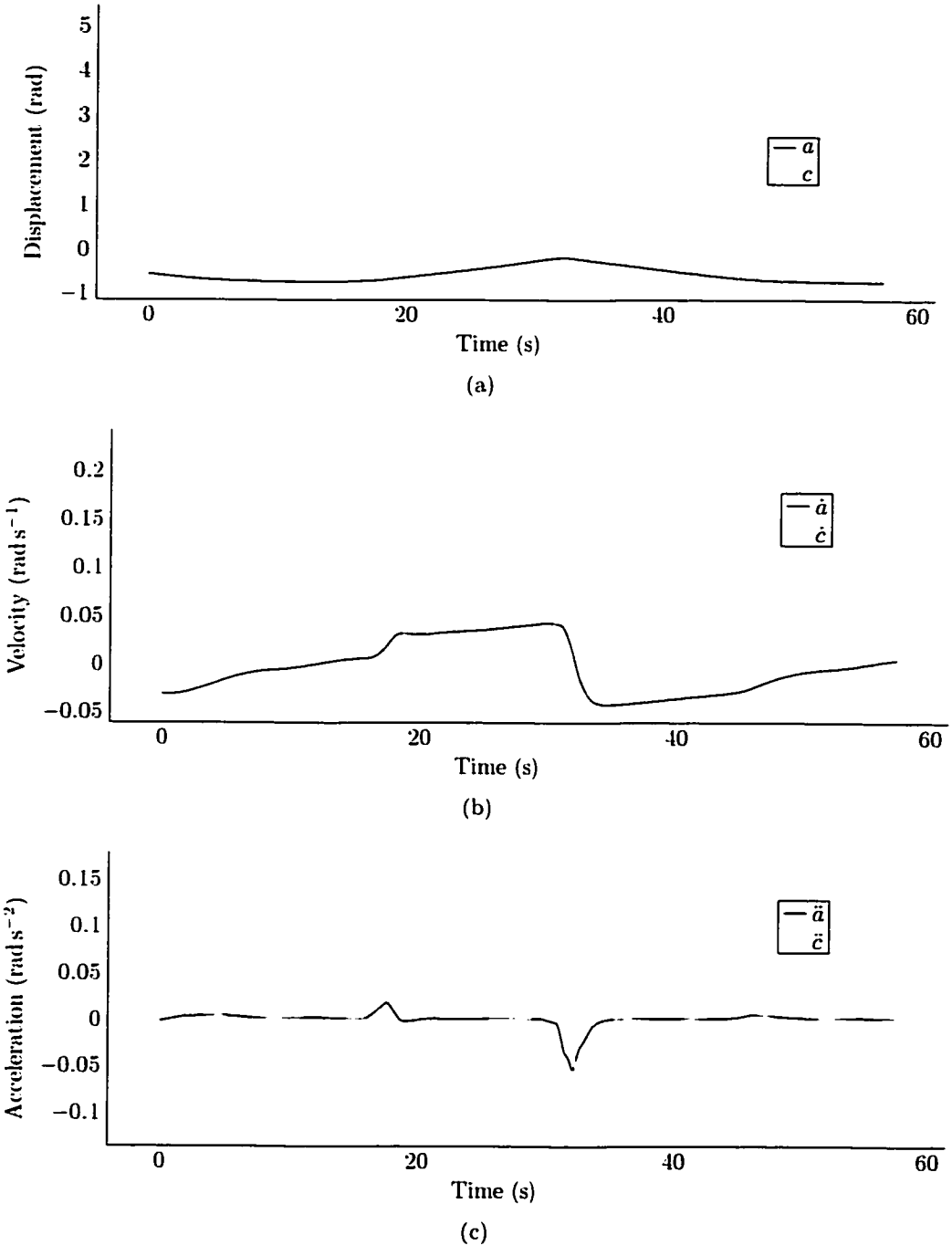


FIGURE 7–9 Plot of simulated (a) rotary axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and constant feedrate.

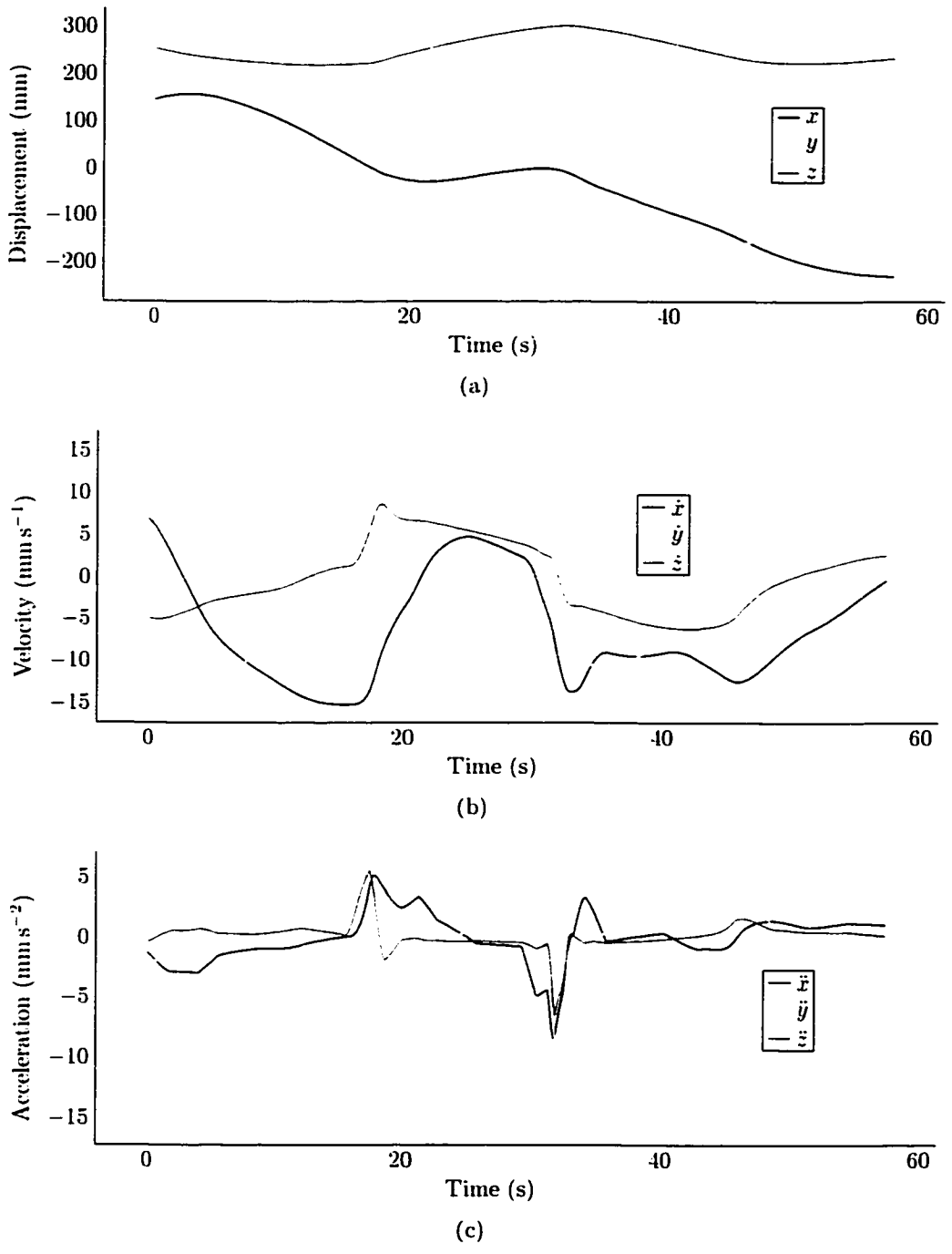


FIGURE 7–10 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the side milling tool-path and constant feedrate.

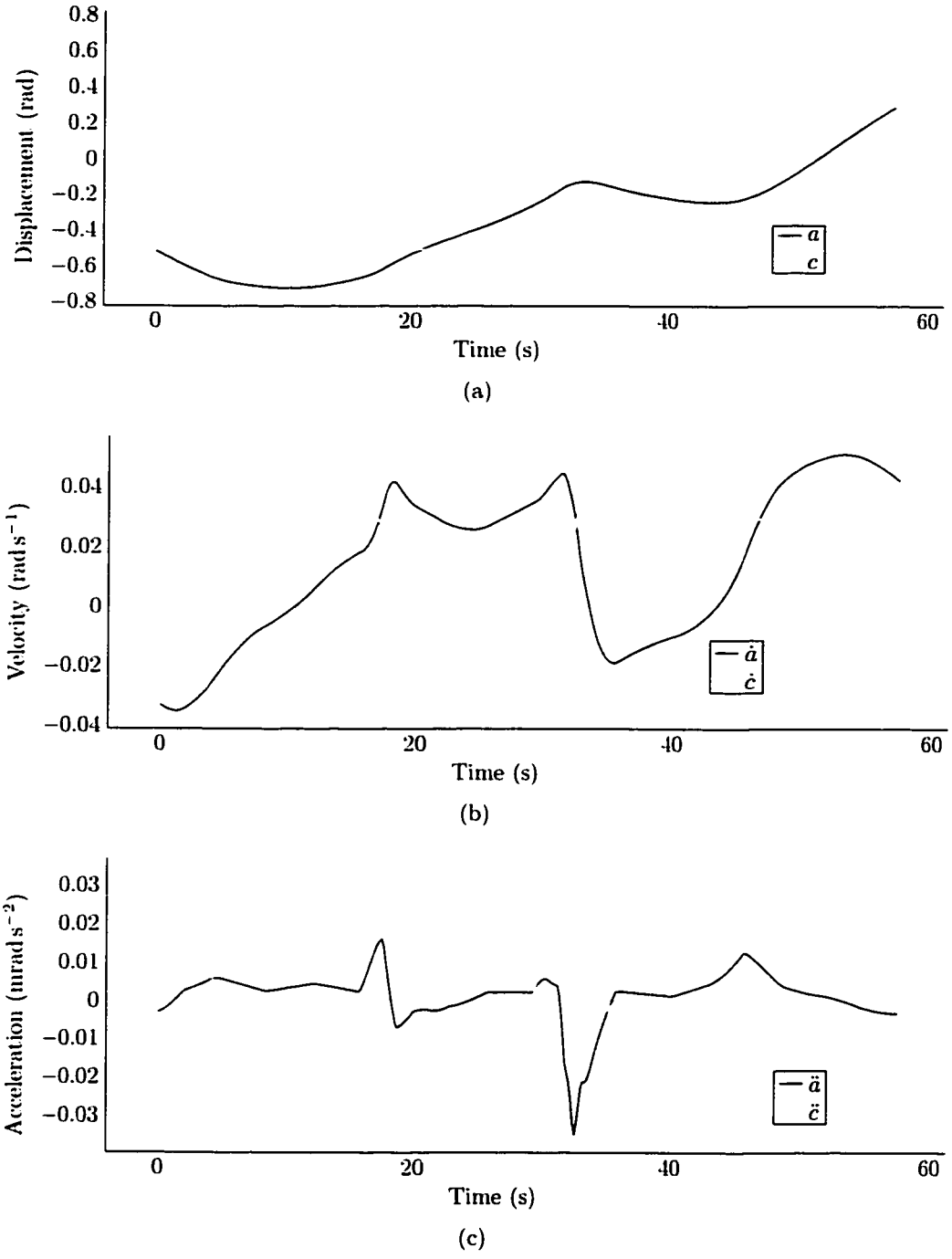


FIGURE 7–11 Plot of simulated (a) rotary axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the side milling tool-path and constant feedrate.

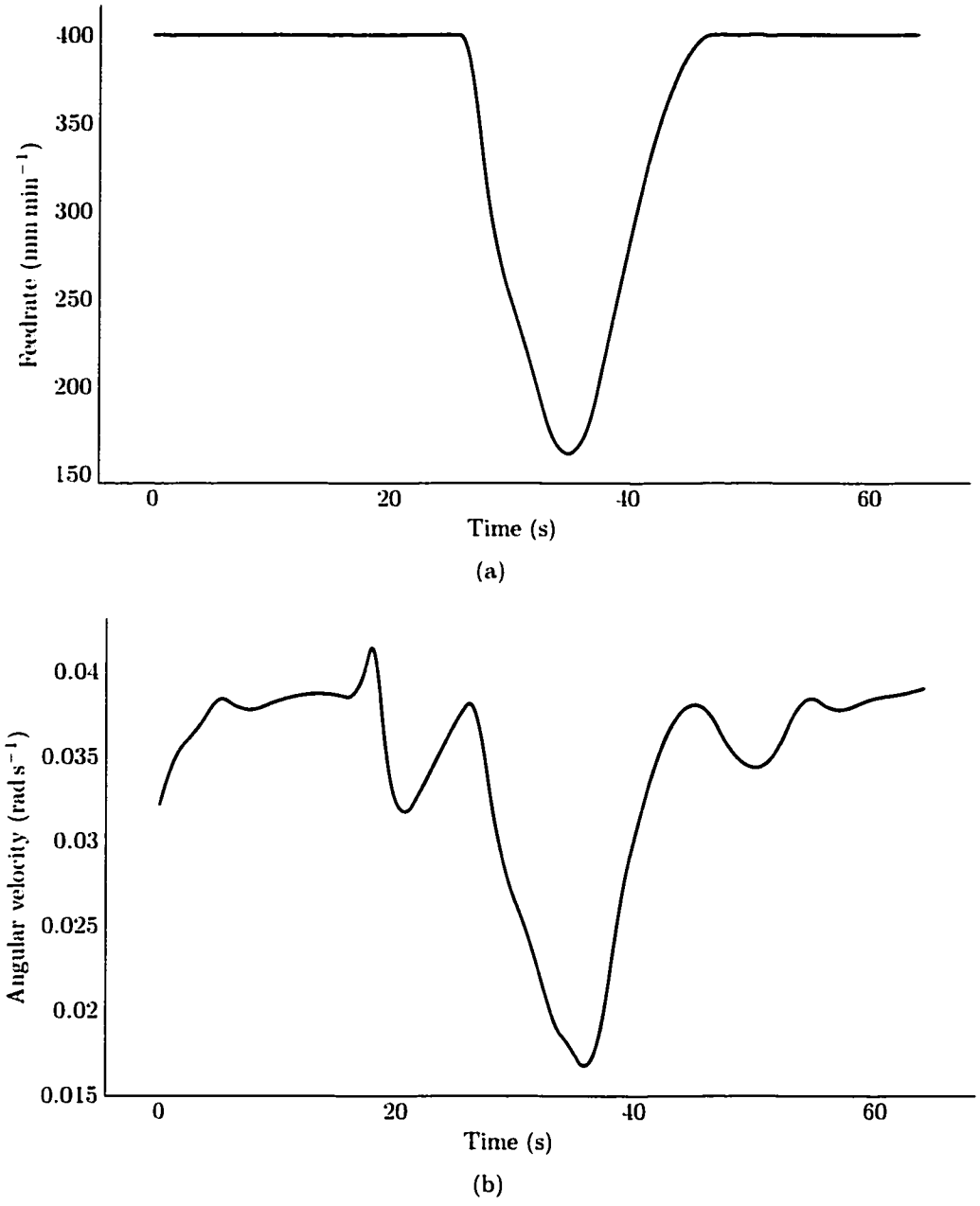


FIGURE 7–12 Plot of simulated (a) feedrate and (b) angular velocity for side milling tool-path with limited axis velocity.

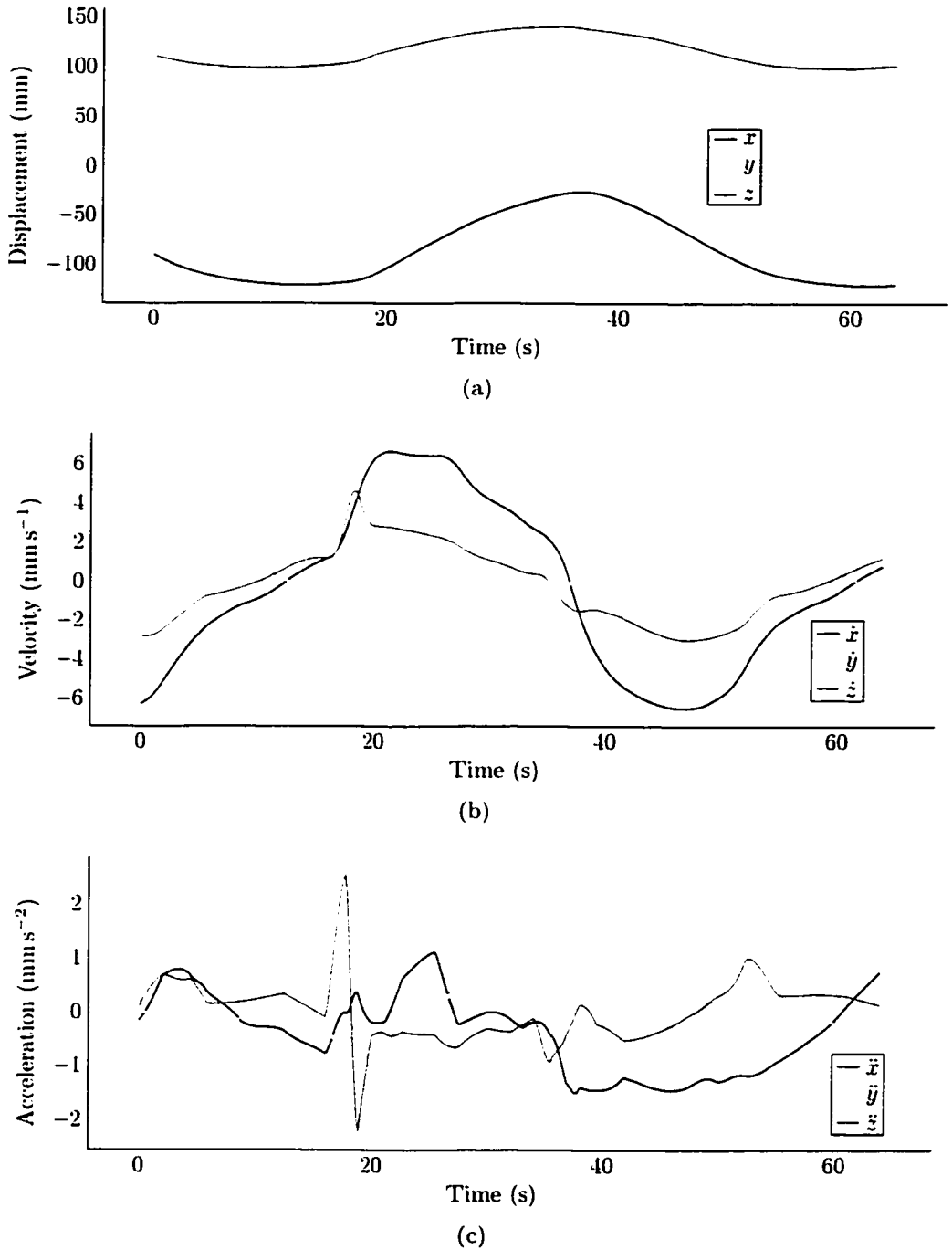


FIGURE 7–13 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and limited axis velocity.

