A real-time configurable NURBS interpolator with bounded acceleration, jerk and chord error

Massimiliano Annoni, Alessandro Bardine, Stefano Campanelli, Pierfrancesco Foglia, Cosimo Antonio Prete

A DipartimentodiIngegneriadell'Informazione, UniversitàdiPisa, Via Diotisalvi 2, 56126 Pisa, Italy
bDipartimentodiMeccanica, Politecnico di Milano, Via La Masa 1, 20136 Milano, Italy

Abstract

Advances in manufacturing technologies and in machine tools allow for unprecedented quality and efficiency in production lines, but also ask for new and increasing requirements on the motion planning and control systems. The increase of CPU processing power has permitted, in traditional CNC systems, the introduction of NURBS interpolation capabilities, thus determining a further increase in machining quality and efficiency. This has posed new and still unsolved issues, such as the need to satisfy multiple opposite constraints like limiting chord error, acceleration and jerk and offering real-time guarantees. In addition, the ability of privileging the production throughput by relaxing one or more of the previous constraints in a simple way, has emerged as another requirement of modern manufacturing plants. Nevertheless, none of the existing NURBS interpolators have these characteristics. In this work, we propose a NURBS interpolator that is able to satisfy all the manufacturing technology requirements and is able to respect, thanks to its bounded computational complexity, the position control real-time constraints. Such an interpolator is easily reconfigurable, i.e., it can relax some of the constraints while maintaining performances better than previously proposed solutions, and can be adapted in order to include constraints that were not originally considered. Performances of the proposed algorithm have been evaluated both by simulations and by real milling experiments.

1. Introduction

Informatics influences the creation process of a workpiece in many ways. Machine tools used to carry out workpieces are controlled by Computer Numerical Control (CNC) systems. The machine tool has to follow a path in order to manufacture the desired workpiece. That path is specified using a particular programming language called G-code. The tool-path is usually generated making use of Computer Aided Manufacturing (CAM) software that produces program codes supporting the operator in the workpiece 3D model analysis. Such a model is designed using Computer Aided Design (CAD) software.

The software architecture of a typical CNC system consists of three main modules: the interpreter decodes the program code to obtain the tool-path information; the interpolator generates reference values for the position control and the position control manages the servomotor control loop [1].

First interpolators allowed the specification of tool-path using straight segments (linear interpolation) and circular arcs (circular interpolation). Modern CAD software, however, supports free-form design of curves so a complex tool-path must be approximated with a sequence of short linear segments and circular arcs. The approximation of tool-path reduces the machining accuracy. Moreover, there are corners between consecutive segments where the machine tool has to reduce feedrate (i.e. the machining speed) in order to change direction. Frequent accelerations and decelerations cause an increased machining time and mechanical vibrations that reduce machining quality. Furthermore, a lot of data is required to approximate complex curves, causing data transmission and storage problems [2].

In order to overcome these limitations, NURBS interpolators were introduced. These interpolators allow the specification of tool-path using NURBS (Non-Uniform Rational B-Spline) [3], a particular kind of parametric curves. NURBS has become the industry de facto standard for representing and designing shapes because it can represent both analytic and free-form surfaces. Algorithms for manipulating and computing NURBS are efficient and numerically stable [4].

Only few NURBS are needed to represent even complex toolpaths so, using NURBS interpolators, program code size is
smaller and the sharp corners are reduced. This fact allows the achievement of higher feedrates along the curve, reducing fluctuations caused by deceleration at corners.

NURBS interpolators can use an offline or online approach. With the offline approach, most of the computations are performed before the interpolation starts in order to save CPU processing power. One issue of this approach is the difficulty to store computed data, moreover, the data must be recalculated if some parameters of CNC machine change, so it is not a suitable approach for changes on the fly. Using the online approach, all computations are executed during interpolation. This fact requires more computational power, but does not suffer from drawbacks described above and it allows to get a feedback during interpolation, that can be used to improve the machining performance. With the increasing processing power capabilities of modern hardware, the online approach has become the most widespread. In order to prevent degradation of machining quality in faster machining processes, interpolators must consider the following constraints:

- chord error (i.e. the distance between the actual curve and the path generated by the interpolator) must be limited in order to achieve high-precision machining [5];
- acceleration (i.e. the rate of change of velocity) must be limited in order to reduce inertia and prevent mechanical shocks that degrade machining quality [1];
- jerk (i.e. the rate of change of acceleration) must be limited to smooth the feedrate profile, improving machining quality [6].

Evolution of NURBS interpolators brought to the development of algorithms satisfying an increasing number of constraints. In particular: chord error is limited in [5,7]; chord error and the tangential acceleration component are limited in [8,9]; chord error and acceleration are limited in [10,11]; chord error, tangential components of acceleration and its derivative (that is only a component of the jerk) are limited in [12,13]; chord error, acceleration and the derivative of tangential acceleration are limited in [14]; chord error, acceleration and tangential jerk are limited in [6,15]. However, to the authors’ best knowledge there is not a NURBS interpolator capable to fulfill all of the constraints cited above, even though this is a requirement desired from both large (Fanuc [16], Siemens [17], Heidenhain [18], etc.) and small (ISAC svl [19], D. Electron [20], etc.) CNC manufacturers.

In addition, an important issue in the design of a NURBS interpolator is that the architecture of a CNC system requires the interpolator to provide reference values for the position control at regular time intervals, fulfilling real-time constraints. Most sophisticated interpolators [6,9,10] use complex iterative algorithms in order to generate position values. Giving an upper bound to their execution time is difficult. So they are not suitable for time-bound implementation.

