REAL TIME TRAJECTORY GENERATION AND INTERPOLATION

by

Anthony Siu

B.A.Sc. University of British Columbia, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(MECHANICAL ENGINEERING)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

JULY 2011

Abstract

This thesis presents a continuous tool motion trajectory generation algorithm for high speed free form surface machining. A NURBS toolpath generation algorithm is presented to fit the discrete motion commands generated from free-form CAD-models. By using a NURBS representation of the machine part, the toolpath is interpolated continuously to direct the synchronized motion of the 5-axis CNC machine. The higher continuity of the motion trajectory allowed for tighter machining tolerances and reduced feedrate fluctuations and the undesired acceleration harmonics in the overall feed motion and in each of the motor motions. An optimal and feasible feedrate profile have been used to continuously maneuver the cutting tool with the interpolated reference tool position and tool orientation commands such that the kinematic constraints of the drives are not violated.

Commonly used least squares curve fitting of discrete data points forces the curve to weave through the data points and results in a fluctuating toolpath. By making use of the defined basis function distributions of the NURBS control points, a higher smoothness fit has been achieved through a minimization on the chord error and the third derivative of the curve. The feasibility of this toolpath generation algorithm has been extended using the double spline representation to represent both the tool position and the tool orientation with minimal fitting error.

The real time interpolation of the fitted NURBS toolpath has also been implemented using the multi-segment Feed Correction Polynomial. This method provides an adaptive mapping between the nonlinear relationship of the NURBS curve parameter and the curve displacement to allow for a consistent feedrate in the cutting motion. Additionally, the kinematic compatibility conditions are considered based on the inverse kinematics of the 5-axis CNC machine. The proposed algorithm ensures that an overall efficient feed constraint is placed such that none of the individual drives are overdriven. The results from experiments and simulations are presented to demonstrate the effectiveness of the developed trajectory generation algorithms.

Table of Contents

List of Tables

List of Figures

Nomenclature

A	Acceleration
α	Objective Function coefficient for Jerk Minimization
$\alpha_0, \cdots, \alpha_7$	Feed Correction Polynomial Coefficients
B_{eq}	Mechanical Plant Equivalent Damping
β	Objective Function coefficient for Chord Error
γ	Objective Function coefficient for Data Point Error
$C(u)$	NURBS Lower-Spline Toolpath Equation
$C^{up}(u)$	NURBS Upper-Spline Toolpath Equation
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CNC	Computer Numerical Control
D	Projection Matrix
ER	Maximum Chord Error for Feedrate Limitation
e_{max}	Maximum fitting error limit at Data Points
e^t	Tool tip Position Error between Data Point and Corresponding Spline Location
e_θ	Orientation Error
ε_{MSE}	Mean Square Tolerance in Feed Correction Polynomial Fit
ε_S	Tolerance in Quadrature Displacement Calculation
f	Feed
H	Homogeneous Transformation matrix
I	Identity Matrix
J	Jerk
J_{eq}	Mechanical Plant Equivalent Inertia
J_K	Objective Function for 3rd derivative minimization

J_L	Objective Function for fit error minimization at Data Points
J_Q	Objective Function for Chord Error minimization
K	Basis Function 3rd derivative Integration Matrix
$K(u)$	Curvature of a curve
K_N	Basis Function Integration Matrix
k	Discrete kth time step
L	Constraint Matrix
L_{Back}	Required Extra length for desired change in feedrate
L_{min}	Minimum Required length for desired change in feedrate
Λ	Lagrange Multiplier
MAL	Manufacturing Automation Laboratory
MSE	Mean Square Error (Variance)
M	Last index of Tool Path Data point data where M + 1 = # of data points
m	Last index in a Knot vector where m + 1 = # of knots
NURBS	Non-Uniform Rational B-Spline
$N_{i,p}$	B-Spline Basis Function: i is the corresponding control point and p is the degree
n	Last index of the B-Spline Control Points where n + 1 = # of Control Points
$O(u)$	Tool Orientation
P_i	Control Point of B-Spline
P_i^{up}	Control Point of Upper B-Spline
p	Degree of B-Spline
Φ	B-Spline Basis Function Matrix
$Q(u)$	Motor Drive Positions in terms of NURBS toolpath curve parameter
$Q(s)$	Motor Drive Positions in terms of Tangential Displacement
Q_s^v	Normalized Motor Drive velocity with maximum drive velocity

Q_s^a, Q_{ss}^a	Normalized Motor Drive velocity, acceleration with maximum drive acceleration
$Q_s^j, Q_{ss}^j, Q_{sss}^j$	Normalized Motor Drive velocity, acceleration, jerk with maximum drive jerk
Q_e	Chord Error Integration
q^O	Upper Spline Data Points
q^t	Tool Tip Location Data Points
$\hat{q}^t$	Tool Tip Location Estimate
Rot	Rotation Matrix
$R_{i,p}$	Rational B-Spline Basis Function (weights included)
$\rho(u)$	Local Curve Radius of Curvature
S	Total Displacement
S_c	Total Chord length formed by Data Points
s	Arc Displacement Parameter
$\hat{s}$	Arc Displacement from Numerical Calculation
s_i	Arc Displacement Data from Numerical Integration
σ_i	Normalized Displacement parameter for Feed Correction Polynomial
TA	Normalized Tool Axis Orientation Vector
t_i	Knot Vector Node
$Transl$	Translation Homogeneous Matrix
T_s	Sampling Time
U	B-Spline Knot Vector
u	Curve Parameter
$\hat{u}$	Curve Parameter estimate using Feed Correction Polynomial
$\bar{u}$	Curve Parameter Estimates for corresponding Data Points
u^*	Knot vector node to be inserted
$V = V(t)$	Tangential Feedrate

V_{max}	Maximum Tangential Feedrate
$w(u)$	NURBS weight function
w_i	Weights of the B-Spline
X, Y, Z	Cartesian Motor Drive Positions
θ_a, θ_c	Rotary Motor Drive Positions
x^t, y^t, z^t	Toolpath Position w.r.t. workpiece
ξ	Constraints Vector

Acknowledgements

The time spent at Manufacturing Automation Laboratory has been an invaluable experience to me. I would like to express my sincere gratitude to my research supervisor, Dr. Yusuf Altintas for his guidance and support. He has shown time and time again that having a good understanding of the underlying picture, and an appreciation for good communication can really help in steering away from frustration and focusing on the matters at hand. His instructions were invaluable throughout my studies and I really appreciate it.

I wish to thank my colleagues for sharing their knowledge, experience and problems with me. I feel lucky to have had a chance to witness the different personalities, and the different attitudes within the lab. It was always thrilling to talk about research to have the discussion drag on to the more philosophical issues around life. I was especially lucky to have an exceptional friend, Ken, who at many times gave me nothing but trouble and frustrations, but also whom I can always show my immature self and was always able to put me at ease. I would also like to acknowledge Dr. Nagamune who gave me invaluable feedbacks and for his kind answers in responding to question that I posed.

This research was sponsored by the Natural Science and Engineering Research Council of Canadian Network for Research and Innovation in Machining Technology (NSERC CANRIMT).

Chapter 1 Introduction

The manufacturing industry today has many interests and applications in precision free-form machined parts, and the demand for high-speed machining (HSM) is ever increasing. Die and mold designs for small-scaled commercial parts, the stamping dies of an automotive body panel, as well as propeller blade designs are typical examples of free-form contour machining. The machining of complex dies, molds, aerospace, automotive and biomedical parts have become a major research focus in the past decades because of the advances seen within the Computer Aided Design (CAD) community to apply smooth spline representations of complex free-form surfaces. Non-uniform rational B-splines (NURBS) are extensively used in CAD today because of their flexibility and precision in handling complex geometries; providing an exact and uniform representation of parametric curves that easily allows for analytical and numerical manipulations. In addition, the wide spread use of robust and high performance five-axis computer-numerically controlled (CNC) machines has allowed the manufacturing industry to take advantage of machining these complex sculptured free-form surfaces. The orientation positioning provided by the additional two axes over traditional three axes CNC machines provide the optimal posture and orientations that are adjusted in real time in response to the varying surface curvatures.

The implementation of NURBS interpolation techniques to high performance CNC machines have been a constant focus. Conventional linearized toolpath trajectories composing of numerous straight line segments cannot provide the smooth surface finish that is desired and forces the machine to decelerate and stop at the joint segments. This increases machining time tremendously and denies accurate tool positioning. In addition, for machines with rotary axes that support significant masses during operation, this imposes severe limitations on the motors' acceleration and torque limits. Furthermore, these abrupt accelerations and decelerations of the tool motion can result in contour error where tangent and curvature continuities are not met at the joint segments. They may also excite the feed drives or the mechanical structures and cause undesired vibrations at the tool tip that cannot be properly tracked with the limited performance of the CNC controllers. By introducing the higher order parametric NURBS scheme into the CNC system, these problems can be reduced and better feedrates, higher accuracies and surface qualities can be achieved.

The major advantage of NURBS interpolation techniques is the overall smoothness and flexibility of the toolpath. In order to acquire a NURBS toolpath from discretized machine code or toolpath data points that are attained from CAD software, there are numerous fitting algorithms. There are local fitting algorithms where the continuity between individual curve segments is maintained, and there are global methods of using least squares fittings to minimize the error between the NURBS toolpath and the data points. Geometrical and smoothness factors are also considered during the fit.

The main problem with using NURBS interpolation techniques is that, unlike traditional linear or circular interpolation techniques where the length of the segment is known analytically, the length of the NURBS toolpath is not. When the curve is interpolated with constant curve parameter increments, the tangential tool travel displacements are not constant and undesired feed fluctuations can occur during real time interpolation. Feed fluctuations can bring about high accelerations and jerks to the motors, leading to the possibilities of feed drive saturations or excitations, thus leaving surface deflects on the part and reducing the overall surface quality. Large contour errors can also result and violate the part tolerances. The methods of Taylor expansion and the Feed Correction Polynomial Interpolation methods are implemented and their effects on feed fluctuations are compared in this thesis.

Feedrate profile scheduling is important in the reduction of cycle time. While having high feedrates will inevitably reduce machining time, considerations on drive constraints as well as part geometry can allow for higher tracking performance and increased overall productivity. Drive saturations severely limit the tracking performance of the CNC system. A jerk limited trajectory profile is used to ensure the smoothness of the trajectory profile and a simple method is used to maximize the feedrate to meet the kinematic constraints of the drives and the geometrical constraints imposed on the feedrate.

The X-Y Table is a common control platform in CNC machines used to direct a work piece during machining operations. X-Y Tables are built and configured to provide high-performance positioning along multiple axes. They are commonly used for moving a large work piece which requires a greater machining area. Compared to the moving spindle mechanism in CNC machining, the moving X-Y table has a greater tolerance to vibrations with its greater system inertia. An industrial X-Y table is implemented and its performance with various controllers is analyzed.

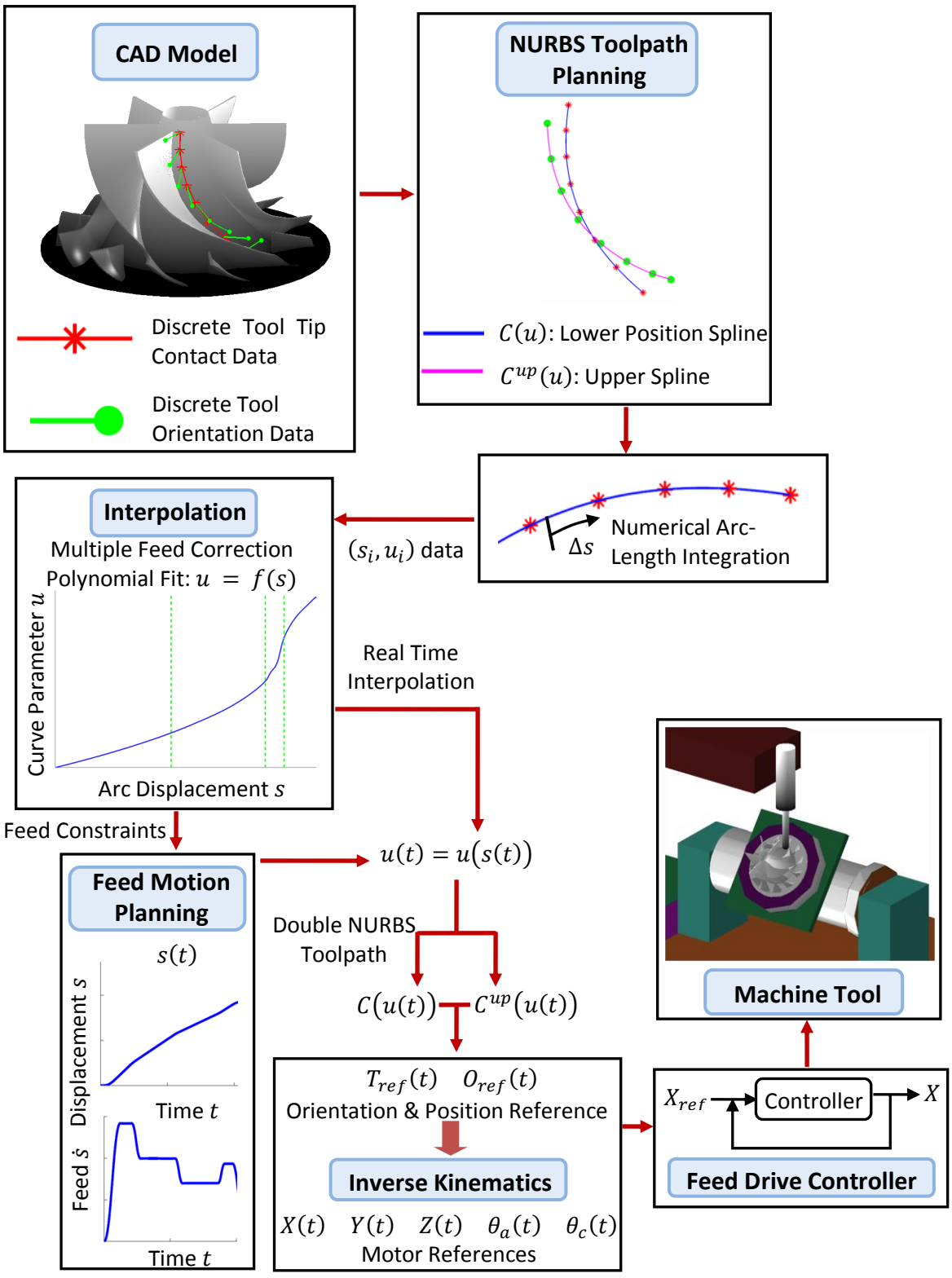


Figure 1.1: Trajectory Generation Overview

The overview of the trajectory generation process is illustrated in Figure 1.1. The review of related literatures is presented in Chapter 2. A smooth NURBS based five-axis toolpath generator is presented in Chapter 3 to represent the discretized CAD model data. A numerical integration method is shown to accurately integrate the displacement arc-length along the toolpath. The real time interpolation methods for the NURBS toolpath and the feedrate scheduling algorithms are explored in Chapter 4. The implementation of an industrial X-Y table and the feed drive controller designs are presented in Chapter 5, followed by the conclusion and a discussion of some possible future research directions in Chapter 6.

Chapter 2 Literature Review

2.1 Introduction/Overview

Toolpath planning and reference command generation play significant roles in the automation of robotics and CNC machine tools. Toolpath planning has to maintain high conformity and ensure the smoothness and geometrical tolerance to the CAD model of the part. In order to properly interpolate the more complex toolpaths of free-form surfaces, more robust and advanced interpolation techniques are required for the tool path. These interpolation techniques have to be capable of being processed in real time as well as complement the feedrate profiles. The feedrate profiles are planned to meet the constraints imposed by the motor limits and the CNC dynamics while maximizing the feedrate to minimize cycle time.

This chapter presents a review of the relevant literature involved with NURBS toolpath generation, interpolation methods for the NURBS toolpath, followed by a review of feedrate optimization.

2.2 Tool Path Fitting for Free-form Machining

The use of splines and other parametric curve representations for complex parts with smooth surfaces in high speed machining operations have not been hugely popular two decades back. However, incentives of processing and transferring large amounts of geometrical information from the much more advanced CAD software models to the CNC system, as well as the increased demands for higher precision parts within the mold and die, and especially in the aerospace industry, advocated the push for using smoother cubic, quintic, Bezier and B-spline curve representations. To represent a free-form surface with linear segments, a large amount of geometric data points are required which leads to significant reduction in curve accuracy. It has been demonstrated that linear interpolation methods have the following limitations: Velocity discontinuities at the linear joint segments, increased machining time from the deceleration of each segment, poor surface finish, loss of geometric information and lower part accuracies. Rather than using conventional linear and circular segments of representing toolpaths, the fitting of splines and the use of direct spline trajectory generation methods have been demonstrated by researchers such as Koren and Lo [27] that the much smoother and continuous motions can reduce these problems.

Earlier works in the 1970s by Pierre Bezier has popularized the use of the Bezier curve in industrial applications and since then, Bezier curves have been widely used in CAD for the modeling of smooth curves. The B-spline curves are a generalization of the Bezier curves, with the use a knot vector and a set of recursive blending functions to allow for local curve modification and for the degree of the curve and the number of control points to be selected independently. The non-uniform rational B-spline (NURBS) curve is a further generalization of the B-spline curve, allowing a non-uniform knot vector and weighted control points. For many computer applications, the use of B-spline representations have been used much earlier in computer vision to smooth out corners in image processing [44] and to give accurate approximations of the original object. Significant data compressions were achieved for using these B-splines representations. Bahr et al. [5] applied a real time cubic parametric curve interpolation scheme to a CNC system and showed that there were low interpolation errors and small variations in jerk. Piegl and Tiller [48] were able to approximate the NURBS curve by using a bi-arc formulation which allowed CNC machines to make use of existing circular interpolations to represent NURBS curves. Erkorkmaz and Altintas [19] proposed the Optimally Arc Length Parameterized quintic splines which optimized the discrepancies between the spline chord length parameter and the travelled arc length in quintic splines. Details involving global and local approximations and exact interpolation fitting algorithms for NURBS curve and surface can be found in Piegl and Tiller [47]. In this thesis, the least squares global approximation method will be implemented.

2.3 Interpolation

At every sampling step of the CNC controller, the spline toolpath is interpolated in real time. This provides the reference command for the position tracking loop of the CNC servo system. The desired tangential displacement of the tool tip at the current feed step is interpolated with the spline toolpath to retrieve the curve parameter necessary to acquire the position reference. One of the main motivations of using parametric interpolation is to reduce the amount of geometric information transferred between the CAD and the CNC system [55]. Instead of the CNC machine processing the large amount of broken down linear line segments with the typical CNC linear interpolator, Shpitalni and Koren [55] have shown that for smooth curves where the first derivatives exist, the first order Taylor expansion has been shown to handle both implicit curves as well as curves in parametric forms and that the first order

approximation is adequate when sampling time is small and the curve maintains a high radius of curvature.

Greenway and Zhang [69] were the first researchers in applying the NURBS interpolation to a six-axis robot with the first order Taylor's expansion to prove the feasibility of the NURBS interpolator in real time motion. Yeh and Hsu [67] applied a compensatory parameter of reducing the drive speed to improve the curve accuracy for parametric curves. A constant curve speed was used together with acceleration and deceleration sections to interpolate a NURBS curve. Zhiming et al. [70] applied the second order Taylor's expansion for the NURBS interpolation with a variable feedrate. Using a dependence on the toolpath curvature, the chordal deviation error and the sudden changes to feedrate were minimized. Cheng et al. [7] compared different numerical algorithms for the NURBS motion command generator: 1^{st} order and 2^{nd} order Taylor's expansion, Runge-Kutta approximation and an iterative corrector-predictor method. While the Runge-Kutta can theoretically outperform the Taylor's Expansion methods in terms of geometrical accuracy, its long computation time makes it inadequate for real time computations. The corrector-predictor method avoids the computation of the curve derivatives, but instead uses the trend from the previous curve parameter values. It was found that unlike the direct approximation methods, the computation time of the corrector-predictor method was unstable and can vary hugely depending on the toolpath. Tsai et al. [60] later improved the structure of the corrector-predictor model; however, it still suffered from problems such as large feedrate errors where the curve exhibited large curvatures, similar to the Taylor's Expansion, and unstable computation times.

The problems associated with the Taylor's expansion are the long computation time involved in calculating the curve derivatives, and that variable feedrate methods face problems with the accumulation of numerical errors and round off errors. Erkorkmaz and Altintas [19] introduced the use of a feed correction polynomial. By using a set of numerically calculated arc-lengths at increments of the spline parameter, a 7^{th} order polynomial can be used to approximate and express the arc-length, curve parameter relationship between the increments. Lei, et al. [32] introduced a similar method using cubic Hermite splines and also incorporated the numerically-calculated arc-length increment to the Taylor's expansion to make it more robust. Heng and Erkorkmaz [22] continued to expand the method by joining multiple polynomials for better fitting results between curve parameter and arc-length.

2.4 Feed Optimization

In machining applications of molds, dies, aerospace and automotive parts, the minimization of cycle time, and having a continuous and smooth feedrate profile are extremely important factors for increasing productivity. The quality and part surface integrity has to be maintained despite the constant push for using higher feedrates. When a constant feed is specified by the process planner in a cutting operation, the CNC machine tries to match the tangential feed of the cutting tool by controlling the velocity, acceleration, and jerk profiles of all the drives. The trajectory profiles of the drives must maintain continuity in position and derivatives to enable better tracking performance and prevent the excitation of the machine. There are also constraints to avoid saturating the individual actuators. When the velocity, acceleration, and jerk of a drive exceeds its physical limitations, nonlinear effects from the saturation can result in instability, poor part surface quality, and potentially damaging the machine. While using conservative feedrates can prevent drive saturations and ensure machining tolerances, the cycle time of the machining operation may increase significantly. The challenge is to minimize cycle time by maximizing feedrate while considering the constraints imposed on the actuators.

One of the main focuses of achieving smooth motion is to limit the jerk of the actuators and the feed motion. Constantinescu and Croft [9] determined the feasibility regions of solving for the velocity, acceleration, and torque rates in a time-optimal problem for robotic manipulators. Erkorkmaz and Altintas [14] presented a jerk-limited trajectory profile by bounding jerk in a piecewise manner. Later Erkorkmaz [18] introduced a cubic jerk trajectory profile that provided continuity between the piecewise jerk bounds. Macfarlane and Croft [9] applied sine waves as the ramps of the trajectory profiles to maintain high continuity and smoothness at the transients. Optimization techniques were later developed to solve for an arbitrary displacement profile that would result in minimum jerk and cycle time. Erkorkmaz and Altintas [19] introduced a quintic displacement profile to iteratively solve for the spine coefficients while being conservative to the drive constraints. This type of feed profile can be scheduled continuously and can be applied to a variety of applications. Rather than using commonly defined profiles, many researchers have also developed algorithms for adaptive feedrate modulations in order to achieve various part tolerances. Yeh et al. [67] has shown the geometrical dependence on feedrate where the curvatures of the toolpath are large and that

feedrate can be adapted to reduce feedrate fluctuations. Lin et al. [36] has shown that contour error can be considered when considering feedrate scheduling, however, an iterative approach, as well as a well known system dynamics are required for their method. Using a heuristic method of satisfying the constraints along the different segments of the toolpath, Heng and Erkorkmaz [22] presented a method to search for the possible feedrate by considering the feasible solution regions of the limited jerk feed profile, allowing feedrate to be iteratively solved and modulated. In this work, the limited jerk multi-segment feed profile is applied to modulate between segments of constant feed to meet the local feedrate constraints to keep the motors within their kinematic limits and the part to be within their geometrical tolerances.

Chapter 3 NURBS Toolpath Generation

3.1 Introduction

A reference toolpath is a key component in CNC machining. The generated toolpath has to maintain a high geometrical tolerance, as well as ensure a smooth profile that can be successfully tracked by the CNC machine. The kinematic profile of the toolpath is also an important consideration in minimizing machining time, and avoiding excitations of natural modes of the CNC machine tool structure.

Traditional CNC toolpaths have always consisted of linear and circular segments. However, to machine parts with complex free form surfaces, the CAD model has to be broken into a large amount of micro linear segments to approximate the original CAD model. The loss of important spline features brings about discontinuities in machining and large surface part errors. A large amount of geometrical information is consequently fed as G codes to the CNC machine to make use of their linear and circular interpolation algorithms to machine the part. This has huge effects on surface finish, part quality and machining time.

In order to overcome these challenges, toolpaths are constructed for cubic and quintic curves to provide continuous motion to the CNC machine as a replacement to linear segmentation. Smoother toolpaths with the cubic polynomial splines ensure C^2 continuity and allow for continuous acceleration profiles.

3.2 NURBS Background

The Non-Uniform Rational B-Spline (NURBS) curve has been widely used in CAD systems due to its many advantages:

1. Intuitive control points
2. Local curve shaping capabilities
3. Curve smoothness

The NURBS curve is a generalization of the B-spline curve where all the control points are weighted. A NURBS curve is illustrated in Figure 3.1. The set of control points, $P_i \in R^3$ and the associated weights, $w_i \in R^+$, form the shape of the curve, and the knot vector, $U = \{u_0, \dots, u_m\}$, determines the distribution and influence of the control points within the interval

of curve parameter $u \in [u_0 \quad u_m]$. As shown in Figure 3.1., increasing the weight of a control point pulls the curve towards the control point. Due to the property of local shaping, only the knot range of $u \in [u_i \quad u_{i+p+1})$ is affected, where p is the degree of the curve.

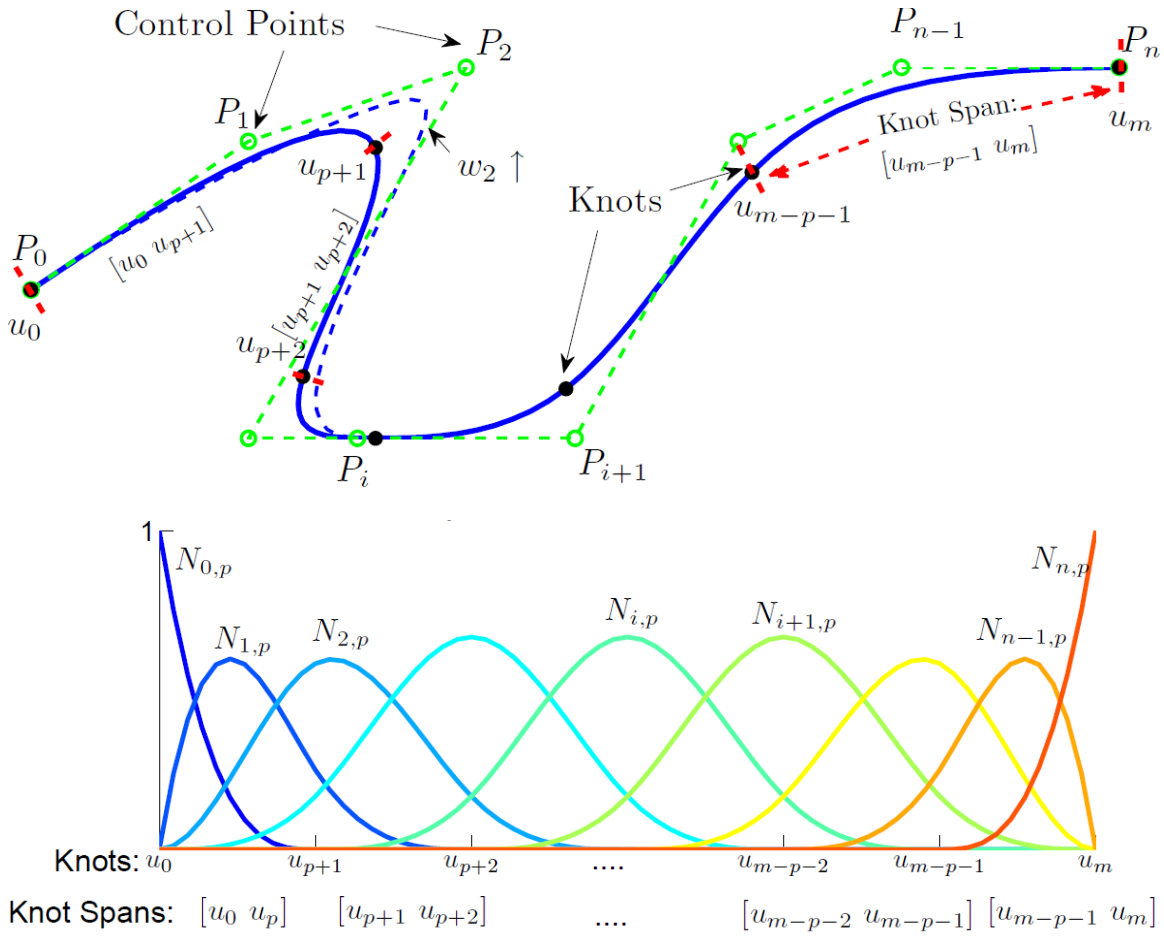


Figure 3.1: NURBS Curve Representation

a). NURBS Curve (Blue), Control Points (Green), Knot Spans (Red)

b). B-Spline Basis Functions

A NURBS curve is defined as:

$$[x(u) \quad y(u) \quad z(u)] = C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) P_i w_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \quad u \in [u_0 \quad u_m] \qquad (3.1)$$

The NURBS curve can also be written as:

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u)\, P_i \tag{3.2}$$

where $R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{i=0}^{n} N_{i,p}(u)w_i}$ is known as the rational basis function. The basis function, $N_{i,p}(u)$, and the rational basis function, $R_{i,p}(u)$, are scalar functions of the curve parameter u.

Some useful properties taken from Piegl [47] regarding the NURBS curve are listed below:

- Nonnegativity: $R_{i,p}(u) \geq 0$ for all $i, p, and\ u\ \epsilon\ [u_0 \quad u_m]$
- Partition of unity: $\sum_{i=0}^{n} R_{i,p}(u) = 1$ for all $u\ \epsilon\ [u_0 \quad u_m]$
- $R_{0,p}(u_0) = R_{n,p}(u_m) = 1$
- For $p > 0, all\ R_{i,p}(u)$attain exactly one maximum on the interval $u\ \epsilon\ [u_0 \quad u_m]$
- Local Support: $R_{i,p}(u) = 0$ everywhere except $u\ \epsilon\ [u_i \quad u_{i+p+1})$
- $C(u)$ is infinitely differentiable on the interior of knot spans and is $p - k$ times differentiable at a knot of multiplicity k

The knot vector U defines the B-Spline basis functions, $N_{i,p}(u)$. The knot vector of NURBS is a set of non-decreasing sequence of real numbers and has the following form of being non-periodic and non-uniform:

$$u_0 = \cdots = u_p < u_{p+1} \leq \cdots \leq u_j \leq u_{j+1} \leq \cdots u_{m-p-1} < u_{m-p} = \cdots = u_m$$

$$U = \big[\underbrace{u_0, \ldots, u_p}_{\text{Repetitive p+1 Knots}}, u_{p+1}, \ldots u_j, \ldots, u_{m-p-1}, \underbrace{u_{m-p}, \ldots, u_m}_{\text{Repetitive p+1 Knots}}\big], \tag{3.3}$$

where $u_j \leq u_{j+1}, j = 0,1, \ldots, m$. Each element within the knot vector is called a *knot*. The knot vector is non-uniform because there are $p + 1$ repeated knots at the beginning and at the end of the knot vector. The purpose of the repeated knots is for the NURBS curve to pass through the first and last control points, P_0 and P_n, and define a curve segment. The internal knots are defined by u_j. Consecutive knots form the half-open interval $[u_j \quad u_{j+1})$, called the j^{th} *knot span*. The key advantage of the NURBS curve over the Bezier curve is that the knot vector allows the NURBS curve designer to define the degree of the curve and the number of control points separately. The number of control points is $n + 1$, and the degree of the curve is p; together

there are $n + p + 2$ or $m + 1$ knots within the knot vector of a curve segment. The relationship of the knot vector size of $m + 1$ knots and $n + 1$ control points is such that the contribution of every control point P_i is in the range of the knot interval $[u_i \quad u_{i+p+1}]$. Outside this range, the influence of the control point P_i is zero, which provides the NURBS curve the property of local shaping. A knot vector can also contain the knots of multiple curve segments. The following is an example of a knot vector for a piecewise cubic (p = 3) NURBS curve:

$$U = [0 \quad 0 \quad 0 \quad 0 \quad 0.2 \quad 0.6 \quad 1 \quad 1 \quad 1 \quad 1.5 \quad 2 \quad 2 \quad 2 \quad 2] \tag{3.4}$$

The two curve segments are defined in the intervals of $u \in [0 \quad 1]$ and $u \in [1 \quad 2]$. The curve is separated at $u = 1$, signified by the repeated p knots. There are 14 knots in the given knot vector, m = # of knots – 1 = 13, and there are $m - p$ = 10 control points, n = # of control points – 1 = 9. The piecewise knot vector can be decomposed into:

$$\left.\begin{aligned} U_1 &= [0 \quad 0 \quad 0 \quad 0 \quad 0.2 \quad 0.6 \quad 1 \quad 1 \quad 1 \quad 1] \\ U_2 &= [1 \quad 1 \quad 1 \quad 1 \quad 1.5 \quad 2 \quad 2 \quad 2 \quad 2] \end{aligned}\right\} \tag{3.5}$$

The first curve segment will have 6 control points and the second curve segment will have 5 control points where $P_5^1 = P_0^2$.

The knot vectors determine the influence of each of the B-Spline basis functions and give the NURBS curve the property of being a piecewise rational polynomial and allow local modifications to the curve.

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.7}$$

The B-Spline basis functions are recursive as indicated in (3.7) where the zero degree basis functions are step functions as shown in (3.6). Higher order, p degree, basis functions are linear combinations of two lower degree, $p - 1$, basis functions. A p degree basis function and its derivatives are equal to zero everywhere except on the half-open $u \in [u_i \quad u_{i+p+1})$.

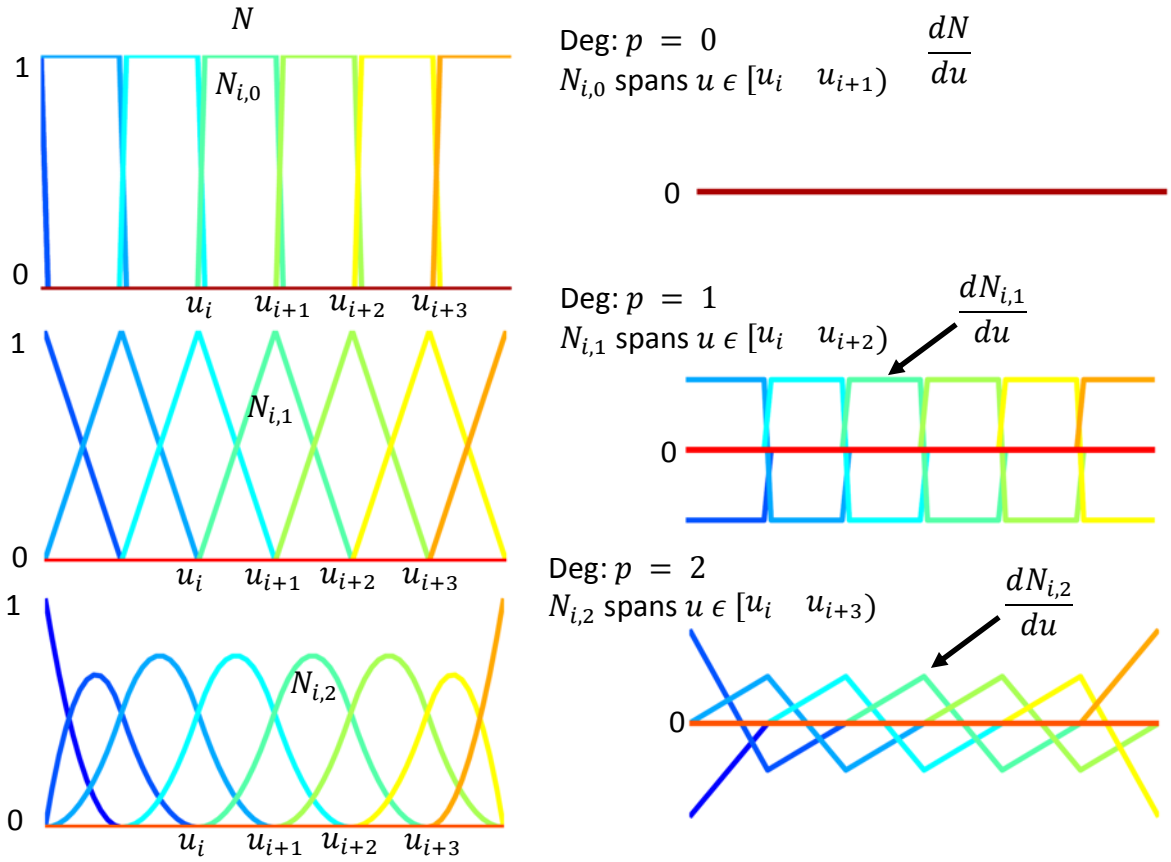


Figure 3.2: B-Spline Basis Functions and Derivatives

The derivatives of the B-spline basis functions are also calculated recursively. The first derivative of the p^{th} degree basis function with respect to the curve parameter u is given as:

$$\frac{dN_{i,p}(u)}{du} = \frac{1}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.8}$$

and the k^{th} derivative is given as:

$$\frac{d^{(k)}N_{i,p}(u)}{du^{(k)}} = p\left(\frac{N_{i,p-1}^{(k-1)}(u)}{u_{i+p} - u_i} - \frac{N_{i+1,p-1}^{(k-1)}(u)}{u_{i+p+1} - u_{i+1}}\right) \tag{3.9}$$

Figure 3.2 shows the lower order piecewise basis functions and their first derivatives. For NURBS curves where all the weights are equal, the derivative of the curve is simply:

$$\frac{d^{(k)}C(u)}{du^{(k)}} = \frac{d^{(k)}N_{0,p}(u)}{du^{(k)}} P_0 + \frac{d^{(k)}N_{1,p}(u)}{du^{(k)}} P_1 + \cdots + \frac{d^{(k)}N_{n,p}(u)}{du^{(k)}} P_n \tag{3.10}$$

When weights are not all equal and are being considered,

$$C(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \frac{w(u)C(u)}{w(u)} = \frac{A(u)}{w(u)} \tag{3.11}$$

where $A(u)$ and $w(u)$ are defined respectively as:

$$A(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i w_i \tag{3.12}$$

$$w(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i \tag{3.13}$$

This form allows the derivatives of the NURBS curve to be solved as the following:

$$C'(u) = \begin{bmatrix} x'(u) \\ y'(u) \\ z'(u) \end{bmatrix} = \frac{A'(u) - w'(u)C(u)}{w(u)} \tag{3.14}$$

$$C''(u) = \begin{bmatrix} x''(u) \\ y''(u) \\ z''(u) \end{bmatrix} = \frac{A''(u) - 2w'(u)C'(u) - w''(u)C(u)}{w(u)} \tag{3.15}$$

$$C^{(3)}(u) = \begin{bmatrix} x^{(3)}(u) \\ y^{(3)}(u) \\ z^{(3)}(u) \end{bmatrix} = \frac{A^{(3)}(u) - 3w'(u)C''(u) - 3w''(u)C'(u) - w^{(3)}(u)C(u)}{w(u)} \tag{3.16}$$

$$C^{(k)}(u) = \frac{A^{(k)}(u) - \sum_{i=1}^{k} \binom{k}{i} w^{(i)}(u) C^{(k-i)}(u)}{w(u)} \tag{3.17}$$

Using the above Eqn. (3.17), the derivatives at the end points of the NURBS curve can be found. The following Eqn. (3.19) is an alternative formulation to the derivatives of just the B-spline (with weighted control points) that is expressed in terms of the control points instead of the derivative of the basis function. This allows the control points at the ends of the NURBS curve to be solved if the curve's boundary conditions are given, i.e. to keep continuity with the previous curve.

$$A^{(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u)(P_i w_i)^{(k)} \tag{3.18}$$

$$(P_i w_i)^{(k)} = \begin{cases} P_i w_i & k = 0 \\ \dfrac{p-k+1}{u_{i+p+1} - u_{i+k}}\left[(P_{i+1} w_{i+1})^{(k-1)} - (P_i w_i)^{(k-1)}\right] & k > 0 \end{cases} \tag{3.19}$$

From the first derivative of the NURBS curve, the second control point can be solved as follows: ($w_1$can be set to 1 and P_0 is equal to the end point of the previous curve segment)

$$w_1 P_1 = \frac{(u_{p+1} - u_1)}{p} C'(u) w_0 + w_1 P_0 \tag{3.20}$$

Similarly the third control point can be found from the second derivative of the NURBS curve, and from the first two control points (giving the curve C^2 continuity) and it is found as the following:

$$\left.\begin{aligned} w_2 P_2 = \Bigg[& \frac{u_{p+1} - u_2}{p(p-1)} A^{(2)}(u_0) - \frac{1}{u_{p+1} - u_1} w_0 P_0 \\ & + \frac{(u_{p+2} - u_2 + u_{p+1} - u_1)}{(u_{p+2} - u_2)(u_{p+1} - u_1)} w_1 P_1 \Bigg] (u_{p+2} - u_2) \end{aligned}\right\} \tag{3.21}$$

where $A^{(2)}(u_0)$ is the following:

$$\left.\begin{aligned} A^{(2)}(u_0) = & \; C^{(2)}(u_0) w_0 + 2w'(u_0) C'(u_0) + w^{(2)}(u_0) C(u_0) \\ = & \; C^{(2)}(u_0) w_0 + 2\frac{p}{u_{p+1} - u_1}(w_1 - w_0) C'(u_0) \\ & + \frac{p(p-1)}{u_{p+1} - u_2}\Bigg[\frac{1}{u_{p+2} - u_2} w_2 - \frac{(u_{p+2} - u_2 + u_{p+1} - u_1)}{(u_{p+2} - u_2)(u_{p+1} - u_1)} w_1 \\ & + \frac{1}{u_{p+1} - u_1} w_0\Bigg] P_0 \end{aligned}\right\} \tag{3.22}$$

In addition to the knots of a knot vector, the peak of a basis function $N_{i,p}(u)$ is known as a node, t_i, defined as:

$$t_i = \frac{1}{p}\sum_{j=1}^{p} u_{i+j} \qquad i = 0, \ldots, n \tag{3.23}$$

These nodes are calculated from the average of the knots. Nodes represent the curve parameter values where the influence from a particular control point is at its maximum. When a particular point on the curve $C(u^*)$ needs to be modified and moved to a specified location, it can be achieved with minimum modification by moving the control point P_i with the closest node t_i to u^*. If there are no existing nodes nearby, either the two control points with nodes closest to u^* need to be moved or a node can be placed at u^*. A knot $\hat{u}$ can be inserted to the existing knot vector to place a node at u^*. The new node is placed as the index of $k + 1$, where the index k is found by the following:

$$|u^* - t_k| = \min_i |u^* - t_i| \tag{3.24}$$

Then to create the node $t_{k+1} = u^*$, the knot $\hat{u}$ is inserted:

$$\left.\begin{aligned} u^* = t_{k+1} &= \frac{1}{p}\left(\hat{u} + \sum_{j=2}^{p} u_{k+j}\right) \\ \hat{u} &= pu^* - \sum_{j=2}^{p} u_{k+j} \end{aligned}\right\} \tag{3.25}$$

Node insertion will be useful in adding additional freedom to the curve during the toolpath fitting process in between each optimization step since the control points are used as the fitting parameters. The additional control point will have its largest influence at the curve parameter value $u = u^* = t_{k+1}$, which can be used by the optimization algorithm to penalize for the local fitting error. Knot insertion is a fundamental technique in NURBS manipulation where the knot $\hat{u} \in [u_k \quad u_{k+1})$ is inserted in the existing knot of $U = \{u_0 \quad \cdots \quad u_m\}$ to form the new knot vector:

$$\bar{U} = \{\bar{u}_0 = u_0 \quad \cdots \quad \bar{u}_k = u_k \quad \bar{u}_{k+1} = \hat{u} \quad \bar{u}_{k+2} = u_{k+1} \quad \cdots \quad \bar{u}_{m+1} = u_m\} \tag{3.26}$$

The weighted set of $n + 1$ control points are denoted as $P_i^w = [P_{x_i}w_i \quad P_{x_i}w_i \quad P_{y_i}w_i \quad w_i]$ and the set of new basis functions found with the new $\bar{U}$ is denoted as $\bar{N}_{i,p}(u)$, then a new set of weighted $n + 2$ control points R_i^w can be found such that the following is satisfied:

$$\sum_{i=0}^{n} N_{i,p}(u)\, P_i^w = \sum_{i=0}^{n+1} \bar{N}_{i,p}(u)\, R_i^w \tag{3.27}$$

The resulting NURBS curve is unchanged, but there is an additional knot and control point that was inserted, which provides an added flexibility for modification. The new weighted set of $n + 2$ control points, R_i^w, can be found as:

$$R_i^w = \alpha_i P_i^w + (1 - \alpha_i) P_{i-1}^w$$

$$\text{where } a_i = \begin{cases} 1 & i \le k - p \\ \frac{u^* - u_i}{u_{i+p} - u_i} & k - p + 1 \le i \le k \\ 0 & i \ge k + 1 \end{cases} \tag{3.28}$$

Figure 3.3 shows an example of knot insertion where the same NURBS Curve is presented with different number of control points.

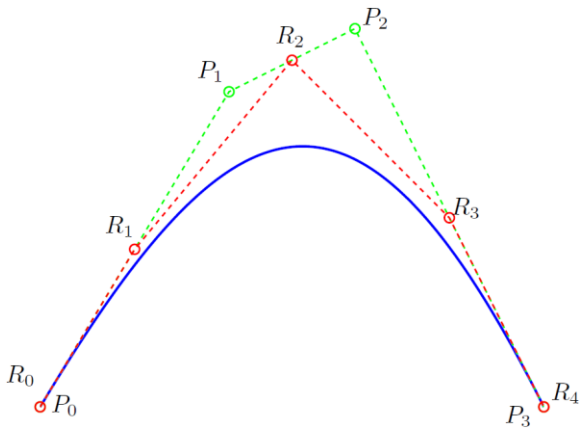


Figure 3.3: Knot Insertion

(Same NURBS Curve (Blue), $n + 1$ control points P (Green), $n + 2$ control points R (Red))

3.3 Double B-Spline Toolpath Representation

The toolpath in five-axis free-form sculptured machining can be expressed as the motion of the tool tip contact point with the work piece, known as the *cutter contact (CC) toolpath* and the *tool orientation*. Every CC toolpath corresponds to one line or curve on the surface. The tool orientations are usually selected to maximize material removal rate and to match the surface curvatures. Tool orientation is also very important in avoiding tool collision and gouging the surface. In order to preserve this desired tool motion, the toolpath format proposed by

Langeron et al. [31] is implemented. Figure 3.4 illustrates this toolpath format consisting of two splines, each being dedicated to the following:

- The CC tool tip positions $q^t = [x^t \quad y^t \quad z^t]$ is fitted with a spline $C(u)$ in part coordinates, i.e. the work piece frame
- Another point on the tool axis $q^O = [x^h \quad y^h \quad z^h]$ is fitted with a spline $C^{up}(u)$

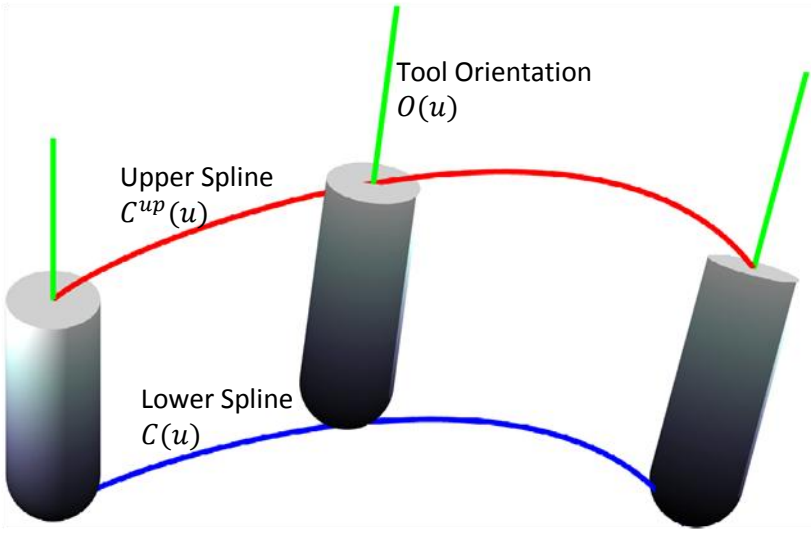


Figure 3.4: Double B-Spline Toolpath Representation

Both the upper spline and the lower spline can be represented with the NURBS formulation. The upper control points are different from the control points of the lower spline, but their knot vector can be the same. For simplicity, the weights can be set to one. The purpose of this representation is such that for every curve parameter value of u, a corresponding tool tip CC position and the tool orientation can simultaneously be obtained from the double splines.

$$\left.\begin{aligned} C(u) &= \begin{bmatrix} x^t(u) \\ y^t(u) \\ z^t(u) \end{bmatrix} = \sum_{i=0}^{n} N_{i,p}(u)\, P_i, \quad u \in U \\ C^{up}(u) &= \begin{bmatrix} x^h(u) \\ y^h(u) \\ z^h(u) \end{bmatrix} = \sum_{i=0}^{n} N_{i,p}(u)\, P_i^{up}, \quad u \in U \end{aligned}\right\} \tag{3.29}$$

The difference between the upper and lower splines gives the tool-axis vector $TA(u)$, which can be normalized to give the unit orientation vector $O(u)$:

$$\left.\begin{aligned} TA(u) = C^{up}(u) - C(u) &= \begin{bmatrix} x^h(u) - x^t(u) \\ y^h(u) - y^t(u) \\ z^h(u) - z^t(u) \end{bmatrix} \\ &= \sum_{i=0}^{n} N_{i,p}(u)\left(P_i^{up} - P_i\right), \quad u \in U \\ O(u) = \begin{bmatrix} O_i \\ O_j \\ O_k \end{bmatrix} &= \frac{TA(u)}{\|TA(u)\|} \end{aligned}\right\} \tag{3.30}$$

3.4 Global Approximation Fit

From standard CAD/CAM software, the tool tip CC toolpath can be retrieved as discrete data points when the planning process breaks the model down into linear segments. Normally, the tool orientation is normal to the free-form surface, and should also be known at these discrete data points. In 5-axis free-form machining, if the large amount of data is interpolated with local individual curves, it may result in non-smooth motion of the cutting process. The least squares global approximation fit for a NURBS curve is used to limit the number of individual curves and tolerances are assigned to limit the error resulting from the least squares fit. The algorithms for the general global approximation least squares fit are taken from Piegl [47].

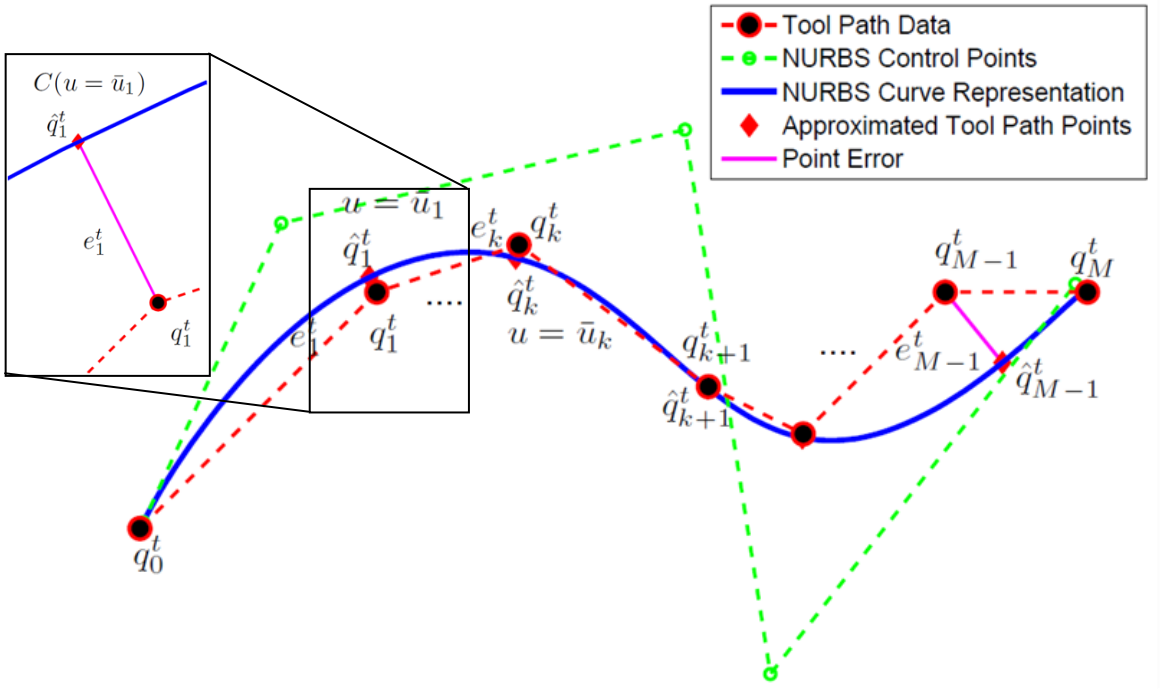


Figure 3.5: NURBS Global Approximation Fit

Let the set of tool tip CC positions be denoted as $q_k^t = \begin{bmatrix} q_{x_k}^t & q_{y_k}^t & q_{z_k}^t \end{bmatrix}$ with $M+1$ data points, $0 \le k \le M$, and let the resulting predictions of the fit from the least squares spline fit be denoted as $\hat{q}_k^t = [x^t(u) \quad y^t(u) \quad z^t(u)]_{|u=\bar{u}_k}$ for the lower spline, then the objective is to minimize the error between the actual data points and the resulting fit predictions as illustrated in Figure 3.5. $\bar{u} = [\bar{u}_0 \quad \cdots \quad \bar{u}_k \quad \cdots \quad \bar{u}_M]$ is the set of spline parameter values that correspond to the given data points, and they are initially calculated using chord length parameterization as given in Eqn. (3.32). This initial parameterization distributes the data points by considering the Euclidian distance between the data points. This is a very rough estimate of the curve parameters since the arc lengths between the corresponding points of the fitted curve will be quite different from the chord lengths of the data points. In addition, there is also no explicit relationship between the arc length of the curve and the curve's parameter.

