AN ABSTRACT OF THE DISSERTATION OF

Shingo Tajima for the degree of Doctor of Philosophy in Robotics and Mechanical Engineering presented on June 7, 2019.

Title: Smooth Trajectory Generation for Machine Tools and Industrial Robots.

Abstract approved: _____

Burak Sencer

This thesis presents accurate and time-optimal smooth reference trajectory generation techniques for manufacturing equipment such as high-speed machine tools (MT) and industrial robots (IR). Typical machining tool-paths for MTs and IRs are defined as a series of discrete linear moves. Although Point-to-Point (P2P) feed motion can be generated by interpolating each linear segment with high-order velocity profiles, the continuous and accurate transition between consecutive segments is necessary to realize a non-stop contouring motion for efficient manufacturing. To generate continuous feed motion along sharp cornered tool-paths, most numerical control (NC) systems blend (smooth) corners locally using various curves and splines. The feed (speed) is reduced around the blend sections so that the motion system's kinematic limits are respected. This thesis proposes 2 novel techniques to enable modern MT and IR to generate non-stop rapid motion along discrete tool-paths. Firstly, a Kinematic Corner Smoothing (KCS) technique has been proposed to generate time-optimal (minimum time) motion trajectories in a real-time within axis kinematic limits. A novel real-time interpolation technique based on Finite Impulse Response (FIR) filtering has also been proposed to suppress residual vibrations for high positioning accuracy of machine tools and motion systems as well. These two techniques are tailored for Cartesian structured motion systems such as 2-3 axis machine tools. Finally, a decoupled FIR filtering technique has been developed to synchronously interpolate tool position and orientation for accurate motion generation for 5-axis MTs and

IRs. These techniques are computationally lightweight and suitable for real-time implementation on modern NC systems. Simulation and experimental validation on Cartesian and 5-axis machine tools are presented to validate the effectiveness of the developed algorithms to interpolate along with discrete commands for high-speed and high-accuracy motion.

Smooth Trajectory Generation for Machine Tools and Industrial Robots

by
Shingo Tajima

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 7, 2019
Commencement June 2019

Doctor of Philosophy dissertation of Shingo Tajima presented on June 7, 2019.

APPROVED:

Major Professor, representing Robotics and Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Shingo Tajima, Author

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my adviser, Dr. Burak Sencer, for providing guidance and support throughout my studies. His hardworking and enthusiastic approach to his work is inspiring, and it has been a pleasure working with him.

I wish to thank my colleagues at Manufacturing Process Control Laboratory, as well as the faculty and staff of School of Mechanical, Industrial and Manufacturing Engineering.

Finally, I would like to gratefully acknowledge the support of my family and friends in my graduate studies.

CONTRIBUTION OF AUTHORS

Dr. Eiji Shamoto assisted as supervising of writing of Chapter 4.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

1. Introductions

Rapid and accurate interpolation of machining tool paths is a broad problem for modern production machinery such as machine tools (MTs) and industrial robots (IRs). Reference tool-paths for those manufacturing equipment are planned by discrete linear moves, or set of waypoints [1], [2]. Once this polygonal tool-path is commanded, modern numerical control (NC) systems try to interpolate continuous tool motion. Nevertheless, discrete nature of the tool-path limits generation of smooth and accurate motion trajectories, and it is a major bottleneck in achieving higher manufacturing efficiency and productivity. 3 major challenges are tacked in this thesis on the interpolation of discrete tool-paths; namely, time optimal blending of polygonal tool-paths, mitigation of unwanted vibrations, and development of computationally efficient real-time algorithms.

High-speed and high-accuracy are the two key targets in modern manufacturing. Corner smoothing (path blending) is one way to generate non-stop smooth tool motion along discrete linear machining tool-paths. Figure 1.1 illustrates application of corner smoothing. The interpolation of consecutive Point-to-Point (P2P) moves must be controlled precisely within the manufacturing tolerances. Furthermore, since most part programs in die-and-mold and aerospace industrial consist of thousands of discrete points, time-optimal path-blending is critical for productivity. In robotics literature, path-blending problem is addressed with spline interpolation techniques [3], [4]. Although, spline interpolation can provide smoother kinematic profiles, control of path errors is difficult and generation of time optimal feed profiles is computationally expensive for real-time implementation. Hence, the use of spline interpolation in limited. Instead, manufacturing and machine tool literature has been focusing on computationally lightweight time-optimal corner smoothing techniques [5]–[8].

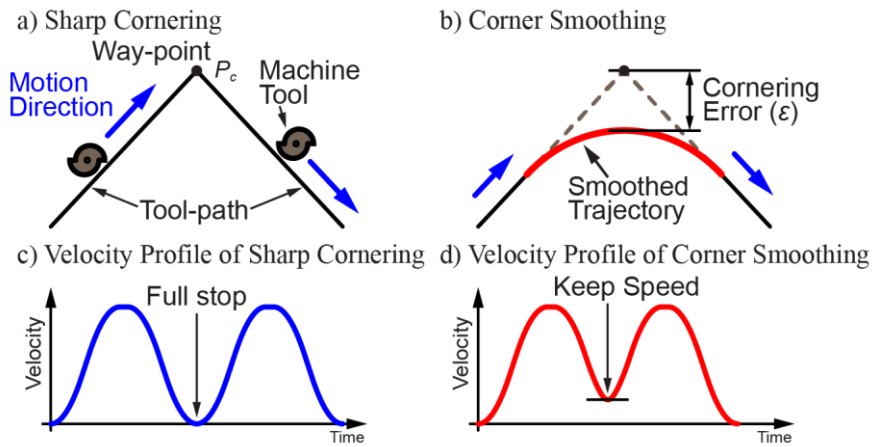

Figure 1.1: Interpolation of linear segments with corner smoothing technique.

In addition to controlling geometric path blending errors, vibrations that originate due to the structural flexibilities in the manufacturing equipment is another source of errors [9]. Figure 1.2 illustrates frequency response model in the time domain (t) and frequency domain (ω). Forced vibrations are either induced by process forces due to interaction of machine with the manufacturing process, or by the reference trajectory as the machine accelerates and decelerates along the tool-path. Typically, high-speed motion trajectories demand high acceleration from axes in motion [10]. This, in return, generates large inertial reaction forces on the machine structure. Frequency content of inertial forces can excite resonances on the machine and cause unwanted forced vibrations which can deteriorate positioning accuracy of MTs or the IRs, and hence the manufacturing quality [11].

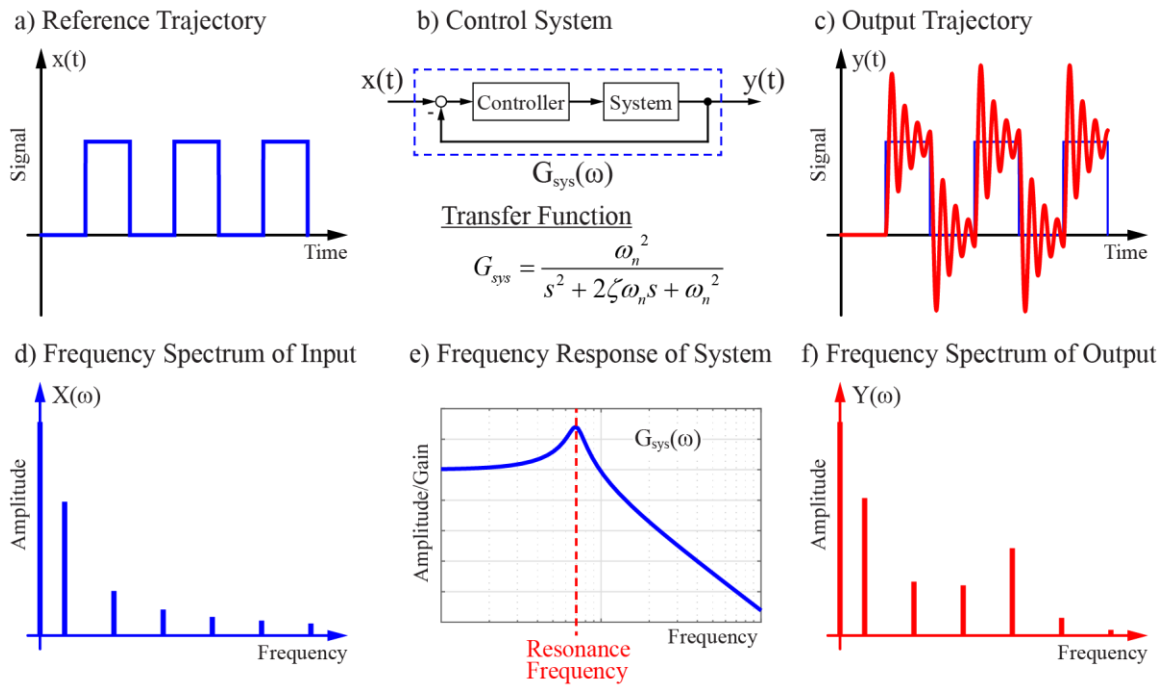

Figure 1.2: Frequency response model.

Accurate blending of discrete moves for 5-axis CNC machines is another challenging problem [12]. Figure 1.3 illustrates interpolation of the discrete tool pose motion of 5-axis CNC machines. As compared to the 3-axis Cartesian machine tools, 5-axis machines can alter tool orientation in synch with the tool-tip during simultaneous 5-axis machining. This functionality enables them to machine complex sculptured surfaces faster and achieve significantly better finish quality. The tool-path contains both translational motion of the tool center point (TCP) and the rotational motion of the tool axis orientation (ORI). Both the translational motion of the tool in Cartesian coordinates and the tool orientation in spherical coordinates [13] must be blended continuously [14] in a synchronized matter [15].

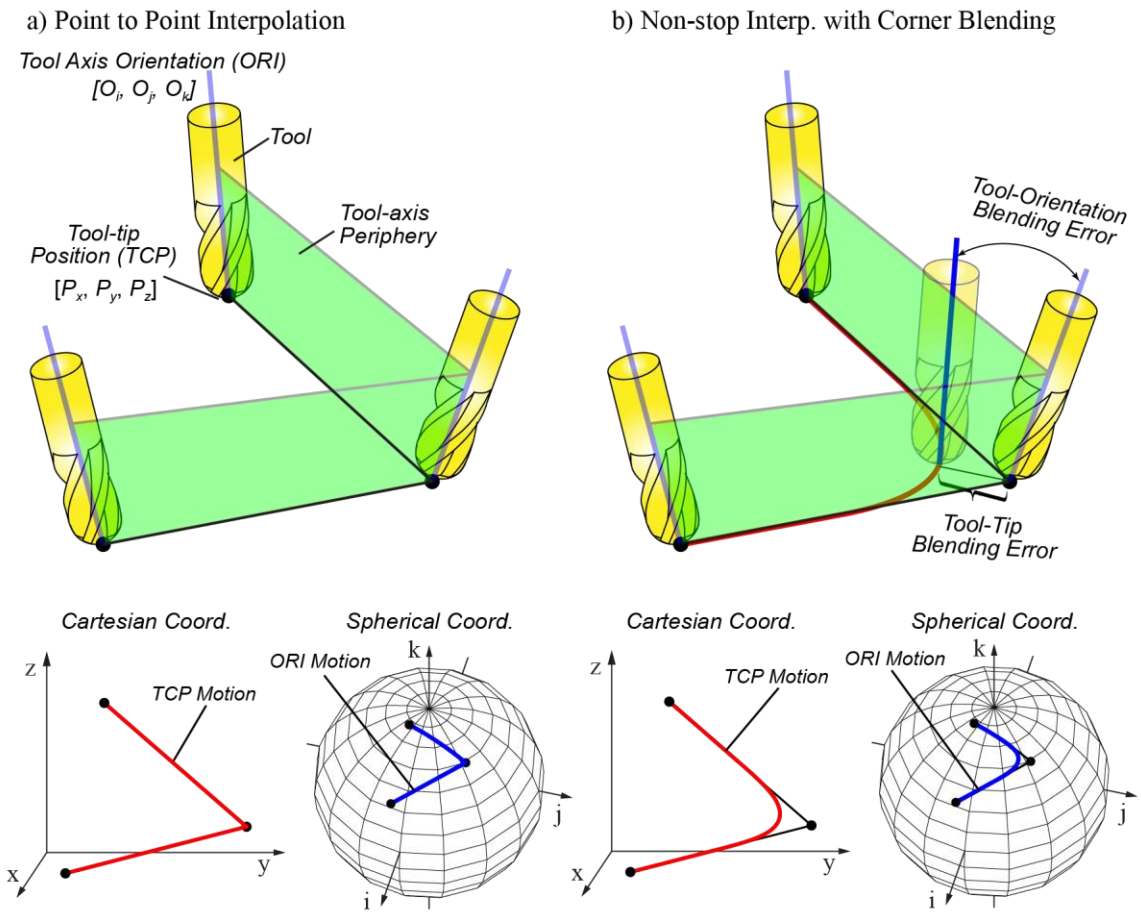

Figure 1.3: Discrete tool pose motion interpolation of 5-axis CNC machines.

The thesis deals with accurate and time-optimal smooth reference trajectory generation and is organized as follows: The kinematic corner smoothing (KCS) technique for optimal accurate motion planning are presented in Chapter 2. This KCS technique is extended in Chapter 3 to generate a smooth motion planning along short-segmented tool-paths. The FIR filtering based trajectory generation technique for non-vibration is accurate motion are presented in Chapter 4. The FIR filtered based tool orientation blending technique for tool pose motion blending is presented in Chapter 5. Finally, conclusions are expressed in Chapter 6.

Kinematic Corner Smoothing for high speed machine tools

Shingo Tajima and Burak Sencer

International Journal of Machine Tools & Manufacture

Volume 108, September 2016, Pages 27-43

2. Kinematic Corner Smoothing for high speed machine tools

This paper presents a novel kinematic corner smoothing technique for high-speed CNC machine tools. Typically, reference tool-paths compromised of short G01 moves are geometrically smoothed by means of arcs and splines within the NC system. In this study, a continuous feed motion is generated by directly planning jerk limited velocity transitions for the drives in the vicinity of sharp corners of the tool-path. This approach completely eliminates the need for geometry based path smoothing and feed planning. Contouring errors at sharp corners are controlled analytically by accurately calculating cornering speed and duration. Since proposed approach bases on kinematically planning axis motion profiles, it exploits acceleration and jerk limits of the drives and delivers a near-time optimal motion. Experimental validation and comparisons are presented to show significant improvement in the cycle time and accuracy of contouring Cartesian tool-paths.

2.1. Introduction

CAD systems are utilized to design complex geometries based on smooth curves such as NURBS or B-splines [16]. Direct interpolation of these curves is proven to be superior in terms of providing smoother and faster motion in high-speed machining [17], [18]. However, most CNC machines are not capable of efficiently interpolating higher order parametric curves in real-time. Accurate calculation of curve lengths [19], planning of time efficient feed profiles [20]–[22] suppression of feed fluctuations [4] and control of chord errors during real-time interpolation are major bottlenecks still being addressed both by academia and NC builders. Instead, CAM systems are aided to discretize original smooth part geometry with numerous short line segments, and NC systems are fed with linear “point-to-point” motion commands. Interpolating paths compromised of linear segments limits productivity. Since linear moves are not continuous, motion has to stop at junction points of linear segments, i.e. corners, leading elongated cycle times and generating rough, cornered surface finishes [23].

Local “corner blending (smoothing)” techniques have been proposed to achieve non-stop continuous motion [5], [24]. The idea is well-known and straightforward. In order to realize a continuous transition between consecutive linear segments, sharp corner is replaced with a smooth blending curve by the NC system. As a result, the corner is no longer sharp, and reference path deviates from the original geometry. As a matter of fact, this deviation is not detrimental during high-speed machining since sharp corners are rarely executed in roughing or semi finishing operations. Instead, corners are programmed to be traversed continuously, subject to manufacturing tolerance constraints and kinematic limits of the machine [25]. Thus, key requirements in continuous cornering are continuity [23], accuracy [26] and speed [27].

Current literature solves corner smoothing problem in two steps, namely; curve fitting followed by feed profile planning. First, corner geometry is smoothed by fitting a highly continuous curve under user specified cornering tolerances. Jouaneh et. al. [5] replaced the corner with a circular arc for fast cornering. However, a circular arc only delivers velocity continuous (C1) motion transition. Later they used two clothoid curves to realize acceleration continuous (C2) motion transition [24]. Yutkowitz and Chester [6] utilized two quartic splines to generate curvature continuous cornering geometry within user-specified tolerances. Sencer et al. [8] solved curve fitting problem with a single Quintic Bezier curve and others [7], [28], [29] used B-spline curves to control both corner geometry and the continuity. Beudaert et. al. [15] extended sharp corner smoothing in five-axis machining paths.

Once sharp corners are smoothened, it becomes a mixture of linear segments continuously blended with splines. Thus, the second step is scheduling of a feed profile. Due to curved corner profile, tangential speed (feedrate) must be lowered so that axis velocity and acceleration limits are not violated at corner sections [23]. Jerk limited acceleration profile (JLAP) is widely used in high-speed machining [30], [31]. It generates trapezoidal acceleration transitions with piecewise constant jerk segments, which helps avoiding excitement of inertial vibrations of feed drive system and provides a practical balance between smoothness and time-optimality. Erkorkmaz and Altintas [32] planned

JLAP based trajectories along spline tool-paths. Others generated JLAP based feed profiles along corner smoothed tool-paths [7], [8], [29].

Nevertheless, separation of corner smoothing problem into curve fitting and feed planning is an inefficient approach. Since smoothened corner geometry is essentially a parametric curve, it suffers from bottlenecks related to real-time interpolation [4], [8], [26]. In addition, planning of a time optimal feed profile along corner blend is computationally stringent [21]. As a result, conservative cornering speeds are selected in real-time implementation [23]. Recent approaches are towards development of kinematic corner blending techniques, which eliminate the need for para-metric curve fitting. Okwudire and Ding [27] used optimal control to generate time-optimal cornering trajectories. Weck and Ye [33] and recently, Sencer et al. [34] investigated on filtering techniques to accurately travel along sharp corners. Tsai and Huang [35] investigated incorporation of servo dynamics into cornering trajectory generation to improve dynamic contouring accuracy.

This paper proposes a novel approach where continuous cornering is achieved without fitting a parametric curve. Instead, we solve the problem "kinematically" by smoothly blending axis velocities from one segment to the other based on the JLAP. Fastest cornering speed, which respects axis velocity, acceleration and jerk limits, and at the same time generates a cornering trajectory within user-specified cornering tolerance computed analytically. Since one of the axis kinematic limits is primarily saturated, proposed "kinematic corner blending" technique provides near time-optimal cornering motion. Section 2 presents the proposed kinematic corner smoothing method based on the JLAP. Section 3 shows illustrative examples, experimental validations and comparisons to past literature. Lastly, Section 4 provides conclusions and discussions.

2.2. Kinematic corner smoothing problem

Majority of NC tool-paths contain series of linear segments as shown in Figure 2.1a. A single planar (x,y) cornering scenario encountered on a Cartesian manufacturing machine is shown in Figure 2.1b. The two consecutive linear segments intersect each other to

generate the sharp corner, $P_c = [x_c, y_c]$. The angles θ_1 and θ_2 define orientation of linear segments, and $\vec{t_s} = [\cos(\theta_1), \sin(\theta_1)]^T$ and $\vec{t_e} = [\cos(\theta_1 + \theta_2), \sin(\theta_1 + \theta_2)]^T$ are the unit vectors defining feed directions along them. As observed, the geometry is position (G^0) continuous, which allows continuous interpolation of axis position commands. However, the feed direction changes discontinuously from t_s to t_e at sharp corner point, P_c. As a result, if sharp corner is to be traveled at constant speed, infinite amount of acceleration is necessary to alter axis velocities at corner point, which saturates the drives. The machine simply has to come to a full-stop at the sharp corner before continuing to the consecutive linear segment. This approach severely elongates cycle time of a manufacturing operation. Therefore, current techniques focus on smoothening sharp corner geometry within specified cornering tolerance so that the machine could traverse non-stop along linear segmented tool-path.

This paper proposes a novel technique where instead of smoothing the path geometry, a smooth and controlled cornering trajectory is generated by designing motion profiles of the axes. Figure 2.2 shows proposed smoothened sharp corner profile. The tool approaches vicinity of the corner at a cornering speed of V_c and an acceleration A_c. As shown in Figure 2.2a–b, the idea is to smoothly blend axis kinematic motion profiles from entry and to the exit of corner so that feed direction can be changed continuously. In order to stay within kinematic limits of the drives, axis kinematic profiles are interpolated at a finite cornering duration of T_c, which in return introduces deviation from original path geometry. Selecting identical V_c and A_c at both ends of the corner generates a symmetrical corner profile around the bisector of the unit tangent vectors, and the maximum deviation from the original sharp cornered geometry occurs at the center (see Figure 2.2a). The problem is to determine the maximum cornering velocity and accelerations, which keeps cornering trajectory within user specified cornering tolerance, ε and utilizes drive's acceleration A_{max} and jerk J_{max} limits to minimize total cornering cycle time.

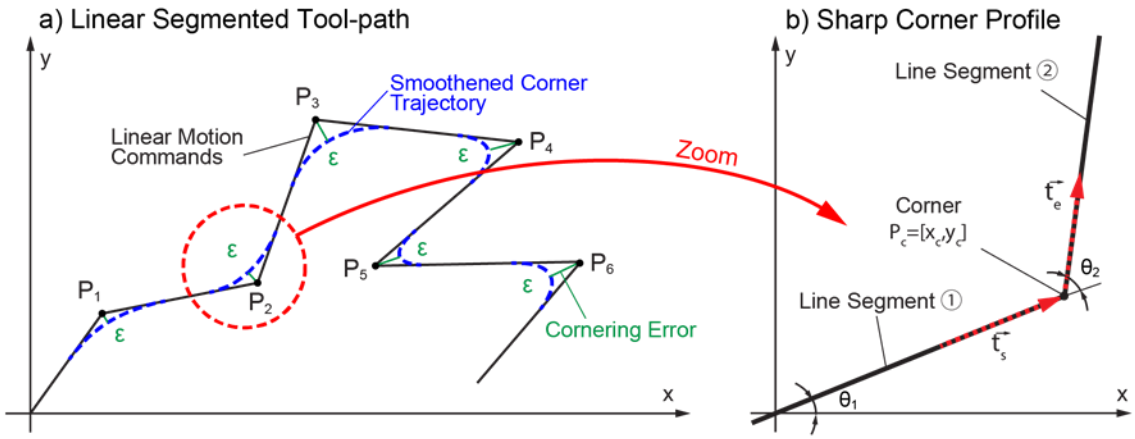

Figure 2.1: Sharp corner smoothing of discrete tool-paths.

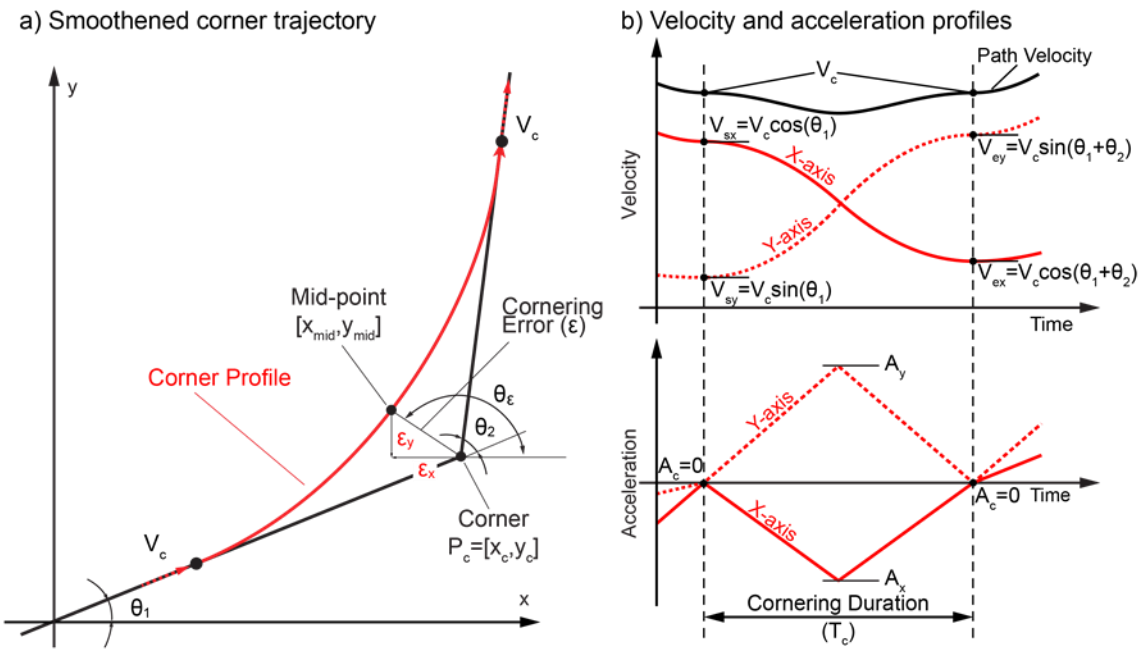

Figure 2.2: Kinematic corner smoothing (KCS) strategy with interrupted acceleration.

2.2.1. Jerk Limited Acceleration Profile

Jerk limited acceleration profile (JLAP) is a widely used trajectory generation scheme in modern CNC machine tools [23], [30], [36]. It is used to accelerate or decelerate the tool from an initial velocity and acceleration, to a final velocity and acceleration within pre-determined acceleration and jerk limits. Figure 2.3 shows the jerk limited acceleration

profile. It consists of 3 phases. In phase 1, acceleration is increased at constant rate controlled by the piecewise constant jerk, J_1. This is followed by a constant (cruise) acceleration phase A, and acceleration is decelerated at a constant rate of J_3 in phase 3. Through this 3-phased acceleration profile, both initial velocity of Vs and acceleration As are smoothly blended with the final velocity V_e and acceleration A_e. If the initial conditions for displacement S_s, velocity V_s and acceleration A_s are known, and the jerk profile is known, acceleration $a(t)$, velocity $v(t)$ and displacement $s(t)$ profiles can be obtained by integrating the jerk $j(t)$ as,

$$a(t) = A_s + \int_0^t j(\tau)d\tau, \quad v(t) = V_s + \int_0^t a(\tau)d\tau, \quad s(t) = S_s + \int_0^t v(\tau)d\tau \,. \quad (2.1)$$

The jerk profile during acceleration/deceleration durations in Figure 2.3 can be written as,

$$j(\tau) = \begin{cases} J_1, & 0 \le t < t_1 \\ 0, & t_1 \le t < t_2 \\ J_3, & t_2 \le t < t_3 \end{cases} \quad (2.2)$$

where t denotes absolute time, t_1, t_2, t_3 denote the time boundaries of each phase; J_1 and J_3 are jerk values in phases 1 and 3. Integrating Eq. (2.2) with respect to time, reveals the trapezoidal acceleration profile

$$a(\tau) = \begin{cases} A_s + J_1\tau_1, & 0 \le t < t_1 \\ A, & t_1 \le t < t_2 \\ A + J_3\tau_3, & t_2 \le t \le t_3 \end{cases} \quad (2.3)$$

where A is the acceleration amplitude, and τ_k is the relative time parameter, which starts at the beginning of the kth phase as shown in Figure 2.3. Similarly, integrating Eq. (2.3) with respect to time generates the velocity profile as,

$$v\ \tau\ = \begin{cases} V_s + A_s\tau_1 + \frac{1}{2}J_1\tau_1^{\ 2}, & 0 \le t \le t_1 \quad V_1 = V_s + A_sT_1 + \frac{1}{2}J_1T_1^2 \\ V_1 + A\tau_2, & t_1 \le t \le t_2 \quad V_2 = V_1 + AT_2 \\ V_2 + A\tau_3 + \frac{1}{2}J_3\tau_3^{\ 2}, & t_2 \le t \le t_3 \quad V_e = V_2 + AT_3 + \frac{1}{2}J_3T_3^2 \end{cases} \quad (2.4)$$

where V_s is the initial velocity, and V_e denotes the final velocity reached at the end of the motion. T_k (k=1,2,3) is the duration of the kth phase, and V_k is the velocity reached at the end of each corresponding phase. Again, integrating Eq. (2.4) with respect to time yields the displacement profile,

$$s\ \tau\ = \begin{cases} S_s + V_s\tau_1 + \frac{1}{2}A_s\tau_1^{\ 2} + \frac{1}{6}J_1\tau_1^{\ 3}, & 0 \le t \le t_1 \quad S_1 = V_sT_1 + \frac{1}{2}A_sT_1^2 + \frac{1}{6}J_1T_1^3 \\ S_1 + V_1\tau_2 + \frac{1}{2}A\tau_2^{\ 2}, & t_1 \le t \le t_2 \quad S_2 = S_1 + V_1T_2 + \frac{1}{2}AT_2^{\ 2} \\ S_2 + V_2\tau_3 + \frac{1}{2}A\tau_3^{\ 2} + \frac{1}{6}J_3\tau_3^{\ 3}, & t_2 \le t \le t_3 \quad S_e = S_2 + V_2T_3 + \frac{1}{2}AT_3^{\ 2} + \frac{1}{6}J_3T_3^3 \end{cases} \tag{2.5}$$

where s_k (k=1,2,3) is the displacement reached at the end of each kth phase.

In most general usage, jerk limited acceleration profile can be employed to generate smooth velocity and acceleration transition between given kinematic boundaries, i.e. V_s, V_e and A_s, A_e. Owing to the nature of the trapezoidal shape of the profile, acceleration amplitude A can be expressed as:

$$A = A_s + J_1T_1 = A_e - J_3T_3\ , \tag{2.6}$$

and correct signs for acceleration and jerk values can be determined from velocity and acceleration boundary conditions:

$$\left.\begin{aligned} A &= \text{sgn}\ \ V_e - V_s\ \ |A| \\ J_1 &= \text{sgn}\ \ A\ \ |J_1| \\ J_3 &= -\text{sgn}\ \ A\ \ |J_3| \end{aligned}\right\}\ . \tag{2.7}$$

Note that a negative A indicates deceleration instead of acceleration in the motion. The signs of jerk amplitudes are modified accordingly. If the value of A is zero, this indicates absence of an acceleration phase.

The jerk limited acceleration profile is constructed by determining durations of all the three phases, T_1, T_2 and T_3. If a constant acceleration phase exists in the motion, maximum acceleration A=A_{max} is reached at the end of phase 1. The maximum allowable jerk, J_{max} is used to minimize the total motion duration. Therefore, durations for phases 1 and 3 can be computed from Eq. (2.6) as:

$$T_1 = \frac{|A_{\max} - A_s|}{J_{\max}}, \quad T_3 = \frac{|A_{\max} - A_e|}{J_{\max}} \ . \tag{2.8}$$

Knowing that both V_e and A_e are reached at the end of velocity transition, T_2 is computed from Eq. (2.4) to be:

$$\begin{aligned} T_2 &= \frac{1}{A_{\max}}\left[|V_e - V_s| - \frac{A_{\max} + A_s}{2}T_1 - \frac{A_{\max} + A_e}{2}T_3\right] \\ &= \frac{1}{A_{\max}}\left[|V_e - V_s| - \frac{A_{\max}{}^2}{J_{\max}} + \frac{A_s{}^2 + A_e{}^2}{2J_{\max}}\right] \end{aligned} \ . \tag{2.9}$$

On the other hand, if the velocity transition ΔV is small, and acceleration capacity of the drives is large, T_2 computed from Eq. (2.9) may yield a negative value, $T_2 < 0$. In this case, $T_2 = 0$ is set, which eliminates any constant acceleration phase, and acceleration amplitude is adjusted from Eq. (2.9):

$$A = \operatorname{sgn}\ V_e - V_s \ \sqrt{J_{\max}\ \ V_e - V_s\ \ + \frac{A_s{}^2 + A_e{}^2}{2}} \tag{2.10}$$

The non-zero jerk durations are then updated accordingly,

$$T_1 = \frac{|A - A_s|}{J_{\max}}, \quad T_3 = \frac{|A - A_e|}{J_{\max}} \ . \tag{2.11}$$

Once all the segment durations are obtained from Eq. (2.8), (2.9) and (2.11), jerk limited acceleration profile can be constructed to realize smooth velocity and acceleration transitions. The total distance traveled during the transition is obtained as:

$$L = \begin{cases} V_s\ \ T_1 + T_2 + T_3\ \ + \frac{1}{2}A_s T_1^2 + A_s T_1\ \ T_2 + T_3\ \ + \frac{1}{2}A\ \ T_2 + T_3\ \ ^2 & \\ \qquad + \frac{1}{6}J_{\max}\ \ T_1^3 - T_3^3\ \ + \frac{1}{2}J_{\max}T_1^2\ \ T_2 + T_3 & \text{if } T_2 \neq 0 \\ V_s\ \ T_1 + T_3\ \ + \frac{1}{2}A_s\ \ T_1 + T_3\ \ + \frac{1}{6}J_{\max}\ \ T_1^3 - T_3^3\ \ + \frac{1}{2}J_{\max}T_1 T_3\ \ T_1 + T_3 & \text{if } T_2 = 0 \end{cases} . \tag{2.12}$$

Please note that, although the most general form of the JLAP is given in above equations, typically the profile is used to generate smooth speed transition between two cruise velocities. In other words, initial and final accelerations are generally zero, $A_s = A_e$

= 0, and construction of the profile becomes simpler. For instance, the acceleration profile becomes symmetrical from Eq. (2.3),

$$A = J_1 T_1 = J_3 T_3 \text{ where } \begin{cases} A = \operatorname{sgn}(V_e - V_s)|A| \\ J_1 = \operatorname{sgn}(A)|J| \text{ and } J_3 = -J \end{cases} \tag{2.13}$$

and durations of phases 1 and 3 can be become:

$$T_1 = T_3 = \frac{A_{\max}}{J_{\max}} \ . \tag{2.14}$$

Knowing that only V_e is reached at the end of velocity transition, T_2 can be computed from Eq. (2.4) to be

$$T_2 = \frac{|V_e - V_s|}{A_{\max}} - \frac{A_{\max}}{J_{\max}} \tag{2.15}$$

and the non-existence of constant acceleration phase, $T_2 < 0$, is handled by setting $T_2 = 0$ and acceleration amplitude is calculated as:

$$A = \operatorname{sgn}(V_e - V_s)\sqrt{J_{\max}|V_e - V_s|} \ , \tag{2.16}$$

and constant jerk durations are adjusted accordingly from Eq. (2.14):

$$T_1 = T_3 = \frac{|A|}{J_{\max}} \ . \tag{2.17}$$

Finally, total distance traveled during a velocity transition yields

$$L = \begin{cases} \dfrac{V_e + V_s \quad A_{\max}^2 + J_{\max} V_e - J_{\max} V_s}{2 A_{\max} J_{\max}} & \text{if } T_2 \neq 0 \\ \dfrac{\sqrt{V_e + V_s \quad J_{\max} \quad V_e - V_s}}{J_{\max}} & \text{if } T_2 = 0 \end{cases} \ . \tag{2.18}$$

The acceleration phase of the JLAP is presented above, and it can be constructed to either interpolate constant velocity transitions, or velocity transitions with initial accelerations. The deceleration phase can be planned by replacing acceleration amplitudes with negative deceleration values. Therefore, it is omitted here. Following sections present

the proposed kinematic corner smoothing (KCS) algorithm designed based on the jerk limited acceleration profile.

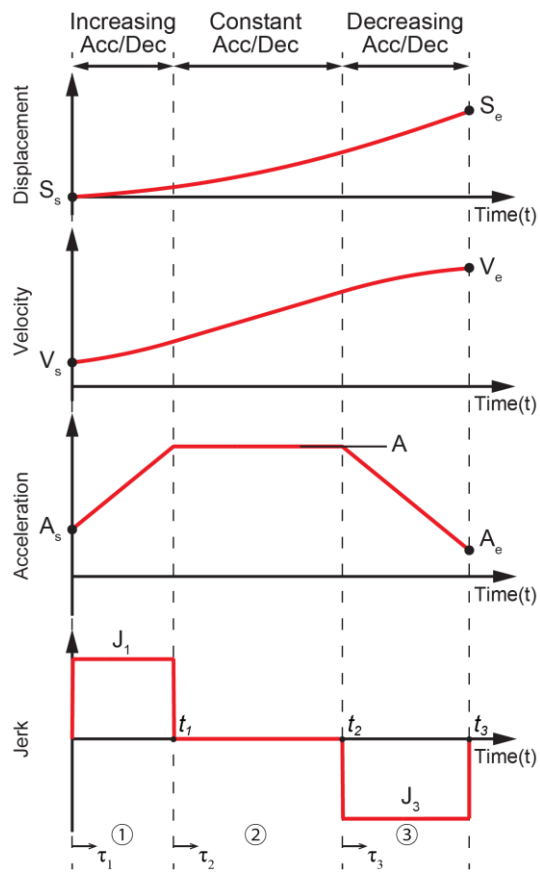

Figure 2.3: Jerk limited acceleration profile (JLAP).

2.2.2. Kinematic corner smoothing (KCS) with interrupted acceleration

This section presents the proposed kinematic corner smoothing (KCS) scheme applied to corners formed by the intersection of long straight lines. For such long line segments, smoothing is highly localized to a corner region that is typically a small fraction of the total length of the line. Thus, we assume that corners do not overlap each other, programmed feedrate along linear segments can be reached, and the machine has capacity to decelerate to a specified cornering speed, V_c. Here, we assume that the cornering motion starts from a constant cornering speed V_c with zero initial acceleration $A_c = 0$. Therefore, we call this method "KCS with interrupted acceleration". As shown in Figure 2.2b, V_c

controls axis velocity boundary conditions at the start (V_{sx}, in V_{sy}) and end (V_{ex}, in V_{ey}) of a cornering trajectory. The objective is to determine the fastest cornering speed, Vc feasible so that axis velocity transitions can be planned, and resultant cornering trajectory deviates from original sharp corner profile by a predetermined geometric tolerance value, ε (See Figure 2.2a).

Assuming that the cornering motion starts and ends at identical tangential cornering velocity V_c, individual *x–y* axis velocity profiles are planned based on the JLAP from Eq. (2.4) as:

$$\begin{aligned} v_x(\tau) &= \left\{ \begin{array}{ll} V_{sx} + \frac{1}{2} J_{1x} \tau_1^2, & 0 \le t < t_1, V_{1x} = V_{sx} + \frac{1}{2} J_{1x} T_1^2 \\ V_{1x} + A_x \tau_2, & t_1 \le t < t_2, V_{2x} = V_{1x} + A_x T_2 \\ V_{2x} + A_x \tau_3 + \frac{1}{2} J_{3x} \tau_3^2, & t_2 \le t \le t_3, V_{ex} = V_{2x} + A_x T_3 + \frac{1}{2} J_{3x} T_3^2 \end{array} \right\}, \\ v_y(\tau) &= \left\{ \begin{array}{ll} V_{sy} + \frac{1}{2} J_{1y} \tau_1^2, & 0 \le t < t_1, V_{1y} = V_{sy} + \frac{1}{2} J_{1y} T_1^2 \\ V_{1y} + A_y \tau_2, & t_1 \le t < t_2, V_{2y} = V_{1y} + A_y T_2 \\ V_{2y} + A_y \tau_3 + \frac{1}{2} J_{3y} \tau_3^2, & t_2 \le t \le t_3, V_{ey} = V_{2y} + A_y T_3 + \frac{1}{2} J_{3y} T_3^2 \end{array} \right\} \end{aligned} \tag{2.19}$$

where starting and ending velocity boundary conditions are calculated form Eq. (2.19) and corner geometry (See Figure 2.3):

$$\underbrace{\left\{ \begin{array}{l} V_{sx} = V_c \cos(\theta_1) \\ V_{sy} = V_c \sin(\theta_1) \end{array} \right\}}_{\text{Starting Velocity Constraints}}, \quad \underbrace{\left\{ \begin{array}{l} V_{ex} = V_c \cos(\theta_1 + \theta_2) = V_{sx} + A_x (T_2 + T_3) + \tfrac{1}{2} J_{1x} T_1^2 + \tfrac{1}{2} J_{3x} T_3^2 \\ V_{ey} = V_c \sin(\theta_1 + \theta_2) = V_{sy} + A_y (T_2 + T_3) + \tfrac{1}{2} J_{1y} T_1^2 + \tfrac{1}{2} J_{3y} T_3^2 \end{array} \right\}}_{\text{Ending Velocity Constraints}} \tag{2.20}$$

where $J_{1x} = -J_{3x}$, $J_{1y} = -J_{3y}$ are axis jerk, and A_x, A_y are axis cornering acceleration amplitudes. For a Cartesian motion system, identical jerk and acceleration limits are generally selected, i.e. $J_{max} = J_{xmax} = J_{ymax}$ and $A_{max} = A_{xmax} = A_{ymax}$. The total velocity transition for each axis during a cornering trajectory can be obtained from Eq. (2.20),

$$\left. \begin{array}{l} \Delta V_x = V_c \left| \cos\ \theta_1 + \theta_2\ - \cos\ \theta_1\ \right| \\ \Delta V_y = V_c \left| \sin\ \theta_1 + \theta_2\ - \sin\ \theta_1\ \right| \end{array} \right\} \tag{2.21}$$

Next, displacement boundary conditions are imposed to control the smoothened corner geometry. Since tangential velocity and acceleration are identical at start and end of the motion, cornering trajectory is symmetrical around the bisector of the unit tangent vectors, t_s and t_e, and hence maximum geometrical deviation from the sharp corner point occurs in the middle of the cornering trajectory. Cartesian coordinates of the mid-point can be computed by integrating Eq. (2.19) and evaluating it at $t = T_1+1/2T_2$ as:

$$\left.\begin{aligned} x_{mid} &= V_c \cos\ \theta_1 \left(T_1 + \frac{T_2}{2}\right) + \frac{1}{2} A_x \left(\frac{T_2}{2}\right)^2 + \frac{1}{6} J_{1x} T_1^3 + \frac{1}{2} J_{1x} T_1^2 \left(\frac{T_2}{2}\right) \\ y_{mid} &= V_c \sin\ \theta_1 \left(T_1 + \frac{T_2}{2}\right) + \frac{1}{2} A_y \left(\frac{T_2}{2}\right)^2 + \frac{1}{6} J_{1y} T_1^3 + \frac{1}{2} J_{1y} T_1^2 \left(\frac{T_2}{2}\right) \end{aligned}\right\} . \quad (2.22)$$

The original sharp corner location $\boldsymbol{P_c} = [x_c, y_c]$ is considered relative from the start of the cornering motion, and it can be defined from the cornering geometry (see Figure 2.2a) as:

$$\left.\begin{aligned} x_c &= L_c \cos\ \theta_1 \\ y_c &= L_c \sin\ \theta_1 \end{aligned}\right\} \quad (2.23)$$

where L_c is the Euclidian length used for corner smoothing. L_c can be calculated based on the corner geometry and total displacement traveled by the drives. For instance, considering X-axis's motion, L_c can be obtained from Eq. (2.18) with the boundary velocity conditions given in Eq. (2.20) as:

$$\begin{aligned} L_c \cdot \cos\ \theta_1 + \cos\ \theta_1 + \theta_2 &= V_c \cos\ \theta_1\ T_1 + \frac{1}{6} J_{1x} T_1^3 + \left(V_c \cos\ \theta_1 + \frac{1}{2} J_{1x} T_1^2\right) T_2 \\ &+ \frac{1}{2} A_x T_2^2 + \left(V_c \cos\ \theta_1 + \frac{1}{2} J_{1x} T_1^2 + A_x T_2\right) T_3 + \frac{1}{2} A_x T_3^2 + \frac{1}{6} J_{3x} T_3^3 \end{aligned} \quad (2.24)$$

and the displacement constraints for the cornering motion is imposed from Eqs. (2.22)-(2.24):

$$\left.\begin{aligned}\varepsilon_x &= x_{mid} - x_c \\ &= V_c \cos\ \theta_1 \left(T_1 + \frac{T_2}{2}\right) + \frac{1}{2} A_x \left(\frac{T_2}{2}\right)^2 + \frac{1}{6} J_{1x} T_1^3 + \frac{1}{2} J_{1x} T_1^2 \left(\frac{T_2}{2}\right) - V_c \cos\ \theta_1 \left(T_1 + \frac{T_2}{2}\right) \\ \varepsilon_y &= y_{mid} - y_c \\ &= V_c \sin\ \theta_1 \left(T_1 + \frac{T_2}{2}\right) + \frac{1}{2} A_x \left(\frac{T_2}{2}\right)^2 + \frac{1}{6} J_{1x} T_1^3 + \frac{1}{2} J_{1x} T_1^2 \left(\frac{T_2}{2}\right) - V_c \sin\ \theta_1 \left(T_1 + \frac{T_2}{2}\right)\end{aligned}\right\} \tag{2.25}$$

where $\varepsilon_x = \varepsilon \cos(\theta_\varepsilon)$, $\varepsilon_y = \varepsilon \sin(\theta_\varepsilon)$ are Cartesian projections of cornering tolerance ε, and $\theta_\varepsilon = \pi/2+\theta_1+\theta_2/2$ is the bisector of the corner (See Figure 2.2a).

Maximum cornering velocity, V_c is sought to generate the fastest cornering speed, which saturates at least one of the axis acceleration or jerk limits. Therefore, V_c is constrained based on the axis, which experiences the largest velocity transition identified from Eq. (2.21). For instance, if $\Delta V_x > \Delta V_y$, X-axis becomes the "limiting axis". Velocity and displacement kinematic conditions are combined from Eqs. (2.20) and (2.25) for the x-axis as:

$$\left.\begin{aligned}\text{Velocity Constraint}: V_{ex} &= V_c \cos\ \theta_1 + \theta_2 \\ &= V_{sx} + A_x\ T_2 + T_3\ + \frac{1}{2} J_{1x} T_1^2 - \frac{1}{2} J_{3x} T_3^2 \\ \text{Displacement Constraint}: \varepsilon_x &= V_c \cos\ \theta_1 \left(T_1 + \frac{T_2}{2}\right) + \frac{1}{2} A_x \left(\frac{T_2}{2}\right)^2 + \frac{1}{6} J_{1x} T_1^3 \\ &\quad + \frac{1}{2} J_{1x} T_1^2 \left(\frac{T_2}{2}\right) - V_c \cos\ \theta_1 \left(T_1 + \frac{T_2}{2}\right)\end{aligned}\right\} \tag{2.26}$$

and motion durations T_1, T_2 and T_3 are identified with respect to axis acceleration and jerk limits as follows.

Firstly, the algorithm assumes that a constant acceleration phase exists during cornering motion. This implies that the acceleration and jerk limits of X axis are fully exploited by setting $A_x = A_{max}$, $J_{1x} = J_{max} = -J_{3x}$, and the duration of constant jerk phase, T_1 and T_3 are computed by the trapezoidal nature of JLAP as:

$$T_1 = T_3 = \frac{A_{\max}}{J_{\max}} \quad . \tag{2.27}$$

Duration of the constant acceleration phase, T_2 is then obtained from Eq. (2.20):

$$T_2 = \frac{\Delta V_x}{A_{\max}} - \frac{A_{\max}}{J_{\max}} \quad , \tag{2.28}$$

and the maximum cornering velocity, Vc is solved from displacement boundary condition given in Eq. (2.26) as:

$$V_c = \frac{\sqrt{8A_{\max}\varepsilon\left|\sin\left(\theta_1 + \frac{\theta_2}{2}\right)\right| - \frac{A_{\max}{}^4}{3J_{\max}{}^2}}}{\left|\cos\ \theta_1 + \theta_2\ - \cos\ \theta_1\ \right|} \quad . \tag{2.29}$$

On the other hand, if the velocity transition is small and acceleration capacity of the drives is large, T_2 computed from Eq. (2.28) may become negative $T_2 < 0$. In this case, constant acceleration phase becomes unnecessary, and it is eliminated by setting $T_2 = 0$. Eq. (2.26) is used to re-calculate maximum axis acceleration as

$$A_x = \sqrt[3]{6J_{\max}{}^2\varepsilon\left|\cos(\theta_\varepsilon)\right|} \quad , \tag{2.30}$$

and the durations of constant jerk section, T_1 and T_3 are updated to:

$$T_1 = T_3 = \frac{A_x}{J_{\max}} \quad . \tag{2.31}$$

As a result, the fastest cornering velocity can be obtained from Eq. (26) as:

$$V_c = \frac{\sqrt[3]{36J_{\max}\varepsilon^2\sin^2\left(\theta_1 + \frac{\theta_2}{2}\right)}}{\left|\cos\ \theta_1 + \theta_2\ - \cos\ \theta_1\ \right|} \quad . \tag{2.32}$$

For some applications where feedrate is slow and the corner geometry is severely obtuse, cornering velocity V_c computed from Eq. (2.29) or (2.32) may exceed the commanded feedrate. In this case, the cornering velocity is set to the programmed feedrate of the linear segment. Cornering acceleration and jerk profiles are solved from Eq. (2.26) to satisfy the desired cornering tolerance. Please note that since cornering velocity is

lowered, maximum acceleration and jerk limits of the drives may not be saturated leading to a near time-optimal cornering motion.

The total cornering duration is calculated by sum of all durations of the JLAP:

$$T_c = T_1 + T_2 + T_3 \tag{2.33}$$

On the other hand, if Y-axis experiences the largest velocity transition, then maximum cornering speed, V_c is computed by replacing cosine terms with sine in Eqs. (2.29) and (2.32). Please note that the JLAP for less demanding, so-called the “trailing”, axis is planned with identical segment durations computed from Eqs. (2.27), (2.28) and (2.31). As a result, the trailing axis is not driven at its kinematic limits, but overall cornering motion is synchronized.