In this work, a configurable NURBS interpolator is presented with the following characteristics:

- It can limit acceleration, jerk and chord error;
- It can give real-time guarantees for the generation of reference values for the position control;
- It is reconfigurable: relaxing some constraints, the algorithm can behave like previous algorithms and it can be extended with new constraints still not considered.

In order to reach such features, new methods to limit jerk, reduce interpolation error and generate a feedrate profile considering interpolation error are introduced.

The rest of this paper is structured as follows. Related works are presented in Section 2. Section 3 describes basic concepts of NURBS Interpolation, while the architecture of the proposed interpolator is presented and improvements are highlighted in Section 4. Section 5 presents the analysis methodology and results. Section 6 concludes the paper.

2. Related work

First NURBS interpolators advanced through the curve with uniform increments of the parameter of the NURBS [21]. This approach does not provide any control over speed or quality of the machining process.

Real-time speed-controlled NURBS interpolators were successively developed so that machining speed could be controlled [22]. Following works focused on interpolators capable of machining a workpiece according to a feedrate profile. First proposals of interpolators [2] used first-order Taylor approximation method to calculate parameter increments according to the desired feedrate.

Approximation error of parameter increments causes undesired feedrate fluctuations, so second-order Taylor approximation method was adopted [23,24]. Variable feedrate algorithms [5,7] were introduced to control the chord error. Since chord error depends on curvature and feedrate, these algorithms adjust the feedrate during interpolation, slowing down when the curvature of the path increases. Interpolators presented in [5,7], however, cannot control the acceleration of the machine tool. So, when curvature changes abruptly, high acceleration/deceleration values may be required, causing strong mechanical shocks.

Some interpolators [11,12,14] adopted an offline approach to detect high curvature zones and generate a feedrate profile that allows to respect desired constraints.

Other algorithms [6,9,10,13] used an online approach, keeping a buffer of interpolated points. When a high curvature corner is found, part of the buffer is recalculated in order to meet the acceleration constraints. As it is not possible to predict how often elements of the buffer must be recalculated nor how many elements need to be recalculated each time, calculating a time-bound for this kind of algorithms is difficult.

Some interpolators [11,12,14] use an online two-phases approach. In the first phase a feedrate profile is generated and in the second phase the actual real-time interpolation is performed. The look-ahead phase of the interpolators presented in [12,14] searches for local minima and maxima of the curvature function of the curve and uses this information to generate a feedrate profile that can limit the derivative of tangential acceleration.

In [11] the NURBS is broken into several segments, each one with similar curvature values and, for each segment, a maximum feedrate is determined.

3. NURBS interpolation

In this section, basic concepts on NURBS interpolation, acceleration/deceleration control and curvature-based feedrate variation are explained.

3.1. NURBS basics

NURBS, like other parametric curves, are represented as a vector function of a scalar parameter $u$. The following elements are needed to define a NURBS [3]:

- $n + 1$ control points ($P_i$);
- a weight for each control point ($w_i$);
- the degree of the curve ($p$);
- a knot vector ($u_0, \ldots, u_m$), where $m = n + p + 1$.

So, the NURBS can be expressed as

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)w_i}{\sum_{i=0}^{n} N_{i,p}(u)} u \in [u_0, u_m].$$  (1)
The basis functions \( N_{i,p}(u) \) are determined by the knot vector in the following way:

\[
N_{i,0}(u) = \begin{cases} 
1 & \text{if } u \in [u_i, u_{i+1}) \\
0 & \text{otherwise}
\end{cases}
\]

\[
N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u).
\]

3.2. Speed-controlled interpolator

A real-time CNC interpolator (Fig. 1) generates reference values for the position control at regular time intervals \( t_k \). So, a CNC interpolator actually performs a sampling of the NURBS. Distance between samples depends on desired feedrate (machining speed) profile \( v(t) \). In order to compute the \( k \)th position using (1), it is necessary to know \( u(t_k) \). Using Taylor series expansion of function \( u(t) \), \( u(t_k) \) can be expressed as

\[
u(t_k) = u(t_{k-1}) + \frac{T^2}{2} u(t_{k-1}) + \frac{T^3}{6} u(t_{k-1}) + \cdots ,
\]

where \( T \) is the interpolator period and derivatives are with respect to time. Using the chain rule, feedrate can be expressed as

\[
v = \frac{ds}{dt} = \left( \frac{ds}{du} \right) \left( \frac{du}{dt} \right),
\]

where \( s \) is the arc-length of the curve. So, the time derivative of \( u(t) \) is calculated as

\[
\dot{u} = \frac{v}{\sigma} = \frac{v}{\sqrt{x^2 + y^2}},
\]

where \( \sigma \) is the parametric speed and is calculated as

\[
\sigma = \sqrt{x^2 + y^2},
\]

where \( x \) and \( y \) are the components of vector \( C(u) \) (1).

In a similar way, second and third time derivatives of \( u(t) \) can be calculated [25] as follows:

\[
\ddot{u} = \frac{\ddot{v}}{\sigma} = \frac{\sigma^2 v}{\sigma^2 u},
\]

\[
\dddot{u} = \frac{\dddot{v}}{\sigma} = \frac{3\sigma v}{\sigma^2 u} - \frac{\sigma^2 v^2}{\sigma^2 u^2}.
\]

The derivatives with prime mark notation are with respect to \( u \), while derivatives with dot notation are with respect to time.

