Smooth and Time-Optimal Trajectory Planning for Multi-Axis Machine Tools

by

Katharine Nancy DiCola

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	Anna Valente Professor, Department of Innovative Technologies University of Applied Sciences and Arts of Southern Switzerland
Supervisor(s)	Kaan Erkorkmaz Professor, Mechanical and Mechatronics Engineering University of Waterloo
Internal Member	Cliff Butcher Associate Professor, Mechanical and Mechatronics Engineering University of Waterloo
Internal Member	William Melek Professor, Mechanical and Mechatronics Engineering University of Waterloo
Internal-external Member	Christopher Nielsen Professor, Electrical and Computer Engineering University of Waterloo

Author's Declaration

Statement of Contribution

An early version of the linear programming (LP) + nonlinear programming (NLP) feedrate optimization methodology, presented in Chapter 3 of this thesis, was published in the following conference paper:

- Chen, C.Q.G., DiCola K., K. Erkorkmaz, and X. Beudaert, "Two-Stage Feedrate Optimization for Freeform Toolpath Contouring", Proceedings of the 7th International Conference on Virtual Machining Process Technology (VMPT), Hamilton, Canada, 2018.

The feed optimization contribution in this thesis is a continuation of the research that was completed by an earlier graduate student, Ms. Christina Chen. Chen had implemented linear programming (LP) based feed optimization. At the time of writing this conference paper, Chen was completing her MASc thesis and the author (DiCola) was starting her PhD. The first implementation of LP + NLP was developed collaboratively between Chen and DiCola, who jointly led the writing of the manuscript. Due to her experience and familiarity with the existing source code, Chen had taken the lead in developing the first software prototype. UW team also received assistance from the industry collaborator Dr. Xavier Beudaert (Ideko, Spain), in developing, improving, and benchmarking various feed optimization ideas. The valuable contributions made by all co-authors, especially Ms. Christina Chen, are gratefully acknowledged by the author.

Following the early and successful demonstration of the LP+NLP concept in the above paper, the author (DiCola) independently realized all of the proceeding formulation, development, integration, programming iterations, and testing of the algorithm in simulations and experiments. She developed and analyzed the conditions for the alignment of the nested long horizon (LP) and short horizon (NLP) windows, to ensure that boundary conditions are passed on correctly between the two different subproblems. She also analyzed the constraint compatibility conditions for switching between different physical constraints during the LP and NLP steps, such that a feasible solution will always theoretically be guaranteed. Using the results of these analyses, the author re-wrote nearly all of the algorithms in code and conducted extensive trouble shooting, with long and challenging toolpaths, which helped catch and correct numerical issues that could cause algorithm failure. The author also analyzed the optimum configuration parameters (e.g., meshing densities, window sizes, overlaps, etc.) for the algorithm to work efficiently and effectively. All of these contributions, detailed in Chapters 3 and 4 of this thesis, are essential for the proposed LP+NLP algorithm to function in a robust and stable manner. Additionally, the toolpath smoothing methods described in Chapter 5, which are not featured in the above conference paper, were developed independently by the author.

Abstract

This thesis presents novel methods for feedrate optimization and toolpath smoothing in CNC machining. Descriptions of the algorithms, simulation test cases, and experimental results are presented.

Both feedrate optimization and toolpath smoothing are essential for increasing manufacturing efficiency while retaining part quality in CNC machining. The application of high-speed machining also necessitates the use of high feedrates, and smooth toolpaths which can be safely traversed at high feeds. However, problems occur when the feedrate is increased without check. High tracking error in machining may cause part tolerance errors. Transient vibrations due to jerky movement can lead to poor part surface quality. High speed trajectories may also demand greater torque than what the feed drives are capable of producing, which affects the motion controller's ability to follow the trajectory correctly. The condition of the machine is also a concern, with the potential for damage or excessive wear on the machine's components, if excessive axis velocity or jerk (i.e., rate of change of acceleration) is commanded.

The feedrate scheduling algorithm developed in this thesis combines linear and nonlinear programming in a dual-windowed implementation. Linear programming (which is computationally fast) is used to quickly provide a near-optimal guess, based on axis velocity, acceleration, and jerk constraints. The solution is then refined through the use of nonlinear optimization. In the latter step, requiring more computations, the commanded motor torque and expected servo error are constrained directly, leading to shorter movement time. A windowing alignment procedure is presented which allows for these two optimization methods, each with different problem constraints and solutions horizons, to work in tandem with one another without risking infeasible boundary conditions between the windows. The algorithm is validated in simulation and experiment studies. Case studies analyzing the parameters of the optimization algorithm are also presented, and the configuration which is most computationally efficient is determined.

A toolpath generation method is presented in which Euler-spiral pairs are used to smooth sharp corners, with an algorithm that integrates directly with the developed feedrate optimization The result is an exactly arc-length parametrized, G2-continuous toolpath whose axis derivatives can be computed very efficiently, which helps reduce the overall computation time.

A repositioning toolpath method is also developed to reduce the cycle time of multi-layer contouring operations. This method replaces circular arc based repositioning segments between contouring passes (commonly used in industry) with a smooth Euler spiral based curve. This avoids tangent and curvature discontinuities, allowing for smoother motion with lower velocity and acceleration demands, while also reducing the overall motion. The repositioning toolpath has also been integrated with feedrate optimization and validated in simulation results.

Acknowledgements

First and foremost, I would like to give my greatest thanks to my supervisor Prof. Kaan Erkorkmaz, for all of the time and effort put into helping me expand my knowledge, develop and implement my research, and write my thesis. He has truly gone above and beyond in his duties as a supervisor, and I am very grateful for his continued support during my PhD.

I would like to thank Dr. Serafettin Engin, Mr. Donald McIntosh, Dr. Jochem Roukema and numerous other colleagues from Pratt & Whitney Canada for their assistance and feedback during my research, and the opportunity to have a fruitful internship at the company. I would also like to thank Dr. Xavier Beudaert (Ideko) for providing ideas, feedback, and benchmarking opportunities during the start of my studies, which helped shape the feed optimization methodology.

I would like to thank my thesis committee members: Professor Anna Valente, Professor Kaan Erkorkmaz, Professor Cliff Butcher, Professor William Melek, and Professor Christopher Nielsen, for taking the time to read my thesis and contribute with constructive feedback.

Sincere thanks to all of the members of Precision Controls Group, past and present, who have assisted me. I would like to thank Christina Chen in particular, who began the research that I have continued in this thesis. I would also like to thank Chia-Pei Wang, for sharing his extensive knowledge of machine tools with me, and Andrew Katz, for helping with experiments, writing, and organization, among many other things.

Finally, I would like to thank my parents, for their constant support. It is thanks to them that I have been able to continue my education for all of these years.

This research was sponsored by Natural Sciences and Engineering Research Council of Canada (NSERC), Pratt & Whitney Canada (P&WC), and Manufacturing Automation Laboratories (MAL) Inc.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the modern manufacturing environment, a significant amount of research and development has been, and continues to be, dedicated to one of the most fundamental goals: increasing the overall throughput of manufacturing. Manufacturing processes often include repetitive motions, with the same toolpath being executed many times. These processes can benefit greatly from trajectory optimization, as any decrease in the motion time will result in an increase in the overall productivity. An additional motivation for fast movement is the proliferation of high-speed machining (HSM). Depending on the materials and surface quality demands for a part, the spindle speed of the cutting tool may be high to take advantage of potential productivity gains. This is especially seen in the machining of aluminum alloys. This, in turn, will require high velocities from the feed drives as they move the cutting tool through the workpiece.

The process of going from a CAD (computer-aided design) model of a part to a finished product presents many opportunities for time saving modifications. Once the CAD model is created, CAM (computer-aided manufacturing) software is used to generate the trajectory a machine must execute to create that part. The computer numerically controlled (CNC) machine's controller converts this trajectory into command signals, which are applied to the feed drive control loops of the machine. The motors then execute the given commands to move the cutting tool along the part.

In this thesis, time saving options are explored for planning the trajectory of the machine, which is defined by the *toolpath* (geometry) and the *feedrate* (tangential velocity profile of the tool as it moves along the toolpath). While reducing the motion time, it is important to retain the final dynamic tool positioning accuracy, and to ensure that the trajectory is within the physical actuation capabilities of the servomotors.

The 'toolpath' in the context of CNC machining is the geometric path which the cutting tool must follow to produce the final part shape. The toolpath is typically generated using CAM software, which analyzes the geometry of the target CAD model, as well as the material blank from which it will be cut, and determines how the tool must travel to produce the target part. Reductions in machining time can be obtained by modifying this toolpath, provided the resulting geometry remains within the specified tolerances. One such modification is to smoothen the toolpath and eliminate sharp corners when allowable. This allows for faster traversal of the path. Potential inaccuracies may also occur during the execution of the trajectory by the machine's feed drives, which when combined with the toolpath modification, may violate the part tolerances. These factors must all be considered if changes to the toolpath are to be made.

The toolpath will also likely contain repositioning segments – movements made by the tool in which no cutting is performed but rather the tool is moved from the end of one cut to the beginning of the next. The toolpath may be modified more liberally in these sections, provided the modification does not cause a collision, and thus they can be replaced with a new shorter or smoother path for the machine to execute more quickly.

Another potential and important modification is to the feedrate (also referred to as ‘feed’). To reduce the time taken to machine a part, it is desirable to increase the feedrate as much as possible without damaging the machine or the part. While many machining operations require a constant feedrate during cutting (such as when removing large amounts of material at once), some operations allow for a variable feedrate (finishing operations, and operations in which the forces on the cutting tool are lower in general). These operations may benefit from feedrate optimization, which is the scheduling of the feed to allow higher speed progression when the part geometry, the cutting process, and especially the dynamic response of the machine tool allows.

The feedrate optimization should solve the variable feedrate throughout the cutting operation to minimize the machining time while maintaining the dimensional accuracy of the resulting part and avoiding any overloading or damage to the machine. The nature of the problem depends on the selection of the optimization constraints, and the computational method selected to optimize the feedrate while enforcing these constraints. For a constraint to be used in an optimization routine, it must accurately reflect the dynamic response which affects the part quality or machine operation, and must be predictable given trajectory candidate. The numerical optimization routine must be selected which can accurately and efficiently solve the problem posed by the constraints.

In this thesis, the following novel contributions are made:

- A new feedrate scheduling algorithm is developed and proposed, in which two optimization algorithms (linear and nonlinear programming) are combined into a forward-looking windowed solution. The advantages of each type of optimization are combined to reduce the cycle time of a machining operation, without requiring excessive computation time. The final generated trajectories have the servo tracking error and motor torque demand constrained directly, in order to ensure the part accuracy and to prevent overloading and potential damage to the machine. Simulation and experimental test cases are presented.
- A novel toolpath smoothing method is developed using Euler spiral pairs which integrates seamlessly with the proposed feed optimization algorithm. The Euler spiral pairs yield a G2-continuous (i.e., twice differentiable), exactly arc-length parametrized toolpath. To significantly reduce the computational load, the algorithm can compute the axis geometric derivatives, needed by the feed optimization, without having to numerically integrate the coordinates for the constraint evaluation points.

- An Euler-spiral based layer transition toolpath has been developed and integrated with the proposed feedrate optimization. This algorithm applies specifically to multi-layer contouring operations with repositioning between the layers. The proposed Euler spiral-based G2-continuous toolpath can be traversed without substantial slowdowns, in comparison to using in-layer circular arc-based transitions, which is the common industrial practice.

Following this introduction (Chapter 1), the literature review is presented in Chapter 2. Chapter 3 describes the dual windowed LP+NLP feedrate optimization method and provides simulation and experimental implementation results. Chapter 4 provides details on numerical considerations when configuring and implementing the LP+NLP algorithm. Chapter 5 describes the developed toolpath corner smoothing and layer transition techniques, and includes simulation and experimental results. The thesis conclusions and future work recommendations are presented in Chapter 6.

2 Literature Review

2.1 Introduction

The first CNC machine tools were developed in the 1950s, with machines that could reproduce motions recorded on punch card programming systems and with rudimentary feedback control implementation [1]. Due to the limited computational technology of the time, generation of advanced trajectories using a computer system was not possible.

Modern machining applications demand significantly greater speed and precision. These operations require much higher spindle speeds and feedrates, which in turn necessitates advanced strategies for numerical control. Industries such as aviation, biomedical, and automotive part manufacturing require a machine tool which can manufacture freeform toolpaths within very tight tolerances of dimensional accuracy and surface quality. High-speed high-accuracy machine tools require trajectory generation methods that can accurately command smooth and high-speed movements within the capabilities of the machine tool. A significant amount of research has been performed, in the context of both robotics and machine tools, to generate algorithms which can produce such trajectories.

Minimum-time motion problems for industrial motion control have been studied for over 50 years. Early research focussed on robotic arms. In 1969, Kahn and Roth studied time-optimal control of a robotic manipulator [2]. This process optimized the traversal between two points (i.e., point-to-point motion), without considering the specific path taken by the machine between points. In 1985, Bobrow et al. studied the problem of a robotic manipulator moving along a designated path in the minimum possible time, without exceeding the torque limitation of the motors [3]. In modern manufacturing, there are many options for time-optimal trajectory planning. In this research, the specific case of CNC machines is studied.

2.2 Smooth toolpath planning

For the machining time to be optimal, the toolpath must be smooth. Discontinuities in the toolpath, such as sharp corners or sudden changes in curvature, would require the machine to slow down significantly or stop completely. If a machine tool were to continue through a sharp corner at a constant speed, this would produce a discontinuous velocity profile, and thus, an unbounded acceleration profile. These sudden high accelerations would exceed the capacities of the feed drive motors and would likely produce huge vibrations, which could damage the machine and have significant negative effect on the quality of the resulting part.

Thus, the toolpath must be modified to eliminate these corners, while still following the original toolpath within a specified tolerance.

CNC machine tools are capable to traversing between commanded positions using different path descriptions, which are covered by different *interpolation methods.* For example, linear interpolation is the simplest movement type – in which all intermediary points between the starting and destination point form a straight line. This mode is used for machining straight edges. While linear interpolation allows for easy calculation, it also creates corners at which toolpath geometric derivatives are not continuous (Figure 2-1a). If such a toolpath is traversed without commanding low velocities at the corner, it will create large fluctuations in the acceleration and jerk during machining, which will induce large servo errors and structural vibrations, that are detrimental to the machine part quality and accuracy.

Circular interpolation allows for the toolpath to be defined as circular arcs – by specifying the start and end points and center of each arc. With this interpolation, discontinuities in direction (i.e. sharp corners) can be eliminated. However, discontinuities in curvature will remain, as the curvature profile will be piecewise constant. Thus, if such toolpaths are traversed without coming to full stops at the connection points, sudden changes in the curvature will result in step changes in the axis acceleration profiles, which are again detrimental.

Advanced Computer Numerical Control (CNC) systems can apply various other and more complex forms of interpolation, in particular spline interpolation, which can eliminate the discontinuity problem encountered when linear and circular interpolation, is used to replicate toolpaths for freeform part machining.

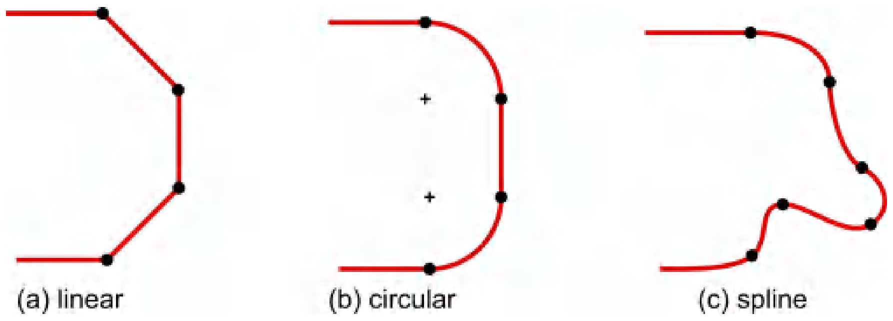


Figure 2-1: Types of interpolation.

2.2.1 Spline representation of toolpaths

A method for generating smooth toolpaths, which enable high traverse rates at increased motion control accuracy, is spline interpolation. Splines are piecewise functions constructed of several segments, each with its own function definition. The boundaries between segments are referred to as “knots”. The most common type of spline is a polynomial spline, in which each segment is defined by a single polynomial equation. These equations are selected such that at each knot, there is continuity in position, direction (i.e., tangent), and in many cases higher order derivatives. The most common type of spline, a cubic polynomial spline, can achieve the continuity of position, and its first and second derivatives (but is discontinuous in higher derivatives). Figure 2-2 shows an example of a toolpath represented with a cubic spline.

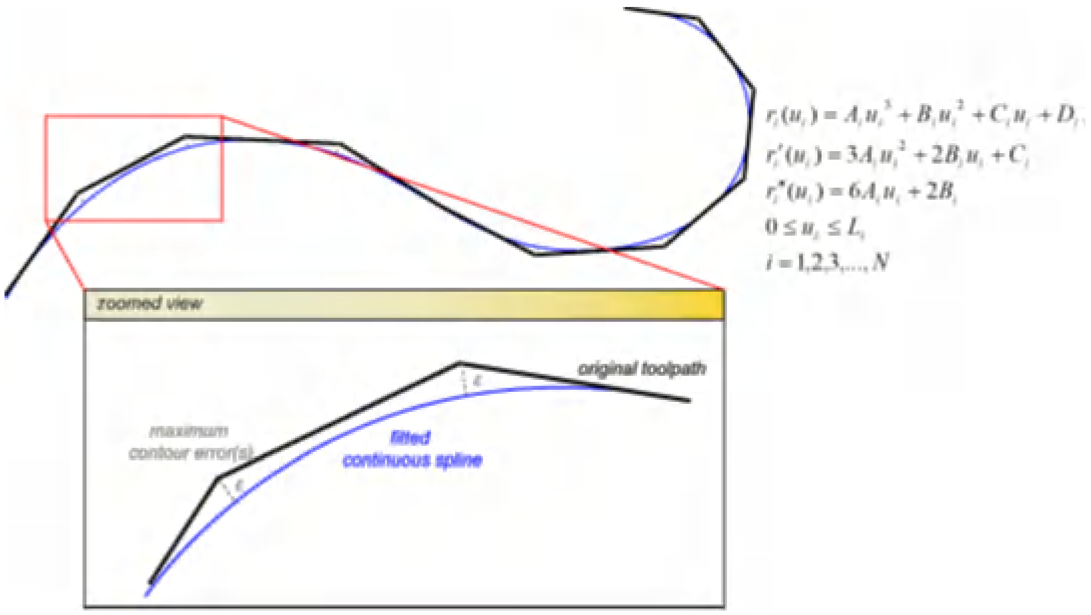


Figure 2-2: Cubic spline toolpath. [4]

Cubic spline interpolation is a well-studied interpolation method which has been in use for many years. In 1985, Huan used cubic splines for a 2nd- derivative continuous toolpath [5]. This eliminates the corners seen when using simple linear interpolation. Cubic splines are continuous in (i.e., differentiable up to) the second derivative, and bounded in the third, implying that they can also be utilized when a jerk-limited trajectory is required. As can be seen from Figure 2-3, with increasing degree of continuity of the motion profile, the frequency content of acceleration can be clustered further into the low frequency range, thereby enabling even smoother motion with potentially less residual vibrations or servo errors (at the expense of requiring increased peak velocity and acceleration magnitudes). Hence, higher-order splines can also be used to obtain an increased degree of continuity [6], and enable smooth motion overall [7], [8]. For example, Erkorkmaz et al. used quintic splines to represent the toolpath [9] and later Yang and Yuen created corner smoothing strategies using quintic splines [6].

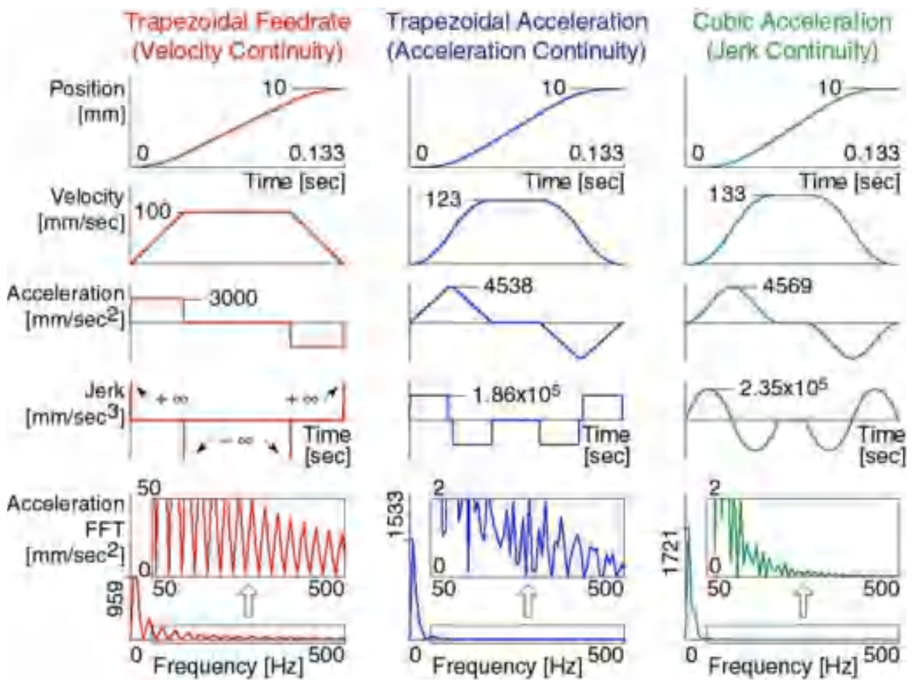


Figure 2-3: Effect of derivative continuity.

As a different and more versatile kind of spline parameterization, basis function splines (B-splines) can be used to represent either the toolpath or the feed profile of a trajectory. An individual B-spline is made of polynomial pieces. Each B-spline is multiplied (or scaled) by a distinct value (the “control point” of that B-spline). A set of B-splines can then be combined by summation into a “B-spline function”. Figure 2-4 shows an example of B-splines (each in a different colour) and the resulting B-spline function (in black). Note that each B-spline occupies only a short portion of the profile. This spline formulation allows for local adjustment without affecting the rest of the function, in that changing any control point will affect only the portion of the function occupied by its corresponding B-spline. This property is essential for achieving numerical efficiency when using a windowed solution (Section 2.3.3). Furthermore, when B-splines are used to parameterize the geometry of a toolpath, this property ensures that the instantaneous commanded axis trajectory always remains within the convex hull of the three (or more, depending on the B-spline order) consecutive control points used to define the local geometry.

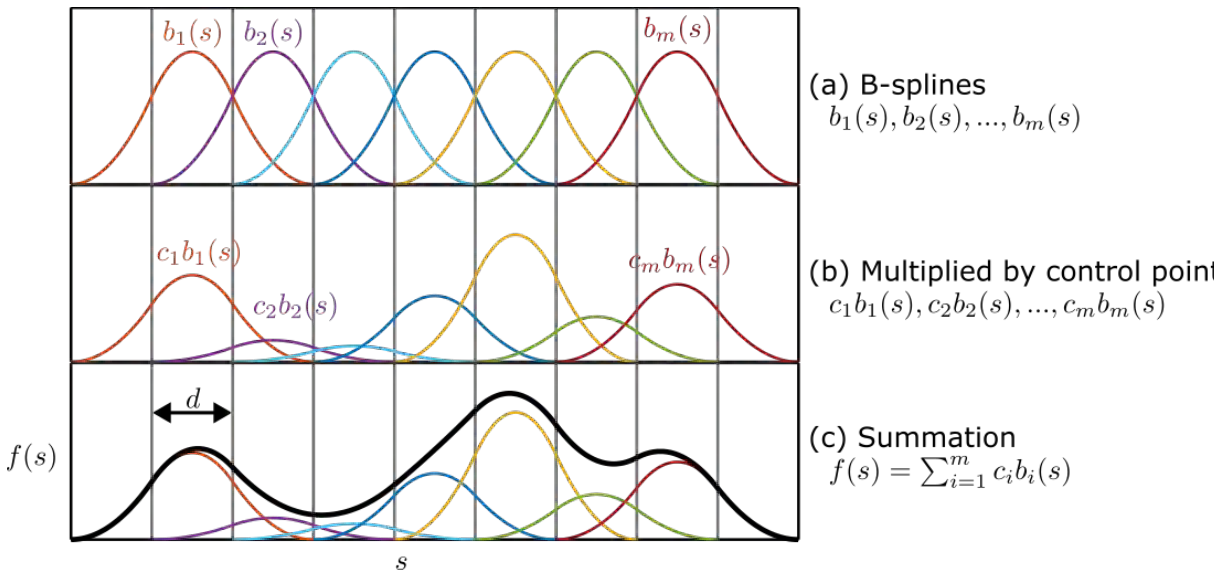


Figure 2-4: Basis function spline, order 3.

Figure 2-5 compares a cubic B-spline function (made of cubic B-splines) and a polynomial spline. A polynomial spline (Figure 2-5 (b)) will pass through each input point exactly, but it will pass outside of the shape created by these points, and depending on the geometry and spacing of the points, the violation may be large. A B-spline function (Figure 2-5 (a)) does not pass through each point directly, but is guaranteed to stay within the convex hull.

Some CNC controllers are able to implement B-spline interpolation natively, in which case, the use of B-spline toolpaths presents an advantage over other types of curves which would need to be computed externally and input as discrete points.

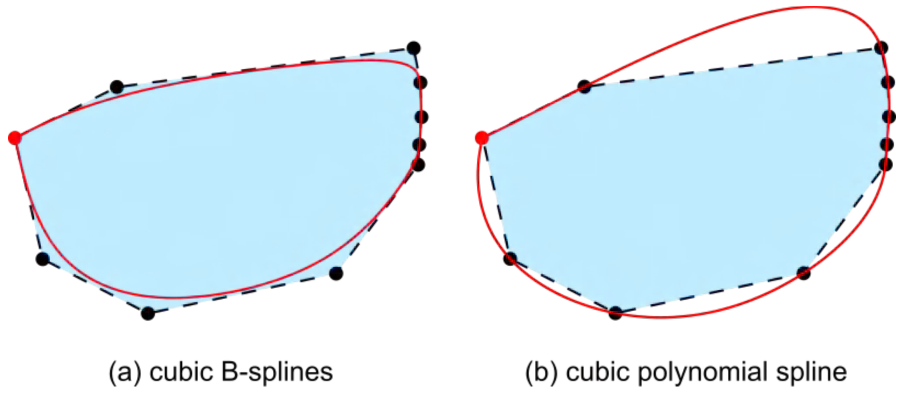


Figure 2-5: Comparison between a toolpath using cubic B-splines and a toolpath using a cubic polynomial spline.

The toolpath may be represented by B-splines of different order. Zhao et al. used cubic B-splines to smooth toolpaths with short linear segments in real-time with curvature continuity [10]. Sencer et al. used quintic B-splines to optimize curvature for toolpaths composed of straight lines [11]. Lartigue et al. used sets of cubic B-splines to machine a complex freeform surface [12].

B-splines may also be used to represent the feed profile, such as in [13]. Okwudire et al. used filtered B-splines to reduce unwanted vibrations during machining [14]. The linear programming (LP) method discussed in Section 3.4 uses B-splines to represent the squared feed profile [15], [16], [17].

NURBS (non-uniform rational B-splines), a modified form of B-splines, include weighting factors for each spline segment, giving more precise control of the resulting profile. Liu et al. used NURBS toolpaths to reduce feed fluctuations in constant-feed machining [18]. Heng and Erkorkmaz developed a NURBS interpolator with the capability to modulate feedrate profile continuously across different toolpath segments [19]. Duan and Okwudire generated an optimal cornering curve and then represented it using NURBS [20]. Wang et al. used a bisection method to interpolate projections of NURBS curves and surfaces [21].

Toolpath smoothing may also be performed by filtering commands sent to the machine's servo controllers. Tajima et al. used FIR filtering on velocity commands [22], which was also extended to the five-axis case [23]. Sencer et al. also used FIR filtering to generate smoothed toolpaths with limited contour error [11]. FIR filtering, in which sudden changes in commanded velocity are smoothed, is used as an option in many machine tool controllers for generating smooth trajectories.

2.2.2 Parametrization

The use of splines to represent smooth toolpaths can be complicated by the preference for arc-length parametrization when designing a trajectory. Cubic spline toolpaths are second derivative continuous, but are parametrized based on the chord length. Figure 2-6 illustrates the difference between incremental arc displacement (ds) and corresponding incremental chord displacement (du). As can be seen, the chord length is not a good approximation of the arc length, and this can cause undesirable fluctuations in the feed, when the feed planning is done based on the assumption that they are equal. Furthermore, between the connections of spline segments, this also results in feed and acceleration discontinuity.

One solution to this problem is to deliberately design the splines such that they are nearly arc-length parametrized. In this context, higher order splines can be used for greater continuity and accuracy in parametrization. In 1993, Wang and Yang used quintic splines to create a toolpath which is close to arc-length parametrized [8]. Another solution may be to quantify the difference between the arc length and the spline parameter and compensate for it, such as in the work of Erkorkmaz and Altintas who used a feed correction polynomial to represent the relationship between the arc length and the spline parameter [7], [24]. A two-step prediction and adjustment method was developed to predict the relationship between path parameter and displacement by Chen and Sun [25].

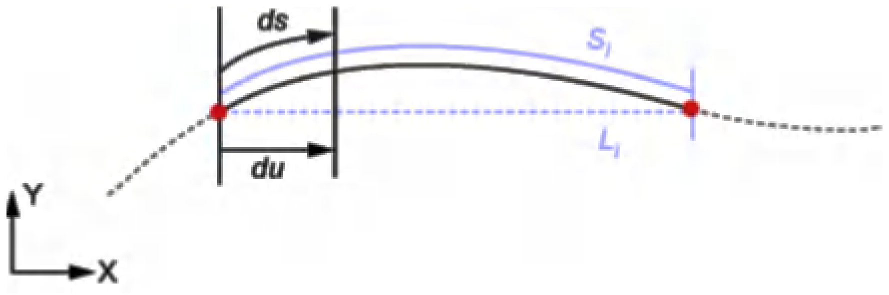


Figure 2-6: Arc displacement compared with chord length.[4]

2.2.3 Euler spirals

Euler spirals, also known as “clothoids” or “Cornu spirals”, can be used to create a high-order differentiable curve [26]. Euler spirals are parameterized such that the relationship between the arc displacement and the curvature is linear by definition, and thus are not subject to minute feed fluctuations caused by variations in the relationship between the arc length and spline parameter. “Euler splines” or “clothoid splines” are defined as piecewise combinations of these curves which produce a single resulting profile. Figure 2-7 illustrates the fundamental geometric shape of an Euler spiral, along with the gradual linear change in curvature.

A method of creating clothoid splines to interpolate between points was proposed in 1989 by Stoer [27]. A method of using clothoid pairs (two segments connected) was proposed in [28] for use in robot path-planning. Jouaneh et al. [29] used clothoid segments to generate simple trajectories for 2-D machining. This method uses clothoid pairs combined with straight-line segments for combined tool and table movement. Walton and Meek [30], [31] developed a method of matching clothoid curves to input polylines (series’ of straight line segments), using symmetrical and asymmetrical clothoid pairs, used in the design of roads, railways and vehicle paths.

Shahzadeh et al. developed a method of CNC machine corner smoothing using clothoids [32]. In this implementation, multiple clothoid segments are generated per corner and a maximum feed is computed for constant-feed machining of the resulting toolpath. Further, Shahzadeh et al. used clothoid pairs to smoothen sharp corners in machining [33]. This method uses iteration to create the clothoid that connect to straight lines and arcs, by matching the tangent angles and curvatures at the boundary conditions.

A modified 3D generalized clothoid was developed by Xiao et al. [34]. The generated curve is not a clothoid by the traditional definition, but still possesses the property of arc-length parameterization. Furthermore, the curve is G3-continuous, whereas the standard clothoid spline (i.e., made up by connecting multiple segments) satisfies only G2.

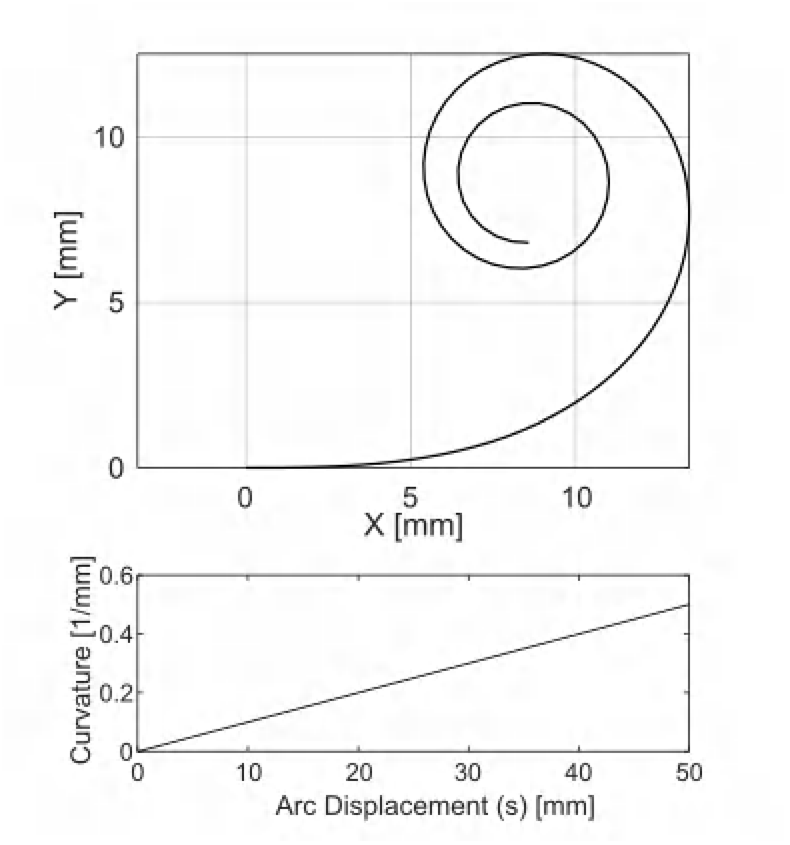


Figure 2-7: Basic shape of an Euler spiral.

Euler spirals have been used extensively in designs of roads and railroads. Eliminating sudden changes in the route’s curvature allows for smooth travel along these paths. Figure 2-8 demonstrates their use in road design. The left-hand side of Figure 2-8 shows a straight road connected directly to a curved road. If a vehicle were to travel along this route at a constant speed, it would experience a sudden change in acceleration due to the sudden change in road curvature. The right-hand side of Figure 2-8 shows a road generated with an Euler spiral. In this case, the acceleration of the vehicle travelling along it would change gradually due to the linear change in curvature.

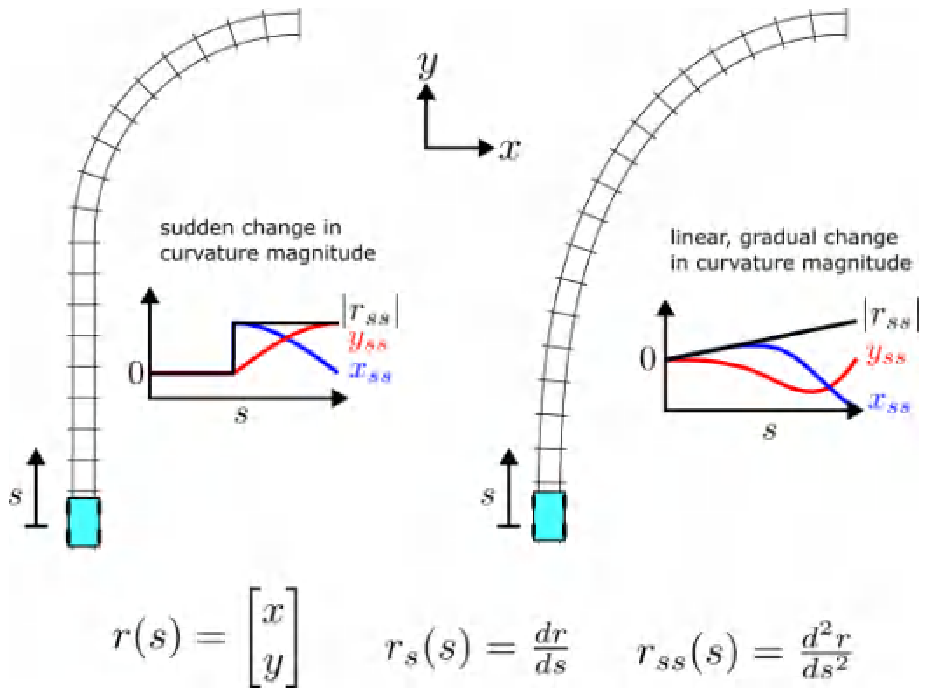

y
x
sudden change in curvature magnitude
linear, gradual change in curvature magnitude
$|r_{ss}|$
y_{ss}
x_{ss}
0
s
$r(s) = \begin{bmatrix} x \\ y \end{bmatrix}$
$r_s(s) = \frac{dr}{ds}$
$r_{ss}(s) = \frac{d^2 r}{ds^2}$

Figure 2-8: Example of Euler spirals used in road design.

As shown by Eqs. (3-18) - (3-20) (in the proceeding chapter), the axis kinematic profiles are computed as products of axis geometric derivatives, the feedrate, and the derivatives of the feedrate. By ensuring that each of these is differentiable, discontinuities will also be avoided in the final axis kinematic profiles.
The second geometric derivative (x_{ss}, y_{ss}) and curvature (r_{ss}) profiles in Figure 2-8 demonstrate the advantages of this shape. If a machine tool's servo axes are travelling along a path at a constant feedrate, the axial velocity is proportional to the tangential velocity and the first geometric derivative r_s. The axis level acceleration is proportional both to r_s multiplied by the tangential component of acceleration ($\ddot{s}$), as well as the second geometric derivative, r_{ss} multiplied by the square of the tangential velocity ($\dot{s}^2$). The linear, continuous shape of the curvature profile prevents any discontinuities in the second derivatives, x_{ss} and y_{ss}. This helps to preserve axis level acceleration continuity (provided that $\ddot{s}$ is also continuous), and thus the axis level jerk can be kept bounded within given limits.

In this thesis, as discussed in Chapter 4, the clothoid has been developed and integrated into the toolpath planning, to work seamlessly with the proposed feedrate optimization strategy. The framework for a 2D clothoid has been studied and extended into 3-axis, by utilizing tilted planes. Then, the clothoid's property of analytical calculability of the geometric derivatives has been integrated and utilized within the feed motion optimization method, also developed in Chapter 3 of this thesis. Afterwards, the use of clothoids has been explored for time-optimal corner rounding within controlled geometric tolerances, as well as producing time-optimal tool repositioning paths for multi-layered contour machining applications.

2.3 Feed optimization

Once a smooth toolpath is planned, the profile of feedrate (i.e., tangential velocity) along this toolpath must be carefully planned and executed. Traditionally, parts can be machined at a constant feedrate when the limiting cutting forces and chip load is of concern, especially in rough machining applications. However, when performing finish machining or when tight regulation of cutting forces is not essential, allowing for the feedrate to be modulated can make machining more efficient by reducing the overall motion time, achieved by travelling faster along the low curvature sections of a toolpath while slowing down during complex and high curvature portions. In the computation of a "minimum-time" trajectory, factors such as the part's targeted accuracy and surface quality, the machine's dynamic and kinematic capabilities, and the computational time of trajectory optimization, must all be considered and balanced.

Constant feed machining is the simplest choice in which the feedrate remains invariant throughout a cutting pass. This approach was used in earlier trajectory generation algorithms, such as by Fleisig and Spence [35]. Erkorkmaz and Altintas generated a jerk-limited profile in which the feed remains constant between the acceleration and deceleration transients [36].

On the other hand, optimized feedrate either attempts the minimize the total travel time, or maximize the integral (or integral square) of the feedrate as a function of travel distance. The feedrate optimization problem is inherently nonlinear and solved subject to a variety of constraints, such as the limitation of actuator level velocity, acceleration, jerk, and torque profiles, etc., as discussed in Section 2.3.1. The methods employed in solving the optimization problem are numerous, ranging from simple heuristics, to linear programming (LP), to nonlinear programming-based (NLP) optimization methods.

2.3.1 Optimization constraints

The selection of an optimization method depends primarily on which factors will be used as constraints. One common approach is to constrain the motion kinematics in each axis with limits on the velocity, acceleration, and jerk, to prevent damage to the machine and to preserve part quality. The velocity constraint is selected based on the physical construction of the axes, determining the safe traverse velocities without inducing early damage to the guideways and motion delivery components. The acceleration constraint is selected based on maximum available motor torque, as the motor torque is correlated to the acceleration and velocity through the equivalent inertia and viscous damping in a feed drive system. The motor cannot supply unlimited torque, so if a very high acceleration is demanded, this will saturate the current delivered by the motor power supply amplifier, thereby leading to nonlinear dynamic behavior and invalidation of the linear stability analysis assumed in the commissioning of the servo feedback control systems.[37]

The jerk constraint is selected to reduce vibrations known to be caused by high rate-of-change in the acceleration [38]. Excessive jerk, as shown in Figure 2-3 (left-hand panel), generates acceleration commands, and therefore torque inputs, with rich high-frequency content. Such content can easily excite structural vibration modes dominated by the machine tool feed drive assembly, the part, the tooling, and/or the fixturing system, and resulting in unwanted vibration marks during machining. Excessive jerk can also lead to premature damage of the machine tool drive components. It has also been shown in literature [39] that a well-tuned feed drive control system typically exhibits a quasi-static servo error profile (excluding transient vibrations) which are correlated to the commanded velocity, acceleration, and jerk. Therefore, limiting jerk also indirectly helps retain a certain level of dynamic accuracy.

There are several advantages to adopting axis-level constraints. No prior knowledge of the machine's dynamics is required to generate the trajectories, unlike with physics-based or physics-inspired models, which must be derived, measured, and calibrated for each individual machine axis. Kinematic constraints are also frequently used, and are sometimes integrated with other constraints, in more elaborate feedrate optimization schemes, [13], [14], [36], [40], [41], [42], [43], as discussed in the proceeding.

In this thesis, a linear problem casting procedure is applied to approximately solve the optimization problem with kinematic constraints (Section 3.4), based on a method previously presented in literature and explained in the following. This method has been adopted to achieve a near-optimum initial guess as part of a larger and more elaborate nonlinear programming-based solution, which is also presented in Chapter 3.