2.2.3. Kinematic corner smoothing (KCS) with uninterrupted acceleration

Previous section presented the kinematic corner smoothing (KCS) approach based on blending constant axis velocities at the entry and exit of a cornering motion. In that algorithm tangential velocity is reduced to the fastest cornering V_c, and cornering motion starts and ends with zero initial acceleration (See Figure 2.2b). In an effort to further reduce overall cornering duration, this section extends the approach presented in Section 2.2.2 by introducing cornering acceleration boundary conditions, A_c. The objective is to realize an uninterrupted tool motion as it is decelerated from segment's programmed feedrate to the cornering velocity and accelerated back to the next segment's feed without interrupting acceleration. Thus, the KCS with uninterrupted acceleration method presented in this section imposes non-zero cornering boundary acceleration conditions to further reduce overall cornering cycle time.

Figure 2.4 presents the approach to blend both axis velocity and accelerations in an uninterrupted manner. Instead of employing all 3 phases of the JLAP as proposed in Section 2.2.2, only the acceleration ramp phase, i.e. phase 1, is employed to smoothly interpolate axis velocity and accelerations from start to the end of the corner. The cornering motion kinematics can be written for X and Y axes as:

$$\underbrace{\left\{\begin{array}{l} j_x(\tau)=J_x \\ a_x(\tau)=A_{sx}+J_x\tau \\ v_x(\tau)=V_{sx}+A_x\tau+\frac{1}{2}J_x\tau^2 \\ s_x(\tau)=V_{sx}t+\frac{1}{2}A_x\tau^2+\frac{1}{6}J_x\tau^3 \end{array}\right\}}_{X-axis} \text{ and } \underbrace{\left\{\begin{array}{l} j_y(\tau)=J_{1y} \\ a_y(\tau)=A_{sy}+J_y\tau \\ v_y(\tau)=V_{sy}+A_y\tau+\frac{1}{2}J_y\tau^2 \\ s_y(\tau)=V_{sy}\tau+\frac{1}{2}A_y\tau^2+\frac{1}{6}J_y\tau^3 \end{array}\right\}}_{Y-axis} 0\le t\le t_1 \ . \quad (2.34)$$

Firstly, identical tangential acceleration magnitudes at the start and end of the cornering motion A_c = -A_s = A_e, are imposed to generate a symmetric cornering trajectory,

$$\underbrace{\left\{\begin{array}{l} A_{sx}=-A_c\cos(\theta_1) \\ A_{sy}=-A_c\sin(\theta_1) \end{array}\right\}}_{\text{Starting Acceleration Constraints}}, \quad \underbrace{\left\{\begin{array}{l} A_{ex}=A_c\cos(\theta_1+\theta_2)=A_{sx}+J_xT_1 \\ A_{ey}=A_c\sin(\theta_1+\theta_2)=A_{sy}+J_yT_1 \end{array}\right\}}_{\text{Ending Acceleration Constraints}} \ . \quad (2.35)$$

Similarly, velocity boundary conditions are imposed as:

$$\underbrace{\left\{\begin{array}{l} V_{sx}=V_c\cos(\theta_1) \\ V_{sy}=V_c\sin(\theta_1) \end{array}\right\}}_{\text{Starting Velocity Constraints}}, \quad \underbrace{\left\{\begin{array}{l} V_{ex}=V_c\cos(\theta_1+\theta_2)=V_{sx}+A_{sx}T_1+\tfrac{1}{2}J_xT_1^2 \\ V_{ey}=V_c\cos(\theta_1+\theta_2)=V_{sy}+A_{sy}T_1+\tfrac{1}{2}J_yT_1^2 \end{array}\right\}}_{\text{Ending Acceleration Constraints}} \ . \quad (2.36)$$

Lastly, displacement boundary conditions are introduced. Planning cornering motion with identical start and ending tangential velocity and accelerations ensures that the trajectory is symmetric around the bisector of the corner, hence maximum cornering error occurs at the mid-point of the trajectory. The mid-point position can be computed from Eq. (2.34) as

$$\left.\begin{array}{l} x_{mid}=V_{sx}\frac{T_1}{2}+\frac{1}{2}A_{sx}\left(\frac{T_1}{2}\right)^2+\frac{1}{6}J_x\left(\frac{T_1}{2}\right)^3 \\ y_{mid}=V_{sy}\frac{T_1}{2}+\frac{1}{2}A_{sy}\left(\frac{T_1}{2}\right)^2+\frac{1}{6}J_y\left(\frac{T_1}{2}\right)^3 \end{array}\right\}, \quad (2.37)$$

and location of the sharp corner point is computed from cornering geometry (See Figure 2.4a) as:

$$P_c=\left\{\begin{array}{l} x_c=L_c\cos\ \theta_1 \\ y_c=L_c\sin\ \theta_1 \end{array}\right. \quad (2.38)$$

where L_c is calculated based on the distance x-axis traveled from Eq. (2.34):

$$L_c = \left| \frac{V_{sx} \cdot T_1 + \frac{1}{2} A_{sx} T_1^2 + \frac{1}{6} J_x T_1^3}{\cos\ \theta_1\ + \cos\ \theta_1 + \theta_2} \right| \tag{2.39}$$

The displacement boundary condition for cornering trajectory is imposed to control cornering tolerance from Eqs. (2.38) and (2.37). For the X-axis it can be written as:

$$\begin{aligned} \varepsilon_x &= x_{mid} - x_c = \varepsilon \cos(\theta_\varepsilon) \\ &= \frac{\left\{ \begin{array}{l} V_{sx} T_1 \frac{1}{2}\ \cos\ \theta_1 + \theta_2\ - \cos\ \theta_1\ + \frac{1}{2} A_{sx} T_1^2 \left(\frac{1}{4} \cos\ \theta_1 + \theta_2\ - \frac{3}{4} \cos\ \theta_1\ \right) \\ + \frac{1}{6} J_x T_1^3 \left(\frac{1}{8} \cos\ \theta_1 + \theta_2\ - \frac{7}{8} \cos\ \theta_1\ \right) \end{array} \right\}}{\cos\ \theta_1\ + \cos\ \theta_1 + \theta_2} \end{aligned} \tag{2.40}$$

Please note that Y-axis component is simply obtained by replacing cosine terms with sine and axis acceleration and jerk amplitudes.

Similar to Section 2.2.2, maximum cornering velocity, V_c is sought to generate the fastest cornering speed, which tries to saturate at least one of the axis' acceleration or jerk limits. The limiting axis is identified as the axis with the largest acceleration transition $\Delta A_x = |A_{ex} - A_{sx}|$ or $\Delta A_y = |A_{ey} - A_{sy}|$. For instance, identifying X-axis as the limiting axis, $\Delta A_x > \Delta A_y$, acceleration, velocity and position constraints for cornering motion are written from Eqs. (2.35), (2.36) and (2.40) as:

$$\left. \begin{array}{l} \text{Acc. Const.}: A_c \cos\ \theta_1 + \theta_2\ = -A_c \cos\ \theta_1\ + J_x T_1 \\ \text{Vel. Const.}: V_c \cos\ \theta_1 + \theta_2\ = V_c \cos\ \theta_1\ - A_c \cos\ \theta_1\ T_1 + \frac{1}{2} J_x T_1^2 \\ \text{Disp. Const.}: \varepsilon \cos\left(\frac{\pi}{2} + \theta_1 + \frac{\theta_2}{2} \right) = V_c \cos\ \theta_1\ \left(\frac{T_1}{2} \right) - \frac{1}{2} A_c \cos\ \theta_1\ \left(\frac{T_1}{2} \right)^2 + \frac{1}{6} J_x \left(\frac{T_1}{2} \right)^3 \\ \qquad - \left(\frac{V_c \cos\ \theta_1\ T_1 - \frac{1}{2} A_c \cos\ \theta_1\ T_1^2 + \frac{1}{6} J_x T_1^3}{\cos\ \theta_1\ + \cos\ \theta_1 + \theta_2} \right) \cos\ \theta_1 \end{array} \right\}. \tag{2.41}$$

Feasible cornering velocity trajectory is then sought by saturating either one of the kinematic limits. For instance, setting cornering jerk, $J_x = J_{max}$ allows computation of the unknown cornering velocity, V_c as

$$V_{c,J\max} = 2\sqrt[3]{\left|\frac{9\varepsilon^2 J_{\max}}{\sin^2\left(\frac{\theta_2}{2}\right)\cos\theta_1 + \cos\theta_1 + \theta_2}\right|} \quad . \tag{2.42}$$

Similarly, the fastest cornering velocity that saturates acceleration limit of the axis is computed by setting $A_x = A_{max}$:

$$V_{c,A\max} = \sqrt{\left|\frac{6A_{\max}\varepsilon}{\sin\left(\frac{\theta_2}{2}\right)}\right|} \quad . \tag{2.43}$$

In order to satisfy both kinematic limits, V_c is selected from Eqs. (2.42) and (2.43) as:

$$V_c = \min\ V_{c,J\max}, V_{c,A\max} \quad . \tag{2.44}$$

The cornering acceleration is then computed from Eq. (2.43). Finally, duration of the cornering motion is solved from Eq. (2.41):

$$T_1 = \frac{12\varepsilon}{V_c \sin\left(\frac{\theta_2}{2}\right)} \quad . \tag{2.45}$$

Based on the cornering velocity and acceleration, fastest cornering motion duration is computed, which saturates kinematic limits one of the drives. The “trailing” axis motion is planned for identical cornering duration T_1 to ensure that the motion is synchronized. The trailing axis’ acceleration and jerk amplitudes, J_y and A_y are computed by re-writing Eq. (2.41), accordingly.

Please note that in case if the cornering velocity computed from Eq. (2.44) exceeds programmed feedrate of the linear block, it is lowered and set to the linear segment’s feedrate. The acceleration and jerk amplitudes are computed from Eq. (2.41), and the motion is re-planned.

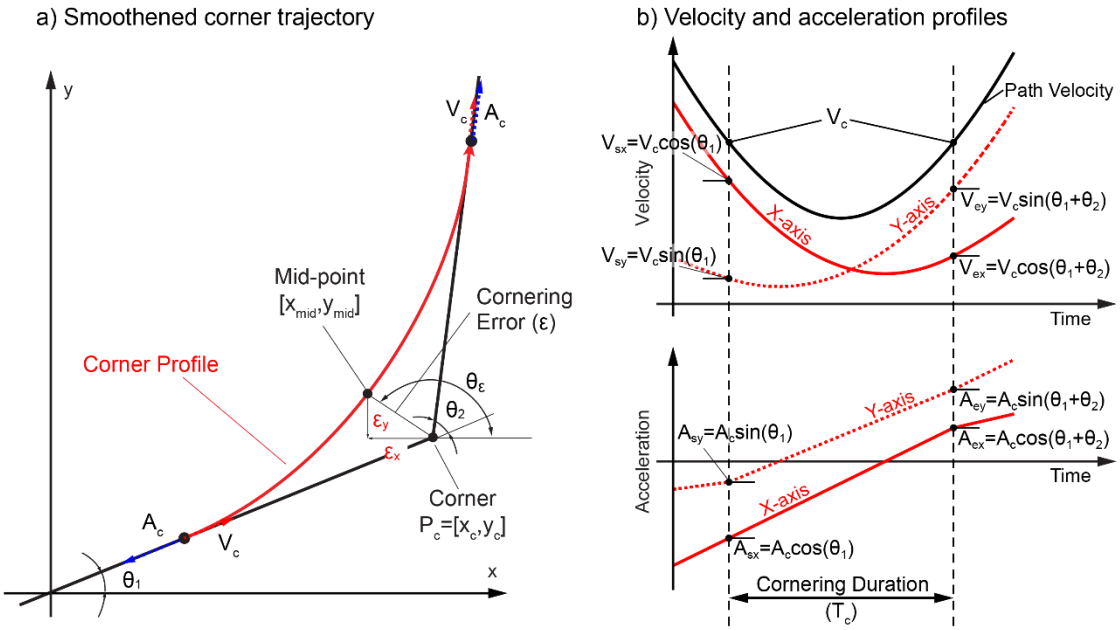

Figure 2.4: Kinematic corner smoothing with uninterrupted acceleration.

2.3. Illustrative example and experimental validation

This section evaluates performance of the proposed kinematic corner blending techniques on various high-speed cornering case scenarios. Experimental results and benchmarks to widely used geometric corner smoothing technique are also presented to validate effectiveness of the proposed cornering smoothing method.

2.3.1. Illustrative example

Firstly, Figure 2.5 and Figure 2.6 show application of the proposed corner smoothing techniques on a right-handed sharp corner, i.e. $\theta_1 = 0°$, and $\theta_2 = 90°$ with two different,10 [μm] and 100 [μm], cornering tolerances. The acceleration and jerk limitations of the drives (X and Y axes) are set to $A_{max} = 2.5\times10^3$ [mm/sec^2] and Jmax = 2×10^5 [mm/sec^3]. Figure 2.2 illustrated the kinematic corner smoothing (KCS) with interrupted acceleration presented in Section 2.2.2. Figure 2.5a shows smoothened cornering geometry with $\varepsilon = 100$ [μm] and $\varepsilon = 10$ [μm] cornering tolerances. Figure 2.5b shows generated axis

and path velocity profiles. As shown, when approached to the corner, tangential velocity is reduced from the programmed feedrate of 100 [mm/sec] to the specified cornering velocity. Depending on the cornering tolerance, the fastest cornering velocity is computed from Eqs. (2.29) or (2.32). For large contouring error, ε = 100 [μm], the resultant cornering velocity is V_c = 32.9 [mm/sec], and total motion time to finish the path is T_Σ = 0.296 [sec]. In contrast, tighter corner tolerance dictates slower cornering speed. When ε = 10 [μm] cornering velocity is calculated as V_c = 7.07 [mm/sec], and resulting cycle time is T_Σ = 0.310 [sec]. Notice from axis motion profiles, when approached to a right hand corner, X-axis simply decelerates to a full stop and Y-axis starts accelerating. The timing of this deceleration/acceleration transition determines the cornering tolerance. Both drives use their full acceleration and/or jerk limits. Particularly, if cornering error is large, acceleration limits of the drives can be saturated. This delivers a faster cornering motion. When cornering tolerance is small, the acceleration limits of the drives cannot be saturated within the allowed jerk bounds.

Figure 2.6, on the other hand, presents results of the kinematic corner smoothing (KCS) algorithm with uninterrupted acceleration presented in Section 2.2.3. As observed from Figure 2.6b, cornering motion has non-zero acceleration at the start and end. Maximum cornering velocities are computed to be V_c = 45.9 and V_c = 14.1, for ε = 100 [μm] and ε = 10 [μm], respectively. The resultant total cycle times are T_Σ = 0.290 [sec] and T_Σ = 0.293 [sec]. Although, KCS with uninterrupted acceleration requires more time during cornering motion, the total cycle time to travel the tool-path is slightly faster. As shown in Figure 2.6c–d, drives decelerate at maximum rate to the corner in an effort to reduce the cycle time. Therefore, the cornering entry and exit acceleration are saturated, i.e. $A_c = A_{max}$ = 2500 [mm/sec^2]. As shown in Figure 2.6d, if cornering tolerance is small, this requires maximum jerk to be utilized to finish the cornering trajectory. However, if the cornering tolerance is large, the motion does not need to utilize full jerk capability to alter its acceleration and velocity. Thus, setting cornering error to ε = 100 [μm] (See Figure 2.6d) only saturates acceleration limits of the drives but does not fully exploit jerk limits, which makes the KCS with uninterrupted acceleration near-time optimal. Figure 2.9 shows a

cycle time comparison between KCS algorithms with interrupted and uninterrupted acceleration profiles.

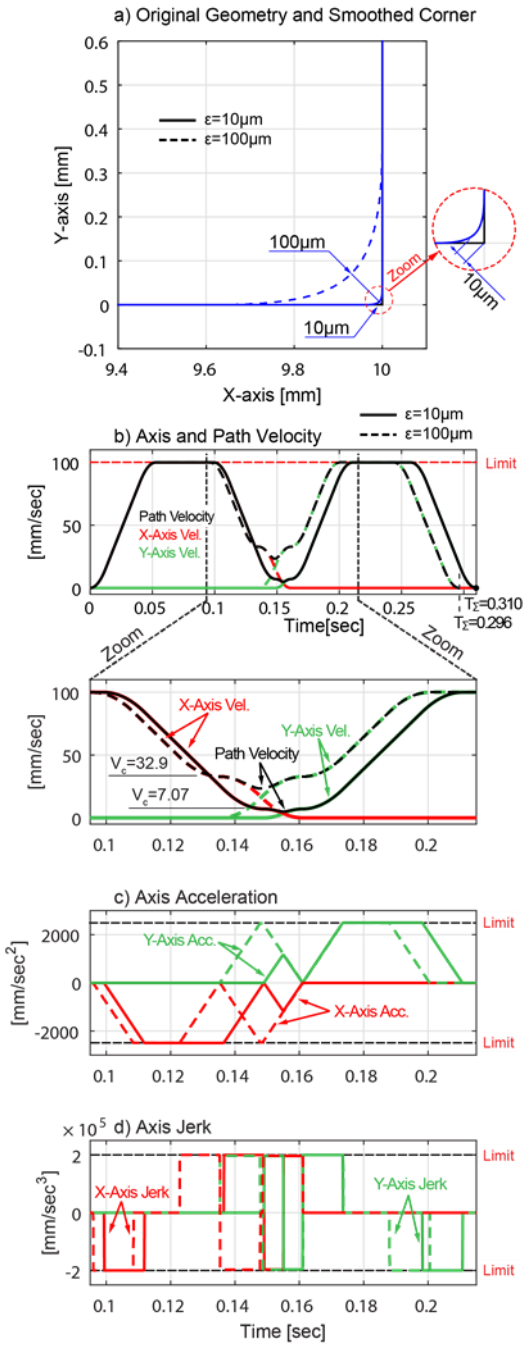

Figure 2.5: Right-handed Sharp Corner Smoothing using KCS method with Interrupted acceleration.

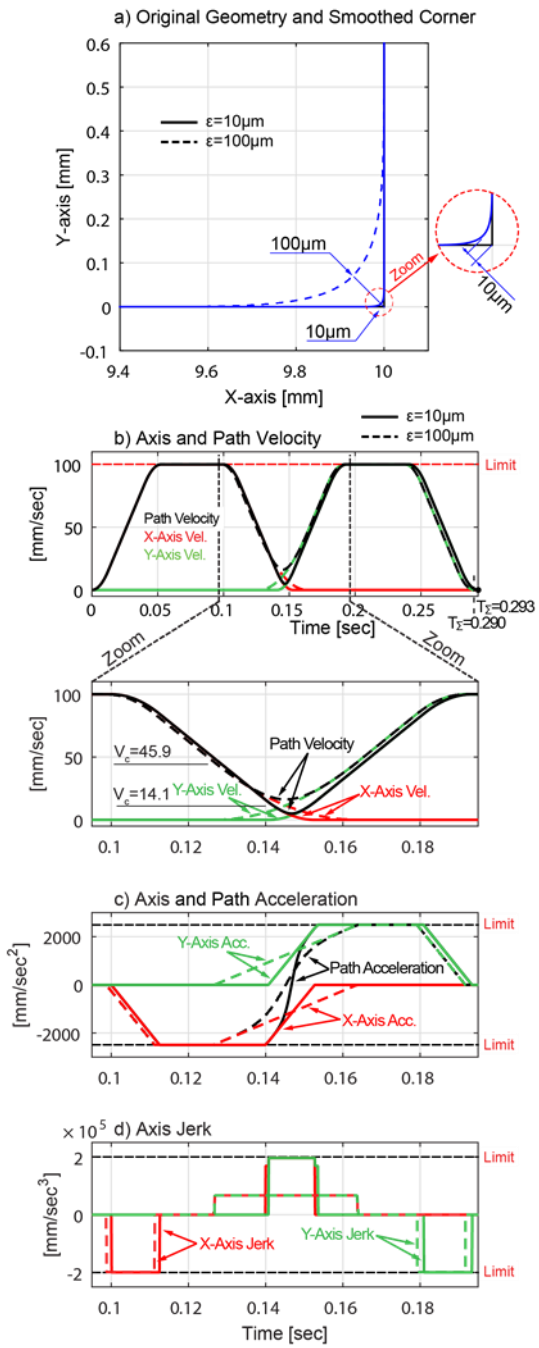

Figure 2.6: Right-handed Sharp Corner Smoothing using KCS method with Uninterrupted acceleration.

Figure 2.7 and Figure 2.8 show kinematic corner smoothing applied to obtuse and acute corners. In both cases cornering tolerance is set to ε = 20 [μm]. Proposed algorithms with interrupted and uninterrupted accelerations can smoothen corners within given cornering

tolerance. In case of the acute corner, resultant cornering velocities are much smaller. This is simply due to fact that X-axis must alter it motion direction, and has to undergo larger velocity traverse. In the obtuse case, Y-axis undergoes similar amount of velocity transition but does not change its motion direction. Notice that since total velocity traverses are similar, total cornering cycle times are actually comparable.

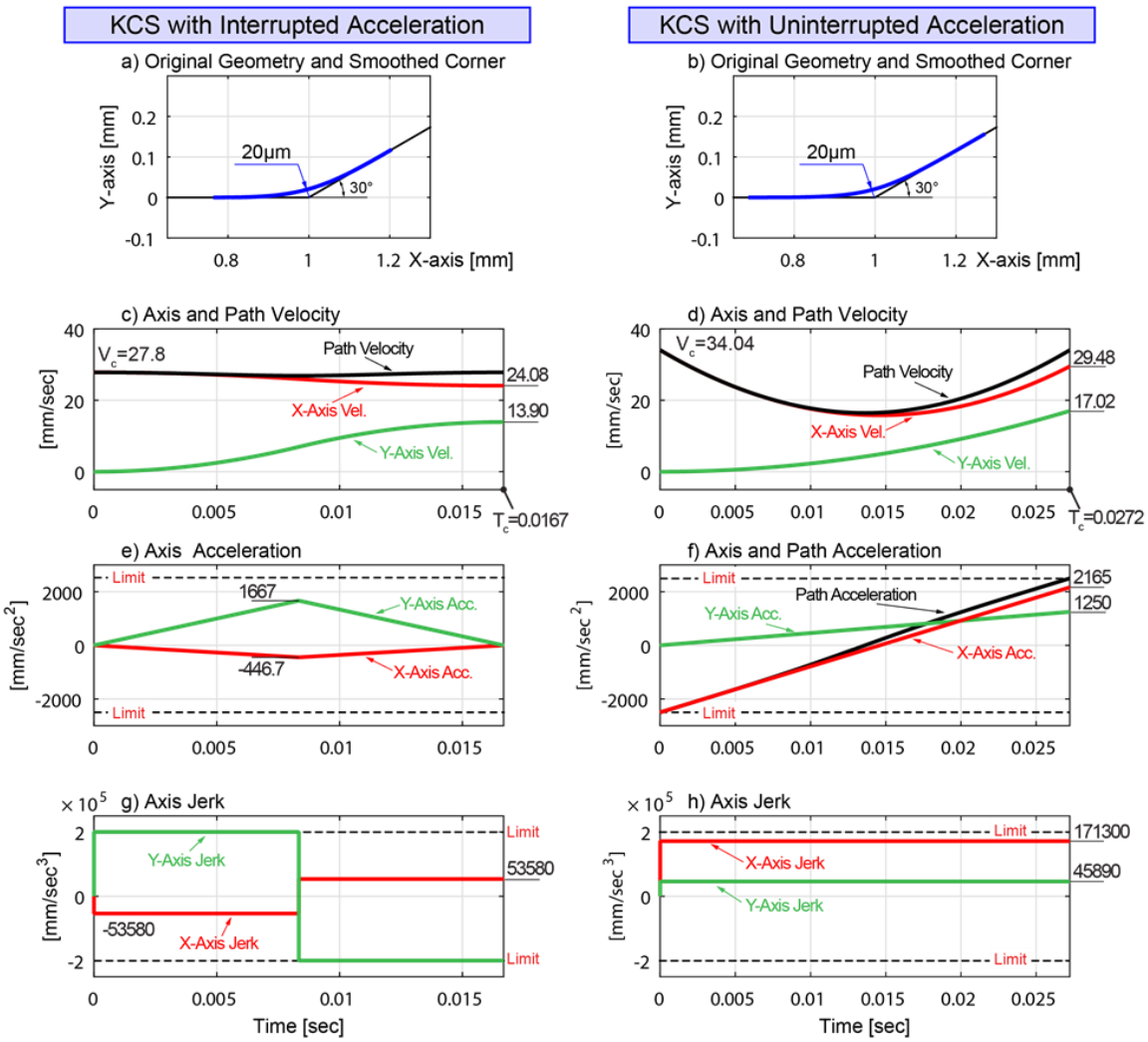

Figure 2.7: Obtuse corner smoothing using KCS with interrupted and uninterrupted acceleration profiles.

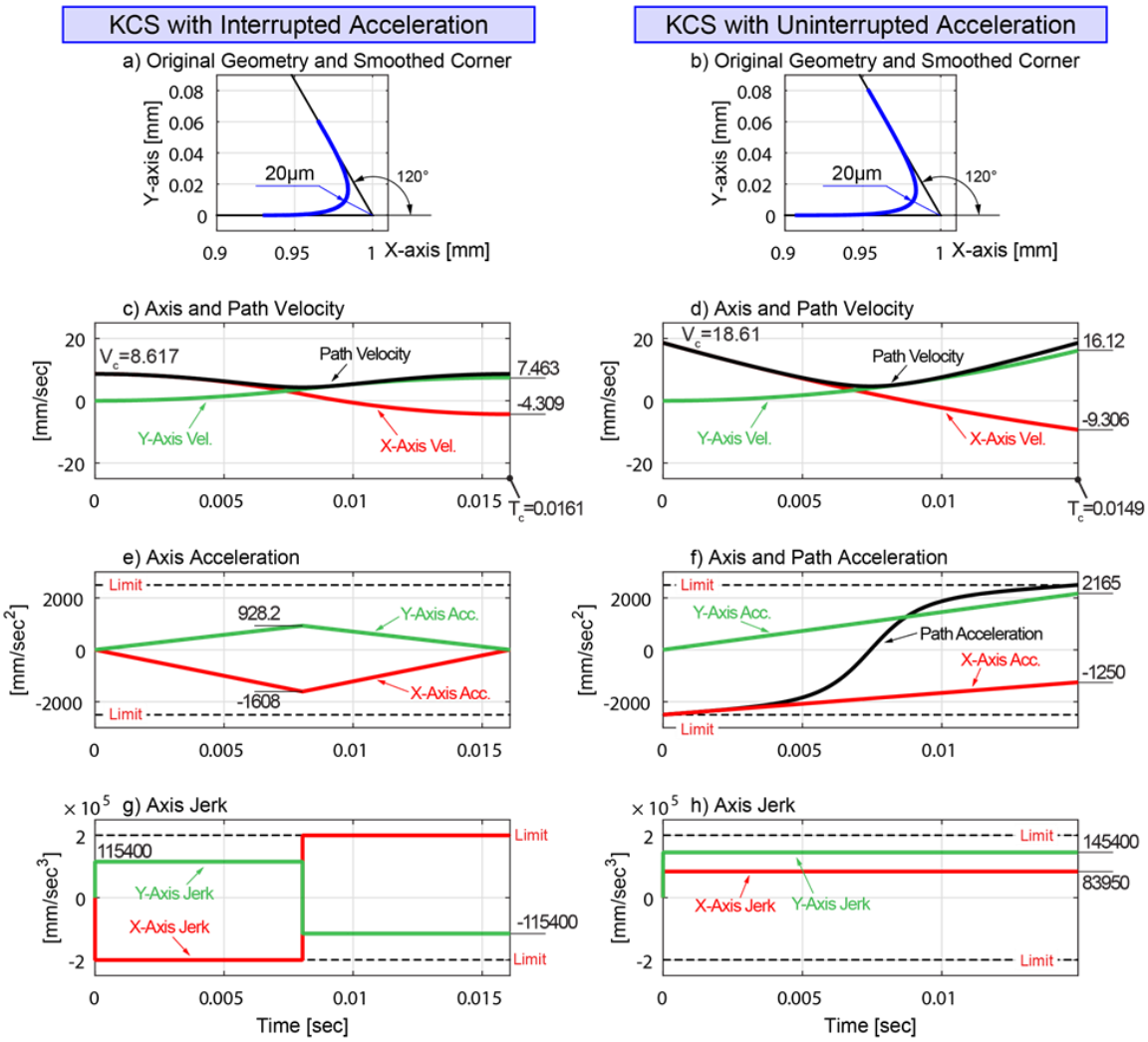

Figure 2.8: Acute corner smoothing based on KCS with interrupted and uninterrupted acceleration profiles.

Next, cycle time performance of the KCS with interrupted and uninterrupted acceleration techniques is compared in Figure 2.9. A single corner is smoothened by two L = 10 [mm] long linear segments. The desired feedrate along the tool-path is set to 100 [mm/sec]. As shown in Figure 2.9a, cornering angle is altered from acute to obtuse to compare the performance of KCS algorithms for different cornering geometries. Figure 2.9b shows total cycle time for different cornering tolerances. As noticed, for very obtuse corners, i.e. cornering angle $\theta_2 < 20$, the KCS algorithm with interrupted acceleration delivers faster cycle time. In contrast, as the corner gets acute the un-interrupted acceleration provides

faster cycle times. This can be attributed to the fact that as the corner gets sharper, cornering velocity becomes smaller. In this case, KCS with uninterrupted acceleration can plan efficient acceleration profiles and minimize the cycle time. Combination of the KCS algorithms with and without uninterrupted acceleration should be used to attain the fastest cycle time.

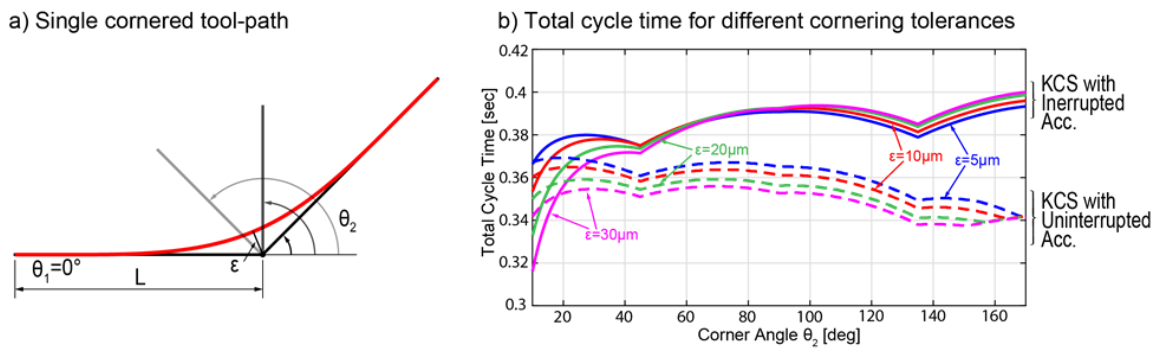

Figure 2.9: Cycle time performance of KCS with interrupted and uninterrupted acceleration profiles.

2.3.2. Experimental results

Lastly, experimental validation and benchmark comparisons are performed. The experimental Cartesian X–Y motion system is shown in Figure 2.10. The planar motion table is driven by 3 linear motors. The heavier X-axis is designed as gantry and carries the lighter Y-axis. In order to implement proposed algorithms servo amplifiers are set to operate in torque (current) control mode. Closed loop control is implemented in the Dspace DS1103® real time control system by reading linear encoder feedback at a resolution of 0.7125 [μm] and commanding torque signal to the servos at a closed loop sampling interval of T_s = 0.1 [msec]. Both X and Y drives are controlled by P–PI cascade [37] motion controllers with velocity feed-forward action. The position feedback control bandwidths of the axes are roughly matched at ω_n = 35 [Hz] to ensure good motion synchronization and path tracking.

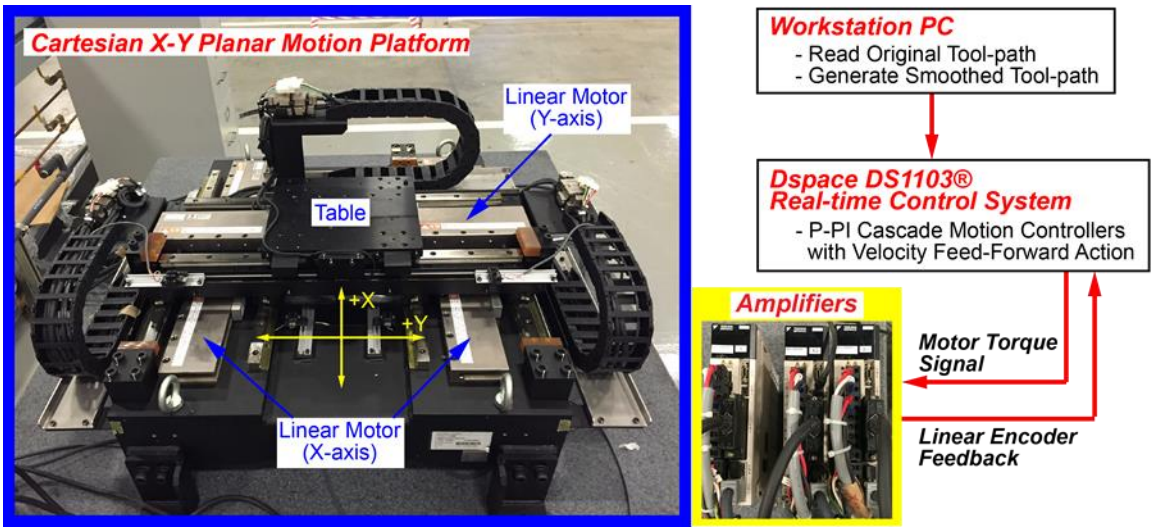

Figure 2.10: Experimental setup.

3 algorithms are implemented and compared to each other on smoothing the tool-path shown in Figure 2.11. Proposed KCS algorithms with interrupted and uninterrupted accelerations are implemented separately. They are compared against a geometric corner smoothing algorithm by Sencer et.al. [8], which fits curvature optimal Beziers around sharp corners and plans jerk limited feedrate profile for minimum cycle time. This method is called as the “Bezier” method. All the algorithms are computed off-line, sampled and commanded in real-time to the motion controllers. Reference motion commands are discretized by rounding the motion durations so that number of samples is integer while keeping the total displacement unchanged.

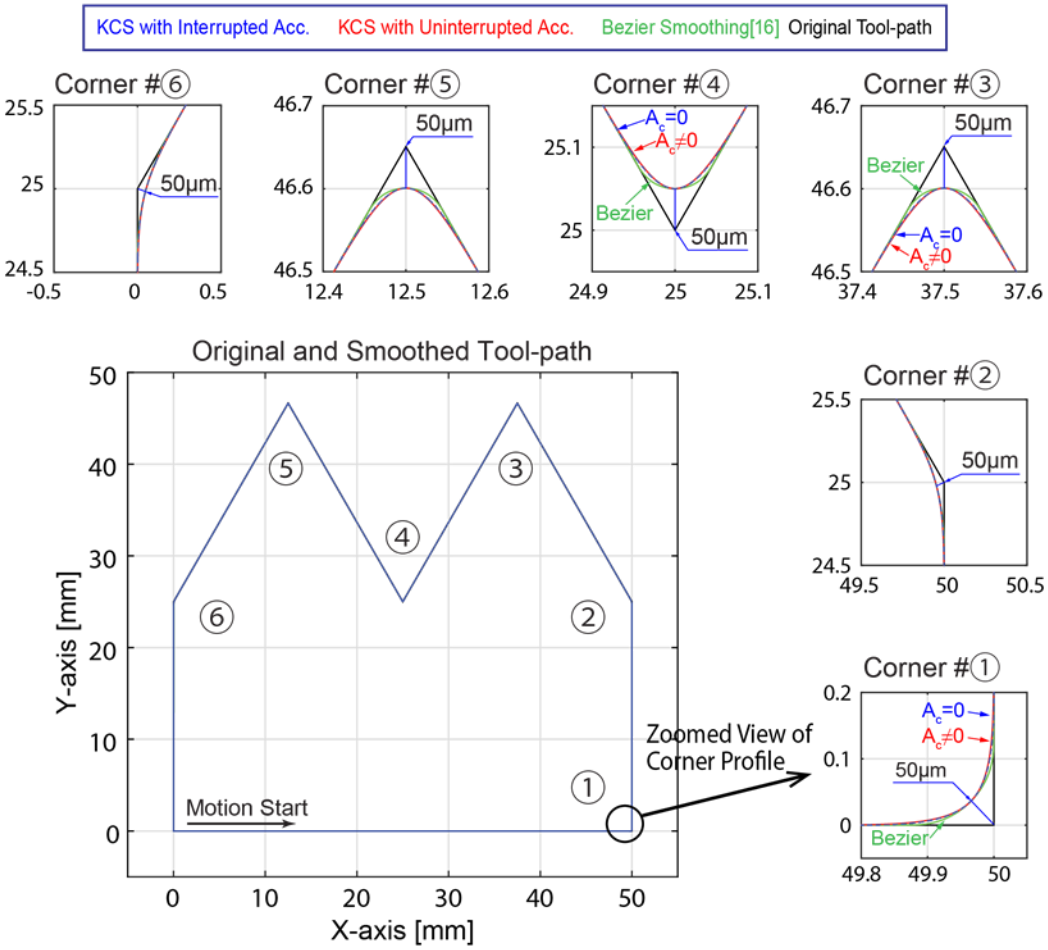

Figure 2.11: Experimental multi-segmented tool-path.

Figure 2.11 shows smoothened tool-path clearly. The cornering error is set to ε = 50 [micron] for all the corners, and all the algorithms successfully smooth corners within given tolerance. Figure 2.12 depicts motion profiles along the tool-path. The feedrate is set to f = 100 [mm/sec], axis acceleration and jerk limits are set to $A_{max} = 2\times10^3$, $J_{max} = 1\times10^5$. Figure 2.12a shows tangential velocity profiles. As shown, proposed KCS technique with uninterrupted acceleration achieves the fastest cycle time amongst all the other methods. This is mainly due to the fact that most corners are acute. Acceleration and jerk profiles for all the methods are compared in Figure 2.12d–g. As shown, all the methods respect acceleration limits of the drives. The Bezier corner-smoothing method fits a curvature optimal Bezier around the corner and selects the fastest cornering speed with respect to acceleration limits of the drives. Therefore, it is able to saturate acceleration limits of drives,

but cannot respect jerk limits. As a matter of fact, if the algorithm is modified to utilize jerk bounds cornering velocity must be reduced greatly. In contrary, the proposed KCS algorithms clearly respect jerk limits of the drives (See Figure 2.12f–g). As will be observed in the contouring errors, this functionality allows generation of a traceable smoother motion.

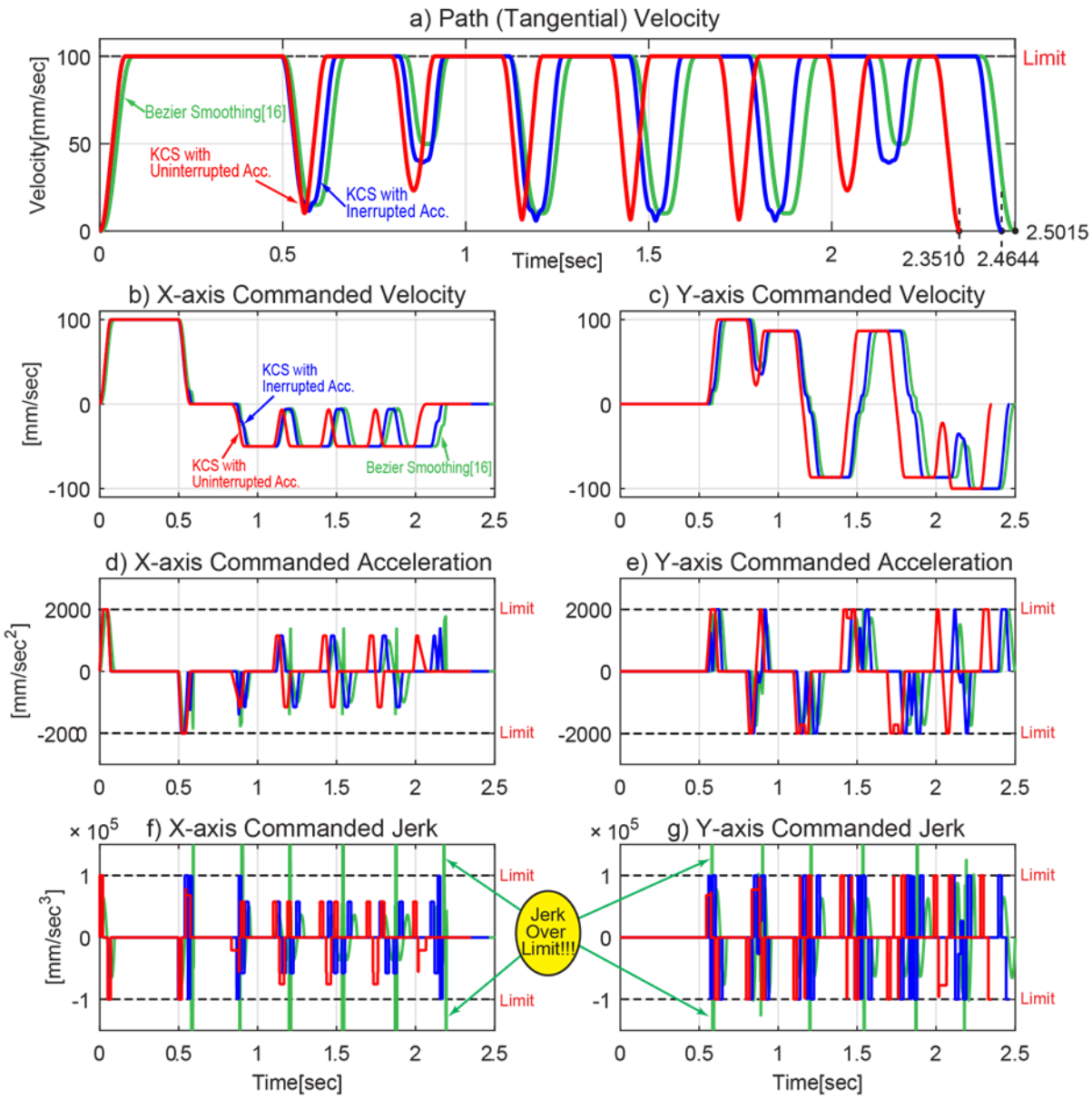

Figure 2.12: Kinematic profiles along corner smoothened tool-path.

It is known that jerk content of the trajectory affects tracking errors and vibratory behavior of feed drive system [36]. Figure 2.13 shows experimentally measured contouring errors along the corner smoothened tool-path. Since the X and Y-axis closed loop bandwidths are matched, contour errors along linear segments are negligible. Largest contouring errors occur at the cornering sections where static friction impacts drives, and acceleration and jerk profiles show large changes when drives alter their motion direction. Table 2.1 summarizes contouring performance of cornering algorithms and overall contour performances. Proposed KCS algorithms and Bezier corner smoothing method deliver similar overall contouring performances. To be exact, proposed KCS methods deliver slightly better RMS and maximum contour errors. The KCS with interrupted acceleration can provide ~20% reduction in maximum contour errors while staying within acceleration and jerk limits of the drives and still deliver faster cycle times. Furthermore, if error profiles are inspected closely Bezier method [8] shows severe error fluctuations around the cornering durations. This is simply due to extremely high jerk amplitude commanded to the drives. Since jerk is not limited to a suitable level, large jerk spikes excite the feedback control system and induce vibrations. These vibrations are visible on the actual trajectory. As noted from Figure 2.13, resultant tool-path fluctuates severally around the cornering sections. These fluctuations will be imprinted on the part surface during an actual manufacturing operation and destroy process tolerances. Proposed technique can limit the jerk and provide a smoother motion with faster cycle time. As noted from Table 2.1, proposed KCS method with uninterrupted acceleration can reduce cycle time around 6–7% for this simple tool-path. For a longer tool-path, which consists large number of corners the effect would be much more pronounced.

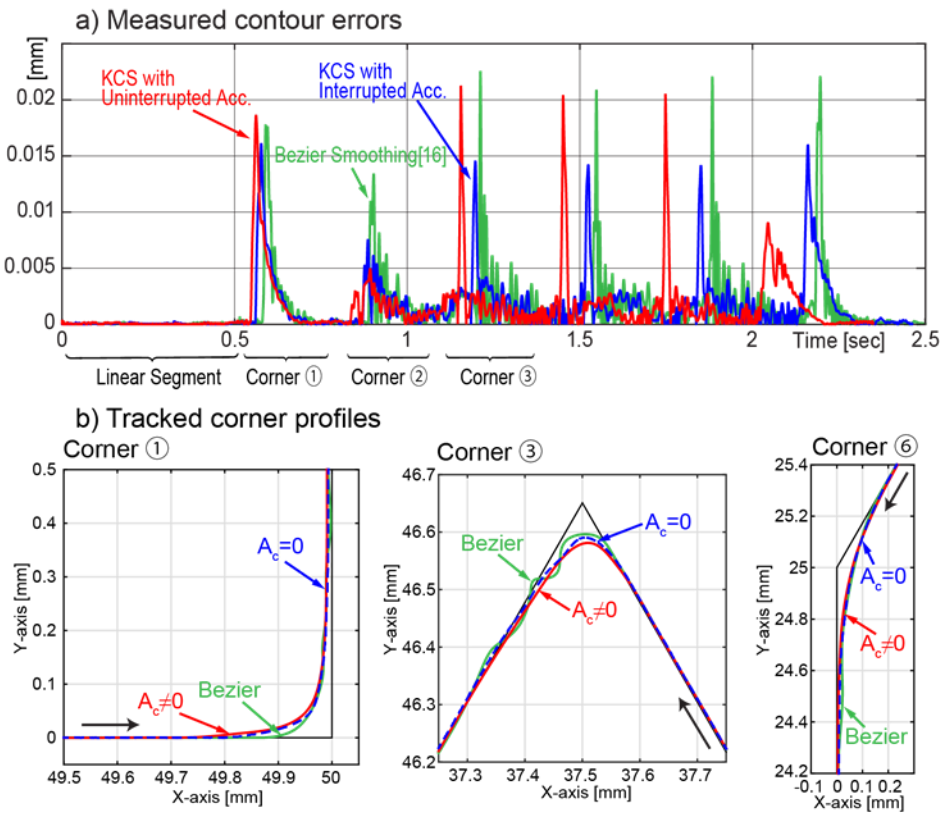

Figure 2.13: Experimentally recorded contouring performance.

Table 2.1: Cycle time and contouring performance comparison.

Algorithms	Cycle Time [sec]	Contour Error	
		RMS [μm]	Max [μm]
KCS with Uninterrupted Acc. (Proposed)	2.3510	3.0637	21.2576
KCS with Interrupted Acc. (Proposed)	2.4644	2.8062	16.1245
Bezier Smoothing ([16])	2.5015	3.1517	22.5685

2.4. Conclusions

This paper proposes novel kinematic corner smoothing (KCS) techniques, which eliminate the need for two-step geometry based corner rounding methods. The proposed algorithms blend axis velocities around sharp corners with jerk limited acceleration transitions and generate symmetric rounded corner profiles with precisely controlled geometric tolerances. The cornering duration is calculated based on the cornering tolerance, axis kinematic limits and sharp corner profile to minimize overall cycle time. Proposed algorithm is fully analytical and provides fast and efficient real-time implementation on 2 to 3 axis Cartesian CNC machine tools. Extensive illustrative examples along obtuse and acute corner profiles validate accuracy and performance of the proposed algorithms. Experimental benchmarks against spline based corner smoothing technique show that proposed algorithms provide better contouring performance while reducing overall cycle time 6–7% for a tool-path with six corners. Considering that longer tool-paths, such as the ones used in high speed die and mold manufacturing, may contain hundred and thousands of sharp corners, proposed techniques provide significant potential to reduce overall cycle times.

Global tool-path smoothing for CNC machine tools with uninterrupted acceleration machine tools

Shingo Tajima and Burak Sencer

International Journal of Machine Tools & Manufacture

Volume 121, October 2017, Pages 81-95

3. Global tool-path smoothing for CNC machine tools with uninterrupted acceleration

Majority of tool-paths for high-speed machining is composed of series of short linear segments, so-called G01 moves. This discrete tool-path format limits the achievable speed and accuracy of CNC machines. To generate continuous feed motion along sharp cornered tool-paths, most NC systems smooth corners locally using a prespecified curve or a spline and slow down to be able to change the feed direction within machine kinematic limits. Path speed is dramatically reduced for accuracy if sharp corners are within close vicinity. This paper proposes a new real-time interpolation algorithm for NC systems to generate continuous rapid feed motion along short segmented linear tool-paths by smoothing local and adjacent corners that are within close vicinity. Instead of locally modifying the corner geometry with a spline, the proposed algorithm directly blends axis velocities between consecutive linear segments based on the jerk limited acceleration profile (JLAP) and generates cornering trajectories within user-specified contour errors and kinematic limits of the drives. A novel Look-Ahead Windowing (LAW) technique is developed to plan tangential feed profile with uninterrupted acceleration to continuously smooth the path. The feed profile is optimized to generate rapid motion along overlapping adjacent corners. Simulation and experimental results demonstrate effectiveness of the proposed method to interpolate accurate Cartesian high-speed motion along short-segmented tool-paths for machining free-form surfaces found in dies, molds and aerospace parts.

3.1. Introduction

Parts for aerospace, and die-and-mould industries contain complex sculptured geometries that are designed on CAM systems using smooth parametric curves such as splines and NURBS [16]. It has been reported that direct interpolation of these curves on CNC (computer numerical controlled) machine tools can provide the smoothest, fastest and accurate motion, and thereby most suitable for high-speed machining [16], [19], [38]. However, deficiencies in the accurate calculation of curve lengths, planning of time-optimal feed profiles, and feed fluctuations hinder penetration of “direct spline

interpolation" in CNC machines [4], [18], [20], [22], [39]. Most CAM systems do not export parametric spline tool-paths [19]. Instead, they discretize original smooth geometry with a series of short line segments, and output a "polygonal" tool-path, which is to be interpolated using basic G01 commands [1]. Based on chord-length specifications, the point-to-points G01 moves range from 1 μm to 1 mm in length. As the curvature of the tool-path increases discretization becomes finer and denser generating NC part programs composed of thousands of lines. Although size of the NC (numerical control) part program is not the bottleneck, its definition is. Polygonal tool-paths are only position continuous, and they do not allow continuous interpolation of the feed motion at the junction points, i.e. corners of consecutive line segments. As a result, motion must stop momentarily at corners; otherwise, acceleration and jerk limits of the drives are violated, which can destroy the surface finish. Regardless, this stop-and-go motion planning leads elongated cycle times and generates rough, cornered surfaces with pronounced feed marks. This process is depicted in Figure 3.1a and b.