Derivatives of parametric speed \( \sigma \) are defined as

\[
\sigma' = \frac{x'x'' + y'y''}{\sigma},
\]

\[
\sigma'' = \frac{x'x'' + y'y'' + x''^2 + y''^2 + \sigma^2}{\sigma^2}.
\]

Most common approximations, in calculation of \( u(t_k) \), are first and second order truncations of Taylor expansion (3). In this work, first, second and third order approximations are considered.

3.3. Acceleration/deceleration control

Abrupt feedrate variations cause mechanical vibrations and shocks that degrade machine quality [1]. Better machining quality is achieved using smooth feedrate profiles, like linear and s-shape feedrate profiles (Fig. 2) that are commonly used in CNC interpolation. The former can limit first derivative of feedrate (tangential acceleration), while the latter can limit both first and second derivatives of feedrate.

The conventional method used to calculate feedrate profiles is ADCBI (Acceleration/Deceleration Control Before Interpolation) [1]. This method consists in calculating the feedrate profile before the interpolation of NURBS is started.

In order to machine the NURBS more efficiently, some algorithms split the NURBS into curve segments and they calculate a different feedrate profile for each segment. In this case, stopping the interpolator at the end of each segment is inefficient, a look-ahead technique [1] has been introduced. Look-ahead calculates the final feedrate of a segment considering length and final feedrate of following segments ensuring that the interpolator feedrate profile of last segment can reach 0 mm/s.
3.4. Geometry-based feedrate limits

Some of the constraints (centripetal acceleration, jerk and chord error) are respected only if feedrate does not exceed a calculated limit that depends on geometric properties of the curve.

The maximum distance between the chord that joins two consecutive interpolated points and the correspondent arc on the nominal curve is called chord error. A feedrate limit that allows to confine chord error has been presented in [5]. It depends on curvature \( \kappa \), that is the geometric property measuring the curve deviation from its tangent.

Curvature is calculated as

\[
\kappa = \frac{|x'y'' - y'x'''|}{\sigma^3}, \tag{11}
\]

Radius of curvature \( \rho \) is defined as the reciprocal of curvature

\[
\rho = \frac{1}{\kappa}. \tag{12}
\]

Chord error \( \epsilon \) can be estimated [5] as

\[
\epsilon = \sqrt{\rho^2 - \left(\frac{v^2}{2}\right)^2}. \tag{13}
\]

And from (13), the maximum feedrate limit that allows to respect chord error tolerance \( \epsilon_{\text{max}} \) is approximated as

\[
v_{\text{max}} = 2 \sqrt{\rho^2 - \left(\frac{v^2}{2}\right)^2}. \tag{14}
\]

However, since (13) is an approximated formula, the tolerance value \( \epsilon_{\text{max}} \) must be chosen in a conservative way, since it could slightly exceed.

Interpolators [6,11] limit centripetal acceleration too.

Centripetal acceleration, that depends on curvature, is calculated as

\[
a_c = \frac{v^2}{\rho}. \tag{15}
\]

In order to confine it, feedrate must always be equal or less than

\[
v_{\text{max,ac}} = \sqrt{a_{\text{max}} \rho}. \tag{16}
\]

where \( a_{\text{max}} \) is the centripetal acceleration limit.

4. Configurable interpolator architecture

The proposed interpolator architecture is shown in Fig. 3. It consists of three modules:

- segmentation module;
- look-ahead module;
- interpolation module.

Each module is mapped into a different task. Segmentation module communicates with look-ahead module through the segmentation buffer, while look-ahead module communicates with interpolation module using look-ahead buffer.

The proposed NURBS interpolator can be configured to limit the following quantities:

- tangential acceleration;
- centripetal acceleration;
- derivative of tangential acceleration;
- tangential jerk;
- centripetal jerk;
- chord error.

It supports different kinds of interpolator approximation, so it can be adapted to be executed on different hardware with different computational power capabilities.

Some constraints can be relaxed to adapt to desired machining quality and speed. In addition, constraints can be easily added if they can be represented as a feedrate limit.

In the following sections, the different modules of the interpolator are described.
that is the actual formula used to calculate the feedrate limit for tangential jerk constraint.

Feedrate limit that allows to respect centripetal jerk constraint \( v_{\text{max,}c} \) is calculated, instead, solving the following inequality:

\[
\frac{v}{\rho} \left( 3\dot{v} + \frac{v}{\rho} \dot{\rho} \right) \leq k_{\text{max}},
\]

(21)

where \( k_{\text{max}} \) is the maximum allowed value for centripetal jerk. Using the chain rule, the time derivative of the curvature radius can be expressed as

\[
\dot{\rho} = \frac{d\rho}{dt} = \frac{ds}{d\rho} \frac{d\rho}{ds} = \frac{d\rho}{ds}.
\]

(22)

where \( s \) is the arc-length and its time derivative is the feedrate. Since it is easier to make computation with curvature, Eq. (11) is used to substitute \( \rho \) in Eq. (22), that becomes

\[
\dot{\rho} = \frac{d}{ds} \left( \frac{1}{\kappa} \right) = -\frac{v \, d\kappa}{\kappa^2 \, ds}.
\]

(23)

The derivative of curvature with respect to arc-length is calculated [25] as

\[
\frac{d\kappa}{ds} = \frac{|x'y''' - y'x''|}{\sigma^4}.
\]

(24)

So, inequality (21) becomes

\[
\left| 3\dot{v} \, v - \frac{d\kappa}{ds} \right| \leq k_{\text{max}}.
\]

(25)

Solving inequality (25), considering the worst case for \( \dot{v} \), feedrate limit \( v_{\text{max,}c} \) that allows to respect centripetal jerk constraint can be calculated. Details of this calculation are found in the Appendix.