The total chord length between all the data points can be found using:

$$\left.\begin{aligned} S_c &= \sum_{k=1}^{M} \left\| q_k^t - q_{k-1}^t \right\| \\ &= \sum_{k=1}^{M} \sqrt{\left(q_{x_k}^t - q_{x_{k-1}}^t\right)^2 + \left(q_{y_k}^t - q_{y_{k-1}}^t\right)^2 + \left(q_{z_k}^t - q_{z_{k-1}}^t\right)^2} \end{aligned}\right\} \tag{3.31}$$

The initial and final parameter values are set as $\bar{u}_0 = 0$ and $\bar{u}_M = 1$, and the internal curve parameter values are normalized against the total chord length as shown:

$$\bar{u}_k = \bar{u}_{k-1} + \frac{\left\| q_k^t - q_{k-1}^t \right\|}{S_c}, \quad k = 1, \cdots, M-1 \tag{3.32}$$

The initial knot vector, U, is predefined and obtained as described by Piegl [47]. The length of the knot vector is based on the number of control points, $n+1$, and the degree, p, of the curve. There are a total of $m+1$ or $n+p+2$ knots, and since the initial and final $p+1$ knots are repetitive and are set as zero and one respectively, there are $n-p$ internal knots and $n-p+1$ knot spans:

$$u_0 = u_1 = \cdots = u_p = 0 \text{ and } u_{m-p} = u_{m-p+1} = \cdots = u_m = 1 \tag{3.33}$$

To distribute the data points amongst the knot spans, d is defined as a positive real number.

$$d = \frac{M+1}{n-p+1} \tag{3.34}$$

The $n - p$ internal knots are found with the follow equations:

$$\left.\begin{aligned} &i = floor(j \cdot d) \\ &\alpha = j \cdot d - i \\ &u_{p+j} = (1-\alpha)\bar{u}_{i-1} + \alpha \bar{u}_i \qquad j = 1, \cdots, n-p \end{aligned}\right\} \tag{3.35}$$

By pre-defining the knot vector and all the weights to be one, Figure 3.6 shows each of the $n+1$ basis functions (color) being evaluated at the $M+1$ chord length parameterized curve values, $\bar{u}$, (dashed) to form the matrix $\Phi_{M+1\times n+1}$. Then a system of linear equations can be formed to calculate the position predictions on the B-spline curve for the data points:

$$\left.\begin{aligned} \hat{q}^t &= \begin{bmatrix} \hat{q}^t_{x_0} & \hat{q}^t_{y_0} & \hat{q}^t_{z_0} \\ \hat{q}^t_{x_1} & \hat{q}^t_{y_1} & \hat{q}^t_{z_1} \\ \vdots & \vdots & \vdots \\ \hat{q}^t_{x_M} & \hat{q}^t_{y_M} & \hat{q}^t_{z_M} \end{bmatrix} \\ &= \begin{bmatrix} N_{0,p}(\bar{u}_0) & N_{1,p}(\bar{u}_0) & N_{2,p}(\bar{u}_0) & \dots & N_{n,p}(\bar{u}_0) \\ N_{0,p}(\bar{u}_1) & N_{1,p}(\bar{u}_1) & N_{2,p}(\bar{u}_1) & \dots & N_{n,p}(\bar{u}_1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ N_{0,p}(\bar{u}_M) & N_{1,p}(\bar{u}_M) & N_{2,p}(\bar{u}_M) & \dots & N_{n,p}(\bar{u}_M) \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} \\ P_{x_1} & P_{y_1} & P_{z_1} \\ \vdots & \vdots & \vdots \\ P_{x_n} & P_{y_n} & P_{z_n} \end{bmatrix} \\ \hat{q}^t{}_{M+1\times 3} &= \Phi_{M+1\times n+1} P_{n+1\times 3} \end{aligned}\right\} \tag{3.36}$$

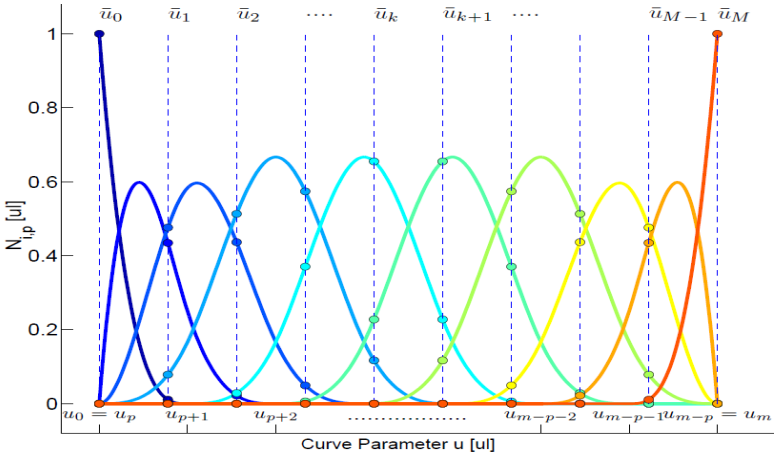


Figure 3.6: Basis Function Evaluation at Chord Length Parameterized Values

The control points, $P_{n+1\times 3}$, are the fit parameters. The errors between the actual data points, q^t, and the computed points on the fitted curve at $\bar{u}$, $\hat{q}^t$ is given as e^t:

$$e^t = q^t - \hat{q}^t = q^t - \sum_{i=0}^{n} N_{i,p}(\bar{u})P = q^t - \Phi P \tag{3.37}$$

The least squares objective function becomes the minimization problem of $\min_x \|Ax - B\|$:

$$J_L = \frac{1}{2} e^{t^T} e^t = \frac{1}{2}(q^t - \Phi P)^T (q^t - \Phi P) \tag{3.38}$$

This least squares problem can be solved by setting the derivative of the objective function with respect to the minimization parameter, the control points, $\frac{\partial J_L}{\partial P}$ to zero.

$$\left.\begin{aligned} &\frac{\partial J_L}{\partial P} = -\Phi^T(q^t - \Phi P) = 0 \\ &\Phi^T q^t = \Phi^T \Phi P \\ &P = (\Phi^T \Phi)^{-1} \Phi^T q^t \end{aligned}\right\} \tag{3.39}$$

This provides the control points for the NURBS curve in the least squares fit. However, since the initial control point is coincident with the initial position of the toolpath, and the second and third control points can be found from the initial first and second derivatives, the first three control points can be found directly using Eqn. (3.20) & (3.21) to achieve C^2 continuity. In that case, ΦP can be broken down into the known and unknown components:

$$\left.\begin{aligned} \Phi_{M+1\times n+1} P_{n+1\times 3} &= {\Phi^{fit}}_{M+1\times n-2} {P^{fit}}_{n-2\times 3} + \psi_{M+1\times 3} P^{known}_{3\times 3} \\ &= \begin{bmatrix} N_{3,p}(\bar{u}_0) & \cdots & N_{n,p}(\bar{u}_0) \\ N_{3,p}(\bar{u}_1) & \cdots & N_{n,p}(\bar{u}_1) \\ \vdots & \ddots & \vdots \\ N_{3,p}(\bar{u}_M) & \cdots & N_{n,p}(\bar{u}_M) \end{bmatrix} \cdot \begin{bmatrix} P_{x_3} & P_{y_3} & P_{z_3} \\ P_{x_4} & P_{y_4} & P_{z_4} \\ \vdots & \vdots & \vdots \\ P_{x_n} & P_{y_n} & P_{z_n} \end{bmatrix} + \\ &\begin{bmatrix} N_{0,p}(\bar{u}_0) & N_{1,p}(\bar{u}_0) & N_{2,p}(\bar{u}_0) \\ N_{0,p}(\bar{u}_1) & N_{1,p}(\bar{u}_1) & N_{2,p}(\bar{u}_1) \\ \vdots & \vdots & \vdots \\ N_{0,p}(\bar{u}_M) & N_{1,p}(\bar{u}_M) & N_{2,p}(\bar{u}_M) \end{bmatrix} \cdot \begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} \\ P_{x_1} & P_{y_1} & P_{z_1} \\ P_{x_2} & P_{y_2} & P_{z_2} \end{bmatrix} \end{aligned}\right\} \tag{3.40}$$

where $\psi_{M+1\times 3} P^{known}_{3\times 3}$ is known and can be lumped with q^t. If this is solved with the minimization function of the Matlab optimization toolbox [42], the new fit parameter can simply replaced by:

$$P_{n+1\times 3} = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ x_{n-2\times 3} \end{bmatrix} \tag{3.41}$$

where x is of size $(n-2)\times 3$ and is the fit parameter for the unknown control points, P_0, P_1, P_2 are the known control points and P is now used in the minimization problem of $\min_P \|AP - B\|$. For simplicity, the minimization will still be denoted as $(q^t - \Phi P)$.

In addition, to cascading known control points onto the fit parameter, additional positional, tangential, or higher derivative constraints can be included using the method of Lagrange Multiplier to compose the constrained least squares minimization problem.

Let specific position constrains be denoted by $\xi_k^{(0)} = [x^t(u) \quad y^t(u) \quad z^t(u)]_{|u=\bar{u}_k}$, $k = 0, \cdots, l_0$,the linear equations are shown as:

$$\left.\begin{bmatrix} \xi_0^{(0)} \\ \vdots \\ \xi_{l_0}^{(0)} \end{bmatrix} = \begin{bmatrix} N_{0,p}(\bar{u}_0) & \cdots & N_{n,p}(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ N_{0,p}(\bar{u}_{l_0}) & \cdots & N_{n,p}(\bar{u}_{l_0}) \end{bmatrix} \cdot \begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} \\ \vdots & \vdots & \vdots \\ P_{x_n} & P_{y_n} & P_{z_n} \end{bmatrix} \right\} \tag{3.42}$$
$$\xi_{l_0+1\times 3}^{(0)} = L_{l_0+1\times n+1}^{(0)} P_{n+1\times 3}$$

Similarly, the constraints for the tangents, $\xi_k^{(1)} = \left[\frac{dx^t(u)}{du} \quad \frac{dy^t(u)}{du} \quad \frac{dz^t(u)}{du}\right]_{|u=\bar{u}_k}$, $k = 0, \cdots, l_1$, and the second derivative, $\xi_k^{(2)} = \left[\frac{d^2x^t(u)}{du^2} \quad \frac{d^2y^t(u)}{du^2} \quad \frac{d^2z^t(u)}{du^2}\right]_{|u=\bar{u}_k}$, $k = 0, \cdots, l_2$, are imposed using the derivatives of the basis functions and are given as $N_{i,p_u}(\bar{u}_k) = \frac{dN_{i,p}}{du}_{|u=\bar{u}_k}$ and $N_{i,p_{uu}}(\bar{u}_k) = \frac{d^2N_{i,p}}{du^2}_{|u=\bar{u}_k}$:

$$\left.\begin{bmatrix} \xi_0^{(1)} \\ \vdots \\ \xi_{l_1}^{(1)} \end{bmatrix} = \begin{bmatrix} N_{0,p_u}(\bar{u}_0) & \cdots & N_{n,p_u}(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ N_{0,p_u}(\bar{u}_{l_1}) & \cdots & N_{n,p_u}(\bar{u}_{l_1}) \end{bmatrix} \cdot \begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} \\ \vdots & \vdots & \vdots \\ P_{x_n} & P_{y_n} & P_{z_n} \end{bmatrix} \right.$$
$$\xi_{l_1+1\times 3}^{(1)} = L_{l_1+1\times n+1}^{(1)} P_{n+1\times 3}$$
$$\left.\begin{bmatrix} \xi_0^{(2)} \\ \vdots \\ \xi_{l_2}^{(2)} \end{bmatrix} = \begin{bmatrix} N_{0,p_{uu}}(\bar{u}_0) & \cdots & N_{n,p_{uu}}(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ N_{0,p_{uu}}(\bar{u}_{l_2}) & \cdots & N_{n,p_{uu}}(\bar{u}_{l_2}) \end{bmatrix} \cdot \begin{bmatrix} P_{x_0} & P_{y_0} & P_{z_0} \\ \vdots & \vdots & \vdots \\ P_{x_n} & P_{y_n} & P_{z_n} \end{bmatrix} \right\} \tag{3.43}$$
$$\xi_{l_2+1\times 3}^{(2)} = L_{l_2+1\times n+1}^{(2)} P_{n+1\times 3}$$

The constraint matrix is cascaded into $L = \begin{bmatrix} L^{(0)} \\ L^{(1)} \\ L^{(2)} \end{bmatrix}_{(l_0+1+l_1+1+l_2+1)\times(n+1)}$, and the constraint vector as $\xi = \begin{bmatrix} \xi^{(0)} \\ \xi^{(1)} \\ \xi^{(2)} \end{bmatrix}_{(l_0+1+l_1+1+l_2+1)\times 3}$. The tangent and higher derivative constraints can either be specified by the designer, or the first and second derivative values at $\bar{u}$ can be estimated with the cubic polynomial derivative estimation method described in Erkorkmaz [14]. Up to the third derivative can be estimated by extending the estimation polynomial to a quintic polynomial. However, due to a lack of extra neighboring points at the beginning and end of the series of estimation polynomial, the derivative estimates at the beginning and end are less accurate and contain undesired noise.

With the additional equality constraints, the optimization problem becomes:

$$\left.\begin{array}{c} J_L = \min_P \frac{1}{2}(q^t - \Phi P)^T (q^t - \Phi P) \\ \text{Subject to: } L \cdot P = \xi \end{array}\right\} \tag{3.44}$$

This is a linear quadratic minimization problem, and can be solved using the Lagrange Multipliers, denoted as: $\Lambda = [\Lambda_0 \quad \cdots \quad \Lambda_r]$, where $r = (l_0 + 1 + l_1 + 1 + l_2 + 1)$, the augmented objective function can be written as:

$$J_L(P, \Lambda) = \frac{1}{2}(q^t - \Phi P)^T (q^t - \Phi P) + \Lambda(L \cdot P - \xi) \tag{3.45}$$

This minimization can be solved by equating the partial derivative of the two control parameters to zero $\left(\frac{\partial J_L(P,\Lambda)}{\partial P} = 0 \text{ and } \frac{\partial J_L(P,\Lambda)}{\partial \Lambda} = 0\right)$, which yields the following linear equation system:

$$\left.\begin{array}{c} \Phi^T \Phi P + L^T \Lambda = \Phi^T q^t \\ L \cdot P = \xi \end{array}\right\} \Rightarrow \begin{bmatrix} \Phi^T \Phi & L^T \\ L & 0 \end{bmatrix} \cdot \begin{bmatrix} P \\ \Lambda \end{bmatrix} = \begin{bmatrix} \Phi^T q^t \\ \xi \end{bmatrix} \tag{3.46}$$

A solution can only be obtained if $\Phi^T \Phi$ is positive definite and non-singular. The method of defining the initial knot vector from Eq. (3.34) & (3.35) ensured that every knot interval $[u_j \quad u_{j+1}]$ contains at least one $\bar{u}$ value to satisfy the non-singularity condition of $\Phi^T \Phi$. In addition, provided that the constraints are linearly independent and less than the number of unknowns $(r \leq n + 1)$, an unique solution can be obtained.

For the global fitting of the lower position spline, $C(u)$, the CC discrete toolpath position data, q^t, are used in the minimization algorithm. The fitting of the upper spline takes place after the lower spline is fitted since the purpose of the upper spline is to provide the desired reference orientation vector with the toolpath, which now has a lower spline curve interpolation as $C(u)$. The reference data points for the fitting of the upper spline is updated as an offset to the lower spline in the direction of the reference orientation $O^r = \begin{bmatrix} O_i^r & O_j^r & O_k^r \end{bmatrix}$, by a scalar value of H. The reason for the update is because there is some associated error with the lower spline fitting and the upper spline fitting, and if q^O is not updated, there will be two sources of error contributing to the interpolated orientation. By updating q^O, the first source of error associated with the lower spline can be eliminated. The updated upper spline data points are:

$$q_k^O = C(\bar{u}_k) + H \cdot O_k^{ref} \tag{3.47}$$

3.4.1 Tolerance Definition and Parameter Correction

The global approximate least squares algorithm minimized the least squares error with the given initial knot vector, U, at the chord length parameterized curve parameter values of $\bar{u}$ against the set of $M + 1$ q^t data points. The problems are that with the flexibility of the limited control points as defined by the size of the knot vector, the desired fitting error limit may not be achievable, and that the $\bar{u}$ curve parameter values may not be a good approximation for the data points. A fitting error limit (e_{max}) is defined to control the deviation of the lower spline from the reference data points. Figure 3.7 illustrates how the fitted toolpath is within the allowable tolerance band as given by e_{max} for all the data points.

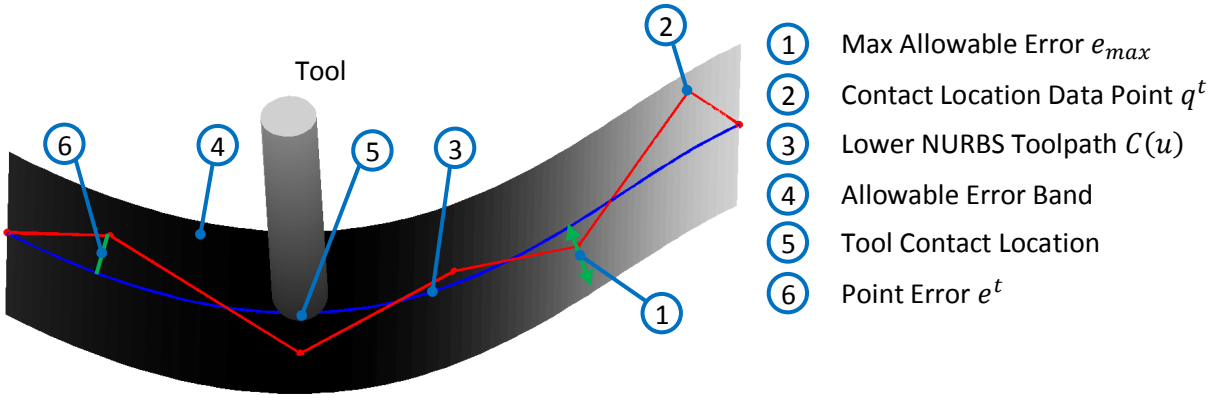


Figure 3.7: Tolerance Definition

After the curve fit, the formulation for the curve is now available, making it possible to compute and update the corresponding $\bar{u}$ values on the curve that are closest to the data points. For each of the $M+1$ data points, the closest curve parameter value can be solved using point projection as shown in Figure 3.8. It is known that the minimum distance occurs where the vector $C(\bar{u}_k) - q_k^t = \begin{bmatrix} x^t(\bar{u}_k) - q_x^t & y^t(\bar{u}_k) - q_y^t & z^t(\bar{u}_k) - q_z^t \end{bmatrix}$ is perpendicular to the tangent of the curve at $\bar{u}_k$ and it can be found using the following dot product:

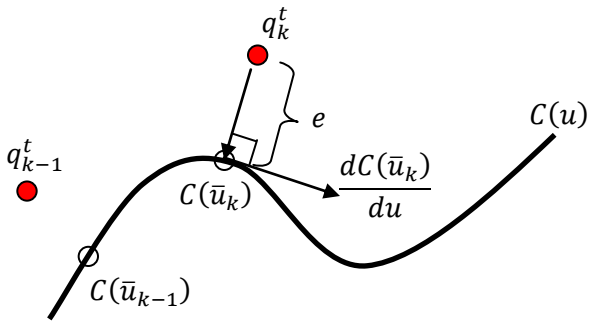


Figure 3.8: Corresponding Spline Parameter

$$f(\bar{u}_k) = \left(C(\bar{u}_k) - q_k^t\right) \cdot \frac{dC(\bar{u}_k)}{du} = \left(C(\bar{u}_k) - q_k^t\right)\left[\frac{dC(\bar{u}_k)}{du}\right]^T = 0 \qquad (3.48)$$

The roots of $f(\bar{u}_k)$ can be found using the Newton-Raphson method.

$$\left.\begin{aligned} & x_{i+1} = x_i - \frac{f(x_i)}{df(x_i)/dx} = \bar{u}_{k\,i+1} = \bar{u}_{k_i} - \frac{f(\bar{u}_{k_i})}{df(\bar{u}_{k_i})/du} \\ & \text{where} \\ & \frac{df(\bar{u}_{k_i})}{du} = \left(\frac{dC(\bar{u}_{k_i})}{du}\right)\left(\frac{dC(\bar{u}_{k_i})}{du}\right)^T + \left(C(\bar{u}_{k_i}) - q_k^t\right)\left(\frac{d^2C(\bar{u}_{k_i})}{du^2}\right)^T \end{aligned}\right\} \qquad (3.49)$$

The convergence of the Newton iteration is highly dependent on the initial guess of $\bar{u}_k$ and the initial guess of $\bar{u}_k$ that was computed based on chord length approximation may not necessarily converge. The lower bound of the search should always be $\bar{u}_{k-1}$, where $\bar{u}_0 = u_0$ and $\bar{u}_M = u_m$, since the curve parameter for the NURBS curve is always monotonically increasing. The upper bound for the search is the last value in the knot vector, u_m. This iteration is terminated when either bound is violated. The iteration is considered to have converged, when either the Euclidean distance or the cosine of the distance vector and the curve tangent (i.e.$f(\bar{u}_k)$) or the change in parameter $\Delta\bar{u}_k$ is small as indicated by the following conditions:

The criteria that the current point computed at $\bar{u}_{k_i}$ is already within Euclidean distance tolerance to the data point is the following:

(1) $$\text{if } \left\| C(\bar{u}_{k_i}) - q_k^t \right\| \leq \varepsilon_1 \tag{3.50}$$

The criteria that the cosine of the angle or $f(\bar{u}_k)$ is small and within the specified tolerance of ε_2:

(2) $$\text{if } \frac{\left| \left(C(\bar{u}_{k_i}) - q_k^t \right) \left[\frac{dC(\bar{u}_{k_i})}{du} \right]^T \right|}{\left\| C(\bar{u}_{k_i}) - q_k^t \right\| \left\| \frac{dC(\bar{u}_{k_i})}{du} \right\|} \leq \varepsilon_2 \tag{3.51}$$

The criteria that the curve parameter does not change significantly during the iteration:

(3) $$\text{if } \left\| (\bar{u}_{k_{i+1}} - \bar{u}_{k_i}) \left(\frac{dC(\bar{u}_{k_i})}{du} \right) \right\| \leq \varepsilon_1 \tag{3.52}$$

A value of $10^{-6}[\text{mm}]$ for ε_1 is typically more than adequate for basic machining operations to indicate that the data point lies on or very close to the curve at $\bar{u}_{k_i}$. The cosine angle tolerance, ε_2, indicated that $\bar{u}_{k_i}$ is a valid root of $f(\bar{u}_k)$ and is selected to be less than ε_1, such as 10^{-8} [ul]. In cases where the iteration is terminated without convergence, the initial guess of $\bar{u}_{k_0}$ needs to be adjusted. Another guess can be picked to be within the current knot span: $u_j \leq \bar{u}_{k_0}^{previous} < \bar{u}_{k_0}^{next} < u_{j+1}$. If convergence is still not achieved after several trials within the current knot span, the next knot span of $[u_{j+1} \quad u_{j+2})$ is searched and this is repeated until either convergence is achieved or until some clipping algorithm determines that the curve is now moving away from the data point. In the case where convergence cannot be achieved, the $\bar{u}_k$ value where $\left\| C(\bar{u}_{k_i}) - q_k^t \right\|$ was minimum during the search process is taken. Once $\bar{u}_k$ is found, the point error is computed as:

$$e_k = \left\| C(\bar{u}_{k_i}) - q_k^t \right\|, \quad k = 0, \cdots, M+1 \tag{3.53}$$

When all the e_k are less than or equal to e_{max}, the lower spline fit is complete. When this is not satisfied, the number of control points used for the fitting process is increased. The goal of the global approximation fit is to satisfy the specified tolerance with the minimum amount of

control point. While having a lower number of control points will result in higher errors at the tool path data points, each control point has its influence on the curve and having more control points would increase the fluctuation along the toolpath. While the initial number of control points required is not defined optimally to adequately capture the overall shape, 5% of the total given toolpath data points, $n = \text{floor}\ ((M+1)/20)$ is used as the initial number of control points. Using Eqn. (3.35), the initial knot vector is generated which allows the lower spline to be fitted. Then, $\bar{u}$ are found and updated using Newton-Raphson iteration and the point error condition is checked. If the error condition is not satisfied, the knot value u^* is found using Eqn. (3.25) to add a node t_k at the $\bar{u}_k$ value where the largest point error occurred, and it is added with Eqn. (3.28). Adding a node at this curve location will reduce the errors at this portion of the curve. The new set of $n+1$ control points can be used as initial conditions for next fitting operation if the Matlab optimization toolbox is used. This process is repeated until the lower spline satisfies the e_{max} tolerance.

While the maximum data point error is a valid characterization for ensuring the geometrical tolerance of the toolpath on the lower spline, orientation error is a more relevant characterization for the upper spline. The orientation is the difference between the upper and lower splines, and the error in orientation is defined as the following:

$$\left.\begin{aligned} &\sin\left(e_{\theta_k}\right) = \frac{\|O_k^r \times O_k\|}{\|O_k^r\|\|O_k\|}, \cos\left(e_{\theta_k}\right) = \frac{O_k^r \cdot O_k}{\|O_k^r\|\|O_k\|} \\ &e_{\theta_k} = \tan^{-1}\left(\frac{\|O_k^r \times O_k\|}{O_k^r \cdot O_k}\right) \\ &\text{where} \\ &O_k = O(\bar{u}_k) = \frac{C^{up}(\bar{u}_k) - C(\bar{u}_k)}{\|C^{up}(\bar{u}_k) - C(\bar{u}_k)\|} \end{aligned}\right\} \tag{3.54}$$

After the lower spline is fitted, the same knot vector is taken and used to fit the upper spline. The set of data points q^O is found using the fitted lower spline and the reference orientation vector, O^{ref}. The upper spline is fitted and the orientation error is checked. It is not likely that the fitting of the upper spline has to be iterated too many times, if at all, since the number of control points that are available to use as the fitting parameter has already satisfied the convergence tolerance of the lower spline and the number of data points, q^t and q^O are both $M+1$. The resulting outputs of the fitting process are the control points of the lower spline,

$P_{n+1\times 3}$, and the upper spline, $P^{\mathrm{up}}_{n+1\times 3}$ as well as the knot vector, U. Figure 3.9 illustrates the flow chart of the fitting process.

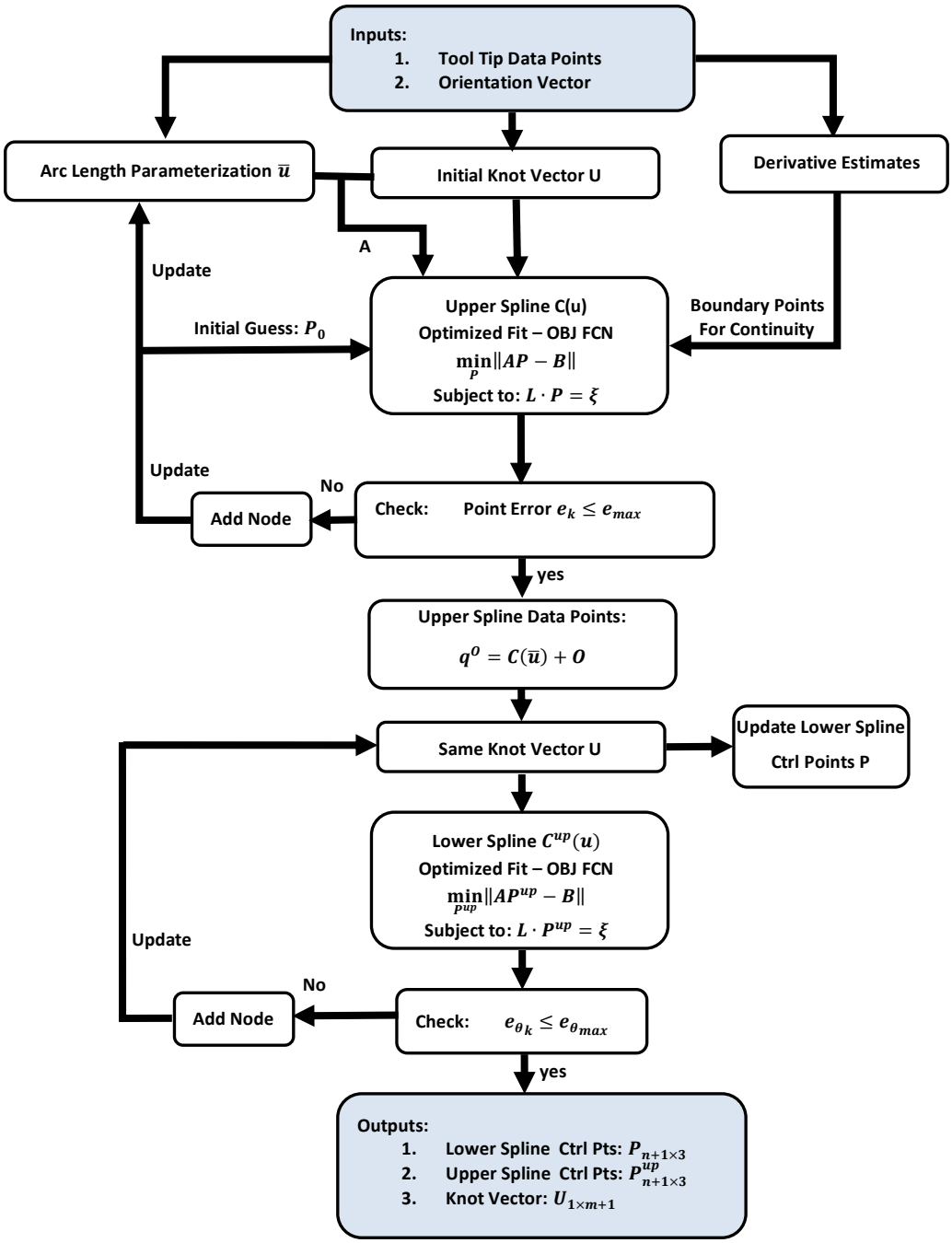
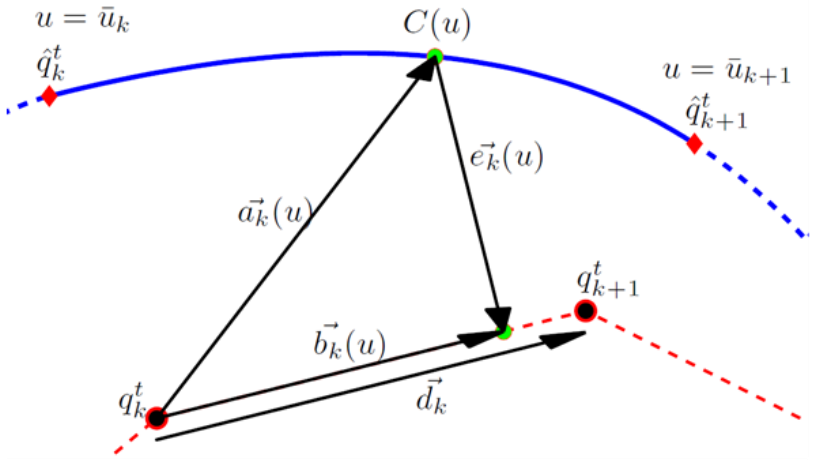


Figure 3.9: Double Spline Toolpath Fitting Flow Chart

3.4.2 Quadrature Chord Error with Minimization

The linear least squares global approximation method was introduced to minimize the error between the set of discrete data points q^t and the NURBS curve at the estimated curve parameter values of $\bar{u}$, which were parameterized based on the chord lengths. Due to the nature that only these selected points along the curve are minimized, in between these points, large errors are often observed between the curve and the chord of consecutive data points q_k^t and q_{k+1}^t. This section will describe a method to quantify these chord errors and attempt to apply a minimization scheme for these errors.

At every curve parameter value u, there is an associated chord error. Assuming that at the estimated curve parameter values of $\bar{u}_k$ on the curve, the $\hat{q}_k^t$ are valid correspondents to the toolpath data points q_k^t, then for the region $u \in [\bar{u}_k \quad \bar{u}_{k+1}]$, the immediate chord formed by $[q_k^t \quad q_{k+1}^t]$, is used. The chord error vector is calculated based on the method of vector projection as illustrated in Figure 3.10:



$$\vec{a_k}(u) = C(u) - q_k^t$$

$$\vec{d_k}(u) = q_{k+1}^t - q_k^t$$

$$\vec{b_k}(u) = \vec{a_k}(u)D_k$$

where D_k is the projection matrix onto $\vec{d_k}(u)$

$$D_k = \frac{\vec{d_k}(u)^T \vec{d_k}(u)}{\|\vec{d_k}(u)\|^2}$$

$$\vec{e_k}(u) = \vec{a_k}(u) - \vec{b_k}(u) = \vec{a_k}(u)(I - D_k)$$

Figure 3.10: Chord Error Projection

The norm of all the chord error vectors can be integrated over the entire curve segment $u \in [u_0 \quad u_m]$ to represent the total chord error. Since there is a different chord for every consecutive toolpath data point q_k^t, the scalar-valued total integration Q_e is separated into M-1 segments of Q_{e_k}. Each of the segment has its own projection 3×3 matrix D_k and its own origin at q_k^t.

$$Q_e = \int_{u_0}^{u_m} \|e(u)\| du = \sum_{k=0}^{M-1} \int_{\bar{u}_k}^{\bar{u}_{k+1}} \|e_k(u)\| du = \sum_{k=0}^{M-1} Q_{e_k} \tag{3.55}$$

$$\left.\begin{aligned} Q_{e_k} &= \int_{\bar{u}_k}^{\bar{u}_{k+1}} \|\vec{e}_k(u)\| du = \int_{\bar{u}_k}^{\bar{u}_{k+1}} \|\vec{a}_k(u)_{(I - D_k)}\| du \\ &= \int_{\bar{u}_k}^{\bar{u}_{k+1}} \left\| \left(C(u) - q_k^t\right)_{(I - D_k)} \right\| du \end{aligned}\right\} \tag{3.56}$$

$C(u)$ is the NURBS curve equation, I is the 3×3 identity matrix. Since the basis functions of the NURBS curve can be expressed analytically, it is possible to solve these Q_{e_k} integrals analytically. However, it is much more computationally efficient to evaluate the integrals numerically using the 3/8 Simpsons Rule:

$$\left.\begin{aligned} Q_{e_k} &= \int_{\bar{u}_k}^{\bar{u}_{k+1}} \|\vec{e}_k(u)\| du = \int_a^b f(u) du \\ &\cong \frac{3h}{8}\left(f(a) + 3f\left(\frac{2a+b}{3}\right) + 3f\left(\frac{a+2b}{3}\right) + f(b)\right) \end{aligned}\right\} \tag{3.57}$$

where $h = \frac{(b-a)}{3}$. The interval of $[a \quad b]$ can be bisected into two subintervals, $[a_1 \quad b_1]$ and $[a_2 \quad b_2]$, where $b_1 = a_2 = \frac{(a+b)}{2}$. This forms the composite 3/8 Simpson's rule where the integral over the region $[a \quad b]$ is approximated by:

$$f(a,b) \approx f(a_1,b_1) + f(a_2,b_2) \tag{3.58}$$

By assigning a tolerance of ε, then satisfying the following condition of:

$$\frac{1}{10}\left|f(a_1,b_1) + f(a_2,b_2) - f(a,b)\right| < \varepsilon \tag{3.59}$$

would imply that the functional variance over $[a \quad b]$ is kept in check. A tolerance value of 10^{-6} [ul] is generally more than adequate for the solution to converge. The intervals are solved from the left to right. Whenever the tolerance given in Eq. (3.59) is not satisfied, the interval is sub-divided further. With every tier of sub-division, the tolerance used is halved (tighter tolerance). It is important to keep track of the tier of sub-divisions, as some regions of the curve are rather consistent while other parts may have large variances and require further divisions. This proceeds until all the sub-intervals of the region $u \, \epsilon \, [\bar{u}_k \quad \bar{u}_{k+1}]$ converge to the tolerance. The total sum of all the sub-divisions will give an approximation of the total Chord error in the interval. This is known as the adaptive quadrature method since the step size on the sections of the curve are adjusted based on the consistency of integration function locally.

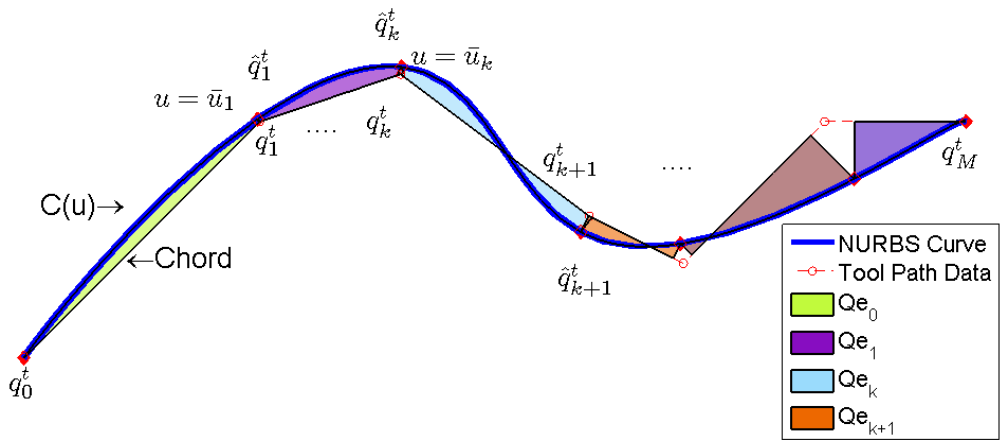


Figure 3.11: Chord Error Integration

Figure 3.11 illustrates the integration of these chord errors. Note that this integration is different from calculating the area between the NURBS curve and the chord lines. For instance, if the norm of the error vector is one throughout the entire path with the knot vector defining the domain as $u \epsilon [u_0 \quad u_m]$, this integration will result in a value of $(u_m - u_0)$, while the computed area can be of any positive value.

With the Quadrature Chord Error defined, for each chord segment, the chord error can be integrated and expressed as the following:

$$\left.\begin{aligned}\int_{\bar{u}_k}^{\bar{u}_{k+1}} e(u)du &= \int_{\bar{u}_k}^{\bar{u}_{k+1}} (\Phi(u)P - q_k^t)(I - D_k)du \\ &= \int_{\bar{u}_k}^{\bar{u}_{k+1}} \Phi(u)du\, P(I - D_k) - (\bar{u}_{k+1} - \bar{u}_k)q_k^t(I - D_k) \\ &= K_{N_k} P(I - D_k) - (\bar{u}_{k+1} - \bar{u}_k)q_k^t(I - D_k)\end{aligned}\right\} \tag{3.60}$$

This leads to the following minimum chord error objective function:

$$\left.\begin{aligned}J_Q &= \frac{1}{2}\sum_{k=0}^{M-1}\Big(K_{N_k}P(I - D_k) - (\bar{u}_{k+1} - \bar{u}_k)q_k^t(I - D_k)\Big)^T \\ &\qquad \cdot \Big(K_{N_k}P(I - D_k) - (\bar{u}_{k+1} - \bar{u}_k)q_k^t(I - D_k)\Big) \\ J_Q &= \frac{1}{2}\sum_{i=0}^{M-1}\big(A_{e_k}P(I - D_k) - B_{e_k}\big)^T\big(A_{e_k}P(I - D_k) - B_{e_k}\big)\end{aligned}\right\} \tag{3.61}$$

where $A_{e_k} = K_{N_k}$ and $B_{e_k} = (\bar{u}_{k+1} - \bar{u}_k)q_k^t(I - D_k)$

In the global approximation fit, in order to minimize the error at the toolpath data points q_k^t, the basis function was derived from the given knot vector and the basis function was evaluated at values of $\bar{u}$. The problem with this is that even though the formulation of the basis functions is known analytically, they are only evaluated at specific locations, effectively neglecting the contribution of intermediate values. The basis functions can be integrated easily using the Composite Simpsons rule (the adaptive method with the integration tolerance is not required since the basis functions are smooth and well-defined) to form the K_{N_k} matrix of size $1 \times n$:

$$\left.\begin{aligned} K_{N_k} &= \int_{\bar{u}_k}^{\bar{u}_{k+1}} \Phi(u)du \\ &\cong \frac{\bar{u}_{k+1} - \bar{u}_k}{8}\left[\Phi(\bar{u}_k) + 3\Phi\left(\frac{5\bar{u}_k + 1\bar{u}_{k+1}}{6}\right)\right. \\ &\quad +3\Phi\left(\frac{4\bar{u}_k + 2\bar{u}_{k+1}}{6}\right) + 2\Phi\left(\frac{3\bar{u}_k + 3\bar{u}_{k+1}}{6}\right) \\ &\quad \left.+3\Phi\left(\frac{2\bar{u}_k + 4\bar{u}_{k+1}}{6}\right) + 3\Phi\left(\frac{1\bar{u}_k + 5\bar{u}_{k+1}}{6}\right) + \Phi(\bar{u}_{k+1})\right] \end{aligned}\right\} \quad (3.62)$$

A graphical representation of the K_{N_k} matrix is shown in Figure 3.12. By having this integration matrix, the contribution of the basis functions within the region is known, which correspondingly provides the contribution of the control points within the region; this provides a much more accurate minimization than just minimizing at specific locations. Note that from the property of the basis function, the sum of all the K_{N_k} $1 \times n$ vectors will yield a single $1 \times n$ vector which represents the area enclosed by each of the basis functions, and the sum of all these areas will yield $(u_m - u_0)$. The K_{N_k} matrix is providing the contribution of the control points in the regions that are of interest, i.e. to the immediate chord and it is scaled accordingly to the particular basis function. In addition, the $(I - D_k)$ is the factor that ensures the control points in the local region are being minimized to the relevant chord.

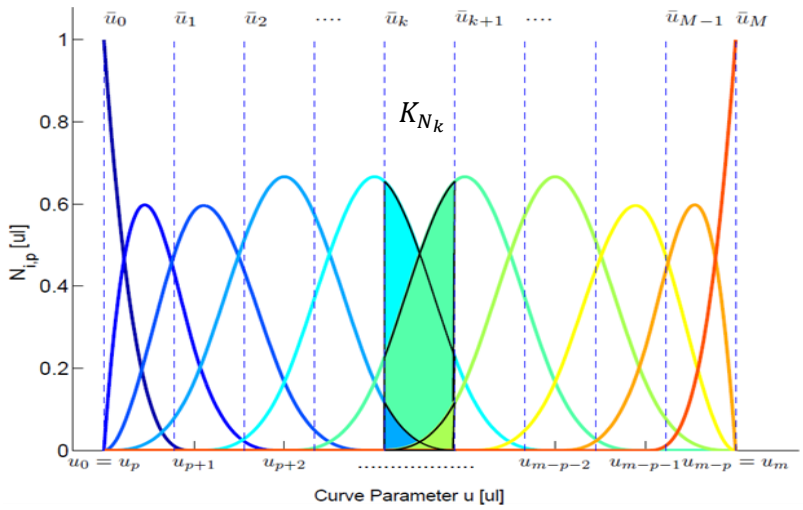


Figure 3.12: Basis Function Integration in Region of Immediate Chord

The previous optimization problem for the least squares fitting was given as:

$$J_L(P, \Lambda) = \frac{1}{2}(q^t - \Phi P)^T(q^t - \Phi P) + \Lambda(L \cdot P - \xi) \tag{3.63}$$

and this is modified by adding the minimum chord error objective function:

$$\min_P[J_L + J_Q] = \min_P \frac{1}{2}\Bigg[\gamma(q^t - \Phi P)^T(q^t - \Phi P) + \beta \sum_{i=0}^{M-1} \left(A_{e_k} P(I - D_k) - B_{e_k}\right)^T \left(A_{e_k} P(I - D_k) - B_{e_k}\right)\Bigg] \tag{3.64}$$

Subject to: $L \cdot P = \xi$

where γ and β are the weights of the multi-objective minimization problem. γ is weight for the fitting effect at the tool data points, while β is the weight for the fitting effect in between the tool data points. Since the effect from β is contributed from the integration of u, and γ is the effect of discrete u values, generally β should be set greater than γ. The effect of these weight factors will be investigated with simulations.

3.4.3 Jerk Minimization

In the least squares fitting process, the curve is fitted based on geometrical error against the set of given discrete toolpath data. If the minimization scheme for the control points were only based on geometrical tolerances, the smoothness of the curve will be neglected and it

would eventually decompose back into having linear segments connecting all the data points, having exactly zero error. In addition, during the fitting process, the knot vector was created based on chord length of the linear segmenting of data points; this will most likely not be the case once a curve is fitted to the points and the segment length along the curve will differ from the original chord lengths. Furthermore, as more knots are being added to the knot vector during the fit, extra control points are added to the curve. While this increases the flexibility of the NURBS curve geometrically, it also means that there is an extra pulling effort. Even though the NURBS curve is infinitely differentiable within the knot spans, it is only $p-k$ times differentiable at the knots of multiplicity k. This can possibly lead to impulses embedded within the curve which could cause oscillations within the derivatives of the curve. These effects may excite the natural modes of the CNC system during operation and deteriorate its tracking performance.

During machining operations, real time interpolation is applied to the NURBS curve toolpath to get the machine tool motion; excessive jerk is undesired as it leads to excitation of vibrations in the CNC assembly. While the jerk on tool motion is usually defined as the third derivative of tool tip displacement, s, with respect to time, $\frac{d^3s}{dt^3}$, in this section, the smoothness on the NURBS toolpath $C(u)$ is optimized. The jerk component of the geometrical curve is minimized and this will ultimately lower the tool motion jerk of $\frac{d^3s}{dt^3}$.

The tool position is given by $C(u) = [x(u) \quad y(u) \quad z(u)]$. The geometrical jerk is defined as the third derivative of $C(u)$ with respect to the curve parameter u, C_{uuu}, and its minimization can be defined as the following minimization problem:

$$\left.\begin{aligned} J_k &= \int_{u_0}^{u_m} {C_{uuu}}^T C_{uuu} du \\ J_k &= \int_{u_0}^{u_m} (N_{uuu}(u)P)^T (N_{uuu}(u)P) du \\ &= P^T \int_{u_0}^{u_m} \big(N_{uuu}(u)\big)^T \big(N_{uuu}(u)\big) du P = P^T K P \end{aligned}\right\} \tag{3.65}$$

where $N_{uuu}(u) = [N_{0,p\,uuu}(u) \quad N_{1,p\,uuu}(u) \quad \cdots \quad N_{n,p\,uuu}(u)]_{1\times n+1}$ and K is a $n+1 \times n+1$ symmetric matrix, which is the integral of the third derivative of the basis function. This K matrix can be expressed analytically, but as extra knots are continuously being added during the fitting

process, solving it using the Simpsons rule as in the previous section is much more computationally efficient.

This jerk minimization is very similar to the chord error minimization except this does not have to be separated into M intervals for the different chords and origins. The purpose of this minimization is to attain the distribution of the third derivative of the basis function and ensure that the combination with the fit parameter, the control points P, is minimized along the path. By adding the minimum jerk objective function, the overall minimization problem becomes:

$$\left.\begin{aligned}&\min_P\left[J_L + J_Q + J_K\right] = \min_P \frac{1}{2}\left[\gamma(q^t - \Phi P)^T(q^t - \Phi P) + \beta J_Q + \alpha(P)^T K(P)\right]\\&\text{Subject to: } L \cdot P = \xi\end{aligned}\right\} \tag{3.66}$$

where J_Q is as defined in Eq. (3.61) and α is the weight for the smoothing effect of the jerk minimization. The effects of the weights will be investigated in the subsequent section. The overall objective function along with the constraints result in a nonlinear optimization problem and it is solved using the Matlab optimization toolbox [42].

3.4.4 Simulation Results

The fitting algorithms are tested in simulations. The goal of the simulations is to fit the double NURBS spline toolpath to the given toolpath data points and try to achieve the specified geometrical tolerances and to verify the effects of the minimum chord error and minimum jerk objective functions. The machining tolerance for aeronautical parts is given around ~ 0.025 [mm][1].

The effects of the objective function coefficients are studied through simulations and fitting results on some test toolpaths. The first toolpath is a fan-shaped toolpath with 89 total data points for operating on an X-Y Table. Half of the toolpath was used, providing 45 discrete tool tip CC points and the overall size of the data points are about $370\ [\mathrm{mm}] \times 185\ [\mathrm{mm}]$. The fitting process proceeded until there were 20 control points. The first 3 control points and the last control points were fixed to provide C^2 continuity at the start. At every step of the fitting process, a node was added at the $\bar{u}_k$ value with the largest point error. The values for the objective function coefficients were varied and the results are analyzed.

First the effect of the chord error minimization objective function was investigated by comparing to just the least squares fit. Mean point error is calculated at the data point where the end point errors are zero. The Quadrature Chord error is calculated as described in the previous section where the error was integrated between every data point and the M integration segments were summed.

$$\text{Mean Point Error: } \frac{\sum_{k=0}^{M} e_k}{M-1} \tag{3.67}$$

Figure 3.13a). is the regular least squares fit without the influence of the chord minimizing coefficient and it has the lowest mean point error at the points. The effect of point error minimization is being subsidized for chord error minimization as β is increased. It can be seen that when $\beta = 10^2$, the mean point error has slightly increased from $1.107\ [\text{mm}]$ to $1.234\ [\text{mm}]$, however, the sum integration of chord error has decreased as intended from $2.125\ [\text{mm}]$ to $1.819\ [\text{mm}]$. As β was further increased to 10^3 and 10^4, the chord error continued to minimize, and the mean point error went down as well. The lengths of these curves were found using numerical integration and as β increased, the smaller length is indicating a smooth and less wavy curve. During the fitting process, when the chord error is lower, it is suspected that the search for the data point correspondent $\bar{u}_k$ had much better results. This effect can be seen at around the X-Y location, $(100, 100)$. In the regular least squares fit, there is an extra effort forcing the curve to pass through the point, causing large chord error and unnecessary oscillation to the curve. With the effect of chord error minimization in Figure 3.13d)., the curve can be seen to have much lower oscillation and very low chord error. The effect of the chord minimization coefficient can be seen to be able to significantly reduce chord error.

Next the effect of the jerk minimization coefficient is analyzed and presented. From Figure 3.14, it can be clearly seen that the mean error at the data points are increasing as the jerk minimization coefficient is increased. There is a significant jump in the data point mean error from $1.107\ [\text{mm}]$ to $1.546\ [\text{mm}]$ when the coefficient is introduced as $\alpha = 10^{-10}$ and an even larger increase to $3.458\ [\text{mm}]$ as α is increased to 10^{-6}. However, it is also clear that the curves for $\alpha = 10^{-8}$ and $\alpha = 10^{-6}$ are significantly smoother than the curve with just the least squares fit. The oscillation at the beginning of the curve due to incompetent knot distribution has been reduced and almost eliminated. The part of the curve going into the location with large

curvature changes where a lot of the data points gather can be seen to have a much better matching with the chord line. In Figure 3.15, the curvature and the third derivative graphs are shown. The curvature K of a general parametric curve is calculated from the following equation:

$$K(u) = \frac{\|C'(u) \times C''(u)\|}{\|C'(u)\|^3} \tag{3.68}$$

With just the regular least squares fit, $\gamma = 1;\ \alpha = 0;\ \beta = 0$, there are sharp curvature peaks seen throughout the entire curve, showing large variations even in regions where the curvature is not very high. When α was set to 10^{-10} and 10^{-8}, the overall variations in curvature decreased and the two peaks with maximum curvatures are also showing lower curvature values. It can be seen that when $\alpha = 10^{-6}$, visually the curve looks a lot smoother and a lot of the curvature variations are reduced, however it does seem that the two curvature peaks have increased in magnitude. While the jerk minimization objective function does not directly target the minimization of curvature, it is really showing an effect in decreasing the influencing of just minimizing at the data points. Focusing on just data point minimization with least squares will ultimately force the curve to artificially pass through the data points, and induce high oscillations to the curve fit geometrically.