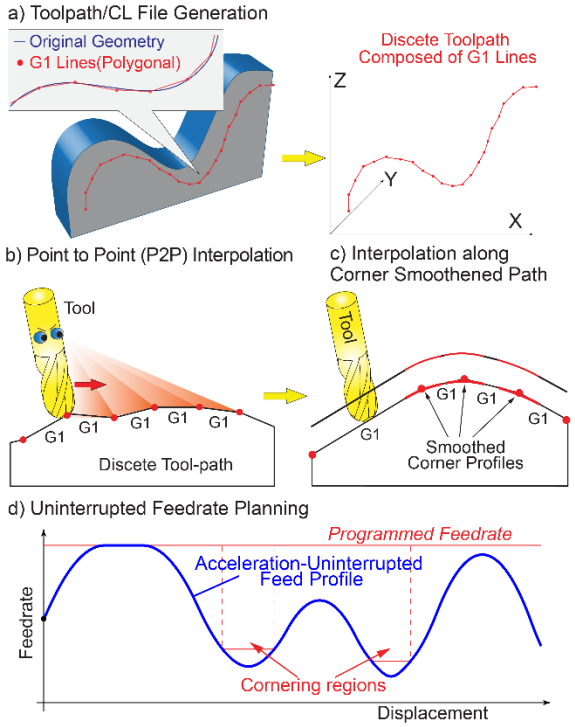

Figure 3.1: Interpolation of discrete tool-paths with corner smoothing.

The well-known approach to generate continuous non-stop motion along polygonal tool-path geometries is to blend consecutive line segments locally [5], [23], [24]. The idea is straightforward. In order to realize a continuous feed motion, the sharp corner geometry between two linear segments is replaced by a smooth blending curve in by the NC system of the machine tool as shown in Figure 3.1b. The final corner profile is no longer sharp, and the reference path deviates from the original corner geometry. As long as the deviation from original corner geometry is controllable, this approach is conveniently applicable to roughing and semi-finishing. Basic arcs, cubic, quadratic, quintic, B-splines or various curves are used to generate a continuous motion transition around the corner profile well documented in the literature [7], [8], [15], [25]–[27], [29], [40].

Geometric local smoothing is then followed by tangential feed planning to generate the final trajectory. Due to curved corner profile, feedrate must be lowered so that axis velocity and acceleration limits are not violated around corner blends [7], [8], [29], [40]. Jerk limited acceleration profile (JLAP) is known to be favorable to realize a smooth and rapid motion for high-speed machining [30]. JALP generates trapezoidal acceleration transitions with piecewise constant jerk segments, which helps avoiding excitement of inertial vibrations of feed drive system and provides a good balance between smoothness and time optimality w.r.t. machine's limits [18], [23], [30], [31]. Thus, most modern CNC machine tools adapt the JALP to plan feed motion and interpolate linear and spline tool-paths.

This 2-step, i.e. geometric path smoothing followed by feed planning based approach, is inefficient. Since the smoothened corner geometry is essentially a parametric curve, it suffers from bottlenecks related to real-time curve interpolation and causes severe feed fluctuations [4], [22]. In addition, planning of a time optimal feed profile along the smoothened path is computationally stringent [22], [24], [25], [32] and typically approximations are employed in the real-time implementation on NC systems [8], [23]. Hence, recent developments have been focusing on 1-step cornering trajectory generation methods. Command filtering [33], [34], the use of optimal control [27], or direct axis velocity pro file blending techniques [35], [41] have recently been proposed. Filtering techniques [33], [34] utilize Finite Impulse Response (FIR) filters [42] that introduce a

predetermined delay to the motion and do not incorporate full kinematic limits of the drives to minimize machining cycle time [43]. Furthermore, currently available filtering techniques are designed for local cornering and thus they are not capable of controlling cornering errors along densely segmented linear paths. Optimal control and velocity profile blending based corner smoothing techniques can generate rapid cornering techniques. But, they are either computationally stringent for real-time implementation or only capable of smoothing local corners that occur between long linear segments.

As mentioned, currently available techniques are mostly geared towards smoothing local corners. They assume that linear moves (G01 lines) are long enough so that a corner could be smoothened "locally" [7], [8], [15], [25]–[27], [29], [31], [33], [34], [40] without interfering with the consecutive corner blend. However, for realistic tool-paths used in the aerospace and die-and-mould industries this assumption is invalid. Typically, highspeed machining tool-paths are composed of linear moves that are shorter than <1mm. If the user selects large cornering error for high speed roughing or semi-finishing, locally smoothened corner geometries overlap each other. As a result, either cornering error tolerance needs to be reduced to eliminate any overlapping and so to plan feasible feed profiles [26], [29], [34], [40], or the motion is simply forced to undergo a full-stop, both of which greatly elongate overall cycle time and limit achievable productivity. Thus, a "global" corner smoothing technique is necessary to handle short-segmented tool-paths with overlapping corners and respect kinematic limits of the machine tool. Current machine tool literature has not addressed this problem thoroughly. The only available 1-step global corner smoothing solution [44] solves the trajectory generation problem with a bang-bang style acceleration profile that is not suitable for modern high-speed machine tools.

Thus, this paper proposes a new real-time interpolation algorithm for NC systems to generate continuous rapid feed motion along short segmented linear tool-paths by smoothing local and adjacent corners that are within close vicinity to generate a global jerk limited high speed motion trajectory. Firstly, the proposed algorithm introduces the 1-step direct corner smoothing technique by blending axis velocities and acceleration between consecutive linear segments based on the JLAP. A novel Look-Ahead Windowing (LAW) technique is then presented to plan tangential jerk limited feed profile with uninterrupted

acceleration to continuously smooth the path. A novel transition algorithm is developed to generate uninterrupted motion along overlapping adjacent corners. Simulation and experimental results demonstrate effectiveness of the proposed method to generate accurate Cartesian high-speed motion along short-segmented tool-paths.

3.2. One-step local corner smoothing with acceleration uninterrupted kinematics

As introduced in Section 3.1, 2-step corner smoothing techniques modify path geometry with parametric curves to enable acceleration continuous feed planning. This section presents the 1-step acceleration-uninterrupted kinematic corner smoothing (AU-KCS) method, which bases on the foundations of the kinematic cornering technique (KCS) presented in [42]. The AU-KCS strategy is depicted in Figure 3.2. As opposed to replacing the sharp corner with a smooth spline and planning the feed profile, the proposed method directly interpolates axis velocities and accelerations around the corner profile to realize a smooth cornering trajectory (Figure 3.2b). Boundary acceleration constraints are introduced so that an acceleration-uninterrupted motion can be planned in and out of the corner, and hence the total cornering cycle time can be reduced.

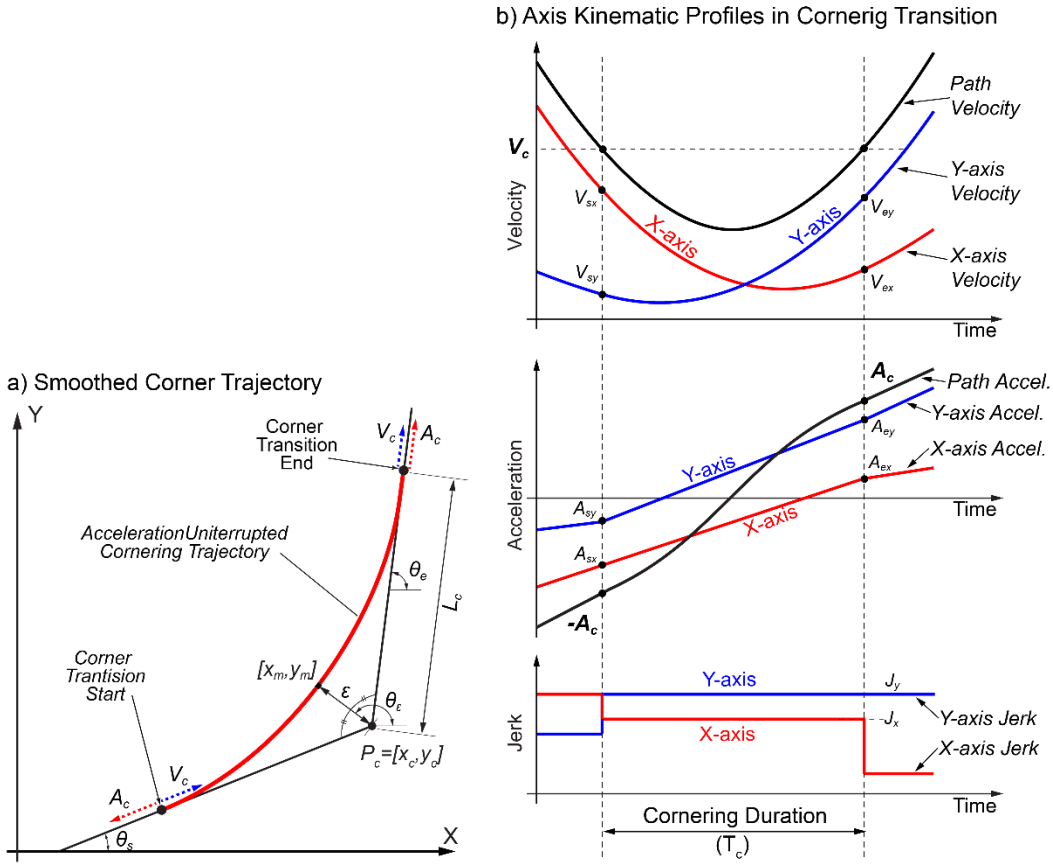

Figure 3.2: Acceleration uninterrupted kinematic corner smoothing (AU-KCS) strategy.

Firstly, we assume that cornering motion starts and ends with identical cornering velocity and accelerations, V_c and A_c. This allows us to generate corner profiles that are symmetric around the corner bisector as shown in Figure 3.2a. Note that, generating symmetric corners is critical for the surface quality. Conventional die and mould parts are composed of back-and-forth parallel rastering paths, and symmetric corners allow generation of smooth surfaces with lower average surface roughness. Kinematics of the axes during corner transition is dictated by the jerk limited acceleration profile (JLAP) [30] to generate a smooth and traceable motion. Based on the above conditions, the objective is to compute a cornering velocity V_c and acceleration A_c that yields the minimum cornering duration, T_c.

Typically, tool motion needs to be decelerated from path speed for cornering transition and accelerated back to the commanded feed of the consecutive linear segment

[8], [29]. As shown in Figure 3.2b, a piecewise constant jerk transition can ensure that tool motion decelerates to a cornering transition and accelerates back to the subsequent segment without interrupting its acceleration. In other words, the motion should decelerate in-and-out of the corner without crossing a zero acceleration so that overall cycle time can be reduced. For a planar case, X and Y axis position $s(t)$, velocity $v(t)$, acceleration $a(t)$, and jerk $j(t)$ profiles can be written from Figure 3.2b as:

$$\left\{\begin{array}{l} j_x(t) = J_x \\ a_x(t) = A_{sx} + J_x t \\ v_x(t) = V_{sx} + A_{sx} t + \frac{1}{2} J_x t^2 \\ s_x(t) = V_{sx} t + \frac{1}{2} A_{sx} t^2 + \frac{1}{6} J_x t^3 \end{array}\right\} \text{ and } \left\{\begin{array}{l} j_y(t) = J_y \\ a_y(t) = A_{sy} + J_y t \\ v_y(t) = V_{sy} + A_{sy} t + \frac{1}{2} J_y t^2 \\ s_y(t) = V_{sy} t + \frac{1}{2} A_{sy} t^2 + \frac{1}{6} J_y t^3 \end{array}\right\} \quad 0 \le t \le T_c \qquad (3.1)$$

where, t is relative cornering time, T_c is cornering duration, and J_x and J_y are jerk magnitudes. Axis velocity (V_s, V_e) and acceleration (A_s, A_e) boundary conditions at the start and end of cornering transition are computed from corner geometry and from Eq. (3.1) as:

$$\begin{Bmatrix} A_{sx} \\ A_{sy} \\ V_{sx} \\ V_{sy} \end{Bmatrix} = \begin{Bmatrix} -A_c \begin{bmatrix} \cos\ \theta_s \\ \sin\ \theta_s \end{bmatrix} \\ V_c \begin{bmatrix} \cos\ \theta_s \\ \sin\ \theta_s \end{bmatrix} \end{Bmatrix} \text{ and } \begin{Bmatrix} A_{ex} \\ A_{ey} \\ V_{ex} \\ V_{ey} \end{Bmatrix} = \begin{Bmatrix} A_c \begin{bmatrix} \cos\ \theta_e \\ \sin\ \theta_e \end{bmatrix} \\ V_c \begin{bmatrix} \cos\ \theta_e \\ \sin\ \theta_e \end{bmatrix} \end{Bmatrix} = \begin{Bmatrix} A_{sx} + J_x T_c \\ A_{sy} + J_y T_c \\ V_{sx} + A_{sx} T_c + ½ J_x T_c^2 \\ V_{sy} + A_{sy} T_c + ½ J_y T_c^2 \end{Bmatrix} \qquad (3.2)$$

where θ_s and θ_e are entry and exit angles to a corner, i.e. orientation of consecutive linear segments. The maximum cornering error ε must be controlled by the user, which introduces the displacement constraint. As shown in Figure 3.2a, cornering error occurs in the middle of the cornering trajectory at $t = T_c/2$ due to symmetry and expressed as:

$$\left.\begin{array}{l} \varepsilon_x = \varepsilon \cos(\theta_\varepsilon) = x_m - x_c \\ \varepsilon_y = \varepsilon \sin(\theta_\varepsilon) = y_m - y_c \end{array}\right\} \qquad (3.3)$$

where $\theta_\varepsilon = (\theta_s + \theta_e)/2 + \pi/2$ is obtained from geometry (See Figure 3.2a). The mid-point of cornering trajectory is evaluated then from Eq. (3.1),

$$\left.\begin{aligned} x_m &= V_{sx}\frac{T_c}{2}+\frac{1}{2}A_{sx}\left(\frac{T_c}{2}\right)^2+\frac{1}{6}J_x\left(\frac{T_c}{2}\right)^3 \\ y_m &= V_{sy}\frac{T_c}{2}+\frac{1}{2}A_{sy}\left(\frac{T_c}{2}\right)^2+\frac{1}{6}J_y\left(\frac{T_c}{2}\right)^3 \end{aligned}\right\} \tag{3.4}$$

and the programmed corner point $P_c = [x_c, y_c]$ location is expressed by the total distance traveled by the axes as:

$$P_c = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \underbrace{\left|\frac{V_{sx}T_c + \frac{1}{2}A_{sx}T_c^{\,2} + \frac{1}{6}J_xT_c^{\,3}}{\cos\ \theta_s\ + \cos\ \theta_e}\right|}_{L_c} \begin{bmatrix} \cos\ \theta_s \\ \sin\ \theta_s \end{bmatrix} \tag{3.5}$$

where L_c is the linear displacement used during the cornering transition (See Figure 3.2a). X-axis component of cornering error can be expressed from Eqs. (3.3)-(3.5) as:

$$\varepsilon_x = \frac{V_{sx}T_c\frac{1}{2}\ \cos\ \theta_e\ - \cos\ \theta_s\ + \frac{1}{2}A_{sx}T_c^{\,2}\left(\frac{1}{4}\cos\ \theta_e\ - \frac{3}{4}\cos\ \theta_s\right) + \frac{1}{6}J_xT_c^{\,3}\left(\frac{1}{8}\cos\ \theta_e\ - \frac{7}{8}\cos\ \theta_s\right)}{\cos\ \theta_s\ + \cos\ \theta_e} \tag{3.6}$$

Please note that the Y-axis component can be obtained by simply plugging corresponding parameters. The minimum cornering duration T_c is sought by simply saturating at least one of the axis' acceleration or jerk limits. The limiting axis can be identified from axis acceleration transitions, $\Delta A_x = |A_{ex} - A_{sx}|$ and $\Delta A_y = |A_{ey} - A_{sy}|$. Assuming that $\Delta A_x > \Delta A_y$, acceleration, velocity and position constraints for cornering motion are collected from Eqs. (3.2) and (3.6) as:

$$\begin{aligned} &\text{Acc. Constraint}: A_c\cos(\theta_e) = -A_c\cos(\theta_s) + J_xT_c \\ &\text{Vel. Constraint}: V_c\cos(\theta_e) = V_c\cos(\theta_s) - A_c\cos(\theta_s)T_c + \frac{1}{2}J_xT_c^{\,2} \\ &\text{Disp. Constraint}: \begin{aligned} \varepsilon\cos(\theta_\varepsilon) &= \left(\frac{T_c}{2}\right)\left(V_c - \frac{1}{2}A_c\frac{T_c}{2}\right)\cos(\theta_s) + \frac{1}{6}J_x\left(\frac{T_c}{2}\right)^3 \\ &- \left|\frac{T_c\left(V_c - \frac{1}{2}A_cT_c\right)\cos(\theta_s) + \frac{1}{6}J_xT_c^{\,3}}{\cos(\theta_s) + \cos(\theta_e)}\right|\cos(\theta_s) \end{aligned} \end{aligned} \tag{3.7}$$

Notice that Eq. (3.7) provides 3 constraints to compute the 4 unknowns of the cornering trajectory, namely; A_c, V_c, J_x and T_c. The near time-optimal cornering trajectory is obtained analytically by either saturating axis acceleration or the jerk limits, i.e. by setting $J_x = J_{max}$ or $A_x = A_{max}$, and it yields from Eq. (3.7) as:

$$\left.\begin{aligned} A_c &= 2\sqrt[3]{\left|\frac{6J_{\max}{}^2\varepsilon\cos\theta_\varepsilon}{\left(\cos\theta_e-\cos\theta_s\right)\left(\cos\theta_s+\cos\theta_e\right)^2}\right|} \\ V_c &= 2\sqrt[3]{\left|\frac{6^2J_{\max}\varepsilon^2\cos^2\theta_\varepsilon}{\left(\cos\theta_e-\cos\theta_s\right)^2\left(\cos\theta_s+\cos\theta_e\right)}\right|} \\ T_c &= 2\sqrt[3]{\left|\frac{6\varepsilon\cos\theta_\varepsilon\left(\cos\theta_s+\cos\theta_e\right)}{J_{\max}\left(\cos\theta_e-\cos\theta_s\right)}\right|} \\ L_c &= \frac{8\varepsilon\cos\theta_\varepsilon}{\cos\theta_e-\cos\theta_s} \end{aligned}\right\}\text{for } J_x = J_{\max} \qquad (3.8)$$

$$\left.\begin{aligned} J_x &= \sqrt{\left|\frac{A_{\max}{}^3\left(\cos\theta_e-\cos\theta_s\right)\left(\cos\theta_e+\cos\theta_s\right)^2}{48\varepsilon\cos\theta_\varepsilon}\right|} \\ V_c &= 2\sqrt{\left|\frac{3A_{\max}\varepsilon\cos\theta_\varepsilon}{\cos\theta_e-\cos\theta_s}\right|} \\ T_c &= 4\sqrt{\left|\frac{3\varepsilon\cos\theta_\varepsilon}{A_{\max}\left(\cos\theta_e-\cos\theta_s\right)}\right|} \\ L_c &= \frac{8\varepsilon\cos\theta_\varepsilon}{\cos\theta_e-\cos\theta_s} \end{aligned}\right\}\text{for } A_x = A_{\max} \qquad (3.9)$$

Eqs. (3.8) or (3.9) can be used to determine the fastest cornering motion duration T_c, which saturates either acceleration or the jerk limits of one of the drives, e.g. X-axis. Based on the smallest T_c, corresponding cornering velocity and acceleration V_c and A_c boundary conditions are obtained analytically from Eqs. (3.8) and (3.9). The motion for the non-saturated axis, e.g. Y-axis, is planned by calculating its jerk magnitude J_y by re-writing Eq. (3.7) and plugging any of the precomputed T_c, A_c, or V_c. Notice that above equations can be used interchangeably. For instance, for a given cornering velocity V_c, the largest cornering error can be computed that saturates either jerk or acceleration limits of the drives.

On the other hand, there is no limit imposed on the cornering velocity V_c in this formulation. Although it is rare, if the corner angle is very obtuse, V_c can become greater than the set feedrate of the G01 command, $V_c > F$. In this case, cornering speed is capped by the block's commanded feedrate, $V_c = F$ and cornering acceleration is set to zero, $A_c = 0$. In this case, cornering motion planning requires 3-segments, e.g. deceleration increase, constant deceleration and deceleration decrease to alter velocities of the drives from start to the end of the cornering transition [42].

Effectiveness of the proposed local AU-KCS scheme is presented in simulations before it is experimentally validated in Section 5. Figure 3.3 shows the AU-KCS technique applied to smoothen local obtuse and acute corners. The cornering tolerance is set to ε = 30 μm in both cases and the programmed feedrate is F = 100mm/s. Acceleration and jerk limitations of the drives (X and Y axes) are set to $A_{max} = 3\times10^3$ mm/s^2 and $J_{max} = 1\times10^5$ mm/s^3, respectively. Eqs. (3.8) and (3.9) are used to compute the cornering velocities for each case. For the obtuse corner, the cornering speed is calculated to be V_c = 48 mm/s and for the acute corner V_c = 23 mm/s. JLAP is used to decelerate and accelerate from cornering transitions. As shown, proposed 1-step AU-KCS technique can generate acceleration continuous motion profiles around a local corner while saturating at least one axis kinematic limit to attain the fastest cornering motion.

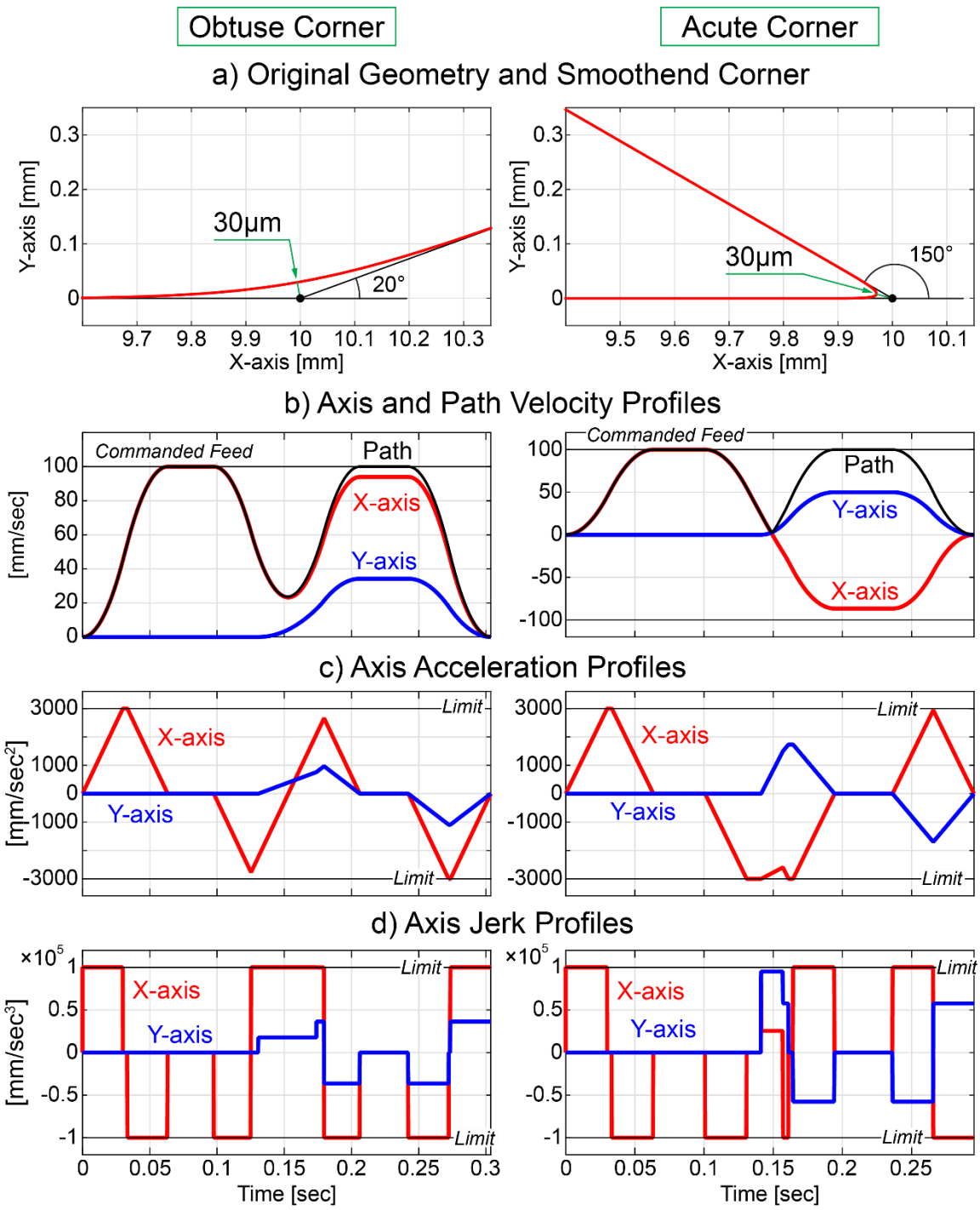

Figure 3.3: Obtuse and acute corner smoothing using AU-KCS.

3.3. Global feed planning along short-segmented tool-paths with uninterrupted acceleration

Previous section introduced the 1-step kinematic corner-smoothing scheme with acceleration uninterrupted motion profiles for smoothing of local corners. An actual high-speed machining tool-path, however, contains dense linear segments and corners that are in close vicinity of each other. This section presents high speed motion planning and accurate interpolation schemes along short-segmented tool-paths based on a look-ahead-windowing (LAW) technique.

Figure 3.4a shows a short-segmented linear tool-path with a series of locally smoothened corners. As shown, when corners are locally smoothened by proposed technique from Section 3.2, the path consists of linear segments and corner transitions. Each corner transition has dedicated cornering velocity V_c and acceleration A_c computed analytically from Eqs. (3.8) and (3.9). A LAW-ing technique is then developed, which reads series of tool-path sections with associated kinematic boundary conditions, and plans the kinematically feasible feed motion. Figure 3.4b shows the proposed LAW based motion planning strategy. As shown, based on the LAW size, motion is planned for N number of segments (G01 blocks) of the tool-path while assuming that a full-stop $F = 0$ is commanded at the end of the LAW. The motion is then planned backwards from the end of LAW to the beginning to reach current block's kinematic conditions. During planning, segment velocities within the window are reduced so that they can be interpolated in a kinematically feasible manner. Once the motion is planned within a LAW, the window is progressed one block and current segment is accepted for interpolation. During planning, compatibility conditions are checked only between consecutive, i.e. kth and $k+1$th segments. Notice that since there is a full-stop planned at the end of the LAW, reducing feedrate of the kth segment to plan a feasible motion towards the $k+1$th block is satisfactory. This type of feed planning approach does not require any iteration, and the rest of the section presents analytical methods to compute maximum speed and boundary conditions for consecutive segments.

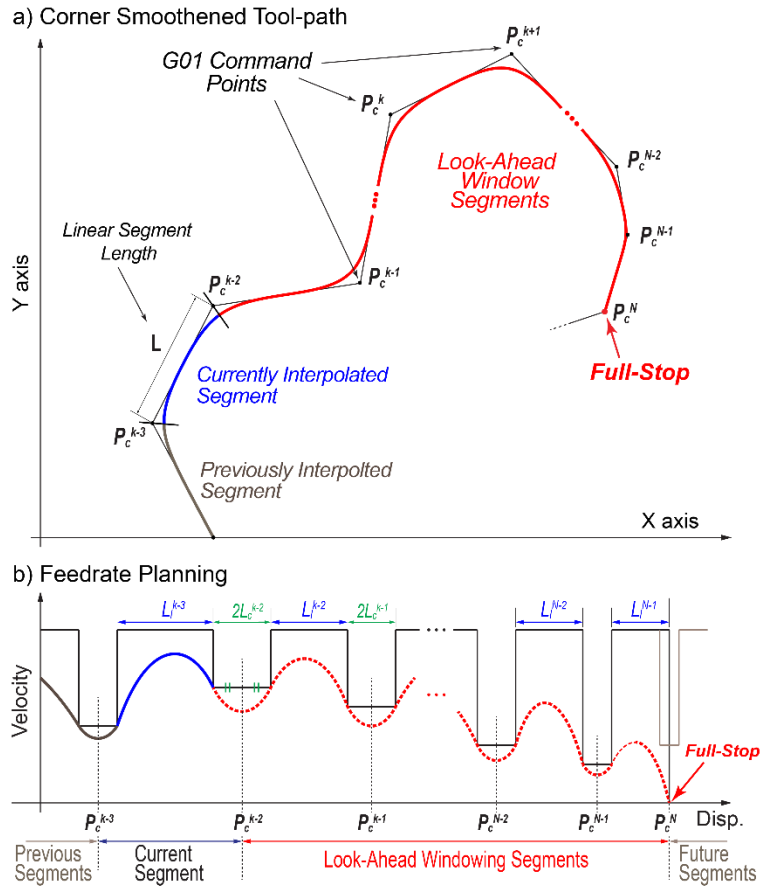

Figure 3.4: Look-ahead windowing (LAW) based motion planning strategy.

3.3.1. Motion planning along separated corners

The feed planning strategy starts with smoothing corners locally based on the AU-KCS presented in Section 3.2. Corner transition distance L_c^k for each corner is computed from Eq. (3.5), and the remaining linear distance between consecutive corner transitions is calculated as:

$$L_l^k = \left\| P_c^k - P_c^{k+1} \right\| - \left(L_c^k + L_c^{k+1} \right) \tag{3.10}$$

As shown in Figure 3.5a if $L_l^k > 0$ consecutive corners are not overlapping, and thus feed motion can be planned to stitch two local corners. This stitching motion is planned based on the 7-segmented JLAP depicted in Figure 3.6, and explained as follows.

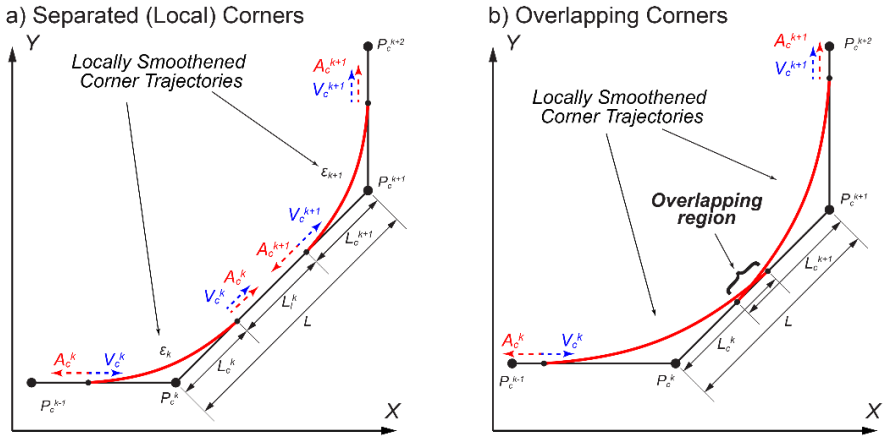

Figure 3.5: Separate (Local) and overlapping cornering geometries.

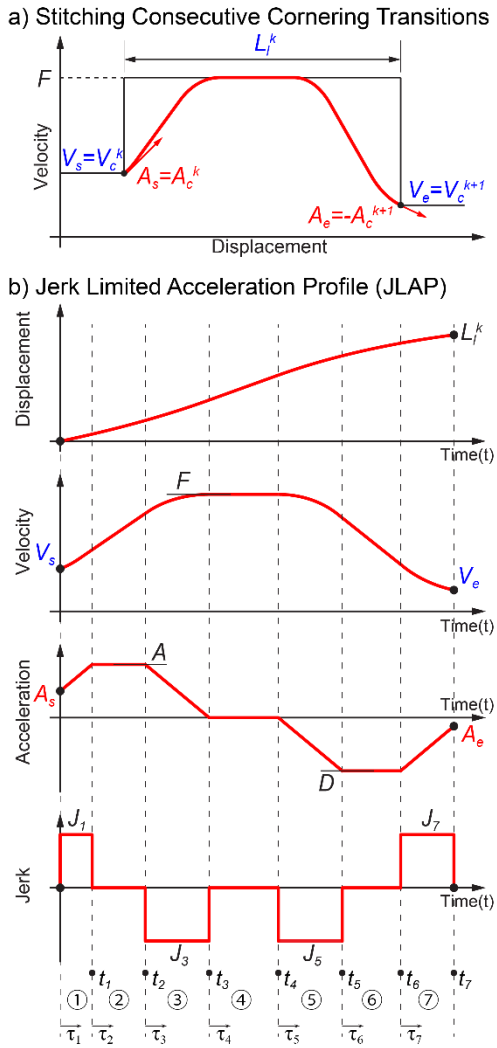

Figure 3.6: Jerk limited acceleration profile (JLAP).

The objective is to attain the maximum possible feedrate along straight sections in-between consecutive corners with respect to velocity V_c^k, V_c^{k+1} and acceleration A_c^k, A_c^{k+1} boundary conditions. The jerk sequence to stitch the motion is planned with respect to Figure 3.6 as:

$$j(\tau)=\begin{cases} J_1 = J_{\max}, & 0 \le t < t_1 \\ J_2 = 0, & t_1 \le t < t_2 \\ J_3 = -J_{\max}, & t_2 \le t < t_3 \\ J_4 = 0, & t_3 \le t < t_4 \\ J_5 = -J_{\max}, & t_4 \le t < t_5 \\ J_6 = 0, & t_5 \le t < t_6 \\ J_7 = J_{\max}, & t_6 \le t \le t_7 \end{cases} \tag{3.11}$$

Integrating Eq. (3.11) with respect to time, t, reveals the trapezoidal acceleration profile as,

$$a(\tau)=\begin{cases} A_c^k + J_1\tau_1, & 0 \le t < t_1, & A_1 = A_c^k + J_1T_1 = A \\ A, & t_1 \le t < t_2, & A_2 = A_1 = A \\ A + J_3\tau_3, & t_2 \le t < t_3, & A_3 = A + J_3T_3 = 0 \\ 0, & t_3 \le t < t_4, & A_4 = A_3 = 0 \\ J_5\tau_5, & t_4 \le t < t_5, & A_5 = J_5T_5 = D \\ D, & t_5 \le t < t_6, & A_6 = A_5 = D \\ D + J_7\tau_7, & t_6 \le t \le t_7, & -A_c^{k+1} = D + J_7T \end{cases} \tag{3.12}$$

where A and D are acceleration and deceleration amplitudes, and τ_i is the relative time parameter, which starts at the beginning of the ith phase of the profile (See Figure 3.6). Similarly, integrating Eq. (3.12) generates the corresponding velocity profile as,

$$v\ \tau\ = \begin{cases} V_c^k + A_s\tau_1 + \frac{1}{2}J_1\tau_1^{\ 2}, & 0 \le t < t_1,\ \ V_1 = V_c^k + A_sT_1 + \frac{1}{2}J_1T_1^2 \\ V_1 + A\tau_2, & t_1 \le t < t_2,\ \ V_2 = V_1 + AT_2 \\ V_2 + A\tau_3 + \frac{1}{2}J_3\tau_3^{\ 2}, & t_2 \le t < t_3,\ \ V_3 = V_2 + AT_3 + \frac{1}{2}J_3T_3^{\ 2} \\ V_3, & t_3 \le t < t_4,\ \ V_4 = V_3 = F \\ V_4 + \frac{1}{2}J_5\tau_5^{\ 2}, & t_4 \le t < t_5,\ \ V_5 = V_4 + \frac{1}{2}J_5T_5^{\ 2} \\ V_5 + D\tau_6, & t_5 \le t < t_6,\ \ V_6 = V_5 + DT_6 \\ V_6 + D\tau_7 + \frac{1}{2}J_7\tau_7^{\ 2}, & t_6 \le t \le t_7,\ \ V_c^{k+1} = V_6 + DT_7 + \frac{1}{2}J_7T_7^{\ 2} \end{cases} \tag{3.13}$$

T_i (i = 1…7) is the duration, V_i is the velocity reached at the end of the ith phase. Consequently, integrating Eq. (3.13) with respect to time yields the displacement profile,

$$s\ \tau\ = \begin{cases} V_s\tau_1 + \frac{1}{2}A_s\tau_1^{\ 2} + \frac{1}{6}J_1\tau_1^{\ 3}, & 0 \le t < t_1,\ \ S_1 = V_sT_1 + \frac{1}{2}A_sT_1^2 + \frac{1}{6}J_1T_1^3 \\ S_1 + V_1\tau_2 + \frac{1}{2}A\tau_2^{\ 2}, & t_1 \le t < t_2,\ \ S_2 = S_1 + V_1T_2 + \frac{1}{2}AT_2^{\ 2} \\ S_2 + V_2\tau_3 + \frac{1}{2}A\tau_3^{\ 2} + \frac{1}{6}J_3\tau_3^{\ 3}, & t_2 \le t < t_3,\ \ S_3 = S_2 + V_2T_3 + \frac{1}{2}AT_3^{\ 2} + \frac{1}{6}J_3T_3^3 \\ S_3 + V_3\tau_4, & t_3 \le t < t_4,\ \ S_4 = S_3 + V_3T_4 \\ S_4 + V_4\tau_5 + \frac{1}{6}J_5\tau_5^{\ 3}, & t_4 \le t < t_5,\ \ S_5 = S_4 + V_4T_5 + \frac{1}{6}J_5T_5^3 \\ S_5 + V_5\tau_6 + \frac{1}{2}D\tau_6^{\ 2}, & t_5 \le t < t_6,\ \ S_6 = S_5 + V_5T_6 + \frac{1}{2}DT_6^{\ 2} \\ S_6 + V_6\tau_7 + \frac{1}{2}D\tau_7^{\ 2} + \frac{1}{6}J_7\tau_7^{\ 2}, & t_6 \le t \le t_7,\ \ S_e = S_6 + V_6T_7 + \frac{1}{2}DT_7^{\ 2} + \frac{1}{6}J_7T_7^{\ 3} = L_l^k \end{cases} \tag{3.14}$$

where s_i (i = 1..7) is the displacement reached at the end of each phase. Durations T_i (i = 1…7) of each phase must be computed to realize the JLAP. Firstly, considering the trapezoidal nature of the profile, acceleration demand is evaluated from,

$$\left.\begin{aligned} A &= \sqrt{J_{\max}\left(F - V_c^k\right) + \frac{\left(A_c^k\right)^2}{2}} \quad \rightarrow \quad T_1 = \frac{A - A_c^k}{J_{\max}}, T_3 = \frac{A}{J_{\max}} \\ D &= -\sqrt{J_{\max}\left(F - V_c^{k+1}\right) + \frac{\left(A_c^{k+1}\right)^2}{2}} \quad \rightarrow \quad T_7 = \frac{A_c^{k+1} - D}{J_{\max}}, T_5 = \frac{D}{-J_{\max}} \end{aligned}\right\} \tag{3.15}$$

Above, J_{max} is tangential jerk limit. If magnitudes A or D exceed their limits A_{max}, they are capped by the limit, and constant acceleration duration T_2 and T_6 are computed as:

$$\left.\begin{aligned} T_2 &= \frac{F - V_c^k}{A_{\max}} - \frac{A_{\max}}{J_{\max}} + \frac{1}{2}\frac{\left(A_c^k\right)^2}{A_{\max} J_{\max}} \\ T_6 &= \frac{F - V_c^{k+1}}{A_{\max}} - \frac{A_{\max}}{J_{\max}} + \frac{1}{2}\frac{\left(A_c^{k+1}\right)^2}{A_{\max} J_{\max}} \end{aligned}\right\} \tag{3.16}$$

Otherwise, constant acceleration phase simply does not exist, $T_2 = T_6 = 0$.

Finally, the distance between cornering transitions, L_l^k, should be traveled by all the stages of interpolation. If the linear distance between corners is long enough, programmed feedrate of the block, F should be achieved implying that $T_4 \geq 0$. The total distance traveled along the 7-segmented JLAP is can be computed from Eqs. (3.14)-(3.16) as:

$$\begin{aligned} L_l^k = {} & T_4 F + \frac{F^2 - \left(V_c^k\right)^2}{2A} + \frac{\left(V_c^{k+1}\right)^2 - F^2}{2D} + \frac{A\left(F + V_c^k\right)}{2J_{\max}} - \frac{D\left(F + V_c^{k+1}\right)}{2J_{\max}} \\ & + \frac{1}{8J_{\max}^2}\left(\frac{\left(A_c^{k+1}\right)^4}{D} - \frac{\left(A_c^k\right)^4}{A}\right) + \frac{\left(A_c^k\right)^3 - \left(A_c^{k+1}\right)^3}{3J_{\max}^2} + \frac{\left(A_c^{k+1}\right)^2 D - \left(A_c^k\right)^2 A}{4J_{\max}^2} \\ & + \frac{1}{2J_{\max}}\left(\frac{\left(A_c^k\right)^2 V_c^k}{A} - \frac{\left(A_c^{k+1}\right)^2 V_c^{k+1}}{D}\right) + \frac{A_c^{k+1} V_c^{k+1} - A_c^k V_c^k}{J_{\max}} \end{aligned} \tag{3.17}$$

and cruise velocity duration T_4 is revealed from Eq. (3.17) as:

$$T_4 = \frac{1}{F}\left[L_l^k - \left\{\begin{array}{l} \dfrac{F^2 - V_c^{k\ 2}}{2A} + \dfrac{V_c^{k+1\ 2} - F^2}{2D} + \dfrac{A\ F + V_c^k}{2J_{\max}} - \dfrac{D\ F + V_c^{k+1}}{2J_{\max}} \\ + \dfrac{1}{8J_{\max}^{\ 2}}\left(\dfrac{A_c^{k+1\ 4}}{D} - \dfrac{A_c^{k\ 4}}{A}\right) + \dfrac{A_c^{k\ 3} - A_c^{k+1\ 3}}{3J_{\max}^{\ 2}} + \dfrac{A_c^{k+1\ 2}D - A_c^{k\ 2}A}{4J_{\max}^{\ 2}} \\ + \dfrac{1}{2J_{\max}}\left(\dfrac{A_c^{k\ 2}V_c^k}{A} - \dfrac{A_c^{k+1\ 2}V_c^{k+1}}{D}\right) + \dfrac{A_c^{k+1}V_c^{k+1} - A_c^k V_c^k}{J_{\max}} \end{array}\right\}\right] \geq 0$$

(3.18)

If Eq. (3.18) does not hold, $T_4 < 0$, it is set to zero $T_4 = 0$, and the maximum reachable feedrate F_{new} is computed from the solution of the following quadratic equation:

$$aF_{new}^{\ 2} + bF_{new} + c = 0 \tag{3.19}$$

where

$$\left.\begin{array}{l} a = \dfrac{1}{2D} - \dfrac{1}{2A}, \quad b = \dfrac{D - A}{2J_{\max}} \\ c = L + \dfrac{1}{8J_{\max}^{\ 2}}\left(\dfrac{A_c^{k\ 4}}{A} - \dfrac{A_c^{k+1\ 4}}{D}\right) + \dfrac{1}{2J_{\max}}\left(\dfrac{A_c^{k+1\ 2}V_c^{k+1}}{D} - \dfrac{A_c^{k\ 2}V_c^k}{A}\right) \\ \quad + \dfrac{A_c^{k+1\ 3} - A_c^{k\ 3}}{3J_{\max}^{\ 2}} + \dfrac{A_c^{k\ 2}A - A_c^{k+1\ 2}D}{4J_{\max}^{\ 2}} \\ \quad + \dfrac{2A_c^k - A\ V_c^k - 2A_c^{k+1} - D\ V_c^{k+1}}{2J_{\max}} + \dfrac{V_c^{k\ 2}}{2A} - \dfrac{V_c^{k+1\ 2}}{2D} \end{array}\right\} \tag{3.20}$$

If Eq. (3.19) possesses complex roots, consecutive cornering velocities V_c^k, V_c^{k+1} and accelerations A_c^k, A_c^{k+1} need to be adjusted to find a feasible solution for planning the JLAP. Non-linear optimization techniques [45] can be used to plan a time optimal motion. However, utilizing iterative schemes for optimizing such a short motion is not feasible in real-time implementation, and should not improve the overall cycle time significantly. Thus, the following analytical approach is developed to determine a sub-optimal cornering velocity to plan a feasible JLAP for stitching the cornering transitions as follows.

Firstly, as shown in Figure 3.7 consecutive block's cornering speeds are lowered to the minimum

$$V_c^k = V_c^{k+1} = \min\left\{V_c^k, V_c^{k+1}\right\} \tag{3.21}$$

Notice that lowering cornering velocities reduces cornering errors ε_k, ε_{k+1} from their preset tolerance as well. Consequently, corner transition lengths L_c^k, L_c^{k+1} are shortened, and this elongates L_l^k (See Figure 3.7). Eq. (3.17) is then used to plan a feasible JLAP motion to stitch consecutive cornering transitions. If a feasible JLAP can be planned, then consecutive cornering velocities are increased to reduce the cycle time. The following analytical technique is developed to search for feasible cornering velocities in a computationally efficient and real-time suitable scheme.

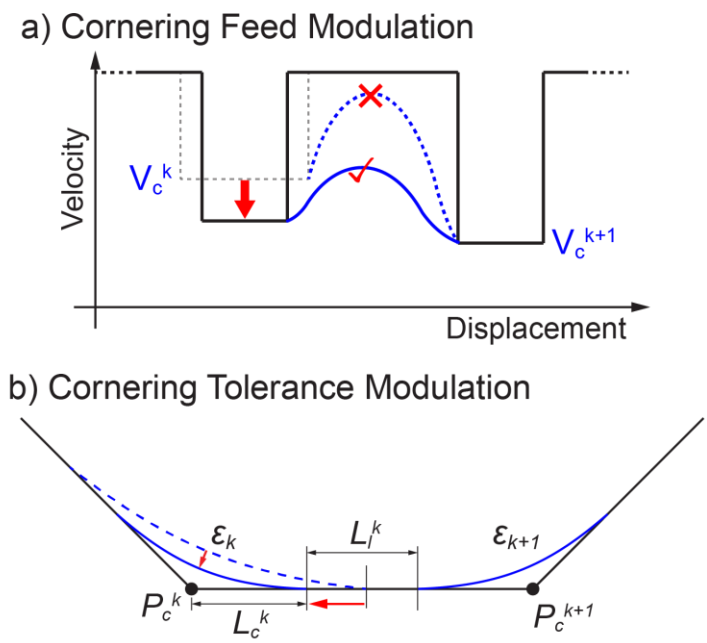

Figure 3.7: Feed adjustment in-between local corners.