To sum up:
- feedrate limit for respecting chord error constraint is calculated using formula (14);
- feedrate limit for respecting centripetal acceleration is calculated using formula (16);
- feedrate limit for respecting tangential jerk is calculated using formula (20);
- feedrate limit for respecting centripetal jerk is calculated solving inequality (25).

If more than a constraint must be respected, the minimum feedrate limit is used.

4.1.2. Segmentation algorithm
Segmentation module splits the NURBS so that consecutive points having a similar feedrate limit (determined by the curve geometry) are put into the same segment.

A set of feedrate ranges \((v_i, v_{i+1}]\) have been defined. Boundaries \( v_i \) are calculated in the following way:

\[
v_i = \begin{cases} 
+\infty & \text{if } i = 0 \\
v_{\text{comm}} & \text{if } i = 1 \\
v_{i-1} & \text{otherwise,}
\end{cases}
\]

(26)

where \( v_{\text{comm}} \) is the commanded feedrate (i.e. the maximum feedrate specified in the program code).

Consecutive points that have a feedrate limit within the same range are put into the same segment, as shown in flow-chart in Fig. 4.

Approximated length and minimum feedrate limit are calculated for each segment. Doing so, the feedrate limit is respected by all the points of the segment.

Segments obtained are stored in the segment buffer. Each segment is described by the following information:
- final value of parameter \( u \);
- final position;
- feedrate limit;
- approximated length.

4.2. Look-ahead module
Look-ahead module takes segments from segment buffer, puts them into the look-ahead buffer and updates their final feedrate so that the last segment of the look-ahead buffer can always reach \( 0 \) mm/s. It implements look-ahead technique described in Section 3.3.

4.3. Interpolation module
Interpolation module generates reference values for the position controller. It is basically a real-time speed-controlled interpolator with configurable order of Taylor approximation. Flow chart of the interpolation module is shown in Fig. 5. The interpolation module generates the feedrate profile for the current interpolation period using the ADCDI (Acceleration/Deceleration Control During Interpolation) method. Then, this profile is corrected according to desired Taylor approximation, in order to reduce interpolation error. If the next knot of the NURBS will be passed in the current interpolation period, there may be discontinuities in curve derivatives that causes an increment of interpolation error, so a correction method is used. Then Taylor approximation method is used and next position is computed. In order to further reduce interpolation error, a configurable number of steps of linear approximation are performed. Finally, the computed position is communicated to the position control.
4.3.1. Feedrate profile computation with ADCDI

ADCDI (Acceleration/Deceleration Control During Interpolation) is the method that we developed in order to generate a feedrate profile for the interpolation interval \([t_{k-1}, t_k]\). Profile is generated considering maximum and final feedrate of the current segment and the remaining length left to the end of the segment. Both linear and s-shape profiles are supported. This approach has been chosen over the conventional ADCBI method described in 3.3 because the latter has two main drawbacks.

First drawback is that ADCBI module also has a time constraint, since it must produce data fast enough for the interpolator. Dimensioning this constraint is difficult, since it depends on the time needed by the interpolator to consume a segment and minimum segment size is not known. ADCDI does not have this problem, since it is embedded into interpolation module.

Second drawback is that ADCBI is very sensitive to approximation errors that cause a mismatch between the estimated segment length and the actual length of the interpolated path corresponding to the segment. So, the generated feedrate profile may end before or after the actual end of the segment. This fact becomes critical in the last segment, where situations shown in Fig. 6 may occur. Since ADCDI computes the feedrate profile at each interpolation cycle, considering the actual covered distance on the segment, it is less sensitive to approximation errors.

4.3.2. Correction of feedrate profile

The correction of feedrate profile is performed in order to change feedrate profile \(v(t)\), calculated using ADCDI, with a modified version that reduces interpolation error (i.e. the difference between the desired distance to cover in the interpolation interval and the actual covered distance).

Feedrate profile \(v(t)\), limited to the \(k\)th interpolation interval \([t_{k-1}, t_k]\), can be expressed as a linear (27) or a quadratic (28) equation, according to the desired profile (respectively, linear or s-shape).

\[
\begin{align*}
v_l(t) &= v(t_{k-1}) + \dot{v}(t_{k-1}) \cdot (t - t_{k-1}) \\
v_s(t) &= v(t_{k-1}) + \dot{v}(t_{k-1}) \cdot (t - t_{k-1}) + \frac{1}{2} \ddot{v}(t_{k-1}) \cdot (t - t_{k-1})^2
\end{align*}
\]

First order Taylor approximation, however, truncates the terms of (3) that includes the derivatives of \(v(t)\), so it considers \(v(t)\) as a constant function. In order to reduce this truncation error, in this work we perform a feedrate correction for the interval \([t_{k-1}, t_k]\), changing the desired profile into a constant feedrate profile that covers the same desired distance \(d\), calculated as:

\[
d = \int_{t_{k-1}}^{t_k} v(t) \, dt.
\]

So, the corrected feedrate function becomes

\[
\begin{align*}
v_{l,c}(t) &= \frac{v_l(t_{k-1}) + v_l(t_k)}{2} \\
v_{s,c}(t) &= v_s(t_{k-1}) + \frac{2 \dot{v}_s(t_{k-1}) + \ddot{v}_s(t_k)}{6} T
\end{align*}
\]

respectively for linear or s-shape profiles. Similarly, the second order Taylor approximation only considers the first derivative of \(v(t)\), so it considers it a linear function. In this case, feedrate correction is necessary only for s-shape profile

\[
v_{s,c}(t) = v_s(t_{k-1}) + \frac{2 \dot{v}_s(t_{k-1}) + \ddot{v}_s(t_k)}{3} (t - t_{k-1}).
\]

Third order Taylor approximation does not need feedrate correction using neither linear nor s-shape profiles. Fig. 7 shows an example of feedrate correction.