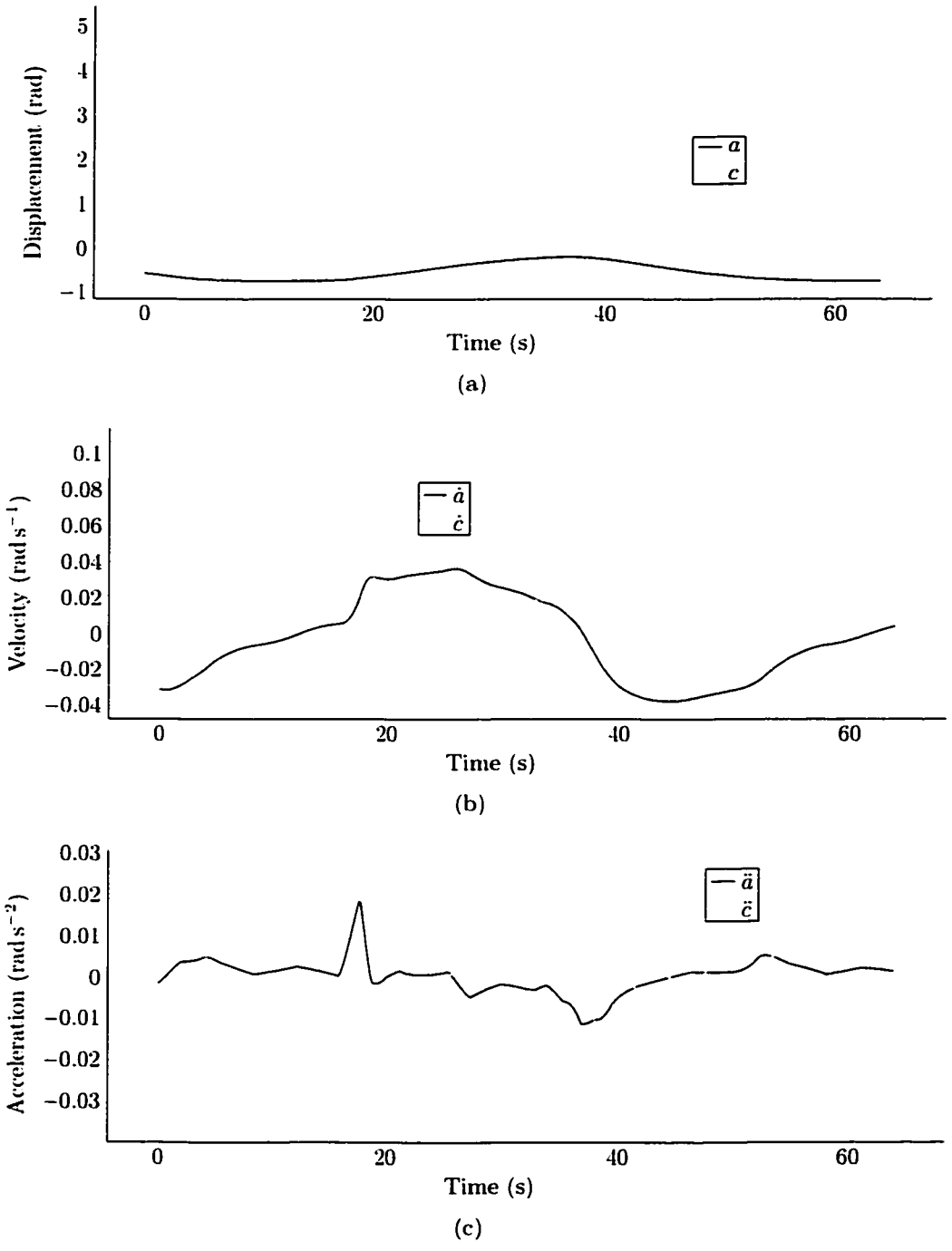
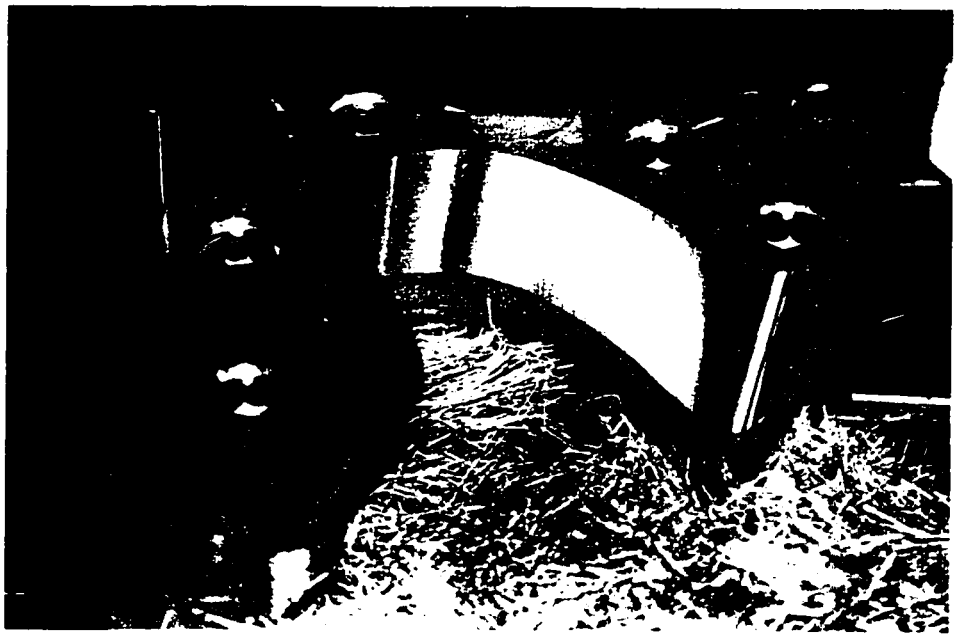


FIGURE 7–14 Plot of simulated (a) rotary axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the side milling tool-path and limited axis velocity.

(a)



(b)

FIGURE 7–15 Photographs of side milling example. The cutter in the right-hand side of the photograph (a) is 25.4 mm in diameter.

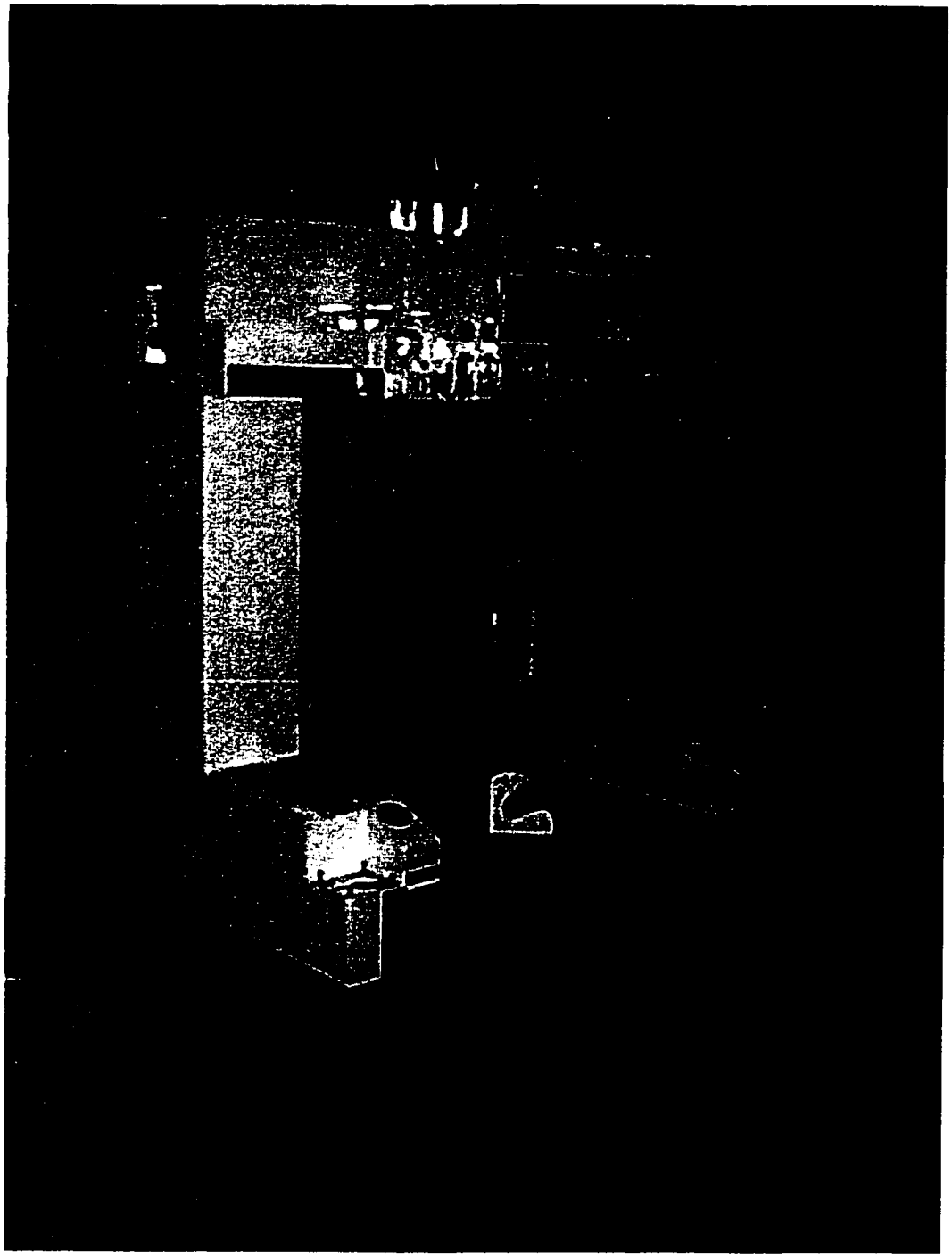


FIGURE 7–16 Photograph of a gantry type CMM.

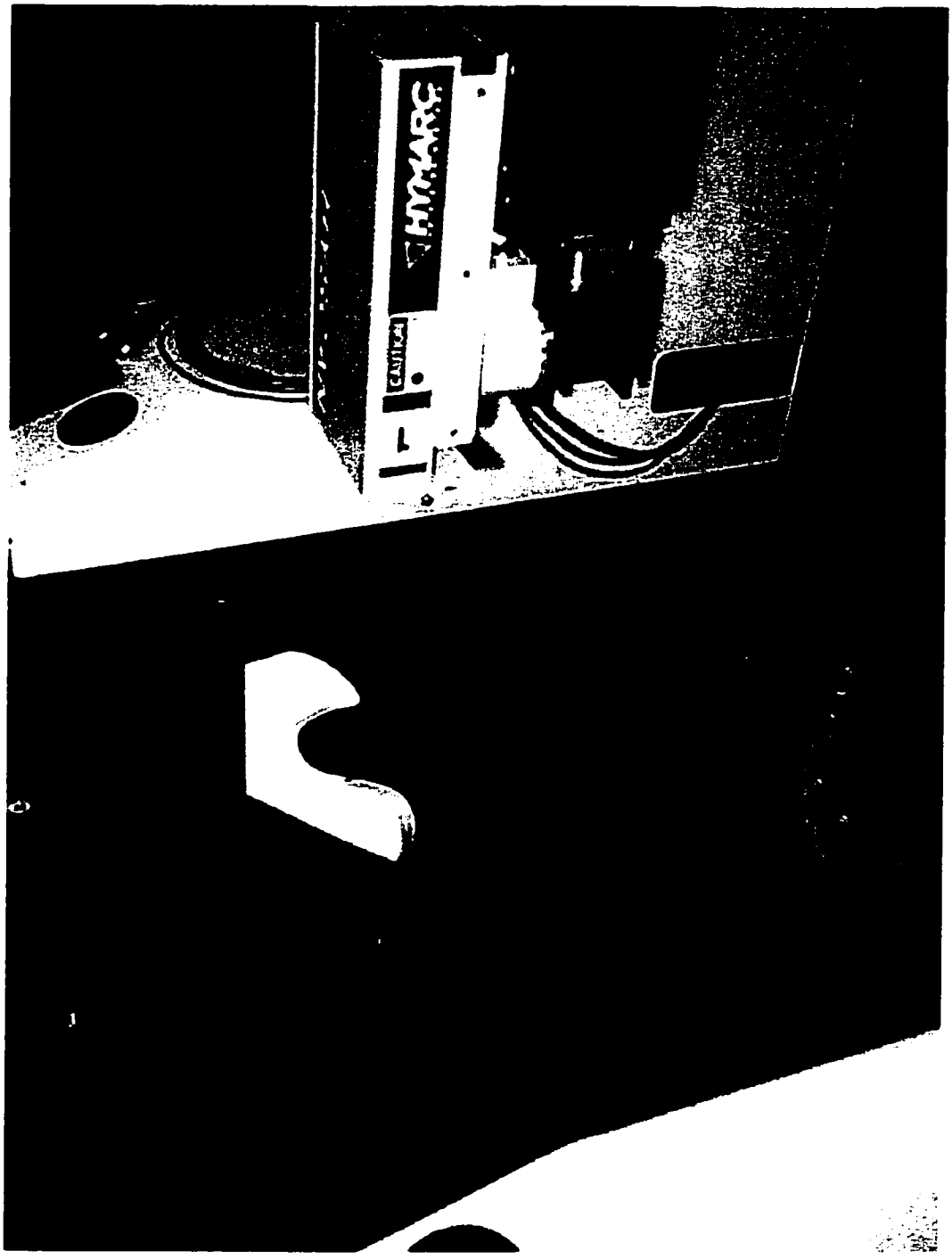


FIGURE 7–17 Photograph of a side milling tool-path part being measure with a scanning laser probe on a CMM.

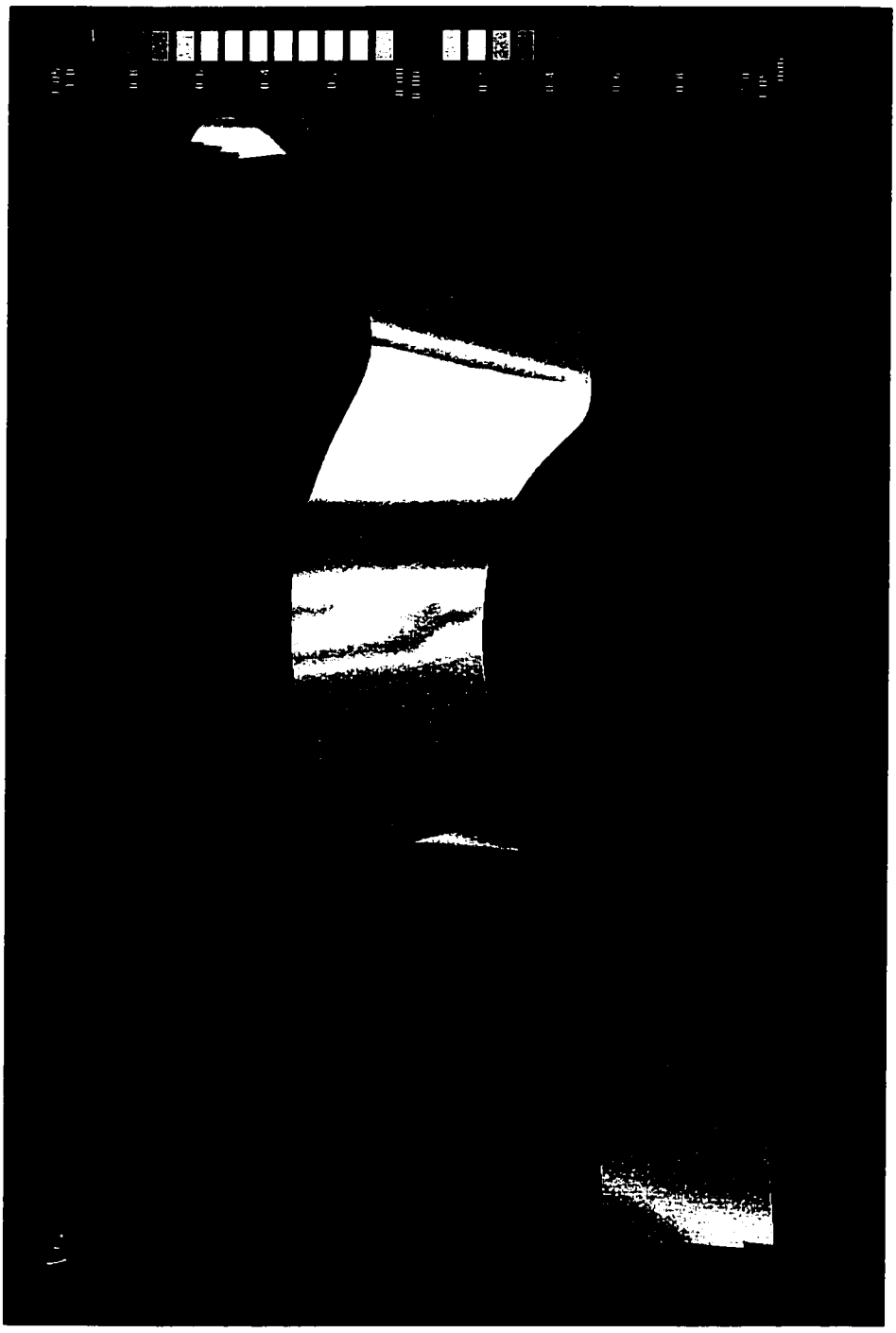


FIGURE 7–18 Colour plot of dimensional error.

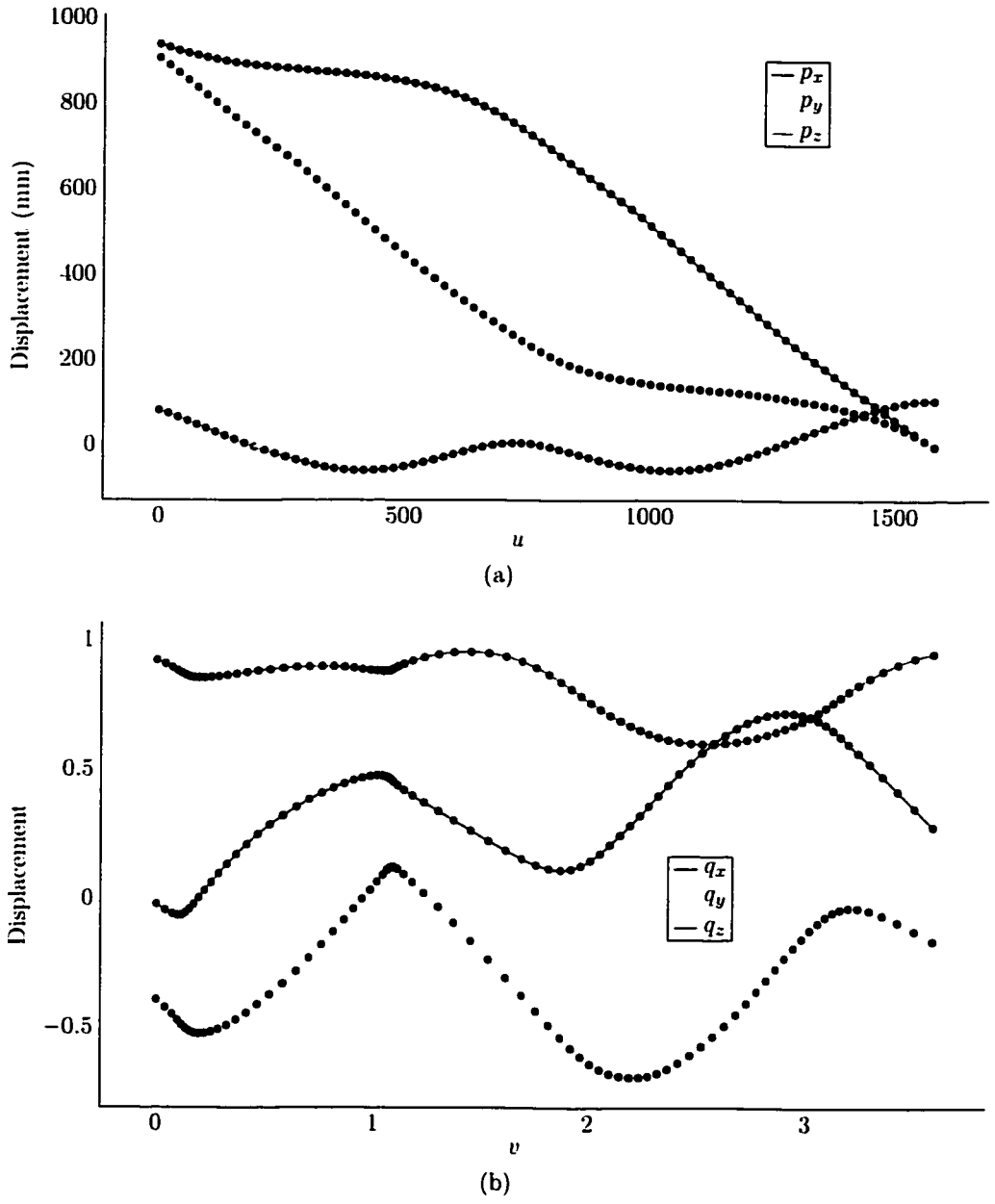


FIGURE 7–19 Plot of simulated (a) position and (b) orientation curve components interpolating the ballnose milling tool-path.