Firstly, linear distance between smoothened corners, L_l^k is typically very small <1 mm in dense short-segmented tool-paths, and this makes planning a complete 7-segmented JLAP unlikely. Instead, a shortened 3-segmented JLAP is planned to interpolate an acceleration uninterrupted cornering transitions. It is assumed that the constant feed, acceleration, and deceleration sections of the 7-segmented JLAP are non-existing, $T_4 = T_2 = T_6 = 0$, and it is assumed that if $V_c^k > V_c^{k+1}$ the motion starts with an acceleration decrease stage ($T_1 = 0$). Similarly, if $V_c^{k+1} > V_c^k$, a deceleration increase section does not exist ($T_7 = 0$). The objective is to analytically compute near time-optimal cornering velocity V_c^k and

the corresponding cornering acceleration A_c^k based on this simplified 3-segmented JLAP. As observed from Eqs. (3.8) and (3.9), altering cornering speed actually alters ε, A_c^k, L_c^k. As a matter of fact, cornering error ε, scales V_c^k, A_c^k, L_c^k, and this relationship can be captured by introducing a scaling factor α as:

$$\left.\begin{aligned} A_c^{'k} &= \sqrt{\alpha} A_c^k \\ V_c^{'k} &= \alpha V_c^k \\ L_c^{'k} &= \alpha^2 L_c^k \end{aligned}\right\} \text{ where } \alpha \geq 1 \tag{3.22}$$

The achievable feedrate F and deceleration D values are determined based on the 3-segmented JLAP as,

$$\left.\begin{aligned} F &= V_c^{'k} + \frac{A_c^{'k}{}^2}{2J_{\max}} \\ D &= -\sqrt{V_c^{'k} - V_c^{k+1}\ J_{\max} + \frac{A_c^{'k}{}^2 + A_c^{k+1}{}^2}{2}} \end{aligned}\right\} \tag{3.23}$$

and corresponding segment durations are obtained;

$$T_3 = \frac{A_c^{'k}}{J_{\max}}, T_5 = \frac{D}{-J_{\max}}, T_7 = \frac{A_c^{'k+1} - D}{J_{\max}} \tag{3.24}$$

The total displacement traveled can be computed by integrating the acceleration profile with the durations from Eq. (3.24) and amplitudes from Eqs. (3.22) and (3.23) as:

$$\begin{aligned} L = {} & \frac{A_c^k{}^3}{3J_{\max}{}^2} + \frac{A_c^k{}^2 A_c^{k+1}}{2J_{\max}{}^2} - \frac{A_c^k{}^2 D}{J_{\max}{}^2} + \frac{A_c^{k+1}{}^3}{6J_{\max}{}^2} - \frac{A_c^{k+1} D^2}{J_{\max}{}^2} + \frac{D^3}{J_{\max}{}^2} \\ & + \frac{\alpha V_c^k}{J_{\max}}\ A_c^k + A_c^{k+1} - 2D\ +\ \alpha^2 - 1\ L_c^k = L_l^k \end{aligned} \tag{3.25}$$

Finally, the scaling factorα is calculated analytically from Eq. (3.25):

$$a\alpha^4 + b\alpha^3 + c\alpha^2 + d\alpha + e = 0$$

$$a = 1,\ \ b = -\frac{V_c^k{}^3}{J_{\max}\ L_c^k{}^2} + \frac{2A_c^k V_c^k}{J_{\max} L_c^k},$$

$$c = -\frac{4\ A_c^k{}^3 + A_c^{k+1}{}^3}{6J_{\max}{}^2 L_c^k} + \frac{3\ A_c^{k+1}{}^2 - A_c^k{}^2\ V_c^k{}^2}{6J_{\max}{}^2\ L_c^k{}^2} + \frac{2V_c^{k+1}A_c^{k+1}}{J_{\max} L_c^k} - \frac{2\ L_c^k + L_l^k}{L_c^k}$$

$$-\frac{V_c^{k+1}\ V_c^k{}^2}{J_{\max}\ L_c^k{}^2}$$

$$d = -V_c^k \left(\begin{array}{l} \frac{8A_c^k\ A_c^{k+1}{}^3}{12J_{\max}{}^3\ L_c^k{}^2} + \frac{A_c^k{}^4}{12J_{\max}{}^3\ L_c^k{}^2} - \frac{3\left(A_c^{k+1}{}^4\right)}{12J_{\max}{}^3\left(L_c^k{}^2\right)} - \frac{\left(A_c^k{}^2\right)\left(A_c^{k+1}{}^2\right)}{12J_{\max}{}^3\left(L_c^k{}^2\right)} \\ -\frac{\left(V_c^{k+1}{}^2\right)}{J_{\max}\left(L_c^k{}^2\right)} + \frac{2A_c^k\left(L_c^k + L_l^k\right)}{J_{\max}\left(L_c^k{}^2\right)} + \frac{\left(A_c^k - A_c^{k+1}{}^2\right)V_c^{k+1}}{J_{\max}{}^2\left(L_c^k{}^2\right)} \end{array} \right)$$

$$e = \frac{\left(A_c^k{}^2\right)\left(A_c^{k+1}{}^2\right)\left(A_c^k - A_c^{k+1}{}^2\right)}{8J_{\max}{}^4\left(L_c^k{}^2\right)} - \frac{\left(\left(A_c^k{}^3\right) - \left(A_c^{k+1}{}^3\right)\right)^2}{72J_{\max}{}^4\left(L_c^k{}^2\right)}$$

$$+ V_c^{k+1}\frac{\left(A_c^{k+1}{}^4\right) - 3\left(A_c^k{}^4\right) + 8\left(A_c^k{}^3\right)A_c^{k+1} - 6\left(A_c^k{}^2\right)\left(A_c^{k+1}{}^2\right)}{12J_{\max}{}^3 L_c^{k2}} \tag{3.26}$$

$$-\frac{2\left(\left(A_c^k{}^3\right) - \left(A_c^{k+1}{}^3\right)\right)\left(L_c^k + L_l^k\right)}{3J_{\max}{}^2\left(L_c^k{}^2\right)} + \frac{-2A_c^{k+1}V_c^{k+1}\left(L_c^k + L_l^k\right)}{J_{\max}\left(L_c^k{}^2\right)} + \frac{\left(L_c^k + L_l^k{}^2\right)}{\left(L_c^k{}^2\right)}$$

$$+\frac{\left(\left(A_c^k{}^2\right) - \left(A_c^{k+1}{}^2\right)\right)\left(V_c^{k+1}{}^2\right)}{2J_{\max}{}^2\left(L_c^k{}^2\right)} + \frac{\left(V_c^{k+1}{}^3\right)}{J_{\max}\left(L_c^k{}^2\right)}$$

The cornering velocity is obtained based on α without the need of iterative procedure from Eq. (22). This approach provides an effective analytical planning of a near-time optimal JLAP to stitch consecutive cornering transitions. If a feasible solution could not be found from Eq. (26), both current and next segment velocities V_c^k and V_c^{k+1} need to be further lowered. This can be done by scaling consecutive cornering velocities simultaneously by

$$\left\{ \begin{array}{l} A_c^{'k} = \alpha A_c^k \\ V_c^{'k} = \alpha^2 V_c^k \\ L_c^{'k} = \alpha^3 L_c^k \end{array} \right\} \text{ and } \left\{ \begin{array}{l} A_c^{'k+1} = \alpha A_c^{k+1} \\ V_c^{'k+1} = \alpha^2 V_c^{k+1} \\ L_c^{'k+1} = \alpha^3 L_c^{k+1} \end{array} \right\} \text{where } 0 \le \alpha \le 1 \tag{3.27}$$

and the associated scaling factor α can be calculated analytically following the similar procedure presented above.

3.3.2. Motion planning along overlapping corners

Previous section presented planning of a continuous and rapid motion along locally corner-smoothed tool-path. If locally smoothed corners are not overlapping as shown in Figure 3.5a, the approach presented in the Section 3.3.1 is effective to plan a jerk limited motion analytically at low computational expense. Nevertheless, if linear segment lengths are short and cornering errors are large, locally smoothened corners overlap each other as shown in Figure 3.5b. This section presents planning of an uninterrupted jerk limited feed motion along overlapping corners.

The general geometry for two locally smoothed overlapping corners is presented in Figure 3.8. Local corner trajectories are planned based on the technique presented in Section 3.2. The idea is to connect mid-points of overlapping corners to generate an acceleration un-interrupted rapid cornering motion (See Figure 3.8a). The mid-point kinematics during cornering transitions can be calculated as:

$$\begin{Bmatrix} a_{mx}^k = \dfrac{a_{sx}^k + a_{ex}^k}{2} \\ v_{mx}^k = \dfrac{v_{sx}^k + v_{ex}^k}{4} \\ s_{mx}^k = \varepsilon_x^k + x_c^k \end{Bmatrix} \text{ and } \begin{Bmatrix} a_{my}^k = \dfrac{a_{sy}^k + a_{ey}^k}{2} \\ v_{my}^k = \dfrac{v_{sy}^k + v_{ey}^k}{4} \\ s_{my}^k = \varepsilon_y^k + y_c^k \end{Bmatrix} \tag{3.28}$$

where $a_{mx}{}^k$, $v_{mx}{}^k$, $a_{mx}{}^k$ and $a_{my}{}^k$, $v_{my}{}^k$, $a_{my}{}^k$ are the midpoint acceleration, velocity and position values for the X and Y axes. The mid-points of consecutive corners are then connected by a 2-segmented jerk profile as shown in Figure 3.8b. The kinematic constraints for interpolating midpoints are written for the X-axis as:

$$\left.\begin{aligned}
&\text{Acc. Constraint :} \quad a_{mx}^{k+1} = a_{mx}^{k} + j_{m1x}\lambda_x T_m + j_{m2x}\left(1-\lambda_x\right) T_m \\
&\text{Vel. Constraint :} \quad \begin{aligned} v_{mx}^{k+1} &= v_{mx}^{k} + a_{mx}^{k}\lambda_x T_m + \frac{1}{2} j_{m1x}\lambda_x^{\,2} T_m^{\,2} \\ &+ \left(a_{mx}^{k} + j_{m1x}\lambda_x T_m\right)\left(1-\lambda_x\right) T_m + \frac{1}{2} j_{m2x}\left(1-\lambda_x\right)^2 T_m^{\,2} \end{aligned} \\
&\text{Disp. Constraint :} \quad \begin{aligned} s_{mx}^{k+1} &= s_{mx}^{k} + v_{mx}^{k}\lambda_x T_m + \frac{1}{2} a_{mx}^{k}\lambda_x^{\,2} T_m^{\,2} + \frac{1}{6} j_{m1x}\lambda_x^{\,3} T_m^{\,3} \\ &+ \left(v_{mx}^{k} + a_{mx}^{k}\lambda_x T_m + \frac{1}{2} j_{m1x}\lambda_x^{\,2} T_m^{\,2}\right)\left(1-\lambda_x\right) T_m \\ &+ \frac{1}{2}\left(a_{mx}^{k} + j_{m1x}\lambda_x T_m\right)\left(1-\lambda_x\right)^2 T_m^{\,2} + \frac{1}{6} j_{m2x}\left(1-\lambda_x\right)^3 T_m^{\,3} \end{aligned}
\end{aligned}\right\} \quad (3.29)$$

Above, T_m is the motion duration, j_{m1x} and j_{m2x} are the jerk amplitudes, and $0 < \lambda_x < 1$ controls timing of jerk transition (See Figure 3.8b). Note that the Y-axis complement of Eq. (3.29) can be written with λ_y, j_{m1y}, j_{m2y}. There are total of 7 unknowns, T_m, λ_x, λ_y, j_{m1x}, j_{m2x}, j_{m1y}, j_{m2y} that needs to be solved from total 6 equalities (X and Y axis contributions) given in Eq. (3.29). The solution can be obtained in 2 steps. Firstly, $\lambda_x = 0.5$ is set to divide X-axis motion into half. Three of the unknowns, T_m, j_{m1x} and j_{m2x} are calculated from Eq. (3.29). Using T_m, the remaining unknowns λ_y, j_{m1y} and j_{m2y} are obtained from the Y-axis complement. Notice that the following kinematic limits need to be satisfied to plan a feasible motion:

$$0 \le T_m, \quad 0 < \begin{Bmatrix} \lambda_x \\ \lambda_y \end{Bmatrix} < 1, \quad -J_{\max} \le \begin{Bmatrix} j_{m1x} \\ j_{m2x} \\ j_{m1y} \\ j_{m2y} \end{Bmatrix} \le J_{\max} \qquad (3.30)$$

If Eq. (3.30) is not satisfied, V_c^k and/or V_c^{k+1} should be reduced to satisfy the constraints. The problem could be formulated as a nonlinear optimization problem and solved numerically. In order to avoid costly iterative schemes, the following analytical approach is developed to determine a sub-optimal cornering velocity.

Firstly, the cornering velocities are set identical to $V_c^k = V_c^{k+1} = \min\{V_c^k, V_c^{k+1}\}$, and existence of a feasible solution is checked from Eqs. (3.29) and (3.30). If a feasible solution could not be obtained, a scaling parameter is used to capture the relationship,

$$\begin{Bmatrix} a'^{k}_{mx} = \alpha a^{k}_{mx} \\ v'^{k}_{mx} = \alpha^{2} v^{k}_{mx} \\ s'^{k}_{mx} = \alpha^{3} \left(s^{k}_{mx} - x^{k}_{c} \right) + x^{k}_{c} \end{Bmatrix}, \begin{Bmatrix} a'^{k}_{my} = \alpha a^{k}_{my} \\ v'^{k}_{my} = \alpha^{2} v^{k}_{my} \\ s'^{k}_{my} = \alpha^{3} \left(s^{k}_{my} - y^{k}_{c} \right) + y^{k}_{c} \end{Bmatrix} \text{where } 0 \le \alpha \le 1 \tag{3.31}$$

and seek for a faster cornering motion. Including the scaling factor, α, the total unknowns to plan overlapping cornering motion becomes 8; namely, α, T_m, λ_x, j_{m1x}, j_{m2x}, λ_y, j_{m1y}, j_{m2y}. Firstly, the interpolation time T_m is approximated using linear distance in between the corners and the average velocity as:

$$T_m = \frac{\sqrt{\left(s^{k}_{mx} - s^{k+1}_{mx} \right)^{2} + \left(s^{k}_{my} - s^{k+1}_{my} \right)^{2}}}{\left(\frac{\sqrt{\left(v^{k}_{mx} \right)^{2} + \left(v^{k}_{my} \right)^{2}} + \sqrt{\left(v^{k+1}_{mx} \right)^{2} + \left(v^{k+1}_{my} \right)^{2}}}{2} \right)} \tag{3.32}$$

Next, in order to seek for rapid motion, one of the X or Y-axis jerk amplitudes is saturated, and the remaining 6 unknowns are solved analytically from Eq. (3.29) by plugging scaled mid-corner position (s_{mx}', s_{my}'), velocity (v_{mx}', v_{my}') and accelerations (a_{mx}', a_{my}') from Eq. (3.31). Notice that there are 4 cases to consider, $j_{m1x} = J_{max}$, $j_{m2x} = J_{max}$, $j_{m1y} = J_{max}$, $j_{m2y} = J_{max}$. The feasibility of the solution is checked from Eq. (3.30), and the parameter set with the largest α is selected. For instance, if $j_{m1x} = J_{max}$ the scaling factor α is computed from

$$\begin{aligned} & a\alpha^{4} + b\alpha^{3} + c\alpha^{2} + d\alpha + e = 0 \\ & \begin{cases} a = 4 \left(v^{k}_{mx} \right)^{2} + 6 a^{k}_{mx} x^{k}_{c} - 6 a^{k}_{mx} s^{k}_{mx} \\ b = -6 a^{k+1}_{mx} x^{k}_{c} + 6 a^{k+1}_{mx} s^{k}_{mx} + 2 a^{k}_{mx} v^{k}_{mx} T_m + 6 j_{m1x} x^{k}_{c} T_m \\ \quad -6 j_{m1x} s^{k}_{mx} T_m \\ c = -8 v^{k}_{mx} v^{k+1}_{mx} + \left(a^{k}_{mx} \right)^{2} T_m^{2} + 6 a^{k+1}_{mx} v^{k}_{mx} T_m - 2 j_{m1x} v^{k}_{mx} T_m^{2} \\ d = -6 a^{k}_{mx} x^{k}_{c} + 6 a^{k}_{mx} s^{k+1}_{mx} - 8 a^{k}_{mx} v^{k+1}_{mx} T_m + 3 a^{k}_{mx} a^{k+1}_{mx} T_m^{2} \\ e = 4 \left(v^{k+1}_{mx} \right)^{2} + 6 a^{k+1}_{mx} x^{k}_{c} - 6 a^{k+1}_{mx} s^{k+1}_{mx} - 6 j_{m1x} x^{k}_{c} T_m \\ \quad +6 j_{m1x} s^{k+1}_{mx} T_m + a^{k+1}_{mx} j_{m1x} T_m^{3} - 4 j_{m1x} v^{k+1}_{mx} T_m^{2} \end{cases} \end{aligned} \tag{3.33}$$

either analytically or using a simple Newton-Raphson [45] type iterative technique, both of which are computationally efficient. If a feasible solution could be obtained, α is used to scale both corner velocities simultaneously similar to Eq. (3.27) (See Section 3.3.1), and all the unknowns can be determined analytically.

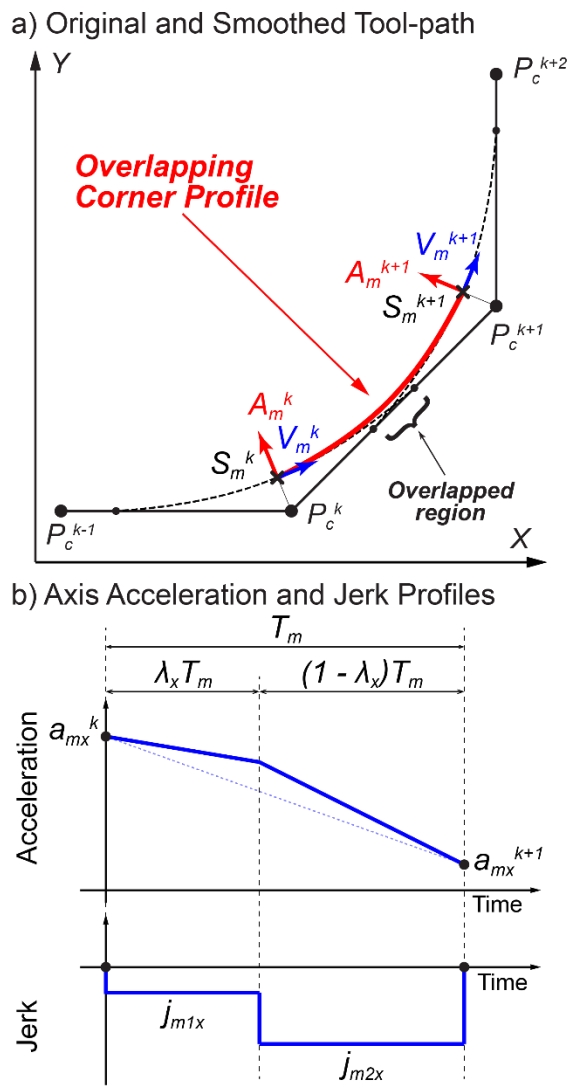

Figure 3.8: Overlapping corner smoothing strategy.

3.4. Illustrative examples

This section presents application of the methods introduced in Sections 3.3.1 and 3.3.2 on a simple short-segmented tool-path. When corners are not overlapping, we identify the case as “local cornering” and apply the acceleration uninterrupted corner smoothing technique presented in Section 3.2 jointly with the feed planning method presented in Section 3.3.1. If local corners are overlapping, we will identify the case as “global cornering” and apply the method presented in Section 3.3.2.

As shown in Figure 3.9, the test tool-path consists of 3 linear moves generating 2 right-handed 45° corners. The length of linear segments before and after the two corners are fixed to 10 mm, and the length of linear segment between corners is set to 0.5 mm. Programmed feedrate is F = 100mm/s, and acceleration and jerk limits of the drives (X and Y axis) are set to $A_{max} = 3 \times 10^3$ mm/s^2 and $J_{max} = 1 \times 10^5$ mm/s^3, respectively.

Figure 3.9 (left hand side) shows application of "local corner smoothing" with a corner tolerance of ε = 9 μm. In this case, corners are well separated and thus they can be smoothened locally. Both axis jerk limits are utilized to generate fast cornering trajectories. Cornering velocities are computed identically as V_c = 13.26mm/s. The JLAP profile to stitch local corners is generated analytically, and the resultant total cycle time is 0.357 s.

On the other hand, when the cornering tolerance increased to ε = 80 μm, locally smoothened corner trajectories overlap as discussed in Section 3.3.2. Since the linear distance between locally smoothened corners is less than zero the algorithm presented in Section 3.3.2 is applied. As shown in Figure 3.9 (right hand side), overlapping corners are traveled in an uninterrupted fashion. The overlapping corner interpolation can keep cornering velocity higher as compared to the local corner interpolation technique. Figure 3.9c and d show generated axis acceleration and jerk profiles. As shown, the acceleration and jerk limits of the drives are fully utilized, the motion is continuous and smooth, and the total cycle time could be reduced to 0.317 s.

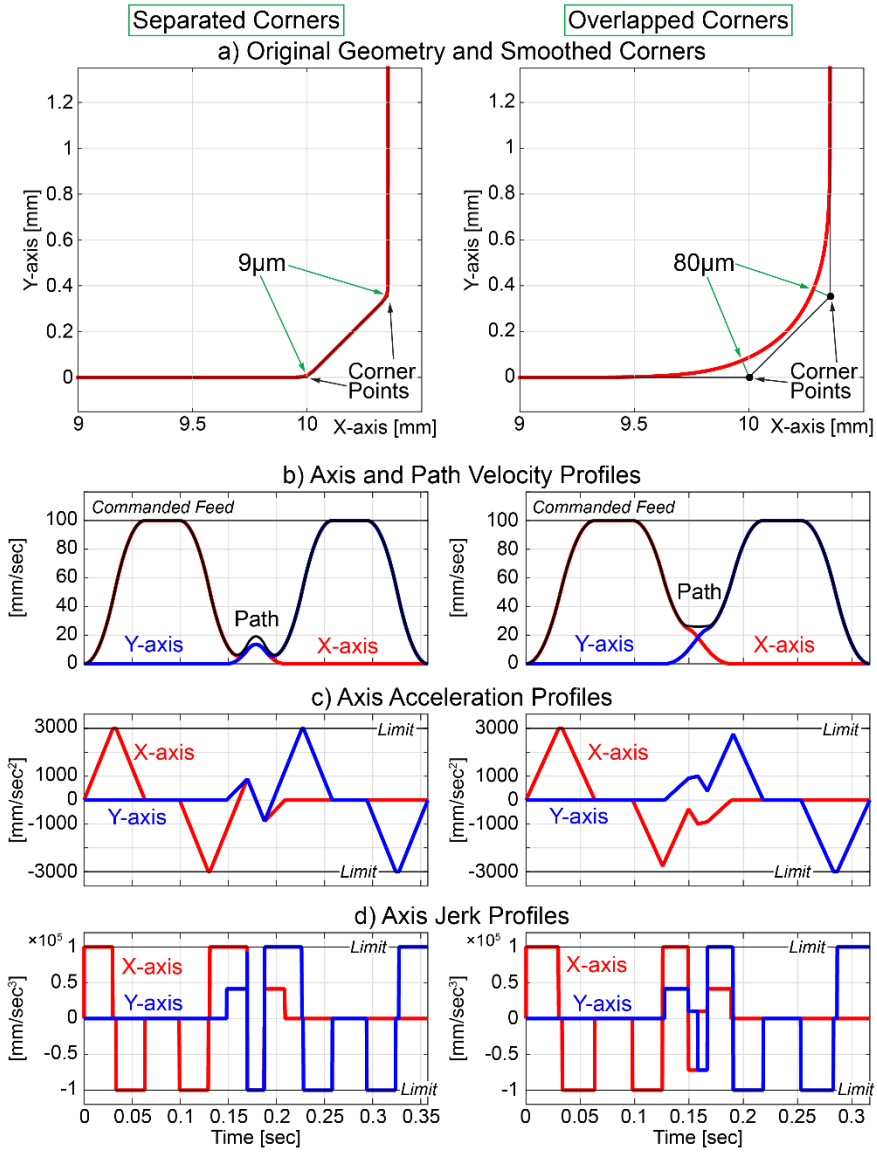

Figure 3.9: Right-Handed Consecutive Corner Smoothing Using Local and Overlapping Cornering Interpolation Techniques.

3.5. Experimental results

Experimental validation and benchmark comparisons of the proposed technique are performed on the Cartesian X-Y motion system shown in Figure 3.10. The planar X-Y motion table is driven by 3 linear motors. The heavier X-axis is designed as gantry and

carries the lighter Y-axis. In order to implement proposed algorithms, servo amplifiers are set to operate in torque (current) control mode. Closed loop control is implemented in the Dspace DS1103® real time control system by reading linear encoder feedback at a resolution of 0.1 µm and commanding torque signal to the servos at a closed loop sampling interval of T_s = 0.1 ms. Both X and Y drives are controlled by P-PI cascade motion controllers with velocity feedforward action. The position feedback control bandwidths of the axes are roughly matched at ω = 50 Hz n to ensure good motion synchronization and contouring [37].

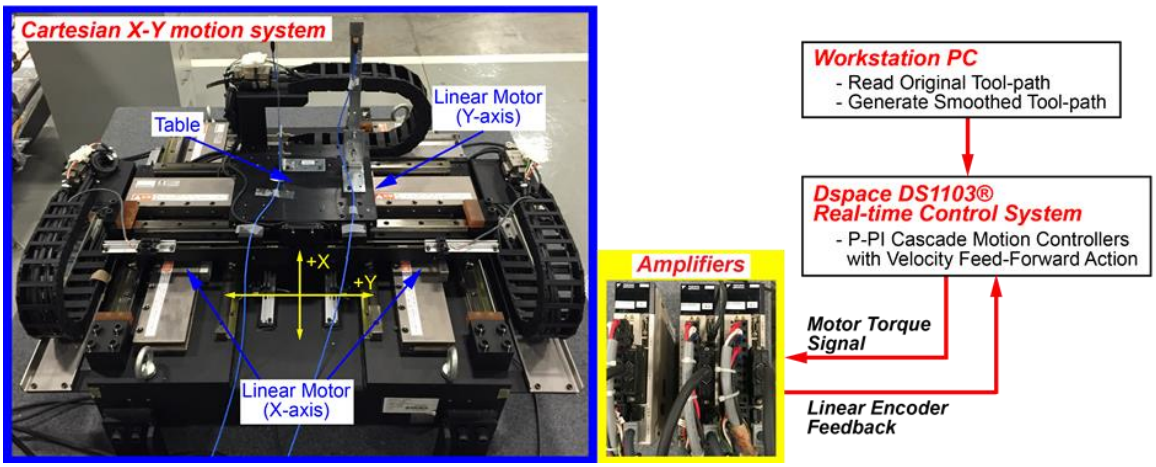

Figure 3.10: Experimental Planar Motion Platform.

Firstly, 3 different interpolation algorithms are implemented and compared experimentally on smoothing a "spiral shaped" short-segmented tool-path shown in Figure 3.11. A part from the first and last long linear moves, the spiral geometry is discretized densely with 1 mm long linear segments. The "local" and "global" corner smoothing algorithms from Sections 3.3.1 and 3.3.2 are implemented separately. These two interpolation algorithms are then compared to a point-to-point (P2P) interpolation technique, which demands a full stop at each corner. All algorithms are sampled and commanded in real-time to the servo controllers. The desired feedrate is set to F = 100 mm/s. Axis acceleration and jerk limits are fixed to A_{max} = 2000 mm/s^2 and J_{max} = 50,000 mm/s^3, respectively.

Figure 3.11 illustrates the tool-path geometry smoothened by “local” and “global” corner smoothing methods. The desired cornering tolerance is set to ε = 50 μm. Both, local and the global corner smoothing algorithms successfully smoothen the discrete path geometry and deliver a continuous motion. Nevertheless, the “local” corner-smoothing algorithm lowers the predefined cornering tolerance if the corners are close to each other. This is because local corner smoothing algorithm cannot interpolate overlapping corners and so it enforces some linear distance in-between corners. Cornering errors are reduced from the preset value down to ε = 0.4-13.3 μm based on the corner angle. On the other hand, the “global” corner interpolation algorithm generates smoothed cornering trajectory while mostly reaching the desired cornering tolerance.

Figure 3.12 presents the resultant kinematic profiles. As expected, the P2P motion strategy shows cyclic motion kinematics and delivers the longest cycle time. Due to short linear moves, commanded feedrate along the tool-path is never reached. Please note that this type of motion scheme is barely utilized in modern machine tools. It is implemented there to highlight the importance of corner smoothing. The local cornering technique, on the other hand, delivers a faster motion time. The acceleration profile is continuous, and limits of the drives are fully utilized at every corner. Due to the obtuse corner geometry, mostly jerk limits of the drives are saturated. Similar to the P2P interpolation scheme, feed profile is fluctuating and motion decelerates and accelerates in-between corners as it attempts to attain higher speeds. Finally, the proposed global corner smoothing technique is applied. As shown in Figure 3.12, the velocity profile is smooth and the fastest cycle time is achieved while respecting drive limits. It is also noticeable that the tangential feedrate is gradually decreasing along the spiral section. This decline in the speed is dictated by the spiral geometry having acute cornering angles towards the end of the path.

Figure 3.13 shows the experimentally measured contouring errors for the tool-path interpolated by each method. As expected, within all the 3 methods, the largest contour errors typically occur at cornering sections when drives undergo large acceleration, and attempt to overcome static friction in the system. Table 3.1 summarizes corresponding cycle times, contour errors, and computational times of each algorithm. The dramatic cycle time reduction, nearly ~45%, by the proposed global corner smoothing algorithm is

obvious. Nevertheless, both proposed global and local corner smoothing techniques deliver nearly identical contouring performance validating the potential for practical application of the developed technique. Lastly, computationally time of each method is compared in Table 3.1. All the 3 methods are implemented in Matlab environment on a PC with Intel i7 2 GHz clocked chipset running on Windows. As observed, when cycle times get longer, computational expense also increases. This is simply due to the fact that more number of samples needs to be interpolated. However, the unit computational expense for interpolating the global and local corner smoothing algorithms with the proposed LAW technique requires very comparable computational effort to local corner smoothing technique. This justifies that the proposed method can be implemented in modern NC systems conveniently.

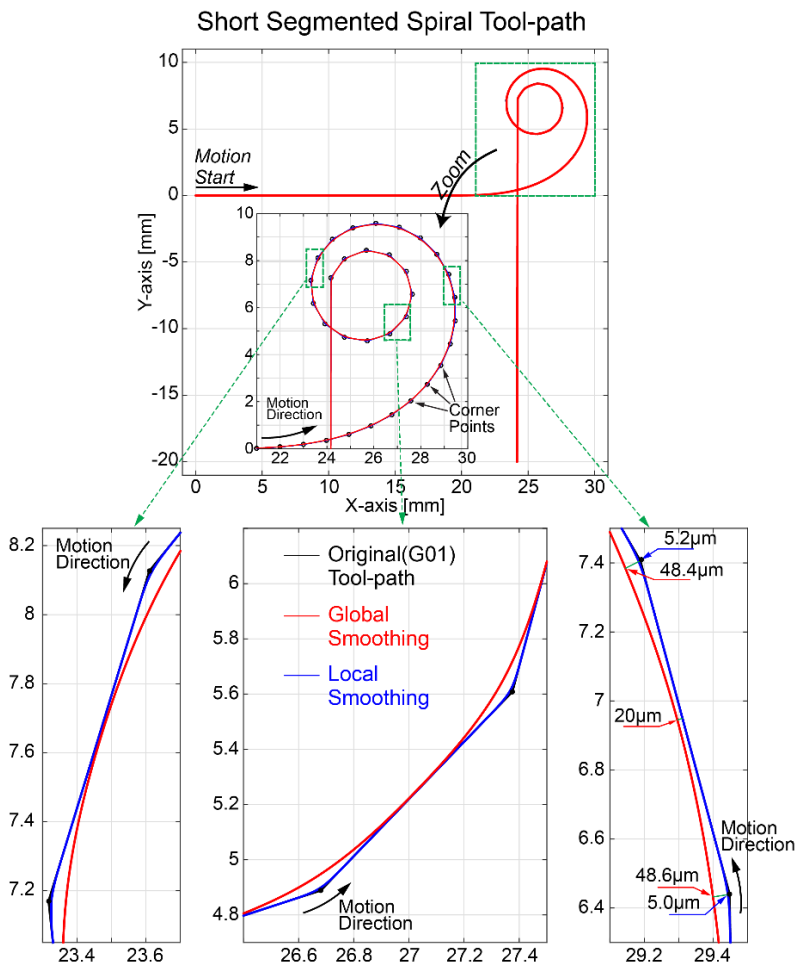

Figure 3.11: Short-segmented Spiral tool-path.

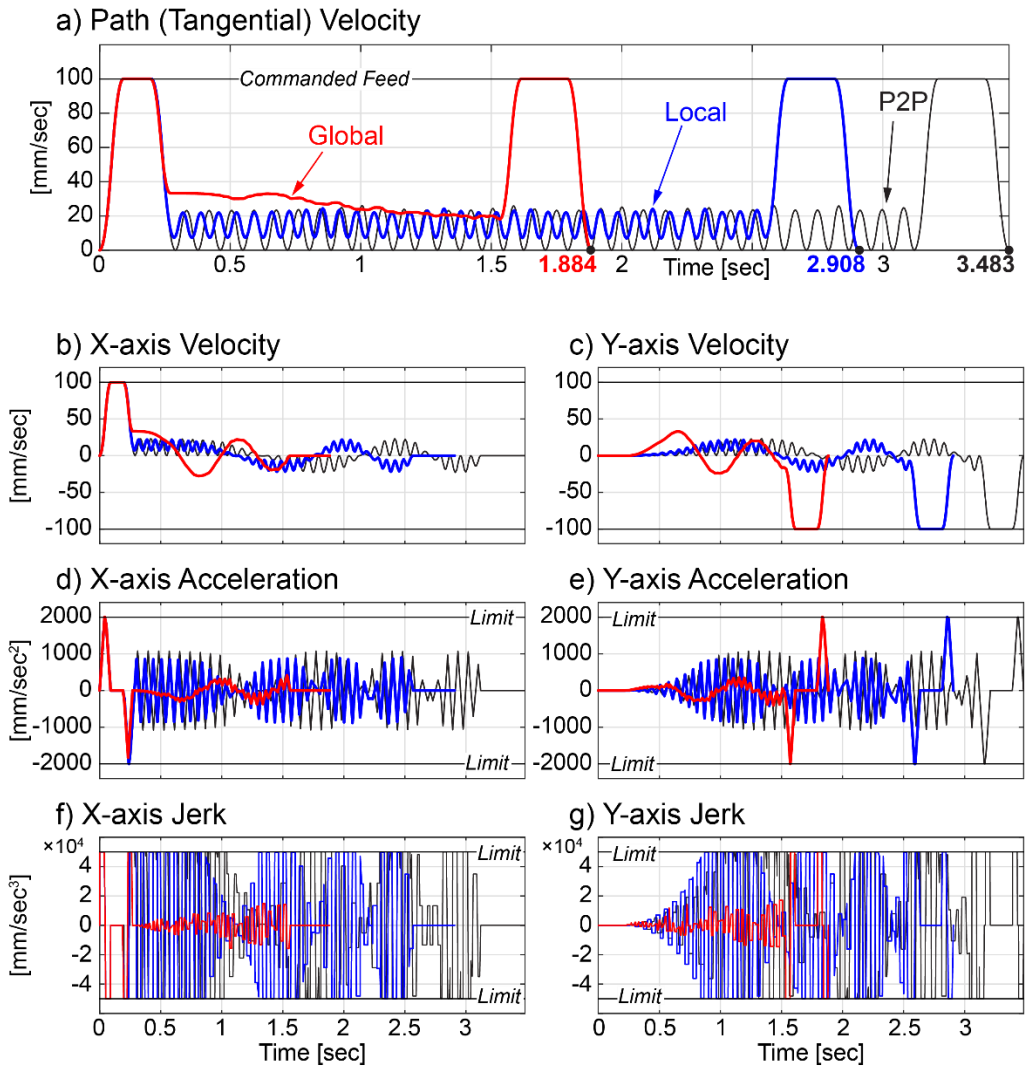

Figure 3.12: Kinematic profiles along spiral tool-path.

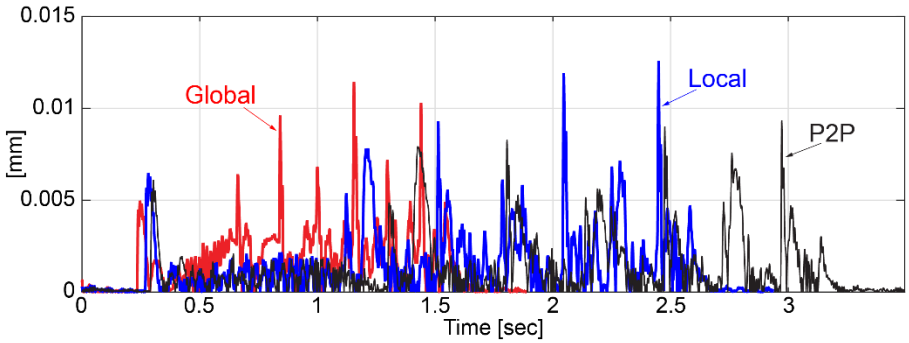

Figure 3.13: Experimentally recorded contouring errors.

Table 3.1: Cycle time and contouring performance comparison along spiral tool-path.

Algorithms	**Cycle Time [sec]**	**Cycle time Reduction in Percentage [%]**	**Contour Error [μm]**		**Computational Time [sec]**	**Unit Computational Effort (Total Computation Time/ Total Cycle Time)**
			RMS	Max		
Global (overlapping)	1.8840	45.9	2.5867	12.8730	0.6497	0.3449
Local	2.9075	16.5	2.2025	12.5822	0.8431	0.2900
Point-to-Point (P2P) interpolation	3.4832	Base	2.0071	9.3338	0.8925	0.2562

Finally, a more complicated tool-path in the shape of a "hearth" shown in Figure 3.14 is smoothened to benchmark contribution of the "global" corner smoothing technique. The tool-path consists of long linear segments with separated corners, and short (<2 mm) dense linear segments in the middle section. Motion direction also changes suddenly in the center of the tool-path. 4 Different methods are compared to each other; namely the proposed 1-step "global" corner smoothing method, the 1-step "FIR" filtering method [34], the "Bezier Spline" based 2-step local corner smoothing technique [8], and finally the basic P2P interpolation. The FIR filtering method is a 1-step method since it directly filters the reference tool-path. It is capable of controlling separated corner tolerances. But, it is not capable of accurately controlling cornering tolerance of overlapping sections. Therefore, FIR filter parameters are hand tuned by trial-and-error to make sure that the cornering tolerance is respected in the middle of the tool-path, both axis acceleration and jerk limits are respected, and a rapid feed motion is achieved. Similarly, the Bezier spline-based corner smoothing technique is only capable of controlling local cornering errors.

The cornering tolerance is set to ε = 100 μm for all the methods, and Figure 3.14 shows actual and smoothened, i.e. interpolated, tool-path geometries. As shown, the proposed method, FIR filtering and the Bezier spline techniques all smoothen "local" corners at ε = 100 μm tolerance. However, along short-segmented regions of the tool-path, Bezier method overrides the set cornering tolerance to generate feasible trajectories. Thus,

it locally smoothens the path, which in return elongates cycle times and generates interrupted motion with large fluctuation in acceleration and jerk profiles. The FIR filtering method cannot cope with short-segmented sections either. In this experiment, FIR filter parameters are iteratively tuned to make sure that the corner tolerance and axis kinematic limits are respected along the short-segmented section of the path. In fact, the FIR technique needs to be re-tuned different tool-paths.

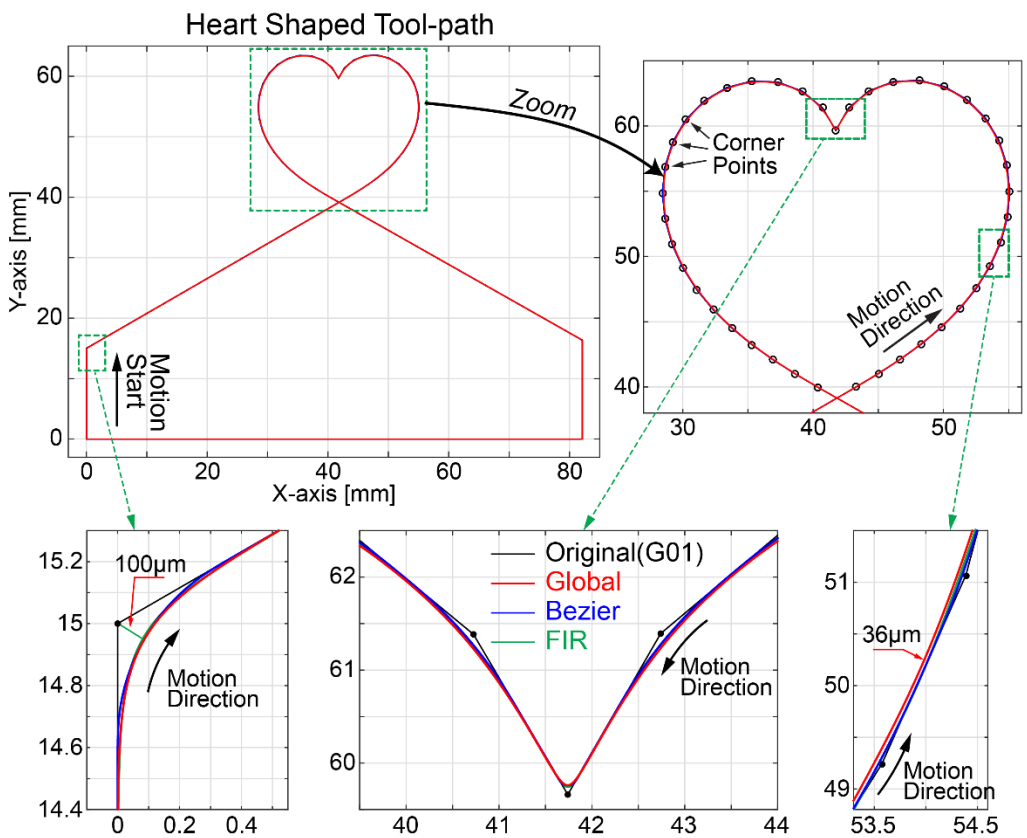

Figure 3.14: “Heart” Shaped Short-Segmented Tool-Path.

Figure 3.15 depicts the kinematic motion profiles of each method. The programmed feedrate is F = 150 mm/s, and axis acceleration and jerk limit are set to A_{max} = 2000 mm/s^2 and J_{max} = 50,000 mm/s^3, respectively. Figure 3.15a shows the path (tangential) velocity profiles. As shown, proposed global interpolation algorithm delivers the fastest cycle time. Observed from the jerk profiles, the motion profiles are smoother with less fluctuation in velocity, acceleration and jerk. Proposed method uses significantly

less jerk to interpolate along densely discretized toolpath section. This is mainly due to the fact that the proposed global smoothing technique can interpolate overlapping corners with uninterrupted acceleration profiles. Notice that Bezier technique exceeds jerk limits of the drives due to deficiencies in the real-time interpolation stated in the literature [4], [20] and mentioned in Section 3.1 of the manuscript. Similarly, the FIR filtering method exceeds jerk limits of the drives during short-segmented section of the tool-path. Both techniques deliver significantly more fluctuating feedrate profiles. This typically results in a rougher surface finish of the manufactured parts. The cycle time comparison is presented in Table 3.2. The proposed global smoothing technique can reduce cycle time almost by half as compared to the basic P2P type interpolation. It also delivers faster cycle time than Bezier and FIR filtering methods.

Finally, experimental contouring errors are presented in Figure 3.16. As shown, the root mean square (RMS) average errors (See Table 3.2) are very similar. The maximum contour errors are also comparable. The Bezier corner smoothing technique generates interrupted acceleration profiles. Hence, this adverse effect is particularly clear from contouring error fluctuations. Similar fluctuations are also observed for the FIR filtering technique. Proposed global corner smoothing technique delivers a less a fluctuating, smoother contouring error profile.

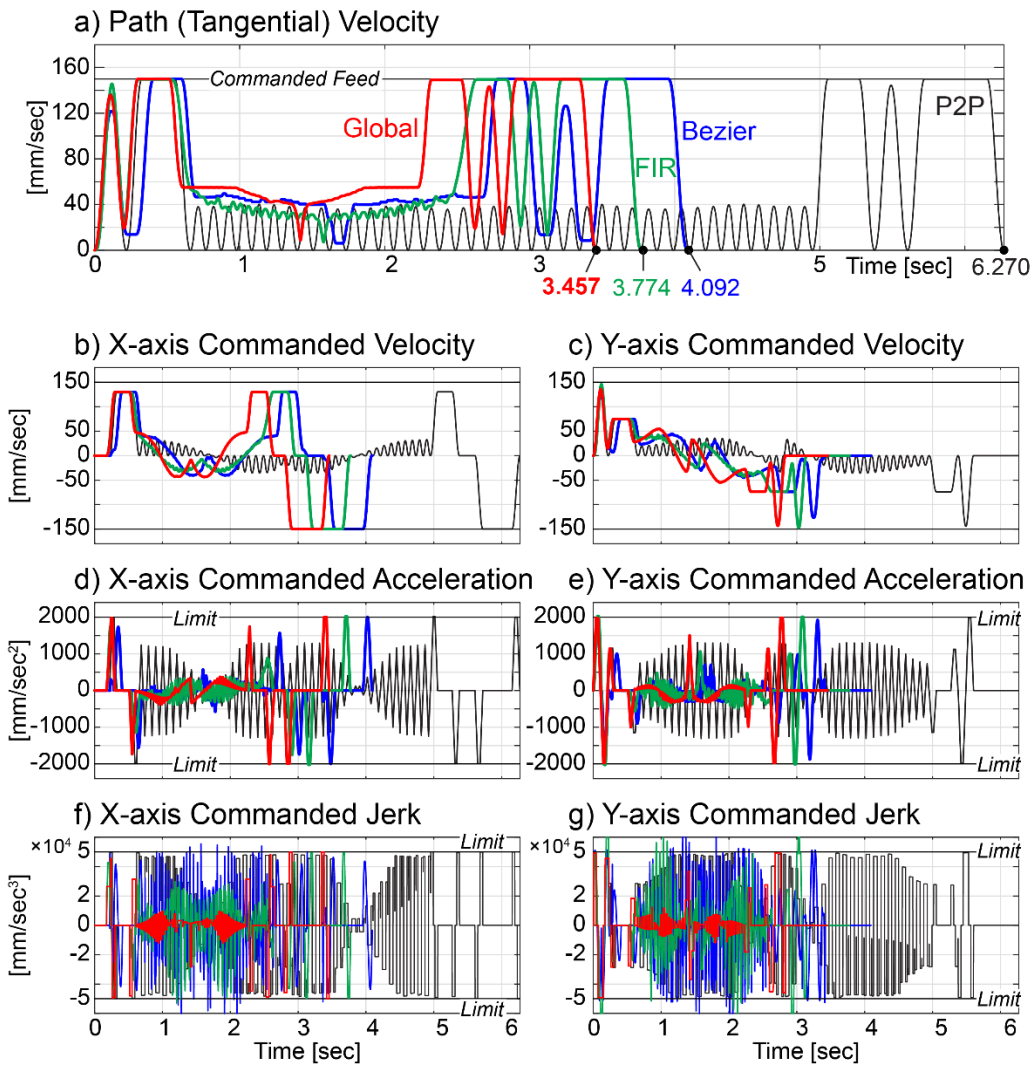

Figure 3.15: Kinematic profiles along heart shaped tool-path.

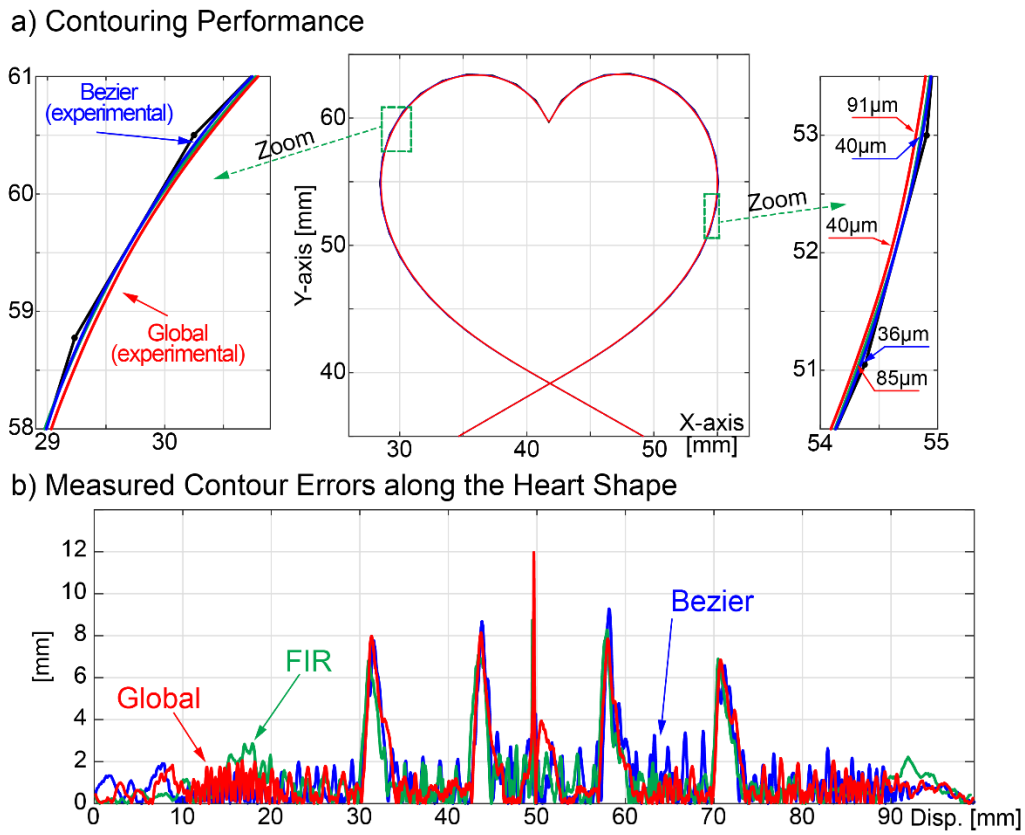

Figure 3.16: Contour error analysis.

Table 3.2: Cycle time and contouring performance comparison.

Algorithms	**Cycle time [sec]**	**Cycle time Reduction in Percentage [%]**	**Contour error [μm]**	
			RMS	Max
Global (overlapping) Interpolation	3.457	45.0	2.4001	12.7891
FIR Interpolation	3.774	39.8	2.1930	8.7736
Bezier (Local) Interpolation	4.092	34.7	2.1038	9.0324
P2P Interpolation	6.270	*Base*	2.1446	12.0504

3.6. Conclusions

This paper proposes a novel global corner smoothing and real-time interpolation technique along short short-segmented linear tool-paths. Corners are smoothed by interpolating axis motions continuously with uninterrupted acceleration profiles. This, in return, generates smooth and rapid motion along short-segmented tool-paths. Furthermore, the cornering transition is planned to be near time-optimal where either acceleration or jerk limits of the drives are saturated. A novel Look-Ahead Windowing (LAW) technique is developed to efficiently plan the feed motion. As compared to already existing techniques in the literature, the hereby-proposed algorithms are 1-step and analytical. Therefore, proposed algorithms are computationally efficient and convenient to implement in real-time on CNC machine tools. Furthermore, the proposed global corner-smoothing scheme considers overlapping corners in short-segmented tool-paths and generates smoother and at the same faster motion suitable for high speed machining. Experimental benchmarks along short segmented toolpaths show that the proposed algorithms can improve the cycle time up to ~45% as compared to P2P path interpolation and 10–15% compared to the existing techniques in the literature while still delivering identical or better contouring performance.

