Arc-Length Parameterized NURBS Tool Path Generation and Velocity Profile Planning for Accurate 3-Axis Curve Milling

Yangtao Li

A Thesis

In the Department

of

Mechanical & Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Mechanical Engineering) at

Concordia University

Montreal, Quebec, Canada

September 2012

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the Thesis prepared,

By: **Yangtao Li**

Entitled: **"Arc-Length Parameterized NURBS Tool Path Generation and Velocity Profile Planning for Accurate 3-Axis curve Milling"**

and submitted in partial fulfillment of the requirements for the Degree of

Master of Applied Science (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. G. Gouw

_____ Examiner
Dr. Y. Zhang

_____ Examiner
Dr. Z.W. Tian External
Concordia Institute for Information Systems Engineering

_____ Supervisor
Dr. Z.C. Chen

Approved by: _____
Dr. S. Narayanswamy, MASc Program Director
Department of Mechanical and Industrial Engineering

_____
Dean Robin Drew
Faculty of Engineering & Computer Science

Date: _____

Abstract

Arc-Length Parameterized NURBS Tool Path Generation and Velocity Profile Planning for Accurate 3-Axis Curve Milling

Yangtao Li

In modern industrial CNC (Computer Numerical Control) machining processes, the pursuing of higher accuracy and efficiency has always been one of the most important tasks to be discussed and studied. A lot of proposed algorithms are developed in order to optimize the machining performance in either of the above focused domains. Nevertheless, there is forever a trade-off between gaining less machining error and providing higher feed rate. As for machining a free-shaped curve (e.g., Bezier curves, B-splines and NURBS) in a three-dimensional space, a better manner to balance out the aforementioned trade-offs turns out to be even more critical and essential.

The conventional iterative function used for tool path generation could cause feed rate fluctuation during the actual machining, and it thus might lead to failure on constraining the error within the machining accuracy requirement. Another potential problem occurs when the machining process comes across into a relatively high curvature segment with the prescribed high feed rate, due to the machine axial acceleration limit, the machine may not be able to maintain the tool tip trajectory within the error tolerance. Therefore, a new approach to NURBS tool path generation

for high feed rate machining is proposed. In this work, several criterions are set for checking the viability of the prescribed feed rate and adjusting it according to the actual shape of the objective curve and the capability of the machine. After the offline feed rate viability check and readjustment, a new iterative algorithm based on the arc-length re-parameterized NURBS function would be implemented to calculate the tool path in real-time.

By using this proposed method, the feed rate fluctuation is diminished and the overall efficiency of the machining process would have been optimized under the condition of accuracy guaranteed.

Acknowledgements

I hereby wish to express my sincere gratitude and thanks to my supervisor, Dr. Chevy Chen, for his precious guidance and patient tutorials that lead me through out this work. Not only to mention how extraordinary he is as my supervisor in my research, but also he is truly a teacher of my spirit. It was him who keeps propelling me forward and changes me a lot with the example of his own meticulous attitude on doing researches.

I also would like to pass my appreciation to all the dear fellows in our lab, for the tireless help you generously offered when I encountered problems and doubts. I could have needed to face a lot more obstacles to finish this work without your help.

Finally, I give my special thanks to my parents, especially my mother, for giving me all their selfless love, as always, and supporting me in every aspect during my study.

CONTENTS

CHAPTER 1 INTRODUCTION 1

1.1 Research Problems 1
1.1.1. B-spline and NURBS Curve *1*
1.1.2. Arc-length Approximation *3*
1.1.3. NURBS Tool Path Interpolation *4*
1.2 Literature Review 5
1.3 Objective of the Research 8
1.4 Thesis Outline 9

CHAPTER 2 NON-ARC-LENGTH NURBS TOOL PATH AND ARC-LENGTH NURBS TOOL PATH GENERATION 10

2.1 Introduction 10
2.2 Algorithm of Non-arc-length NURBS Tool Path 11
2.3 Algorithm of Arc-length NURBS Tool Path 14
2.3.1 Properties of Arc-length NURBS Curve *14*
2.3.2 Arc-Length NURBS Interpolation *15*
2.4 Comparison of Non-Arc-Length and Arc-Length NURBS Tool Path Generation Methods 16
2.4.1 Comparison on Tool Path Error and Feed Rate Error *16*
2.4.2 Comparison on Computational Efficiency *18*

CHAPTER 3 ARC LENGTH CALCULATION USING QUADRATURE METHODS 19

3.1 Introduction 19
3.2 Derivation of Mathematical Equations for Arc Length Approximation 19
3.3 Common Quadrature Methods 22
3.3.1 Midpoint Rule and Trapezoid Rule *22*
3.3.2 Simpson's Method *24*
3.3.2.1 Derivation of Simpson's Rule 24
3.3.2.2 Error Estimation for Simpson's Rule 27
3.4 Arc length calculation for B-splines and NURBS 28
3.5 Curvature-adaptive based Arc Length Calculation for Parametric Curves Using Composite Simpson's Method 29
3.5.1 Rough Tempt on Optimizing Calculation Efficiency *29*
3.5.2 Curvature-recognition based B-spline arc length approximation *34*

3.6 A New Approach to Arc-length Approximation for NURBS Curves with Sharp Corners 39
3.7 A Potential Problem in NURBS Arc-Length Approximation When Using Adaptive Simpson's Method 49

CHAPTER 4 PROPOSED ARC-LENGTH PARAMETERIZED NURBS TOOL PATH GENERATION 54

4.1 Introduction 54
4.2 Accuracy Bounded Piecewise Feed Rate Adjustment 55
4.2.1 Principle of Finding Maximum Valid Feed Rate According to Machine Properties and Curvature of the Curve 55
4.2.2 Curvature-Based Segmentation with Afterwards Feed Rate Adjustment 57
4.3 Pre-machining Valid Feed Rate Adjustment with Bounded Axial Acceleration 59
4.3.1 Derivation of Axial Acceleration Functions for Each Axis 60
4.3.2 Feed Rate Re-adjustment Method According to the Machine Axial Acceleration Limit 63
4.4 Feed Rate Profile Derivation During the Speed Change Procedure between Adjacent Sub-intervals 66
4.5 Arc-length Parameterized NURBS Tool Path Generation Method 73

CHAPTER 5 APPLICATIONS 77

5.1 Introduction 77
5.2 Example of 3-Axis Arc-length NURBS Tool Path Generation 78
5.3 Summery 87

CHAPTER 6 CONCLUSIONS AND FUTURE WORK 88

6.1 Conclusions 88
6.2 Future work 89

CHAPTER 7 REFERENCES 91

List of Tables

Table 2.1 Comparison on governing functions of non-arc-length and arc-length NURBS..... 18

Table 3.1 Performance comparison on function $f(x)=\sin(8\cdot x)$ 31

Table 3.2 Performance comparison on function $f(x)=5\cdot\sin(8\cdot x)$ 32

Table 3.3 Performance comparison on function $f(x)=10\cdot\sin(8\cdot x)$ 32

Table 3.4 Average time saved on each period with different value of A 33

Table 3.5 An example B-spline curve 'C'..... 36

Table 3.6 Efficiency comparison of finding the arc length of 'C' using traditional way and curvature-based algorithm (Accuracy is set to be 1.0×10^{-6}) 38

Table 3.7 Comparison between the aforementioned algorithms when approximating the arc-length of the example NURBS curve *C(u).* (Accuracy is set to be 1.0×10^{-6})..... 45

Table 3.8 Specifications of the example NURBS curve *S(u)* 46

Table 3.9 The specifications of the example NURBS curve C(u) 49

Table 4.1 Declaring list of the Parameters..... 56

Table 4.2 Declaring list of the notations..... 71

Table 5.1 Control points of the example NURBS tools path 78

Table 5.2 Prescribed tolerances and machine configurations 78

Table 5.3 Appropriate Feed rates over each curve segments after the adjustment 82

Table 5.4 Comparison on Computational time using different interpolation method..... 84

List of Figures

Figure 2.1 An example of 14 chosen points with equal parametric intervals 10

Figure 2.2 An example of 14 chosen points with equal arc length intervals..... 10

Figure 2.3 Illustrative scheme for traditional NURBS tool path generation 11

Figure 2.4 The unit tangent vector at a arbitrary point *A* 15

Figure 2.5 An exaggerated, illustrative tool trajectory of a non-arc-length NURBS scrap with the unit free parameter u 17

Figure 3.1 Approximation of the arc length based on Pythagoras' theorem 20

Figure 3.2 A microscopic view on parameterized curve length approximation 21

Figure 3.3 An illustrative diagram of Midpoint Rule's principle 23

Figure 3.4 An illustrative diagram of Trapezoid Rule's principle 23

Figure 3.5 An illustrative diagram of Simpson's Rule's principle 24

Figure 3.6 Composite Simpson's Rule's principle 26

Figure 3.7 Function $f(x)=5\cdot\sin(8\cdot x)$ 30

Figure 3.8 One sample period of Function $f(x)=5\cdot\sin(8\cdot x)$ after being segmented..... 30

Figure 3.9 Shape of the B-spline curve 'C' 35

Figure 3.10 Curvature profile of 'C' and with the threshold lines '$y=\pm DP$' (where DP was preset to be 1 in this case)..... 36

Figure 3.11 Corresponding selected curve segments in 'S' 37

Figure 3.12 The shape of the example NURBS curve *C(u)*..... 39

Figure 3.13 Image of the Integrand $f(u)=\left|C'(u)\right|$ 40

Figure 3.14 Detailed computational steps during the actual approximation of the traditional method 41

Figure 3.15 Detailed computational steps during the actual approximation when setting the dividing point as $u=1$ 43

Figure 3.16 Detailed computational steps during the actual approximation using the proposed method 44

Figure 3.17 The shape of the example NURBS curve *S(u)* 47

Figure 3.18 Detailed computational steps of the example NURBS curve *S(u)* using the traditional method 47

Figure 3.19 Image of the Integrand $g(u)=\left|S'(u)\right|$ 48

Figure 3.20 The example NURBS curve C(u) and its control points 50

Figure 3.21 An illustration of marking technique over the exaggerated and disproportional parametric interval of the example NURBS curve C(u) 51

Figure 3.22 An illustration of recorded bisect points when using the proposed method and the traditional method 52

Figure 3.23 A zoomed view of the marked sub-interval (199.975, 200.000) 53

Figure 4.1 An exaggerated demonstration of Large chord errors at high curvature parts 55

Figure 4.2 An exaggerated illustration of machining error during one sampling period 56

Figure 4.3 Shape of the example curve $C(t)$ 58

Figure 4.4 Curvature profile of the curve $C(t)$ with the threshold line $k=k_r$ in red 59

Figure 4.5 Geometric relationship between the tool locations at the two sampling moments 62

Figure 4.6 An example acceleration profile of x-axis 64

Figure 4.7 Acceleration profile of x-axis after feed rate adjustment 66

Figure 4.8 The 'Staircase-Shaped' feed rate profile 67

Figure 4.9 Feed rate profile result from traditional method 67

Figure 4.10 Comparison on feed rate profile of traditional method and proposed method during deceleration procedure 68

Figure 4.11 Comparison on feed rate profile of traditional method and proposed method during acceleration procedure 69

Figure 4.12 Feed rate profile result from proposed method 70

Figure 4.13 Illustration of the actual acceleration at a random point *P* on the curve 71

Figure 4.14 An finalized illustrative feed rate profile with bounded axial acceleration 73

Figure 4.15 Flowchart of the arc-length NURBS tool path generation method 74

Figure 4.16 Flowchart of feed rate profile generation module 75

Figure 4.17 A simplified illustrative flowchart for arc-length NURBS tool path generation.....76

Figure 5.1 The shape of the given general NURBS tool path in space 79

Figure 5.2 Recognition of the curve segments that cannot be machined with the prescribed feed rate 80

Figure 5.3 Curvature profile of the theoretical tool path with the threshold line 81

Figure 5.4 Acceleration profile of each axis at the current invalid feed rate..... 81

Figure 5.5 Acceleration profile of each axis after implementing the feed rate re-adjustment method 82

Figure 5.6 Actual feed rate profile during the machining process using the arc-length NURBS interpolation..... 83

Figure 5.7 Actual feed rate profile using the traditional NURBS interpolation 84

Figure 5.8 Trajectory error curve of proposed method..... 86

Figure 5.9 The limit-bounded axial acceleration profile using the proposed arc-length NURBS tool path generation method 86

Chapter 1 Introduction

1.1 Research Problems

For advanced functions and appealing shapes, many parts are designed with free-form curves and surfaces and they are required to be efficiency machined with high accuracy and smoothness. The machining accuracy requirements of these components are becoming higher and higher. To achieve this goal, the cutting tool should cut the workpiece in high feed rate, and its trajectory in machining should comply with the preplanned tool path. The freeform curves and surfaces can be represented with B-splines and non-uniform rational B-splines, namely NURBS. To understand NURBS tool path better, the basics of the NURBS are introduced in this section.

1.1.1. B-spline and NURBS Curve

A general B-spline curve of *p*th-degree can be represented as:

$$S(u)=\sum_{i=0}^{n} N_{i,p}(u)P_i \qquad (a\le u\le b) \tag{1.1}$$

where

$$\begin{aligned} N_{i,0}(u)&=\begin{cases}1 & (u_i\le u\le u_{i+1})\\ 0 & (otherwise)\end{cases}\\ N_{i,p}(u)&=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u)\end{aligned} \tag{1.2}$$

and the control points $\boldsymbol{P_i}$ are given in the form of (x_i, y_i, z_i) coordinates in a three-dimensional space. A knot vector generally consists *n+k+1* non-decreasing elements with the first and last *k* elements equal to the lower and upper boundaries of the parameter range respectively, where *k*=*p*+1. For example, a knot vector of a *k*th-order B-spline curve with its parameter *u* varies from *a* to *b* can be defined as

$$U = \left\{ \underbrace{a,...,a}_{k}, u_k, ..., u_n, \underbrace{b,...,b}_{k} \right\}.$$

The iteration functions of Eq. (1.2) are the blending functions of the B-splines. B-splines are a special type of NURBS, they use the same rules of defining knot vectors and same blending functions. The mainly difference between NURBS and B-splines is that a NURBS has a weight specified for each control point. The equation of NURBS curves is:

$$S(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad (a \le u \le b) \tag{1.3}$$

where w_i is the weight of point $\boldsymbol{P_i}$ and the blending functions $N_{i,p}(u)$ are derived from Eq. (1.2).

B-splines and NURBS have the following main properties:

1. A popular B-spline curve starts and ends at the first and last control points respectively, namely if *u* varies from 0 to 1, then we have $S(0) = P_0$ and $S(1) = P_n$.
2. Each curve segment only affected by *k* control points. A control point affects at most the number of *k* curve segments in its neighborhood.

3. For a NURBS curve, the weight function affects the same neighboring curve segments.

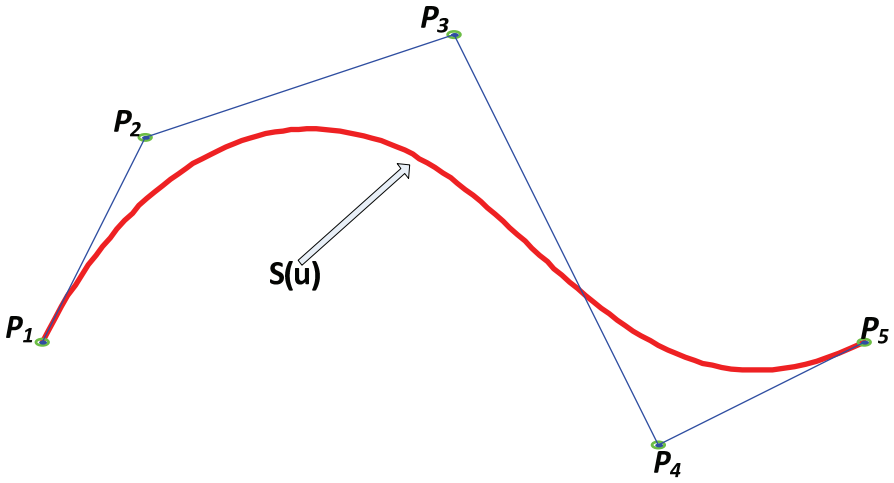


Figure 1.1 A typical NURBS curve with its control polygon

1.1.2. Arc-length Approximation

A curve can be approximated with a polygon that formed by connecting a finite number of points on that curve. The total arc length of the curve can be approximated with the sum of the lengths of all the polygon edges. By increasing the number of line segments on the curve with smaller lengths, a better approximation result could be obtained.

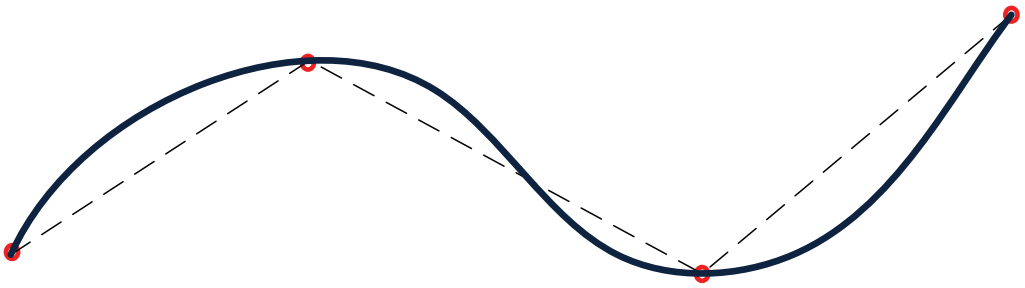


Figure 1.2 Illustration of multiple linear segments arc length approximation

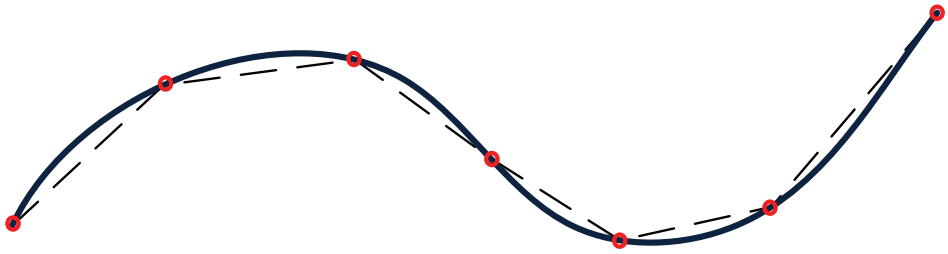
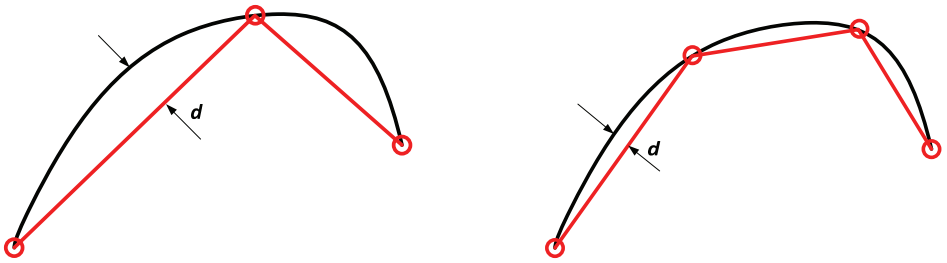


Figure 1.3 Illustration of multiple linear segments arc length approximation with more dividing segments

To improve the approximation accuracy, a maximum displacement between each polygon segments and the curve can be defined as *d* in the Figure 1.4(a) and 1.4(b). The smaller *d*, the higher approximation accuracy is obtained.



(a) (b)

Figure 1.4 Approaching method of arc length approximation

However, using this method to approximate the length of a curve is inaccurate due to the randomness, roughness and inflexibility when segmenting the curve. A more accurate way of calculating arc lengths will be proposed in the Chapter 3.

1.1.3. NURBS Tool Path Interpolation

In CNC machining, the tool path interpolation is to calculate cutter locations on the tool path and the tool moves linearly from cutter locations to cutter locations.

After a theoretical NURBS tool path is fed into a CNC controller, with the desired machining feed rate, a large number of cutter locations are interpolated on the given tool path, and the tool moves along the cutter locations linearly, forming the tool trajectory.

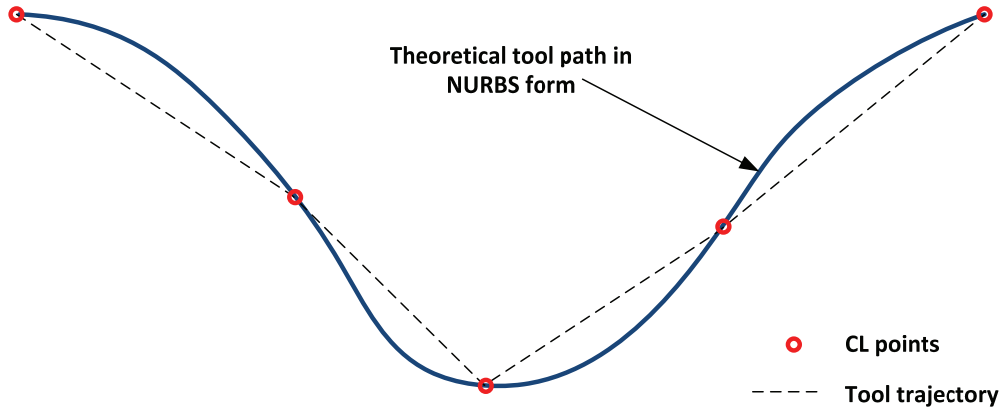


Figure 1.5 Actual tool trajectory after implementing NURBS interpolation

The deviation between the actual tool trajectory and the theoretical tool path is called trajectory error or machining error. This error can be diminished or bounded within the smaller prescribed tolerance by choosing a more proper, in our case, NURBS interpolation method.

1.2 Literature Review

There are a lot of technical articles and relative researches can be found in the field of tool path generation strategies for CNC machining. Feng et al. [5] took axial acceleration into consideration and developed an NURBS tool path interpolation method. However, the tangent acceleration varies along the curve too often and causes feed rate fluctuations that could affect the machining accuracy. Chen and Li

[6] proposed several methods to control the feed rate fluctuation in actual CNC machining. Basically, this method is considered as an add-on algorithm which recursively check for feedrate fluctuation error and maintains it within the prescribed tolerance in real-time. Heng and Erkorkmaz [7] presented a method that fits multiple polynomials and keep the mean squared error of the parameter prediction within a tolerance, to decreases the feed fluctuations. Zhang et al. [8] introduced their NURBS interpolator algorithms with 'look-ahead' control modules. With these strategies, the real-time feedrate is checked and manipulated in a look-ahead buffer, in order to ensure the machining constrains. Emami and Arezoo [9] developed a synchronized look-ahead and real-time parametric interpolator for NURBS curves. The look-ahead module determines the consecutive feedrate blocks in advance to smooth the tool motion. Lee et al. [10] studied an off-line feedrate scheduling method, which divides the original tool path into several NURBS pieces according to its curvature, and also uses a feedrate compensation method to smooth the actual feedrate profile. Meanwhile, a feedrate adjustment method according to the acceleration/deceleration was developed by Wang et al. [11] in 2009. In their work, a direct digital convolution method is proposed for velocity planning of NURBS interpolator. Annoni et al. [12] published another article on NURBS interpolation method, which uses a real-time configurable NURBS interpolator to easily set or change multiple constrains, such as accuracy, acceleration and jerk before the real machining. These constrains are ensured within their limits during real-time calculations of machining process. Ki N.Y. [13] studied a velocity profile generation

methodology and improved the CNC machining efficiency. The feedrate does not need to decrease to zero during the changing procedure of two adjacent tool motions. Also, this method has an added jerk control option when generating the velocity profile. Yeh and Hsu [14] developed an adaptive-feedrate interpolation for parametric curves, which can bound the chord error within the tolerance by decreasing the feedrate. An integrated NURBS path interpolation is presented by Lei et al. [15], in which, the trade-offs between less feedrate fluctuations and less computing time has been balanced through the introduction of their pre-processing module. Similarly, Yong and Narayanaswami [16] introduced another parametric interpolator with confined chord errors, but they also took acceleration and deceleration in to their considerations. However, a FPGA-based motion controller is introduced by Yau et al. [17]. The calculation of basic functions and derivatives of NURBS curve are completed separately by the FPGA chip. Therefore the time consumption during the real-time machining is shortened. X. Zhiming et al. [18] presented a real-time interpolation algorithm for NURBS curves. In which, the contour error and feedrate fluctuation can be constrained and stabilized within the prescribed tolerances. In addition, the feedrate components, acceleration components and the driving force of the servomotor of each axis are pre-calculated before real machining process. Luan et al. [19] developed a pre-scheduled NURBS interpolation method, which calculates the feedrate off-line according to cutting path and machine-tool capability. The acceleration and jerk are effectively reduced and the contour precision is enhanced.