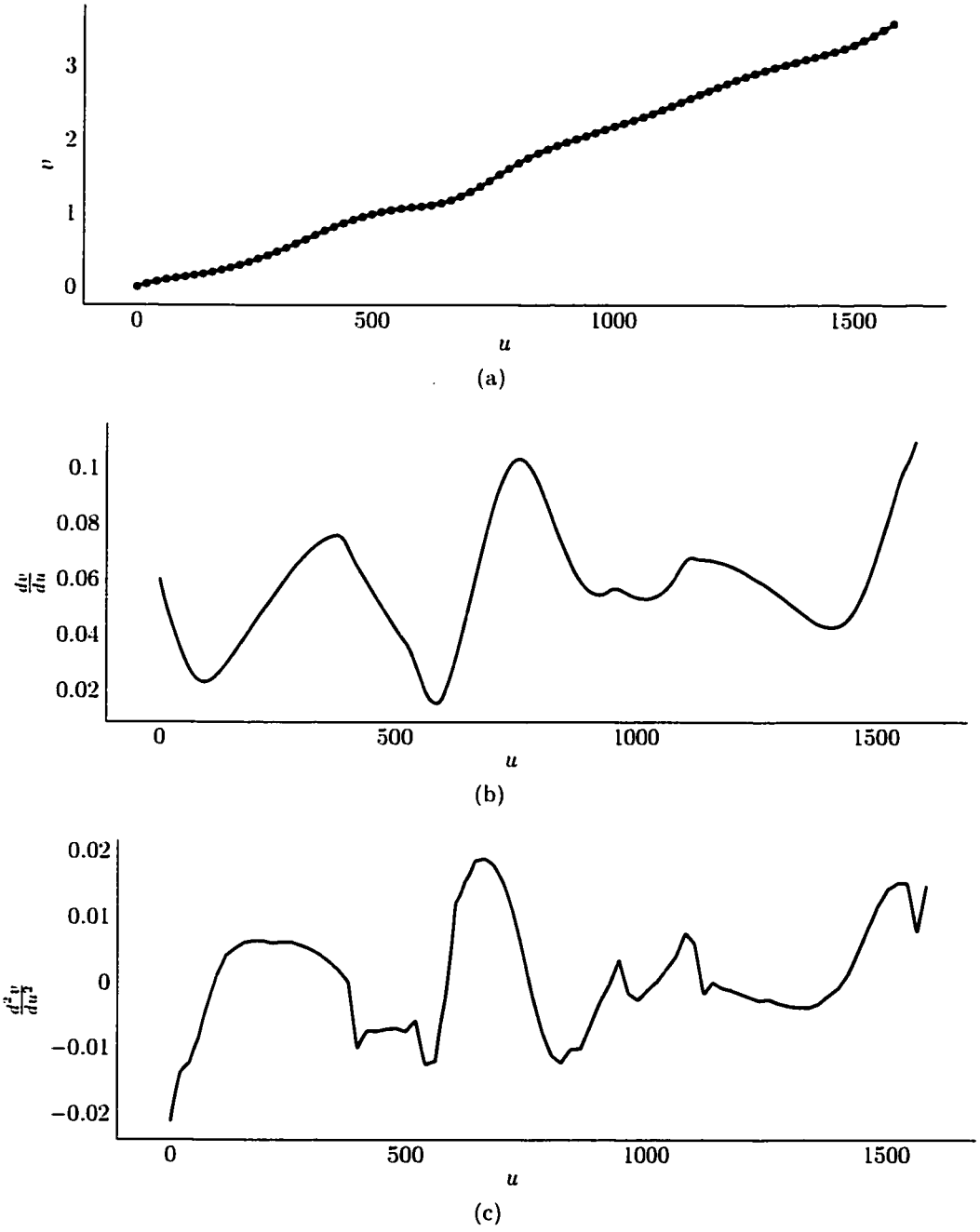


FIGURE 7–20 Plot of (a) reparameterization curve, (b) first and (c) second derivatives interpolating the ballnose milling tool-path.

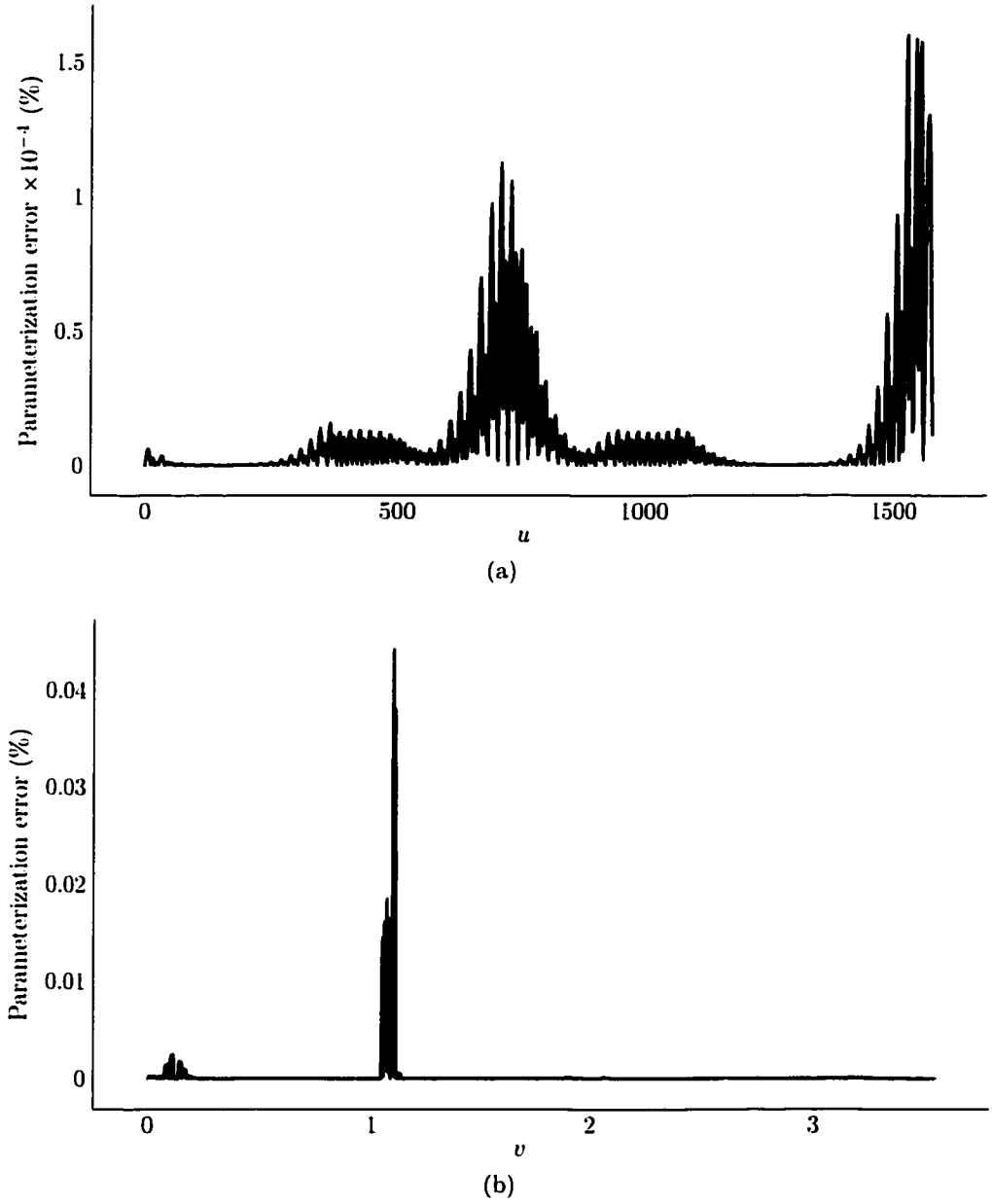


FIGURE 7–21 Plot of parameterization error of the (a) position and (b) orientation curves for the ballnose milling tool-path.

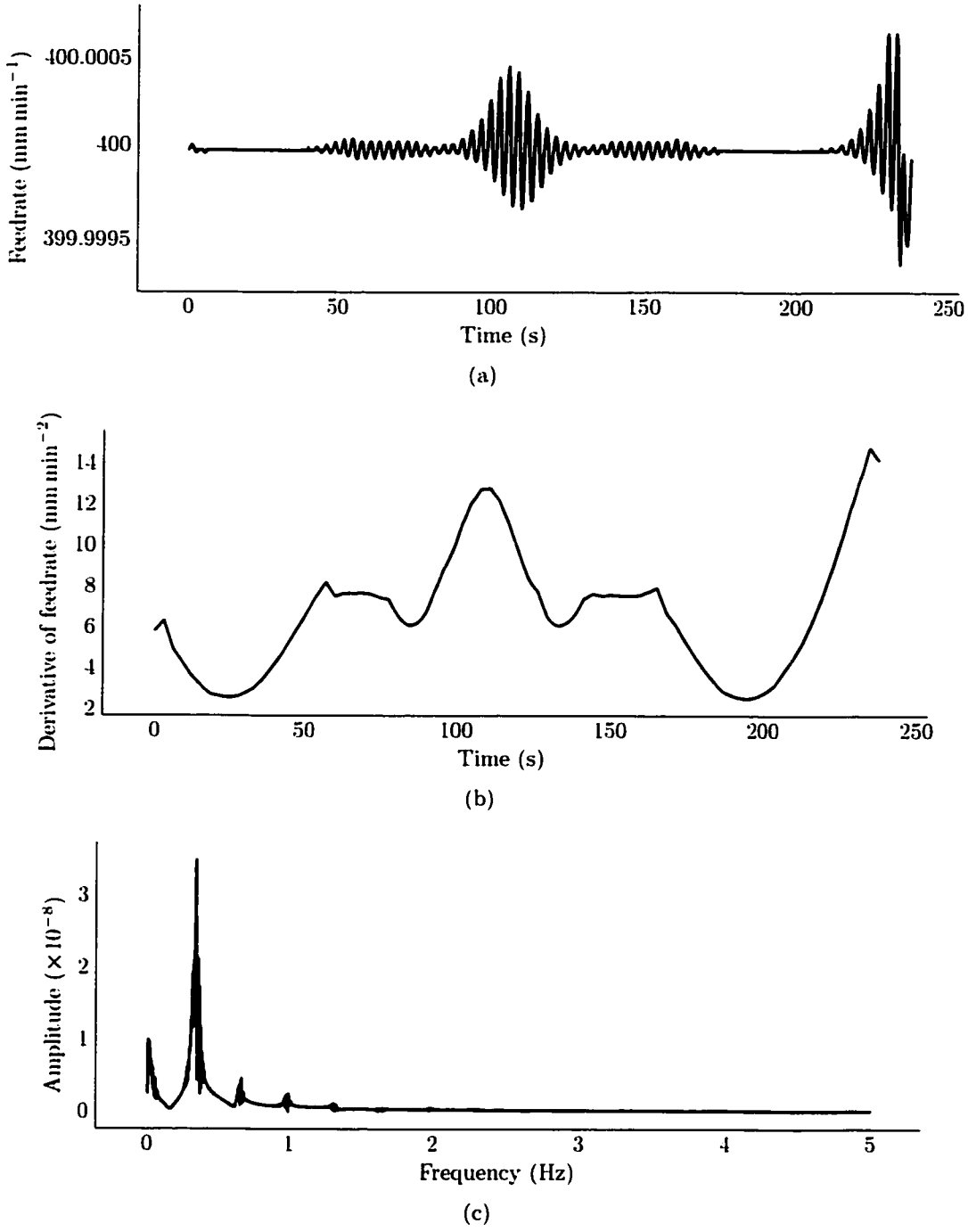


FIGURE 7–22 Plot of simulated (a) feedrate, (b) derivative and (c) FFT for ballnose milling tool-path.

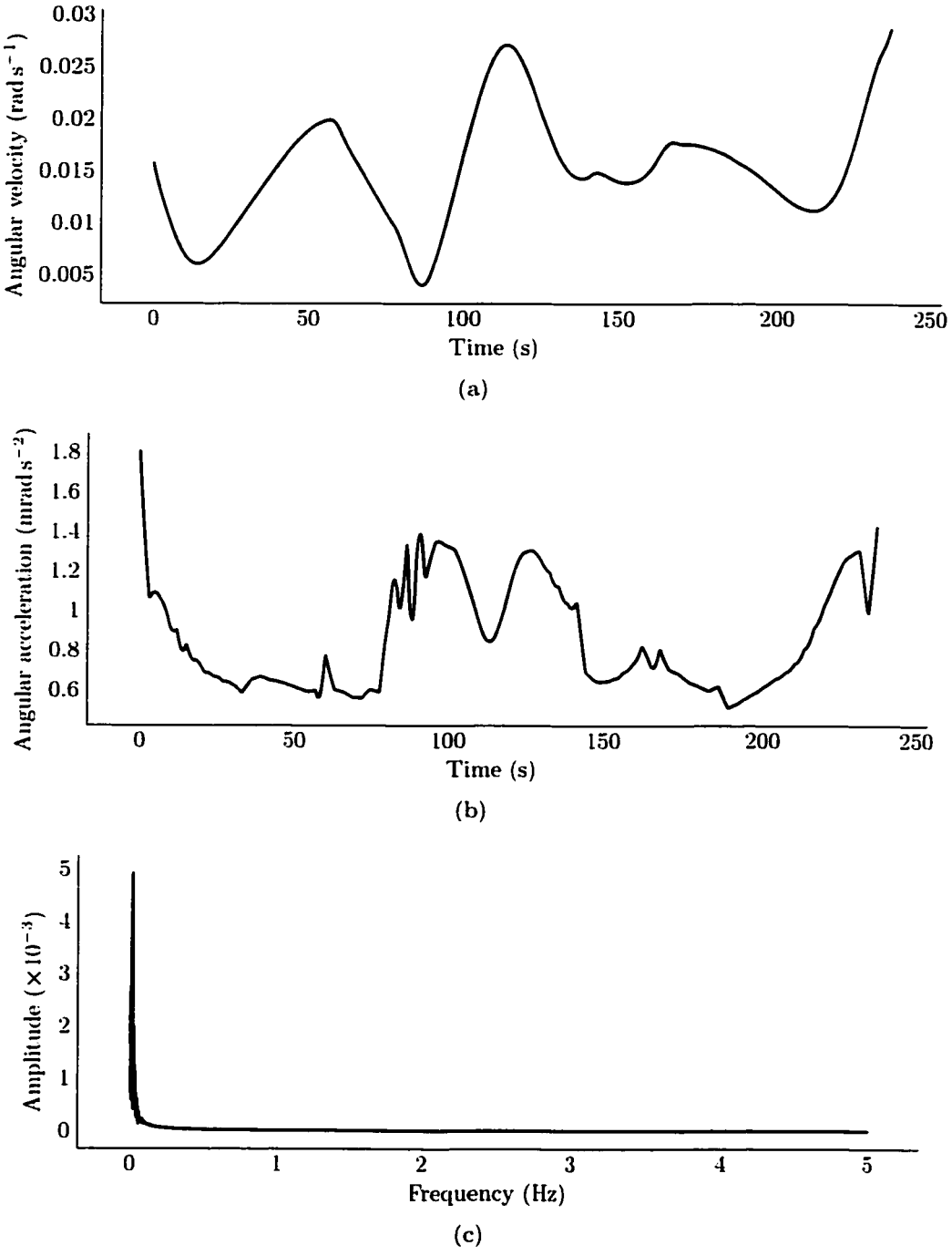


FIGURE 7–23 Plot of simulated (a) angular speed, (b) derivative and (c) FFT for ballnose milling tool-path.

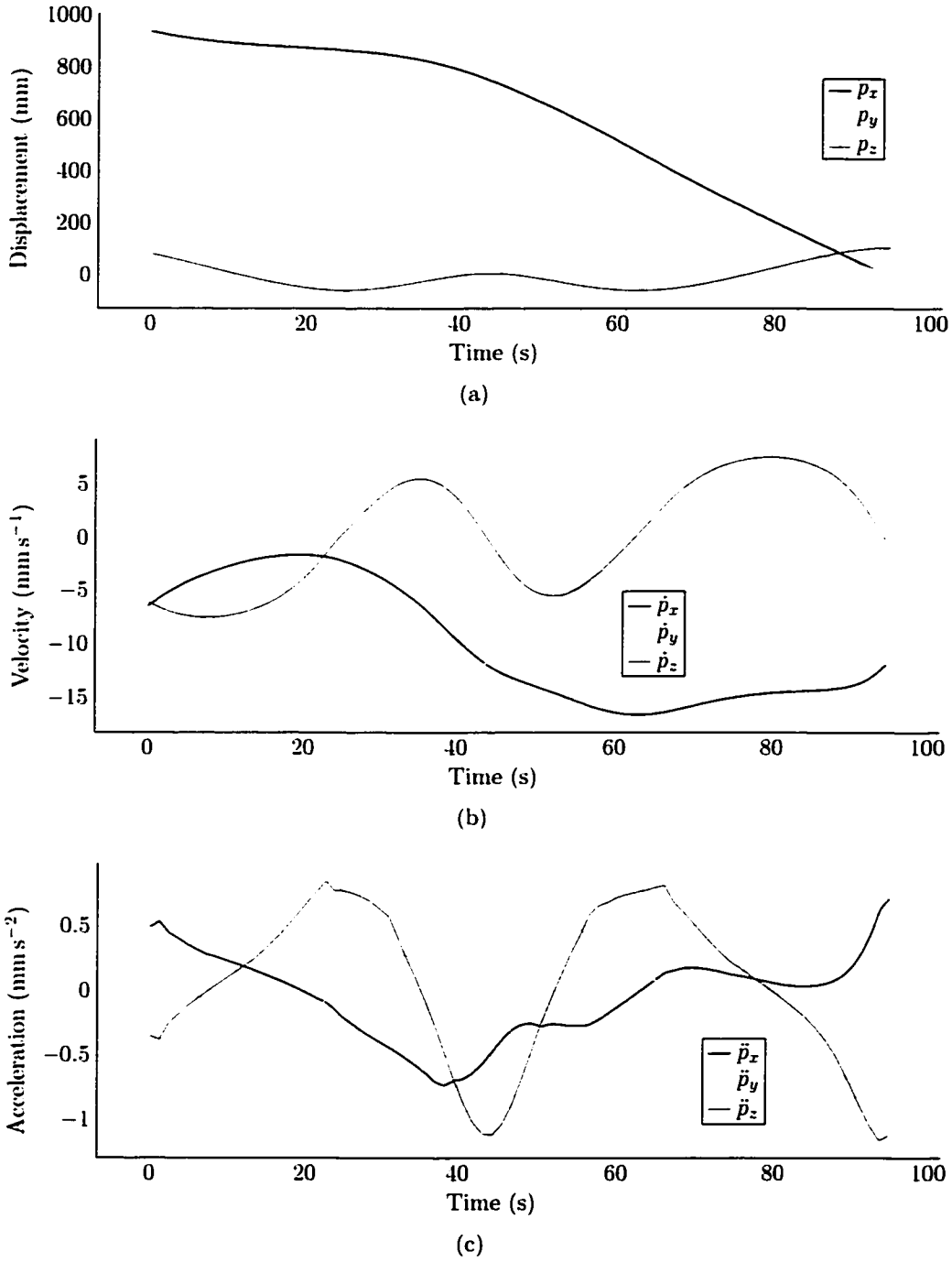


FIGURE 7–24 Plot of simulated (a) position curve components, (b) first and (c) second derivatives interpolating the ballnose milling tool-path at the feedrate 400 mm min^{-1}.

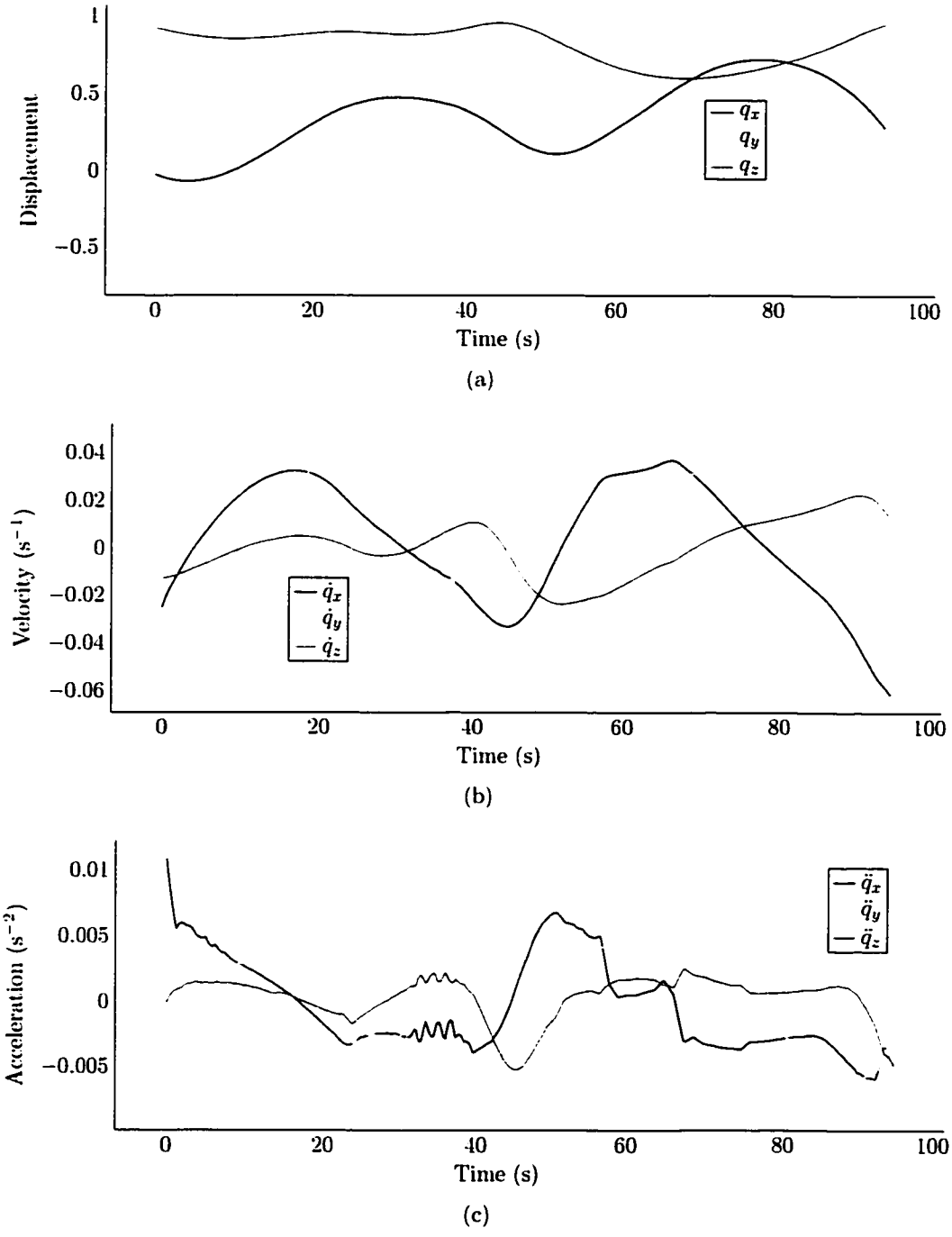


FIGURE 7–25 Plot of simulated (a) orientation curve components, (b) first and (c) second derivatives interpolating the ballnose milling tool-path at the feedrate 400 mm min^{-1}.