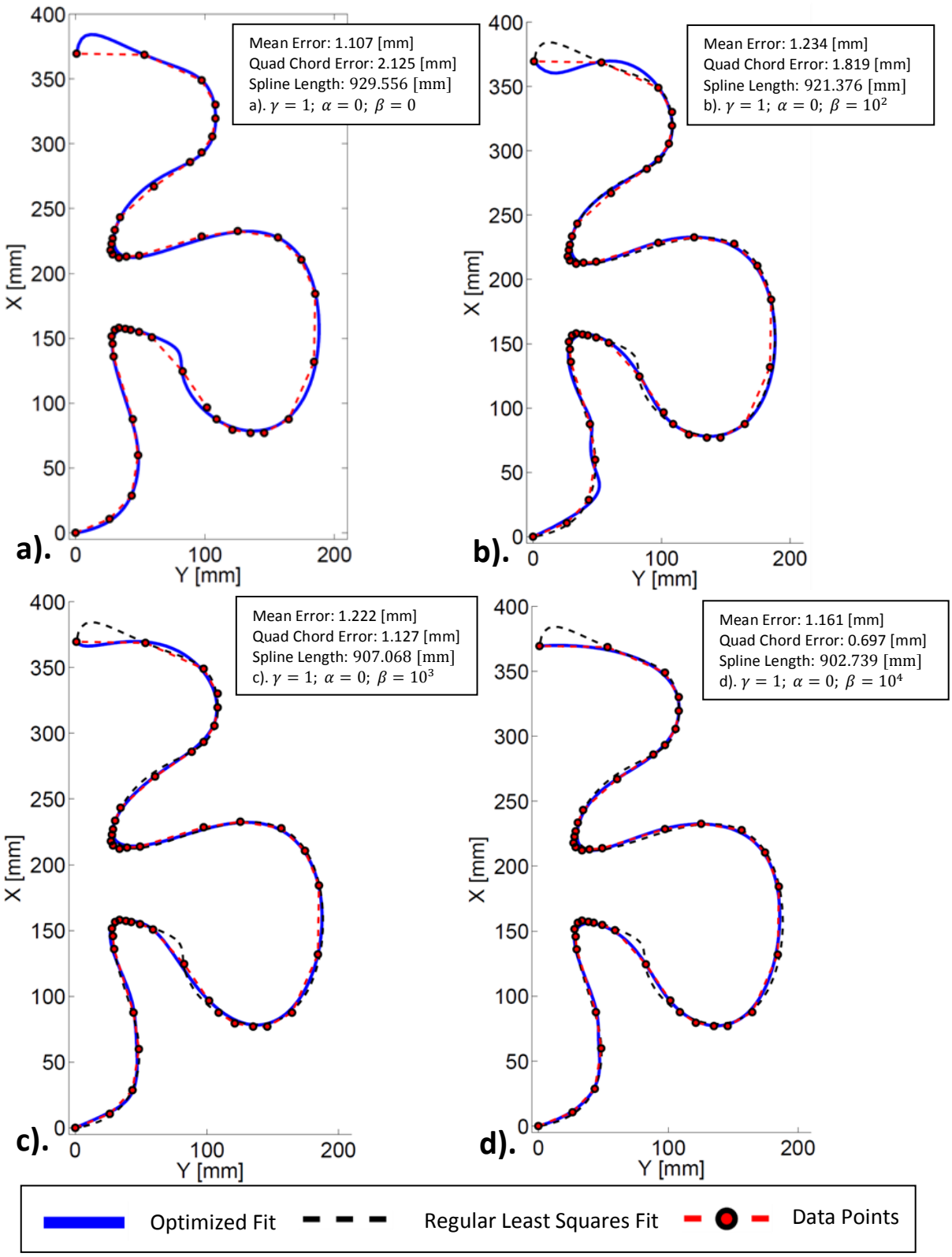


Figure 3.13: Chord Error Objective Comparison

a). Regular Least Squares, b-d). Optimized Chord Error Minimization

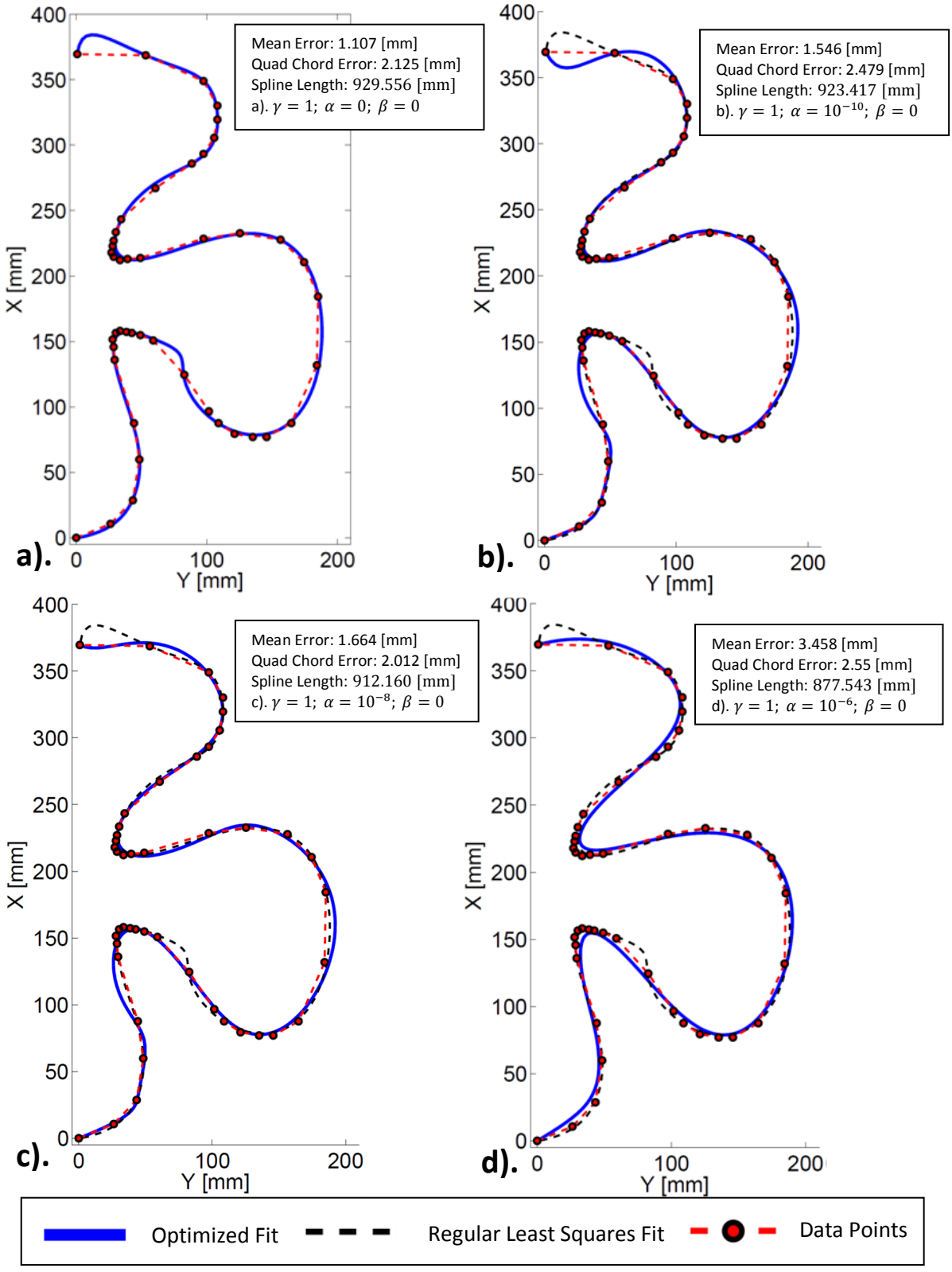


Figure 3.14: Jerk Objective Comparison

a). Regular Least Squares, b-d). Optimized Jerk Minimization

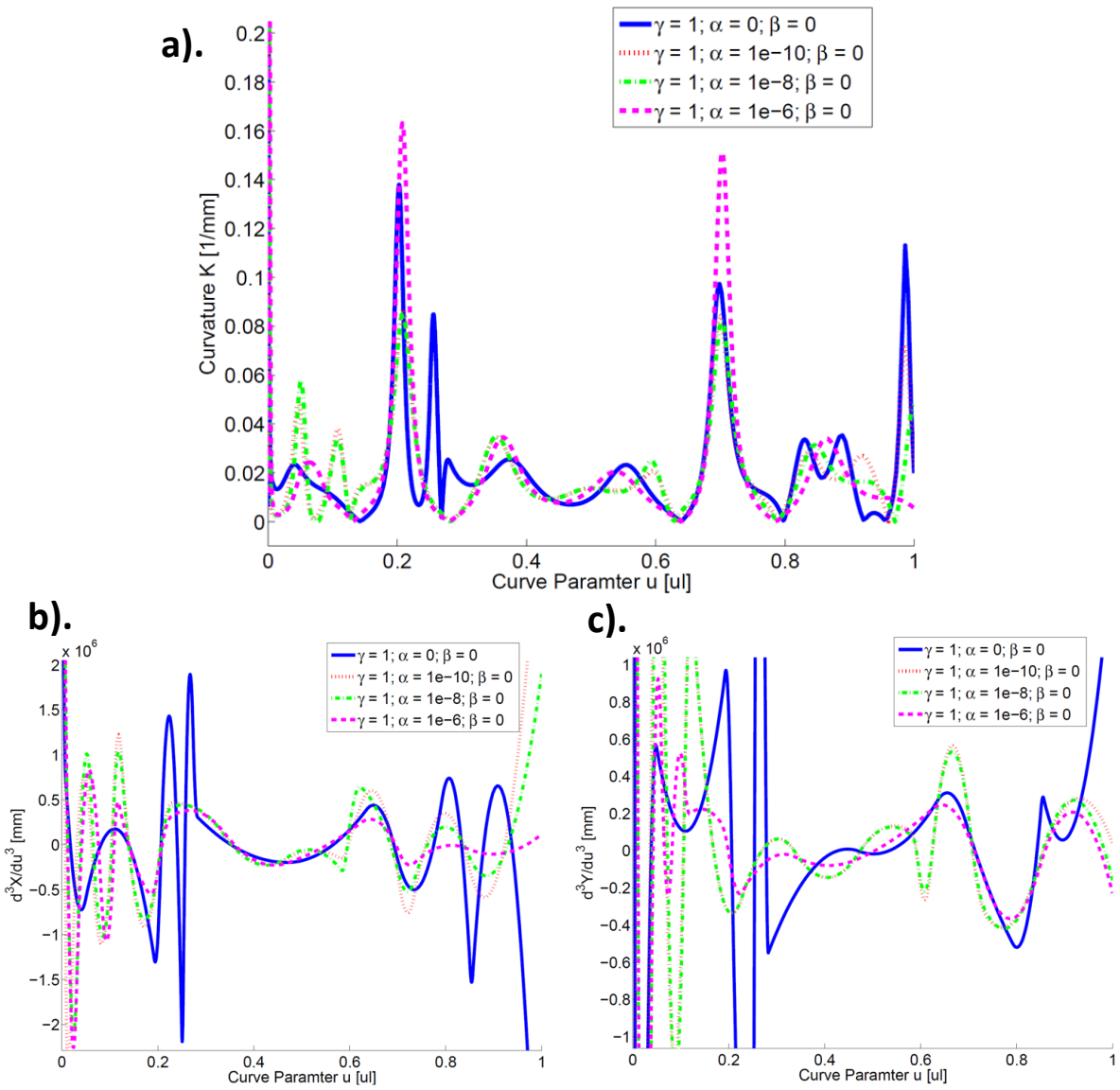


Figure 3.15: Jerk Minimized NURBS Fit

a). Curvature, b-c). 3rd derivative in X and Y

From the multi-objective simulations of the X-Y toolpath, it was shown that while the regular least squares fit provides good fitting results at the data points, showing the lowest mean point error; focusing at just the data points has an adverse effect of increasing the geometrical fluctuation of the NURBS curve. By adding the effects of chord error minimization and jerk minimization, the focus at just the data points is reduced, and a higher emphasis is placed on the curve as a whole; at the data points and in between the data points. From the figures shown, using a β value of 10^3 and an α value of 10^{-8} seems to provide decent results and are not in any way too aggressive for the fitting process. The following simulations are done

with three dimensional surfaces to verify the double spline fitting scheme. The following coefficients: $\gamma = 1;\ \alpha = 10^{-8};\ \beta = 10^{3}$, are used for the multi-objective fits.

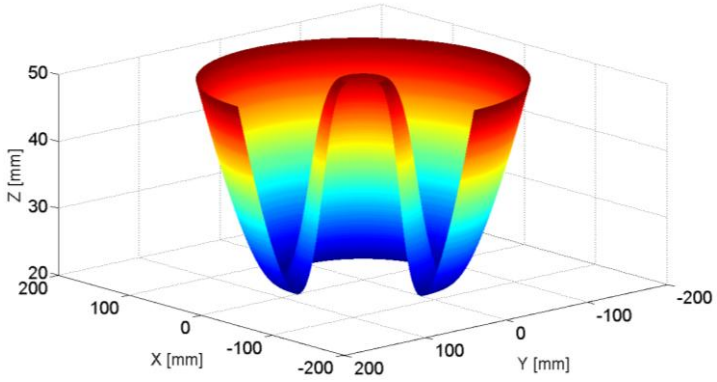


Figure 3.16: Parametric Ruled Surface Toolpath

This first test five-axis toolpath was taken and modified from Lo [39]. It is from a simple parametric revoluted surface shown in Figure 3.16 where a single strip was taken at $v = -0.98\,\pi$ as the toolpath.

$$\left.\begin{array}{c} x = (60u^3 - 90u^2 + 90u + 20)\cdot cos(v) \\ y = (60u^3 - 90u^2 + 90u + 20)\cdot sin(v) \\ z = (60u^3 - 90u^2 + 90u + 50) \\ 0 \le u \le 1.5, -\pi \le v \le 0.5\pi \end{array}\right\} \tag{3.69}$$

The orientation vector was taken as the normal of the surface:

$$n(u,v) = \left[\frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}\right]_{|v=-0.98\,\pi} \tag{3.70}$$

300 discrete CC toolpath data points were taken with natural spacing in the u direction, and the corresponding 300 discrete orientation data vectors were taken as the normal to the surface. These data points are shown in Figure 3.17. The double NURBS representation was used to represent the toolpath and the orientation. The following tolerances were used to terminate the fitting algorithm: $e_{max} = 0.02\ [\text{mm}]$ at the data points; integration of all the chord error $\sum Q_{max} = 0.005\ [\text{mm}]$; the maximum orientation $e_{\theta\,max} = 0.001\ [\text{rad}]$ at the data points.

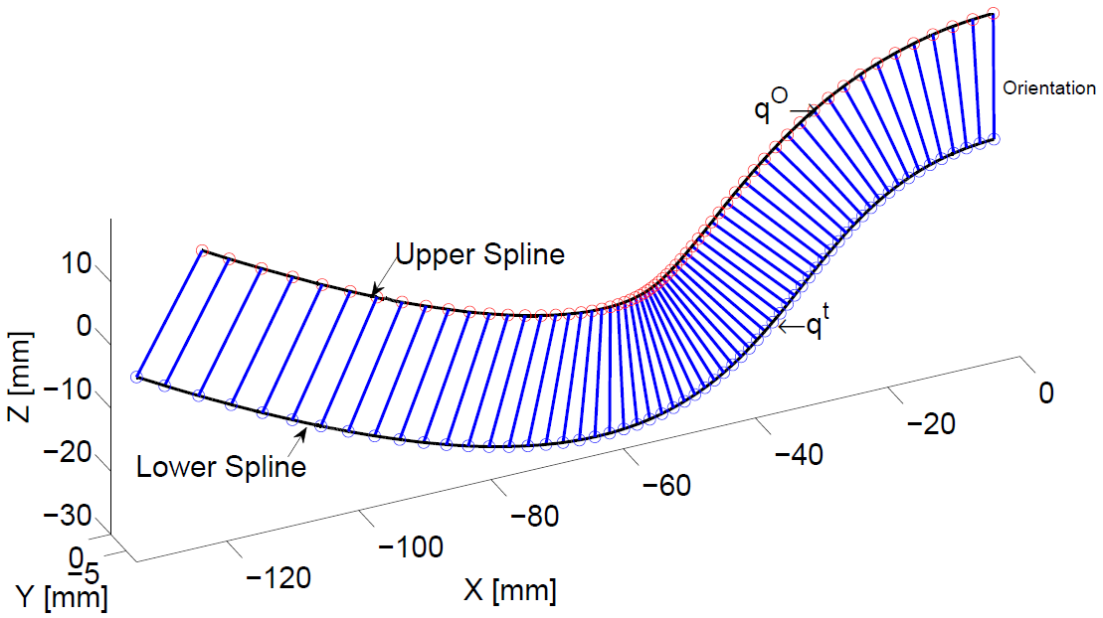


Figure 3.17: Five-Axis Toolpath 1 Data

Initially 16 control points were used to fit Toolpath 1 which resulted in a maximum of 0.358 [mm] of error at the data points, and 0.110 [mm] of quadrature chord error summed through the entire path on the lower spline. After 17 iterations, the prescribed tolerances are met and the resulting curve fit ended up with 33 control points as shown in Figure 3.18. From the fit, the interpolated toolpath position and orientation with respect to the curve parameter is shown in Figure 3.19. The errors between the fitted toolpath and the original data were evaluated at each of the data point and shown in Figure 3.20. The mean error was found to be 0.00448 [mm] for the data points with a maximum error of 0.0129 [mm], and 0.00450 [mm] of quadrature chord error was summed through the entire path. A maximum of 0.9001 [mrad] was found for the orientation error.

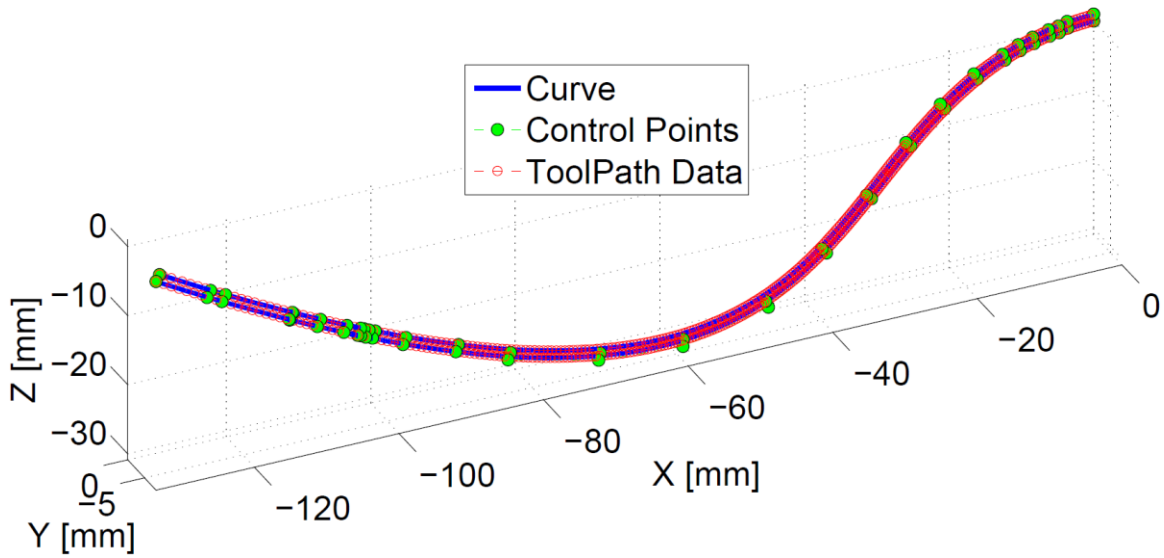


Figure 3.18: Toolpath 1 Fit

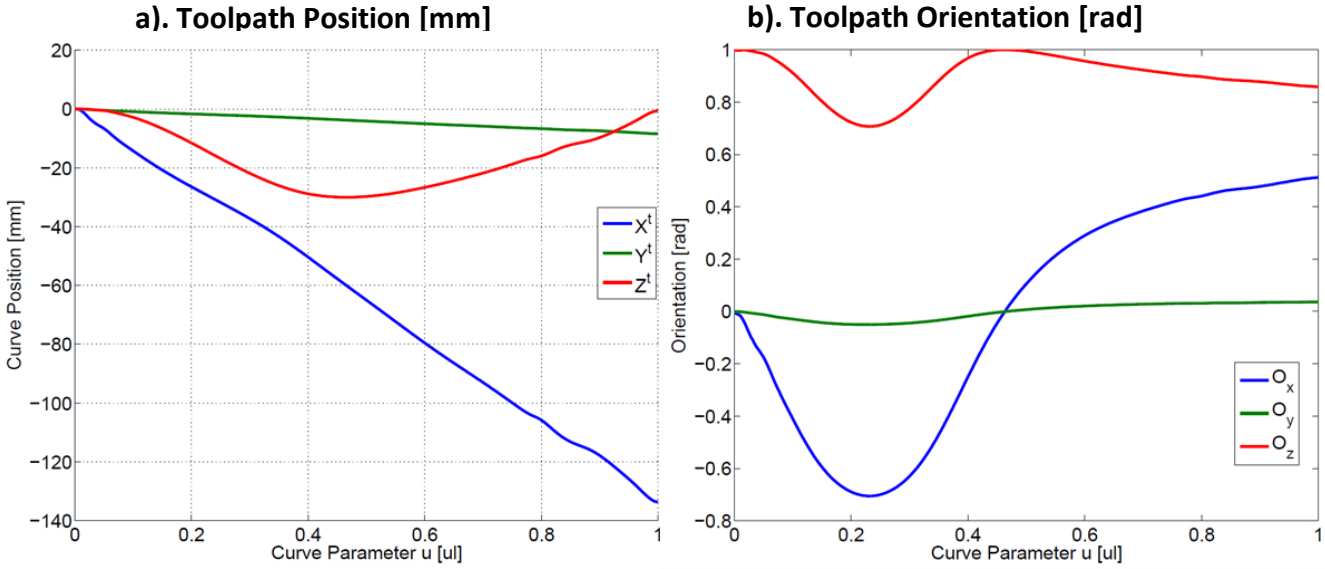


Figure 3.19: Toolpath 1 Fit Interpolated Position and Orientation

a). Position and b). Orientation

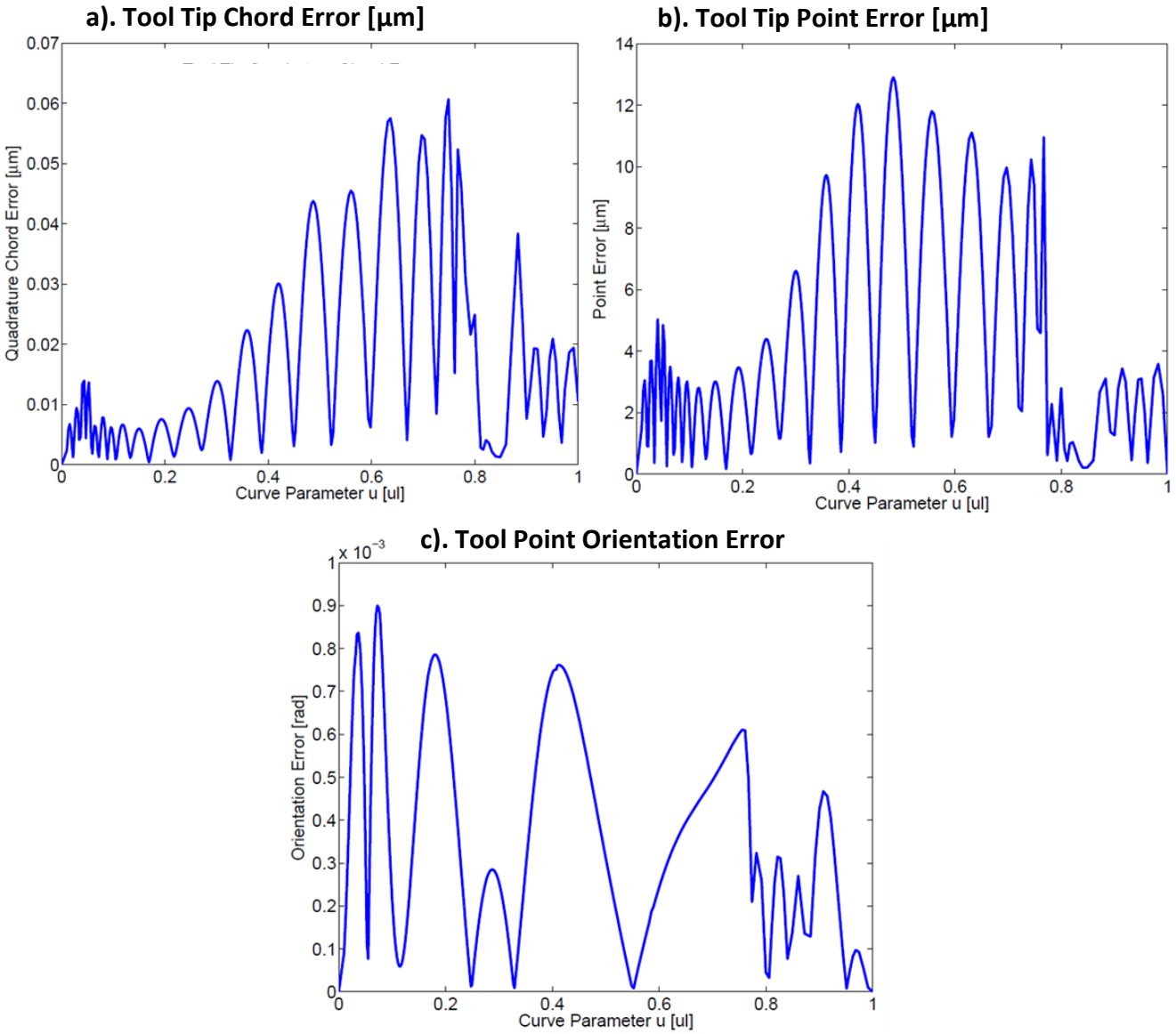


Figure 3.20: Toolpath 1 Fitting Errors

a). Position Local Chord Error, b). Position Point Error, c). Orientation Point Error

The reason for the variations seen in the quadrature chord error and the point error graph is due to the influence of the control points locally. It is impossible for a curve with only 33 control points to interpolate for 300 CC data points perfectly, therefore a tolerance scheme was set up to establish a band to keep all the data points relatively close to the curve. It is also notable that at locations with more control points, the general mean point error and chord error is lower and that these two graphs are very similar. For the orientation toolpath orientation,

there are notably two local peaks at around $u = 0.2$ and $u = 0.4$, where the orientation in z changes direction. Large orientation error peaks occurring at these locations are expected.

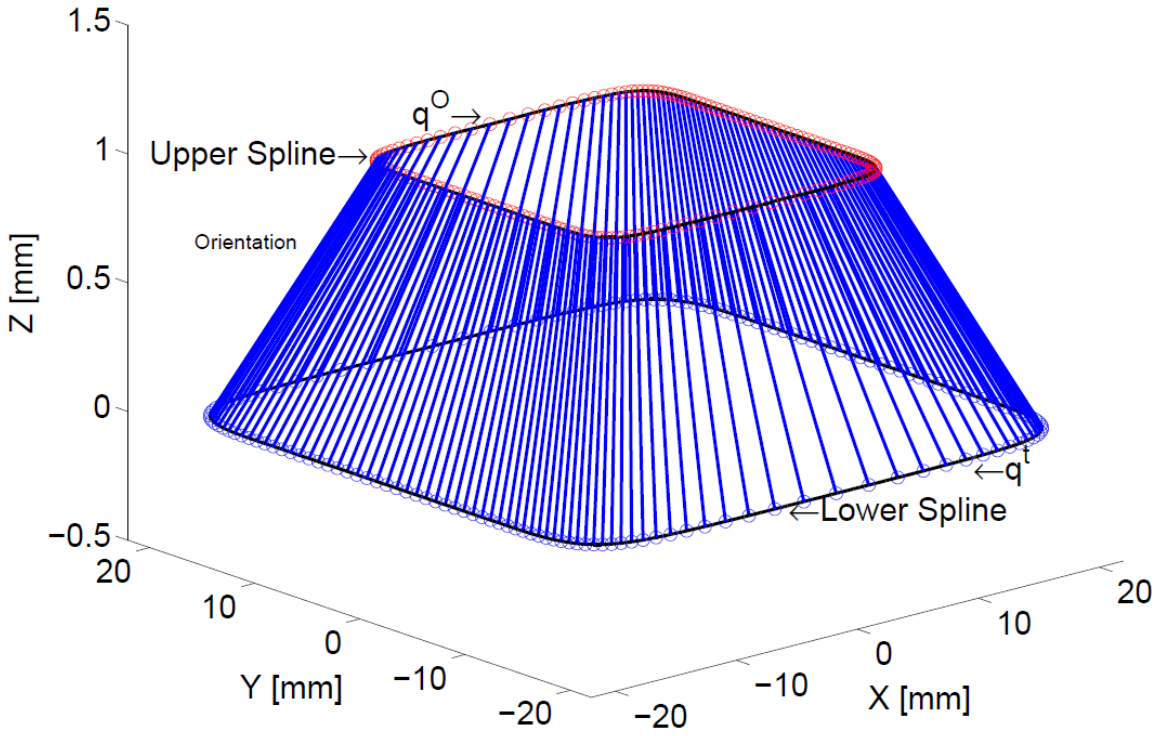


Figure 3.21: Five-Axis Toolpath 2 Data

The next test five-axis toolpath, as shown in Figure 3.21, is a closed square with rounded corners and a consistent change in orientation. 613 discrete CC toolpath data points were taken along with the corresponding 613 orientation vectors. The toolpath starts at the bottom center of the square and ends at the same location. A notable feature of the toolpath data is that the distributation of data points are the same for the bottom and the top and the same for the right and left sides, with a higher data point density for the right and left sides. By taking 5% of the number of data points as the number of initial control points, the fitting process started with 31 control points. The first 3 control points and the last control points were fixed. The same tolerances and the same multi-objective coefficients are kept the same as the previous test toolpath. From the result of the lower spline fit as shown in Figure 3.22, the distribution of the control points is consistent with the distribution of the data points, with more control points on the left and right sides. In general, the control points are distributed very well with respect to the length of the curve. The right side of Figure 3.22 is the zoomed in view showing how the local influence of the control points result in a wavy toolpath, which is why a lower number of

control point is desired in the global fit. This wavy toolpath is one of the disadvantage of the global fit, as a piecewise fit will easily allow the straight sections to remain straight. However, in general, the piecewise fit will result in a lot more control points throughout the path.

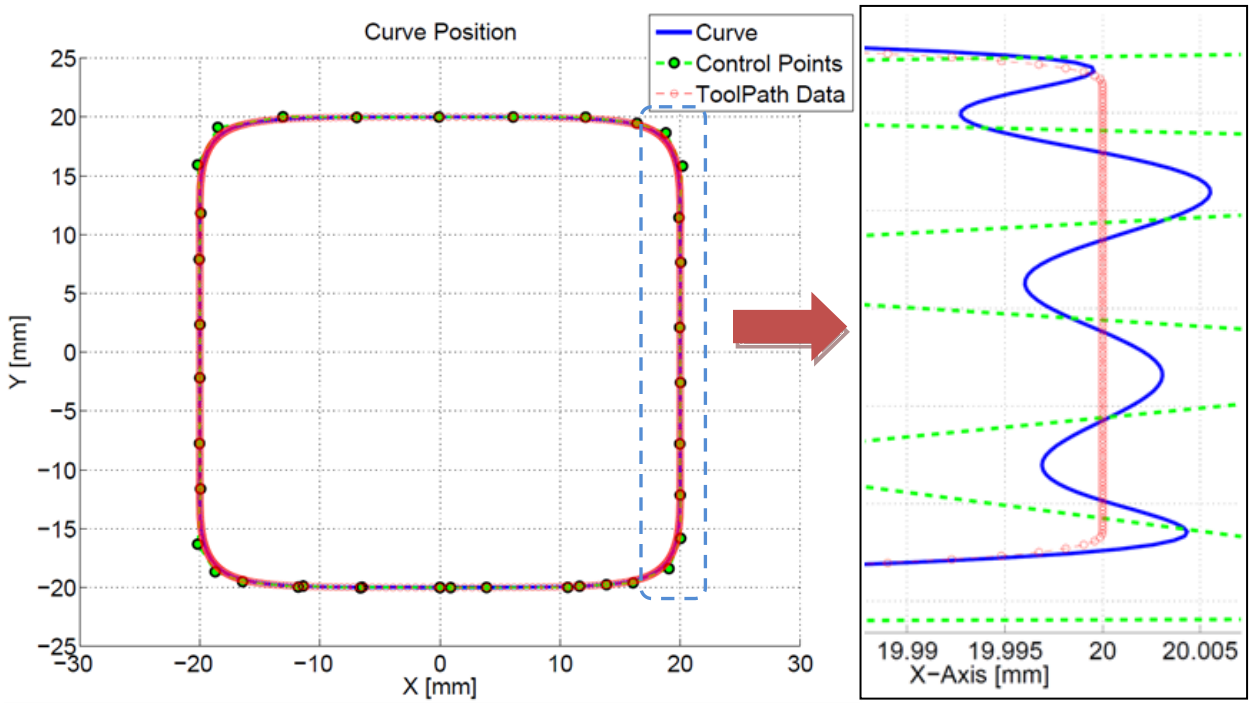


Figure 3.22: Toolpath 2 Lower Spline Fit

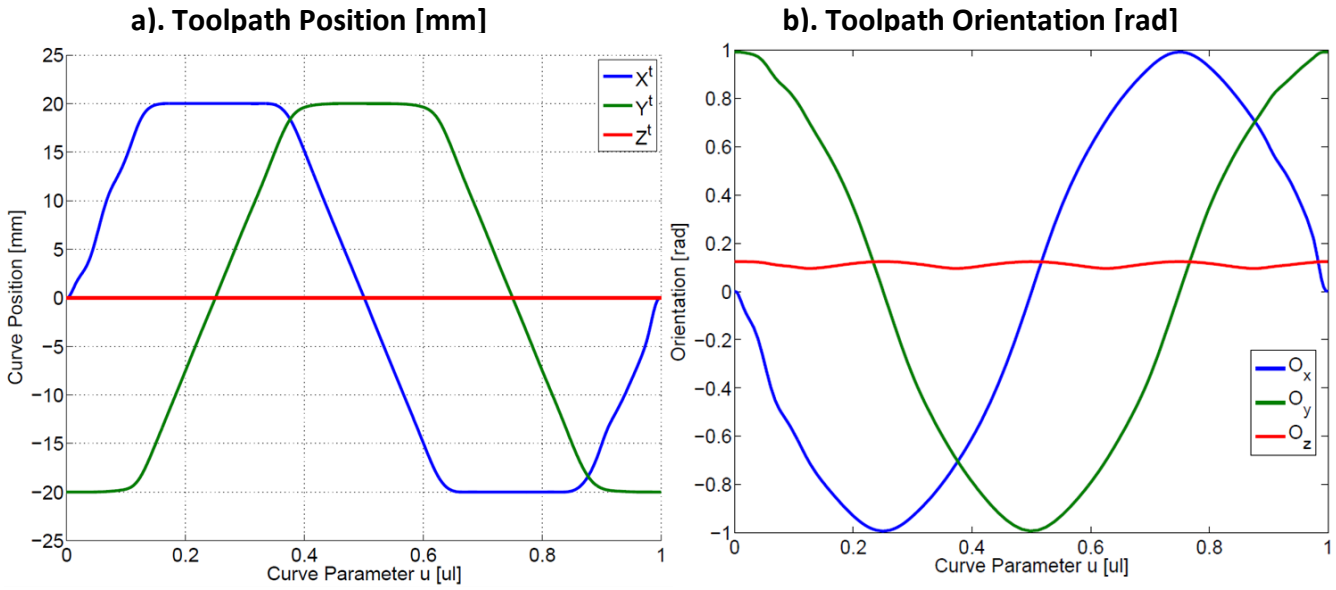


Figure 3.23: Toolpath 2 Fit Interpolated Position and Orientation

a). Position and b). Orientation

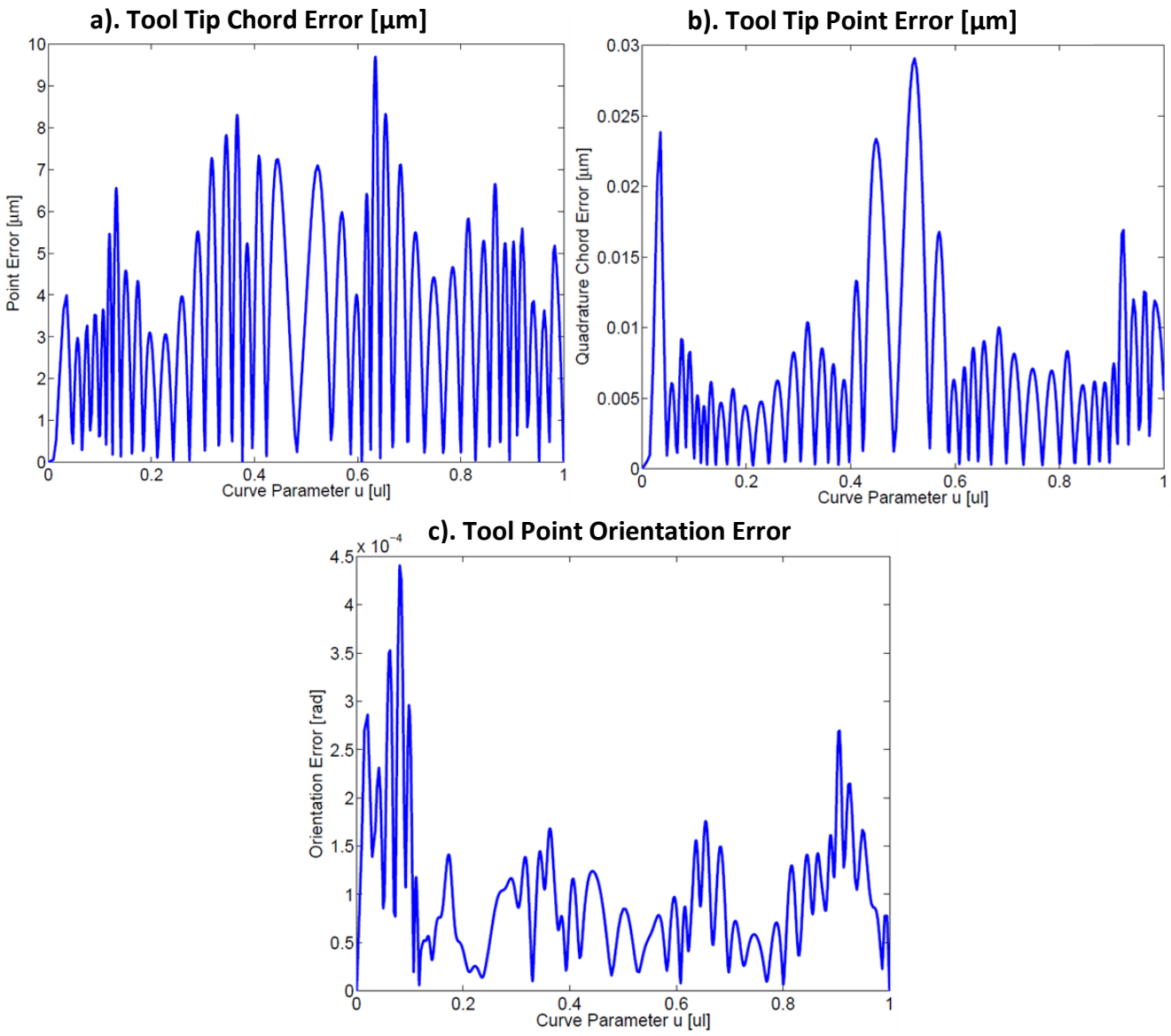


Figure 3.24: Toolpath 2 Fitting Errors

Figure 3.23 shows the interpolated position and orientation graphs from the toolpath fit. In terms of the fitting errors, there was an initial maximum point error of $0.3173\ [\mathrm{mm}]$, a mean point error of $0.0608\ [\mathrm{mm}]$ and a sum chord error integration of $0.0548\ [\mathrm{mm}]$ with 31 control points. The tolerances were met with 43 control points and the final maximum point error was $0.00970\ [\mathrm{mm}]$ as shown in Figure 3.24b)., with the mean point error of $0.00361\ [\mathrm{mm}]$ and $0.00343\ [\mathrm{mm}]$ for the chord error integration. The maximum orientation error was $0.441\ [\mathrm{mrad}]$ near the beginning of the curve as shown in Figure 3.24c). As this toolpath is very consistent, the mean point error and chord error integration are very well matched. The fluctuations within the point error and chord error graphs (Figure 3.24a,b).) are due to the

shaping of the control points locally; this can be clearly illustrated from the zoomed portion of right side of the tool path as shown in Figure 3.22. The larger chord error at the center of the curve is due to having less control points. The larger chord and orientation error at the beginning can be attributed to the continuity conditions set at the beginning by the initial control points, otherwise, throughout the curve, orientation error is maintained to be consistently low, since the orientation reference is quite consistent. Although it does seem that there is a slightly higher orientation error closer to the curve corners, this is expected.

From the simulation results, it was shown that by including the effects of chord error minimization, and jerk minimization, the overall fitting of the curve is smoother geometrically than just the least squares fitting algorithm where point error is the only minimization objective. It is concluded that by using a higher coefficient for the chord error objective than the point error objective, the overall curve can be minimized based on local chords, $(\bar{u}_{k+1} - \bar{u}_k)$. However, the influence of the point error objective cannot be removed completely, as the error at the data points are still very important. It has been shown that when the chord error is minimized, a better parameterization $\bar{u}_k$ can be found, allowing both the chord error and mean point error to reduce simultaneously. By using the following multi-objective coefficients of $\gamma = 1;\ \alpha = 10^{-8};\ \beta = 10^{3}$, the fitting algorithm is refrained from being too aggressive and can provide results that can achieve tight tolerances. These coefficient values may be dependent on the number of data points used in the fit as that would generally affect the range of curve parameter for the data points. These coefficient values were used for two test five-axis toolpaths using the double spline representation. Both the orientation and tool tip position can be simultaneously represented with a single curve parameter value.

3.5 Adaptive Quadrature Length

During the machining operation, the machine tool has to travel along the defined NURBS toolpath with the feedrate specified by the feed scheduler. The feed scheduler has to ensure that the tool travels smoothly within the toolpath, and decelerates to a full stop once the toolpath is complete. Any inaccuracy with the length estimate can result in overshooting or undershooting along the path, causing undesired discontinuities and jerk within the system. Therefore, it is extremely important to find the true path length along the generated NURBS toolpath. While the NURBS curve possesses many advantages in its curve shaping properties, the path length cannot be solved analytically using the NURBS curve's parameters; numerical

methods have to be used. In general, the displacement function S(u) of a parametric curve C(u) is the summation of the infinitesimally small segments along the curve, where the length of each segment is based on the Pythagorean Theorem.

$$S(u) = \int_a^b \frac{ds}{du} du = \int_a^b |C'(u)| du = \int_a^b \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}\, du \tag{3.71}$$

The Simpson's 3/8 Rule can be applied to this to obtain an approximation of the arc length within the region $u \epsilon [a \quad b]$ using a tolerance value denoted as ε_S as mentioned in previous sections. This ε_S value is especially important in bounding the functional variation of the integration function.

$$U = \left[\underbrace{u_0, \dots, u_p}_{\text{Repetitive p+1 Knots}}, u_{p+1}, \dots u_j, \dots, u_{m-p-1}, \underbrace{u_{m-p}, \dots, u_m}_{\text{Repetitive p+1 Knots}}\right], \tag{3.72}$$

To apply this to the NURBS curve, the internal knot spans $[u_p \quad u_{p+1}]$, … $[u_{m-p-1} \quad u_{m-p}]$ are used as the intervals. The intervals are solved from the left to right, whenever the tolerance ε_S given in Eq. (3.59) is not satisfied for any interval, the interval is sub-divided further, e.g. $\left[u_p \quad \frac{u_p+u_{p+1}}{2}\right]$ and $\left[\frac{u_p+u_{p+1}}{2} \quad u_{p+1}\right]$. By accounting for all the knot spans as intervals, this ensures that all the important features on the NURBS curve are considered. Since the knot spans determine which control points are contributing locally, if the integration intervals do not take the knots into account, the influence of some control points may be missed, due to the nature that numerical integrations only evaluate the intervals at specific locations. Extreme knot conditions where several knots are close together can be quite problematic if the knots are not considered. This can be demonstrated in the following example as shown in Figure 3.25:

Degree: 3

Control Points: P_0(0,0), P_1(10,0), P_2(20,0), P_3(50,0), P_4(0,10), P_5(0,30), P_6(20,50),P_7(40,50),

P_8(50,0), P_9(50,30), P_{10}(50,40), P_{11}(50,50)

Weights: [1 1 1 5 1 1 1 1 5 1 1 1]

Knot Vector U:

[0 0 0 0 0.40001 0.40002 0.40003 0.40004 0.40005 0.40006 0.40007 0.40008 1 1 1 1];

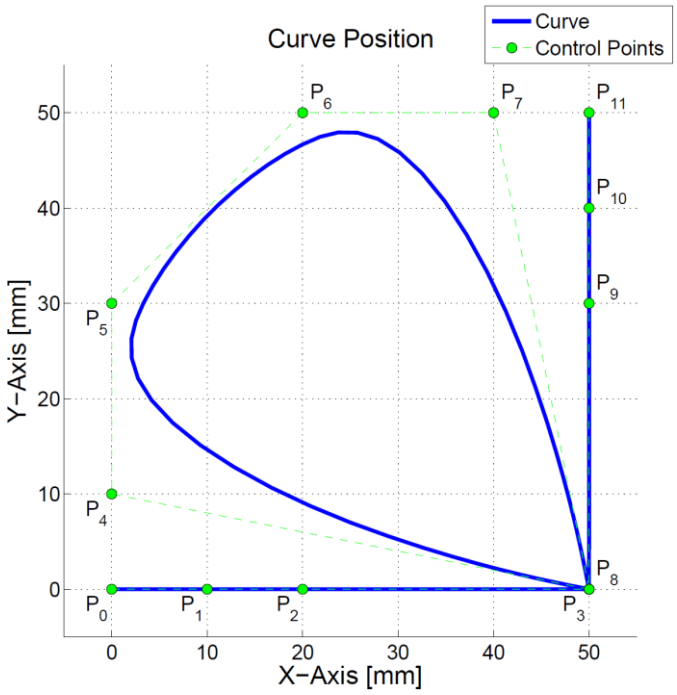


Figure 3.25: A Cubic NURBS Curve with Extreme Knot Conditions

Using the adaptive quadrature method that considers the knot intervals, the path displacements across the knot spans are shown:

S = [0 49.9996 51.9604 75.9953 111.6196 135.5532 171.1776 195.2125 197.1735 247.1732]

$247.1732\ [\mathrm{mm}]$ is the true path length that was calculated with a tolerance of $\varepsilon_S = 1e^{-8}\ [\mathrm{ul}]$. The following Table 3.1 shows the path length found with and without knot interval considerations with given tolerance:

Tolerance	With Knot Consideration	Without Knot Consideration
$\varepsilon_S[\mathrm{ul}]$	Path Length S [mm]	Path Length S[mm]
1	247.0847	99.8895
$1e^{-1}$	247.0829	100.0399
$1e^{-2}$	247.1751	247.1741

Tolerance	With Knot Consideration	Without Knot Consideration
$1e^{-3}$	247.1739	247.1732
$1e^{-4}$	247.1733	247.1732
$1e^{-5}$	247.1732	247.1732

Table 3.1: Path Length with Varying Adaptive Quadrature Tolerance

When the tolerance is large, the path length cannot be adequately calculated without knot considerations and a length of roughly $100\ [\mathrm{mm}]$ is found. This $100\ [\mathrm{mm}]$ is the sum of the $50\ [\mathrm{mm}]$ lines at the beginning and end of the NURBS curve, the middle fan portion of the path is completely left out of consideration. The characteristic of the knot vector shows that it is a degree 3 curve, with 4 repeated knots at the front and end with all the internal knots being extremely close together. Due to the property that a control point P_i is only active within the knot range of $[u_i \quad u_{i+p+1}]$, only the first 4 control points are relevant in the large range of $[0 \quad 0.40004]$, while only the last 4 control points are relevant in the large range of $[0.40005 \quad 1]$, leaving the middle 4 control points only relevant within the confined range of $[0.40001 \quad 0.40008]$. Figure 3.26 shows the basis functions for this NURBS curve.

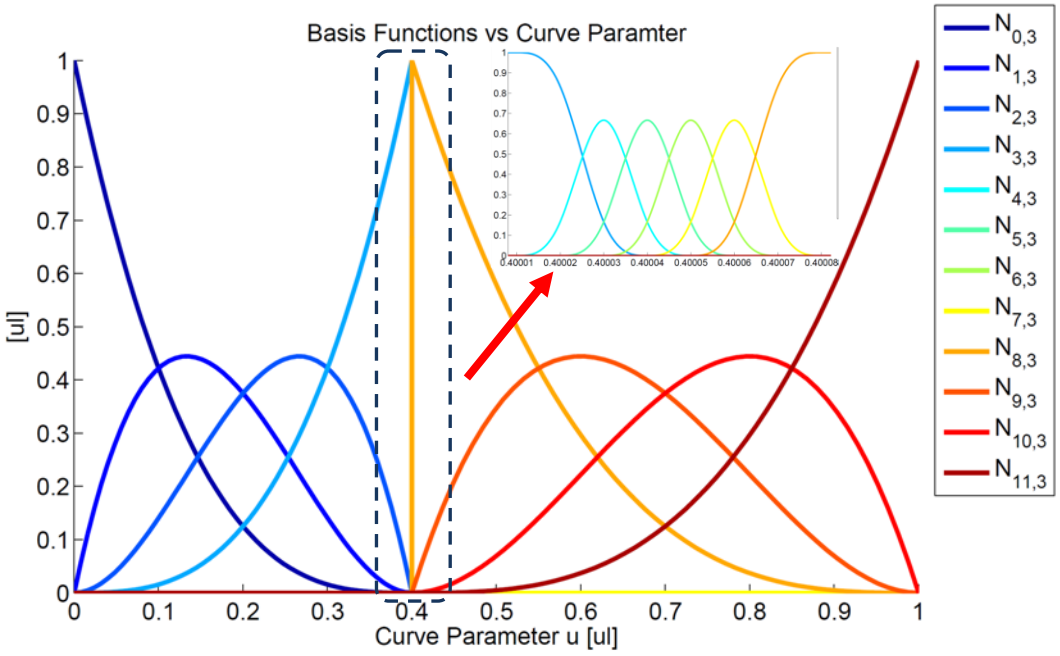


Figure 3.26: Basis Functions with Extreme Knot Conditions

From Table 3.1, it can be seen that it is possible to calculate the true length even without knot interval considerations by having tighter tolerances. However tolerances cannot be increased indefinitely due to computational constraints. At first glance of the table, it may seem that the lengths found without knot interval consideration converges earlier with lower tolerance than considering the knots, however, this is not true.

For instance when $\varepsilon_S = 1e^{-2}$ [ul], in the case of knot consideration, this tolerance s applied to every knot interval $[0 \quad 0.40001], [0.40001 \quad 0.4002], \cdots$, and so on. In comparison to the case of without knot considerations, the tolerance of $1e^{-2}$ [ul] is applied to $[0 \quad 1]$, where the tolerance is not met and the interval subdivides into $[0 \quad 0.5]$, $[0.25 \quad 0.5]$, $[0.375 \quad 0.5]$, $[0.375 \quad 0.4375]$ and so on, until $[0.40001 \quad 0.40002]$. When the division finally reaches $[0.40001 \quad 0.40002]$, the division has already been subdivided 16 times, which means a 2^{16} times tighter tolerance than the base tolerance. In other words, given the same base tolerance, the tolerance is actually a lot tighter when not considering knot intervals, which in turn means longer computation time.

In addition to just computing the total path length of the NURBS Curve using this method, the intermediate subdivisions are also recorded as data of (u_i, s_i) along the path. Figure 3.27 shows the path length of the NURBS curve in terms of the curve parameter u. There is a large jump in path length at $u = 0.4$ due to the extreme knot conditions where all the control points pertaining to the fan are located. The data of (u_i, s_i) will be used later for the Correctional Polynomial interpolation method.