Zhang et al. optimized a robotic manipulator trajectory by reformulating the problem into linear constraints through variable transformation and applying convex optimization [16], with a method introduced by Verscheure et al. [44]. These constraints are based on manipulator joint trajectories rather than the end effector position. A near-optimal solution can be solved by using linear programming (LP) [16]. Fan et al. applied the same method to five-axis CNC machining [15]. They presented a method in which the chord error, acceleration and jerk constraints are considered, with the jerk constraint being replaced by its linearized upper bound [15]. In later work, Erkorkmaz et al. [17] showed that the linear programming formulated solution can also be used to optimize different portions of a feedrate profile for a long toolpath independent of one another, in a method conducive to parallel programming rather than having to solve the complete toolpath sequentially in a forward progressing manner. The latter is the strategy typically used in CNC systems. Following the Principle of Optimality [45], which implies that a short section taken out of a long trajectory that is optimal, must itself be optimal as well, the work in [17] demonstrated that different portions of the overall optimum feed profile can be solved and connected by utilizing the local feed minima, which are influenced by the geometry of the toolpath, as shown in Figure 2-9. The work also demonstrated that the LP solution (used for producing a close initial guess in Chapter 3) is indeed robust and computationally efficient.

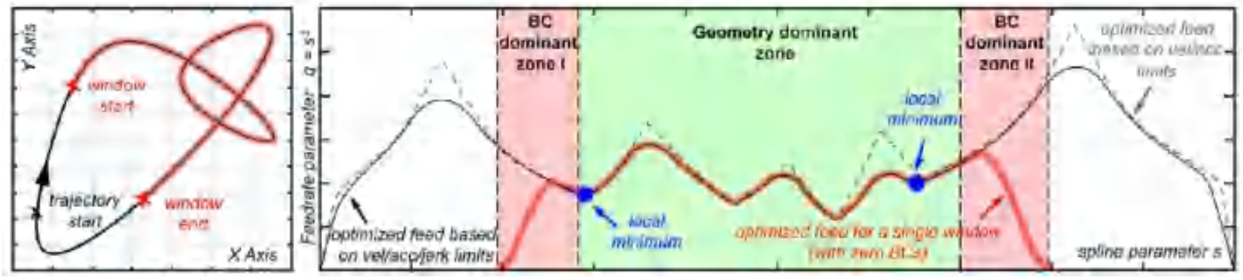


Figure 2-9: Optimized feed profile for a long (global) toolpath and its windowed (local) portion, considering zero boundary conditions for both cases [17].

For increased technological utility and sometimes better performance, it is possible to constrain certain physical factors directly, rather than relying on axis level constraints. One common option is to constrain the cutting force, as excessive cutting force will damage the tool and part. In a collaboration with the research team of Prof. Lazoglu (Koç University, Turkey), Erkorkmaz et al. combined axis kinematic constraints and cutting force constraints to generate optimal trajectories [46]. The result, shown in Figure 2-10 shows the effect of both sets of constraints on the final optimized feed profile.

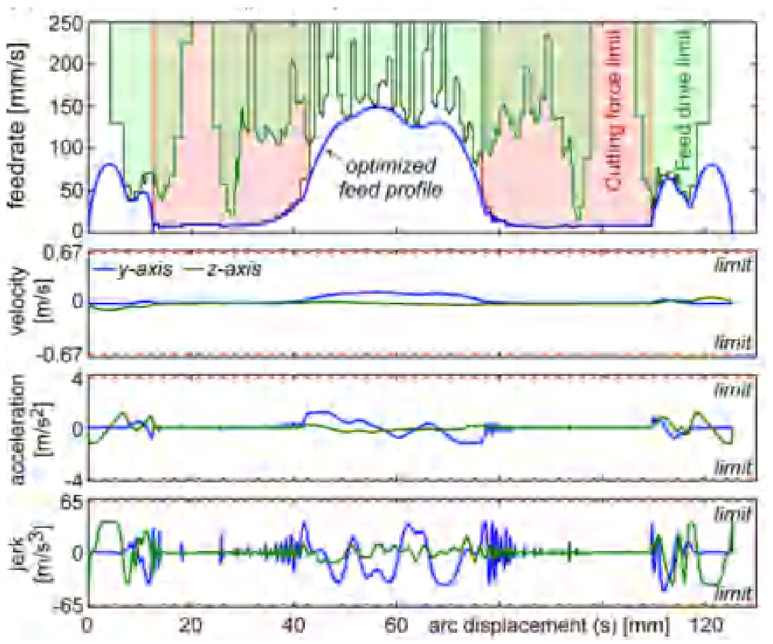


Figure 2-10: Application of axis kinematic and cutting force constraints in feed optimization. [46]

An earlier example is Kim and Kim (1996), who predicted tool force by measuring the current draw of AC motors, and used this as feedback in a real-time controller [47]. Other methods allow for the prediction of cutting force directly from the commanded trajectory. Ridwan et al. used fuzzy adaptive control to constrain the cutting force, and therefore the tool power required for cutting [48]. Xu and Tang controlled deflection cutting force using a "force-area quotient function" (a model of the relationship between the maximum material removal rate and the feed direction in a five-axis machining operation) to reduce the machining time [49].

Feed optimization can also be applied to constrain the dynamic positioning error of the machine tool servo control system, in order to achieve a certain degree of machined part accuracy. Feng and Su constrained scallop height and machining error in five-axis machining of 3-D surfaces.[50]

The torque demand from the motors may also be constrained to within safe limits. If a trajectory requires more torque than a feed drive motor is able to produce, the machine will not be able to follow it. This constraint was considered in early feed optimization for robots, such as in Bobrow et al. in 1985 [3]. More modern optimization methods may also apply torque constraints. Ferry and Altintas considered tool shank bending stress, tool deflection, max chip load, and torque limit as constraints in five-axis machining of jet engine impellers [51].

For any attribute to be considered as a constraint in the optimization, it must be modeled. In this thesis, as two practical constraints, the motor torque and servo error are considered, as discussed in Section 3.6.1. The motor torque (u) is computed by modelling the drive as an inertial element (i.e., mass - m), upon which viscous friction (b) and Coulomb friction (d_{coul}) are also active:

$$u = ma + bv + d_{coul} * sign(v) \qquad (2\text{-}1)$$

The tracking error (e) is modelled as a linear combination of the commanded axis level velocity (v_r), acceleration (a_r), and jerk (j_r). This formula is based on approximating the true servo error with a Maclaurin series (i.e., around zero frequency) in the frequency domain, and was developed and validated by Gordon and Erkorkmaz in [39]. The formula is not able to capture the influence of transient vibrations or stick-slip friction induced errors, but is successful in predicting the general waveform of the servo error, as can be see in Figure 2-11.

$$e = K_j j_r + K_a a_r + K_v v_r \qquad (2\text{-}2)$$

When considered in open form to include full expressions of the axis level acceleration and jerk profiles, both Eqs. (2-1) and (2-2) lead to nonlinear inequality constraints as a function of the optimization variables, as will be explained in detail in Chapter 3. However, their solution also can achieve trajectories that are both faster than those solved using linear programming for only axis velocity, acceleration, and pseudo-jerk, and which can also be tracked with comparable dynamic accuracy.

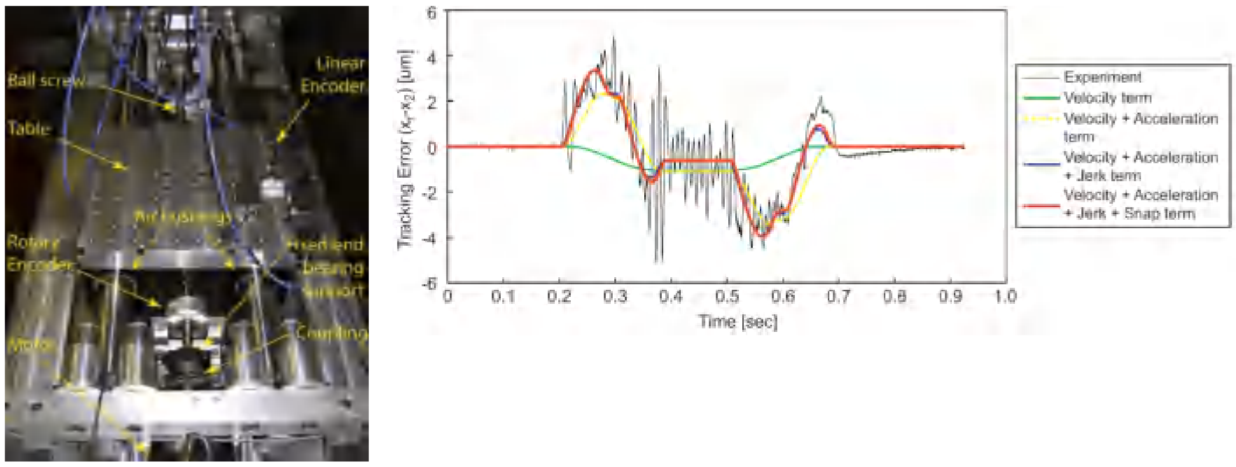
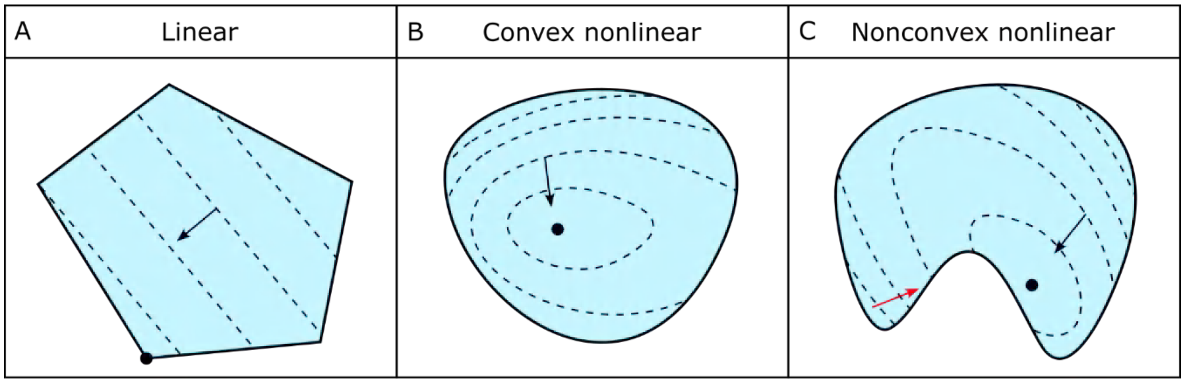


Figure 2-11: Quasi-static servo error response for a ball screw driven feed drive (left), linear error model based on the profiles of commanded velocity, acceleration, jerk, and snap (i.e., time-derivative of jerk).[39]

2.3.2 Review of optimization methods

The field of computational optimization is richly complex –countless algorithms exist, tailored to a variety of different scenarios and types of problem. The selection of an optimization method depends on the shape of the problem to be solved – notably whether the problem is *linear* and whether the problem is *convex.* A set can be described as convex if: for any two points in the set, the line connecting those two points lies entirely within the set. This is represented visually in the Figure 2-12, with Panels A and B (convex) compared with Panel C (nonconvex). In a convex optimization problem, both the feasible set and the objective function must be convex [52]. The result of this is that any *locally* optimal point will also be *globally* optimal. A nonconvex optimization problem may have many locally optimal solutions which are not the absolute (i.e., global) optimal point. Linear problems can be considered a special subset of convex optimization problems. All linear problems are convex, though the reverse is not necessarily true. In this subsection, several optimization methods are reviewed, and their advantages and disadvantages are discussed.



$$\min_{x\in\mathbb{R}^n} p(x) \qquad s.t. \begin{cases} a_i(x) = b \\ a_i(x) \geq b \end{cases}$$

Figure 2-12: Linear, nonlinear, and nonconvex optimization. Blue shading designates feasible solution set.

The optimization problem in Figure 2-12 is a minimization problem, to find the value of x which minimizes the function $p(x)$, subject to equality constraints $a_i(x) = b$ and/or inequality constraints $a_i \geq b$. Optimization problems in this section will be presented in this format.

2.3.2.1 Linear programming (LP)

Linear programming is a type of optimization in which all constraint and objective functions are affine (i.e. linear with respect to the free variables). An LP problem can be written as in Eq. (2-3).

$$\begin{aligned}&\min c^T x \\ &such\ that \\ &a_i x \le b_i\end{aligned} \tag{2-3}$$

In which, $c^T x$ represents a linear function of the variable x, and $a_i x \le b_i$ represents any number of linear constraint equations. Linear programming problems are relatively simple to solve computationally. The earliest examples of numerical optimization algorithms were designed to solve linear programming problems, using the very limited computational capacity of the time.

Panel A of Figure 2-12 shows a graphical representation of a 2-D linear programming problem, in which the goal is to minimize the objective function. The objective function is represented topologically, with dotted lines indicating iso-levels of the function. The polygon represents the "feasible region", i.e. the intersection of feasible solutions for all constraint functions. In an LP problem, all constraint functions must be affine – meaning that the feasible region is constructed from the intersection of "half-spaces" - the space to one side of a line in 2D or of a plane in 3D, which may be extended similarly into higher dimensions. The feasible region resulting from the intersection of linear constraints will take the shape of a polygon in 2D, or a polyhedron in higher dimensions. These shapes are always convex. The arrow in Figure 2-12 represents the decreasing direction of the objective function. The point at the bottom is the solution - the point at which the objective function is smallest within the feasible region. More detailed descriptions of linear programming can be found in [52], [53].

The solution of a linear programming optimization problem assumes one of three possible cases. The first case is one single solution, as shown in Figure 2-12. As can be seen, the optimum solution occurs at one of the corner points of the polygon (or polyhedron), which is a basic property utilized in the Simplex search method [54]. The second case is a problem with no solution, when the size of the feasible region is zero, and there are no points which satisfy all constraints. The third case is when there are multiple optimal points, which will occur only if the iso-level contour of the objective function is parallel to one of the constraints. In this situation, all optimal points will be on a single line, or plane (or hyperplane in higher dimensions). Linear programming problems are convex by definition. The linearity of the constraints guarantees that the resulting feasible region is convex.

While the axis-level constrained feedrate optimization problem is not linear, it can be approximated into a linear optimization problem, at the expense of reformulating the problem by considering a linearized upper bound approximation of jerk (i.e., pseudo-jerk), instead of the true jerk. This approach, explained in Section 3.4, leads to a slightly more conservative solution, but is very efficient and robust to solve.

2.3.2.2 Nonlinear programming (NLP)

NLP refers to a set of methods by which nonlinear optimization problems can be solved. Unlike with LP, NLP can find optimum solutions for problems defined with nonlinear objective functions and nonlinear constraints [52]. Due to the complexity of nonlinear optimization, numerous algorithms have been developed. Many NLP algorithms are suited to specific optimization conditions, and may be inefficient or inaccurate if used in other conditions. As such, the selection of a nonlinear optimization algorithm depends strongly on the type of problem being solved. In this thesis, sequential quadratic programming (SQP) and interior point (IP) optimization methods are discussed.

NLP - Interior Point (Barrier) Methods

Interior point methods make up a large category of optimization methods. These are also referred to as "barrier methods" due to the property that all iterative steps towards the solution (though not necessarily all tested points) must satisfy the inequality constraints of the problem. Thus, each step taken is in the "interior" of the feasible set. There are many ways to enforce boundary conditions. One method is through the use of logarithmic barrier functions. Such is presented as an illustrative example in the following, but other barrier enforcement methods may also be used when implementing interior point optimization.

A constrained minimization function (Eq. (2-4)) may be converted to an unconstrained minimization function (Eq. (2-5)) by representing the constraints as logarithmic barrier functions which influence the objective function.

$$\begin{cases} \min\limits_{\mathrm{x}\in\mathbb{R}^{\mathrm{n}}} p(x) \\ such\ that\ (s.t.) \qquad g_i(x) < 0, i = 1, \dots, m \end{cases} \tag{2-4}$$

$$\min_{\mathrm{x}\in\mathbb{R}^{\mathrm{n}}} p(x) + \frac{1}{t}\log\big(-g_i(x)\big) + \cdots + \frac{1}{t}\log(-g_m(x)) \tag{2-5}$$

This transformation offers several benefits. This new function is undefined outside of the feasible region, and approaches infinity towards the boundaries – so the descent direction of the function is steered away from the boundary. Therefore, by principle the optimization solver avoids cases where the function is undefined. It is important to add that the new objective function is analytically differentiable at all feasible points.

In Eq. (2-5), the parameter "t" controls the transition rate of the barrier functions. As $t \to \infty$, the influence of the logarithmic terms will diminish for solutions within the feasible set. The steps of the interior point method will typically start with a lower value of t, and will increase t gradually to higher values.

A numerical example of an interior point optimization is shown in Figure 2-13. The original optimization problem has a 2-variable quadratic objective function and six linear constraints. Due to the shape of the objective function, iterative steps will be guided towards the boundary, where the solver is prone to selecting an infeasible iterate. In this example, the solution lies on the boundary, which means that the solver may have difficulty selecting steps towards the solution without choosing infeasible iterates.

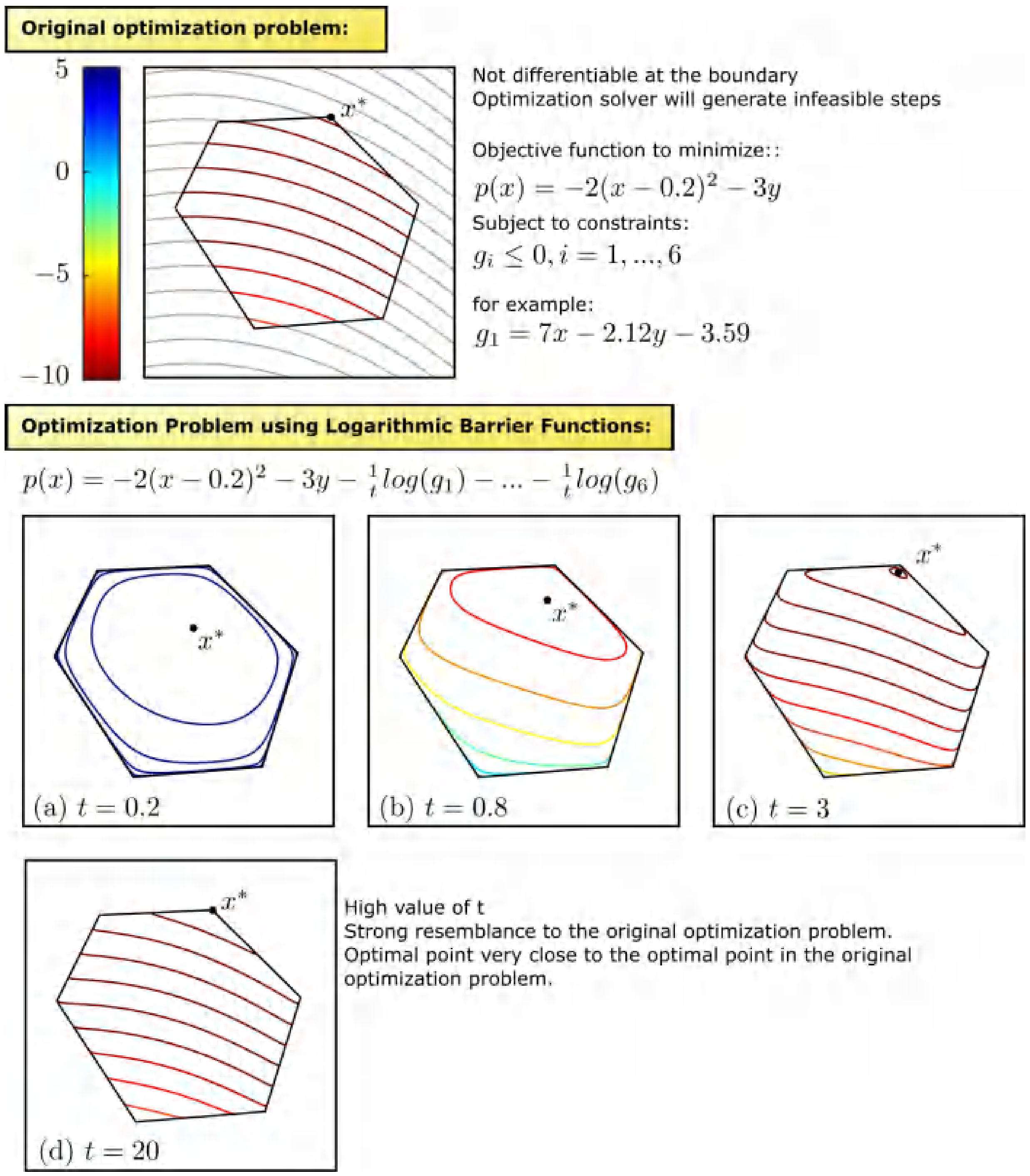


Figure 2-13: Interior point optimization using logarithmic barrier functions.

In the modified optimization problem, logarithmic barrier functions are applied. Now, the function is differentiable over all feasible points. It also goes to infinity towards the boundary – in this way, the solver is discouraged from leaving the feasible region. When t is small, the modified objective function does not resemble the original objective function very well (such as in Figure 2-13 (a) and (b)), but avoids the very sharp, sudden transitions around the boundary. For example, when $t = 0.2$, the barrier functions dominate, and the original objective function is not very distinguishable. However, as t is made larger in consecutive solution iterations (such as in Figure 2-13 (c) and (d)), the modified objective function now closely resembles the original (unconstrained) objective function as evaluated within the feasible solution space. In fact, with a value of $t = \infty$, the original problem is obtained.

A typical interior point optimization routine will alternate solving the modified objective function, and increasing the barrier parameter, t. The decision of when and how much to increase t to obtain the most efficient solution can be complex and will vary significantly depending on the solver.

After each modified optimization problem is solved, the obtained solution x^* is used as the starting point for solving the next modified optimization problem, following the update of the barrier parameter. If each increase to the barrier parameter is small, x^* will change very little, which guarantees the starting guess will be very close to the new solution. In this way, the modified optimization problem can be solved efficiently at each step. However, this will also require solving more modified problems before obtaining a solution to the full problem. If each increase to the barrier parameter is larger, then the starting guess in the modified optimization problem will be less accurate, and thus each modified optimization problem will be less efficient. There will, however, be fewer problems to solve overall. Detailed descriptions of interior point optimization are available in [52], [53].

When using logarithmic barrier functions, the starting guess of an interior point algorithm must always be a feasible point. This may be difficult to find, depending on the complexity of the constraints. In the case of a feed optimization problem, however, a feasible starting guess can always be found by setting the feedrate profile to be arbitrarily small. Many modern interior point algorithms are designed such that infeasible starting guesses are acceptable – the solver can take an infeasible starting guess by starting with another smaller optimization problem, which serves to generate a feasible point [55].

NLP – Sequential Quadratic Programming (SQP)

SQP is another method through which nonlinear optimization problems can be solved. Constraints and objective functions considered in SQP problems may be nonlinear and nonconvex – they are not required to be quadratic, despite the name. SQP problems may have any starting point, i.e., non-feasible starting points and iterates can also be used.

SQP is accomplished by creating a localized *quadratic sub-problem* at each step, then solving this problem using quadratic programming (QP) – for which there are many effective algorithms. SQP starts with a nonlinear optimization problem, such as the one in Eq. (2-4), which aims to minimize $p(x)$. A quadratic programming subproblem will then be created, solving for $p'(x)$, which is an approximation of $p(x)$ at the current iterate. This is shown in Eq. (2-6), comprising a quadratic objective function ($p'(x)$) and linear constraints. Information about quadratic programming can be obtained from [52].

$$\begin{cases} \min\limits_{x} p'(x) = \dfrac{1}{2} x^T G x + x^T c \\ \text{Subject to:} \\ a_i^T x = b_i, \quad i \in \mathcal{E} \\ a_i^T x \geq b_i, \quad i \in \mathcal{I} \end{cases} \tag{2-6}$$

The creation of the quadratic subproblem is depicted in Figure 2-14. A quadratic objective function is selected that best fits the actual objective function around the current iterate point (shown in red), typically by estimating the local gradient (c) and Hessian (G, i.e., second derivative matrix). Additionally, the nearby nonlinear constraints are approximated as linear constraints. This quadratic sub-problem is then solved to find the next iterate.

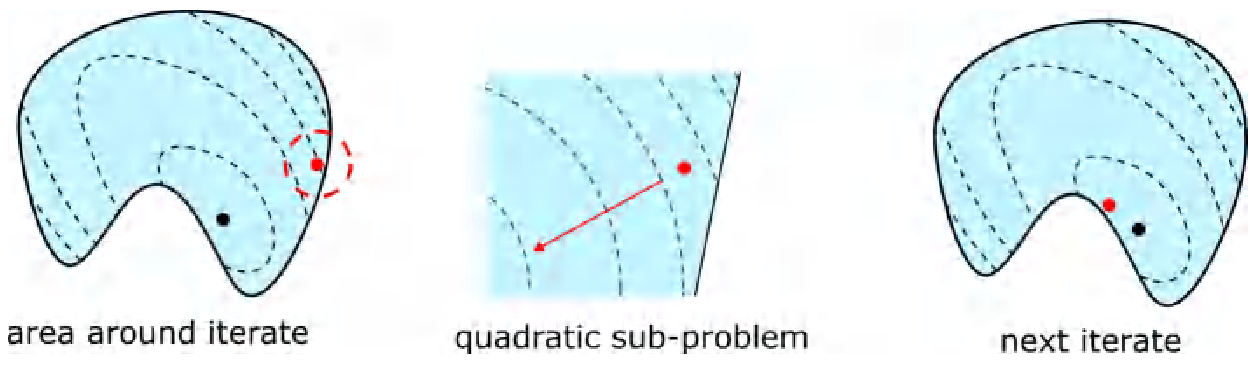


Figure 2-14: Sequential quadratic programming (SQP).

While the starting point and the intermediate steps in SQP may be *non-feasible*, the objective function can still be evaluated at these points. Therefore, it's not essential to find a feasible starting guess. However, the computational efficiency of these problems benefits significantly from a near-optimal starting guess. A mathematical overview of SQP can also be found in [56].

The feedrate optimization algorithms in this thesis were implemented in MATLAB, using the built-in optimization functions included in the software package. The two MATLAB nonlinear optimization functions used in this thesis are the "Interior Point" method and "SQP" optimization. Detailed descriptions and guidelines for their implementation can be found in [57].

The interior point optimization implements logarithmic barrier functions, as described in the preceding paragraphs. It implements a combination of optimization steps. The first is a "direct step" – which solves for the KKT (Karush-Kuhn-Tucker) equations, a set of conditions which determine whether a given solution is optimal in a specific problem. If this is not possible, the solver can also produce solution steps using a conjugate gradient method. These steps are interspersed with updates to the barrier parameter. This algorithm also includes steps which can compensate for infeasible starting conditions. It was found via trial-and-error that this solver is better able to recover from infeasible starting guesses, without increasing the solver time excessively. The interior point solver is not as efficient as the SQP solver, but may be more successful in cases with an infeasible starting guess. MATLAB's interior point algorithm is able to switch into a "feasibility mode" in the case of an infeasible starting guess, in which the solver switches to a different optimization routine whose only goal is finding a feasible iterate.

SQP optimization begins with obtaining the Hessian matrix at the current step. Following this, a quadratic approximation of the function is realized, which is solved via quadratic programming (QP). MATLAB's QP method is an "active set" method, in which at each step, only certain constraints considered "active" are included in the computation of the next step.

SQP was found to not be able to handle infeasible starting conditions well. It does include an additional "initialization" step, which should ideally be able to handle an infeasible starting guess. However, for the developed feed optimization problem it was observed by trial-and-error that this step usually increased the solution time significantly or caused the solver to fail. The SQP solver is the most time-efficient of the two methods, but only if the starting guess can be guaranteed to be feasible.

2.3.3 Windowing procedures to handle long toolpaths

As typical freeform machining toolpaths comprise of thousands to ten-thousands of spline segments, each of which may have tens to hundreds of free variables defining the feedrate profile control points, it is extremely difficult and also impractical to solve the optimum feed profile for a long and complex toolpath all in one shot. This creates problems both in terms of computational efficiency and available memory to store and process the necessary computations [17]. Instead, a "windowing" process is integrated with the feedrate optimization, to solve the optimum feed profile in parts, i.e., "windows" along the toolpath. This results in solving several smaller optimization problems with enforced boundary conditions, so that they can connect seamlessly with the neighbouring windows.

A variety of different window configurations have been used previously. Forward-looking algorithms are the most common [10],[13]. Tajima and Sencer created a smoothing process for linear toolpaths that uses a look-ahead window method to blend multiple corners [42].

Windows may be split into individual sections which can be solved separately as in [17], in which the algorithm is designed for a parallel computing environment and each window can be solved simultaneously. An implementation of this concept is shown in Figure 2-15.

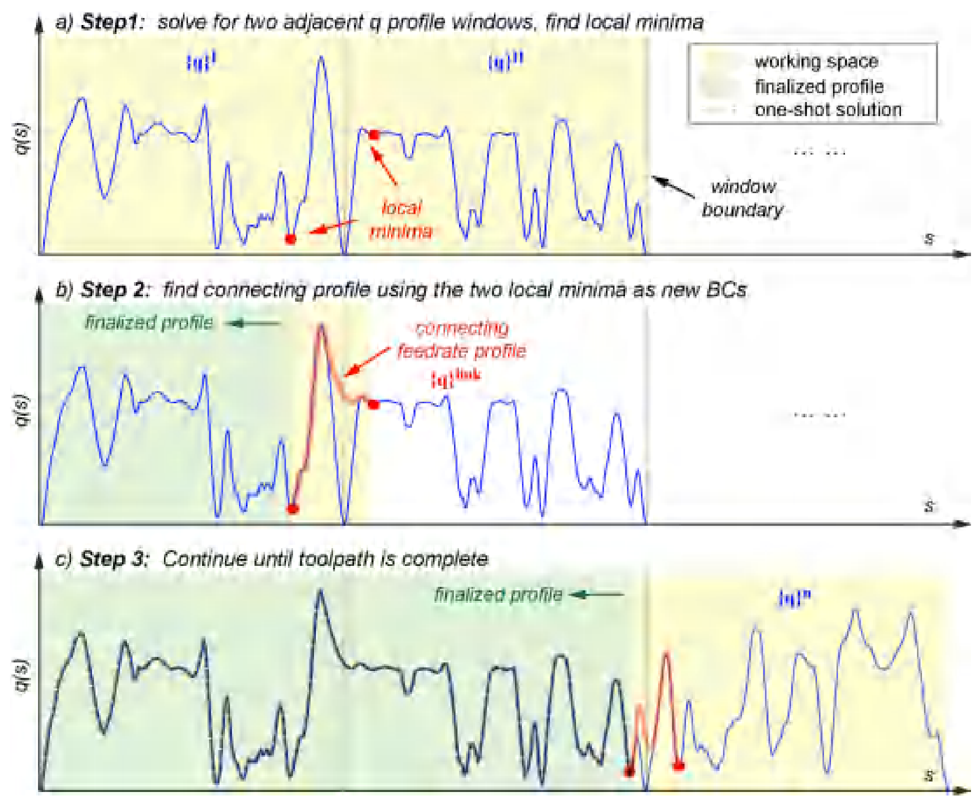


Figure 2-15: Concept of solving the optimum feedrate profile in connecting windows [4].

A toolpath may be designed such that each window has specific maxima of velocity, acceleration, and jerk, such as in [19], [58], [59].

In this thesis, the windowing method has been applied in a dual (nested) structure, as will be explained in Section 3.2. As the proposed feed optimization method utilizes a combination of LP (for fast and approximate solution of a near optimal feed profile) and then a consecutive NLP pass (for refining the optimality of the solution, subject to less conservative but nonlinear constraints), the LP steps are taken in the form of large windows, within which the NLP solutions, which are computationally more intensive, are then implemented as smaller window steps. Thus, feed profiles for indefinitely long toolpaths can be processed in a forward streaming manner, similar to the implementation structure in CNCs, while taking advantage of the speed of LP and further solution optimality that can be gained via NLP.

2.4 Conclusions

Extensive study has been devoted to the problem of reducing machining time while preserving part quality and machine condition in CNC machining.

Toolpath smoothing methods have been discussed in the literature review. Multiple different spline types have been used for CNC trajectories, including polynomial splines of different order, many varieties of B-splines, and other types of splines, all of which have their own advantages in obtaining a smoother toolpath, a better parametrization, or a more computationally efficient generation algorithm. In this thesis, a toolpath-smoothing method is developed which uses Euler spiral pairs to obtain G2-continuous toolpaths which are arc-length parametrized. The increased computational load of computing Euler spiral coordinate points (due to the required numerical integration) is partly compensated for by using an algorithm which integrates directly with a feed optimization procedure.

Many trajectory optimization strategies have been discussed in this literature review. Some of these strategies are based on the limitation of axis kinematic profiles (i.e., velocity, acceleration, and jerk). These methods are more conservative, but strategies have been developed that increase the computational efficiency of the optimization significantly. They also require no prior knowledge of the machine on which the trajectory will run. Other strategies seek to predict physical factors that affect part quality, and modify the trajectory to control these factors directly. Many possible constraints can be predicted and controlled in this way, such the force on the tool during cutting or the material removal rate. Among the factors which have been studied are commanded motor torque and tracking error. In this thesis, the computation efficiency of the velocity, acceleration and jerk constrained problem formulated for linear programming is combined with the optimality obtained from a torque and tracking error constrained nonlinear optimization.

3 Feedrate Scheduling through Synergistic Application of Linear and Nonlinear Optimization via Dual-Windowing

3.1 Introduction

The feedrate scheduling algorithm developed in this thesis applies a combination of two types of optimization in a windowed configuration, organized to utilize the distinct advantages of each method.

Using linear programming (LP), which is fast and robust, it is possible to obtain a near-optimal solution to the axis kinematics-constrained feed optimization problem (i.e., the minimization of motion time with respect to axis level velocity, acceleration, and jerk magnitude limits). This is achieved by replacing the jerk profile with a slightly conservative upper bound, named 'pseudo-jerk' [15], [16], [17]. Conveniently, such axis kinematics (vel/acc/jerk) are commonly used in CNC machine tool controllers for feedrate optimization. However, using these limits is also a conservative means to guarantee servo accuracy and motor current limiting, thus can lead to the underutilization of the equipment.

In contrast, directly solving the trajectory optimization subject to only servo error and motor current magnitude constraints, which are nonlinear by nature, can produce trajectories with even shorter motion time. However, such trajectories cannot be solved by LP, and their solution necessitates application of nonlinear optimization – also referred to as nonlinear programming (NLP). NLP is computationally more expensive, but its convergence can be significantly accelerated if the objective function and constraints are analytically differentiable, and a feasible and near-optimal initial guess is provided. The feed optimization problems defined in this research already satisfy the differentiability requirement (i.e., the objective and constraint functions are all continuous with respect to the optimization parameters). Furthermore, the dual windowing LP+NLP method proposed in this thesis (introduced in Section 3.2), applies LP to obtain a near-optimal and feasible initial guess to the nonlinear optimization problem.

In the proposed method, a near-optimal feed profile is first solved for a sufficiently long look-ahead window by employing LP, based on velocity, acceleration, and jerk limit constraints. Then, the solution obtained via LP is refined by applying NLP over shorter and overlapping portions. As NLP is computationally much more expensive, the window size is kept smaller. In order to process indefinitely long toolpaths in a forward-looking manner, as CNC code is normally read and parsed, the LP optimization is then applied to the consecutive portion of the toolpath with some overlap with the earlier LP window. The new portion of the toolpath is afterwards refined via NLP, and this algorithm is repeated while shifting the optimization windows for LP and NLP forwards along the toolpath.

Thus, the advantages of both methods, i.e., fast initial computation with LP and motion time optimality with NLP, are effectively utilized. Furthermore, the windowing and boundary condition matching functionalities, which are detailed in the proceeding subsections, enable efficient processing of indefinitely long toolpaths while guaranteeing the existence of a feasible solution during both LP and NLP steps.

Both LP and NLP optimization methods utilize B-spline representation to formulate the progression profile to be optimized, with the free optimization variables coinciding to the control points of the B-spline function. Since B-spline formulation is used, a solution within a window can be expediently computed and updated by simply modulating the control points. The resulting effect on the kinematic profiles and constraint functions is easily computable and confined only within a known localized portion of the toolpath. This 'local' property of B-spline has been key in developing and implementing the proposed LP+NLP windowed solution.

The proceeding sections of this chapter are organized as follows: The general progression of the proposed dual windowed feed optimization is described in Section 3.2. The B-spline parameterization of the progression (which coincides with feedrate $\dot{s}$ in NLP, and its squared term $q = \dot{s}^2$ in the case of LP) is explained in Section 3.3. This is followed by the LP solution for $q = \dot{s}^2$ in Section 3.4, its re-parametrization as a B-spline in terms of $\dot{s}$ in Section 3.5. The casting and solution of the NLP problem is then explained in Section 3.6. The dual windowing method requires careful coordination and updating of the boundary conditions while switching between overlapping LP and NLP solutions, in order to preserve the kinematic compatibility and constraint feasibility conditions throughout the toolpath. The details of these tasks are explained in Section 3.7. The feed optimization is solved in LP and NLP as a function of the arc displacement ($\dot{s} = \dot{s}(s)$). However, the arc displacement profile ultimately needs to be reconstructed in the time domain as $s = s(t)$. The numerical integration implemented for this functionality is explained in Section 3.8. The simulation and experimental results obtained from the implementation of the algorithm are presented and discussed in Sections 3.9 and 0. The conclusions for this chapter are then presented in Section 3.11.

3.2 Dual windowing LP+NLP optimization

The windowing procedure, introduced in Section 2.3.3, splits the toolpath into smaller sections on which optimization can be performed sequentially. Since the problem size is diminished, faster convergence in each iteration can be achieved. By reducing the memory requirements, the possibility of the optimization algorithm taking an excessive amount of time, or failing to converge is also mitigated.

The proposed optimization method uses a novel combination of LP and NLP in a serial windowed configuration, as shown schematically in Figure 3-1. The computationally faster, but performance-wise more conservative LP solution is obtained first for a large look-ahead window. The trajectory is afterwards refined for better motion time optimality using shorter range windows with NLP.

As mentioned earlier, the main difference between the LP and NLP optimization is the choice of constraints. Feed optimization via LP optimization (described in detail in Section 3.4) uses velocity, acceleration, and jerk constraints, with jerk being capped by a slightly conservative upper bound, 'pseudo-jerk' to enable linear constraint formulation. Feed optimization developed via NLP (described in detail in Section 3.6) focuses directly on the motor torque (u) and tracking error (e) as constraints and relaxes the bounds on velocity, acceleration and jerk (although the latter three could also be included into the NLP if needed). Switching from the velocity, acceleration, jerk constraints in LP to the torque and servo error constraints in NLP yields a solution that is less conservative, i.e., allowing faster movements with higher accelerations and shorter motion time. However, after NLP is performed, the resulting trajectory may be *infeasible* according to the original LP constraints, which would lead to infeasible boundary conditions or trajectory sections being passed onto the adjacent LP window. Therefore, at the starting and ending locations of the windows, the boundary conditions imposed for the LP and NLP sub-problems must be carefully coordinated.

Considering Figure 3-1, the dual windowing algorithm progresses as follows:

- **Step 1:** The LP problem is defined and solved,
    - A single large window is processed. The travel length, W_{LP}, is several-fold longer than the required distance for the slowest axis to accelerate from rest to full speed, based on the given velocity, acceleration, and jerk constraints, and return to rest again. The first LP window is between $s = 0$ and $s = W_{LP}$.
    - LP optimization is performed on this part of the trajectory.
- **Step 2:** The NLP problem is defined and solved,
    - The NLP solution is solved in overlapping 'short' window steps of length W_{NLP}, starting from $s = 0$, and traversing to the arc distance of $s = W_{LP} - P_{LP}$. Here, P_{LP} is a buffer distance to "pause" the NLP solution exactly at the planned start of the consecutive (large) LP window (see panel titled 'Step 1, repeat'). This guarantees that NLP does not modify the already feasible starting boundary condition for the proceeding LP window, that is produced by the current LP solution. Otherwise, modification of the state at this point could potentially produce an infeasible problem definition for the proceeding LP window.
    - The nonlinear optimization is performed sequentially in sub-windows overlapping by the distance (P_{NLP}), until the end of the current LP section is reached.
- **Step 1, repeat:** LP is performed on the consecutive section, from $s = W_{LP} - P_{LP}$ to $s = 2W_{LP} - P_{LP}$.
- **Step 2, repeat:** Nonlinear optimization is performed in small sub-windows, this time across the section from $s = W_{LP} - P_{LP} - P_{NLP}$ to $s = 2W_{LP} - 2P_{LP}$. Hence, the solution continues with

adequate overlap, from where the last NLP in the previous large lookahead window had left off. Note that this overlaps the very last NLP solution by P_{NLP}

- This process continues, with regular shifts of the LP and NLP windows per the above pattern, until the end of the toolpath is reached.

As a general term, considering the *m*th LP window, the solution range can be formulated as in Eq. (3-1)

$$s_{LP} = s \in [(m-1)W_{LP} - (m-1)P_{LP}, mW_{LP} - (m-1)P_{LP}] \tag{3-1}$$

Similarly, the *n*th NLP (small) window within the *m*th LP window is active in optimizing the feed for the arc displacement range given in Eq. (3-2):

$$\begin{aligned} s_{NLP} = \ s \in [&(m-1)W_{LP} - (m-1)P_{LP} + (n-1)W_{NLP} - (n-1)W_{NLP}, \\ &(m-1)W_{LP} - (m-1)P_{LP} + nW_{NLP} - (n-1)W_{NLP}] \end{aligned} \tag{3-2}$$

And these small windows continue until, $s = mW_{LP} - mP_{LP}$.

A practical example of the application of this windowing integrated optimization is shown in Figure 3-2. Detailed descriptions of the algorithm are given in the remainder of this chapter. Methods to determine window sizes (W_{LP}, W_{NLP}) and overlap (P_{LP}, P_{NLP}) are discussed in Section 4.1.