On the other hand, some arc-length parameterized NURBS tool path generation algorithms had also been developed. Wang et al. [20] presented an abstract arc-length parameterized spline path theory which pointed out the relationship between the parameter variable and the curve length is generally non-linear. Khan M.A. [21] presented an arc-length parameterized tool path generation method and a gouging-free NURBS theoretical tool path interpolation algorithm. The feedrate can now also adaptive to the local shape of the curve.

1.3 Objective of the Research

The objective of this thesis can be concluded by the two following central aspects:

1) Propose a new approach to accurately calculating the arc-length of the NURBS tool path.
2) Generate arc-length NURBS tool paths to maintain the high accuracy during machining process with less feed rate lose and computational time.
3) After the re-parameterization, a feasible feed rate profile need to be found according to machining error tolerance and the acceleration limit of the machine. Finally all the cutter locations can moves with a more stable and smoother feed rate in real time.

Based on the three main components of my work from above, a new 3-axis NURBS tool path generation method is proposed.

1.4 Thesis Outline

This thesis comprises of seven chapters. Chapter one gives a general introduction of some relative concepts of this work, followed by reviewed literatures on this topic, and points out the research objectives. Chapter two reveals the properties and differences between the arc-length NURBS tool path generation and the traditional method. Chapter three introduces several arc length calculation methods and proposes a new curvature-based NURBS arc-length calculation method. Chapter four presents a new arc-length NURBS tool path generation method that is global axial acceleration and accuracy bounded. The new strategy for feed rate profile generation is also shown in this chapter. Chapter five conducts a test of generating an actual tool path. Comparison between the traditional method and the proposed method based on the gathered date from the example will be shown and analyzed. Chapter six concludes the central work of this thesis and proposes some possible future works. In Chapter seven, references are listed.

Chapter 2 Non-arc-length NURBS Tool Path and Arc-length NURBS Tool Path Generation

2.1 Introduction

For a general B-spline or NURBS curve, its parameter *u* is unit-free. This indicates non-arc-length equally spaced parametric values of *u* do not represent equally spaced points on the curve. These kinds of curves are called non-arc-length B-splines and non-arc-length NURBS.

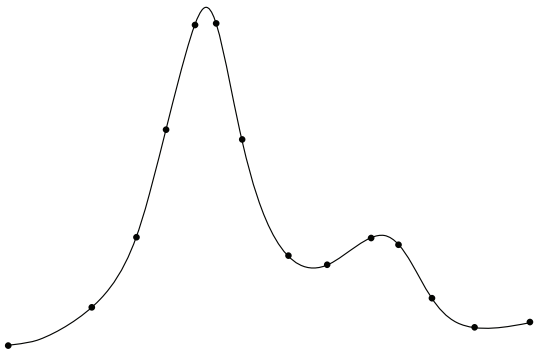


Figure 2.1 An example of 14 chosen points with equal parametric intervals

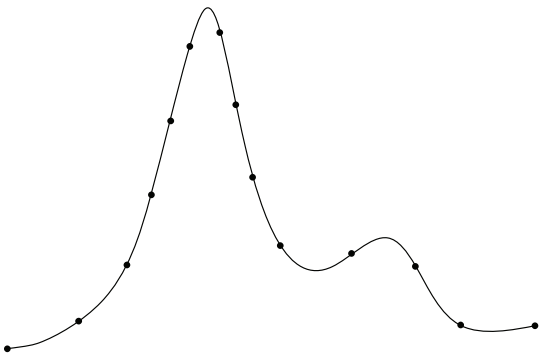


Figure 2.2 An example of 14 chosen points with equal arc length intervals

2.2 Algorithm of Non-arc-length NURBS Tool Path

In 3-axis CNC machining, the actual cutter locations are illustrated in a procedure shown in Figure (2.3).

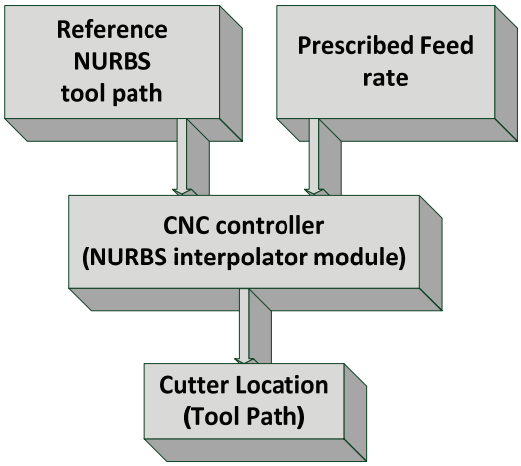


Figure 2.3 Illustrative scheme for traditional NURBS tool path generation

A reference tool path (also called theoretical tool path) in the form of NURBS expression and a prescribed feed rate are sent to the NURBS interpolator of the CNC controller. The CNC interpolator then generates the accordingly feed rate profile and instantaneously calculates the cutter locations in real time.

Considering the feed rate profile generated in Figure (2.3) is $V(t)$, to a non-arc-length NURBS reference tool path

$$S(u)=\begin{bmatrix}x(u) & y(u) & z(u)\end{bmatrix}^T=\frac{\sum_{i=0}^{n}N_{i,p}(u)w_iP_i}{\sum_{i=0}^{n}N_{i,p}(u)w_i} \qquad (a\le u\le b) \tag{2.1}$$

with the given knot vectors

$$U=\left\{\underbrace{a,...,a}_{k},u_k,.....,u_n,\underbrace{b,...,b}_{k}\right\}$$

and the control points P_i, the feed rate function $V(t)$ can be expressed as

$$V(t) = \frac{dS(u)}{dt}. \tag{2.2}$$

We can re-write the above equation into

$$V(t) = \left\| \frac{dS(u)}{du} \right\| \cdot \frac{du}{dt}, \tag{2.3}$$

hence, we can deduce

$$\frac{du}{dt} = \frac{V(t)}{\left\| \frac{dS(u)}{du} \right\|} \tag{2.4}$$

By using the first order approximation of Taylor expansion formula, the Eq. (2.4) can be processed as follows.

$$u_{m+1} = u_m + \frac{V(t_m) \cdot T_s}{\left\| \frac{dS(u)}{du} \right\|_{u=u_m}} \tag{2.5}$$

where u_m and T_s denote the instant value of parameter u at the time $t = t_m$ and the machine sampling period, respectively. The parametric value u_{m+1} of the next sampling period can be found. Normally, Eq. (2.5) is adequate to be applied on curve segments with small curvatures. However, for relatively high curvature segments, the second order approximation is more properly for use.

$$u_{m+1} = u_m + \frac{V(t_m) \cdot T_s}{\left\| \frac{dS(u)}{du} \right\|_{u=u_m}} - \frac{V^2(t_m) \cdot T_s^2}{2 \cdot \left\| \frac{d^2 S(u)}{du^2} \right\|_{u=u_m}^4} \tag{2.6}$$

In Eq. (2.5) and (2.6), the first derivative of $S(u)$ can be computed as

$$\frac{dS(u)}{du}=\frac{\sum_{i=0}^{n}N'_{i,p}(u)w_i}{\sum_{i=0}^{n}N_{i,p}(u)w_i}P_i-\frac{\sum_{i=0}^{n}N'_{i,p}(u)w_i\cdot\sum_{i=0}^{n}N_{i,p}(u)w_i}{\left(\sum_{i=0}^{n}N_{i,p}(u)w_i\right)^2}P_i \tag{2.7}$$

as well as the second derivative can be computed as

$$\begin{aligned}\frac{d^2S(u)}{du^2}=&\frac{\sum_{i=0}^{n}N_{i,p}(u)w_i\cdot\sum_{i=0}^{n}N''_{i,p}(u)w_iP_i}{\left(\sum_{i=0}^{n}N_{i,p}(u)w_i\right)^2}-\frac{\sum_{i=0}^{n}N''_{i,p}(u)w_i\cdot\sum_{i=0}^{n}N_{i,p}(u)w_iP_i}{\left(\sum_{i=0}^{n}N_{i,p}(u)w_i\right)^2}-\\&\frac{2\sum_{i=0}^{n}N'_{i,p}(u)w_i\cdot\sum_{i=0}^{n}N_{i,p}(u)w_i\sum_{i=0}^{n}N'_{i,p}(u)w_iP_i}{\left(\sum_{i=0}^{n}N_{i,p}(u)w_i\right)^3}-\\&\frac{2\sum_{i=0}^{n}N'_{i,p}(u)w_i\cdot\sum_{i=0}^{n}N_{i,p}(u)w_iP_i\cdot\sum_{i=0}^{n}N'_{i,p}(u)w_i}{\left(\sum_{i=0}^{n}N_{i,p}(u)w_i\right)^3}\end{aligned} \tag{2.8}$$

where the *k*th order derivative of $N_{i,p}(u)$ is

$$N_{i,p}^{(k)}(u)=p\cdot\left[\frac{N_{i,p-1}^{k-1}(u)}{u_{i+p-1}-u_i}-\frac{N_{i+1,p-1}^{k-1}(u)}{u_{i+p}-u_{i+1}}\right] \tag{2.9}$$

Thus, all the *CL* points can be calculated by substituting a series of *u* values which are found iteratively using the Taylor expansion approximation of (2.5) or (2.6), into Eq. (2.1). As the real-time coordinates of all CL points

$$CL(u_i)\big|_{i=1}^{m}=\left\{\begin{bmatrix}x(u_1)\\y(u_1)\\z(u_1)\end{bmatrix},\begin{bmatrix}x(u_2)\\y(u_2)\\z(u_2)\end{bmatrix},\cdots,\begin{bmatrix}x(u_m)\\y(u_m)\\z(u_m)\end{bmatrix}\right\} \tag{2.10}$$

are calculated (where *m* denotes the number of the CL points) in the CNC controller iteratively, the tool moves from the current position directly to the next position in the coordinate sequence of Eq.(2.10). The actual tool path is finally formed.

2.3 Algorithm of Arc-length NURBS Tool Path

2.3.1 Properties of Arc-length NURBS Curve

To understand arc-length NURBS tool path well, the main properties of the paths are introduced here. The arc-length NURBS uses the actual arc length *s* as its parameter instead of *u*. For example, considering a point $A=\begin{bmatrix}x(u_A) & y(u_A) & z(u_A)\end{bmatrix}^T$ is on the curve $C(u)$. This point can also be found as

$$A=\begin{bmatrix}x(s_A) & y(s_A) & z(s_A)\end{bmatrix}^T$$

where the parameter s_A denotes the arc-length measuring from the starting point of the curve to the point *A*. An arc-length NURBS tool path can be represented as:

$$C(s)=\begin{bmatrix}x(s) & y(s) & z(s)\end{bmatrix}^T=\frac{\sum_{i=0}^{n}H_{i,p}(s)h_iQ_i}{\sum_{i=0}^{n}H_{i,p}(s)h_i} \quad (0\le s\le L), \tag{2.11}$$

in which the parameter *s* is the genuine arc length between the initial point of the curve and the current point, and *L* is the total length of the curve. $H_{i,p}$, h_i, Q_i are the corresponding base function, weights and control points, respectively. Ideally, the main properties of an arc-length NURBS can be easily recognized as

$$\left\|C'(s)\right\|=\frac{\left\|d(C(s))\right\|}{ds}=1 \tag{2.12}$$

and

$$\left\|C''(u)\right\|=0 \tag{2.13}$$

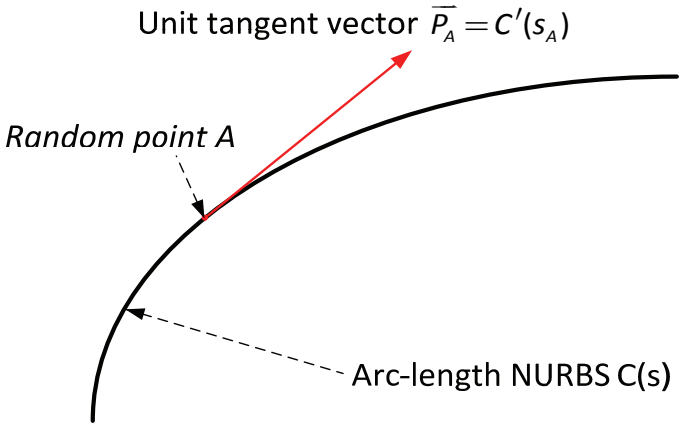


Figure 2.4 The unit tangent vector at a arbitrary point *A*

This indicates the value of its first order derivative with respect to *s* at any point on the curve is the coordinate of the unit tangent vector $\vec{P}$ of that point (Figure 2.4).

2.3.2 Arc-Length NURBS Interpolation

By inputting the arc length NURBS into the CNC controller, the feed rate profile $V(t)$ is generated. The iterative interpolation function of parameter *u* in Eq. (2.5) and (2.6) are replaced by

$$s_{m+1} = s_m + V(t_m) \cdot T_s \tag{2.14}$$

where s_m denotes the value of parameter *s* at the moment of $t = t_m$, and T_s is the machine sampling period. Then, a cluster of *CL* points can be found. According to Eq. (2.14), the following equation holds.

$$\lim_{V(t)\cdot T_s \to 0} \left| \overline{CL(s_m)CL(s_{m+1})} \right| - \left| CL(s_m)CL(s_{m+1}) \right| = 0 \tag{2.15}$$

Since the sampling period T_s is constant, the lower the prescribed feed rate is set, the closer the tool trajectory can get to the reference tool path. In another word, for any two neighboring *CL* points, the chord and the arc length in between are

approximately equal to each other if the given feed rate is not high.

Therefore, by finding a valid feed rate profile that can bound the machining error within the tolerance, the final tool path is formed by connecting all the CL points.

2.4 Comparison of Non-Arc-Length and Arc-Length NURBS Tool Path Generation Methods

As aforementioned, the basic procedures of generating the machining tool path from using either conventional NURBS or arc-length NURBS are similar. Only that the NURBS interpolation algorithms are different. Then the question why bother to use arc-length NURBS may arise. In fact, there are two main differences between these two algorithms that would explain the advantages of using arc-length NURBS tool paths.

2.4.1 Comparison on Tool Path Error and Feed Rate Error

First, when conducting conventional NURBS interpolation, due to the highly non-linear relationship between the increments of parameter u and the corresponding displacements along curve

$$\frac{\Delta u_0}{\Delta s_0} \neq \frac{\Delta u_1}{\Delta s_1} \neq \frac{\Delta u_2}{\Delta s_2} \neq \cdots \frac{\Delta u_n}{\Delta s_n}, \tag{2.16}$$

the chord length $\left|CL(u_i)CL(u_{i+1})\right|$ oscillates globally. This will lead to a feed rate

fluctuation over the whole machining process, resulting unpredictable acceleration and deceleration in tool motion. In addition, the tool trajectory error may fatally violate the prescribed tolerance at some times even with a normal or low feed rate (Figure 2.5).

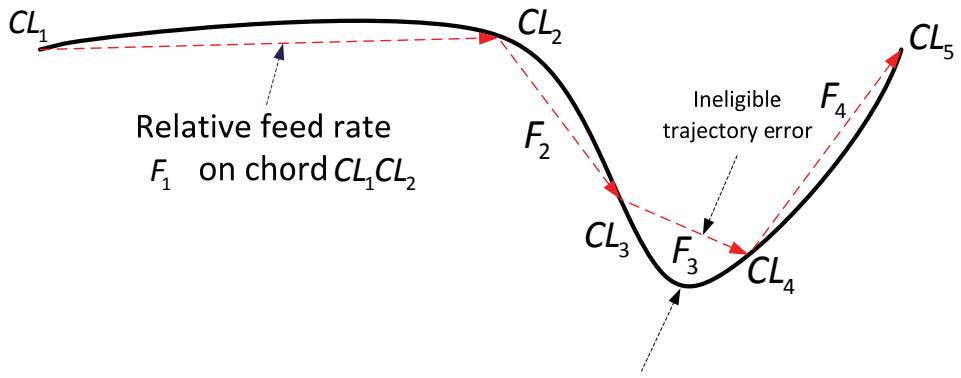


Figure 2.5 An exaggerated, illustrative tool trajectory of a non-arc-length NURBS scrap with the unit free parameter u

The local feed rate for each sampling period can be calculated by

$$F_i = \left|CL_iCL_{i+1}\right| / T_s \tag{2.17}$$

According to the example shown in Figure (2.5), the instantaneous feed rates F_1, F_2, F_3 and F_4 truly varies rapidly causing fluctuation. However, with the arc-length parameterized NURBS, this problem is eliminated in favor of all the neighboring chords are with equally lengths.

Second, the governing functions of Eq. (2.5) and Eq. (2.6) of non-arc-length NURBS are actually approximations from Taylor expansion. The neglect of the residue from Taylor expansion would more or less cause inaccuracy during the approximation. On contrary, the iteration function of Eq. (2.14) from the arc-length NURBS interpolation algorithm does not have such problem.

2.4.2 Comparison on Computational Efficiency

By comparing the interpolation functions of non-arc-length NURBS and arc-length NURBS as follows

Table 2.1 Comparison on governing functions of non-arc-length and arc-length NURBS

	Arc-length NURBS interpolation	Non-arc-length NURBS interpolation	
		First order Taylor approximation	Second order Taylor approximation
Governing Function	$s_{m+1} = s_m + V(t_m)\cdot T_s$	$u_{m+1} = u_m + \frac{V(t_m)\cdot T_s}{\left\|\frac{dS(u)}{du}\right\|_{u=u_m}}$	$u_{m+1} = u_m + \frac{V(t_m)\cdot T_s}{\left\|\frac{dS(u)}{du}\right\|_{u=u_m}} - \frac{V^2(t_m)\cdot T_s^2}{2\cdot\left\|\frac{d^2S(u)}{du^2}\right\|^4_{u=u_m}}$
Velocity Profile	V(t)	V(t)	V(t)
Need of First order derivative evaluation	No	Yes	Yes
Need of Second order derivative evaluation	No	No	Yes

it is reasonable to believe the CNC controller could suffer a much heavier computational burden when embedded with the conventional NURBS interpolation algorithm. Consequently, when both the NURBS interpolators calculate in real time, the one with the arc-length NURBS interpolation algorithm turns out can run potentially faster. Hence, a higher computational efficiency is gained.

Chapter 3 Arc length Calculation Using Quadrature Methods

3.1 Introduction

Apart from the general approach which had been brought up in Section 1.1.2, there are still many ways to approximate curve lengths. In this chapter, several quadrature methods are introduced for evaluating the arc lengths of free-form curves in a numerical way. For mathematical equations are applied when evaluating the actual arc lengths, by using these methods, the approximation accuracy can be easier and better controlled. In fact, quadrature methods are a typical set of ways to find numerical evaluation of definite integrals. These methods are basically referring to the elementary technique for approximating the enclosed area that underneath the objective curve within a certain interval. There is not only one but a series of methods of quadrature. Every which of them provide us with certain accuracy according to our demands.

3.2 Derivation of Mathematical Equations for Arc Length Approximation

To derive a more accurate and formulated method for calculating the length of general curves, By letting the displacement *d* in Figure (1.4) tends to be zero, the

length of each polygon segment can be measured as equal to the length of the corresponding arc of the actual curve.

Consider a real function *f(x)* such that *f(x)* and $f'(x)=\frac{dy}{dx}$, its derivative with respect to *x*, are continuous on geometric interval $[a,b]$. The length *s* of the curve *f(x)* within the interval $[a,b]$ can be found as follows:

According to Pythagoras' theorem, an infinitesimal part of the polygon segment *ds* is shown in Figure (3.1).

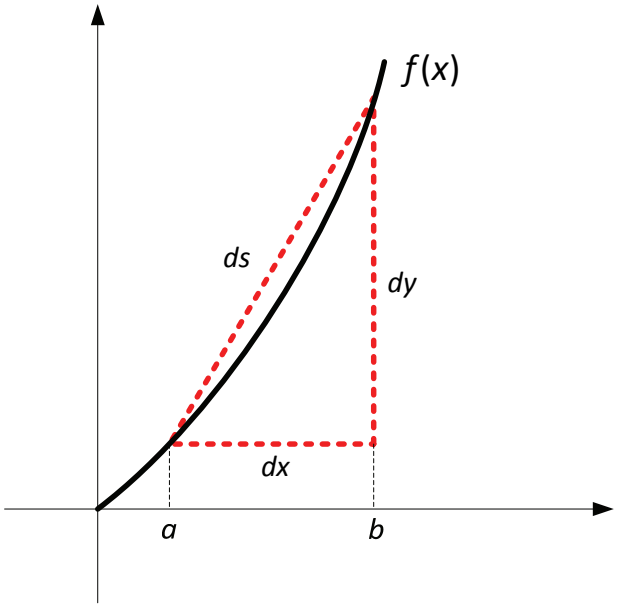


Figure 3.1 Approximation of the arc length based on Pythagoras' theorem

Since the equation

$$ds^2 = dx^2 + dy^2 \tag{3.1}$$

holds, so that by processing the above equation throughout from Eq. (3.2) to Eq. (3.5),

$$\frac{ds^2}{dx^2}=1+\frac{dy^2}{dx^2} \tag{3.2}$$

$$\frac{ds^2}{dx^2} = 1 + \left(\frac{dy}{dx}\right)^2 \tag{3.3}$$

$$\frac{ds}{dx} = \sqrt{1 + \left(\frac{dy}{dx}\right)^2} \tag{3.4}$$

$$ds = \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx \tag{3.5}$$

the final expression of arc length approximation function is derived as Eq. (3.6).

$$s = \int_a^b \sqrt{1 + \left[f'(x)\right]^2}\, dx \tag{3.6}$$

Instead of segmenting the curve and summing them up, the arc length in certain intervals can be calculated if the representation equation of the curve and its first order derivative were acquired. However, for parameterized curves, the arc length approximation function is slightly different.

Let $C(t)$ to be a parameterized curve with respect to t. An infinitesimal curve segment from point $C(T)$ to $C(T + \Delta t)$ is shown as Figure (3.2). Similarly,

$$\Delta s = \sqrt{\Delta x^2 + \Delta y^2}\ . \tag{3.7}$$

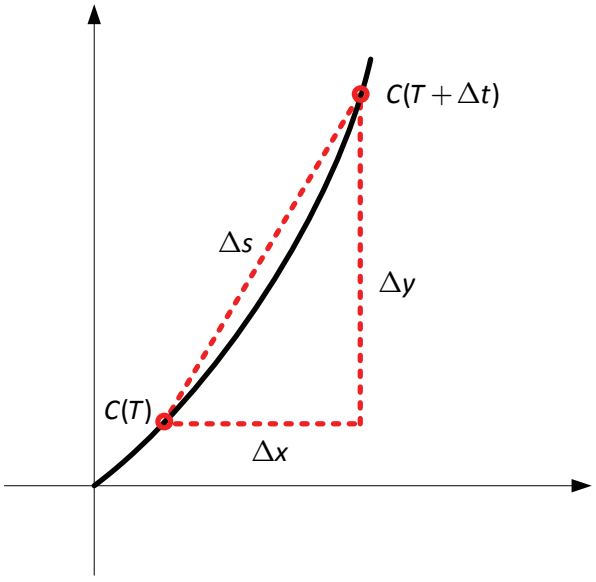


Figure 3.2 A microscopic view on parameterized curve length approximation

Assuming the whole objective parametric interval $[a,b]$ is divided up into *n* pieces, the approximation of the curve length *S* from point *C(a) to C(b)* can be written as:

$$S \approx \sum_{i=1}^{n} \sqrt{\Delta x_i^{\,2} + \Delta y_i^{\,2}} \tag{3.8}$$

By letting $n \to \infty$ and multiplying $\frac{\Delta t}{\Delta t}$ to the right-side of Eq. (3.8), then

$$S = \lim_{n\to\infty} \sum_{i=1}^{n} \sqrt{\left(\frac{\Delta x_i}{\Delta t}\right)^2 + \left(\frac{\Delta y_i}{\Delta t}\right)^2} \cdot \Delta t \tag{3.9}$$

Eq. (3.9) can eventually be written in the form of

$$S = \int_a^b \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \cdot dt \tag{3.10}$$

or

$$S = \int_a^b \left|C'(t)\right| \cdot dt \tag{3.11}$$

where $\left|C'(t)\right|$ indicates the norm of the vector $C'(t)$.