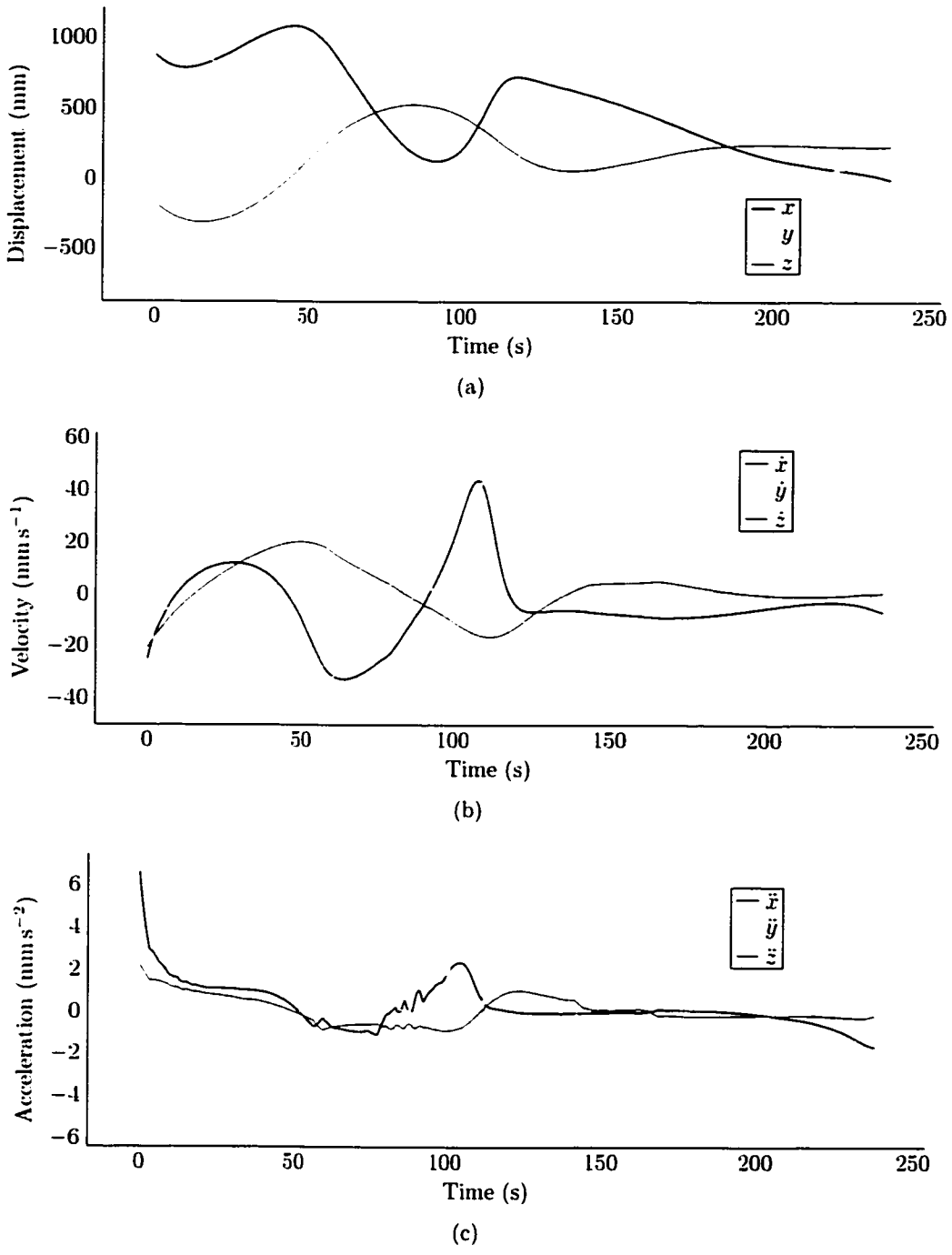


FIGURE 7–26 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the ballnose milling tool-path and constant feedrate.

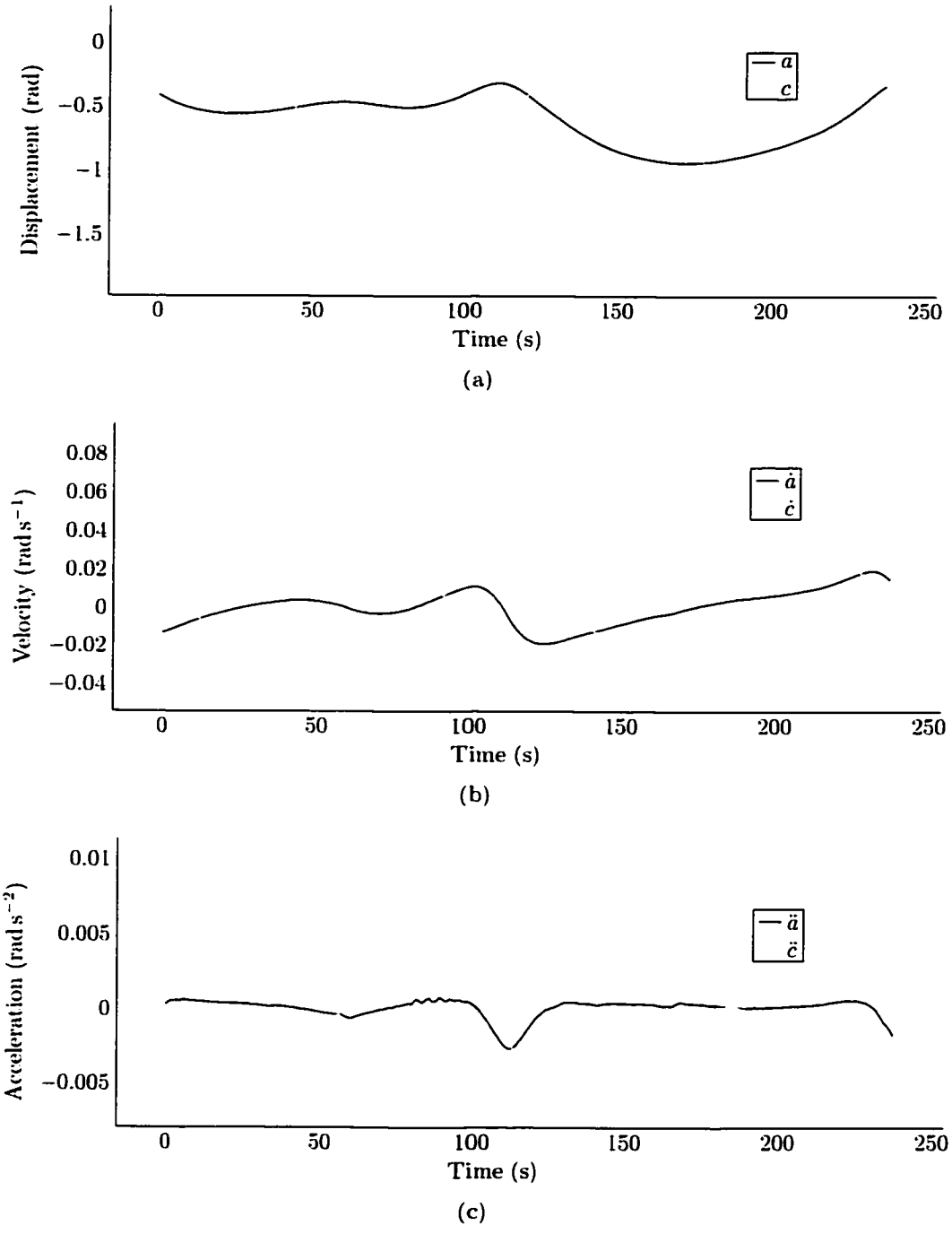


FIGURE 7–27 Plot of simulated (a) rotary axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in horizontal position for the ballnose milling tool-path and constant feedrate.

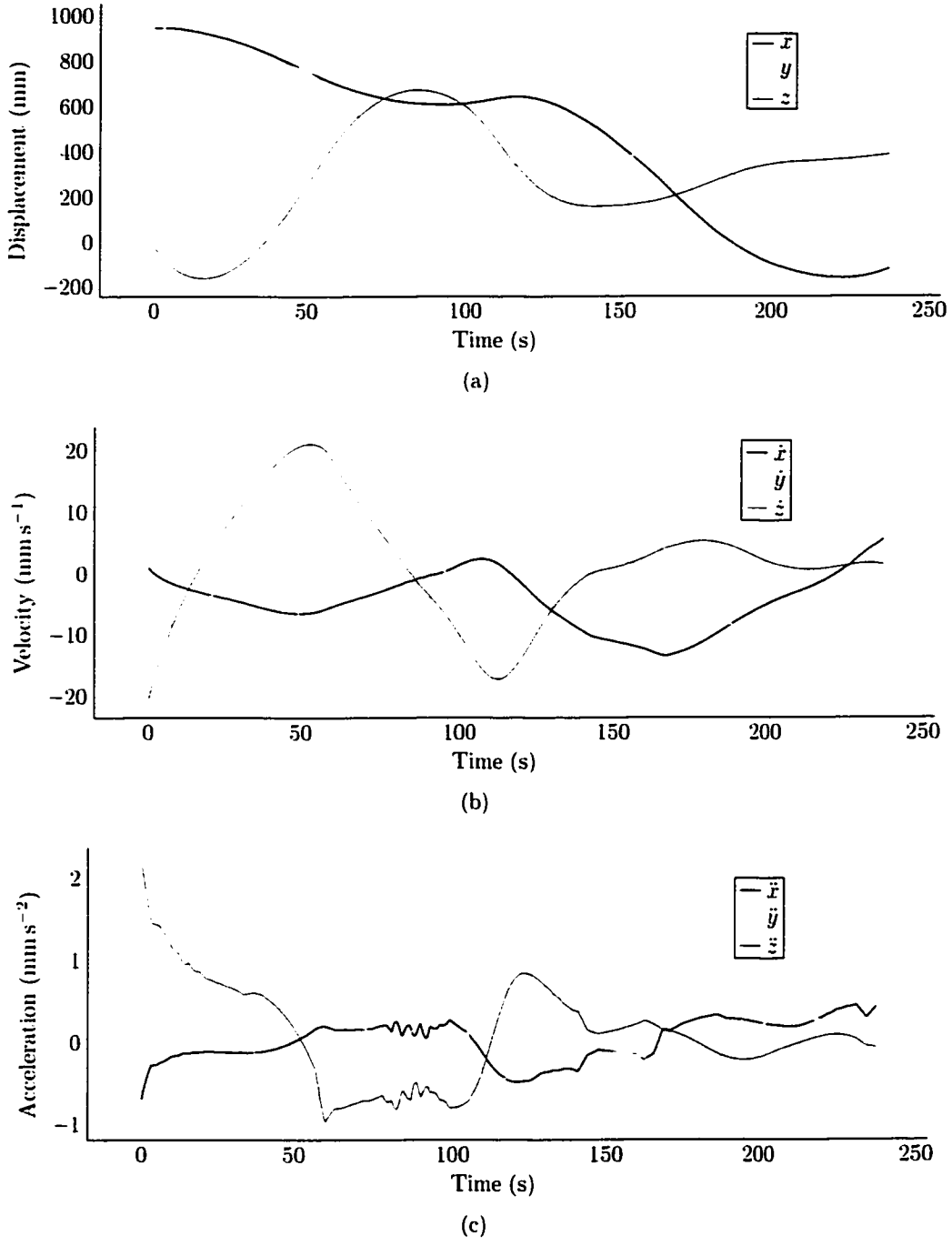


FIGURE 7–28 Plot of simulated (a) linear axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the ballnose milling tool-path and constant feedrate.

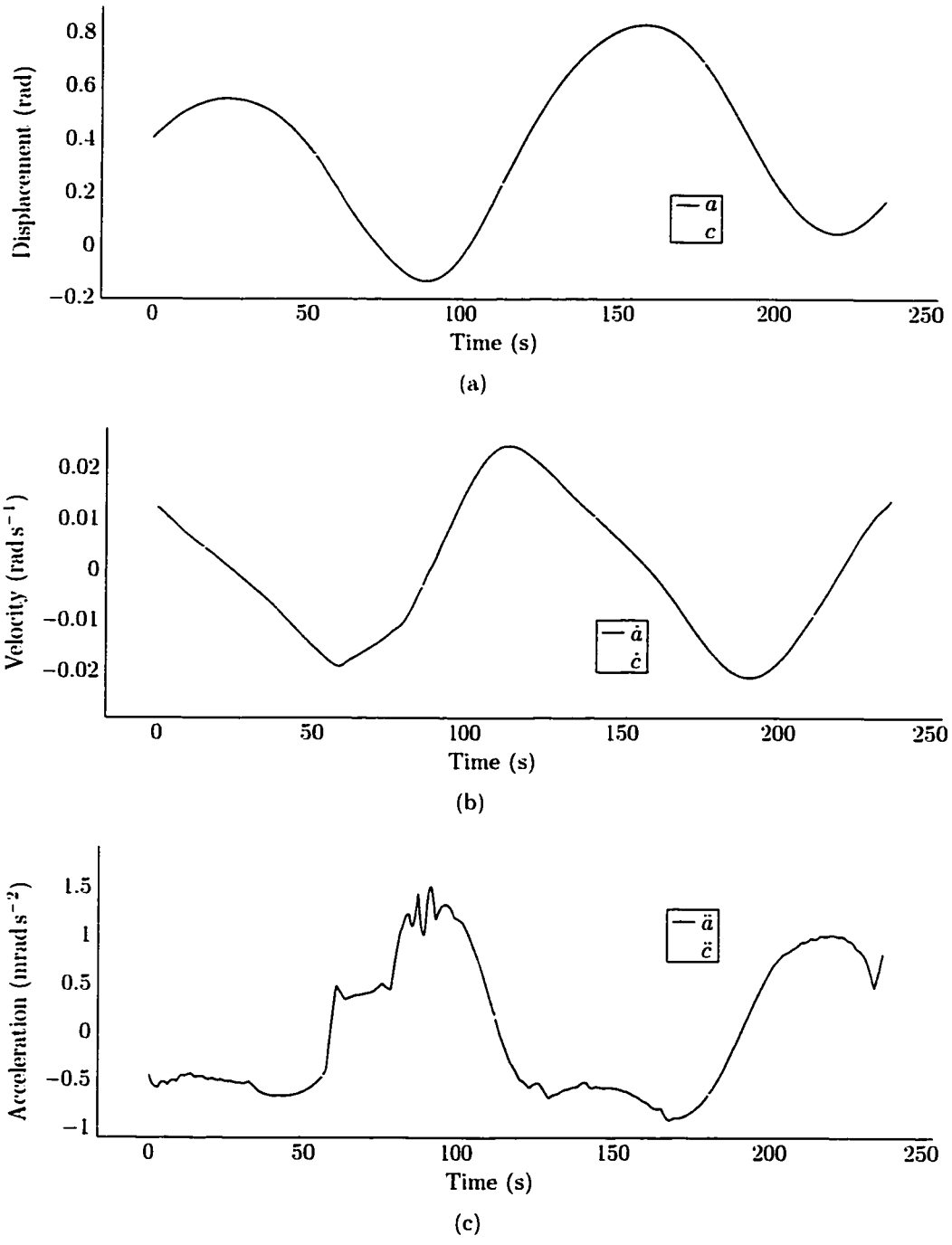


FIGURE 7–29 Plot of simulated (a) rotary axes, (b) first and (c) second derivatives of a tilting-rotary five-axis mill with table in vertical position for the ballnose milling tool-path and constant feedrate.

CHAPTER 8
Conclusions

This thesis has developed a general five-axis interpolation architecture and algorithms using parametric splines. The architecture involves sampling the tool-path and computing inverse kinematics in real-time on the CNC machine tool controller. This particular arrangement has two advantages with respect to conventional practice. Firstly, changes in the kinematics of the machine tool such as workpiece relocation or a tool change do not require off-line repost-processing of the tool-path. This saves time and eliminates possible errors. Secondly, the improved architecture enables more accurate placement of the cutter on the part surface. By sampling the tool-path curves or the design surface in real-time and then performing the kinematics computations, the cutter location will be more accurate. In contrast, the use of linear interpolation to interpolate axis commands, as is often done by commercial software, introduces velocity discontinuities and uncontrolled feedrate oscillations. Furthermore, because the inverse kinematics calculation step is performed before rather than after interpolation the tool-path followed varies with changes in kinematics including workpiece location and cutter geometry. In this thesis, it was shown that with the chosen near arc-length parameterization technique a spline can be tested to determine its devia-

tion from unit speed and improved to the desired tolerance which is a valuable tool in limiting feedrate error.

The algorithms developed consist of an interpolated C^2 continuous tool-path with near constant feedrate based on three splines:

1. A near arc-length parameterized C^2 continuous polynomial quintic spline interpolating the position.

2. A near arc-length parameterized C^2 continuous spherical Bézier spline interpolating the orientation.

3. A monotonic reparameterization spline to synchronize the orientation and position splines while reducing angular acceleration.

The position spline is developed from previous near arc-length parameterized splines. The method is improved by normalizing the first derivative at the spline junctions to the tangent, and also normalizing the second derivative to the curvature at the junctions. Consequently, the average parameterization error is reduced by 50% in comparison to the best published results.

Interpolating the orientation spline on the unit sphere has its roots in theoretical kinematics. The method proposed is unique in the application of near arc-length methods to a spherical Bézier spline.

To ensure synchronous motion of the position and orientation spline, the orientation spline is reparameterized with a monotonic spline. This method of synchronization as opposed to using a naturally parameterized spherical Bézier spline is demonstrated to reduce angular acceleration.

Simulations were run to demonstrate the feasibility of the proposed algorithms for two sample tool-paths: one side milling and another ballnose milling. From the

simulations. the interpolated tool-path is C^2 continuous, with near constant feedrate, and velocities and accelerations which correspond well with previous work on three-axis interpolators.

Further simulations demonstrate the effect of the interpolated tool-path on axes commands. For this purpose, two kinematics configurations were used: a tilting-rotary table five-axis vertical machining centre with the workpiece mounted in two orientations. Again C^2 continuity was demonstrated. From the simulations it was determined that in one case the axis velocity was outside the normal range. A feedrate interpolation method was developed to reduce the feedrate such that the axis velocity was within normal operating bounds. By interpolating the feedrate with a C^1 spline, C^2 continuity of the axes commands was maintained.

The efficiency of the interpolation algorithms presented in this thesis may be improved. It may be possible to convert the Bézier control points of the orientation curve to deBoor points without altering the shape or parameterization of the curves. This would result in a significant reduction in the amount of data required for the canonical representation of a five-axis tool-path.

To achieve automation of the proposed algorithms, the solution to the non-linear equations for the orientation spline must be deterministic in nature. The numerical stability of Broyden's method for solving the spherical spline control points has not been investigated. The investigation of a linear method would make the proposed interpolation technique practicable in industrial situations.

The solution to the interpolation problem is inextricably linked to the as of yet poorly researched question—which points on the design surface should be chosen for interpolation? This is a question of both quantity and quality. The number of points and their locations must be investigated. The location of points will determine

how well the curve follows the design surface and the parameterization error of the interpolated tool-path curves. Furthermore, using tangent and curvature information from the design surface will aid in improving the accuracy of the interpolated tool-path and may eliminate the first step of interpolating a cubic spline from which tangent and curvatures were obtained.

The parameterized tool-path is a valuable not only in generating axis commands but as a general format for the representation of tool-paths in both the CNC machine tool controller and also in CAD/CAM systems. Although, more complex than the conventional CL-data representation of five-axis tool-paths, a parameterized tool-path provides first and second derivative information as well as a smoothly variable feedrate schedule. Obtaining derivative from information from a noisy signal such as a discrete tool-path represents a danger since the derivatives inevitably amplify the noise and obscure the signal.