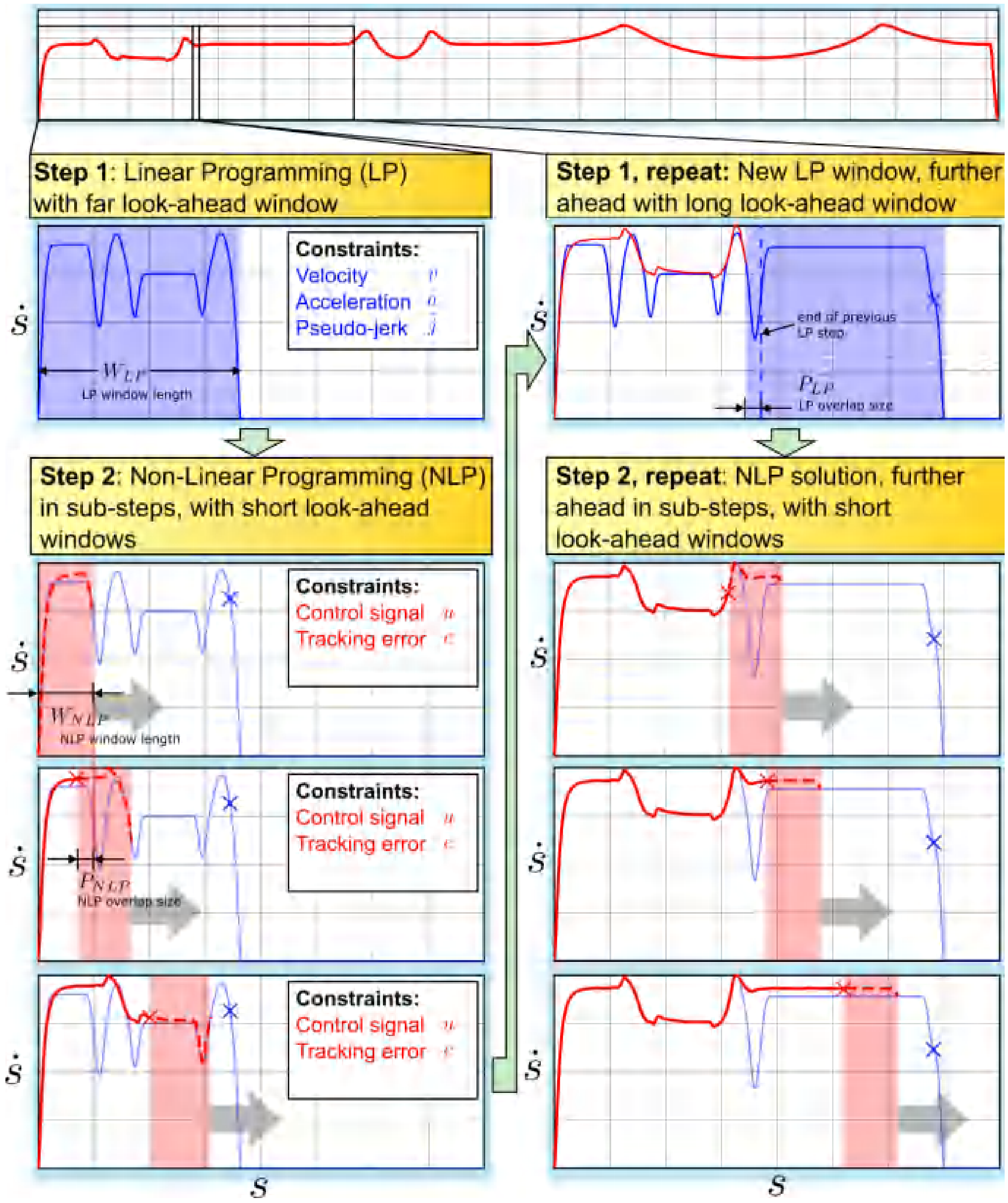

Step 1: Linear Programming (LP) with far look-ahead window
Constraints:
Velocity v
Acceleration a
Pseudo-jerk $\bar{j}$
W_{LP}
LP window length
$\dot{s}$
Step 2: Non-Linear Programming (NLP) in sub-steps, with short look-ahead windows
Constraints:
Control signal u
Tracking error ϵ
W_{NLP}
NLP window length
P_{NLP}
NLP overlap size
s
Step 1, repeat: New LP window, further ahead with long look-ahead window
end of previous LP step
P_{LP}
LP overlap size
Step 2, repeat: NLP solution, further ahead in sub-steps, with short look-ahead windows

Figure 3-1: Sample section of dual windowing optimization procedure.

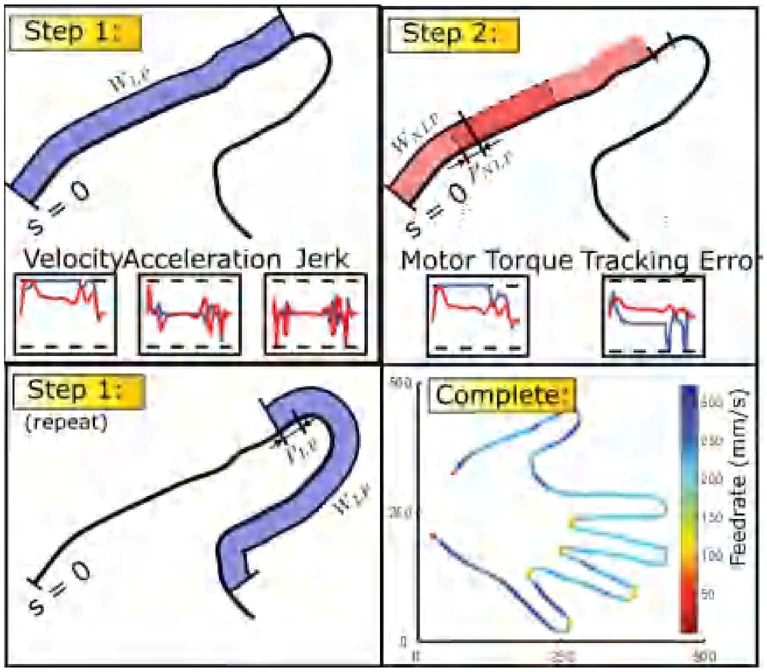


Figure 3-2: Illustration of dual windowing steps.

3.3 Parameterization of the tool progression (i.e., feedrate $\dot{s}$ or its square $\dot{s}^2$) as a B-spline function

3.3.1 Introduction

A B-spline function is constructed as the summation of a series of B-splines, or “basis” splines. The B-splines are constructed using piecewise polynomial curves. They are defined by *knots*, which control the shape and placement of individual B-splines, and *control points*, which act as scaling values for each individual B-spline.

Figure 3-3 shows an example of a B-spline function. The knots are indicated by the vertical lines. Figure 3-3a illustrates a set of B-splines $b_i(s)$ created based on a chosen spline order and knot profile (which in this case is monotonically increasing). Figure 3-3b demonstrates the influence of each control point c_i, in scaling its respective B-spline $b_i(s)$. Then, the summation of the scaled B-splines is shown in Figure 3-3c, which is designated as the final ‘B-spline function’ $f(s)$.

Note that in this thesis, B-splines are used to represent either the feedrate, $f(s) = \dot{s}$, or its square $q(s) = \dot{s}^2$, in the context of solving $\dot{s}$ or $\dot{s}^2$ as a nonlinear or linear optimization problem, respectively. In the general descriptions of B-splines, $f(s)$ is used for exemplary notation, but the same mathematical formulations apply also to $q(s)$.

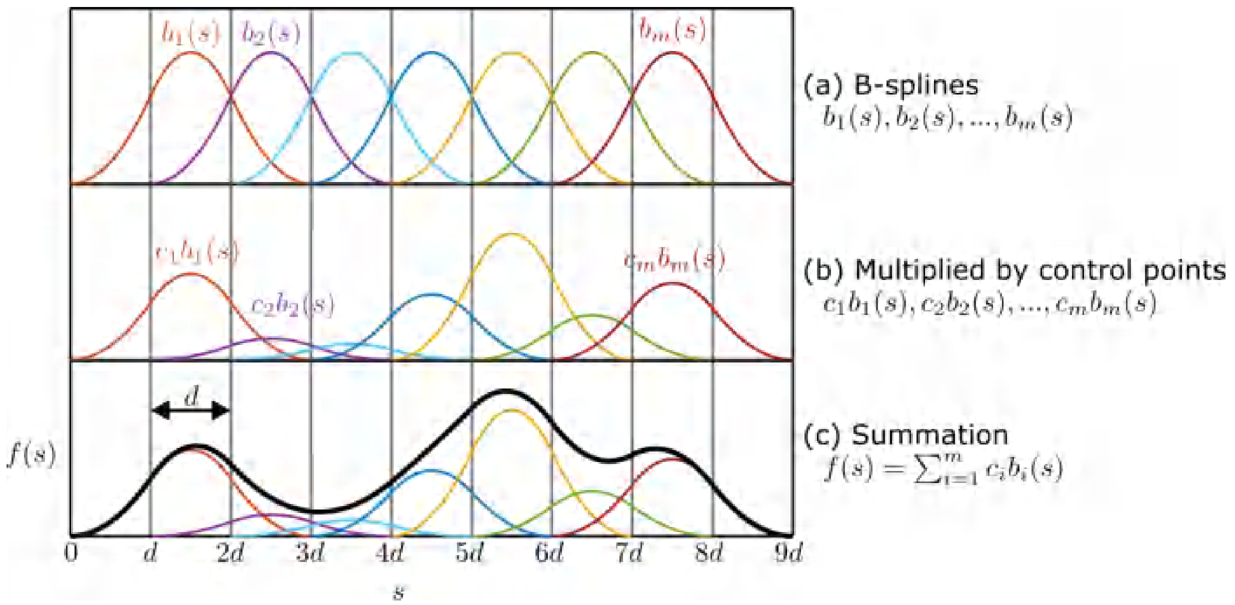


Figure 3-3: B-spline function from individual B-splines.

B-splines vary based on several factors, including order and knot placement. A B-spline of order n will span a range of $n + 1$ knots, and will be constructed of n polynomial curve segments, where each polynomial is of degree $n - 1$. The B-spline function $f(s)$, within a range of two consecutive knots, will have a maximum of n local B-splines influence it. This means that the control points not attributed to the influencing (or 'active') B-splines have no effect on $f(s)$ within this range. Thus, B-spline functions offer the advantage of being able to accommodate local changes simply by adjusting the adjacent control points, without impacting the rest of the B-spline function.

The B-splines shown in Figure 3-3 are *quadratic* – i.e., order = 3 and degree = 2. Each individual B-spline spans 4 knots, and is constructed of 3 polynomial segments, each of degree 2. Figure 3-3 also shows a simple distribution of quadratic B-splines with evenly spaced knots, which is a special case adopted in this thesis for mathematical convenience. B-splines can, in general, be created with uneven non-negative spacing between the knots, including zero spacing – in which multiple knots exist at the same location (i.e. "knot multiplicity"). The latter can be used to enforce certain boundary conditions. Detailed mathematical descriptions of B-splines are available in [60], [61], [62].

3.3.2 General description of the B-spline function

A B-spline can be generated using the following recurrence relation [61], originally developed by de Boor. [62]. In a B-spline function of order n, with knot sequence ξ, the ith B-spline is given by $b_{i,\xi}^{n}$, using Eqs. (3-3) and (3-4). Note that the B-spline order "n" is expressed as a superscript in this notation, and it is *not* an exponent.

$$b_{i,\xi}^{n}(s) = \gamma_{i,\xi}^{n}(s) \cdot b_{i,\xi}^{n-1}(s) + \left(1 - \gamma_{i+1,\xi}^{n}(s)\right) \cdot b_{i+1,\xi}^{n-1}(s) \tag{3-3}$$

Where:

$$\gamma_{i,\xi}^{n}(s) = \frac{s - \xi_i}{\xi_{i+n-1} - \xi_i} \tag{3-4}$$

For which the starting point is the "characteristic function" in Eq. (3-5), which is the definition of a first order (i.e., zeroth degree) B-spline.

$$b_{i,\xi}^{1}(s) = \begin{cases} 1 & for\ \xi_i \leq s < \xi_{i+1} \\ 0 & otherwise \end{cases} \tag{3-5}$$

This recurrence relation can be solved for any nondecreasing set of knots $[\xi_1, \xi_2, \dots]$, with the desired spline order n. However, when the nature of the B-spline is a special case, the formula can be significantly simplified. One such case is a *uniform B-spline function*, which is composed of B-splines with evenly spaced knots, such as the ones in Figure 3-3.

3.3.3 Specific case: Uniform B-spline function

Applying evenly spaced knots in the definition of the B-splines leads to a uniform B-spline function. This simplifies the analysis of the B-spline function and streamlines its computation. In this case, the knot separation distance d is specified, and all knots are defined as $\xi_i = d \cdot i$. Derivation of the uniform B-spline equations, including their derivatives, can be found in Appendix A.1.

For a quadratic B-spline function with evenly spaced knots, such as the one depicted in Figure 3-3, the B-spline between $\xi_i = d \cdot i$ and $\xi_{i+3} = d \cdot (i+3)$ is defined by three segments, given in Eq. (3-6).

$$b_i^3(s) = \begin{cases} \dfrac{(s-di)^2}{2d^2} & for\ di \leq s < d(i+1) \\ \dfrac{-2s^2 + sd(4i+6) - d^2[2i^2+6i+3]}{2d^2} & for\ d(i+1) \leq s < d(i+2) \\ \dfrac{(d(i+3)-s)^2}{2d^2} & for\ d(i+2) \leq s < d(i+3) \\ 0 & otherwise \end{cases} \tag{3-6}$$

Above, s represents the overall arc displacement from the very beginning of the toolpath. In the computer implementation, the above formulation can be further simplified by re-parameterizing with respect to $s' = s - d \cdot i$ (i.e., the arc displacement from the starting knot of each B-spline).

Similarly, a cubic uniform B-spline is as a special case of the B-spline, is defined between $\xi_i = d \cdot i$ and $\xi_{i+4} = d(i+4)$ as four segments as shown in in Eq. (3-7).

$$b_i^4(s) = \tag{3-7}$$

$$
\begin{cases}
\dfrac{(s-di)^3}{6d^3} & for\ di \le s < d(i+1) \\
\dfrac{-3s^3+(9i+12)ds^2-3d^2(3i^2+8i+4)s+d^3(3i^3+12i^2+12i+4)}{6d^3} & for\ d(i+1) \le s < d(i+2) \\
\dfrac{s^3-(3i+8)ds^2+(3i^2+16i+20)d^2s-\left(i^3+8i^2+20i+\frac{44}{3}\right)d^3}{2d^3} & for\ d(i+2) \le s < d(i+3) \\
\dfrac{(d(i+4)-s)^3}{6d^3} & for\ d(i+3) \le s < d(i+4) \\
0 & otherwise
\end{cases}
$$

In this thesis, quadratic B-splines with uniform knot spacing have been used. This function achieves first order continuity of its derivative with respect to the arc displacement (e.g., $\dot{s}_s = d\dot{s}/ds$- here, a subscript of s indicates derivative with respect to s, for instance $f_s(s) = df(s)/ds$ and $f_{ss}(s) = d^2f(s)/ds^2$). This enables the resulting the tangential acceleration profile to be continuous ($\ddot{s} = \dot{s}_s\dot{s}$), and the tangential jerk ($\dddot{s} = \ddot{s}_{ss}\dot{s}^2 + \dot{s}_s\dot{s}$) and axis level jerk ($\dddot{\boldsymbol{r}} = [\dddot{x} \quad \dddot{y} \quad \dddot{z}]^T$) to be bounded.

For simplicity, in the remainder of this chapter and throughout the rest of the thesis, the order indication via superscript will be dropped. Thus, a single B-spline is henceforth expressed as in Eq. (3-8), and will be a quadratic B-spline, unless explicitly stated otherwise.

$$b_i^3(s) = b_i(s) \qquad (3\text{-}8)$$

In this case, the B-spline function, defined as the summation of each B-spline b_i multiplied by its corresponding control point c_i, using the notation in Eq. (3-8) can be expressed as in Eq. (3-9).

$$f(s) = \sum_{i=1}^{m} c_i b_i(s) \qquad (3\text{-}9)$$

Thus, the structure of every B-spline in the entire (feedrate or feedrate-square) profile is conveniently represented using the same set of equations, which greatly simplifies the formulation and computational implementation. Section 3.3.5 describes the meshing of constraint checkpoints along the B-spline function, and the utilization of the uniform knot spacing to gain the advantage of reducing the computational burden during the evaluation of the optimization constraints.

3.3.4 Derivative profiles for uniform B-splines

Besides their ‘locality’ attribute, another major advantage of B-spline functions is that they are analytically differentiable, and the derivatives (which can be determined easily) are linear with respect to the control points. Figure 3-4 shows uniform quadratic and cubic B-spline functions along with their derivatives with respect to s.

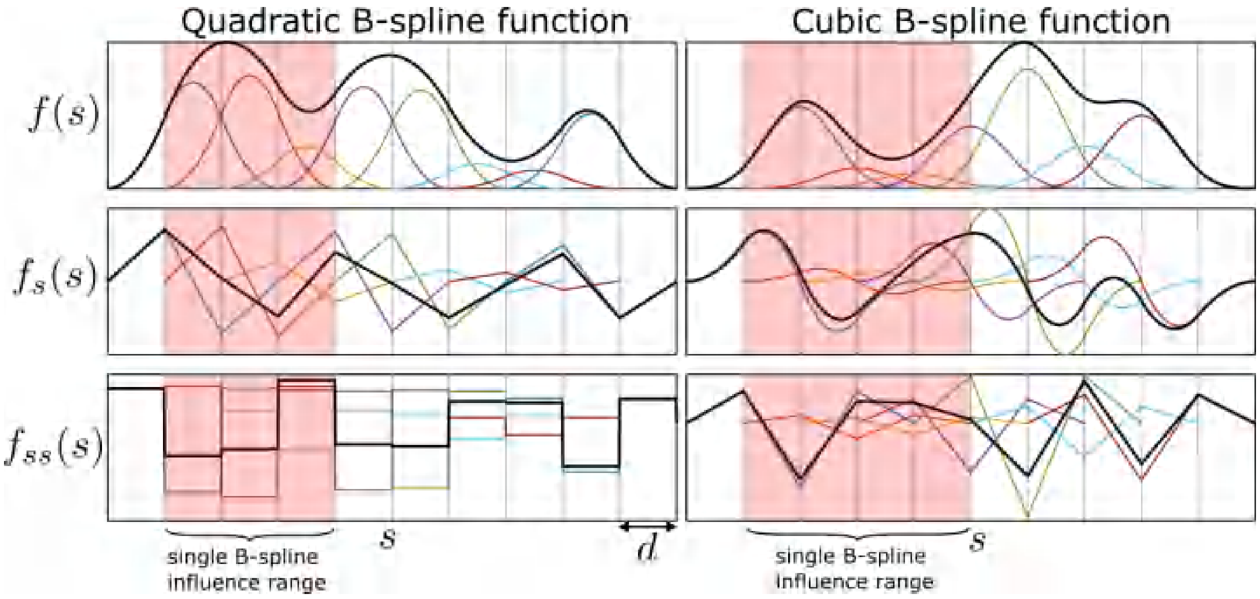


Figure 3-4: Quadratic and cubic B-splines.

In the general case, the first and second derivatives of $f(s)$ with respect to s can be expressed as:

$$f_s(s) = \frac{df(s)}{ds} = \sum_{i=1}^{m} c_i [b_i]_s(s) \tag{3-10}$$

$$f_{ss}(s) = \frac{d^2 f(s)}{ds^2} = \sum_{i=1}^{m} c_i [b_i]_{ss}(s) \tag{3-11}$$

Extending from Eq. (3-6), the first and second derivatives (b_s and b_{ss}) of a single uniform quadratic B-spline with a knot separation distance d can be represented as in Eqs. (3-12) and (3-13).

$$[b_i]_s(s) = \frac{d}{ds}\big(b_i(s)\big) = \begin{cases} \dfrac{s-di}{d^2} & for\ di \le s < d(i+1) \\ \dfrac{-2s + d(2i+3)}{d^2} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{1}{d^2}s - \dfrac{i+3}{d} & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases} \tag{3-12}$$

$$[b_i]_{ss}(s) = \frac{d^2}{ds^2}\big(b_i(s)\big) = \begin{cases} \dfrac{1}{d^2} & for\ di \le s < d(i+1) \\ \dfrac{-2}{d^2} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{1}{d^2} & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases} \tag{3-13}$$

In the case of a cubic B-spline, the first and second derivatives (b_s and b_{ss}) can be expressed as in Eqs. (3-14) and (3-15).

$$[b_i]_s(s) = \frac{d}{ds}\big(b_i(s)\big)$$
$$= \begin{cases} \dfrac{s^2 - 2dis + i^2d^2}{2d^3} & for\ di \le s < d(i+1) \\ \dfrac{-3s^2 + 2(3i+4)ds - d^2(3i^2+8i+4)}{2d^3} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{3s^2 - (6i+16)ds + (3i^2+16i+20)d^2}{2d^3} & for\ d(i+2) \le s < d(i+3) \\ \dfrac{-d^2(i+4)^2 + 2ds(i+4) - s^2}{2d^3} & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases} \quad (3\text{-}14)$$

$$[b_i]_{ss}(s) = \frac{d^2}{ds^2}\big(b_i(s)\big) = \begin{cases} \left(\dfrac{1}{d^3}\right)s - \left(\dfrac{i}{d^2}\right) & for\ di \le s < d(i+1) \\ -\dfrac{3}{d^3}s + \dfrac{(3i+4)d}{d^3} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{3s - (3i+8)d}{d^3} & for\ d(i+2) \le s < d(i+3) \\ \dfrac{2d(i+4) - 2s}{2d^3} & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases} \quad (3\text{-}15)$$

The development of the above derivative expressions is presented in Appendix A.1.

The ease of computing the derivatives brings important advantages. Notably in feedrate optimization, the profile derivatives must be computed frequently and repeatedly. Since the spline function and its derivatives are linear with respect to the control points, when evaluating the constraints (and their gradients), the intermediate calculations of the progression ($\dot{s}$ or $q = \dot{s}^2$) and the derivatives (e.g., $\dot{s}_s$, $\dot{s}_{ss}$, which are then used to construct $\ddot{s}$, $\dddot{s}$), are simply solved by multiplying the control points with a pre-computed constant matrix (based on the constraint checkpoint resolution), as shown in Section 3.3.5. This greatly accelerates the constraint evaluations. Furthermore, the ability to directly compute the constraint gradients analytically, without relying on numerical differentiation, also enhances the convergence during NLP.

In this thesis, the objective was to obtain trajectories with limited jerk, for which using a quadratic B-spline function to formulate the feedrate (or its square) was sufficient (utilizing Eqs. (3-6), (3-12), (3-13)). If jerk continuity is also required, then the optimization methods can be expanded to rely on cubic B-splines (as provided in Eqs. (3-7), (3-14), (3-15)), albeit at the cost of slightly increased complexity and computational load.

3.3.5 Control and evaluation points for the B-spline function

The numerical optimization represents the continuous B-spline function as a series of points at which the constraint values are checked. As mentioned earlier, the function is modulated by the control points c_i, separated by the knot distance d. However, in evaluating the constraints (which are a function of the progression and its derivatives), the B-spline needs to be reconstructed at a higher resolution. For implementation convenience and numerical efficiency, this resolution has been chosen, in this thesis, as an integer subdivision of the knot distance.

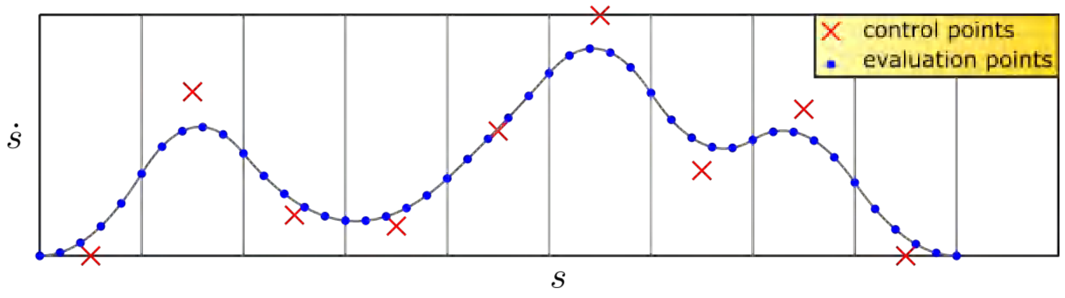


Figure 3-5: Placement of control and evaluation points throughout a trajectory.

To generate the evaluation points, an integer constraint meshing factor N_c is specified defined as the number of evaluation points per control point. N_c remains constant throughout the optimization process. For a function with r control points, the number of evaluation points is defined as N, where $N = rN_c$. The space between evaluation points is then a subdivision of the knot spacing, i.e., d/N_c.

In determining the meshing factor, a good practice would be to determine the smallest possible curvature in the toolpath and to assign the constraint checking arc displacement accordingly, e.g., to be several times smaller than the minimum radius of curvature, in determining the adequate meshing factor N_c for a given knot spacing d.

For a B-spline function with r control points and N evaluation points, the function at all evaluation points can be computed using Eq. (3-16):

$$f_{(N\times 1)} = [B]_{(N\times r)} * c_{(r\times 1)} \tag{3-16}$$

Above, the matrix B is precalculated by enumerating the B-spline equations based on the knot separation distance, d, the control points, and the displacement parameter (s) at the corresponding evaluation points (Eq. (3-3) - (3-5)). The B matrix for the entire toolpath is large but sparse. Each LP and NLP step is associated with a smaller matrix, B', which is a subset of B. As the structure of B' repeats itself along the block diagonal of B, it is not necessary compute and store the complete B matrix. Figure 3-6 exemplarily shows the structure of the B and B' matrices for a quadratic B-spline profile with 3 evaluation points per control point ($N_c = 3$).

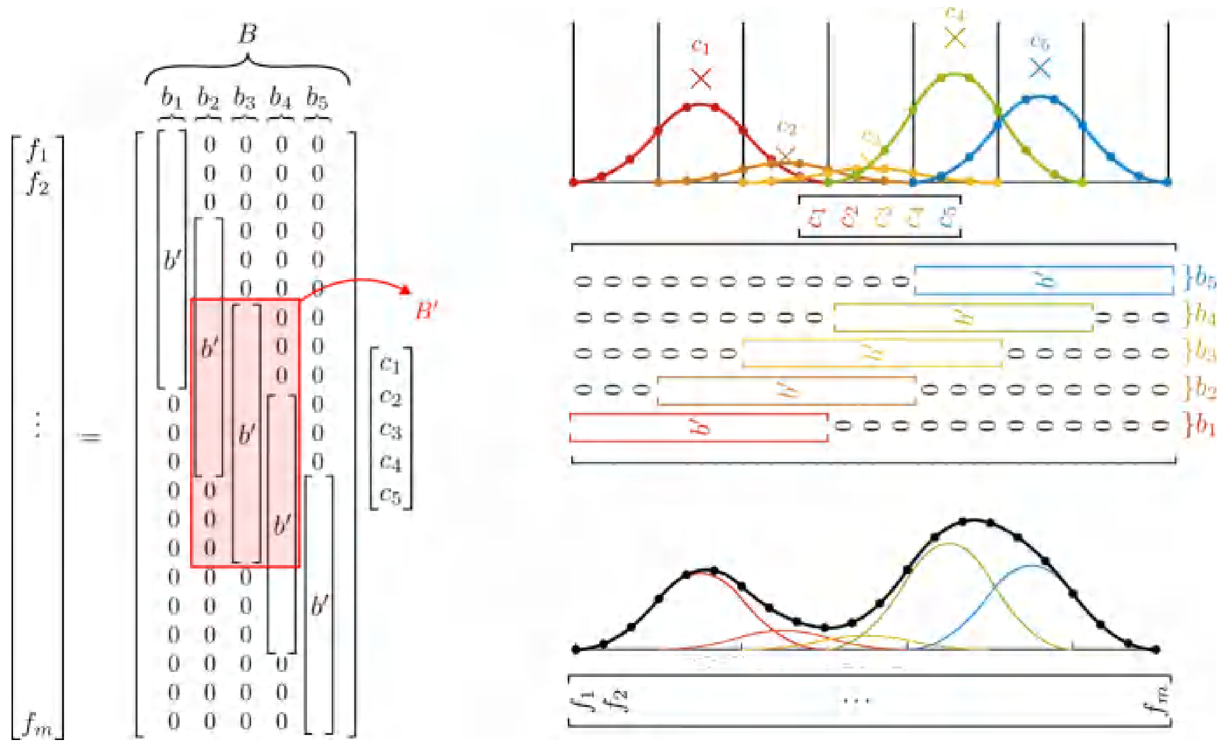


Figure 3-6: Structure of the B matrix and B' submatrices in B-spline computation.

In the interest of computational efficiency, the point alignment is set such that the B' matrix does not change. This is true when the number and alignment of evaluation points per control point remains constant throughout the optimization procedure, and thus the size of each window remains fixed. Due to this alignment, every B-spline in the function is defined by the same set of points, which are then multiplied by the corresponding control point c_i. In Figure 3-6 this is represented as b', which must only be computed once for a single B-spline basis function and distributed into the submatrix B'. The dimensions of b', for a quadratic B-spline, are $[3N_c \times 1]$. The entire B matrix, or any submatrix B', can be computed by aligning b'. Two B' matrices will be used throughout most of the optimization – one for the LP windows (with longer range) and one for the NLP windows (which are shorter). As needed, additional B' matrices may also be truncated for the windows at the end of the toolpath which do not require a complete window progression.

3.4 Linear programming (LP) optimization

The velocity, acceleration and jerk constrained problem can be converted into a linear problem with some simplifications. Linear problem formulation methods presented by Zhang et al. and Fan et al. [15], [16] were modified into a windowed solution by Erkorkmaz et al. [17]. The method applied in [17] is used in the first step of the optimization described in this document. Building on the earlier works [15], [16], the jerk constraint is replaced with “pseudo-jerk”, a conservative estimate of the jerk which ensures the actual jerk constraint is not violated, albeit with some loss of optimality.

The optimization problem is stated in Eq. (3-17), where L is the total length of the toolpath and $q = \dot{s}^2$ is the square of the tangential velocity. The expressions for axis level velocity, acceleration, and jerk are given in Eqs. (3-18) to (3-20). The x-axis equations are shown as an example – but the y- and z-axis equations are identical in structure. In these equations, x_s represents the geometric derivative of x, ($x_s = dx/ds$).

$$\max \int_0^L q\,ds \quad s.t. \quad \begin{bmatrix} |\dot{x}| \\ |\dot{y}| \\ |\dot{z}| \end{bmatrix} \le \begin{bmatrix} v_{x,\max} \\ v_{y,\max} \\ v_{z,\max} \end{bmatrix}, \qquad \begin{bmatrix} |\ddot{x}| \\ |\ddot{y}| \\ |\ddot{z}| \end{bmatrix} \le \begin{bmatrix} a_{x,\max} \\ a_{y,\max} \\ a_{z,\max} \end{bmatrix}, \qquad \begin{bmatrix} |\dddot{x}| \\ |\dddot{y}| \\ |\dddot{z}| \end{bmatrix} \le \begin{bmatrix} j_{x,\max} \\ j_{y,\max} \\ j_{z,\max} \end{bmatrix} \tag{3-17}$$

Most available optimization algorithms in software are set up to solve minimization problems. To use these algorithms directly, the maximization of $\int q$ was instead expressed as a minimization of $-\int q$.

The velocity, acceleration and jerk in each axis are computed from the prespecified toolpath $x = x(s)$ with known geometric derivatives (x_s, x_{ss}, x_{sss} y_s, ...) using Eqs. (3-18) to (3-20).

$$\dot{x} = \frac{dx}{dt} = x_s\dot{s} \tag{3-18}$$

$$\ddot{x} = \frac{d\dot{x}}{dt} = x_s\ddot{s} + x_{ss}\dot{s}^2 \tag{3-19}$$

$$\dddot{x} = \frac{d\ddot{x}}{dt} = x_s\dddot{s} + 3x_{ss}\dot{s}\ddot{s} + x_{sss}\dot{s}^3 \tag{3-20}$$

If the geometric derivatives are not available in closed form, they can be computed numerically (by solving values of ‘s’) and applying the chain rule for differentiation as needed. Substituted into the constraints, they can be written as:

$$|\dot{x}| \le v_{x,\max} \Leftrightarrow -v_{x,\max} \le \dot{x} \text{ and } \dot{x} \le v_{x,\max} \tag{3-21}$$

$$|\ddot{x}| \le a_{x,\max} \Leftrightarrow -a_{x,\max} \le \ddot{x} \text{ and } \ddot{x} \le a_{x,\max} \tag{3-22}$$

$$|\dddot{x}| \le j_{x,\max} \Leftrightarrow -j_{x,\max} \le \dddot{x} \text{ and } \dddot{x} \le j_{x,\max} \tag{3-23}$$

The constraints are reformulated by introducing a variable transformation parameter, $q = \dot{s}^2$ (which is used in Eq. (3-17)) which is equal to the square of the feed rate, $\dot{s}$.

$$q = \dot{s}^2 \tag{3-24}$$

After this transformation, the velocity, acceleration, and jerk equations can be represented as in Eqs. (3-25)-(3-27). Once this parameter is introduced, the axial velocity, acceleration, and jerk equations, can be represented as linear constraints.

$$|\dot{x}| = \left|\frac{dx}{dt}\right| = \left|x_s\sqrt{q}\right| \le v_{x,\max} \tag{3-25}$$

$$|\ddot{x}| = \left|\frac{d\dot{x}}{dt}\right| = \left|x_{ss}q + \frac{1}{2}x_s q_s\right| \le a_{x,\max} \tag{3-26}$$

$$|\dddot{x}| = \left|\frac{d\ddot{x}}{dt}\right| = \left|\left(x_{sss}q + \frac{3}{2}x_{ss}q_s + \frac{1}{2}x_s q_{ss}\right)\sqrt{q}\right| \leq j_{x,\max} \tag{3-27}$$

The axis velocity constraint (3-25) is put in linear form by squaring both sides:

$$x_s^2 q \leq v_{x,\max}^2 \tag{3-28}$$

As earlier indicated, the derivative terms (q_s, q_{ss}) are linear with respect to the profile q, because q is parameterized using 3rd order (quadratic) basis function splines. Hence, the function and derivative calculations follow the identical structure earlier shown in Eqs. (3-9)-(3-11), based on the control points c_i.

$$q(s) = \sum_{i=1}^{m} c_i b_i(s) \tag{3-29}$$

$$q_s(s) = \sum_{i=1}^{m} c_i [b_i]_s(s) \tag{3-30}$$

$$q_{ss}(s) = \sum_{i=1}^{m} c_i [b_i]_{ss}(s) \tag{3-31}$$

As x_s and x_{ss} are precomputed constant arrays, determined by the toolpath geometry, the acceleration constraint in Eq. (3-26) is already linear with respect to c_i.

To satisfy the jerk constraint, the $\sqrt{q}$ term is replaced with an array q^* serving as its upper bound. q^* is the unique solution for only the velocity- and acceleration-constrained problem (and is considered as an array of constants in the context of solving q in the LP optimization). Then, defining the pseudo-jerk as:

$$\tilde{\dddot{x}} = \left(x_{sss}q + \frac{3}{2}x_{ss}q_s + \frac{1}{2}x_s q_{ss}\right)\sqrt{q^*} \tag{3-32}$$

it follows that since $q \leq q^*$, the pseudo-jerk will be an upper bound on the actual axis-level jerk.

$$|\dddot{x}| \leq |\tilde{\dddot{x}}| \quad \Leftrightarrow \quad \left|\left(x_{sss}q + \tfrac{3}{2}x_{ss}q_s + \tfrac{1}{2}x_s q_{ss}\right)\sqrt{q}\right| \leq \left|\left(x_{sss}q + \tfrac{3}{2}x_{ss}q_s + \tfrac{1}{2}x_s q_{ss}\right)\sqrt{q^*}\right| \tag{3-33}$$

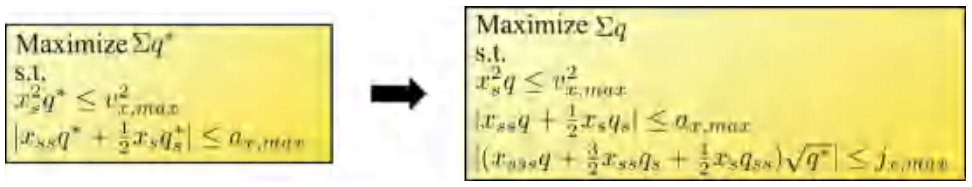
Hence, bounding the pseudo-jerk magnitude by $j_{x,\max}$ will guarantee that the actual jerk is bounded by the same limit (i.e., $|\dddot{x}| \leq |\tilde{\dddot{x}}| \leq j_{x,\max}$).

In this case, limiting the pseudo jerk takes the following form:

$$\left|\left(x_{sss}q + \frac{3}{2}x_{ss}q_s + \frac{1}{2}x_s q_{ss}\right)\sqrt{q^*}\right| \leq j_{x,\max} \tag{3-34}$$

Since the terms x_s, x_{ss}, x_{sss} and q^* are all computed ahead of time, the pseudo-jerk constraint is linear with respect to q.

Now, the velocity, acceleration, and pseudo-jerk constraints can be considered jointly in context of the LP problem, as shown schematically Figure 3-7. Of course, there is some loss of optimality in resulting motion time, compared to using the actual jerk constraint. However, the loss of optimality was benchmarked to be around only 5%, which can be remedied by NLP to account for exact jerk. The main advantage of LP and linear problem formulation is the solution of a near-optimal feed profile at around ten times the speed of applying full-fledged NLP solution, thus obtaining a good initial guess for the proceeding NLP steps.



Maximize Σq^*
s.t.
$x_s^2 q^* \le v_{x,max}^2$
$|x_{ss} q^* + \frac{1}{2} x_s q_s^*| \le a_{x,max}$

$\rightarrow$

Maximize Σq
s.t.
$x_s^2 q \le v_{x,max}^2$
$|x_{ss} q + \frac{1}{2} x_s q_s| \le a_{x,max}$
$|(x_{sss} q + \frac{3}{2} x_{ss} q_s + \frac{1}{2} x_s q_{ss})\sqrt{q^*}| \le j_{x,max}$

Figure 3-7: LP optimization procedure.

3.4.1 Selection of kinematic V/A/J limits for compatibility with the proceeding NLP step

The velocity, acceleration, and jerk limits used in the LP step must accurately reflect the expected values of motor torque and tracking error to ensure that these latter limits are not violated. In the proposed methodology, the velocity, acceleration, and jerk limits are selected using the motor torque and tracking error equations, described in further detail in Section 3.6.1.

$$u_{x,\max} = m_x a_{x,\max} + b_x v_{x,\max} + c_x + d_{coul,x} * \operatorname{sign}(v_{x,\max}) \tag{3-35}$$

$$e_{x,\max} = K_{v,x} v_{x,\max} + K_{a,x} a_{x,\max} + K_{j,x} j_{x,\max} \tag{3-36}$$

The constraints in Eqs. (3-35) and (3-36) are expressed for the x-axis, and identical structures also apply to other axes. With known values of u_{max} and e_{max} for each axis, solving Eqs. (3-35) and (3-36) for v_{max}, a_{max}, and j_{max} would yield multiple solutions. It may be desirable to identify a specific value of v_{max} first and then compute the corresponding a_{max} and j_{max}. A sample calculation is given below, using the parameters from Table 3-4. In the sample case, the desired torque and tracking error are u_{max} = 10V and e_{max} = 0.02 mm. (Note that torque here is presented as a percentage of the maximum control signal). Additionally, it is assumed that a maximum velocity of v_{max} = 150 mm/s is specified.

In the Y axis, using Eq. (3-35) and applying the triangle inequality ($|A + B| \le |A| + |B|$), the maximum acceleration is computed as $a_{\max}$ = 3.34×10^3 mm/s^2. This value is then combined with Eq. (3-36) to compute the maximum jerk $j_{\max}$ = 1.56×10^5 mm/s^3.

In the X axis, using Eq. (3-35), a maximum acceleration of $a_{\max} = 1.23 \times 10^3$ mm/s^2 is found. However, when combining this with Eq. (3-36), this acceleration value is found to be invalid, as the acceleration contribution to tracking error is found to be $K_{a,x} a_{x,\max} = 0.0245$, which exceeds the assigned tracking error limit. The acceleration constraint in the X axis is therefore scaled down to $a_{\max} = 500$ mm/s^2, and the computed jerk constraint is found to be 6.36×10^4 mm/s^3.

3.5 Reparameterization of the LP solution to a B-spline function directly representing feedrate

The LP solution is a B-spline function defining the feedrate square $q(s) = \dot{s}^2$ with respect to the path parameter s. However, the NLP problem needs to be cast in terms of feedrate, in order to be able to compute the nonlinear constraints. Thus, the LP solution needs to be re-parameterized in terms of B-spline functions that directly represent the progression as $\dot{s}$. To achieve this, first, the $\dot{s}$ values are computed at the constraint checkpoints by taking the square root of the $q(s)$ profile. Then, a new B-spline is fitted using a Least Squares formulation parameter estimation [63]. A sample result of this fit is shown in Figure 3-8.

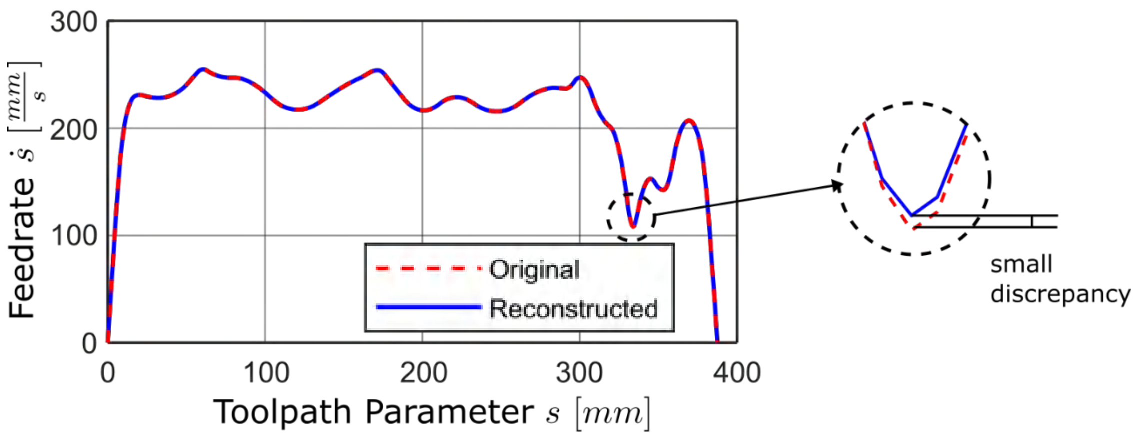


Figure 3-8: Results of B-spline reconstruction of feedrate profile from feedrate squared profile.