3.3 Common Quadrature Methods

3.3.1 Midpoint Rule and Trapezoid Rule

On one hand, as one of the most basic quadrature rules, the Midpoint Rule approximate the integral $\int_a^b f(x)dx$ by the area of a rectangle with its base length equals to the span of the target interval, namely $|b-a|$, and its height equals to integrand value at the midpoint of the above interval. The result can be written as Eq. (3.12), where $h = b - a$.

$$\int_a^b f(x)dx = hf(\frac{a+b}{2}) \tag{3.12}$$

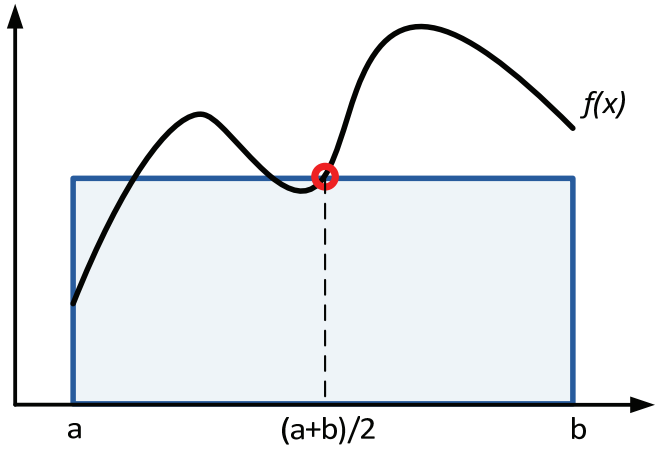


Figure 3.3 An illustrative diagram of Midpoint Rule's principle

On the other hand, The Trapezoid Rule approximates the same integral by the area of a trapezoid with its base length set as same as the Midpoint Rule, and sides equal to the integrand values at the two endpoints which are $f(a)$ and $f(b)$ respectively. The approximation result can be written as

$$\int_a^b f(x)dx = h\frac{f(a)+f(b)}{2} \tag{3.13}$$

where still $h = b - a$.

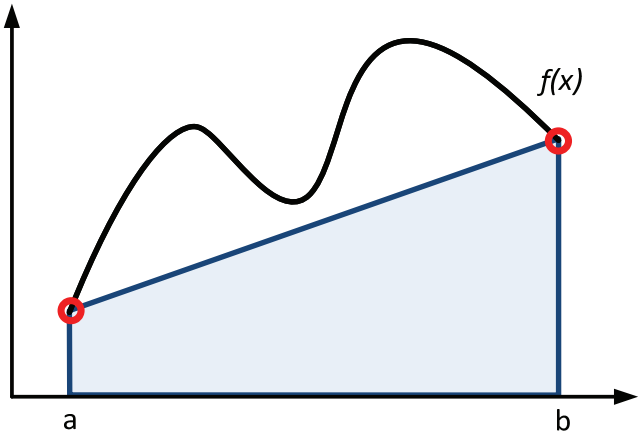


Figure 3.4 An illustrative diagram of Trapezoid Rule's principle

Apparently, both of the methods above are capable for evaluating linear

functions, but neither of them could achieve an expected accuracy when implemented on a quadratic function.

3.3.2 Simpson's Method

The Simpson's rule is also a method for numerical integration approximation for definite integrals. This method approximates the integrand $f(x)$ with a quadratic function, namely the $P(x)$ profile in Figure (3.5), which passes through both the end-points and the point $f(\frac{a+b}{2})$.

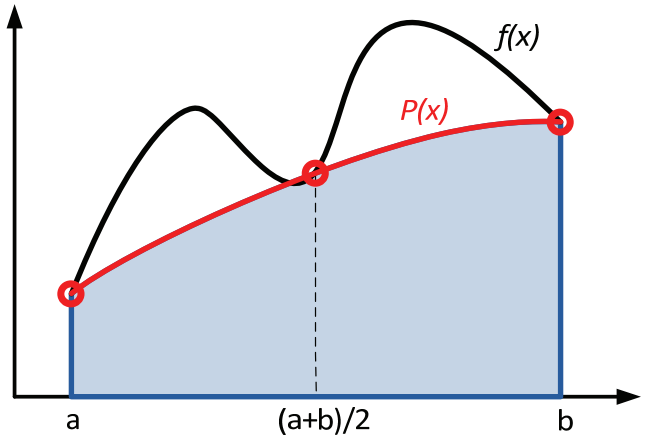


Figure 3.5 An illustrative diagram of Simpson's Rule's principle

The approximation result can be written as Eq. (3.14), where S is the approximation of the integral $\int_a^b f(x)dx$.

$$S = \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right] \tag{3.14}$$

3.3.2.1 Derivation of Simpson's Rule

As a matter of fact, Simpson's rule is a combination of the previous two basic quadrature rules, namely Midpoint rule and trapezoid rule. By implementing the

approximation equations of Eq. (3.12) and Eq. (3.13) of these two methods to tons of test examples, it turns out the approximation error of Midpoint rule is always roughly -2 times of that from Trapezoid rule. Therefore, by representing the approximation results from Midpoint and Trapezoid rule with M and T respectively, it can be found the exact value S of the integral would have the relationship of

$$S - T = -2(S - M) \tag{3.15}$$

if the aforementioned errors had exactly -2 times difference from each other. Thus, by solving S in Eq. (3.15), a more accurate approximation is obtained:

$$S = \frac{2}{3}M + \frac{1}{3}T. \tag{3.16}$$

By substituting M and T with the right-hand side of the Eq. (3.12) and Eq. (3.13) respectively, the Simpson's method is derived.

$$S = \frac{h}{6}[f(a) + 4f(c) + f(b)] \tag{3.17}$$

where c denotes the midpoint of the interval $[a,b]$.

If the whole objective interval $[a,b]$ were equally divided into two parts, d and e are found as the midpoints of the two new subintervals $[a,c]$ and $[c,b]$ respectively. By individually performing the procedure of Eq. (3.17) over the two subintervals, a closer approximation result S' of composite Simpson's method can be found as

$$S' = \frac{h}{12}[f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)]. \tag{3.18}$$

In most cases, the result from the composite Simpson's rule of Eq. (3.18) tends to be more accurate than the result from Eq. (3.17).

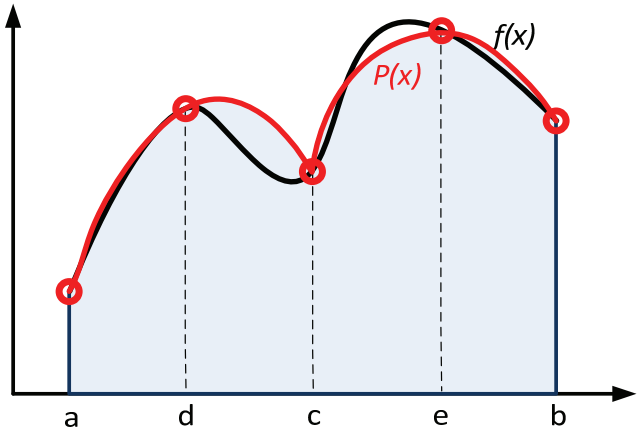


Figure 3.6 Composite Simpson's Rule's principle

Since at least both the end points and the midpoint of the objective interval are required for the implementation of the Simpson's rule. The most basic Simpson's formula of Eq. (3.17) is called the Three Point Simpson's Rule.

Therefore, in order to implement more steps or, in another word, a composite Simpson's rule, we may need a number of *2W+1* points to carry out *W* steps of Simpson's rule, e.g. the above Eq. (3.18) is a five points double Simpson's rule.

Now by assuming that the known interval [*a*, *b*] has been divided into a number of *2W* sub-intervals, and *h* = *b* - *a*. The general composite Simpson's rule can be written as:

$$\int_a^b f(x)dx = \frac{h}{3\cdot 2W}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+\cdots \\ \cdots+2f(x_{2W-2})+4f(x_{2W-1})+f(x_{2W})] \tag{3.19}$$

or

$$\int_a^b f(x)dx = \frac{h}{3\cdot 2W}\sum_{k=1}^{W}\left(f(x_{2k-2})+4f(x_{2k-1})+f(x_{2k})\right) \tag{3.20}$$

where $x_{2k-1}=\frac{x_{2k-2}+x_{2k}}{2}$, $x_0=a$, $x_{2W}=b$.

Thus, by keep sub-dividing the objective interval, also as know as to increase the recursive counts *W*, higher approximation accuracy could be obtained.

3.3.2.2 Error Estimation for Simpson's Rule

Since the Eq. (3.17) and Eq. (3.18) are approximating the same integral over $[a,b]$, the error can be denoted as:

$$E = (A_2 - A_1) \tag{3.21}$$

where A_1 and A_2 stand for the approximation result from Eq. (3.17) and Eq. (3.18), respectively. As comparing Figure (3.6) from Figure (3.5), the step size of Eq. (3.18) is recognized as half as that of Eq. (3.17). Therefore, the approximation result A_2 can be considered as roughly 2^4 accurate as A_1. Therefore by solving the ideal accurate value *R* in the following equation

$$16(R - A_2) = R - A_1 \tag{3.22}$$

a more accurate approximation based on the existing integral evaluation results is derived.

$$R = A_2 + (A_2 - A_1)/15 \tag{3.23}$$

Due to both *R* and A_2 were approximating over the same interval [*a*, *b*], the relatively approximation error of *R,* thus, can be found as

$$E_R = (R - A_2) \tag{3.24}$$

Therefore, the approximation accuracy can be easily controlled by keeping

$$\left(E_R\right)_{\max} \le \varepsilon_t \tag{3.25}$$

where ε_t is the prescribed accuracy tolerance.

3.4 Arc length calculation for B-splines and NURBS

Assuming we have a B-spline curve

$$C(u)=\sum_{i=0}^{n}N_{i,p}(u)P_i \qquad (a\le u\le b) \tag{3.26}$$

with its knot vector defined as

$$U=\left\{\underbrace{a,\cdots,a}_{p},u_{p+1},\cdots,u_{m-p-1},\underbrace{b,\cdots,b}_{p}\right\}$$

where p is the degree of the curve, $P_0\cdots P_n$ are the control points. The blending function $N_{i,p}(u)$ is defined as

$$\begin{aligned}N_{i,0}(u)&=\begin{cases}1 & (u_i\le u\le u_{i+1})\\ 0 & (otherwise)\end{cases}\\ N_{i,p}(u)&=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u)\end{aligned} \tag{3.27}$$

Now that, according to the Eq. (3.11), the curve length S measured from $u=c$ to $u=d$ ($c\ge a,d\le b,c\le d$) can be found as

$$S=\int_c^d\left|C'(u)\right|\cdot du \tag{3.28}$$

where the derivative of the B-spline is calculated from

$$C'(u)=\sum_{i=0}^{n-1}N_{i,p-1}(u)Q_i \tag{3.29}$$

$$Q_i=p\frac{P_{i+1}-P_i}{u_{i+p+1}-u_{i+1}} \tag{3.30}$$

with its knot vector

$$U'=\left\{\underbrace{a,\cdots,a}_{p-1},u_{p+1},\cdots,u_{m-p-1},\underbrace{b,\cdots,b}_{p-1}\right\} \tag{3.31}$$

The blending functions $N_{i,p-1}(u)$ are computed on U'. By implementing *W*-steps Composite Simpson's rule from Eq. (3.19) or Eq. (3.20) on the definite integral of Eq. (3.28) has been found as

$$S = \frac{d-c}{3 \cdot 2W} \sum_{k=1}^{M} \left(\left| C'(u_{2k-2}) \right| + 4 \left| C'(u_{2k-1}) \right| + \left| C'(u_{2k}) \right| \right) \tag{3.32}$$

3.5 Curvature-adaptive based Arc Length Calculation for Parametric Curves Using Composite Simpson's Method

3.5.1 Rough Tempt on Optimizing Calculation Efficiency

To calculate the curve length of $C(t)$ in section (3.2) within the interval $[a,b]$, the result is given by Eq. (3.11). However, according to the properties of integrals, Eq. (3.11) can also be written as

$$S = \int_a^c \left| C'(t) \right| \cdot dt + \int_c^b \left| C'(t) \right| \cdot dt \tag{3.33}$$

where $a < c < b$.

When trying to reach the same accuracy, the total time consumption during the calculation could be shortened by dividing the objective interval into several segments according to the curvature and continuity of the curve instead of directly setting the integrated interval to define the upper and lower boundaries of the integral. The following example using the function $f(x) = 5 \cdot \sin(8 \cdot x)$ illustrates the idea.

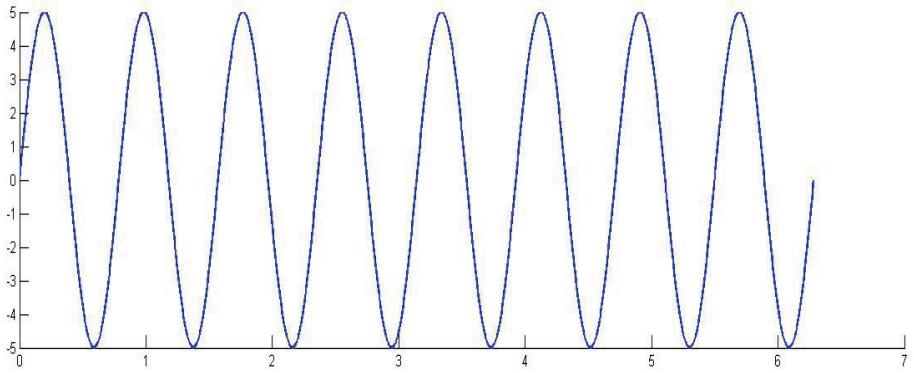


Figure 3.7 Function $f(x) = 5 \cdot \sin(8 \cdot x)$

As it appears, the shape of the curve is more twisted at the peaks and troughs, which means the curvature is greater at these parts than the rest of the curve.

Then, a rough segmenting procedure is implemented here which divides each period of the wave into 6 equal pieces as Figure (3.8) shows.

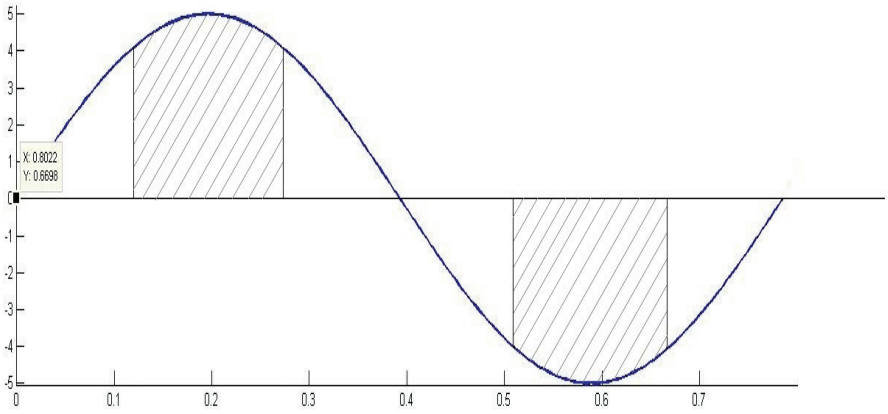


Figure 3.8 One sample period of Function $f(x) = 5 \cdot \sin(8 \cdot x)$ **after being segmented**

The curve pieces within the shaded interval can be considered to be relatively more curved than the other parts of the curve. Now by calculating the length of all these shaded segments and the rest un-shaded parts separately, the sum of all these sub-results from above will be the length of the curve.

When attempting to reach the same accuracy using the composite Simpson's method to solve this, this method consumes less time than doing direct calculation over the whole objective interval.

This is because the calculations within the shaded interval need more recursive steps to reach the expected accuracy due to its relatively high curvature, while the rest pieces of the curve need not. Consequently, redundant computations may occur while calculating those other relatively smoothed pieces of the curve. The following test function $f(x)=A\cdot\sin(8\cdot x)$ demonstrates this (where A and N denotes the amplitude coefficient of the function and the number of the periods that is being calculated respectively in the upcoming contents).

While A=1, accuracy=1.0×10^{-6}:

Table 3.1 Performance comparison on function $f(x)=\sin(8\cdot x)$

	Curve length	Time consumed using segmentation method (s)	Time consumed from direct approximation	Difference of average time consumed (s)	Average time saved on each period (s)
N=1	4.1238	0.578266	0.656773	0.078507	0.078507
N=5	20.6188	1.765949	2.102958	0.337009	0.067402
N=20	82.4753	6.163200	7.325077	1.161877	0.058094
N=40	164.9505	12.014090	14.310630	2.296540	0.054135

While A=5, accuracy=1.0×10^{-6}:

Table 3.2 Performance comparison on function $f(x)=5\cdot\sin(8\cdot x)$

	Curve length	Time consumed using segmentation method (s)	Time consumed from direct approximation	Difference of average time consumed (s)	Average time saved on each period (s)
N=1	20.0348	0.750659	0.801483	0.050824	0.050824
N=5	100.1742	2.639361	3.022017	0.382656	0.076531
N=20	400.6969	9.750296	11.022530	1.272234	0.063612
N=40	801.3937	19.131970	21.772795	2.640825	0.066021

While A=10, accuracy=1.0×10^{-6}:

Table 3.3 Performance comparison on function $f(x)=10\cdot\sin(8\cdot x)$

	Curve length	Time consumed using segmentation method (s)	Time consumed from direct approximation	Difference of average time consumed (s)	Average time saved on each period (s)
N=1	40.0196	0.803874	0.860551	0.056677	0.056677
N=5	200.0979	2.918330	3.140983	0.222653	0.045306

N=20	800.3918	10.920671	11.734121	0.813450	0.040672
N=40	1.6008×10^3	21.507276	23.198515	1.691239	0.042281

Even though the Simpson's method and its initialization steps are repeatedly executed within this method, the total time consumed is still shorter comparing with the direct approximation method. That clearly shows the calculation time is saved during the effective calculation, in another word, the overall number of recursive operations is decreased.

The result has proved the theory of this method is credible, but since the way of segmenting the objective interval is random and rough, it still could save more time by choosing the dividing points to be more dependent on the property of the curve.

Table 3.4 Average time saved on each period with different value of A

	Time saved on each period (s)	
A=1	0.078507	
	0.067402	
	0.058094	
	0.054135	
	Average (s)	0.064535
A=5	0.050824	
	0.076531	
	0.063612	
	0.066021	
	Average (s)	0.064247
A=10	0.056677	
	0.045306	
	0.040672	
	0.042281	
	Average (s)	0.046234

According to the statistics above, the average time saved on each period is decreasing as the amplitude of the wave *A* increases. This is due to the curvature within each sub-intervals is changing more severely when near the peaks. In this case, we should adjust the dividing points to re-narrow down those intervals that corresponding to the segments with relatively high curvature.

3.5.2 Curvature-recognition based B-spline arc length approximation

Based on the previous rough idea, which indicates that those curve segments with relatively high curvature would somehow have a hold back effect on the arc-length calculation efficiency over the entire objective interval, a way to recognize those segments is introduced.

A threshold value, denoted as '*DP*' in the following contents, is set to divide the whole curve into several segments according to the curvature of the curve. That means, if the curvature of some parts of the curve is greater than the pre-set value '*DP*', then these segments will be extracted from the rest of the curve and be calculated separately.

A general B-spline curve $C(u)$ from Eq. (3.26) and its first order derivative $C'(u)$ is computed as shown before. Since $C'(u)$ is also a B-spline curve, the second order derivative $C''(u)$ can also be found.

If we write $x'(u)$ and $y'(u)$ as the value of *x* component and *y* component respectively of the first order derivative value $C'(u)$ at *u,*

$$C'(u)=\begin{bmatrix} x'(u) \\ y'(u) \end{bmatrix} \tag{3.34}$$

and write $x''(u)$ and $y''(u)$ as the value of x component and y component respectively of the second order derivative value $C''(u)$ at u,

$$C''(u) = \begin{bmatrix} x''(u) \\ y''(u) \end{bmatrix} \tag{3.35}$$

The curvature κ at point u can be expressed as:

$$\kappa = \frac{x'(u)y''(u) - y'(u)x''(u)}{(x'(u)^2 + y'(u)^2)^{3/2}} \tag{3.36}$$

Assuming a general B-spline curve 'C' shown in (Figure 3.9), with its relative parameters defined in Table 3.5,

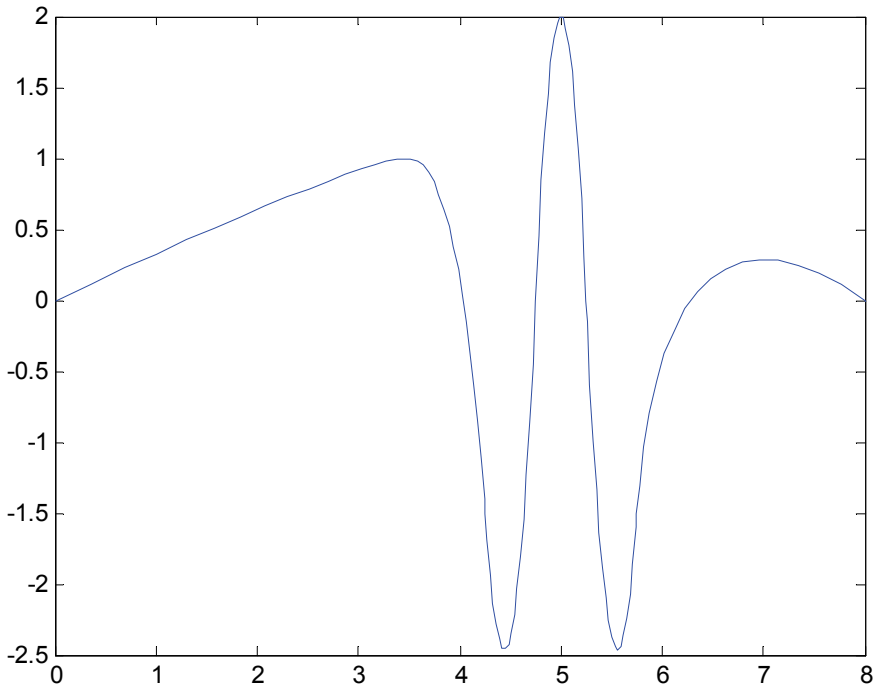


Figure 3.9 Shape of the B-spline curve 'C'

by drawing out the curvature of 'C' using Eq. (3.36), and visualizing the threshold line $y = \pm DP$ in red (Figure 3.10), the segments with their curvature value exceeded the threshold limit will be pulled out from the original curve, shown in (Figure 3.11).