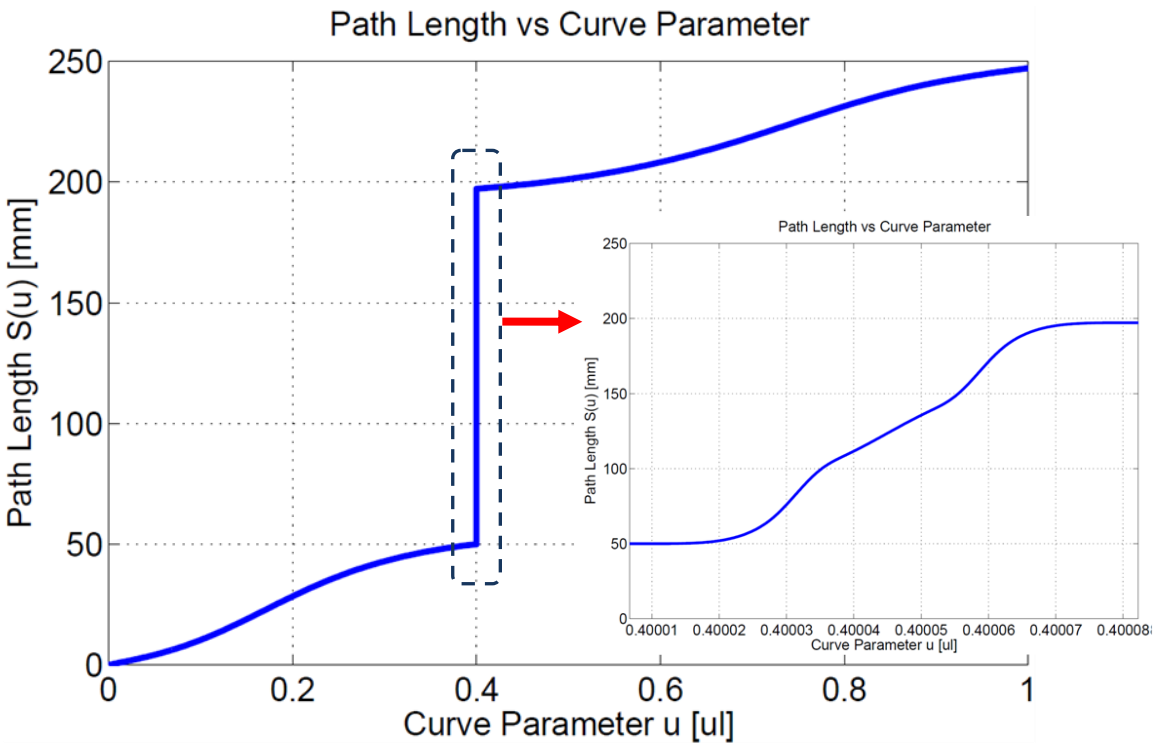


Figure 3.27: Path Displacement vs. Curve Parameter with Extreme Knot Conditions

3.6 Conclusion

A smooth and continuous toolpath is essential for producing high quality parts for High Speed Machining. In this chapter, the non-uniform rational B-spline (NURBS) curve was implemented to provide a toolpath fit on a set of discrete toolpath data that is exported from CAD systems. The advantages of the NURBS curve over traditional linear and circular segmentation methods were discussed, and some common techniques for shaping the NURBS curve were presented.

The general scheme of using the least squares fit to globally approximate the curve to a set of discrete toolpath data was borrowed from Piegl [47] and was presented. To overcome the irregularities and fluctuations of fitting the curve only to the data points, the chord integration error was introduced to quantify the chord error in between the data points. An objective based on local chord was presented and integrated along with the objective to minimize the overall jerk of the curve fit that was introduced by Sencer [50]. Since the distribution of the B-spline basis functions is known entirely with a given knot vector, using a numerical integration technique to efficiently compute for the overall influence of each of the control points with

respect to the local chord region allows the set up of a multi-objective fitting function that takes full advantage of the known basis function. The effects of the multi-objective function were shown through simulations and were implemented on test five-axis toolpaths. The double spline scheme introduced by Langeron et al. [31] was implemented to simultaneously represent both the tool tip position as well as the tool orientation. The multi-objective fitting algorithm was integrated with this toolpath representation scheme and the simulation results for five-axis toolpaths were presented.

The adaptive quadrature length method presented by Lei et al. [32] is integrated in this chapter to find the true length of the NURBS toolpath. Using this method of finding the displacement along the toolpath, provides a set of curve parameter and displacement (u_i, s_i) data points which are essential for the interpolation of the toolpath in real time.

Chapter 4 Real Time Interpolation and Feedrate Profiling

4.1 Introduction

From a set of discrete data point of a CAD model, Chapter 3 demonstrated that a smooth NURBS curve can be fitted and used to accurately represent the toolpath. This chapter presents the process and constraints involved in scheduling a feedrate profile for a five-axis CNC system and the methods available to interpolate the fitted toolpath in real time.

In real time interpolation, feed fluctuation is a problem that arises from incorrectly interpolating the curve parameter to match the desired travel displacement for the time step. The "Feed Correction Polynomial method" was developed by Erkorkmaz and Altintas [19] and was implemented on B-spline toolpaths for feed consistent interpolation. Similar concepts were developed by Lei [32], where he applied the adaptive quadrature length method to find the true length of the toolpath and fitted the data with a cubic hermite spline to acquire an inverse length function. These concepts were extended further by Heng and Erkorkmaz [22] to connect multiple segments of the polynomial together in order to express the large changes in curve parameter within small arc displacement changes without the need to increase the order of the polynomial. The "Feed Correction Polynomial method" is implemented and is compared to the widely used "Taylor Series Expansion" interpolation method.

A continuous feedrate modulation strategy is applied to the 5-axis CNC machine. The 5-axis machine kinematics is presented and considerations to the axis drive's kinematic limits such as velocity, acceleration, and jerk, as well as geometrical constraints for the toolpath such as chord error and curvature are included to schedule a smooth feedrate profile.

4.2 Interpolation

In real time interpolation, the toolpath is converted into a machine trajectory. For an arc-length parameterized toolpath such as the linear segments used in linear interpolation $C(s)$, where s is the arc-length displacement, L is the length of the line, x_s and x_e are the starting and ending position respectively, the position given by the linear toolpath is the following:

$$C(t) = C(s(t)) = x(s(t)) = x_s + \frac{(x_e - x_s)}{L} \cdot s(t) \tag{4.1}$$

Let the sampling interval of the CNC control loop be denoted as T_s, then time can be expressed as $t = kT_s$ where k is an integer value representing the k^{th} time step. In trajectory generation, a reference command of $x_{ref}(k)$ is generated at every time step:

$$\left.\begin{aligned} C\big(s(k)\big) = x_{ref}\big(s(k)\big) = x_s + \frac{(x_e - x_s)}{L} \cdot s(k) \\ s(k) = \sum_{k=0}^{k} \Delta s(k) \cdot T_s \end{aligned}\right\} \tag{4.2}$$

where $\Delta s(k)$ is the desired arc length increment at the k^{th} step. For arc-length parameterized toolpaths such as the linear and circular toolpaths, the machine axis increment at every time step can be easily calculated. Toolpaths that are not parameterized by arc-length but rather, parametrically, such as the NURBS curve $C(u)$, requires an additional conversion where the curve parameter $u(s)$ needs to be mapped from arc-length displacement to get $C\left(u\big(s(t)\big)\right)$. The challenge is in the mapping between u and s to satisfy the desired arc length increment at every time step. This is the process of interpolation as shown in Figure 4.1, where the next curve parameter value of $u(k+1)$ is computed to satisfy $\Delta s(k)$. A straight forward approach in computing the curve parameter u, where u is defined within the knot vector interval of $u \in [u_0 \quad u_m]$, is the natural interpolation. This is the method of linearly incrementing the curve parameter in proportion to the ratio of arc length increment $\Delta s(k)$ to the total arc length S_{total}.

$$\Delta u(k) = \frac{(u_m - u_0)}{S_{total}} \cdot \Delta s(k) \tag{4.3}$$

The linear relationship used in natural interpolation is mostly only valid where the spline is arc-length parameterized, and in general, this is not valid for the entire spline toolpath. Using an improperly calculated curve parameter, can cause undesirable feed fluctuations, resulting in higher accelerations and jerk components on the axis drives. While the high frequency components may not be tracked by the limited performance of the CNC control, it can lead to the excitation of the CNC machine tool's natural modes for the structure or the drives. In addition, having aggressive accelerations for the motor drives can result in motor drive saturations, further increasing the vibrations at the machine tool. All these effects are huge detriments for the proper tracking of the reference path, and will manifest as rough surface finishes, vibration marks, poor geometrical tolerances and reduced overall machine life.

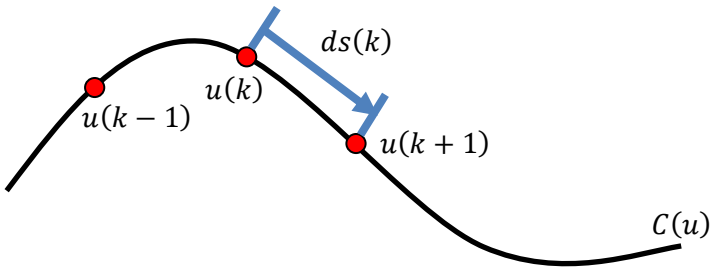


Figure 4.1: Interpolation

4.2.1 Taylor Expansion

The method of Taylor series interpolation was first introduced by Shipitalni, et al. [55] to estimate the next curve parameter at every time step interval with the current curve parameter. It is based on the assumption that when the arc-length increment is small, and that the curve is parametrically smooth, i.e. the first derivative of the curve is defined and continuous, the tangent direction of the curve is sufficient to compute for the incremental value in curve parameter. The toolpath is defined along the lower tool tip spline as $C(u) = [x(u) \quad y(u) \quad z(u)]$ and the feedrate profile is denoted as $V = V(t)$. Since feedrate is defined as the rate of change in tool tip displacement with respect to time, it can be expanded as the following:

$$V(t) = \frac{ds}{dt} = \frac{ds}{du}\frac{du}{dt} \Rightarrow \frac{du}{dt} = \frac{V(t)}{ds/du} \tag{4.4}$$

The change in displacement for a parametric curve is the following:

$$\left.\begin{aligned} ds &= \left\|\frac{dC(u)}{du}\right\| du = \sqrt{\left(\frac{dx(u)}{du}\right)^2 + \left(\frac{dy(u)}{du}\right)^2 + \left(\frac{dz(u)}{du}\right)^2}\, du \\ \frac{ds}{du} &= \|C'(u)\| \end{aligned}\right\} \tag{4.5}$$

This gives the 1st order Taylor Expansion as:

$$\left.\begin{aligned} \frac{du}{dt} &= \frac{V(t)}{\|C'(u)\|} \\ u(k+1) &= u(k) + \frac{V(k)\cdot T_s}{\|C'(u(k))\|} \end{aligned}\right\} \tag{4.6}$$

When the curvature of the curve is low, the first order Taylor approximation is generally adequate; however, when the curvature is large, then higher order approximation becomes necessary:

$$u(k+1) = u(k) + T_s\left(\frac{du}{dt}\right)_{|u=u(k)} + \frac{{T_s}^2}{2}\left(\frac{d^2u}{dt^2}\right)_{|u=u(k)} + H.O.T. \tag{4.7}$$

The following is the second order Taylor Expansion:

$$\left.\begin{aligned} \frac{d^2u}{dt^2} &= \frac{d\left({du}/{dt}\right)}{du}\frac{du}{dt} = -V(t)\frac{(C'(u)\cdot C''(u))}{\|C'(u)\|^3}\frac{V(t)}{\|C'(u)\|} \\ u(k+1) &= u(k) + \frac{V(k)\cdot T_s}{\|C'(u(k))\|} - \frac{(V(k)\cdot T_s)^2}{2}\frac{(C'(u(k))\cdot C''(u(k)))}{\|C'(u(k))\|^4} \end{aligned}\right\} \tag{4.8}$$

In order to apply the Taylor Expansion for real time interpolation, the derivatives of the NURBS curve have to be evaluated at $u(k)$ at every time step of the interpolation. Due to the recursive nature of the B-spline basis function, especially in the evaluation of derivatives, the computational load becomes more significant when the degree of the curve, the number of control points and the number of axis are increased.

Both the 1st and 2nd order Taylor Expansion methods are widely used. However, this method is not reliable when the feedrate is varying or if there are sharp curvature changes along the toolpath. When the feedrate is varying, such as in a typical kinematic profile consisting of sections of speeding up, maintaining constant feed and then slowing down at the end of the path, the total number of time step is already determined. However, as there is a slight discrepancy between the actual displacement and the desired displacement at every interpolation step, this discrepancy is being accumulated. Towards the end of the curve, when the total number of time steps has been reached, the curve parameter may exceed or fall short of the available knot vector $u \,\epsilon\, [u_0 \quad u_m]$ range. In addition, when there are sharp curvature changes along the toolpath, by only computing for the derivatives at the current location, the sharp change that occurs just slightly behind the current location may not be captured by the approximation. Lei et al. [33] presented an improved Taylor expansion method that is more robust against extreme knot distributions where large changes on the curve can occur across very short intervals of the curve parameter. In situations where the knots are close together, important features from local control points can be missed entirely if the computed Δu from

Taylor expansion passes through the knots and into different knot spans. In order to avoid this, it is necessary to check at the computation cycle whether the new curve parameter value of $u(k+1)$ is still within the same knot interval as $u(k)$. By having the displacement for all the knot spans pre-calculated using the adaptive quadrature length in pre-processing, whenever a knot span is passed through, the remaining path length, l_i, from the previous knot span has to be accurately calculated and subtracted from the total desired displacement of the current step to get a residual desired displacement, $\Delta s^*(k) = \Delta s(k) - l_i$. If the residual desired displacement is within the path length of the next knot span, l_{i+1}, then the new curve parameter, $u(k+1)$, will be calculated within this knot. If the residual desired displacement is greater than the path length of the next knot span, then the knot span after that is taken, once again updating the residual desired displacement $\Delta s^*(k) = \Delta s^*(k) - l_{i+1}$. This knot span search continues until the available path length from a knot span can accommodate the remaining residual desired displacement. While doing a knot for knot check as described will increase the robustness of the Taylor Expansion, it will further increase the computational cost of the method, making it more difficult for real time implementations.

4.2.2 Feed Correction Polynomial

From the NURBS toolpath formulation, it is generally not possible for arc-length parameterization. In order to face the challenge of unwanted feedrate fluctuations that is inherent to non-arc-length parameterized parametric curves, the feed correction polynomial interpolation method was introduced by Erkorkmaz and Altintas [19]. Following this work, Heng and Erkorkmaz [22], adopted the adaptive quadrature length method and the multiple inverse length function method as proposed by Lei et al. [33], and convincingly capture the mapping between arc length displacement and curve parameter.

The scalar valued function $u = f(s)$ is established as a 7th order polynomial to map the desired arc length displacement, s to the curve parameter, u. In chapter 3, the adaptive quadrature method was used to find the approximate path length along the toolpath. The method resulted in a set of (u_i, s_i) data, as well as the final path length. The 7th order polynomial is fitted to approximate the data such that the boundary conditions on point position, and the first and second derivatives at the start and end points can be imposed, which would require at least 5th order. The original reasoning introduced by Erkorkmaz and Altintas [19] for using 7th order is to avoid numerical problems while characterizing the nonlinear

relationship of the (u_i, s_i) data, however, for each spline toolpath segment, only a single 7th order polynomial was fitted using least squares. The reasoning used by Heng and Erkorkmaz [22] to have the additional two degrees of freedom to better approximate the data, however, as multiple polynomials are used when a single polynomial fails to capture the data, the additional freedom is not totally necessary.

The conditions used in determining whether the polynomial is properly capturing the curve parameter and arc-length relationship are that the polynomial must be monotonically increasing (curve parameter can only increase) and that the mean square error between the data points and the fit must stay below a specified tolerance value. Whenever these conditions fail, the data points are split into two halves to be individually fitted. The splitting continues until all the polynomials meet the two conditions. This adaptive splitting method is preferred over increasing the order of the polynomial.

Each curve is fitted using the least squares method to the set of $(u_i, s_i),\ i = 0, \cdots, N_\Sigma$ data that was computed using the adaptive quadrature length method. To avoid any ill-conditioning, the parameter range is normalized by total arc length in the given data.

$$\sigma_i = \frac{s_i - s_0}{s_{N_\Sigma} - s_0}, \quad i = 0, \cdots, N_\Sigma \tag{4.9}$$

The feed polynomial and its derivatives in terms of the normalized arc-length parameter σ and the normalized coefficients are the following:

$$\left.\begin{aligned} \hat{u} &= a_0\sigma^7 + a_1\sigma^6 + a_2\sigma^5 + a_3\sigma^4 + a_4\sigma^3 + a_5\sigma^2 + a_6\sigma^1 + a_7 \\ \hat{u}_s &= \frac{d\hat{u}}{d\sigma}\frac{d\sigma}{ds} = \frac{7a_0\sigma^6 + 6a_1\sigma^5 + 5a_2\sigma^4 + 4a_3\sigma^3 + 3a_4\sigma^2 + 2a_5\sigma^1 + a_6}{s_{N_\Sigma} - s_0} \\ \hat{u}_{ss} &= \frac{42a_0\sigma^5 + 30a_1\sigma^4 + 20a_2\sigma^3 + 12a_3\sigma^2 + 6a_4\sigma^1 + 2a_5}{\left(s_{N_\Sigma} - s_0\right)^2} \end{aligned}\right\} \tag{4.10}$$

where $\hat{u}$ is the curve parameter prediction calculated using feed correction polynomial. A least squares fit is obtained over the normalized arc length displacement and curve parameter values:

$$\left.\begin{aligned}\begin{bmatrix}\hat{u}_0\\ \hat{u}_1\\ \hat{u}_2\\ \vdots\\ \hat{u}_{N_\Sigma}\end{bmatrix} &= \begin{bmatrix}0 & 0 & 0 & \cdots & 1\\ {\sigma_1}^7 & {\sigma_1}^6 & {\sigma_1}^5 & \cdots & 1\\ {\sigma_2}^7 & {\sigma_2}^6 & {\sigma_2}^5 & \cdots & 1\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & 1 & 1 & \cdots & 1\end{bmatrix}\begin{bmatrix}a_0\\ a_1\\ a_2\\ \vdots\\ a_7\end{bmatrix}\\ \hat{u}_{(N_\Sigma+1)\times 1} &= \Phi_{(N_\Sigma+1)\times 8}\theta_{8\times 1}\end{aligned}\right\} \tag{4.11}$$

and the objective function to minimize is the following:

$$J = \frac{1}{2}e^T e = \frac{1}{2}(u-\Phi\theta)^T(u-\Phi\theta) \tag{4.12}$$

where $u = [u_0 \quad u_1 \quad \cdots \quad u_{N_\Sigma}]^T$. The first and second derivatives constraints are imposed to the beginning and end of the polynomial and are calculated as the following:

$$\left.\begin{aligned}u_s &= \frac{du}{ds} = \frac{1}{\sqrt{\left(\frac{dx(u)}{du}\right)^2 + \left(\frac{dy(u)}{du}\right)^2 + \left(\frac{dz(u)}{du}\right)^2}} = \frac{1}{\|C'(u)\|}\\ u_{ss} &= \frac{d^2u}{ds^2} = -\frac{C'(u)\cdot C''(u)}{\|C'(u)\|^4}\end{aligned}\right\} \tag{4.13}$$

Then denoting the starting point as $u^{init} = u_0$, the ending point as $u^{final} = u_{N_\Sigma}$, the first and second derivative boundary conditions for each polynomial are $u_s^{init} = u_{s_{|u=u^{init}}}$, $u_s^{final} = u_{s_{|u=u^{final}}}$ and $u_{ss}^{init} = u_{ss_{|u=u^{init}}}$, $u_{ss}^{final} = u_{ss_{|u=u^{final}}}$, the constraints matrix is written as the following:

$$\left.\begin{aligned}\begin{bmatrix}u^{init}\\ \left(s_{N_\Sigma}-s_0\right)u_s^{init}\\ \left(s_{N_\Sigma}-s_0\right)^2 u_{ss}^{init}\\ u^{final}\\ \left(s_{N_\Sigma}-s_0\right)u_s^{final}\\ \left(s_{N_\Sigma}-s_0\right)^2 u_{ss}^{final}\end{bmatrix} &= \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0\\ 42 & 30 & 20 & 12 & 6 & 2 & 0 & 0\end{bmatrix}\begin{bmatrix}a_0\\ a_1\\ a_2\\ a_3\\ a_4\\ a_5\\ a_6\\ a_7\end{bmatrix}\\ \xi_{6\times 1} &= L_{6\times 8}\theta_{8\times 1}\end{aligned}\right\} \tag{4.14}$$

The Lagrange multiplier method is applied as was described in the optimization problem described in Chapter 3 to solve the following minimization problem:

$$\left.\begin{array}{c} J = \min_{\theta} \frac{1}{2}(u - \Phi\theta)^T(u - \Phi\theta) \\ \text{Subject to: } L \cdot \theta = \xi \end{array}\right\} \tag{4.15}$$

The feed correction polynomial coefficients can then be computed by solving the following system of linear equations:

$$\left.\begin{array}{c} \Phi^T\Phi\theta + L^T\Lambda = \Phi^T u \\ L \cdot \theta = \xi \end{array}\right\} \Rightarrow \begin{bmatrix} \Phi^T\Phi & L^T \\ L & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \Lambda \end{bmatrix} = \begin{bmatrix} \Phi^T u \\ \xi \end{bmatrix} \tag{4.16}$$

Initially, the whole set of data points that were computed using the adaptive quadrature length method are used for the first fit. The mean squared error check is done with the following:

$$MSE = \sum_{i=0}^{N_\Sigma} \frac{(u_i - \hat{u}_i)^2}{N_\Sigma + 1} < \varepsilon_{MSE} \tag{4.17}$$

The other condition is to ensure that the correction polynomial is monotonically increasing. The roots of the 1st derivative are found:

$$\hat{u}_\sigma(\sigma = 0) = 7a_0\sigma^6 + 6a_1\sigma^5 + 5a_2\sigma^4 + 4a_3\sigma^3 + 3a_4\sigma^2 + 2a_5\sigma^1 + a_6 = 0 \tag{4.18}$$

The real parts of the roots are checked to ensure that there are no local maxima within $0 \leq \sigma \leq 1$. If either the root condition or the MSE conditions fail, the set of (u_i, s_i), $i = 0, \cdots, N_\Sigma$ data points are split into two. Then, the polynomial is once again fitted to the first half of the data, and it will continue to split until the conditions are met, in which case the next segment of data points is fitted. The number of data points being fitted is decreased with every sub-division. The process is terminated when all segments of data points are fitted. In the case where the number of data points has reduced to less than the order of the polynomial, it means that the tolerance, ε_S, used in finding the path length was not tight enough and that not enough data points were generated or that the MSE tolerance, ε_{MSE}, is too tight.

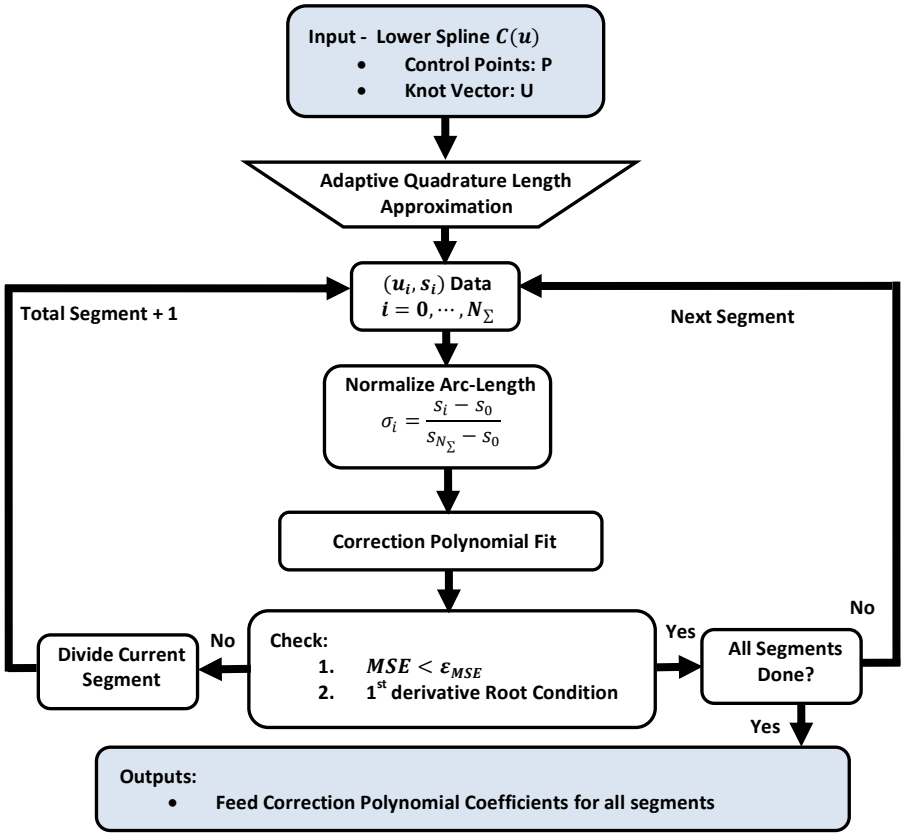


Figure 4.2: Feed Correction Polynomial Fit Flowchart

The feed correctional polynomial fitting process is outlined in Figure 4.2. The outputs are the polynomial coefficients for all the polynomial segments. Suppose there are n polynomial segments, this implies that there are n sets of polynomial coefficients:

$$\theta^j = \begin{bmatrix} a_0^j & a_1^j & a_2^j & \cdots & a_7^j \end{bmatrix}^T; \quad j = 1, \cdots, n \tag{4.19}$$

The endpoints of every feed correctional polynomial segments are also known and denoted as $\begin{bmatrix} s_0^j & s_{N_{\Sigma_j}}^j \end{bmatrix}$. During real time interpolation, the feed scheduler provides the desired change in the tool tip tangential displacement, $ds(k)$ at the given time step, $t = kT_s$. The displacement position of the next step is then:

$$s(k+1) = s(k) + ds(k) \tag{4.20}$$

The corresponding feed correction polynomial segment has to be used in order to compute for the curve parameter estimate of the next step, $\hat{u}(k+1)$. The matching polynomial segment is

the maximum j^{th} segment such that $s(k+1) \leq s^j_{N_{\Sigma_j}}$. The next curve parameter estimate is then found as:

$$\left.\begin{aligned} \sigma(k+1) &= \frac{s(k+1) - s_0^j}{s^j_{N_{\Sigma_j}} - s_0^j} \\ \hat{u}(k+1) &= [\sigma(k+1)^7 \quad \sigma(k+1)^6 \quad \cdots \quad \sigma(k+1)^1 \quad 1] \begin{bmatrix} a_0^j \\ \vdots \\ a_7^j \end{bmatrix} \end{aligned}\right\} \tag{4.21}$$

With the next curve parameter estimate of $\hat{u}(k+1)$, the equations for the lower and upper splines can be used to retrieve the tool tip position as well as the tool orientation vector.

The feed correction polynomial coefficients can be solved in pre-processing on the CNC machine. For real time implementation, the polynomial coefficients and the segment displacement position end points can all be stored within look-up tables and can be easily selected with a comparison operator. This allows the feed correction polynomial interpolation method to be extremely fast for real time computations unlike the Taylor Expansion method where the derivatives of the curve have to be computed in real time.

It can also be shown that by minimizing the error for curve parameter estimation, feed fluctuations are also reduced. Given a desired feedrate, $\frac{ds_d}{dt}$, and considering that the position from the NURBS tool path is in parametric form, where the actual curve parameter u is approximated by using the correction polynomial $u \cong \hat{u} = f(s_d)$. The tool path position and its derivative with respect to time can then be parameterized as $x = x\left(u\left(s_d(t)\right)\right)$ and $\frac{dx(u)}{dt} = \frac{dx(u)}{du}\frac{d\hat{u}}{ds_d}\frac{ds_d}{dt}$. By substituting this into the expression for actual feedrate:

$$\left.\begin{aligned} \dot{s} &= \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \\ \dot{s} &= \sqrt{x'(u)^2 + y'(u)^2 + z'(u)^2}\left(\frac{d\hat{u}}{ds_d}\right)\left(\frac{ds_d}{dt}\right) \end{aligned}\right\} \tag{4.22}$$

Since $u_s = \frac{1}{\sqrt{\left(\frac{dx(u)}{du}\right)^2 + \left(\frac{dy(u)}{du}\right)^2 + \left(\frac{dz(u)}{du}\right)^2}}$, it can be seen that if the polynomial can closely approximate $\hat{u} \cong u$ to within a mean square tolerance, $\sqrt{x'(u)^2 + y'(u)^2 + z'(u)^2}\left(\frac{d\hat{u}}{ds_d}\right)$ will approach unity and the fluctuation between the actual feedrate $\dot{s}$ and $\dot{s}_d$ will be minimized.

It will be shown within the simulation/experimental section that using tighter tolerances of ε_{MSE} will reduce the overall feed fluctuation. The downside of using a tighter tolerance is that there will be more segments of the feed correction polynomial, which requires a higher computation time to obtain in pre-processing as well as take up addition CNC memory during run time. In addition, the transitions between connecting polynomials can also induce some transient feedrate fluctuations.

4.3 Inverse Kinematics

The feed motion of the CNC machine is defined as the tangent displacement motion of the tool tip contact with respect to the work piece. This trajectory motion throughout the machining operation is described by the toolpath, providing information for both the tool tip contact position as well as the tool orientation. However, for typical servo control systems such as the CNC machine, the reference commands for the position control loop are the individual position of the axis motor drives. Therefore a structural dependent inverse kinematics transformation must be applied to the tool tip position and tool orientation to generate the drive commands for the Cartesian and rotary axis drives. The challenge associated with such a mapping is that five-axis machine tools generally exhibit non-Cartesian and nonlinear kinematics and that the same mapping must also be used in maintaining the physical limits of the drives.

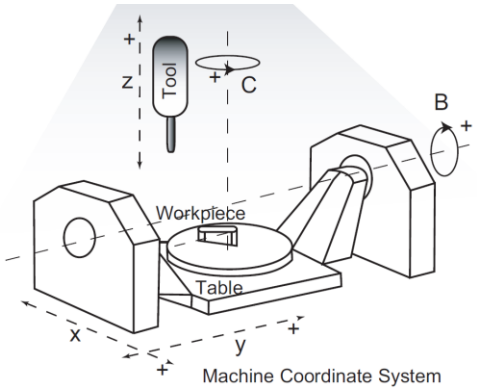


Figure 4.3: 5-Axis Machine Tool

Typical five-axis machines can be classified as three basic types: (1). Table-tilting, with two rotations on the table, (2). Spindle-tilting, with two rotations on the spindle, and (3) Hybrid, with one rotation on the table and one rotation on the spindle. The table-tilting configuration shown in Figure 4.3 is the most economical type with a major advantage of having fewer loads being imposed on the spindle, i.e. less tool vibrations. The disadvantage is that a large torque is

required to rotate/tilt the table if there is a large and heavy workpiece. This imposes major limitations on the rotary drives and heavily constrains the tangential feed, whereas the spindle–rotating type does not suffer from this issue.

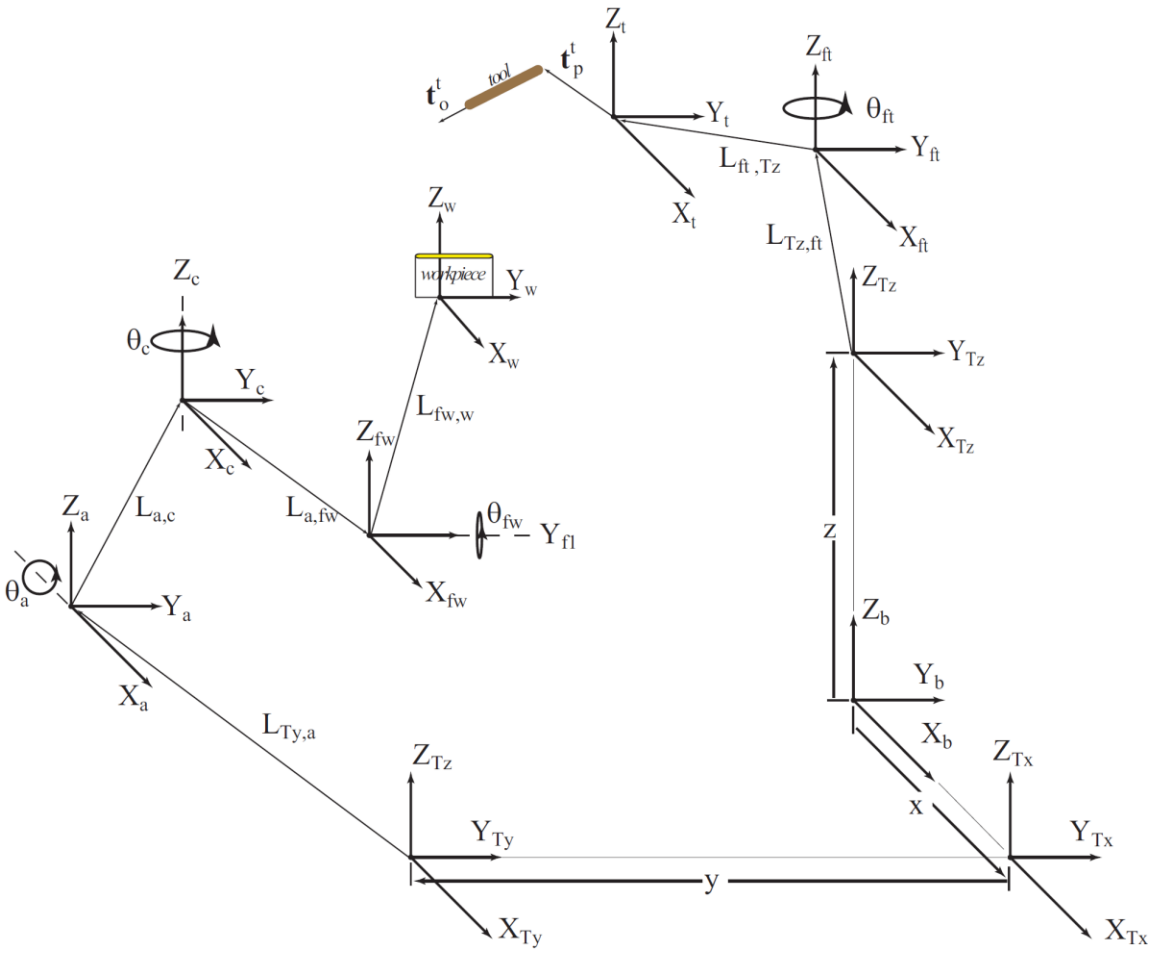


Figure 4.4: 5-Axis Machine Tool Reference Frames

The five-axis table tilting configuration kinematics is used and Figure 4.4 illustrates the coordinate systems for the table-tilting 5-axis machine tool. The kinematic algorithms of this configuration are borrowed from Sencer [54]. The transformations between the coordinate frames are expressed using 4×4 homogeneous transformation matrices expressed as:

$$H_{i+1}^{i} = \begin{bmatrix} R_{i,i+1}{}_{3\times3} & L_{i,i+1}^{i}{}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}_{4\times4} \tag{4.23}$$

where the system in the $i+1^{th}$ frame is expressed in the i^{th} frame. i.e. the coordinate of a point p in the $i+1^{th}$ frame, denoted as p^{i+1}, can be expressed in the i^{th} frame as the following:

$$p^i = R_{i,i+1}p^{i+1} + L^i_{i,i+1} \tag{4.24}$$

$L^i_{i,i+1} = L^i_{i,i+1,x}\boldsymbol{i} + L^i_{i,i+1,y}\boldsymbol{j} + L^i_{i,i+1,z}\boldsymbol{k}$ is the relative distance from the origin O_i to the origin O_{i+1} measured in the i^{th} frame. $R_{i,i+1}$is the orientation of the $i+1^{th}$ frame with respect to the i^{th} frame where:

$$R_{i,i+1} = \begin{bmatrix} \boldsymbol{i}_i \cdot \boldsymbol{i}_{i+1} & \boldsymbol{i}_i \cdot \boldsymbol{j}_{i+1} & \boldsymbol{i}_i \cdot \boldsymbol{k}_{i+1} \\ \boldsymbol{j}_i \cdot \boldsymbol{i}_{i+1} & \boldsymbol{j}_i \cdot \boldsymbol{j}_{i+1} & \boldsymbol{j}_i \cdot \boldsymbol{k}_{i+1} \\ \boldsymbol{k}_i \cdot \boldsymbol{i}_{i+1} & \boldsymbol{k}_i \cdot \boldsymbol{j}_{i+1} & \boldsymbol{k}_i \cdot \boldsymbol{k}_{i+1} \end{bmatrix} \tag{4.25}$$

The homogeneous transformation matrix can also be expressed as:

$$\left.\begin{aligned} H^i_{i+1} &= trans(L^i_{i,i+1,x}, L^i_{i,i+1,y}, L^i_{i,i+1,z}) \cdot \begin{bmatrix} R_{i,i+1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \\ \text{where } trans\left(L^i_{i,i+1,x}, L^i_{i,i+1,y}, L^i_{i,i+1,z}\right) &= \begin{bmatrix} & & L^i_{i,i+1,x} \\ \boldsymbol{I}_{3\times3} & & L^i_{i,i+1,y} \\ & & L^i_{i,i+1,z} \\ \mathbf{0}_{1\times3} & & 1 \end{bmatrix} \end{aligned}\right\} \tag{4.26}$$

The following is a list of the coordinate frames:

- $O_bX_bY_bZ_b$: Base
- $O_wX_wY_wZ_w$: Workpiece
- $O_tX_tY_tZ_t$: Tool
- $O_aX_aY_aZ_a$ and $O_cX_cY_cZ_c$: Primary rotary drive A and secondary rotary drive C
- $O_{Tx}X_{Tx}Y_{Tx}Z_{Tx}$, $O_{Ty}X_{Ty}Y_{Ty}Z_{Ty}$, and $O_{Tz}X_{Tz}Y_{Tz}Z_{Tz}$: Cartesian X,Y,Z drives
- $O_{fw}X_{fw}Y_{fw}Z_{fw}$ and $O_{ft}X_{ft}Y_{ft}Z_{ft}$: Miscellaneous Offsets/Tilts to workpiece and tool

The purpose of the frames $O_{fw}X_{fw}Y_{fw}Z_{fw}$ and $O_{ft}X_{ft}Y_{ft}Z_{ft}$ is to allow the offsets and tilts for the miscellaneous fixtures between the table and the workpiece and between the table and the tool to be assigned. Forward kinematics described the motion and orientation of a frame as a function of the motor joint positions, $[X \quad Y \quad Z \quad \theta_a \quad \theta_c]$. The workpiece system can be expressed in the base frame with the following series of homogeneous transformations:

$$\left.\begin{aligned} H_w^b &= H_{Tx}^b H_{Ty}^{Tx} H_a^{Ty} H_c^a H_{fw}^c H_w^{fw} \\ H_w^b &= Trans(X,0,0)\cdot Trans(0,Y,0)\cdot Trans(L_{Ty,a,x}^{Ty}, L_{Ty,a,y}^{Ty}, L_{Ty,a,z}^{Ty}) \\ &\quad \cdot TRot(X_a,\theta_a)\cdot Trans(L_{a,c,x}^{a}, L_{a,c,y}^{a}, L_{a,c,z}^{a}) \\ &\quad \cdot TRot(Z_c,\theta_c)\cdot Trans(L_{c,fw,x}^{c}, L_{c,fw,y}^{c}, L_{c,fw,z}^{c}) \\ &\quad \cdot TRot(Y_{fw},\theta_{fw})\cdot Trans(L_{fw,w,x}^{fw}, L_{fw,w,y}^{fw}, L_{fw,w,z}^{fw}) \end{aligned}\right\} \quad (4.27)$$

Similarly, the cutting tool system can be expressed in the base frame:

$$\left.\begin{aligned} H_t^b &= H_{Tz}^b H_{ft}^{Tz} H_t^{ft} \\ H_t^b &= Trans(0,0,Z)\cdot\ Trans\left(L_{Tz,ft,x}^{Tz}, L_{Tz,ft,y}^{Tz}, L_{Tz,ft,z}^{Tz}\right) \\ &\quad \cdot TRot\left(Z_{ft},\theta_{ft}\right)\cdot Trans(L_{ft,t,x}^{ft}, L_{ft,t,y}^{ft}, L_{ft,t,z}^{ft}) \end{aligned}\right\} \quad (4.28)$$

where $TRot(X_a,\theta_a)_{4\times4}$ is the homogeneous basic rotation matrix about the x axis by angle θ_a. The 3 dimensional basic rotation matrix will be denoted as $Rot(X_a,\theta_a)_{3\times3}$. The tool tip position and tool orientation are denoted as t_p^t and t_o^t in the tool frame and can be expressed in the workpiece frame as t_p^w and t_o^w:

$$\left.\begin{aligned} \begin{bmatrix} t_p^w & t_o^w \\ 1 & 0 \end{bmatrix} &= H_t^w \begin{bmatrix} t_p^t & t_o^t \\ 1 & 0 \end{bmatrix} \\ H_t^w = (H_b^w)\left(H_t^b\right) &= \left(H_w^b\right)^{-1}\left(H_t^b\right) \end{aligned}\right\} \quad (4.29)$$

The orientation of the cutting tool within the tool frame is aligned with the z axis:

$$t_o^t = [0 \quad 0 \quad 1]^T \quad (4.30)$$

Taking only the rotation components in calculating the tool orientation in the workpiece frame,

$$\left.\begin{aligned} \begin{bmatrix} t_o^w \\ 0 \end{bmatrix} &= \left(TRot(X_a,\theta_a)\cdot TRot(Z_c,\theta_c)\cdot TRot(Y_{fw},\theta_{fw})\right)^{-1}\cdot TRot\left(Z_{ft},\theta_{ft}\right)\begin{bmatrix} t_o^t \\ 0 \end{bmatrix} \\ \begin{bmatrix} t_o^w \\ 0 \end{bmatrix} &= TRot(Y_{fw},-\theta_{fw})\cdot TRot(Z_c,-\theta_c)\cdot TRot(X_a,-\theta_a)\cdot TRot\left(Z_{ft},\theta_{ft}\right)\begin{bmatrix} t_o^t \\ 0 \end{bmatrix} \end{aligned}\right\} \quad (4.31)$$

Assuming that the frames $O_{fw}X_{fw}Y_{fw}Z_{fw}$ and $O_{ft}X_{ft}Y_{ft}Z_{ft}$ are aligned with the previous frames, $\theta_{ft} = \theta_{fw} = 0$, this simplifies to:

$$\left.\begin{aligned}\begin{bmatrix} t_{o,i}^{w} \\ t_{o,j}^{w} \\ t_{o,k}^{w} \\ 0 \end{bmatrix} &= \begin{bmatrix} \cos(\theta_c) & \sin(\theta_c) & 0 & 0 \\ -\sin(\theta_c) & \cos(\theta_c) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_a) & \sin(\theta_a) & 0 \\ 0 & -\sin(\theta_a) & \cos(\theta_a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} O_i \\ O_j \\ O_k \\ 0 \end{bmatrix} &= \begin{bmatrix} t_{o,i}^{w} \\ t_{o,j}^{w} \\ t_{o,k}^{w} \\ 0 \end{bmatrix} = \begin{bmatrix} \sin(\theta_a)\sin(\theta_c) \\ \sin(\theta_a)\cos(\theta_c) \\ \cos(\theta_a) \\ 0 \end{bmatrix}\end{aligned}\right\} \tag{4.32}$$

The orientation toolpath given by the difference between the upper and lower spline is expressed as the tool orientation with respect to the workpiece, which is $[O_i(u) \quad O_j(u) \quad O_k(u)]$. The inverse kinematics solution for the rotary axes is then solved as:

$$\left.\begin{aligned}\theta_a(u) &= \arccos\left(O_k(u)\right) \\ \theta_c(u) &= \arctan2\big(O_i(u), O_j(u)\big)\end{aligned}\right\} \tag{4.33}$$

where $\arctan2$ returns an angle in the range of $(-\pi \quad \pi]$.

The tool tip position in the tool frame is represented by (usually at the origin):

$$t_p^t = \left[t_{p,x}^t \quad t_{p,y}^t \quad t_{p,z}^t\right]^T = [0 \quad 0 \quad 0]^T \tag{4.34}$$

and this is transformed into the workpiece frame by:

$$\begin{bmatrix} t_{p,x}^{w} \\ t_{p,y}^{w} \\ t_{p,z}^{w} \\ 1 \end{bmatrix} = H_t^w \begin{bmatrix} t_{p,x}^{t} \\ t_{p,y}^{t} \\ t_{p,z}^{t} \\ 1 \end{bmatrix} \tag{4.35}$$

where H_t^w can be rewritten as the following to factor out the Cartesian motor joint positions:

$$\left.\begin{aligned}H_t^w &= (H_b^w)\big(H_t^b\big) \\ H_t^w &= \big(H_{Ty}^w\big)\big(H_b^{Ty}\big)\big(H_{Tz}^b\big)(H_t^{Tz}) \\ H_t^w &= \big(H_{Ty}^w\big) \cdot Trans(-X, -Y, Z) \cdot (H_t^{Tz})\end{aligned}\right\} \tag{4.36}$$

$$\left.\begin{aligned}\begin{bmatrix} t_{p,x}^{w} \\ t_{p,y}^{w} \\ t_{p,z}^{w} \\ 1 \end{bmatrix} = \left(H_{Ty}^{w}\right) \cdot diag(-1,-1,1,1) \cdot \begin{bmatrix} \boldsymbol{I}_{3\times3} & \begin{matrix} X \\ Y \\ Z \end{matrix} \\ 0 & 1 \end{bmatrix} \\ \cdot diag(-1,-1,1,1) \cdot \left(H_{t}^{Tz}\right) \begin{bmatrix} -t_{p,x}^{t} \\ -t_{p,y}^{t} \\ t_{p,z}^{t} \\ 1 \end{bmatrix}\end{aligned}\right\} \tag{4.37}$$

The tool tip contact position with respect to the workpiece is given by the lower spline $C(u) = [x^t(u) \quad y^t(u) \quad z^t(u)]^T = [t_{p,x}^w(u) \quad t_{p,y}^w(u) \quad t_{p,z}^w(u)]^T$ and assuming that $\theta_{ft} = \theta_{fw} = 0$, this can be simplified into:

$$\left.\begin{aligned}\begin{bmatrix} t_{p,x}^{w}(u) \\ t_{p,y}^{w}(u) \\ t_{p,z}^{w}(u) \\ 1 \end{bmatrix} = \left(H_{Ty}^{w}\right) \cdot diag(-1,-1,1,1) \\ \cdot \left\{ \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} + \begin{bmatrix} & \boldsymbol{I_{3\times3}} & & \begin{matrix} -\left(L_{Tz,ft,x}^{Tz} + L_{ft,t,x}^{ft}\right) \\ -\left(L_{Tz,ft,y}^{Tz} + L_{ft,t,y}^{ft}\right) \\ +\left(L_{Tz,ft,z}^{Tz} + L_{ft,t,z}^{ft}\right) \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} -t_{p,x}^{t} \\ -t_{p,y}^{t} \\ t_{p,z}^{t} \\ 1 \end{bmatrix} \right\}\end{aligned}\right\} \tag{4.38}$$

This gives the inverse kinematic solution for the Cartesian drives, as:

$$\left.\begin{aligned}\begin{bmatrix} X(u) \\ Y(u) \\ Z(u) \\ 1 \end{bmatrix} = \left\{\left(H_{Ty}^{w}\right) \cdot diag(-1,-1,1,1)\right\}^{-1} \begin{bmatrix} t_{p,x}^{w}(u) \\ t_{p,y}^{w}(u) \\ t_{p,z}^{w}(u) \\ 1 \end{bmatrix} \\ \begin{bmatrix} -\left(L_{Tz,ft,x}^{Tz} + L_{ft,t,x}^{ft} + t_{p,x}^{t}\right) \\ -\left(L_{Tz,ft,y}^{Tz} + L_{ft,t,y}^{ft} + t_{p,y}^{t}\right) \\ \left(L_{Tz,ft,z}^{Tz} + L_{ft,t,z}^{ft} + t_{p,z}^{t}\right) \\ 1 \end{bmatrix}\end{aligned}\right\} \tag{4.39}$$

This solution can be simplified with the following expansion:

$$\left.\begin{aligned}\left\{\left(H_{Ty}^{w}\right)\cdot diag(-1,-1,1,1)\right\}^{-1} &= diag(-1,-1,1,1)^{-1}\left(H_{Ty}^{w}\right)^{-1}\\ &= diag(-1,-1,1,1)\left(H_{w}^{Ty}\right) = diag(-1,-1,1,1)\left(H_{a}^{Ty}H_{c}^{a}H_{fw}^{c}H_{w}^{fw}\right)\\ &= \begin{bmatrix} diag(-1,-1,1) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix} Rot(X,\theta_a(u)) & L_{Ty,a}^{Ty}\\ 0 & 1\end{bmatrix}\begin{bmatrix} Rot(Z,\theta_c(u)) & L_{a,c}^{a}\\ 0 & 1\end{bmatrix}\begin{bmatrix} \boldsymbol{I}_{3\times3} & L_{c,w}^{c}\\ 0 & 1\end{bmatrix}\\ &= \begin{bmatrix} R(u) & L(u)\\ 0 & 1\end{bmatrix}\end{aligned}\right\} \quad (4.40)$$

where

$$\left.\begin{aligned} R(u) &= diag(-1,-1,1)\cdot Rot(X,\theta_a(u))\cdot Rot(Z,\theta_c(u))\\ L(u) &= R(u)\cdot L_{c,w}^{c} + diag(-1,-1,1)\cdot Rot\left(X,\theta_a(u)\right)\cdot L_{a,c}^{a} + diag(-1,-1,1)\cdot L_{Ty,a}^{Ty}\end{aligned}\right\} \quad (4.41)$$

substituting these expressions back into Eqn. (4.39) and simplifying the constant translation from the table to tool frame:

$$\left.\begin{aligned}\begin{bmatrix} X(u)\\ Y(u)\\ Z(u)\\ 1\end{bmatrix} &= \begin{bmatrix} R(u)_{3\times3} & L(u)_{3\times1}\\ 0 & 1\end{bmatrix}\begin{bmatrix} C(u)_{3\times1}\\ 1\end{bmatrix} - diag(-1,-1,1)\cdot\begin{bmatrix} L_{Tz,ft}^{Tz} + L_{ft,t}^{ft} + t_p^t\\ 1\end{bmatrix}\\ \begin{bmatrix} X(u)\\ Y(u)\\ Z(u)\end{bmatrix} &= R(u)C(u) + L(u) - diag(-1,-1,1)\cdot\left(L_{Tz,ft}^{Tz} + L_{ft,t}^{ft} + t_p^t\right)\end{aligned}\right\} \quad (4.42)$$

It can be seen that the orientation of the cutting tool is only a function of the rotary axis drives, A and C, whereas the tool tip position is dependent on all five of the drives. Given the tool motion data as the tool tip position and tool orientation, from the tool path, $R(u) = \begin{bmatrix} x^t(u) & y^t(u) & z^t(u) & O_i(u) & O_j(u) & O_k(u)\end{bmatrix}^T$, the inverse kinematic solution of Eqn. (4.33) is first used to solve for the joint commands of the rotary drives, $[\theta_a(u) \quad \theta_c(u)]$ and then Eqn. (4.42) is used to solve for the Cartesian axis drive commands, $[X(u) \quad Y(u) \quad Z(u)]$.

4.4 5-Axis Constraints Formulations

In 5-axis machining operations, all five drives are simultaneously driven to produce the tangential feeding motion and the proper orientation of the cutting tool. As mentioned in the previous section, for the rotary table 5-Axis CNC configuration, the tool orientation is only dependent on the rotary drives and that the tool tip position is dependent on all five of the drives. Since each drive has its own respective velocity, and torque limits, the resultant feed of the motion is limited by the drive whose limit cannot handle the control effort as demanded by the CNC motion controller. When a drive motor saturates, the nonlinear effect of the saturation can induce undesired vibrations to the CNC system. In addition, the tracking performance for

the saturated axis will deteriorate whereas the other axes are still performing optimally. This will increase the overall contour error and reduce the quality of the part. Therefore an overall constraint on feed has to be imposed to keep all the axis drives within their operational velocity, acceleration, and jerk limits. In addition, the geometrical machining tolerances are also considered in feed scheduling.

