Real-time jerk limited feedrate profiling and interpolation for linear motor multi-axis machines using NURBS toolpaths

Krystian Erwinski *Member, IEEE*, Andrzej Wawrzak and Marcin Paprocki

***Abstract*—In this paper a NURBS toolpath feedrate profile generation algorithm for a biaxial linear motor control system is presented. High achievable velocities and accelerations of linear motor machines present new computational challenges in implementing feedrate generation and toolpath interpolation algorithms in real-time controllers. The proposed algorithm is capable of on-line generation of the feedrate profile with axial acceleration and jerk constraints. Each stage of the feedrate profiling algorithm is described with attention being given to both performance and implementation aspects. Furthermore an alternative to the commonly used Taylor series interpolation method is also tested to ensure minimal interpolation errors. The feedrate profiling and interpolation algorithms' implementation in a PC-based controller with real-time Linux kernel is described. Experimental results are presented that confirm that the algorithm is capable of limiting acceleration and jerk in the machine's axes and it's low computation time enables real-time on-line operation in a PC-based CNC controller.**

***Index Terms*—CNC, NURBS, feedrate profile, s-curve, linear motor, Linux RTAI, LinuxCNC, real-time**

I. Introduction

IN MODERN industry computerized numerically controlled (CNC) machines are in widespread use. Different areas of application pose different requirements and unique challenges on manufacturing speed, precision and efficiency. One area that is exhibiting rapid innovation are high performance cutting machines especially laser cutters [1]. Because cutting is done by a laser beam there is no physical contact between the tool and the workpiece so no load forces are present. Combined with the availability of high power fiber lasers this allows for very high working speeds and accelerations in the order of several meters per second and tens of meters per second squared respectively, especially for thin metal sheets. In order to achieve such high performance machine drivetrains based on rack and pinion and ballscrew mechanisms are replaced by direct drive permanent magnet linear motors (PMLM). This eliminates backlash and elasticity of the drivetrain which are significant factors that degrade machines' dynamic performance. All of the aforementioned qualities provide linear motor machines with unparalleled dynamic performance which greatly increases productivity [2].

This research was supported by the "Excellence Initiative—Research University" programme of Nicolaus Copernicus University, Poland as part of the Emerging Field: Automation and Control Systems group.

The authors are with the Departament of Automatics and Measurement Systems, Institute of Engineering and Technology, Faculty of Physics Astronomy and Informatics, Nicolaus Copernicus University in Torun, Torun, Poland. E-mails: erwin@umk.pl, awawrzak@umk.pl, marcin.paprocki@umk.pl

In order to utilize the high dynamic performance of linear motor driven machines a high performance control system is required. Nowadays high performance CNC controllers are often based on PC's with real-time operating systems and real-time communication fieldbus [3]. These controllers can run advanced motion control algorithms that are essential for fully utilizing the machines' capabilities.

Two of the most important tasks in CNC control are feedrate profiling and toolpath interpolation. Feedrate profiling is the process of computing the velocity tangent to the toolpath (feedrate) as a function of time or the toolpath parameter such as path length. Profiling is essential to ensure that the desired motion does not exceed the machine's capabilities in terms of drive power or dynamics of the mechanical components. This is done by shaping the feedrate profile so that it's derivatives such as acceleration and jerk are constrained. Toolpath interpolation is the process of determining successive position setpoints based on the feedrate profile and the toolpath geometry. The position setpoints are generated in constant time intervals which corresponds to the cycle time of the communication bus between the CNC controller and axis servo drives.

Some CNC controllers use trapezoid feedrate profiling. In this type of profiling only feedrate and tangent acceleration are limited. This means that acceleration changes sharply between zero, minimum and maximum values. This is not acceptable for high performance machines as large acceleration jumps can cause excessive vibration and premature wear of mechanical components. In such cases an additional constraint is used which limits the acceleration derivative or jerk. In this case feedrate consists of S-shaped segments. Many of the feedrate profiling methods presented in literature consider S-shaped jerk limited profiling [4], [5]. Some authors introduce higher order profiles [6] or profiles based on trigonometric functions [7]. This is useful to further limit vibrations but does not offer significant advantages for highly stiff machines such as those driven by linear motors.

Many of the algorithms proposed in literature only consider tangent velocity, acceleration and jerk limits. Some toolpaths with large curvature changes can cause significant acceleration and jerk violations despite limiting tangent values. This can cause the axis drives to reach their current limits and lead to an emergency stop. Therefore some authors introduce explicit axis constraints. Such an approach usually requires utilization of an optimization algorithm to simultaneously include axial constraints [8], [9], [10] as well as other limitations such as tracking error [11] or contour error [12]. Many feedrate optimization algorithms presented in the literature while employing sophisticated optimization methods do not consider the implementation aspects and computational burden placed on the CNC controller. Computationally demanding algorithms may be impractical for real-time implementation as the time it takes to execute the feedrate optimization is longer than the actual trajectory execution [13], [14]. This makes such methods difficult to apply in practice using most industrial PC-based controllers. The implementation and computational aspects considered in this paper are especially important in case of linear motor based machines capable of highly dynamic motion.

In this paper an algorithm is proposed for jerk-limited feedrate profile generation which directly limits acceleration and jerk in the machine's axes. The algorithm was developed and tested considering highly dynamic performance of linear motor machines such as laser cutters. Short 250 microsecond interpolation cycles are assumed which are common in high performance controllers and servo drives. The authors also integrate the proposed algorithm with an efficient and accurate RK2-PC interpolation method for NURBS toolpaths. Implementation of the proposed feedrate profiling method along with the interpolation algorithm in a real-time PC-based CNC controller is described. Experimental results are given that verify correct operation of the feedrate profile generation algorithm and interpolation algorithm as well as run times of both algorithms' implementations. The feedrate profile generation algorithm is also compared with results from several related papers.

II. NURBS TOOLPATH INTERPOLATION

Interpolation in most CNC machines involves determining positions on linear or circular toolpath segments defined by G-Code programs. If required shapes are complex they have to be approximated by a large number of linear segments. A state of the art solution is the direct interpolation of complex shapes defined as polynomial splines. Currently CAD software uses Non-Uniform Rational B-Splines for defining part shapes so this is the type of polynomial spline toolpath usually used in literature and will be considered in this paper. NURBS curves are also part of the STEP-NC standard which aims to replace G-Code in the role of a standard manufactured part definition.

NURBS curves are defined by a series of control points in cartesian space which form a control polygon [15]. Changing the location of the control point changes the curve's shape locally. Each point also has a weight value which defines how close to the control point the curve is located. Example NURBS curves ("bird" and "flower") with their control points and polygons are presented in Fig.1. NURBS toolpath interpolation is the process of determining the cartesian position on the NURBS curve in each interpolation cycle based on current feedrate and toolpath geometry. In each cycle the curve's parameter is incremented and then the corresponding point on the curve is determined using DeBoor's algorithm [15]. This algorithm also provides the curve's derivatives required for determining the parameter increment. NURBS

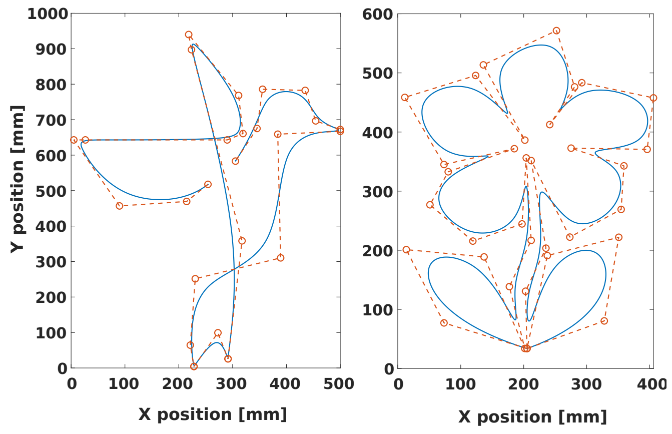

Fig. 1: "Bird" and "flower" NURBS toolpaths (blue) with marked control points and polygons (orange)

curves are usually parametrized with a unit-less parameter u where $u = 0$ marks the beginning of the curve and $u = 1$ marks it's end. Ideally a single timestep parameter increment should correspond to an arc-length increment of $V(t_i) \cdot \triangle t$ (where $V(t_i)$ - current feedrate, $\triangle t$ - time increment). The relationship between this parameter and the curve's arc length is non-linear and unique for every curve. This means that during interpolation the appropriate parameter increment has to be determined by approximating this relationship.