Table 3.5 An example B-spline curve 'C'

Control points 'CP'	Knot vectors 'U'
(0, 0)	0
(3, 1)	0
(4, 1)	0
(4.5, -4)	0.5
(5, 4)	1
(5.5, -4)	1.5
(6, 1)	2
(8, 0)	2.5
	3
	3
	3

After finding out each end points of those selected curve segments and marking them out with the blue circles in (Figure 3.11), the whole curve has been segmented

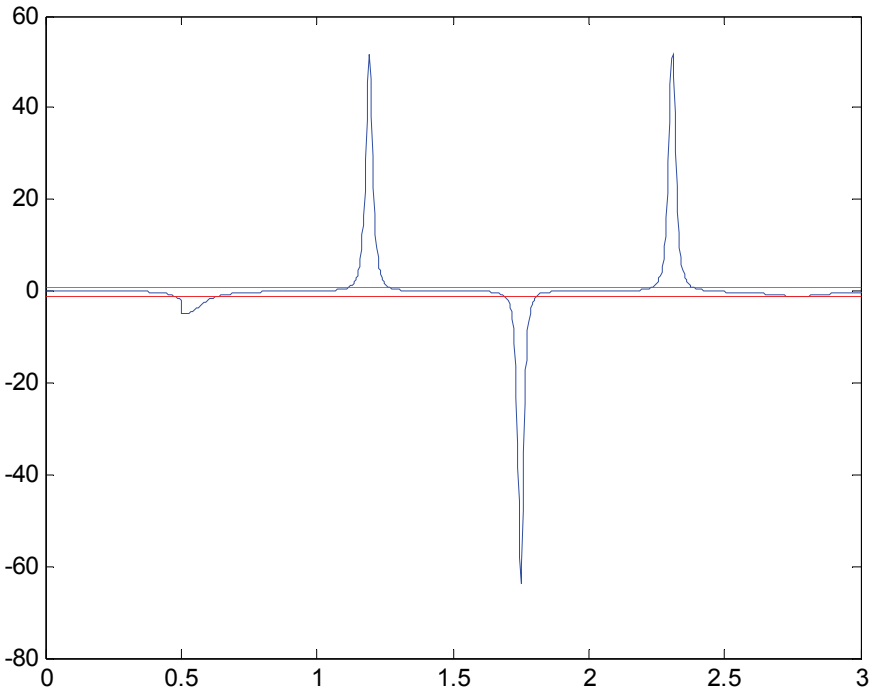


Figure 3.10 Curvature profile of 'C' and with the threshold lines '$y = \pm DP$'

(where DP was preset to be 1 in this case)

into several individual parts with each part either to be believed as relatively smooth

or relatively curved. Considering the very end points of the whole objective interval, namely $C(0)$ and $C(3)$ in this specific example, along with the points that I marked out, these points are recognized as the *Dividing Points.*

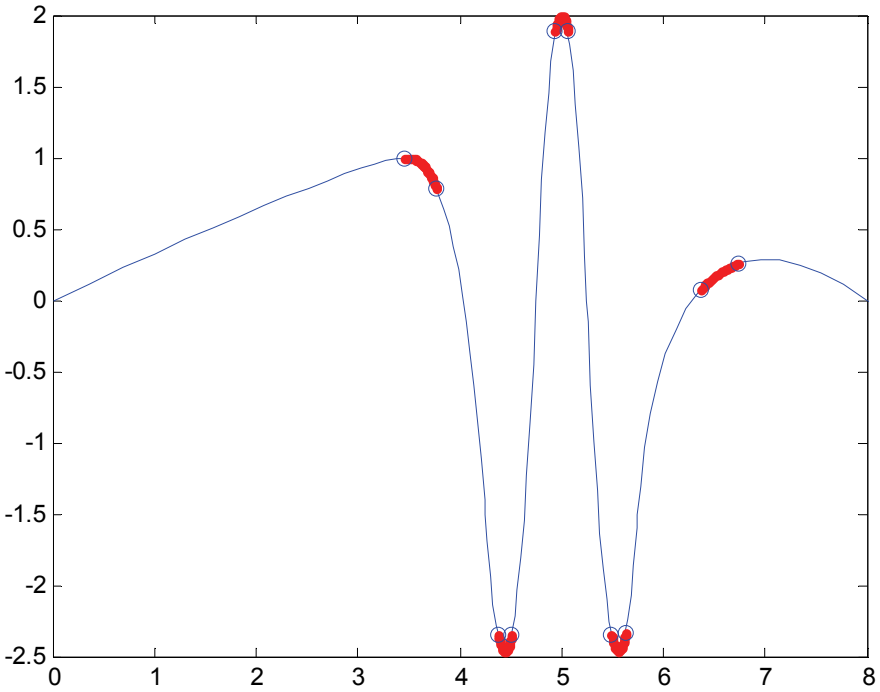


Figure 3.11 Corresponding selected curve segments in 'S'

If the corresponding parameter value u of these dividing points are denoted as

$$Udivid = \{u_1, u_2, u_3, \cdots u_n\}, \tag{3.37}$$

the total curve length within the objective interval

$$S = \int_{u_1}^{u_n} |C'(u)| \cdot du \tag{3.38}$$

can also be written in the form of

$$S = \int_{u_1}^{u_2} |C'(u)| \cdot du + \int_{u_2}^{u_3} |C'(u)| \cdot du + \cdots \int_{u_{n-1}}^{u_n} |C'(u)| \cdot du\,. \tag{3.39}$$

As for this specific curve 'C', n is equal to 12.

Since the dividing points are selected out based on the curvature features, the calculation efficiency of this algorithm for obtaining the curve length should be even higher comparing to the previous rough segmentation algorithm in Section 3.5.1. It is indubitable that the total curve length of 'C' is fixed, as been found as 20.868231 units long in this case. However, the time consumption from using Eq. (3.38) and Eq.(3.39) are different:

Table 3.6 Efficiency comparison of finding the arc length of 'C' using traditional way and curvature-based algorithm (Accuracy is set to be 1.0×10^{-6})

	Traditional Method	Curvature-based segmentation Method
Functions	$S=\int_{u_1}^{u_n}\left\|C'(u)\right\|\cdot du$	$S=\int_{u_1}^{u_2}\left\|C'(u)\right\|\cdot du+\int_{u_2}^{u_3}\left\|C'(u)\right\|\cdot du+\cdots$ $\cdots+\int_{u_{n-1}}^{u_n}\left\|C'(u)\right\|\cdot du$
Time Consumed	0.5011 (s)	0.3622 (s)
Time saved	0.1275 (s)	

As expected, the calculation time has been shortened for around 30%, even a lot of initialization code lines in the program has been executed repeatedly. In another word, the core algorithm from curvature based method could be even more effective and speedy than it looks like from this example.

3.6 A New Approach to Arc-length Approximation for NURBS Curves with Sharp Corners

As a matter of fact, it is common that for some industrial designs, the contour of some parts with sharp corners may also be represented by the presence of repeated knots as aforementioned. To evaluate the total length of such a curve, the traditional adaptive Simpson's rule could be used. However, for these kinds of special cases, by simply apply the traditional adaptive Simpson's rule over the entire objective parametric interval might either consume longer computational time or lose approximation accuracy. To elaborate this, an illustrative example is presented as follows.

A general NURBS curve *C(u)* of order three with the knot sequence $[0,0,0,1,1,2,3,3,3]$ is shown in (Figure 3.12).

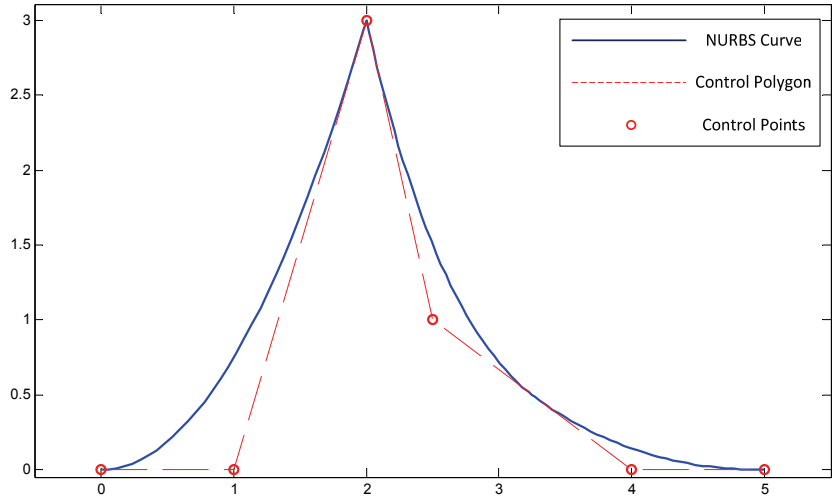


Figure 3.12 The shape of the example NURBS curve *C(u)*

According to Eq. (3.11), the total length of *C(u)* can be written as:

$$S = \int_0^3 \left| C'(u) \right| \cdot du \tag{3.40}$$

For a clearer understanding, Figure (3.13) shows the shape of the integrand of Eq. (3.40), namely

$$f(u) = \left| C'(u) \right| . \tag{3.41}$$

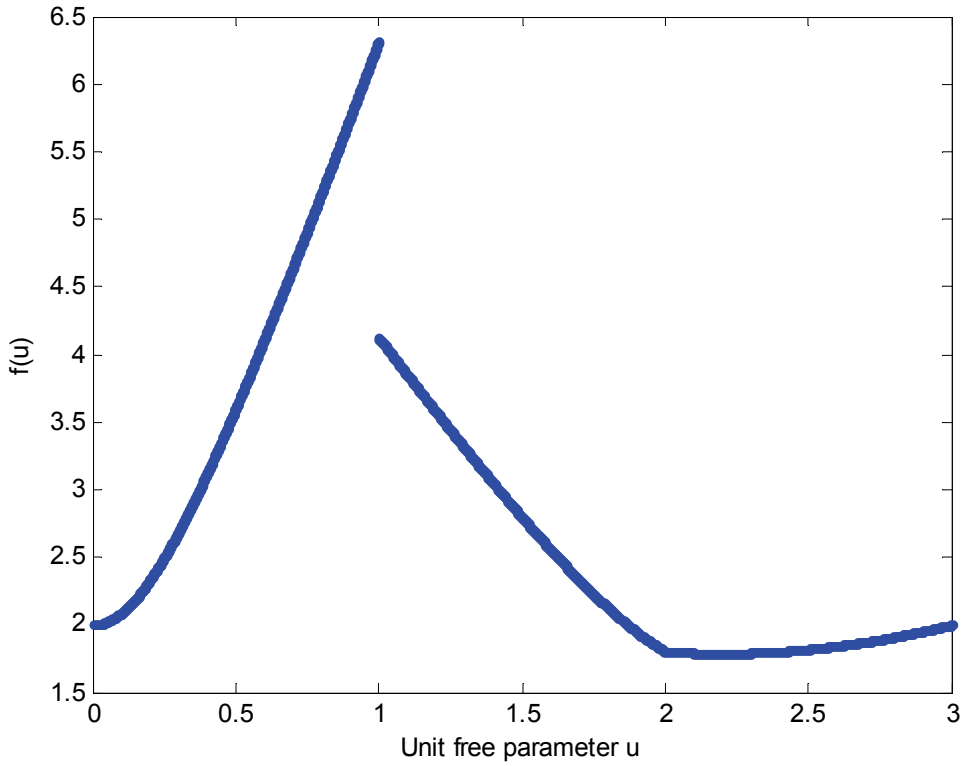


Figure 3.13 Image of the Integrand $f(u) = \left| C'(u) \right|$

According to the aforementioned approximation technique of the adaptive Simpson's method, the whole objective parametric interval will keep bisecting into more sub-intervals until the determination condition is met. As Figure (3.5) shows, a parabola is always passing through those three points, namely $f(a)$, $f(b)$ and $f(\frac{a+b}{2})$, on the objective function to fit and approximate the curve of the integrand over its corresponding parametric sub-interval [a, b].

Guided by this basic principle, the arc-length approximation procedure of the example above can be carried out by directly implementing the adaptive Simpson's rule over the entire parametric interval. The final approximation result would be obtained when the iteration procedure automatically stopped by the determination condition. Surely the error of the result can be controlled within the prescribed tolerance, for the determination condition was set based on accuracy demand. However, with this traditional way, a considerable computational redundancy situation may occur. The following Figure (3.14) illustrates step-by-step how the algorithm bisects its current parametric interval and when it determines to move on to the next section during the actual process.

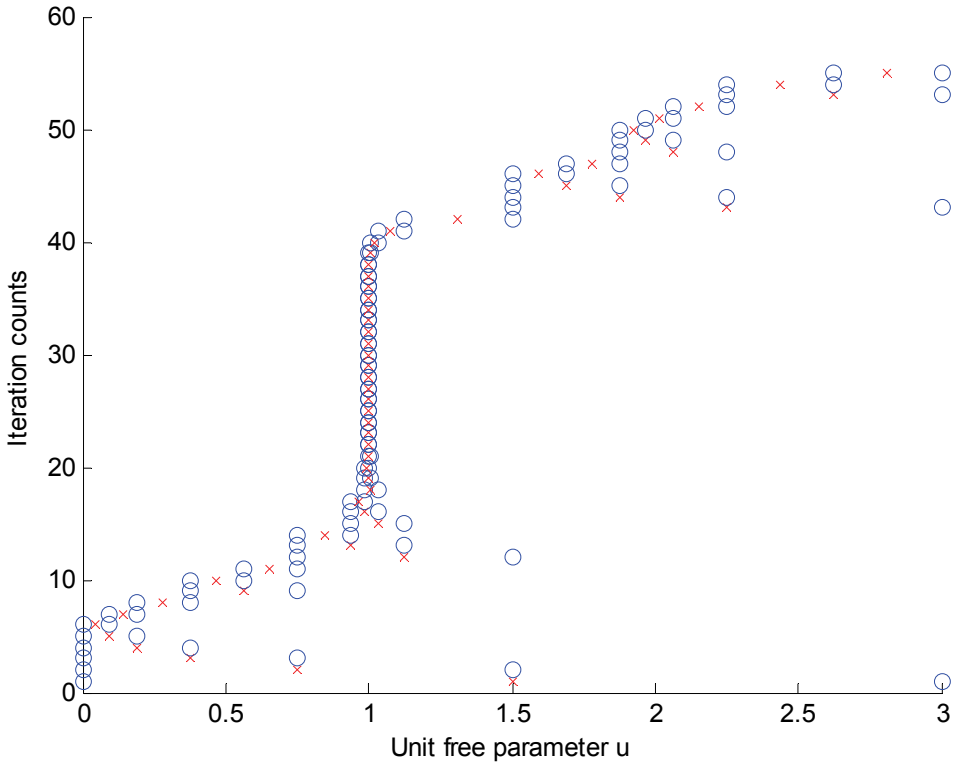


Figure 3.14 Detailed computational steps during the actual approximation of the traditional method

As shown above, for each iteration count, the blue circles indicate the lower and upper parametric boundary of the current sub-interval which is under the evaluation procedure. While the red 'x' cross at the same horizontal level marks out the corresponding parametric mid-point. Noticeably, a considerable amount of iteration counts has been found when the algorithm try to approximate the curve length at the neighborhood of the parametric value 1. The actual point *C(1)* is located exactly at the only sharp corner on the example NURBS curve *C(u)*. And consequently, to approximate the total length of this NURBS curve, the basic three-point Simpson's method has been executed 55 times when using the traditional way.

The reason why the traditional adaptive Simpson's method requires a relatively greater number of recursive procedures at the sharp corner is that the norm of the derivative function of the original NURBS curve, namely $\left|C'(u)\right|$, is discontinues at that point. Therefore, to avoid such redundant computational loads, a new piecewise adaptive Simpson's method is proposed.

By applying the proposed new method, the overall computational load could be decreased significantly. Since the cause of the redundant iterations has been revealed, the new proposed algorithm will be illustrated as follows.

If there is any sharp corner in the designed curve, according to the given information of the objective NURBS, namely the knot sequence and the control points, the corresponding parametric value of the exact points of those sharp corners can be found. Subsequently, instead of approximating the full length of the curve directly, the entire objective parametric interval is actually spitted up by the

afore-obtained parametric values. However, the parametric boundaries of the new sub-intervals are not simply defined by the exact dividing parametric values but the infinite close values from both left and right sides of the dividing points.

As in this example, the corresponding parametric value at the only sharp corner is $u=1$. To illustrate the abovementioned method and for better comparison purpose, the entire objective parametric interval is firstly divided into two parts by the exact value $u=1$. Therefore, the arc-length approximation equation of *C(u)* can be found as:

$$S=\int_0^1 \left|C'(u)\right|\cdot du+\int_1^3 \left|C'(u)\right|\cdot du \tag{3.42}$$

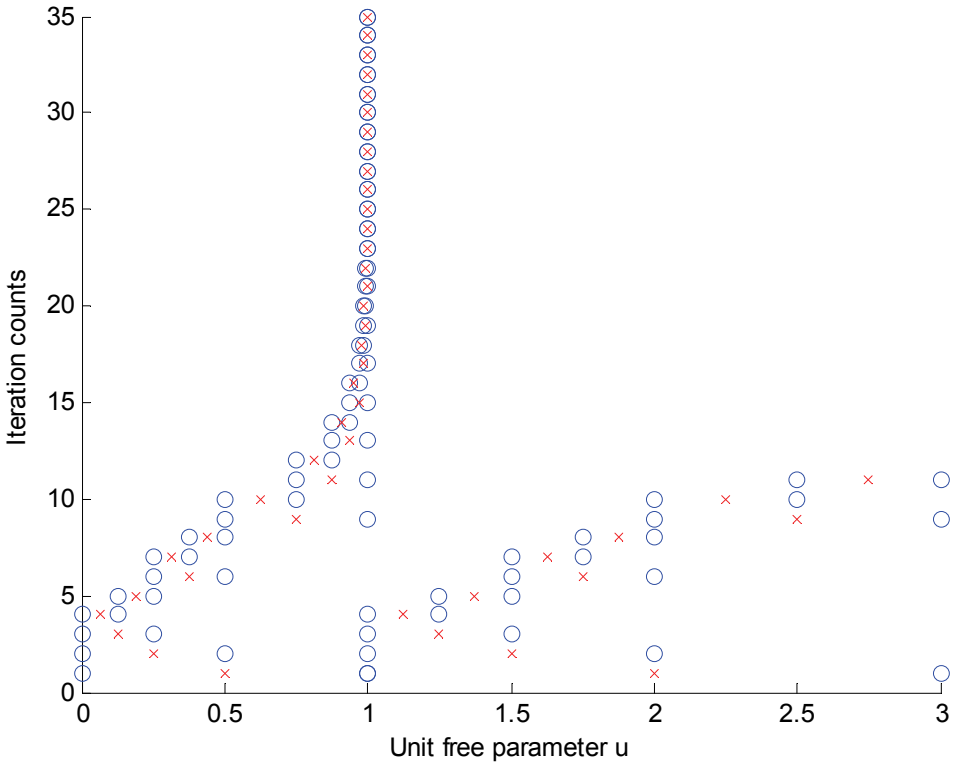


Figure 3.15 Detailed computational steps during the actual approximation when setting the dividing point as $u=1$

By drawing out the illustrative Figure of the detailed iteration procedure, it can be found that the redundant computational loads for approximating the arc-length at point $C(1)$ and its neighborhood remain still.

However, according to the proposed dividing method, the resultant sub-parametric intervals are [0, 1), and (1, 3] respectively. Thus, instead of using the traditional arc-length approximation method of Eq. (3.40) and the dividing method from Eq. (3.42), the new approach can be written as:

$$S = \int_0^{1-\mu} \left| C'(u) \right| \cdot du + \int_{1+\mu}^{3} \left| C'(u) \right| \cdot du \tag{3.43}$$

where $\mu \to +0$.

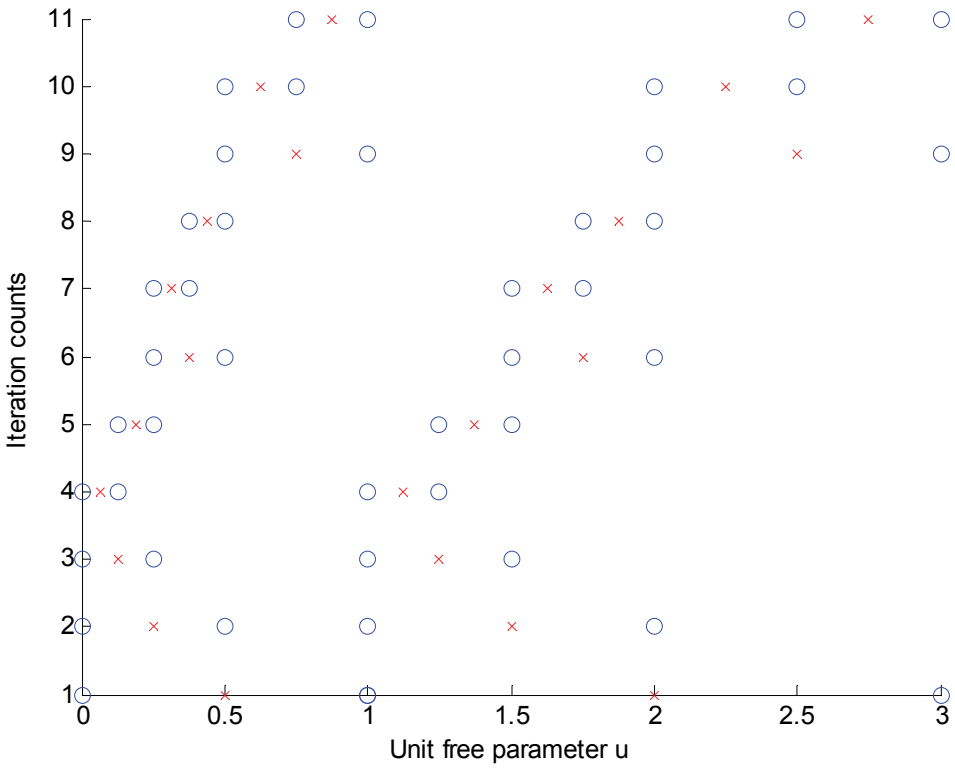


Figure 3.16 Detailed computational steps during the actual approximation using the proposed method

By implementing this new method of Eq. (3.43), the calculation efficiency can be significantly increased compare to either of the two methods above. This fact is illustrated by Figure (3.16), which shows the detailed computation steps during the actual approximation.

Clearly, According to Figure (3.16), the iteration count has been significantly reduced, especially at the neighborhood of the sharp corner point *C(1)*.

Table 3.7 Comparison between the aforementioned algorithms when approximating the arc-length of the example NURBS curve *C(u)*. (Accuracy is set to be 1.0×10^{-6})

Algorithms	Iteration counts	Time consumption (s)	Approximation result
Traditional method	55	0.067	8.4634987
Direct-dividing method	46	0.056	8.4634986
Proposed method	22	0.029	8.4634882

Evidentially, in the shown example, the proposed method eliminates the redundant iterations by about 60% of that from the traditional method, and shortens the computational time by more than 50%. Nevertheless, the application of this new approach can only benefit to certain kinds of situations.

Generally, there are two ways to make sharp corners appear in a NURBS curve. To a NURBS curve with order *k*, one way to form sharp corners in the curve, is to have repeated knots in its knot sequence besides the first and the last *k* elements.

Furthermore, the number of those repeated knots N_{knots} should satisfy:

$$N_{knots} = k - 1 \tag{3.44}$$

The other way to achieve so is to have consecutive repeated control points along with the NURBS curve. Similarly, to form a sharp corner in this way, the number of those repeated control points N_{CP} has also the following prerequisite to meet:

$$N_{CP} \geq k - 1 \tag{3.45}$$

However, our proposed method may not be such effective or to have notable advantages upon the NURBS curves that use repeated control points to generate sharp corners. For better understanding, a typical NURBS example of such case is presented as follow.

Assuming the specifications of a NURBS curve *S(u),* which uses repeated control points to form the designed sharp corner, is defined in Table (3.8) with its actual curve shape shown in Figure (3.17). (The corresponding weights for all the control points are set to be 1.)

Table 3.8 Specifications of the example NURBS curve *S(u)*

Control points	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
Coordinates (mm)	0 0	0.5 2.5	2.0 3.0	2.0 3.0	2.0 3.0	3.0 0.0	4.0 0.5	5.0 2.0
Knot vector	u_0	u_1	u_2	u_3	u_4	u_5		
values	0	0	0	0	1	2		
Knot vector	u_6	u_7	u_8	u_9	u_{10}	u_{11}		
values	3	4	5	5	5	5		

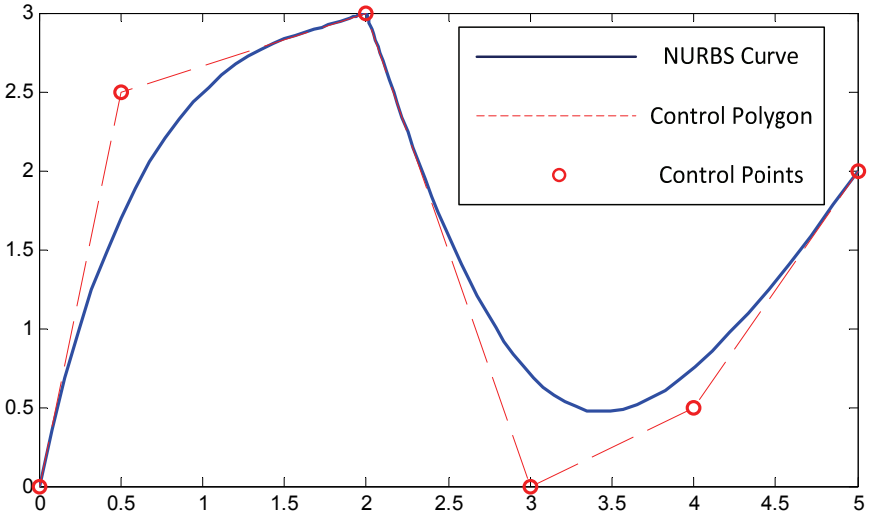


Figure 3.17 The shape of the example NURBS curve *S(u)*

To calculate the arc-length of this curve, the traditional adaptive Simpson's method is directly implemented over the entire parametric interval. The detailed computational step has also been traced:

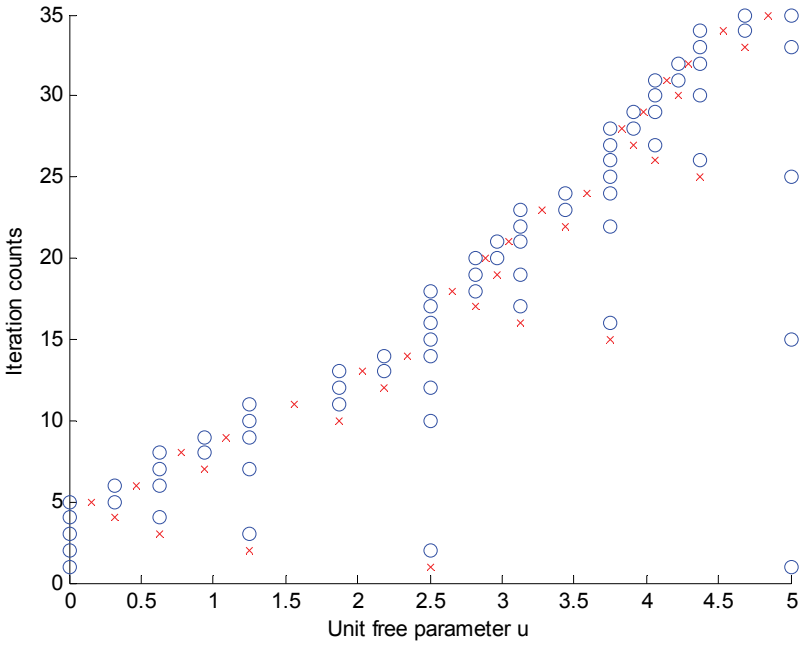


Figure 3.18 Detailed computational steps of the example

NURBS curve *S(u)* using the traditional method

As shown in Figure (3.18), the overall trend of the collected data turns out to be relatively smooth and evenly distributed along the entire parametric range. There is no such serious computational redundancy as Figure (3.14) shown from the previous example. This is due to the different shape of the functions: $f(u)=\left|C'(u)\right|$ and $g(u)=\left|S'(u)\right|$ derived from those two different cases.