The motor drive joint positions are obtained by applying the inverse kinematics transformation using Eq.(4.33) and Eq.(4.42). , $Q(u) = [X(u) \quad Y(u) \quad Z(u) \quad \theta_a(u) \quad \theta_c(u)]^T$. Therefore, in order to impose the axis velocity, acceleration, and jerk profile constraints $\left(\dot{Q}(t), \ddot{Q}(t), \dddot{Q}(t)\right)$, the dynamics from the inverse kinematics have to be included. First, the derivatives of the drive axis commands are obtained with respect to the tool tip contact displacement profile $s = s(t)$ as the following:

$$\left.\begin{aligned} Q(t) &= Q\big(s(t)\big) \\ \dot{Q}(t) &= Q_s(s) \cdot \dot{s}(t) \\ \ddot{Q}(t) &= Q_{ss}(s) \cdot \dot{s}(t)^2 + Q_s(s) \cdot \ddot{s}(t) \\ \dddot{Q}(t) &= Q_{sss}(s) \cdot \dot{s}(t)^3 + 3Q_{ss}(s) \cdot \dot{s}(t) \cdot \ddot{s}(t) + Q_s(s) \cdot \dddot{s}(t) \end{aligned}\right\} \tag{4.43}$$

where Q_s, Q_{ss}, and Q_{sss} are obtained from the parameterization between the curve parameter and the displacement profile as the following:

$$\left.\begin{aligned} Q(s) &= Q\big(u(s)\big) \\ Q_s(s) &= \frac{dQ(u)}{du}\frac{du}{ds} \\ Q_{ss}(s) &= \frac{d^2Q(u)}{du^2}\left(\frac{du}{ds}\right)^2 + \frac{dQ(u)}{du}\frac{d^2u}{ds^2} \\ Q_{sss}(s) &= \frac{d^3Q(u)}{du^3}\left(\frac{du}{ds}\right)^3 + 3\frac{d^2Q(u)}{du^2}\frac{du}{ds}\frac{d^2u}{ds^2} + \frac{dQ(u)}{du}\frac{d^3u}{ds^3} \end{aligned}\right\} \tag{4.44}$$

The derivatives of the curve parameter with respect to the displacement position are:

$$\left.\begin{aligned} u_s &= \frac{1}{\|C'(u)\|} \\ u_{ss} &= -\frac{\left(\frac{dC(u)}{du} \cdot \frac{d^2C(u)}{du^2}\right)}{\|C'(u)\|^4} \\ u_{sss} &= 4\frac{\left(\frac{dC(u)}{du} \cdot \frac{d^2C(u)}{du^2}\right)^2}{\|C'(u)\|^7} - \frac{\left(\frac{d^2C(u)}{du^2} \cdot \frac{d^2C(u)}{du^2} + \frac{dC(u)}{du} \cdot \frac{d^3C(u)}{du^3}\right)}{\|C'(u)\|^5} \end{aligned}\right\} \tag{4.45}$$

To incorporate the dynamics introduced from the inverse kinematics of the toolpath to the motor joint positions, the following derivates are calculated:

$$\left.\begin{aligned}
\frac{dQ(u)}{du} &= \left[\frac{dX(u)}{du} \quad \frac{dY(u)}{du} \quad \frac{dZ(u)}{du} \quad \frac{d\theta_a(u)}{du} \quad \frac{d\theta_c(u)}{du}\right]^T \\
\frac{d^2Q(u)}{du^2} &= \left[\frac{d^2X(u)}{du^2} \quad \frac{d^2Y(u)}{du^2} \quad \frac{d^2Z(u)}{du^2} \quad \frac{d^2\theta_a(u)}{du^2} \quad \frac{d^2\theta_c(u)}{du^2}\right]^T \\
\frac{d^3Q(u)}{du^3} &= \left[\frac{d^3X(u)}{du^3} \quad \frac{d^3Y(u)}{du^3} \quad \frac{d^3Z(u)}{du^3} \quad \frac{d^3\theta_a(u)}{du^3} \quad \frac{d^3\theta_c(u)}{du^3}\right]^T
\end{aligned}\right\} \tag{4.46}$$

The rotary A motor axis is only dependent on the z component of the orientation:

$$\left.\begin{aligned}
&\theta_a(O_k(u)) = \arccos\left(O_k(u)\right) \\
&\frac{d\theta_a}{du} = \frac{d\theta_a}{dO_k}\frac{dO_k}{du} \\
&\frac{d^2\theta_a}{du^2} = \frac{d^2\theta_a}{{dO_k}^2}\left(\frac{dO_k}{du}\right)^2 + \frac{d\theta_a}{dO_k}\frac{d^2O_k}{du^2} \\
&\frac{d^3\theta_a}{du^3} = \frac{d^3\theta_a}{{dO_k}^3}\left(\frac{dO_k}{du}\right)^3 + 3\frac{d^2\theta_a}{{dO_k}^2}\frac{d^2O_k}{du^2} + \frac{d\theta_a}{dO_k}\frac{d^3O_k}{du^3}
\end{aligned}\right\} \tag{4.47}$$

where the derivatives of motor A with respect to the z orientations are:

$$\left.\begin{aligned}
\frac{d\theta_a(O_k)}{dO_k} &= \frac{-1}{\sqrt{1-{O_k}^2}} \\
\frac{d^2\theta_a(O_k)}{{dO_k}^2} &= \frac{-O_k}{\left(1-{O_k}^2\right)^{\frac{3}{2}}} \\
\frac{d^3\theta_a(O_k)}{{dO_k}^3} &= \frac{-1}{\left(1-{O_k}^2\right)^{\frac{3}{2}}} - \frac{3{O_k}^2}{\left(1-{O_k}^2\right)^{\frac{5}{2}}}
\end{aligned}\right\} \tag{4.48}$$

The rotary C motor axis is dependent on the x and y components of the orientation:

$$\begin{aligned}
&\theta_c(u) = \arctan2\left(O_i(u), O_j(u)\right) \\
&\frac{d\theta_c(u)}{du} = \begin{cases}
\dfrac{\frac{dO_i(u)}{du}O_j(u) - O_i(u)\frac{dO_j(u)}{du}}{O_i(u)^2 + O_j(u)^2} & O_j(u) \neq 0 \\
0 & O_j(u) = 0 \\
\text{undefined} & O_j(u) = 0, O_j(u) = 0
\end{cases}
\end{aligned} \tag{4.49}$$

In order to compute for the orientation component derivatives, the upper spline is subtracted from the lower spline to get the tool axis vector, and the orientation is obtained from normalizing the tool axis:

$$\left.\begin{aligned} &TA(u) = C^{up}(u) - C(u) && \frac{dTA(u)}{du} = \frac{dC^{up}(u)}{du} - \frac{dC(u)}{du} \\ &\frac{d^2TA(u)}{du^2} = \frac{d^2C^{up}(u)}{du^2} - \frac{d^2C(u)}{du^2} && \frac{d^3TA(u)}{du^3} = \frac{d^3C^{up}(u)}{du^3} - \frac{d^3C(u)}{du^3} \end{aligned}\right\} \quad (4.50)$$

The norm of the tool axis is denoted by vector $\|TA\| = \sqrt{(TA_x)^2 + (TA_y)^2 + (TA_z)^2}$ and its derivatives with respect to the spline parameter are computed and denoted as $(\|TA\|_u, \|TA\|_{uu}, \|TA\|_{uuu})$, then the resulting orientation derivatives can be calculated as follows:

$$\left.\begin{aligned} &O(u) = \frac{TA}{\|TA\|} \\ &\frac{dO(u)}{du} = \begin{bmatrix} O_{i_u} \\ O_{j_u} \\ O_{k_u} \end{bmatrix} = \frac{TA_u}{\|TA\|} - \frac{TA\|TA\|_u}{\|TA\|^2} \\ &\frac{d^2O(u)}{du^2} = \begin{bmatrix} O_{i_{uu}} \\ O_{j_{uu}} \\ O_{k_{uu}} \end{bmatrix} = \frac{TA_{uu}}{\|TA\|} - \frac{TA\|TA\|_{uu} - 2TA_u\|TA\|_u}{\|TA\|^2} + 2\frac{TA\|TA\|_u{}^2}{\|TA\|^3} \\ &\frac{d^3O(u)}{du^3} = \begin{bmatrix} O_{i_{uuu}} \\ O_{j_{uuu}} \\ O_{k_{uuu}} \end{bmatrix} = \frac{\|TA\|^3 A_{uuu} - \|TA\|^2(TA\|TA\|_{uuu} + 3\|TA\|_{uu}TA_u + 3\|TA\|TA_{uu})}{\|TA\|^4} \\ &\qquad + \frac{\|TA\|\left(6TA_u\|TA\|_u{}^2 + 6TA\|TA\|_{uu}\|TA\|_u\right) - 6TA\|TA\|_u{}^3}{\|TA\|^4} \end{aligned}\right\} \quad (4.51)$$

The Cartesian motor axes are dependent on the rotary axes and the lower spline tool contact toolpath. From Eqn. (4.42), the simplified inverse kinematics solution is written in the following form where $C(u)$ is the position of the lower spline toolpath and T is a constant translation:

$$\left.\begin{aligned} &R(u) = diag(-1,-1,1) \cdot Rot(X, \theta_a(u)) \cdot Rot(Z, \theta_c(u)) \\ &L(u) = R(u) \cdot L^c_{c,w} + diag(-1,-1,1) \cdot Rot\big(X, \theta_a(u)\big) \cdot L^a_{a,c} + diag(-1,-1,1) \cdot L^{Ty}_{Ty,a} \\ &\begin{bmatrix} X(u) \\ Y(u) \\ Z(u) \end{bmatrix} = R\big(\theta_a(u), \theta_c(u)\big)C(X(u), Y(u), Z(u)) + L\big(\theta_a(u), \theta_c(u)\big) - T \end{aligned}\right\} \quad (4.52)$$

Then the derivatives of the Cartesian motor axes can be found:

$$\left.\begin{aligned}
\begin{bmatrix} \frac{dX(u)}{du} \\ \frac{dY(u)}{du} \\ \frac{dZ(u)}{du} \end{bmatrix} &= \frac{dR(u)}{du}C(u) + R(u)\frac{dC(u)}{du} + \frac{dL(u)}{du} \\
\begin{bmatrix} \frac{d^2X(u)}{du^2} \\ \frac{d^2Y(u)}{du^2} \\ \frac{d^2Z(u)}{du^2} \end{bmatrix} &= \frac{d^2R(u)}{du^2}C(u) + 2\frac{dR(u)}{du}\frac{dC(u)}{du} + R(u)\frac{d^2C(u)}{du^2} + \frac{d^2L(u)}{du^2} \\
\begin{bmatrix} \frac{d^3X(u)}{du^3} \\ \frac{d^3Y(u)}{du^3} \\ \frac{d^3Z(u)}{du^3} \end{bmatrix} &= \frac{d^3R(u)}{du^3}C(u) + 3\frac{d^2R(u)}{du^2}\frac{dC(u)}{du} + 3\frac{dR(u)}{du}\frac{d^2C(u)}{du^2} + R(u)\frac{d^3C(u)}{du^3} + \frac{d^3L(u)}{du^3}
\end{aligned}\right\} \quad (4.53)$$

where the set of rotation matrices are dependent on both θ_a and θ_c and their derivatives are shown as:

$$\left.\begin{aligned}
\frac{dR(u)}{du} &= \frac{\partial R}{\partial \theta_a}\frac{d\theta_a}{du} + \frac{\partial R}{\partial \theta_c}\frac{d\theta_c}{du} \\
\frac{d^2R(u)}{du^2} &= \frac{\partial^2 R}{\partial {\theta_a}^2}\left(\frac{d\theta_a}{du}\right)^2 + \frac{\partial R}{\partial \theta_a}\frac{d^2\theta_a}{du^2} + \frac{\partial^2 R}{\partial {\theta_c}^2}\left(\frac{d\theta_c}{du}\right)^2 + \frac{\partial R}{\partial \theta_c}\frac{d^2\theta_c}{du^2} \\
\frac{d^3R(u)}{du^3} &= \frac{\partial^3 R}{\partial {\theta_a}^3}\left(\frac{d\theta_a}{du}\right)^3 + 3\frac{\partial^2 R}{\partial {\theta_a}^2}\frac{d\theta_a}{du}\frac{d^2\theta_a}{du^2} + \frac{\partial R}{\partial \theta_a}\frac{d^3\theta_a}{du^3} \\
&\quad + \frac{\partial^3 R}{\partial {\theta_c}^3}\left(\frac{d\theta_c}{du}\right)^3 + 3\frac{\partial^2 R}{\partial {\theta_c}^2}\frac{d\theta_c}{du}\frac{d^2\theta_c}{du^2} + \frac{\partial R}{\partial \theta_c}\frac{d^3\theta_c}{du^3}
\end{aligned}\right\} \quad (4.54)$$

The expression for the motor positions in terms of the displacement position, $Q(s)$, can be derived by applying the inverse kinematics to the toolpath and then approximating the curve parameter parameterization u using the feed correction polynomial as $Q\big(u(s)\big)$. Furthermore, using the sets of derivatives shown above, the first, second, and third motor position derivatives, $Q_s(s)$, $Q_{ss}(s)$,and $Q_{sss}(s)$ can be found. There is now a complete mapping between the upper and lower spline toolpaths, which provide the simultaneous tool position and orientation information, to the motor joint commands of the Cartesian and rotary drives in terms of the toolpath arc-length displacement. Now a relationship can be formed to impose a limit on the feedrate as scheduled by the feed scheduler to ensure that none of the motors

violate their kinematic constraints. The five axis drive constraints algorithms are taken from Sencer [51]. The set of nonlinear constraints will be formed based on the arc-length parameterized motor axis commands, as various portions of the toolpath are more demanding for some motors, while other portions may be more lenient. Given the velocity saturation limits of the motors to be $[v_{x,max} \quad v_{y,max} \quad v_{z,max} \quad v_{\theta_a,max} \quad v_{\theta_c,max}]^T$, travelling along a 5-axis toolpath from $0 \leq s \leq S_{\sum}$, the following constraints equations have to be met:

$$-\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \leq \begin{bmatrix} X_s(s)/v_{x,max} \\ Y_s(s)/v_{y,max} \\ Z_s(s)/v_{z,max} \\ \theta_{a_s}(s)/v_{\theta_a,max} \\ \theta_{c_s}(s)/v_{\theta_c,max} \end{bmatrix} \dot{s}(s) \leq \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \tag{4.55}$$

Using the normalized partial derivatives of the motor axis commands, the maximum allowable feed is where the maximum partial derivative is equated to the limit, such that none of the drive actually violates the velocity limit condition:

$$\left.\begin{aligned} Q^v_{s,max}(s) = \max \left| \begin{bmatrix} \frac{X_s(s)}{v_{x,max}} & \frac{Y_s(s)}{v_{y,max}} & \frac{Z_s(s)}{v_{z,max}} & \frac{\theta_{a_s}(s)}{v_{\theta_a,max}} & \frac{\theta_{c_s}(s)}{v_{\theta_c,max}} \end{bmatrix}^T \right| \\ \dot{s}_{V,max}(s) = \frac{1}{Q^v_{s,max}(s)} \end{aligned}\right\} \tag{4.56}$$

or expressed as a single velocity inequality as:

$$C_V(s) = Q^v_{s,max}(s)\dot{s}(s) - 1 \leq 0 \tag{4.57}$$

Similar to the velocity constraints, the second path displacement derivatives of the motor positions have to lie within the maximum allowable range of the motor acceleration limits:

$$-\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \leq \begin{bmatrix} X_{ss}(s)/a_{x,max} \\ Y_{ss}(s)/a_{y,max} \\ Z_{ss}(s)/a_{z,max} \\ \theta_{a_{ss}}(s)/a_{\theta_a,max} \\ \theta_{c_{ss}}(s)/a_{\theta_c,max} \end{bmatrix} \dot{s}^2(s) + \begin{bmatrix} X_s(s)/a_{x,max} \\ Y_s(s)/a_{y,max} \\ Z_s(s)/a_{z,max} \\ \theta_{a_s}(s)/a_{\theta_a,max} \\ \theta_{c_s}(s)/a_{\theta_c,max} \end{bmatrix} \ddot{s}(s) \leq \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \tag{4.58}$$

By normalizing the motor axis command derivatives with the maximum acceleration limits, denoted by Q^a_s, Q^a_{ss}:

$$\left.\begin{aligned} Q_s^a &= \left[\frac{X_s(s)}{a_{x,max}} \quad \frac{Y_s(s)}{a_{y,max}} \quad \frac{Z_s(s)}{a_{z,max}} \quad \frac{\theta_{a_s}(s)}{a_{\theta_a,max}} \quad \frac{\theta_{c_s}(s)}{a_{\theta_c,max}}\right]^T \\ Q_{ss}^a &= \left[\frac{X_{ss}(s)}{a_{x,max}} \quad \frac{Y_{ss}(s)}{a_{y,max}} \quad \frac{Z_{ss}(s)}{a_{z,max}} \quad \frac{\theta_{a_{ss}}(s)}{a_{\theta_a,max}} \quad \frac{\theta_{c_{ss}}(s)}{a_{\theta_c,max}}\right]^T \end{aligned}\right\} \tag{4.59}$$

The following acceleration inequality is formed:

$$\left|Q_{ss,i}^a(s)\right|\dot{s}^2(s) + \left|Q_{s,i}^a(s)\right||\ddot{s}(s)| \le 1; \;\; i = 1, \cdots, 5$$

There is an inverse relationship between the allowable feed and the allowable acceleration. The characteristics of a typical profile, allows for different operating regimes where either feed or acceleration can be more dominant, such as when the motors are starting from low speeds, a high tangential acceleration can be used to quickly reach the desired feed. As a constant high feed is maintained, the required tangential acceleration is zero. It is possible to optimally determine the combination of feed and acceleration values from the five motors, however, it is more computationally efficient to bound the feasible regions where the maximum feed and accelerations can be found by setting the other to zero:

$$\left.\begin{aligned} \dot{s}(s) = 0 \Rightarrow \ddot{s}_{a,max}(s) &= \frac{1}{Q_{s,max}^a(s)} \\ \ddot{s}(s) = 0 \Rightarrow \dot{s}_{a,max}(s) &= \sqrt{\frac{1}{Q_{ss,max}^a(s)}} \end{aligned}\right\} \tag{4.60}$$

Then the feasible acceleration inequality constraint becomes:

$$C_A(s) = Q_{s,max}^a(s)|\ddot{s}(s)| + Q_{ss,max}^a(s)\dot{s}^2(s) - 1 \le 0 \tag{4.61}$$

The following set of motor constraints are set for jerk:

$$-\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \le \begin{bmatrix} X_{sss}(s)/J_{x,max} \\ Y_{sss}(s)/J_{y,max} \\ Z_{sss}(s)/J_{z,max} \\ \theta_{a_{sss}}(s)/J_{\theta a,max} \\ \theta_{c_{sss}}(s)/J_{\theta_c,max} \end{bmatrix}\dot{s}^3(s) + 3\begin{bmatrix} X_{ss}(s)/J_{x,max} \\ Y_{ss}(s)/J_{y,max} \\ Z_{ss}(s)/J_{z,max} \\ \theta_{a_{ss}}(s)/J_{\theta a,max} \\ \theta_{c_{ss}}(s)/J_{\theta_c,max} \end{bmatrix}\dot{s}(s)\ddot{s}(s) + \begin{bmatrix} X_s(s)/J_{x,max} \\ Y_s(s)/J_{y,max} \\ Z_s(s)/J_{z,max} \\ \theta_{a_s}(s)/J_{\theta a,max} \\ \theta_{c_s}(s)/J_{\theta_c,max} \end{bmatrix}\ddot{s}(s) \le \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \tag{4.62}$$

Similar to the acceleration inequality, the jerk constraint is written in the following form:

$$\left|Q_{sss,i}^j(s)\right||\dot{s}^3(s)| + 3\left|Q_{ss,i}^j(s)\right||\dot{s}(s)||\ddot{s}(s)| + \left|Q_{s,i}^j(s)\right||\dddot{s}(s)| \le 1; \;\; i = 1, \cdots, 5 \tag{4.63}$$

where the jerk limited maximum feed can be evalutated by setting acceleration and jerk to zero:

$$\ddot{s}(s) = \dddot{s}(s) = 0 \Rightarrow \dot{s}_{j,max}(s) = \sqrt[3]{\frac{1}{\left|Q^{j}_{sss,max}(s)\right|}} \tag{4.64}$$

Then similarily the feasible jerk region is defined using the following equality:

$$C_J(s) = Q^{j}_{sss,max}(s)|\dot{s}^3(s)| + 3Q^{j}_{ss,max}|\dot{s}(s)||\ddot{s}(s)| + Q^{j}_{s,max}(s)|\dddot{s}(s)| - 1 \leq 0 \tag{4.65}$$

By imposing the axis constraints in terms of velocity, acceleration, and jerk, the kinematic limits of the motors can be kept from saturation and operate within a feasible range. In addition to the motor saturation constraints, researchers such as Xu [66] and Yeh [68] have shown that a geometrical dependence of feedrate can be applied to limit the the contour error that results on the machined part.

An inherent source of error in the interpolation process of parametric curves is the error that arises from using discrete reference commands as the input to the servo controller. While the toolpath is a smooth parametric curve, the real time interpolation of the toolpath will generate discrete reference position commands with the desired displacement ds of the tool tip at every sampling time interval of T_s as shown in Figure 4.5.

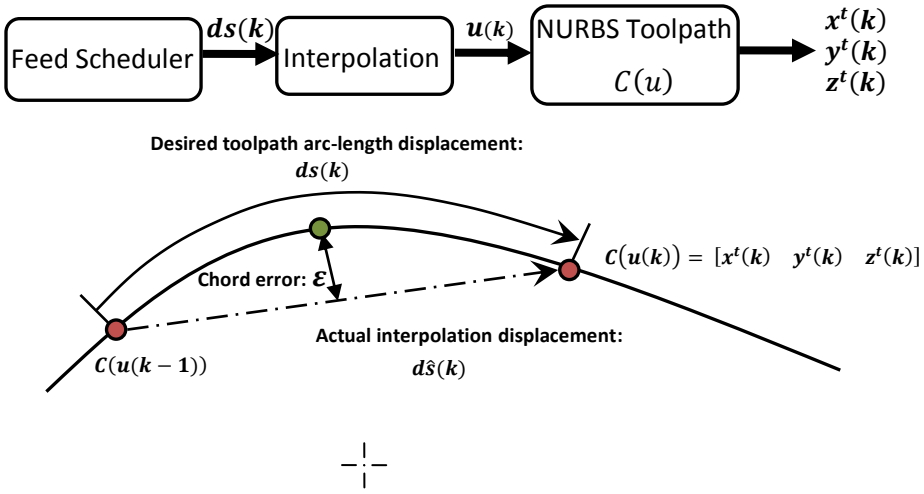


Figure 4.5: Interpolated Discrete Reference Position Commands

ds is the desired arc-length on the toolpath:

$$ds(k) = \int_{u(k-1)}^{u(k)} \left\| \frac{dC(u)}{du} \right\| du \tag{4.66}$$

However the actual displacement between the discrete reference command is:

$$d\hat{s}(k) = \sqrt{\left(x^t(k) - x^t(k-1)\right)^2 + \left(y^t(k) - y^t(k-1)\right)^2 + \left(z^t(k) - z^t(k-1)\right)^2} \quad (4.67)$$

This discrepency leads to two problems: feedrate fluctuation and chord error. While it is not possible to eliminiate these problems completely, it is possible to reduce them by limiting the feedrate. The discrepency between the actual displacement and the desired displacement ds is highly prevalent at locations of the toolpath where there is a high curvature (low radius of curvature). High feed fluctuations around sharp curves can lead to various vibrations and contour tracking problems. In order to reduce this problem, Zhiming [70] applied a variable feedrate method with dependence on the toolpath curvature. The toolpath curvature is calculated as follows:

$$K(u) = \frac{|C'(u) \times C''(u)|}{|C'(u)|^3} \quad (4.68)$$

The curvature dependent feedrate is expressed in the following form such that the feedrate $V(u)$ approaches V_{max} (desired feedrate) when the curvature approaches zero.

$$V(u) = \frac{k^*}{K(u) + k^*} V_{max} \quad (4.69)$$

where a specific pair of curvature and feedrate values can be used to determine k^*. By reducing the feedrate at locations of high curvature, additional interpolation steps are required to complete the curve, but each of the smaller steps contains a much lower displacement descrepency from the desired displacement.

In addition to the feed fluctuation problem, chord error, as defined by the distance from the actual interpolated chord and the toolpath parametric curve is also highly dependent on toolpath curvature. When there is a large local radius of curvature in the toolpath, i.e. the curve is relatively straight, the toolpath can be approximated by the interpolated discrete commands sent to the CNC position controller and the chord error is minimal and high feedrates can be used. For locations with a large curvature, large chord error can occur. By specifying a maximum chord error denoted as ER, an adaptive feedrate can be applied as follows:

$$V(u) = \begin{cases} V_{max} & \text{if } \frac{2}{T_s}\sqrt{\rho(u)^2 - (\rho(u) - ER)} > V_{max} \\ \frac{2}{T_s}\sqrt{\rho(u)^2 - (\rho(u) - ER)} & \text{if } \frac{2}{T_s}\sqrt{\rho(u)^2 - (\rho(u) - ER)} \leq V_{max} \end{cases} \quad (4.70)$$

where $\rho(u) = 1/K(u)$, is the local radius of curvature and is the inverse of the curve curvature. The curve parameter value can be approximated using the fitted correction polynomial as $\hat{u}(s)$. The set of feedrate constraints has to be considered by the CNC feed scheduler to properly schedule a feedrate profile that does not violate the velocity, acceleration and jerk limits of the drive, and considers the geometrical machining tolerance of the toolpath.

4.5 Feedrate Profiling

The objective of feedrate profiling is to generate a minimum cycle time displacement profile to direct the cutting tool. It is common for the process planner to specify a desired feedrate for the CNC machine to cut the workpiece. However, the entire cutting process will not only consist of the machine tool travelling with a constant feed, as the motor axes have to be accelerated from rest at the beginning of the toolpath and decelerated at the end. In addition, the tangential feed exhibited by the cutting tool is the result of all the motor axes. Travelling at a constant high feedrate throughout the entire toolpath where the demand of each of the motor is varying according to the toolpath is not feasible, as the velocity, acceleration, and jerk constraints of the individual motors may be violated. The geometrical feedrate constraints as presented from the previous section are also important considerations.

The most common type of feedrate profile is the trapezoidal velocity profile, where the motors are ramped up in velocity to meet a desired feed and are ramped back down at the end of the toolpath. The jerk limited profiling for a single curve segment was presented by Erkorkmaz and Altintas [14]. It provides a feedrate kinematics profile that bounded the tangential jerk displayed by the cutting tool, allowing for a smooth acceleration and deceleration profile. The resulting profile has significantly lower high frequency content in acceleration as opposed to a trapezoidal feedrate profile. A C^3 displacement profile was later introduced [17] using a quintic polynomial as:

$$s_k(\tau) = A_k\tau^5 + B_k\tau^4 + C_k\tau^3 + D_k\tau^2 + E_k\tau + F_k, \qquad 0 \leq \tau \leq T_k \quad (4.71)$$

It provided a multi-segment displacement profile where the jerk content could be summed in an overall jerk objective function given as:

$$J_v = \int_0^{T_0} [\dddot{s}_0(\tau)]^2 d\tau + \int_0^{T_1} [\dddot{s}_1(\tau)]^2 d\tau + \cdots + \int_0^{T_{Ns-1}} [\dddot{s}_{Ns-1}(\tau)]^2 d\tau \tag{4.72}$$

Then the jerk minimization problem can be used to solve for the coefficients of the polynomials. In order to optimize the total cycle time, a second cost function is formulated as the sum of the time duration of each polynomial:

$$\min T_\Sigma = \min \sum_{k=0}^{Ns-1} T_k \,, \text{ subject to: } C(t) \leq 0 \text{ for } 0 \leq t \leq T_\Sigma \tag{4.73}$$

The minimum time feed optimization accounts for the motor constraints by forming an overall constraint equation cascading the feasible motor velocity, acceleration, and jerk constraints given in Eqns. (4.57),(4.61), and (4.65) as:

$$C(t) = \begin{bmatrix} C_F^-(t) & C_F^+(t) & C_V(t) & C_A(t) & C_J(t) \end{bmatrix} \tag{4.74}$$

where $C_F^-(t) = -\dot{s}(t) \leq 0$ is the lower feed limit to ensure forward motion, and $C_F^+(t) = \dot{s}(t) - V_{max} \leq 0$ is the process planner specified upper feed limit. Sencer [51] extended this optimization by expressing the feed profile in B-spline form and optimized by maximizing the control points, f_i, of the feed profile as shown in Figure 4.6:

$$\min_{\dot{s}} T_\Sigma = \max_{f_i} \sum_{i=1}^{n} f_i \,, \text{ subject to: } C(s) \leq 0 \text{ for } 0 \leq s \leq S_\Sigma \tag{4.75}$$

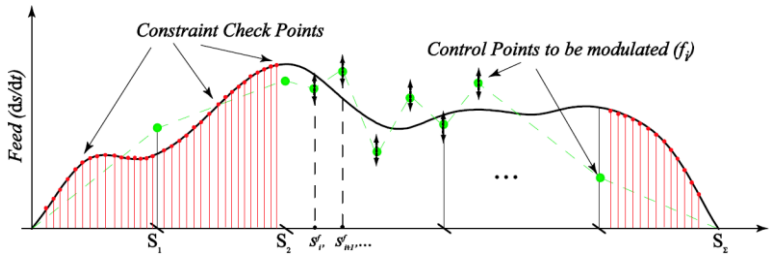


Figure 4.6: B-Spline Feed Profile

The feed profile is solved iteratively by using a forward windowing technique of incremental Δs, to solve for the highest feed control points that will not violate the constraints. The feedrate optimization algorithms are particularly effective in rough milling, or semi-finishing operations,

where rapid feed alterations are not vital for part tolerances and feedrate fluctuations are more tolerable. While these optimization approaches can minimize the overall jerk content throughout the entire toolpath, as well as greatly increase productivity with minimal cycle time, the algorithms also introduce unnecessary feedrate fluctuations throughout the toolpath. In finishing operations where tangential feedrate must be scheduled smoothly with minimal fluctuations, constant feed sections have to be used where they are connected to each other using smooth acceleration transients. While using constant feedrate sections will result in a sub-optimal solution, it has a lower computational cost and is more robust to a wider variety of applications. Heng and Erkorkmaz [20], [22] have shown that a heuristic approach to applying the limited jerk feed profile can yield favorable results with short cycle time and short computation time.

The limited jerk trajectory profile is computationally easy to implement compared to the optimization methods and it has a well-defined profile that can be taken advantage of. By imposing a bounded acceleration, A_{max}, and a bounded maximum jerk, J_{max}, on the feed motion, the acceleration and jerk constraints for the motor drives from Eqn. (4.61) and (4.65) where the set of feasible $|\dot{s}(s)|$, $|\ddot{s}(s)|$, $|\dddot{s}(s)|$ feed, acceleration, and jerk values had to be identified iteratively along the tool-path can be simplified as $|\ddot{s}(s)| = A_{max}$ and $|\dddot{s}(s)| = J_{max}$.

$$\left.\begin{aligned} C_A(s) = Q^a_{s,max}(s)|\ddot{s}(s)| + Q^a_{ss,max}(s)\dot{s}^2(s) - 1 \le 0 \\ \dot{s}(s) \le \sqrt{\frac{1 - Q^a_{s,max}(s)|A_{max}|}{Q^a_{ss,max}(s)}} \end{aligned}\right\} \tag{4.76}$$

Since Q^a_s is normalized by the individual motor limits, there is an inherent assumption that $|A_{max}|$ is less than the individual motor drive acceleration limits, i.e. $|A_{max}| \le a_{x,max}$.

$$\left.\begin{aligned} C_J(s) = Q^j_{sss,max}(s)|\dot{s}^3(s)| + 3Q^j_{ss,max}|\dot{s}(s)||\ddot{s}(s)| + Q^j_{s,max}(s)|\dddot{s}(s)| - 1 \le 0 \\ Q^j_{sss,max}(s)|\dot{s}^3(s)| + 3Q^j_{ss,max}|\dot{s}(s)||A_{max}| + Q^j_{s,max}(s)|J_{max}| - 1 \le 0 \end{aligned}\right\} \tag{4.77}$$

where the real root from the cubic equation can be taken as the maximum feedrate imposed by jerk. With the set of drive motor constraints and the set of geometrical constraints defined, the constraints equation in terms of $C(s)$ is known for the entire tool-path:

$$\left.\begin{aligned}
&C_F^-(s) = -\dot{s}(s) \le 0 \\
&C_F^+(s) = \dot{s}(s) - V_{max} \le 0 \\
&C_V(s) = Q^v_{s,max}(s)\dot{s}(s) - 1 \le 0 \\
&C_A(s) = Q^a_{s,max}(s)|A_{max}| + Q^a_{ss,max}(s)\dot{s}^2(s) - 1 \le 0 \\
&C_J(s) = Q^j_{sss,max}(s)|\dot{s}^3(s)| + 3Q^j_{ss,max}|\dot{s}(s)||A_{max}| + Q^j_{s,max}(s)|J_{max}| - 1 \le 0 \\
&C_K(s) = \dot{s}(s)\big(K\big(\hat{u}(s)\big) + k^*\big) - k^* V_{max} \le 0 \\
&C_{CH}(s) = \dot{s}(s) \cdot \frac{2}{T_s}\sqrt{\rho\big(\hat{u}(s)\big)^2 - \big(\rho\big(\hat{u}(s)\big) - ER\big)} \le 0
\end{aligned}\right\} \quad (4.78)$$

The effective feedrate constraint at any particular displacement value s is the minimum of all the constraints. The general case of the limited jerk profile is shown in the following Figure 4.7, composing of seven phases: 3 phases for acceleration, 1 for constant feed, and 3 for deceleration. However, this profile can easily be set up with only four sections by having only acceleration, $f = f_e$, or only deceleration, $f = f_s$. To preserve generality, the signed value for acceleration is used where $A > 0$, indicates an acceleration stage, and $A < 0$, indicates a deceleration stage. The same applies for jerk where $J_1 = -J_3 > 0$ or $J_5 = -J_7 > 0$, indicates acceleration and $J_1 = -J_3 < 0$ or $J_5 = -J_7 < 0$, indicates deceleration.

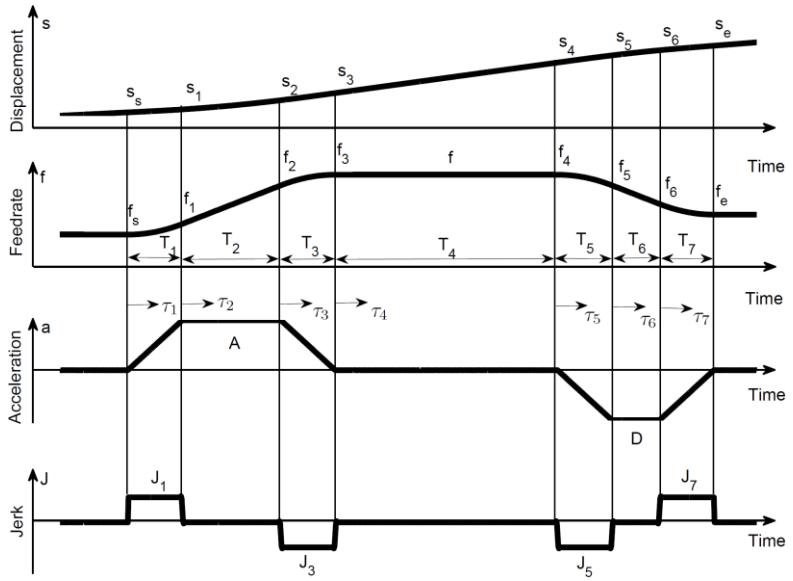


Figure 4.7: Limited Jerk Trajectory Profile

The feed and displacement equations for the profile are listed as the following [14]:

$$f(\tau)=\begin{Bmatrix} f_s+\frac{1}{2}J_1\tau_1^2, & f_s\text{: initial feed} & 0\le\tau<T_1 \\ f_1+A\tau_2, & f_1=f_s+\frac{1}{2}J_1T_1^2, & T_1\le\tau<T_2 \\ f_2+A\tau_3+\frac{1}{2}J_3\tau_3^2, & f_2=f_1+AT_2, & T_2\le\tau<T_3 \\ f_3, & f_3=f_2+AT_3+\frac{1}{2}J_3T_3^2, & T_3\le\tau<T_4 \\ f_4+\frac{1}{2}J_5\tau_5^2, & f=f_3=f_4, & T_4\le\tau<T_5 \\ f_5+D\tau_6, & f_5=f_4+\frac{1}{2}J_5T_5^2, & T_5\le\tau<T_6 \\ f_6+D\tau_7+\frac{1}{2}J_7\tau_7^2, & f_6=f_5+DT_6, & T_6\le\tau\le T_7 \end{Bmatrix} \quad (4.79)$$

$$s(\tau)=\begin{Bmatrix} s_s+f_s\tau_1+\frac{1}{6}J_1\tau_1^3, & s_s\text{:initial position} & 0\le\tau<T_1 \\ s_1+f_1\tau_2+\frac{1}{2}A\tau_2^2, & s_1=s_s+f_sT_1+\frac{1}{6}J_1T_1^3, & T_1\le\tau<T_2 \\ s_2+f_2\tau_3+\frac{1}{2}A\tau_3^2+\frac{1}{6}J_3\tau_3^3, & s_2=s_1+f_1T_2+\frac{1}{2}AT_2^2, & T_2\le\tau<T_3 \\ s_3+f_3\tau_4, & s_3=s_2+f_2T_3+\frac{1}{2}AT_3^2+\frac{1}{6}J_3T_3^3, & T_3\le\tau<T_4 \\ s_4+f_4\tau_5+\frac{1}{6}J_5\tau_5^3, & s_4=s_3+f_3T_4, & T_4\le\tau<T_5 \\ s_5+f_5\tau_6+\frac{1}{2}D\tau_6^2, & s_5=s_4+f_4T_5+\frac{1}{6}J_5T_5^3, & T_5\le\tau<T_6 \\ s_6+f_6\tau_7+\frac{1}{2}D\tau_7^2+\frac{1}{6}J_7\tau_7^3, & s_6=s_5+f_5T_6+\frac{1}{2}DT_6^2, & T_6\le\tau\le T_7 \end{Bmatrix} \quad (4.80)$$

By using a specified A_{max} and J_{max}, the following expressions can be used where the jerk values and the acceleration / deceleration values are the same:

$$\left.\begin{array}{l} A_{max}=A=-D;\ J_{max}=J_1=-J_3; \\ T_1=T_3=\dfrac{A}{J_1}=\dfrac{A_{max}}{J_{max}};\ \ T_2=\dfrac{f-f_s}{A_{max}}-\dfrac{A_{max}}{J_{max}}\ge 0 \end{array}\right\} \quad (4.81)$$

Since feed must be attained within the first three phases, the bounded acceleration stage may not exist; in which case, a reachable acceleration is found as:

$$A=\begin{Bmatrix} A_{max} & \text{if} \quad T_2\ge 0 \\ sgn(A)\cdot\sqrt{J_{max}(f-f_s)} & \text{if} \quad T_2<0 \end{Bmatrix} \quad (4.82)$$

where T_1 and T_3 are updated with the reachable acceleration. Similarly, for the deceleration stage:

$$\left.\begin{aligned} &D = -A_{max} = -A;\ J_5 = -J_{max} = -J_7 \\ &T_5 = T_7 = \frac{D}{J_5} = \frac{A_{max}}{J_{max}};\ T_6 = \frac{f - f_e}{A_{max}} - \frac{A_{max}}{J_{max}} \geq 0 \\ &D = \begin{cases} -A_{max} & \text{if} \quad T_6 \geq 0 \\ sgn(D) \cdot \sqrt{J_{max}(f - f_e)} & \text{if} \quad T_6 < 0 \end{cases} \end{aligned}\right\} \tag{4.83}$$

The condition of total travel length must be satisfied. If there are excess travel lengths, after the lengths for acceleration and deceleration are accounted for, the constant feed stage will exist, otherwise, the reachable feed for the given length replaces the desired feed, f.

In order to generate a limited jerk trajectory profile, the inputs are:

- the length of the path, L
- the maximum bounded jerk, J_{max}
- the bounded acceleration, A_{max}
- the sampling time, T_s
- the initial feed, f_s
- the desired running feed, F
- the end feed, f_e

The goal is to maximize the nominal running feed, while not violating the constraints. By assigning a constant bounded acceleration, A_{max}, and jerk, J_{max}, for the entire path, the only adjustment parameters are the nominal feedrate, F, and the end feed, f_e. Rather than using constant lengths as incremental steps, a number of minimum peaks are taken from the constraints curve. The starting feed, f_s, is continued from the previous segment, and the end feed, f_e, is set to be just below the next minimum constraint value. The outline of the process is shown in the following Figure 4.8:

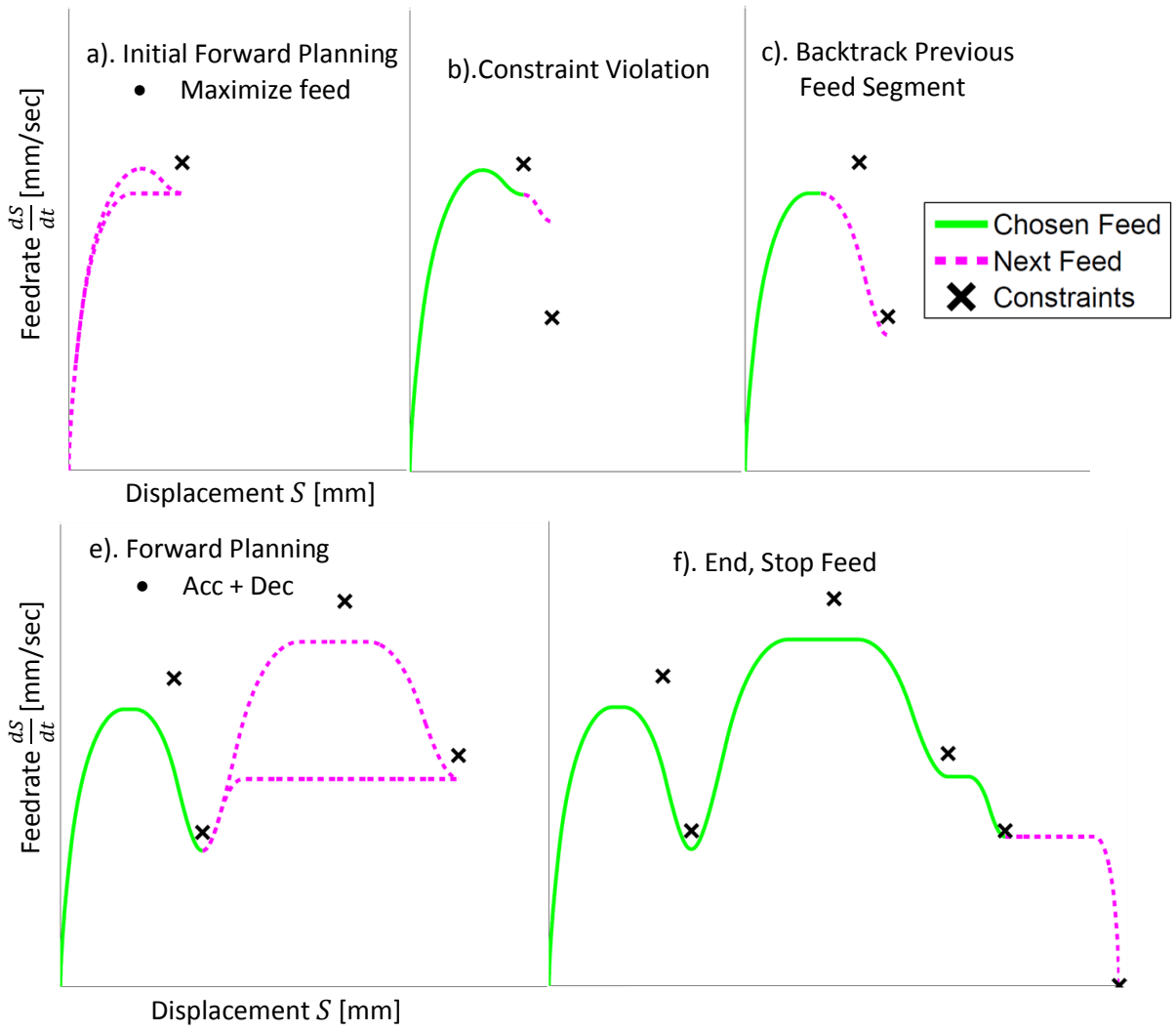


Figure 4.8: Multi-Segment Limited Jerk Feed Profile Planning

A major advantage of applying the multi-segmenting method to a well-defined trajectory profile is that the feasible search region is known based on the current feedrate. Initially, the feed can be set as high as possible with the ending feed meeting the first minimum constraint. The nominal feed can be lowered if the constraints are violated midway. In the case that the next constraint is lower than the current constraint, i.e. deceleration, the end feed boundary condition needs to be met with the available length between the two constraints. If this is not possible, as shown in Figure 4.8b), the previously chosen feed segment has to be backtracked.

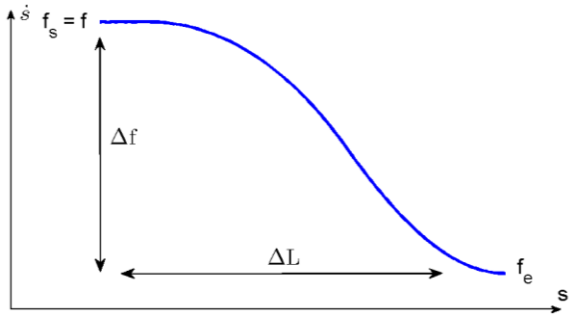


Figure 4.9: Required Length for Deceleration

In the case of deceleration as shown in Figure 4.9, the nominal feed is the same as the starting feed, then the length travelled by the deceleration portion of the profile is the following:

$$\left.\begin{aligned} L &= fT_5 + \frac{J_5}{6}T_5^3 + f_5T_6 + \frac{D}{2}T_6^2 + f_6T_7 + \frac{D}{2}T_7^2 + \frac{J_7}{6}T_7^2 \\ L &= f\frac{D}{J_5} + \frac{D^3}{6{J_5}^2} + \left(f + \frac{D^2}{2J_5}\right)T_6 + \frac{D}{2}{T_6}^2 + \left(f + \frac{D^2}{2J_5} + DT_6\right)\frac{D}{J_5} + \frac{D^3}{2{J_5}^2} - \frac{D^2}{6J_5} \\ L &= \frac{D}{2}{T_6}^2 + \left(f + \frac{3D^2}{2J_5}\right)T_6 + \frac{2D}{J_5}f + \frac{D^3}{{J_5}^2} \end{aligned}\right\} \quad (4.84)$$

where time period for the constant deceleration stage can be expressed as:

$$T_D = T_6 = \frac{\Delta f}{D} - \frac{D}{J_5} \quad (4.85)$$

If $T_6 < 0$, the maximum deceleration value is reduced and the new length becomes:

$$\left.\begin{aligned} T_6 &= 0; \quad D = -\sqrt{|J_5(\Delta f)|} \\ L &= -\frac{2\sqrt{|J_5(\Delta f)|}}{J_5}f + \frac{-(|J_5(\Delta f)|)^{\frac{3}{2}}}{{J_5}^2} \end{aligned}\right\} \quad (4.86)$$

Given the current feedrate of, f, and the change in feedrate, Δf, required to meet the next constraint, the minimum length required to satisfy this is given as:

$$L_{min} = \begin{cases} \dfrac{-A_{max}}{2}(T_D)^2 + \left(f - \dfrac{3A_{max}^2}{2J_{max}}\right)(T_D) + \dfrac{2A_{max}}{J_{max}}f - \dfrac{A_{max}^3}{{J_{max}}^2} & \text{if} \quad |\Delta f| \geq \dfrac{A_{max}^2}{J_{max}} \\ \dfrac{2\sqrt{|J_{max}(\Delta f)|}}{J_{max}}f - \dfrac{(|J_{max}(\Delta f)|)^{\frac{3}{2}}}{{J_{max}}^2} & \text{if} \quad |\Delta f| < \dfrac{A_{max}^2}{J_{max}} \end{cases} \quad (4.87)$$

Denoting the available length for the current step as ΔL, L_{min} as the minimum to realize the desired Δf transition, then the amount to back track into the previously chosen feed is:

$$L_{Back} = L_{min} - \Delta L \tag{4.88}$$

This leaves a shorter working length for the previous segment to satisfy its own Δf transitions. If the previous segment had been a deceleration, that had barely made its end boundary constraint, backtracking might be required again.

$$L_{k-1}^{new} = L_{k-1}^{old} - L_{Back} \tag{4.89}$$

Note that Eqn. (4.87) can also be written for the minimum length required to accelerate to a desired feed as:

$$L_{min} = \begin{cases} \dfrac{A_{max}}{2}(T_a)^2 + \left(f_s + \dfrac{3A_{max}^2}{2J_{max}}\right)(T_a) + \dfrac{2A_{max}}{J_{max}} f_s + \dfrac{A_{max}^3}{{J_{max}}^2} & \text{if} \quad |\Delta f| \geq \dfrac{A_{max}^2}{J_{max}} \\ \dfrac{2\sqrt{|J_{max}(\Delta f)|}}{J_{max}} f_s + \dfrac{(|J_{max}(\Delta f)|)^{\frac{3}{2}}}{{J_{max}}^2} & \text{if} \quad |\Delta f| < \dfrac{A_{max}^2}{J_{max}} \end{cases} \tag{4.90}$$

where

$$T_a = \frac{f - f_s}{A_{max}} - \frac{A_{max}}{J_{max}} \tag{4.91}$$

These two expressions can also be rewritten as

$$\begin{matrix} \dfrac{1}{2A_{max}} f^2 + \dfrac{A_{max}}{2J_{max}} f - \left(\dfrac{f_s}{2A_{max}} + \dfrac{A_{max}}{2J_{max}}\right) f_s + \dfrac{A f_s}{J_{max}} - L = 0 & \text{if} \quad T_a \geq 0 \\ (|J_{max}(f - f_s)|)^{\frac{3}{2}} + 2f_s \cdot J_{max}(|J_{max}(f - f_s)|)^{\frac{1}{2}} - L \cdot J_{max}^2 = 0 & \text{if} \quad T_a < 0 \end{matrix} \tag{4.92}$$

where the roots of the quadratic and cubic can be used to solve for the maximum reachable feed given the available length, L, and the current feed, f_s. For Eqn. (4.92), the positive root is taken from the quadratic equation and the real root is taken from the cubic equation. T_a is then calculated from each feed value to determine which one is valid. This provides a limit to the search space when searching for a feed for acceleration forward planning. During acceleration planning, there is no problem if the desired end feed cannot be reached as a lower feed can guarantee that the constraints will not be violated. At the end of the profile planning process, the end constraint has to ensure that feed comes to a complete stop at the end of the toolpath, $f_e = 0$. The length of the entire toolpath had been previously found using the Quadrature length integration method. The L_{min} for the deceleration, Eqn. (4.87), can be solved to check whether

it is possible to decelerate to zero end feed given the current length left till the end of the path and the current feed value; f_s is set to f and f_e is zero. Backtracking may be required.

By using a multi-segment limited jerk profile, a smooth feedrate profile can be planned without unnecessary fluctuations from an optimization algorithm. Even though the cycle time will be increased, the computational time and the part quality will be improved.

4.7 Implementation and Experimental Results

The real time implementation of the NURBS interpolation algorithms was conducted on a high speed X-Y table as shown in Figure 4.10. The two axis router is driven by two linear DC motors. Position feedback is obtained from the incremental position encoders, with a resolution of $1\ [\mu m]$. The workspace consists of $300\ [mm]$ of stroke in both the X and Y directions.

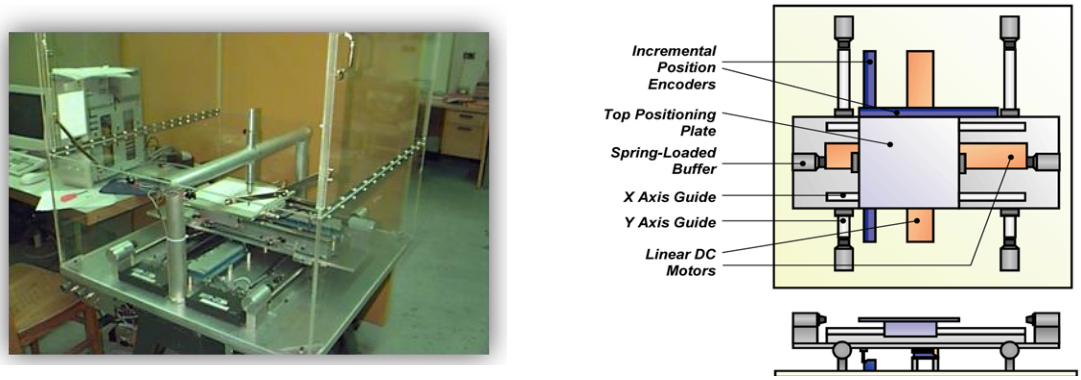


Figure 4.10: High Speed X-Y Table

The presented real time interpolation algorithms are developed using Matlab/Simulink and are implemented using the real time dSPACE hardware system. The general system block diagram used for the X-Y table implementation is shown in Figure 4.11:

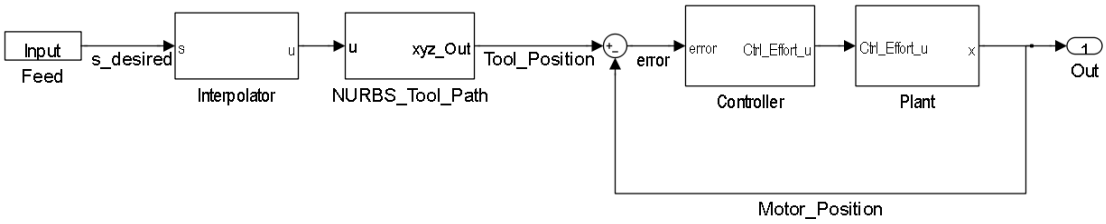


Figure 4.11: X-Y Table Implementation Block Diagram

The displacement profile is generated from feed scheduler and is used as the input. The interpolator translates the displacement into the curve parameter using one of the presented interpolation algorithms. Then the curve parameter is passed into the NURBS toolpath block

where the NURBS equation is evaluated to extract the tool position. A Simulink S-function is used for the computation of the NURBS equation, allowing for customized C code to be written. The curve parameter value is compared to the knot vector to determine the non-zero B-spline basis functions to evaluate and avoid unnecessary computations. When there are multiple curves, the control points and the knot vectors for the particular curve are also selected. Following the NURBS toolpath evaluation, the tool position reference command is sent into the position control loop which is used to control the motors using the analog output ports of the dSPACE board. The sampling time for the interpolation and the position control loop is $T_s = 1\,[\mathrm{ms}]$.

The test toolpath was generated from 89 fan-shaped data points that were scaled to an overall size of $125\,[\mathrm{mm}] \times 125\,[\mathrm{mm}]$ as shown in Figure 4.12. A quintic NURBS toolpath was fitted with 55 control points, using the fitting algorithm outlined in Chapter 3, with the following multi-objective fitting coefficients: $\gamma = 1;\ \alpha = 10^{-8};\ \beta = 10^{3}$. The total path length was found to be $602.075\,[\mathrm{mm}]$. From the curvature graph shown in Figure 4.13, there are 4 sharper curves along the toolpath.