To approximate this relationship the second or third order Taylor series method is often used. The Taylor series method is a closed form solution which enables computation of next interpolated position in one step. It's accuracy is often sufficient if the toolpath's curvature does not vary significantly and feedrates are not large. In linear motor driven laser cutting machines feedrates can be even 10 times higher than in typical milling machines. Under these conditions truncation errors in the Taylor expansion method cause interpolation errors which result in feedrate fluctuations [16]. Furthermore these errors accumulate over each interpolation step as subsequent parameter increments are based on previous parameter values. Several solutions have been proposed. A simple improvement is to increase the order of Taylor's interpolation formula. This however requires computation of higher order curve derivatives and does not improve accuracy significantly for high feedrates and curvatures. Another solution is to globally approximate the relationship between the NURBS toolpath parameter and arc-length [17], [18]. This can usually be done by a Feed Correction Polynomial (FCP) but requires additional computation steps before actual interpolation. For large, complex curves this approximation would require large amounts of memory and a high order spline to reduce interpolation errors. Finally a general family of methods called predictor-corrector interpolation have been developed [19], [20], [21]. The general idea of this approach is to produce a coarse parameter increment or increments first, compute corresponding arc-length and then produce the final parameter value based on arc-length error.

In this work a second order Runge-Kutta predictor-corrector (RK2-PC) method presented in [21] is used as a real-time interpolator. A second order method is used due to simple implementation using only first order derivatives while simultaneously offering good interpolation accuracy provided by the correction step. This is always a two-step method and no additional corrector iterations are required. In the first stage of this method an initial parameter value is computed using the following equations.

$$\tilde{u}_{i+1} = u_i + \frac{1}{2}(K_1 + K_2)\Delta t \tag{1}$$

$$K_1 = \frac{du}{dt} = \frac{F(t_i)}{||C'(u_i)||} \quad K_2 = \frac{F(t_i)}{||C'(u_i + K_1\Delta t)||} \tag{2}$$

where: $C'(u)$ is a vector first derivative of the NURBS curve and $F(t_i)$ is the current feedrate. The obtained parameter value is based on the averaged parametric velocity of the curve at the initial location (K_1) and parametric velocity at a secondary location (K_2). Secondary location parameter value is computed by a trial step obtained by executing a first order Taylor interpolation increment. This step is equivalent to Heun's method [19]. In order to obtain curve position and first derivative for the trial step DeBoor's algorithm is used.

In the corrector phase of the interpolation algorithm a parameter correction increment Δu_{i+1} is computed so that the arc-length difference between the initial position and the final position is equal to the desired arc-length.

$$||C(\tilde{u}_{i+1} + \Delta u_{i+1}) - C(u_i)|| = F(t_i) \cdot \Delta t \tag{3}$$

The final position can be rewritten using first order Taylor expansion and substituted into equation 3. After squaring both sides the following equation is obtained.

$$||C(\tilde{u}_{i+1}) + C'(\tilde{u}_{i+1})\Delta u_{i+1} - C(u_i)||^2 = F(t_i)^2 \cdot \Delta t^2 \tag{4}$$

Equation 4 can be rewritten as a quadratic equation:

$$a \cdot (\Delta u_{i+1})^2 + b \cdot \Delta u_{i+1} + c = 0 \tag{5}$$

$$a = ||C'(\tilde{u}_{i+1})||^2 \tag{6}$$

$$b = 2 \cdot C'(\tilde{u}_{i+1}) \cdot (C(\tilde{u}_{i+1}) - C(u)) \tag{7}$$

$$c = ||C(\tilde{u}_{i+1}) - C(u)||^2 - F^2(t_i)\Delta t^2 \tag{8}$$

In order to increase accuracy of the RK2-PC method the distance $||C(\tilde{u}_{i+1}) - C(u)||$ is replaced by actual arc-length increment of the prediction step between curve points at u_i and $\tilde{u}_{i+1}$. This arc-length is computed by numerical integration using Simpson's method.The correction increment Δu_{i+1} can be found by finding one of the roots of the quadratic equation.

$$u_{i+1} = \tilde{u}_{i+1} + \frac{-b + \sqrt{a^2 - 4a \cdot b}}{2a} \tag{9}$$

The described predictor-corrector method can significantly reduce interpolation errors compared to the standard Taylor's expansion method. The method's accuracy compared to other commonly used interpolation methods is investigated in the experimental results section.

III. Feedrate profiling algorithm

The feedrate profile for a NURBS curve can be defined as a function of time, curve parameter or arc-length. In this paper time parameterization of the feedrate profile is used due to real-time requirements placed on the profiling algorithm. The feedrate profile can be expressed in closed form equations with appropriate coefficients determining the velocity, acceleration and jerk limits. The S-curve shape of the feedrate profile is assumed. This profile is commonly used in practice. It consists of up to 7 phases - 3 acceleration phases, one constant feedrate phase and 3 deceleration phases. Jerk which is the rate of change of acceleration is limited so the velocity is smooth which helps alleviate vibration and machine component wear. An example feedrate profile with corresponding acceleration and jerk profiles is presented in fig.2.

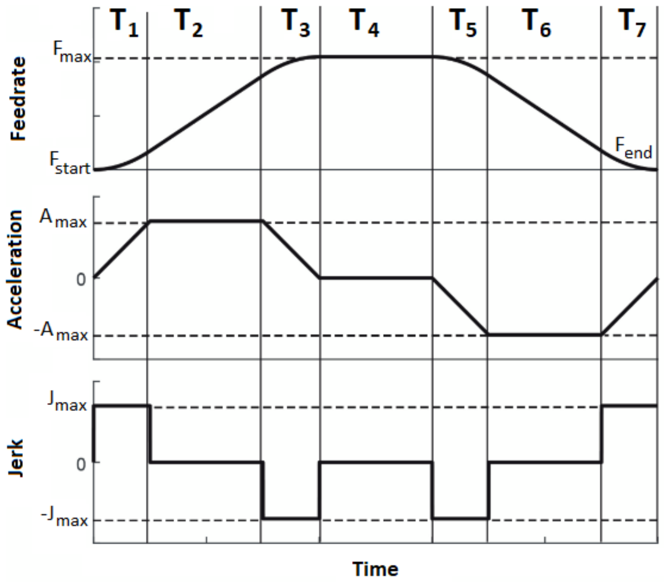

Fig. 2: Example feedrate, acceleration and jerk profiles. T_1–T_7 are the durations of each phase of the profile.

The simplest feedrate profile would consist of only one acceleration and one deceleration phase at the beginning and end of the toolpath. Because the toolpath is a curve, constant feedrate does not imply constant velocity in each of the machine's axes. Feedrate has to be limited in fragments of the toolpath with large curvature such as sharp corners due to sharp changes in axial velocity. This induces high acceleration and jerk values which can cause drive saturation and excessive tracking errors. This is even more important for linear motor machines capable of high feedrates and accelerations. On the other hand the feedrate should be as high as possible in order to maximize productivity.