The machining test of the side milling tool-path demonstrates that the location and orientation of the workpiece in the workspace of the machine tool has a significant effect on the kinematics of command generation. A poor choice of workpiece location and orientation must result in poor surface finish and geometric inaccuracy. The parameterized tool-path solution is an invaluable tool in determining the best possible fixturing of a workpiece to improve the surface finish and accuracy of the part.

APPENDIX A

Derivatives of the Spherical Bézier Curve and Spline

The derivative of Eq. 5–54 is

$$\frac{d}{dv}\mathbf{B}_k^j(v) = \left[\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\sin(\theta(1-v)) + \mathbf{B}_k^{j-1}(v)\cos(\theta(1-v))\,(\theta'(1-v)-\theta)\right.$$
$$\left. + \frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v) + \mathbf{B}_{k+1}^{j-1}(v)\cos(\theta v)(\theta' v+\theta)\right]\,[\sin(\theta)]^{-1}$$
$$+ \left[\mathbf{B}_k^{j-1}(v)\sin(\theta(1-v)) + \mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v)\right]\left[-\theta'\cos(\theta)(\sin(\theta))^{-2}\right] \tag{A-1}$$

where

$$\theta = \arccos(\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v)),$$

$$\theta' = -\left(\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v)\right.$$
$$\left. + \mathbf{B}_k^{j-1}(v)\cdot\frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v)\right)\left(1-\left(\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v)\right)^2\right)^{\frac{1}{2}}.$$

The second derivative of Eq. 5–54 is

$$\begin{aligned}
\frac{d^2}{dv^2}\mathbf{B}_k^j(v) = \Bigg[\Bigg[&\frac{d^2}{dv^2}\mathbf{B}_k^{j-1}(v)\sin(\theta(1-v)) \\
&+ 2\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\cos(\theta(1-v))(\theta'(1-v)-\theta) \\
&- \mathbf{B}_k^{j-1}(v)\sin(\theta(1-v))(\theta'(1-v)-\theta)^2 \\
&+ \mathbf{B}_k^{j-1}(v)\cos(\theta(1-v))(\theta''(1-v)-2\theta') \\
&+ \frac{d^2}{dv^2}\mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v) + 2\frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v)\cos(\theta v)(\theta' v+\theta) \\
&- \mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v)(\theta' v+\theta)^2 + \mathbf{B}_{k+1}^{j-1}(v)\cos(\theta v)(\theta'' v+2\theta')\Big]\sin^{-1}(\theta)\Big] \\
&- 2\left[\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\sin(\theta(1-v)) + \mathbf{B}_k^{j-1}(v)\cos(\theta(1-v))(\theta'(1-v)-\theta)\right. \\
&\left.+ \frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v) + \mathbf{B}_{k+1}^{j-1}(v)\cos(\theta v)(\theta' v+\theta)\right]\cos(\theta)\theta'\sin^{-2}(\theta) \\
&+ \left[2\cos^2(\theta)\theta'^2\sin^{-3}(\theta)\theta'^2\sin^{-1}(\theta) - \right. \\
&\left.+ \cos(\theta)\theta''\sin^{-2}(\theta)\right]\left(\mathbf{B}_k^{j-1}(v)\sin(\theta(1-v)) + \mathbf{B}_{k+1}^{j-1}(v)\sin(\theta v)\right)
\end{aligned} \tag{A-2}$$

where

$$\theta = \arccos(\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v)),$$

$$\begin{aligned}
\theta' = -\Bigg(&\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v) \\
&+ \mathbf{B}_k^{j-1}(v)\cdot\frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v)\Bigg)\left(1-\left(\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v)\right)^2\right)^{\frac{1}{2}}.
\end{aligned}$$

$$\theta'' = -\sin^{-3}(\theta)\left[(\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v))\left(\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v) + \mathbf{B}_k^{j-1}(v)\cdot\frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v)\right)^2 + (1 - (\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v))^2)\left(\frac{d^2}{dv^2}\mathbf{B}_k^{j-1}(v)\cdot\mathbf{B}_{k+1}^{j-1}(v) + 2\frac{d}{dv}\mathbf{B}_k^{j-1}(v)\cdot\frac{d}{dv}\mathbf{B}_{k+1}^{j-1}(v) + \mathbf{B}_k^{j-1}(v)\cdot\frac{d^2}{dv^2}\mathbf{B}_{k+1}^{j-1}(v)\right)\right].$$

For Bézier curves the first derivative at $v = 0$ is affected by only the first and second control point. Hence, after differentiating Eq. 5–57 with respect to v and evaluating the result at zero (the subscript i has been dropped for clarity)

$$\left.\frac{d}{dv}\mathbf{Q}^j(v)\right|_{v=0} = \mathbf{q}' = \frac{j\theta}{\lambda\sin(\theta)}(\mathbf{d}_2 - \mathbf{d}_1\cos(\theta)) \tag{A–3}$$

where

$$\theta = \arccos(\mathbf{d}_1\cdot\mathbf{d}_2).$$

Similarly, when the first derivative of the spherical Bézier spline segment is evaluated at λ

$$\left.\frac{d}{dv}\mathbf{Q}^j(v)\right|_{v=\lambda} = \mathbf{q}' = \frac{j\theta}{\lambda\sin(\theta)}(\mathbf{d}_{j+1}\cos(\theta) - \mathbf{d}_j) \tag{A–5}$$

where

$$\theta = \arccos(\mathbf{d}_j\cdot\mathbf{d}_{j+1}).$$

Eq. A–3 and Eq. A–5 can be inverted to yield

$$\mathbf{d}_2 = \mathbf{d}_1\cos\left(\frac{\lambda|\mathbf{q}'|}{j}\right) + \frac{\mathbf{q}'}{|\mathbf{q}'|}\sin\left(\frac{\lambda|\mathbf{q}'|}{j}\right) \tag{A–7}$$

and

$$\mathbf{d}_j = \mathbf{d}_{j+1}\cos\left(\frac{\lambda|\mathbf{q}'|}{j}\right) - \frac{\mathbf{q}'}{|\mathbf{q}'|}\sin\left(\frac{\lambda|\mathbf{q}'|}{j}\right) \tag{A-8}$$

respectively.

Given the second derivative and the first two control points the third control point can be found because the second derivative at the end $v = 0$ is only affected by the first three control points as evidenced when Eq. 5-57 is differentiated twice and evaluated first at zero

$$\begin{aligned}\left.\frac{d^2}{dv^2}\mathbf{Q}^j(v)\right|_{v=0} = \mathbf{q}'' = \frac{j}{\lambda^2}\left(\frac{\theta}{\sin(\theta)}\left(\left.\frac{d}{dv}\mathbf{B}_2^{j-1}(v)\right|_{v=0} - \left.\frac{d}{dv}\mathbf{B}_1^{j-1}(v)\right|_{v=0}\cos(\theta)\right)\right. \\ \left. + \left.\frac{d^2}{dv^2}\mathbf{B}_1^1(v)\right|_{v=0} + \theta'\left(\frac{1}{\theta}\left.\frac{d}{dv}\mathbf{B}_1^1(v)\right|_{v=0} - \frac{1}{\sin(\theta)}\left.\frac{d}{dv}\mathbf{B}_1^1(v)\right|_{v=1}\right)\right)\end{aligned} \tag{A-9}$$

where

$$\theta = \arccos(\mathbf{d}_1 \cdot \mathbf{d}_2)$$

$$\theta' = -\frac{\mathbf{d}_1 \cdot \left.\frac{d}{dv}\mathbf{B}_2^{j-1}(v)\right|_{v=0} + \mathbf{d}_2 \cdot \left.\frac{d}{dv}\mathbf{B}_1^{j-1}(v)\right|_{v=0}}{\sin(\theta)}$$

and then at λ

$$\begin{aligned}\left.\frac{d^2}{dv^2}\mathbf{Q}^j(v)\right|_{v=\lambda} = \mathbf{q}'' = \frac{j}{\lambda^2}\left(\frac{\theta}{\sin(\theta)}\left(\left.\frac{d}{dv}\mathbf{B}_{j+1}^{j-1}(v)\right|_{v=1}\cos(\theta) - \left.\frac{d}{dv}\mathbf{B}_j^{j-1}(v)\right|_{v=1}\right)\right. \\ + \left.\frac{d^2}{dv^2}\mathbf{B}_{j+1}^1(v)\right|_{v=1} \\ \left. + \theta'\left(\frac{1}{\theta}\left.\frac{d}{dv}\mathbf{B}_{j+1}^1(v)\right|_{v=1} - \frac{1}{\sin(\theta)}\left.\frac{d}{dv}\mathbf{B}_j^1(v)\right|_{v=0}\right)\right)\end{aligned} \tag{A-10}$$

where

$$\theta = \arccos(\mathbf{d}_j \cdot \mathbf{d}_{j+1})$$

$$\theta' = -\frac{\mathbf{d}_j \cdot \frac{d}{dv}\mathbf{B}_{j+1}^{j-1}(v)\Big|_{v=1} + \mathbf{d}_{j+1} \cdot \frac{d}{dv}\mathbf{B}_j^{j-1}(v)\Big|_{v=1}}{\sin(\theta)}$$

Solving for the third control point is achieved in two steps. Firstly it is recognized that the unknown, $\mathbf{d}_3$, appears in Eq. A-9 only within the right hand side of

$$\mathbf{q}' = \frac{d}{dv}\mathbf{B}_2^{j-1}(v)\Big|_{v=0}. \tag{A-11}$$

Consequently, the unknown control is obtained by first solving for $\mathbf{q}'$ and then solving for the unknown with

$$\mathbf{d}_3 = \mathbf{d}_2 \cos\left(\frac{\|\mathbf{q}'\|}{j-1}\right) + \frac{\mathbf{q}'}{\|\mathbf{q}'\|} \sin\left(\frac{\|\mathbf{q}'\|}{j-1}\right). \tag{A-12}$$

The vector $\mathbf{q}'$ is obtained by rewriting Eq. A-9 as

$$\mathbf{q}' + (\mathbf{d}_1 \cdot \mathbf{q}')\,\mathbf{r} + \mathbf{s} = \mathbf{0} \tag{A-13}$$

where

$$\mathbf{r} = \frac{1}{\theta}\left(\frac{1}{\sin(\theta)}\frac{d}{dv}\mathbf{B}_1^1(v)\Big|_{v=1} - \frac{1}{\theta}\frac{d}{dv}\mathbf{B}_1^1(v)\Big|_{v=0}\right)$$

$$\mathbf{s} = \frac{\sin(\theta)}{j\theta}\left(j\frac{d^2}{dv^2}\mathbf{B}_1^1(v)\Big|_{v=0} - \lambda^2\mathbf{q}''\right) + \mathbf{r}\left(\mathbf{d}_2 \cdot \frac{d}{dv}\mathbf{B}_1^{j-1}(v)\Big|_{v=0}\right) - \cos(\theta)\frac{d}{dv}\mathbf{B}_1^{j-1}(v)\Big|_{v=0}.$$

To isolate $\mathbf{q}'$ note that $\mathbf{q}'$ and $\mathbf{d}_2$ must be orthogonal. Taking the dot product of $\mathbf{d}_2$ and Eq. 5–16 eliminates the leftmost $\mathbf{q}'$. From the remaining equation $\mathbf{d}_1 \cdot \mathbf{q}'$ is isolated and substituted back into Eq. 5–16. This results in

$$\mathbf{q}' = \left(\frac{\mathbf{s} \cdot \mathbf{d}_2}{\mathbf{r} \cdot \mathbf{d}_2}\right) \mathbf{r} - \mathbf{s}. \tag{A-14}$$

The result for $\mathbf{d}_{j-1}$ is obtained from the second derivative of the spherical Bézier spline evaluated at λ

$$\mathbf{d}_{j-1} = \mathbf{d}_j \cos\left(\frac{\|\mathbf{q}'\|}{j-1}\right) - \frac{\mathbf{q}'}{\|\mathbf{q}'\|} \sin\left(\frac{\|\mathbf{q}'\|}{j-1}\right) \tag{A-15}$$

and

$$\frac{d}{dv}\mathbf{B}_j^{j-1}(v)\bigg|_{v=1} = \mathbf{q}' = \left(\frac{\mathbf{s} \cdot \mathbf{d}_j}{\mathbf{r} \cdot \mathbf{d}_j}\right) \mathbf{r} - \mathbf{s} \tag{A-16}$$

where

$$\mathbf{r} = \frac{1}{\theta}\left(\frac{1}{\theta}\frac{d}{dv}\mathbf{B}_{j+1}^1(v)\bigg|_{v=1} - \frac{1}{\sin(\theta)}\frac{d}{dv}\mathbf{B}_j^1(v)\bigg|_{v=0}\right)$$

$$\mathbf{s} = \frac{\sin(\theta)}{j\theta}\left(\lambda^2\mathbf{q}'' - j\frac{d^2}{dv^2}\mathbf{B}_{j+1}^1(v)\bigg|_{v=1}\right) + \mathbf{r}\left(\mathbf{d}_j \cdot \frac{d}{dv}\mathbf{B}_{j+1}^{j-1}(v)\bigg|_{v=1}\right) - \cos(\theta)\frac{d}{dv}\mathbf{B}_{j+1}^{j-1}(v)\bigg|_{v=1}.$$

APPENDIX B

Derivatives of the Feedrate Spline

The first derivative of Eq. 5–89 with respect to time is given as

$$\frac{d}{dt}U(t) = \begin{cases} g_1, & \text{if } g_2 = g_3 = 0 \\ g_1 \exp(g_2 t), & \text{if } g_3 = 0 \\ g_1\left(1 + \tan^2\left(\sqrt{g_1 g_3}\, t\right)\right), & \text{if } g_2 = 0 \\ \dfrac{g_2^2}{g_3(g_2 t - 2)^2}, & \text{if } g_2^2 = 4 g_1 g_3 \\ \dfrac{\varrho^2(g_2 - \varrho)(g_2 + \varrho)\exp(\varrho t)}{g_3(\exp(\varrho t)(g_2 - \varrho) - g_2 - \varrho)^2}, & \text{if } g_2^2 > 4 g_1 g_3 \\ \dfrac{\rho^2\left(1 + \tan^2\left(\frac{\varrho t}{2} + \arctan(g_2, \rho)\right)\right)}{4 g_3}, & \text{otherwise} \end{cases} \tag{B-1}$$

The second derivative of Eq. 5–89 with respect to time is given as

$$\frac{d^2}{dt^2}U(t) = \begin{cases} 0. & \text{if } g_2 = g_3 = 0 \\ g_1 g_2 \exp(g_2 t). & \text{if } g_3 = 0 \\ 2g_1\sqrt{g_1 g_3}\tan\left(\sqrt{g_1 g_3}t\right)\left(1+\tan^2\left(\sqrt{g_1 g_3}t\right)\right). & \text{if } g_2 = 0 \\ \dfrac{-2g_2^3}{g_3(g_2 t-2)^3}. & \text{if } g_2^2 = 4g_1 g_3 \\ \dfrac{\varrho^3(g_2-\varrho)(g_2+\varrho)\exp(\varrho t)(g_2+\varrho+(g_2-\varrho)\exp(\varrho t))}{g_3(\exp(\varrho t)(\varrho-g_2)+g_2+\varrho)^3}. & \text{if } g_2^2 > 4g_1 g_3 \\ \dfrac{\rho^3\tan\left(\frac{\varrho t}{2}+\arctan(g_2,\rho)\right)\left(1+\tan^2\left(\frac{\varrho t}{2}+\arctan(g_2,\rho)\right)\right)}{4g_3}. & \text{otherwise} \end{cases} \tag{B–2}$$

APPENDIX C

Example Tool-path Data

TABLE C–1 Interpolatory data for the example peripheral milling tool-path.

	p_x (mm)	p_y (mm)	p_z (mm)	q_x	q_y	q_z
1	113.560775	7.735266	-2.209314	-0.107258	0.624902	0.773300
2	117.864949	-10.950074	-0.974065	-0.003002	0.653008	0.757345
3	115.502860	-34.808781	0.779567	0.135034	0.648777	0.748902
4	104.086026	-55.840098	2.682644	0.263922	0.596758	0.757776
5	95.225652	-63.959571	4.073524	0.317154	0.551822	0.771301
6	88.820589	-65.972318	6.021818	0.329520	0.516103	0.790603
7	80.162816	-65.096397	7.031464	0.326785	0.478040	0.815285
8	72.478246	-61.109537	6.863103	0.313696	0.446069	0.838223
9	65.985754	-54.666365	6.143037	0.288698	0.416448	0.862105
10	54.251993	-39.568096	4.931174	0.217010	0.357492	0.908354
11	38.038952	-23.111532	3.536073	0.129479	0.261821	0.956391
12	31.679054	-18.711329	3.017623	0.105499	0.220895	0.969575
13	26.192567	-16.781277	2.454873	0.096827	0.184942	0.977968
14	22.337845	-16.486398	1.907427	0.097931	0.159739	0.982290
15	18.769843	-17.937940	0.258718	0.110602	0.137889	0.984253
16	17.004164	-21.333422	-1.375463	0.133457	0.127499	0.982819
17	16.763098	-27.082862	-2.573817	0.171104	0.127727	0.976939
18	20.059322	-38.210737	-3.573619	0.238991	0.153346	0.958837

continued on next page

continued from previous page

	p_x (mm)	p_y (mm)	p_z (mm)	q_x	q_y	q_z
19	26.991221	-67.786318	-5.570258	0.399932	0.201303	0.894165
20	28.080737	-86.648047	-6.205088	0.487937	0.208205	0.847684
21	21.813110	-103.571377	-4.442501	0.567908	0.182153	0.802683
22	6.156323	-114.179587	-2.044104	0.628723	0.098457	0.771372
23	-12.661488	-117.988771	-0.872310	0.654180	-0.006643	0.756310
24	-31.816238	-116.324006	0.514512	0.651997	-0.117213	0.749107
25	-49.438878	-108.784390	2.089537	0.618930	-0.223905	0.752856

TABLE C–2 Interpolatory data for the example ballnose milling tool-path.