The B-spline representation of $\dot{s}$ will always have some discrepancy with the original points, although it is usually small. In Figure 3-8 the discrepancy is not large enough to be visible (maximum fitting error of 0.6 mm/s, typically less than 1% of the function value). After the reconstruction, the new B-spline representation is verified, to make sure it does not violate any constraints. If there is constraint violation, the feedrate control points are scaled down until all constraints are satisfied. Similarly, non-negativity of the new control points c_i is also checked, to ensure that the feed profile always progresses forwards.

3.6 Nonlinear programming (NLP) optimization

The NLP method offers a more optimal solution than the LP in terms of motion time, but it is more computationally intensive, and may require a near-optimal starting guess. The LP solution efficiently generates such a near-optimal starting guess. The selection of the optimization algorithm for NLP is also important, and is discussed in this section.

The main advantage of NLP is its ability to handle nonlinear constraints. One application can be to refine the solution of the axis-level velocity, acceleration, and pseudo-jerk constrained problem to considering the actual jerk instead of pseudo-jerk, which was observed, in benchmarks conducted, to produce further 4-5% motion time reduction, which is a moderate gain. More importantly, replacing the velocity, acceleration, and (pseudo-)jerk constraints with direct servo error and motor torque constraints, which are also nonlinear but technologically more relevant, can result in even more significant decrease in motion time, as demonstrated in the subsequent sections.

Hence, NLP problem considered in this thesis can be formulated as:

$$\max \int_0^L \dot{s}\, ds \quad s.t. \quad \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \le \begin{bmatrix} u_{x,\max} \\ u_{y,\max} \\ u_{z,\max} \end{bmatrix}, \qquad \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \le \begin{bmatrix} e_{x,\max} \\ e_{y,\max} \\ e_{z,\max} \end{bmatrix} \tag{3-37}$$

And optionally,

$$\begin{bmatrix} |\dot{x}| \\ |\dot{y}| \\ |\dot{z}| \end{bmatrix} \le \begin{bmatrix} v_{x,\max} \\ v_{y,\max} \\ v_{z,\max} \end{bmatrix}, \qquad \begin{bmatrix} |\ddot{x}| \\ |\ddot{y}| \\ |\ddot{z}| \end{bmatrix} \le \begin{bmatrix} a_{x,\max} \\ a_{y,\max} \\ a_{z,\max} \end{bmatrix}, \qquad \begin{bmatrix} |\dddot{x}| \\ |\dddot{y}| \\ |\dddot{z}| \end{bmatrix} \le \begin{bmatrix} j_{x,\max} \\ j_{y,\max} \\ j_{z,\max} \end{bmatrix}, \qquad \dot{s}_{min} \le \dot{s} \le \dot{s}_{\max}$$

Naturally, when velocity, acceleration, and jerk constraints are also used, the resulting solution can be more conservative in terms of motion time, as expected from the triangle inequality ($|A + B| \le |A| + |B|$).

In some cases, to ensure the forward traversal of the trajectory (i.e. to avoid unnecessary stops or backward movements due to numerical errors), a lower bound can be imposed on the feedrate as $0 < \dot{s}_{min} \le \dot{s}$, where $\dot{s}_{min}$ is a very small positive feed which would not cause any constraint violation. Similarly, to limit the magnitude of cutting forces, motor power and torque, and tool deflections during machining, an upper bound can also be imposed on the feedrate, as $\dot{s} \le \dot{s}_{\max}$.

In implementation, optimization algorithms tend to take the form of minimization problems. Modifying the main statement of the problem, as in Eq. (3-38), allows the use commercially available optimization algorithms.

$$\min \left[- \int_0^L \dot{s}\, ds \right] \tag{3-38}$$

Once again, the geometric derivatives used in the constraints determined by the toolpath. The derivatives of feedrate with respect to the arc parameter are computed per Eqs. (3-10)-(3-13). The time derivatives are computed using Eqs. (3-18)-(3-20). Finally, the predicted control signal and tracking error are computed according to Eqs. (3-39) and (3-40), which are described in Section 3.6.1.

3.6.1 Control signal and tracking error models

The tracking error and motor torque provide more direct indicators of a CNC machine tools motion performance (and manufacturing accuracy) and utilization compared using the classical kinematic limits of velocity, acceleration, and jerk. The NLP optimization problem stated in Eq. (3-37) includes direct constraints on motor torque (or voltage command sent to the motor amplifiers, u) and the servo tracking error (e). Like kinematic constraints, these constraints must be calculated at point along the toolpath and sufficiently close intervals. A given trajectory will typically result in different torque commands and tracking errors on different machines. Thus, parameter identification is required and performed to quantify the relationships between the commanded trajectory profile, and the resulting motor torque and servo accuracy responses. This is typically achieved by executing a movement on the target machine, and recording the velocity, acceleration, jerk, motor torque command, and tracking error for each axis, and applying suitable model identification techniques.

The motor torque demand can be represented by approximating the CNC drives as an inertia element subject to viscous and Coulomb friction, as shown by Eq. (3-39). This open-loop model neglects the transient dynamics contributed by the feedback servo controller. However, these dynamics (except for stick-slip friction) are typically not excited when smooth trajectories are used. In the equation, mass (m_x), viscous friction (b_x), torque offset (c_x) (due to preload or balancing), and Coulomb friction ($d_{coul,x}$) can be identified using Least Squares (LS) estimation [63] on the experimental data collected during trajectory tracking. Eq. (3-39) applies separately to each axis, assuming that they are inertially decoupled.

$$u_x = m_x\ddot{x} + b_x\dot{x} + c_x + d_{coul,x}sign(\dot{x}) \tag{3-39}$$

The tracking error can be approximated as a linear combination of the instantaneous velocity, acceleration, and jerk commands ($\dot{x}_r, \ddot{x}_r, \dddot{x}_r$) as shown by Eq. (3-40), by assuming that the feed drive dynamics are predominantly rigid and do not contain poorly damped (i.e., oscillatory) poles that which significantly influence the servo response. Here, K_j, K_a, and K_v are model parameters that can be identified using the LS approach. e_x is recorded tracking error from the CNC machine.

$$e_x = K_{j,x}\dddot{x}_r + K_{a,x}\ddot{x}_r + K_{v,x}\dot{x}_r \tag{3-40}$$

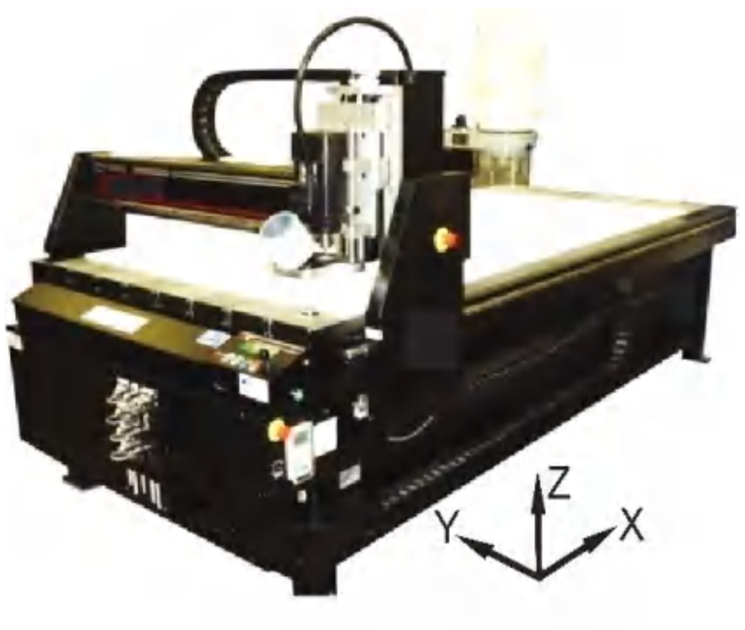


Figure 3-9: Router machine tool used in the experiments.

The testbed machine, a 3-axis router as seen in Figure 3-9, was used to record torque and tracking error profiles during a short movement. The parameters for the models in Eqs. (3-39) and (3-40) were then estimated, and validated by reconstructing the motor torque and tracking error signals. The result is shown in Figure 3-10.

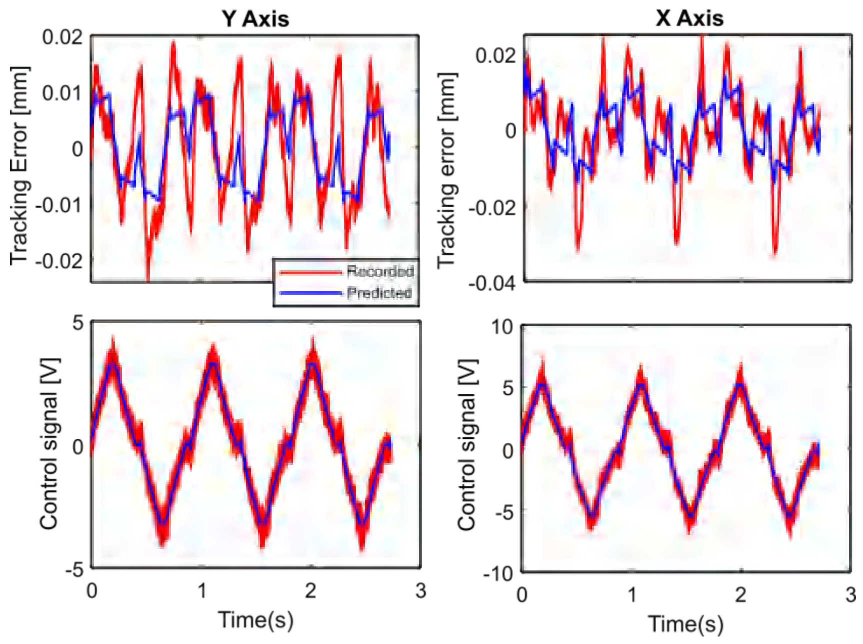


Figure 3-10: Results of a parameter identification routine.

The control signal predictions are in very good agreement. However, the prediction of tracking error, which is much more sensitive to minute effects and nonlinearities, is limited in accuracy by the nature of the experimental setup. The setup used is a low-cost flatbed router with large amounts of friction (in the guideways and ballscrew–nut interface), backlash (i.e., motion loss), lead errors (position-dependent harmonic transmission error) and flexibilities (originating from the mechanical frame and assembly and also the drive system (including the belt-driven connections between the motors and ballscrew-nuts mechanisms). These all contribute to imperfections and unmodeled effects in the motion delivery, which become more significant at the micron-level and lead to observed deviations from the predictions of Eq. (3-40). For example, several of the large peaks in the tracking error plots in Figure 3-10 correspond to zero-velocity transitions in the testing trajectory, at which stick-slip friction (currently not considered in the model) becomes significant. However, in high-end CNC feed drives, which have much less friction and are constructed with much more rigid components and connections, it has been shown that the quasi-static tracking error model in Eq. (3-40) actually yields fairly accurate predictions [39], [64]. Therefore, in spite of the mechanical disadvantages of the experimental setup that was used, the servo error model in Eq. (3-40) has been adopted and used in this thesis.

Table 3-1 and Table 3-2 show the identified parameters used in simulation and experimental studies, respectively.

Table 3-1: Parameters used in simulation case studies.

	X-axis	Y-axis
M	2.13×10^{-3}	1.25×10^{-3}
b	4.67×10^{-2}	3.80×10^{-2}
c	1.48×10^{-1}	-1.19×10^{-3}
d_{coul}	2.41×10^{-1}	1.31×10^{-1}
K_v	-4.49×10^{-5}	8.07×10^{-5}
K_a	1.99×10^{-5}	1.52×10^{-6}
K_j	-5.20×10^{-8}	-1.80×10^{-8}

Table 3-2: Identified parameters of router machine used in experiment.

	X-axis	Y-axis
M	1.94×10^{-3}	1.15×10^{-3}
b	5.06×10^{-2}	3.28×10^{-2}
c	-1.34×10^{-1}	1.32×10^{-1}
d_{coul}	4.96×10^{-1}	1.84×10^{-1}
K_v	5.34×10^{-5}	2.93×10^{-5}
K_a	1.45×10^{-5}	1.45×10^{-5}
K_j	7.17×10^{-7}	2.39×10^{-7}

3.6.2 NLP Method I - Sequential quadratic programming (SQP)

The basic SQP algorithm is described in Section 2.3.2.2. The iterates of this algorithm are computed by first solving for the Hessian (second derivative) matrix of the Lagrangian of the problem, the latter which is a modified objective function that integrates equality constraints. Using the Hessian, the quadratic approximation of the objective function is created. Then, the quadratic program is solved. This solver is able to deal with infeasible iterates – if the solution of the QP is infeasible according to the constraints, the solver will go back and try again with a smaller step size. However, in experimentation, it was found that infeasible starting conditions will cause the solution time to increase significantly – up to 5 times in comparison to starting with a feasible guess.

This algorithm computes full matrices of the problem. In the best of the author's knowledge, it does not make adjustments that take advantage of sparse matrices. Implementation of the windowing solution is therefore essential to the use of this algorithm, and the solution becomes very slow when trying to solve a long toolpath. However, the efficiency of SQP can be increased by providing a near-optimal starting guess, which the LP solution is able to generate.

3.6.3 NLP Method II - Interior point algorithm (IP)

In some cases, the SQP algorithm was prone to failure – for example, if a non-feasible initial guess was arrived at numerical errors passed on from previous computational steps. There were also instances with long toolpaths when the SQP, in an inexplicable manner, failed to converge. To overcome this drawback, the interior point algorithm was also implemented for feed optimization.

The concept of interior point algorithms is described in Section 2.3.2.2. MATLAB's interior point algorithm implements logarithmic barrier functions and alternates between iterative steps to solve this function, and the modification of the barrier function parameter. While this algorithm is separate from the SQP implementation, it also internally uses SQP in some of its solution steps.

Unlike in the SQP algorithm, this problem operates using sparse approximations and sparse linear algebra – this allows for the capacity to solve problems using large, sparse matrices. This would be a significant advantage if a single one-shot solution to the NLP problem was desired, but is not needed in the windowed solution presented in this thesis. In the developed algorithm, IP has been implemented as a secondary slower but more robust solution, in the rare case the SQP algorithm fails to successfully converge. The user has to manually select which NLP method is to be used.

3.7 Coordination of boundary conditions and additional numerical considerations

The progression parameters which must be set in the overall optimization algorithm are as follows:

Table 3-3: Progression parameters for the dual windowing feed optimization algorithm.

	LP specific	NLP specific
Total toolpath length	L (distance) with r control points	
Window size	W_{LP}	W_{NLP}
Overlap size	P_{LP}	P_{NLP}
Constraint meshing factor i.e. number of constraint evaluation points per control point	N_c	

Considering Figure 3-1, the window sizes, W_{LP} and W_{NLP}, are assigned based on certain factors: 1) An excessively small window will be computationally inefficient and may not reach the actual achievable optimal feedrate. 2) An excessively large window may cause the optimization to fail, or exceed the available computational memory (which varies depending on the platform used to compute the optimization).

Note that most of the parameters in Table 3-3 are referred to in terms of distance along the toolpath. However, in the implementation, these parameters are represented in terms of integer numbers of control points. Thus, all values of L, W, and P must be integer multiples of the knot spacing distance d.

The overlap sizes, P_{LP} and P_{NLP}, have similar considerations. If the overlap is too small, the feedrate will be suboptimal at the connection points between windows. This is because boundary condition influences causing the feedrate to drop to zero or small values will be inevitable. If the overlap is too large, the computation time will be excessive, as long and overlapping portions of the trajectory will be unnecessarily reoptimized. The minimum overlap is computed based on the worst-case deceleration distance of the machine, based on the maximum kinematic parameters. The deceleration distance for a machine axis from its maximum velocity is given in Eq. (3-41), which is derived in detail in Appendix A.2.

$$l_{dec} = l_{acc} = \begin{cases} \dfrac{v_{\max}^{\frac{3}{2}}}{\sqrt{j_{\max}}} & if\ v_{\max} \le \dfrac{a_{\max}^2}{j_{\max}} \\ \dfrac{1}{2} v_{\max}\left(\dfrac{a_{\max}}{j_{\max}} + \dfrac{v_{\max}}{a_{\max}}\right) & if\ v_{\max} > \dfrac{a_{\max}^2}{j_{\max}} \end{cases} \tag{3-41}$$

The minimum deceleration distance is computed separately for each axis. Then, the largest one is used to determine the minimum window overlap distance. From this, a minimum window size can be expressed in control points as p using Eq. (3-42) where r is the number of control points in the trajectory, L is the total toolpath length, and α_{overlap} is an applied safety factor (for instance, $\alpha_{\text{overlap}} = 2$ will enforce that the overlap must be twice as long as the expected deceleration distance).

$$p = \alpha_{\text{overlap}}\, l_{acc} \frac{r}{L} \tag{3-42}$$

The LP overlap, P_{LP}, requires additional consideration, as shown in Figure 3-2. This overlap occurs at the boundary between NLP-optimized and LP-optimized-only solutions, which have different constraints. Once the nonlinear optimization has been performed based on servo error and motor torque constraints, the solution will likely no longer be a feasible solution to the LP problem which considers velocity, acceleration, and pseudo-jerk constraints. With infeasible initial boundary conditions, the consecutive LP optimization step would fail. The use of a B-spline feed profile ensures that the boundary conditions remain feasible for the LP at the beginning of the next LP step, as shown in Figure 3-11.

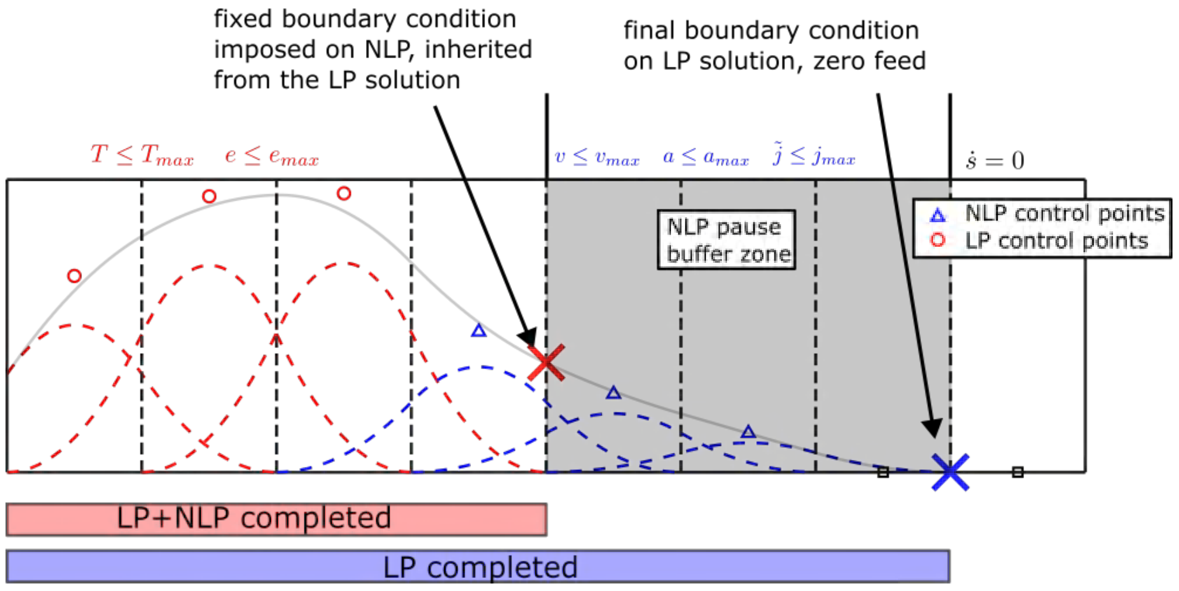


Figure 3-11: Boundary condition enforcement at window boundaries in LP+NLP windowing optimization.

The minimum size of the pause boundary is related specifically to the order of the B-spline feedrate profile. In a B-spline function of order n, any individual B-spline will have a range of $[\xi_k, \xi_{k+n}]$. Using Eqs.

(3-6),(3-12), and (3-13), it can be seen that the effect of any quadratic B-spline will be limited to the space between four knots – hence, the length of the NLP pause buffer range would have to be at least $3d$.

3.8 Time-domain reconstruction

The solution obtained from the dual-windowing algorithm is a feedrate profile expressed in terms of arc length, $\dot{s}(s)$, as well as functions that map a given value of s to its corresponding x, y, z coordinates. This must be converted to arc displacement ($s(t)$) and feedrate expressed in terms of time ($\dot{s}(t)$), in order to be interpolated and sent as a discrete-time motion command to the servo controllers, ultimately as $x(t)$, $y(t)$, and $z(t)$.

Eq. (3-43) defines the relationship between t and s, which depends on the feedrate $\dot{s}$. In this work, it has been solved using third-order Taylor series expansion.

$$t_i = \int_{s_{i-1}}^{s_i} \frac{1}{\dot{s}} ds \tag{3-43}$$

Third-order Taylor series expansion was selected because the second and third derivatives of s have already been computed during the optimization.

The process of time reconstruction is described as follows:

1. An array of desired time values is created. In this case, a set of points with a constant sampling period T_s are used, which is the most common form in sampled motion control systems.
2. The process will fail at any point when $\dot{s}$, $\ddot{s}$, and $\dddot{s}$ are all sufficiently close to zero. To avoid this, a minimum threshold for $\dot{s}$ is selected and any parts of the trajectory with feedrates below this threshold are marked as "breakpoints". As most trajectories start and end at zero velocity, the beginning and end of the toolpath are expected to be breakpoints.
3. If starting from a breakpoint: the first arc displacement will be zero, plus some small offset.
4. At this point, the following values are known: s_k (arc displacement at sample k), a B-spline profile defining the feedrate at any given position, and a desired time interval, T_s.
5. Values of the derivatives $\dot{s}(k)$, $\ddot{s}(k)$, $\dddot{s}(k)$ are computed at the sample k.
6. The time and arc displacement for sample $k+1$, $(t(k+1), s(k+1))$ are computed using Eqs. (3-44) and (3-45).

$$t(k+1) = t(k) + T_s \tag{3-44}$$

$$s(k+1) = s(k) + \dot{s}(k)T_s + \frac{1}{2}\ddot{s}(k)T_s^2 + \frac{1}{6}\dddot{s}(k)T_s^3 \tag{3-45}$$

7. This iteration continues until reaching either the end of the toolpath or a designated breakpoint.
8. If a breakpoint is reached, then the same "feedrate offset" as the beginning is applied before continuing as before.

Sample result for reconstruction of the arc displacement (and thus feed) profile in the time domain is shown in Figure 3-12. The optimization output $f(s)$ is converted to a set of sample points (t, s) at the desired time intervals, which would ultimately be sent as displacement commands to the parametric toolpath interpolation algorithms (i.e., $x = x(s(t))$, $y = y(s(t))$, $z = z(s(t))$. For the purpose of verification, in this sample an additional step was performed in which the feedrate profile was generated as a function of arc displacement, and these two (phase plane and time-domain reconstruction data) were overlaid for comparison on the bottom plot of Figure 3-12. As can be seen, there are only negligible differences between the two, which can be reduced further, if needed, if applying smaller time steps.

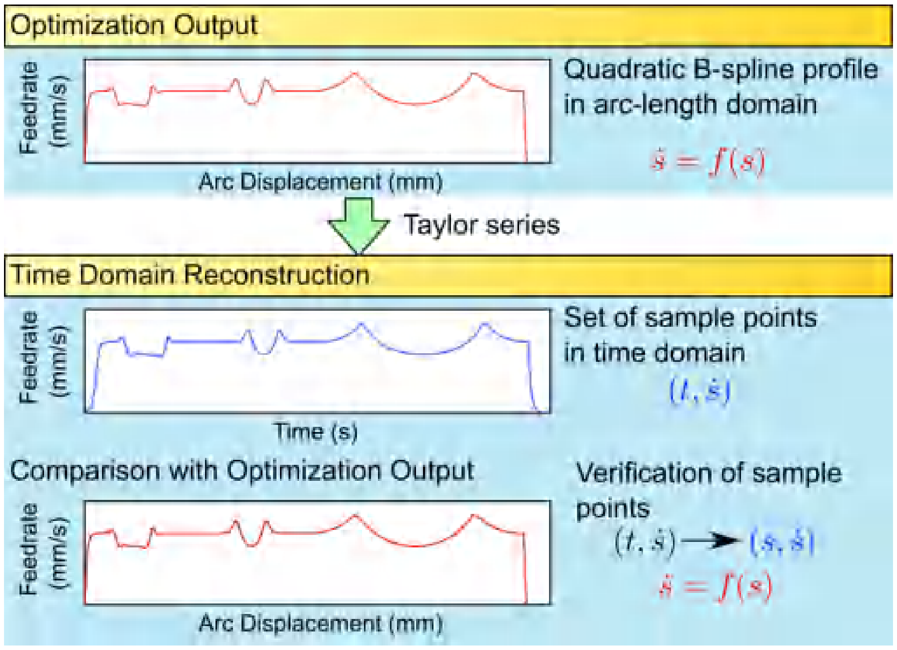

Optimization Output
Feedrate (mm/s)
Arc Displacement (mm)
Quadratic B-spline profile in arc-length domain
$\dot{s} = f(s)$
Taylor series
Time Domain Reconstruction
Feedrate (mm/s)
Time (s)
Set of sample points in time domain
$(t, \dot{s})$
Comparison with Optimization Output
Feedrate (mm/s)
Arc Displacement (mm)
Verification of sample points
$(t, \dot{s}) \longrightarrow (s, \dot{s})$
$\dot{s} = f(s)$

Figure 3-12: Verification of time-domain reconstruction of a arc displacement profile.

3.9 Simulation benchmarks

The results of applying the proposed dual-windowed feed optimization are shown in Figure 3-13. A hand-shaped sample toolpath is chosen due to its familiarity and variety of large and small curvatures. The feedrate is displayed by varying colour throughout the toolpath. Additionally, plots of the constrained motor torque and tracking error are provided below the toolpath. As can be seen, the constraints are satisfied throughout the toolpath. The optimization was performed using the parameters in Table 3-1. The acceleration and jerk limits were determined following the procedure in Section 3.4.1, to ensure that the LP solution would also be feasible for the NLP problem. The resulting constraint limits are shown in Table 3-4.

Table 3-4: Constraints applied in simulation case studies.

	X-axis	Y-axis
Maximum velocity [mm/s]	150	150
Acceleration [mm/s^2]	500	3340
Jerk [mm/s^3]	63000	156000
Control signal [V]	10	10
Tracking error [mm]	0.02	0.02

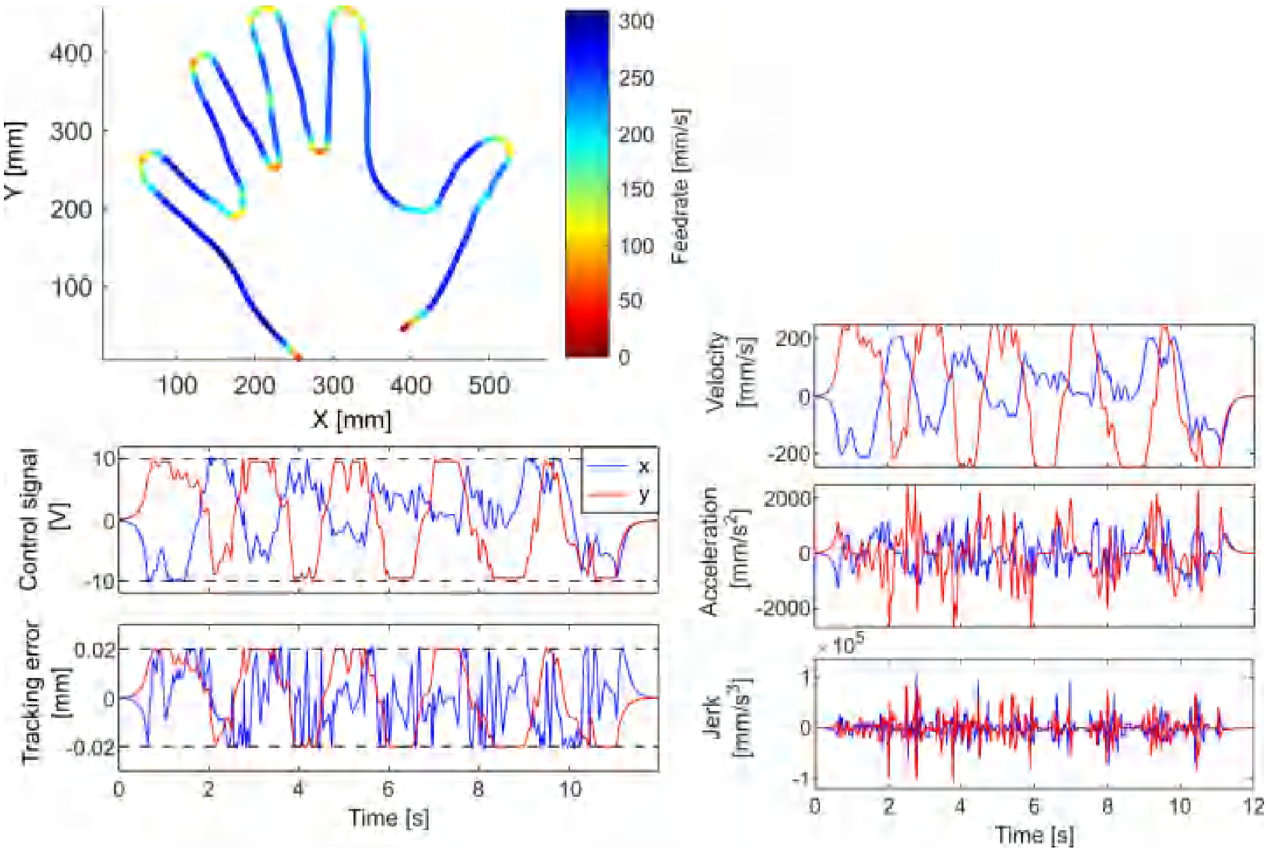


Figure 3-13: Results of an optimization procedure.

This case reveals several notable features of the optimization strategy:

- As predicted, the feedrate is faster during long, low-curvature segments and slows down during sections with tighter curvature.
- The optimization trajectory can sometimes reach a very high feedrate. It may be desirable to implement a feedrate limitation constraint as well (e.g., to keep machining forces limited). This is possible using both LP and NLP strategies, although it was not implemented in this example.

- The torque and tracking error are within the constraint envelope. Constraint violations due to numerical error in time reconstruction do occur, but the violation sizes are less than 1% of the assigned maximum and are not visible on the constraint profiles.

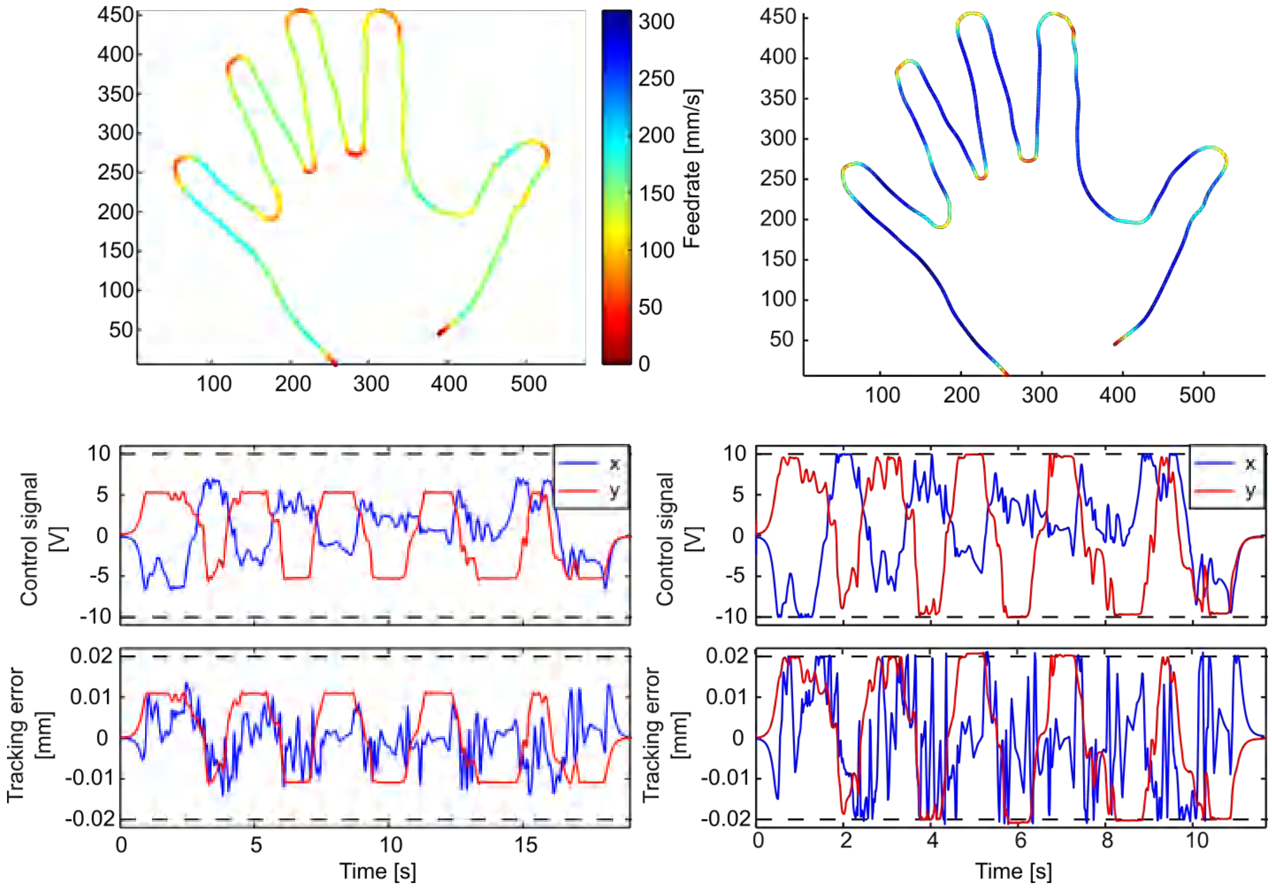


Figure 3-14: Comparison between LP (left) and LP+NLP (right) feed optimization optimizations.

A comparison between the LP-only algorithm and the LP+NLP algorithm was made and is presented in Figure 3-14, with computational time and movement time results presented in Table 3-5. In the LP-only case (left side of Figure 3-14), the axis velocity, acceleration, and jerk limits are used as an indirect method to constrain the control signal and servo error. In the LP+NLP implementation (right side of Figure 3-14), the control signal and servo error are constrained. It can be seen that this ultimately reduces the motion time by 29.7%, beyond the result obtained with only LP, without exceeding these prescribed limits. However, it is also important to acknowledge the increased computational time (around 3-fold), in exchange for the improved motion performance.

It can be seen that in the Y-axis, the profiles of control signal and tracking error closely resemble each other. This is caused by the weighting of the kinematic profiles in the identified parameters of the models for u and e, in which for the particular machine, both show strong correlation to the velocity commands.

Table 3-5: Computation time and movement cycle time for LP vs LP+NLP algorithms.

	LP only	LP+NLP
Computation time (s)	3.47	11.94
Movement time (s)	19.00	11.80

One of the strengths of the dual-windowing approach is that it enables the processing of indefinitely long toolpaths in a forward-moving (i.e., streaming) manner, similar to the architecture of an industrial CNC. Both types of optimization (LP and NLP) are performed along the toolpath in a carefully coordinated manner. An example of testing the proposed optimization on a longer toolpath is shown in Figure 3-15. This sample toolpath has a length of 18500 [mm], and was processed using 4000 control points and 20000 constraint evaluation points. Noting that on a 3-axis machine, LP+NLP requires 27 inequality constraints to hold at each evaluation point (i.e., (1 velocity, 2 acceleration, 2 pseudo-jerk, and 2 control signal and 2 tracking error) x (2 axes)), this translates to 540,000 inequality constraints.

Due to memory limitations on the optimization computer, the computation time was long, but the optimization was overall successful. Small constraint violations due to numerical error in time reconstruction did occur occasionally, as can be seen in the constraint plots of Figure 3-15, but outside of these errors, the constraints remain within their prescribed limits. Trajectories of this length can be very computationally expensive. The applied velocity, acceleration, and jerk limits are equal to those in Table 3-4. The control signal and tracking error limits used were 10 V and 0.02 mm, respectively.

The computation and motion time are provided in Table 3-6.

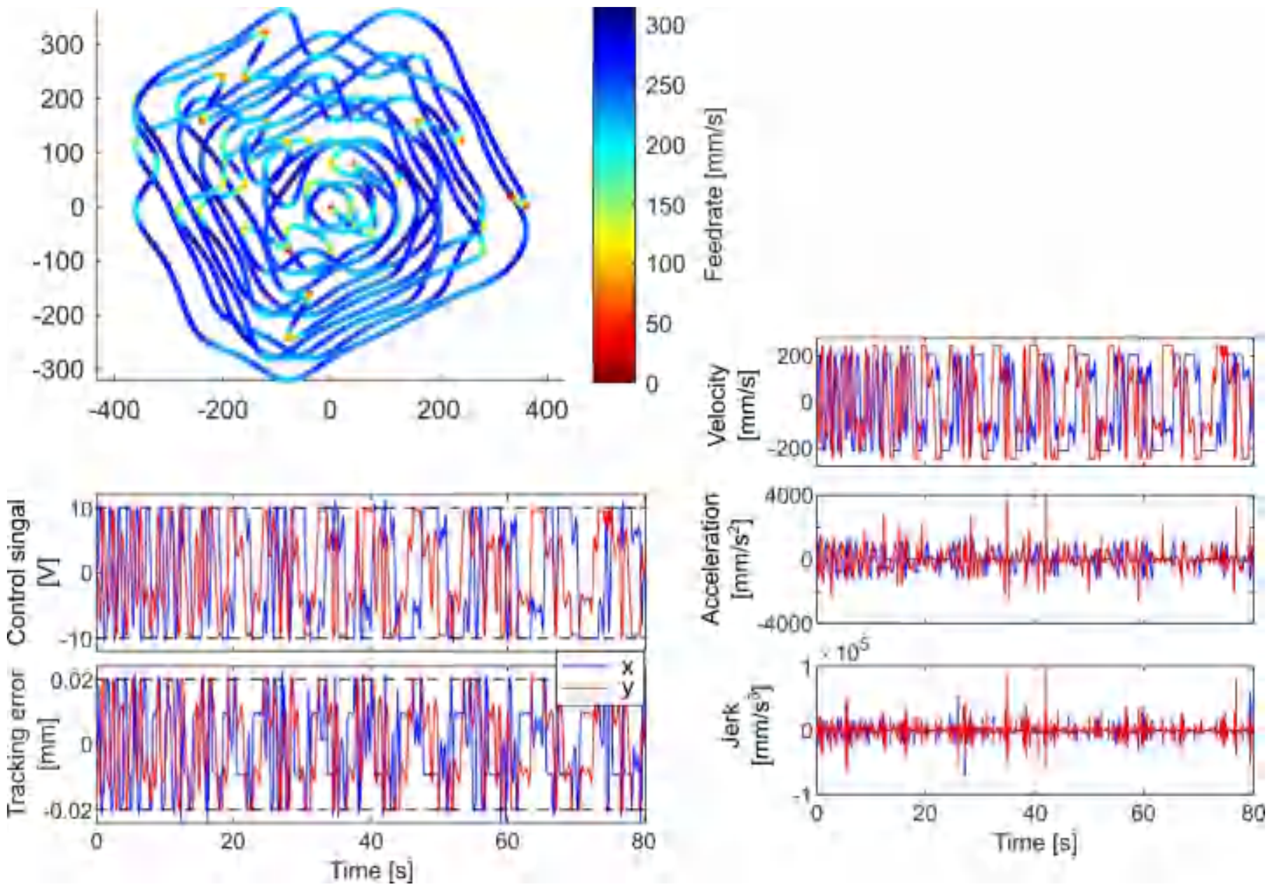


Figure 3-15: LP+NLP optimization of longer toolpath.

In comparison, the LP-only optimization of the same toolpath is presented below in Figure 3-16. The LP-only optimized trajectory is much longer in time, approximately twice the movement time of the LP+NLP optimization. Small, brief constraint violations continue to occur, caused by the time-domain reconstruction (discussed further in Section 4.2). With the exception of these, the tracking error and motor torque, again, stay well-below the limits due to the use of more conservative velocity, acceleration, and jerk constraints. It can be verified that the velocity, acceleration, and jerk limits were set to their maximum possible values, to not violate the control signal and servo error constraints. However, due to the conservativeness this brings (from the triangle inequality), the control signal (i.e., motor torque) and tracking error capacity are clearly underutilized, thus resulting in longer motion time.