Directly maximizing feedrate, acceleration and jerk while simultaneously keeping these values within limit for each axis is a difficult optimization problem usually requiring significant computation effort. An alternative solution is to limit the feedrate by using a Feedrate Limit Function (FLF). The FLF is a combination of linearized axial constraints combined in a single feedrate constraint dependent on the current curve geometry. There are different formulations of Feedrate Limit Functions in literature which usually combine axial velocity, acceleration and jerk constraints [13], [22]. One such formulation used in this paper can be expressed as:

$$F_{cc}^{lim} = \frac{2}{\Delta t}\sqrt{\rho^2(u) - (\rho(u) - \epsilon_{max})^2} \tag{10}$$

$$F_{ac}^{lim} = \sqrt{\frac{\min(A_{x_{max}}, A_{y_{max}})}{\kappa(u)}} \tag{11}$$

$$F_{jc}^{lim} = \sqrt[3]{\frac{\min(J_{x_{max}}, J_{y_{max}})}{\kappa^2(u)}} \tag{12}$$

$$\kappa(u) = \frac{1}{\rho(u)} = \frac{||C'(u) \times C''(u)||}{||C'(u)||^3} \tag{13}$$

$$FLF = \min\left(F_{max}, F_{c}^{lim}, F_{ac}^{lim}, F_{jc}^{lim}\right) \tag{14}$$

where: κ - curvature, ρ - curvature radius, ϵ_{max} - maximum allowable chord error.

The FLF defines the maximum allowable feedrate limited by a linearization of axial acceleration and jerk constraints as well as chord error constraints. This is done by limiting the centripetal acceleration and jerk at each point. Chord error is also limited which is the distance between a line connecting two adjacent interpolation points and an arc connecting these two points with a radius equal to the curvature radius (inverse of curvature) at that point. This error is caused by discretization of the toolpath curve during the interpolation process. The FLF is only an approximation of the feedrate limit imposed by aforementioned constraints. Planning the s-curve profile by only considering the FLF can result in axial acceleration and jerk limit violations even if the feedrate doesn't violate the FLF. This can usually occur when rapidly accelerating or decelerating in high curvature regions of the curve. The main advantage of this approach is significantly reduced computation effort by avoiding the solution of a constrained optimization problem which allows real-time implementation. The main contribution of this paper is a computationally efficient feedrate profile generation algorithm which utilizes the Feedrate Limit Function but also directly handles axial and jerk limits.

In order to plan the feedrate profile for a NURBS curve toolpath a linear forward scan of the toolpath is first performed. The scan determines critical points of the curve which correspond to areas of high curvature. In these points the feedrate has to be significantly reduced to satisfy the velocity, acceleration and jerk constraints. The feedrate profiles are planned in segments between these critical points. The algorithm to search for minimums of the Feedrate Limit Function (FLF) can be summarized as follows:

1. Sample the FLF at 500 equidistant points with respect to the NURBS curve parameter for each NURBS control point.
2. For each sample point check 5 adjacent points on both sides of each point to make sure that this point is a local minimum. Small minima which may result from numerical errors are therefore ignored.
3. Neighboring points of this point are then used as a bracket to refine the minimum using Brent's method [23], [24] with the coarse minimum point used as an initial guess.
4. If the NURBS arc-length between critical points is longer than the distance required to accelerate or decelerate for given A_{max} and J_{max} the segment is deemed to short and the starting critical point is discarded. The previous starting point is used (segments are merged)
5. An array of refined local minima of FLF (without infeasible minima) is obtained that indicates critical points on the NURBS curve for given limits of axial velocity, acceleration, jerk and chord error. Each of these points is characterized by the curve parameter value, maximum allowable feedrate value obtained from the FLF and arc-length value. The arc-length is determined by performing numerical integration of the arc-length function between each of the identified critical points. Integration is performed by using Simpson's method. The number of 500 test points per control point was chosen by trial and error to keep a balance between step size (accuracy of locating minima) and computation speed. It was found to locate all relevant critical points in all tested cases. The value of 5 neighboring points per test point was chosen by trial and error to ensure that the point contains an actual local minimum of the FLF and is not a glitch due to numerical errors.

The next stage of the algorithm involves computing the S-curve feedrate profiles for each segment between identified critical points. To fit the profile between the critical points for given maximum allowable feedrate, acceleration and jerk, the arc-length distance and initial and final feedrates of the segment have to be taken into account. The initial and final feedrates are equal to the feedrate minima determined by the FLF.

Each phase of the S-curve profile is characterized by it's duration which is given by:

$$T_1 = T_3 = T_5 = T_7 = \frac{A_{max}}{J_{max}} \tag{15}$$

$$T_2 = \frac{F_{max} - F_{start}}{A_{max}} - T_1 \tag{16}$$

$$T_6 = \frac{F_5 - F_6}{A_{max}} \tag{17}$$

For short distances and low jerk and acceleration limits the trapezoid acceleration profile is reduced to a triangular profile with the peak accelerations A_2 and A_6 equal or less than A_{max}. For triangular acceleration profile phase 2 is omitted if $A_{max}^2/J_{max} \geq F_{max} - F_{start}$ which means that the total increase in feedrate in phases 1 and 3 is equal or greater than the difference between start and maximum feedrate. Analogously for a triangular deceleration profile phase 6 is omitted if $A_{max}^2/J_{max} \geq F_{max} - F_{end}$. If segments 2 or 6 are omitted their duration is 0 and:

$$T_1 = T_3 = \sqrt{\frac{F_{max} - F_{start}}{J_{max}}} \tag{18}$$

$$T_5 = T_7 = \sqrt{\frac{F_{max} - F_{end}}{J_{max}}} \tag{19}$$

Certain combination of parameters may cause distance S_4 to be negative which means that the total distance when accelerating from F_{start} to F_{max} and decelerating from F_{max} to F_{end} is greater than the arc-length of the segment between the critical points. Phase has to be omitted and a peak feedrate value has to be determined numerically. A bisection search is performed in order to find F_{peak} for which the total acceleration and deceleration profile distance is less or equal to the segment length. The search is performed between $F_a = F_{max}$ and $F_b = \max(F_{start}, F_{end})$. In each iteration a new feedrate value is computed according to the following equation:

$$F_k = \begin{cases} \frac{F_{max}+\max(F_{start},F_{end})}{2} & k = 0 \\ F_{k-1} + \frac{\Delta F}{2^k} & S_{tot} < S_{seg} \\ F_{k-1} - \frac{\Delta F}{2^k} & S_{tot} > S_{seg} \end{cases} \tag{20}$$

$$\Delta F = F_{max} - \max(F_{start}, F_{end}) \tag{21}$$

Each iteration involves recomputing the feedrate profile with F_{max} being replaced by F_k. Then the total distance is determined and used to determine the next bisection step. Iterations are performed until maximum number of 30 iterations are reached or the change in candidate feedrate is less than 0.001 mm/s.

In order to directly handle axial acceleration and jerk constraints a novel extension to the typical S-curve planning method is proposed in this paper. A one dimensional search is used to find A_{max} and J_{max} values that do not violate axial constraints. The outline of the proposed methods is presented below:

1. After planning the s-curve feedrate profile between identified critical points according to the previously described steps the segment is interpolated using this profile and the previously described RK2 method.
2. At each interpolation point values of axis acceleration and jerk are computed. A sum of normalized acceleration and jerk constraint violations A_{viol} and J_{viol} is computed for all N interpolation points as well as the total duration of the segment T_{tot}:

$$A_{viol} = \sum_{i=0}^{N} \frac{A_{x,i} - A_{x_{max}}}{A_{x_{max}} \cdot N} + \sum_{i=0}^{N} \frac{A_{y,i} - A_{y_{max}}}{A_{y_{max}} \cdot N} \tag{22}$$

$$J_{viol} = \sum_{i=0}^{N} \frac{J_{x,i} - J_{x_{max}}}{J_{x_{max}} \cdot N} + \sum_{i=0}^{N} \frac{J_{y,i} - A_{y_{max}}}{J_{y_{max}} \cdot N} \tag{23}$$

3. If jerk is violated, new candidate values of jerk xL and xR are computed according to the golden-section search algorithm:

$$xL = B–(1/G) \cdot (B–A) \tag{24}$$

$$xR = A + (1/G) \cdot (B–A) \tag{25}$$

where: A, B - limits of the search space, initially $A = 0, B = J_{max}$, $G = \frac{\sqrt{5}+1}{2}$ - golden ratio.