	p_x (mm)	p_y (mm)	p_z (mm)	q_x	q_y	q_z
1	988.073	987.783	103.169	0.703165	0.703165	0.10544
2	975.432	972.001	101.114	0.703663	0.702477	0.106694
3	963.993	955.616	97.5212	0.705143	0.701009	0.106585
4	953.715	938.853	92.5996	0.707512	0.698832	0.105175
5	944.515	921.873	86.5854	0.71066	0.696012	0.102611
6	936.284	904.775	79.7047	0.714482	0.692604	0.0990723
7	928.911	887.611	72.1548	0.718884	0.688645	0.0947338
8	922.295	870.408	64.1003	0.723787	0.684162	0.0897493
9	916.344	853.17	55.6767	0.729128	0.679172	0.0842452
10	910.979	835.895	46.9951	0.734856	0.673686	0.0783238
11	906.132	818.574	38.145	0.740925	0.66771	0.0720644
12	901.744	801.196	29.2073	0.747291	0.661257	0.0655367
13	897.764	783.744	20.2585	0.753909	0.654342	0.0588058
14	894.159	766.195	11.3717	0.760742	0.646973	0.0519329
15	890.895	748.529	2.61717	0.767759	0.639158	0.0449769
16	887.934	730.726	-5.93548	0.774928	0.630906	0.0379959
17	885.24	712.766	-14.2159	0.782217	0.622232	0.031049
18	882.775	694.632	-22.1521	0.789596	0.61315	0.0241971
19	880.498	676.305	-29.6552	0.797025	0.603692	0.0175185
20	878.367	657.765	-36.6239	0.804473	0.593885	0.0111026
21	876.34	639	-42.9531	0.811915	0.583753	0.00504369
22	874.37	620.004	-48.5312	0.81933	0.573322	-0.000553919
23	872.411	600.785	-53.2414	0.826698	0.562619	-0.00557536

continued on next page

continued from previous page

	p_x (mm)	p_y (mm)	p_z (mm)	q_x	q_y	q_z
24	870.413	581.363	-56.9664	0.834	0.551676	-0.00989648
25	868.328	561.78	-59.5961	0.84122	0.540527	-0.0133874
26	866.107	542.092	-61.0608	0.848362	0.529177	-0.0159304
27	863.682	522.368	-61.4147	0.855482	0.517538	-0.0174814
28	860.966	502.683	-60.6782	0.86254	0.505668	-0.0180007
29	857.885	483.111	-58.8748	0.86949	0.493642	-0.0174523
30	854.368	463.725	-56.0412	0.876285	0.481533	-0.0158112
31	850.345	444.596	-52.2327	0.882881	0.469416	-0.0130664
32	845.754	425.784	-47.5274	0.889235	0.457358	-0.00922529
33	840.533	407.339	-42.033	0.895315	0.445413	-0.00431538
34	834.616	389.266	-35.9577	0.901162	0.43348	0.00160093
35	827.929	371.552	-29.5365	0.906787	0.421506	0.00837741
36	820.39	354.189	-23.0036	0.912185	0.409472	0.0158326
37	811.91	337.17	-16.6086	0.91736	0.39735	0.0237553
38	802.396	320.506	-10.6206	0.922317	0.385116	0.0318936
39	791.809	304.203	-5.33505	0.927074	0.372744	0.0399373
40	780.178	288.266	-1.05304	0.93167	0.360187	0.0475013
41	767.53	272.763	1.95123	0.936142	0.34743	0.054139
42	753.932	257.807	3.45934	0.940514	0.334528	0.0593688
43	739.5	243.541	3.34679	0.944792	0.321608	0.0627357
44	724.379	230.11	1.61355	0.948957	0.308868	0.0638916
45	708.716	217.632	-1.61143	0.952959	0.296548	0.0626728
46	692.629	206.177	-6.10031	0.956735	0.284889	0.0591275
47	676.187	195.773	-11.5635	0.960216	0.274089	0.0534884
48	659.406	186.423	-17.6895	0.963336	0.264307	0.0461121
49	642.27	178.111	-24.1702	0.96604	0.255669	0.0374139
50	624.792	170.731	-30.7296	0.968301	0.248231	0.027817
51	606.967	164.187	-37.1173	0.970118	0.241987	0.017726
52	588.782	158.395	-43.1003	0.971497	0.236932	0.00753504
53	570.235	153.284	-48.4605	0.972455	0.233077	-0.00236379
54	551.34	148.788	-52.9963	0.973014	0.230455	-0.0115746
55	532.14	144.847	-56.6014	0.973187	0.229163	-0.01977
56	512.697	141.399	-59.2266	0.973006	0.229229	-0.0267097
57	493.083	138.372	-60.8253	0.972507	0.230642	-0.0321601
58	473.37	135.695	-61.3658	0.971721	0.233388	-0.0358912
59	453.63	133.29	-60.8248	0.97067	0.237446	-0.0376717
60	433.938	131.08	-59.1812	0.969363	0.242789	-0.0372636

continued on next page

continued from previous page

	p_x (mm)	p_y (mm)	p_z (mm)	q_x	q_y	q_z
61	414.369	128.99	-56.4082	0.967794	0.24938	-0.0344176
62	394.988	126.978	-52.5178	0.965918	0.257235	-0.0288531
63	375.822	125	-47.6464	0.963652	0.266383	-0.0203859
64	356.891	123.007	-41.9109	0.960901	0.276748	-0.00893464
65	338.199	120.953	-35.4284	0.957545	0.288231	0.00557938
66	319.74	118.793	-28.3127	0.953433	0.300708	0.0232502
67	301.498	116.483	-20.6716	0.948381	0.314037	0.0442003
68	283.451	113.982	-12.6069	0.942165	0.328057	0.0685894
69	265.573	111.249	-4.21374	0.934508	0.34258	0.0966154
70	247.846	108.247	4.433	0.92509	0.357355	0.128478
71	230.25	104.938	13.2619	0.913509	0.372088	0.164475
72	212.763	101.284	22.1993	0.899262	0.386424	0.204951
73	195.362	97.2475	31.1709	0.881726	0.399911	0.250261
74	178.025	92.792	40.1007	0.860133	0.411977	0.300741
75	160.734	87.8707	48.909	0.83355	0.421878	0.356669
76	143.48	82.4179	57.5116	0.800881	0.428608	0.418191
77	126.261	76.3637	65.8173	0.76087	0.430854	0.485224
78	109.081	69.6288	73.7198	0.712092	0.426973	0.557332
79	91.9549	62.1223	81.0871	0.653011	0.414987	0.633532
80	74.9146	53.7428	87.7579	0.582175	0.392583	0.712004
81	58.0201	44.3824	93.54	0.498527	0.357257	0.789834
82	41.3734	33.9368	98.2138	0.401872	0.306622	0.862834
83	25.1319	22.3237	101.549	0.293487	0.23896	0.925615
84	9.513	9.50774	103.341	0.176715	0.153962	0.972146
85	-5.22119	-4.47594	103.454	0.0571643	0.0534376	0.996934

APPENDIX D

Derivatives of Interpolated Tool-path and Axis Commands

The derivatives of the position parameter u with respect to time for constant feedrate are

$$\dot{u} = \frac{d}{dt}U(t) = f, \tag{D-1}$$

$$\ddot{u} = \frac{d^2}{dt^2}U(t) = 0 \tag{D-2}$$

and for variable feedrate are given by the Eq. 5-89.

The time derivatives of the position and orientation components of the tool-path are

$$\dot{\mathbf{p}} = \frac{d}{dt}\mathbf{P}(t) = \frac{d}{du}\mathbf{P}(U(t))\ \frac{d}{dt}U(t), \tag{D-3}$$

$$\ddot{\mathbf{p}} = \frac{d^2}{dt^2}\mathbf{P}(t) = \frac{d^2}{du^2}\mathbf{P}(U(t))\left(\frac{d}{dt}U(t)\right)^2 + \frac{d}{du}\mathbf{P}(U(t))\ \frac{d^2}{dt^2}U(t), \tag{D-4}$$

and

$$\dot{\mathbf{q}} = \frac{d}{dt}\mathbf{Q}(t) = \frac{d}{du}\mathbf{Q}(V(U(t)))\ \frac{d}{du}V(U(t))\ \frac{d}{dt}U(t). \tag{D-5}$$

$$\begin{aligned}\ddot{\mathbf{q}} = \frac{d^2}{dt^2}\mathbf{Q}(t) = &\frac{d^2}{du^2}\mathbf{Q}(V(U(t)))\left(\frac{d}{dv}V(U(t))\right)^2\left(\frac{d}{dt}U(t)\right)^2 \\ &+ \frac{d}{du}\mathbf{Q}(V(U(t)))\ \frac{d^2}{dv^2}V(U(t))\left(\frac{d}{dt}U(t)\right)^2 \\ &+ \frac{d}{du}\mathbf{Q}(V(U(t)))\ \frac{d}{dv}V(U(t))\ \frac{d^2}{dt^2}U(t).\end{aligned} \tag{D-6}$$

Since an axis command is a function of the tool-path, its first and second time derivatives are

$$\dot{a} = \frac{d}{dt}A(t) = \frac{d}{dt}A(\mathbf{P}(t), \mathbf{Q}(t)) = \frac{\partial}{\partial \mathbf{p}}A(t)\ \frac{d}{dt}\mathbf{P}(t) + \frac{\partial}{\partial \mathbf{q}}A(t)\ \frac{d}{dt}\mathbf{Q}(t), \tag{D-7}$$

$$\begin{aligned}\ddot{a} = \frac{d^2}{dt^2}A(t) = \frac{d^2}{dt^2}A(\mathbf{P}(t), \mathbf{Q}(t)) = &\frac{\partial^2}{\partial \mathbf{p}^2}A(t)\ \frac{d}{dt}\mathbf{P}(t) + \frac{\partial}{\partial \mathbf{p}}A(t)\ \frac{d^2}{dt^2}\mathbf{P}(t) \\ &+ \frac{\partial^2}{\partial \mathbf{q}^2}A(t)\ \frac{d}{dt}\mathbf{Q}(t) + \frac{\partial}{\partial \mathbf{q}}A(t)\ \frac{d^2}{dt^2}\mathbf{Q}(t).\end{aligned} \tag{D-8}$$

APPENDIX E

Derivatives of Machine Tool Inverse Kinematics

The first and second derivatives of the inverse kinematics solution for a tilting-rotary table five-axis vertical machining centre with the table in the horizontal position:

$$\dot{a} = \mp \frac{\dot{q}_z}{(1 - q_z^2)^{\frac{1}{2}}} \tag{E-1}$$

$$\dot{c} = \frac{\dot{q}_x q_y - q_x \dot{q}_y}{q_x^2 + q_y^2} \tag{E-2}$$

$$\dot{x} = (d_x \dot{c} + \dot{p}_y) \sin c - (d_y \dot{c} - \dot{p}_x) \cos c \tag{E-3}$$

$$\begin{aligned} \dot{y} = &((d_x \dot{c} + \dot{p}_y) \cos c - (d_y \dot{c} - \dot{p}_x) \sin c - d_z \dot{a}) \cos a \\ &- (d_x \dot{a} \sin c + d_y \dot{a} \cos c + \dot{p}_z) \sin a \end{aligned} \tag{E-4}$$

$$\begin{aligned} \dot{z} = &(d_x \dot{a} \sin c + d_y \dot{a} \cos c + \dot{p}_z) \cos a \\ &+ ((d_x \dot{c} + \dot{p}_y) \cos c + (\dot{p}_x - d_y \dot{c}) \sin c - d_z \dot{a}) \sin a \end{aligned} \tag{E-5}$$

$$\ddot{a} = \pm \frac{q_z \dot{q}_z^2 + (1 - q_z^2)\ddot{q}_z}{(1 - q_z^2)^{\frac{3}{2}}} \tag{E-6}$$

$$\ddot{c} = \frac{(q_x^2 + q_y^2)(q_x \ddot{q}_y - q_y \ddot{q}_x) + 2(q_x \dot{q}_x + q_y \dot{q}_y)(q_y \dot{q}_x - q_x \dot{q}_y)}{(q_x^2 + q_y^2)^2} \tag{E-7}$$

$$\ddot{x} = (-(d_x \dot{c} - d_y \ddot{c} + 2\dot{p}_y)\dot{c} + \ddot{p}_x)\cos c + (-d_x \ddot{c} + (d_y \dot{c} - 2\dot{p}_x)\dot{c} - \dot{p}_y)\sin c \tag{E-8}$$

$$\begin{aligned}\ddot{y} = &((d_x \ddot{c} - d_y(\dot{a}^2 + \dot{c}^2) + 2\dot{c}\dot{p}_x + \ddot{p}_y)\cos c + \\ &(-d_x(\dot{a}^2 + \dot{c}^2) - d_y \ddot{c} + \ddot{p}_x - 2\dot{c}\dot{p}_y)\sin c - \\ &d_z \ddot{a} - 2\dot{a}\dot{p}_z)\cos a + ((-2(d_x \dot{c} + \dot{p}_y)\dot{a} - d_y \ddot{a})\cos c + \\ &(-d_x \ddot{a} + 2(d_y \dot{c} - \dot{p}_x)\dot{a})\sin c + d_z \dot{a}^2 - \ddot{p}_z)\sin a\end{aligned} \tag{E-9}$$

$$\begin{aligned}\ddot{z} = &((2(d_x \dot{c} + \dot{p}_y)\dot{a} + d_y \ddot{a})\cos c + \\ &(d_x \ddot{a} - 2(d_y \dot{c} + \dot{p}_x)\dot{a})\sin c - d_z \dot{a}^2 + \\ &\ddot{p}_z)\cos a + ((d_x \ddot{c} - d_y(\dot{a}^2 + \dot{c}^2) + 2\dot{c}\dot{p}_x + \ddot{p}_y)\cos c + \\ &(-d_x(\dot{a}^2 + \dot{c}^2) - d_y \ddot{c} + \ddot{p}_x - 2\dot{c}\dot{p}_y)\sin c - \\ &d_z \ddot{a} - 2\dot{a}\dot{p}_z)\sin a\end{aligned} \tag{E-10}$$

The first and second derivatives of the inverse kinematics solution for a tilting-rotary table five-axis vertical machining centre with the table in the vertical position:

$$\dot{a} = \frac{q_y \dot{q}_z - \dot{q}_y q_z}{q_y^2 + q_z^2} \tag{E-11}$$

$$\dot{b} = \frac{\dot{q}_x}{\sqrt{1 - q_x^2}} \tag{E-12}$$

$$\dot{x} = (\dot{p}_x - d_z \dot{b})\cos b - (\dot{p}_x + d_x \dot{b})\sin b \tag{E-13}$$

$$\begin{aligned}\dot{y} = &(\dot{a}d_z \cos b + \dot{a}d_x \sin b + \dot{p}_y)\cos a + ((\dot{p}_z + d_x \dot{b})\cos b \\ &+ (\dot{p}_x - d_z \dot{b})\sin b - \dot{a}d_y)\sin a\end{aligned} \tag{E-14}$$

$$\begin{aligned}\dot{z} =&((\dot{p}_z + d_x\dot{b})\cos b + (\dot{p}_x - d_z\dot{b})\sin b - \dot{a}d_y)\cos a \\ &- (\dot{a}d_z\cos b + \dot{a}d_x\sin b + \dot{p}_y)\sin a\end{aligned} \tag{E-15}$$

$$\ddot{a} = \frac{(q_y^2 + q_z^2)(q_y\ddot{q}_z - \ddot{q}_y q_z) - 2(q_y\dot{q}_z - \dot{q}_y q_z)(q_y\dot{q}_y + q_z\dot{q}_z)}{(q_y^2 + q_z^2)^2} \tag{E-16}$$

$$\ddot{b} = \frac{\ddot{q}_x(1 - q_x^2) + \dot{q}_x^2 q_x}{(1 - q_x^2)^{\frac{3}{2}}} \tag{E-17}$$

$$\ddot{x} = (\ddot{p}_x - 2\dot{p}_z\dot{b} - d_z\ddot{b} - d_x\dot{b}^2)\cos b + (d_z\dot{b}^2 - d_x\ddot{b} - \ddot{p}_z - 2\dot{p}_x\dot{b})\sin b \tag{E-18}$$

$$\begin{aligned}\ddot{y} =&((d_z\ddot{a} + 2d_x\dot{a}\dot{b} + 2\dot{p}_z\dot{a})\cos b \\ &+ (d_x\ddot{a} - 2d_z\dot{a}\dot{b} + 2\dot{p}_x\dot{a})\sin b - d_y\dot{a}^2 + \ddot{y})\cos a \\ &+ ((d_x\ddot{b} - d_z(\dot{b}^2 + \dot{a}^2) + \ddot{p}_z + 2\dot{p}_x\dot{b})\cos b \\ &+ (\ddot{p}_x - 2\dot{p}_z\dot{b} - d_x(\dot{a}^2 + \dot{b}^2) - d_z\ddot{b})\sin b - d_y\ddot{a} - 2\dot{p}_y\dot{a})\sin a\end{aligned} \tag{E-19}$$

$$\begin{aligned}\ddot{z} =&((d_x\ddot{b} - d_z(\dot{a}^2 + \dot{b}^2) - 2\dot{p}_x\dot{b} + \ddot{p}_z)\cos b \\ &+ (\ddot{p}_x - 2\dot{p}_z\dot{b} - d_x(\dot{a}^2 + \dot{b}^2) - d_z\ddot{b})\sin b - d_y\ddot{a} - 2\dot{p}_y\dot{a})\cos a \\ &+ ((-d_z\ddot{a} - 2d_x\dot{a}\dot{b} - 2\dot{p}_z\dot{a})\cos b + (2d_z\dot{a}\dot{b} - d_x\ddot{a} - 2\dot{p}_x\dot{a})\sin b \\ &+ d_y\dot{a}^2 - \ddot{y})\sin a\end{aligned} \tag{E-20}$$