`

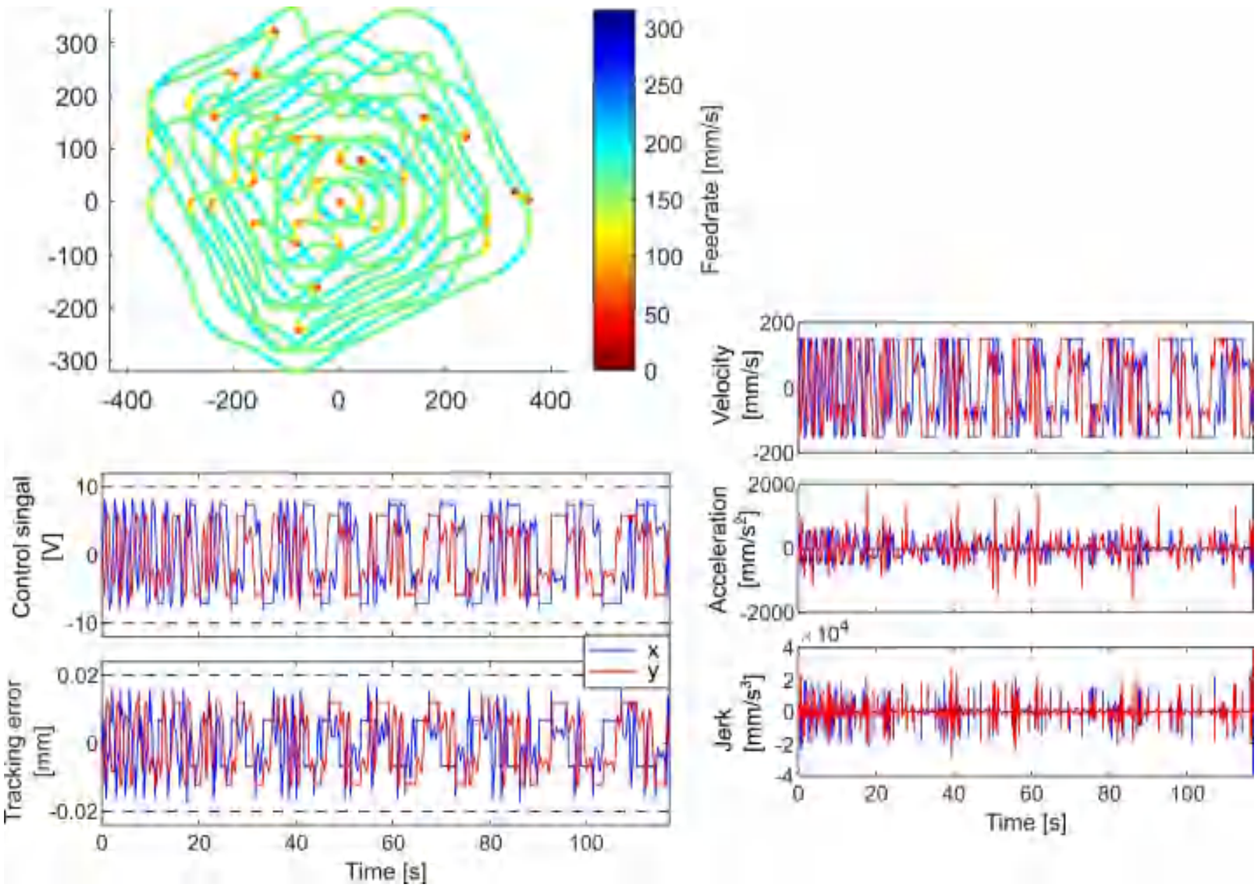


Figure 3-16: LP-only optimization of longer toolpath.

Table 3-6: Computation and motion time of very long toolpath.

	LP only	LP+NLP
Computation time (s)	37	353
Movement time (s)	117	80

This can be more clearly analyzed by considering the impact of limiting u_x and e_x indirectly by capping the magnitudes of velocity, acceleration, and jerk:

$$|u_x| \leq u_{x,\max} \Rightarrow \tag{3-46}$$

$$|u_x| = \left|m_x\ddot{x} + b_x\dot{x} + c_x + d_{coul,x}sign(\dot{x})\right| \leq m_x|\ddot{x}| + b_x|\dot{x}| + |c_x| + d_{coul,x} \leq$$
$$\leq m_x a_{x,\max} + b_x v_{x,\max} + |c_x| + d_{coul,x} \leq u_{x,\max} \tag{3-47}$$

$|e_x| \leq e_{x,\max} \Rightarrow$

$$|e_x| = \left|K_{j,x}\dddot{x}_r + K_{a,x}\ddot{x}_r + K_{v,x}\dot{x}_r\right| \leq \left|K_{j,x}\right| \cdot |\dddot{x}_r| + \left|K_{a,x}\right| \cdot |\ddot{x}_r| + \left|K_{v,x}\right| \cdot |\dot{x}_r|$$
$$\leq \left|K_{j,x}\right| \cdot j_{x,\max} + \left|K_{a,x}\right| \cdot a_{x,\max} + \left|K_{v,x}\right| \cdot v_{x,\max} \leq e_{x,\max}$$

When the numerical values are considered,

$$m_x a_{x,\max} + b_x v_{x,\max} + |c_x| + d_{coul,x} = u_{x,\max} \tag{3-48}$$

$$\underbrace{1.25 \times 10^{-3} \cdot 3.34 \times 10^{3}}_{4.17} + \underbrace{3.8 \times 10^{-2} \cdot 1.5 \times 10^{2}}_{5.7} + 1.19 \times 10^{-3} + 1.31 \times 10^{-1} = 10$$

$$\left|K_{j,x}\right| \cdot j_{x,\max} + \left|K_{a,x}\right| \cdot a_{x,\max} + \left|K_{v,x}\right| \cdot v_{x,\max} = e_{x,\max}$$

$$\underbrace{|-1.8 \times 10^{-8}| \cdot 1.56 \times 10^{5}}_{2.82 \times 10^{-3}} + \underbrace{|1.52 \times 10^{-6}| \cdot 3.34 \times 10^{3}}_{5.08 \times 10^{-3}} + \underbrace{|8.07 \times 10^{-5}| \cdot 1.5 \times 10^{2}}_{12.1 \times 10^{-3}} = 0.02$$

Considering Eqs. (3-46 and (3-47), while the terms $\left|m_x\ddot{x} + b_x\dot{x} + c_x + d_{coul,x}sign(\dot{x})\right|$ and $\left|K_{j,x}\dddot{x}_r + K_{a,x}\ddot{x}_r + K_{v,x}\dot{x}_r\right|$represent the actual magnitudes of control signal and servo error that must be limited, the latter expressions to the right-hand side of these terms, which are more conservative, assume the worst case scenarios. This is convenient for mathematical purposes, but not fully realistic. For example, jerk, acceleration, and velocity do not always have the same sign through a toolpath (which is what is assumed when computing their limits $j_{x,\max}$, $a_{x,\max}$, $v_{x,\max}$ used for LP). Therefore, there will be instances where larger velocity and acceleration magnitudes can be used, if they have cancelling signs in the torque equation (e.g., decelerating during positive velocity). Similarly, the values of $\dot{x}$, $\ddot{x}$, and $\dddot{x}$ are never simultaneously at their maximum limits, although $v_{x,\max}$, $a_{x,\max}$, $j_{x,\max}$ are computed considering this extreme situation. Thus, at many instances throughout the toolpath, the feed profile can be modulated so that the terms $\left|m_x\ddot{x} + b_x\dot{x} + c_x + d_{coul,x}sign(\dot{x})\right|$ and $\left|K_{j,x}\dddot{x}_r + K_{a,x}\ddot{x}_r + K_{v,x}\dot{x}_r\right|$ stay within their prescribed limits, while the individual profiles of $\dot{x}$, $\ddot{x}$, and $\dddot{x}$ may exceed $v_{x,\max}$, $a_{x,\max}$, and $j_{x,\max}$, respectively. This is the key reason that LP+NLP, which facilitates direct limiting of the nonlinear constraints for u and e is able to yield a faster trajectory and shorter motion time over LP.

3.10 Experimental results

Experimental validation was performed on the three-axis router as seen in Figure 3-9 using air-cut tests (in which the tool moves through the specified trajectory without any material being cut). Such feedrate optimization is particularly suited for operations with low cutting force, such as finish machining, in which the servo errors are induced primarily by the trajectory commands and not the lightweight cutting forces.

The machine tool is driven by two x-axis drives, one on each side of the machine, moving the gantry. The single y-axis drive which runs across the gantry. The drives are ball screw type (x-axes: rotating nut, y-axis: rotating screw), with synchronous belt mechanisms providing the linkage between the motors.
When obtaining experimental results, the x-axis tracking error and torque model were predicted based on one of the x-axis drives. The vertical z-axis was not fully functional at the time of conducting the experiments, and therefore was not used. The parameters identified and used in the experiment are presented in Table 3-7.

Table 3-7: Constraints considered in the experimental trajectory generation for the router.

Velocity [mm/s]	150
Acceleration [mm/s^2]	500
Jerk [mm/s^3]	10000
Tracking error [mm]	0.018
Control signal [V]	8

Figure 3-17 presents the experimental result for testing the hand-shaped toolpath shown in Figure 3-13. The limits constraint limits were modified to match the limits of the experimental setup.

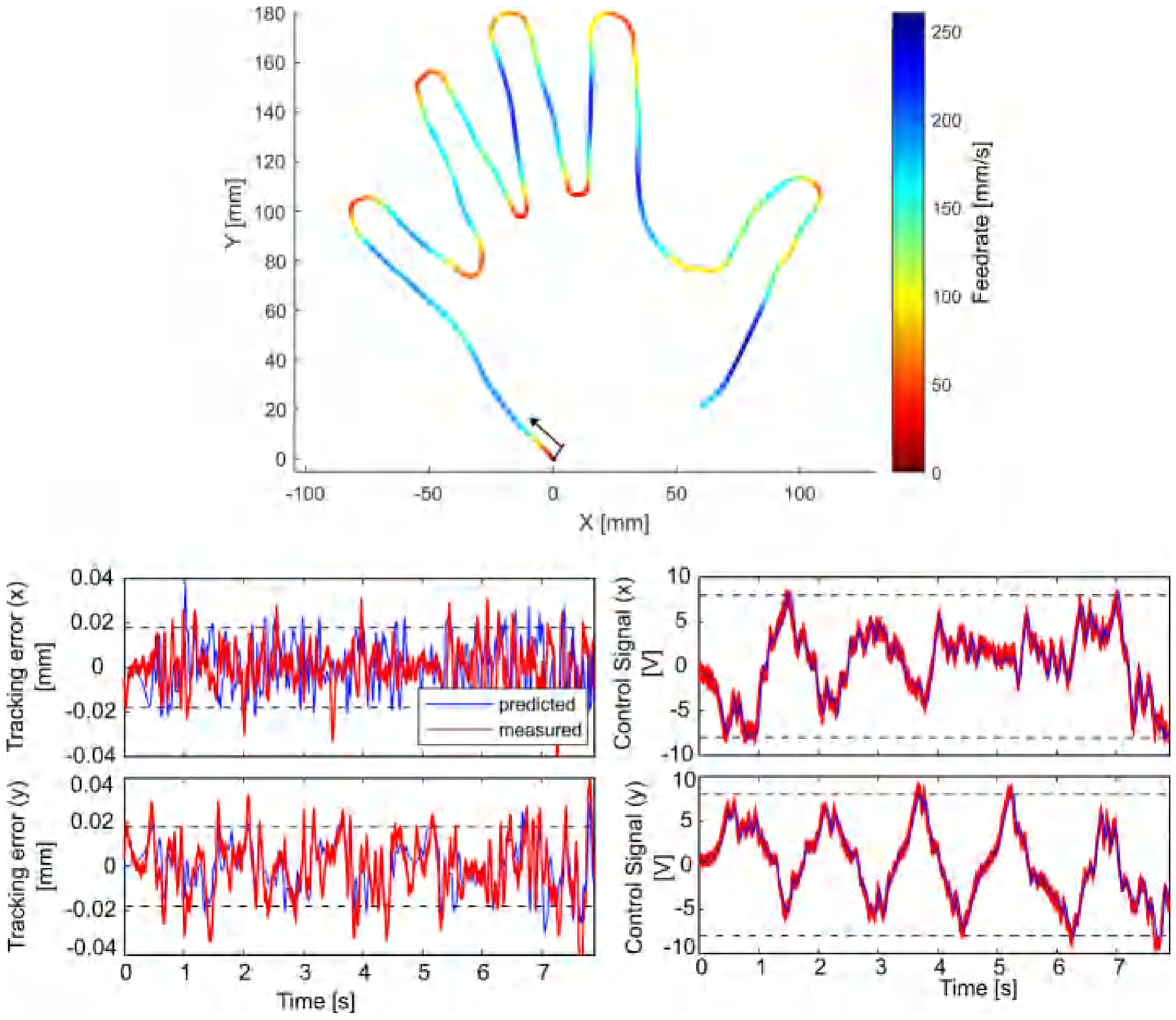


Figure 3-17: Experimental result of hand-shaped trajectory with LP+NLP dual-windowed optimization.

During the experiments, the tool was observed to travel smoothly along the assigned path, speeding up during long and low curvature portions and slowing down at higher curvature sections.

The control signal plot (bottom right of Figure 3-17) shows accurate experimental reproduction of the predicted motor control signal – the signal is successfully constrained to within the applied limit of 8 V (with very small violations due to the noise of the recorded signal).

The tracking error plot (bottom left of Figure 3-17) shows some discrepancy, due to the factors explained at the end of Section 3.6.1, originating from the mechanical imperfections of the motion transmission in the experimental setup, such as heavy and non-uniform stick-slip friction, backlash, lead errors, and structural vibrations. The actual and predicted tracking errors are shown in further detail in the zoomed in plot in Figure 3-18. The measured servo errors show noticeably higher peaks than the prediction, which in the experiments does cause some violation of the error constraints. Nevertheless, majority of the trends for the servo error are still captured, especially for the y-axis which is more rigid compared to the x-axis.

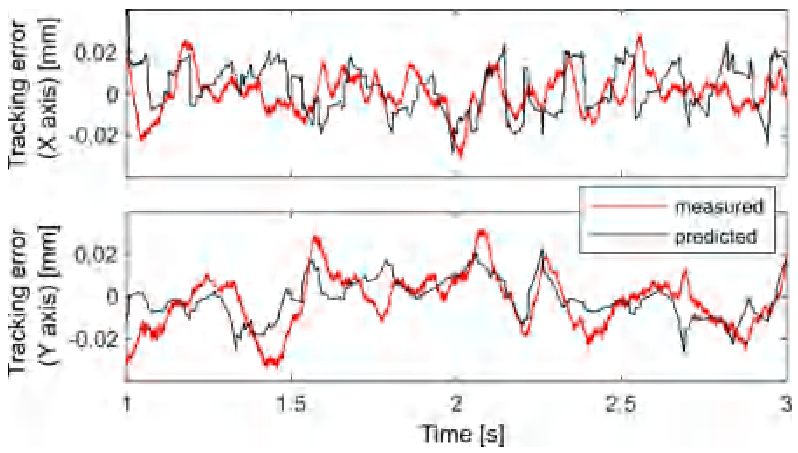


Figure 3-18: Predicted and experimental tracking errors from the router.

The tracking error model (described in Section 3.6.1) is based on a linear combination of the commanded velocity, acceleration, and jerk. It does not account for the contribution of static (stick-slip) friction and other nonlinearities, which make significant contributions to positioning errors in this case. However, the linear model works very well for high-end machine tool drives [39], [64], which have much lower friction influence compared to inertial forces, better motion transmission, and also are more rigid by nature. In future research, to improve the prediction accuracy for machines with large friction, an extended model can be developed which includes an analytical approximation of the anticipated servo error profile contributed by additional stick slip friction. However, in the context of validating the developed LP+NLP method, the experimental results from the router are promising, in terms of demonstrating the effectiveness of the proposed new trajectory generation method. Also, the prediction and limiting of motor torque is very successful.

3.11 Conclusions

In this chapter, a novel feedrate optimization method has been presented, which combines two different optimization methods into a single solution, applying two windowing methods in careful coordination with one another. The proposed LP+NLP windowing method uses the advantages of each type of optimization technique, the LP method's robustness and computational efficiency, and the increased optimality of the NLP solution. For each section of the toolpath, a near-optimal solution is generated using LP to solve the velocity, acceleration, and jerk constrained linear problem. Following this, the solutions is refined using NLP, with the constraints being swapped to limit the tracking error and commanded motor torque. The NLP optimization generates a faster trajectory, while keeping the performance of the motion (i.e., torque consumption and dynamic accuracy) within a specified envelope. The coordination of LP and NLP solution windows and evaluation of boundary conditions are designed specifically to avoid generating optimization problems with infeasible boundary conditions or initial solution.

The feedrate profile is represented using uniform quadratic B-splines, which allow for adjustment to the feedrate in a small local area of the toolpath without any effect on the rest of the trajectory. The numerical structure is determined so that may of the matrices and sub-matrices can be reused, and different feed profiles evaluated very fast without chancing the structure or values of the many constant arrays. In addition, the forward progressing windowed solution allows for the computation of optimized feedrate profile for indefinitely long toolpaths without running into memory capacity issues, as demonstrated for a sample toolpath with 9000 control points and 810,000 inequality constraints that must be satisfied while minimizing cycle time for the overall motion.

While the optimization algorithm has been successfully developed, implemented, and tested, a possible improvement, in the future, would be the refinement of the constraint equations, e.g., to facilitate more accurate prediction of the servo error on a machine with large friction. As expected, the result of any given optimization is machine-specific, and will not produce the same result on a different machine. The predictive models for tracking error and motor torque rely on assumptions about the machine dynamics which, may not always be fully accurate. Thus, improved machine characterization (e.g., considering multibody dynamics or nonlinear effects like friction) can certainly be implemented, at least in an approximate manner. For example, the open-loop multibody model can be used, or friction can be approximated with a suitable nonlinear function as a function of arc displacement and the feedrate. The good news is that the core optimization algorithm (i.e., LP + NLP) can be re-used without modification. The only parts that would need to be changed are how the velocity, acceleration, and jerk limits are defined for the LP step, and the necessary functions that handle calculation of motor torque and servo error.

The primary direction for future work would be an extension of the method into 5-axis. Prototype 5-axis optimization methods have been developed in the course of this work; however, the axes are represented as independent from each other. A 5-axis method which correctly models the interdependence of the linear and rotary axes is the next step. These constraints are inherently nonlinear, and would be well-served by the NLP optimization method.

Another issue is the adequate selection of the window and progressions sizes, as this can have a major impact on the solution efficiency and optimality for LP and NLP methods. The subsequent chapter investigates this problem with further numerical studies.

4 Impact of Configuration Choices When Utilizing the LP+NLP Method

The LP+NLP method includes parameters which must be tuned for optimal performance. These comprise of the windows sizes for the larger LP and smaller NLP windows ($W_{LP}, \quad W_{NLP}$), overlap lengths for the two methods (P_{LP}, P_{NLP}) and the number of constraint evaluation points per control point (N_c).

This chapter focusses on the impact of these parameter on the computational load for the optimization. The window size is expressed not as a travel length, but instead in terms of number of control points, which is the main factor affecting the computation time and optimality of the solution. The relationship between control points and physical length is strongly dependent on the toolpath geometry and machine limits, and must be assigned specifically based on the machining operation in question. One possibility, especially when using cubic B-splines to represent the toolpath, is to enforce a minimum number of control points per polynomial segment. This ensures that the feed is modulated with sufficient granularity as a function of changing curvature along the toolpath.

The following presented computation time tests were performed on the same computer – a desktop PC (Intel i7-3820 CPU at 3.60 GHz, with 16 GB of memory, running 64-bit Windows 10) running MATLAB version 2023b. The computation time will vary significantly based on the computer used to perform the optimization. Therefore, the computation times reported in this chapter are comparative with respect to one another. Additionally, the assumption is made that the computer performing the optimization has sufficient memory. On a computer with very limited memory the window sizes may need to be kept smaller than the ideal sizes identified in this chapter.

4.1 Window size and overlap

Sets of sample calculations were performed to analyze the effect of changing window size. The ideal LP window size was found by trial-and-error to be approximately 100 control points. However, variation in this size had only very small effects on the overall computation time. The LP algorithm is much faster than either of the available NLP algorithms, and is performed significantly less frequently than the NLP algorithms. Therefore, fine-tuning the LP window size offers only marginal possibilities for improvement. Initially tested values of LP window size ranging from 50 to 200 control points all resulted in average computation times within approximately 3% of the fastest computation case. Table 4-1 shows the results of LP window size tests. Each test was performed five times at identical configuration and the resulting computational times were averaged for reporting. The computational times were quite consistent for tests conducted under identical parameters.

As there is little loss of computational efficiency, shorter toolpaths, (such as some of the examples presented in Section 3.9 and 3.10) may be solved in LP with a one-shot solution by setting the window length to encompass the entire toolpath.

Table 4-1: Comparison of optimization time with different LP window sizes.

LP window size (# of control points)	50	75	100	150	200
Computation time (s)	10.44	10.16	10.12	10.28	10.26
Percentage increase compared with ideal size [%]	3.1	0.36	(reference)	1.5	1.3

When considered in isolation, outside of the LP+NLP optimization, the LP optimization time may be examined over a more extensive range of window sizes. Figure 4-1 presents the computation times for a total of 1000 control points, divided into various window sizes, with each successive window overlapping the previous window by 8 points (which corresponds to four times the minimum required overlap distance explained in Section 3.7). When the window size is very short, the overlap takes up a huge portion of each window, and the total number of windows required to complete the toolpath becomes very large. Therefore, the computation time is increased. At a window size of 10 control points, 496 windows are required to complete the toolpath. When the window size is very large, only a few windows are required – at a window size of 400 control points, only 3 windows are required to complete the toolpath. However, the computation time for each window also keeps increasing, which affects the total computation time.

In isolation, based on the computational platform used, the ideal window size for LP optimization was found to be 50 points. This size is large enough that the total window number remains reasonable, but small enough that each LP optimization does not take excessively long. In the combined LP+NLP optimization, the ideal LP window size was found to be 100 points, due to the additional computational load of converting the LP solution prior to NLP optimization (Section 3.5), a process that occurs after each LP window solution.

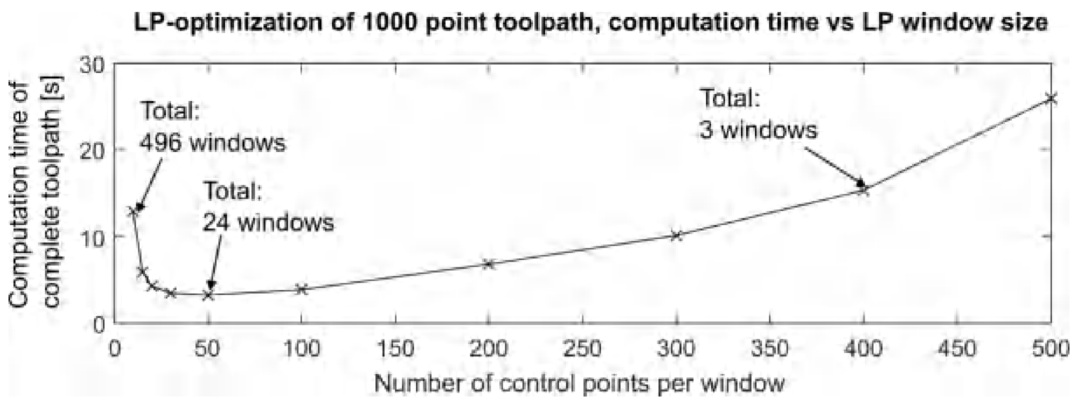


Figure 4-1: LP window size and computation time for LP-only optimization.

Changes in NLP window size have a much more significant influence on the LP+NLP optimization time. As with the LP solution, a balance must be struck between the increased computational load of a larger windows, versus the increase in redundant re-optimization of sections due to the NLP window overlap, as discussed in the proceeding paragraphs. The ideal NLP window size was found to be 20 control points on the computation platform used. Comparable window sizes of 15, 30 and 40 were also tested, and all produced increases in computational time in the range of 5-20%. Table 4-2 shows the results of NLP window size tests of the same LP+NLP optimization. In this case the LP window size was constraints to 50 control points.

Table 4-2: Comparison of optimization time with different NLP window sizes.

NLP window size (# of control points)	15	20	30	40
Computation time (s)	11.44	9.55	10.12	11.14
Percentage increase compared with ideal size [%]	19.9	(reference)	6.0	16.7

In selecting the value of overlap between windows, the priority is to avoid negative effects on the solution profile. Based on the principle of optimality described in Section 2.3.1, an optimized trajectory solved with a windowed solution should give the same result as a one-shot optimization, provided that the windows overlap sufficiently to eliminate the boundary-condition dependent slowdowns at the beginning and end of a window. Insufficient overlap will produce unnecessary local decelerations at the boundaries between windows. The theoretical minimum overlap was computed in Section 3.7, but in practice, the overlap is required to be significantly greater than the minimum, to account for numerical error, control point alignment, and constraint differences between LP and NLP. The choice of overlap was made through observing the local decelerations during many different optimization trials.

The control-point overlap between LP windows is set to be only slightly higher than the theoretical minimum value. If small local decelerations occur, they will be overwritten by the proceeding NLP step.

The LP window overlap was set to be 4 times the theoretical minimum. This was determined by converting the required overlap length into number of control points, considering the control point meshing step size. (i.e., feedrate B-spline knot distance).

The NLP window size must have a relatively large overlap, due to both the increased feedrate of the solution and to eliminate unnecessary decelerations / accelerations. If small and unnecessary local decelerations are present between NLP windows, they will remain in the final solution. In experimentation, it was found that NLP window overlap sizes of lower than 5 times the theoretical minimum could occasionally produce such decelerations.

As an example, an optimization was performed with a total length of 2400 mm which comprised 200 control points, and therefore resulted in 12 mm per control point. The theoretical minimum deceleration distance was computed as 23.1 mm, a distance which is occupied by just under 2 control points. Then, the LP overlap distance is set to be 4x this number, or 8 control points. The NLP overlap distance is set to be 5x this number, or 10 control points.

4.2 Selection of constraint evaluation resolution with respect to the feedrate control variables

Section 3.3.5 describes the implementation of the constraint meshing factor N_c, defined as the number of evaluation points per control point. A larger value of N_c will result larger optimization matrices and reduced computational efficiency. However, an excessively small value of N_c is not sufficient to verify that the constraints are met throughout the entire trajectory.

In the simplest case, where constraint violations are only checked at each control point (i.e., $N_c = 1$), the computational time is the lowest. However, the constraint checking points are placed so sparsely that large constraint violations can occur in-between checking points. These violations only become noticeable after optimization is complete and the time-domain reconstruction has been performed. A test case is presented in Figure 4-2 to illustrate this effect. The reconstructed motor torque command (i.e., control signal) and tracking error profiles are presented. The specified maximum limits for these variables are shown with the black dashed lines. As can be seen, significant constraint violations are visible, including servo errors anticipated six times that of the specified limit. During the optimization itself, no constraint violation was detected in the trajectory – they occur so briefly that they can “slip through the cracks” in relation to the sparsely-placed constraint checkpoints, only to appear after time-domain interpolation.

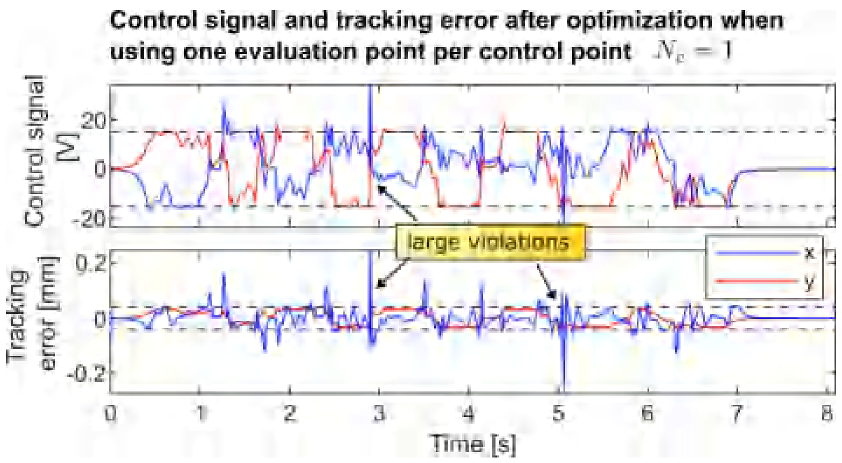


Figure 4-2: Constraint violations caused by low constraint meshing factor.

The solution to avoid these constraint violations is to increase the number of points for constraint enforcement, by increasing the value of N_c. However, each increase in N_c also increases the computational load of the optimization proportionally. The result of optimization with a meshing factor of $N_c = 10$ is presented in Figure 4-3. In comparison with Figure 4-2, only negligible constraint violations are present (less than 1% in excess of the specified limits).

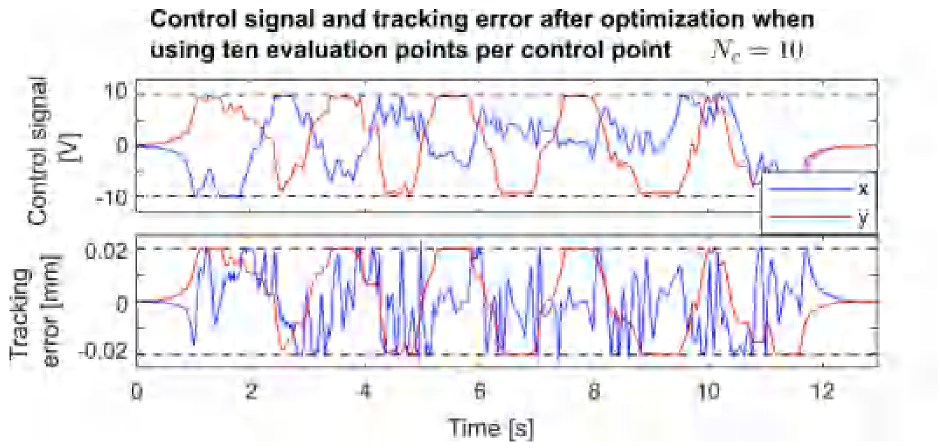


Figure 4-3: Constraint agreement when meshing factor is sufficiently high.

A set of sample optimizations was run in which different values of N_c were tested, and the result is given in Table 4-3. The ideal value of N_c may depend on the type of toolpath and manufacturing operation. For a less precise operation in which small, brief constraint violations are not problematic, a of $N_c = 5$ can perhaps be tolerated. For more precise operations, a value of $N_c = 10$ may be used to avoid small errors. Increases past 10 points are not recommended, as the computation time increases significantly, and the resulting trajectory does not differ significantly from the $N_c = 10$ case.

Table 4-3: Influence of number of constraint checkpoints per control point (N_c) on the optimization outcome.

	$N_c = 1$	$N_c = 5$	$N_c = 10$
Computation time [s]	6.35	9.51	20.33
Constraint violation (%)	400%	≤20%	≤1%
Constraint violation likelihood	very likely	occasional	(no major violations)

4.3 Comparison between sequential quadratic versus interior point NLP algorithms

Both SQP and IP algorithms described in Section 3.6 are able to solve the NLP problem, but the computational efficiency varies greatly between the two methods. In most conditions, the interior point algorithm takes significantly longer to solve an optimization problem compared to SQP. The most important factor in maintaining computational efficiency of the NLP step is to avoid infeasible starting conditions from the LP step.

Table 4-4 presents a sample of computation times of the entire optimization process of a single toolpath, when the NLP solver is set to use either the SQP or the IP algorithm. The test was performed on the sample hand shaped toolpath depicted in Section 3.9 and 3.10, with a total length of 200 control points (control point spacing = 10 mm). The LP window size was 40 points with an overlap of 8 points, and the NLP window size was 20 points with an overlap of 10 points. The computation time of Table 4-4 includes the following steps: initializations of the algorithm, LP steps, NLP steps, and time-domain reconstruction at a sampling rate of 80 Hz. To compare the tolerance for constraint violations between NLP algorithms, the LP step is solved with scaled-down or scaled up constraint limits (yielding guaranteed feasible versus marginally infeasible and significantly infeasible starting guesses for the NLP solver).

As can be seen from Table 4-4, the IP algorithm is overall slower than the SQP in normal conditions. However, if there is any violation of constraint values in the initial guess, the performance of SQP suffers significantly. The IP is better able to handle such inputs. While under normal conditions, the LP constraints are selected to avoid violation of the NLP constraints, it is nonetheless possible that infeasible starting guesses may be generated by the LP – for instance, due to small numerical errors in approximation or calculations. If constraint violations are expected from LP, it is computationally faster to scale down the LP constraints to near-zero (thus avoiding all violations entirely) and continue using SQP, rather than switch to the IP algorithm.

Table 4-4: Sample computation times of SQP and IP for toolpath in Figure 3-13.

Initial solution guess provided to NLP	SQP computation time [s]	IP computation time [s]
Zero initial conditions for NLP step (all control points are set to zero. LP solution was computed but not used as the initial guess.)	5.27	16.37
LP solved with scaled-down constraint limits (80%)	3.70	9.47
LP solved with original constraint limits (default algorithm)	3.66	8.97
LP solved with increased constraint limits (110%) ⇨ passing on minor 'infeasible' start to NLP	**19.10**	8.75
LP solved with significantly increased constraint limits (150%) ⇨ passing on significantly 'infeasible' start to NLP	(did not finish computing within the testing period of 300 seconds)	23.44

Based on this analysis, the SQP algorithm was selected for the majority of the nonlinear feed optimization problems performed. However, some cases were identified in the testing in which numerical error occurs led to a feasible solution not being found. In these cases, the optimization process may be repeated with the IP algorithm. Currently, the selection between algorithms is achieved manually. In future implementation, this can be automated so that if the SQP solution for a particular window does not converge within a given computational time, or a feasible result it not obtained with SQP, that window would be solved using IP.

4.4 Performance benchmark of LP+NLP on a higher-speed machine

The simulation and experimental examples presented previously in Sections 3.9 and 3.10 were performed considering the kinematic and dynamic limits of the wood router (Figure 3-9). This is a comparatively slow machine with lower acceleration capabilities and worse servo accuracy compared to CNC machines commonly used in industry. To evaluate the effectiveness of the algorithm for faster machines, an additional simulation was performed with higher axis kinematic limits and greater accuracy requirements. The parameters and limits considered in the 'fast machine' case are summarized in Table 4-5 and Table 4-6.

Table 4-5:Parameters used in higher-speed machine simulation case study.

	X-axis	Y-axis
M	6.0×10^{-3}	3.0×10^{-3}
b	1.2×10^{-1}	9.0×10^{-2}
c	4.5×10^{-1}	-4.5×10^{-3}
d_{coul}	7.5×10^{-1}	3.9×10^{-1}
K_v	-2.24×10^{-5}	4.03×10^{-5}
K_a	1.0×10^{-5}	7.6×10^{-7}
K_j	-2.6×10^{-8}	-9.0×10^{-4}

Table 4-6: Limits used in higher-speed machine simulation case study.

	X-axis	Y-axis
Maximum velocity [mm/s]	250	250
Acceleration [mm/s^2]	5000	5000
Jerk [mm/s^3]	40000	40000
Control signal [%]	50	50
Tracking error [mm]	0.02	0.02

The result of the optimization is given in Figure 4-4.

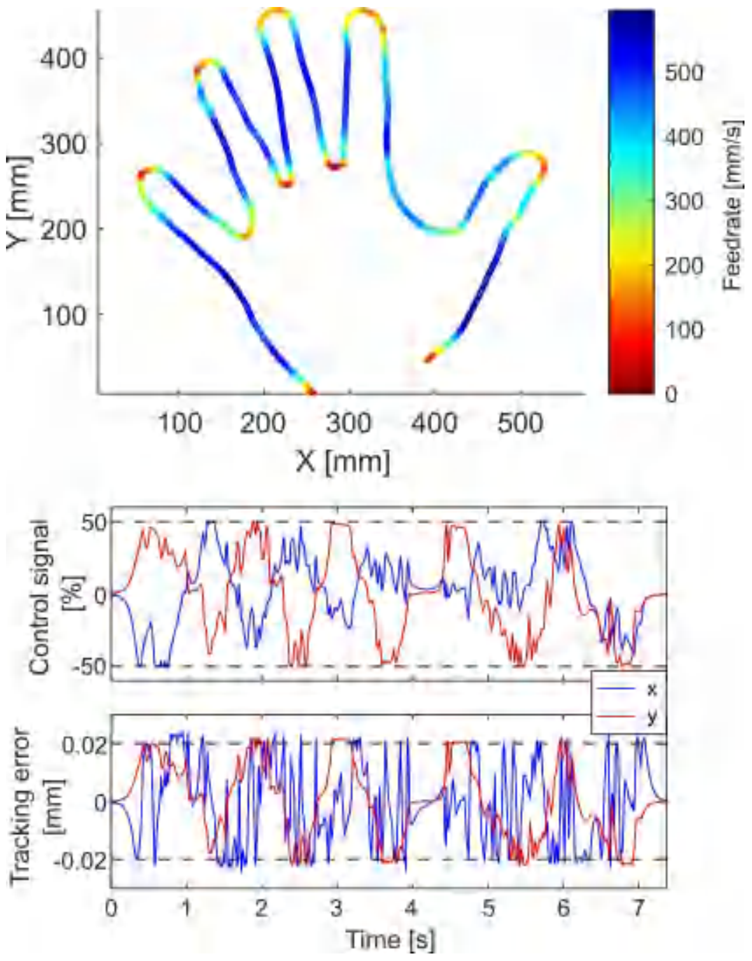


Figure 4-4: Simulated trajectory optimization on higher-speed machine.

Trajectories generated for the faster machine are successful in simulation. As with the trajectories generated considering the router dynamics, the control signal and tracking error remain within their limits with only minor numerical errors, particularly in the tracking error profile.

4.5 Conclusions

In this chapter, the selection of appropriate values for LP and NLP window sizes, overlap, and constraint meshing factor are discussed. The motivation by which the parameters are selected is to reduce computation time as much as possible while reducing the risk of constraint violation in the output trajectory.

Ideal window sizes selected, based on the used PC computational platform, sample toolpath geometries, and machine limits considered, were 100 control points for LP and 20 for NLP. In general, LP window sizes between 50 and 200 control points were found to be acceptable, as the LP window size has a secondary effect on the total computation time. The NLP window size has a greater influence. The overlap was selected as a multiple of the theoretical minimum overlap discussed in Section 3.7, with 4 times the minimum for LP and 5 times the minimum for NLP windows. The large overlaps are selected to avoid unnecessary local decelerations that occur at window boundaries, due to zero velocity and acceleration initial and final conditions being imposed in each LP window problem definition.

The constraint meshing factor describes the number of constraint evaluation points per control point. An overly small meshing factor results in a final (time-domain) trajectory with many large constraint violations. Excessively large meshing factor, on the other hand, increases computation time. Bad on the toolpath that was tested, a value of 5 was found to be ideal for computational efficiency if some constraint violation (e.g., 20%) errors could be tolerated. A minimum value of N_c=10 was required for high-precision results in which constraint violation has to be smaller (e.g., $\leq$1%).

Comparison was made between the SQP and IP algorithms for fulfilling the NLP window step. SQP is significantly faster than the IP algorithm for all windows with a feasible initial guess. As the LP step is designed to ensure feasibility of the starting guess, so the SQP algorithm can be selected as the primary choice of NLP solver in nearly every case. Exceptions were noted, from rare cases (encountered during more extensive testing) in which unpredictable numerical errors caused the SQP solver to fail. It was observed that the IP algorithm is better able recover from these errors. Hence, in the final version of the proposed algorithm, NLP has been implemented such that for each new window SQP, is first utilized and only if this method does not succeed, the solution is obtained via IP.

An additional simulation was presented which validated the effectiveness of the algorithm when used to generate an optimized trajectory for a higher-speed machine tool in comparison to the router, like ones commonly used in the metal-cutting industry.

From the insights gained in this chapter, the selection of appropriate parameters can be achieved, that leads to efficient and effective optimization of the feedrate profile in an error-free manner.

5 Smooth Connection Between Toolpaths using Euler Spirals

5.1 Introduction

Two toolpath smoothing methods are presented in this chapter. The first modifies the Euler spiral-based corner smoothing method to combine synergistically with the optimization method discussed in Chapter 3, to reduce the computational time of the combined smoothing and optimization. The second method uses Euler spirals to generate transition toolpaths between the passes of a layered contour machining operation. The Euler spiral is an arc length parametrized curve in which the curvature changes linearly with the arc displacement. This continuous, gradual change in curvature makes it ideal for curves requiring tangent and curvature transitions. For which reason, it has been used extensively in road and railroad design. This benefit also applies for CNC toolpath planning. The G2-continuity of the Euler spiral profile means that, when combined with a continuous feedrate profile, they generate continuous velocity and acceleration profiles, and bounded jerk. The formulae for computing the coordinate points of Euler spirals require the use of numerical integration, which does bring a minor amount of computational load during initialization, similar to that of polynomial splines.

The corner smoothing method developed in this thesis applies Euler spiral “pairs” to replace sharp corners in straight-line toolpaths, such as in [28], [29], [30], [31], resulting in a toolpath with continuity in both first and second geometric derivatives. The smoothed path remains within a designated maximum contouring deviation when compared with the original toolpath. The methodology developed in this thesis can be applied in 2- or 3-dimensional straight-line toolpaths and differing from the earlier works, the toolpath smoothing has also been integrated with automated feedrate optimization.

The combination of the corner smoothing and feed optimization takes advantage of a specific property of Euler spirals. While they are computationally expensive due to the requirement for numerical integration, their derivatives are much simpler to compute. This is beneficial in the context of optimization, which requires the axis-level geometric derivatives to be determined a priori. There is no requirement for the position coordinates to be computed until the optimization is complete and the trajectory is being generated for commanding the machine tool. The computational time of the derivative-checking step is reduced by 90%, and therefore the total time of the optimization is significantly lower due to this.

Also in this chapter, an Euler-spiral based method for generating connecting movements between flat contouring passes is presented. This method is designed for operation in which the contouring passes are layered one on top of another. This technique affects only the repositioning segments between the contouring passes and does not change the cutting toolpath or feedrate, which can be advantageous in the case of high-precision machining operations with strict requirements for cutting feedrates. This transition toolpath has been developed for reducing the total cycle time for machining such layered contouring toolpaths, while preserving the to final part quality due to not making changes to the actual ‘cutting’ portion of the operation.

5.2 Euler spirals

The Euler spiral is defined as a two-dimensional curve in which the curvature changes linearly with respect to the arc displacement [30]. In Figure 5-1, a sample segment is shown which includes a straight line (zero curvature), followed by a linear increase in the curvature (relative to the arc displacement s) throughout the red highlighted Euler spiral segment, and then a circular arc of radius R_c, shown on the curvature plot as a constant curvature of $\frac{1}{R_c}$.