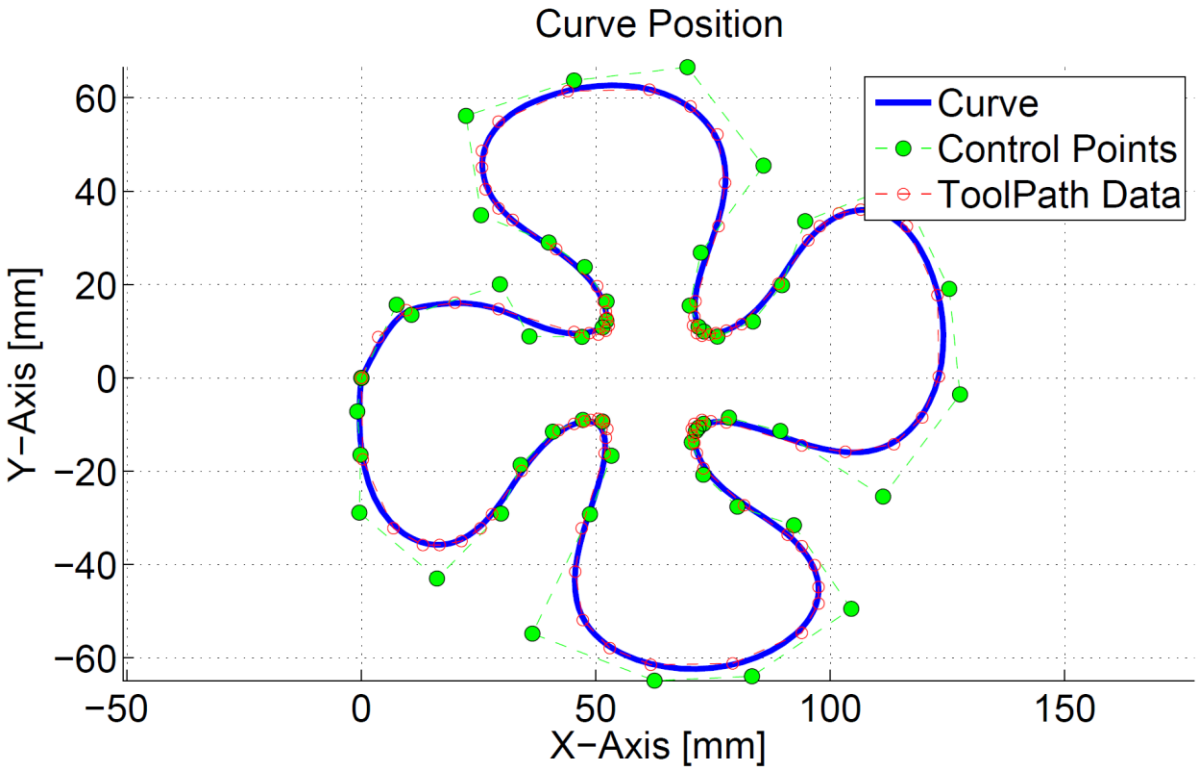


Figure 4.12: Fan-Shaped NURBS Toolpath

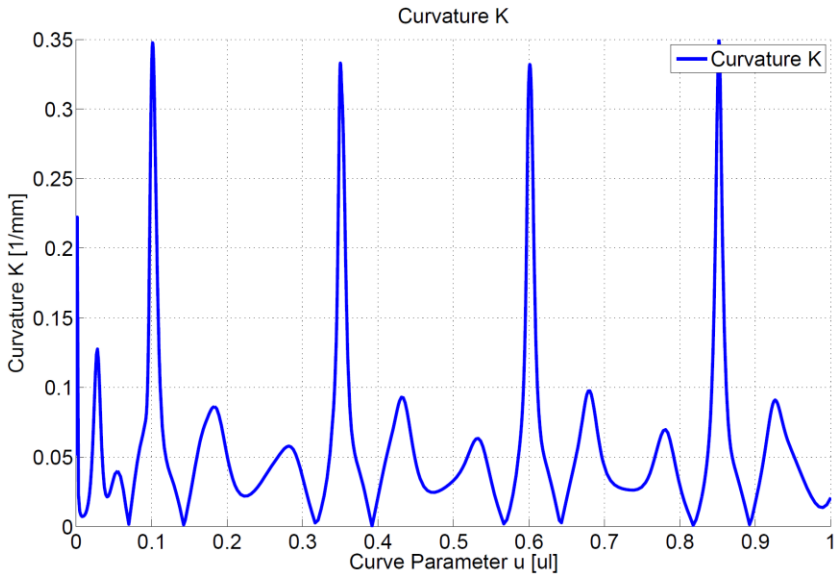


Figure 4.13: Fan-Shaped NURBS Toolpath Curvature

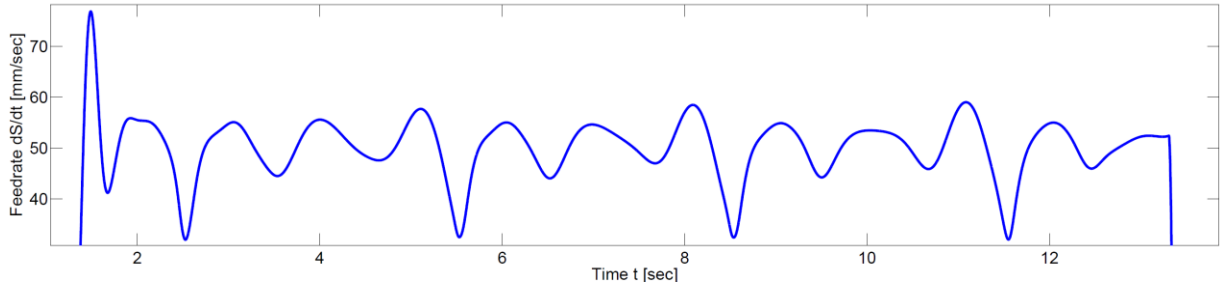
In order to compare the performance of the interpolation methods in terms of feedrate consistency, and fluctuations, a constant feedrate of $50\ [\mathrm{mm/sec}]$ profile is generated using limited jerk. The maximum acceleration and jerk for the toolpath were set at $4905\ [\mathrm{mm/sec^2}]$ and $5 \times 10^4\ [\mathrm{mm/sec^3}]$. To calculate and compare the feed of the tool reference command, the derivatives are calculated numerically over two sample points. The derivative of the x and y positions are estimated using:

$$\frac{d\hat{x}_k}{dt} = \frac{x_{k+1} - x_{k-1}}{2 \cdot T_s};\ \frac{d\hat{y}_k}{dt} = \frac{y_{k+1} - y_{k-1}}{2 \cdot T_s};\ 2 \le k \le N_t - 1 \qquad (4.93)$$

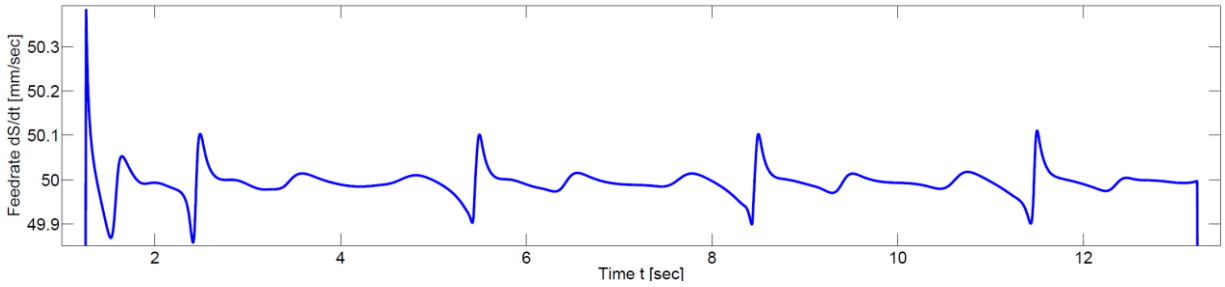
where k is the sample index, denoting the time $t = k \cdot T_s$ and N_t is the total number of samples. The same numerical calculation is then used to estimate the feedrate $\frac{d\hat{s}_k}{dt}$, acceleration $\frac{d^2\hat{s}_k}{dt^2}$, and jerk $\frac{d^3\hat{s}_k}{dt^3}$ of the toolpath as:

$$\frac{d\hat{s}_k}{dt} = \sqrt{\left(\frac{d\hat{x}_k}{dt}\right)^2 + \left(\frac{d\hat{y}_k}{dt}\right)^2};\ \frac{d^2\hat{s}_k}{dt^2} = \frac{\frac{d\hat{s}_{k+1}}{dt} - \frac{d\hat{s}_{k-1}}{dt}}{2 \cdot T_s};\ \frac{d^3\hat{s}_k}{dt^3} = \frac{\frac{d^2\hat{s}_{k+1}}{dt^2} - \frac{d^2\hat{s}_{k-1}}{dt^2}}{2 \cdot T_s} \qquad (4.94)$$

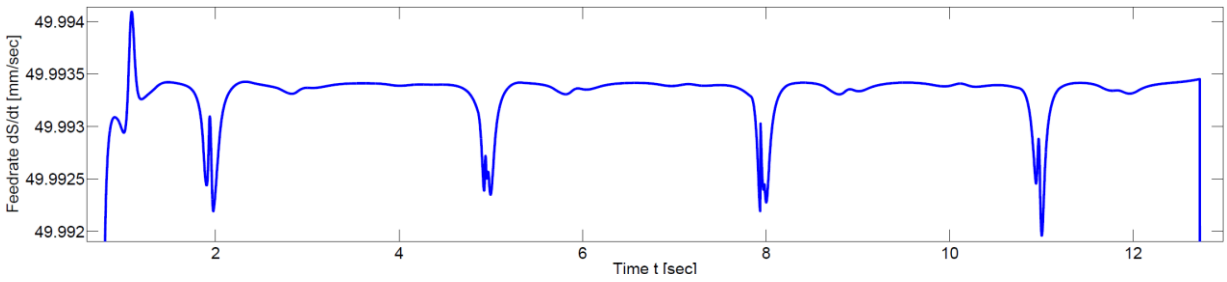
The feedrate at the reference to the position control loop is shown in the following Figure 4.14:



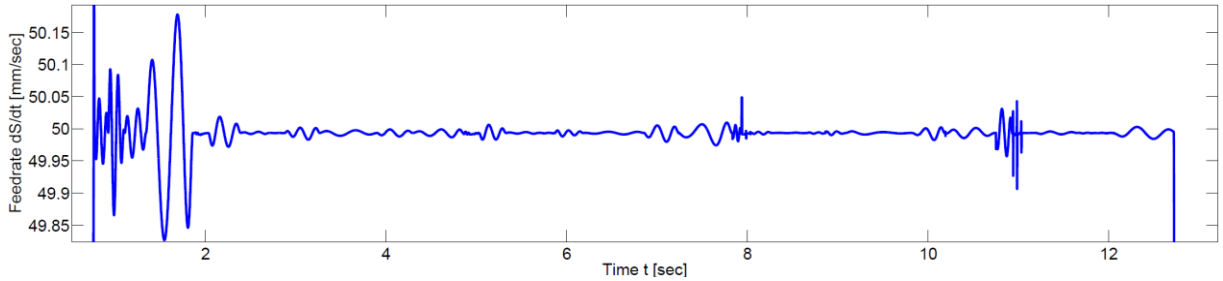
a) Natural Interpolation



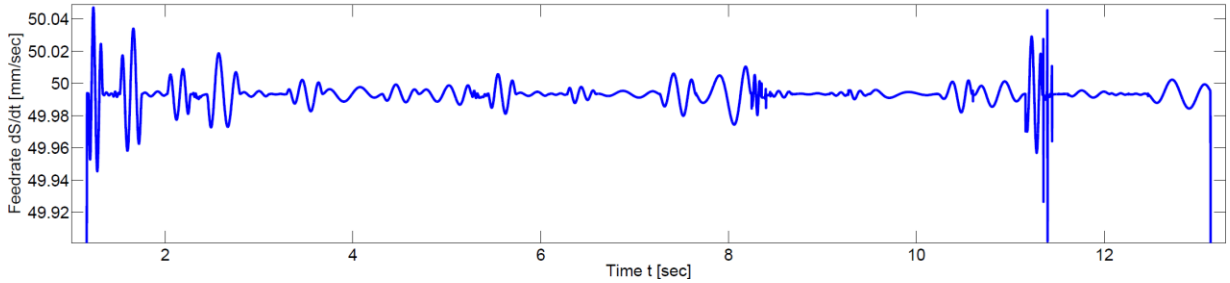
b) 1st Order Taylor Expansion



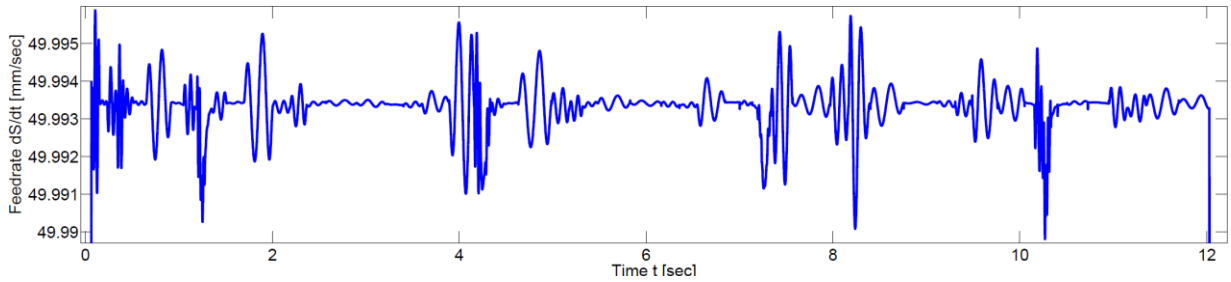
c) 2nd Order Taylor Expansion



d) Feed Correction Polynomial $\varepsilon_{MSE} = 1 \times 10^{-8}$



e) Feed Correction Polynomial $\varepsilon_{MSE} = 1 \times 10^{-10}$



f) Feed Correction Polynomial $\varepsilon_{MSE} = 1 \times 10^{-12}$

Figure 4.14: Feedrate Fluctuation for Interpolation Algorithms

With natural interpolation, the feedrate at the position reference can be seen to reach as high as $75\ [\mathrm{mm/sec}]$, and as low as $35\ [\mathrm{mm/sec}]$, showing a maximum fluctuation of about $35\ [\%]$ and a mean fluctuation of about $10\ [\%]$. Since the natural interpolation method increments the curve parameter in portion to the arc length, fluctuations occur due to the nonlinearity between the curve parameter and the curve displacement. A plot of the curve parameter and displacement relationship shown in Figure 4.15 between the multi-segment correctional polynomial with a mean square error fit of $\varepsilon = 10^{-8}$ on the (u_i, s_i) data from the numerical adaptive integration and a linear line shows that the difference between them is as large as $2\ [\mathrm{mm}]$. This difference shows that it has very similar shape as the feed fluctuations as seen in Figure 4.14a) above.

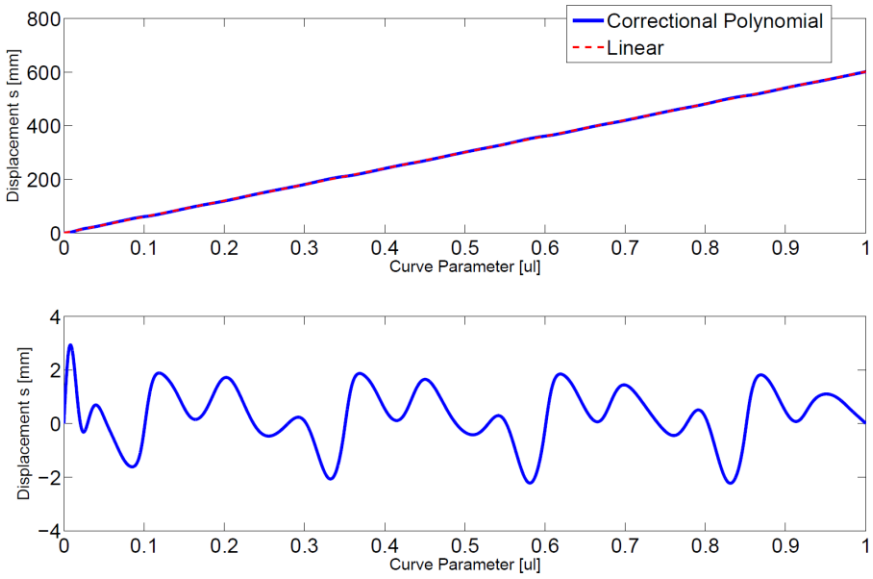


Figure 4.15: Nonlinear Curve Parameter and Displacement Relationship

For the 1st order Taylor interpolation method in Figure 4.14b)., it shows decent feedrate integrity along the toolpath except for the four sharp locations. This is expected as the slope is very volatile at locations of high curvature and cannot be accurately used to predict the curve parameter and displacement relationship. It shows good results with only a maximum fluctuation of $0.24\ [\%]$. The 2nd order Taylor interpolation in Figure 4.14c). shows huge improvement in terms of feed fluctuation, with a maximum fluctuation of $0.00285\ [\%]$, however, the maximum fluctuations are also occurring at the high curvature peaks. In addition, the computation time of the 2nd order Taylor method has increased over the 1st order method as a higher order computation was required. The correctional polynomial with a mean square fitting error of $\varepsilon_{MSE} = 1 \times 10^{-8}$ shown in Figure 4.14d). shows comparable feed fluctuation results as the 1st order Taylor method, with a maximum fluctuation of $0.358\ [\%]$. However, the computation time for the feed correction polynomial methods are only about $0.02\ [\mathrm{msec}]$, since a series of pre-fitted polynomial coefficients are already stored in look-up tables. This short computation time is ideal for real time implementations. A summary of the feed fluctuations and computation times are listed in Table 4.1. Feed fluctuation gets better when using a tighter mean square error tolerance to fit the correction polynomials. At $\varepsilon_{MSE} = 1 \times 10^{-12}$, in Figure 4.14e)., feed fluctuation is a lot better at a maximum of $0.00480\ [\%]$ and is comparable to the 2nd order Taylor method, while have very low computation time.

Interpolation Type	Feed Fluctuation (Max) [%]	Feed Fluctuation (Average) [%]	Computation Time [msec]
Natural Interpolation	35	10	0.0128
1st Order Taylor	0.24	0.049	6.786
2nd Order Taylor	0.00285	0.000158	7.116
Correction Polynomial: $\varepsilon_{MSE} = 1 \times 10^{-8}$	0.358	0.0253	0.0204
Correction Polynomial: $\varepsilon_{MSE} = 1 \times 10^{-10}$	0.183	0.0120	0.0212
Correction Polynomial: $\varepsilon_{MSE} = 1 \times 10^{-12}$	0.00480	0.000810	0.0202

Table 4.1: Interpolation Methods Comparison of Feed Fluctuation and Computation Time

To validate the feasibility of the presented multi-segmenting method of the limited jerk feed profile, the fan-shaped toolpath is tested with respect to the kinematic constraints of the X-Y table. The velocity, acceleration, and jerk limits of the linear drives are given in the following Table 4.2:

X-Y Table Limits	Velocity Limits	Acceleration Limits	Jerk Limits
X Linear Drive	$338.67 \frac{\text{mm}}{\text{sec}}$	$9810 \frac{\text{mm}}{\text{sec}^2}$	$10^5 \frac{\text{mm}}{\text{sec}^3}$
Y Linear Drive	$338.67 \frac{\text{mm}}{\text{sec}}$	$49050 \frac{\text{mm}}{\text{sec}^2}$	$10^5 \frac{\text{mm}}{\text{sec}^3}$

Table 4.2: X-Y Table Kinematic Limits

The maximum feedrate limit was set at $200\ [\text{mm/sec}]$, with acceleration and jerk bounded at $4905\ [\text{mm/sec}^2]$ and $5 \times 10^4\ [\text{mm/sec}^3]$. The maximum allowable chord error was set at $1\ [\mu\text{m}]$ and the feedrate is limited to half of the maximum feedrate when the curvature gets to $2\ [\text{mm}^{-1}]$. The feedrate generation process took $7.61\ [\text{sec}]$ to generate the constraints curve, and the feasible multi-segment feed profile; a 2.16 GHz Pentium Dual-Core laptop computer with 3 GB of RAM was used running MATLAB. In Figure 4.16, the constraints curve is seen to be dominated by the kinematic jerk constraints of the x and y drives.

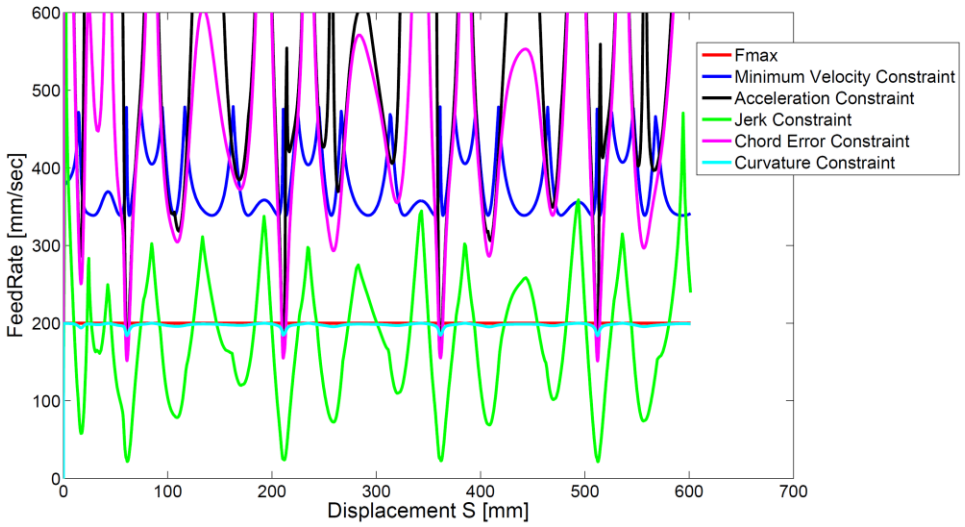


Figure 4.16: Feedrate Constraints

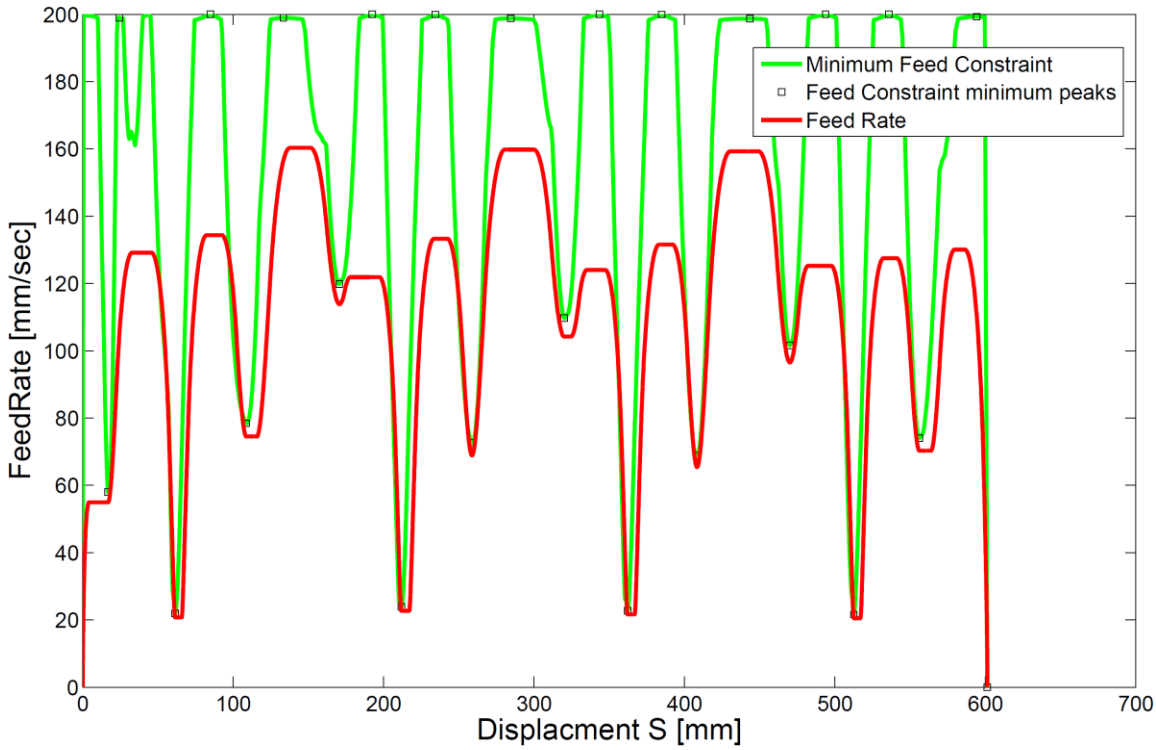


Figure 4.17: Multi-Segment Limited Jerk Feed Profile

A multi-segment limited jerk feed profile is scheduled as described in Section 4.5. The resulting feed profile is seen in Figure 4.17 to be quite adaptive to the shape of the fan, as there are four low feedrate segments for the sharp curves along the toolpath, followed by straighter segments, which lead to a smaller curve. At the sharp regions, the feedrate is $20\ [\mathrm{mm/sec}]$. The total motion duration for this profile is $6.795\ [\mathrm{sec}]$, for a travelling displacement of

601.371 [mm], while a constant feedrate at 20 [mm/sec] through the entire toolpath would require 30.16 [sec] and a constant feedrate at 50 [mm/sec] would require 12.17 [sec]. In addition, with a constant feedrate at 50 [mm/sec], the kinematic jerk constraints of the linear drives will be violated at the four sharp locations. The linear drives were controlled using a loop shaping controller with a bandwidth of 10 [Hz]. The following Table 4.3 shows the experimentally identified drive dynamics of the X-Y table:

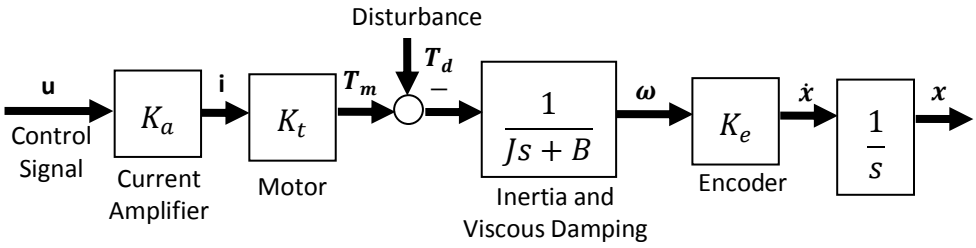


Figure 4.18: Linear Rigid Body Feed Drive Dynamics Model

	X-Axis	Y-Axis
Amplifier Gain: K_a [A/V]	0.28568	0.2802
Motor Force Constant: K_t [N/A]	99.5447	135.1173
Encoder Gain: K_e [mm/m]	1000	1000
Mass: m [kg]	5.447	31.482
Friction: b [Nsec/m]	18.0928	124.9376

Table 4.3: X-Y Table Drive Dynamics

The Feed Correction Polynomial with a mean square error of $\varepsilon_{MSE} = 10^{-12}$ [ul] was used with (u_i, s_i) data that was generated with an integration tolerance of $\varepsilon_S = 10^{-12}$ [ul]. The results can be summarized by the kinematic profiles of the resulting tool motion as shown:

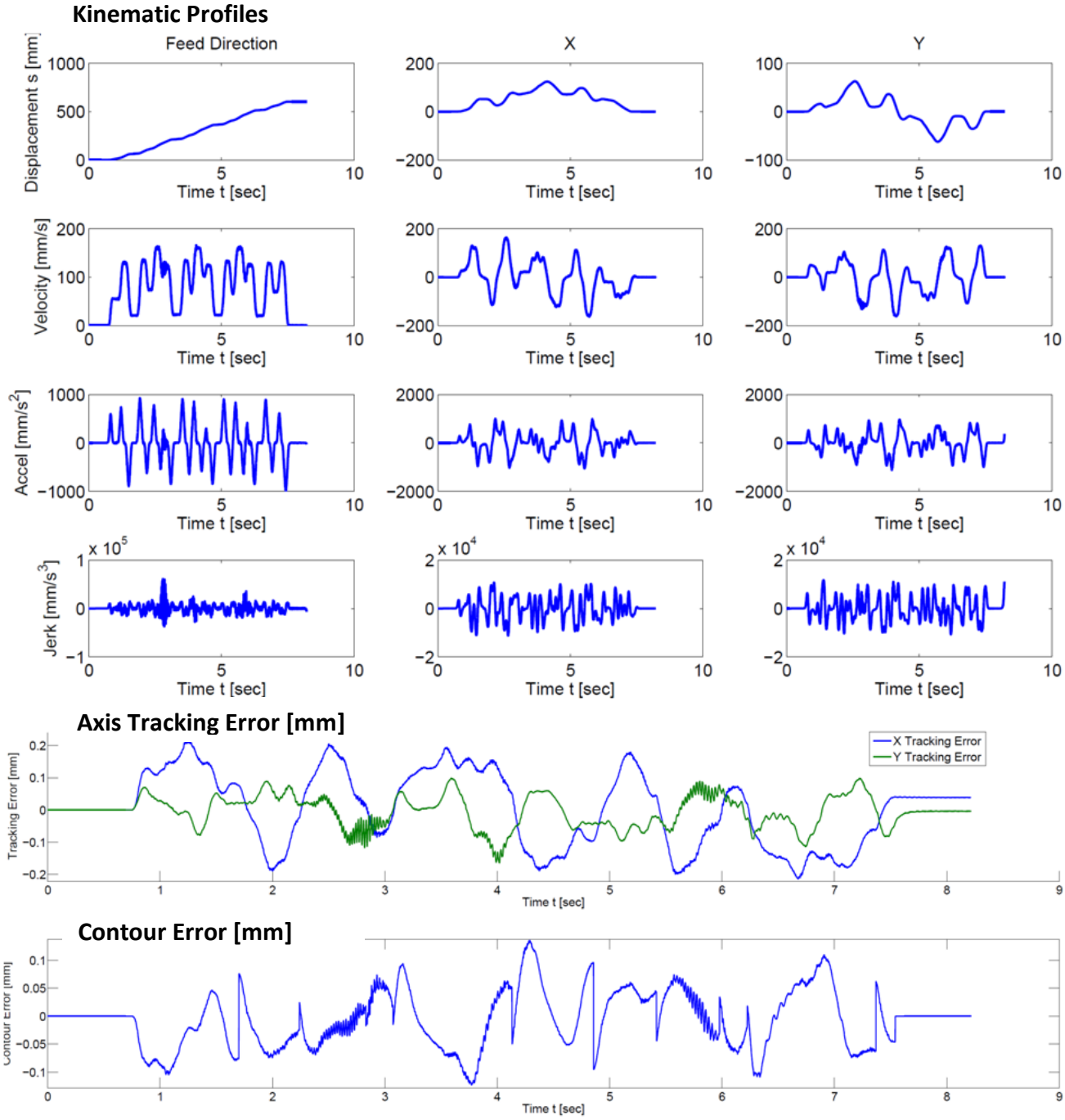


Figure 4.19: Kinematic and Error Profiles of Fan-Shaped NURBS Contouring

From the kinematic profiles in Figure 4.19, none of the axes exceeded the specified kinematic velocity, acceleration and jerk limits. The maximum recorded tracking error is 0.215 [mm] for the x axis and 0.165 [mm] for the y axis. Contour error is calculated as:

$$\varepsilon_c = -E_x \cdot \sin\phi + E_y \cdot \cos\phi \tag{4.95}$$

where ϕ is the angle between the tangent of the tool path command with the x axis and E_x and E_y are the x and y tracking errors and a maximum of $0.136\ [\mathrm{mm}]$ of contour error was recorded. From the results, the multi-segmenting method of feed profiling shows its feasibility in computation time and in cycle time. While the tool travelled at a high speed, the large size sample toolpath showed decent contouring capabilities and none of the kinematic limitations for the motors were violated.

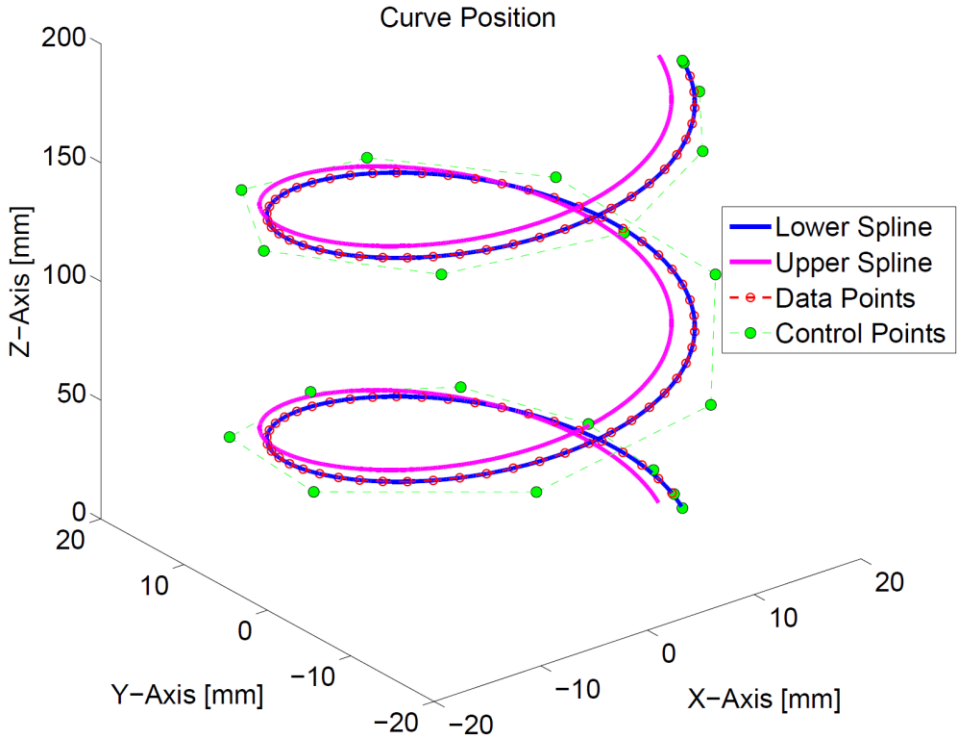


Figure 4.20: 5-Axis Spiral Toolpath

Next, in order to validate the feasibility of using the double spline, as well as the inclusion of the five-axis kinematics of the CNC machine into feedrate scheduling, a sample 5-axis toolpath as shown in Figure 4.20 is tested through simulation. A uniform spiral toolpath is used where the tool position with respect to the workpiece and the rotary motor axes are given by the following equations:

$$\left.\begin{array}{c} T_{xref}(u) = 15\cos(u) + 5\sin(u)\ [mm] \\ T_{yref}(u) = 15\sin(u) - 5\cos(u)\ [mm] \\ T_{zref}(u) = 15u\ [mm] \\ A_{ref}(u) = -\cos^{-1}\left(\frac{sin(u)}{\sqrt{5}}\right)\text{[rad]} \\ C_{ref}(u) = -\tan^{-1}\left(\frac{\cos(\mathrm{u})}{\sqrt{5}}\right)\text{[rad]} \\ 0 \le u \le 2\pi \end{array}\right\} \quad (4.96)$$

100 data points are extracted from Eqn. (4.96) linearly in terms of the parameter u to get the orientation and the tool position information, and 21 control points were used to fit these data points using the presented algorithms toolpath generation algorithms. The motor drive constraints used to form the feedrate constraints are listed in Table 4.5. The acceleration and jerk values for the tangential feed are bounded at $1000\ [\mathrm{mm/sec^2}]$ and $10^4\ [\mathrm{mm/sec^3}]$. The resulting feed profile is shown in Figure 4.21 where the maximum feed is at $84\ [\mathrm{mm/sec}]$ and the total tool motion duration is $4.19\ [\mathrm{sec}]$.

Drives	Velocity [unit/sec]	Acceleration [unit/sec^2]	Jerk [unit/sec^3]
X [mm]	250	2000	30000
Y [mm]	250	2000	30000
Z [mm]	250	2000	30000
A [rad]	2.897	34.907	200
C [rad]	5.061	43.633	250

Table 4.4: 5-Axis Drive Limits

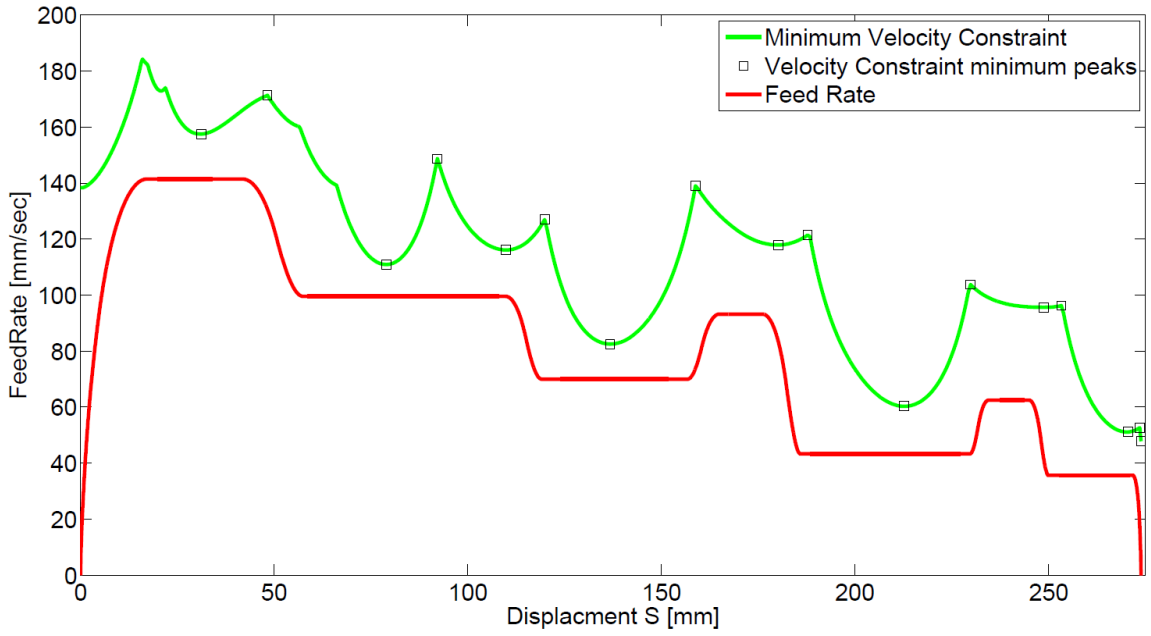


Figure 4.21: 5-Axis Sample Toolpath Feed Profile

This was tested in simulation using the block diagram shown in Figure 4.22. A tolerance of $\varepsilon = 10^{-12}$ [ul] was used to generate the (u_i, s_i) data points from the numerical adaptive integration of the toolpath length, leading to a total tool path length of 273.860 [mm]. A mean square error of $\varepsilon_{MSE} = 10^{-12}$ [ul] was used to fit the relationship between the curve parameter and toolpath displacement relationship and it resulted in 16 segments of the Feed Correction polynomial.

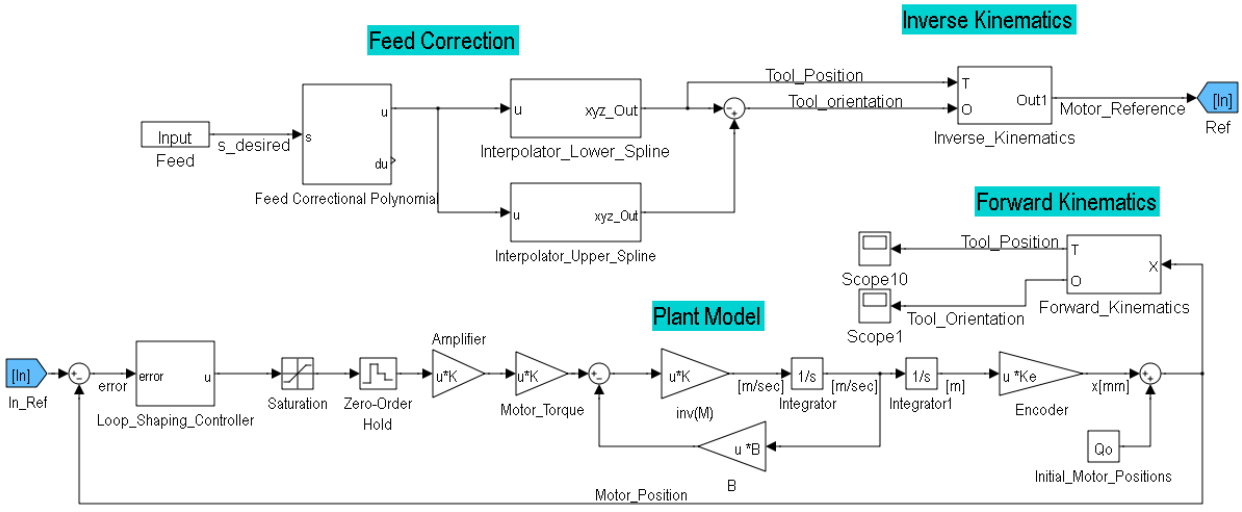


Figure 4.22: 5-Axis Simulation Block Diagram

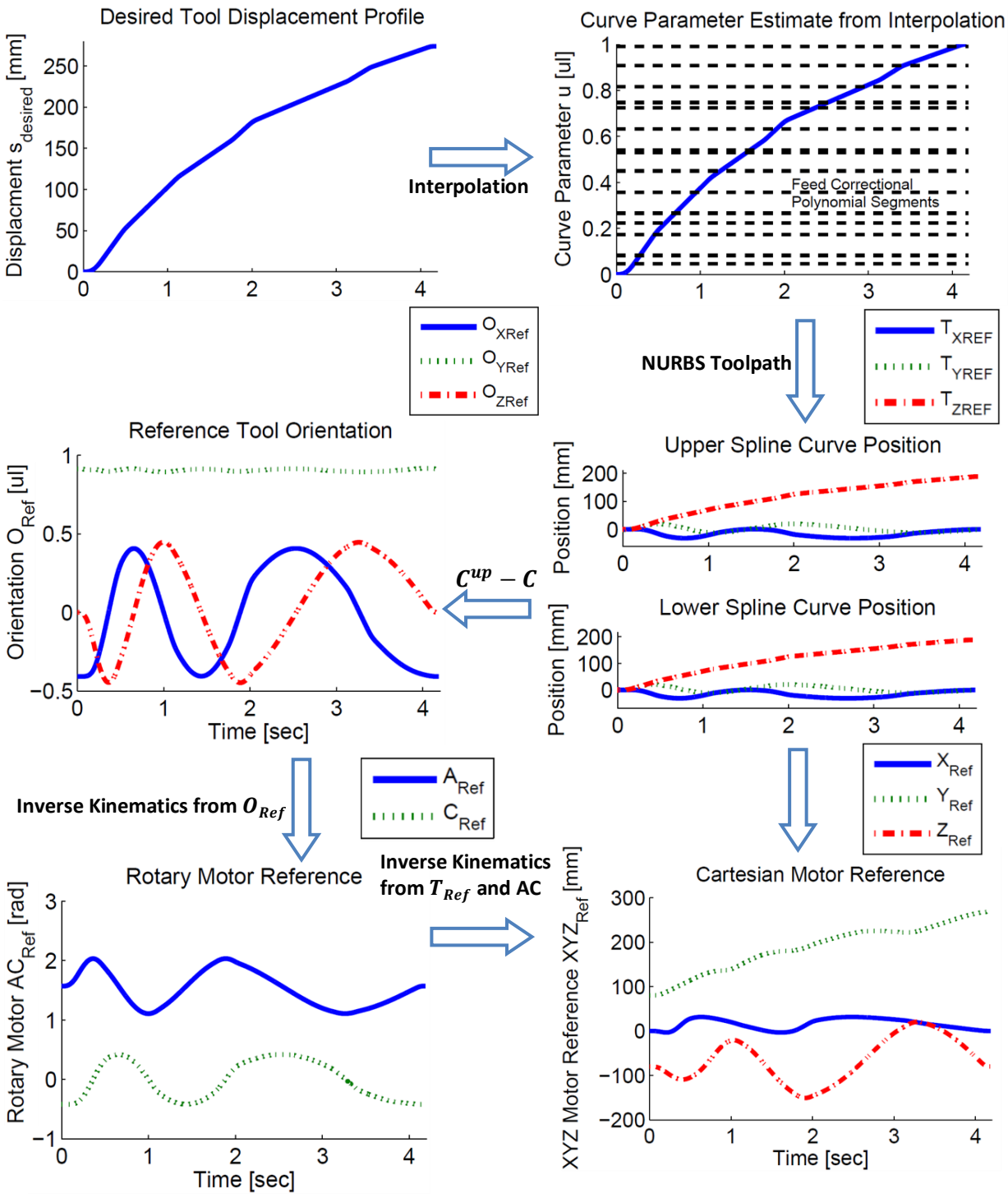


Figure 4.23: 5-Axis Simulation: Motor Reference Command Generation

Figure 4.23 shows the graphs for the motor reference generation process. From the feed scheduler, the displacement profile is interpolated at every sampling time using the correctional polynomial. The valid correction polynomial segment is selected based on the current displacement. The current displacement is then normalized based on the displacement range of the polynomial where the estimate of the curve parameter is obtained from the normalized displacement and the coefficients of the polynomial. The curve parameter estimate is then used to obtain the positions of both the upper and lower splines. The lower spline position is subtracted from the upper spline and normalized to obtain the reference orientation given by the double spline. Using the obtained orientation, the position reference of the Rotary drives AC can be obtained, then along with the tool tip reference of the lower spline, the Cartesian Motor drives XYZ references are also obtained through inverse kinematics. The following Table 4.5 shows the drive dynamics used for the simulation:

	X-axis	Y-axis	Z-axis	A-axis	C-axis
$m = \frac{J}{K_a K_t K_e}$	0.00162 $[\text{Vsec}^2/\text{mm}]$	0.00174 $[\text{Vsec}^2/\text{mm}]$	0.00296 $[\text{Vsec}^2/\text{mm}]$	0.00682 $[\text{Vsec}^2/\text{rad}]$	0.00054 $[\text{Vsec}^2/\text{rad}]$
$c = \frac{B}{K_a K_t K_e}$	0.00681 $[\text{Vsec}/\text{mm}]$	0.00863 $[\text{Vsec}/\text{mm}]$	0.01518 $[\text{Vsec}/\text{mm}]$	0.01964 $[\text{Vsec}/\text{rad}]$	0.00368 $[\text{Vsec}/\text{rad}]$

Table 4.5: 5-Axis Simulation Drive Dynamics

The feed drive dynamics are simplified as shown in Figure 4.24 and are written in a matrix form:

$$\left.\begin{gathered}\begin{bmatrix} m_X & 0 & 0 & 0 & 0 \\ 0 & m_Y & 0 & 0 & 0 \\ 0 & 0 & m_Z & 0 & 0 \\ 0 & 0 & 0 & m_{\theta_a} & 0 \\ 0 & 0 & 0 & 0 & m_{\theta_c} \end{bmatrix}\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \\ \ddot{\theta}_a \\ \ddot{\theta}_c \end{bmatrix} + \begin{bmatrix} c_X & 0 & 0 & 0 & 0 \\ 0 & c_Y & 0 & 0 & 0 \\ 0 & 0 & c_Z & 0 & 0 \\ 0 & 0 & 0 & c_{\theta_a} & 0 \\ 0 & 0 & 0 & 0 & c_{\theta_c} \end{bmatrix}\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{\theta}_a \\ \dot{\theta}_c \end{bmatrix} = \begin{bmatrix} u_X \\ u_Y \\ u_Z \\ u_{\theta_a} \\ u_{\theta_c} \end{bmatrix} - \begin{bmatrix} T_{d_X} \\ T_{d_Y} \\ T_{d_Z} \\ T_{d\,\theta_a} \\ T_{d\,\theta_c} \end{bmatrix} \\ M\ddot{Q}(t) + C\dot{Q}(t) = u(t) - T_d(t) \\ \ddot{Q} = M^{-1}\left(u(t) - T_d(t) - C\dot{Q}(t)\right)\end{gathered}\right\} \quad (4.97)$$

where $m = J/K_a K_t K_e$, $c = B/K_a K_t K_e$, $Q(t)$ is the motor joint position, $u(t)$ is the controller control effort, and $T_d(t)$ is the disturbance. For the simulation, the disturbance used is zero. This allows the feed drive dynamics diagram to be redrawn as the following:

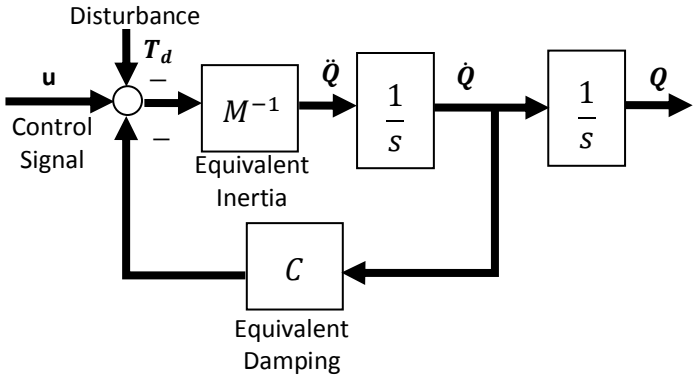


Figure 4.24: Equivalent Feed Drive Dynamics Model

The simulation was carried out using a sampling time of $T_s = 1\,[\mathrm{msec}]$, with five independent loop shaping controllers designed at a bandwidth of $15\,[\mathrm{Hz}]$, for each of the position loops of the five motor axes. The resulting motor output profile graphs are shown in Figure 4.25. The velocity and acceleration of both the Cartesian motors and the Rotary motors are within their specified drive limits. However, the jerk limitation is violated at several locations. Since the inverse kinematics of the CNC structures have shown that the Cartesian drive axes are dependent on the Rotary drives AC, violations in A and C drives have also led to violations in the X drive along several locations on the path and the Z drive at the end of the path. The performance of the position controller is also an important consideration. Even if the motor reference commands are totally within the kinematic constraints, the actual motor position is a result of position tracking. In addition, the jerk components in the reference toolpath are also inducing a transient response in the motor jerk trajectory. Figure 4.26 shows the simulated motor tracking errors. It can be clearly seen that tracking error is largest when the jerk limits of the motors were violated. In the actual CNC structure, the 5 motor positions will result in the position and orientation change of the tool. To simulate this, the motor positions are translated into the tool tip position with respect to the workpiece and into the tool orientation through forward kinematics; this is shown in Figure 4.27. Figure 4.28 shows the tracking error between the output tool tip position and tool orientation against the reference tool position and orientation from the double spline interpolation. The tool tip position tracking error reached as high as $40\,[\mu\mathrm{m}]$ where the A rotary motor was first violated at $0.17\,[\mathrm{sec}]$. The resulting tool tip displacement and feed profile is calculated off the numerical derivatives of the tool tip position and is shown in Figure 4.29. The largest differences between the initial desired displacement profile and the simulated displacement occur at the transitions between the

segments of constant feed where limited jerk was used to induce the change in the feed modulation. The transient responses that occur at these locations are leading to the large fluctuation in feedrate at these points. At the segments of constant feedrate, the resulting feed fluctuations is seen to be less than $0.152\ [\%]$. Due to the cumulative contributions of tracking error, the resulting toolpath displacement is $274.168\ [\mathrm{mm}]$ which overshot the desired tool tip displacement by $0.308\ [\mathrm{mm}]$. However, it should be noted that this resulting toolpath displacement also includes the transient fluctuations that occur throughout the entire toolpath, which manifests into the additional displacement exhibited by the tool.