APPENDIX F

Source Code for Real-time Tool-path Sampling

The ISO C++ source code on the following pages, is for a command line program for sampling a tool-path and feedrate schedule with the algorithms proposed in this thesis. Inverse kinematics calculations are performed for a tilting-rotary table vertical machining centre with the table in the "horizontal" position. At least two parameters are required on the command line. The first is the servo update period (Δt) in seconds (i.e., `885.4e-6`). The second parameter is a filename for a file containing the MATLAB matrices defining the canonical parameters of a position spline, orientation spline, reparameterization spline, feedrate spline and inverse kinematics constant. A third command line argument may be given optionally. It specifies a filename to contain the output of the run. If this argument is not given then the program will execute the computations and for each servo update 1000 times and calculate benchmarks based on the results.

```
////////////////////////////////////////////////////////////////////////////////
//
// Interpolated Tool Path Sampling and Inverse Kinematics
// for a Tilting Rotary Table Five-axis CNC Machine Tool
```

```
//
// Copyright (c) 1999,2000 Robert V Fleisig
//
////////////////////////////////////////////////////////////////////////////

#include <cassert>
#include <complex>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <memory>
#include <string>
#include <vector>

using namespace std;

#ifdef _WIN32
#include <windows.h>
#endif
#include <mat.h>

const int no_output_repetitions = 1000;

template <class T> complex<T> tan( const complex<T>& number ) {
  return sin( number )/cos( number );
}

class Vector {
public:
  double x;
  double y;
  double z;

  Vector() {}

  Vector( const double _x, const double _y, const double _z ) :
    x( _x ), y( _y ), z( _z ) {}
};

inline Vector operator+( const Vector& left, const Vector& right ) {
  return Vector( left.x + right.x, left.y + right.y, left.z + right.z );
}

inline Vector operator-( const Vector& left, const Vector& right ) {
  return Vector( left.x - right.x, left.y - right.y, left.z - right.z );
}

inline double dot( const Vector& left, const Vector& right ) {
  return left.x*right.x + left.y*right.y + left.z*right.z;
```

```
}

inline Vector operator*( const double scalar, const Vector& _vector ) {
  return Vector( scalar*_vector.x, scalar*_vector.y, scalar*_vector.z );
}

inline Vector operator*( const Vector& _vector, const double scalar ) {
  return scalar*_vector;
}

inline Vector operator/( const Vector& _vector, const double scalar ) {
  return Vector( _vector.x/scalar, _vector.y/scalar, _vector.z/scalar );
}

class Matrix {

private:
  double* _matrix;
  mxArray* _matlab_matrix;

public:
  inline Matrix() : _matrix( NULL ), _matlab_matrix( NULL ) {}

  inline void initialize( MATFile* input_file, const string& matrix_name ) {

    _matlab_matrix = matGetArray( input_file, matrix_name.c_str() );
    assert( _matlab_matrix );

    _matrix = mxGetPr( _matlab_matrix );
    assert( _matrix );

    return;
  }

  inline int get_dimensions() const {
    assert( _matlab_matrix );
    return mxGetNumberOfDimensions( _matlab_matrix );
  }

  inline int get_first_dimension() const {
    assert( get_dimensions() >= 1 );
    const int* dim_array = mxGetDimensions( _matlab_matrix );
    return dim_array[0];
  }

  inline int get_second_dimension() const {
    assert( get_dimensions() >= 2 );
    const int* dim_array = mxGetDimensions( _matlab_matrix );
    return dim_array[1];
  }
```

```
  inline int get_third_dimension() const {
    assert( get_dimensions() >= 3 );
    const int* dim_array = mxGetDimensions( _matlab_matrix );
    return dim_array[2];
  }

  inline double operator[]( const int index ) const {
    return _matrix[index];
  }

  inline virtual ~Matrix() { mxDestroyArray( _matlab_matrix ); }
};

class MatlabOutputFile {
private:
  MATFile* _file;

public:
  inline MatlabOutputFile( const string& file_name ) : _file( NULL ) {
    _file = matOpen( file_name.c_str(), "w" );
    assert( _file );
  }

  inline void write_real_matrix( const string& name, const int rows,
                                 const vector<double>& array ) {

    assert( array.size() % rows == 0 );

    mxArray* matlab_array =
      mxCreateDoubleMatrix( rows, array.size()/rows, mxREAL );
    assert( matlab_array );

    copy( array.begin(), array.end(), mxGetPr( matlab_array ) );

    mxSetName( matlab_array, name.c_str() );
    ( void )matPutArray( _file, matlab_array );
    mxDestroyArray( matlab_array );

    return;
  }

  inline ~MatlabOutputFile() {
    ( void )matClose( _file );
  }
};

#ifdef _WIN32
class RealTimeProcess {
private:
```

```
  const HANDLE _process, _thread;
  DWORD _priority_class;
  int _thread_priority;
  bool _real_time;

public:
  RealTimeProcess() : _process( GetCurrentProcess() ),
    _thread( GetCurrentThread() ),
    _priority_class( GetPriorityClass( _process ) ),
    _thread_priority( GetThreadPriority( _thread ) ),
    _real_time( false ) {}

  void set_real_time() {
    if( !_real_time ) {
      ( void )SetPriorityClass( _process, REALTIME_PRIORITY_CLASS );
      ( void )SetThreadPriority( _thread, THREAD_PRIORITY_TIME_CRITICAL );
      _real_time = true;
    }
    return;
  }

  void reset_real_time() {
    if( _real_time ) {
      ( void )SetPriorityClass( _process, _priority_class );
      ( void )SetThreadPriority( _thread, _thread_priority );
      _real_time = false;
    }
    return;
  }

  DWORD get_priority_class() const { return GetPriorityClass( _process ); }

  int get_thread_priority() const { return GetThreadPriority( _thread ); }

  virtual ~RealTimeProcess() {
    reset_real_time();
  }
};
#endif

void display_copyright() {

  cout << endl
       << "Interpolated Tool Path Sampling and Inverse Kinematics "
       << endl
       << "for a Tilting-Rotary Table Five-axis CNC Machine Tool"
       << endl << endl
       << "Copyright (c) 1999,2000 Robert V Fleisig"
       << endl << endl;
```

```
  return;
}

void display_help() {

  cout << "Tool_Path_Sampling.exe sampling_period "
       << "input_file_name [output_file_name]"
       << endl << endl
       << "  sampling_period in seconds and >= zero"
       << endl
       << "  input_file_name must contain the requisite Matlab matrices"
       << endl
       << "  ouput_file_name will contain the output as Matlab matrices"
       << endl << endl;

  return;
}

bool parse_arguments( int argc, char* argv[],
                      double& sampling_period,
                      string& input_file_name,
                      string& output_file_name,
                      bool& output ) {

  enum { sampling_period_arg = 1,
         input_file_name_arg = 2,
         output_file_name_arg = 3 };

  const int last_arg = output_file_name_arg;

  if( !( argc == last_arg || argc == last_arg + 1 ) ) {
    display_help();
    return false;
  }

  sampling_period = atof( argv[sampling_period_arg] );

  if( sampling_period <= 0. ) {
    display_help();
    return false;
  }

  input_file_name = argv[input_file_name_arg];

  if( input_file_name.length() <= 0 ) {
    display_help();
    return false;
  }

  if( argc == last_arg )
```

```
    output = false;

  else {

    output_file_name = argv[output_file_name_arg];

    if( output_file_name.length() <= 0 ) {
      display_help();
      output = false;
    } else
      output = true;
  }
  return true;
}

void read_input_parameters( const string& input_file_name,
                            Matrix& feedrate_coefficients,
                            Matrix& feedrate_ranges,
                            Matrix& position_coefficients,
                            Matrix& position_ranges,
                            Matrix& reparameterization_coefficients,
                            Matrix& reparameterization_ranges,
                            Matrix& orientation_control_points,
                            Matrix& orientation_ranges,
                            Matrix& fixed_point,
                            Matrix& part_offset ) {

  MATFile* matlab_matrix_file = matOpen( input_file_name.c_str(), "r" );
  assert( matlab_matrix_file );

  feedrate_coefficients.initialize( matlab_matrix_file,
                                    "Feedrate_Coefficients" );

  assert( feedrate_coefficients.get_dimensions() == 2 );
  assert( feedrate_coefficients.get_first_dimension() == 3 );
  assert( feedrate_coefficients.get_second_dimension() > 0 );

  feedrate_ranges.initialize( matlab_matrix_file,
                              "Feedrate_Ranges" );

  assert( feedrate_ranges.get_dimensions() == 2 );
  assert( feedrate_ranges.get_first_dimension() == 1 );
  assert( feedrate_ranges.get_second_dimension() > 0 );

  assert( feedrate_coefficients.get_second_dimension() ==
          feedrate_ranges.get_second_dimension() );

  position_coefficients.initialize( matlab_matrix_file,
                                    "Position_Coefficients" );
```

```
assert( position_coefficients.get_dimensions() == 3 );
assert( position_coefficients.get_first_dimension() == 3 );
assert( position_coefficients.get_second_dimension() == 6 );
assert( position_coefficients.get_third_dimension() > 0 );

position_ranges.initialize( matlab_matrix_file,
                            "Position_Ranges" );

assert( position_ranges.get_dimensions() == 2 );
assert( position_ranges.get_first_dimension() == 1 );
assert( position_ranges.get_second_dimension() > 0 );

assert( position_coefficients.get_third_dimension() ==
        position_ranges.get_second_dimension() );

reparameterization_coefficients.
  initialize( matlab_matrix_file, "Reparameterization_Coefficients" );

assert( reparameterization_coefficients.get_dimensions() == 2 );
assert( reparameterization_coefficients.get_first_dimension() == 6 );
assert( reparameterization_coefficients.get_second_dimension() > 0 );

reparameterization_ranges.initialize( matlab_matrix_file,
                                      "Reparameterization_Ranges" );

assert( reparameterization_ranges.get_dimensions() == 2 );
assert( reparameterization_ranges.get_first_dimension() == 1 );
assert( reparameterization_ranges.get_second_dimension() > 0 );

assert( reparameterization_coefficients.get_second_dimension() ==
        reparameterization_ranges.get_second_dimension() );

orientation_control_points.initialize( matlab_matrix_file,
                                       "Orientation_Control_Points" );

assert( orientation_control_points.get_dimensions() == 3 );
assert( orientation_control_points.get_first_dimension() == 3 );
assert( orientation_control_points.get_second_dimension() == 6 );
assert( orientation_control_points.get_third_dimension() > 0 );

orientation_ranges.initialize( matlab_matrix_file,
                               "Orientation_Ranges" );

assert( orientation_ranges.get_dimensions() == 2 );
assert( orientation_ranges.get_first_dimension() == 1 );
assert( orientation_ranges.get_second_dimension() > 0 );

assert( orientation_control_points.get_third_dimension() ==
        orientation_ranges.get_second_dimension() );
```

```
  fixed_point.initialize( matlab_matrix_file, "Fixed_Point" );

  assert( fixed_point.get_dimensions() == 2 );
  assert( fixed_point.get_first_dimension() == 3 );
  assert( fixed_point.get_second_dimension() == 1 );

  part_offset.initialize( matlab_matrix_file, "Part_Offset" );

  assert( part_offset.get_dimensions() == 2 );
  assert( part_offset.get_first_dimension() == 3 );
  assert( part_offset.get_second_dimension() == 1 );

  matClose( matlab_matrix_file );

  return;
}

double get_duration( const Matrix& ranges ) {

  double sum = 0.;

  for( int index = 0; index < ranges.get_second_dimension();
       index++ )
    sum += ranges[index];

  return sum;
}

inline double sample_local_feedrate( const double local_parameter,
                                     const double coefficient1,
                                     const double coefficient2,
                                     const double coefficient3 ) {

  if( coefficient1 == 0. && coefficient2 == 0. )

    return coefficient3*local_parameter;

  else if( coefficient1 == 0. )

    return ( (exp( coefficient2*local_parameter +
                   log( complex<double>( coefficient3 ) ) ) -
              coefficient3 )/coefficient2 ).real();

  else if( coefficient2 == 0. ) {

    return ( -sqrt( complex<double>( coefficient3/coefficient1 ) )*
             tan( sqrt( complex<double>( coefficient1*coefficient3 ) )*
                  local_parameter ) ).real();
```

```
  } else if( coefficient2*coefficient2 == 4.*coefficient1*coefficient3 )

    return coefficient2*( 1./( 2.*coefficient1 -
                               coefficient1*coefficient2*local_parameter ) -
                          1./( 2.*coefficient1 ) );

  else if( coefficient2*coefficient2 > 4.*coefficient1*coefficient3 ) {

    const double alpha = sqrt( coefficient2*coefficient2 -
                               4.*coefficient1*coefficient3 );

    const complex<double> beta =
      exp( alpha*local_parameter +
           log( complex<double>( ( coefficient2 - alpha )/
                                 ( coefficient2 + alpha ) ) ) );

    return ( ( beta*( alpha + coefficient2 ) - coefficient2 + alpha )/
             ( 2.*coefficient1*( 1. - beta ) ) ).real();

  } else {

    const double alpha = sqrt( 4.*coefficient1*coefficient3 -
                               coefficient2*coefficient2 );

    return ( (alpha*tan( complex<double>( alpha*local_parameter/2. +
                                          atan2( coefficient2, alpha ) ) ) -
              coefficient2 )/( 2.*coefficient1 ) ).real();
  }
}

inline double sample_feedrate( const double parameter,
                               double& segment_parameter,
                               int& segment,
                               double& integration_constant,
                               const Matrix& coefficients,
                               const Matrix& ranges ) {

  while( parameter > segment_parameter + ranges[segment] ) {
    integration_constant +=
      sample_local_feedrate( ranges[segment],
                             coefficients[segment*3],
                             coefficients[segment*3+1],
                             coefficients[segment*3+2] );
    segment_parameter += ranges[segment++];
  }

  return sample_local_feedrate(parameter - segment_parameter,
                               coefficients[segment*3],
                               coefficients[segment*3+1],
                               coefficients[segment*3+2]) +
```

```
    integration_constant;
}

inline Vector sample_position( const double parameter,
                               double& segment_parameter,
                               int& segment,
                               const Matrix& coefficients,
                               const Matrix& ranges ) {

  while( parameter > segment_parameter + ranges[segment] )
    segment_parameter += ranges[segment++];

  const double local_parameter = parameter - segment_parameter;

  return Vector( ( ( ( ( coefficients[segment*18+15]*
                         local_parameter +
                         coefficients[segment*18+12] )*
                       local_parameter +
                       coefficients[segment*18+9] )*
                     local_parameter +
                     coefficients[segment*18+6] )*
                   local_parameter +
                   coefficients[segment*18+3] )*
                 local_parameter +
                 coefficients[segment*18],
                 ( ( ( (coefficients[segment*18+16]*
                        local_parameter +
                        coefficients[segment*18+13] )*
                       local_parameter +
                       coefficients[segment*18+10] )*
                     local_parameter +
                     coefficients[segment*18+7] )*
                   local_parameter +
                   coefficients[segment*18+4] )*
                 local_parameter +
                 coefficients[segment*18+1],
                 ( ( ( ( coefficients[segment*18+17]*
                         local_parameter +
                         coefficients[segment*18+14] )*
                       local_parameter +
                       coefficients[segment*18+11] )*
                     local_parameter +
                     coefficients[segment*18+8] )*
                   local_parameter +
                   coefficients[segment*18+5] )*
                 local_parameter +
                 coefficients[segment*18+2] );
}

inline double sample_reparameterization( const double parameter,
```

```
                                        double& segment_parameter,
                                        int& segment,
                                        const Matrix& coefficients,
                                        const Matrix& ranges ) {

  while( parameter > segment_parameter + ranges[segment] )
    segment_parameter += ranges[segment++];

  const double local_parameter = parameter - segment_parameter;

  return ( ( coefficients[segment*6]*local_parameter +
             coefficients[segment*6+1] )*
           local_parameter +
           coefficients[segment*6+2] )/
    ( ( coefficients[segment*6+3]*local_parameter +
        coefficients[segment*6+4] )*local_parameter +
      coefficients[segment*6+5] );
}

inline Vector spherical_interpolation( const double local_parameter,
                                       const Vector& vector1,
                                       const Vector& vector2 ) {

  const double alpha = dot( vector1, vector2 );

  const double beta = sqrt( 1. - alpha*alpha );

  const double gamma = acos( alpha );

  const double delta = sin( gamma*(1. - local_parameter ) );
  const double epsilon = sin( gamma*local_parameter );

  return ( vector1*delta + vector2*epsilon )/beta;
}

inline Vector sample_orientation( const double parameter,
                                  double& segment_parameter,
                                  int& segment,
                                  const Matrix& orientation_control_points,
                                  const Matrix& orientation_ranges ) {

  while( parameter > segment_parameter + orientation_ranges[segment] )
    segment_parameter += orientation_ranges[segment++];

  const double local_parameter = ( parameter - segment_parameter )/
    orientation_ranges[segment];

  const Vector control_point11 =
    spherical_interpolation( local_parameter,
                             Vector( orientation_control_points[segment*18],
```

```
                                orientation_control_points[segment*18+1],
                                orientation_control_points[segment*18+2]),
                        Vector( orientation_control_points[segment*18+3],
                                orientation_control_points[segment*18+4],
                                orientation_control_points[segment*18+5] ) );

const Vector control_point12 =
  spherical_interpolation( local_parameter,
                        Vector( orientation_control_points[segment*18+3],
                                orientation_control_points[segment*18+4],
                                orientation_control_points[segment*18+5] ),
                        Vector( orientation_control_points[segment*18+6],
                                orientation_control_points[segment*18+7],
                                orientation_control_points[segment*18+8] ) );

const Vector control_point13 =
  spherical_interpolation( local_parameter,
                        Vector( orientation_control_points[segment*18+6],
                                orientation_control_points[segment*18+7],
                                orientation_control_points[segment*18+8] ),
                        Vector( orientation_control_points[segment*18+9],
                                orientation_control_points[segment*18+10],
                                orientation_control_points[segment*18+11] ) );

const Vector control_point14 =
  spherical_interpolation( local_parameter,
                        Vector( orientation_control_points[segment*18+9],
                                orientation_control_points[segment*18+10],
                                orientation_control_points[segment*18+11] ),
                        Vector( orientation_control_points[segment*18+12],
                                orientation_control_points[segment*18+13],
                                orientation_control_points[segment*18+14] ) );

const Vector control_point15 =
  spherical_interpolation( local_parameter,
                        Vector( orientation_control_points[segment*18+12],
                                orientation_control_points[segment*18+13],
                                orientation_control_points[segment*18+14] ),
                        Vector( orientation_control_points[segment*18+15],
                                orientation_control_points[segment*18+16],
                                orientation_control_points[segment*18+17] ) );

const Vector control_point21 = spherical_interpolation( local_parameter,
                                                        control_point11,
                                                        control_point12 );

const Vector control_point22 = spherical_interpolation( local_parameter,
                                                        control_point12,
                                                        control_point13 );
```

```
  const Vector control_point23 = spherical_interpolation( local_parameter,
                                                          control_point13,
                                                          control_point14 );

  const Vector control_point24 = spherical_interpolation( local_parameter,
                                                          control_point14,
                                                          control_point15 );

  const Vector control_point31 = spherical_interpolation( local_parameter,
                                                          control_point21,
                                                          control_point22 );

  const Vector control_point32 = spherical_interpolation( local_parameter,
                                                          control_point22,
                                                          control_point23 );

  const Vector control_point33 = spherical_interpolation( local_parameter,
                                                          control_point23,
                                                          control_point24 );

  const Vector control_point41 = spherical_interpolation( local_parameter,
                                                          control_point31,
                                                          control_point32 );

  const Vector control_point42 = spherical_interpolation( local_parameter,
                                                          control_point32,
                                                          control_point33 );

  return spherical_interpolation( local_parameter,
                                  control_point41,
                                  control_point42 );
}

inline void inverse_kinematics( const Vector& fixed_point,
                                const Vector& part_offset,
                                const Vector& _position,
                                const Vector& orientation,
                                double& axis_x,
                                double& axis_y,
                                double& axis_z,
                                double& axis_a,
                                double& axis_c ) {

  const Vector position = _position + part_offset;

  axis_a = -acos( orientation.z );
  axis_c = atan2( -orientation.x, -orientation.y );

  const Vector delta = position - fixed_point;
  const double alpha = 1. - orientation.z*orientation.z;
```

```
  const double beta = sqrt(alpha);
  const double gamma = delta.x*orientation.x + delta.y*orientation.y;

  axis_x = ( delta.y*orientation.x - delta.x*orientation.y )/beta +
    fixed_point.x;
  axis_y = ( delta.z*alpha - orientation.z*gamma )/beta + fixed_point.y;
  axis_z = gamma + delta.z*orientation.z + fixed_point.z;

  return;
}

void write_output_data( const string& output_file_name,
                        vector<double>& sampled_time,
                        vector<double>& sampled_arc_distance,
                        vector<double>& sampled_position,
                        vector<double>& sampled_angular_arc_distance,
                        vector<double>& sampled_orientation,
                        vector<double>& sampled_axis_commands ) {

  auto_ptr<MatlabOutputFile>
    output_file( new MatlabOutputFile( output_file_name ) );

  output_file->write_real_matrix( "Sampled_Time", 1, sampled_time );

  output_file->write_real_matrix( "Sampled_Arc_Distance", 1,
                                  sampled_arc_distance );

  output_file->write_real_matrix( "Sampled_Position", 3, sampled_position );

  output_file->write_real_matrix( "Sampled_Angular_Arc_Distance", 1,
                                  sampled_angular_arc_distance );

  output_file->write_real_matrix( "Sampled_Orientation", 3,
                                  sampled_orientation);

  output_file->write_real_matrix( "Sampled_Axis_Commands", 5,
                                  sampled_axis_commands);

  return;
}

void print_results( const double best_time_seconds,
                    const double best_time_sampling_period,
                    const double worst_time_seconds,
                    const double worst_time_sampling_period,
                    const double average_time_seconds,
                    const double average_time_sampling_period ) {

  cout << "Best time: "
       << best_time_seconds
```

```
        << " seconds" << endl
        << "            "
        << best_time_sampling_period
        << "% of sampling period"
        << endl << endl
        << "Worst time: "
        << worst_time_seconds
        << " seconds" << endl
        << "             "
        << worst_time_sampling_period
        << "% of sampling period"
        << endl << endl
        << "Average time: "
        << average_time_seconds
        << " seconds" << endl
        << "              "
        << average_time_sampling_period
        << "% of sampling period"
        << endl << endl;

  return;
}

int main( int argc, char* argv[] ) {

  display_copyright();

  double sampling_period;
  string input_file_name, output_file_name;
  bool output;

  if( !parse_arguments( argc, argv, sampling_period,
                        input_file_name, output_file_name, output ) )
    return EXIT_FAILURE;

  Matrix feedrate_coefficients,
    feedrate_ranges,
    position_coefficients,
    position_ranges,
    reparameterization_coefficients,
    reparameterization_ranges,
    orientation_control_points,
    orientation_ranges,
    fixed_point,
    part_offset;

  read_input_parameters( input_file_name,
                         feedrate_coefficients,
                         feedrate_ranges,
                         position_coefficients,
```

```
                        position_ranges,
                        reparameterization_coefficients,
                        reparameterization_ranges,
                        orientation_control_points,
                        orientation_ranges,
                        fixed_point,
                        part_offset );

  vector<double> sampled_time,
    sampled_arc_distance,
    sampled_position,
    sampled_angular_arc_distance,
    sampled_orientation,
    sampled_axis_commands;

  double time = 0.,
    segment_time = 0.,
    feedrate_integration_constant = 0.,
    arc_distance,
    segment_arc_distance = 0.,
    segment_reparameterization = 0.,
    angular_arc_distance,
    segment_angular_arc_distance = 0.,
    axes[5];

  Vector position,
    orientation;

  int feedrate_segment = 0,
    position_segment = 0,
    reparameterization_segment = 0,
    orientation_segment = 0;

  const double duration = get_duration( feedrate_ranges );

  const int repetitions = output ? 1 : no_output_repetitions;

#ifdef _WIN32
  RealTimeProcess real_time_process;
  real_time_process.set_real_time();
#endif

  unsigned long sample = 0;
  clock_t worst_time = 0,
    best_time = CLOCKS_PER_SEC,
    total_time_elapsed = 0,
    start_sample = clock();

  while( time <= duration ) {
```

```
    sample++;

    for( int repetition = 0; repetition < repetitions; repetition++ ) {

      arc_distance = sample_feedrate( time,
                                      segment_time,
                                      feedrate_segment,
                                      feedrate_integration_constant,
                                      feedrate_coefficients,
                                      feedrate_ranges );

      position = sample_position( arc_distance,
                                  segment_arc_distance,
                                  position_segment,
                                  position_coefficients,
                                  position_ranges );

      angular_arc_distance =
        sample_reparameterization( arc_distance,
                                   segment_reparameterization,
                                   reparameterization_segment,
                                   reparameterization_coefficients,
                                   reparameterization_ranges );

      orientation = sample_orientation( angular_arc_distance,
                                        segment_angular_arc_distance,
                                        orientation_segment,
                                        orientation_control_points,
                                        orientation_ranges );

      inverse_kinematics( Vector( fixed_point[0],
                                  fixed_point[1],
                                  fixed_point[2] ),
                          Vector( part_offset[0],
                                  part_offset[1],
                                  part_offset[2] ),
                          position,
                          orientation,
                          axes[0],
                          axes[1],
                          axes[2],
                          axes[3],
                          axes[4] );
    }

    const clock_t end_sample = clock();
    const clock_t sampled_time_elapsed = end_sample - start_sample;
    total_time_elapsed += sampled_time_elapsed;

    if(sampled_time_elapsed > worst_time) worst_time = sampled_time_elapsed;
```

```
    if(sampled_time_elapsed < best_time) best_time = sampled_time_elapsed;

    if( output ) {
      sampled_time.push_back( time );
      sampled_arc_distance.push_back( arc_distance );
      sampled_position.push_back( position.x );
      sampled_position.push_back( position.y );
      sampled_position.push_back( position.z );
      sampled_angular_arc_distance.push_back( angular_arc_distance );
      sampled_orientation.push_back( orientation.x );
      sampled_orientation.push_back( orientation.y );
      sampled_orientation.push_back( orientation.z );
      sampled_axis_commands.push_back( axes[0] );
      sampled_axis_commands.push_back( axes[1] );
      sampled_axis_commands.push_back( axes[2] );
      sampled_axis_commands.push_back( axes[3] );
      sampled_axis_commands.push_back( axes[4] );
    }
    start_sample = clock();
    time += sampling_period;
  }
  const unsigned long samples = sample;

#ifdef _WIN32
  real_time_process.reset_real_time();
#endif

  if( output )
    write_output_data( output_file_name,
                       sampled_time,
                       sampled_arc_distance,
                       sampled_position,
                       sampled_angular_arc_distance,
                       sampled_orientation,
                       sampled_axis_commands );
  else {

    const double best_time_seconds =
      best_time/( double )CLOCKS_PER_SEC/repetitions;

    const double best_time_sampling_period =
      best_time/( double )CLOCKS_PER_SEC/sampling_period/repetitions*100.;

    const double worst_time_seconds =
      worst_time/( double )CLOCKS_PER_SEC/repetitions;

    const double worst_time_sampling_period =
      worst_time/( double )CLOCKS_PER_SEC/sampling_period/repetitions*100.;

    const double average_time_seconds =
```

```
      total_time_elapsed/( double )CLOCKS_PER_SEC/samples/repetitions;

    const double average_time_sampling_period =
      total_time_elapsed/( double )CLOCKS_PER_SEC/samples/
      sampling_period*100./repetitions;

    print_results( best_time_seconds,
                   best_time_sampling_period,
                   worst_time_seconds,
                   worst_time_sampling_period,
                   average_time_seconds,
                   average_time_sampling_period );
  }
  return EXIT_SUCCESS;
}
```

References

[1] Farid Abrari. *Multi-axis milling of flexible parts.* PhD thesis, McMaster University, 1998.