Fig. 6. Effects of approximation errors on feedrate profile with ADCBI. (a) Feedrate profile ends before the actual end of curve. The last portion of the curve is interpolated at very low feedrate and machining time is highly increased. (b) Feedrate profile ends after the actual curve end. There is an abrupt machine stop that degrades the machining quality.
4.3.3. Correction of discontinuities of curve derivatives

This method allows to correctly compute the Taylor approximation when there are discontinuities of curve derivatives.

A function can be expressed as a Taylor series expansion only if all of its derivatives are continuous in the interval of interest. NURBS derivatives may have discontinuities when parameter \( u \) assumes values equal to one of the knots of the curve \([u_0, \ldots, u_n]\). So, if a knot \( u_i \) such that \( u(k_i-1) < u_i < u(k_i) \) exists, formula (31) is not applicable. In this work such cases are handled considering \( u(t_k) \) in Taylor expansion instead of \( u(t_k) \):

\[
u(t_k) = u(t_k) + T_1 \dot{u}(t_k) + \frac{T_1^2}{2} \ddot{u}(t_k) + \frac{T_1^3}{6} \dddot{u}(t_k) + \cdots\]

(33)

where \( t_k \) is calculated according to the feedrate profile so that \( u(t_k) = u_i \), solving the following equation:

\[
\| C(u_i) - C(u(t_{k-1})) \| = \int_{t_{k-1}}^{t_k} v(t)dt.
\]

(34)

The discontinuity can be skipped this way and the Taylor series expansion is still applicable.

4.3.4. Linear approximation

This method allows to further reduce interpolation error after the Taylor approximation has been computed. It uses an iterative approach that calculates a new value for \( u(t_k) \) and iterates until a desired precision is met. However, since the execution time of the interpolation module must be bounded, a maximum number of steps is imposed.

The function \( d_i(u) \) that links the parameter \( u \) with the distance between position at \( u \) and last position is defined as

\[
d_i(u) = \| C(u) - C(u(t_{k-1})) \|.
\]

(35)

Using linear approximation method, \( d_i(u) \) is approximated with a linear function, as shown in Fig. 8, and it can be inverted obtaining

\[
u(d_r) = u_o + \frac{u_0 - u_k}{d_k - d_o} (d_r - d_o).
\]

(36)

Formula (36) is used to calculate \( u(d) \) at each linear approximation step, with \( d \) as the desired distance obtained from (29); \( u_k \) is the parameter value calculated in the previous step (or Taylor approximation for the first step) and \( u_o \) is initially \( u(t_{k-1}) \) and after each step, if \( d > d_o \) is updated with value of \( u_k \) so that \( d_o \) is always smaller than \( d \). The procedure is iterated until the desired precision is met, or until the maximum number of linear approximation steps is reached.

4.4. Concurrency and real-time considerations

Segmentation module, look-ahead module and interpolation module (shown in Fig. 3) can run simultaneously. Segmentation and look-ahead modules have no time constraints and communicate through the segmentation buffer with a producer/consumer paradigm. Interpolation module produce commands for the position control at constant time intervals, so it must be implemented as a real-time task with period \( T \).

Flow-chart of interpolation module is shown in Fig. 5. Computation of position and curve derivatives are realized using algorithms described in [4] that have complexity \( O(p^n) \), where \( p \) is the degree of NURBS curve. Other steps of the flow chart are realized without iterations, so their complexity is constant \( O(1) \). Computation of position and curve derivatives is iterated for \( n_{steps} + 1 \) times, where \( n_{steps} \) is the maximum number of linear approximation steps, however, \( n_{steps} \) is a configurable parameter of the algorithm and it does not depends on the input data, so the total complexity of the algorithm is \( O(p^n) \). In order to calculate a time-bound for the interpolation module, a maximum value for the degree of the curve must be imposed. Conventional CNC systems usually support NURBS up to the third degree.

Note that, while the proposed interpolator can be time-bound, once a maximum curve degree is fixed, most sophisticated existing algorithms [6,9,10] use iterative computations that depend on the curve shape in addition to the degree.

Look-ahead module and interpolation module communicate using look-ahead buffer. If the interpolation module consumes data too fast and the look-ahead module cannot provide more data, the interpolator slows down according to desired feedrate profile and stops without negative consequences. This behavior is ensured by the look-ahead technique. However, such situations should not happen in efficient NURBS interpolations since they significantly increase machining time.

4.5. Configurability considerations

The proposed algorithm can be configured in order to meet different requirements of machining quality, production
throughput and computational power. If some of the constraints are relaxed, machining time of a workpiece decreases, increasing the throughput. Furthermore, the algorithm can be adapted in order to respect new constraints that can be expressed as a feedrate limit calculated in function of geometric properties of the curve. The new feedrate limit will be added to limits specified in Section 4.1.1.