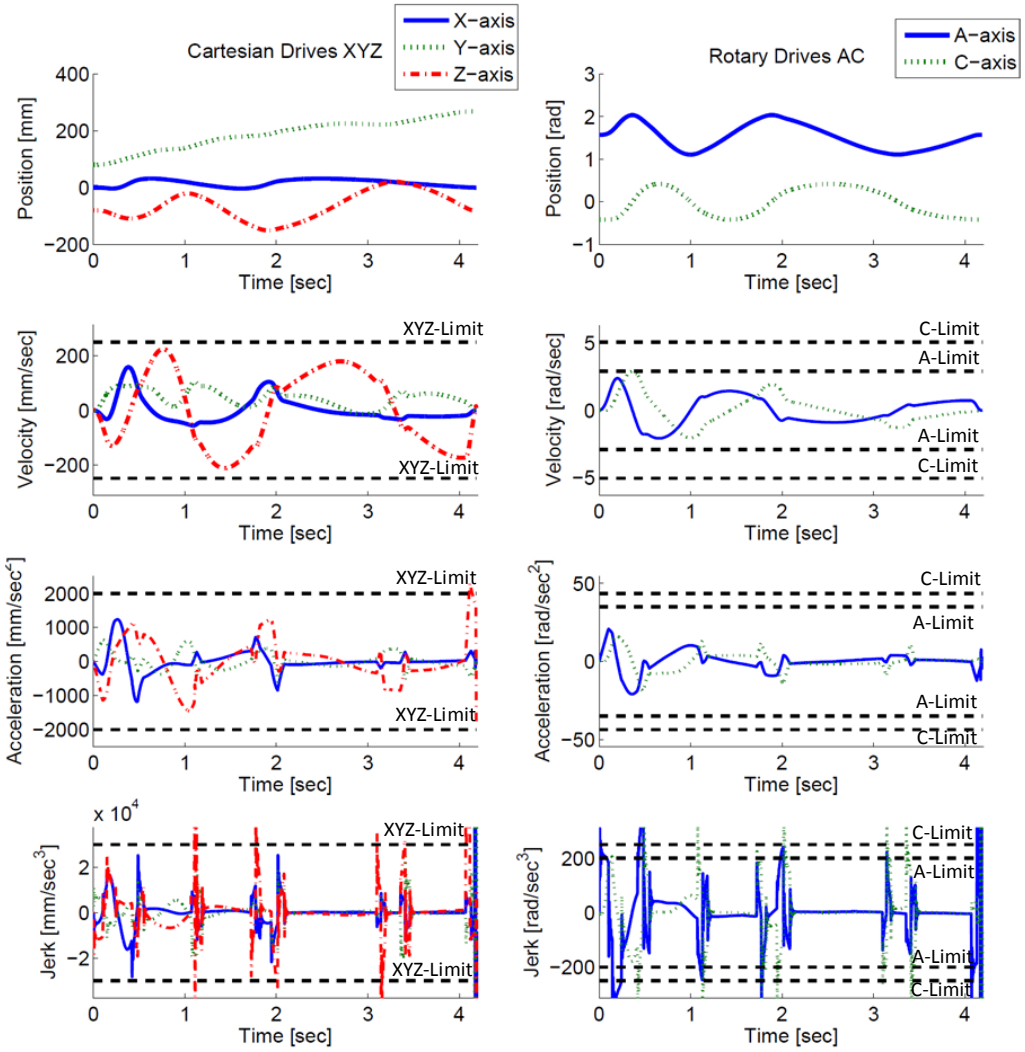


Figure 4.25: Simulation – Motor Output Profiles

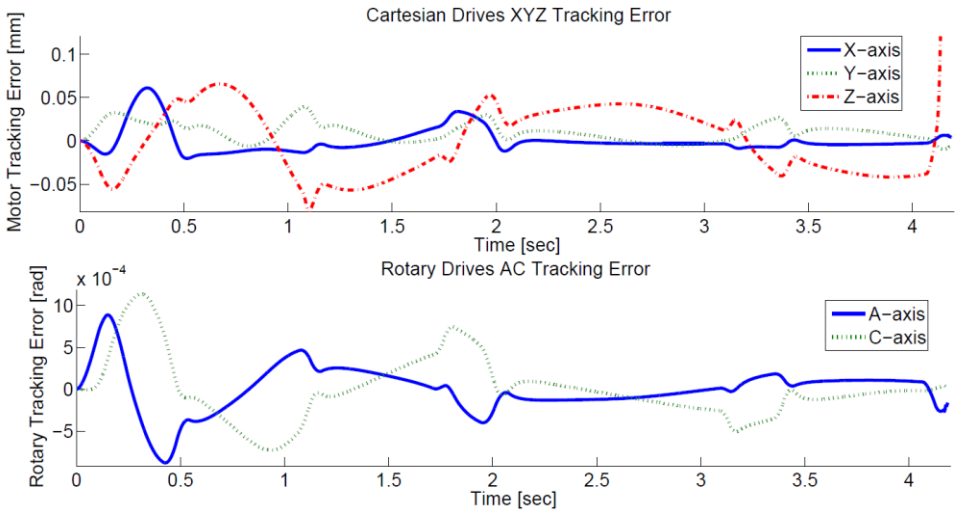


Figure 4.26: Simulation – Motor Tracking Errors

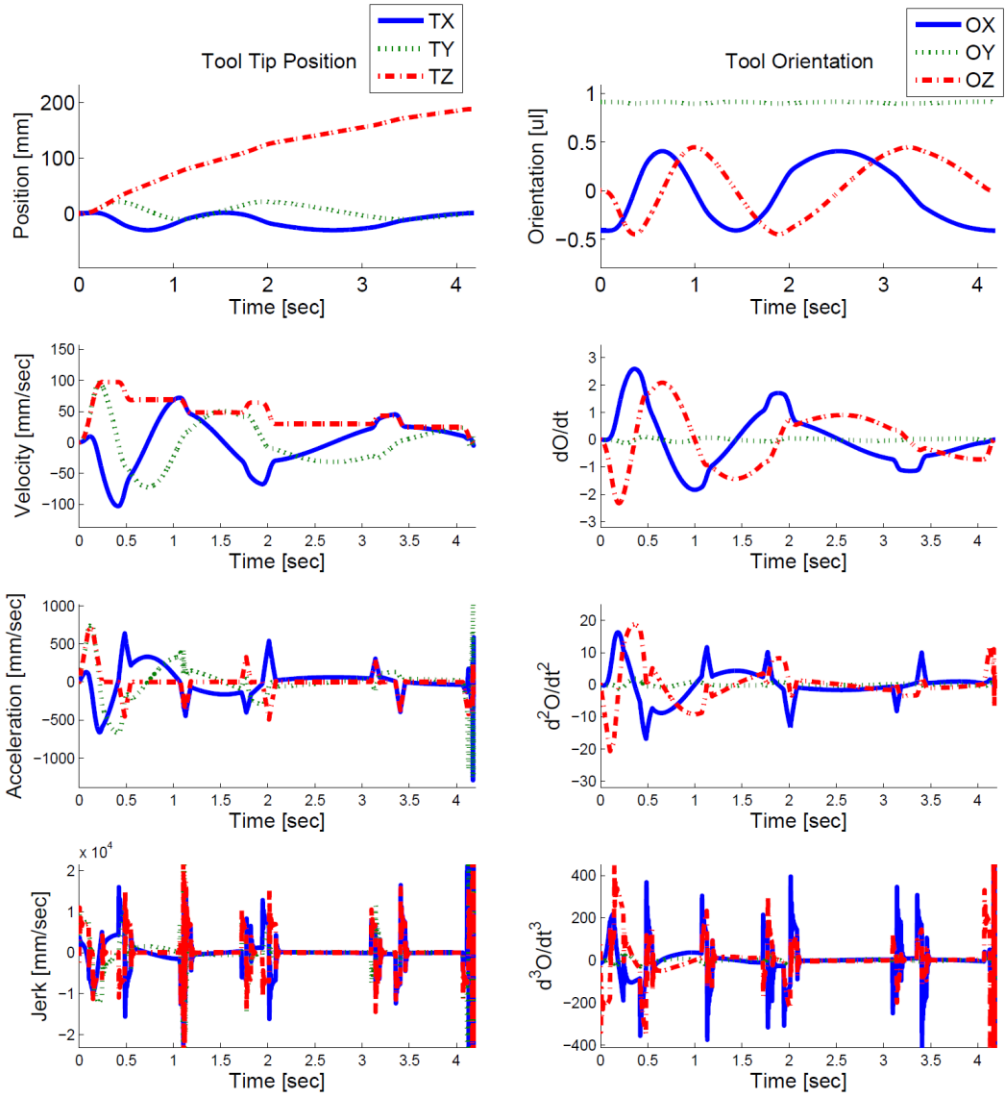


Figure 4.27: Simulation – Tool Tip Position and Tool Orientation

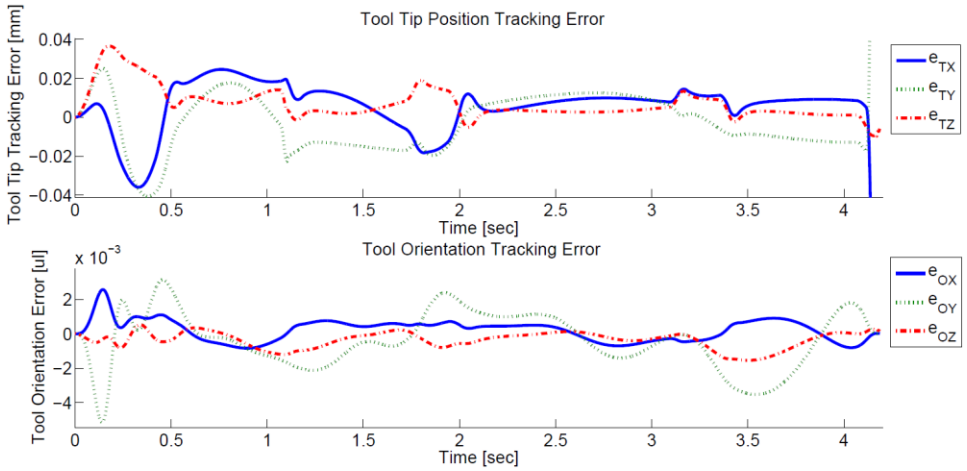


Figure 4.28: Simulation – Tool Tip Position and Tool Orientation Tracking Error

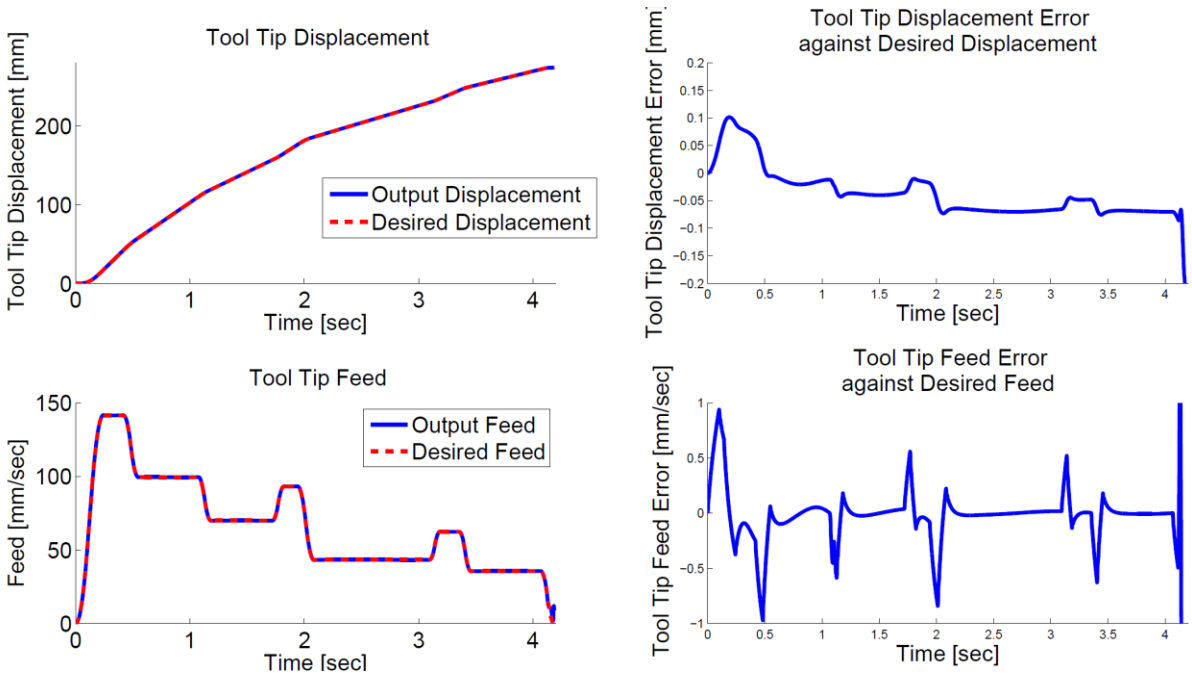


Figure 4.29: Simulation – Tool Tip Displacement and Feed Profile

4.8 Conclusion

In this chapter, the real time interpolation methods of Taylor Expansion [55] and Feed Correction Polynomial [22] were presented. Feed fluctuation is a real challenge when the relationship between the curve parameter and the displacement of the NURBS toolpath cannot be accurately represented. The 1st Order Taylor expansion has been shown to provide a decent mapping for smooth toolpaths; however, inaccuracies in the mapping where the toolpath has a large curvature can lead to undesired feedrate fluctuations. While the 2nd order Taylor Expansion showed considerable improvement in terms of feed fluctuations, the disadvantage of the Taylor Expansion in accommodating for variable feedrate and the long computation time can be problematic for real time implementations. The Feed Correction Polynomial method utilizes the highly accurate data from the adaptive quadrature integration of the toolpath length, and fits the data using multiple 7th order polynomials to capture the nonlinear relationship. It was shown that using a tighter tolerance for the fit would require additional segments of the polynomial and improve the overall feed fluctuation. While the fitting process takes increasingly more time using a tight tolerance, it can be done in pre-processing; allowing for a more practical computation time for real time implementation.

For a 5-axis CNC machine, the feed motion of the cutting tool is a combination of the 5 motor axes. In order to properly schedule the feedrate, inverse kinematics has to be applied to the reference toolpath and the reference orientation to obtain the motor reference that will realize the tool motion. The kinematics algorithms for the 5-axis rotary table CNC configuration are borrowed from Sencer [54] and a series of relationships were presented to incorporate this kinematics into feed scheduling. In order to prevent any of the motors from violating their own respective velocity, acceleration, and jerk limits, the reference toolpath and the reference orientations were mapped into the motor joint positions expressed as functions of the tool displacement. This ultimately allowed a limitation to be placed on the overall feedrate. As different portions of the toolpath have varying demands on each of the motors, a modulating feedrate is effective in dictating the tool motion and decreasing the overall cycle time. By using a multi-segment bounded jerk feedrate profile inspired by Heng [20], the feedrate can remain constant depending on the local constraints and be modulated to a different feedrate to heuristically satisfy the feedrate limitations along the entire toolpath. Since the bounded jerk profile introduced by Erkorkmaz [14] is a well-defined profile, the computation is relatively simple as opposed to an objective function optimization approach.

The presented interpolation algorithms were implemented and tested in real time on the high speed X-Y table with a sample NURBS toolpath that was generated from the toolpath fitting algorithm presented in Chapter 3 and a comparison between feedrate fluctuation and computation time was made. A variable feedrate profile was applied with the correction polynomial where the tool motion slowed down along the sharp regions of the toolpath, and travelled with high speed everywhere else. The kinematic profiles of the resulting motor motion were presented. To show the feasibility of the 5-axis toolpath using the double spline representation, and to incorporate the 5-axis dynamics into feed scheduling as well as motor reference generation, a simple and well-defined toolpath was used. A feedrate was scheduled based on the constraints imposed by the individual motor's velocity, acceleration, and jerk limitations. The results of the simulation were presented and it was found that the output motor jerk motion violated the jerk limitations for the rotary motors which also lead to the violations for the Cartesian motors. This can be attributed to the transient effects of the position control loop induced by the jerk pulses that are used to modulate the feedrate. This suggests that a smoother feedrate profile such as the cubic acceleration or the cubic jerk profile may improve on the contouring results.

Chapter 5 Hardware and Controller Implementations of the X-Y Table

5.1 Introduction

The next stage after generating the motor axis commands with feed scheduling and the interpolation of the toolpath is to realize the tool motion on the CNC machine through the proper tracking of the motors. The performance of the CNC motor controller is crucial especially in 5-axis tool motions, as the performance in one motor axis, especially for the rotary drives, directly affects the performance of every axis in the motion of the cutting tool with respect to the workpiece. In addition, in the actual cutting operation, undesired vibrations with the tool motion is particularly damaging to the overall quality of the workpiece. This chapter presents the implementation of an industrial X-Y Table using a real time control system.

The X-Y Table is an integral set-up to almost every CNC systems. This particular setup as shown in Figure 5.1 has an overall dimension of $2.06 \times 1.24 \times 0.40\ [m^3]$ with a stroke distance of $0.64\ [\mathrm{m}]$ along the X-axis and $0.40\ [\mathrm{m}]$ in the Y-axis. It is a combination of two linear stages using a ball-screw mechanism with a motor drive mounted on each axis. The control algorithms are designed with MATLAB Simulink and the commands are processed through dSPACE to control the Siemens SIMODRIVE611U motor driver. A number of different control algorithms were implemented and their performances were analyzed.

Figure 5.1: LAMB X-Y Table

5.2 System Implementation

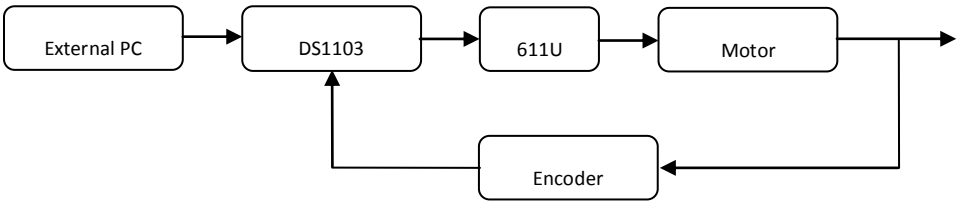
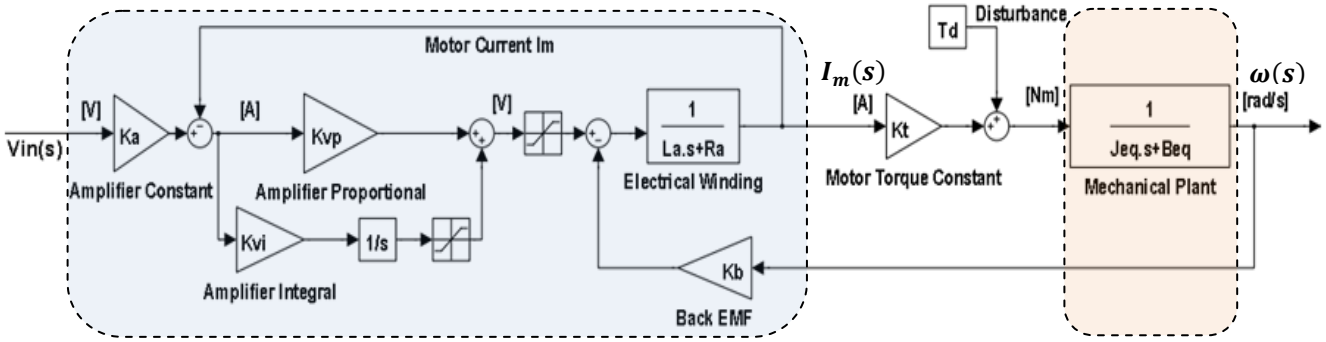


Figure 5.2: X-Y Table System Overview

This section will provide an overview of the system setup and a brief description of each of the components. A master external PC is setup to develop the control algorithms using Matlab Simulink. The control algorithms are uploaded onto the real time platform of the dSPACE DS1103 control board. Using the real time code that is compiled and uploaded on the DS1103, an analog voltage signal (-10 V to + 10 V) from the digit to analog (D/A) output of the DS1103 is sent to the SIMODRIVE 611 Universal (611U) motor driver/control board module.

The SIMODRIVE 611 Universal (611U) is a control board module providing a communication interface with the motors and encoder systems. It is a 2-axis board that provides two independent drive controls capable of driving the motors in speed, torque or position mode. The 611U can be used independently with Siemens' internal control algorithms, but it can also be used as a standard motor driver by operating in torque mode where the analog voltage signal from the DS1103 becomes the torque command for the motor. This allows custom control algorithms to be implemented. The encoder signal is fed back to both the 611U (required to operate the motor – temperature checks, etc.) and to the DS1103 to complete the control loop. The open loop diagram of torque mode operation is given in Figure 5.3 where the 611U acts as a standard motor driver and takes care of the current amplifier loop. The 611U has built-in information on the standard Siemens motors. The motors used were the three-phase Siemens Servomotors, 1FT6084.



a). Current Amplifier Loop b). Mechanical System

Figure 5.3: Torque Mode Open Loop Diagram

The transfer function of the current amplifier loop can be written as the following:

$$\frac{I_m(s)}{V_{in}(s)} = K_a \frac{\left(K_{vp} + {}^{K_{vi}}/_{s}\right)\left(J_{eq}s + B_{eq}\right)}{\left(J_{eq}s + B_{eq}\right)\left(L_a s + R_a + K_{vp} + {}^{K_{vi}}/_{s}\right) + K_b K_t} \cong K_a \text{ @ low freq} \tag{5.1}$$

Since the bandwidth of the electrical current loop is several orders higher than the mechanical system's bandwidth, the current loop can be approximated with a constant gain K_a in the operational range of the X-Y Table. The transfer function of the mechanical system and the reduced open loop transfer function are given as:

$$\frac{\omega(s)}{I_m(s)} = \frac{K_t}{J_{eq}s + B_{eq}} \tag{5.2}$$

$$\frac{\omega(s)}{V_{in}(s)} = \frac{K_a K_t}{J_{eq}s + B_{eq}} \tag{5.3}$$

Since the range of the voltage input and the desired torque from the motor can be adjusted within the settings of the 611U, essentially, $K_a K_t$ is an adjustable value.

5.3 Controller Designs

This section discusses the different control algorithms that were implemented on the two-axis X-Y table and notes some of the expected observations and results. The open loop transfer functions are not identical for the X and Y axis due to their physical differences such as the X axis table being heavier than the Y axis table (Y on top of X). The goals used for the controller designs are:

- To maximize bandwidth/response speed
- To ensure stability with at least 60 deg of phase margin
- To eliminates steady-state error
- To minimizes overshoot

The general procedure of controller design is to model the transfer function of the machine plant, $G(s)$, in the s-domain and apply the zero-order hold (ZOH) equivalent to the plant. After which, the controllers will be designed in the discrete Z-domain, $C(z)$. Including the amplifier, motor constants, the mechanical plant and the encoder gain, the open loop transfer function of the system from input voltage to position in the s domain is:

$$G_{ol} = \frac{X(s)}{V_{in}(s)} = \frac{K_a K_t K_e}{s} \cdot \frac{1}{J_{eq}s + B_{eq}} = \frac{K}{s(s + \frac{1}{\tau_v})} \tag{5.4}$$

where $K = {K_a K_t K_e}/{J_e}$ and $\tau_v = {J_e}/{B_e}$. Since this system has to be controlled by a digital discrete control system with a sampling time of T_s, to compute for the equivalent system in the discrete Z-domain, the zero order hold is applied:

$$\left.\begin{aligned}
G_{ZOH,ol} &= ZOH[G_{ol}(s)] = (1 - z^{-1})Z\left[\frac{G_{ol}(s)}{s}\right] = (1 - z^{-1})\, Z\left[\frac{K}{s^2(s + \frac{1}{\tau_v})}\right] \\
G_{ZOH,ol} &= (1 - z^{-1}) \cdot K \cdot \tau_v \cdot Z\left[\frac{1}{s^2} + \frac{-\tau_v}{s} + \frac{\tau_v}{s + {1}/{\tau_v}}\right] \\
G_{ZOH,ol} &= (1 - z^{-1}) \cdot K\tau_v \left[\frac{T_s z}{(z-1)^2} - \frac{\tau_v z}{z - 1} + \frac{\tau_v z}{z - e^{-\frac{T_s}{\tau_v}}}\right] \\
G_{ZOH,ol}(z) &= K\tau_v \left[\frac{\left(T_s + \tau_v e^{-\frac{T_s}{\tau_v}} - \tau_v\right) \cdot z + \left(\tau_v - T_s e^{-\frac{T_s}{\tau_v}} - \tau_v e^{-\frac{T_s}{\tau_v}}\right)}{z^2 - \left(1 + e^{-\frac{Ts}{\tau_v}}\right) \cdot z + e^{-\frac{Ts}{\tau_v}}}\right] \\
G_{ZOH,ol}(z) &= K\tau_v \left[\frac{\left(T_s + \tau_v e^{-\frac{T_s}{\tau_v}} - \tau_v\right) \cdot z + \left(\tau_v - T_s e^{-\frac{T_s}{\tau_v}} - \tau_v e^{-\frac{T_s}{\tau_v}}\right)}{z^2 - \left(1 + e^{-\frac{T_s}{\tau_v}}\right) \cdot z + e^{-\frac{T_s}{\tau_v}}}\right]
\end{aligned}\right\} \tag{5.5}$$

The Euler approximation is used where $s \cong \frac{z-1}{Ts \cdot z}$. The Bode plot of the system transfer function in the continuous s-domain and discrete Z-domain can be plotted and compared. The expected

difference is that in the Z-domain, the dynamics of the zero order hold is included which provides additional phase from the time delay. The systems will end up with different phase and gain margins. However, this difference is minimized with a smaller sampling time.

5.3.1 PID Controller

The PID (Proportional Integral Derivative) controller is one of the simplest and most robust controllers and it is widely used in industry. With a proportional controller gain, K_p, the closed loop transfer function is the following:

$$\left.\begin{aligned} G_{cl} &= \frac{K_p G_{ol}(z)}{1 + K_p G_{ol}(z)} \\ G_{cl} &= \frac{K_p K \tau_v \left[\left(T_s + \tau_v e^{-\frac{T_s}{\tau_v}} - \tau_v\right)\cdot z + \left(\tau_v - T e^{-\frac{T_s}{\tau_v}} - \tau_v e^{-\frac{T_s}{\tau_v}}\right)\right]}{z^2 - \left(1 + e^{-\frac{T_s}{\tau_v}}\right)\cdot z + e^{-\frac{T_s}{\tau_v}} + K_p K \tau_v \left[\left(T_s + \tau_v e^{-\frac{T_s}{\tau_v}} - \tau_v\right)\cdot z + \left(\tau_v - T_s e^{-\frac{T_s}{\tau_v}} - \tau_v e^{-\frac{T_s}{\tau_v}}\right)\right]} \end{aligned}\right\} \quad (5.6)$$

Given a desired damping ratio and natural frequency, there would be corresponding poles on the characteristic equation of this transfer function which will determine the K_p value to use. These desired poles have to be designed such that their magnitude is less than one. In the s-domain, K_p can be chosen to shift the magnitude plot of the loop transfer function up to cross the 0 dB line at the desired frequency – desired crossover frequency, ω_c.

The PID controller has a transfer function with the following form:

$$\text{PID Controller: } C(s) = K_p + \frac{K_I}{s} + K_d s \quad (5.7)$$

The general rule in tuning these three gain parameters is the following:

- Add K_p to increase bandwidth → decrease time constant → improve rise time
- Add K_d to decrease tracking error and improve overshoot
- Add K_I to remove steady state error

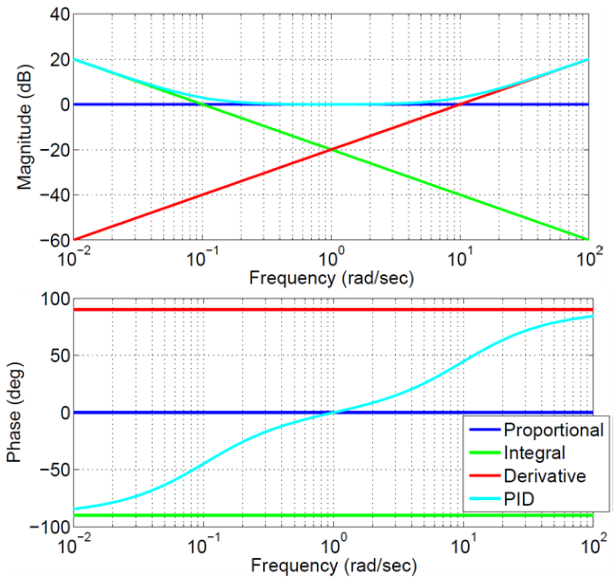


Figure 5.4: PID Controller Bode Plot

$$K_p = 10, K_I = 0.1, K_d = 0.1$$

Since the whole PID controller is the sum of the proportional (flat line), integral (negative slope), and the derivative (positive slope), in the magnitude plot of the controller, the largest of the three dominates at a given frequency, as shown in Figure 5.4 above. Consider a general control system shown in Figure 5.5 below, the error transfer function is given by:

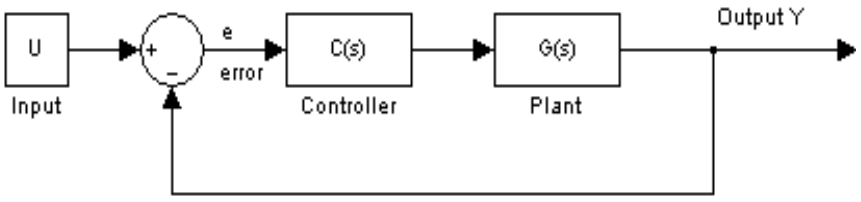


Figure 5.5: General Close Loop Control System

$$\frac{e(s)}{U(s)} = \frac{1}{1 + C(s)G(s)} \tag{5.8}$$

The error transfer function suggests that a large loop gain will minimize the tracking error. Moreover, as suggested by the Initial Value Theorem:

$$\lim_{t \to 0} f(t) = \lim_{s \to +\infty} sF(s) \tag{5.9}$$

and the Final Value Theorem:

$$\lim_{t\to+\infty} f(t) = \lim_{s\to+0} sF(s) \tag{5.10}$$

The behavior of the frequency response is describing the inverse behavior of the time domain. Since the integral portion of the controller keeps the magnitude high at low frequencies (large time), its effect is apparent with steady state error and since the derivative portion keeps the magnitude high at high frequencies, it improves tracking error and overshoot. Given some desired performance, these three parameters can be tuned to satisfy the design requirements.

5.3.2 Lead Lag Controller

The lead-lag controller is a lead compensator cascaded with a lag compensator to introduce a pole-zero pair into the open loop transfer function. Lead compensation, shown in Figure 5.6, is a conventional frequency-domain design approach, which uses the derivative to improve stability by directly increasing the phase margin. Lead compensation is used to compensate for the phase delay of the plant, allowing for a higher system bandwidth and faster response speed. The system bandwidth can be increased further by increasing the DC gain of the control loop, which also improves the steady state tracking.

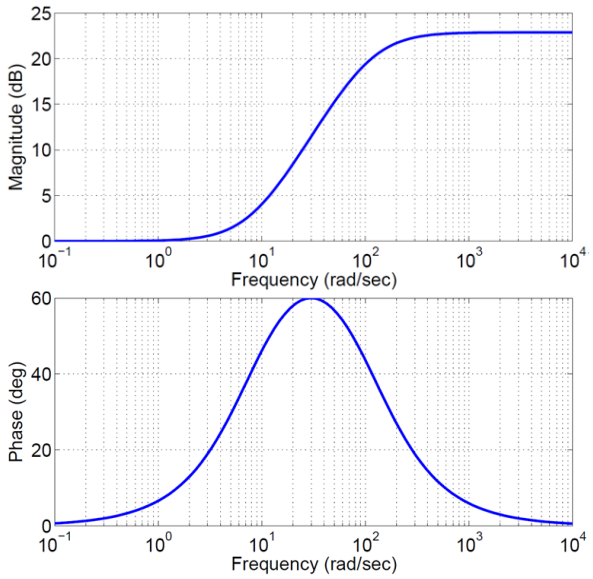


Figure 5.6: Lead Compensator Bode Plot

Lag compensator, shown in Figure 5.7, can be used to simultaneously improve both the steady-state accuracy and the stability of the system. Since the lag compensator adds a phase lag to the system, it actually has a destabilizing effect. But since the crossing frequency is reduced by the lag compensator, the phase lag will be lower in the neighborhood of the new crossing

frequency, thereby increasing the phase margin and improving system stability. The lag compensator inherently improves the low-frequency behavior, steady-state accuracy in particular, of the system. Furthermore, since a lag compensator is essentially a low-pass filter, it has the added advantage of filtering out high-frequency noise.

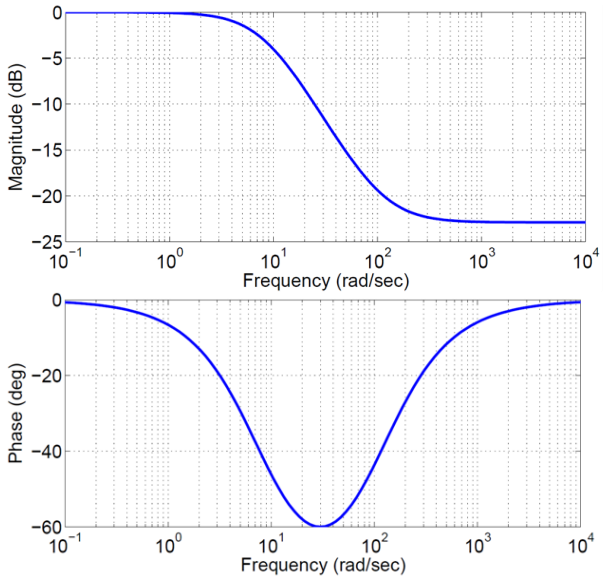


Figure 5.7: Lag Compensator Bode Plot

An integrator can also be added to eliminate steady state error. In Laplace domain, the Lead Lag compensator with integrator transfer function is written as:

$$\text{Lead Lag Controller: } C(s) = K_c \frac{1 + \alpha T_c s}{1 + T_c s} \cdot \frac{K_i + s}{s} \tag{5.11}$$

The design goals for the lead lag controller are the desired cross over frequency, ω_c, and the phase margin, PM. Phase margin is defined as the difference between the phase at the cross over frequency and -180 degrees. The integrator can be lumped with the plant using the calculation where a common value of the integrator constant K_i is picked to be $\omega_c/10$. Then the phase contribution by the lead lag controller is given as:

$$\phi = [tan^{-1}(\alpha T_c \omega) - tan^{-1}(T_c \omega)] \tag{5.12}$$

where

$$\left.\begin{aligned} \alpha = \frac{1+\sin\phi}{1-\sin\phi};\ \begin{cases}\alpha > 1 \text{ for phase lead or high pass filter}\\ \alpha < 1 \text{ for phase lag or low pass filter}\end{cases}\\ T_c = \frac{1}{\omega_c\sqrt{\alpha}} \end{aligned}\right\} \tag{5.13}$$

In order for the resulting phase at ω_c to be PM, the total phase of the loop transfer function at the cross over frequency is expressed, where ϕ can be solved:

$$\phi + \measuredangle G(j\omega_c) = \frac{(-180^\circ + PM)\pi}{180^\circ} \tag{5.14}$$

The gain, K_c, is added to ensure the loop transfer function passes through the desired cross over frequency at $0\ dB$ or a gain of one.

$$K_c = \frac{1}{\left|\frac{1+\alpha T_c \cdot j\omega_c}{1+T_c s}\right| \cdot |G(j\omega_c)|} \tag{5.15}$$

Using the Euler approximation, the lead lag controller can be written in the discrete Z-domain as:

$$C(z) = K_c \frac{z(T_s + \alpha T_c) - \alpha T_c}{z(T_s + T_c) - T_c} \cdot \frac{z(K_i T_s + 1) - 1}{z - 1} \tag{5.16}$$

5.3.3 Loop Shaping Controller

Loop shaping is a control method to improve the system performance and maintain the system stability by tuning the controller to shape the negative loop transmission (NLT) to achieve the desired characteristics. Depending on the frequency response function of the plant, it is common to shape the $\text{NLT} = C(s)G(s)$ to cross over $0\ dB$ at the desired crossover frequency, ω_c. Since the closed loop transfer function is given as $G_{CL} = \frac{NLT}{1+NLT}$, this would place the $-3\ dB$ drop to occur near ω_c. A design requirement is to ensure that the NLT has a positive phase margin at ω_c and it is desirable to reduce the NLT gain for $\omega > \omega_c$ and increase NLT gain for$\omega < \omega_c$. At high frequencies, $\omega > \omega_c$, it is desirable to minimize the control output and not amplify the noise. At low frequencies, $\omega < \omega_c$, a higher NLT gain can improve the tracking

performance. The loop shaping controller is very similar to the lead-lag controller and is shown as follows:

$$\text{Loop Shaping Controller: } C(s) = K_c \cdot \frac{\frac{s}{\omega_c/\alpha} + 1}{\frac{s}{\alpha \cdot \omega_c} + 1} \cdot (1 + \frac{K_i}{s}) \tag{5.17}$$

In the Z-domain, this is given as:

$$C(z) = K_c \cdot \frac{z(\alpha + T_s\omega_c) - \alpha}{z\left(\frac{1}{\alpha} + T_s\omega_c\right) - \frac{1}{\alpha}} \cdot \frac{z(K_i T_s + 1) - 1}{z - 1} \tag{5.18}$$

For a second order system, $G(s) = \frac{1}{s^2}$, a proportional controller will always have a zero phase margin with a -2 slope crossing ω_c on the magnitude plot. An integrator controller is worse with a -3 slope (-270 degrees phase). The idea of the loop shaping controller is to shape the NLT to cross $0\ dB$ with a higher phase than 180 degrees to have a positive phase margin. In the case of a 2nd order plant, the controller needs to provide a positive slope at ω_c to increase the phase at ω_c. The loop shaping controller is shown in Figure 5.8. A positive slope is only necessary around ω_c.

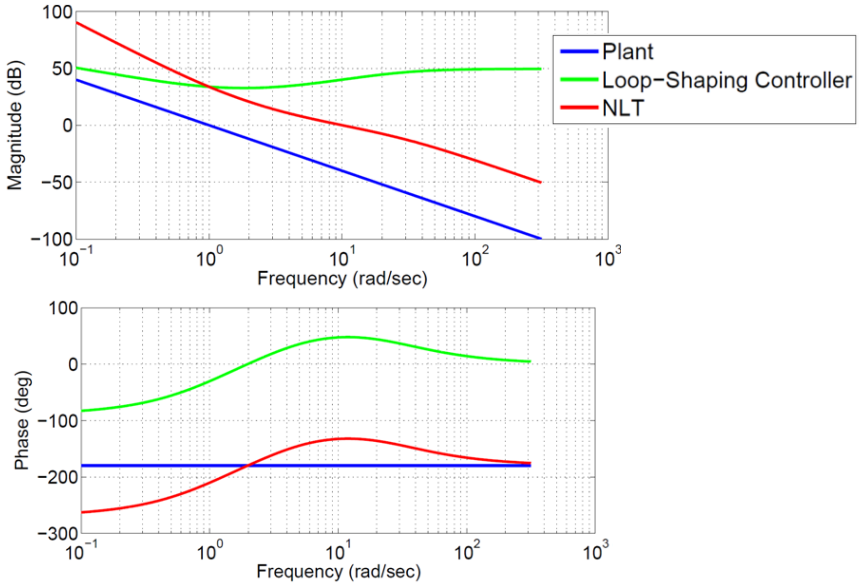


Figure 5.8: Loop Shaping Controller Bode Plot

The Bode plot of the loop shaping (green) is shown with $\omega_c = 10\ rad/sec$ and $\alpha = 3$. A low α value should be chosen to satisfy the phase margin requirement as it would provide a higher

NLT gain for $\omega < \omega_c$ and improve the tracking performance. To remove the steady state error, an integrator is introduced to increase the NLT gain at low frequencies.

5.3.4 Pole Placement Controller

The pole placement controller block diagram is shown in Figure 5.9 and its control strategy [4] is to assign the desired poles to the closed loop transfer function of the system to define the system characteristic.

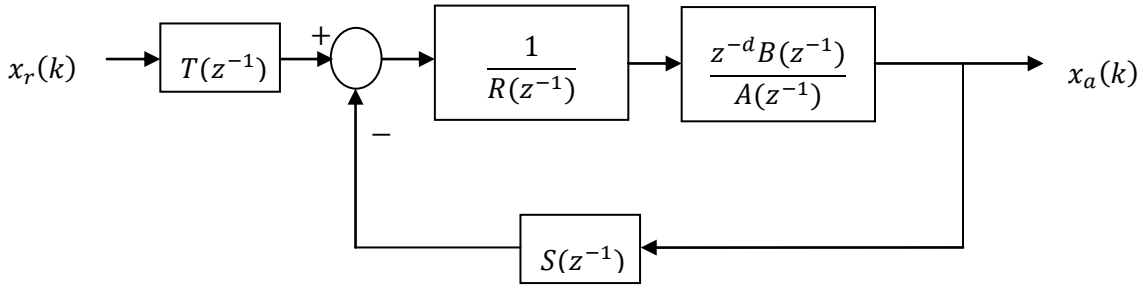


Figure 5.9: Pole Placement Controller Block Diagram

The plant's transfer function is written in the form of $\frac{z^{-d}B(z^{-1})}{A(z^{-1})}$ and it has a delay of d sampling intervals with the following open loop polynomials:

$$\left.\begin{aligned} B(z^{-1}) &= b_1 z^{-1} + b_2 z^{-2} + \ \dots + b_n z^{-n} \\ A(z^{-1}) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \ \dots + a_m z^{-m} \end{aligned}\right\} \tag{5.19}$$

The polynomials $S(z^{-1})$, $T(z^{-1})$ and $R(z^{-1})$ represent the feedback, feed-forward and error regulators respectively, and form the control law.

$$\left.\begin{aligned} R(z^{-1}) &= 1 + r_1 z^{-1} + r_2 z^{-2} + \ \dots + r_p z^{-p} \\ S(z^{-1}) &= s_0 + s_1 z^{-1} + s_2 z^{-2} + \ \dots + s_f z^{-f} \\ T(z^{-1}) &= t_0 \end{aligned}\right\} \tag{5.20}$$

The closed loop transfer function of the control system is given by:

$$\frac{x_a(k)}{x_r(k)} = \frac{T(z^{-1})z^{-d}B(z^{-1})}{A(z^{-1})R(z^{-1}) + z^{-d}B(z^{-1})S(z^{-1})} \tag{5.21}$$

If the plant has a stable zero inside the unit circle, B can be partitioned into two parts: $B = B^+B^-$ where B^+ has the stable and B^- has the unstable zeros. Stable zeros (B^+) can be

cancelled by selecting the control polynomial $R = R'B^+$, which leads to the following transfer function:

$$R(z^{-1}) = R'(z^{-1})B^+(z^{-1}) \tag{5.22}$$

$$\left.\begin{aligned} \frac{x_a(k)}{x_r(k)} &= \frac{T(z^{-1})z^{-d}B^+(z^{-1})B^-(z^{-1})}{A(z^{-1})B^+(z^{-1})R'(z^{-1}) + z^{-d}B^+(z^{-1})B^-(z^{-1})S(z^{-1})} \\ \frac{x_a(k)}{x_r(k)} &= \frac{T(z^{-1})z^{-d}B^-(z^{-1})}{A(z^{-1})R'(z^{-1}) + z^{-d}B^-(z^{-1})S(z^{-1})} \end{aligned}\right\} \tag{5.23}$$

where the stable zeros are cancelled out, leaving the unstable zeros. The goal of the pole placement design is for the closed loop transfer function to obey the following desired model dynamics:

$$\frac{x_a(k)}{x_r(k)} = \frac{z^{-d}B_m(z^{-1})}{A_m(z^{-1})} \tag{5.24}$$

where $A_m(z^{-1})$ is the desired characteristic equation of the closed loop system which is designed to the desired transient response characteristics. A second order dynamic model represents the desired response with a damping ratio of ξ_m and natural frequency of ω_m,

$$\left.\begin{aligned} A_m(s) &= s^2 + 2\xi_m\omega_m s + {\omega_m}^2 = (s - s_{m1})(s - s_{m2}) \\ s_{m1} &= -\xi_m\omega_m + j\omega_m\sqrt{1 - {\xi_m}^2} \\ s_{m2} &= -\xi_m\omega_m - j\omega_m\sqrt{1 - {\xi_m}^2} \end{aligned}\right\} \tag{5.25}$$

The desired poles can be mapped to the Z-domain by,

$$\left.\begin{aligned} z_{m1} &= e^{s_{m1}T_s} = e^{-\xi_m\omega_m T_s} \cdot e^{j\omega_m\sqrt{1-{\xi_m}^2}T_s} \\ z_{m2} &= e^{s_{m2}T_s} = e^{-\xi_m\omega_m T_s} \cdot e^{-j\omega_m\sqrt{1-{\xi_m}^2}T_s} \\ A_m(z) &= (z - z_{m1})(z - z_{m2}) \\ &= z^2 - 2e^{-\xi_m\omega_m T_s}(e^{j\omega_m\sqrt{1-{\xi_m}^2}T_s} + e^{-j\omega_m\sqrt{1-{\xi_m}^2}T_s}) + e^{-2\xi_m\omega_m T_s} \end{aligned}\right\} \tag{5.26}$$

A damping ratio of $\xi_m \geq 0.707$ is desirable. The acceleration is determined by the maximum acceleration that the motors provide without going over the torque limit. The peak torque is T_d for accelerating the inertia J_e, then the maximum acceleration becomes:

$$\ddot{\theta}_{max} = \frac{T_d}{J_e} \tag{5.27}$$

If the drive maximum velocity is $\dot{\theta}_{max}$, $t_r = \frac{\dot{\theta}_{max}}{\ddot{\theta}_{max}}$ is the rise time. The natural frequency ω_m [rad/sec] is approximated with the following equation:

$$\omega_m \cong \frac{1 - 0.4167\xi + 2.917\xi^2}{t_r} \rightarrow \xi < 1 \tag{5.28}$$

With this approximation, the discrete closed loop transfer function with a unit gain is:

$$\frac{x_a(k)}{x_r(k)} = \frac{z^{-d}B_m(z^{-1})}{A_m(z^{-1})} = \frac{z^{-d}B_m(z^{-1})}{1 + m_1 z^{-1} + m_2 z^{-2}} \tag{5.29}$$

The order of the desired characteristic equation (A_m) must be equal to the order of plant's poles ($A(z^{-1})$). The following equation must hold in order for the system to behave like the desired model.

$$\frac{z^{-d}BT}{AR + z^{-d}BS} = \frac{z^{-d}B_m(z^{-1})}{A_m(z^{-1})} \tag{5.30}$$

The highest powers of z^{-1} are $z^{-(p+m)}$ and $z^{-(d+n+f)}$, and their powers are equal to obtain $p + m = d + n + f$ number of equations. The number of unknowns in control polynomials in $R(z^{-1})$ and $S(z^{-1})$ is $p + f + 1$ which must be equal to the number of simultaneous equations. The degree of control polynomials are found as:

$$\left.\begin{array}{c} d + n + f = p + f + 1 \rightarrow \deg(R) = p = d + n - 1 = d + \deg(B) - 1 \\ m + p = p + f + 1 \rightarrow \deg(S) = f = m - 1 = \deg(A) - 1 \end{array}\right\} \tag{5.31}$$

The numerators must have a balanced power as well,

$$\left.\begin{array}{c} z^{-d}B(z^{-1})T = z^{-d}B_m(z^{-1}) \\ \left[\frac{z^{-d}B(z^{-1})b_m}{A_m(z^{-1})}\right]_{z=1} = 1 \rightarrow b_m = \left[\frac{A_m(z^{-1})}{z^{-d}B(z^{-1})}\right]_{z=1} \\ T = t_0 = b_m \end{array}\right\} \tag{5.32}$$

By separating the unknowns in a matrix form, the identification of control parameters are reduced to a least square problem.

5.3.5 Sliding Mode Controller

Sliding mode control [3] is a nonlinear control technique with high robustness against hysteresis, nonlinearity, and disturbance while having a high frequency bandwidth. It makes use of a sliding surface that is being tracked, which in turn dictates how the positions are being tracked. An energy function is formed to slide the system onto the surface, with considerations to the disturbance.

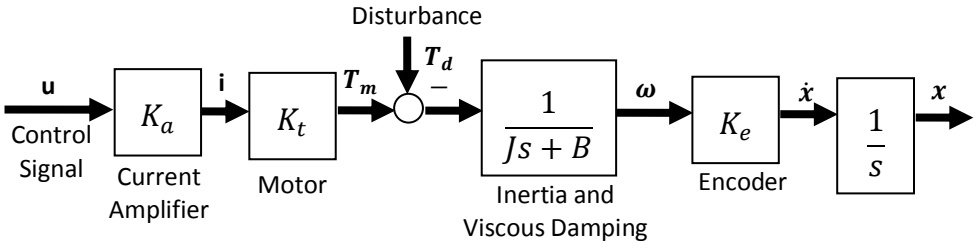


Figure 5.10: Open Loop Linear Rigid Body Feed Drive Dynamics Model

The system equation for the feed drive model shown in Figure 5.10 can be written as:

$$\left.\begin{aligned} &x(s) = \frac{K_a\, K_t\, K_e}{s(Js+B)}\left[u(s) - \frac{1}{K_a\, K_t} T_d(s)\right] \\ &\frac{J}{K_a\, K_t\, K_e}\ddot{x}(t) + \frac{B}{K_a\, K_t\, K_e}\dot{x}(t) = u(t) - \frac{1}{K_a\, K_t} T_d(t) \\ &J_e\, \ddot{x}(t) + B_e\, \dot{x}(t) = u(t) - d(t) \end{aligned}\right\} \tag{5.33}$$

where $d(t) = \frac{1}{K_a\, K_t} T_d(t)$, x is the output position, u is the control input, and d is the system disturbance. The fundamental steps in designing a sliding mode controller are the selections of a sliding surface and a Lyapunov function. For accurate tracking of position and velocity, the sliding surface, S is selected as:

$$S = \lambda(x_r - x) + (\dot{x}_r - \dot{x}) \tag{5.34}$$

where λ $[1/\text{sec}]$ is the desired but achievable bandwidth of the drive, x_r is the reference position command, and $\dot{x}_r$ and $\dot{x}$ are the respective velocities. The d^+ and d^- are upper and lower disturbance limits measured on the machine. The external disturbance can be tracked by the following simple observer,

$$\left.\begin{aligned} &\dot{\hat{d}} = \rho\kappa S \\ &\hat{d}(k) = \hat{d}(k-1) + \rho\kappa S \cdot T_s \end{aligned}\right\} \tag{5.35}$$

where T_s is the control period, k is the control interval counter in discrete time domain, ρ is parameter adaptation gain ($\rho \approx 0.005$) and κ is the coefficient for the integral control of the disturbance as,

$$\kappa = \begin{cases} 0 & \text{if} \quad \hat{d}(k) \le d^{-} \text{and } S \le 0 \\ 0 & \text{if} \quad \hat{d}(k) \le d^{+} \text{and } S \ge 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.36}$$

The second step in sliding mode controller design is to select a Lyapunov function which is used to obtain a stable control law for a nonlinear system. The Lyapunov function is as follows:

$$V(t) = \frac{1}{2}\left[J_e\, S^2 + \frac{(d-\hat{d})^2}{\rho} \right] \tag{5.37}$$

which resembles the summation of kinetic energy and the square of disturbance prediction error. For asymptotic stability of nonlinear systems, the derivative of the Lyapunov function must be negative or zero, meaning that the rate of change in the energy and prediction error must decrease.

$$\left.\begin{aligned} &\dot{V}(t) = J_e\, S\, \dot{S} - \dot{\hat{d}}\,\frac{(d-\hat{d})}{\rho} \le 0 \\ &\dot{V}(t) = J_e\, S\, [\lambda(\dot{x}_r - \dot{x}) + \ddot{x}_r] + SB_e\dot{x} - Su + Sd - S\kappa(d-\hat{d}) \le 0 \\ &\dot{V}(t) = J_e\, S\, [\lambda(\dot{x}_r - \dot{x}) + \ddot{x}_r] + SB_e\dot{x} - Su + S\hat{d} = -K_s S^2 \end{aligned}\right\} \tag{5.38}$$

where $K_s > 0$ is the control gain to be selected. Therefore, the control law u is obtained as the following:

$$\left.\begin{aligned} &u(k) = J_e\left[\lambda\left(\dot{x}_r(k) - \dot{x(k)}\right) + \ddot{x}_r(k)\right] + B_e\dot{x}(k) + \hat{d}(k) + K_s S(k) \\ &S(k) = \lambda(x_r(k) - x(k)) + (\dot{x}_r(k) - \dot{x(k)}) \end{aligned}\right\} \tag{5.39}$$

5.4 Implementation Results

With the desired reference X and Y positions given from trajectory generation, an important measure of performance is the contour error. While the axis tracking error applies only for the particular motor axis, the contour error requires a high tracking performance from both axes. The frequency response of the open loop plant system was taken and it provides an accurate representation to design the controllers. The results of the contouring experiment with the implemented controllers are presented. The feedforward controller was also used in

conjunction with the presented controllers using the experimentally determined plant system dynamics. It helps to reduce the control effort that is required by the test controller.

5.4.1 Frequency Response Function

The experimental open loop frequency response is taken to assess the performance of the actual system. It allows the controllers to be designed more accurately based on the actual behavior of the system. For the FRF experiment, different frequencies (sinusoidal) of voltage commands were applied with constant voltage offsets to the open loop of each drive axis. The offset voltage ensures that the drive is constantly moving in one direction to avoid introducing unnecessary static friction when the drive changes directions. For each frequency, the magnitude ratio of the angular velocity and the input voltage, and their phase differences are measured. The FRF is given as the output position in millimeters over the input voltage. The results are plotted in Figure 5.11 and 5.12 as the measured frequency response function for each drive.