Accurate interpolation of machining tool-paths based on FIR filtering

Shingo Tajima, Burak Sencer and Eiji Shamoto

Precision Engineering

Volume 52, April 2018, Pages 332-344

4. Accurate interpolation of machining tool-paths based on FIR filtering

This paper presents a novel real-time (online) interpolation algorithm based on Finite Impulse Response (FIR) filters to generate smooth and accurate reference motion trajectories for machine tools and motion systems. Typically, reference tool-paths are composed of series of linear (G01) or circular (G02) segments. Basic point-to-point (P2P) feed motion can be generated by interpolating each segment with trapezoidal or S-curved velocity profile. However, smooth and accurate transitions between path segments are necessary to realize non-stop contouring motion. In this study, FIR filters are utilized, and the reference tool-path is filtered to interpolate a non-stop rapid feed motion. By using a chain of FIR filters, acceleration and jerk continuous motion profiles are generated from velocity pulse commands. A segment interpolation timing technique is developed to control the contour errors during non-stop real-time interpolation of tool-paths. Furthermore, by utilizing FIR filters for interpolation, frequency spectrum of the interpolated motion profiles is controlled. The time constant (delay) of the filter is tuned to create notches around the lightly damped vibration modes of the motion system, which allows mitigation of unwanted vibrations and thus enables delivering accurate feed motion. Simulation studies and industrial scale experimental validations are provided to illustrate effectiveness of the developed interpolation technique.

4.1. Introduction

Reference trajectory generation plays a key role in the computer control of machine tools and motion systems. Generated trajectories must not only describe the desired tool path accurately, but must also have smooth kinematic profiles in order to maintain high tracking accuracy, and avoid exciting natural vibration modes of the mechanical structure or servo control system. As a matter of fact, most machining tool-paths are defined in terms of series of linear (G01) segments or circular (G02) arcs [1], [46]. This imposes serious limitations in terms of delivering a non-stop smooth and rapid motion for productivity, and to achieve the desired final part geometry.

There are several challenges associated with interpolating a smooth motion along these discrete tool-paths. Consider interpolation on a single path segment; feedrate (tangential velocity) profile needs to be planned with smooth acceleration and decelerations to avoid excitation of the machine tool's structural modes [36] and at the same time respect kinematic limits, i.e. torque, acceleration and jerk, of the drives [30], [47], [48]. Polynomial based feed profiles, such as trapezoidal velocity [46], acceleration [30] and jerk profiles [49] are well-known to the machine tool literature. They can be planned to fully exploit machine limits and generate time-optimal feed motion along predetermined paths [10], [47], [48]. However, these methods suffer from two bottlenecks. Firstly, they don't provide any quantitative means to control the frequency spectrum of the interpolated acceleration commands. In practice, the jerk limit is used to mitigate any residual vibrations [36]. Note that, tuning the jerk limit smoothens acceleration profile. But, it does not directly control the frequency spectrum. Robotics literature adapted exponential [50], trigonometric [51] or minimum jerk spline [48] based acceleration profiles to help attenuate frequency spectrum of reference trajectories. In precision machine tool literature, input shapers (IS) [52], [53] and notch filtering are utilized to filter the reference motion commands to attenuate the excitation around the lightly damped resonant frequencies of the machine. These techniques are easy to implement and robust against parameter variations [52], which makes them suitable for practice. However, input shaping distorts interpolated tool trajectories due to shaper dynamics and induces interpolation errors. Either machining velocity (feedrate) has to be lowered to reduce those errors, which is widely employed in practice; or, model-based compensation techniques that are mostly computationally costly are proposed in the literature [54].

Another bottleneck is the computational load of reference trajectory generators. As the degree and complexity of the acceleration profile becomes higher, computational cost to plan polynomial based trajectory generation increases [55]. Recent efforts are directed towards generating online, real-time suitable interpolation techniques [56]–[58]. These approaches essentially utilize a tuned dynamic system to filter and smoothen velocity/displacement commands. They are designed with a chain of integrators and cascaded feedback loops [59]. Online path smoothers can be implemented in the form of

recursive difference equations. To attain time-optimal motion, nonlinear feedback elements such as saturation blocks are also introduced [60], [61]. Nevertheless, unless combined with an input shaper, methods cannot control frequency spectrum of generated trajectories.

Finite Impulse Response (FIR) filters provide a computationally efficient framework for online trajectory generation. The use of FIR filtering for real-time interpolation and trajectory generation is known to the machine tool literature [62]. Chain of *1st* order FIR filters can be used to generate smooth reference trajectories with trapezoidal acceleration and jerk profiles [34], [43]. Time constants of filters can be assigned to realize time optimal motion. Furthermore, frequency response of the filter can be tuned so that the excitation of the reference trajectory is shifted away from the resonances of the machine tool. Finally, it can be implemented as a moving average filter on modern microprocessors with minimum computational effort [43].

Although FIR filtering is an effective technique for online interpolation of reference trajectories, so far, its use is constrained in simple point-to-point (P2P) moves. If consecutive moves, e.g. linear or circular segments, are interpolated continuously without a full stop at the segment junctions, large interpolation contouring errors occur due to sudden change in the feed direction and the dynamics of the filter. Unless these contour errors are confined, the use of FIR filtering for generating uninterrupted, rapid and accurate feed motion in precision motion systems is limited. Recent literature recognized these shortcomings and proposed compensation techniques [34], [35], [54]. However, these techniques are either computationally expensive because they need to estimate errors through dynamic models. This greatly limits their application in real-time implementation. Or, they consider contouring errors only around junction of linear segments [34], which is not realistic since conventional machining tool-paths consist of both mixture of linear and circular segments, and transitions in-between those segments must be considered for a non-stop high-speed contouring motion.

This paper, for the first time, presents comprehensive interpolation techniques for generating uninterrupted and accurate feed motion along multi-segmented machining tool-paths based on FIR filtering. Contributions of the paper are laid out as follows. Section 4.2

first analyses high-order trajectory generation based on FIR filtering technique. It is followed by the introduction of accurate interpolation of linear and circular paths. A feedrate control technique is presented to control contour errors during interpolation of circular paths. Section 4.3 presents online interpolation of multi-segmented toolpaths based on FIR filtering. Dwell time control technique is presented to control the interpolation errors that occur during non-stop transition between linear and circular segments. Finally, Sections 4.4 and 4.5 present illustrative examples and rigorous experimental validations along complex tool-paths.

4.2. Online trajectory generation based on FIR filtering

4.2.1. Generation of high order kinematic profiles

Typically, “trapezoidal acceleration” or “trapezoidal jerk” based feed profiling is employed to generate reference trajectories for highspeed and precision motion systems [30], [49]. This section outlines the basic methodology to generate high-order trajectories utilizing a chain of FIR filters [43].

A 1st order FIR filter is defined in Laplace (s) domain by the following transfer function [43], [63]:

$$M_i(s) = \frac{1}{T_i}\frac{1-e^{-sT_i}}{s}, \quad i = 1...N \tag{4.1}$$

where T_i is the time constant (delay) of the ith FIR filter. Observed from Eq. (4.1), a FIR filter consists of an integrator ($1/s$) and a pure delay e^{-sTi} resembling a simple moving average filter [63]. The impulse response is evaluated by taking inverse Laplace transform of Eq. (4.1) as:

$$m(t) = L^{-1}\ M_i(s)\ = \frac{u(t)-u(t-T_i)}{T_i} \text{ where } u = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases} \tag{4.2}$$

and as shown in Figure 4.1, it becomes a simple rectangular pulse with a duration of T_i having a magnitude of $1/T_i$. This implies that for any $T_i > 0$, the area underneath the impulse response is unitary. As a result, when an arbitrary signal is convolved with the FIR filter,

area underneath the original signal does not alter. Furthermore, since the filter has a free integrator (1/*s*) it increases degree of the filtered (convolved) signal. This property can be used to generate high order real-time motion trajectories as follows.

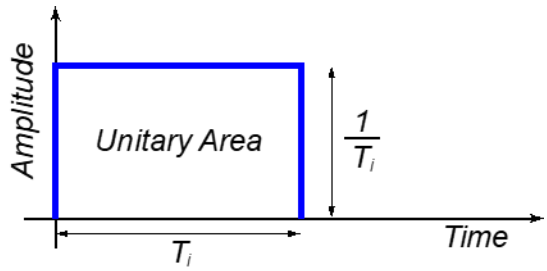

Figure 4.1: Impulse response of a 1st order FIR filter.

Let us consider a simple linear movement for a length of L commanded at a velocity of F. This trajectory can be commanded by a rectangular pulse for a duration of $T_v = L/F$ as shown in Figure 4.2a. FIR filtering this rectangular velocity pulse generates the well-known trapezoidal velocity profile [46] with piecewise constant acceleration segments (See Figure 4.2b). Subsequently, another FIR filter can be convolved with the trapezoidal velocity profile to generate smoother trapezoidal (jerk limited) acceleration profile [30]. As outlined in Figure 4.2, high-order reference kinematic profiles can be generated by filtering a reference velocity pulse though chain (series) of FIR filters [34], [43]. Finally, the resultant velocity profile is integrated to obtain reference displacement profile as shown in Figure 4.2c.

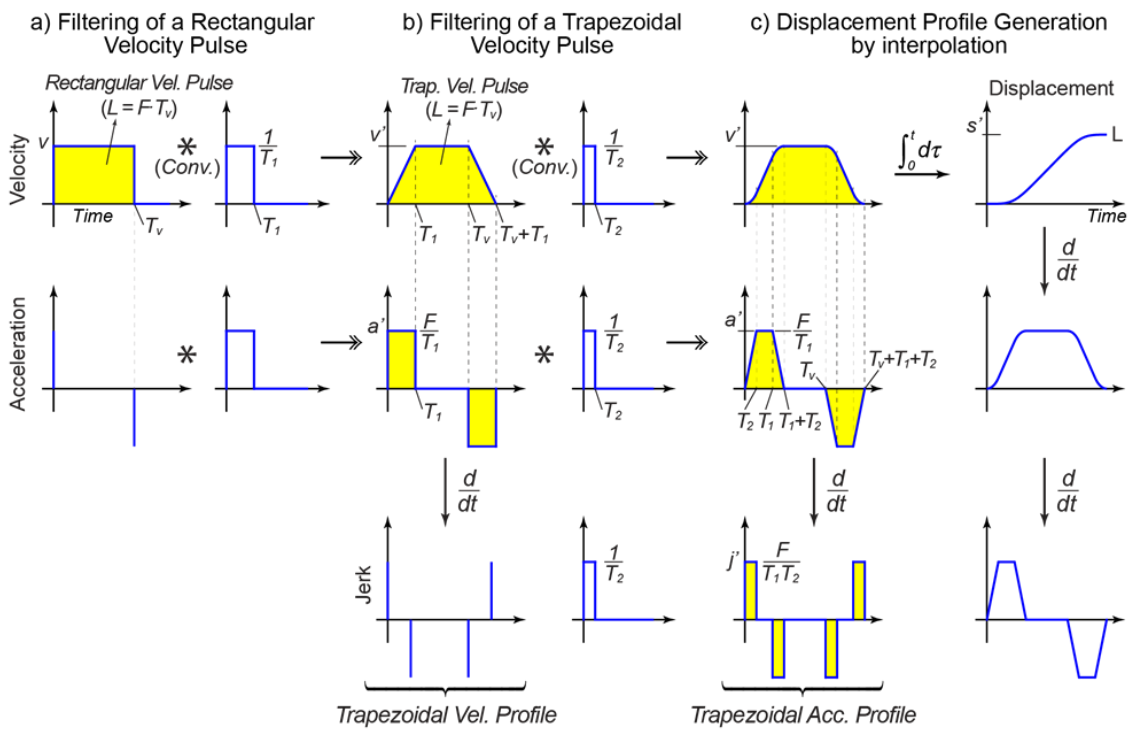

Figure 4.2: FIR filtering based smooth trajectory generation.

Filtered kinematic profiles can be analyzed through analytical solution of convolution [63]. Consider a simple trapezoidal velocity profile. Convolution of a rectangular velocity pulse command $v(t)$, with the impulse response of the FIR filter from Eq. (4.2) is written as:

$$\begin{aligned} v'(t) = v(t) * m(t) &= \frac{1}{T_1}\int_0^t \Big(\big[v(\tau)-v(\tau-T_v)\big]\big[u(t-\tau)-u(t-T_1-\tau)\big]\Big)d\tau \\ &= \frac{1}{T_1}\begin{bmatrix} \int_0^t v(\tau)u(t-\tau)d\tau - \int_0^t v(\tau)u(t-T_1-\tau)d\tau \\ -\int_0^t v(\tau-T_v)u(t-\tau)d\tau + \int_0^t v(\tau-T_v)u(t-T_1-\tau)d\tau \end{bmatrix} \end{aligned} \tag{4.3}$$

and the filtered velocity signal $v'(t)$ is derived by evaluating above integrals,

$$v'(t) = \begin{cases} (F/T_1)t & , \ 0 \le t < T_1 \\ F & , \ T_1 \le t < T_v \\ F/T_1(-t+T_v+T_1) & , \ T_v \le t < T_v + T_1 \\ 0 & , \ T_v + T_1 \le t \end{cases} \tag{4.4}$$

where the resultant acceleration signal is obtained by differentiation as:

$$a'(t) = \begin{cases} F/T_1 & , \ 0 \le t < T_1 \\ 0 & , \ T_1 \le t < T_v \\ -F/T_1 & , \ T_v \le t < T_v + T_1 \end{cases} \tag{4.5}$$

Finally, smooth displacement profile $s'(t)$ is generated by integration of the velocity profile.

The use of convolution enables analytical derivation of filtered profile kinematics. Above trapezoidal velocity profile is derived for the case of $T_v > T_1$ and also illustrated in Figure 4.3a. The peak acceleration depends on the filter's time constant and the commanded velocity, $A_{peak} = F/T_1$. Commanded velocity F is reached at filter's time constant, $t = T_1$, and the remaining cruise velocity duration becomes $T_v - T_1$. On the other hand, when the reference velocity pulse duration is equal or shorter than the filter's time delay $T_v \le T_1$, motion kinematics alters. As depicted in Figure 4.3b, when $T_v = T_1$, no velocity cruise section occurs. As opposed, Figure 4.3c illustrates the case of $T_v < T_1$. In this case, commanded velocity cannot be reached and peak velocity becomes $V_{peak} = L/T_1$ with a cruise phase duration of $T_1 - T_v$. Although omitted here, convolution can be used to analytically derive all these kinematic profiles. The overall motion duration is elongated by the amount of filter's time constant $T_v + T_1$.

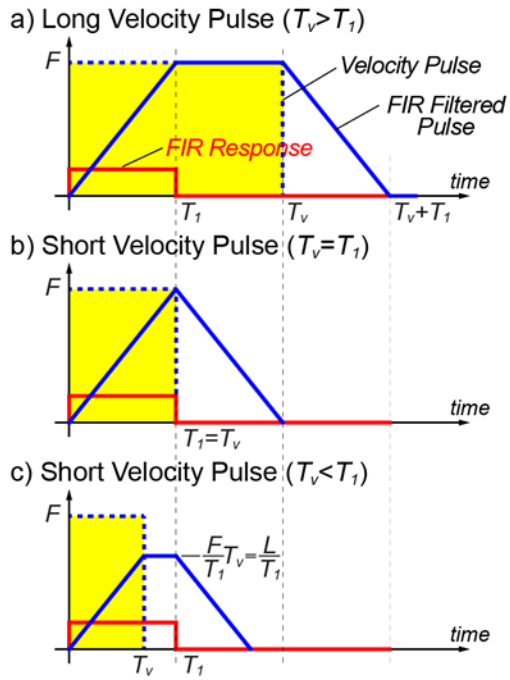

Figure 4.3: Trapezoidal velocity profile generated by single FIR filter.

In precision motion systems jerk [30] or even snap [49] limited velocity profiles are favored to generate smoother, more traceable reference motion profiles. As illustrated in Figure 4.2, utilizing 2 FIR filters with time constants T_1 and T_2 generates the well-known trapezoidal acceleration (jerk limited) feed profile. Figure 4.4 illustrates the motion profile generated by filtering a trapezoidal velocity pulse. The profile kinematics can be computed analytically by replacing rectangular velocity pulse command with the trapezoidal one in Eq. (4.3), and for the case of $T_v > T_1 > T_2$, the generated velocity profile is derived as:

$$v'(t)=\begin{cases}\frac{1}{2}\frac{F}{T_1T_2}t^2 & 0\le t<T_2\\ \frac{1}{2}\frac{FT_2}{T_1}+\frac{F}{T_1}\left(t-T_2\right) & T_2\le t<T_1\\ F-\frac{1}{2}\frac{F}{T_1T_2}\left(\left(T_1+T_2\right)-t\right)^2 & T_1\le t<T_1+T_2\\ F & T_1+T_2\le t<T_v\\ F-\frac{1}{2}\frac{F}{T_1T_2}\left(t-T_v\right)^2 & T_v\le t<T_v+T_2\\ F-\frac{1}{2}\frac{FT_2}{T_1}-\frac{F}{T_1}\left(t-\left(T_v+T_2\right)\right) & T_v+T_2\le t<T_v+T_1\\ \frac{1}{2}\frac{F}{T_1T_2}\left(\left(T_v+T_1+T_2\right)-t\right)^2 & T_v+T_1\le t<T_v+T_1+T_2\\ 0 & T_v+T_1+T_2\le t\end{cases} \tag{4.6}$$

and the corresponding filtered acceleration profile $a'(t)$ becomes:

$$a'(t)=\begin{cases}\frac{F}{T_1T_2}t & 0\le t<T_2\\ \frac{F}{T_1} & T_2\le t<T_1\\ \frac{F}{T_1}-\frac{F}{T_1T_2}\left(t-T_1\right) & T_1\le t<T_1+T_2\\ 0 & T_1+T_2\le t<T_v\\ -\frac{F}{T_1T_2}\left(t-T_v\right) & T_v\le t<T_v+T_2\\ -\frac{F}{T_1T_2}\left(t-T_v\right) & T_v\le t<T_v+T_2\\ -\frac{F}{T_1} & T_v+T_2\le t<T_v+T_1\\ -\frac{F}{T_1}+\frac{F}{T_1T_2}\left(t-\left(T_v+T_1\right)\right) & T_v+T_1\le t<T_v+T_1+T_2\end{cases} \tag{4.7}$$

As observed from Eq. (4.6), filtered (interpolated) motion profile is determined by the filter delays, T_1 and T_2, and the length of the reference velocity pulse, T_v. There are in total 8 combinations. Owing to the linearity of filtering operation, the order of filters in the

chain does not matter, and 3 main cases characterize the interpolated profiles; namely, $T_v > T_1 + T_2$, $T_v = T_1 + T_2$, or $T_v < T_1 + T_2$. Figure 4.4a depicts the most common case when the velocity pulse is longer than total sum of filter time constants, $T_v > T_1 + T_2$ and $T_1 > T_2$. Note that in this case, a full 7-segmented jerk limited acceleration profile [30] with cruise phase can be generated (Eq. (4.6)). However, if a rapid (high speed) move on a short travel distance is commanded, T_v may become smaller. For instance, cruise velocity phase may disappear completely if $T_v = T_1 + T_2$ as shown in Figure 4.4b. Furthermore, if $T_v < T_1 + T_2$, commanded path velocity cannot be reached. In this case, peak velocity is computed as $V_{peak} = L/T_1$ (See Figure 4.4c). The total motion duration is elongated by the filter delay as, $T_v + T_d$, where $T_d = T_1 + T_2$.

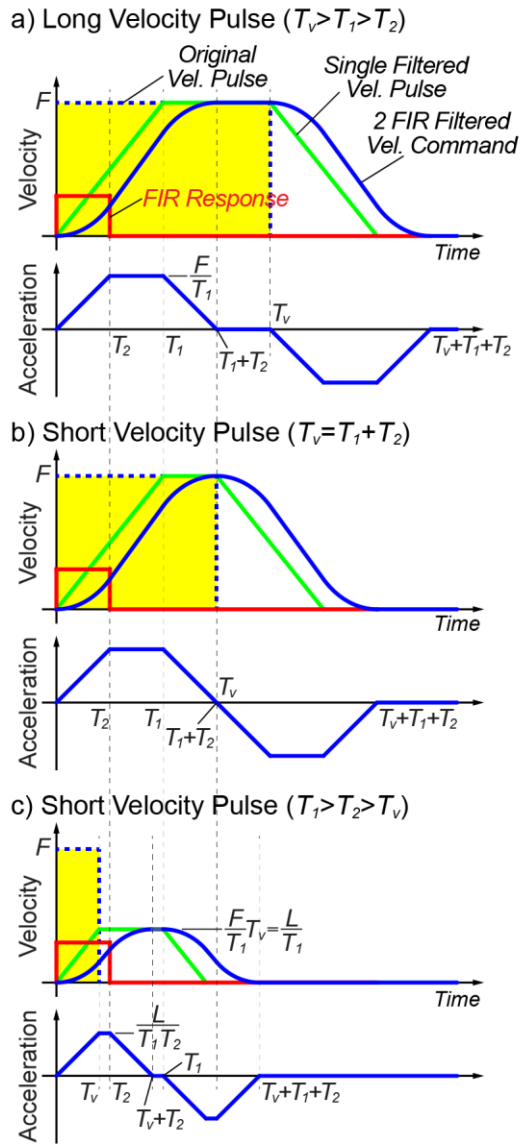

Figure 4.4: Trapezoidal acceleration profile generated by 2 FIR filters.

4.2.2. Frequency shaping of interpolated trajectories

The FIR filter structure also provides effective means to control frequency spectrum of the generated trajectories. Filtered acceleration $a'(t)$ profile controls the torque/force delivered by the feed drive, which induces excitation to the overall motion system. If frequency spectrum of the reference acceleration profile contains components near the

lightly damped structural modes of the machine structure, it initiates forced vibrations [52], [54].

For a rectangular pulse velocity input, the acceleration command consists of set of impulses separated by Tv (See Figure 4.2a). Consider only a single acceleration impulse with a magnitude *h*

$$a(t) = \begin{cases} h, & t = 0 \\ 0, & t \neq 0 \end{cases} \tag{4.8}$$

convolved with the chain of FIR filters; the frequency spectrum of resultant acceleration profile simply becomes frequency response of the FIR filters in the chain, evaluated as:

$$a'(j\omega) = hM_1(j\omega)M_2(j\omega)...M_n(j\omega) \tag{4.9}$$

and frequency (ω) response of a single FIR filter can be computed from Eq. (4.1) as:

$$M_i \quad j\omega \quad = \frac{1}{T_i}\frac{1-e^{-j\omega T_i}}{j\omega} \rightarrow \left|M_i \quad j\omega\right| = \frac{\sin\left(\frac{\omega T_i}{2}\right)}{\frac{\omega T_i}{2}} \tag{4.10}$$

Consequently, frequency spectrum of the acceleration profile becomes multiplication of sinc [64] functions from Eq. (4.10) as:

$$\left|a' \quad j\omega\right| = \prod_{i=1}^{N}\left|\frac{\sin\left(\omega T_i/2\right)}{\omega T_i/2}\right| = \prod_{i=1}^{N}\left|\frac{\sin\left(\pi\omega/\omega_i\right)}{\pi\omega/\omega_i}\right|, \quad \text{where} \quad \omega_i = \frac{2\pi}{T_i} \tag{4.11}$$

The above property can be exploited to choose time constant of the FIR filter to avoid exciting lightly damped structural frequencies of the machine tool. Every *sinc* function creates periodic notches (ripples), which can be matched with the resonant frequency of the motion system by setting,

$$\omega_i = \frac{\omega_r}{k} \leftrightarrow T_i = k\frac{2\pi}{\omega_r} \tag{4.12}$$

An example is presented in Figure 4.5. Simply setting time constant of the filters in the chain to the natural periods of the resonant modes $k = 1$, $T_i = 2\pi/\omega_r$ introduces shortest

filter delay into the motion while avoiding excitation of resonances. It is also notable that there is close resemblance between Input Shaper [52], [53] and the FIR filter. Input Shapers have the property to cancel any vibration at the half of the vibration period, π/ω_r.

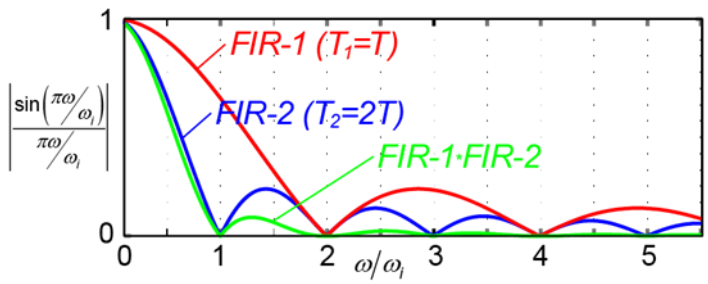

Figure 4.5: Frequency response of FIR filter.

4.2.3. FIR based interpolation of linear and circular paths

4.2.3.1. Linear Interpolation

Interpolation of single axis motion based on FIR filtering is presented in the previous sections. This technique can be extended to generate point-to-point (P2P) multi-axis linear motion. Figure 4.6 outlines the process to interpolate planar P2P linear motion between two points, $P_s = [x_s, y_s]^T$ and $P_e = [x_e, y_e]^T$. Firstly, the path length is computed from the Euclidean distance, $L = \|P_e - P_s\|$, and the tangential feed pulse F for a duration of $T_v = L/F$ is generated as shown in Figure 4.6b. The feed pulse is dissolved into its Cartesian velocity pulse components based on the path geometry,

$$\left.\begin{aligned} v_x(t) &= v(t)\frac{x_e - x_s}{L} = v(t)\cos\ \alpha \\ v_y(t) &= v(t)\frac{y_e - y_s}{L} = v(t)\sin\ \alpha \end{aligned}\right\} \text{ where } \alpha = \tan^{-1}\left(\frac{y_e - y_s}{x_e - x_s}\right) \qquad (4.13)$$

and smooth axis velocity commands are interpolated by applying FIR filtering. Note that, time constants of the filters are set identical so that the resultant motion is coordinated. Finally, the interpolated axis velocity commands are integrated to interpolate position commands.

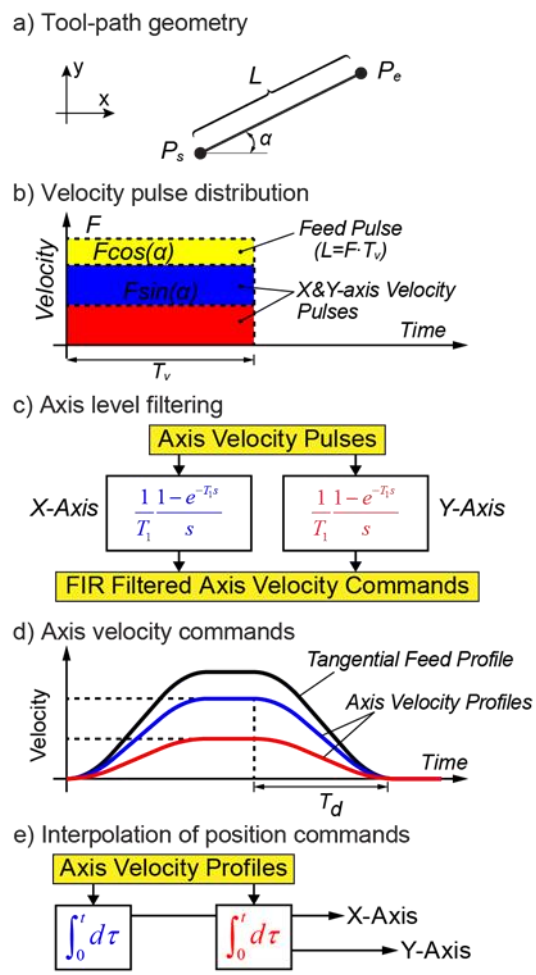

Figure 4.6: Multi-axis interpolation based on FIR filtering.

4.2.3.2. Circular interpolation

Next, the approach for linear interpolation is adapted to interpolate circular paths. During circular interpolation, the total travel length becomes the arc length of the circular path $L = R\Delta\theta$ where R is the arc radius and $\Delta\theta = \theta_e - \theta_s$ is the difference between starting and ending angular positions (See Figure 4.7). Omitting the arc center, rectangular feed pulse F is dissolved into its axis components based on the circular geometry as:

$$\underbrace{\left\{\begin{aligned} v_x(t) &= -F\sin\left(\frac{F}{R}t+\theta_s\right) \\ v_y(t) &= F\cos\left(\frac{F}{R}t+\theta_s\right) \end{aligned}\right\}}_{\text{Velocity Commands}} \text{ and } \underbrace{\left\{\begin{aligned} s_x(t) &= R\cos\left(\frac{F}{R}t+\theta_s\right) \\ s_y(t) &= R\sin\left(\frac{F}{R}t+\theta_s\right) \end{aligned}\right\}}_{\text{Position Commands}} \tag{4.14}$$

Notice that as opposed to the linear motion, reference axis velocity commands during circular interpolation are not in rectangular pulse form but rather sinusoidal signals. The reference axis commands are then filtered through chain of FIR filters and integrated to interpolate the circular path. Figure 4.7 illustrates the reference and interpolated velocity commands during circular interpolation.

Notice that sinusoidal axis motion commands are generated at the rotational frequency of $\omega_c = F/R$, and they are modulated by the frequency response of FIR filters. At steady state, the filtered axis motion commands can be written from Eqs. (4.14) and (4.1) as:

$$\left.\begin{aligned} s'_x(t) &= R\left|G_{FIR}\right|_{\omega=\omega_c}\cos\left(\frac{F}{R}t+\theta_{init}\right) \\ s'_y(t) &= R\left|G_{FIR}\right|_{\omega=\omega_c}\sin\left(\frac{F}{R}t+\theta_{init}\right) \end{aligned}\right\} \tag{4.15}$$

where $G_{FIR} = \prod_{i=1}^{N} M_i(j\omega)$ is frequency response function of FIR filter. The discrepancy between reference and interpolated (filtered) circular motion commands result in an interpolation contour error ε as shown in Figure 4.7. This contour error is measured normal to the commanded circle, and its steady state value can be calculated from Eqs. (4.14) and (4.15) as:

$$\begin{aligned} \varepsilon &= \sqrt{\left(s_x - s'_x\right)^2 + \left(s_y - s'_y\right)^2} \\ &= \sqrt{\left(R\left(1-G_{FIR}\right)\cos\left(\frac{F}{R}t+\theta_s\right)\right)^2 + \left(R\left(1-G_{FIR}\right)\sin\left(\frac{F}{R}t+\theta_s\right)\right)^2} \\ &= R\left(1-G_{FIR}\right) \end{aligned} \tag{4.16}$$

The steady state value of the contour error is controlled by magnitude of the frequency response of the FIR filter at the fundamental frequency of the circular motion,

ω_c. By lowering the feedrate F, excitation frequency can be altered, and ε can be confined by a user-specified tolerance value. Without losing generality, let us consider a trapezoidal acceleration profile generated by 2-FIR filters. The magnitude of the frequency response at ω_c can be evaluated from Eq. (4.11) as:

$$\left|G_{FIR}\right|_{\omega=\omega_c} = \left| \frac{\sin\left(\frac{T_1 F}{2R}\right)}{\frac{T_1 F}{2R}} \frac{\sin\left(\frac{T_2 F}{2R}\right)}{\frac{T_2 F}{2R}} \right| \tag{4.17}$$

4-term Taylor expansion can be applied to Eq. (4.17) to obtain a polynomial expression:

$$\left|G_{FIR}\right| \approx \left(1 - \frac{\left(T_1\omega_c/2\right)^2}{3!} + \frac{\left(T_1\omega_c/2\right)^4}{5!}\right)\left(1 - \frac{\left(T_2\omega_c/2\right)^2}{3!} + \frac{\left(T_2\omega_c/2\right)^4}{5!}\right) \tag{4.18}$$

and substituting Eq. (4.18) into Eq. (4.16) yields the relationship between the contouring error and the feedrate as:

$$\begin{aligned} &\varepsilon - R\left(1 - \left|G_{FIR}\right|_{\omega=\omega_c}\right) = 0 \\ &\Rightarrow x^4 - 80\left(\frac{1}{T_1^2} + \frac{1}{T_2^2}\right)x^3 + 640\left(\frac{10}{T_1^2 T_2^2} + \frac{3}{T_1^4} + \frac{3}{T_2^4}\right)x^2 \\ &\quad - 153600\left(\frac{1}{T_1^4 T_2^2} + \frac{1}{T_1^2 T_2^4}\right)x + 3686400\frac{\varepsilon}{T_1^4 T_2^4 R} = 0 \end{aligned} \tag{4.19}$$

where

$$x = \omega_c^2 = \left(\frac{F}{R}\right)^2 \tag{4.20}$$

Eq. (4.19) is a *4th* order polynomial whose roots can be solved conveniently in real-time, and the maximum feedrate to bound the contour error by a tolerance value can be obtained from Eq. (4.20). It should be noted that the polynomial approximation used in Eq. (4.18) only approximates |GFIR| at low frequency; namely, below the first notch (ripple) of FIR filters. The first notch is typically matched with one of the structural resonances to mitigate vibrations [34], [46], [53] and to minimize overall filter delay. Hence, the 4-term Taylor expansion is suitable for practice.

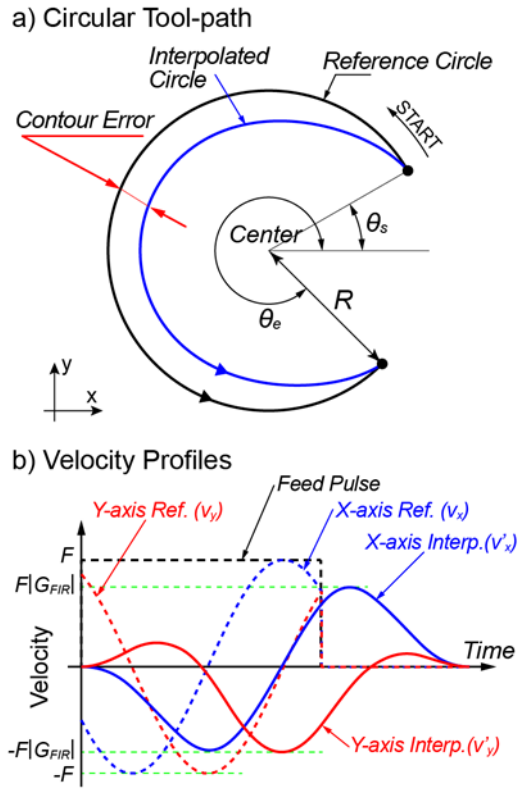

Figure 4.7: Circular interpolation and corresponding velocity profiles.

4.3. Multi-segmented tool-path interpolation strategy

Previous section discussed generation of basic linear and circular trajectories based on FIR filtering of axis velocity pulse commands. This section focuses on continuous and accurate interpolation of multi-segmented tool-paths.

Figure 4.8 presents the overall strategy for online FIR based interpolation of multi-segmented NC tool-paths. Each segment of the tool-path in a NC block/G-code is represented by a timed feed pulse. As introduced in the previous section, depending on the interpolation type, e.g. linear or circular, tangential feed pulse is dissolved into its axis velocity pulses, which are then filtered through chain of FIR filters to generate smooth axis reference motion commands. The NC part program can be interpolated based on two types of motion; namely, “point-to-point” (P2P), or non-stop “contouring”. The motion type can

be controlled by adjusting the "dwell" time between consecutive feed pulses (See Figure 4.8). For instance, in P2P mode, an instantaneous stop between the programmed segments is desired. This kind of motion strategy is typically employed in pick-and-place operations, ultra-precision machining and measurement. P2P motion can be achieved by simply accounting for the FIR filter delay and adding a *dwell* time between consecutive feed pulses that is equal to the filter delay Td as outlined in Figure 4.8. Figure 4.9a shows an example P2P motion generated along 2 linear segments. Consecutive feed pulses are filtered with a dwell time of $T_d = T_1 + T_2$ (in case of 2 FIR filters) to generate a full stop at P_1 and the corresponding velocity profile is shown in Figure 4.9b.

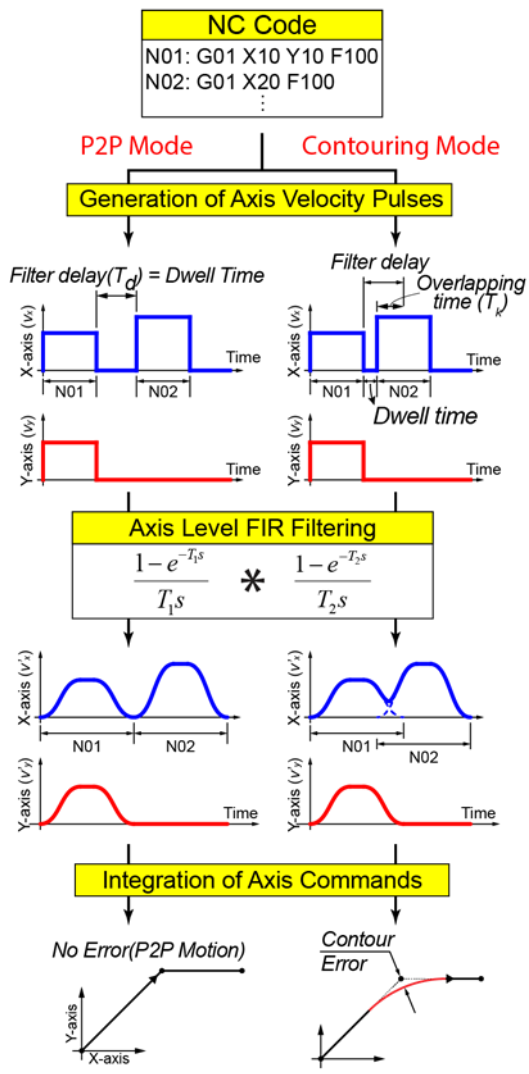

Figure 4.8: Overall path interpolation strategy.

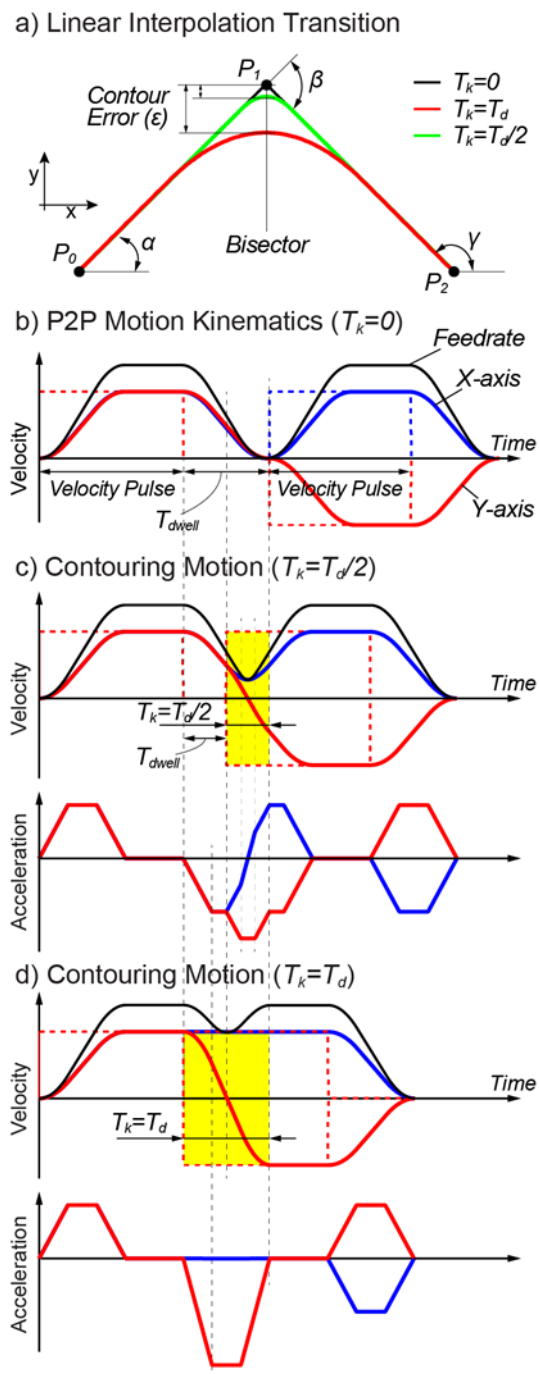

Figure 4.9: Kinematic profiles during contouring motion.

4.3.1. Contour error control during non-stop linear interpolation

On the other hand, in high-speed machining, uninterrupted accurate contouring motion is desired. As presented in Figure 4.8, non-stop "*contouring*" motion can be generated by interpolating consecutive feed pulses without fully waiting for the filter delay to die out. This enforces convolution of consecutive feed pulse to begin with non-zero initial conditions, and through precise control of the dwell time, contouring errors [23], [65] along segment transitions can be confined.

Firstly, let us define an "overlapping time", T_k, to control the *overlap* of convolution of the consecutive feed pulses:

$$0 \leq T_k \leq T_d \tag{4.21}$$

If $T_k = 0$, the *dwell* time is equal to the total filter delay T_d, and as shown in Figure 4.9a and b, a P2P motion is generated. Figure 4.9c illustrates the case when $T_k = T_d/2$. In this case, consecutive segment interpolation is initiated before feed motion of the *1st* block comes to a full-stop. As a result, feed direction is altered continuously. Due to this gradual change in the feed direction an interpolation error, ε, occurs around the junction point of consecutive path segments (See Figure 4.9a). When the overlapping time T_k is increased to its upper limit $T_k = T_d$, *no dwell* time is inserted between consecutive feed pulses. Since change in the feed direction is also initiated earlier, larger interpolation contour error occurs as shown in Figure 4.9d.

The contour error around the segment junction due to non-stop change in the feed direction can be controlled analytically. Consider the generic path shown in Figure 4.9, the deceleration motion towards midpoint, P_1 starts at $t = T_v$. When a non-zero overlapping time is set $T_k > 0$, feed direction towards the endpoint P_2 is initiated with the start of convolution of the consecutive segment at $t = T_v + T_d - T_k$. Note that convolution of the *1st* segment finishes at $t = T_v + T_d$, which marks the completion of feed direction change. If the feedrate F at consecutive segments is identical, total axis (x and y) velocity traverse, i.e. change in feed direction, is controlled directly by the angle between linear segments as:

$$\underbrace{\begin{Bmatrix} F_x^- = F\cos(\alpha) \\ F_y^- = F\sin(\alpha) \end{Bmatrix}}_{1^{st}\text{ Segment Velocities}}, \underbrace{\begin{Bmatrix} F_x^+ = F\cos(\gamma) \\ F_y^+ = F\sin(\gamma) \end{Bmatrix}}_{2^{nd}\text{ Segment Velocities}} \text{ where } \begin{Bmatrix} F_x^+ \\ F_y^+ \end{Bmatrix} = \underbrace{\begin{Bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{Bmatrix}}_{\text{Rotation Matrix}} \begin{Bmatrix} F_x^- \\ F_y^- \end{Bmatrix} \quad (4.22)$$

where $F_{x,y}^-$ represent axis velocities along the *1st* linear segment and $F_{x,y}^+$ represent velocities on the *2nd* (consecutive) segment, and β is the angle between the linear segments. Thus, when feed pulses with identical amplitude F are commanded, deceleration/acceleration kinematics around the bisector from the segment junction P_1 becomes mirror-imaged (See Figure 4.9a). As a result, tangential feedrate exhibits its minimum in the middle of the segment transition at $t = T_v + T_d - T_k/2$. Similarly, the interpolated trajectory also becomes symmetric where the maximum deviation from the junction point occurs along the bisector at $t = T_v + T_d - T_k/2$.

Kinematic profiles during segment transition depend on the overlapping time, T_k, and the filter delay, T_d. They can be computed by superimposing filtered velocity profiles of consecutive segments as shown in Figure 4.10. For a 7-segmented trapezoidal acceleration profile the *x* and *y*-axis velocities during segment transition can be calculated from Eq. (4.6) as:

$$v'_{x,y}(t)=\begin{cases} F^-_{x,y}-\frac{1}{2}\frac{F^-_{x,y}}{T_1T_2}\left(t-T_v\right)^2 & T_v\le t\le T_v+T_2 \\ F^-_{x,y}-\frac{1}{2}\frac{F^-_{x,y}T_2}{T_1}-\frac{F^-_{x,y}}{T_1}\left(t-\left(T_v+T_2\right)\right) & T_v+T_2\le t\le T_v+T_1 \\ \frac{1}{2}\frac{F^-_{x,y}}{T_1T_2}\left(\left(T_v+T_1+T_2\right)-t\right)^2 & T_v+T_1\le t\le T_v+T_1+T_2-T_k \\ \frac{1}{2}\frac{F^-_{x,y}}{T_1T_2}\left(\left(T_v+T_1+T_2\right)-t\right)^2 & \\ \quad+\frac{1}{2}\frac{F^+_{x,y}}{T_1T_2}\left(t-\left(T_v+T_1+T_2-T_k\right)\right)^2 & T_v+T_1+T_2-T_k\le t\le T_v+T_1+T_2 \\ \frac{1}{2}\frac{F^+_{x,y}}{T_1T_2}\left(t-\left(T_v+T_1+T_2-T_k\right)\right)^2 & T_v+T_1+T_2\le t\le T_v+T_1+2T_2-T_k \\ \frac{1}{2}\frac{F^+_{x,y}T_2}{T_1}+\frac{F^+_{x,y}}{T_1}\left(t-\left(T_v+T_1+2T_2-T_k\right)\right) & T_v+T_1+2T_2-T_k\le t\le T_v+2T_1+T_2-T_k \\ F^+_{x,y}-\frac{1}{2}\frac{F^+_{x,y}}{T_1T_2}\left(\left(T_v+2T_1+2T_2-T_k\right)-t\right)^2 & T_v+2T_1+T_2-T_k\le t\le T_v+2T_1+2T_2-T_k \end{cases} \tag{4.23}$$

for $T_k < 2T_2$. Similarly, maximum contouring error along the bisector can be calculated by superimposing the remaining distance towards the midpoint P_1 during interpolation of the *1st* segment, and the distance traveled due to the convolution of the *2nd* segment. Based on Figure 4.10 Cartesian components of the maximum contour error can be written as:

$$\left.\begin{aligned} \varepsilon_x &= -\int_{T_v+T_d-T_k/2}^{T_v+T_d} v'^-_x d\tau+\int_0^{T_k/2} v'^+_x\, d\tau \\ \varepsilon_y &= -\int_{T_v+T_d-T_k/2}^{T_v+T_d} v'^-_y d\tau+\int_0^{T_k/2} v'^+_y\, d\tau \end{aligned}\right\} \tag{4.24}$$

and integrating axis velocity profiles $v'_{x,y}^-$ and $v'_{x,y}^+$ from Eq. (4.6) yields the maximum contour error during uninterrupted interpolation of consecutive linear segments from Eq. (4.24) as:

$$\varepsilon=\sqrt{\varepsilon_x^2+\varepsilon_y^2}=\begin{cases} \frac{T_k^{\,3}}{24T_1T_2}F\sin\left(\frac{\beta}{2}\right) & 0\le T_k\le 2T_2 \\ \frac{4T_2^{\,2}-6T_2T_k+3T_k^{\,2}}{12T_1}F\sin\left(\frac{\beta}{2}\right) & 2T_2<T_k\le T_1+T_2 \end{cases} \tag{4.25}$$

Finally, for a predetermined contour error tolerance the overlapping time can be solved from Eq. (4.25) as:

$$T_k = \begin{cases} \sqrt[3]{\dfrac{24T_1T_2\varepsilon}{F\sin(\beta/2)}} & 0 \le T_k \le 2T_2 \\ T_2 + \sqrt{\dfrac{4T_1\varepsilon}{F\sin(\beta/2)} - \dfrac{T_2^{\,2}}{3}} & 2T_2 < T_k \le T_1 + T_2 \end{cases} \tag{4.26}$$

As presented above, the *dwell* time between consecutive interpolation of feed pulses is controlled by the amount of Tk from Eq. (4.26), and interpolation contour error is confined by a predetermined value efficiently.

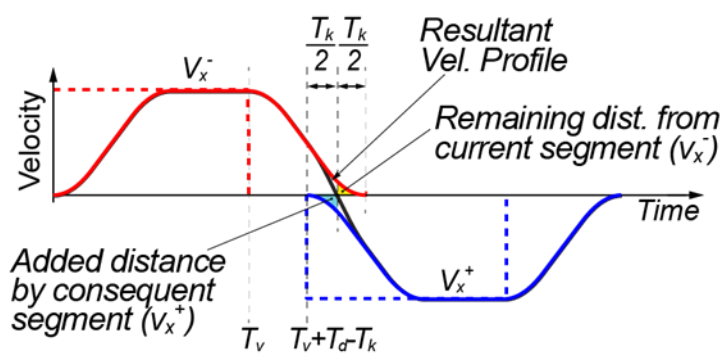

Figure 4.10: Axis kinematic profiles during segment transition.

4.3.2. Control of contour errors during non-stop linear and circular interpolation

As shown in Figure 4.11, contour errors occur during non-stop interpolation through circular (G2/G3) and linear (G1) segments as well. The dwell time control method presented in the previous section can be adapted to confine these contour errors by approximating the change in the feed direction.

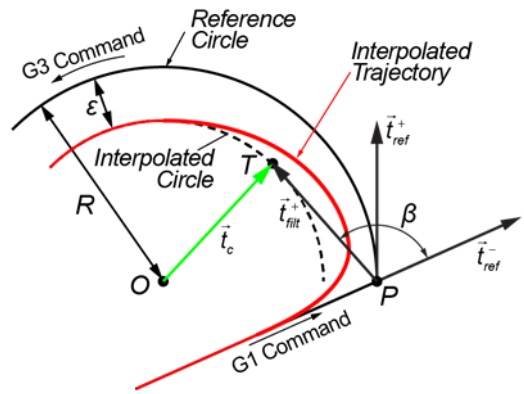

Figure 4.11: Linear to circular interpolation transition.