In order to adapt the algorithm to the available computational power, the interpolation module can be configured to use different approximations of Taylor expansion (first, second and third order). In addition, the maximum number of linear approximation steps can also be configured. A better approximation method reduces interpolation errors that cause feedrate fluctuations, degrading machining quality.

5. Results

The test case is presented and performances of various configurations of the interpolator are discussed in this section. The provided results are obtained from both simulations and experimental tests.

5.1. Test case

NURBS interpolators performances are evaluated using test curves. Fig. 9 shows the NURBS used as test case in this study. Its parameters are:

- degree \(p = 3\)
- control points \(P_i = [10, 10, 0, 20, 10, 20, 20, 10, 10] (\text{mm})\);
- weights \(w_i = [1, 200, 200, 1, 200, 200, 1]\);
- knots \(u = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1]\).

This test curve has been chosen because it has both low curvature and high curvature zones, so it can show the advantages of using a segmented approach over a classical algorithm such as [2]. Under the technological point of view, the selected curve is interesting too, since high curvature zones are critical for workpiece quality due to both feedrate variations and inertial effects on the cutting tool. This path is at the same time quite simple, which could make it easier to detect quality defects on the workpiece due to the various algorithms.

The algorithm has been implemented on an industrial CNC controller [27] with a Core 2 Duo CPU and Windows XP Embedded operating system with RTX real-time extensions. Software simulations were performed using this configuration and their results are discussed in Section 5.2. Simulations are useful for checking if the feedrate profile produced by the interpolator can respect the constraints. However, simulations cannot provide any information about the effect of the feedrate profile on the position control system. It has to be pointed out as one of the main advantages of the proposed approach is the capability to limit centripetal acceleration and jerk: such motion characteristics play a fundamental role on the mill dynamics during milling operations and could interact with the process forces producing positional drifts. Experimental evaluation is necessary to verify that the proposed interpolator does not introduce undesired effects during machining and that it can actually improve machining quality. So, algorithm performances were evaluated also through experimental tests on a KERN Evo milling machine [28] controlled by an Heidenhain iTNC 530 CNC [18]. Experimental results are discussed in Section 5.3.

Table 1 lists the interpolator parameters used for the simulations, while Table 2 lists the parameters used for experimental tests. The choice of acceleration and jerk limits depends on the servo systems characteristics, the chord error tolerance is chosen depending on the desired machining precision and the interpolator period must be dimensioned so that the real-time constraints are not violated. Values in Table 1 represent a typical scenario. Note that \(J_{\text{c max}}\) must be greater than \(J_{\text{a max}}\) in order to correctly use Eq. (19).

The interpolator is configured to respect all of the constraints and to use third order Taylor approximation with two steps of linear approximation.

5.2. Simulations results

In this section, the results of the simulations are described. The performances of the proposed algorithm with all the constraints enabled are shown in order to verify the fulfillment of the constraints. Furthermore, the reconfiguration capability of the interpolator is also verified.

5.2.1. Basic results

An evaluation of the interpolation of the test curve (Fig. 9) has been performed. The interpolator was configured to respect all the supported constraints: chord error, acceleration and jerk. Fig. 10

![Fig. 9. The test NURBS curve. (a) Curve representation on the plane. (b) Curvature of the NURBS with respect to parameter \(u\).](image-url)
Fig. 10. Results for the proposed interpolator. (a) Feedrate over time. (b) Tangential acceleration over time. (c) Centripetal acceleration over time. (d) Tangential jerk over time. (e) Centripetal jerk over time. (f) Chord error over time. Dashed lines mark limits that should not be exceeded to fulfill constraints.

The total machining time is 2.3065 s. Graph (a) shows the feedrate during time. It slows down on high curvature zones in order to respect constraints. Graphs (b) and (c) show acceleration tangential and centripetal components. They both are within the limits (marked with the dashed lines). Graphs (d) and (e) show tangential and centripetal components of jerk. They are within the limits (marked with the dashed lines) too. Finally, graph (f) shows chord error over time. Chord error too is within the given tolerance. Performances of the algorithm are compared to the ones of an interpolator using a similar approach, but a different segmentation scheme [12]. It will be referred as “max-segmented”, since it splits the curve where its curvature function has local maxima. With max-segmented approach, each segment starts with a maximum that determines initial feedrate, contains a minimum that determines maximum feedrate and ends with another maximum that determines final feedrate. This approach, however, ensures that constraints are respected only for points where curvature has maxima and not for the whole curve. Performances of the algorithm are also compared to the adaptive interpolator [5] that changes the feedrate in order to limit chord error, without using any acceleration control method. Finally, performances of the algorithm are compared to the classical speed-controlled interpolator [2] that does not consider any geometry-based constraint and uses a linear feedrate profile that limits tangential acceleration. Fig. 11 shows results of other interpolators (max-segmented, adaptive and speed-controlled). The graphs show that other algorithms cannot respect the constraint limits (marked by dashed lines).

5.2.2. Configurability results

In this section, we release some constraints of the proposed interpolator so that it behaves like max-segmented interpolator, adaptive interpolator and speed-controlled interpolator; performances are then compared.