4. A candidate value is chosen based on the following rules:
4.1. If both values cause jerk violations choose one with smaller J_{viol}
4.2. If both values do not cause violations choose one with shorter segment duration T_{tot}
4.3. Otherwise choose value without violation
5. Update variables depending on the previous choice:

$$xLB = xR;\ xR = xL;\ xL = B–(1/G) \cdot (B–A) \tag{26}$$

$$xR : A = xL;\ xL = xR;\ xR = A + (1/G) \cdot (B–A) \tag{27}$$

6. Repeat points 3-5 for acceleration.
7. Terminate search when $B - A < 0.1$
8. If the profile distance becomes too large to fit into the segment the end point feedrate also using golden-section search
9. In case the profile is still infeasible when the acceleration and jerk adjustment process terminates a failsafe mechanism is employed. The current profile is merged with the previous one. The current starting point is rejected and starting point of previous segment is used. The s-curve fitting procedure is repeated for new segment starting point.

The procedure described above is presented as a flow diagram on fig.3.

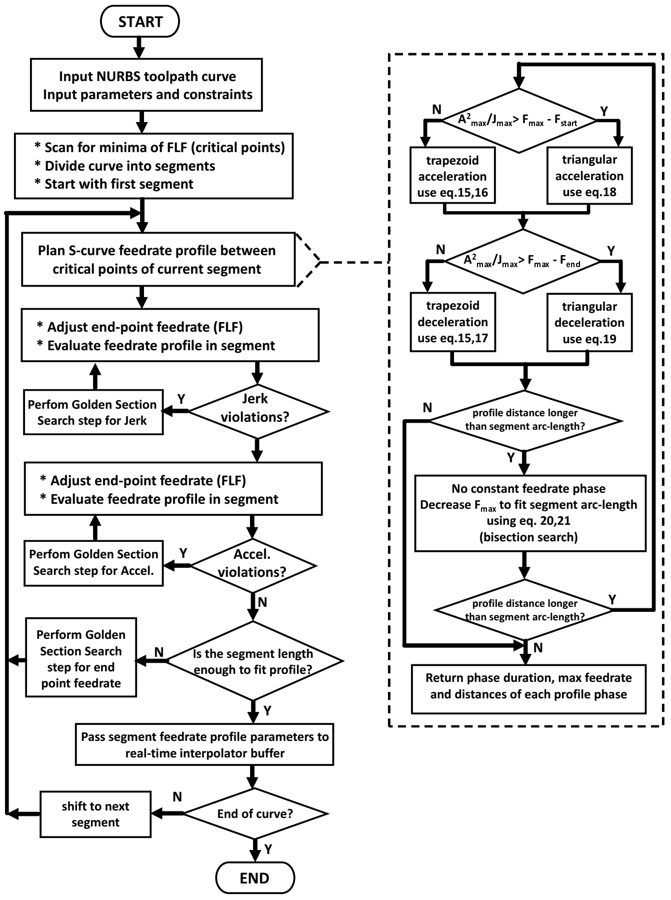
START
Input NURBS toolpath curve
Input parameters and constraints
* Scan for minima of FLF (critical points)
* Divide curve into segments
* Start with first segment
Plan S-curve feedrate profile between critical points of current segment
* Adjust end-point feedrate (FLF)
* Evaluate feedrate profile in segment
Jerk violations?
Y
Perfom Golden Section Search step for Jerk
* Adjust end-point feedrate (FLF)
* Evaluate feedrate profile in segment
Accel. violations?
Y
Perfom Golden Section Search step for Accel.
N
Is the segment length enough to fit profile?
N
Perform Golden Section Search step for end point feedrate
Y
Pass segment feedrate profile parameters to real-time interpolator buffer
End of curve?
N
shift to next segment
Y
END
$A^2_{max}/J_{max} > F_{max} - F_{start}$
N
trapezoid acceleration use eq.15,16
Y
triangular acceleration use eq.18
$A^2_{max}/J_{max} > F_{max} - F_{end}$
N
trapezoid deceleration use eq.15,17
Y
triangular deceleration use eq.19
profile distance longer than segment arc-length?
N
Y
No constant feedrate phase
Decrease F_{max} to fit segment arc-length using eq. 20,21 (bisection search)
profile distance longer than segment arc-length?
Y
N
Return phase duration, max feedrate and distances of each profile phase

Fig. 3: Flow diagram of the feedrate profile generation algorithm

When all parameters for the current segment are computed they are stored in a buffer waiting to be utilized by the real-time NURBS interpolator. Next segment is then processed in similar manner until the toolpath's end point is reached. In each discrete time step the interpolator computes the current feedrate by using the following formula:

$$F(t) = \begin{cases} J_{max} \cdot t^2 + F_0 & 0 \geq t \geq \tau_1 \\ A_1 \cdot t + F_1 & \tau_1 > t \geq \tau_2 \\ -\frac{1}{2} J_{max} \cdot t^2 + A_2 \cdot t + V_2 & \tau_2 > t \geq \tau_3 \\ F_{max} & \tau_3 \geq t > \tau_4 \\ -\frac{1}{2} J_{max} \cdot t^2 + F_4 & \tau_4 > t \geq \tau_5 \\ A_5 \cdot t + F_5 & \tau_5 > t \geq \tau_6 \\ \frac{1}{2} J_{max} \cdot t^2 + A_6 \cdot t + F_6 & \tau_6 > t \geq \tau_7 \end{cases} \tag{28}$$

$$\tau_k = \sum_{i=1}^{k} T_i \tag{29}$$

where: τ_k is the total duration of the feedrate profile at the end of phase k. The formula is given for a full profile with all seven phases. If a phase is not present (it's duration is 0) the relevant equation is omitted. The feedrate for each t is input into the predictor-corrector NURBS interpolation algorithm (RK2-PC) in order to determine the parameter value and position on the NURBS toolpath. This in turn allows to determine position setpoints for each of the machine's axes.

IV. Laboratory setup

To test the performance of the developed algorithm in an environment close to an actual CNC controller a PC-based control system was developed. PC-based controllers are often used to control high performance cutting machines [25]. The laboratory setup consists of a PC-based controller and two linear motor axes arranged to form a biaxial cartesian system. The PC used as the controller has a Core i5-6600 3.3Ghz processor, 8GB RAM and an SSD drive. The motors are Tecnotion TM6 iron core linear motors with a continuous force of 120N and peak force of 240N. The motors are mounted on aluminum frames with linear guides. Position feedback is provided by Renishaw RESOLUTE optical linear encoders with a nominal resolution of 5nm. Each linear motor axis is controlled by a Kollmorgen AKD P307 servo drive with 3A constant and 9A peak current. Each drive is in turn controlled by the PC-based controller via EtherCAT industrial fieldbus. The maximum feedrate of each unit F_{max} is limited to 2500mm/s, maximum acceleration A_{max} is set to 20000mm/s^2 and maximum jerk J_{max} is set to 200000mm/s^3. These values were determined experimentally to keep drive and motor current within nominal range of +/-3A with only short term overloads acceptable (+/-6A). While individually the linear motor units are capable of even higher values the additional load of the upper unit requires a decrease of acceleration and jerk in order to limit overloads and tracking error.