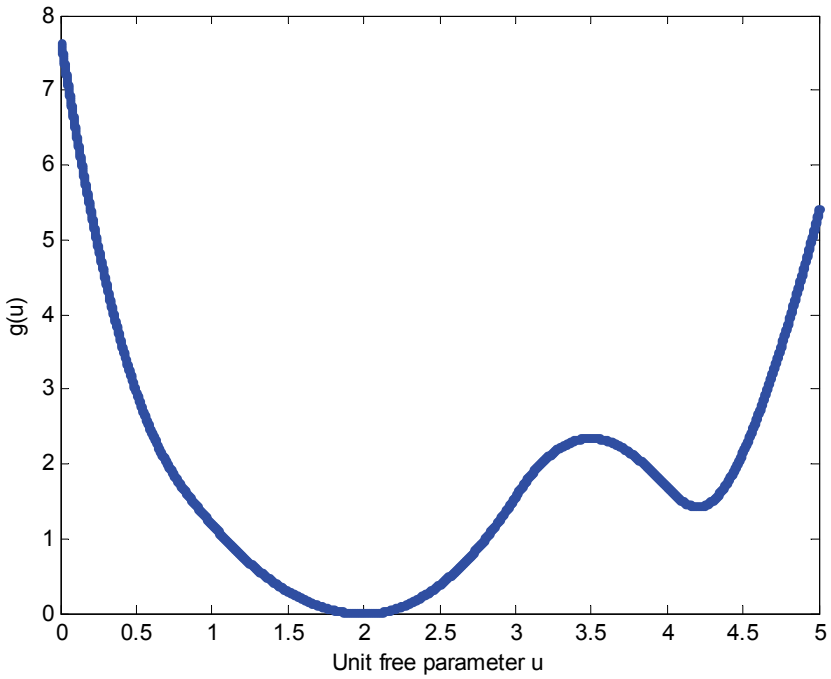


Figure 3.19 Image of the Integrand $g(u)=\left|S'(u)\right|$

By comparing Figure (3.19) to Figure (3.13), it is not difficult to recognize the major difference between these two dominating integrands. Unlike the function f(u), g(u) is continuous over the whole parametric interval. According to the previous deduced explanation that pointed out the cause of the computational redundancy is due to the discontinuity of the dominating integrands, the presence of the difference while implementing the same traditional method on these two NURBS cases has now been given a reasonable explanation. This means, although both the NURBS curve

C(u) and *S(u)* have the presence of a sharp corner geometrically, the norm functions of their tangent vector f(u) and g(u) can be essentially different. Consequently, the arc-length approximation procedure could be totally diverse.

Fortunately, with our proposed method, the arc-length calculation procedure for NURBS curve can now be treated more efficient, especially for those special cases that may cause the abovementioned computational redundancy.

3.7 A Potential Problem in NURBS Arc-Length Approximation When Using Adaptive Simpson's Method

Generally, the adaptive Simpson's rule is applicable to approximate arc-lengths for most kind of successive curves. However, for some NURBS curves (Bezier and B-splines as well) with enormous changes of knot spans, unexpected errors may occur during the mathematical approximation procedure of this quadrature method. To demonstrate this, an example NURBS curve with its weight vector [1, 1, 1, 1, 1, 1, 1, 1, 1] is specified in Table (3.9) and plotted in Figure (3.20).

Table 3.9 The specifications of the example NURBS curve C(u)

Control points	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Coordinates (mm)	0 0	100 0	100 0	200 0	200 100	100 100	100 100	100 100	0 100

Knot vector	u_0	u_1	u_2	u_3	u_4	u_5
values	0	0	0	199.975	199.980	199.985

Knot vector	u_6	u_7	u_8	u_9	u_{10}	u_{11}
values	199.990	199.995	200	400	400	400

By implementing the traditional Simpson's rule, the final approximation result for the arc-length of this curve is 200.000 *mm*. However, this result can be easily identified and geometrically proved to be wrong by Figure (3.20). This is due to the presence of the short knot spans from u_3 to u_8.

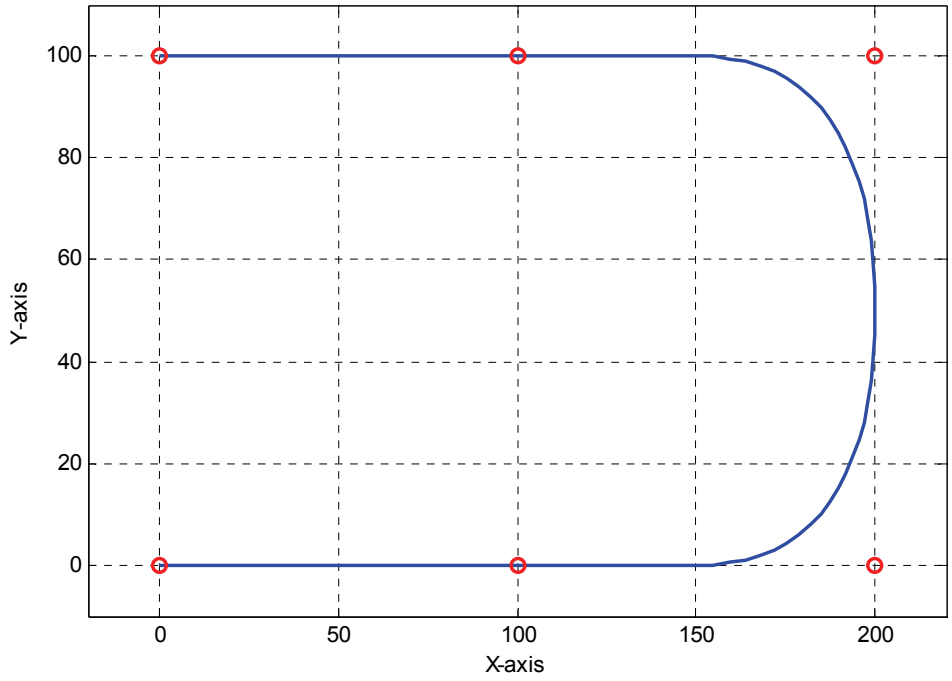


Figure 3.20 The example NURBS curve C(u) and its control points

As introduced earlier in this Chapter, the Simpson's rule needs to bisect the objective parametric interval and evaluates the corresponding value of the function to approximate the curve length. In this given example, however, it appears that the midpoints of all the sub-intervals do not fall into the knot span $[u_3, u_8]$, but the determination condition of the algorithm is met. As a result, the actual curve length

within this parametric interval is neglected, which indubitably would lead us to an incorrect result. To solve this, an improved composite Simpson's method with marking technique is proposed here.

As shown in Figure (3.21a), instead of directly using the Simpson's method, a mark '0' would be attached to each knot span before implementing the core quadrature algorithm. Then the traditional Simpson's method is applied over the entire parametric interval. During this procedure, not only the current approximation result, denoted as S_1, would be stored but also the midpoints of all the parametric sub-interval at each iteration step would also be recorded in a vector *V*. Subsequently, as shown in Figure (3.21b), our proposed algorithm would check each knot span and change the original mark '0' into '1' for those who has at least one element from vector ***V*** falls into their knot span.

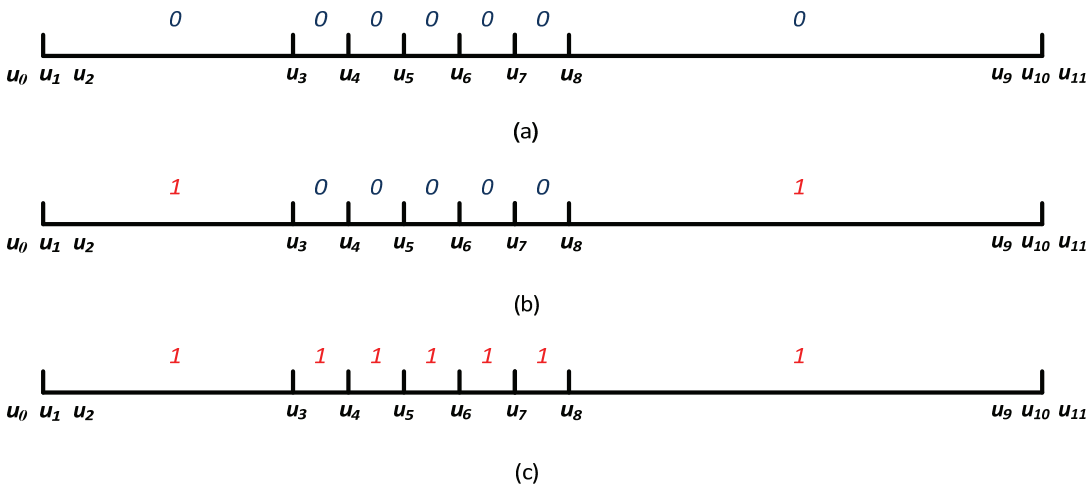


Figure 3.21 An illustration of marking technique over the exaggerated and disproportional parametric interval of the example NURBS curve C(u)

Once the previous steps were done, the core quadrature algorithm is implemented over the ignored knot span separately, which can be simply recognized

by their unchanged mark '0'. Ultimately, in order to make sure that the approximation algorithm does not skip or ignore any knot span no matter how close the knots are, the proposed algorithm would be stopped by not only the determination conditions from the traditional Simpson's rule, but also on the condition that all the marks turn to '1', as Figure (3.21c) shows.

To better illustrate this, the detailed bisect points of both the proposed method and the traditional method has been recorded and visualized as follows:

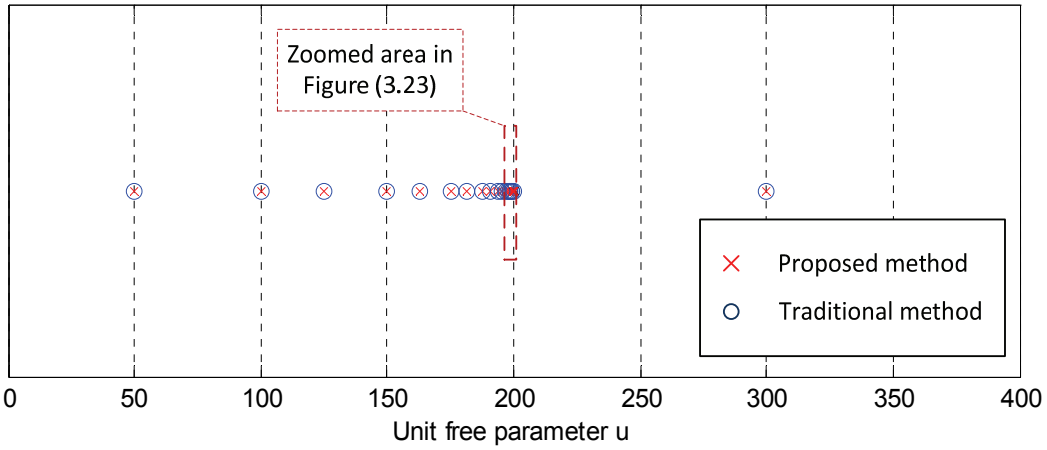


Figure 3.22 An illustration of recorded bisect points when using the proposed method and the traditional method

Figure (3.22) shows the global view of all the bisect points during the whole process, while Figure (3.23) provides a zoomed view of the marked parametric sub-interval (199.975, 200.000) in Figure (3.22), to better demonstrate and compare the difference between the aforementioned two algorithms.

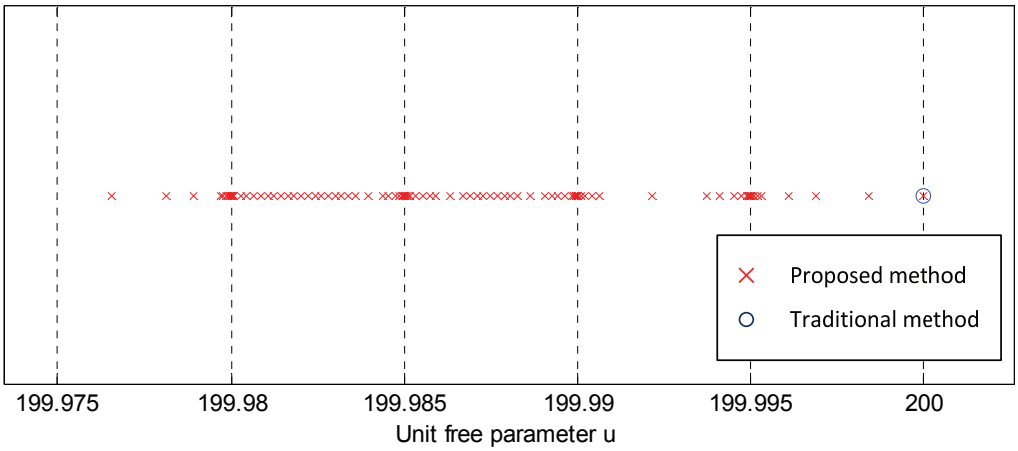


Figure 3.23 A zoomed view of the marked sub-interval (199.975, 200.000)

As shown above, it is intuitive that the traditional algorithm does not have any bisect points landed within the parametric interval (199.975, 200.000), as a consequence, the previous mistaken result is derived. On contrary, our proposed method solved this problem and came up with the accurate result.

By denoting S_2 as the approximation result derived from the later complementary quadrature procedure in this example, the final arc length of *C(u)* can be found as

$$S_u = S_1 + S_2 = 200.000 + 262.323 = 462.323(mm)\,, \tag{3.46}$$

by our proposed method.

Chapter 4 Proposed Arc-Length Parameterized NURBS Tool Path Generation

4.1 Introduction

Assuming a set of points $P_i = [P_1 \quad P_2 \quad \cdots \quad P_m]$ are sampled from the genuine NURBS tool path curve $S(u)$, the corresponding arc length $S_i = [S_1 \quad S_2 \quad \cdots \quad S_m]$ that measures from the starting point of the curve to the sample points can be calculated using the aforementioned Curvature-based Simpson's Method. By minimizing $\sum_{i=1}^{m}\left(C(S_i) - P_i\right)^2$ from using the least square method, an arc-length NURBS $C(s)$ can be found as

$$C(s) = \begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix} = \sum_{i=0}^{n} N_{i,p}(s) \cdot Q_i \qquad (0 \le s \le L) \tag{4.1}$$

where the parameter *s* is the actual arc-length.

After deriving the theoretical arc-length NURBS tool path, another indispensable procedure for finding all the cutter locations is the feed rate profile generation. In this chapter, a new feed rate profile generation algorithm in the CNC interpolation module is proposed. In which, both the accuracy demand and machine axial acceleration limit are globally bounded. Finally, a more smooth and accurate tool trajectory can be determined via using this new method.

4.2 Accuracy Bounded Piecewise Feed Rate Adjustment

When performing a high prescribed feed rate machining, it is possible that the expected feed rate may not be appropriate during the entire machining process for the accuracy demand could be violated. This is mainly because the machine sampling period T_s might not be short enough to refresh and change the current tool motion status under such a high feed rate over those segments. Consequently, due to this kind of lagging, those curve segments with relatively high curvatures could suffer from accuracy loss as Figure (4.1) shows. Therefore, to maintain the tool trajectory error within the machining tolerance, the feed rate must be revised according to the machine properties and the paths' local curvatures.

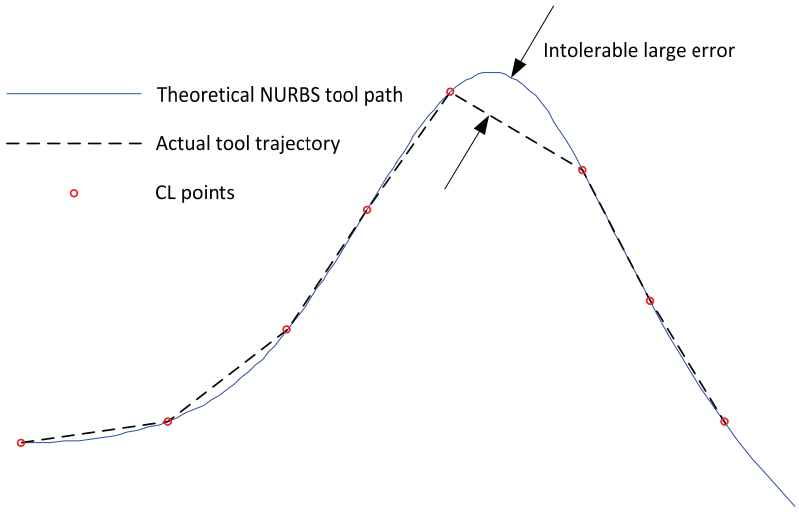


Figure 4.1 An exaggerated demonstration of Large chord errors at high curvature parts

4.2.1 Principle of Finding Maximum Valid Feed Rate According to Machine

Properties and Curvature of the Curve

Assuming a circular shaped curve with its curvature radius *R* is known as the theoretical tool path that is about to be processed. The relative machine configuration and the machining accuracy requirement are denoted as follows.

Table 4.1 Declaring list of the Parameters

Subjects	Values
Machine Sampling Time	T_s
Accuracy Tolerance	ε_{err}

Since the radius of the curve is constant, the largest machining error occurs at half way between the two adjacent *CL* points. An exaggerated illustrative actual tool trajectory during one machine sampling period is shown in Figure (4.2).

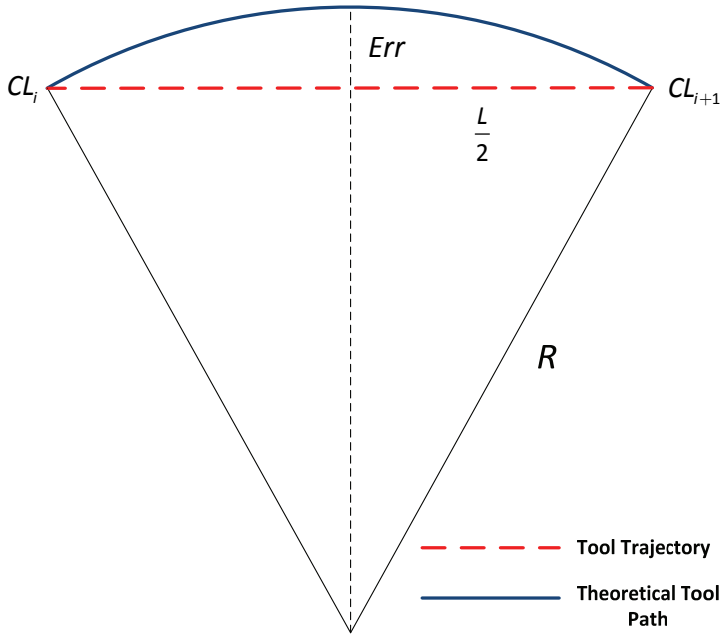


Figure 4.2 An exaggerated illustration of machining error during one sampling period

According to Figure (4.2), the relationship between the actual machining error *Err* and the chord length *L* of $\left|CL_iCL_{i+1}\right|$, or also as known as the step-size in the following contents, can be found as

$$\left(R-Err\right)^2+\left(\frac{L}{2}\right)^2=R^2, \tag{4.2}$$

since the step-size can not be negative, thus,

$$L=\left|2\sqrt{R^2-\left(R-Err\right)^2}\right|. \tag{4.3}$$

In order to assure the accuracy is bounded as required, the machining error *Err* can be no larger than ε_{err}. Thus, by letting

$$Err=\varepsilon_{err} \tag{4.4}$$

In Eq. (4.3), the maximum valid step-size $L_{\max}$ can be founded. Therefore, the maximum valid feed rate can be calculated:

$$F_{\max}=\frac{L_{\max}}{T_s}=\left|\frac{2\sqrt{R^2-\left(R-\varepsilon_{err}\right)^2}}{T_s}\right| \tag{4.5}$$

4.2.2 Curvature-Based Segmentation with Afterwards Feed Rate Adjustment

Since the radius of the curvature *R* can also be replaced by $\frac{1}{k}$, where *k* is the curvature of the curve, the function that specifies the relationship between the valid feed rate and the local curvature of the curve for general cases can be written as,

$$F=\frac{\left|2\sqrt{\left(\frac{1}{k}\right)^2-\left(\frac{1}{k}-\varepsilon_{err}\right)^2}\right|}{T_s} \tag{4.6}$$

where T_s and ε_{err} are treated as constant for they were given and fixed before machining.

According to Eq. (4.6), and let the expected feed rate be denoted by F_r, a corresponding curvature k_r can be found. This k_r indicates the maximum curvature value that the theoretical tool path curve can reach without violating the accuracy requirement at the machining speed of F_r. In another word, k_r is the threshold value that separates the unprocesseable curve segments which can not be machined at the prescribed feed rate F_r from the feasible ones.

For example, when interpolating the following curve $C(t)$, shown as Figure (4.3), with its curvature profile shown as Figure (4.4), the highlighted red parts are the non-machinable curve segments when processing with the prescribed feed rate F_r.

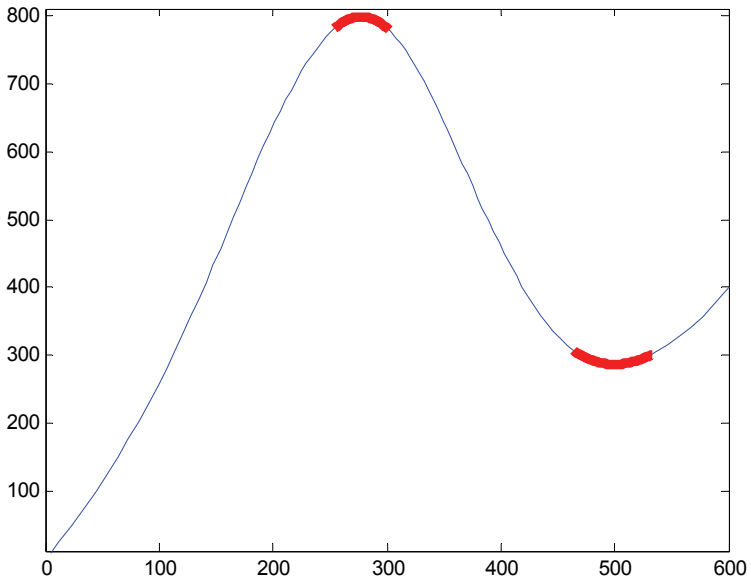


Figure 4.3 Shape of the example curve $C(t)$

The corresponding parameter value at each end point of the segments

$$P_{divide} = \begin{bmatrix} PD_1 & PD_2 & \cdots & PD_n \end{bmatrix} \tag{4.7}$$

are recognized as the dividing points (n=6 in this example). They divide the original parametric interval into several sub-intervals if there were curve segments with their

curvature higher than the threshold value k_r.