5.2.2.1. Max-segmented interpolator. Max-segmented interpolator cannot limit centripetal jerk, centripetal acceleration and tangential jerk (but it uses an s-shape profile, so it can limit the derivative of tangential acceleration), so those constraints have been disabled in the configurable interpolator. Fig. 12 shows simulation results. Graphs (a) and (b) show feedrate results for the proposed interpolator and max-segmented interpolator, respectively. Both interpolators slow down on high curvature zones in order to respect chord error tolerance. Max-segmented interpolator decelerates until the minimum feedrate is met, then immediately starts to accelerate. The proposed interpolator, instead, maintains the feedrate at the minimum limit for a whole segment, so it is a little slower. Graphs (c) and (d) show the derivative of tangential acceleration for the

Fig. 11. Simulation results for other algorithms. (a)–(c) Feedrate over time for max-segmented, adaptive and speed-controlled, respectively. (d) Centripetal jerk over time for max-segmented. (e) Tangential acceleration over time for adaptive. (f) Chord error over time for speed-controlled. Dashed lines mark limits that should not be exceeded to fulfill constraints.
5.2.2. Adaptive interpolator. Adaptive interpolator only limits chord error, so the proposed interpolator is configured to use only chord error constraint and acceleration/deceleration control is disabled so that the feedrate used during interpolation is always the feedrate limit of the segment. Fig. 13 shows results. Graphs (a) and (b) show feedrate results for proposed interpolator and adaptive interpolator, respectively. The adaptive interpolator suffers from feedrate fluctuations in high curvature zones that cause the exceeding of the commanded feedrate. Fluctuations are caused by the interpolation error, since the adaptive interpolator uses a first order Taylor approximation. The proposed algorithm does not suffer from noticeable feedrate fluctuations since it uses third order Taylor approximation and two steps of linear approximation. Graphs (c) and (d) show chord error results for proposed interpolator and adaptive interpolator, respectively. In both cases, chord error is confined even though the tolerance value is slightly exceeded because of the approximation error of (13).

The machining time of the adaptive interpolator is 0.912 s (60.46% less than the proposed interpolator with all the constraints), while the machining time of the proposed interpolator with only the chord error limit is 0.9275 s (59.79% less than when all the constraints are enabled).

5.2.2.3. Speed-controlled interpolator. The speed-controlled interpolator does not respect any constraints and it uses a linear feedrate profile that limits tangential acceleration. So, the proposed interpolator is configured not to consider any constraint during segmentation and to use a linear feedrate profile. Fig. 14 shows simulation results. Graphs (a) and (b) show feedrate results for proposed interpolator and speed-controlled interpolator, respectively. Since speed-controlled interpolator uses a first order Taylor approximation method, the feedrate fluctuations at high curvature
zones are very high and cause the exceeding of commanded feedrate. Graphs (c) and (d) show tangential acceleration results for proposed interpolator and speed-controlled interpolator, respectively. Feedrate fluctuations cited above cause high acceleration peaks, that are the first four peaks in graph (d). The last peak is due to the mismatch between desired feedrate profile and actual feedrate. The actual curve ends before the generated profile can reach 0 mm/s, causing high instantaneous deceleration that exceeds the constraints. This mismatch is caused by high interpolation error. So, the speed-controlled interpolator cannot confine tangential acceleration for all points of the curve, while the proposed interpolator can, since it has low interpolation error (it uses third order Taylor approximation and two steps of linear approximation).

The machining time of the speed-controlled interpolator is 0.9815 s (57.45% less than the proposed interpolator with all the constraints), while the machining time of the proposed interpolator with tangential acceleration constraints is 0.9865 s (57.23% less than when all the constraints are enabled).

### 5.3. Experimental results

Simulation results have shown that the feedrate profile generated by the proposed interpolator fulfills all the constraints. In order to prove that it does not introduce vibrations and that it can improve machining quality, the results of the experimental tests are shown in this section.

Experimental tests were performed on a KERN Evo milling machine. The tool-path shown in Fig. 9 is machined on a 2017A aluminum alloy block using a 6 mm diameter HSS mill (Garant 191200, 3 teeth), a spindle rotational speed of 18,568 rpm and a commanded federate of 900 mm/min (15 mm/s). The proposed interpolator is configured in order to respect all the constraints and the interpolation period is set to 3 ms. A picture of the machined tool-path is shown in Fig. 15.

In order to compare results, experimental tests of the max-segmented interpolator were performed too. Only this interpolator has been chosen for comparison because it has already been proved in [12] that it can improve machining quality compared to the adaptive and the speed-controlled interpolators. Moreover, these interpolators cannot limit the jerk (as shown in simulations), so they could be dangerous for the applied milling machine.

Contour error (i.e. the distance between the actual tool position and the desired tool-path) is the metric used to measure the machining quality. Results also include the tracking error for each axis, that allows the evaluation of the effects of the interpolator on the position control system.

Table 3 shows a comparison of tracking error, contour error and machining time between the max-segmented interpolator and the proposed interpolator. The results indicate that the proposed interpolator is able to reduce both tracking error and contour error. Fig. 16 shows the errors behavior over time. High contour and tracking error values can be found in correspondence of high curvature zones. The proposed interpolator, however, can reduce these peaks since it slows down more than the max-segmented interpolator to keep acceleration and jerk controlled. Max-segmented interpolator cannot directly limit centripetal acceleration and jerk, so it does not take them into account even if they could affect the cutting quality, especially on geometrical features, as thin walls, which are particularly sensible to mill inertial reactions.