[2] J Harold Ahlberg, Edwin N Nilson, and Joseph L Walsh. *The theory of splines and their applications*, volume 38 of *Mathematics in science and engineering.* Academic Press, New York, NY, USA, 1967.

[3] American National Standards Institute. *Information technology—programming language APT: processor input language and system-neutral CLfile.* ANSI NCITS 37-1999 edition, 1999.

[4] Allan H Barr, Bena Currin, Steven Gabriel, and John F Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Computer Graphics (ACM)*, 26(2):313-320, July 1992.

[5] Sanjeev Bedi, I Ali, and N Quan. Advanced interpolation techniques for N.C. machines. *ASME Journal of Engineering for Industry*, 115(3):329-336, August 1993.

[6] C Bergren. A simple algorithm for circular interpolation. *Control Engineering*. 18:57–59. September 1971.

[7] Oene Bottema and Bernard Roth. *Theoretical kinematics*. North-Holland Publishing Company. Amsterdam. Netherlands. 1979.

[8] YD Chen. J Ni. and SM Wu. Real-time CNC tool path generation for machining IGES surfaces. *ASME Journal of Engineering for Industry*. 115(4):480–486. 1993 1993.

[9] Inhaeng Cho. Kunwoo Lee. and Jongwon Kim. Generation of collision-free cutter location data in five-axis milling using the potential energy method. *International Journal of Advanced Manufacturing Technology*. 13(8):523–529. 1997.

[10] Byong K Choi and Robert B Jerard. *Sculptured surface machining*. Kluwer Academic Publishers. Dordrecht. Netherlands. 1998.

[11] Jui-Jen Chou and Daniel CH Yang. Command generation for three-axis CNC machining. *ASME Journal of Engineering for Industry*. 113(3):305–310. August 1991.

[12] Jui-Jen Chou and Daniel CH Yang. On the generation of coordinated motion for five-axis CNC/CMM machines. *ASME Journal of Engineering for Industry*. 114(1):15–22. February 1992.

[13] Intel Corporation. *Intel Architecture Software Developer's Manual. Volume 1: Basic Architecture*. 1999.

[14] Intel Corporation. *Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference Manual*, 1999.

[15] Intel Corporation. *Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide*, 1999.

[16] Per E Danielsson. Incremental curve generation. *IEEE Transactions on Computers*, C-19(9):783–93, September 1970.

[17] R Delbourgo and JA Gregory. C^2 rational quadratic spline interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 3:141–152, 1983.

[18] Don Dragomatz and Stephen Mann. A classified bibliography of literature on NC milling path generation. *Computer-Aided Design*, 29(3):239–247, March 1997.

[19] Electronic Industries Alliance. *Machines, numerically controlled, interchangeable perforated tape variable block format for contouring & contouring/positioning*, EIA-RS274D-79 edition, March 1979.

[20] Electronic Industries Alliance. *Subset of USA standard code for information interchange for numerical control perforated tape*, EIA-RS358B-79 edition, March 1979.

[21] Gerald Farin. *Curves and surfaces for computer aided geometric design—a practical guide*. Academic Press, Boston, MA, USA, third edition, 1993.

[22] Rida T Farouki and Takis Sakkalis. Real rational curves are not 'unit speed'. *Computer Aided Geometric Design*, 8(2):151–157, May 1991.

[23] Ivor D Faux and Michael J Pratt. *Computational geometry for design and manufacture*. Ellis Horwood, Chichester, UK, 1979.

[24] Andrew D Fisher, Robert V Fleisig, and Allan D Spence. A distributed object architecture CMM interface pendant. In *Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference PROLAMAT 98 The Globalization of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility and the Virtual Enterprise*, Trento, Italy, September 1998. University of Trento.

[25] Robert V Fleisig. Compressor and turbine blade tip profiling. Master's thesis, McMaster University, Hamilton, Ontario, Canada, December 1996.

[26] Robert V Fleisig, Peter V Saturley, and Allan D Spence. A research testbed for common, bi-directional information flow between design and manufacturing. In *Proceedings 31st CIRP International Seminar on Manufacturing Systems*, pages 58–63, Berkeley, California, USA, May 1998. University of California Berkeley.

[27] Robert V Fleisig, Peter V Saturley, and Allan D Spence. A research testbed for common, bi-directional information flow between design and manufacturing. *CIRP Journal of Manufacturing Systems*, 29(1):45–49, 1999.

[28] Robert V Fleisig and Allan D Spence. CAD/CAM integration for five-axis closed loop machining. In *Proceedings CSME Forum 1996 Engineering Applications of Mechanics "Manufacturing Science and Engineering"*, pages 402–409, Hamilton, Ontario, Canada, May 1996. McMaster University.

[29] Robert V Fleisig and Allan D Spence. Integrated digitizing, path planning, and five-axis machining for the refurbishment of compressor and turbine blades. In *Proceedings of the ASME dynamic systems and control division*, volume 59, pages 31–38, Dallas, Texas, USA, November 1997.

[30] Robert V Fleisig and Allan D Spence. Ballnose cutter contact point calculation for five-axis milling. In *17th Canadian Congress of Applied Mechanics CANCAM 99*, pages 365–366, 1999.

[31] Robert V Fleisig and D Spence, Allan. A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining. *Computer-Aided Design*, 2000.

[32] Thomas A Foley, David A Lane, and Gregory M Nielson. Visualizing functions over a sphere. *IEEE Computer Graphics and Applications*, 10(1):32–40, January 1990.

[33] FN Fritsch and RE Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 22(2):386–400, April 1980.

[34] Q Jeffrey Ge. *Computational kinematics*, volume 28 of *Solid mechanics and its applications*, chapter Computational geometry and motion approximation, pages 229–238. Kluwer Academic Publishers, Dordrecht, Netherlands, October 1993.

[35] Q Jeffrey Ge. Kinematics-driven geometric modeling: a framework for simultaneous NC tool-path generation and sculptured surface design. In *Proceedings of the 1996 13th IEEE International Conference on Robotics and Automation*, volume 2, pages 1819–1824, 1996.

[36] Q Jeffrey Ge and Kang Donglai. Motion interpolation with G^2 composition Bézier motions. *ASME Journal of Mechanical Design*, 117(4):520–525, December 1995.

[37] Q Jeffrey Ge and Bahram Ravani. Computer aided geometric design of motion interpolants. *ASME Journal of Mechanical Design*, 116(2):756–762, September 1994.

[38] Q Jeffrey Ge and Bahram Ravani. Geometric construction of Bézier motions. *ASME Journal of Mechanical Design*, 116(3):749–755, September 1994.

[39] Herbert Goldstein. *Classical mechanics*. Addison-Wesley Press, Cambridge, MA, USA, second edition, 1980.

[40] JA Gregory and R Delbourgo. Piecewise rational quadratic interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 2:123–130, 1982.

[41] Cristoph M Hoffmann. *Solid and geometric modeling: an introduction*. Morgan Kaufmann Publishers, Inc., 1989.

[42] Rediar Hovtun. LOOPS: 5-axis milling for all machining jobs. In *Proceedings 31st CIRP International Seminar on Manufacturing Systems*, pages 431–435, May 1996.

[43] Jung-Tang Huang. *Design and application of a new CNC command generator for CAD/CAM integration*. PhD thesis, University of California Los Angeles, 1992.

[44] Jung-Tang Huang and Daniel CH Yang. A generalized interpolator for command generation of parametric curves incomputer-controlled machines. In *Proceedings of 1992 Japan-USA Symposium on Flexible Automation*, volume 1, pages 393–399. ASME, 1992.

[45] Yunching Huang and James H Oliver. Integrated simulation, error assessment, and tool path correction for five-axis NC milling. *Journal of Manufacturing Systems*. 14(5):331–344. 1995.

[46] JS Hwang. Interference-free tool-path generation in the NC machining of parameric compound surfaces. *Computer-Aided Design*. 24(12):667–676. December 1992.

[47] CG Jensen and DC Anderson. Accurate tool placement and orientation for finish surface machining. In *Proceedings of the ASME Production Engineering Division. Manufacturing Science and Engineering*, volume 59, pages 127–145, 1992.

[48] CG Jensen and DC Anderson. A review of numerically controlled methods for finish-sculptured-surface machining. *IIE Transactions*. 28(1):30–39. January 1996.

[49] Bert Jüttler. Visualization of moving objects using dual quaternion curves. *Computers & Graphics*. 18(3):315–326. 1994.

[50] Bert Jüttler and Michael G Wagner. Computer-aided design with spatial rational B-spline motions. *ASME Journal of Mechanical Design*, 118(2):193–201, June 1996.

[51] Myung-Soo Kim and Kee-Won Nam. Interpolating solid orientations with circular blending quaternion curves. *Computer-Aided Design*, 27(5):385–398, May 1995.

[52] Dimitris Kiritsis. High precision interpolation algorithm for 3D parametric curve generation. *Computer-Aided Design*, 26(11):850–856, November 1994.

[53] Yoram Koren. Interpolator for a computer numerical control system. *IEEE Transactions on Computers*, C-25(1):32–37, January 1976.

[54] Yoram Koren. Design of computer control for manufacturing systems. *ASME Journal of Engineering for Industry*, 101:326–332. August 1979.

[55] Yoram Koren and Rong-Shine Lin. Five-axis surface interpolators. *CIRP Annals*, 44(1):379–382. 1995.

[56] Yoram Koren, Chih-Ching Lo, and Moshe Shpitalni. CNC interpolators: algorithms and analysis. In *Proceedings of the ASME Production Engineering Division. Manufacturing Science and Engineering*, volume 64, pages 83–92, 1993.

[57] Alain Liégeois. *Performance and computer-aided design*, volume 7 of *Robot technology*. Kogan Page, London, UK, 1985.

[58] Anis Limaiem. *Computer-aided CMM inspection, planning and verification*. PhD thesis, McMaster University, 1996.

[59] Rong-Shine Lin and Yoram Koren. Real-time five-axis interpolators for machining ruled surfaces. In *Proceedings of the ASME Dynamics Systems and Control Division*, volume 55, pages 951–960, 1994.

[60] Rong-Shine Lin and Yoram Koren. Efficient tool-path planning for machining free-form surfaces. *ASME Journal of Engineering for Industry*, 118(1):20–28, February 1996.

[61] Rong-Shine Lin and Yoram Koren. Real-time interpolators for multi-axis CNC machine tools. *Journal of Manufacturing Systems*, 25(2):145–149, 1996.

[62] Xiong-Wei Liu. Five-axis NC cylindrical milling of sculptured surfaces. *Computer-Aided Design*. 27(12):887–894. 1995.

[63] Chih-Ching Lo. *Cross-coupling control of multi-axis manufacturing*. PhD thesis. University of Michigan. 1992.

[64] Chih-Ching Lo. Real-time generation and control of cutter path for 5-axis CNC machining. *International Journal of Machines Tools & Manufacture*. 39:471–488. 1999.

[65] H Makino. Clothoidal interpolation—a new tool for high-speed continuous path control. *CIRP Annals*. 37(1):25–28. 1988.

[66] H Makino and T Ohde. Motion control of the direct drive actuator. *CIRP Annals*. 40(1):375–378. 1991.

[67] ME Martellotti. An analysis of the milling process. *Transactions of the ASME*. 63(1):677–700. January 1941.

[68] ME Martellotti. An analysis of the milling process part II: down milling. *Transactions of the ASME*. 67(1):233–251. January 1945.

[69] O Masory and Yoram Koren. Reference-pulse circular interpolators for CNC systems. *ASME Journal of Engineering for Industry*, 103(1):131–136, February 1981.

[70] O Masory and Yoran Koren. Reference-word circular interpolators for CNC systems. *ASME Journal of Engineering for Industry*, 104(4):400–405, November 1982.

[71] J Michael McCarthy. The differential geometry of curves in an image space of spherical kinematics. *Mechanism and Machine Theory*, 22(3):205–211. 1987.

[72] J Michael McCarthy. *An introduction to theoretical kinematics*. The MIT Press. Cambridge. MA. USA. February 1990.

[73] Koichi Morishige. Yoshimi Takeuchi. and Kiwamu Kase. Tool path generation using C-space for 5-axis control machining. *ASME Journal of Manufacturing Science and Engineering*. 121:144–149. February 1999.

[74] Erik Oberg. Franklin D Jones. Holbrook L Horton. and Henry H Ryffel. *Machinery's handbook*. Industrial Press. New York. NY. USA. 25th edition. 1996.

[75] Spiros G Papaioannou. Interpolation algorithms for numerical control. *Computers in Industry*. 1:27–40. 1979.

[76] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer-Verlag. Berlin. Germany. second edition. 1997.

[77] Daniel Pletinckx. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer*. 5:2–13. 1989.

[78] William H Press. Saul A Teukolsky. William T Vetterling. and Brian P Flannery. *Numerical recipes in C the art of scientific computing*. Cambridge University Press. Cambridge. UK. second edition. 1992.

[79] Bahram Ravani and Frank C Park. Bézier curves on Riemannian manifolds and lie groups with kinematic applications. *ASME Journal of Mechanical Design*. 117(1):36–40. March 1995.

[80] Jean-Max Redonnet, Walter Buio, and Gilles Dessein. Side milling of ruled surfaces: optiumum positioning of the milling cutter and calculation of interference. *International Journal of Advanced Manufacturing Technology*, 14(7):459–467, 1998.

[81] Gabor Renner. A method of shape description for mechanical engineering practice. *Computers in Industry*, 3:137–142, 1982.

[82] Walter Rubio, Pierre Lagarrigue, Gilles Dessein, and François Paster. Calculation of tool paths for a torus mill on free-form surfaces on five-axis machines with detection and elimination of interference. *International Journal of Advanced Manufacturing Technology*, 14(1):13–20, 1998.

[83] A Rüegg and P Gygax. Generalized kinematics model for three- to five-axis milling machines and their implementation in a CNC. *CIRP Annals*, 41(1):547–550, 1992.

[84] Radha Sarma and Aarthi Rao. Discretizers and interpolators for five-axis CNC machines. In *Proceedings of the ASME Dynamics Systems and Control Division*, volume 64, pages 447–454, 1998.

[85] T Sata, F Kimura, N Okada, and M Hosaka. A new method of NC interpolation for machining the sculptured surfaces. *Annals of the CIRP*, 30(1):369–372, 1981.

[86] Steven Schofield and Paul K Wright. Open architecture controllers for machine tools, part 1: design principles. *ASME Journal of Manufacturing Science and Engineering*, 120:417–424, 1998.

[87] Milton C Shaw. *Metal cutting principles*, volume 3 of *Oxford Series on Advanced Manufacturing*. Oxford University Press, Oxford, UK, 1989.

[88] Ken Shoemake. Animating rotation with quaternion curves. *Computer Graphics (ACM)*. 19(3):245–254. July 1985.

[89] Moshe Shpitalni. Yoram Koren, and Chih-Ching Lo. Realtime curve interpolators. *Computer-Aided Design*. 26(11):832–838. November 1994.

[90] Alexander H Slocum. *Precicion machine design*. Prentice-Hall. Englewood Cliffs. NJ. USA. 1988.

[91] Sandeep Sood. Paul K Wright. and Jane MacFarlane. Process planning: a review. In *Proceedings of the ASME Design Engineering Division*. volume 66. pages 45–54. 1993.

[92] Allan D Spence. *Solid modeller based milling process simulation*. PhD thesis. University of British Columbia. March 1992.

[93] Allan D Spence. Farid Abrari. and MA Elbestawi. Integrated solid modeller based solutions for machining. *Computer-Aided Design*. 32:553–568. 2000.

[94] Allan D Spence. Robert V Fleisig. and Nael A Barakat. Thermally corrected volumetric error compensation for coordinate measuring machines. In *Proceedings of the 33rd International MATADOR Conference*. 2000.

[95] Lakshmi N Srinivasan and Q Jeffrey Ge. Parametric continuous and smooth motion interpolation. *ASME Journal of Mechanical Design*. 118(4):494–498. December 1996.

[96] Lakshmi N Srinivasan and Q Jeffrey Ge. Fine tuning of rational B-spline motions. *ASME Journal of Mechanical Design*. 120(1):46–51, March 1998.

[97] R Stadelmann. Computation of nominal path values to generate varisou spacial curves for machine tools. *CIRP Annals*, 38(1):373–376, 1989.

[98] Suk-Hwan Suh, Jung-Hoon Cho, and Jean-Yves Hascoet. Incorporation of tool deflection in tool path computation: simulation and analysis. *Journal of Manufacturing Systems*, 15(3):190–199, 1996.

[99] Richard Teltz and M A Elbestawi. Hierarchical, knowledge-based control in turning. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 115(1):122–132, March 1993.

[100] Richard W Teltz, Ali Shawky, Keith Urbasik, and Mohamed A Elbestawi. Sensor based planning and control for open architecture control. In *Technical Paper—Society of Manufacturing Engineers*, volume MS95-145. NAMRI SME, 1995.

[101] Stephen Veldhuis and Mohamed A Elbestawi. Strategy for the compensation of error in five-axis machining. *CIRP Annals*, 44(1):373–377, 1995.

[102] GW Vickers and Quan KW. Ball-mills versus end-mills for curved surface machining. *ASME Journal of Engineering for Industry*, 111(22):22–26, February 1989.

[103] Fu-Chung Wang, Steven Schofield, and Paul K Wright. Real time quintic spline interpolator for an open architecture machine tool. In *Proceedings of the ASME Dynamics Systems and Control Division*, volume 58, pages 291–297, 1996.

[104] Fu-Chung Wang and Paul K Wright. Open architecture controllers for machine tools, part 2: a real time quintic spline interpolator. *ASME Journal of Manufacturing Science and Engineering*, 120:425–432, 1998.

[105] Fu-Chung Wang and Daniel CH Yang. Nearly arc-length parameterized quintic-spline interpolation for precision machining. *Computer-Aided Design*. 25(5):281–288. May 1993.

[106] Andrew Warkentin. Sanjeev Bedi. and Fadi Ismail. Five-axis milling of spherical surfaces. *International Journal of Machines Tools & Manufacture*. 36(2):229–243. 1996.

[107] Paul K Wright. Principles of open-architecture manufacturing. *Journal of Manufacturing Systems*. 14(3):187–202, 1995.

[108] Daniel CH Yang and Jui-Jen Chou. Automatic generation of piecewise constant speed motion with smooth transition for multi-axis machines. *ASME Journal of Mechanical Design*. 116(2):581–586. June 1994.

[109] Daniel CH Yang and Alex D Golub. Precision trajectory generation for CNC milling of arbitrary contours and surfaces. *International Journal of Machines Tools & Manufacture*. 34(7):1005–1018. October 1994.

[110] Daniel CH Yang and Tom Kong. Parametric interpolator versus linear interpolator for precision CNC machining. *Computer-Aided Design*, 26(3):225–234. March 1994.

[111] Daniel CH Yang and Fu-Chung Wang. A quintic spline interpolator for motion command generation of computer-controlled machines. *ASME Journal of Mechanical Design*, 116:226–231, March 1994.

[112] S Yang, J Yuan, and J Ni. Accuracy enhancement of a horizontal machining center by real-time error compensation. *Journal of Manufacturing Systems*, 15(2):113–124, 1996.

[113] Millan K Yeung and Desmond J Walton. Curve fitting with arc splines for NC toolpath generation. *Computer-Aided Design*. 26(11):845–8459. November 1994.

[114] Chun-Fong You and Chih-Hsing Chu. Tool-path verification in five-axis machining of sculptured surfaces. *International Journal of Advanced Manufacturing Technology*. 13(4):248–255. 1997.