During circular interpolation, interpolated tool motion settles down on a circular path that has a smaller radius than the reference one due to the FIR filter dynamics (See Figure 4.7). As shown in Figure 4.11, the feed direction at the start of the circular interpolation is bounded between the tangent vector $\vec{\mathbf{t}}^{+}_{\mathbf{ref}}$ of the reference path and the tangent vector $\vec{\mathbf{t}}^{+}_{\mathbf{filt}}$ of the interpolated path shown. $\vec{\mathbf{t}}^{+}_{\mathbf{ref}}$ is known from the reference path geometry, and $\vec{\mathbf{t}}^{+}_{\mathbf{filt}}$ can be computed by the geometry through the following relationship:

$$\vec{t}_c \perp \vec{t}^{+}_{filt} \rightarrow \vec{t}_c \cdot \vec{t}^{+}_{filt} = \left|\vec{t}_c\right|\left|\vec{t}^{+}_{filt}\right|\cos\left(\frac{\pi}{2}\right) = 0 \tag{4.27}$$

where

$$\left.\begin{aligned} \left|\vec{t}_c\right| &= \sqrt{\left(T_x - O_x\right)^2 + \left(T_y - O_y\right)^2} \\ \left|\vec{t}_{filt}\right| &= \sqrt{\left(T_x - P_x\right)^2 + \left(T_y - P_y\right)^2} \end{aligned}\right\} \tag{4.28}$$

Note that norm of $\vec{\mathbf{t}}_{\mathbf{c}}$ is known from Eq. (4.16),

$$\left|\vec{t}_c\right| = \sqrt{\left(T_x - O_x\right)^2 + \left(T_y - O_y\right)^2} = R - \varepsilon \tag{4.29}$$

and substituting Eqs. (4.29) and (4.28) in (4.27) yields:

$$\vec{t}_{filt} = \left(\frac{(R-\varepsilon)^2 \left(P_x - O_x\right) \pm (R-\varepsilon)\sqrt{R^2-(R-\varepsilon)^2}\left(P_y - O_y\right)}{R^2} + O_x - P_x \right)\vec{i} + \left(\frac{(R-\varepsilon)^2 \left(P_y - O_y\right) \mp (R-\varepsilon)\sqrt{R^2-(R-\varepsilon)^2}\left(P_x - O_x\right)}{R^2} + O_y - P_y \right)\vec{j} \tag{4.30}$$

where $\vec{\mathbf{i}}$ and $\vec{\mathbf{j}}$ are the unit directional vectors in x and y directions, respectively. Feed direction during linear to circular segment G01→G02/03 transition is then bounded between the $\vec{\mathbf{t}}^{+}_{\mathbf{ref}}$ and $\vec{\mathbf{t}}^{+}_{\mathbf{filt}}$, and hence the largest change can be approximated as shown in Figure 4.11 as:

$$\beta = \arccos\left(\min\left(\frac{\vec{t}^{-}_{ref} \cdot \vec{t}^{+}_{ref}}{\left|\vec{t}^{-}_{ref}\right|\left|\vec{t}^{+}_{ref}\right|}, \frac{\vec{t}^{-}_{ref} \cdot \vec{t}^{+}_{filt}}{\left|\vec{t}^{-}_{ref}\right|\left|\vec{t}^{+}_{filt}\right|} \right) \right) \tag{4.31}$$

where $\vec{\mathbf{t}}^{-}_{\mathbf{ref}}$ is the feed vector along the linear path. Eq. (4.31) is used to calculate the overlapping time T_k and control the maximum value of the contour error. In a similar fashion, transition from circular to linear segment G02/03→G01 is depicted in Figure 4.12a, and Eq. (4.31) can be adapted for this case as:

$$\beta = \arccos\left(\min\left(\frac{\vec{t}^{-}_{ref} \cdot \vec{t}^{+}_{ref}}{\left|\vec{t}^{-}_{ref}\right|\left|\vec{t}^{+}_{ref}\right|}, \frac{\vec{t}^{-}_{filt} \cdot \vec{t}^{+}_{ref}}{\left|\vec{t}^{-}_{filt}\right|\left|\vec{t}^{+}_{ref}\right|} \right) \right) \tag{4.32}$$

Finally, the transition between two consecutive circular segments G02/G03→ G02/G03 is illustrated in Figure 4.12b. In this particular case, Eqs. (4.31) and (4.32) needs to be expanded to contain all possible combinations to bound the feed direction, and the largest angular change, i.e. worst case, is determined as:

$$\beta = \arccos\left(\min\left(\frac{\vec{t}^{-}_{ref} \cdot \vec{t}^{+}_{ref}}{\left|\vec{t}^{-}_{ref}\right|\left|\vec{t}^{+}_{ref}\right|}, \frac{\vec{t}^{-}_{ref} \cdot \vec{t}^{+}_{filt}}{\left|\vec{t}^{-}_{ref}\right|\left|\vec{t}^{+}_{filt}\right|}, \frac{\vec{t}^{-}_{filt} \cdot \vec{t}^{+}_{ref}}{\left|\vec{t}^{-}_{filt}\right|\left|\vec{t}^{+}_{ref}\right|}, \frac{\vec{t}^{-}_{filt} \cdot \vec{t}^{+}_{filt}}{\left|\vec{t}^{-}_{filt}\right|\left|\vec{t}^{+}_{filt}\right|} \right) \right) \tag{4.33}$$

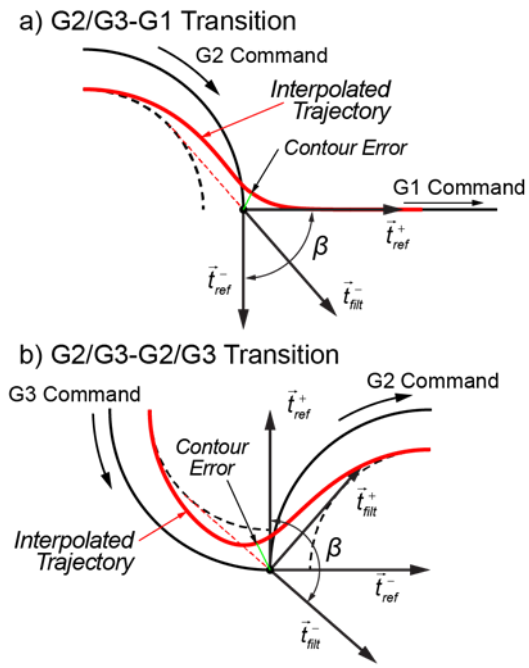

Figure 4.12: Feed direction during circular and linear transitions.

4.4. Illustrative example

This section demonstrates application of the proposed FIR based block timing control technique to accurately interpolate machining tool-paths. The reference tool-path shown in Figure 4.13 is given in the Gcode/CL program defined by two G01 linear segments followed by circular move (G02). In order to interpolate it with a jerk limited trapezoidal acceleration profile, 2 FIR filters are used with time constants set to T_1 = 50 [ms] and T_2 = 30 [msec]. The command feedrate is set to F = 200 [mm/sec], and the maximum interpolation error tolerance is ε = 100 [μm].

The path is interpolated based on the P2P and the proposed non-stop contouring type interpolation techniques (See Figure 4.8). Interpolation results are summarized in Figure 4.13. Figure 4.13a and b compare P2P and the contouring type interpolated tool trajectories, and resultant feedrate profiles. As shown, in case ofP2P interpolation, the motion undergoes a full stop at each segment junction. A *dwell* time identical to the total FIR filter delay T_d = 30 + 50 = 80 [ms] is inserted between the blocks. The total cycle time

for P2P motion results to T_{total} = 1.137 [s]. Proposed FIR filtering based contouring type interpolation technique can generate accurate non-stop feed motion. Contouring errors around transition of linear and circular segment junctions as well as the circular contour are precisely kept at and below the ε = 100 [μm] tolerance value. The circular interpolation error is bounded by lowering the feedrate to 84.2 [mm/s] from Eqs. (4.19) and (4.20). The contour error around segment transitions are controlled by calculating the overlapping time T_k based on the change in the feed direction from Eq. (4.26). The overall cycle time is reduced to T_{total} = 1.074 [s]. Next section presents interpolation of a more complex tool-path on an actual motion stage.

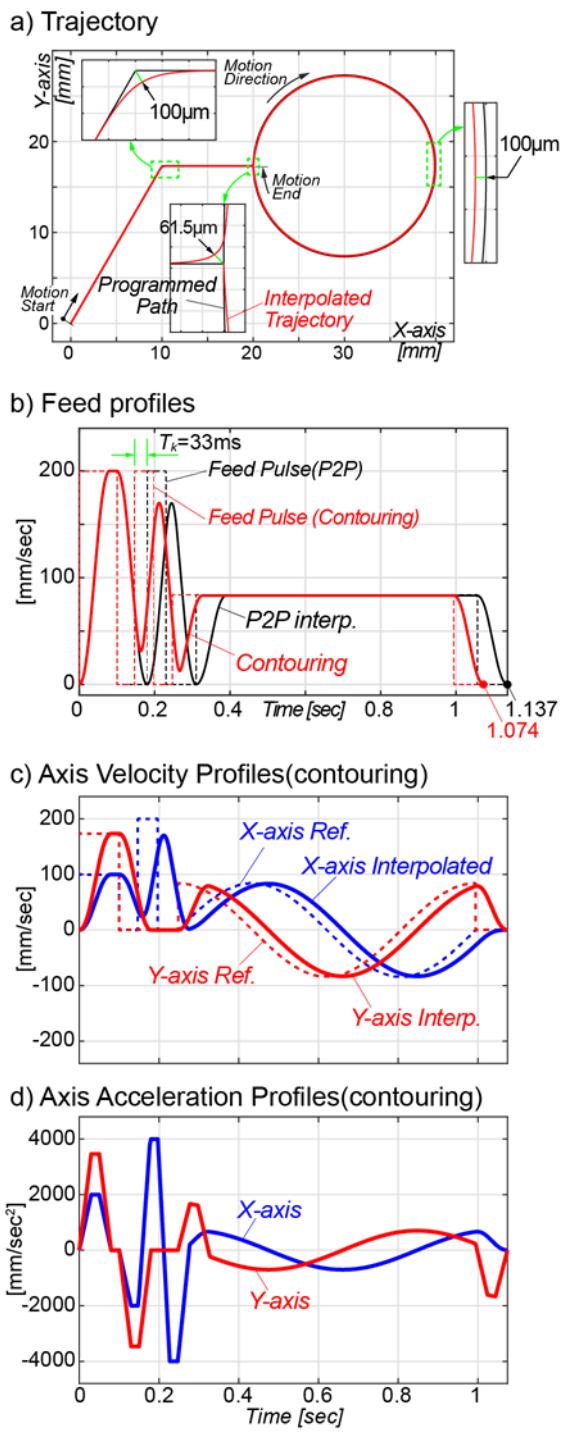

Figure 4.13: FIR based interpolation of multi-segmented path.

4.5. Experimental validation

4.5.1. Setup and implementation

Experimental validation and benchmark comparisons of the proposed technique are performed on the Cartesian X-Y motion system shown in Figure 4.14. The planar *X-Y* motion table is driven by 3 linear motors. The heavier X-axis is designed as gantry and carries the lighter Y-axis. In order to implement proposed algorithms, servo amplifiers are set to operate in torque (current) control mode. Closed loop control is implemented in the Dspace DS1103® real time control system by reading linear encoder feedback at a resolution of 0.1 [μm] and commanding torque signal to the servos at a closed loop sampling interval of T_s = 0.1 [msec]. Both X and Y drives are controlled by P-PI cascade motion controllers with velocity feed-forward action. The position feedback control bandwidths of the axes are roughly matched at ω_n = 50 [Hz] to ensure good motion synchronization and contouring [23].

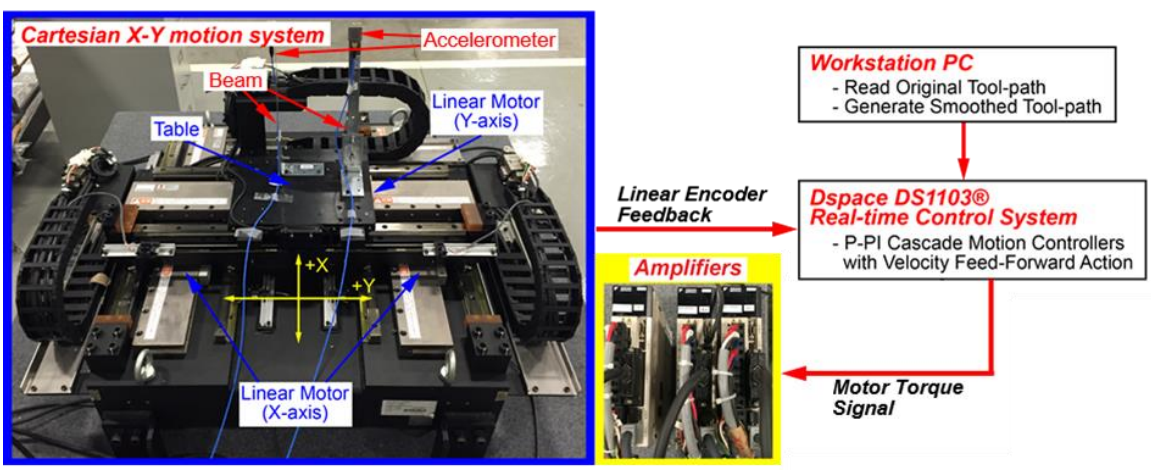

Figure 4.14: Experimental XY motion platform.

Next, to generate motion commands at discrete time instants kT_s, FIR filter's transfer function from Eq. (4.1) needs to be discretized. A simple Euler's backward differentiation technique [63] is applied to derive the *z-domain* transfer function as:

$$M_i(z) = \frac{1}{N_i}\frac{1-z^{-N_i}}{1-z^{-1}} \tag{4.34}$$

where $N_i = T_i/T_s$ is the number of (delay) samples of the filter. The filtered velocity commands $v'_{x,y}$ are generated by implementing Eq. (4.34) through the following simple difference equation:

$$v'_{x,y}(k) = v'_{x,y}(k-1) + \frac{1}{N_i}\ v_{x,y}(k) - v_{x,y}(k-N_i) \tag{4.35}$$

where k is the sample counter. Note that generation of the filtered velocity commands from above difference equation requires only 2 additions and 1 multiplication for a single FIR filter.

4.5.2. Experimental results

In the *1st* experiment, the clover shaped tool-path shown in Figure 4.15 is interpolated with the P2P and contouring interpolation techniques. As shown, this tool-path consists of 5 linear and 5 circular segments. The feedrate along the tool-path is set to F = 200 [mm/sec], and the maximum interpolation contour tolerance is ε = 100 [μm]. 2 FIR filters are used to interpolate the tool-path with time constants set to T_1 = 30 and T_2 = 25 [msec]. As shown in Figure 4.15, the tool-path is interpolated non-stop within desired contouring tolerance. Maximum contouring errors both around linear and circular segment transitions as well as along the circular sections are respected. Kinematic profiles are shown in Figure 4.16. As shown, feedrate is lowered along circular sections and also at segment junctions to generate accurate transition. Figure 4.16b depicts the feed pulse timing and the resultant feedrate profiles. The cycle time during contouring interpolation is clearly shorter and axis motion profiles are acceleration and jerk limited (See Figure 4.16c and d). Contouring errors are measured experimentally [23] and presented in Figure 4.17 as well. As shown, the tool-path is interpolated within the given contour error tolerance. Experimentally measured contour errors show small discrepancy from the interpolated ones due to feedback tracking dynamics of the servo system. These errors are not accounted for in the proposed technique, and although small in this experiment (< 15 [μm]), they can be further improved by well-known feed-forward control techniques [63].

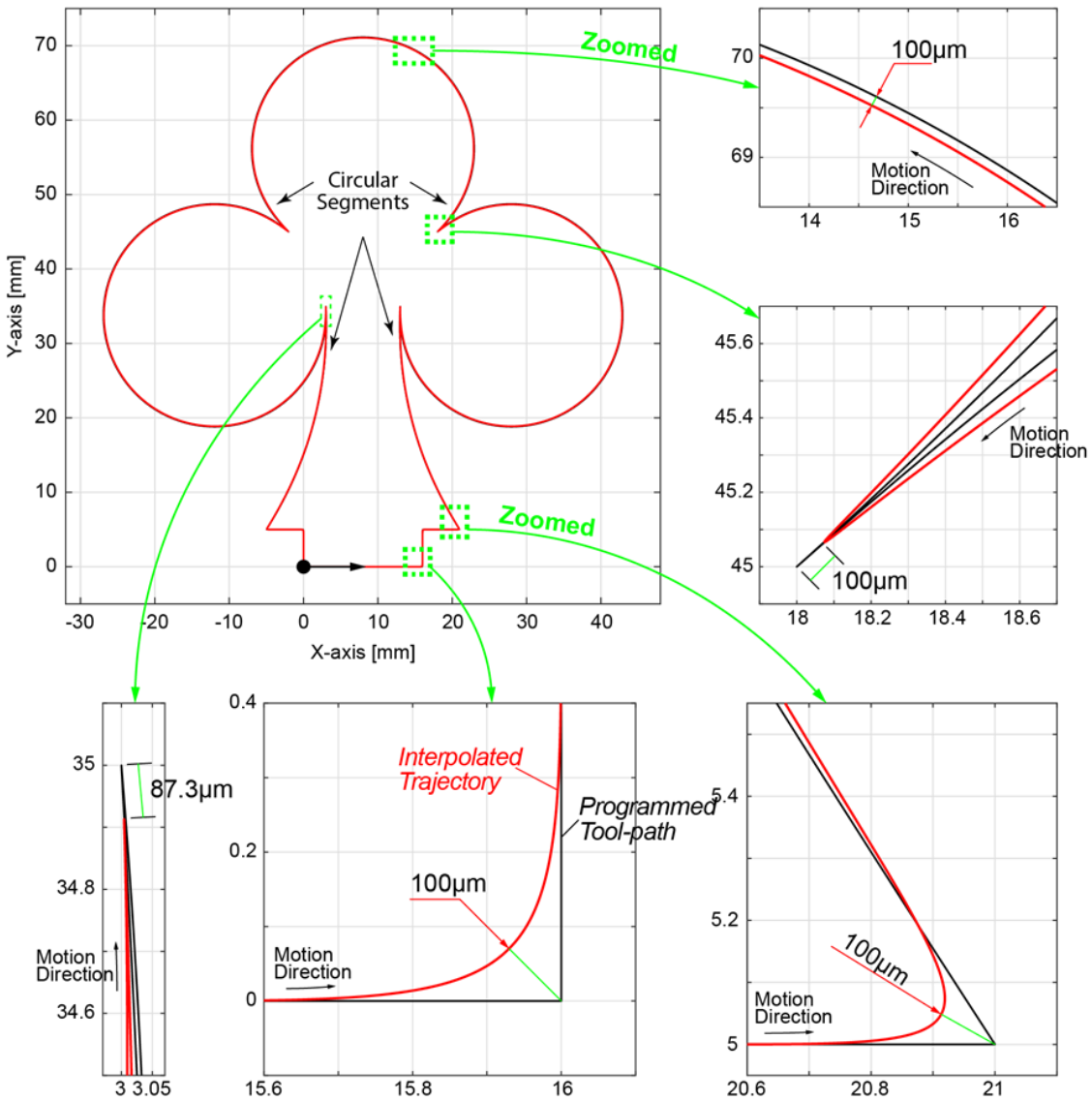

Figure 4.15: Clover shaped tool-path.

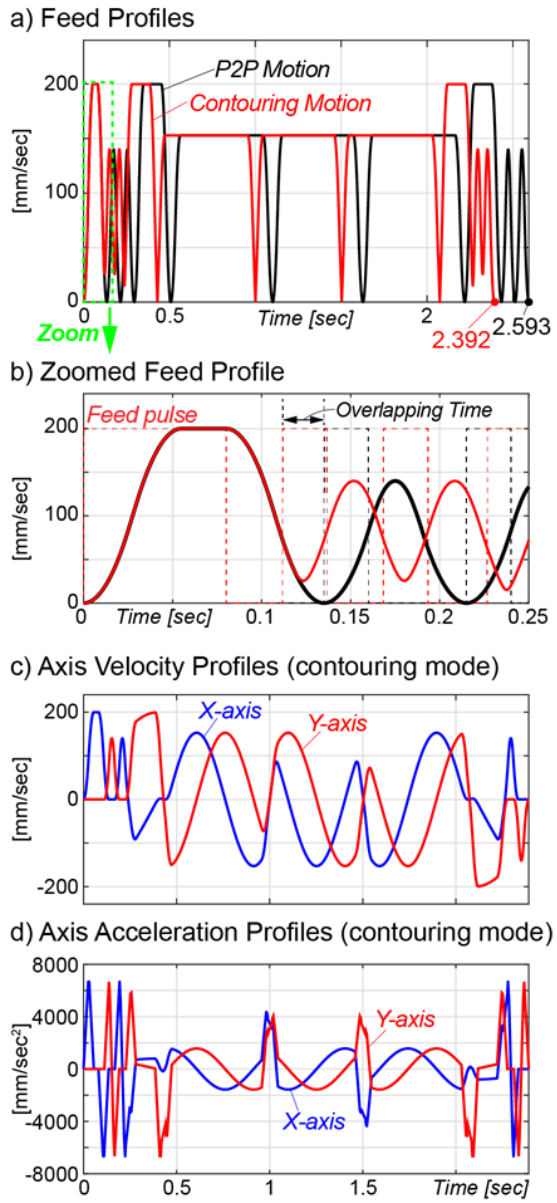

Figure 4.16: Interpolated kinematic profiles along clover shaped tool-path.

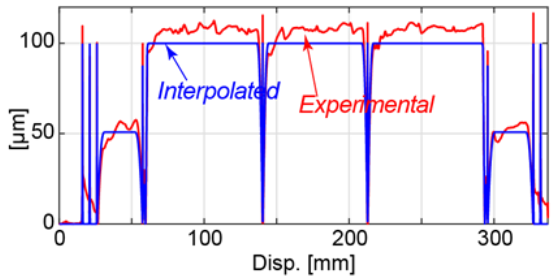

Figure 4.17: Contour errors during clover tool-path.

A *2nd* experiment is performed to showcase vibration suppression capabilities of the proposed interpolation scheme. The starfish shaped tool-path shown in Figure 4.18 is interpolated using the proposed FIR filtering based technique and compared against another technique that fits small Bezier segments at segment junctions to realize a continuous motion transition and thereby non-stop contouring motion [8]. The tool-path consists of 125 short linear segments. The programmed feedrate is set to F = 50 [mm/sec], and the contouring tolerance is ε = 30 [μm]. 2 FIR filters are used to interpolate the path with trapezoidal acceleration profiles. The filter time constants are tuned to avoid unwanted vibrations. As shown in Figure 4.14, 2 flexible beams are placed on the X-Y table. The beams are flexible in orthogonal directions, e.g. X and Y, where their *1st* bending modes are identified with accelerometers (PCB-3711E1110G) mounted on the top of the beams as: $\omega_x \approx 7.4$ [Hz] and $\omega_y \approx 9.2$ [Hz]. In order to avoid exciting the lightly damped beam resonances, FIR filter delays are set to T_1 = 136.1 [msec] and T_2 = 110.9 [msec], accordingly. Figure 4.19 shows feedrate profiles of the interpolation techniques. Note that, due to short linear segments, the programmed feedrate is never reached. As a result, interpolated feed profiles and the corresponding accelerations fluctuate aggressively. Nevertheless, the proposed FIR based filtering technique delivers the fastest cycle time while respecting desired contouring tolerance along the entire tool-path. Figure 4.20 compares frequency spectrum of the interpolated acceleration profiles. As shown, FIR based interpolation technique exhibits attenuated frequency spectrum especially around the resonances of flexible beam structures. Most of the excitation is kept in the lower frequencies. However, the Bezier based technique simply spreads the excitation at a much wider bandwidth. Figure 4.21 presents beam accelerations measured through the attached accelerometers. As shown, the level of acceleration of the beams are significantly less as the motion stage tracks the FIR based interpolated trajectory. The maximum acceleration is same as the interpolated one from the reference trajectory. As compared, the acceleration level is significantly higher on the Bezier based trajectory generation technique. Finally, Figure 4.22 shows the frequency spectrum of the measured beam accelerations. As shown, the Bezier based technique clearly excites the resonances and causes beams to vibration heavily. On the other hand, FIR based technique does not induce any unwanted vibrations

and the beams move as a rigid body with the motion table. This experiment clearly demonstrates that the proposed technique can interpolate complex tool-paths accurately, deliver rapid non-stop contouring motion and at the same time mitigate unwanted residual vibrations.

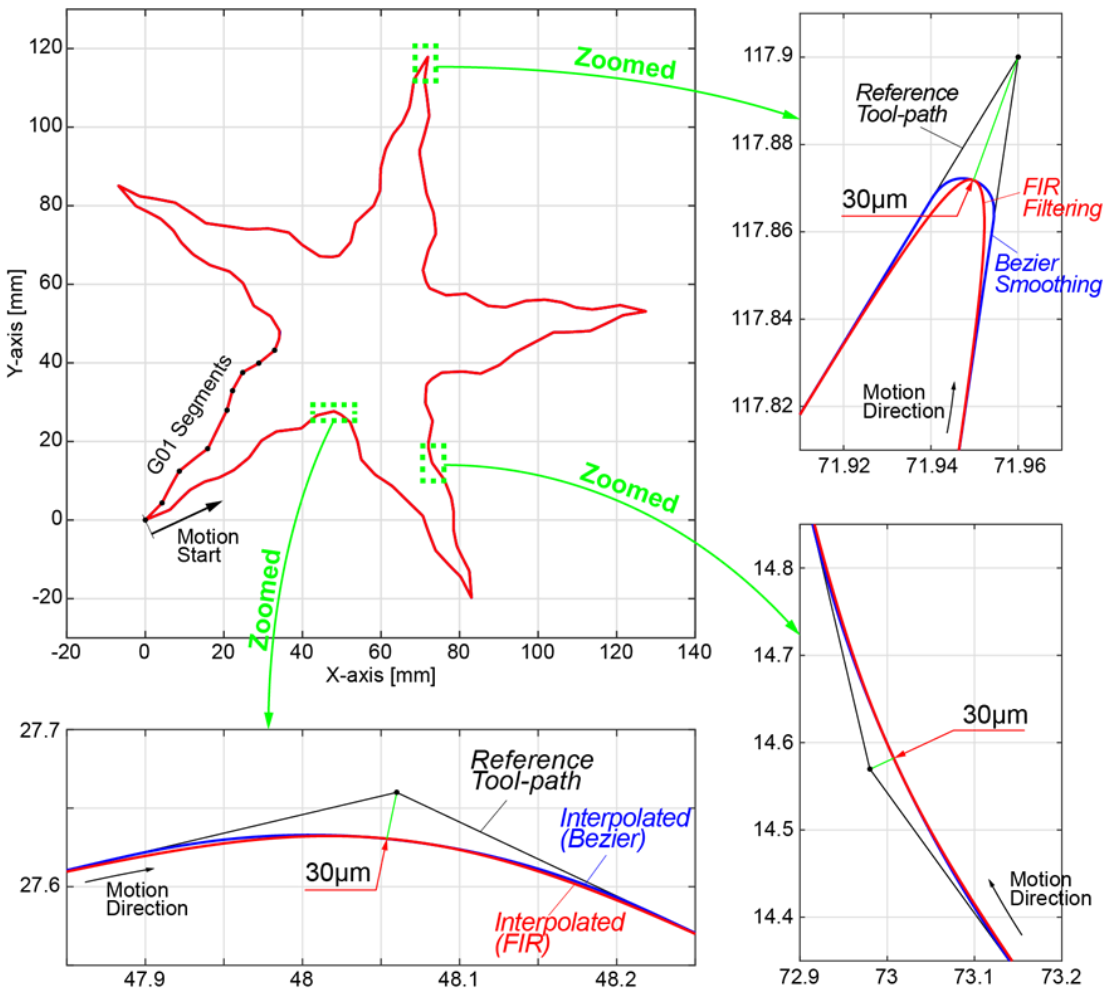

Figure 4.18: Starfish shaped tool-path.

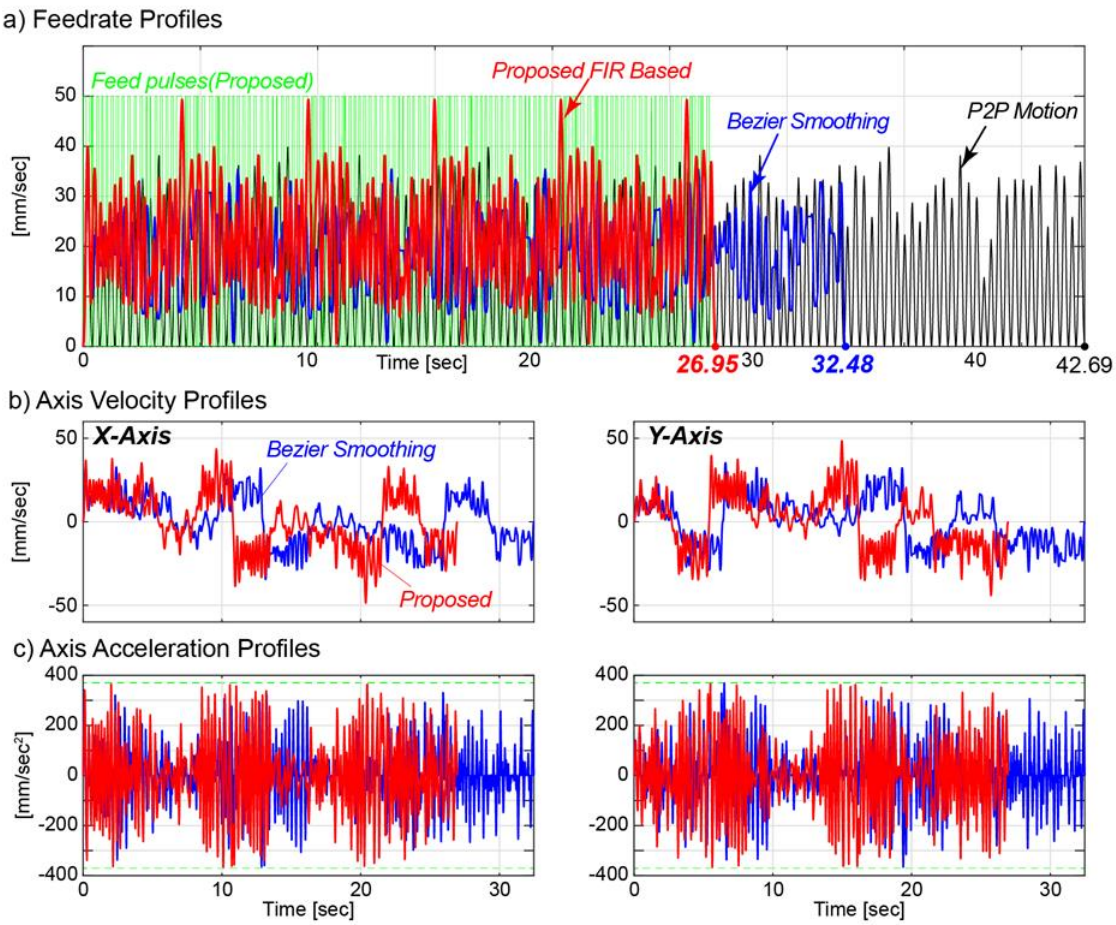

Figure 4.19: Interpolated kinematic profiles along starfish shaped tool-path.

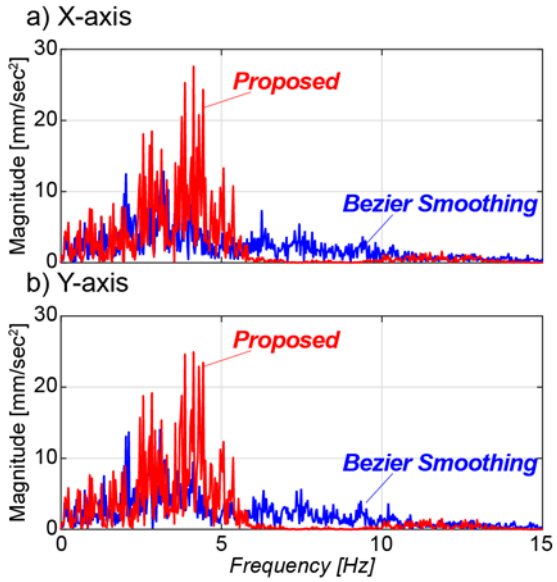

Figure 4.20: DFT of interpolated axis acceleration profiles.

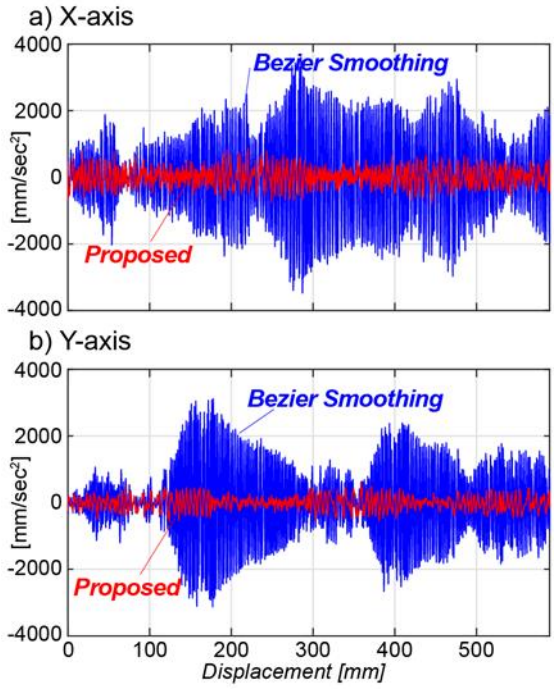

Figure 4.21: Experimentally measured beam accelerations.

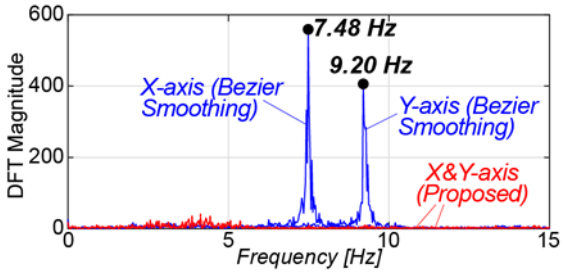

Figure 4.22: DFT of beam accelerations.

4.6. Conclusions

A novel online trajectory generation scheme has been proposed for Cartesian machines and motion systems to generate high-speed and accurate feed motion. Owing to its simple filtering structure, proposed scheme can interpolate linear and circular paths with high kinematic continuity and minimum computational load making it suitable for real-time processors. The proposed block timing technique considers the change in the feed

direction and the total delay in the filter chain to generate accurate non-stop rapid feed motion. For the first time, interpolation errors that occur during both linear and circular segment transitions as well as circular arcs are considered making the proposed scheme comprehensive for multi-segmented paths. Furthermore, by tuning the filter delays with respect to the dynamics of the motion system, frequency spectrum of the acceleration profile is shaped and unwanted residual vibrations are avoided. Experimental results validated that the proposed technique can interpolate multi-segmented tool-paths accurately. As compared to the state of the art technique, the proposed interpolation method can eliminate unwanted vibrations and reduce the cycle time up to ~20% while utilizing same level of acceleration proving it to be a practical and effective online interpolation technique form modern NC systems.

Accurate real-time interpolation of 5-axis tool-paths with local corner smoothing

Shingo Tajima and Burak Sencer

International Journal of Machine Tools & Manufacture

Volume 142, July 2019, Pages 1-15

5. Accurate real-time interpolation of 5-axis tool-paths with local corner smoothing

5-axis machining tool-paths, when programmed in workpiece coordinates, translational and rotational tool motion in terms of Cartesian tool center points (TCPs), and unit tool orientation vectors (ORI). This paper presents a computationally efficient real-time trajectory generation algorithm for 5-axis machine tools to interpolate translational and rotational tool motion synchronously for accurate 5-axis machining. Finite Impulse Response (FIR) filters are used to generate jerk limited motion trajectories in real-time. Linear translational motion of the tool center point (TCP) is interpolated by FIR filtering of Cartesian velocity pulses in G01 blocks. In order to generate constant speed tool axis rotation, spherical linear interpolation is used, and unit tool orientation vectors (ORI) are filtered directly in the spherical coordinates. Precise tool motion synchronization is realized by matching time-constants of FIR filters utilized for translational and rotational interpolation. Non-stop path interpolation is achieved by locally blending consecutive linear G01 commands. Instead of fitting geometric blending curves and solving feed scheduling problem, smoothing functionality of FIR filtering is used, and a direct 1-step path smoothing algorithm is proposed for real-time implementation. The algorithm considers path blending errors in Cartesian (Euclidian) as well as in spherical (orientation) coordinates due to transient response of the FIR filter. As a result, both tool-tip and the tool-orientation errors are controlled accurately. Effectiveness of the developed algorithms are validated in simulations and also experimentally on an open-NC controlled 5-axis machine tool.

5.1. Introduction

5-axis machines have become a crucial tool for modern die and mold and aerospace industries. As compared to their 3-axis counterparts, 5-axis machines can alter tool orientation in synch with the tool-tip in simultaneous 5-axis machining. This functionality enables them to machine complex sculptured surfaces faster and thereby achieve significantly better finish quality [66]. The speed, accuracy and overall productivity of 5-axis machine tools are greatly affected by 2 key factors: i) Reference tool-path, and ii)

Performance of the numerical control (NC) system [9]. This paper proposes a novel real-time trajectory generation technique for accurate interpolation of discrete 5-axis machining tool-paths.

Typical 3 or 5-axis machining tool-paths are generated using Computer-Aided Manufacturing (CAM) systems and programmed by series of linear G01 moves [9], [67]. Deficiencies of this point-to-point (P2P) linear tool-path format are well-known to 3-axis machining literature. Since linear tool-paths are only position (G^0) continuous, feed motion must undergo instantaneous stops at segment junctions, which yields elongated cycle times, puts strain on the drives and generates stair-cased rough surface finish. A well-known approach to generate non-stop continuous tool motion is to blend consecutive linear segments locally using micro-splines [8], [23], which ensures acceleration continuous transition between consecutive linear moves. This operation is performed by the NC system in real-time. The key is to control the corner rounding (blending) geometry so that the blending error is within the allowable part tolerance, and its curvature is minimized [8] to attain rapid feed motion. Most techniques are based on this 2-step approach where firstly, a micro-spline curve is fitted to generate a smooth corner geometry. Next, smooth feedrate profile is planned along the blended path geometry and interpolated in real-time while minimizing feed fluctuations and chord errors [4], [22], [23], [29]. Yutkowitz [6] patented a corner smoothing method, which uses two 4th order polynomial curves to blend two linear feed blocks with user-specified tolerances and ensure an acceleration continuous feed motion. Erkorkmaz et al. [23] used a quintic (5th order) splines to round a sharp corner accurately. Sencer et al. [8] used a curvature optimized quinitc Bezier spline to generate both accurate and rapid cornering motion. Ernesto and Farouki [25] employed a Bezier conic and optimized the feedrate along the curve under acceleration bounds. Variety of spline types and feed optimization algorithms are proposed in the literature in an effort to achieve faster cycle times and also minimize the computational load required for real-time interpolation [7], [10], [31].

Currently available 5-axis local path smoothing algorithms are generally build on this existing 2-step approach developed for 3-axis machining. Nevertheless, 5-axis machining brings additional challenges. A crucial challenge arises immediately in the 1st

step. Typical 5-axis machining tool-paths are programmed in workpiece coordinates where tool axis translation is described by Cartesian tool center points (TCPs) and tool axis orientation is defined in spherical coordinates by unit tool-orientation vectors (ORI). Both translational and rotational tool motion must be blended locally in their corresponding coordinate systems and interpolated synchronously. As illustrated in Figure 5.1, this requires accurate control of Euclidian tool-tip and angular tool-orientation blending errors at segment junctions, and also synchronized interpolation of those curves in their corresponding coordinate systems (cartesian and spherical). Fleasing and Spence [14] were the first one to propose a de-coupled approach where Quintic splines are used in Cartesian and spherical coordinates to smoothly interpolate TCP and ORI paths separately. They reported that the use of linear spherical interpolation for tool orientation vectors reduces angular velocity fluctuations and provides smoother tool motion. Nevertheless, they have to use an extra polynomial to synchronize linear and angular tool velocities, which is reported as a challenge in this de-coupled interpolation strategy [14]. This approach requires actually 3 steps; namely, geometric spline fitting, feed-planning and TCP and ORI synchronization. Tulsyan and Altintas [26] adapted this approach for local corner smoothing, and derived analytical solutions to minimize the computational cost of fitting higher order splines in cartesian and spherical coordinates. System of non-linear equations must be solve to fit corner blending splines within user specified Euclidian and angular error tolerances [14], which greatly limits real-time implementation on modern NC systems. Yang and Yuen [68], [69] further provided closed-form solutions in an attempt to reduce the computational burden.

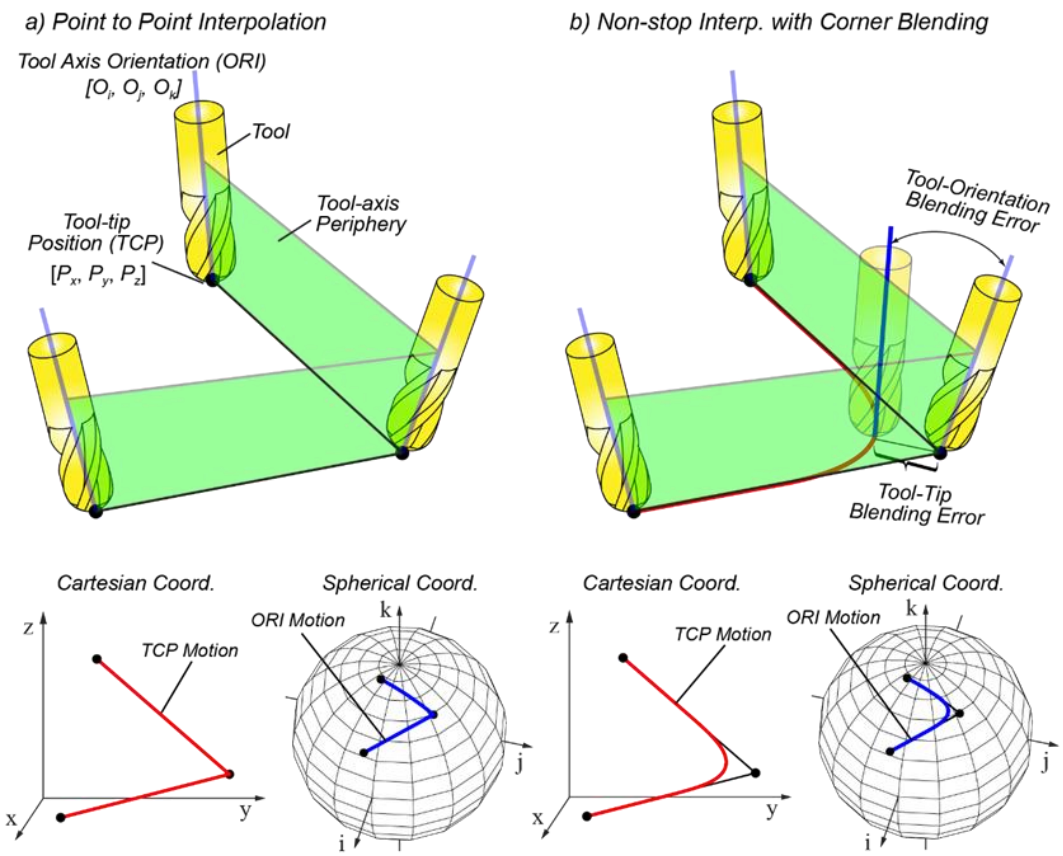

Figure 5.1: 5-Axistool-pathinterpolation.

Although de-coupled blending in Cartesian and Spherical coordinates allows accurate control of tool motion, it is computationally expensive [70], [71]. Dual-spline interpolation in cartesian coordinates is proposed to reduce the computational expense [10], [15], [67], [72], [73]. In this approach, TCP and another point along the tool-shank is blended in cartesian coordinates. The analytical framework generated for the 3-axis case can be deported to 5-axis machining for this approach. Notice that, since two local corner smoothing curves for TCP and another tool-axis point are used, computational load is actually only doubled from the 3-axis case. As long as those blending curves are parameterized identically, this approach potentially reduces feed fluctuations arising from synchronization of linear and angular tool motion kinematics as well. Bezier and B-spline [15], [72], or PH curves [73] can be used for corner blending. The blending step is then followed by feedrate profiling, and interpolation of those blending curves [10], [47], [74]. As noted, lifting the 2-step corner blending approach for 5-axis machining requires at least

doubles the computational load and still suffers from deficiencies related to real-time interpolation.

This paper presents novel online interpolation of 5-axis machining tool-paths in workpiece coordinates. Drawbacks of geometric smoothing and interpolation are avoided, and a “direct” path interpolation approach is proposed. Firstly, Finite Impulse Response (FIR) filters [34], [43], [75] are used to interpolate translational and rotational tool motion in workpiece coordinate system. TCP motion is interpolated and locally blended in Cartesian coordinates by FIR filtering of axial velocity components [34]. Spherical linear interpolation [14] is employed, and tool orientation is blended directly in spherical coordinates. Rotational blending errors are also controlled analytically. Henceforth, the paper is organized as follows. Section 5.2 presents real-time P2P interpolation of 5-axis tool-paths using FIR filtering. Section 5.3 shows local corner blending for tool translation and tool orientation based on two novel approaches. Finally, experimental validations conducted on a 5-axis machine tool are presented in section 5.4.

5.2. Point-to-point (P2P) linear interpolation of 5-axis tool-paths

As shown in Figure 5.1, typical simultaneous 5-axis machining tool-paths are defined in the workpiece coordinate system, and they command synchronized linear interpolation of tool-tip (TCP) and tool-axis orientation (ORI) [67], [76]. TCP is defined by its Cartesian coordinates $\boldsymbol{P} = [P_x, P_y, P_z]^T$, and ORI is given by set of unit orientation vectors $\boldsymbol{O} = [O_i, O_j, O_k]^T$ in spherical coordinates. During P2P interpolation, TCP must be interpolated linearly between successive points. At the same time, ORI must be interpolated linearly between successive tool-axis orientation vectors on the unit sphere [14], [26].

This section first introduces fundamentals of Point to Point (P2P) interpolation based on FIR filtering. This is then followed by P2P linear interpolation of 5-axis toolpaths in workpiece coordinate system.

5.2.1. FIR filtering based real-time interpolation

In this work, Finite Impulse Filters (FIR) are utilized to realize linear interpolation of tool motion with trapezoidal (jerk-limited) acceleration kinematics so that generated trajectories are suitable for high-speed 5-axis machining. A *1st* order FIR filter is defined in Laplace(s) [77] domain by the following transfer function (TF) [43], [75]:

$$M_i(s) = \frac{1}{T_i}\frac{1-e^{-sT_i}}{s}, \quad i = 1...N \tag{5.1}$$

where T_i is the time constant (delay) of the *i*th FIR filter. FIR filter's TF consists of an integrator (1/s) and a pure delay e^{-sT_i} yielding a rectangular impulse response as:

$$m_i(t) = L^{-1}\ M_i(s)\ = \frac{u(t)-u(t-T_i)}{T_i} \text{ where } u = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases} \tag{5.2}$$

When Eq. (5.2) is convolved ($*$) with a rectangular velocity pulse v for a length of T_v, it produces a trapezoidal velocity profile. Convolving resultant trapezoidal velocity profile with another FIR filter (i = 2) having a constant T_2 increases continuity of the motion (due to the integrator in FIR's TF from Eq. (5.1)), and generates the well-known jerk-limited velocity profile v' with trapezoidal acceleration transients [22], [75] also shown in Figure 5.2 as:

$$v'(t) = v(t) * m_1(t) * m_2(t) = \begin{cases} \frac{1}{2}\frac{F}{T_1T_2}t^2 & 0 \le t < T_2 \\ \frac{1}{2}\frac{FT_2}{T_1} + \frac{F}{T_1}\ t - T_2 & T_2 \le t < T_1 \\ F - \frac{1}{2}\frac{F}{T_1T_2}\ \ T_1 + T_2\ \ - t\ ^2 & T_1 \le t < T_1 + T_2 \\ F & T_1 + T_2 \le t < T_v \\ F - \frac{1}{2}\frac{F}{T_1T_2}\ t - T_v\ ^2 & T_v \le t < T_v + T_2 \\ F - \frac{1}{2}\frac{FT_2}{T_1} - \frac{F}{T_1}\ t -\ T_v + T_2 & T_v + T_2 \le t < T_v + T_1 \\ \frac{1}{2}\frac{F}{T_1T_2}\ \ T_v + T_1 + T_2\ \ - t\ ^2 & T_v + T_1 \le t < T_v + T_1 + T_2 \\ 0 & T_v + T_1 + T_2 \le t \end{cases}$$

(5.3)

Notice that total motion duration becomes summation of the original velocity pulse length and time constants of the FIR filters, $T_{total} = T_v + T_1 + T_2$. Filter's time constant controls shape of the profile and a detailed analysis on the motion kinematics is already provided in [43], [75]. For instance, if $T_v > T_1 > T_2$, a full 7 segmented jerk limited trajectory could be generated. Peak acceleration is controlled by filter time constant and tangential speed, F/T_1. Similarly, peak jerk value becomes $F/T_1/T_2$ (see Figure 5.2). Hence, filter time constants or block's feedrate can be selected to adjust the peak acceleration and jerk demanded by the trajectory.

Kinematic profiles, i.e. position, velocity, acceleration and jerk, can be computed analytically in polynomial form by formally solving the convolution integral. For the real-time implementation on NC systems; however, this may not be necessary. FIR filter is discretized,

$$M_i(z) = M_i(s)\big|_{s=\ 1-z^{-1}/T_s} = \frac{1}{N_i}\frac{1-z^{-N_i}}{1-z^{-1}} \tag{5.4}$$

with $z = \exp(sT_s)$ being the discrete time operator, T_s is servo loop's sampling time and $N_i = \text{int}(T_i/T_s)$ is the number of delay samples. Eq. (5.4) is then simply implemented in the

form of a moving average filter [43], [75] in real-time by evaluating following difference equation:

$$v'_i(k) = v'_i(k-1) + \frac{1}{N_i}\left(v'_{i-1}(k) - v'_{i-1}(k-N_i)\right) \tag{5.5}$$

where k is the sample counter and v'_i is the ith filtered signal. As observed, real-time FIR filtering-based trajectory generation requires only 1 multiplication and 2 summation operation. As shown in Figure 5.2, jerk-limited acceleration profiles require 2 consecutive FIR filtering. Therefore, 2 multiplication and 4 addition operation must be performed at most to generate the profile in real-time, which is conveniently realizable by most modern NC units or even low-cost micro-processors.