Fig. 17 shows the differences between the desired tool-path and the actual machined tool-path for both max-segmented and the proposed interpolator in high curvature corners of the figure. It can be noted that when approaching high curvature zones, the control action necessary to turn the tool causes a deviation from the programmed path. Since max-segmented interpolator approaches these sensitive zones at higher speed, it causes a deviation that
is more marked in both the magnitude of the deviation and the length of the deviated path. The proposed interpolator reduces this deviation, machining these zones at smaller feedrate, according to the constraints.

6. Conclusions

In this work, a NURBS interpolator that can limit chord error, acceleration and jerk is presented. Configurability is an important characteristic of the proposed interpolator. It can be easily reconfigured so that some constraints can be relaxed in order to privilege production throughput over machining quality. In addition, the algorithm can be adapted to include constraints that were not originally considered. The proposed interpolator offers real-time guarantees, since its module that has real-time constraints can be time-bounded. Additionally, some methods that improve interpolation accuracy have been introduced.

The proposed interpolator has been evaluated through both simulations and experimental tests.

Simulation results have shown that the proposed interpolator can fulfill all of the considered constraints, while other existing algorithms cannot and that, relaxing some constraints, the production throughput is increased almost as much as using other existing algorithms. Experimental results have shown that no undesired effects are caused by the algorithm on the workpiece.

Acknowledgments

The authors would like to thank Luciano Dal Lago (HEIDENHAIN Italiana srl) and Francesco Cacciatore (Politecnico di Milano) for their advices during the experimental tests execution and Gualtiero Fantoni (Università di Pisa) for his suggestions on how to improve this work.

Appendix

In this appendix, details of method used to solve (25) are provided.

Inequality (25) can be expressed as a system of inequalities

\[
\begin{align*}
-\frac{d\kappa}{ds}v^3 + 3\kappa \dot{v}v - J_{c\text{max}} &\leq 0 \\
-\frac{d\kappa}{ds}v^3 + 3\kappa \dot{v}v + J_{c\text{max}} &\geq 0.
\end{align*}
\] (A.1)

Since the value of the time derivative of the feedrate (i.e., the tangential acceleration) is unknown during the segmentation, worst case is considered.

If \( \frac{d\kappa}{ds} \) is positive, the worst case is when \( \dot{v} = -a_{t\text{max}} \), so the feedrate limit that allow to respect centripetal jerk constraint is calculated solving

\[
-\frac{d\kappa}{ds}v_{\text{max},c}^3 - 3\kappa a_{t\text{max}} v_{\text{max},c} + J_{c\text{max}} = 0,
\] (A.2)

that has exactly one real solution.

If \( \frac{d\kappa}{ds} \) is negative, instead, the worst case is when \( \dot{v} = a_{t\text{max}} \) and the feedrate limit is calculated solving

\[
-\frac{d\kappa}{ds}v_{\text{max},c}^3 + 3\kappa a_{t\text{max}} v_{\text{max},c} - J_{c\text{max}} = 0,
\] (A.3)

that has exactly one real solution.

If \( \frac{d\kappa}{ds} \) is null, then feedrate limit is calculated as

\[
v_{\text{max},c} = \frac{J_{c\text{max}}}{3\kappa a_{t\text{max}}}. \] (A.4)
If a s-shape feedrate profile is used, however, the absolute value of maximum tangential acceleration that can be reached depends on maximum feedrate of the segment and may be smaller than $a_{t\text{ max}}$. From the equations of motion, this maximum value is calculated as

$$|a_{t\text{ max},j}| = \min \left( a_{t\text{ max}}, \sqrt{j_{m\text{ ax}}}, v_{\text{max},j}\right). \quad (A.5)$$

Assuming that $v_{\text{max},j}$ is small enough that $|a_{t\text{ max},j}| < a_{t\text{ max}}$, it can be expressed as

$$v_{\text{max},j} = \frac{\alpha_{t\text{ max},j}}{\lambda_{\text{max}}}. \quad (A.6)$$

So, the procedure to solve system of inequalities (A.1) is changed in the following way.

If $\frac{dx}{ds}$ is positive, $a_{t\text{ max},j}$ is calculated solving

$$-\frac{dx}{ds} \frac{a^2_{t\text{ max},j}}{\lambda_{\text{max}}} + 3 \kappa \frac{a^2_{t\text{ max},j}}{\lambda_{\text{max}}} \pm j_{\text{c max}} = 0, \quad (A.7)$$

that has two real solution and the correct one is the negative one, since the worst case is being considered.

If $\frac{dx}{ds}$ is negative, $a_{t\text{ max},j}$ is calculated solving

$$-\frac{dx}{ds} \frac{a^2_{t\text{ max},j}}{\lambda_{\text{max}}} + 3 \kappa \frac{a^2_{t\text{ max},j}}{\lambda_{\text{max}}} \pm j_{\text{c max}} = 0, \quad (A.8)$$

that has two real solution and the correct one is the positive one, since the worst case is being considered.

If $\frac{dx}{ds}$ is null, then $a_{t\text{ max},j}$ is calculated as

$$a_{t\text{ max},j} = \sqrt{\frac{j_{\text{c max}} \cdot j_{\text{m max}}}{3\kappa}}. \quad (A.9)$$

So, if the initial assumption (that $|a_{t\text{ max},j}| < a_{t\text{ max}}$) is verified, the feedrate limit must be calculated in the regular way (i.e. solving (A.2), (A.3) or (A.4), according to $\frac{dx}{ds}$).

References