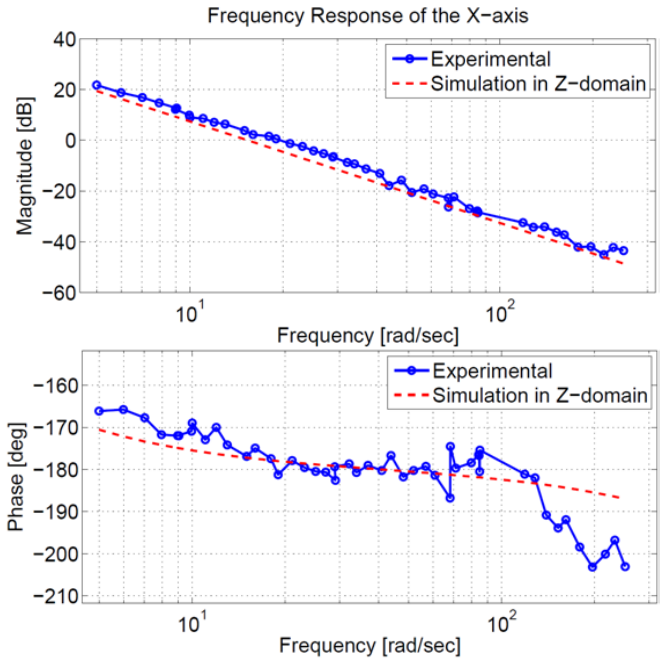


Figure 5.11: X-Axis FRF Experiment

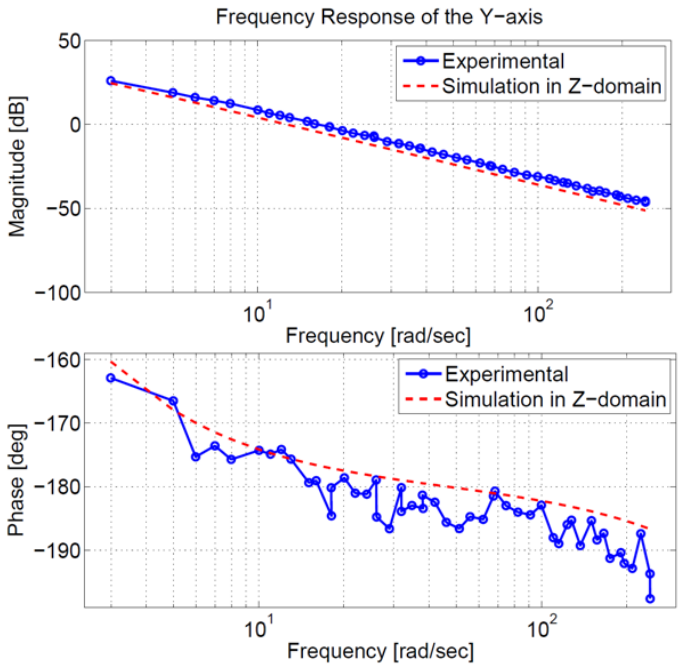


Figure 5.12: Y-Axis FRF Experiment

The experimental FRF are plotted against the theoretical model that was obtained from the 2nd order feed drive dynamics Eqn. (5.4) using the experimentally determined inertia equivalent, J_{eq}, and the experimentally determined viscous damping equivalent, B_{eq}. These values were identified using open-loop experimentations and are given in the following Table 5.1:

	$J_{eq}\ [kgm^2]$	$B_{eq}\ \left[\frac{Nm}{rad/sec}\right]$
X-axis	0.030	0.0253
Y-axis	0.022	0.0237

Table 5.1: Experimentally Determined Equivalent Inertia and Damping

The experimental results are very similar to the theoretical results. It is important to pick a feasible cross over frequency to design the controllers. Since the actual system is a high order system with a lot of coupled modes, it is desirable to select a cut off frequency that is relatively consistent in terms of both magnitude and phase. A basic requirement for stability is for the magnitude plot of the loop function to only cross unity gain at one frequency. In addition the limits of the controllers are also considered. For instance, it may not be possible for the controllers to have really high gains due to power limitations and there are also limits to the

amount of phase compensation that can be provided by the controller. The desired cross over frequencies of 20 [rad/sec] for X axis and 15 [rad/sec] for Y axis are selected as the design targets of the controllers.

5.4.2 Basic Controller Contouring Evaluations

A sample NURBS tool path trajectory is used to test the maneuverability of the large-sized X-Y table. The sample toolpath was interpolated with the 1st order Taylor's expansion; the contouring performance of the loop shaping, pole placement and the sliding mode controllers were evaluated. For simplicity, a trapezoidal feedrate profile was used. The axis commands were generated with a sampling time of 1 [ms]. The total length of the tool path is 318.329 [mm] and the displacement and feedrate profile used are shown as Figure 5.13:

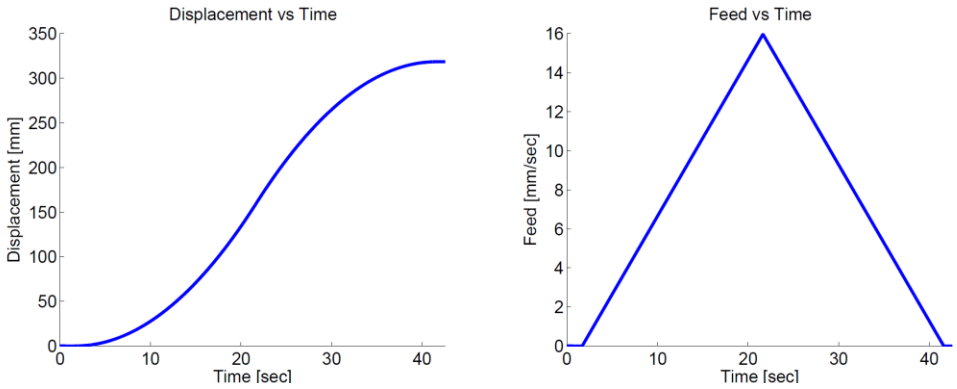


Figure 5.13: Controller Evaluation: Displacement and Feedrate Profile

The loop shaping controller was tested with the parameters listed in Table 5.2 – the design is based on designing the cross over frequencies of 20 [rad/sec] for X axis and 15 [rad/sec] for Y axis, with a phase compensation to increase the phase margin to 60 [deg].

Controller Parameters	X axis	Y axis
K_p	2.883	1.969
K_i	2.000	1.500
α	4	4
Cutoff frequency ω_c	20	15

Table 5.2: Loop Shaping Controller Parameters

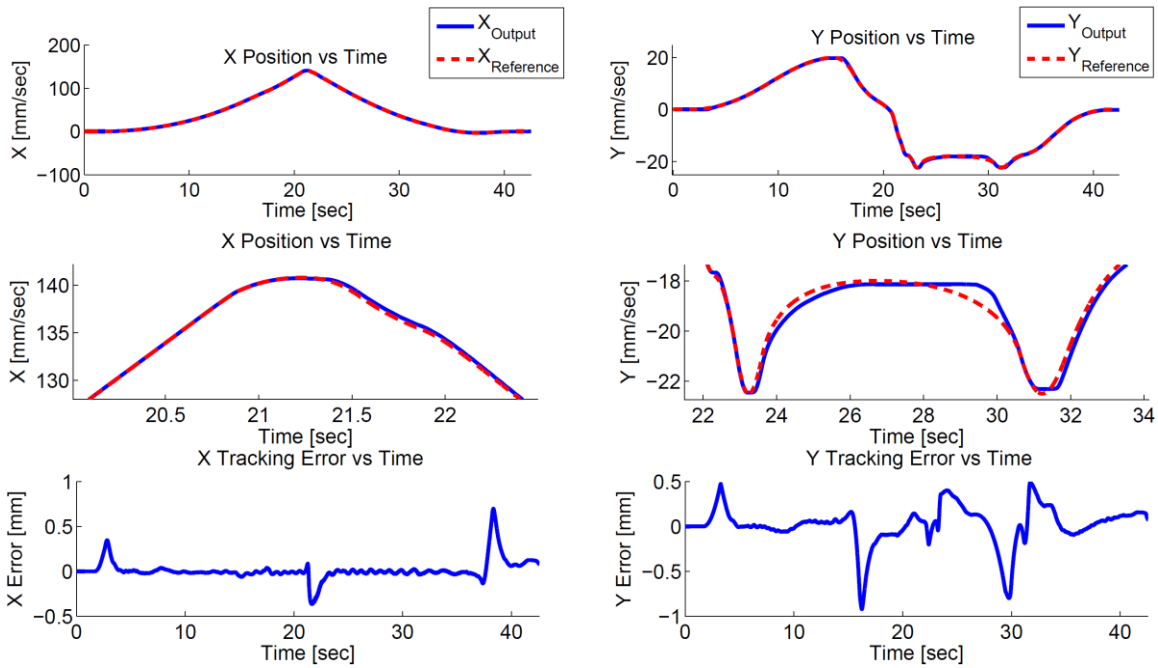


Figure 5.14: Loop Shaping Controller Position and Tracking Error

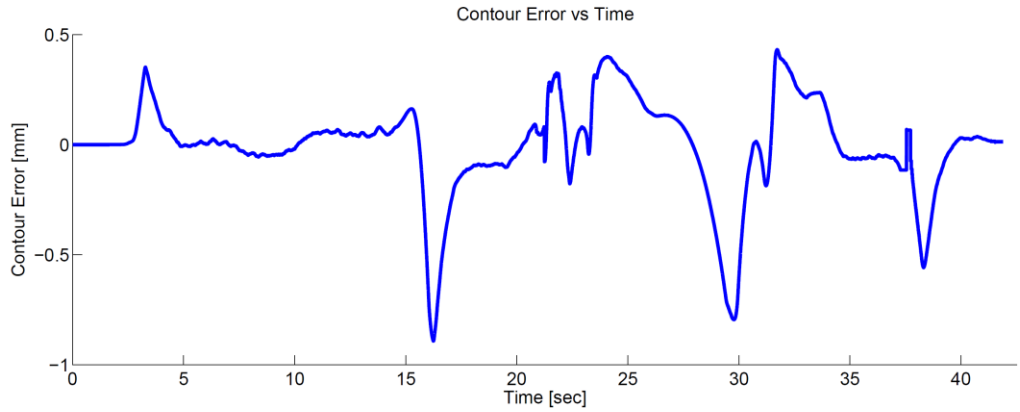


Figure 5.15: Loop Shaping Controller Contour Error

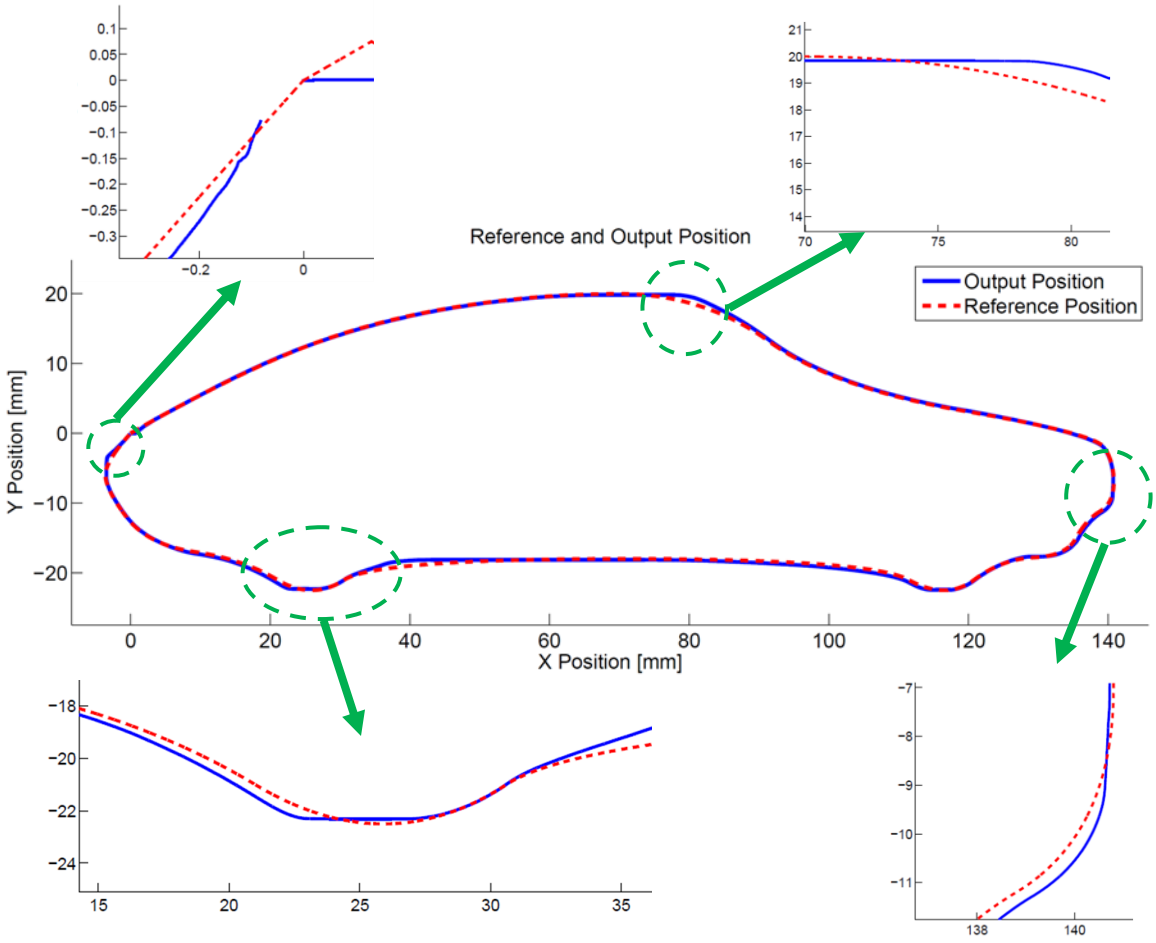


Figure 5.16: Loop Shaping Controller Contouring Performance

For the loop shaping controller, as shown in Figure 5.14, the largest tracking error for the x axis is occurring at the start and the end of the toolpath, as well as in the middle of the tool path where the direction is flipped. When the toolpath starts, there is some delay between the reference and the actual position since the motion is starting up and there is also static friction involved. As the toolpath flips in direction, there is a rapid change in the reference command, and for this large scale X-Y table, the bandwidth or the response time for the controller designs cannot be too high without losing stability. Similarly, the y axis suffers from the highest tracking error at the wheels where the sharp change in the reference is demanding a large control effort from the controller. From Figure 5.16, the overall performance of the loop shaping controller is very good as there were no observable vibrations throughout the entire toolpath and the maximum contour error resulted in 0.892 [mm] at the first wheel. It is also clear that the tracking errors are very small for the smoother regions of the toolpath.

The pole placement controller parameters are the desired natural frequency and damping ratio of the resulting closed loop response. A high natural frequency as listen in Table 5.3 was selected to achieve a high response time for the controller and a slightly underdamped system was implemented.

Controller Parameters	X axis	Y axis
Natural frequency ω_n	50	40
Damping ratio ξ	0.9	0.9

Table 5.3: Pole Placement Controller Parameters

The faster response time of the designed pole placement controller leads to a lower maximum contour error of $0.269\ [\mathrm{mm}]$ than the loop shaping controller. From the resulting contouring performance shown in Figure 5.19, the observed overshoots at the curves are very small as the system is very close to being critically damped. At the end of the toolpath, however, there is a residue error of $0.3\ [\mathrm{mm}]$, whereas the loop shaping controller only had $0.15\ [\mathrm{mm}]$ of residual error. This can be attributed to the loop shaping controller being cascaded to an integrator which serves to eliminate steady state error. However, due to the faster response of the pole placement controller, the transient contouring performance where the curve switches directions or where the toolpath has a high curvature performs better than the loop shaping controller.

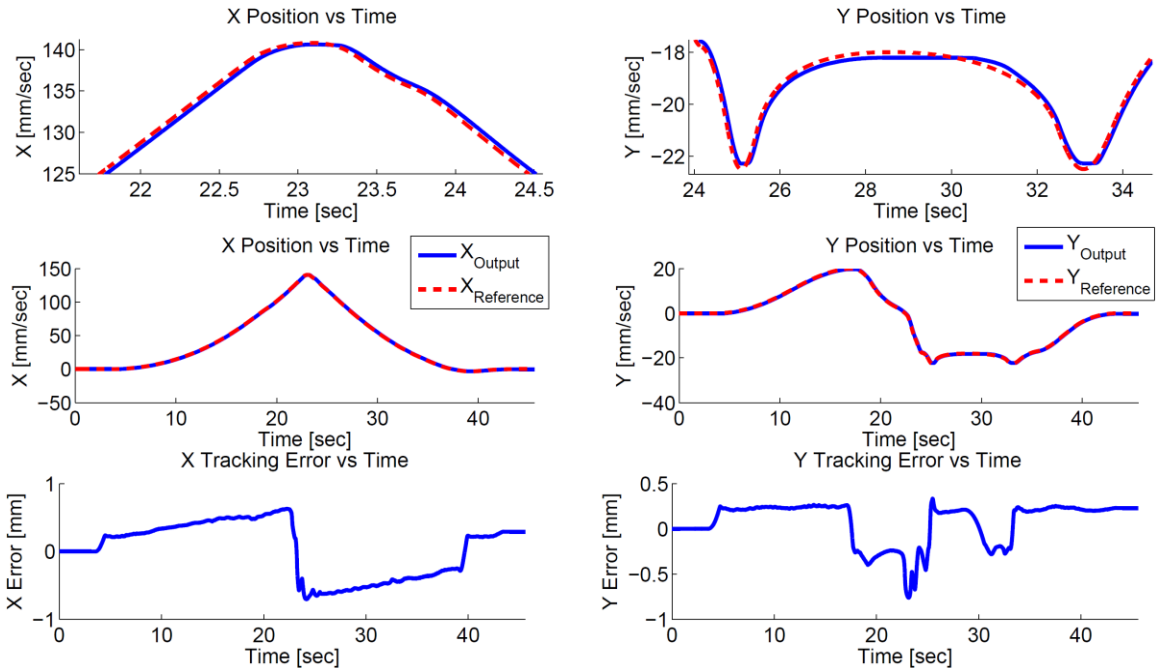


Figure 5.17: Pole Placement Controller Position and Tracking Error

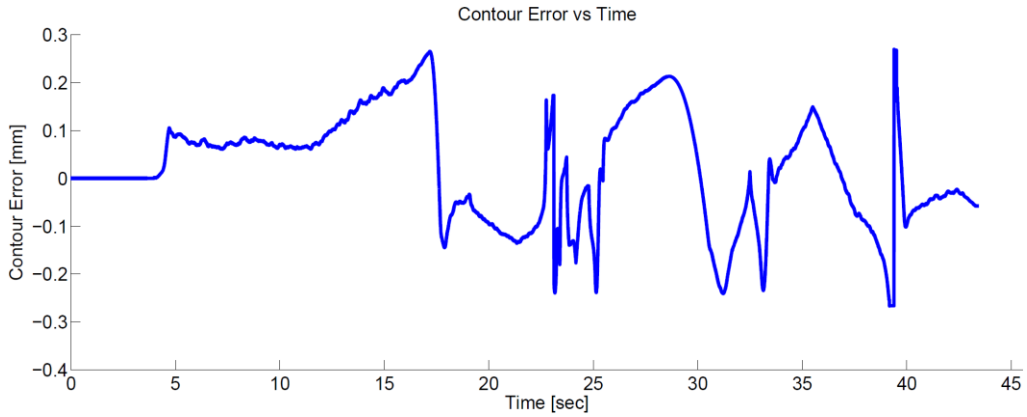


Figure 5.18: Pole Placement Controller Contour Error

For the sliding mode controller, some tuning was required for selecting the values for the adaptive gain, ρ and the sliding gain, K_s. The maximum and minimum disturbance values were obtained from the open loop friction experiment where the maximum observed static friction are taken for both directions of travel. The resulting controller parameters that were implemented are listed in Table 5.4:

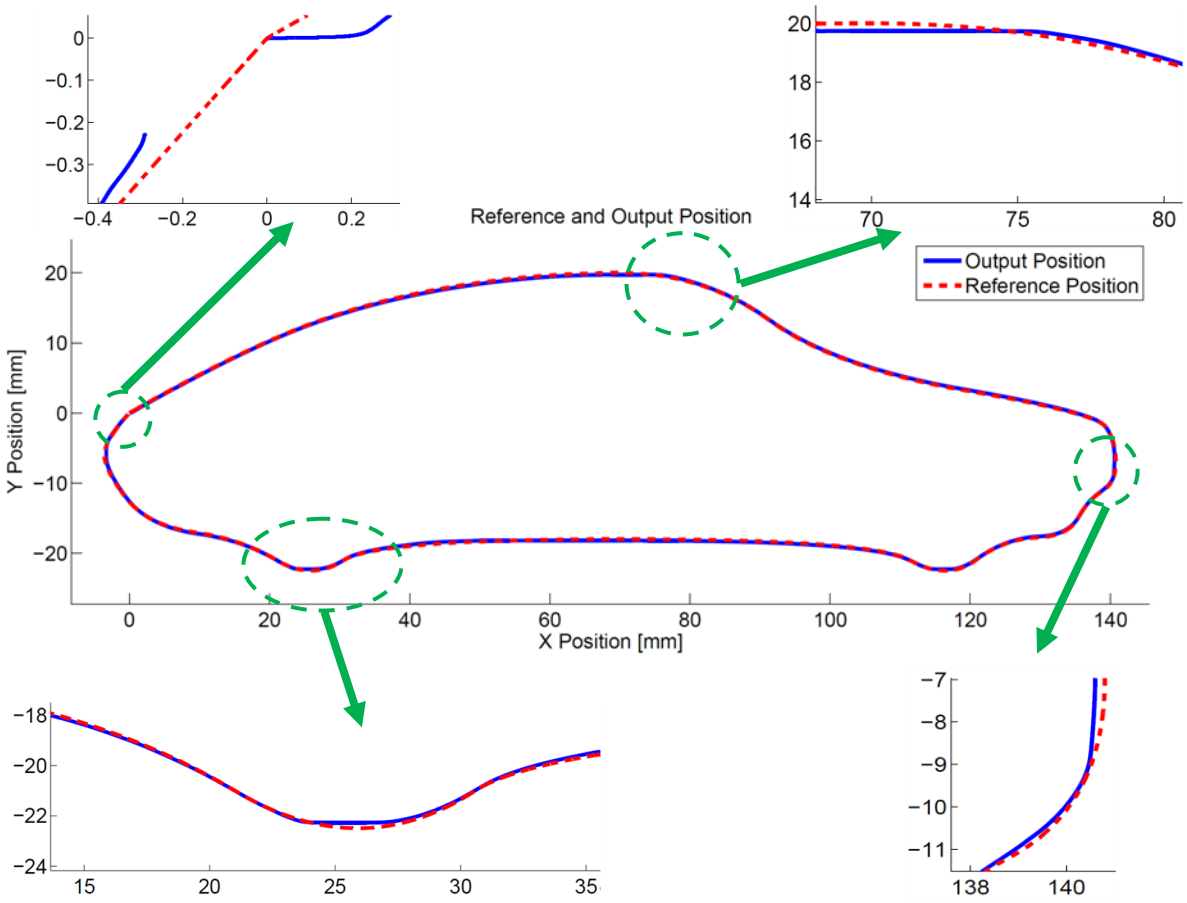


Figure 5.19: Pole Placement Controller Contouring Performance

Controller Parameters	X axis	Y axis
Bandwidth gain λ	15	13
Adaptive gain ρ	0.003	0.004
Sliding gain K_s	1	0.8
Min disturbance d^-	-8	-3.52
Max disturbance d^+	8	3.52

Table 5.4: Sliding Mode Controller Parameters

It was observed that during the X-Y table motion as shown in Figure 5.22, the sliding mode controller was very aggressive and that led to slight vibrations along the toolpath. A wavy position output can be observed from Figure 5.20. The maximum contour error was recorded as

0.334 [mm]. Further tuning may be required to improve the performance of this controller, as a less aggressive controller will reduce the observed vibrations.

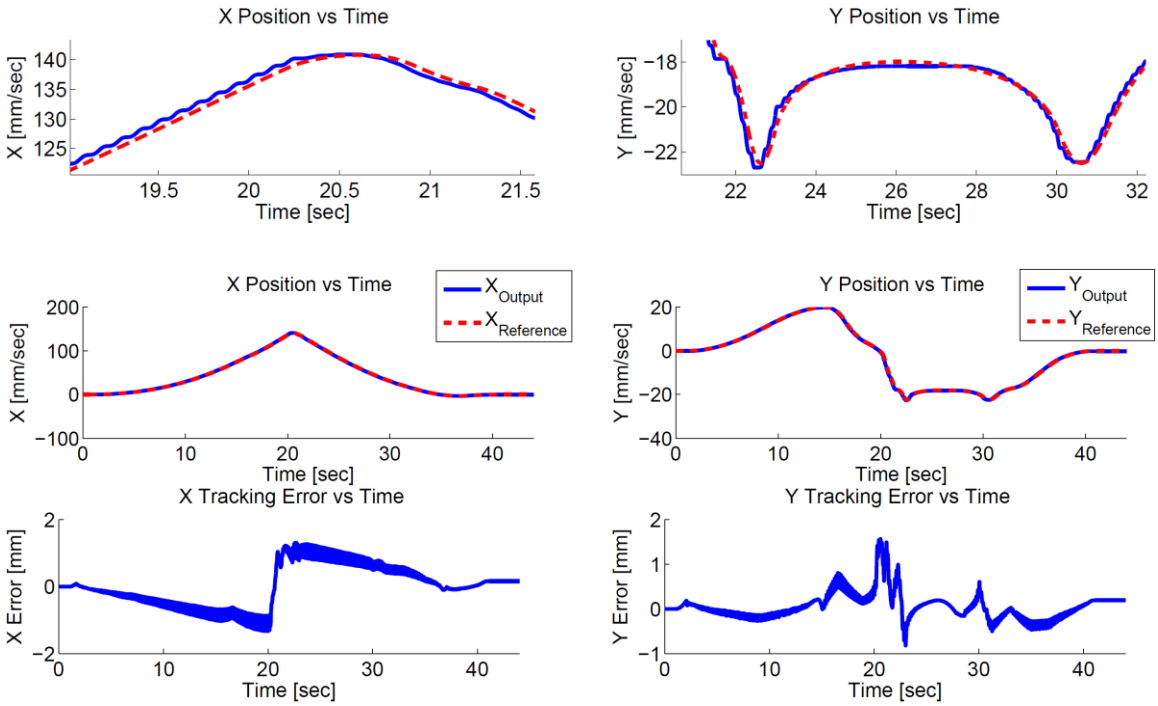


Figure 5.20: Sliding Mode Controller Position and Tracking Error

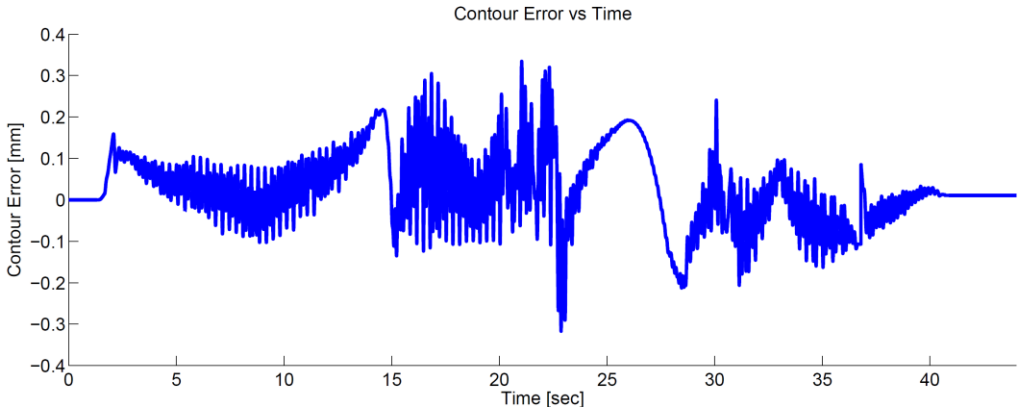


Figure 5.21: Sliding Mode Controller Contour Error

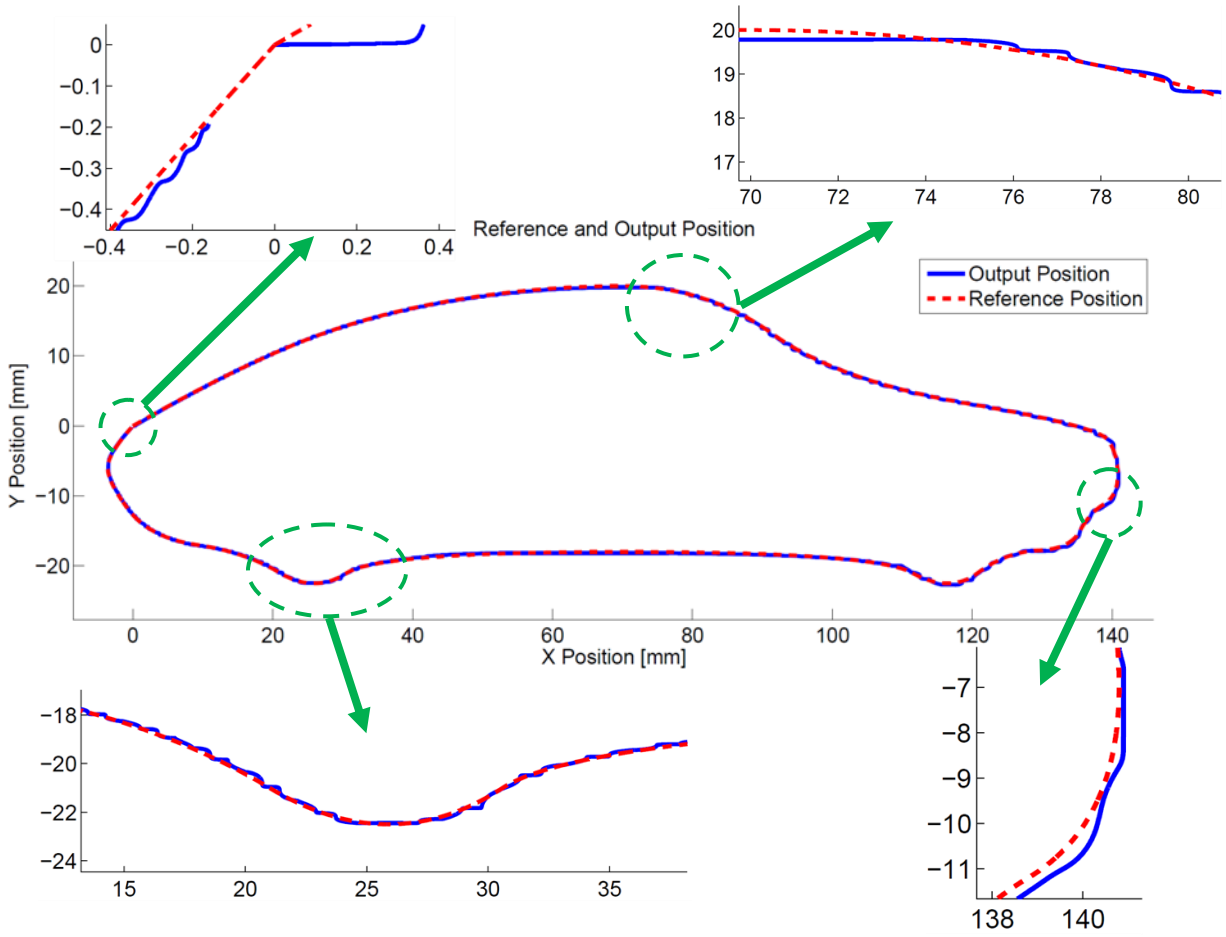


Figure 5.22: Sliding Mode Controller Contouring Performance

Overall, the pole placement controller was shown to have the lowest contour error compared to the loop shaping controller and the sliding mode controller. The loop shaping controller had the least steady state error by the end of the toolpath due to the cascading of an integrator. Further tuning of the sliding mode controller gains are required to improve its contouring performance.

Chapter 6 Conclusion

6.1 Conclusion

The continuous motion of the cutting tool is highly appreciated in decreasing machining cost and improving part quality. Complex free-form geometries are often broken down into short linear segments in NC programs, and results in a piece-wise motion of the CNC machine tool. In this thesis, a smooth NURBS toolpath trajectory generation scheme with real time interpolation techniques is presented. The proposed techniques allow for the coordinated movements of the motor drives within their kinematic limits and are applicable for both 3-axis and 5-axis high speed machining.

The NURBS representation of the toolpath was presented. A global approximation least squares fit allows a large number of tool data points to be fitted with a relatively small amount of control points. By using a pre-assigned knot vector, the control points can be solved linearly as the minimization parameter to minimize the error between the data points and the corresponding locations on the NURBS curve. However, it was shown that if the minimization consideration is solely at the data points, an undesirable wavy toolpath may be produced as the least squares algorithm forces the curve to discriminatively pass through the data points. The chord error minimization objective function was introduced to reduce the absolute influence at the data points, and strives to minimize each respective region of the curve to the respective chord formed by the data points. This minimization method scales the influence of each region through their range in the curve parameter and it was observed that the chord error results have a close resemblance to the mean error results at the data points. In addition, the jerk minimization objective function was used to minimize the jerk content of the curve with respect to the curve parameter. The effectiveness of the objective functions was proven using the sample fan shaped tool path, where the commonly used least squares fit at the data points was used as a controlled comparison as the objective coefficients for the chord minimization and the jerk minimization were varied. While increasing the influence of the chord error minimization affected the curve positively in terms of the observed fluctuations, the influence of the least squares objective at the data points was still required to preserve the overall shape of the curve and ensure that poor parameterization of the curve does not occur. This was the same for the jerk minimization objective, as the resulting curve's curvature decreased; however increasing

the jerk minimization coefficient too much started to have a detrimental effect. It was concluded that the chord error minimization and the jerk minimization should be used conservatively while preserving the influence of the least squares error at the data points.

By combining the smooth curve fitting technique and using the double spline format to simultaneously represent both the tool tip position and the tool orientation, it was shown that a single curve parameter can be used to accurately represent and interpolate the toolpath information. The fitting algorithm was tested on sample position and orientation toolpaths and feasibility was shown as the specified fitting tolerances were met. This representation allows for a convenient integration into a 5-axis system without the unnecessary computation from additional interpolations steps. Using the double spline format, one of the interpolation techniques can be applied to the lower spline in real time on the CNC machine to generate the reference tool tip position and the tool orientation. The real time interpolation techniques of the Taylor expansion and the Feed Correction Polynomial were analyzed. The Taylor expansion method is difficult to implement using a variable feedrate and shows high feed fluctuations where the toolpath has a high curvature. In addition, the computation of the higher order derivatives of the NURBS curve with the Taylor expansion can be computationally expensive due to the recursive nature of the B-spline basis functions. The Feed Correction Polynomial on the other hand, has extremely low real time computation time, as the polynomial coefficients are pre-calculated in pre-processing. It is also robust to using a variable feedrate and provides an accurate mapping between the spline length and the curve parameter. In addition, using a higher mean square error tolerance of fitting the polynomial has been shown to have good results with little feed fluctuations.

While proper interpolation can reduce feed motion fluctuation of the cutting tool, the toolpath fitting process is also extremely important. Since the overall feed motion is the combined coordinated motion of all the motor axes, if the toolpath is inherently full of fluctuations, (high jerk and curvature content, etc.) the resulting motion for the individual axes can be oscillatory. The interpolation methods tries to output the next curve parameter value, u, within the given sampling time to provide a consistent tool displacement, ds. The fluctuation of the individual axes can cancel each other out for the resulting feed to exhibit very low fluctuations. This is a limitation of the interpolation methods, since the individual axes can become excited and cause undesirable vibration with the respective axis. This ultimately means

that both the toolpath planning and the interpolation need to work effectively to produce a smooth and continuous tool motion.

A multi-segment limited jerk feed profile was presented to continuously modulate the feedrate between segments of constant feed to meet the local constraints that were imposed by the kinematic motor constraints and the part geometrical constraints such as chord error and curvature. In order to properly apply these constraints, the inverse kinematics was applied to determine how the toolpath motion is translated into the individual motion of the motors. Due to the varying demands on the different motors along different regions of the toolpath, an overall feedrate constraint was incurred. From the experimental and simulation results, the reference commands of the motors did not violate the limits; however, the actual motor positions may end up violating the limits due to the transient response from the position tracking. Particularly, it was noted that the largest contour errors occurred where the limited jerk was applied to modulate the feedrate. This problem can be improved by applying a smoother feed profile than the limited jerk, such as the cubic acceleration. The advantage of using the limited jerk profile over an optimized profile is because it is well-defined, providing insights on the feasible feedrate regions and is able to modulate between constant feedrate easily. The cubic acceleration profile should be capable of the same thing, with much lower frequency within its jerk content, and thus is more robust from exciting the system.

While toolpath generation, toolpath interpolation and feedrate scheduling are responsible for generating the motor reference commands, the CNC motion controller is responsible with the tracking performance of the reference commands. Some commonly used controller algorithms were compared using a simple toolpath and they were tested on a large-sized X-Y table that was driven by industrial Siemens motors.

6.2 Future Research Direction

In continuation to this work, some possible future research direction is to analyze how the fitting of the toolpath can affect the frequency content of the reference command under constant feed. If the natural modes of the CNC machine system are known, the toolpath fitting may be adjusted to specifically avoid these modes. Another possible research direction is to analyze the spatial frequencies of the reference commands, as high frequencies often indicate sharp edges. Ideas from optics design of spatial filters may be used to filter out the high

frequencies; however, algorithms will have to be developed such that the geometrical machining tolerances are not affected and can still be met.

During the toolpath fitting process, the corresponding curve parameters on the NURBS curve for the data points, were first estimated based on the chord length and were later found using the minimum distance condition with the Newton Raphson method, this can be improved computationally by narrowing down the search regions and there may be inherent properties within the NURBS curve that can be used to better estimate these corresponding curve parameter values, rather than through curve displacement. It was mentioned that curve displacement generally has a nonlinear relationship with the curve parameter; thus using this criteria in estimating the parameters may not be the best option.

For the feed profile, the limited jerk profile was used because of its simple and well defined formulations, allowing for a known feasible feedrate range for the feedrate modulation. However, the non-continuous jerk profile was problematic and caused undesirable tracking error as the feedrate was modulated. It also contains a high frequency content that is capable of exciting the motor drives. A cubic acceleration or a cubic jerk feed profile also have well defined formulations. The multi-segmenting feed profile scheduling method can be easily extended to these other types of profile that have shown higher continuity and smoother motion. Additionally, as the feed scheduling becomes more developed in real time applications, it should be possible to apply control laws to the overall process; including the feedrate scheduling. It is undeniable that having a lower feedrate can improve contour error, and including the feedrate as part of the control loop can be robust to operational disturbances that are hard to estimate in pre-processing. However, as it is desirable to keep feedrate as constant as possible, look-ahead algorithms as well as effective scheduling algorithms will have to be developed.

In terms of controller design, the CNC position controllers can be designed to specifically reduce contour error rather than the individual axis's tracking error. The dynamics of the CNC kinematics has to be accounted for to formulate an expression of the contour error.

The overall scheme of continuous tool motion can be applied to robotic applications in general. There are many types of industrial robots that can benefit from the optimization of their motion to increase productivity. New CNC configurations involving robotic arms are gaining popularity, having entirely different workspace and constraints than conventional CNC machines.

Bibliography

[1] Affouard, A., Duc, E., Lartigue, C. Langeron, J.-M., Bourdet, P., 2004, "Avoiding 5-Axis Singularities using Tool Path Deformation", Journal of Machine Tools and Manufacture, Vol. 44, Iss. 4, pp. 415-425.

[2] Altintas, Y., 2000, "Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations and CNC Design", Cambridge University Press.

[3] Altintas, Y., Erkorkmaz, K., Zhu, W.-H., 2000, "Sliding Mode Controller Design for High Speed Drives", CIRP Annuals – Manufacturing Technology, Vol. 49, Iss. 1, pp. 265-270.

[4] Altintas, Y., 2009, "Computer Control of Mechatronics Systems (Course Notes)", Mechanical Engineering, The University of British Columbia, Vancouver.

[5] Bahr, B., Xiao, X., Krishnan, K., 2001, "A Real-time Scheme of Cubic Parametric Curve Interpolations for CNC Systems", Computers in Industry, Vol. 45, Iss. 3, pp. 309-317.

[6] Chen, X.-D., Yong, J.-H., Wang, G., Paul, J.-C., Xu, G., 2008, "Computing the Minimum Distance between a Point and a NURBS Curve", Computer-Aided Design, Vol. 40, Iss. 10-11, pp. 1051-1054.

[7] Cheng. M.-Y., Tsai, M.-C., Kuo, J.-C., 2002, Real-time NURBS Command Generators for CNC Servo Controllers", Journal of Machine Tools and Manufacture, Vol. 42, Iss. 7, pp. 801-813.

[8] Cheng, C.-W., Tsai, M.-C., 2004, "Real-Time Variable Feedrate NURBS Curve Interpolator for CNC Machining", Journal of Advanced Manufacturing Technology, Vol. 23, No. 11-12, pp. 865-873.

[9] Constantinescu, D., Croft, E.A., 2000, "Smooth and Time Optimal Trajectory Planning for Industrial Manipulators Along Specified Paths", J. Rob. Syst., Vol.17, pp. 233-249.

[10] DSPACE Release, Quick Software Installation Guide, Release 6.1.

[11] DSPACE Systems, First Work Steps, Release 6.0.

[12] DSPACE PPC Controller Board, Hardware Installation and Configuration, Release 6.0.

[13] Erkorkmaz, K., 1999, M.A.Sc. Thesis: "High Speed Contouring Control for Machine Tool Drives", University of British Columbia, Department of Mechanical Engineering, BC, Canada.

[14] Erkorkmaz, K., Altintas, Y., 2001, "High Speed CNC System Design: Part I – Jerk Limited Trajectory Generation and Quintic Spline Interpolation", International Journal of Machine Tools and Manufacture, Vol. 41, No. 9, pp. 1323-1345.

[15] Erkorkmaz, K., Altintas, Y., 2001, "Trajectory Generation for High Speed Milling of Molds and Dies", Proceedings of the 2nd International Conference and Exhibition on Design and Production of Does and Molds, Kusadasi, Turkey, DM_46.

[16] Erkorkmaz, K., Altintas, Y., 2002, "Optimal Spline Interpolation for High Speed Machine Tools", Proceedings of 6th Biennial Conference on Engineering Systems Design and Analysis, Istanbul, Turkey.

[17] Erkorkmaz, K., Altintas, Y., 2003, "Feedrate Optimization for Spline Interpolation in High Speed Machine Tools", CIRP Annals – Manufacturing Technology, Vol. 52, Iss. 1, pp. 297-302.

[18] Erkorkmaz, K., 2004, Ph.D. Thesis: "Optimal Trajectory Generation and Precision Tracking Control for Multi-Axis Machines", University of British Columbia, Department of Mechanical Engineering, BC, Canada.

[19] Erkorkmaz, K., Altintas, Y., 2005, "Quintic Spline Interpolation with Minimal Feed Fluctuations", Journal of Manufacturing Science and Engineering, Vol. 127, No. 2, pp. 339-349.

[20] Erkorkmaz, K., M. Heng, 2008, "A Heuristic Feedrate Optimization Strategy for NURBS Toolpaths", CIRP Annuals - Manufacturing Technology, Vol. 57, Iss. 1, pp. 407-410.

[21] Heng, M., 2008, "Smooth and Time-Optimal Trajectory Generation for High Speed Machine Tools", University of Waterloo, Department of Mechanical Engineering, Ontario, Canada.

[22] Heng, M., Erkorkmaz, K., 2010, "Design of a NURBS Interpolator with Minimal Feed Fluctuations and Continuous Feed Modulation Capability", International Journal of Machine Tools and Manufacture, Vol. 50, Iss. 3, pp. 281-293.

[23] Ho, M.-C., Hwang, Y.-R., Hu, C.-H., 2003, "Five-Axis Tool Orientation Smoothing Using Quaternion Interpolation Algorithm", International Journal of Machine Tools and Manufacture, Vol. 43, Iss. 12, pp. 1259-1267.

[24] Hsu, Y.Y., Wang, S.S., 2007, "A New Compensation Method for Geometry Errors of Five-Axis Machine Tools", International Journal of Machine Tools and Manufacture, Vol. 47, Iss. 2, pp. 352-360.

[25] Huang, J.-T., Yang, D. C.H., 1992, “Precision Command Generation for Computer Controlled Machines”, ASME Precision Machining: Technology and Machine Development and Improvement, Vol. 58, pp. 89–104.

[26] Jung, Y. H., Lee, D.W., Kim, J. S., Mok, H. S., 2002, “NC Post-Processor for 5-axis Milling Machine of Table-rotation/tilting type”, Journal of Materials Processing Technology, Vol 130-131, pp. 641-646.

[27] Koren, Y., Lo, C.C., Shpitalni, M., 1993, “CNC interpolators: Algorithms and Analysis”, ASME Production Engineering Division, Proceedings of the 1993 ASME Winter Annual Meeting, Vol. 64, pp. 83-92.

[28] Koren, Y. Lin, R. S.,1994, “Real-Time Five Axis Interpolator for Machining Ruled Surfaces”, Proceedings of the 1994 International Mechanical Engineering Congress and Exposition, ASME Dynamic System and Control Division, Vol. 55-2, pp. 951-959.

[29] Koren, Y., Lin, R.-S., 1995, “Five-Axis Surface Interpolators”, Annals of CIRP, Vol. 44, No. 1, pp. 379-382.

[30] Lai, Y.-L., 2010, “Tool-Path Generation of Planar NURBS Curve”, Robotics and Computer-Integrated Manufacturing, Vol. 26, Iss. 5, pp. 471-482.

[31] Langeron, J.M., Duc, E., Lartigue, C., Bourdet, P., 2004, “A New Format for 5-Axis Tool Path Computation, using B-Spline Curves”, Vol. 36, Iss. 12, pp. 1219-1229.

[32] Lei, W.T., Sung, M.P., Lin, L.Y., Huang, J.J., 2007, “Fast Real-Time NURBS Path Interpolation for CNC Machine Tools”, International Journal of Machine Tools and Manufacture, Vol. 47, Iss. 10, pp. 1530-1541.

[33] Lei, W.T., Wang, S.B., 2009, “Robust Real-Time NURBS Path Interpolators”, International Journal of Machine Tools and Manufacture, Vol. 49, Iss. 7-8, pp 625-633.

[34] Li, W., Liu, Y., Yamazaki, K., Fujisima, M., 2008, “The Design of a NURBS Pre-Interpolator for Five-Axis Machining”, International Journal of Advanced Manufacturing Technology, Vol. 36, No. 9-10, pp. 927-935.

[35] Lin, H., Wang, G., Dong, C., 2003, “Constructing Iterative Non-Uniform B-Spline Curve and Surface to Fit Data Points”, Science in China Series F: Information Sciences, Vol. 47, No. 3, pp. 315-331.

[36] Lin, M.-T., Tsai, M.-S., Yau, H.-T., 2007, “Development of a dynamic-based NURBS Interpolator with Real-Time Look-Ahead Algorithm”, International Journal of Machine Tools and Manufacture, Vol. 47, Iss. 15, pp. 2246-2262.

[37] Lin, R.-S., 2000, "Real-Time Surface Interpolator for 3-D Parametric Surface Interpolator for 3-D Parametric Surface Machining on 3-Axis Machine Tools", International Journal of Machine Tools and Manufacture, Vol. 40, Iss. 10, pp. 1513-1526.

[38] Liu, X., Ahmad, F., Yamazaki, K., Mori, M., 2004, "Adaptive Interpolation Scheme for NURBS Curves With the Integration of Machining Dynamics", International Journal of Machine Tools and Manufacture, Vol. 45, Iss. 4-5, pp. 433-444.

[39] Lo, C.-C., 1998, "A New Approach to CNC Tool Path Generation", Computer-Aided Design, Vol. 30, Iss. 8, pp. 649-655.

[40] Macfarlane, S., Croft, E.A., 2003, "Jerk-Bounded Manipulator Trajectory Planning: Design for Real-time Applications", Transactions of IEEE on Robotics and Automation, Vol. 19, No. 1, pp. 42-52.

[41] Makhanov, S.S., 2010, "Adaptive Geometric Patterns for Five-Axis Machining: A Survey", International Journal of Advanced Manufacturing Technology, Vol. 47, No. 9-12, pp. 1167-1208.

[42] Matlab version 7.9.0 Natick, Massachusetts: The MathWorks Inc., 2009.

[43] Mathews, J.H., Fink, K.K., 2004, "Numerical Methods Using Matlab (4^{th} Edition), Prentice-Hall Inc.

[44] Medioni, G., Yasumoto, Y., 1987, "Corner Detection and Curve Representation using Cubic B-Splines", Computer Vision, Graphics, and Image Processing, Vol. 39, Iss. 3, pp. 267-278.

[45] Omirou, S.L., 2003, "Space Curve Interpolation for CNC Machines", Journal of Materials Processing Technology, Vol. 141, Iss. 3, pp. 343-350.

[46] Park, J., Nam, S., Yang, M., 2005, "Development of a Real-Time Trajectory Generator for NURBS Interpolation Based On The Two-Stage Interpolation Method", Vol. 26, No. 4, pp. 359-365.

[47] Piegl, L., Tiller, W., 1997, "The NURBS Book", Springer Verlag.

[48] Piegl, L., Tiller, W., 2002, "Biarc Approximation of NURBS Curves", Computer-Aided Design, Vol. 34, Iss. 11, pp. 807-814.

[49] Selimovic, I., 2006, "Improved Algorithms for the Projection of Points on NURBS curves and surfaces", Computer Aided Geometry Design, Vol. 23, Iss.5, pp. 439-445.

[50] Sencer, B., 2005, M.A.Sc. Thesis: "Five-Axis Trajectory Generation Methods", University of British Columbia, Department of Mechanical Engineering, BC, Canada.

[51] Sencer, B., Altintas, Y., Croft, E. A., 2008, "Feed Optimization for Five-Axis CNC Machine Tools with Drive Constraints", International Journal of Machine Tools and Manufacture, Vol. 48, Iss. 7-8, pp. 733-745.

[52] Sencer, B., Altintas, Y., Croft, E. A., 2009, "Modeling and Control of Contouring Errors for Five-Axis Machine Tools. Part I – Modeling", Transactions of ASME Journal of Manufacturing Science and Engineering, Vol. 131, Iss. 3, 8 pages.

[53] Sencer, B., Altintas, Y., Croft, E. A., 2009, "Modeling and Control of Contouring Errors for Five-Axis Machine Tools. Part II – Precision Contour Controller Design", Transactions of ASME Journal of Manufacturing Science and Engineering, Vol. 131, Iss. 3, 10 pages.

[54] Sencer, B., 2009, Ph.D. Thesis: "Smooth Trajectory Generation and Precision Control of 5-Axis CNC Machine Tools", University of British Columbia, Department of Mechanical Engineering, BC, Canada.

[55] Shpitalni, M., Koren, Y., Lo, C.C., 1994, "Realtime Curve Interpolators", Computer-Aided Design, Vol. 26, Iss. 11, pp. 832-838.

[56] Siemens, "Quick Startup Guide for SIMODRIVE 611 Universal", Version 1.0, 2000

[57] Siemens, "SIMODRIVE 611 digital – Drive Converters – Configuration Manual", 05/2008 Edition

[58] Siemens, "SIMODRIVE 611 MASTERDRIVES MC – Synchronous Servomotors 1FT6 – Configuration Manual", 10/2005 Edition

[59] Siemens, "SIMODRIVE 611 Universal – Control Components for Closed-Loop Speed Control and Positioning – Function Manual", 09/2008 Edition

[60] Tsai, M.-C., Cheng, C.-W., 2003, "A Real-Time Predictor-Corrector Interpolator for CNC Machining", Journal of Manufacturing Science and Engineering, Vol. 125, Iss. 3, pp. 449-460.

[61] Tsai, Y.-F., Farouki, R.T., Feldman, B., 2001, "Performance Analysis of CNC Interpolators for Time-Dependent Feedrates along PH Curves", Computer Aided Geometric Design, Vol. 18, No. 3, pp. 245-265.

[62] Wang, F.-C., Yang, D.C.H., 1993, "Nearly Arc-Length Parameterized Quintic-Spline Interpolation for Prescision Machine", Computer-Aided Design, Vol. 25, No. 5, pp. 281-288.

[63] Wang, F.-C., Schofield, S., Wright, P.K., 1996, "Real Time Quintic Spline Interpolator for an Open Architecture Machine Tool", ASME Dynamic Systems and Control Division,

Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exposition, Vol. 58, pp. 291-297.

[64] Wang, F.-C. Wright, P.K., 1998, "Open Architecture Controllers for Machine Tools, Part 2: A Real Time Quintic Spline Interpolator", Journal of Manufacturing Science and Engineering, Vol. 120, No. 2, pp. 425-432.

[65] Xu, H.-Y., Dai, J., Tam, H.-Y., Zhou, Y., 2003, "Angular Feedrate Interpolation for Three-Dimensional Implicit Curves", International Journal of Production Research, Vol. 41, No. 15, pp. 3461-3478.

[66] Xu, H.-Y., Tam, H.-Y., Zhou, Z., Tse, P.W., 2001, "Variable Feedrate CNC Interpolation for Planar Implicit Curves", International Journal of Advanced Manufacturing Technology, Vol. 18, No. 11, pp. 794-800.

[67] Yeh, S.-S., Hsu, P.-L., 1999, "The Speed-Controlled Interpolator for Machining Parametric Curves", Comptuer-Aided Design, Vol. 31, Iss. 5, pp. 349-357.

[68] Yeh, S.-S., Hsu, P.-L., 2001, "Adaptive-Feedrate Interpolation for Parametric Curves with a Confined Chord Error", Computer-Aided Design, Vol. 34, No. 3, pp. 229-237.

[69] Zhang, Q.G., Greenway, R.B., 1998, "Development and Implementation of a NURBS Curve Motion Interpolation", Robotics and Computer-Integrated Manufacturing, Vol. 14, No. 1, pp. 27-36.

[70] Zhiming, X., Jincheng, C., Zhengjin, F., 2002, "Performance Evaluation of a Real-Time Interpolation Algorithm for NURBS Curves", International Journal of Advanced Manufacturing Technology, Vol. 20, No. 4, pp. 270-276.

[71] Zhou, K., Wang, G., Jin, H., Tan, Z., 2008, "NURBS Interpolation Based On Exponential Smoothing Forecasting", International Journal of Advanced Manufacturing Technology, Vol. 39, No. 11-12, pp. 1190-1196.