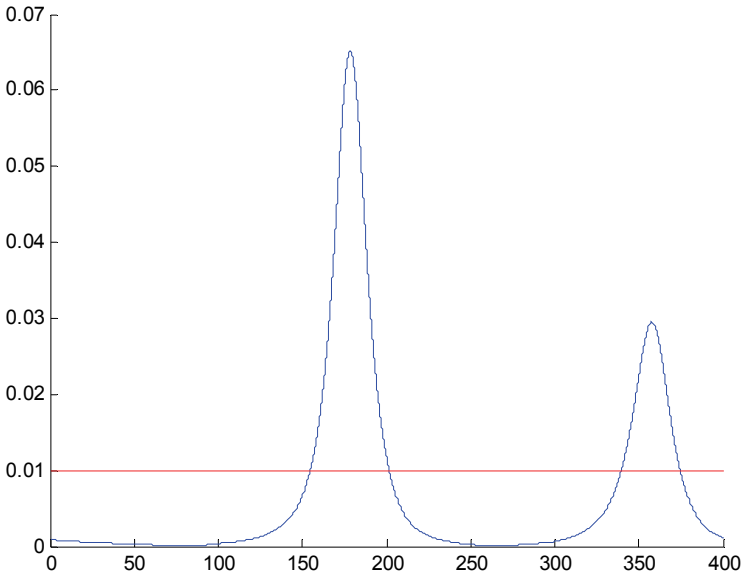


Figure 4.4 Curvature profile of the curve $C(t)$ with the threshold line $k = k_r$ in red

After identifying those segments that the prescribed feed rate is considered unqualified and over-high for them, new local feed rates need to be assigned to each of those segments individually. Since the curvature profile is known, as well as the dividing points, the highest curvature k_m within the parameter interval $\left[PD_i,\ \ PD_{i+1}\right]$ can be evaluated.

By substituting k_m into the Eq. (4.6), the maximum valid feed rate F_m over this part of the curve is finally determined. Now that, the overall feed rate profile has been piecewisely re-adjusted to ensure the machining accuracy.

4.3 Pre-machining Valid Feed Rate Adjustment with Bounded Axial Acceleration

4.3.1 Derivation of Axial Acceleration Functions for Each Axis

let one of the theoretical arc-length NURBS tool path segments from the previous step to be denoted as

$$C(s)=\begin{bmatrix}x(s)\\y(s)\\z(s)\end{bmatrix} \qquad (a\le s\le b) \tag{4.8}$$

and also the appropriate constant feed rate within the corresponding interval [a, b] as F_{ab}. According to the aforementioned arc-length NURBS properties in section 2.3.1, the unit tangent vector at any point on the curve can be found by evaluating the first order derivative of Eq. (4.8) with respect to the arc-length *s*

$$C'(s)=\begin{bmatrix}\frac{dx(s)}{ds} & \frac{dy(s)}{ds} & \frac{dz(s)}{ds}\end{bmatrix}^T=\begin{bmatrix}x'(s) & y'(s) & z'(s)\end{bmatrix}^T. \tag{4.9}$$

Now, since both the direction and the magnitude of the Feed rate were known, the velocity function over [a, b], therefore, can be derived as

$$\overrightarrow{F_{AB}}(s)=\begin{bmatrix}V_x(s)\\V_y(s)\\V_z(s)\end{bmatrix}=\begin{bmatrix}x'(s)\cdot F_{ab}\\y'(s)\cdot F_{ab}\\z'(s)\cdot F_{ab}\end{bmatrix} \qquad (a\le s\le b) \tag{4.10}$$

Assuming that t_1 represents any arbitrary moment during the machining process of this curve segment, and

$$t_2=t_1+\Delta t \tag{4.11}$$

is the upcoming moment, where Δt is set to be an infinitesimal time increase. For x-axis, the instantaneous velocities at t_1 and t_2, in this occasion, are denoted as v_1 and v_2 respectively. Since the time increase Δt is way more shorter than the machine sampling period T_s, the axial acceleration a_x during the time t_1 to t_2 ,thus, can be considered as uniform. Therefore, the uniform rectilinear kinematic

equation can be applied here as Eq. (4.12) shows.

$$v_2^2 - v_1^2 = 2 \cdot a_x \cdot (l_2 - l_1) \tag{4.12}$$

In which, l_1 and l_2 are the corresponding displacement of the tool along x-axis at moment t_1 and t_2. By solving a_x in the above equation, the acceleration provided by x-axis during this time span can be found as

$$a_x = \frac{v_2^2 - v_1^2}{2 \cdot (l_2 - l_1)} \tag{4.13}$$

The Eq. (4.13) can be simply expanded two steps further into

$$a_x = \frac{(v_2 + v_1)(v_2 - v_1)}{2 \cdot (l_2 - l_1)} \tag{4.14}$$

$$a_x = \frac{(v_2 + v_1)}{2} \cdot \frac{dv}{dl} \tag{4.15}$$

By post-multiplying $\frac{ds}{ds}$ to the right hand side of the above equation, it becomes

$$a_x = \frac{(v_2 + v_1)}{2} \cdot \frac{dv}{ds} \cdot \frac{ds}{dl} \tag{4.16}$$

For better demonstration purpose, an illustrative example curve which lies in XY plane is shown in Figure (4.5). P_1 and P_2 are the two tool location points at time t_1 and t_2 respectively. Since the time span is extremely short, the chord length $\left|P_1P_2\right|$ is considered to be equivalent to the arc-length $\left|\overset{\frown}{P_1P_2}\right|$, and the unit tangent vector at point P_1 is also considered as aligned with P_1P_2.

Therefore, if the unit tangent vector at point P_1 is found as

$$\vec{P_1} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}, \tag{4.17}$$

According to the geometric ratio relationship shown in Figure (4.5), the following Eq.

(4.18) holds.

$$\frac{dl}{ds} = \frac{x_p}{\left|\overrightarrow{P_1}\right|} = x_p \tag{4.18}$$

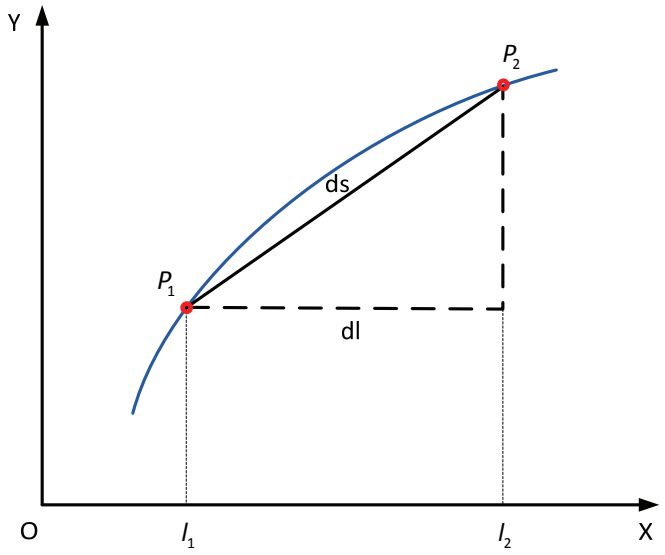


Figure 4.5 Geometric relationship between the tool locations at the two sampling moments

Since $\Delta t \to 0$, then from equation

$$v_2 = v_1 + a_x \Delta t \tag{4.19}$$

the first factor in Eq. (4.16) can be approximated as

$$\frac{(v_2 + v_1)}{2} \approx v_1 \tag{4.20}$$

Finally, by substituting Eq. (4.18) and (4.20) into Eq. (4.16), the axial acceleration function of x-axis with respect to parameter *s* is derived as

$$a_x(s) = V_x(s) \cdot V_x'(s) \cdot \frac{1}{x'(s)} \tag{4.21}$$

As the fact that the feed rate F_{ab} is not changing along the curve over the interval [a, b], the resultant acceleration $a(s) = \left[a_x(s) \quad a_y(s) \quad a_z(s)\right]^T$ at any point *s* is exactly the centripetal acceleration $a_n(s)$ at that point. Similarly, by finding out the

axial acceleration of y-axis and z-axis, the acceleration at point *s* can be written as

$$a(s)=\begin{bmatrix}a_x(s)\\a_y(s)\\a_z(s)\end{bmatrix}=\begin{bmatrix}V_x(s)\cdot V_x'(s)\cdot\dfrac{1}{x'(s)}\\V_y(s)\cdot V_y'(s)\cdot\dfrac{1}{y'(s)}\\V_z(s)\cdot V_z'(s)\cdot\dfrac{1}{z'(s)}\end{bmatrix} \tag{4.22}$$

where $V_x(s)$, $V_y(s)$ and $V_z(s)$ can be found from Eq. (4.10). Therefore, the axial acceleration is now represented as a function of *s*.

4.3.2 Feed Rate Re-adjustment Method According to the Machine Axial Acceleration Limit

After deriving the axial acceleration functions for each axis from the previous procedure, for some part of the theoretical tool path could still not be processed at the current feed rate due to the limit of the machine axial acceleration. By denoting A_x, A_y and A_z as the maximum axial acceleration and deceleration limit of each axis, the current feed rate F_{ab} over parametric interval $[a,b]$ can be found applicable if

$$\begin{cases}a_x(s)\le A_x\\a_y(s)\le A_y\\a_z(s)\le A_z\end{cases}\qquad (a\le s\le b) \tag{4.23}$$

holds.

If not, the expected feed rate turns out to be unsuitable, and has to be re-adjusted to meet the machine settings. Otherwise, intolerable trajectory error could be gained during the machining. Assume that an example axial acceleration function of x-axis has been visualized as Figure (4.6) shows.

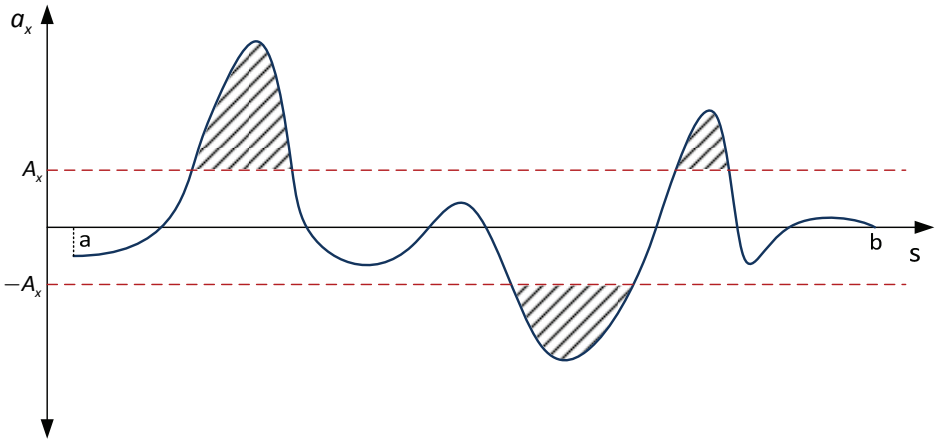


Figure 4.6 An example acceleration profile of x-axis

As fore-mentioned, the feed rate F_{ab} is recognized to be qualified over those curve segments whose axial acceleration curves are caught in the region formed by the two red dash lines, $a_x = A_x$ and $a_x = -A_x$. However, the shaded parts shown in Figure (4.6) indicate the intervals that the curve segments within them can not be processed by the machine at such a high feed rate. Therefore, in order to guarantee the accuracy and viability of the actual machining, the following method is introduced to correct the unfeasible feed rate.

Since the axial acceleration functions for each axis have been found, the maximum absolute value $a_{x\max}$ within the relative interval [a, b] can also be calculated by finding the maximum value of

$$f(s) = \left|a_x(s)\right| \tag{4.24}$$

over $s \in [a,b]$ using linear search algorithms. Similarly, after finding $a_{y\max}$ and $a_{z\max}$ for y-axis and z-axis, it can be found that the maximum axial acceleration of each axis exceeds their corresponding limits by the percentages of $(1-\alpha)$, $(1-\delta)$ and $(1-\gamma)$ respectively, where

$$\alpha = \frac{A_x}{a_{x\max}} \cdot 100\% \tag{4.25}$$

$$\delta = \frac{A_y}{a_{y\max}} \cdot 100\% \tag{4.26}$$

and

$$\gamma = \frac{A_z}{a_{z\max}} \cdot 100\% . \tag{4.27}$$

If the value of α, δ and γ are all greater than 1, the current feed rate then need not to be changed. Otherwise, in order to make sure the machine has the capability to maintain the tool trajectory on its path, the valid feed rate F'_{ab} over [a, b] is actually computed by

$$F'_{ab} = \sqrt{W} \cdot F_{ab} \tag{4.28}$$

where *W* denotes the lowest percentage value among α, δ and γ. The derivation of Eq. (4.28) is as follows.

Since the feed rate over this objective interval [a, b] is constant, which in another word, is indicating that there are no acceleration or deceleration motions along the tool path. The resultant acceleration provided by all the three axes is actually works only as the centripetal acceleration. Therefore, according to the formula

$$a_n = F_{ab}^2 \cdot k \tag{4.29}$$

where a_n is the centripetal acceleration and *k* is the relative local curvature, we multiply the percentage *W* to both sides of the Eq. (4.29):

$$W \cdot a_n = W \cdot F_{ab}^2 \cdot k \tag{4.30}$$

in which, the curvature *k* is fixed for the theoretical tool path is known. So that the Eq.

(4.30) can be re-written into

$$\frac{W \cdot a_n}{W \cdot F_{ab}^2} = \frac{W \cdot a_n}{\left(\sqrt{W} \cdot F_{ab}\right)^2} = k \tag{4.31}$$

This means, if the original centripetal acceleration is scaled by the factor *W*, the original feed rate needs to be scaled by $\sqrt{W}$. Consequently, by reconfigure the feed rate into F'_{ab}, the previous unviable axial acceleration profile of Figure (4.6) would now be qualified to the machine configuration as shown in Figure (4.7).

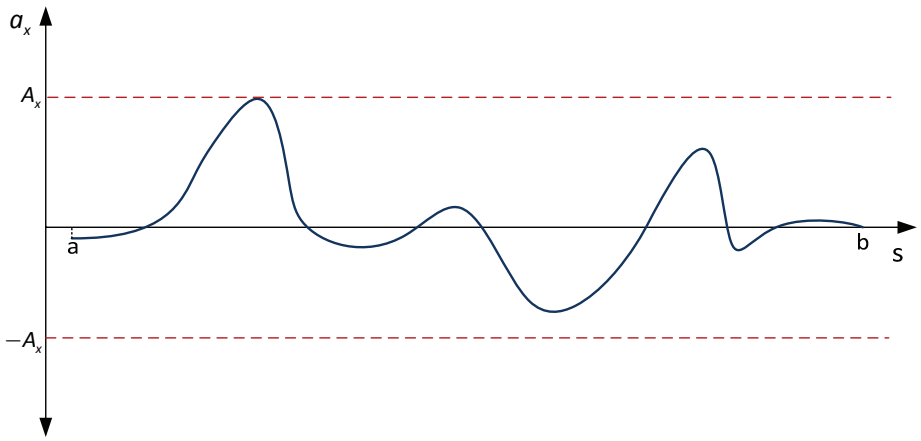


Figure 4.7 Acceleration profile of x-axis after feed rate adjustment

With this method, an appropriate feed rate over this sub-interval [a, b] can be found to prevent the axial acceleration from exceeding the machine's capability.

4.4 Feed Rate Profile Derivation During the Speed Change Procedure between Adjacent Sub-intervals

Now that the whole arc-length parameter interval $[0, \quad L]$ (*L* is the total length of the theoretical tool path) has been divided into several sub-intervals through the previous steps, and also assigned with their matching feed rates. As Figure (4.8)

shows, the feed rate profile is actually in the shape of discontinuous staircases.

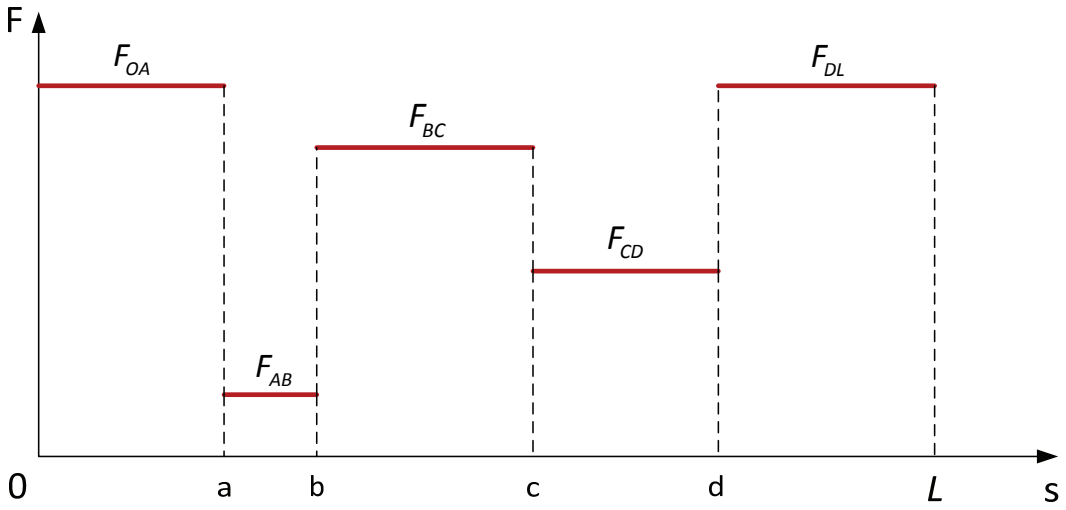


Figure 4.8 The 'Staircase-Shaped' feed rate profile

In order to avoid unpredictable trajectory error and to improve machining efficiency, the feed rate profile at the joint of each two adjacent sub-intervals need to be confirmed before feed this velocity profile into the NURBS interpolator. In traditional machining strategy [13] pp. (29-30), the tool movement has to be completely stopped before the next starts. The final feed rate profile gained according to this method is given as Figure (4.9).

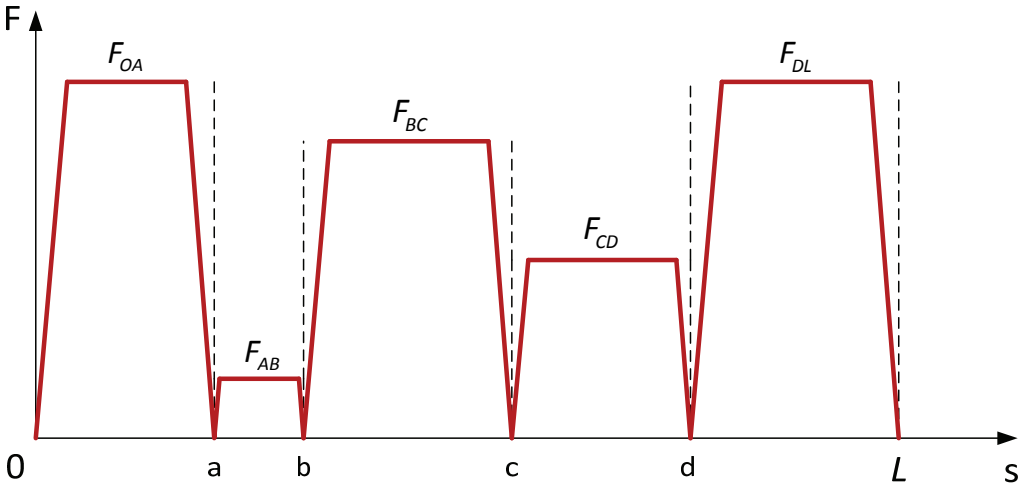


Figure 4.9 Feed rate profile result from traditional method

Since the feed rate has to decrease to zero by the end of the current interval and re-accelerate to the subsequently expected value after entering the next, a waste of machining time is caused due to the redundant acceleration and deceleration procedures. Therefore, a new method is proposed to compensate the aforesaid time lose as follows.

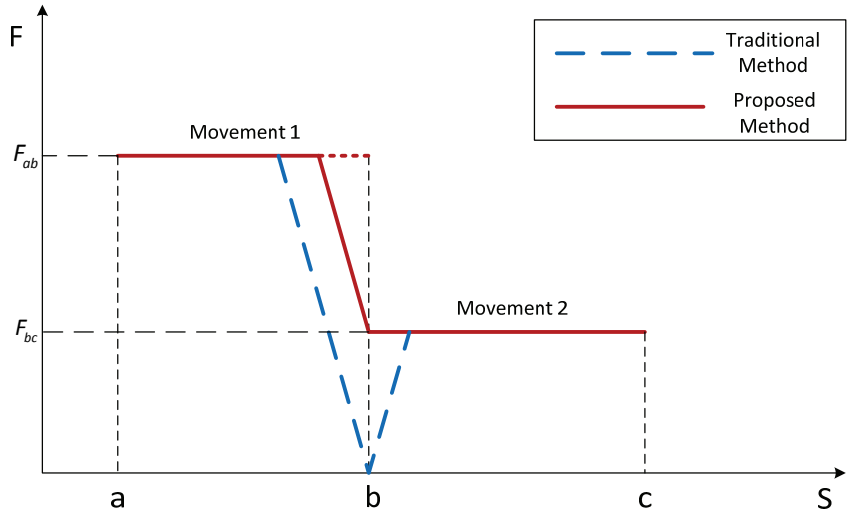


Figure 4.10 Comparison on feed rate profile of traditional method and proposed method during deceleration procedure

Basically, this proposed method can be known consist of the two following cases. For the first case, whenever the tool is finishing the current movement and about to enter the next interval whose relative feed rate is lower than the recent one, a deceleration procedure is carried out within the present interval. Instead of completely halt the tool motion at the end point of the present parameter interval, the feed rate directly decreases to the expected value of to the next parameter interval before the next movement takes place. As shown in Figure (4.10), the feed rate would have been already decreased to the expected value of F_{bc}, when the tool starts the movement of the next interval [b, c].

For the other case, the feed rate changing procedure would be conducted after the tool passes the boundary of the two adjacent intervals if the prescribed feed rate of the upcoming interval is higher than that at current. As Figure (4.11) shows, the tool moves at its prescribed speed over the whole span of [a, b], but starts to accelerate the moment it enters the next interval [b, c] until the feed rate meet the expected feed rate F_{bc}.

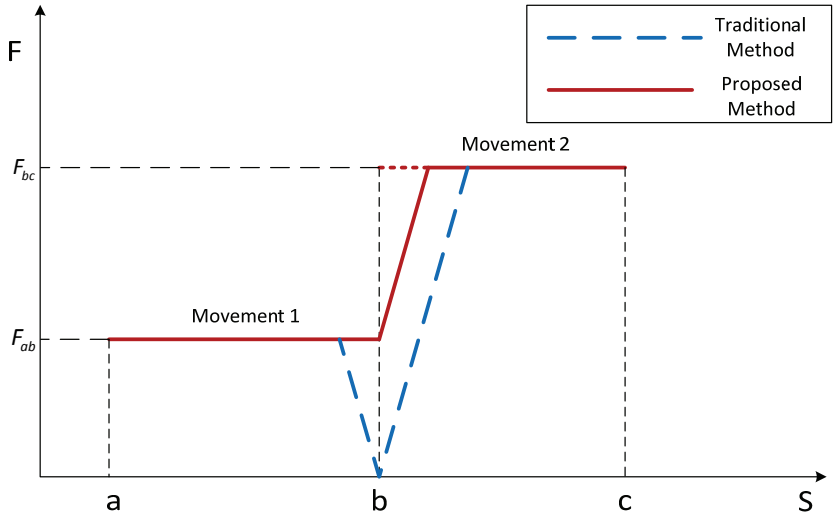


Figure 4.11 Comparison on feed rate profile of traditional method and proposed method during acceleration procedure

This proposed method can also be described as a 'postponed-accelerated and pre-decelerated' method based on its characteristic. Because the feed rate changing procedure always take place within the interval that with a higher expected feed rate. By using this method, the feed rate profile of Figure (4.9) which is resulted from the traditional method is now optimized as Figure (4.12) shows.

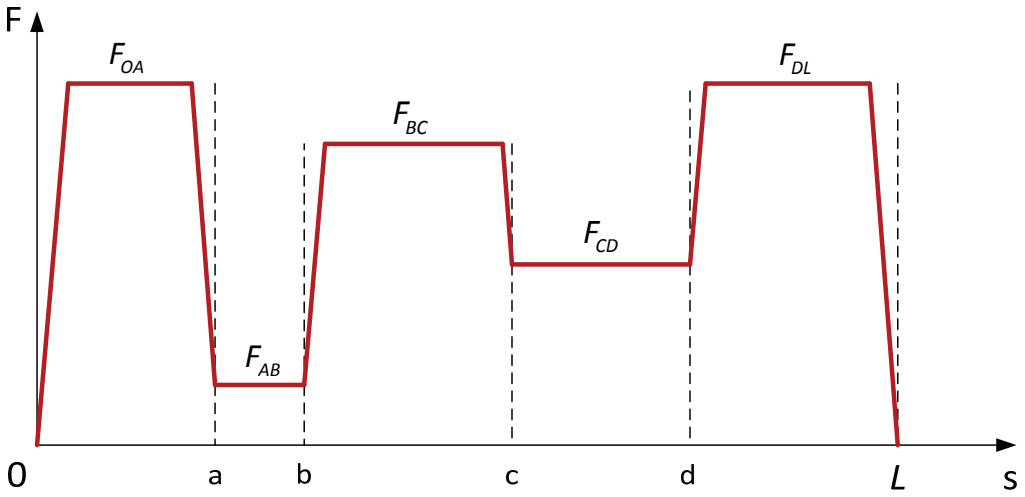


Figure 4.12 Feed rate profile result from proposed method

However, the feed rate profile gathered from actual machining can still be different from what has shown in Figure (4.12) due to the acceleration and deceleration motions along the curve may not turn out to be uniform. This is because not only does each axis need to provide the acceleration or deceleration along the curve, but in the meantime, they also need to give the relative centripetal acceleration as well.