The PC runs a Linux operating system with a specialized real-time RTAI kernel. LinuxCNC open-source machine control software is installed on the system. It provides an additional real-time APIs (RTAPI and HAL) over RTAI's API which facilitates integration of user-made control modules. LinuxCNC itself has a CNC control module but it only offers trapezoidal feedrate profiling and no direct NURBS interpolation and is not used in this work. The feedrate profiling algorithm is implemented on the PC using LinuxCNC's RTAPI and HAL.

The structure of the feedrate profile generation and NURBS interpolation algorithms implemented on the PC-based controller is presented on fig.4. All processes running on the PC can be run as user-space or in kernel-space. The feedrate profiling algorithm is implemented as a user-space process. The NURBS interpolator is implemented as a real-time module running on the same real-time thread as the EtherCAT master stack. This thread is called every 250μs which is the lowest possible cycle of the Kollmorgen AKD servo drives. In every cycle new setpoints are computed by the interpolator and sent

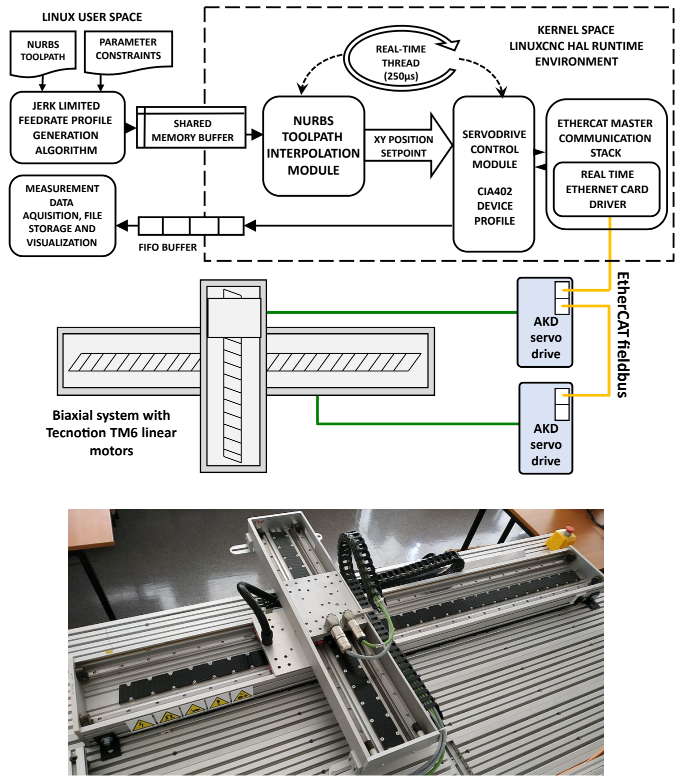
LINUX USER SPACE
NURBS TOOLPATH
PARAMETER CONSTRAINTS
JERK LIMITED FEEDRATE PROFILE GENERATION ALGORITHM
SHARED MEMORY BUFFER
MEASUREMENT DATA AQUISITION, FILE STORAGE AND VISUALIZATION
FIFO BUFFER
REAL-TIME THREAD (250µs)
KERNEL SPACE LINUXCNC HAL RUNTIME ENVIRONMENT
NURBS TOOLPATH INTERPOLATION MODULE
XY POSITION SETPOINT
SERVODRIVE CONTROL MODULE
CIA402 DEVICE PROFILE
ETHERCAT MASTER COMMUNICATION STACK
REAL TIME ETHERNET CARD DRIVER
EtherCAT fieldbus
AKD servo drive
AKD servo drive
Biaxial system with Tecnotion TM6 linear motors

Fig. 4: Laboratory setup schematic and picture

to the drives. Beside the main interpolation functions auxiliary functions such as homing, error handling or data acquisition are implemented. The profile generator and the interpolator are connected via a shared memory buffer. Feedrate profile computation is performed asynchronously to the interpolator so that the more complex and non-deterministic feedrate profiling algorithm can be run concurrently. The profiling module generates enough feedrate values to fill its local buffer and waits until the buffer is emptied to fill it again. A 10ms thread polls the user-space buffer and copies blocks of data from the profiling module. The interpolator module constantly reads the shared buffer and uses the stored values to interpolate the NURBS toolpath in 250μs intervals. This cycle continues until the curve is fully interpolated. The real-time interpolator module and the user-space profile generator run on different processor cores with the real-time part being isolated from the Linux operating system. This ensures real-time performance and stability which is paramount when controlling highly dynamic linear motor drives. The feedrate profile user-space module has full access to the Linux file system and can be easily extended with a GUI or other modules. The real-time module which works in kernel space is much more restricted. It cannot access files, and can interact directly only with other real-time kernel modules.

In order to communicate with the servo drives an EtherCAT software master stack (IGH EherCAT master) was also integrated in the system. The EtherCAT master is implemented as a real-time network card driver based on Linux e1000e driver and a kernel stack module. A software interface module between the HAL real-time API and the EtherCAT stack was developed which enables control of the Kollmorgen AKD drives in various operating modes. This interface conforms with the Can In Automation device profile 402 (CiA402) which is required by the servo drives. The software interface allows direct communication with the NURBS real-time interpolator module. On the hardware level communication is handled by a standard Intel network interface card therefore no dedicated hardware devices are required.

V. Experimental results

To validate the performance of the developed feedrate profiling algorithm the example NURBS toolpaths from fig.1 are used. The toolpaths were run on the linear motor lab setup with a maximum feedrate of $F_{max} = 2500 mm/s$ and two sets of acceleration and jerk limits:
1) $A_{max} = 10000\, mm/s^2$, $J_{max} = 60000\, mm/s^3$
2) $A_{max} = 15000\, mm/s^2$, $J_{max} = 200000\, mm/s^3$
The interpolation period was set to 0.25ms and the chord error limit to 0.001mm for all cases. Figures 5 - 8 present the feedrate profile, acceleration profiles for both axes and jerk profiles for both axes. Dashed lines on the feedrate profile plot indicate feedrate constraints generated by limiting chord error (orange), centripetal acceleration (green), centripetal jerk (violet). Critical points (marked as purple dots) indicate minima's of the Feedrate Limit Functions.

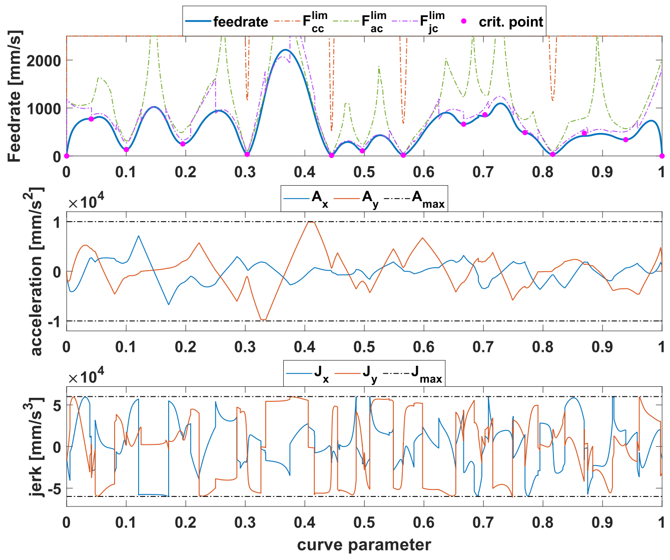

Fig. 5: Feedrate, axial acceleration and jerk profiles for "bird" curve for $A_{max} = 1e^4 mm/s^2$, $J_{max} = 6e^4 mm/s^3$

It can be seen that the axial constraints are never violated for any of the generated feedrate profiles. Some of the critical points had their feedrates lowered by the algorithm compared to initial minima obtained from the FLF. This was necessary to enable fitting of the s-curve profile while keeping the axial limits within bounds. It can also be seen that in some regions the feedrate is lower than the constraint components of the FLF. This is because the FLF is just an approximation and cannot be relied upon to constrain the axial acceleration and jerk at all times. The proposed method effecively and consistently limits these values.