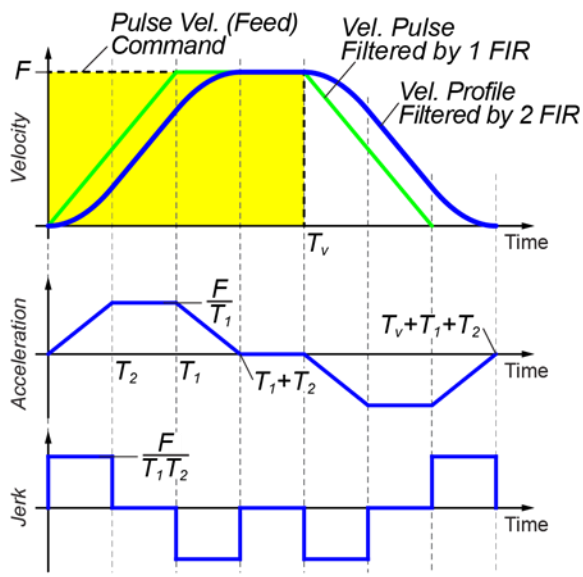

Figure 5.2: Jerk limited trajectory generated by FIR filtering.

5.2.2. Real-time point to Point (P2P) interpolation of 5-axis toolpaths

This sub-section presents linear interpolation of simultaneous P2P 5-axis motion trajectories based on FIR filtering directly in workpiece coordinates. Figure 5.3 outlines the proposed de-coupled interpolation scheme. As shown, translational and rotational motion of the tool are interpolated in a de-coupled fashion. Translational tool motion is generated by linear interpolation of TCPs in the Cartesian workpiece coordinates.

Synchronously, tool-orientation is interpolated by spherical linear interpolation of ORI vectors in Spherical coordinates.

Consider starting and ending tool poses on a G-line, [$\boldsymbol{P_s}$, $\boldsymbol{O_s}$] and [$\boldsymbol{P_e}$, $\boldsymbol{O_e}$]. The Euclidian distance and angle of rotation are calculated as:

$$L = \left\| \mathbf{P_e} - \mathbf{P_s} \right\| \text{ and } \theta = \arccos \mathbf{O_s} \cdot \mathbf{O_e} \tag{5.6}$$

If the machining feedrate F is defined based on linear tool-tip velocity, then a linear feed pulse is generated for the duration of

$$T_v = \frac{L}{F} = \frac{\left\| \mathbf{P_e} - \mathbf{P_s} \right\|}{F} \tag{5.7}$$

In order to ensure synchronized tool motion, identical angular velocity pulse duration T_v is set and corresponding angular velocity ω pulse magnitude is calculated:

$$\omega = \frac{\theta}{T_v} = \frac{\arccos\left(\mathbf{O_s} \cdot \mathbf{O_e}\right)}{T_v} \tag{5.8}$$

Typically, federate in 5-axis machining is programmed w.r.t to linear TCP velocity. If inverse-time-mode interpolation [47] is programmed, then T_v may be already provided in the G-code. In this case, linear and angular speed pulse amplitudes are updated to $F = L/T_v$ and $\omega = \theta/T_v$.

Next, the TCP velocity pulse is dissolved into its Cartesian components based on the path geometry:

$$\frac{d\mathbf{P}(t)}{dt} = \dot{\mathbf{P}}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} = v(t)\vec{\mathbf{t}}, \quad \text{where } \vec{\mathbf{t}} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \frac{\mathbf{P_e} - \mathbf{P_s}}{\left\| \mathbf{P_e} - \mathbf{P_s} \right\|} \tag{5.9}$$

As shown in Figure 5.3d and e, Cartesian velocity pulses v_x, v_y, v_z are convolved (filtered) with parallel FIR filters directly in the workpiece coordinates,

$$\frac{d\mathbf{P}'(t)}{dt} = \dot{\mathbf{P}}'(t) = \begin{bmatrix} v'_x(t) \\ v'_y(t) \\ v'_z(t) \end{bmatrix} = \dot{\mathbf{P}}(t) * m_1(t) * m_2(t) \tag{5.10}$$

where m_1 and m_2 are the impulse responses of FIR filters. Filtered (smoothened) velocity commands are then integrated to generate the final TCP commands:

$$\mathbf{P}'(t) = \begin{bmatrix} p'_x(t) \\ p'_y(t) \\ p'_z(t) \end{bmatrix} = \int_0^t \begin{bmatrix} v'_x(t) \\ v'_y(t) \\ v'_z(t) \end{bmatrix} d\tau \tag{5.11}$$

As shown in Figure 5.3, the tool orientation vectors (ORI) are interpolated in parallel. The pseudo angular velocity pulse components $d\boldsymbol{O}/dt = [v_i, v_j, v_k]$ are generated on the unit sphere by making use of the spherical linear interpolation [14], [26] as:

$$\frac{d\mathbf{O}(t)}{dt} = \dot{\mathbf{O}}(t) = \begin{bmatrix} v_i(t) \\ v_j(t) \\ v_k(t) \end{bmatrix} = \frac{-\omega\cos\left(\left(T_v - t\right)\omega\right)\mathbf{O_s} + \omega\cos\left(t\omega\right)\mathbf{O_e}}{\sin\left(\theta\right)}, \quad \left\|\mathbf{O_s}\right\| = \left\|\mathbf{O_e}\right\| = 1 \tag{5.12}$$

and they are smoothened by convolving with parallel FIR filters in the spherical coordinates:

$$\frac{d\mathbf{O}'(t)}{dt} = \dot{\mathbf{O}}'(t) = \begin{bmatrix} v'_i(t) \\ v'_j(t) \\ v'_k(t) \end{bmatrix} = \dot{\mathbf{O}}(t) * m_1(t) * m_2(t) \tag{5.13}$$

Finally, numerical integration is performed to interpolate the final ORI vectors directly on the unit sphere:

$$\mathbf{O}'(t) = \begin{bmatrix} o'_i(t) \\ o'_j(t) \\ o'_k(t) \end{bmatrix} = \int_0^t \left(\dot{\mathbf{O}}(t) * m_1(t) * m_2(t)\right) d\tau \tag{5.14}$$

As depicted in Figure 5.3, FIR filtering based linear interpolation of TCP and ORI vectors is performed directly in the workpiece system. Since parallel FIR filters with identical time constants are used, tool translational and orientation motion start and end at the same time in a synchronized manner. Furthermore, the linear and angular speed of the tool is kept constant eliminating any unwanted fluctuations.

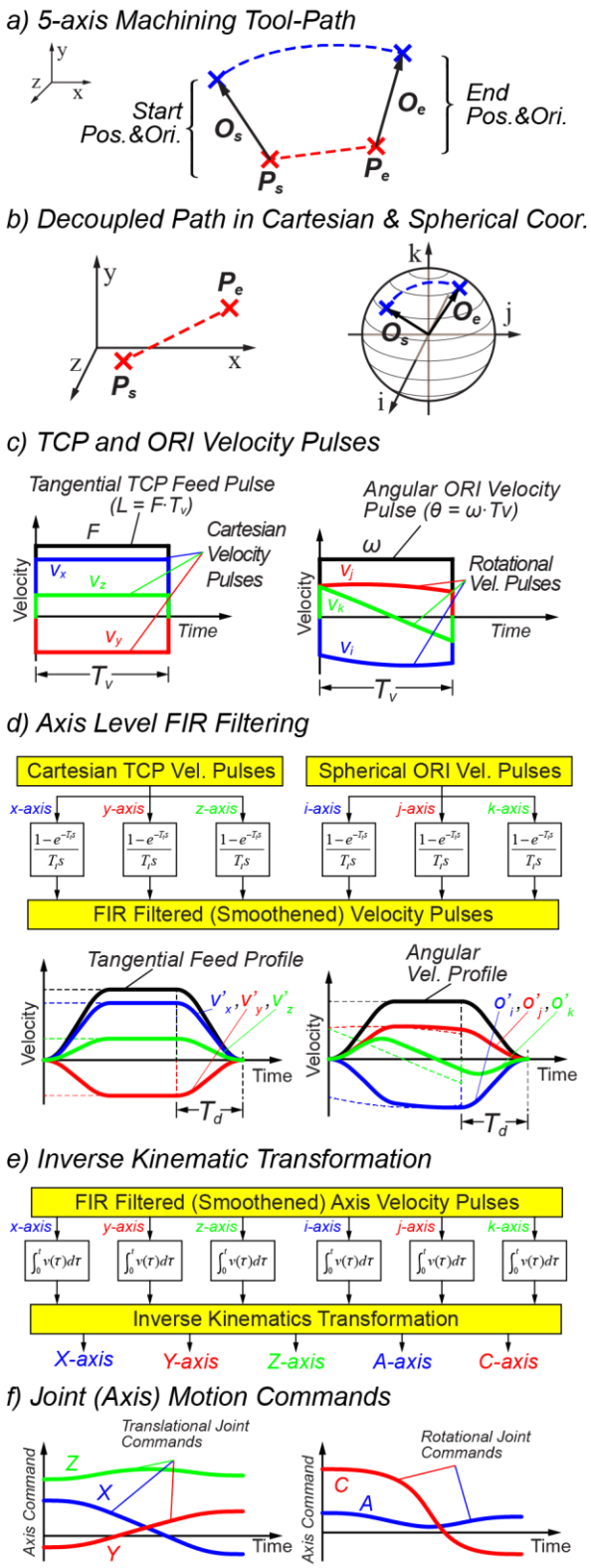

Figure 5.3: Point to Point (P2P) tool-motion interpolation.

5.2.3. Illustrative example

An illustrative example is provided hereby to showcase application of the proposed interpolation scheme. A 5-axis machining tool-path shown in Figure 5.4 is programmed with 3 consecutive tool-pose points. Tangential feedrate is set to F = 100 [mm/sec]. Feed pulse durations are computed from linear feed and TCP displacement. Tool's angular speed is calculated from Eq. (5.8). 2 FIR filters with time constants of T_1 = 100 [msec] and T_2 = 60 [msec] are utilized. Notice that FIR filtering elongates each block's duration by the total filter delay of $T_d = T_1 + T_2$. Thus, successive block velocity pulses are commanded with a "dwell" time of $T_{dwell} = T_d$ = 160 [msec]. As a result, at the end of each linear interpolation the tool undergoes a full-stop and P2P motion is achieved. As shown in Figure 5.4, both TCP and ORI vectors are interpolated with trapezoidal acceleration profiles. Linear and angular velocities are synchronized precisely. Furthermore, owing to direct spherical interpolation constant angular tool-axis' velocity [14] is preserved. Based on the interpolated tool-pose, inverse kinematics is applied, and joint position commands are generated. Readers should refer to Section 5.4 for the kinematics of the in-house controlled 5-axis machine tool. Since the kinematic transformation is continuous, all the drive commands are interpolated with smooth jerk-limited acceleration kinematics as demonstrated in Figure 5.4.

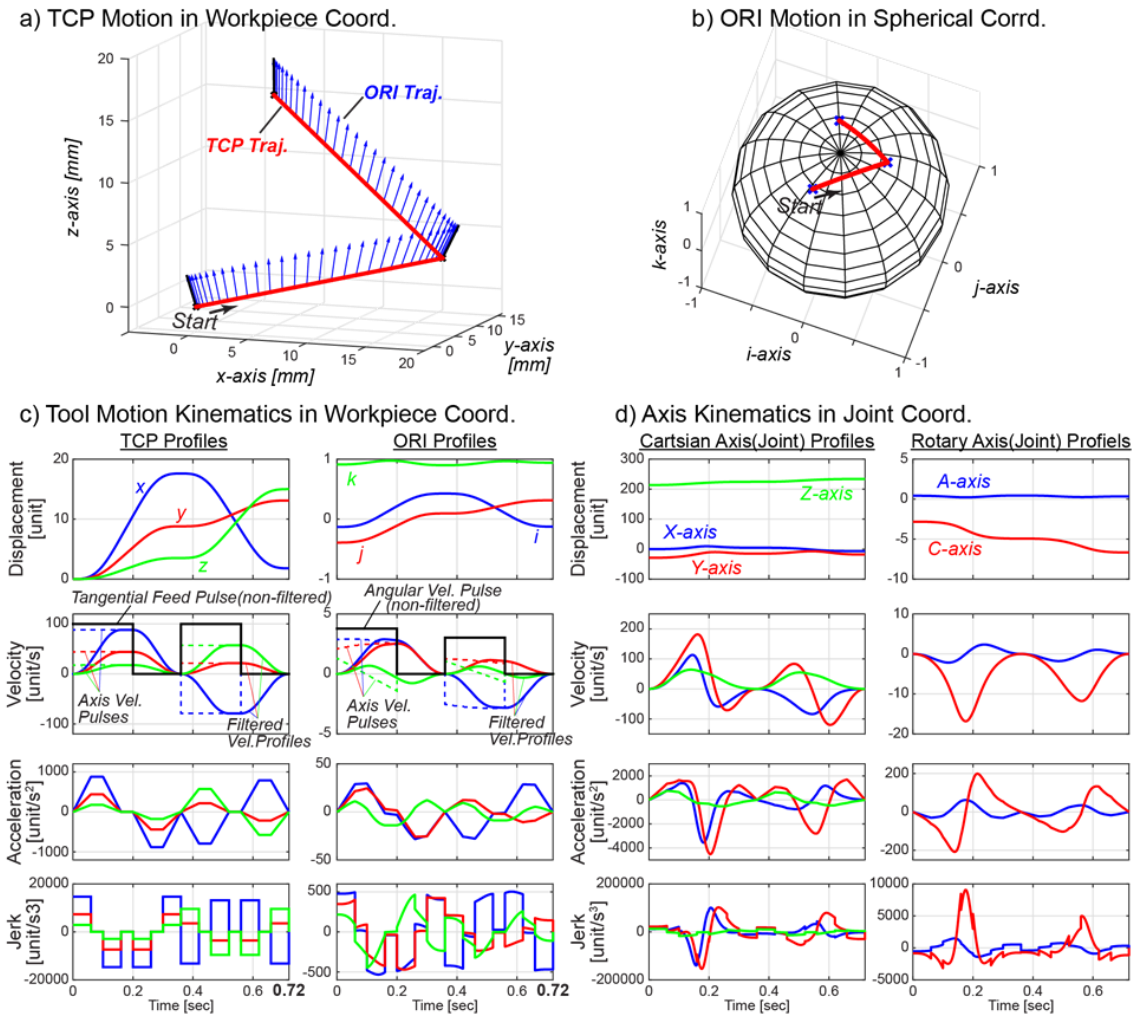

Figure 5.4: P2P interpolation example.

5.3. Non-stop linear interpolation of 5-axis tool-paths with local corner blending

Previous section discussed P2P linear interpolation of 5-axis machining tool-paths in workpiece coordinates. This section focuses on non-stop interpolation of 5-axis machining tool-paths with accurate local corner blending (smoothing). Two methods are presented for real-time interpolation targeting high-speed or high high-smoothness motion.

5.3.1. Non-stop interpolation based on dwell-time controlled blending (NS-DCB)

The first method is based on the dwell-time control, and it is called the non-stop dwell-time control path blending (NS-DCB). As presented in proceeding section, if consecutive blocks are commanded with a dwell-time of equal or larger than the filter delay, a full-stop and an exact P2P motion is realized. On the other hand, if consecutive velocity pulses are commanded without fully waiting for the filter delay to die out, tool's linear and angular feed directions are changed before the target pose is reached, and a corner blend is generated. The idea of utilizing dwell-time control to blend 2-axis Cartesian tool-paths with confined corner blending tolerance is already presented in [34], [75]. This approach is extended here for local blending of 5-axis tool-paths with synchronized TCP and ORI kinematics directly in the workpiece coordinates. This strategy is outlined in Figure 5.5, and detailed mathematical analysis is provided in the following.

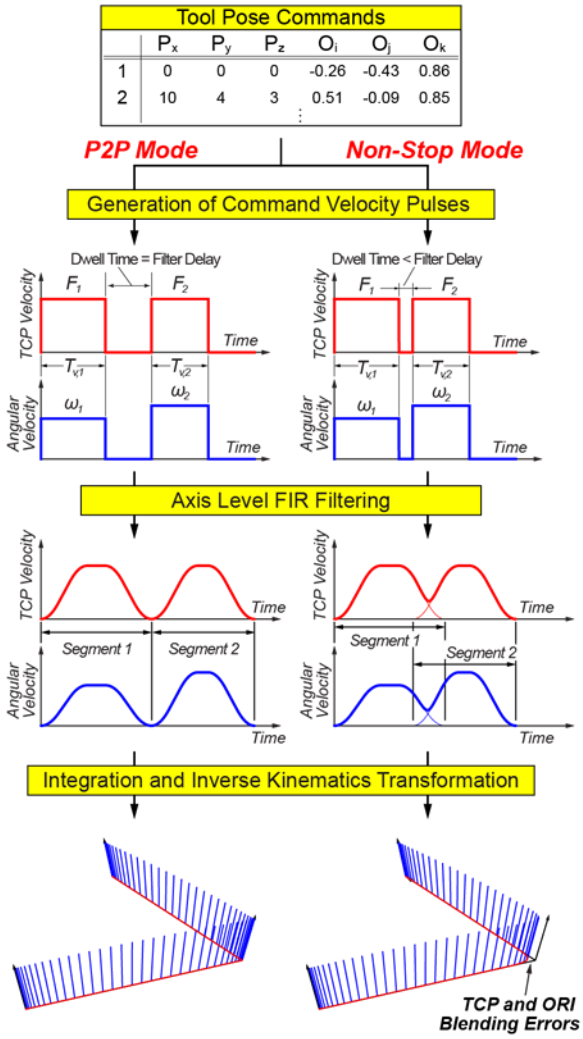

Figure 5.5: Overall Non-stop Interpolation of 5-axis Tool-paths based on Dwell-time Control Strategy.

Figure 5.6 illustrates application of dwell-time control for non-stop interpolation with TCP motion blending. A simple example with two consecutive feed blocks is shown in Figure 5.6a. As observed, when the dwell-time between consecutive blocks is set less than the total FIR filter delay $T_{dwell} < T_1 + T_2$, convolution of the successive feed block (F_2) is initiated before the current one is completed. As a result, feed direction is transitioned from t_1 to t_2 without stopping at the corner. Local blending (smoothing) starts at $\boldsymbol{P_s}$, at $t = T_{v,1} + T_{dwell}$ and ends at $\boldsymbol{P_e}$ when FIR filter's transient response dies out at $t = T_{v,1} + T_d$. Shortening the dwell-time initiates earlier cornering transition, and thus the time spend

during cornering transition increases yielding a larger blending trajectory and a greater deviation from the corner, i.e. segment's junction point, $\boldsymbol{P_2}$. Thus, cornering duration T_c plays a critical role in controlling corner blend geometry. The total FIR filtering delay, $T_d = T_1 + T_2$ is fixed. Cornering duration T_c becomes:

$$T_c = T_d - T_{dwell} \tag{5.15}$$

If the successive block feeds are identical $F_1 = F_2$, maximum corner blending error occurs in the middle of the cornering trajectory at $t = T_v + T_d - T_c/2$ as shown in Figure 5.6:

$$\varepsilon_{TCP} = \left\| \mathbf{P_m} - \mathbf{P_2} \right\| \tag{5.16}$$

The midpoint coordinates $\boldsymbol{P_m}$ can be calculated by the remaining distance towards $\boldsymbol{P_2}$ from current (*1st*) block's deceleration profile, which is donated as l_1, and by the added distance due to acceleration towards the successive (*2nd*) block donated as l_2 (See Figure 5.6a and b). Velocity profile during current block's deceleration is simply based on the trapezoidal acceleration profile from Eq. (5.3):

$$v_1^{'}(t) = \begin{cases} F_1 - \dfrac{1}{2}\dfrac{F_1}{T_1T_2}\ t - T_v\ ^2 & T_v \le t \le T_v + T_2 \\ F_1 - \dfrac{1}{2}\dfrac{F_1T_2}{T_1} - \dfrac{F_1}{T_1}\ t - \ T_v + T_2 & T_v + T_2 \le t \le T_v + T_1 \\ \dfrac{1}{2}\dfrac{F_1}{T_1T_2}\ \ T_v + T_1 + T_2\ \ - t\ ^2 & T_v + T_1 \le t \le T_v + T_1 + T_2 - T_k \end{cases} \tag{5.17}$$

and by setting the integration boundaries to $t_s = T_{v,1} + T_{dwell}$ and $t_e = T_{v,1} + T_d$ in Eq. (5.17), yields the remaining distance towards $\boldsymbol{P_2}$:

$$l_1 = \left\| \int_{T_v+T_d-T_c/2}^{T_v+T_d} v_1^{'} d\tau \right\| = \begin{cases} \dfrac{T_c^{\,3}}{48T_1T_2}F_1 & , \quad 0 \le T_c \le 2T_2 \\ \dfrac{4T_2^{\,2} - 6T_2T_c + 3T_c^{\,2}}{24T_1}F & , \quad 2T_2 < T_c \le T_1 + T_2 \end{cases} \tag{5.18}$$

Next, velocity profile during acceleration towards $\boldsymbol{P_3}$ is integrated to compute the l_2

$$l_2 = \left\| \int_0^{T_c/2} v_2^{'} d\tau \right\| = \begin{cases} \dfrac{T_c^{\,3}}{48T_1T_2} F_2 & , \quad 0 \le T_c \le 2T_2 \\ \dfrac{4T_2^{\,2} - 6T_2T_c + 3T_c^{\,2}}{24T_1} F_2 & , \quad 2T_2 < T_c \le T_1 + T_2 \end{cases} \tag{5.19}$$

Hence, midpoint of cornering blend is evaluated from the geometry as shown in Figure 5.6a,

$$\overrightarrow{\mathbf{P_m}} - \overrightarrow{\mathbf{P_2}} = l_2 \overrightarrow{\mathbf{t_2}} - l_1 \overrightarrow{\mathbf{t_1}} \tag{5.20}$$

and its Euclidian norm yields TCP blending error:

$$\varepsilon_{TCP} = \left\| l_2 \overrightarrow{\mathbf{t_2}} - l_1 \overrightarrow{\mathbf{t_1}} \right\| = \sqrt{l_1^{\,2} + l_2^{\,2} - 2l_1l_2\left(\overrightarrow{\mathbf{t_1}} \cdot \overrightarrow{\mathbf{t_2}}\right)} \tag{5.21}$$

It is worthwhile to notice that the inner product of consecutive feed vectors yields the cornering angle

$$\cos\left(\theta_{TCP}\right) = \vec{\mathbf{t}}_{\mathbf{1}} \cdot \vec{\mathbf{t}}_{\mathbf{2}} \tag{5.22}$$

and hence by plugging Eqs. (5.18) and (5.19) into Eq. (5.21), provides the final TCP blending error expression as a function of cornering duration T_c, FIR filter delays, T_1, T_2 and the cornering angle θ_{TCP} as:

$$\varepsilon_{TCP} = \begin{cases} \dfrac{T_c^{\,3}}{48T_1T_2} \sqrt{F_1^{\,2} + F_2^{\,2} - 2F_1F_2\cos\left(\theta_{TCP}\right)} & , \quad 0 \le T_c \le 2T_2 \\ \dfrac{4T_2^{\,2} - 6T_2T_c + 3T_c^{\,2}}{24T_1} \sqrt{F_1^{\,2} + F_2^{\,2} - 2F_1F_2\cos\left(\theta_{TCP}\right)} & , \quad 2T_2 < T_c \le T_1 + T_2 \end{cases} \tag{5.23}$$

Closed-form solution to the T_c for a desired TCP blending tolerance is calculated from above;

$$T_c = \begin{cases} \sqrt[3]{\dfrac{48T_1T_2\varepsilon_{TCP}}{\sqrt{F_1^{\,2} + F_2^{\,2} - 2F_1F_2\cos\left(\theta_{TCP}\right)}}} & , \quad 0 \le T_c \le 2T_2 \\ T_2 + \sqrt{\dfrac{8T_1\varepsilon_{pos}}{\sqrt{F_1^{\,2} + F_2^{\,2} - 2F_1F_2\cos\left(\theta_{TCP}\right)}} - \dfrac{T_2^{\,2}}{3}} & , \quad 2T_2 < T_c \le T_1 + T_2 \end{cases} \tag{5.24}$$

and the final dwell-time between consecutive feed blocks is simply:

$$T_{dwell,TCP} = T_d - T_c \tag{5.25}$$

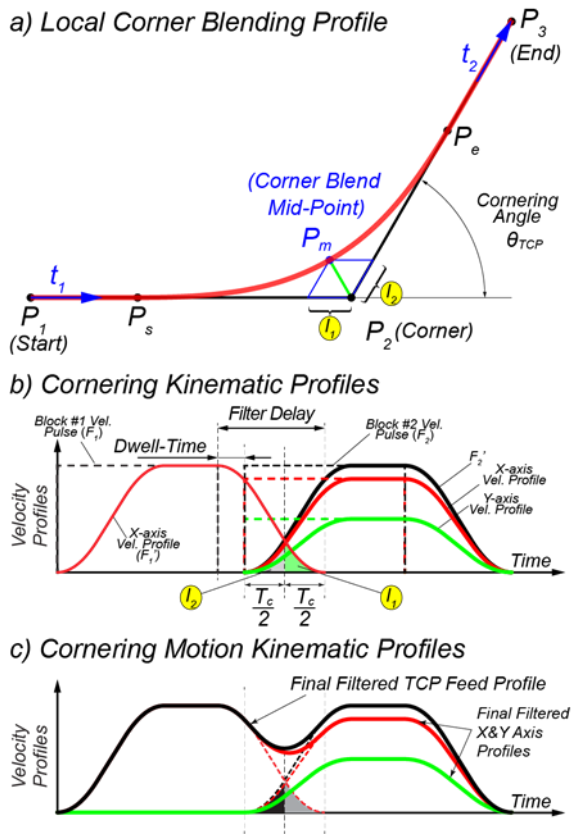

Figure 5.6: TCP blending error control.

Notice that the cornering angle, θ_{TCP} plays a critical role. Acute corners, $\theta_{TCP} > \pi/2$ demand larger velocity traverse, and thus require shorter cornering duration to satisfy desired blending tolerance. Eq. (5.23) captures this geometrical effect. Also notice that, proposed formulation assumes that the largest blending error occurs in the middle of the cornering trajectory. If $F_1 \neq F_2$, cornering trajectory is no longer symmetric around the bi-sector of $\boldsymbol{P_1P_2P_3}$. In this particular case, proposed formulation overestimates the blending error and can still be utilized safely for non-stop real-time tool-tip interpolation. This is due to the fact that proposed method computes contour error at the midpoint of the cornering trajectory.

In 5-axis machining, blending of ORI vectors and controlling the angular deviation is critical especially during flank milling operations [67], [76]. Above dwell-time control approach can also be applied for blending ORI vectors within specified angular orientation error tolerance ε_{ORI}. Figure 5.7 shows corner blend trajectory on the unit sphere and

corresponding angular velocity kinematics for $T_{dwell} < T_d$. As illustrated in Figure 5.7, when pseudo angular velocities (dO_i/dt, dO_j/dt, dO_k/dt) are FIR filtered, tool-orientation is blended with C^2 continuity and make a smooth transition around $\boldsymbol{O_2}$. Henceforth, the ORI blending error is controlled as follows.

$\boldsymbol{O_s}$ is the ORI vector at the start of the blend when successive block convolution starts, and $\boldsymbol{O_e}$ in Figure 5.7 is measured at the end of the blend. Similar to the Cartesian case, cornering trajectory is symmetric on the unit sphere when $\omega_1 = \omega_2$, and maximum angular deviation occurs at the midpoint, $\boldsymbol{O_m}$. θ_1 and θ_2 are analogous to l_1 and l_2 denoting remaining and added rotations of successive interpolation blocks around the corner. They are obtained by integrating the angular velocity kinematics:

$$\theta_i = \begin{cases} \dfrac{T_c^{\,3}}{48T_1T_2}\omega_i & , \quad 0 \le T_c \le 2T_2 \\ \dfrac{4T_2^{\,2} - 6T_2T_c + 3T_c^{\,2}}{24T_1}\omega_i & , \quad 2T_2 < T_c \le T_1 + T_2 \end{cases} \qquad \text{for } i = 1, 2 \tag{5.26}$$

Next, $\boldsymbol{O_2O_s^*O_mO_e^*}$ forms a spherical square on the unit sphere [78]. Considering the fact that the corner blending tolerance is typically very small, the geometry can be assumed as a parallelogram. Hence, the spherical cosine [78] law can be applied, and the orientation blending error ε_{ORI} can be expressed as a function of cornering angles:

$$\cos\left(\varepsilon_{ORI}\right) = \cos\left(\theta_1\right)\cos\left(\theta_2\right) + \sin\left(\theta_1\right)\sin\left(\theta_2\right)\cos\left(\theta_{ORI}\right) \tag{5.27}$$

where

$$\theta_{ori} = \arccos\left(\frac{\left(\mathbf{O_1} \times \mathbf{O_2}\right)\cdot\left(\mathbf{O_2} \times \mathbf{O_3}\right)}{\left\|\mathbf{O_1} \times \mathbf{O_2}\right\|\left\|\mathbf{O_2} \times \mathbf{O_3}\right\|}\right) \tag{5.28}$$

By plugging Eqs. (5.26) and (5.28) into Eq. (5.27) blending error ε_{ORI} for any given dwell-time T_{dwell} and FIR filter delay T_d can be calculated. The goal is to calculate the dwell-time required to satisfy desired angular blending tolerance. In order to find a closed-form solution, Maclaurin Series for the sine/cosine terms are used,

$$\begin{cases} \sin(\theta_i) = \theta_i - \dfrac{\theta_i^3}{3!} \\ \cos(\theta_i) = 1 - \dfrac{\theta_i^2}{2!} + \dfrac{\theta_i^4}{4!} \end{cases} \tag{5.29}$$

and Eq. (5.27) is linearized

$$\cos(\varepsilon_{ori}) = \left(1 - \frac{\theta_1^2}{2!} + \frac{\theta_1^4}{4!}\right)\left(1 - \frac{\theta_2^2}{2!} + \frac{\theta_2^4}{4!}\right) + \left(\theta_1 - \frac{\theta_1^3}{3!}\right)\left(\theta_2 - \frac{\theta_2^3}{3!}\right)\cos(\theta_{ORI}) \tag{5.30}$$

and organized as a *4th* order polynomial as:

$$aT_{temp}{}^4 + bT_{temp}{}^3 + cT_{temp}{}^2 + dT_{temp} + e = 0 \tag{5.31}$$

Notice that, accuracy of the above Maclaurin series approximation can be improved by adding more terms in Eq. (5.29). The use of only 2 terms leads to the above 4th order polynomial, which can be solved conveniently in real-time, and it does not introduce significant errors in practice.

When $T_c \leq 2T_2$, (See Eq. (5.26)) the polynomial coefficients in Eq. (5.31) become

$$\begin{cases} a = \dfrac{\omega_1{}^4\omega_2{}^4}{48^8 T_1^8 T_2{}^8 4!4!} \\ b = \dfrac{1}{48^6 T_1^6 T_2{}^6}\left(-\dfrac{\omega_1{}^4\omega_2{}^2 + \omega_1{}^2\omega_2{}^4}{2!4!} + \dfrac{\omega_1{}^3\omega_2{}^3}{3!3!}\cos(\theta_{ORI})\right) \\ c = \dfrac{1}{48^4 T_1^4 T_2{}^4}\left(\dfrac{\omega_1{}^4 + \omega_2{}^4}{4!} + \dfrac{\omega_1{}^2\omega_2{}^2}{2!2!} - \dfrac{\omega_1{}^3\omega_2 + \omega_1\omega_2{}^3}{3!}\cos(\theta_{ORI})\right) \\ d = \dfrac{1}{48^2 T_1^2 T_2{}^2}\left(-\dfrac{\omega_1{}^2 + \omega_2{}^2}{2!} + \omega_1\omega_2\cos(\theta_{ORI})\right) \\ e = 1 - \cos(\varepsilon_{ORI}) \end{cases} \tag{5.32}$$

where $T_c = \sqrt[6]{T_{temp}}$. When $T_c > 2T_2$ (See Eq. (5.26)), Eq. (5.31) must be solved using the updated polynomial coefficients as:

$$\begin{cases} a = \dfrac{\omega_1^{\,4}\omega_2^{\,4}}{24^8 T_1^8\, 4!4!} \\ b = \dfrac{1}{24^6 T_1^6}\left(-\dfrac{\omega_1^{\,4}\omega_2^{\,2}+\omega_1^{\,2}\omega_2^{\,4}}{2!4!}+\dfrac{\omega_1^{\,3}\omega_2^{\,3}}{3!3!}\cos\left(\theta_{ORI}\right)\right) \\ c = \dfrac{1}{24^4 T_1^4}\left(\dfrac{\omega_1^{\,4}+\omega_2^{\,4}}{4!}+\dfrac{\omega_1^{\,2}\omega_2^{\,2}}{2!2!}-\dfrac{\omega_1^{\,3}\omega_2+\omega_1\omega_2^{\,3}}{3!}\cos\left(\theta_{ORI}\right)\right) \\ d = \dfrac{1}{24^2 T_1^2}\left(-\dfrac{\omega_1^{\,2}+\omega_2^{\,2}}{2!}+\omega_1\omega_2\cos\left(\theta_{ORI}\right)\right) \\ e = 1-\cos\left(\varepsilon_{ORI}\right) \end{cases} \tag{5.33}$$

where $T_c = T_2 + \sqrt{\left(\sqrt{T_{temp}} - T_2^{\,2}\right)/3}$. In order to realize fastest cornering duration, the smallest real root of Eq. (5.31) should be selected. The dwell-time for satisfying angular blending tolerance then becomes:

$$T_{dwell,ORI} = T_d - T_c \tag{5.34}$$

Finally, the minimum of the dwell times calculated for TCP and ORI (Eq. (5.25) and (5.34)) errors must be selected so that both tolerances are respected:

$$T_{dwell} = \max\left(T_{dwell,TCP}, T_{dwell,ORI}\right) \tag{5.35}$$

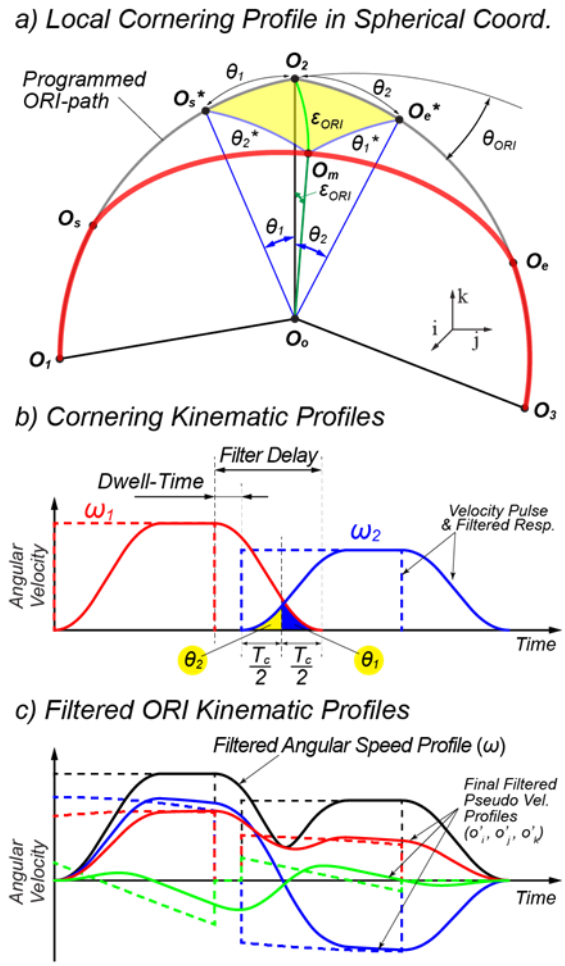

Figure 5.7: ORI blending error control.

5.3.2. Non-stop interpolation based on Velocity Controlled blending (NS-VCB)

A novel 5-axis cornering strategy is presented in the previous section. Dwell-time between consecutive feed blocks is adjusted to ensure non-stop and accurate interpolation of discrete 5-axis tool-paths. This strategy commands the feedrate to be altered from current block to the next one within the dwell-time. When the dwell time is short, it commands a large velocity change requiring higher acceleration. Although this approach provides rapid cornering that is suitable for roughing or semi-finishing operations, rapid change in machining feed may leave cutter marks on the part surface due to cutting force fluctuations and residual forced vibrations [9], [76]. This may jeopardize surface quality in

fine finishing operations. This section proposes a reduced acceleration cornering strategy tailored for precision and fine finishing on 5-axis machine tools. The approach presented in this subsection tries to control the cornering speed [23] to yield a reduced acceleration 5-axis TCP and ORI blending trajectory, and it is denoted as non-stop interpolation based on velocity-controlled blending (NS-VCB). When utilized interchangeably with the previous strategy, both rapid and accurate interpolation of 5-axis machining toolpaths can be realized for roughing and finishing operations.

In this approach, the dwell-time around corners is set to zero $T_{dwell} = 0$ and only the cornering speed is adjusted. Two feed pulses, F_1^* and F_2^*, at a duration of $T_d/2$ are created around the corner to be blended (See Figure 5.8). Those feed pulses are adjusted to control the blending error during cornering transition. Note that the original TCP feed pulses F_1 and F_2 are altered only around the corner (See Figure 5.8). Similarly, two angular velocity pulses ω_1^* and ω_2^* are introduced to the angular motion for blending the ORI vectors. Cornering duration becomes the filter delay, $T_c = T_d$. Assigned cornering velocity pulses F_1^*, F_2^*, ω_1^* and ω_2^* are determined based on the corner blending tolerance as follows.

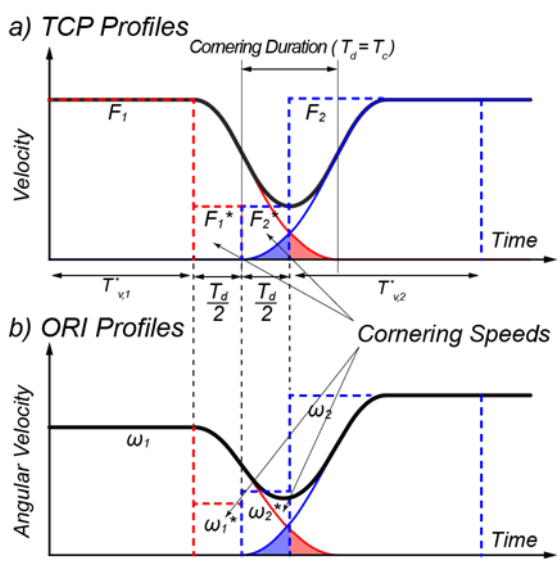

Figure 5.8: TCP and ORI blending kinematics based on velocity control.

Firstly, Eq. (5.23) is used as a basis, and T_{dwell} is set to zero to yield the following relationship between cornering speed and the TCP blending errors:

$$\varepsilon_{TCP} = \sqrt{F_1^{*2} + F_2^{*2} - 2F_1^* F_2^* \cos\left(\theta_{TCP}\right)}\left(\frac{3T_1^2 + T_2^2}{24T_1}\right) \tag{5.36}$$

Notice that F_1^* and F_2^* are the adjusted feed pulses, where α is the corresponding scaling factor,

$$\begin{cases} F_1^* = \alpha F_1 \\ F_2^* = \alpha F_2 \end{cases} \tag{5.37}$$

Plugging Eq. (5.37) into Eq. (5.36) shows that increasing cornering feed (or the scaling factor) linearly increases corner blending errors as:

$$\begin{aligned} \varepsilon_{TCP}^* &= \sqrt{F_1^{*2} + F_2^{*2} - 2F_1^* F_2^* \cos\left(\theta_{TCP}\right)}\left(\frac{3T_1^2 + T_2^2}{24T_1}\right) \\ &= \alpha\sqrt{F_1^2 + F_2^2 - 2F_1 F_2 \cos\left(\theta_{TCP}\right)}\left(\frac{3T_1^2 + T_2^2}{24T_1}\right) \\ &= \alpha\varepsilon_{TCP} \end{aligned} \tag{5.38}$$

Hence, the scaling factor α for a desired cornering tolerance can be solved from Eq. (5.38) as:

$$\alpha = \frac{\dfrac{24T_1}{3T_1^2 + T_2^2}}{\sqrt{F_1^2 + F_2^2 - 2F_1 F_2 \cos\left(\theta_{TCP}\right)}}\varepsilon_{TCP} \tag{5.39}$$

As shown in Figure 5.8b similar approach can be followed to control ORI blending errors as well. The dwell-time in Eq. (5.30) is set to zero and the ORI blending error ε_{ORI} is evaluated as a function of angular velocity pulses of consecutive segments as:

$$\begin{aligned} &\varepsilon_{ORI} = \arccos\left(\left(1 - \frac{\theta_1^2}{2!} + \frac{\theta_1^4}{4!}\right)\left(1 - \frac{\theta_2^2}{2!} + \frac{\theta_2^4}{4!}\right) + \left(\theta_1 - \frac{\theta_1^3}{3!}\right)\left(\theta_2 - \frac{\theta_2^3}{3!}\right)\cos\left(\theta_{ORI}\right)\right) \\ &\text{where } \theta_i = \frac{3T_1^2 + T_2^2}{24T_1}\omega_i\ , \quad i = 1, 2 \end{aligned} \tag{5.40}$$

Next, angular feedrate is also scaled by $\omega_1^* = \alpha\omega_1$ and $\omega_2^* = \alpha\omega_2$ around the corner, and Eq. (5.40) is expanded,

$$\begin{aligned}
&\frac{\theta_1^{*4}\theta_2^{*4}}{4!4!}+\left\{-\frac{\theta_1^{*4}\theta_2^{*2}}{2!4!}-\frac{\theta_1^{*2}\theta_2^{*4}}{2!4!}+\frac{\theta_1^{*3}\theta_2^{*3}}{3!3!}\cos\left(\theta_{ORI}\right)\right\}\\
&+\left\{\frac{\theta_1^{*4}}{4!}+\frac{\theta_2^{*4}}{4!}+\frac{\theta_1^{*2}\theta_2^{*2}}{2!2!}-\frac{\theta_1^{*3}\theta_2^{*}}{3!}\cos\left(\theta_{ORI}\right)-\frac{\theta_1^{*}\theta_2^{*3}}{3!}\cos\left(\theta_{ORI}\right)\right\}\\
&+\left\{-\frac{\theta_1^{*2}}{2!}-\frac{\theta_2^{*2}}{2!}+\theta_1^{*}\theta_2^{*}\cos\left(\theta_{ORI}\right)\right\}+1=\cos\left(\varepsilon_{ORI}{}^{*}\right)\\
&\text{where } \theta_1^{*}=\frac{3T_1^{\,2}+T_2^{\,2}}{24T_1}\omega_1\alpha,\ \text{and } \theta_2^{*}=\frac{3T_1^{\,2}+T_2^{\,2}}{24T_1}\omega_2\alpha
\end{aligned} \tag{5.41}$$

The scaling factor α to satisfy orientation blending errors is determined through solution of the following *4th* order polynomial,

$$a\alpha'^4+b\alpha'^3+c\alpha'^2+d\alpha'+e=0 \tag{5.42}$$

where

$$\begin{cases}
a=\dfrac{\left(3T_1^{\,2}+T_2^{\,2}\right)^8\omega_1^{\,4}\omega_2^{\,4}}{24^8T_1^8\,4!4!}\\
b=\dfrac{\left(3T_1^{\,2}+T_2^{\,2}\right)^6}{24^6T_1^6}\left(-\dfrac{\omega_1^{\,4}\omega_2^{\,2}+\omega_1^{\,2}\omega_2^{\,4}}{2!4!}+\dfrac{\omega_1^{\,3}\omega_2^{\,3}}{3!3!}\cos\left(\theta_{ORI}\right)\right)\\
c=\dfrac{\left(3T_1^{\,2}+T_2^{\,2}\right)^4}{24^4T_1^4}\left(\dfrac{\omega_1^{\,4}+\omega_2^{\,4}}{4!}+\dfrac{\omega_1^{\,2}\omega_2^{\,2}}{2!2!}-\dfrac{\omega_1^{\,3}\omega_2+\omega_1\omega_2^{\,3}}{3!}\cos\left(\theta_{ORI}\right)\right)\\
d=\dfrac{\left(3T_1^{\,2}+T_2^{\,2}\right)^2}{24^2T_1^2}\left(-\dfrac{\omega_1^{\,2}+\omega_2^{\,2}}{2!}+\omega_1\omega_2\cos\left(\theta_{ORI}\right)\right)\\
e=1-\cos\left(\varepsilon_{ORI}\right)
\end{cases} \tag{5.43}$$

and $\alpha=\sqrt{\alpha'}$

The smallest scaling factor obtained from Eq. (5.39) and Eqs. (5.42)-(5.43) is used to determine the cornering speed that satisfy both TCP and ORI blending errors.

On the other hand, if the original consecutive feedrate commands are not identical, $F_1 \neq F_2$ then the algorithm needs to be modified slightly. If the feed difference is large, $F_1 \ll F_2$ or $F_1 \gg F_2$, only the larger federate is reduced to satisfy the blending error tolerance. The algorithm is summarized in Figure 5.9 and blending errors are controlled as follows.

For instance, if only F_1 needs to be adjusted; $F_1^* = \alpha F_1$, ε_{TCP} is determined by plugging F_1^* into Eq. (5.36) as

$$\begin{aligned}\varepsilon_{TCP}^{\ *} &= \sqrt{F_1^{*2} + F_2^{\ 2} - 2F_1^* F_2 \cos\left(\theta_{TCP}\right)}\left(\frac{3T_1^{\ 2} + T_2^{\ 2}}{24T_1}\right) \\ &= \sqrt{\left(\alpha F_1\right)^2 + F_2^{\ 2} - 2\left(\alpha F_1\right)F_2 \cos\left(\theta_{TCP}\right)}\left(\frac{3T_1^{\ 2} + T_2^{\ 2}}{24T_1}\right)\end{aligned} \tag{5.44}$$

The scaling factor α for satisfying the desired tool-tip cornering tolerance is calculated;

$$\alpha = \frac{F_2 \cos\left(\theta_{TCP}\right) + \sqrt{F_2^{\ 2}\cos^2\left(\theta_{TCP}\right) - \left(F_2^{\ 2} - \varepsilon_{TCP}^{\ 2}\left(\frac{24T_1}{3T_1^{\ 2} + T_2^{\ 2}}\right)^2\right)}}{F_1} \tag{5.45}$$

When F_2 is adjusted as $F_2^* = \alpha F_2$ instead of F_1, then scaling factor α can be determined in the same way of Eqs. (5.44)-(5.45). Similarly, when one of the angular velocity pulses is only adjusted, i.e. $\omega_1^* = \alpha\omega_1$, the scaling factor α is computed by substituting the adjusted angular velocity pulse ω_1^* into Eq. (5.40) as

$$a\alpha^4 + b\alpha^3 + c\alpha^2 + d\alpha + e = 0 \tag{5.46}$$

where

$$\begin{cases} a = \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^8}{24^8 T_1^8}\dfrac{\omega_1^{\ 4}\omega_2^{\ 4}}{4!4!} - \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^6}{24^6 T_1^6}\dfrac{\omega_1^{\ 4}\omega_2^{\ 2}}{2!4!} + \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^4}{24^4 T_1^4}\dfrac{\omega_1^{\ 4}}{4!} \\ b = \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^6}{24^6 T_1^6}\dfrac{\omega_1^{\ 3}\omega_2^{\ 3}}{3!3!}\cos\left(\theta_{ORI}\right) - \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^4}{24^4 T_1^4}\dfrac{\omega_1^{\ 3}\omega_2}{3!}\cos\left(\theta_{ORI}\right) \\ c = -\dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^6}{24^6 T_1^6}\dfrac{\omega_1^{\ 2}\omega_2^{\ 4}}{2!4!} + \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^4}{24^4 T_1^4}\dfrac{\omega_1^{\ 2}\omega_2^{\ 2}}{2!2!} - \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^2}{24^2 T_1^2}\dfrac{\omega_1^{\ 2}}{2!} \\ d = -\dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^4}{24^4 T_1^4}\dfrac{\omega_1\omega_2^{\ 3}}{3!}\cos\left(\theta_{ORI}\right) + \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^2}{24^2 T_1^2}\omega_1\omega_2\cos\left(\theta_{ORI}\right) \\ e = \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^4}{24^4 T_1^4}\dfrac{\omega_2^{\ 4}}{4!} - \dfrac{\left(3T_1^{\ 2} + T_2^{\ 2}\right)^2}{24^2 T_1^2}\dfrac{\omega_2^{\ 2}}{2!} + 1 - \cos\left(\varepsilon_{ORI}\right) \end{cases} \tag{5.47}$$

And when ω_2 is adjusted as $\omega_2^* = \alpha\omega_2$, the scaling factor α can be determined in the same way as in ω_1 is adjusted.

Finally, original feed pulse durations T_{v1} and T_{v2} are adjusted so that the total commanded displacement is kept unchanged. Notice that the area under original feed pulse is the total travel distance of the linear block, $L_i = F_i T_{vi}$, $i = 1,2$. Once feed pulses amplitudes around the corner (see Figure 5.8) are reduced, duration of the original feed pulse must also be modified to keep the total travel distance unchanged. Adjusted pulse durations can be computed from Figure 5.8 by using the geometrical relationships as:

$$T_{v,1}^{\ *} = \frac{L_1 - F_1^{\ *}\dfrac{T_d}{2}}{F_1} \text{ and } T_{v,2}^{\ *} = \frac{L_1 - F_2^{\ *}\dfrac{T_d}{2}}{F_2} \tag{5.48}$$

and the total cycle time elongation by this cornering approach becomes $(T_{v,1} + T_{v,2}^{*}) - (T_{v,1} + T_{v,2}^{*}) + T_d$.