As shown in Figure (4.13), a theoretical tool path $C(s)$ lies in XY-plane, and the tool moves from *A* to *B* along the curve. A random point *P* with its instant centripetal acceleration and linear acceleration along the curve denoted by $\overrightarrow{a_n}$ and $\overrightarrow{a_F}$ respectively is marked out. According to the equation for calculating centripetal acceleration:

$$a_n = V^2 / R \quad \text{or} \quad a_n = V^2 \cdot k \tag{4.32}$$

where *R* is the local radius of the curve and *k* is the corresponding curvature, it is clear that the centripetal acceleration is determined by both the tangential velocity and curvature at any reference point.

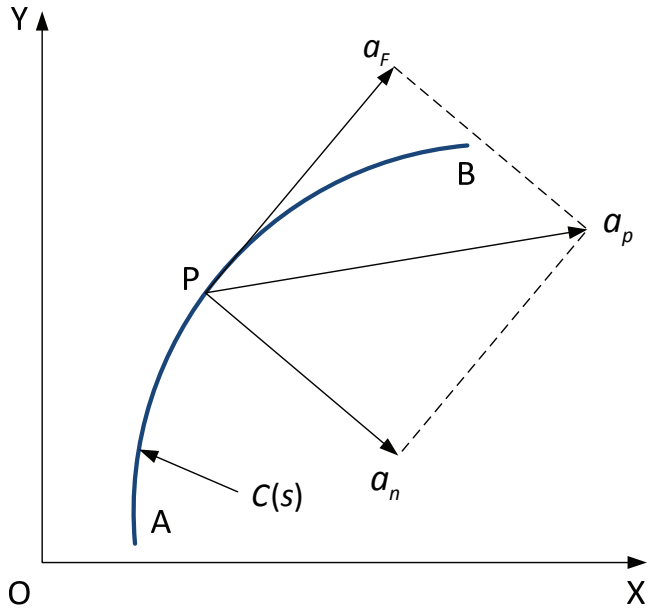


Figure 4.13 Illustration of the actual acceleration at a random point *P* on the curve

By combining the two vectors $\overrightarrow{a_n}$ and $\overrightarrow{a_F}$, the resultant acceleration vector $\overrightarrow{a_p}$ at point ***P*** can also be drown. By using the notations declared in Table (4.2),

Table 4.2 Declaring list of the notations

Subjects	Notations
Machine Sampling Time	T_s
Feed rate at current sampling period	F_o
Feed rate at next sampling period	F_t
Curvature at present location	k
Centripetal acceleration at current location	a_n
Linear acceleration at current location	a_F
Resultant acceleration of a_n and a_F	a_p

the following equations hold:

$$a_n = F_o^{\,2} \cdot k \tag{4.33}$$

$$F_t = F_o + a_F \cdot T_s \tag{4.34}$$

$$a_p = \sqrt{a_F^{\,2} + a_n^{\,2}} \tag{4.35}$$

From the equations above, it is clear that the higher the feed rate is at the current location, the greater centripetal acceleration is going to be needed. Consequently, the linear acceleration a_F is always changing along with the change of the curvature.

Assuming that $\vec{m}$ and $\vec{n}$ are the projection of $\vec{a_p}$ on x-axis and y-axis respectively, the machine is only capable to keep the tool trajectory within the error tolerance if

$$\left|\vec{m}\right| \le A_{\max} \tag{4.36}$$

and

$$\left|\vec{n}\right| \le A_{\max} \tag{4.37}$$

where $A_{\max}$ is the axial acceleration limit of the machine.

Since $\vec{m}$ and $\vec{n}$ are perpendicular to each other, the magnitude of vector $\vec{a_p}$ is always either greater than $\left|\vec{m}\right|$ and $\left|\vec{n}\right|$, or equal to one of them. Thus, by assigning

$$a_p = A_{\max} \tag{4.38}$$

and solving the equations of Eq. (4.33), Eq. (4.34) and Eq. (4.35) simultaneously, the maximum valid feed rate F_t during next sampling period can be found iteratively. Therefore, a valid feed rate profile that with bounded axial acceleration is finally generated as Figure (4.14) shows. The acceleration and deceleration procedures for

each sub-interval are actually not identical because of the required centripetal acceleration during each sampling period is constantly changing due to the variation of the curvature along the tool path. And thus, the linear acceleration and deceleration need to adapt itself according to Eq. (4.35)

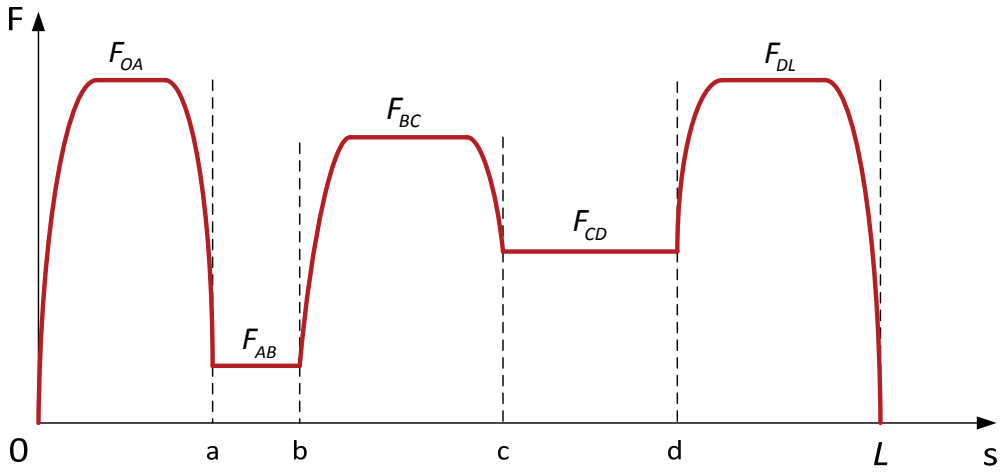


Figure 4.14 An finalized illustrative feed rate profile with bounded axial acceleration

4.5 Arc-length Parameterized NURBS Tool Path Generation Method

After deriving the feasible feed rate profile from the aforementioned method, the cutter locations can now be generated in the new arc-length NURBS interpolation module. According to the iteration function for generating the arc-length NURBS tool cutter locations,

$$s_{m+1} = s_m + V(t_m) \cdot T_s \tag{4.39}$$

where s_m denotes the value of parameter s at the moment of $t = t_m$, and $V(t)$ is the final feed rate function, the overall flowchart for implementing this arc-length

NURBS tool path generation method is shown in Figure (4.15)

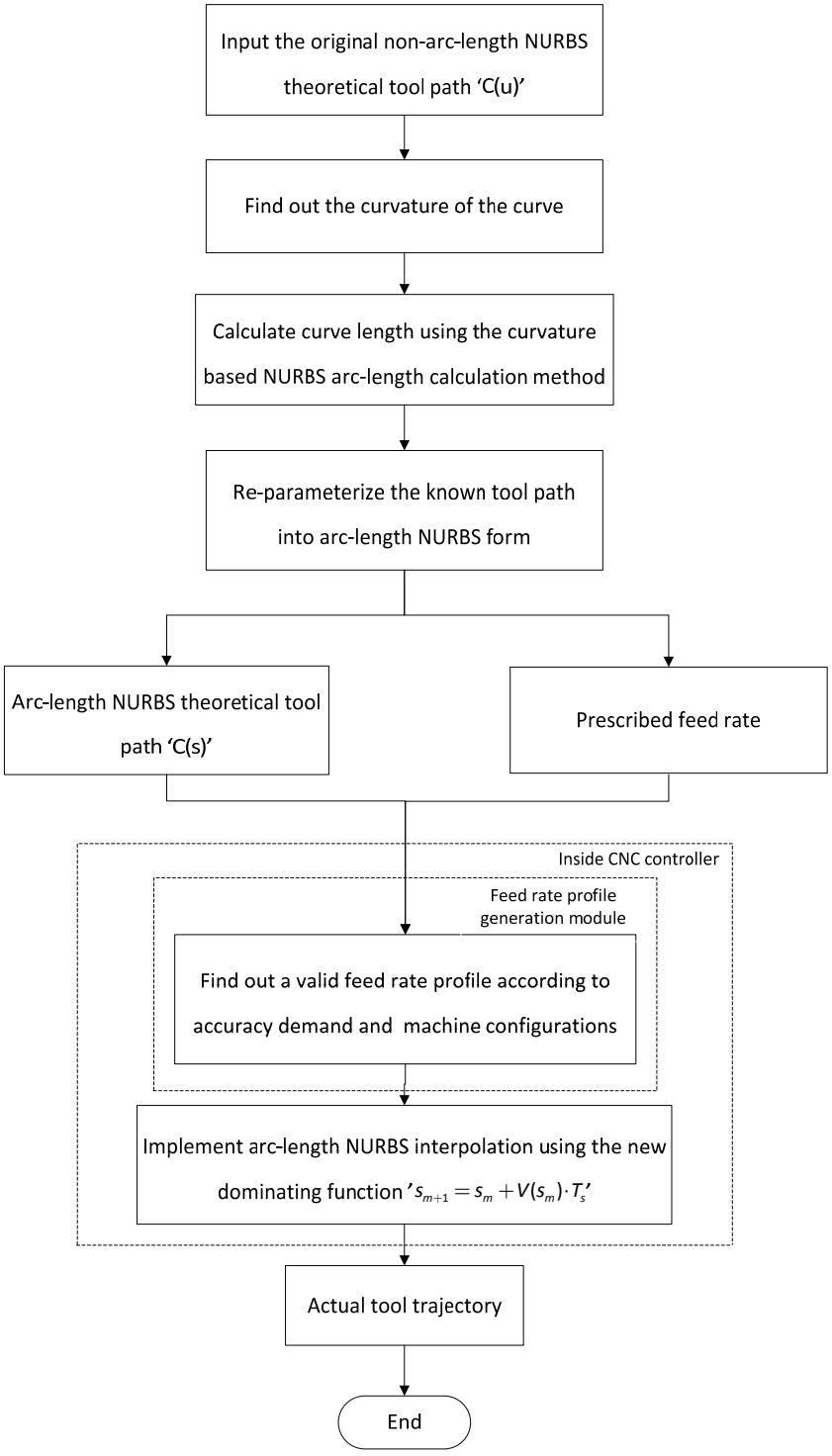


Figure 4.15 Flowchart of the arc-length NURBS tool path generation method

In which, the feed rate profile was generated within a separate module inside the CNC controller. The flowchart for the feed rate profile generation module is

shown in Figure (4.16).

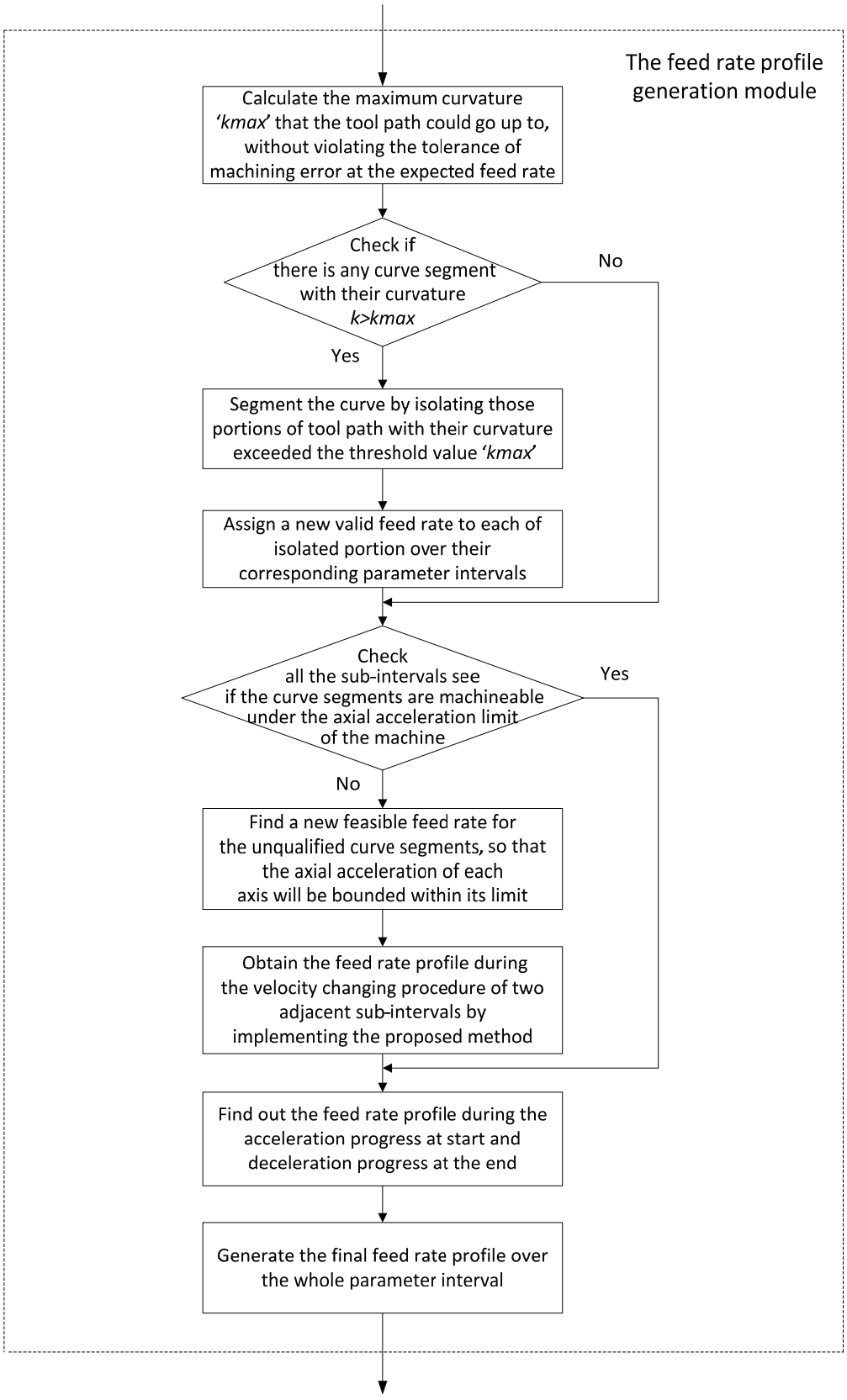


Figure 4.16 Flowchart of feed rate profile generation module

A simplified illustrative flowchart of this arc-length NURBS tool path generation method drown as Figure (4.17)

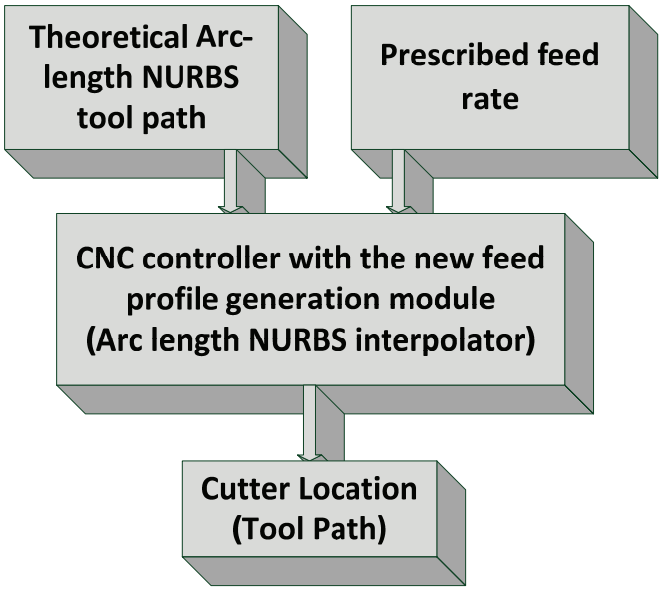


Figure 4.17 A simplified illustrative flowchart for arc-length NURBS tool path generation

Comparing the traditional NURBS tool path generation method shown in Figure (2.3), my proposed method not only changes the source NURBS tool path into the form of arc-length parameterized NURBS, but also uses a different interpolation algorithm in the CNC controller that comply with my new feed rate profiles.

Chapter 5 Applications

5.1 Introduction

By implementing the previous introduced tool path generation method, the actual feed rate during real time machining process can be considered adaptive to the curve shape and machine capabilities under the condition of the prescribed accuracy requirement could be fairly assured.

Due to the arc-length NURBS tool path generation method, as mentioned, can provide a smoother tool motion during the actual machining than the traditional way could, the chance of encountering unexpected big errors caused by inaccurate approximation from Taylor expansion is eliminated. Furthermore, even if an add-on method were conducted during the real time interpolation to compensate part of the feed rate fluctuation, the calculation time could be way longer than our proposed method.

In this chapter, a series of statistics are gathered from simulation results of a 3-axis machining example to demonstrate this new approach of tool path generation method. And also analysis and comparison will be included afterwards between the proposed method and the traditional one. The chosen theoretical tool path is a random 3-D NURBS curve in the general form; also the accuracy requirement and the machine configurations are set in a practical way.

5.2 Example of 3-Axis Arc-length NURBS Tool Path Generation

Considering a general NURBS tool path with its weights equal to 1, and control points listed in Table (5.1),

Table 5.1 Control points of the example NURBS tools path

Control points	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
Coordinates (mm)	0	1.8734	40.3782	30.2134	41.1982	54.0097	70.1092	77.0922
	0	51.4502	19.2108	40.0112	18.8291	3.0021	0.1109	14.1209
	0	5.7707	29.31815	69.9741	59.3641	1.51535	-30.1169	62.0139
Control points	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}
Coordinates (mm)	84.1902	92.2918	95.8217	127.8190	126.7021	129.4502	131.7289	140
	28.6765	50.1144	64.2100	124.1012	60.0010	42.5002	26.8745	0
	64.3768	74.5015	32.0989	11.9117	49.9851	41.2281	33.3931	0

Is about to be machined under the following prescribed tolerances and machine configurations shown in Table (5.2).

Table 5.2 Prescribed tolerances and machine configurations

Subjects	Prescribed values
Arc-length approximation accuracy δ	1.0×10^{-6} mm
Tool trajectory error tolerance ε	0.005 mm
Expected feed rate F_E	400 mm/s
Machine sampling period T_s	0.002 s

Machine axial acceleration limit	A_{max}	1g (9800 mm/s^2)

The actual shape of this tool path is shown in Figure (5.1).

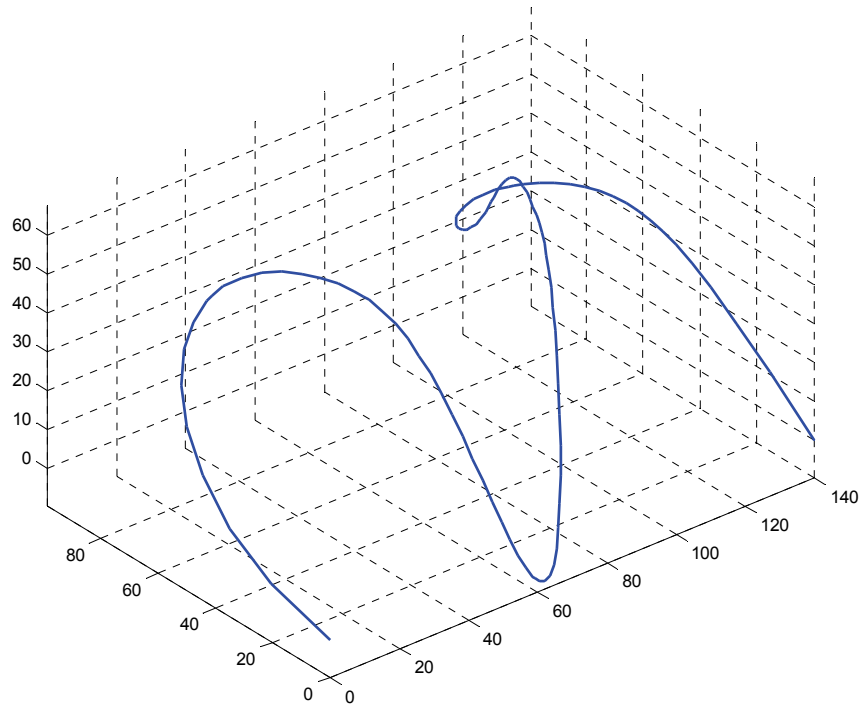


Figure 5.1 The shape of the given general NURBS tool path in space

First, the general NURBS curve is re-parameterized into an arc-length NURBS form by using the curvature-based Simpson's method to evaluate all the arc-lengths measuring from the starting point of the curve to all data points respectively. Then the least square method is implemented to derive the arc-length NURBS that fits to all these data points.

Second, by using the segmentation algorithm given in Section 4.2.2, three curve segments are isolated from the original curve as shown in Figure (5.2) and assigned with new valid feed rates. These segments are corresponding to the curvature profile that exceeds the threshold value in Figure (5.3).

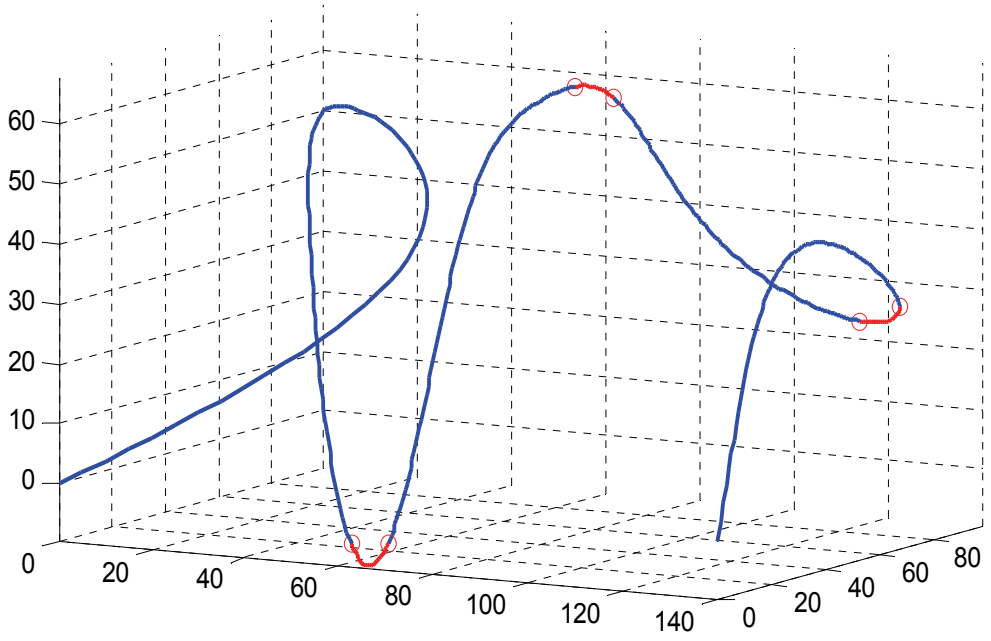


Figure 5.2 Recognition of the curve segments that cannot be machined with the prescribed feed rate

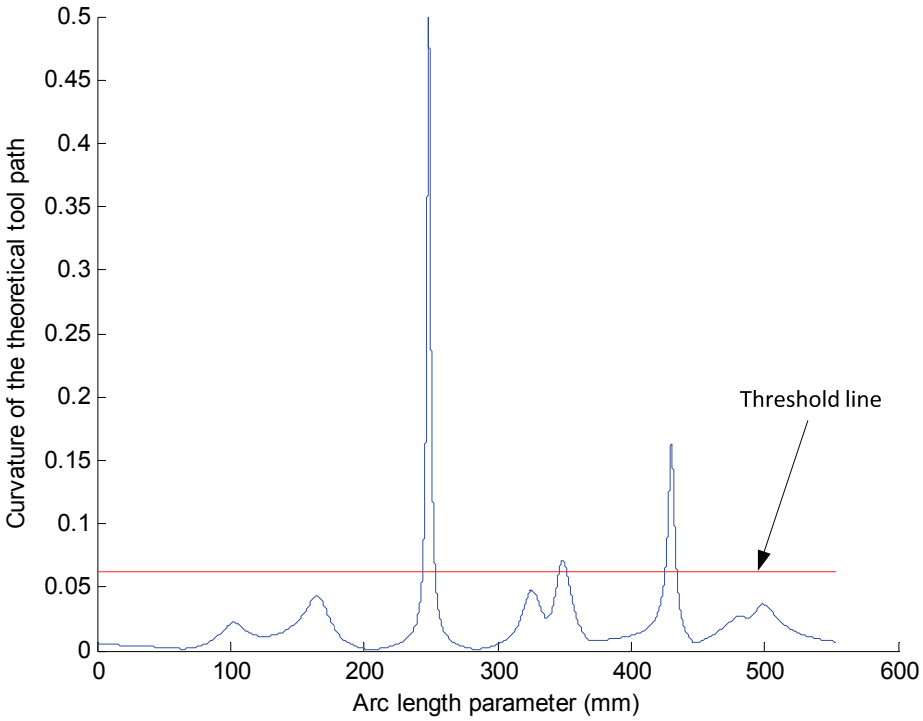


Figure 5.3 Curvature profile of the theoretical tool path with the threshold line

Take the first segment for example, its corresponding parameter varies from 0 to 243.7806, and the expected feed rate over this sub-interval is at 400 mm/s. By using the method introduced in Section 4.3.1, the acceleration profile of each axis can be found as Figure (5.4).