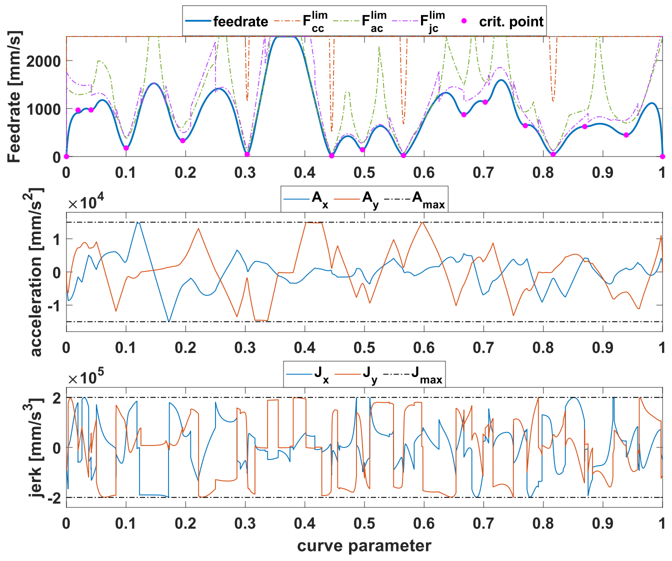

Fig. 6: Feedrate, axial acceleration and jerk profiles for "bird" curve for $A_{max} = 1.5e^4 mm/s^2$, $J_{max} = 2e^5 mm/s^3$

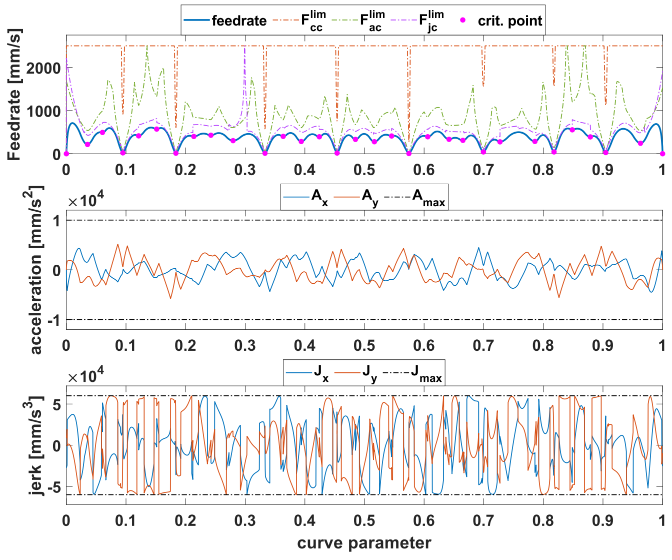

Fig. 7: Feedrate, axial acceleration and jerk profiles for "flower" curve for $A_{max} = 1e^4 mm/s^2$, $J_{max} = 6e^4 mm/s^3$

The plots on figures 9-12 present actual tracking error recorded from linear motor drives during interpolation of the test toolpaths. The errors are relatively small with peak values well below 0.1 mm. The tracking error exhibits larger values corresponding to steeper sections of the acceleration profile which is to be expected. No sudden error spikes are present which proves that the algorithm generated a smooth feedrate profile and it's interpolation was done correctly. The tracking error perturbations are caused by strong cogging forces typical for iron core linear motors. These forces are an external disturbance of approximately sinusoidal characteristics. The tracking error perturbation can be seen because linear motors are direct drives and a high resolution linear optical encoder is used. This effect is especially visible for lower velocities

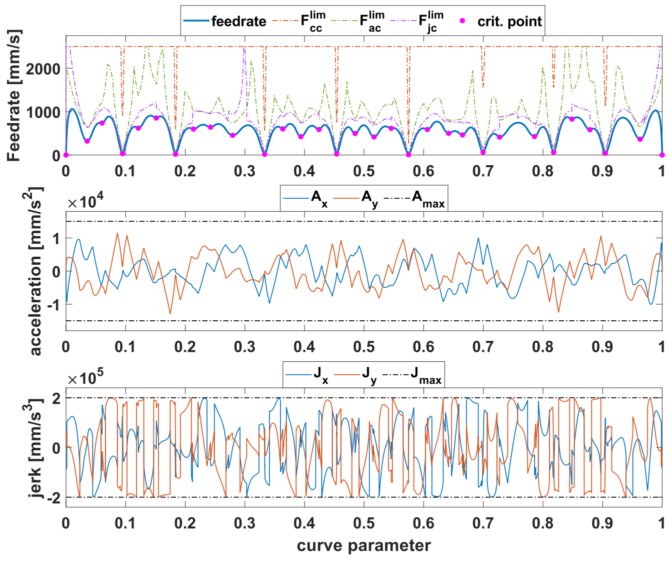

Fig. 8: Feedrate, axial acceleration and jerk profiles for "flower" curve for $A_{max} = 1.5e^4 mm/s^2$, $J_{max} = 2e^5 mm/s^3$

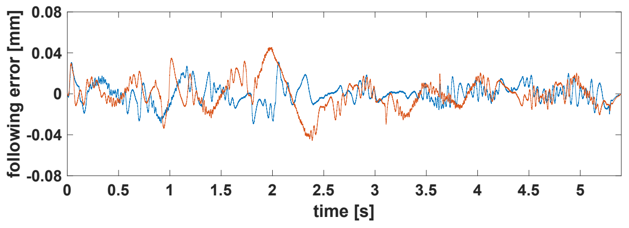

Fig. 9: Tracking error for X (blue) and Y (orange) axes for "bird" curve, $A_{max} = 1e^4 mm/s^2$, $J_{max} = 6e^4 mm/s^3$

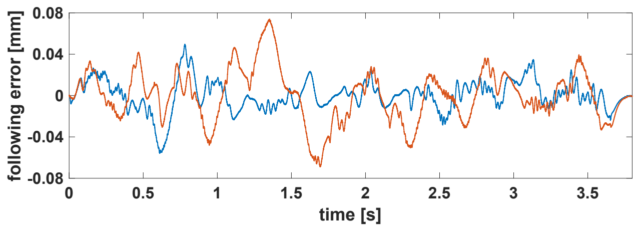

Fig. 10: Tracking error for X (blue) and Y (orange) axes for "bird" curve, $A_{max} = 1.5e^4 mm/s^2$, $J_{max} = 2e^5 mm/s^3$

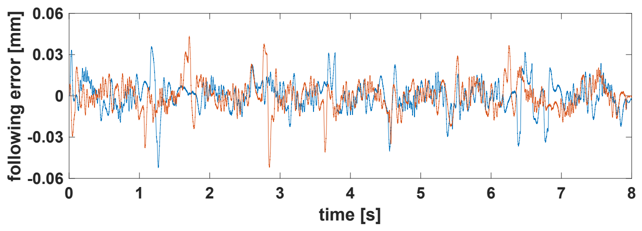

Fig. 11: Tracking error for X (blue) and Y (orange) axes for "flower" curve, $A_{max} = 1e^4 mm/s^2$, $J_{max} = 6e^4 mm/s^3$

because at higher velocities the cogging effect is opposed by inertia and hidden by resolution of the encoder. The velocities in the "flower" curve are lower than for the "bird" curve which is the reason why the disturbances are more pronounced in

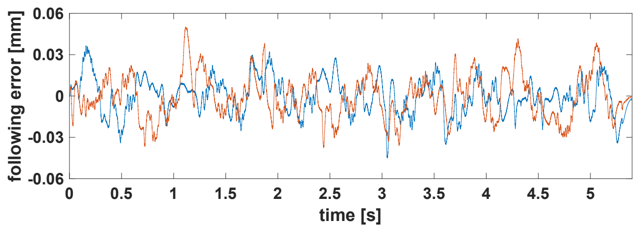

Fig. 12: Tracking error for X (blue) and Y (orange) axes for "flower" curve, $A_{max} = 1.5e^4 mm/s^2$, $J_{max} = 2e^5 mm/s^3$

fig.11 and 12. Compensation of cogging forces via current feedforward over the EtherCAT bus is planned to be added in future work.