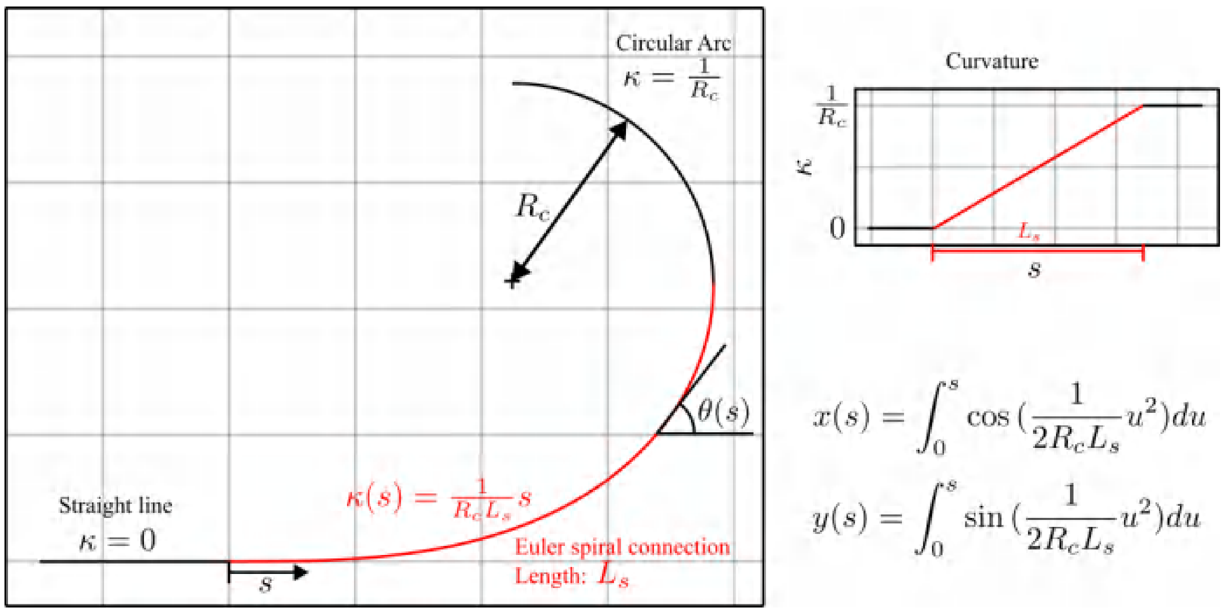


Figure 5-1: Euler spiral between a straight line and a circular arc.

5.2.1 Derivation of the Euler spiral equation

The derivation of the x- and y-coordinates of the Euler spiral starts with the linear expression of curvature, κ, shown in Eq.(5-1). In this equation, c is a constant that defines the rate at which the curvature changes as a function of arc displacement s.

$$\kappa(s) = cs \tag{5-1}$$

The value of c can be computed based on the length of the Euler spiral segment, L_s, and the ending radius of curvature R_c.

$$c = \frac{1}{R_c L_s} \tag{5-2}$$

From Eq. (5-2), the expression for the angle of the line $\theta(s)$, depicted in Figure 5-1, can be derived, as in Eq. (5-3). The expression is simple and does not require numerical integration.

$$\theta(s) = \int_0^s cu\,du = \frac{1}{2}cs^2 \tag{5-3}$$

The x- and y-coordinates of the resulting curve can be derived from the angle. For a small segment, ds

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} ds\cos\theta(s) \\ ds\sin\theta(s) \end{bmatrix} \tag{5-4}$$

Which can then be integrated over the whole curve, as given in Eqs. (5-5) and (5-6).

$$x(s) = \int_0^s \cos\left(\frac{1}{2}cu^2\right) du \tag{5-5}$$

$$y(s) = \int_0^s \sin\left(\frac{1}{2}cu^2\right) du \tag{5-6}$$

The above equations can also be normalized with a scaling factor, a:

$$a = \sqrt{\frac{c}{2}} \tag{5-7}$$

Once normalized, the x- and y-coordinates can be expressed in terms of the Fresnel integrals, as in Eqs. (5-10) and (5-11). Fresnel integrals, defined as the cosine and sine of the square of the input, have been significantly studied. Methods for accurate numerical computation of these integrals are available in literature, as discussed in 5.3.4.

$$\kappa(s) = 2a^2 s \tag{5-8}$$

$$\theta(s) = a^2 s^2 \tag{5-9}$$

$$x(s) = \frac{1}{a}\int_0^{as} \cos u^2\,du \tag{5-10}$$

$$y(s) = \frac{1}{a}\int_0^{as} \sin u^2\,du \tag{5-11}$$

At this point, in the best of the author's knowledge, the x- and y-coordinates of the Euler spiral cannot be simplified further. Numerical integration is required to compute the coordinates along the line.

5.2.2 Geometric derivatives of the Euler spiral

For a given Euler spiral segment (Eq. (5-12)), the first, second, and third geometric derivatives are given by Eqs. (5-13) to (5-15).

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \frac{1}{a}\begin{bmatrix} \int_0^{as} \cos u^2\,du \\ \int_0^{as} \sin u^2\,du \end{bmatrix} \tag{5-12}$$

$$\begin{bmatrix} x_s(s) \\ y_s(s) \end{bmatrix} = \begin{bmatrix} \cos(a^2 s^2) \\ \sin(a^2 s^2) \end{bmatrix} \tag{5-13}$$

$$\begin{bmatrix} x_{ss}(s) \\ y_{ss}(s) \end{bmatrix} = 2a^2 s \begin{bmatrix} -\sin(a^2 s^2) \\ \cos(a^2 s^2) \end{bmatrix} \tag{5-14}$$

$$\begin{bmatrix} x_{sss}(s) \\ y_{sss}(s) \end{bmatrix} = 2a^2 \begin{bmatrix} -2a^2 s^2 \cos a^2 s^2 - \sin a^2 s^2 \\ -2a^2 s^2 \sin a^2 s^2 + \cos a^2 s^2 \end{bmatrix} \tag{5-15}$$

While determining the x- and y-axis coordinates requires numerical computation of the Fresnel integrals in Eq. (5-12), the axis derivatives are expressed with simple formulae which can easily be computed analytically. This offers an advantage in computation time for trajectory optimization routines that rely on axis derivatives. The resulting derivative profiles are shown in Figure 5-2. When used for corner smoothing, only a short segment of this profile is necessary, but the profiles in Figure 5-2 have been extended well beyond these parts to verify the position and geometric derivatives while traversing multiple quadrants of the same Euler spiral.

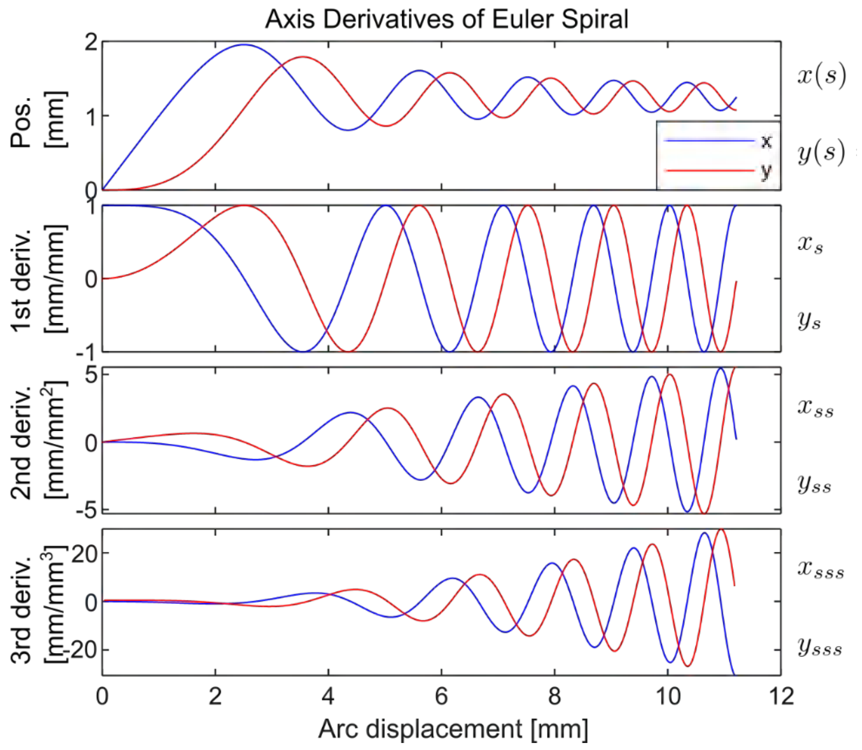


Figure 5-2: Euler spiral position coordinates and its geometric derivatives.

As shown earlier in Eqs. (3-18)-(3-20), the kinematic profiles of a trajectory are products of the axis geometric derivatives with respect to arc length, the feedrate ($\dot{s}$), and its time derivatives ($\ddot{s}$, $\dddot{s}$). Ensuring that the axis first and second derivatives are continuous is necessary to ensure that the kinematic profiles of a trajectory along this path remain bounded. An additional advantage is the arc-length parametrization of the Euler spiral segments. Indeed, given the desired arc length to be traveled (s), computing the Fresnel integrals in Eq. (5-12) provides directly the x- and y-axis coordinates corresponding to that displacement, eliminating the need to perform feed fluctuation corrections, which is a common issue with polynomial type splines (like cubic or quintic), which are not arc length parameterized [7], [8].

5.3 Euler spiral pair connections between linear toolpath segments

In this section, a general method for applying Euler spirals to smooth the corners of toolpaths is developed and described. Clothoid pair methods were proposed in literature before [28], [29], [31], [33]. In this thesis, they have been synergistically integrated with the feed optimization in Chapter 3 to yield quick and time-optimal cornering motion, while also providing advantageous mathematical computation properties. The developed methodology applies to 2D paths, and has also been extended to 3D paths as described in Section 5.3.3.

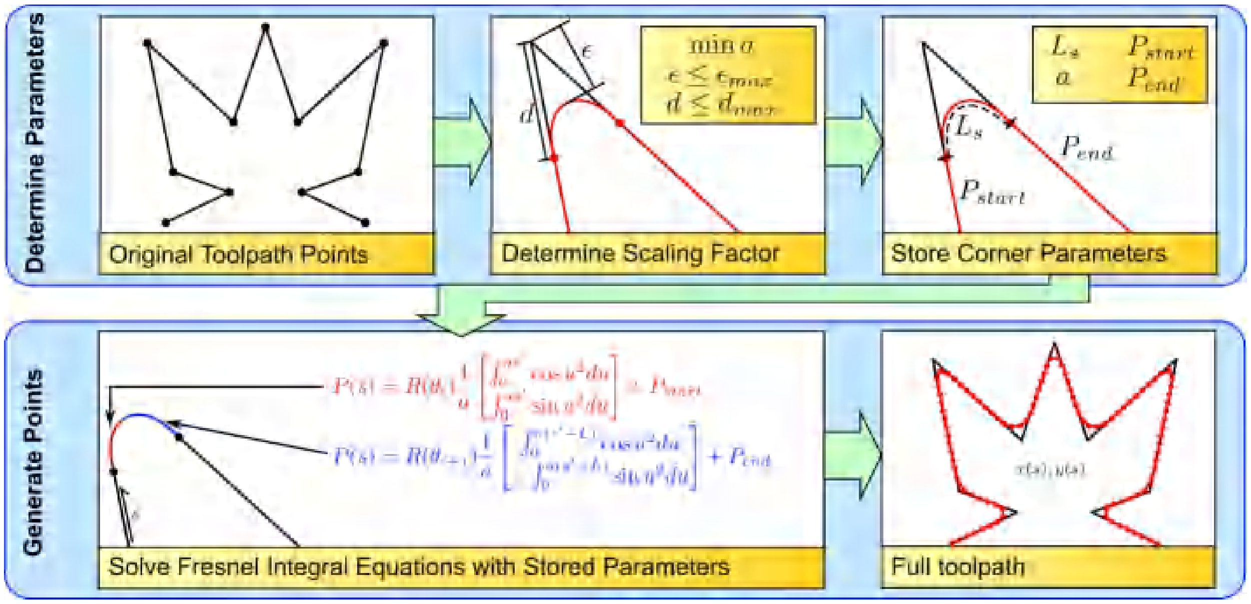


Figure 5-3: Summary of corner smoothing method.

This method directly constrains the maximum contouring deviation $\epsilon_{\max}$ with respect to the original path, and the maximum cornering distance $d_{\max}$. These dimensions are indicated in Figure 5-3, and their computation is detailed in Section 5.3.1.

The Euler spirals are computed based on the parameter, s', which denotes the local arc displacement of any corner, relative to its starting point s_{start}, (the arc displacement at the point P_{start}).

$$s' = s - s_{start} \tag{5-16}$$

Figure 5-3 summarizes the method of smoothing using a symmetrical Euler-spiral pair. Given a set of input points, the smoothed profile of each corner must be generated. In the top half of Figure 5-3, a set of parameters is computed that defines the shape of each corner (as will be detailed in Section 5.3.1). The stored values of these parameters for all corners provide a complete definition of the toolpath, from which x- and y-coordinate points may be computed for any arc displacement value s, as shown in the bottom half of Figure 5-3.

5.3.1 Determination of Euler spiral parameters for a single corner

For each corner in the original toolpath, the following parameters, designated in Figure 5-4, must be computed:

- Scaling factor, a
- Corner length, L_s
- Start and end points, P_{start} and P_{end}

Because the final size of the clothoid pair is not yet known, a temporary scaling factor is assumed, as $a = 1$. Using the definition of the angle in Eq. (5-9), the total arc length of the corner can be computed from the original angle of the corner, ϕ:

$$\theta\left(\frac{L_s}{2}\right) = \frac{\phi}{2} \tag{5-17}$$

$$L_s = \sqrt{2\phi} \tag{5-18}$$

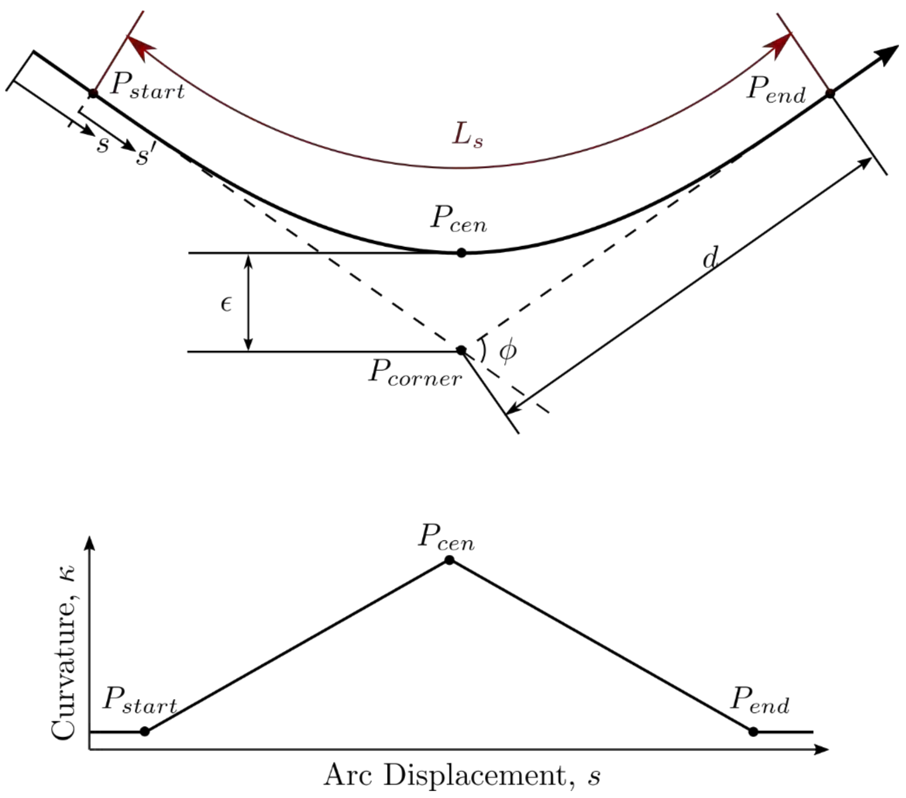


Figure 5-4: Single Euler spiral corner.

Figure 5-5 shows the contouring deviation and corner blending distance computed from the centre point of the Euler spiral. In the second step, the correct scaling factor is computed such that the Euler spiral fits within the applied contouring deviation and cornering distance constraints.

The values of contour deviation and cornering distance which correspond to the temporary scaling factor $a = 1$ can be computed. These are denoted with ϵ' and d'. A temporary centre point P'_{cen} can be computed from Eq. (5-19).

$$P'_{cen} = \begin{bmatrix} x'_{cen} \\ y'_{cen} \end{bmatrix} = \begin{bmatrix} \int_0^{\sqrt{\frac{\phi}{2}}} cos\, u^2\, du \\ \int_0^{\sqrt{\frac{\phi}{2}}} sin\, u^2\, du \end{bmatrix} \tag{5-19}$$

The temporary contour deviation ϵ' and cornering distance d' can be computed using Eqs. (5-20) and (5-21).

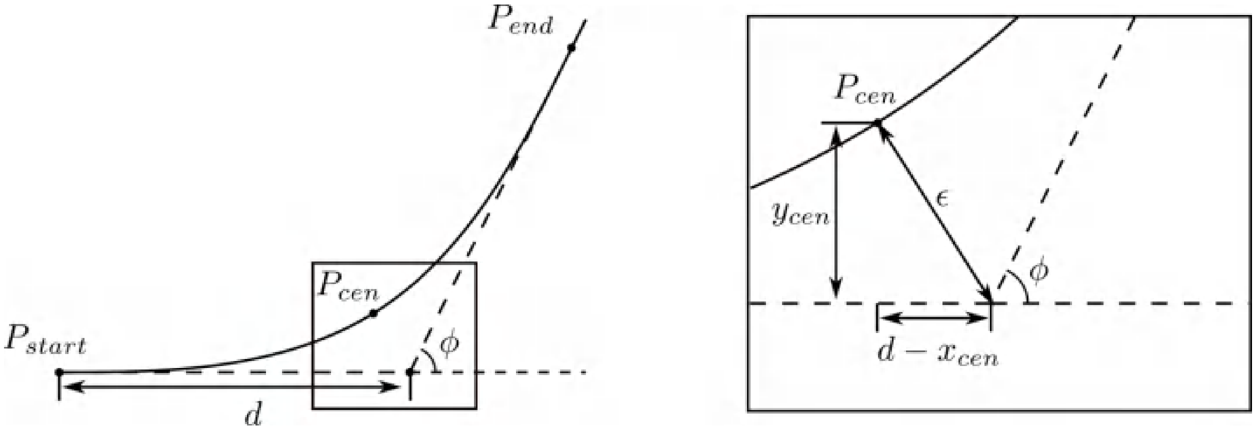


Figure 5-5: Maximum path error and corner blending distance.

$$\epsilon' = \left| \frac{y'_{cen}}{\sin\left(\frac{\pi - \phi}{2}\right)} \right| \tag{5-20}$$

$$d' = \left| \epsilon' \cos\left(\frac{\pi - \phi}{2}\right) + x'_{cen} \right| \tag{5-21}$$

Because a corresponds to a linear scaling of the size of the corner, it can be assigned by comparing the temporary contouring deviation and cornering distance with the assigned constraints.

$$a = \max\left(\frac{\epsilon'}{\epsilon_{max}}, \frac{d'}{d_{max}}\right) \tag{5-22}$$

Finally, the start and end points of the line can both be easily computed – they are both at a distance d from the corner point P_{corner} in the directions of incoming and outgoing lines to and from the corner point, respectively.

5.3.2 Converting the double clothoid curve into toolpath coordinates

Using the computed parameters, the x- and y-coordinates of the line can be computed for any value of s. First, the local arc displacement parameter of a corner is computed. If s_{start} indicates the value of s at P_{start}, then the local parameter s' is computed as in Eq (5-16), from which the values of x and y can be computed in Eqs. (5-23) and (5-24), where θ_i is the angle of the line immediately before the corner point, and θ_{i+1} is the angle of the line immediately after.

$$\begin{bmatrix} x(s') \\ y(s') \end{bmatrix} = R(\theta_i)\frac{1}{a}\begin{bmatrix} \int_0^{as'} \cos u^2 \, du \\ \int_0^{as'} \sin u^2 \, du \end{bmatrix} + P_{start}, 0 \leq s' \leq \frac{L}{2} \tag{5-23}$$

$$\begin{bmatrix} x(s') \\ y(s') \end{bmatrix} = R(\theta_{i+1})\frac{1}{a}\begin{bmatrix} \int_0^{a(s\prime - L)} \cos u^2 \, du \\ -\int_0^{a(s\prime - L)} \sin u^2 \, du \end{bmatrix} + P_{end}, \frac{L}{2} \leq s' \leq L \tag{5-24}$$

Above, the rotation matrices $R(\theta_i)$ and $R(\theta_{i+1})$ can be expressed as,

$$R(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \quad , \quad R(\theta_{i+1}) = \begin{bmatrix} \cos\theta_{i+1} & -\sin\theta_{i+1} \\ \sin\theta_{i+1} & \cos\theta_{i+1} \end{bmatrix} \tag{5-25}$$

The values of the function's geometric derivatives can also be expressed as in Eqs. (5-13) to (5-15) with rotations applied similarly to the x- and y-coordinate equations.

5.3.3 Extension of corner smoothing to 3-axis

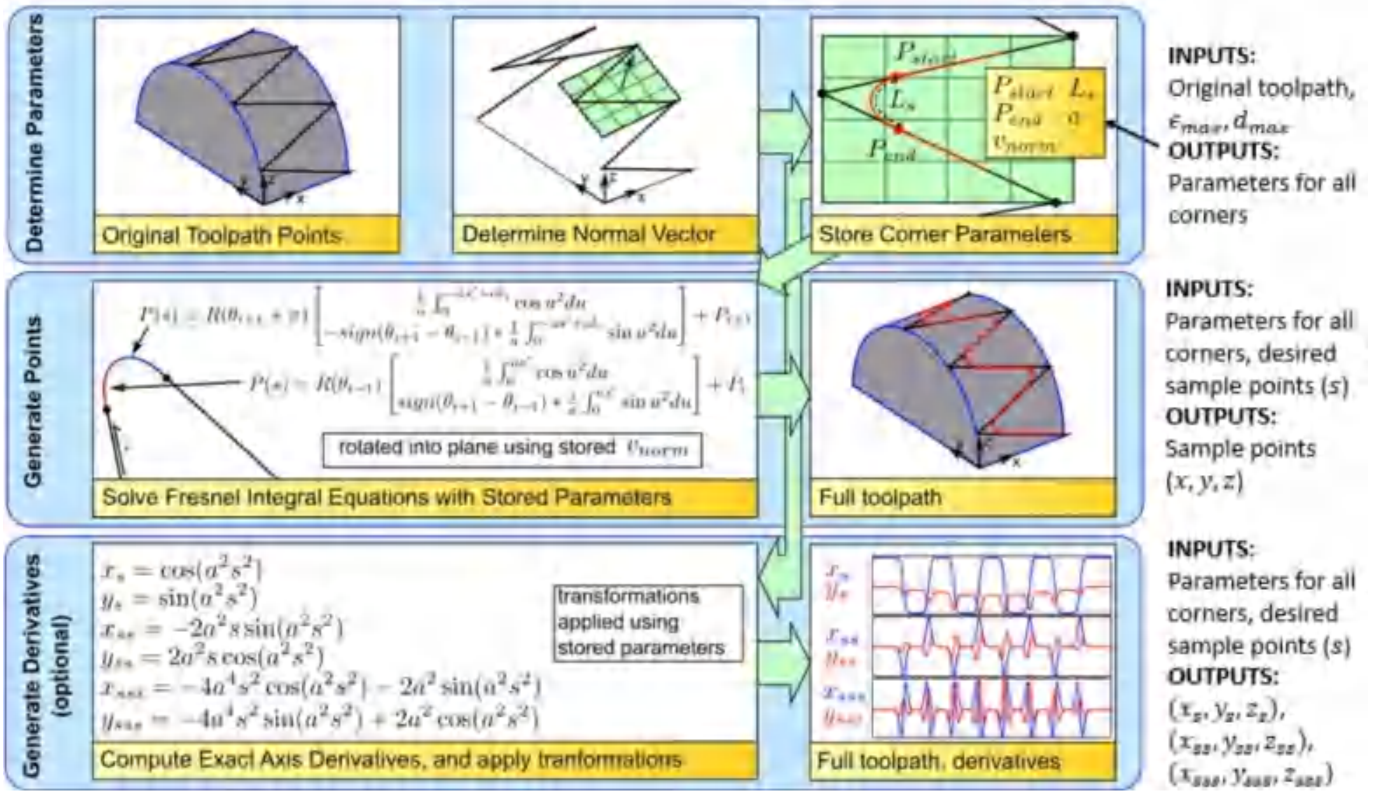

Determine Parameters
Original Toolpath Points
Determine Normal Vector
Store Corner Parameters
INPUTS:
Original toolpath, ϵ_{max}, d_{max}
OUTPUTS:
Parameters for all corners
Generate Points
Solve Fresnel Integral Equations with Stored Parameters
rotated into plane using stored v_{norm}
Full toolpath
INPUTS:
Parameters for all corners, desired sample points (s)
OUTPUTS:
Sample points (x, y, z)
Generate Derivatives (optional)
transformations applied using stored parameters
Compute Exact Axis Derivatives, and apply tranformations
Full toolpath, derivatives
INPUTS:
Parameters for all corners, desired sample points (s)
OUTPUTS:
(x_s, y_s, z_s), (x_{ss}, y_{ss}, z_{ss}), $(x_{sss}, y_{sss}, z_{sss})$

Figure 5-6: Three-dimensional Euler spiral based toolpath smoothing.

This method can also be implemented for 3D toolpaths by applying a suitable rotation. Each corner is defined by two lines – and therefore a single tilted plane. Each corner is created in 2-D in the corresponding plane, and then rotated into place to match the original linear toolpath segments. To facilitate the extension to 3D, an additional vector is computed for each plane, a normal vector v_{norm} which defines the plane's orientation.

The process of creating 3D corners takes place in three steps. First, the tilted plane on which the corner fits is rotated on the x-y plane. Second, the desired sampling points for corner are created using the 2D approach, defined in Sections 5.3.1and 5.3.2. Last, these points are rotated back into their intended orientations.

To identify the angle of the tilted plane of the corner, v_{norm} is computed as in Eq. (5-26).

$$v_{norm} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = (P_{corner} - P_{start}) \times (P_{end} - P_{corner}) \tag{5-26}$$

Above, $\times$ represents the vector product.

Following this calculation, a set of transformations is generated, which rotates the tilted plane containing the corner to the x-y plane, such that v_{norm} aligns with the corresponding z-axis.

For any point P, the modified P_r which is in the x-y plane is computed using Eq. (5-27). The rotation matrices are defined in Eq. (5-28), based on the components of v_{norm}.

$$P_r' = R_2 R_1 P \tag{5-27}$$
$$P_r = R_2 R_1 P - P_{r,z}'$$

$$R_1 = \begin{bmatrix} \frac{v_x}{\sqrt{v_x^2 + v_y^2}} & \frac{v_y}{\sqrt{v_x^2 + v_y^2}} & 0 \\ -\frac{v_y}{\sqrt{v_x^2 + v_y^2}} & \frac{v_x}{\sqrt{v_x^2 + v_y^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_2 = \begin{bmatrix} \frac{v_z}{\sqrt{v_x^2 + v_y^2 + v_z^2}} & 0 & -\frac{\sqrt{v_x^2 + v_y^2}}{\sqrt{v_x^2 + v_y^2 + v_z^2}} \\ 0 & 1 & 0 \\ \frac{\sqrt{v_x^2 + v_y^2}}{\sqrt{v_x^2 + v_y^2 + v_z^2}} & 0 & \frac{v_z}{\sqrt{v_x^2 + v_y^2 + v_z^2}} \end{bmatrix} \tag{5-28}$$

The rotation is performed in two steps: R_1 corresponds to a rotation that aligns v_{norm} with the x-z plane, and R_2 corresponds to a rotation about the y-axis to align v_{norm} with the z-axis. After this rotation, the point is then translated into the x-y plane by subtracting its z position $P_{r,z}'$. Once these transformations are applied to P_{start}, P_{end}, and P_{corner}, the sample trajectory points along the Euler corner smoothing is generated in the same way as the 2D approach, resulting in a set of points $x_r(s)$, $y_r(s)$. The final set of corner points is then rotated and translated using the above transformation in reverse, as in Eq. (5-29).

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = R_1^{-1} R_2^{-1} \left(\begin{bmatrix} x_r(s) \\ y_r(s) \end{bmatrix} + P_{r,z}' \right) \tag{5-29}$$

5.3.4 Numerical computation of Fresnel integrals

The generation of toolpaths relies on fast and accurate numerical integration methods. Selection of a numerical method for Fresnel integrals must usually account for their oscillatory nature, which causes them to be ill-conditioned and prone to large numerical errors [65]. However, in this application, the integration is performed only on a limited portion of the function, which avoids the oscillatory section altogether. High accuracy is also needed for interpolation, otherwise the planned velocity, acceleration, and jerk profiles will not be replicated correctly. During testing, it was observed that simple first order or trapezoidal cumulative summation produced excessive error and caused misalignment of the Euler spiral segments. Thus, the Fresnel integrals have been computed in this thesis using MATLAB’s ‘quadgk’ function. This function uses the Gauss-Kronrod quadrature method, which is particularly effective both for polynomial curves and also Fresnel integrals. Further discussion of the Gaussian quadrature method can be found in [65].

5.3.5 Computation of only the geometric derivatives at the constraint evaluation points

To compute the sampled axis coordinates along the smoothed toolpath (x, y, z), computation of integrals is required. These integrals take up significant computational load. However, computing the axis derivatives of the toolpath requires only simple formulae. In benchmarking, it was found that evaluating the actual position coordinates on the Euler spiral required roughly 9 times more computational time compared to evaluating the first, second, and third geometric derivatives. However, the feedrate optimization algorithm needs only the geometric derivatives, and not the position coordinates themselves. Hence, once the optimized feed profile is determined, the numerical integration applied to compute the position coordinates only during the interpolation of the final axis trajectory.

5.3.6 Simulation case study: Euler spiral based corner smoothing integrated with feed optimization

A sample combined trajectory planning, involving smoothing of the 'crown' shape in Figure 5-3, and then applying feedrate optimization via LP+NLP is shown in Figure 5-7.

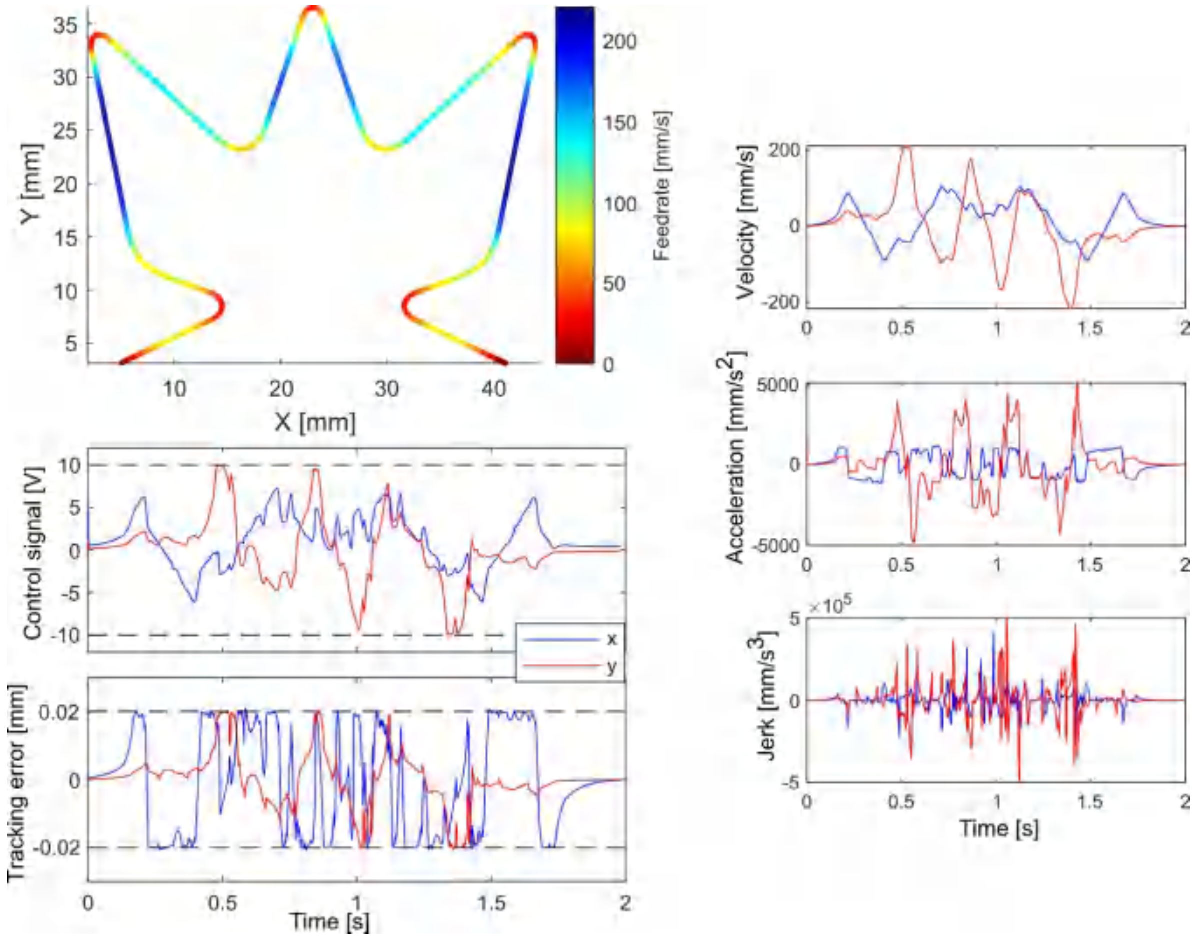
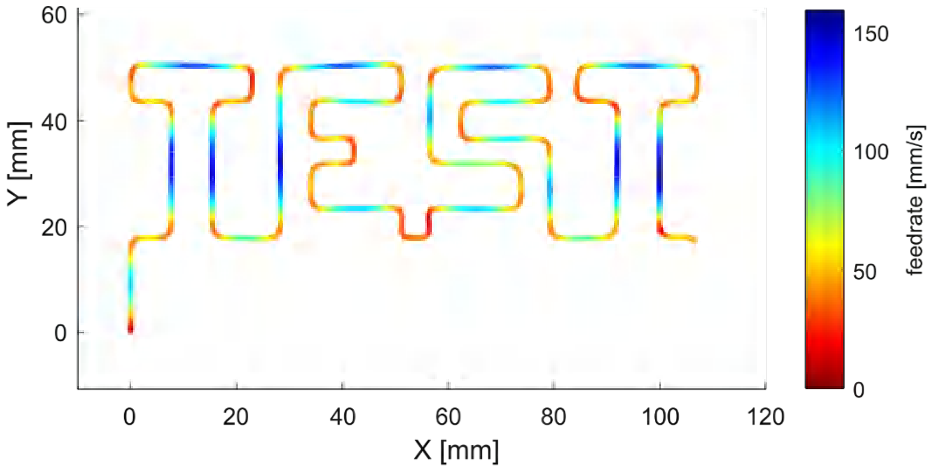


Figure 5-7: Result of combined corner smoothing and feedrate optimization.

In processing the toolpath, the corner smoothing path deviation tolerance was set to 2 [mm], and the maximum cornering distance to 40% of the length of any line. After executing the algorithm, it was verified via inspection that at each acute corner, the contour deviation was indeed exactly 2 mm. At the obtuse corners, the cornering distance constraint became active, and the path deviation was lower than 2 mm. Plots of the predicted maximum control signal and servo tracking error are also shown. Each of these is kept within the enforced limit, apart from small numerical errors caused by the time-domain reconstruction. By the time NLP is complete, the velocity, acceleration, and jerk exceed the original LP-enforced constraints, as is expected in the LP+NLP optimization.

5.3.7 Experimental validation for corner smoothing



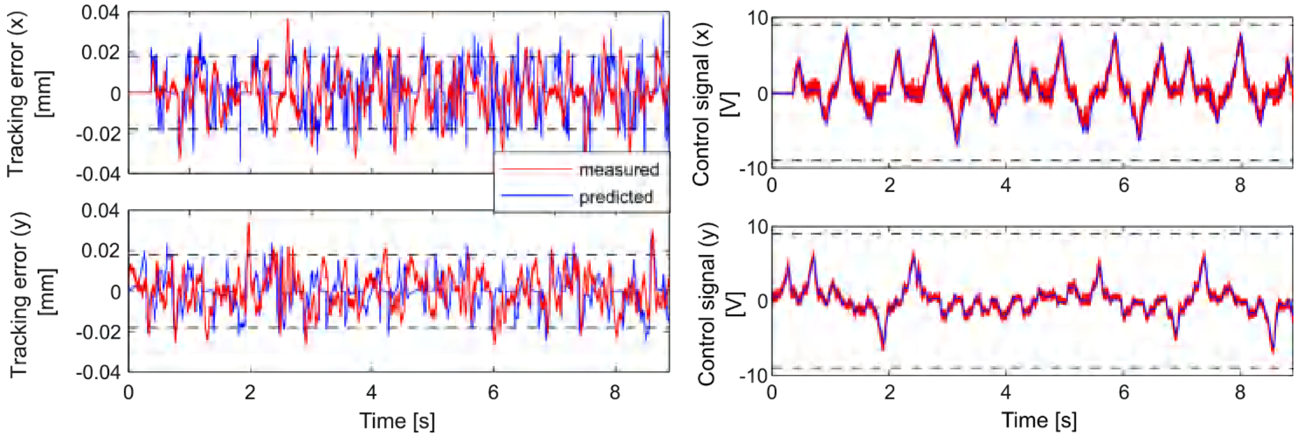


Figure 5-8: Experimental result of Euler-spiral smoothed toolpath with LP+NLP feedrate optimization.

Combined toolpath smoothing and feedrate optimization was performed on a corner-smoothed toolpath on the same experimental setup described in Section 3.10. The result is presented in Figure 5-8. The experimental control signals are in excellent agreement with prediction results. Once again, due to imperfections of the machine (e.g. friction and other nonlinearities), some inconsistency between the predicted and measured servo is evident. Nevertheless, servo errors are in the range of prediction and on a higher-end machine tool with better rigidity and lower friction, closer correlation can naturally be expected. The tool is able to traverse the toolpath at the variable feedrate shown in Figure 5-9, reaching high speeds during the straight-line segments, and continuing at moderate feedrates when turning around the corners.

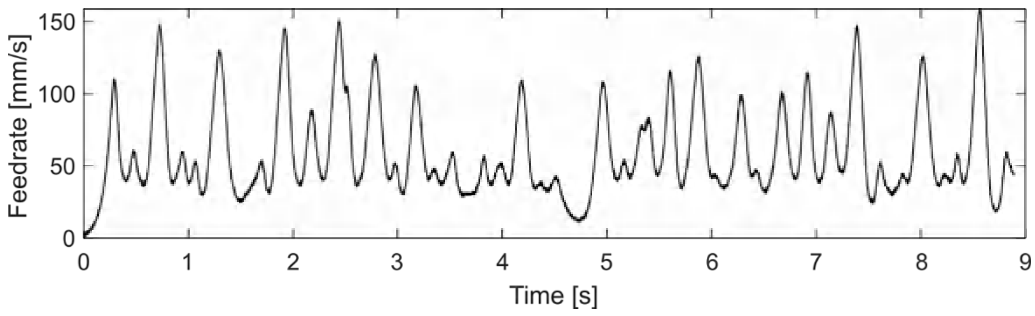


Figure 5-9:Feedrate profile of experimentally tested Euler-spiral corner smoothing and LP+NLP optimization result.

5.4 Connecting multi-layer toolpaths using Euler spirals

Euler spirals can also be used to plan smooth in-between connections for multi-layered contouring toolpaths. The toolpaths considered here are flat two-dimensional passes around, or within, a shape with each pass being at a different height in the orthogonal direction. Figure 5-10 illustrates such an operation for a ‘star-shaped’ contour.

Various methods exist for generating 3D machining toolpaths. When layer-by-layer contouring is required with the z-axis feed occurring between the passes, an Euler spiral based toolpath can be used to reduce motion time for repositioning between the layers. This is achieved by eliminating redundant decelerations and accelerations.

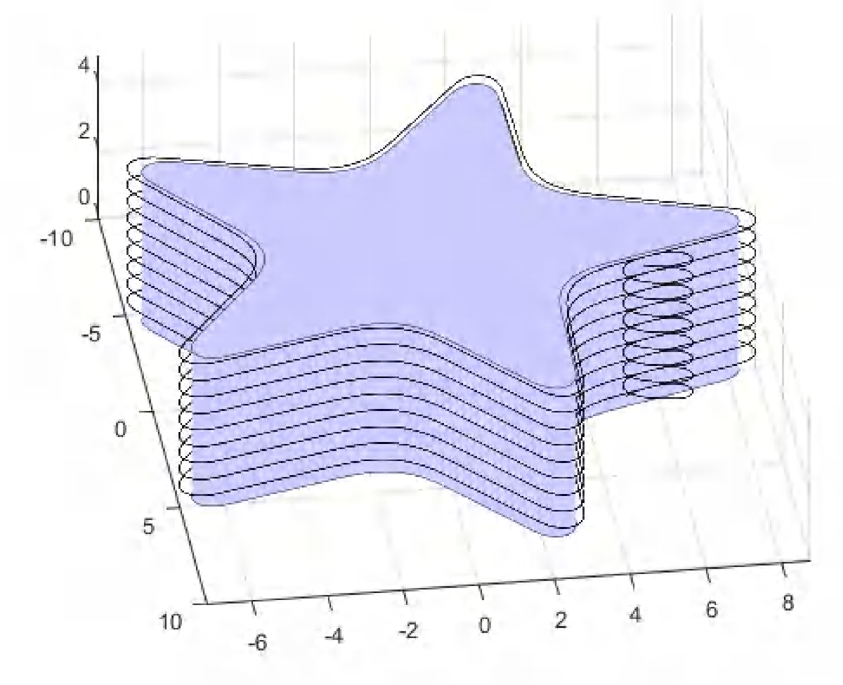


Figure 5-10: Multi-layer contouring operation connected via Euler spiral-based toolpaths.

Figure 5-11a shows a top-down view of a typical contouring operation with lead-in/lead-out circular arcs, which is a method commonly used in industry. Figure 5-11b shown an Euler spiral-based layer transition curve, which has been developed in this thesis and integrated with feedrate optimization.