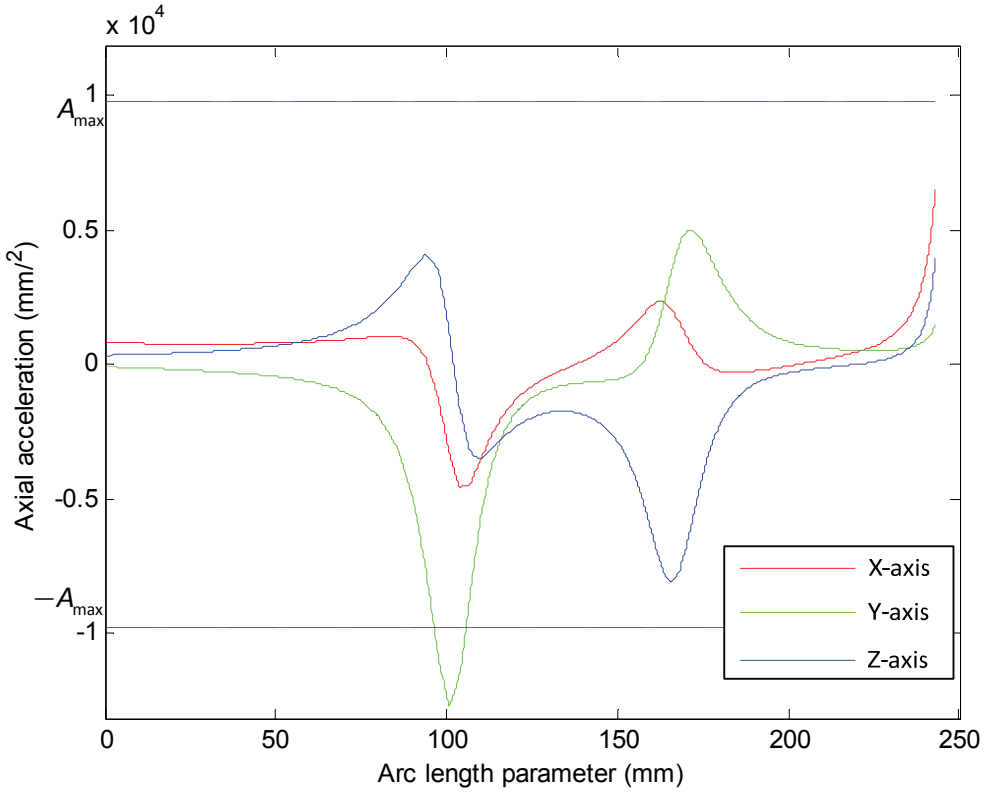


Figure 5.4 Acceleration profile of each axis at the current invalid feed rate

Apparently, the axial acceleration on Y-axis would exceed the machine axial acceleration limit if the tool were moving at its prescribed feed rate. In another word, the machine could not provide sufficient centripetal acceleration to maintain the tool trajectory error within the machining accuracy under such a high feed rate. Therefore, the feed rate over this curve segment has now been re-adjusted according to the

method introduced in Section 4.3.2, which is shown in Figure (5.5).

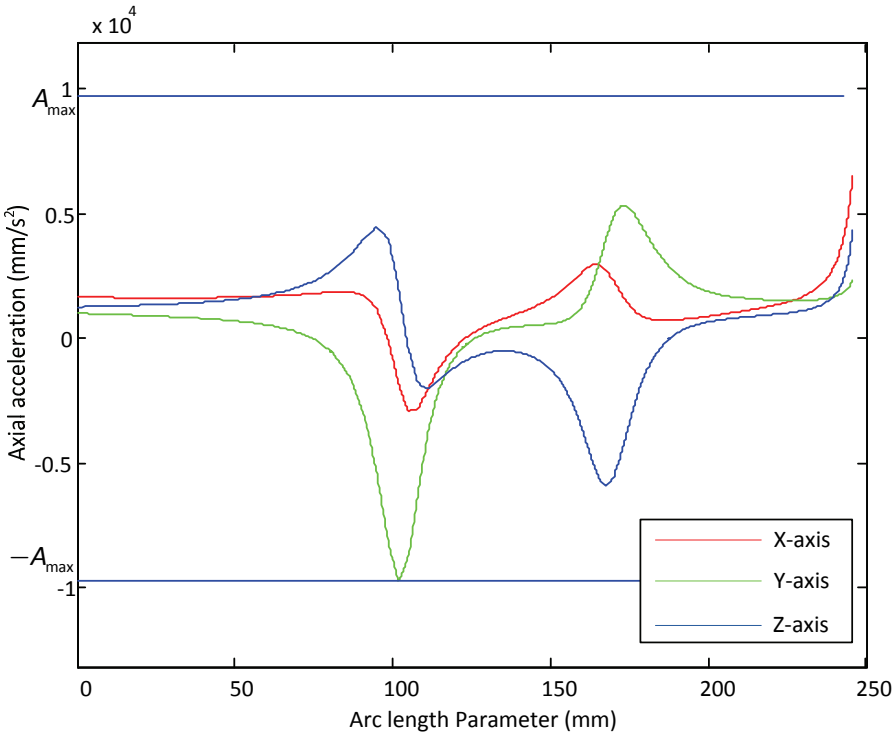


Figure 5.5 Acceleration profile of each axis after implementing the feed rate re-adjustment method

Similarly, by carrying out this procedure over all the parametric sub-intervals, the valid feed rate for each segments are listed in Table (5.3)

Table 5.3 Appropriate Feed rates over each curve segments after the adjustment

Arc-length Parametric intervals	Corresponding feed rates
$[0, \quad 243.7806)$	351 mm/s
$[243.7806, \quad 253.2825)$	141 mm/s
$[253.2825, \quad 346.1469)$	396 mm/s
$[346.1469, \quad 351.9481)$	372 mm/s
$[351.9481, \quad 425.5877)$	400 mm/s
$[425.5877, \quad 434.2540)$	183 mm/s

$[434.2540, \quad 553.6312]$	345 mm/s

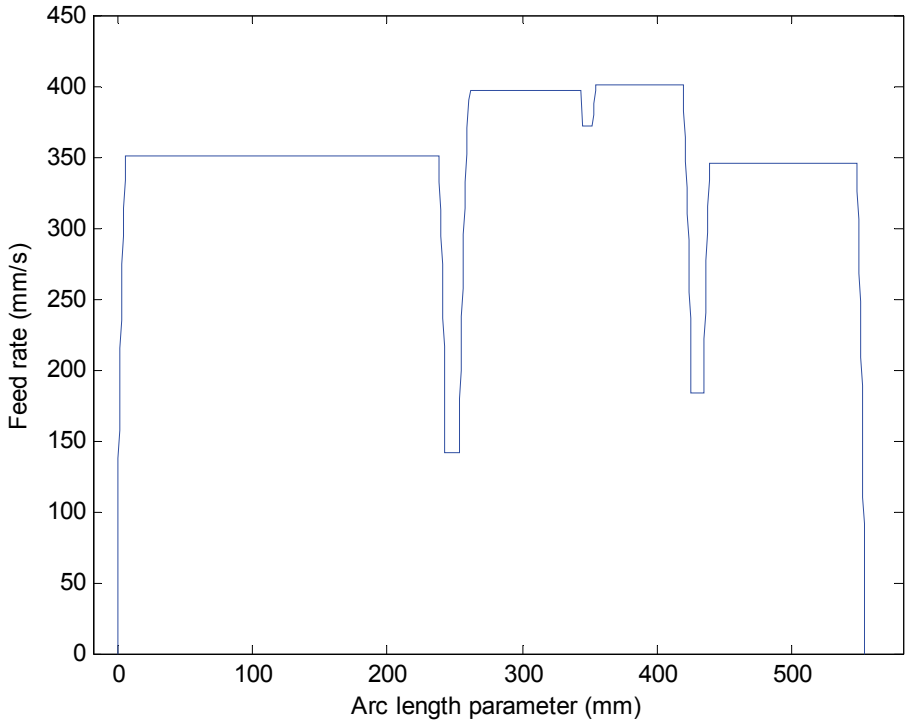


Figure 5.6 Actual feed rate profile during the machining process using the arc-length NURBS interpolation

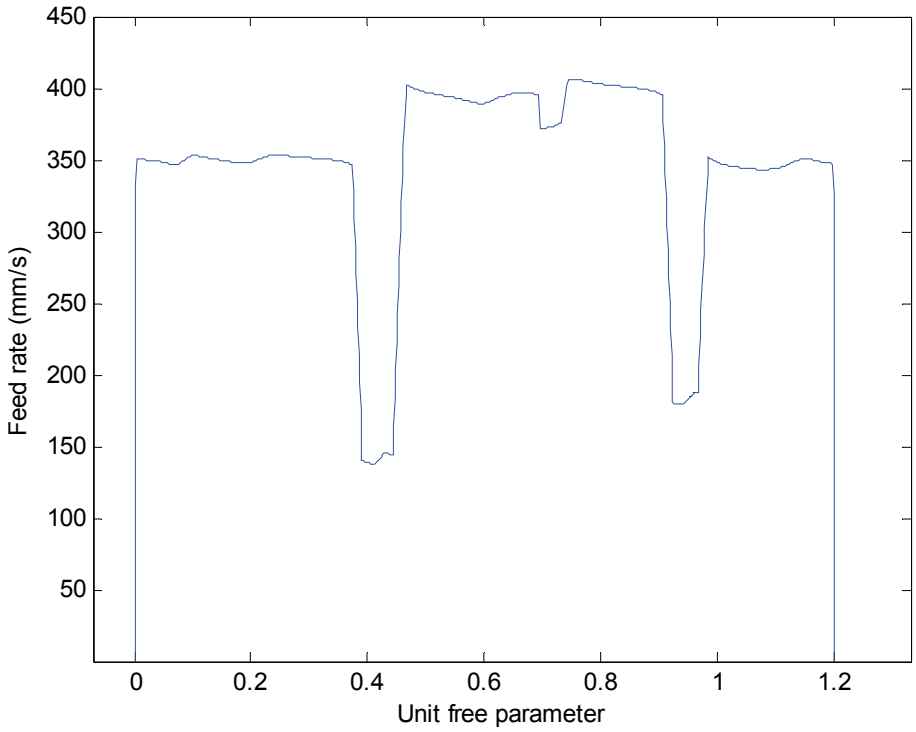


Figure 5.7 Actual feed rate profile using the traditional NURBS interpolation

Finally, after all the cutter location points were generated using arc-length NURBS interpolation iteration function of Eq. (4.39), the actual feed rate during the machining is shown in Figure (5.6).

As shown in Figure (5.7), when using the traditional NURBS interpolation method, the actual feed rate fluctuates as anticipated due to the inaccurate approximation of the iteration function of Eq. (2.5) with the unit free parameter. Although this kind of feed rate fluctuation can somehow be compensated with some method that had been developed, such as the one that introduced in 'NURBS Interpolation Method with Feedrate Correction in 3-axis CNC System' by Chen and Li, the computational time during the real-time machining could be significantly increased. By using their compensation method, the feed rate fluctuation would decrease within a prescribed tolerance *e* by implementing an additional search procedure to find the next parameter more precisely. Indubitably, The following Table (5.4) shows the computational time of different interpolation method during the actual machining of this particular example.

Table 5.4 Comparison on Computational time using different interpolation method

Interpolation method	Total time consumption during computation
Traditional NURBS interpolation with feed rate compensation tolerance at *e*=1 (mm/s)	6.03 s
Traditional NURBS interpolation with feed rate compensation tolerance at *e*=0.5 (mm/s)	8.15 s

Arc-length NURBS interpolation	1.32 s

As Table (5.4) shows, the calculation time is almost 5 to 8 times longer than the arc-length NURBS interpolation method when the traditional methods try to ease this kind of fluctuation. However, the machine can only execute its next movement after the sequential cutter location had been calculated. Therefore, the increased calculation time makes it more likely to cause unexpected halts, which are another source of fluctuation, if the machine has finished its current movement but waits on the next instruction being generated.

Since either the feed rate fluctuation or axial acceleration exceedance during the machining could cause intolerable trajectory errors, the machining quality would be degraded if such situation occurs when using the traditional method. However, with the proposed method in this work, these problems are solved. The tool trajectory error along the whole tool path is shown in Figure (5.8) and the bounded axial acceleration profile of each axis is shown in Figure (5.9).

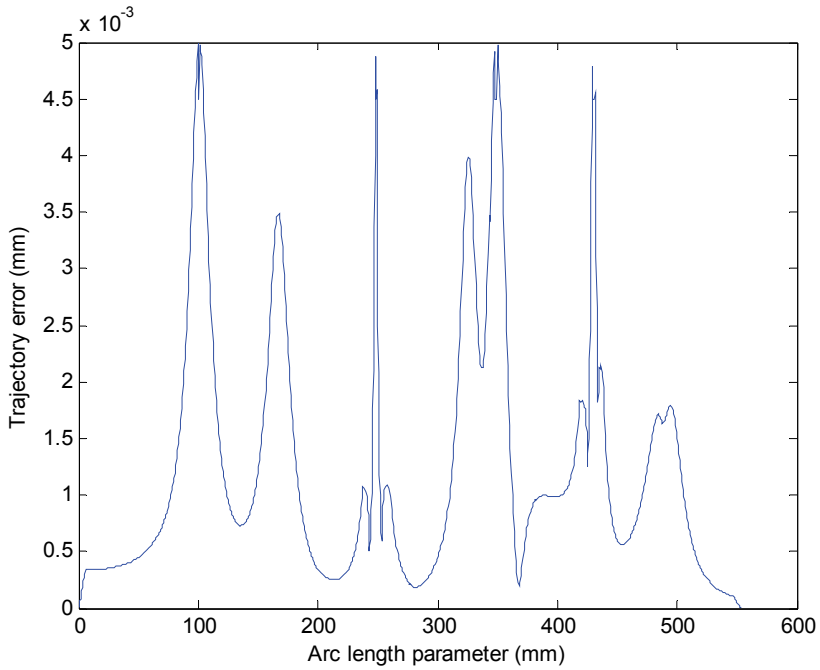


Figure 5.8 Trajectory error curve of proposed method

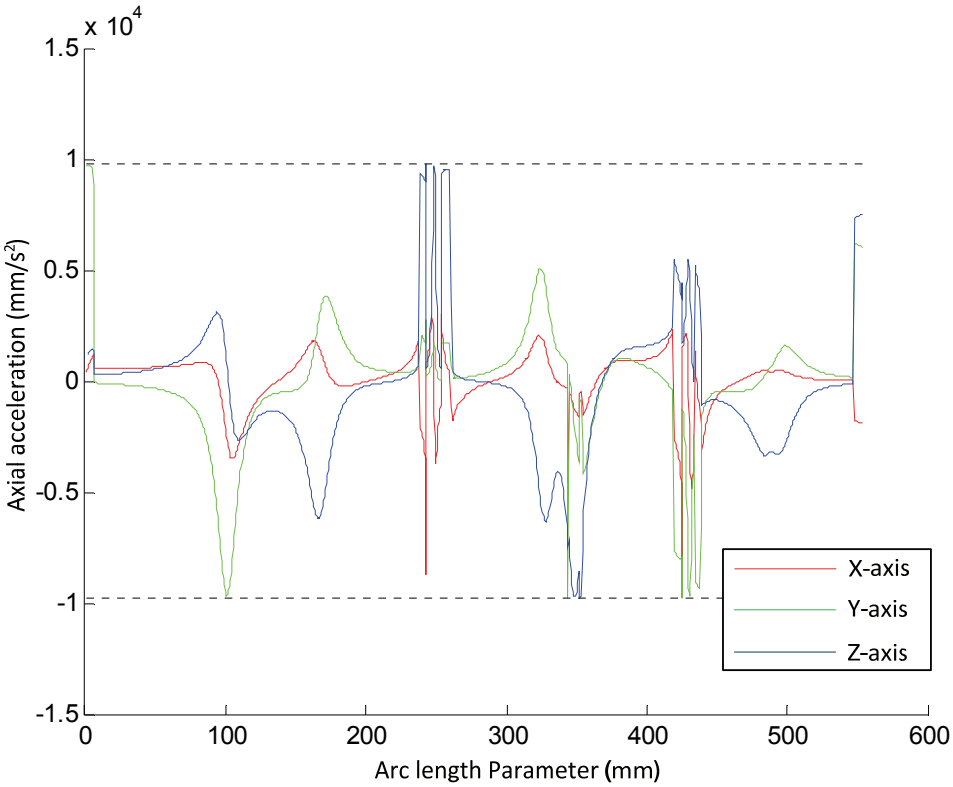


Figure 5.9 The limit-bounded axial acceleration profile using the proposed arc-length NURBS tool path generation method

5.3 Summery

According to the results given in the above example, it is reasonable to believe that this proposed arc-length NURBS tool path generation method has better performance than the traditional non-arc-length NURBS method. Not only because the axial acceleration is globally bounded to their limits, so that the machine can actually finish the assigned task without encountering unpredictable problems; but also a potential tool trajectory error source that comes with the traditional NURBS tool path generation method is eliminated, due to the feed rate fluctuation caused by inaccurate approximations from the traditional iteration function has been avoided. Moreover, the real-time cutter location calculation procedure during each machine sampling period is faster than the traditional one, which means, comparing to the traditional method, the requirement of the minimum central processor's calculation speed is somehow lowered when trying to conduct a same machining task and achieve a similar performance.

Chapter 6 Conclusions and Future work

6.1 Conclusions

In this work, an accurate arc-length calculation algorithm is introduced, followed by its implementation in a new parameterized NURBS tool path generation method. Instead of using the unit free parameter, arc-length parameterized source data are fed into the CNC controller that with the proposed arc-length NURBS interpolation algorithm, to generate the machining tool path in real-time process. Comparing to the traditional NURBS tool path generation method, the major improvements and contributions of this thesis can be summed up in the following aspects:

- An advanced approach to NURBS arc-length calculation is developed. In which, both the calculation efficiency and accuracy has been improved in addition to the elimination of the potential problems that may cause mistaken results.
- A piecewise feed rate adjustment strategy has been implemented in this research. All the corresponding feed rates over each parametric sub-interval are checked for feasibilities. This ensures the machine is fully capable for accomplishing the task with a valid feed rate profile. The axial acceleration limit and the tool trajectory error are globally bounded to their tolerances according to the machine properties and

the curvature of the theoretical tool path.

- Since the traditional theoretical NURBS tool path has been replaced by the arc-length NURBS, the feed rate fluctuation caused by the inaccurate approximation on finding tool locations from the traditional NURBS interpolation method has been avoided. And thus, these resultant unpredictable tool trajectory errors will no longer exist when using the proposed method.
- The calculation time for finding the next tool location during real-time machining is shorter than the traditional method. The difference between the time consumption of these two methods turns out to be more and more significant when the traditional method tries to compensate the aforementioned feed rate fluctuation with other linear search algorithms.

6.2 Future work

For future research, the following topics can be considered to expand the present work：

- During acceleration or deceleration procedure along the curve, another factor ‘jerk’ can be taken into consideration.
- Develop a jerk bounded method to provide better machine kinematics and better machining quality.

- Take tool angel into consideration and try developing an arc-length parameterized NURBS tool path generation method for 5-axis machining.
- Try to improve and reorganize the feed rate adjustment method to be more flexible, and then develop a pre-machining feed rate correction module based on the arc-length NURBS machining.

Chapter 7 References

[1] Piegl L., Tiller W., The NURBS Book, second edition. *Springer*.

[2] Lee K., Principles of CAD/CAM/CAE systems. *Seoul National University*.

[3] Mathews J.H., Fink K.D. Numerical Methods Using Matlab, fourth edition. *Prentice-Hall Inc*.

[4] Gander W., Gautschi W., Adaptive Quadrature–Revisited. *BIT Numerical Mathematics*, 40(2000), pp. 84-101.

[5] Feng J.C., Li Y.H., Wang Y.H., Chen M., Design of a real-time adaptive NURBS interpolator with axis acceleration limit. *Springer*.

[6] Chen L.J., Li H.Y., NURBS Interpolation Method with Feedrate Correction in 3-axis CNC System. *2009 International Conference on Computer Engineering and Technology*.

[7] Heng M., Erkorkmaz K., Design of a NURBS interpolator with minimal feed fluctuation and continuous feed modulation capability. *International Journal of Machine Tools & Manufacture*, 50(2010), pp.281-293.

[8] Zhang X.H., Yu D., Hu Y., Hong H.T., Sun W.T., Development of a NURBS curve interpolator with look-ached control and feedrate filtering for CNC system. *IEEE Industrial Electronics and Applications*, ICIEA 2009, pp. 2755-2759.

[9] Emami M.M., Arezoo B., A look-ahead command generator with control over trajectory and chord error for NURBS curve with unknown arc length. *Computer-Aided Design*, 42(2010), pp.625-632.

[10] Lee A.C., Lin M.T., Pan Y.R., Lin W.Y., The feedrate scheduling of NURBS interpolator for CNC machine tools. *Computer-Aided Design*, 43(2011), pp. 612-628.

[11] Wang X., Wang J.W., Rao Z., An adaptive parametric interpolator for trajectory planning. *Advances in Engineering Software*, 41(2010), pp. 180-187.

[12] Annoni M., Bardine A., Campanelli S., Foglia P., Prete C.A., A real-time configurable NURBS interpolator with bounded acceleration, jerk and chord error. *Computer-Aided Design*, 44(2012), pp. 509-521.

[13] Ng Y.K., A new velocity profile generation for high efficiency CNC machining application. *City University of Hong Kong*.

[14] Yeh S.S., Hsu P.L., Adaptive feed-rate interpolation for parametric curves with a confined chord error. *Computer-Aided Design*, 34(2002), pp. 229-237.

[15] Lei W.T., Sung M.P., Lin L.Y., Huang J.J. Fast real-time NURBS path interpolation for CNC machine tools. *International Journal of Machine Tools and Manufacture*, 47(2007), pp. 1530-1541.

[16] Yong T., Narayanaswami R., A parametric interpolator with confined chord errors, acceleration and deceleration for NC machining. *Computer-Aided Design*, 35(2003), pp. 1249-1259.

[17] Yau H.T., Lin M.T., Tsai M.S., Real-time NURBS interpolation using FPGA for high speed motion control. *Computer-Aided Design*, (38)2006, pp. 1123-1133.

[18] X. Zhiming, C. Jincheng, F. Zhengjin, Performance evaluation of a real-time interpolation algorithm for NURBS curves. *International Journal of Advanced*

Manufacturing Technology, (20)2002, pp.270-276.

[19] Luan Z.T., Jiang G.D., Bai B., Zhang Y., Mei X.S., A feedrate pre-schedule NURBS interpolation method for high-speed machining. *2010 8th IEEE international Conference on Control and Automation*, pp.1672-1677.

[20] Wang H.L., Kearney J., Atkinson K., Arc-length Parameterized Spline Curves for Real-Time Simulation. *Curve and Surface Design*, 2002, pp 387-396.

[21] Khan M.A., Piecewise arc-length parameterized NURBS tool paths generation for 3-axis CNC machining of Accurate, Smooth Sculptured Surfaces. *Concordia University*, Canada, 2010.