The interpolation accuracy and computation time of the RK2-PC NURBS interpolation method was verified. At each interpolation step the actual arc-length between the previous and current parameter value was computed using the adaptive Simpson's method. The interpolation error is the difference between this arc-length and the desired distance $F_i(t) \cdot \Delta t$. Second aim of the experiment was to determine real-time capabilities of the presented interpolation algorithm and it's implementation. The interpolation algorithm is implemented as a real-time module running on a separate processor core from the profiling algorithm. The interpolation module cycle time was measured by reading the processor's time stamp code (TSC) at the start of each real-time interpolation cycle and computing the time difference between cycles. Computation time and interpolation error measurement was carried out for each toolpath and parameter set using four different interpolation algorithms. Interpolation errors and computation times for the second and third order Taylor series methods as well as the Feed Correction Polynomial method were also measured for comparison. This amounted for over 200000 interpolation cycles for each algorithm. Results are presented in table I. It can be seen that the interpolation error of the RK2-

TABLE I: NURBS INTERPOLATION ERRORS AND COMPUTATION TIME

	Mean Squared Error [mm]	**Total Accumulated Error [mm]**	**Average Computation Time [μs]**
SOT	$5.412e^{-7}$	7.537	3.015
TOT	$1.311e^{-7}$	3.524	3.231
FCP	$4.301e^{-10}$	0.055	2.874
RK2 P-C	$1.618e^{-20}$	$5.748e^{-8}$	4.249

PC method is very small which means that the interpolation algorithm is accurate even for very high feedrates and highly variable curvatures. On the other hand the interpolation error of the commonly used 2nd and 3rd order Taylor (SOT, TOT) series method is much larger. This error is present at each interpolation period. This means that for long toolpaths the error can accumulate and cause the feedrate profile to become misaligned with the curve's curvature minima. This in turn drastically increases axial acceleration and jerk violations leading to drive current saturation and excessive tracking error. Another popular method - the Feed Correction Polynomial offers the lowest interpolation time and errors lower than the

Taylor series method. This is however offset by the necessity of performing an initial fit of the arc-length function.

The RK2-PC algorithm was the most computationally demanding of the four compared algorithms. Maximum recorded runtime was just below 6μs with an average of 4.249μs. This is just a fraction of the total thread cycle time of 250μs. This shows that the much improved accuracy was achieved at negligible increase in computation effort compared to the total thread cycle time. In fact EtherCAT communication stack has a much bigger impact on the total thread time than interpolation with the total running time of interpolation, communication and data acquisition less than 30μs. This means that the proposed algorithm can be easily implemented on a PC with a real-time operating system. Short computation times ensure that on-line profiling and interpolation is viable. Such a short interpolation cycle means that the proposed method can be used even with state of the art servo drives with position loop cycles less that 100μs [26]. This is important as shorter interpolation cycles mean less interpolation error when machining complex trajectories at high speed. Compared to other methods presented in literature the RK2-PC method is a two step method (one prediction step and one correction). Many proposed predictor-corrector methods use an iterative process to determine the correct parameter increment [27], [20], [28]. This can lead to an increase in computation time. This is important because interpolation is performed many times during the feedrate planning process to determine axial constraints violation. Interpolation is finally performed in hard-real time context and is coupled with fieldbus communication stack so any disruptions to real-time capabilities can lead to errors in the control system. The RK2-PC method can achieve computation times only marginally longer than single step methods such as Taylor's series but offer very good accuracy which is typical of predictor-corrector methods. Multiple iterations are not required however which is an advantage of the RK2-PC method in this application. Many papers do not consider the real-time computation aspects of their algorithms so in many cases no computation time is given. The method proposed in this paper is tested on a real-time control system and computation times were verified to be very low.

The feedrate profile generation algorithm computation time was verified separately from the interpolation agorithm as the two run as separate software modules. Running time of the feedrate profile generation algorithm was measured by reading the clock() function at algorithm initialization and termination. The function returns a value of a monotonic system clock with a resolution of 1ms. The difference allows determination of the total computation time of the feedrate profile for the whole toolpath. The algorithm was run 100 times for each toolpath and each parameter set. Average running times for each toolpath and parameter set are given in table II compared with total execution time of the trajectory. It can be seen that the computation time of the profiling algorithm is much less than the total execution time of the trajectory. The profiling is run concurrently with trajectory interpolation and execution on the drives. The computation and execution time difference means that the profiling algorithm can supply profiled feedrate input to the interpolator without the risk of depleting the shared memory buffer between the two software modules.

TABLE II: PROFILING COMPUTATION TIME AND TRAJECTORY EXECUTION TIME

trajectory	**profiling average computation time [s]**	**trajectory execution time [s]**
”bird” set 1	0.608	5.665
”bird” set 2	0.412	3.728
”flower” set 1	1.109	7.971
”flower” set 2	0.778	5.374

The experimental results prove that the proposed algorithm is real-time capable and can be used for on-line generation of feedrate profiles with constrained axial acceleration and jerk. This is a significant advantage over most methods that utilize feedrate profile optimization to generate profiles with axial acceleration and jerk limitation. The algorithm is also more accurate than simple methods utilizing the feedrate limit function because the axial constraints are explicitly handled instead of relying on the approximate feedrate limit profided by the FLF.

VI. Comparison to similar works

The main advantage of the feedrate planning method proposed in this paper compared with similar works is its short computation time with explicit limitation of axis acceleration and jerk. In many papers the computation time can be longer than the trajectory execution time. For example in [13] the feedrate profile for a planar curve is computed in almost 14s while trajectory execution time is just above 8s. Other proposed methods suffer from similar problems [14]. Furthermore some papers do not even consider the computation time aspect. Some methods such as those presented in [29], [30], [21] that use the Feedrate Limit Function for feedrate scheduling use long constant feedrate sections which causes the axis capabilities to be significantly underutilized. It can be seen that the method proposed in this paper keeps the axial acceleration and jerks close to their limits where possible. Some other methods which use constrained optimization methods can also directly handle axial constraints. However these have the disadvantage of long computation time [10], [31]. Many of the feedrate generation methods do not consider practical implementation aspects and the presented algorithms are often implemented in Matlab. In this paper a full implementation is considered and the proposed feedrate profiling algorithm is tested on a PC-based real-time capable CNC controller connected to servo drives over the EtherCAT fieldbus. The control system used in this paper for experiments is similar to high-speed machine control systems. Many methods proposed in literature can only be computed off-line and only then carried over to an actual machine but the method proposed in this paper can perform efficient on-line feedrate scheduling in the machine controller. The algorithm proposed in this paper was compared with several related works from the literature. Algorithms were chosen that use the ”butterfly” NURBS planar curve in their experimental results section and the implementation platform (PC) is similar to that used in this paper. Values of feedrate, acceleration, jerk and chord error limits were set to values identical as those stated in the referenced papers. For each method the profiling computation time and trajectory execution time were measured. The comparative results were given in table III. Feedrate, axial accelration and jerk profiles generated for each compared paper were shown on figures 13,14,15