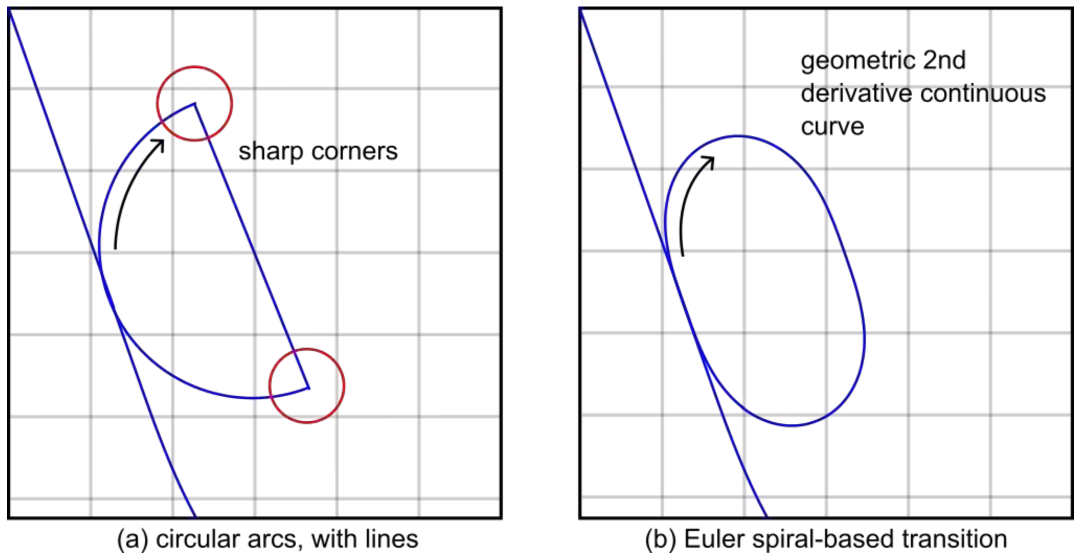

sharp corners
(a) circular arcs, with lines
geometric 2nd
derivative continuous
curve
(b) Euler spiral-based transition

Figure 5-11: Comparison between circular arc lead-in and lead-out and Euler spiral based transition, top-down view.

The repositioning procedure in Figure 5-11a includes two circular arcs. The movement in the z-axis is executed entirely in one straight line while connecting the sharp corners. In comparison, the Euler spiral-based layer transition in Figure 5-11b can smoothly execute the movement from one z-level to the next, so that the axes do not have to stop or slow down significantly, as they would with the sharp corners that occur in the circular arc based repositioning strategy.

The Euler spiral based transition toolpath is shown in 3D in Figure 5-12. The section denoted ‘cutting’ remains unchanged for the contour machining portion. The ‘lead-in’ and ‘lead-out’ are accomplished with Euler spiral segments, which remain within the same x-y plane within the consecutive contouring layers. The ‘layer transition’ section is smooth between the z-levels of the current and proceeding layers, and also blends the x-y coordinates of the toolpaths between the two layers.

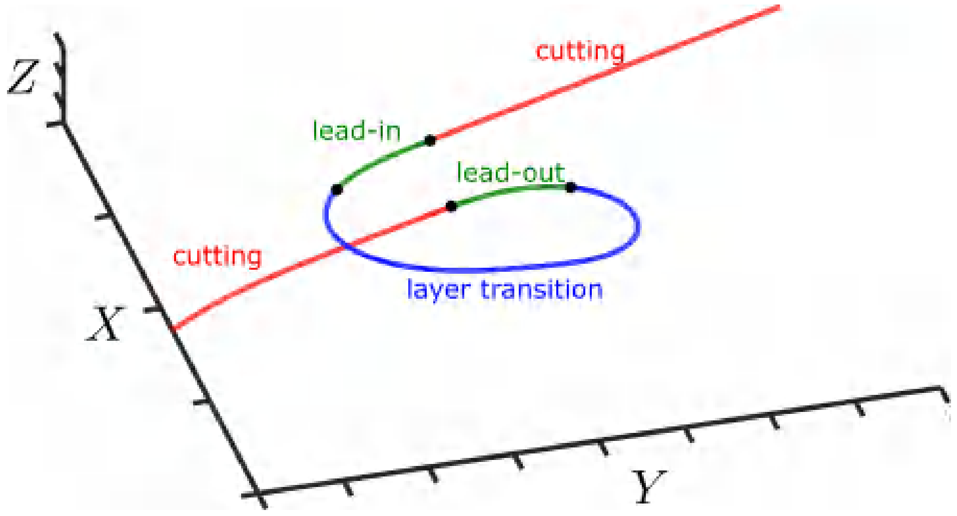


Figure 5-12: Smooth transition between contour machining layers, 3D view.

5.4.1 Parameterization of the layer transition curve

The layer transition curve is created in two steps, combining Euler spiral segments aligned with each other, which are then blended together. The base curve, shown in Figure 5-13a, is the starting point. An additional curve (the layer blending function) is applied to accomplish the change in the z-level, and also to accommodate minor changes in the x-y location and direction of the start and end points in each layer.

It should be noted that after the application of the layer blending function, the resulting toolpath is no longer an exact Euler spiral – the curvature plot is no longer strictly linear, and the Euler spline parameter s no longer equal to the exact arc displacement. However, the resulting curve still remains very close to an Euler spiral shape, provided that the z-axis step size is small and that the consecutive passes possess aligned entry and exit points and tangents. The G2-continuity of the transition curve is also maintained.

Creating a base curve $p_k(s)$ that aligns with a given layer, Layer k, can be accomplished in two steps: First, the shape $p_n(s)$ (shown in Figure 5-13a) is defined with Eqs. (5-30)-(5-32). This shape is then rotated and aligned with the end of Layer k (as $p_k(s)$), using Eq. (5-33), as illustrated in Figure 5-13b.

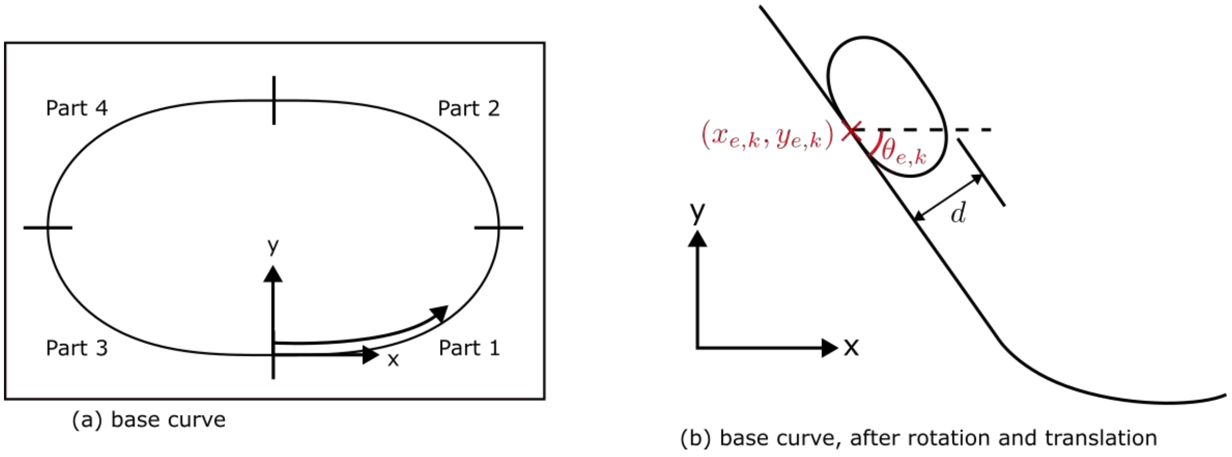


Figure 5-13: Base curve of layer transition curve, before and after rotation and alignment.

$$p_n(s) = \begin{bmatrix} x_n(s) \\ y_n(s) \end{bmatrix} \tag{5-30}$$

$$x_n(s) = \begin{cases} \frac{1}{a}\int_0^{as} \cos(u^2)\,du & 0 \le s < \frac{L}{4} \\ \frac{1}{a}\left[\int_{\frac{aL}{4}}^{as} \cos\left(-u^2 + aLu - \frac{a^2L^2}{8}\right)du + \int_0^{\frac{aL}{4}} \cos(u^2)\,du\right] & \frac{L}{4} \le s < \frac{L}{2} \\ \frac{1}{a}\int_{\frac{aL}{2}}^{as} \cos\left(u^2 - aLu + \frac{3a^2L^2}{8}\right)du & \frac{L}{2} \le s < \frac{3L}{4} \\ \frac{1}{a}\left[\int_{\frac{3aL}{4}}^{as} \cos\left(-u^2 + 2auL - \frac{3a^2L^2}{4}\right)du - \int_0^{\frac{aL}{4}} \cos(u^2)\,du\right] & \frac{3L}{4} \le s < L \end{cases} \tag{5-31}$$

$$y_n(s) = \begin{cases} \frac{1}{a}\int_0^{as} \sin(u^2)\,du & 0 \le s < \frac{L}{4} \\ \frac{1}{a}\left[\int_{\frac{aL}{4}}^{as} \sin\left(-u^2 + aLu - \frac{a^2L^2}{8}\right)du + \int_0^{\frac{aL}{4}} \sin(u^2)\,du\right] & \frac{L}{4} \le s < \frac{L}{2} \\ \frac{1}{a}\left[\int_{\frac{aL}{2}}^{as} \sin\left(u^2 - aLu + \frac{3a^2L^2}{8}\right)du + 2\int_0^{\frac{aL}{4}} \sin(u^2)\,du\right] & \frac{L}{2} \le s < \frac{3L}{4} \\ \frac{1}{a}\left[\int_{\frac{3aL}{4}}^{as} \sin\left(-u^2 + 2auL - \frac{3a^2L^2}{4}\right)du + \int_0^{\frac{aL}{4}} \sin(u^2)\,du\right] & \frac{3L}{4} \le s < L \end{cases} \tag{5-32}$$

$$p_k(s) = \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} \cos(\theta_{e,k}) & -\sin(\theta_{e,k}) \\ \sin(\theta_{e,k}) & \cos(\theta_{e,k}) \end{bmatrix} p_n(s) + \begin{bmatrix} x_{e,k} \\ y_{e,k} \end{bmatrix} \tag{5-33}$$

In the above expressions, the parameters a and L are defined based on the desired size of the transition. A clearance distance d must be specified, as shown in Figure 5-13b. From this, the minimum radius of curvature is computed in Eq. (5-34), followed by the scaling factor in Eq. (5-35), and the normalized length L' in Eq. (5-36), and the total toolpath length in Eq. (5-37). The empirical factor '2.75' in Eq. (5-34) was determined by trial and error, when investigating the use of an Euler spiral to develop a layer transition toolpath.

$$r \cong \frac{d}{2.75} \tag{5-34}$$

$$a = \frac{1}{r\sqrt{2\pi}} \tag{5-35}$$

$$L' = a\pi r \tag{5-36}$$

$$L = \frac{4L'}{a} \tag{5-37}$$

In the above equations, L represents the perimeter for traveling around the full base oval shape, once. Each of the four sections of this curve is an Euler spiral of the same shape, with the only differences being the position and orientation of the shape. If the constraint evaluation points are placed identically on all segments, Eqs. (5-31) and (5-32) may be solved only for the range of $0 \leq s < L/4$, and the remaining three segments can be generated by rotating and positioning exact copies of this segment.

As earlier mentioned, the base oval is rotated and positioned into alignment as shown in Figure 5-13b and Eq. (5-33). To generate the final transition curve, two versions of the base oval shape are created, which can be represented in arrays $[x_k, y_k]$ aligning with the end of Layer k, and $[x_{k+1}, y_{k+1}]$ aligning with the beginning of Layer $k+1$. A blending function is applied that transitions between these two oval shapes. This way, the blending curve is able to account both for the z-level change, and any small discrepancies in the x- and y-coordinate positions of each layer.

Before the blending function is applied, a certain portion of the transition toolpath is designated as 'lead-in' or 'lead-out' for which no movement in the z-axis is desired. This way, the cutting tool would be prevented from moving up or down (axially) while cutting a flat layer. To enforce the lead-in and lead-out behaviour, a buffer factor b is applied as in Eq. (5-38). This produces L_z – the parameter length for the vertical transition portion of the blending curve. The value of b can be assigned based on the cutting operation in question. As an example, when $b = 0.15$, 15% of the total curve is designated as lead-out, and 15% is designated as lead-in, leaving 70% for the actual vertical (z-axis) transition.

$$L_z = L(1 - 2b) \tag{5-38}$$

The local Euler spline parameter for the transition portion of the blending curve is designated in Eq. (5-39).

$$s_z = s - Lb \tag{5-39}$$

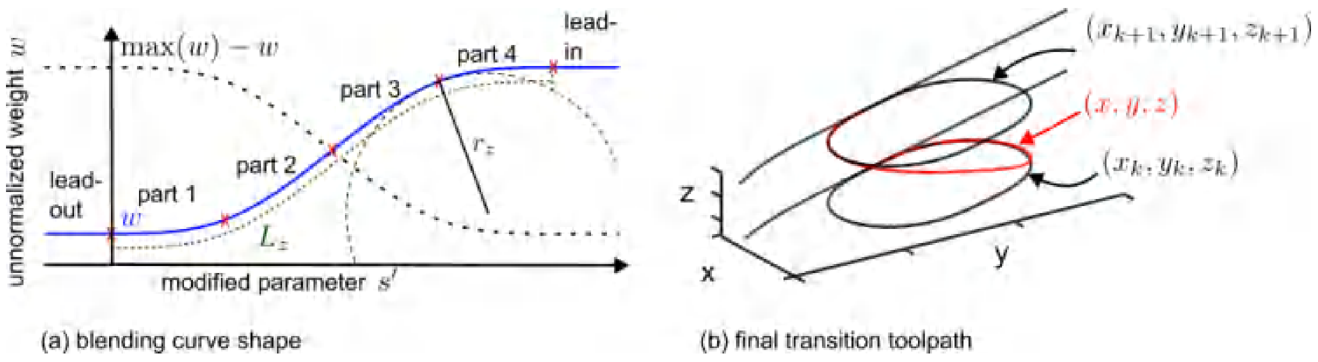


Figure 5-14: Blending function of Euler spiral layer transition.

Figure 5-14a shows the blending function as two parametric weights, to be applied on the two base oval functions. This curve, made up of three connecting Euler spirals, is represented by Eqs. (5-40) and (5-41), which are defined using the auxiliary functions detailed in (5-42) and (5-43). Similar to the base oval, it can be viewed as one single segment repeated four times, with applied rotations and translations, which may be used to reduce the number of numerical integrations that must be performed.

$$s'(s_z) = \begin{cases} f_{s',1}(s_z) & 0 \le s_z < \frac{L_z}{4} \text{ (part 1)} \\ f_{s',2}(s_z) + f_{s',1}\left(\frac{L_z}{4}\right) - f_{s',2}\left(\frac{L_z}{4}\right) & \frac{L_z}{4} \le s_z < \frac{3L_z}{4} \text{ (parts 2\&3)} \\ -f_{s',1}(-s_z + L_z) + 2\left(f_{s',2}\left(\frac{L_z}{2}\right) + f_{s',1}\left(\frac{L_z}{4}\right) - f_{s',2}\left(\frac{L_z}{4}\right)\right) & \frac{3L_z}{4} \le s_z < L_z \text{ (part 4)} \end{cases} \tag{5-40}$$

$$w(s_z) = \begin{cases} f_{w,1}(s_z) & 0 \le s_z < \frac{L_z}{4} \text{ (part 1)} \\ f_{w,2}(s_z) + f_{w,1}\left(\frac{L_z}{4}\right) - f_{w,2}\left(\frac{L_z}{4}\right) & \frac{L_z}{4} \le s_z < \frac{3L_z}{4} \text{ (parts 2\&3)} \\ -f_{w,1}(-s_z + L_z) + 2\left(f_{w,2}\left(\frac{L_z}{2}\right) + f_{w,1}\left(\frac{L_z}{4}\right) - f_{w,2}\left(\frac{L_z}{4}\right)\right) & \frac{3L_z}{4} \le s_z < L_z \text{ (part 4)} \end{cases} \tag{5-41}$$

These functions are defined in terms of individual Euler spiral segments in (5-42) and (5-43).

$$f_{s',1}(s_z) = \int_0^{s_z} \cos\left(\frac{2}{r_z L_z} u^2\right) du \tag{5-42}$$

$$f_{s',2}(s_z) = \int_0^{s_z} \cos\left(-\frac{2}{r_z L_z} u^2 + \frac{2}{r_z} u - \frac{L_z}{4 r_z}\right) du$$

$$f_{w,1}(s_z) = \int_0^{s_z} \sin\left(\frac{2}{r_z L_z} u^2\right) du \tag{5-43}$$
$$f_{w,2}(s_z) = \int_0^{s_z} \sin\left(-\frac{2}{r_z L_z} u^2 + \frac{2}{r_z} u - \frac{L_z}{4 r_z}\right) du$$

In order to compute these segments, a maximum radius of curvature must be assigned (r_z), which also influences the transition rate of the weighting function. By trial and error, it was found that using a value $r_z = 10L_z$ produced a suitable weighting function shape. After the curve is created, a factor is applied per Eq. (5-44) which scales the weighting range to be between 0 and 1.

$$\alpha = \frac{1}{w(4L_z)} \tag{5-44}$$

After the blending function is normalized, it is applied as a weighting factor which transitions between the previously generated base curves $[x_k, y_k, z_k]$ and $[x_{k+1}, y_{k+1}, z_{k+1}]$, aligning with the end of Layer k and beginning of Layer $k+1$, respectively. The z levels of each layer are represented as constant values z_k, and z_{k+1}. The blending function is scaled with the factor α, given in Eq. (5-44), after which the final curve can be computed with Eq. (5-45).

$$f(s_z) = \begin{bmatrix} x(s_z) \\ y(s_z) \\ z(s_z) \end{bmatrix} = \big(1 - \alpha w(s_z)\big) \begin{bmatrix} x_k\big(s'(s_z)\big) \\ y_k\big(s'(s_z)\big) \\ z_k \end{bmatrix} + \big(\alpha w(s_z)\big) \begin{bmatrix} x_{k+1}\big(s'(s_z)\big) \\ y_{k+1}\big(s'(s_z)\big) \\ z_{k+1} \end{bmatrix} \tag{5-45}$$

Figure 5-14b shows the two base curves, and the resulting transition toolpath after the weighting function is applied.

5.4.2 Sample trajectory using Euler spiral based layer transitions

Figure 5-15 depicts the toolpath generated for a star-shaped contouring operation with Euler spiral based layer transitions. They can be seen on the right-hand side of the figure, linking smoothly from one layer to the next. This toolpath has been considered in the case study presented here.

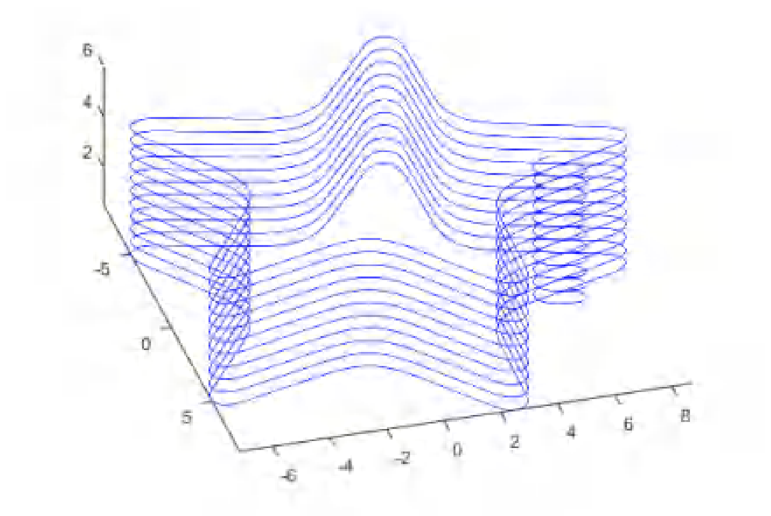


Figure 5-15: Layer transition on 11-layer contouring operation of star shape.

The layer transition curve has been integrated with the feed optimization. A scenario is considered in which the contour machining is to be conducted at a constant preassigned feedrate (for example, constrained due to the machining process specifications), and the feedrate optimization is performed only on the transition curves (to speed up the repositioning). To test the main idea, at this stage only LP-based feed optimization was used. Table 5-1 lists the constraints that were considered in the trajectory generation and optimization. It is customary in CNC toolpath programming for the entry portion into a contour machining operation to have a feedrate typically lower (e.g. by around 50%) compared to the cutting feed value. This is referred to as the 'lead-in' feedrate. In developing the simulation case study, to reflect industrial practice, a lead-in feedrate of 50% was also imposed in the 'lead-in' portion of the repositioning toolpath.

Table 5-1: Parameters used in the validating Euler spiral based layer transition.

Cutting feedrate $[mm/s]$	40
Maximum feed (lead-in) $[mm/s]$	20
Maximum feed during layer transition $[mm/s]$	(not limited)
Axis velocity limit $[mm/s]$	250
Axis acceleration limit $[mm/s^2]$	3000
Axis jerk limit $[mm/s^3]$	50000

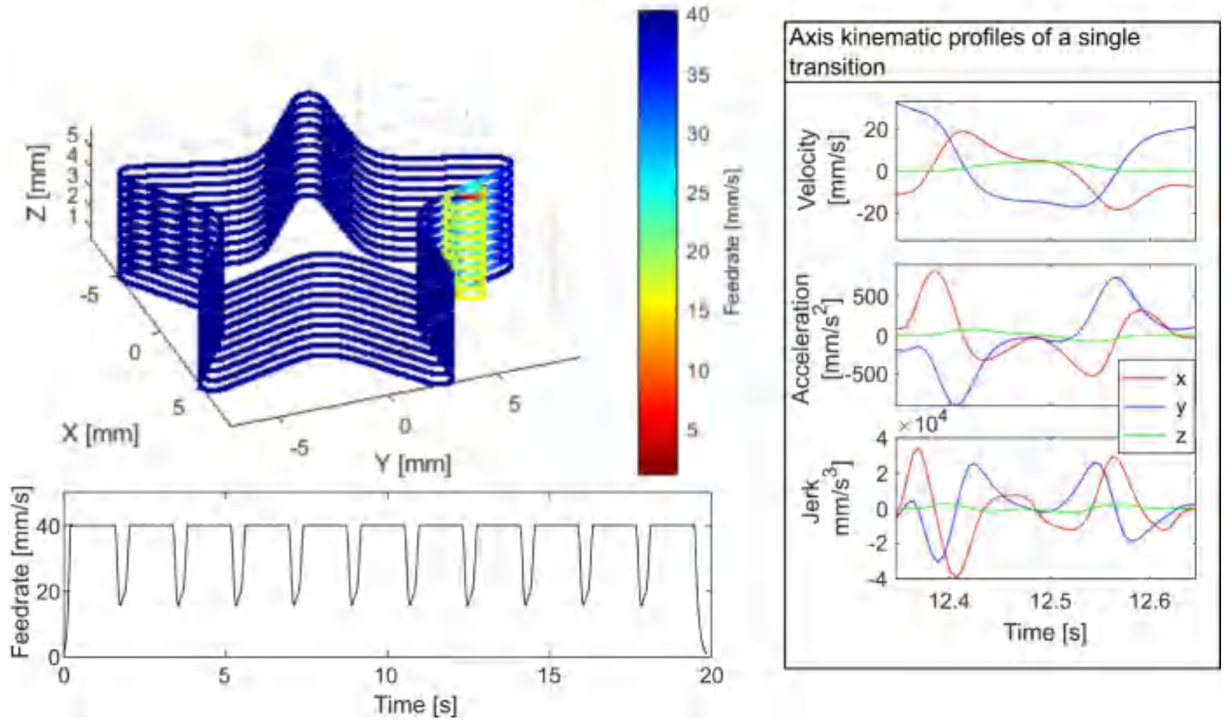


Figure 5-16: Optimized layer transition of 11-layer star-shaped contouring operation.

Figure 5-16 shows the completed contouring toolpath with layer transitions, color coded to represent feedrate. Each machining pass is visible as a section of constant feedrate. In between the cutting passes, the feed decreases for the smaller transition curves – both due to the smaller geometry of the curve, and to accommodate the enforced maximum lead-in feedrate, which was observed to be the active constraint. The deceleration is smooth and the feedrate does not decrease to zero or near-zero, as would be the case with a circular arc and corner based transition curve shown in Figure 5-11a. The axis-level velocity, acceleration, and jerk are shown for a typical single transition, and as can be seen, they remain within the assigned limits in Table 5-1.

In typical CNC programming, a circular arc transition toolpath like the one in Figure 5-11a is often traversed at constant feedrate. This can be improved with optimization, but still remains slower than the Euler spiral transition. Figure 5-17 presents a comparison between LP-based feed optimization performed on the Euler spiral transition and the circular arc transition (considering the same limits in Table 5-1). The trajectory for the circular arc toolpath has four noticeable slow-down points, each associated with a discontinuity: The two corner points at which the direction (i.e., tangent) is discontinuous, and the connection points out of and into the cutting portion, at which the curvature is discontinuous. The Euler spiral eliminates all of these discontinuities, and therefore does not require the tool to slow down unnecessarily. Unless imposed by geometry (e.g., very small radius of curvature in the transition), the feedrate does not have to slow down below the specified lead-in feed value. Thus, in comparison to arc-based transitioning, in this example the time required per tool positioning is decreased by 23%.

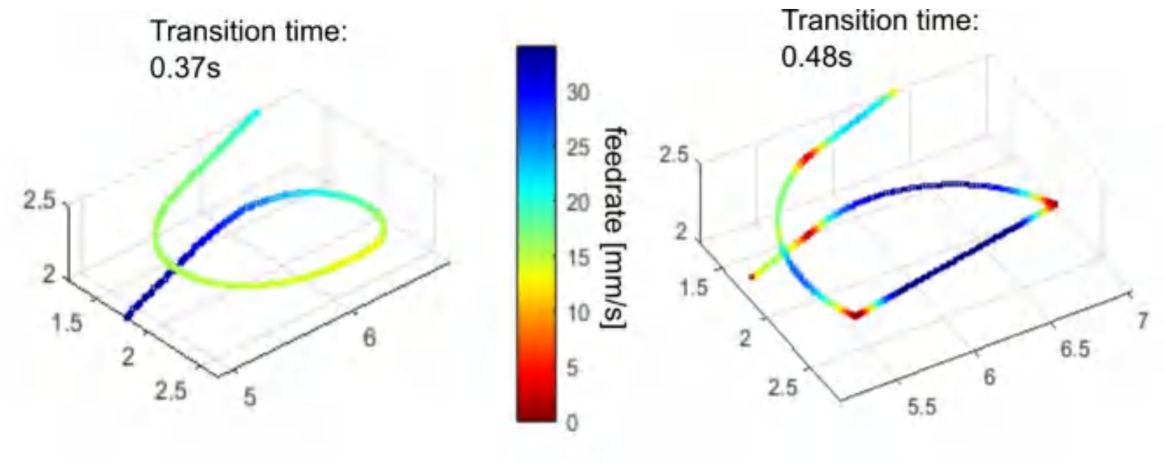


Figure 5-17: Comparison between Euler spiral based transition and circular arc based transition, with feed optimization.

5.5 Conclusions

In this chapter, two toolpath smoothing methods have been developed based on the Euler spiral. The first is an extension of a known Euler spiral 'pair' method from literature, implemented to work directly with the feedrate optimization method presented in Chapter 3. The Euler spiral pair method was developed and implemented in both 2D and 3D. The second is a method designed to reduce machining time for multi-level contouring operations, by enabling non-stop transitions between layers.

The Euler spiral pair method is advantageous as it generates an exactly arc-length parametrized curve which is continuous in first and second geometric derivatives of the axes. This has been implemented in a method in which the toolpath parameters are stored, and derivatives can be pre-computed for any given arc displacement. Euler spiral based position coordinates can be inefficient to compute, as there is no closed-form analytical solution and numerical integration is needed. By contrast, the axis geometric derivatives of Euler spirals are represented with simple formulae, which is a convenient feature they offer for direct calculation, in comparison to interpolating the toolpath ahead of time and computing the derivatives by numerical differentiation.

The layer transition toolpath is advantageous in machining multi-layer contouring toolpaths. While the machining portion remains unchanged, in comparison to using industry-standard circular arc-based repositioning, Euler spirals can achieve non-stop tool repositioning while the tool is not cutting, with shorter motion time and as well as lower acceleration and velocity requirements.

Future work for both methods would involve extending them into 5-axes. Prototype 5-axis smoothing methods were tested in development, but the combination of tool tip position and orientation profiles occasionally produced discontinuities or unwanted geometries.

6 Conclusions and Future Work

6.1 Conclusions

In this thesis, a novel feedrate scheduling method was developed combining linear programming and nonlinear programming optimization algorithms in a dual-windowed configuration. Linear programming, which is fast and robust, is used to generate a near-optimal starting guess of the feedrate profile, followed by nonlinear programming to refine the profile. Nonlinear optimization is more computationally expensive, but can directly constrain a wide variety of target responses which are nonlinear with respect to the feedrate. The linear programming solution is a near-optimal solution to the velocity, acceleration, and jerk constrained problem, which is used as a conservative means of limiting the servo tracking error and commanded motor torque for the trajectory. The nonlinear optimization step implements motor torque and tracking error constraints directly, allowing for shorter cycle times without exceeding the imposed maximums on these two responses. The window size and alignment strategies have been developed specifically for the dual-windowed strategy, in order to coordinate the boundary conditions of the two optimization algorithms, as non-matching constraints have the potential to produce an infeasible intermediate solution, which can cause the algorithm to fail.

Simulation case studies were generated that demonstrate the effectiveness of the algorithm, with a cycle time reduction of approximately 30% when compared with the LP-only solution, and the predicted motor torque and tracking error successfully constrained within the target envelope. Experimental case studies were implemented that showed promise, with the simulated trajectories being implemented smoothly, and the motor torque being successfully constrained. Accurately constraining the tracking error was more challenging, due to the difficulty of predicting this response for the used testbed (a flatbed router), which had significant nonlinearities, like friction, and low rigidity. Further experimental case studies are suggested on a modern, industrial CNC machine, as the tracking error prediction model has already been validated in literature.

Two toolpath smoothing methods were developed with the use of Euler spirals:

Euler spiral pair connections were used to generate a G2-continuous, exactly arc-length parametrized curves for smoothing sharp corners in the toolpath. The G2-continuity of the toolpath, when combined with a continuous feedrate profile, guarantees that the velocity and acceleration of the trajectory along that path are continuous and the jerk is bounded. The algorithm has been integrated with the LP+NLP feed optimization to reduce the overall computation time, taking advantage of the ease of computing geometric derivatives for Euler spirals without having to solve the toolpath itself. Once the feedrate is optimized and the feed profile and toolpath are ready to be interpolated, numerical integration is used for determining the position profiles on the Euler spiral. For this purpose, Gauss-Kronrod quadrature method was determined to be suitable.

An Euler spiral based layer-transition toolpath was designed for multilevel contouring toolpaths. This method replaces the circular arc based repositioning segments, thus achieving smooth and uninterrupted connection between the layers. Euler spiral based layer transitioning requires less motion time compared to circular arc-based connections. Since tangent and curvature discontinuities are avoided, the required axis velocity and acceleration commands are also smaller in magnitude.

The novel contributions of the thesis are as follows:

- A novel dual-windowing LP+NLP feedrate scheduling algorithm, in which two optimization algorithms are combined into a forward-looking windowed solution. This algorithm is able to generate optimized trajectories with motor torque and tracking error constrained directly, which gives lower cycle times than the velocity, acceleration, and jerk constrained optimization. This algorithm was verified in simulation and experiment.
- A toolpath corner smoothing method using Euler spiral pairs, which has been integrated in a novel manner with the proposed LP+NLP feedrate optimization. Taking advantage of the derivative property of Euler spirals, the method computes the geometric derivatives for the constraint checkpoints without requiring computationally expensive numerical integration for computing the checkpoint coordinates. The latter is normally required when working with other kinds of splines (e.g. B-spline, polynomial, etc.).
- An Euler-spiral based layer transition toolpath to reduce machining time in multi-layer contouring. This method has also been integrated with feed optimization. Compared to the industrial practice of planning lead-in and lead-out movements via in-layer circular arcs, the proposed transition continuously modulates the tangent and curvature, while integrating smooth z-axis motion. This results in quicker movement while requiring less velocity and acceleration from the drives.

6.2 Future work

The research in this thesis opens several avenues for future work. Some of the topics recommended for further research are:

- Continuous and streaming implementation: The developed feed optimization algorithm has a structure amenable to continuous processing of indefinitely long toolpaths. However, the current implementation in Matlab utilizes large-scale arrays for recording and displaying the final optimization profile outputs. In future work, these can be removed so the output is written directly to a file or a continuous buffer.
- Using different basis functions: The feed optimization in this thesis considered B-splines as the fundamental curve to modulate. In the future, other options, such as radial basis functions and wavelets can also be investigated.

- Variable-step meshing of control and constraint evaluation points in feed optimization: Using fixed distance meshing, which is adjusted to consider only the smallest feature size, can be conservative if many of the toolpath segments are actually larger. One possible solution could be to split each toolpath segment, typically defined by a cubic B-spline, into a fixed minimum number of control points and constraint checkpoints. Larger features and toolpath segments, which have less abrupt curvature changes, would thus utilize 'as required' feed modulation and constraint checkpoints, rather than a high density of fixed mesh points. However, this would mean that the 'B' matrix structure in Section 3.3.5 would not be constant anymore, but would need to be updated regularly in the algorithm. It is worth investigating whether this modification can yield further computational efficiency, especially when handling long toolpaths with segments that vary substantially in size with respect to one another (e.g., 1:10 or 1:100).
- Extension of the algorithm to 5-axis: The feed optimization and toolpath smoothing algorithms presented in this thesis can be extended to work with position and orientation vectors of the tool. 5-axis machining is becoming increasingly important in industrial CNC, and is essential for producing complex parts in many applications, like aerospace, biomedical, and automotive. Extending the algorithms to 5 axes would broaden the range of machining operations that can be improved for efficiency and quality.
- Use of a multibody model: The current optimization algorithm considers each axis to be independent of one another. However, in the case of 5-axis machining, the inertial coupling between axes can have a large effect on the dynamic response of the machine, which must be accurately modeled and considered. The dynamic response of 5-axis machine tools, especially containing direct drive rotary axes, are highly nonlinear, comprising of multibody coupling effects, friction, and position-dependent disturbances (e.g., torque ripple). Therefore, nonlinear optimization is most suitable in handling them.
- Integrating feed optimization with machine learning (ML): The current feed scheduling method developed is based on classical optimization methods, which can take a considerable time to converge. Future research should consider utilizing ML to speed up the feed optimization, where ML can be trained to detect patterns and yield near-optimum profiles based on the optimization output for given axis derivative profiles and constraints. Then, a trained ML can be used to potentially replace parts of the feed optimization (such as the initial guess generation via LP), or maybe to perform the optimization altogether (with some safety and constraint compliance checks, as applicable).
- Position dependent modulation of constraint boundaries: In certain 3D machining applications, some passes require only low positioning tolerance, whereas other portions of the toolpath may need tight tolerances. To take advantage of the opportunities to 'speed-up' the manufacturing throughput when

possible, while ensuring compliance in the necessary portions for contour machining, the constraint boundaries considered in the feed optimization can be made path dependent.

- Integrating toolpath and feedrate optimization into a single simultaneous optimization algorithm: Allowing the feedrate and toolpath modifications to occur within the same optimization routine would enable further reduction in motion time. It was observed by the author, while conducting this thesis, that minute changes to the toolpath geometry can facilitate smoother geometric derivatives, allowing an increase in the feedrate profile. To explore the benefit of simultaneous toolpath and feed optimization, developing an algorithm first to work on short one-shot toolpaths, and then extending the approach into a windowed solution, similar to the one presented in this thesis, is recommended for future research.
- Extension to other processes: In the future, the methodologies in this thesis can also be extended to processes other than machining, such as laser cutting, additive manufacturing (deposition), and freeform surface polishing, which also require contour tracing. In such processes, feedrate may need to be limited with both lower and upper bounds, as a function of arc displacement, which would be incorporated into the optimization constraints.

Letters of Copyright permission

License Number	5724920709271
License date	Feb 09, 2024
Licensed Content Publisher	Elsevier
Licensed Content Publication	Precision Engineering
Licensed Content Title	Accurate control of ball screw drives using pole-placement vibration damping and a novel trajectory prefilter
Licensed Content Author	Dan J. Gordon,Kaan Erkorkmaz
Licensed Content Date	Apr 1, 2013
Licensed Content Volume	37
Licensed Content Issue	2
Licensed Content Pages	15
Start Page	308
End Page	322
Type of Use	reuse in a thesis/dissertation
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	2
Format	both print and electronic
Are you the author of this Elsevier article?	No
Will you be translating?	No
Title of new work	Smooth and Time-Optimal Trajectory Planning for Multi-Axis Machine Tools
Institution name	University of Waterloo
Expected presentation date	Feb 2024
Portions	Figure 2, page 310 Figure 7, page 313
Requestor Location	Katharine DiCola Attn: Katharine DiCola
Publisher Tax ID	GB 494 6272 12
Total	0.00 CAD

ELSEVIER LICENSE
TERMS AND CONDITIONS

Feb 13, 2024

This Agreement between Katharine DiCola ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

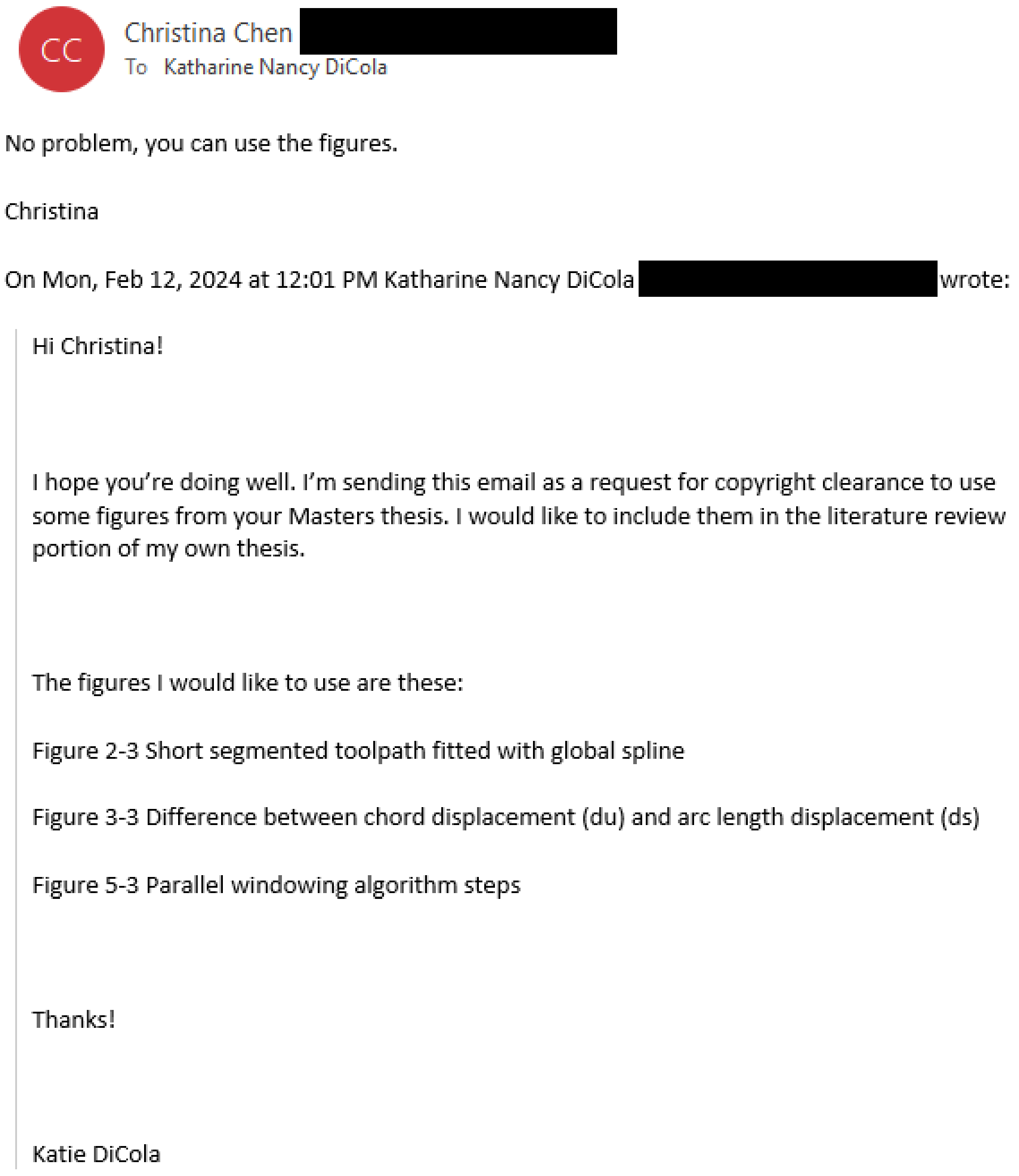
License Number	5724920454397
License date	Feb 09, 2024
Licensed Content Publisher	Elsevier
Licensed Content Publication	CIRP Annals - Manufacturing Technology
Licensed Content Title	Feedrate optimization for freeform milling considering constraints from the feed drive system and process mechanics
Licensed Content Author	Kaan Erkorkmaz,S. Ehsan Layegh,Ismail Lazoglu,Huseyin Erdim
Licensed Content Date	Jan 1, 2013
Licensed Content Volume	62
Licensed Content Issue	1
Licensed Content Pages	4
Start Page	395
End Page	398
Type of Use	reuse in a thesis/dissertation
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	1
Format	both print and electronic
Are you the author of this Elsevier article?	No
Will you be translating?	No
Title of new work	Smooth and Time-Optimal Trajectory Planning for Multi-Axis Machine Tools
Institution name	University of Waterloo
Expected presentation date	Feb 2024
Portions	Figure 7, page 398
Requestor Location	Katharine DiCola Attn: Katharine DiCola
Publisher Tax ID	GB 494 6272 12
Total	0.00 CAD

ELSEVIER LICENSE
TERMS AND CONDITIONS

Feb 13, 2024

This Agreement between Katharine DiCola ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

License Number	5724920163294
License date	Feb 09, 2024
Licensed Content Publisher	Elsevier
Licensed Content Publication	CIRP Annals - Manufacturing Technology
Licensed Content Title	Linear programming and windowing based feedrate optimization for spline toolpaths
Licensed Content Author	Kaan Erkorkmaz,Qing-Ge (Christina) Chen,Ming-Yong Zhao,Xavier Beudaert,Xiao-Shan Gao
Licensed Content Date	Jan 1, 2017
Licensed Content Volume	66
Licensed Content Issue	1
Licensed Content Pages	4
Start Page	393
End Page	396
Type of Use	reuse in a thesis/dissertation
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	1
Format	both print and electronic
Are you the author of this Elsevier article?	No
Will you be translating?	No
Title of new work	Smooth and Time-Optimal Trajectory Planning for Multi-Axis Machine Tools
Institution name	University of Waterloo
Expected presentation date	Feb 2024
Portions	Figure 2, page 395
Requestor Location	Katharine DiCola Attn: Katharine DiCola
Publisher Tax ID	GB 494 6272 12
Total	0.00 CAD



CC

Christina Chen [redacted]
To Katharine Nancy DiCola

No problem, you can use the figures.