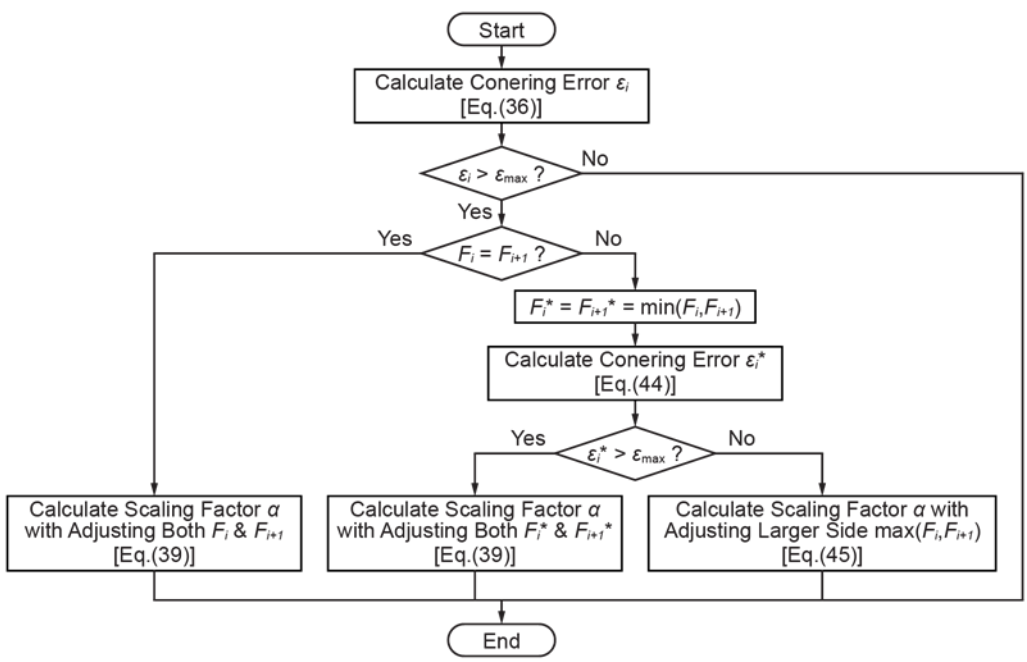

Figure 5.9: Blending velocity calculation flow chart.

5.3.3. Illustrative examples

This section provides simulation-based case studies to validate effectiveness and functionality of proposed schemes presented in section 5.3. Figure 5.10 shows the non-stop

path interpolation and corner blending techniques applied to a single corner. The G-code consist of 3 discrete tool poses given in Table 5.1. Programmed feedrate is F = 100 [mm/sec], and cornering tolerances are set as ε_{TCP} = 800 [μm] for TCP and ε_{ORI} = 0.02 [rad] for the tool orientation, respectively. 2 FIR filters with time constants, T_1 = 100 [msec] and T_2 = 60 [msec], are used to interpolate the toolpath. Notice that each linear pass is 20 [mm] long, which commands 2 identical 200 [msec] feed pulses. Corresponding angular velocity pulse amplitudes are calculated to generated synchronized tool motion as described in Section 5.2.

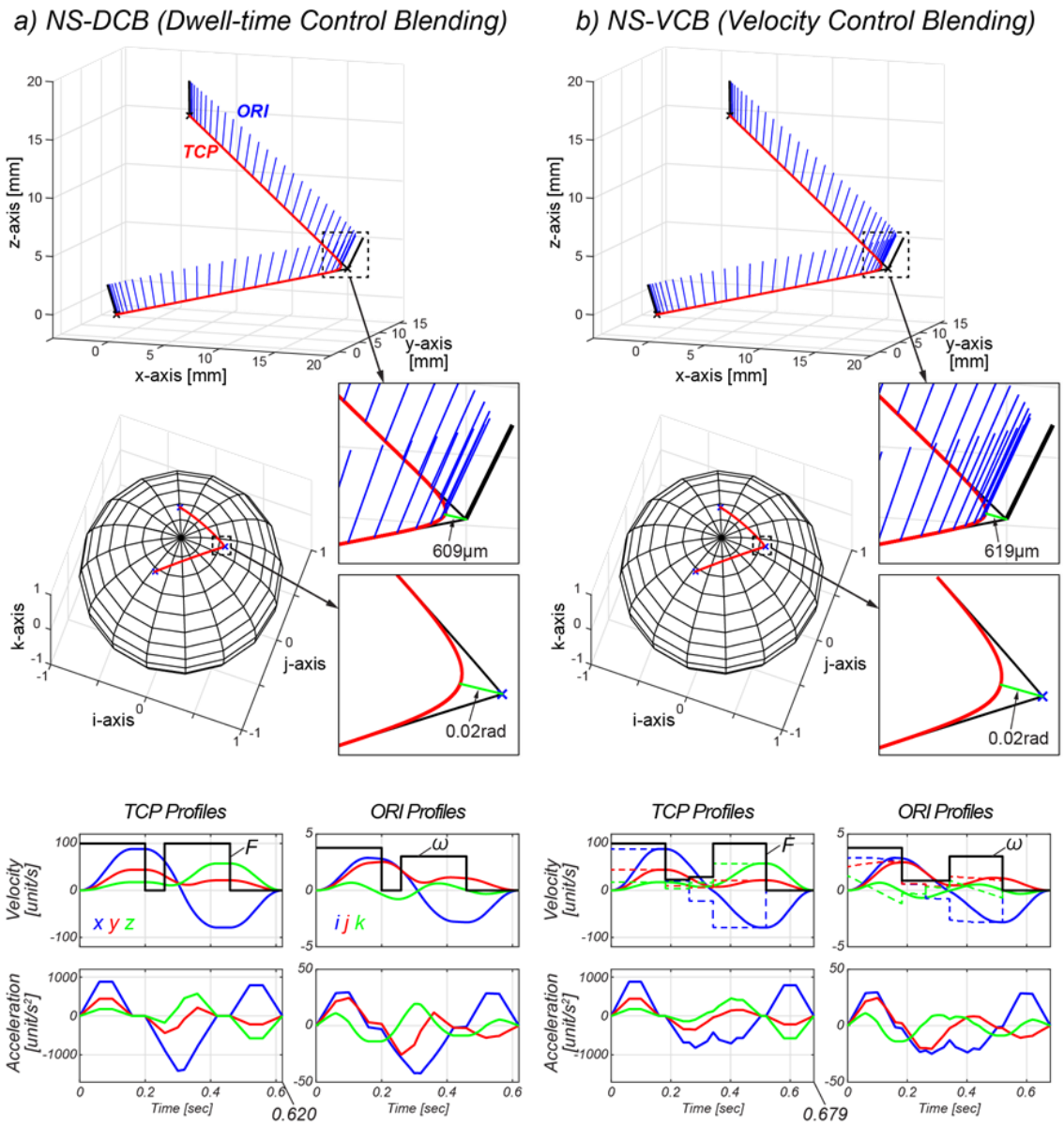

Figure 5.10: Non-stop interpolation illustrative example.

Table 5.1: Single corner tool-path pose points.

	Position [mm]			*Orientation []*		
	P_x	P_y	P_z	O_i	O_j	O_k
1	0.0000	0.0000	0.0000	-0.1302	-0.3906	0.9113
2	17.6090	8.8045	3.5218	0.4262	0.0947	0.8997
3	1.8059	13.1032	15.0015	-0.1255	0.3137	0.9412

The dwell-time based blending technique NS-DCB is applied in Figure 5.10a, and velocity-controlled blending one NS-VCB is applied in Figure 5.10b. As shown, in both cases, cornering tolerances are well respected. Notice that the ORI tolerance is the limiting one and the TCP tolerance is below the set value. This is just a coincidence. It all depends on the tool-path and user set tolerance values. As described in the previous sections, NS-DCB and NS-VCB methods provide different cycle times. Total cycle time for interpolating this tool-path is 0.620 [sec] and 0.679 [sec], respectively. The cornering durations are measured as 100.4 [msec] and 160 [msec], respectively. The NS-DCB technique is 8.7% faster than the NS-VCB method. In return, maximum acceleration of the dwell-time control technique is 70% larger. Hence, NS-DCB is more suitable for roughing and semi-finishing operations whereas the NS-VCB is more favorable in finishing passes since it demands less acceleration and induces less velocity fluctuation. Experimental results section benchmarks overall contouring errors to showcase the difference as well.

Additionally, Figure 5.11 is provided to clearly illustrate the cycle-time difference between the proposed techniques. As shown, for the same cornering angle, the NS-VCB approach delivers longer cycle times and demands less acceleration. As a result, velocity fluctuation in corner blending motion is smaller. Please notice that the trend in the cycle time with cornering error is linear and accurately captured by Eqs. (5.38) and (5.39). Figure 5.11c depicts required cycle time as the cornering angle changes.

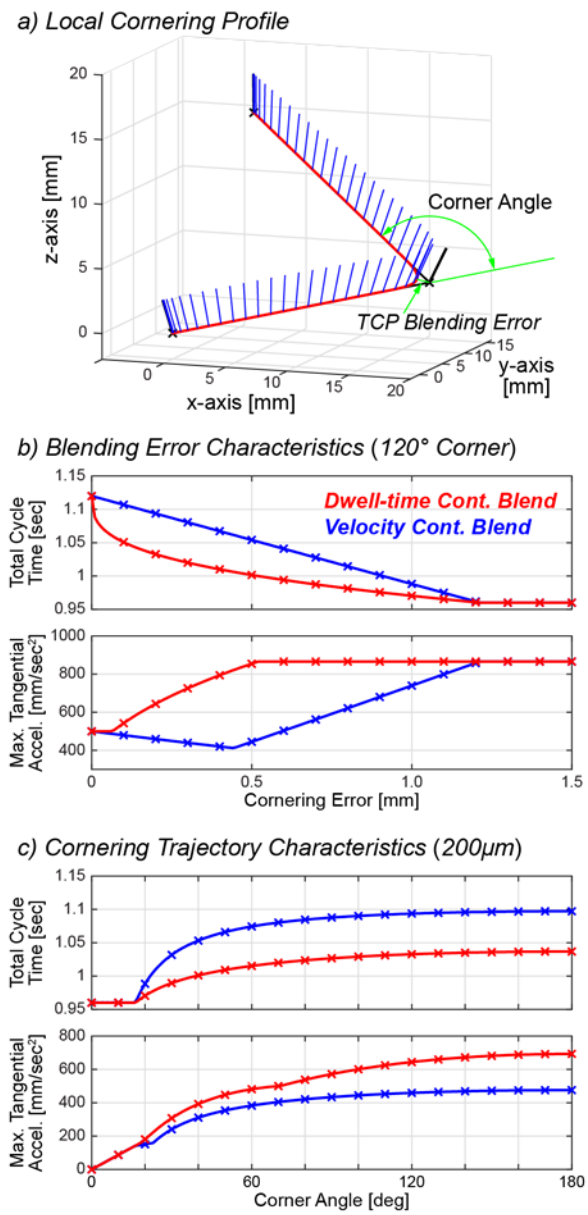

Figure 5.11: Benchmark between NS-DCB and NS-VCB schemes.

5.4. Experimental results

In addition to the simulation studies provided in the previous section, experimental validations on an actual 5-axis machine tool are presented in this section. The experimental machine tool is shown in Figure 5.12. It is a vertical 5-axis milling machine with A and C

rotary axes on the worktable side. The machine's inverse kinematics is provided in [68], [69]. The axes of the machine are controlled by in-house developed NC system based on the Dspace® hardware system. PID controllers [77] for each axis are implemented and position bandwidths are matched for synchronized motion at ~20 [Hz] [69]. PID tuning for 5-axis machine requires caution. Rotary and cartesian position loop dynamics must be tuned so that their tracking bandwidths are matched. This helps attenuate contouring errors due to dynamic synchronization. The closed-loop sampling time for the servo control system is set to 1 [kHz].

Figure 5.12: Experimental 5-axis machine tool.

The star shaped tool-path shown in Figure 5.13 is commanded for experimental validation. The tool-path consists of 15 number of linear segments and the corresponding G-code is summarized in Table 5.2. Figure 5.13a shows Cartesian TCP path and Figure 5.13b shows the ORI path on spherical coordinates. In order to provide a fair and informative comparison, the tool-path is interpolated with 4 different FIR based interpolation schemes: (1) P2P interpolation scheme presented in Section 5.2, (2) Non-stop interpolation scheme using NS-DCB is presented in Section 5.3, (3) Non-stop interpolation scheme using the NS-VCB presented in Section 5.3 and (4) direct blending of joint

commands (NS-JCB). In NS-JCB, non-stop motion is realized by FIR filtering the joint (axis) velocity pulses directly in the joint coordinates. Therefore, linear interpolation is not performed in workpiece coordinates, instead joint positions are interpolated linearly between start and end points of each linear block in the tool-path file [71]. Notice that this approach does not generate coordinated tool motion and introduces large path following (contour) errors. In order to attenuate severity of those fluctuations, extra way-points are added along long linear segments.

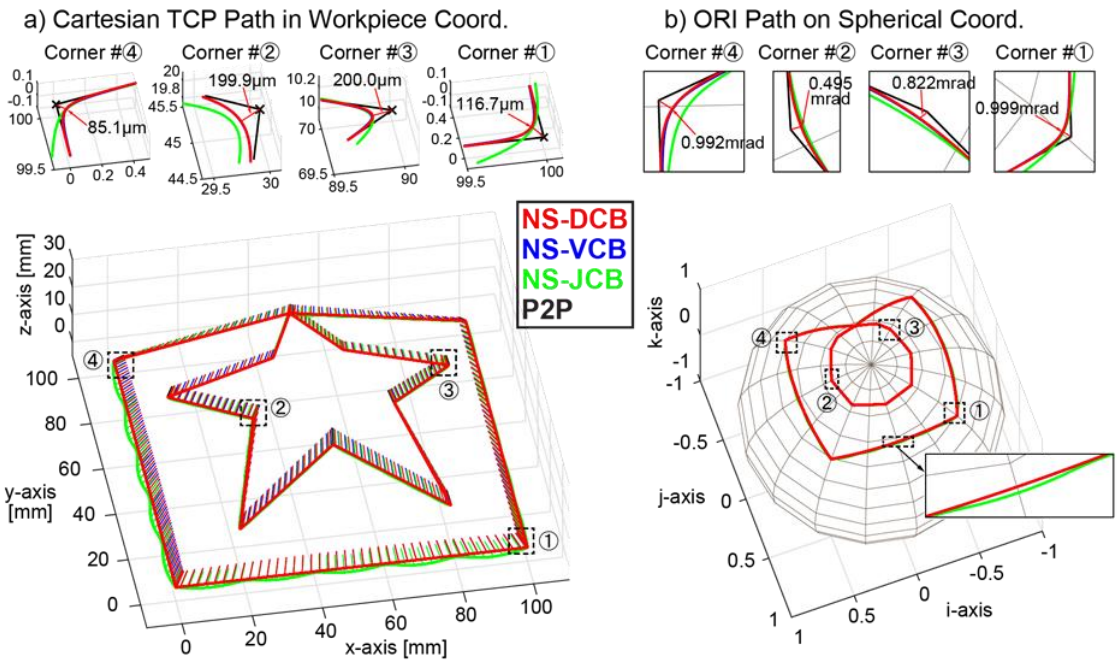

Figure 5.13: Star shaped tool-path.

Table 5.2: Shaped tool-path points.

	Position [mm]			*Orientation []*		
	P_x	P_y	P_z	O_i	O_j	O_k
1	0	0	0	0.5025	0.5025	0.7035
2	100	0	0	-0.5025	0.5025	0.7035
3	100	100	0	-0.5025	-0.5025	0.7035
4	50	100	10	0.0000	-0.3511	0.9363
5	40	70	20	0.1562	-0.3123	0.9370
6	10	70	10	0.2357	-0.2357	0.9428
7	30	45	20	0.3162	0.0000	0.9487
8	20	10	10	0.2357	0.2357	0.9428
9	50	30	20	0.0000	0.3162	0.9487
10	80	10	10	-0.2357	0.2357	0.9428
11	70	45	20	-0.3162	0.0000	0.9487
12	90	70	10	-0.2357	-0.2357	0.9428
13	60	70	20	-0.1562	-0.3123	0.9370
14	50	100	10	0.0000	-0.3511	0.9363
15	0	100	0	0.5025	-0.5025	0.7035
16	0	0	0	0.5025	0.5025	0.7035

For all the cases 2 FIR filters are used with time constants of $T_{d,1}$ = 100 [msec] and $T_{d,2}$ = 60 [msec]. As described in Section 5.1 this only requires 2 multiplications and 4 additions at each sampling interval. Since interpolation is performed in workpiece coordinates, calculations must be done for each Cartesian (x,y,z) and spherical (i,j,k) coordinates resulting in 12 multiplication and 24 addition operations to be performed in real-time. This is still very well within the computational power of modern real-time processors. It should also be noted that a polynomial root-finding algorithm must implemented for confining the orientation smoothing errors from Eqs. (5.31) and (5.42). However, this is a very standard procedure and can be realized in real-time.

The corner smoothing tolerance is set to ε_{TCP} = 200 [micron] and ε_{ORI} = 1 [mrad]. Figure 5.13 shows the final interpolated tool-path. Firstly, as expected, sharp corners are realized by the P2P interpolation scheme. The non-stop interpolation methods NS-DCB and NS-VCB proposed in this paper generate smooth corner profiles within the set error tolerances (See zoomed figures). Linear synchronous interpolation of tool-tip and tool-orientation is achieved. Both TCP and ORI trajectories are blended accurately around corners. Joint-space interpolation (NS-JCB) also generates non-stop motion. However, as seen in the tool-path it cannot guarantee linear tool motion interpolation in workpiece coordinates. Due to non-linear kinematics of 5-axis machine tool, linearly interpolating joint positions does not ensure linear tool interpolation in workpiece coordinates. The resultant tool-path deviates from the reference leaves large contour errors due to interpolation being performed in the joint coordinates. In order to attenuate the contour deviation, extra points are added along long linear segments during NS-JCB interpolation. Although this reduces the contouring errors; in return, it creates large velocity fluctuations in motion kinematics potentially degrading surface finish quality. This is another known drawback of joint space interpolation. Most CAM systems generate densely discretized sparse tool-paths to alleviate this problem [67].

Figure 5.14 presents TCP and ORI velocity profiles. The tangential TCP feedrate profile is shown in Figure 5.14a. As observed P2P motion planning generates the slowest cycle time since it commands stop and go motion at every corner. Both NS-DCB and NS-JCB approaches provide faster trajectories. However, as shown in Figure 5.13, the NS-JCB cannot be used for accurate 5-axis machining. It cannot ensure linear tool interpolation and violent feed fluctuations may destroy the surface finish. The NS-VCB approach is the second fastest interpolation scheme. Tool motion kinematics are shown in Figure 5.15, and axis kinematic profiles are presented in Figure 5.16. They validate that fact that jerk-limited smooth axis motion kinematics are generated by the proposed interpolation schemes.

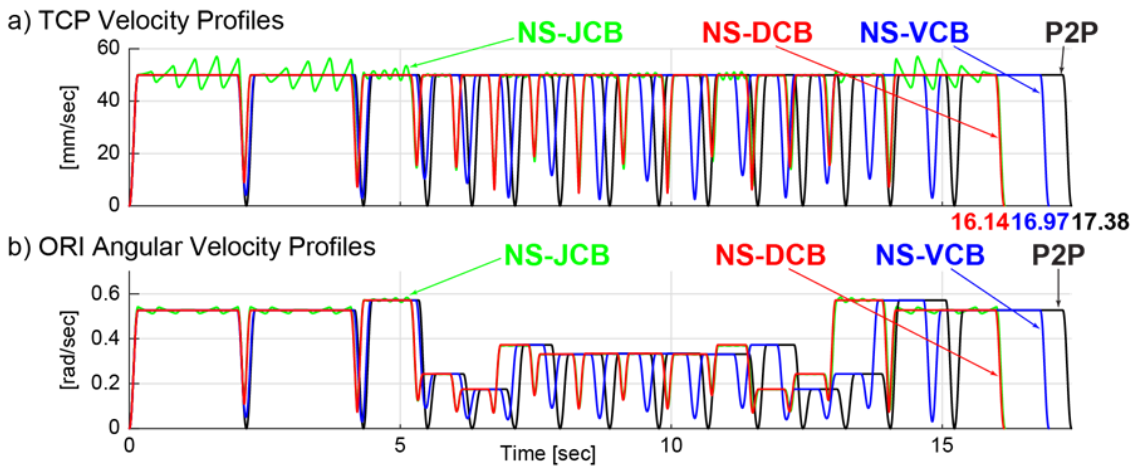

Figure 5.14: Interpolated kinematic profiles along star shaped tool-path.

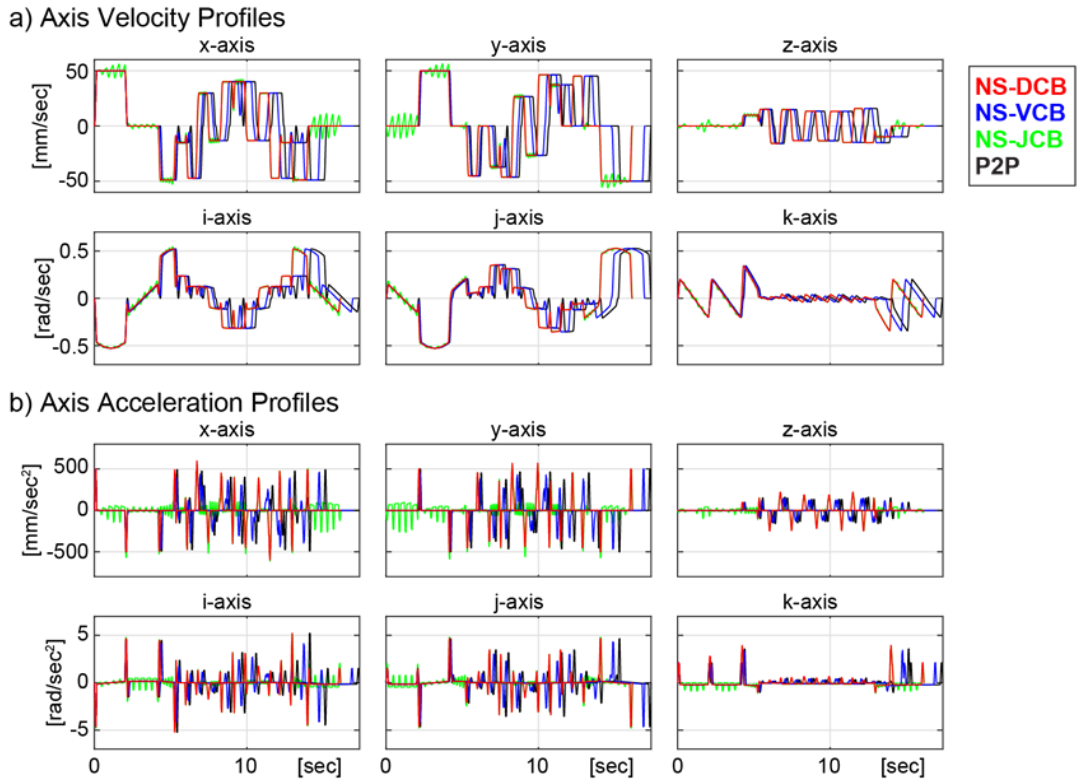

Figure 5.15: Interpolated tool motion kinematic profiles along star shaped tool-path.

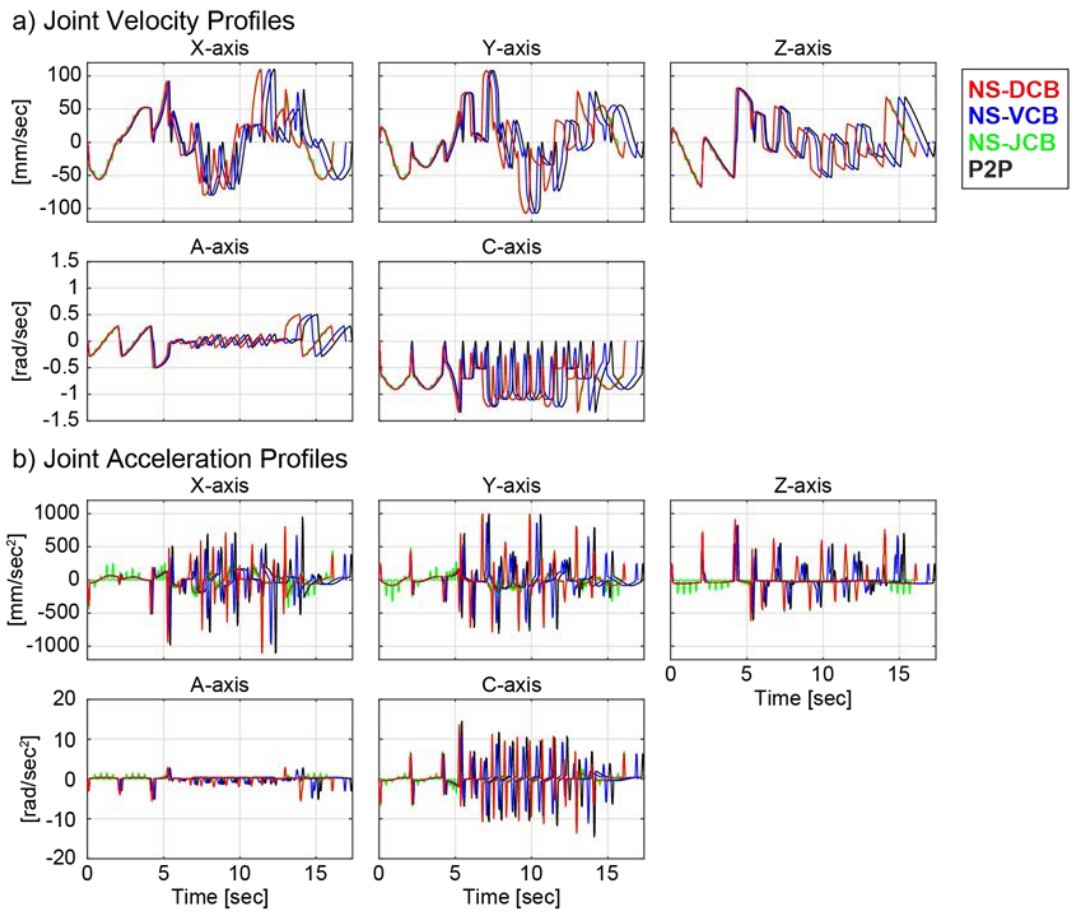

Figure 5.16: Interpolated axis kinematic profiles along star shaped tool-path.

Finally, the 5-axis contouring performance for the interpolation schemes are measured in Figure 5.17. Contour errors are calculated as the normal deviation from the commanded TCP and ORI paths [79]. They assess how accurately feedback controllers [77] track generated kinematic trajectories. If the interpolation scheme commands smooth feed profiles, generated tool-path is followed more accurately. In contrast, if the interpolation scheme demands rapidly varying feedrate and large accelerations, overall contouring performance degrades, which may deteriorate the surface finish quality as well. The following observations can be made from the contour error trend measured in Figure 5.17. Firstly, during linear interpolation contouring errors are small if the axis tracking bandwidths are matched [77]. This characteristic is observed in Figure 5.17 with an exception. A large contour error peak is visible at t = 4.2 [sec]. This peak occurs precisely at a velocity reversal where all the 5 axes change their motion direction. When the axes

change their motion direction, coulomb friction disturbance with Stribeck effect [9] kicks in and induces large tracking errors. Next, every time a corner is interpolated contour errors show smaller peaks. This characteristic is expected since velocity and acceleration transients are commanded, and hence servo lag induces contour error peaks. The NS-DCB algorithm shows largest contour error peaks since it demands greater acceleration. In contrast, the NS-VCB method shows smallest peaks. Overall RMS value of the contour errors also validate these characteristics. The NS-VCB demands overall ~15% less acceleration and hence it also induces 10% less contour errors. The NS-DCB provides the fastest cycle time whereas the NS-VCB requires longer cycle time, but in return it commands smoother provides and induces smaller contour errors making it more suitable for precision finishing operations. Finally, joint space interpolation showcases the largest contour errors. As explained in the previous paragraphs, this is mainly due to linear interpolation performed in the joint space rather than in the workpiece coordinates.

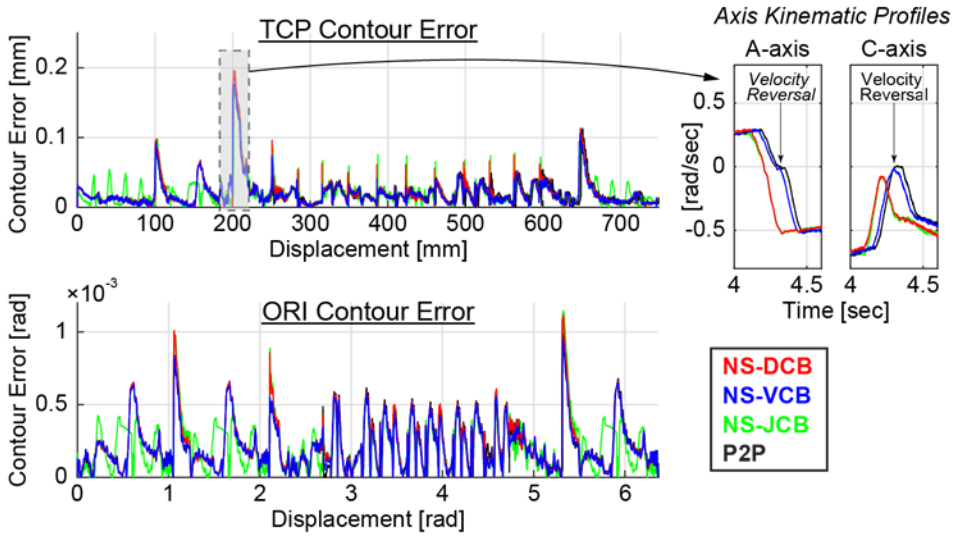

Figure 5.17: Experimental contouring errors.

5.5. Conclusions

This paper presented novel real-time (online) interpolation techniques for 5-axis machine tools. It shows that synchronized translational and rotational motion of the tool

can be interpolated directly in the workpiece coordinates using FIR (moving average) filtering. It is computationally lightweight and ensures accurate linear interpolation of tool motion. Direct linear interpolation of tool orientation in the workpiece coordinates has been a bottleneck in 5-axis machining. This paper provides a very novel solution to this problem. Tool orientation can be interpolated linearly on the great sphere in synch with the tool tip. In addition, 2 types of corner smoothing algorithms are also developed based on the FIR interpolation scheme. Both algorithms are simple and provide accurate smoothing of sharp corners to realize non-stop 5-axis interpolation. Illustrative examples and experimental results validate the significant contribution of the proposed algorithms. Overall cycle time can be reduced up to 40~50% as compared to conventional P2P interpolation. On the other hand, overall contouring accuracy can be improved up to 75% as compared to basic joint based interpolation. The algorithms and techniques proposed in this paper can be applied to increase efficiency and dynamic accuracy of modern 5-axis machine tools.

6. Conclusions

This thesis has proposed trajectory generation algorithms for machine tools and industrial robots to generate smooth and accurate motion. Novel kinematic corner smoothing techniques are developed to interpolate high-speed feed motion along discrete tool-paths by fully utilizing kinematic limits of the drives. FIR filtering based trajectory generation techniques are also introduced to mitigate any unwanted vibrations originating due to structural dynamics of the machinery. Finally, 6DOF tool pose interpolation is addressed, and a novel FIR based blending algorithm is developed for synchronized interpolation of tool's translational and rotational motion. Contributions of this thesis are summarized as follows.

Chapter 2 presents a Local Kinematic Corner Smoothing (LKCS) algorithm, which can blend cartesian axis velocities around sharp corners with jerk limited acceleration transitions and generates symmetric rounded corner profiles within confined geometric tolerances. This novel algorithm does not require any splining and provides fully analytical solution to the time optimal corner blending problem. The proposed algorithms can increase productivity up to 30-50% as compared to the state of the art blending and P2P interpolation techniques.

The LKCS technique from Chapter 2 is extended to interpolate short-segmented linear tool-paths in Chapter 3, and it is called the Global Kinematics Corner Smoothing (GKCS). Short segmented tool-paths is widely used in aerospace and die-and-mould machining. The proposed GKCS technique can generate near time optimal, smooth and rapid motion. As compared to blending local corners smoothly, GKCS can generate continuous contouring paths and improve the cycle time up 10–15%. The cycle time improvement against P2P interpolation can be up to 50%.

Chapter 4 presented FIR filtering based interpolation techniques, which can generate a non-stop contouring motion along cartesian machining tool-paths. Owing to the simple FIR filtering structure, proposed scheme can interpolate linear and circular paths with high kinematic continuity and minimum computational load making it suitable for real-time processors. The proposed block timing technique considers change in the feed

direction and the total delay in the filter chain to generate accurate non-stop rapid feed motion. These FIR based trajectory generation techniques control both contour errors and frequency spectrum of reference trajectories. As compared to the state-of-the-art technique, these techniques eliminate unwanted vibrations and reduce the cycle time up to ~20% while utilizing same level of acceleration for modern NC systems.

Chapter 5 presented FIR filtering based interpolation of tool-pose motion for accurate 5-axis machining. This novel technique can interpolate tool tip and tool orientation motion by applying FIR filtering synchronously in cartesian and spherical coordinates. As compared to the state of the art joint coordinate based interpolation, proposed technique eliminates fluctuations of the finishing surface and realizes accurate feed motion. It can improve overall machining cycle times on 5-axis machine tools and can be applied on high speed industrial robots.

Bibliography

[1] Y.-K. Choi and A. Banerjee, "Tool path generation and tolerance analysis for free-form surfaces," *Int. J. Mach. Tools Manuf.*, vol. 47, no. 3, pp. 689–696, Mar. 2007.

[2] X. G. P. Schuurbiers, "Blending in pick and place applications," pp. 1–67, 2007.

[3] K. Zhang, J.-X. Guo, and X.-S. Gao, "Cubic spline trajectory generation with axis jerk and tracking error constraints," *Int. J. Precis. Eng. Manuf.*, vol. 14, no. 7, pp. 1141–1146, 2013.

[4] K. Erkorkmaz and Y. Altintas, "Quintic spline interpolation with minimal feed fluctuation," *J. Manuf. Sci. Eng.*, vol. 127, no. 2, pp. 339–349, 2005.

[5] M. K. Jouaneh, Z. Wang, and D. A. Dornfeld, "Trajectory planning for coordinated motion of a robot and a positioning table. I. Path specification," *IEEE Trans. Robot. Autom.*, vol. 6, no. 6, pp. 735–745, 1990.

[6] S. J. Yutkowitz, "Apparatus and method for smooth cornering in a motion control system," Jul-2005.

[7] H. Zhao, L. Zhu, and H. Ding, "A real-time look-ahead interpolation methodology with curvature-continuous B-spline transition scheme for CNC machining of short line segments," *Int. J. Mach. Tools Manuf.*, vol. 65, pp. 88–98, 2013.

[8] B. Sencer, K. Ishizaki, and E. Shamoto, "A curvature optimal sharp corner smoothing algorithm for high-speed feed motion generation of NC systems along linear tool paths," *Int. J. Adv. Manuf. Technol.*, vol. 76, no. 9–12, pp. 1977–1992, 2015.

[9] Y. Altintas, A. Verl, C. Brecher, L. Uriarte, and G. Pritschow, "Machine tool feed drives," *CIRP Ann.*, vol. 60, no. 2, pp. 779–796, 2011.

[10] B. Sencer, Y. Altintas, and E. Croft, "Feed optimization for five-axis CNC machine tools with drive constraints," *Int. J. Mach. Tools Manuf.*, vol. 48, no. 7–8, pp. 733–745, 2008.

[11] J. M. Hyde and W. P. Seering, "Using input command pre-shaping to suppress multiple mode vibration," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 2604–2609.

[12] Y.-R. Hwang and C.-S. Liang, "Cutting errors analysis for spindle-tilting type 5-axis NC machines," *Int. J. Adv. Manuf. Technol.*, vol. 14, no. 6, pp. 399–405, 1998.

[13] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*, vol. 2. Citeseer, 1998.

[14] R. V. Fleisig and A. D. Spence, "A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining," *Comput.-Aided Des.*, vol. 33, no. 1, pp. 1–15, 2001.

[15] X. Beudaert, S. Lavernhe, and C. Tournier, "5-axis local corner rounding of linear tool path discontinuities," *Int. J. Mach. Tools Manuf.*, vol. 73, pp. 9–16, 2013.

[16] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 2012.

[17] Y. Koren and R.-S. Lin, "Five-axis surface interpolators," *CIRP Ann.*, vol. 44, no. 1, pp. 379–382, 1995.

[18] Q. G. Zhang and R. B. Greenway, "Development and implementation of a NURBS curve motion interpolator," *Robot. Comput.-Integr. Manuf.*, vol. 14, no. 1, pp. 27–36, 1998.

[19] J. M. Langeron, E. Duc, C. Lartigue, and P. Bourdet, "A new format for 5-axis tool path computation, using Bspline curves," *Comput.-Aided Des.*, vol. 36, no. 12, pp. 1219–1229, 2004.

[20] F.-C. Wang, P. K. Wright, B. A. Barsky, and D. C. H. Yang, "Approximately arc-length parametrized C3 quintic interpolatory splines," *J. Mech. Des.*, vol. 121, no. 3, pp. 430–439, 1999.

[21] S. D. Timar, R. T. Farouki, T. S. Smith, and C. L. Boyadjieff, "Algorithms for time–optimal control of CNC machines along curved tool paths," *Robot. Comput.-Integr. Manuf.*, vol. 21, no. 1, pp. 37–53, 2005.

[22] M. Heng and K. Erkorkmaz, "Design of a NURBS interpolator with minimal feed fluctuation and continuous feed modulation capability," *Int. J. Mach. Tools Manuf.*, vol. 50, no. 3, pp. 281–293, 2010.

[23] K. Erkorkmaz, C.-H. Yeung, and Y. Altintas, "Virtual CNC system. Part II. High speed contouring application," *Int. J. Mach. Tools Manuf.*, vol. 46, no. 10, pp. 1124–1138, 2006.

[24] M. K. Jouaneh, D. A. Dornfeld, and M. Tomizuka, “Trajectory planning for coordinated motion of a robot and a positioning table. II. Optimal trajectory specification,” *IEEE Trans. Robot. Autom.*, vol. 6, no. 6, pp. 746–759, 1990.

[25] C. A. Ernesto and R. T. Farouki, “High-speed cornering by CNC machines under prescribed bounds on axis accelerations and toolpath contour error,” *Int. J. Adv. Manuf. Technol.*, vol. 58, no. 1–4, pp. 327–338, 2012.

[26] S. Tulsyan and Y. Altintas, “Local toolpath smoothing for five-axis machine tools,” *Int. J. Mach. Tools Manuf.*, vol. 96, pp. 15–26, 2015.

[27] M. Duan and C. Okwudire, “Minimum-time cornering for CNC machines using an optimal control method with NURBS parameterization,” *Int. J. Adv. Manuf. Technol.*, vol. 85, no. 5–8, pp. 1405–1418, 2016.

[28] V. Pateloup, E. Duc, and P. Ray, “Bspline approximation of circle arc and straight line for pocket machining,” *Comput.-Aided Des.*, vol. 42, pp. 817–827, 2010.

[29] L. B. Zhang, Y. P. You, J. He, and X. F. Yang, “The transition algorithm based on parametric spline curve for high-speed machining of continuous short line segments,” *Int. J. Adv. Manuf. Technol.*, vol. 52, no. 1–4, pp. 245–254, 2011.

[30] K. Erkorkmaz and Y. Altintas, “High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation,” *Int. J. Mach. Tools Manuf.*, vol. 41, no. 9, pp. 1323–1345, 2001.

[31] M.-T. Lin, M.-S. Tsai, and H.-T. Yau, “Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm,” *Int. J. Mach. Tools Manuf.*, vol. 47, no. 15, pp. 2246–2262, 2007.

[32] Y. Altintas and K. Erkorkmaz, “Feedrate optimization for spline interpolation in high speed machine tools,” *CIRP Ann.*, vol. 52, no. 1, pp. 297–302, 2003.

[33] M. Weck and G. Ye, “Sharp corner tracking using the IKF control strategy,” *CIRP Ann.*, vol. 39, no. 1, pp. 437–441, 1990.

[34] B. Sencer, K. Ishizaki, and E. Shamoto, “High speed cornering strategy with confined contour error and vibration suppression for CNC machine tools,” *CIRP Ann.*, vol. 64, no. 1, pp. 369–372, 2015.

[35] M.-S. Tsai and Y.-C. Huang, "A novel integrated dynamic acceleration/deceleration interpolation algorithm for a CNC controller," *Int. J. Adv. Manuf. Technol.*, vol. 87, no. 1–4, pp. 279–292, 2016.

[36] P.-J. Barre, R. Bearee, P. Borne, and E. Dumetz, "Influence of a jerk controlled movement law on the vibratory behaviour of high-dynamics systems," *J. Intell. Robot. Syst.*, vol. 42, no. 3, pp. 275–293, 2005.

[37] S.-S. Yeh and P.-L. Hsu, "Perfectly matched feedback control and its integrated design for multiaxis motion systems," *J. Dyn. Syst. Meas. Control-Trans. Asme*, vol. 126, no. 3, pp. 547–557, 2004.

[38] C. Brecher *et al.*, "NURBS based ultra-precision free-form machining," *CIRP Ann.*, vol. 55, no. 1, pp. 547–550, 2006.

[39] M.-C. Tsai, C.-W. Cheng, and M.-Y. Cheng, "A real-time NURBS surface interpolator for precision three-axis CNC machining," *Int. J. Mach. Tools Manuf.*, vol. 43, no. 12, pp. 1217–1227, 2003.

[40] Q. Bi, Y. Wang, L. Zhu, and H. Ding, "A practical continuous-curvature Bezier transition algorithm for high-speed machining of linear tool path," in *International Conference on Intelligent Robotics and Applications*, 2011, pp. 465–476.

[41] X. Pessoles, Y. Landon, and W. Rubio, "Kinematic modelling of a 3-axis NC machine tool in linear and circular interpolation," *Int. J. Adv. Manuf. Technol.*, vol. 47, no. 5–8, pp. 639–655, 2010.

[42] S. Tajima and B. Sencer, "Kinematic corner smoothing for high speed machine tools," *Int. J. Mach. Tools Manuf.*, vol. 108, pp. 27–43, 2016.

[43] L. Biagiotti and C. Melchiorri, "FIR filters for online trajectory planning with time- and frequency-domain specifications," *Control Eng. Pract.*, vol. 20, no. 12, pp. 1385–1399, 2012.

[44] Q. Zhang, X.-S. Gao, H.-B. Li, and M.-Y. Zhao, "Minimum time corner transition algorithm with confined feedrate and axial acceleration for nc machining along linear tool path," *Int. J. Adv. Manuf. Technol.*, vol. 89, no. 1–4, pp. 941–956, 2017.

[45] T. Coleman, M. A. Branch, and A. Grace, "Optimization toolbox," *Use MATLAB User's Guide MATLAB 5 Version 2 Relaese II*, 1999.

[46] Y. Altintas, *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design*. Cambridge university press, 2012.

[47] K. Erkorkmaz, A. Alzaydi, A. Elfizy, and S. Engin, "Time-optimal trajectory generation for 5-axis on-the-fly laser drilling," *CIRP Ann.*, vol. 60, no. 1, pp. 411–414, 2011.

[48] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, 1985.

[49] P. Lambrechts, M. Boerlage, and M. Steinbuch, "Trajectory planning and feedforward design for electromechanical motion systems," *Control Eng. Pract.*, vol. 13, no. 2, pp. 145–157, 2005.

[50] Z. Rymansaib, P. Iravani, and M. N. Sahinkaya, "Exponential trajectory generation for point to point motions," in *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2013, pp. 906–911.

[51] A. Visioli, "Trajectory planning of robot manipulators by using algebraic and trigonometric splines," *Robotica*, vol. 18, no. 6, pp. 611–631, 2000.

[52] W. E. Singhose, W. P. Searing, and N. C. Singer, "Improving repeatability of coordinate measuring machines with shaped command signals," *Precis. Eng.*, vol. 18, no. 2–3, pp. 138–146, 1996.

[53] N. C. Singer and W. P. Seering, "Preshaping command inputs to reduce system vibration," *J. Dyn. Syst. Meas. Control*, vol. 112, no. 1, pp. 76–82, 1990.

[54] Y. Altintas and M. R. Khoshdarregi, "Contour error control of CNC machine tools with vibration avoidance," *CIRP Ann.*, vol. 61, no. 1, pp. 335–338, 2012.

[55] J. W. Jeon and Y.-K. Kim, "FPGA based acceleration and deceleration circuit for industrial robots and CNC machine tools," *Mechatronics*, vol. 12, no. 4, pp. 635–642, 2002.

[56] S. Jones, R. Goodall, and M. Gooch, "Targeted processor architectures for high-performance controller implementation," *Control Eng. Pract.*, vol. 6, no. 7, pp. 867–878, 1998.

[57] R. A. Osornio-Rios, R. de Jesus Romero-Troncoso, G. Herrera-Ruiz, and R. Castañeda-Miranda, "Computationally efficient parametric analysis of discrete-time

polynomial based acceleration–deceleration profile generation for industrial robotics and CNC machinery," *Mechatronics*, vol. 17, no. 9, pp. 511–523, 2007.

[58] D.-I. Kim, J. W. Jeon, and S. Kim, "Software acceleration/deceleration methods for industrial robots and CNC machine tools," *Mechatronics*, vol. 4, no. 1, pp. 37–53, 1994.

[59] M. Bonfe and C. Secchi, "Online smooth trajectory planning for mobile robots by means of nonlinear filters," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4299–4304.

[60] R. Zanasi, C. G. L. Bianco, and A. Tonielli, "Nonlinear filters for the generation of smooth trajectories," *Automatica*, vol. 36, no. 3, pp. 439–448, 2000.

[61] R. Zanasi and R. Morselli, "Discrete minimum time tracking problem for a chain of three integrators with bounded input," *Automatica*, vol. 39, no. 9, pp. 1643–1649, 2003.

[62] C.-S. Chen and A.-C. Lee, "Design of acceleration/deceleration profiles in motion control based on digital FIR filters," *Int. J. Mach. Tools Manuf.*, vol. 38, no. 7, pp. 799–825, 1998.

[63] K. Ogata and Y. Yang, *Modern control engineering*, vol. 4. Prentice-Hall, 2002.

[64] N. Wiener, *The Fourier integral and certain of its applications*. CUP Archive, 1988.

[65] S. Goto, M. Nakamura, and N. Kyura, "Trajectory generation of industrial mechatronic systems to achieve accurate contour control performance under torque saturation," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 1995, vol. 3, pp. 2401–2406.

[66] T. Moriwaki, "Multi-functional machine tool," *CIRP Ann.*, vol. 57, no. 2, pp. 736–749, 2008.

[67] C. Lartigue, E. Duc, and A. Affouard, "Tool path deformation in 5-axis flank milling using envelope surface," *Comput.-Aided Des.*, vol. 35, no. 4, pp. 375–382, 2003.

[68] A. Yuen, K. Zhang, and Y. Altintas, "Smooth trajectory generation for five-axis machine tools," *Int. J. Mach. Tools Manuf.*, vol. 71, pp. 11–19, 2013.

[69] J. Yang and A. Yuen, "An analytical local corner smoothing algorithm for five-axis CNC machining," *Int. J. Mach. Tools Manuf.*, vol. 123, pp. 22–35, 2017.

[70] M.-C. Ho, Y.-R. Hwang, and C.-H. Hu, “Five-axis tool orientation smoothing using quaternion interpolation algorithm,” *Int. J. Mach. Tools Manuf.*, vol. 43, no. 12, pp. 1259–1267, 2003.

[71] Y. Wang, X. Ma, L. Chen, and Z. Han, “Realization methodology of a 5-axis spline interpolator in an open CNC system,” *Chin. J. Aeronaut.*, vol. 20, no. 4, pp. 362–369, 2007.

[72] Y. Sun, Y. Zhao, Y. Bao, and D. Guo, “A smooth curve evolution approach to the feedrate planning on five-axis toolpath with geometric and kinematic constraints,” *Int. J. Mach. Tools Manuf.*, vol. 97, pp. 86–97, 2015.

[73] J. Shi, Q. Bi, L. Zhu, and Y. Wang, “Corner rounding of linear five-axis tool path by dual PH curves blending,” *Int. J. Mach. Tools Manuf.*, vol. 88, pp. 223–236, 2015.

[74] M.-T. Lin, J.-C. Lee, C.-C. Shen, C.-Y. Lee, and J.-T. Wang, “Local Corner Smoothing with Kinematic and Real-time Constraints for Five-axis Linear Tool Path,” in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018, pp. 816–821.

[75] S. Tajima, B. Sencer, and E. Shamoto, “Accurate interpolation of machining tool-paths based on FIR filtering,” *Precis. Eng.*, vol. 52, pp. 332–344, 2018.

[76] E. Budak, E. Ozturk, and L. T. Tunc, “Modeling and simulation of 5-axis milling processes,” *CIRP Ann.*, vol. 58, no. 1, pp. 347–350, 2009.

[77] B. C. Kuo, *Automatic control systems*. Prentice Hall PTR, 1987.

[78] G. Van Brummelen, *Heavenly mathematics: The forgotten art of spherical trigonometry*. Princeton University Press, 2012.

[79] B. Sencer, Y. Altintas, and E. Croft, “Modeling and control of contouring errors for five-axis machine tools—part I: modeling,” *J. Manuf. Sci. Eng.*, vol. 131, no. 3, p. 031006, 2009.