Christina

On Mon, Feb 12, 2024 at 12:01 PM Katharine Nancy DiCola [redacted] wrote:

> Hi Christina!
>
> I hope you're doing well. I'm sending this email as a request for copyright clearance to use some figures from your Masters thesis. I would like to include them in the literature review portion of my own thesis.
>
> The figures I would like to use are these:
>
> Figure 2-3 Short segmented toolpath fitted with global spline
>
> Figure 3-3 Difference between chord displacement (du) and arc length displacement (ds)
>
> Figure 5-3 Parallel windowing algorithm steps
>
> Thanks!
>
> Katie DiCola

References

[1] W. Pease, “An Automatic Machine Tool,” *Scientific American*, vol. 187, no. 3, pp. 101–115, 1952.

[2] M. E. Kahn and B. Roth, “The near-minimum-time control of open-loop articulated kinematic chains,” 1971.

[3] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, “Time-optimal control of robotic manipulators along specified paths,” *The international journal of robotics research*, vol. 4, no. 3, Art. no. 3, 1985.

[4] C. Q. G. Chen, “Time-Optimal Feedrate Planning for Freeform Toolpaths for Manufacturing Applications,” Master’s Thesis, University of Waterloo, 2018.

[5] J. Huan, “Direct Spline Interpolation of CNC-Machine Tool,” *IFAC Proceedings Volumes*, vol. 18, no. 9, Art. no. 9, Aug. 1985, doi: 10.1016/S1474-6670(17)60297-0.

[6] J. Yang and A. Yuen, “An analytical local corner smoothing algorithm for five-axis CNC machining,” *International Journal of Machine Tools and Manufacture*, vol. 123, pp. 22–35, Dec. 2017, doi: 10.1016/j.ijmachtools.2017.07.007.

[7] K. Erkorkmaz and Y. Altintas, “Quintic Spline Interpolation With Minimal Feed Fluctuation,” *Journal of Manufacturing Science and Engineering*, vol. 127, no. 2, Art. no. 2, Apr. 2005, doi: 10.1115/1.1830493.

[8] F.-C. Wang and D. C. H. Yang, “Nearly arc-length parameterized quintic-spline interpolation for precision machining,” *Computer-Aided Design*, vol. 25, no. 5, Art. no. 5, May 1993, doi: 10.1016/0010-4485(93)90085-3.

[9] K. Erkorkmaz, C.-H. Yeung, and Y. Altintas, “Virtual CNC system. Part II. High speed contouring application,” *International Journal of Machine Tools and Manufacture*, vol. 46, no. 10, pp. 1124–1138, Aug. 2006, doi: 10.1016/j.ijmachtools.2005.08.001.

[10] 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,” *International Journal of Machine Tools and Manufacture*, vol. 65, pp. 88–98, Feb. 2013, doi: 10.1016/j.ijmachtools.2012.10.005.

[11] 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,” *The International Journal of Advanced Manufacturing Technology*, vol. 76, no. 9–12, Art. no. 9–12, 2015.

[12] C. Lartigue, C. Tournier, M. Ritou, and D. Dumur, “High-Performance NC for HSM by means of Polynomial Trajectories,” *CIRP Annals*, vol. 53, no. 1, Art. no. 1, Jan. 2004, doi: 10.1016/S0007-8506(07)60706-9.

[13] B. Sencer, Y. Altintas, and E. Croft, “Feed optimization for five-axis CNC machine tools with drive constraints,” *International Journal of Machine Tools and Manufacture*, vol. 48, no. 7, Art. no. 7, Jun. 2008, doi: 10.1016/j.ijmachtools.2008.01.002.

[14] C. Okwudire, K. Ramani, and M. Duan, “A trajectory optimization method for improved tracking of motion commands using CNC machines that experience unwanted vibration,” *CIRP Annals*, vol. 65, no. 1, Art. no. 1, Jan. 2016, doi: 10.1016/j.cirp.2016.04.100.

[15] W. Fan, X.-S. Gao, C.-H. Lee, K. Zhang, and Q. Zhang, “Time-optimal interpolation for five-axis CNC machining along parametric tool path based on linear programming,” *Int J Adv Manuf Technol*, vol. 69, no. 5, Art. no. 5, Nov. 2013, doi: 10.1007/s00170-013-5083-x.

[16] Q. Zhang, S. Li, J.-X. Guo, and X.-S. Gao, “Time-optimal path tracking for robots under dynamics constraints based on convex optimization,” *Robotica*, vol. 34, no. 9, Art. no. 9, Sep. 2016, doi: 10.1017/S0263574715000247.

[17] K. Erkorkmaz, Q.-G. (Christina) Chen, M.-Y. Zhao, X. Beudaert, and X.-S. Gao, “Linear programming and windowing based feedrate optimization for spline toolpaths,” *CIRP Annals*, vol. 66, no. 1, Art. no. 1, Jan. 2017, doi: 10.1016/j.cirp.2017.04.058.

[18] H. Liu, Q. Liu, S. Zhou, C. Li, and S. Yuan, “A NURBS interpolation method with minimal feedrate fluctuation for CNC machine tools,” *The International Journal of Advanced Manufacturing Technology*, vol. 78, no. 5–8, Art. no. 5–8, 2015.

[19] M. Heng and K. Erkorkmaz, “Design of a NURBS interpolator with minimal feed fluctuation and continuous feed modulation capability,” *International Journal of Machine Tools and Manufacture*, vol. 50, no. 3, pp. 281–293, Mar. 2010, doi: 10.1016/j.ijmachtools.2009.11.005.

[20] 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, Art. no. 5, Jul. 2016, doi: 10.1007/s00170-015-7969-2.

[21] X.-R. Wang, Z.-Q. Wang, Y.-S. Wang, T.-S. Lin, and P. He, “A bisection method for the milling of NURBS mapping projection curves by CNC machines,” *The International Journal of Advanced Manufacturing Technology*, vol. 91, no. 1–4, Art. no. 1–4, 2017.

[22] S. Tajima, B. Sencer, and E. Shamoto, “Accurate interpolation of machining tool-paths based on FIR filtering,” *Precision Engineering*, vol. 52, pp. 332–344, Apr. 2018, doi: 10.1016/j.precisioneng.2018.01.016.

[23] S. Tajima and B. Sencer, “Accurate real-time interpolation of 5-axis tool-paths with local corner smoothing,” *International Journal of Machine Tools and Manufacture*, vol. 142, pp. 1–15, Jul. 2019, doi: 10.1016/j.ijmachtools.2019.04.005.

[24] K. Erkorkmaz, “Efficient Fitting of the Feed Correction Polynomial for Real-Time Spline Interpolation,” *Journal of Manufacturing Science and Engineering*, vol. 137, no. 4, Art. no. 4, Aug. 2015, doi: 10.1115/1.4030300.

[25] M. Chen and Y. Sun, “Contour error–bounded parametric interpolator with minimum feedrate fluctuation for five-axis CNC machine tools,” *Int J Adv Manuf Technol*, vol. 103, no. 1–4, pp. 567–584, Jul. 2019, doi: 10.1007/s00170-019-03586-5.

[26] R. Levien, “The Euler spiral: a mathematical history,” p. 16.

[27] J. STOER, “Curve Fitting With Clothoidal Splines,” *Journal of Research of the National Bureau of Standards*, vol. 87, no. 4, pp. 317–346, 1982.

[28] Y. Kanayama, “Trajectory generation for mobile robots,” *Int. J. Robotics Res*, vol. 3, pp. 333–340, 1986.

[29] 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 Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 735–745, Dec. 1990, doi: 10.1109/70.63274.

[30] D. S. Meek and D. J. Walton, “Clothoid spline transition spirals,” *Math. Comp.*, vol. 59, no. 199, pp. 117–133, 1992, doi: 10.1090/S0025-5718-1992-1134736-8.

[31] D. J. Walton and D. S. Meek, “A controlled clothoid spline,” *Computers & Graphics*, vol. 29, no. 3, pp. 353–363, Jun. 2005, doi: 10.1016/j.cag.2005.03.008.

[32] A. Shahzadeh, A. Khosravi, and S. Nahavandi, “Path Planning for CNC Machines Considering Centripetal Acceleration and Jerk,” in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2013, pp. 1759–1764. doi: 10.1109/SMC.2013.303.

[33] A. Shahzadeh, A. Khosravi, T. Robinette, and S. Nahavandi, “Smooth path planning using biclothoid fillets for high speed CNC machines,” *International Journal of Machine Tools and Manufacture*, vol. 132, pp. 36–49, Sep. 2018, doi: 10.1016/j.ijmachtools.2018.04.003.

[34] Q.-B. Xiao, M. Wan, Y. Liu, X.-B. Qin, and W.-H. Zhang, “Space corner smoothing of CNC machine tools through developing 3D general clothoid,” *Robotics and Computer-Integrated Manufacturing*, vol. 64, p. 101949, Aug. 2020, doi: 10.1016/j.rcim.2020.101949.

[35] R. V. Fleisig and A. D. Spence, “A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining,” *Computer-Aided Design*, vol. 33, no. 1, Art. no. 1, Jan. 2001, doi: 10.1016/S0010-4485(00)00049-X.

[36] K. Erkorkmaz and Y. Altintas, “High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation,” *International Journal of Machine Tools and Manufacture*, vol. 41, no. 9, Art. no. 9, Jul. 2001, doi: 10.1016/S0890-6955(01)00002-5.

[37] K. Kong, H. C. Kniep, and M. Tomizuka, “Output Saturation in Electric Motor Systems: Identification and Controller Design,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 132, no. 051002, Aug. 2010, doi: 10.1115/1.4001792.

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

[39] D. J. Gordon and K. Erkorkmaz, “Accurate control of ball screw drives using pole-placement vibration damping and a novel trajectory prefilter,” *Precision Engineering*, vol. 37, no. 2, Art. no. 2, Apr. 2013, doi: 10.1016/j.precisioneng.2012.09.009.

[40] H. Li *et al.*, “A novel feedrate scheduling method based on Sigmoid function with chord error and kinematic constraints,” *Int J Adv Manuf Technol*, vol. 119, no. 3, pp. 1531–1552, Mar. 2022, doi: 10.1007/s00170-021-08092-1.

[41] S. Tajima and B. Sencer, “Real-time trajectory generation for 5-axis machine tools with singularity avoidance,” *CIRP Annals*, vol. 69, no. 1, pp. 349–352, 2020.

[42] S. Tajima and B. Sencer, “Global tool-path smoothing for CNC machine tools with uninterrupted acceleration,” *International Journal of Machine Tools and Manufacture*, vol. 121, pp. 81–95, Oct. 2017, doi: 10.1016/j.ijmachtools.2017.03.002.

[43] Y. Altintas and K. Erkorkmaz, “Feedrate Optimization for Spline Interpolation In High Speed Machine Tools,” *CIRP Annals*, vol. 52, no. 1, pp. 297–302, Jan. 2003, doi: 10.1016/S0007-8506(07)60588-5.

[44] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Time-Optimal Path Tracking for Robots: A Convex Optimization Approach,” *IEEE Transactions on Automatic Control*, vol. 54, no. 10, Art. no. 10, Oct. 2009, doi: 10.1109/TAC.2009.2028959.

[45] D. E. Kirk, *Optimal Control Theory: An Introduction*. Courier Corporation, 2012.

[46] K. Erkorkmaz, S. E. Layegh, I. Lazoglu, and H. Erdim, “Feedrate optimization for freeform milling considering constraints from the feed drive system and process mechanics,” *CIRP Annals*, vol. 62, no. 1, Art. no. 1, Jan. 2013, doi: 10.1016/j.cirp.2013.03.084.

[47] T.-Y. Kim and J. Kim, “Adaptive cutting force control for a machining center by using indirect cutting force measurements,” *International Journal of Machine Tools and Manufacture*, vol. 36, no. 8, Art. no. 8, Aug. 1996, doi: 10.1016/0890-6955(96)00097-1.

[48] F. Ridwan, X. Xu, and F. C. L. Ho, “Adaptive execution of an NC program with feed rate optimization,” *Int J Adv Manuf Technol*, vol. 63, no. 9, Art. no. 9, Dec. 2012, doi: 10.1007/s00170-012-3959-9.

[49] K. Xu and K. Tang, “Five-axis tool path and feed rate optimization based on the cutting force–area quotient potential field,” *Int J Adv Manuf Technol*, vol. 75, no. 9, Art. no. 9, Dec. 2014, doi: 10.1007/s00170-014-6221-9.

[50] H.-Y. Feng and N. Su, “Integrated tool path and feed rate optimization for the finishing machining of 3D plane surfaces,” *International Journal of Machine Tools and Manufacture*, vol. 40, no. 11, Art. no. 11, Sep. 2000, doi: 10.1016/S0890-6955(00)00025-0.

[51] W. B. Ferry and Y. Altintas, “Virtual Five-Axis Flank Milling of Jet Engine Impellers—Part I: Mechanics of Five-Axis Flank Milling,” *Journal of Manufacturing Science and Engineering*, vol. 130, no. 1, Art. no. 1, Jan. 2008, doi: 10.1115/1.2815761.

[52] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[53] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[54] G. B. Dantzig, “Origins of the simplex method,” in *A history of scientific computing*, S. G. Nash, Ed., New York, NY, USA: ACM, 1990, pp. 141–151. doi: 10.1145/87252.88081.

[55] F. A. Potra and S. J. Wright, “Interior-point methods,” *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 281–302, Dec. 2000, doi: 10.1016/S0377-0427(00)00433-7.

[56] P. T. Boggs and J. W. Tolle, “Sequential Quadratic Programming *,” *Acta Numerica*, vol. 4, pp. 1–51, Jan. 1995, doi: 10.1017/S0962492900002518.

[57] "Constrained Nonlinear Optimization Algorithms - MATLAB & Simulink." Accessed: Oct. 23, 2023. [Online]. Available: https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

[58] A.-C. Lee, M.-T. Lin, Y.-R. Pan, and W.-Y. Lin, "The feedrate scheduling of NURBS interpolator for CNC machine tools," *Computer-Aided Design*, vol. 43, no. 6, pp. 612–628, 2011.

[59] K. Erkorkmaz and M. Heng, "A heuristic feedrate optimization strategy for NURBS toolpaths," *CIRP annals*, vol. 57, no. 1, pp. 407–410, 2008.

[60] J. H. Gallier, *Curves and surfaces in geometric modeling: theory and algorithms*. Morgan Kaufmann, 2000.

[61] K. Höllig and J. Hörner, *Approximation and modeling with B-splines*. SIAM, 2013.

[62] C. De Boor, *A practical guide to splines*, vol. 27. springer-verlag New York, 1978.

[63] L. Ljung, *System Identification: Theory for the User*. Prentice Hall PTR, 1999.

[64] G. W. G. Tseng, C. Q. G. Chen, K. Erkorkmaz, and S. Engin, "Digital shadow identification from feed drive structures for virtual process planning," *CIRP Journal of Manufacturing Science and Technology*, vol. 24, pp. 55–65, Jan. 2019, doi: 10.1016/j.cirpj.2018.11.002.

[65] R. M. Corless and N. Fillion, *A Graduate Introduction to Numerical Methods: From the Viewpoint of Backward Error Analysis*. New York, NY: Springer New York, 2013. doi: 10.1007/978-1-4614-8453-0.

A. Appendices

A.1 Derivation of uniform quadratic and cubic B-spline equations and their derivatives

This appendix presents the mathematical derivations of the B-spline equations used in the main optimization algorithm. Note that equation numbers will be used only for major equations and not intermediary derivation steps.

The resulting quadratic B-spline equation is given in Eq. (A.1-6), and its first and second derivatives are given in Eqs. (A.1-8) and (A.1-9) respectively. The resulting cubic B-spline equation is given in Eq. (A.1-7), and its first and second derivatives are given in Eqs. (A.1-10) and (A.1-11) respectively.

A.1.1 Starting Equations

De Boor's algorithm for B-spline computation is given in Eqs. (A.1-1), and (A.1-2). The starting point is the characteristic function defined in Eq. (A.1-3), which is also the definition of a first-order B-spline.

$$b_{i,\xi}^{n}(s) = \gamma_{i,\xi}^{n}(s) \cdot b_{i,\xi}^{n-1}(s) + \left(1 - \gamma_{i+1,\xi}^{n}(s)\right) \cdot b_{i+1,\xi}^{n-1}(s) \tag{A.1-1}$$

$$\gamma_{i,\xi}^{n}(s) = \frac{s - \xi_i}{\xi_{i+n-1} - \xi_i} \tag{A.1-2}$$

$$b_{i,\xi}^{1}(s) = \begin{cases} 1 & for\ \xi_i \le s < \xi_{i+1} \\ 0 & otherwise \end{cases} \tag{A.1-3}$$

The notation $b_{i,\xi}^{n}$ represents the ith B-spline, of order n, with knot profile $\xi = [\xi_1, \xi_2, \dots, \xi_k]$. The knot profile contains k knots in total.

Additionally, because of the enforced uniform spacing, the knot positions can be defined in terms of a constant knot separation distance d, as described in Eq. (A.1-4).

$$\xi_i = di \tag{A.1-4}$$

A.1.2 Second order B-spline

$$b_i^2 = \gamma_i^2 b_i^1 + \left(1 - \gamma_{i+1}^2\right) b_{i+1}^1$$

$$b_i^1(s) = \begin{cases} \gamma_i^2 & for\ di \le x < d(i+1) \\ \left(1 - \gamma_{i+1}^2\right) & for\ d(i+1) \le x < d(i+2) \\ 0 & otherwise \end{cases}$$

For which we can compute γ:

$$\gamma_i^2(s) = \frac{s - di}{d}$$

The result is:

$$b_i^2(s) = \begin{cases} \dfrac{s - di}{d} & for\ di \le s < d(i+1) \\ \dfrac{d(i+2) - s}{d} & for\ d(i+1) \le x < d(i+2) \\ 0 & otherwise \end{cases} \tag{A.1-5}$$

A.1.3 Third order B-spline

$b_i^3 = \gamma_i^3 b_i^2 + (1-\gamma_{i+1}^3) b_{i+1}^2$

First, we can fill in the pieces of b_i^2 and b_{i+1}^2 from Eq. (A.1-5)

$$b_i^3(s) = \begin{cases} \gamma_i^3\left(\frac{s-di}{d}\right) & for\ di \le s < d(i+1) \\ \gamma_i^3\left(\frac{d(i+2)-s}{d}\right) + (1-\gamma_{i+1}^3)\left(\frac{s-d(i+1)}{d}\right) & for\ d(i+1) \le s < d(i+2) \\ (1-\gamma_{i+1}^3)\left(\frac{d(i+3)-s}{d}\right) & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases}$$

And compute γ

$$\gamma_i^3(s) = \frac{s-di}{d(i+2)-di} = \frac{s-di}{2d}$$

The result is:

$$b_i^3(s) = \begin{cases} \left(\frac{s-di}{2d}\right)\left(\frac{s-di}{d}\right) & for\ di \le s < d(i+1) \\ \left(\frac{s-di}{2d}\right)\left(\frac{d(i+2)-s}{d}\right) + \left(\frac{-s+d(i+3)}{2d}\right)\left(\frac{s-d(i+1)}{d}\right) & for\ d(i+1) \le s < d(i+2) \\ \left(\frac{-s+d(i+3)}{2d}\right)\left(\frac{d(i+3)-s}{d}\right) & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases}$$

Which simplifies to:

$$b_i^3(s) = \begin{cases} \frac{(s-di)^2}{2d^2} & for\ di \le s < d(i+1) \\ \frac{-2s^2 + sd(4i+6) - d^2[2i^2+6i+3]}{2d^2} & for\ d(i+1) \le s < d(i+2) \\ \frac{(d(i+3)-s)^2}{2d^2} & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases} \qquad \text{(A.1-6)}$$

A.1.4 Fourth order B-spline

$$b_i^4 = \gamma_i^4 b_i^3 + (1-\gamma_{i+1}^4) b_{i+1}^3$$

$$b_i^4(s) = \begin{cases} \gamma_i^4\left(\frac{(s-di)^2}{2d^2}\right) & for\ di \le s < d(i+1) \\ \gamma_i^4\left(\frac{-2s^2+sd(4i+6)-d^2[2i^2+6i+3]}{2d^2}\right) + (1-\gamma_{i+1}^4)\left(\frac{(s-d(i+1))^2}{2d^2}\right) & for\ d(i+1) \le s < d(i+2) \\ \gamma_i^4\left(\frac{(d(i+3)-s)^2}{2d^2}\right) + (1-\gamma_{i+1}^4)\left(\frac{-2s^2+sd(4(i+1)+6)-d^2[2(i+1)^2+6(i+1)+3]}{2d^2}\right) & for\ d(i+2) \le s < d(i+3) \\ (1-\gamma_{i+1}^4)\left(\frac{(d(i+4)-s)^2}{2d^2}\right) & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases}$$

First, γ is computed:

$$\gamma_i^4(s) = \frac{s - di}{d(i+3) - di} = \frac{s - di}{3d}$$

$$\left(1 - \gamma_{i+1}^4\right) = 1 - \frac{s - d(i+1)}{3d} = \frac{3d - s + d(i+1)}{3d} = \frac{d(4+i) - s}{3d}$$

Due to complexity, each of the four polynomial pieces is solved outside the large equation.

First polynomial piece:

$$\gamma_i^4\left(\frac{(s-di)^2}{2d^2}\right) = \left(\frac{s-di}{3d}\right)\left(\frac{(s-di)^2}{2d^2}\right) = \frac{(s-di)^3}{6d^3}$$

Second polynomial piece:

$$\gamma_i^4\left(\frac{-2s^2 + sd(4i+6) - d^2[2i^2+6i+3]}{2d^2}\right) + \left(1 - \gamma_{i+1}^4\right)\left(\frac{\left(s - d(i+1)\right)^2}{2d^2}\right)$$

Expansion of first half of second polynomial piece:

$$\left(\frac{s-di}{3d}\right)\left(\frac{-2s^2 + sd(4i+6) - d^2[2i^2+6i+3]}{2d^2}\right)$$

$$= \left(\frac{1}{6d^3}\right)(s-di)(-2s^2 + sd(4i+6) - d^2[2i^2+6i+3])$$

$$= \left(\frac{-2s^3 + 6ds^2(i+1) - d^2s(6i^2+12i+3) + (d^3i[2i^2+6i+3])}{6d^3}\right)$$

Expansion of second half of second polynomial piece:

$$\left(\frac{d(4+i) - s}{3d}\right)\left(\frac{\left(s - d(i+1)\right)^2}{2d^2}\right)$$

$$= \left(\frac{1}{6d^3}\right)\left(-s + d(4+i)\right)\left(s - d(i+1)\right)^2$$

$$= \left(\frac{1}{6d^3}\right)\left(-s + d(4+i)\right)(s^2 - 2sd(i+1) + d^2(i+1)^2)$$

$$= \left(\frac{-s^3 + (3i+6)ds^2 - 3d^2(i^2+4i+3)s + d^3(i^3+6i^2+8i+i+4)}{6d^3}\right)$$

Combined:

$$\frac{-3s^3 + (9i+12)ds^2 - 3d^2(3i^2+8i+4)s + d^3(3i^3+12i^2+12i+4)}{6d^3}$$

Third polynomial piece:

$$\left(\frac{s-di}{3d}\right)\left(\frac{(d(i+3)-s)^2}{2d^2}\right) + \left(\frac{d(4+i)-s}{3d}\right)\left(\frac{-2s^2+sd(4(i+1)+6)-d^2[2(i+1)^2+6(i+1)+3]}{2d^2}\right)$$

Minor simplifications:

$$\left(\frac{s-di}{3d}\right)\left(\frac{(d(i+3)-s)^2}{2d^2}\right)+\left(\frac{d(4+i)-s}{3d}\right)\left(\frac{-2s^2+sd(4i+10)-d^2(2i^2+10i+11)}{2d^2}\right)$$

Expansion of first half of third polynomial piece:

$$\left(\frac{s-di}{3d}\right)\left(\frac{(d(i+3)-s)^2}{2d^2}\right)$$

$$\left(\frac{1}{6d^3}\right)(s-di)(d^2(i+3)^2-2sd(i+3)+s^2)$$

$$\left(\frac{s^3-d(3i+6)s^2+d^2(3i^2+12i+9)s-d^3i(i+3)^2}{6d^3}\right)$$

Expansion of second half of third polynomial piece:

$$\left(\frac{d(4+i)-s}{3d}\right)\left(\frac{-2s^2+sd(4i+10)-d^2(2i^2+10i+11)}{2d^2}\right)$$

$$\left(\frac{1}{6d^3}\right)(d(4+i)-s)\left(-2s^2+sd(4i+10)-d^2(2i^2+10i+11)\right)$$

$$\frac{2s^3+(-18-6i)ds^2+(36i+6i^2+51)d^2s+(-2i^3-18i^2-51i-44)d^3}{6d^3}$$

Combined:

$$\left(\frac{s^3-d(3i+6)s^2+d^2(3i^2+12i+9)s-d^3i(i+3)^2}{6d^3}\right)$$

$$\frac{2s^3+(-18-6i)ds^2+(36i+6i^2+51)d^2s+(-2i^3-18i^2-51i-44)d^3}{6d^3}$$

$$\left(\frac{1}{6d^3}\right)(3s^3-(9i+24)ds^2+(9i^2+48i+60)d^2s-(i(i+3)^2+2i^3+18i^2+51i+44)d^3)$$

$$\left(\frac{1}{2d^3}\right)\left(s^3-(3i+8)ds^2+(3i^2+16i+20)d^2s-\left(i^3+8i^2+20i+\frac{44}{3}\right)d^3\right)$$

$$\frac{s^3-(3i+8)ds^2+(3i^2+16i+20)d^2s-\left(i^3+8i^2+20i+\frac{44}{3}\right)d^3}{2d^3}$$

Fourth polynomial piece:

$$\left(\frac{d(4+i)-s}{3d}\right)\left(\frac{(d(i+4)-s)^2}{2d^2}\right)$$

$$\left(\frac{(d(i+4)-s)^3}{6d^3}\right)$$

Final equation:

$$b_i^4(s) \tag{A.1-7}$$
$$= \begin{cases} \frac{(s-di)^3}{6d^3} & for\ di \le s < d(i+1) \\ \frac{-3s^3+(9i+12)ds^2-3d^2(3i^2+8i+4)s+d^3(3i^3+12i^2+12i+4)}{6d^3} & for\ d(i+1) \le s < d(i+2) \\ \frac{s^3-(3i+8)ds^2+(3i^2+16i+20)d^2s-\left(i^3+8i^2+20i+\frac{44}{3}\right)d^3}{2d^3} & for\ d(i+2) \le s < d(i+3) \\ \left(\frac{(d(i+4)-s)^3}{6d^3}\right) & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases}$$

A.1.5 Third order B-spline derivatives

Original third order equation:

$$b_i^3(s) = \begin{cases} \frac{(s-di)^2}{2d^2} & for\ di \le s < d(i+1) \\ \frac{-2s^2+sd(4i+6)-d^2[2i^2+6i+3]}{2d^2} & for\ d(i+1) \le s < d(i+2) \\ \frac{(d(i+3)-s)^2}{2d^2} & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases}$$

First polynomial piece:

$$\frac{(s-di)^2}{2d^2} = \frac{s^2-2dis+d^2i^2}{2d^2} = \left(\frac{1}{2d^2}\right)s^2 - \frac{is}{d} + \frac{i^2}{2}$$

First derivative:

$$\frac{d}{ds}\left(\frac{(s-di)^2}{2d^2}\right) = \left(\frac{1}{d^2}\right)s - \frac{i}{d} = \frac{s-di}{d^2}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{s-di}{d^2}\right) = \frac{1}{d^2}$$

Second polynomial piece:

$$\frac{-2s^2 + sd(4i+6) - d^2[2i^2 + 6i + 3]}{2d^2}$$

$$\left[-\frac{1}{d^2}\right]s^2 + \left[\frac{(4i+6)}{2d}\right]s - [i^2 + 3i + 1.5]$$

First derivative:

$$\frac{d}{ds}\left(\left[-\frac{1}{d^2}\right]s^2 + \left[\frac{(4i+6)}{2d}\right]s - [i^2 + 3i + 1.5]\right) = \left[-\frac{2}{d^2}\right]s + \left[\frac{(4i+6)}{2d}\right]$$

$$= \left[-\frac{4}{2d^2}\right]s + \left[\frac{d(4i+6)}{2d^2}\right]$$

$$= \frac{-2s + d(2i+3)}{d^2}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{-2s + d(2i+3)}{d^2}\right) = -\frac{2}{d^2}$$

Third polynomial piece:

$$\frac{(d(i+3) - s)^2}{2d^2} = \frac{d^2(i+3)^2 - 2d(i+3)s + s^2}{2d^2} = \left[\frac{1}{2d^2}\right]s^2 + \left[-\frac{(i+3)}{d}\right]s + \left[\frac{(i+3)^2}{2}\right]$$

First derivative:

$$\frac{d}{ds}\left(\left[\frac{1}{2d^2}\right]s^2 + \left[-\frac{(i+3)}{d}\right]s + \left[\frac{(i+3)^2}{2}\right]\right) = \frac{1}{d^2}s - \frac{i+3}{d}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{1}{d^2}s - \frac{i+3}{d}\right) = \frac{1}{d^2}$$

Final equations:

$$\frac{d}{ds}\left(b_i^3(s)\right) = \begin{cases} \dfrac{s - di}{d^2} & for\ di \le s < d(i+1) \\ \dfrac{-2s + d(2i+3)}{d^2} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{1}{d^2}s - \dfrac{i+3}{d} & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases} \qquad \text{(A.1-8)}$$

$$\frac{d^2}{ds^2}\left(b_i^3(s)\right) = \begin{cases} \frac{1}{d^2} & for\ di \le s < d(i+1) \\ -\frac{2}{d^2} & for\ d(i+1) \le s < d(i+2) \\ \frac{1}{d^2} & for\ d(i+2) \le s < d(i+3) \\ 0 & otherwise \end{cases} \tag{A.1-9}$$

A.1.6 Fourth order B-spline derivative

$$b_i^4(s)$$

$$= \begin{cases} \frac{(s-di)^3}{6d^3} & for\ di \le s < d(i+1) \\ \frac{-3s^3 + (9i+12)ds^2 - 3d^2(3i^2+8i+4)s + d^3(3i^3+12i^2+12i+4)}{6d^3} & for\ d(i+1) \le s < d(i+2) \\ \frac{s^3 - (3i+8)ds^2 + (3i^2+16i+20)d^2s - \left(i^3+8i^2+20i+\frac{44}{3}\right)d^3}{2d^3} & for\ d(i+2) \le s < d(i+3) \\ \frac{(d(i+4)-s)^3}{6d^3} & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases}$$

First polynomial piece:

$$\frac{(s-di)^3}{6d^3} = \frac{(s-di)(s^2-2sdi+d^2i^2)}{6d^3} = \frac{s^3 - 2s^2di + sd^2i^2 - dis^2 + 2sd^2i^2 - d^3i^3}{6d^3}$$

$$= \frac{s^3 - (3di)s^2 + (3d^2i^2)s + (d^3i^3)}{6d^3}$$

First derivative:

$$\frac{d}{ds}\left(\frac{s^3 - (3di)s^2 + (3d^2i^2)s + (d^3i^3)}{6d^3}\right) = \left(\frac{1}{2d^3}\right)s^2 - \left(\frac{i}{d^2}\right)s + \left(\frac{i^2}{2d}\right)$$

$$= \left(\frac{1}{2d^3}\right)s^2 - \left(\frac{2id}{2d^3}\right)s + \left(\frac{i^2d^2}{2d^3}\right)$$

$$= \frac{s^2 - 2dis + i^2d^2}{2d^3}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{s^2 - 2dis + i^2d^2}{2d^3}\right) = \left(\frac{1}{d^3}\right)s - \left(\frac{i}{d^2}\right)$$

Second polynomial piece:

$$\frac{-3s^3 + (9i+12)ds^2 - 3d^2(3i^2+8i+4)s + d^3(3i^3+12i^2+12i+4)}{6d^3}$$

First derivative:

$$\frac{d}{ds}\left(\frac{-3s^3 + (9i + 12)ds^2 - 3d^2(3i^2 + 8i + 4)s + d^3(3i^3 + 12i^2 + 12i + 4)}{6d^3}\right)$$

$$= \left(-\frac{9}{6d^3}\right)s^2 + \left(\frac{2(9i + 12)d}{6d^3}\right)s - \left(\frac{3d^2(3i^2 + 8i + 4)}{6d^3}\right)$$

$$= \frac{-3s^2 + 2(3i + 4)ds - d^2(3i^2 + 8i + 4)}{2d^3}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{-3s^2 + 2(3i + 4)ds - d^2(3i^2 + 8i + 4)}{2d^3}\right) = -\frac{3}{d^3}s + \frac{(3i + 4)d}{d^3}$$

Third polynomial piece:

$$\frac{s^3 - (3i + 8)ds^2 + (3i^2 + 16i + 20)d^2s - \left(i^3 + 8i^2 + 20i + \frac{44}{3}\right)d^3}{2d^3}$$

First derivative:

$$\frac{d}{ds}\left(\frac{s^3 - (3i + 8)ds^2 + (3i^2 + 16i + 20)d^2s - \left(i^3 + 8i^2 + 20i + \frac{44}{3}\right)d^3}{2d^3}\right) =$$

$$\left(\frac{3}{2d^3}\right)s^2 - \left(\frac{2(3i + 8)d}{2d^3}\right)s + \left(\frac{(3i^2 + 16i + 20)d^2}{2d^3}\right)$$

$$\frac{3s^2 - (6i + 16)ds + (3i^2 + 16i + 20)d^2}{2d^3}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{3s^2 - (6i + 16)ds + (3i^2 + 16i + 20)d^2}{2d^3}\right) = \frac{3s - (3i + 8)d}{d^3}$$

Fourth polynomial piece:

$$\frac{(d(i + 4) - s)^3}{6d^3} = \frac{d^3(i + 4)^3 - 3sd^2(i + 4)^2 + 3ds^2(i + 4) - s^3}{6d^3}$$

First derivative:

$$\frac{d}{ds}\left(\frac{d^3(i + 4)^3 - 3sd^2(i + 4)^2 + 3ds^2(i + 4) - s^3}{6d^3}\right) = \frac{-d^2(i + 4)^2 + 2ds(i + 4) - s^2}{2d^3}$$

Second derivative:

$$\frac{d}{ds}\left(\frac{-d^2(i + 4)^2 + 2ds(i + 4) - s^2}{2d^3}\right) = \frac{2d(i + 4) - 2s}{2d^3}$$

Final equations:

$$\frac{d}{ds}\left(b_i^4(s)\right)$$

$$= \begin{cases} \dfrac{s^2 - 2dis + i^2d^2}{2d^3} & for\ di \le s < d(i+1) \\ \dfrac{-3s^2 + 2(3i+4)ds - d^2(3i^2+8i+4)}{2d^3} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{3s^2 - (6i+16)ds + (3i^2+16i+20)d^2}{2d^3} & for\ d(i+2) \le s < d(i+3) \\ \dfrac{-d^2(i+4)^2 + 2ds(i+4) - s^2}{2d^3} & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases} \tag{A.1-10}$$

$$\frac{d^2}{ds^2}\left(b_i^4(s)\right) = \begin{cases} \left(\dfrac{1}{d^3}\right)s - \left(\dfrac{i}{d^2}\right) & for\ di \le s < d(i+1) \\ -\dfrac{3}{d^3}s + \dfrac{(3i+4)d}{d^3} & for\ d(i+1) \le s < d(i+2) \\ \dfrac{3s - (3i+8)d}{d^3} & for\ d(i+2) \le s < d(i+3) \\ \dfrac{2d(i+4) - 2s}{2d^3} & for\ d(i+3) \le s < d(i+4) \\ 0 & otherwise \end{cases} \tag{A.1-11}$$

A.2 Derivation of theoretical minimum deceleration distance

In this Appendix, the minimum distance required to decelerate from a given feedrate v_{max} is computed.

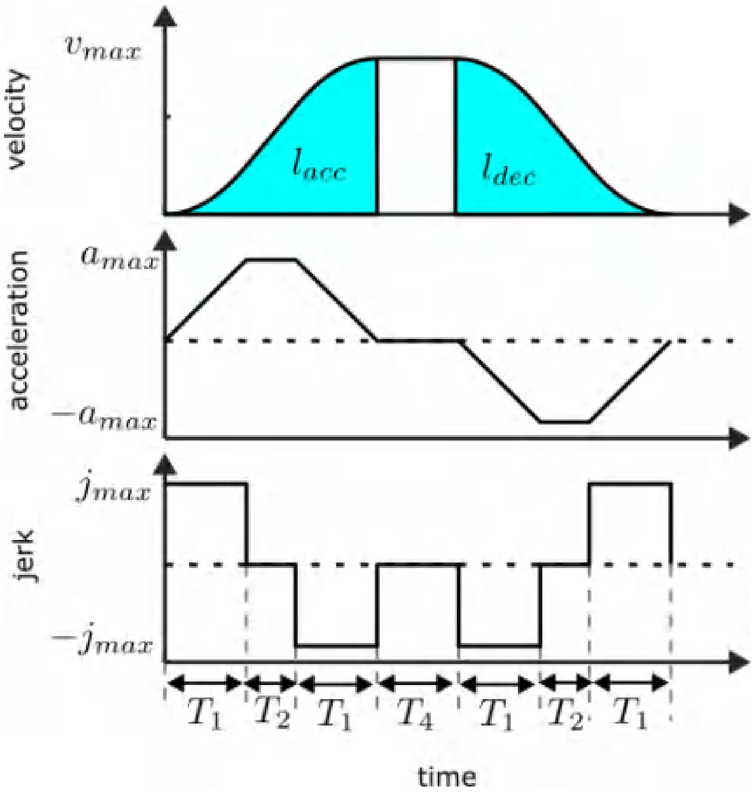


Figure A.2-6-1: Acceleration and deceleration distance with maximum velocity, acceleration, and jerk.

In the example in Figure A.2-6-1, both l_{acc} (the acceleration distance) and l_{dec} (the deceleration distance) are indicated, as the area of shaded region underneath the velocity curve. This trajectory is symmetrical, and therefore $l_{acc} = l_{dec}$. As we need only to compute one of these, the following equations apply to the l_{acc} portion of the curve.

The time interval occupied by each phase of the motion is marked on the plot as T_1, T_2, T_4.

T_1 is first computed. During the first motion segment:

$j = j_{\max}$

$a = j_{\max} t$

$v = \frac{1}{2} j_{\max} t^2$

If, at the end of the first segment, the maximum acceleration is reached, the resulting change in feed would be given by Eq. (A.2-2).

$a_{\max} = j_{\max} T_1$

$$T_1 = \frac{a_{\max}}{j_{\max}} \tag{A.2-1}$$

$$v = \frac{1}{2} j_{\max}(T_1)^2 \rightarrow v = \frac{1}{2} j_{\max}\left(\frac{a_{\max}}{j_{\max}}\right)^2 = \frac{1}{2}\frac{a_{\max}^2}{j_{\max}} \tag{A.2-2}$$

Throughout the course of one acceleration, there are two T_1 segments, so the change in feed is given by Eq. (A.2-3).

$$v = \frac{a_{\max}^2}{j_{\max}} \tag{A.2-3}$$

After this computation, there are two possible cases:

Case 1:

$v \leq \frac{a_{\max}^2}{j_{\max}}$: The acceleration never reaches its maximum value a_{max}. The segment with length T_2 will therefore not exist.

Case 2:

$v > \frac{a_{\max}^2}{j_{\max}}$: The acceleration will reach its maximum value a_{max} and stay there for some time, T_2.

Case 1:

As the acceleration does not reach its maximum, the previously computed T_1 will not apply. Therefore, the value of T_1 is recomputed in Eq. (A.2-4).

$$v_{max} = 2\left[-\frac{1}{2} j_{\max} T_1^2\right]$$

$$T_1 = \sqrt{\frac{v_{\max}}{j_{\max}}} \tag{A.2-4}$$

Because of the symmetry of the curve, the value of T_1 is doubled, and the acceleration distance is computed.

$$l_{acc} = \frac{1}{2} v_{\max}(2T_1)$$

$$l_{acc} = v_{\max}\sqrt{\frac{v_{\max}}{j_{\max}}} = \frac{v_{\max}^{\frac{3}{2}}}{j_{\max}}$$

Case 2:

The value of T_1 computed in Eq. (A.2-1) remains true. We must then compute T_2:

$$\left(v_{\max} - \frac{a_{\max}^2}{j_{\max}}\right) = a_{\max} T_2$$

$$T_2 = \frac{v_{\max}}{a_{\max}} - \frac{a_{\max}}{j_{\max}}$$

Then, same as before the curve is symmetrical:

$$l_{acc} = \frac{1}{2} f_{wc}(2T_1 + T_2)$$

$$l_{acc} = \frac{1}{2} v_{\max} \left(2\ \frac{a_{\max}}{j_{\max}} + \frac{v_{\max}}{a_{\max}} - \frac{a_{\max}}{j_{\max}}\right)$$

$$l_{acc} = \frac{1}{2} v_{\max} \left(\frac{a_{\max}}{j_{\max}} + \frac{v_{\max}}{a_{\max}}\right)$$

Result:

The final equation for acceleration and deceleration distance is given in Eq. (A.2-5).

$$l_{acc} = \begin{cases} \dfrac{v_{\max}^{\frac{3}{2}}}{\sqrt{j_{\max}}} & if\ v_{\max} \le \dfrac{a_{\max}^2}{j_{\max}} \\ \dfrac{1}{2} v_{\max} \left(\dfrac{a_{\max}}{j_{\max}} + \dfrac{v_{\max}}{a_{\max}}\right) & if\ v_{\max} > \dfrac{a_{\max}^2}{j_{\max}} \end{cases} \tag{A.2-5}$$