TABLE III: COMPARISON OF COMPUTATION TIME AND EXECUTION TIME (PROPOSED / ORIGINAL)

trajectory	**computation time [s]**	**execution time [s]**
Jia et. al [21]	0.162 / 4.8	4.61 / 8.7
Liu et. al [32]	0.239 / 5.3	6.31 / 6.86
Erwinski et. al [10]	0.32 / 90	8.8 / 9.2

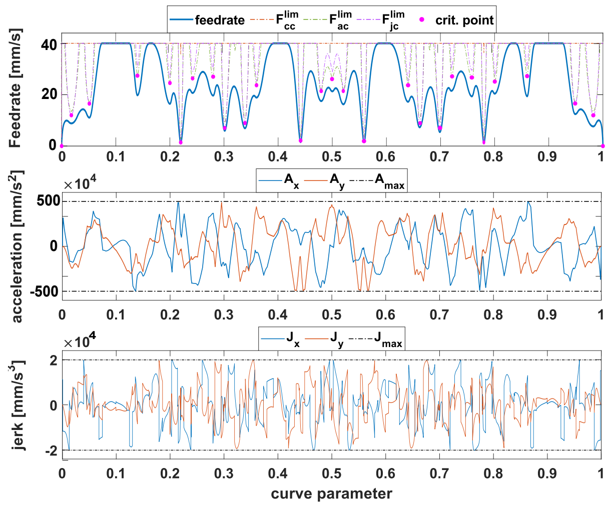

Fig. 13: Feedrate, axial acceleration and jerk profiles for "butterfly" curve (Jia et. al [21]) for $F_{max} = 40mm/s$,$A_{max} = 500mm/s^2$, $J_{max} = 2e^4mm/s^3$

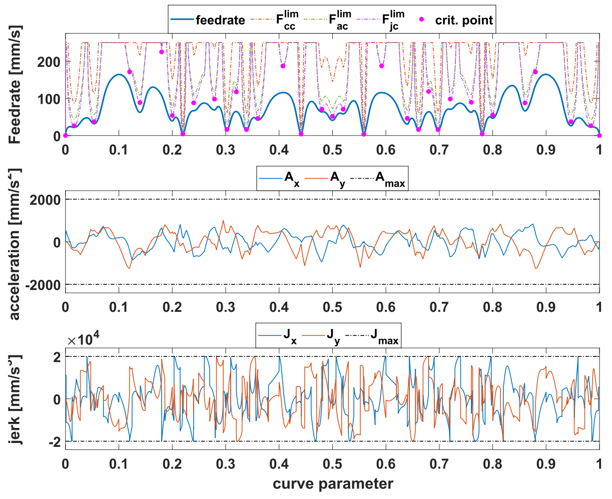

Fig. 14: Feedrate, axial acceleration and jerk profiles for "butterfly" curve (Liu et. al [32]) for $F_{max} = 250mm/s$, $A_{max} = 2000mm/s^2$, $J_{max} = 2e^4mm/s^3$

In all cases the proposed algorithm generated a proper feedrate profile without violations of axial acceleration and

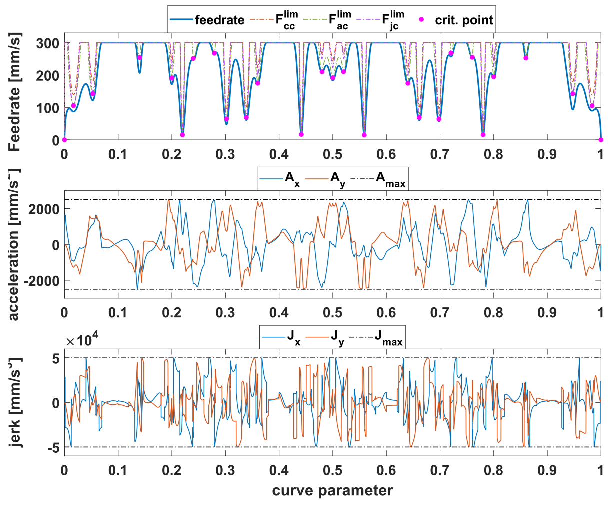

Fig. 15: Feedrate, axial acceleration and jerk profiles for "butterfly" curve (Erwinski et. al [10]) for $F_{max} = 300mm/s$, $A_{max} = 2500mm/s^2$, $J_{max} = 5e^4mm/s^3$

jerk limits. The computation time was significantly shorter than all of the compared methods. The trajectory execution times of the proposed method were also shorter. This proves that the algorithm can generate feedrate profiles both for highly dynamic linear motor machines as well as typical milling machines.

VII. CONCLUSION

In this paper a real-time capable feedrate profiling algorithm for NURBS curve toolpaths is presented which is the main contribution of this paper. It's main advantage is the ability to constrain axial acceleration and jerk while maintaining computational efficiency. A Feedrate Limit Function serves to indicate critical points on the toolpath between which an S-curve profile is planned with maximum values. Axial acceleration and jerk constraints are handled by using Golden-section one dimensional search of jerk and acceleration values. This avoids using highly demanding optimization approaches and allows the algorithm to be easily implemented on PC-based CNC controllers. This algorithm also uses only forward scanning of the toolpath instead of the popular bi-directional scanning which also decreases computation time.

A secondary contribution of this paper is the consideration of real-time performance of the proposed algorithm on a PC-based control system comparable to ones used in laser and plasma cutting machines. Experimental results prove that the algorithm is effective at generating the feedrate profile with constrained axial acceleration and jerk and is capable of on-line real-time operation. Tests were performed on an actual PC-based control system with a NURBS interpolation algorithm (2nd order Runge-Kutta) as well as an EtherCAT communication stack for direct control of linear servo motors. The accuracy and run time of the utilized interpolation method was verified and compared to other popular methods. The results show that using a more accurate interpolation method does not influence the computational effort of the controller in a significant way.

The algorithm proposed in this paper can perform efficient on-line feedrate scheduling. Short toolpath execution times as well as computation times much shorter than the trajectory execution tims can be obtained compared to other methods presented in literature. This means that the method can be directly applied to PC-based controllers of high speed linear motor based CNC machines such as laser cutters.

Future work will focus on improving the presented method including additional constraints such as tracking error and drive current. Further research is also planned which will focus on testing the proposed algorithm for machines with parallel kinematics such as Delta or H-Bot as well as 5-axis machines.

References

[1] M. Hajad, V. Tangwarodomnukun, C. Jaturanonda, and C. Dumkum, “Laser cutting path optimization using simulated annealing with an adaptive large neighborhood search,” *Int. J. Adv. Manuf. Technol.*, vol. 103, no. 1-4, pp. 781–792, 2019.
[2] Y. Altintas, A. Verl, C. Brecher, L. Uriarte, and G. Pritschow, “Machine tool feed drives,” *CIRP annals*, vol. 60, no. 2, pp. 779–796, 2011.
[3] K. Erwinski, M. Paprocki, L. M. Grzesiak, K. Karwowski, and A. Wawrzak, “Application of ethernet powerlink for communication in a linux rtai open cnc system,” *IEEE Trans. Ind. Electron.*, vol. 60, no. 2, pp. 628–636, 2012.
[4] X. Du, J. Huang, and L.-M. Zhu, “A complete s-shape feed rate scheduling approach for nurbs interpolator,” *J. Comput. Des. Eng.*, vol. 2, no. 4, pp. 206–217, 2015.
[5] H. Ni, C. Zhang, S. Ji, T. Hu, Q. Chen, Y. Liu, and G. Wang, “A bidirectional adaptive feedrate scheduling method of nurbs interpolation based on s-shaped acc/dec algorithm,” *IEEE Access*, vol. 6, pp. 63 794–63 812, 2018.
[6] H. Ni, J. Yuan, S. Ji, C. Zhang, and T. Hu, “Feedrate scheduling of nurbs interpolation based on a novel jerk-continuous acc/dec algorithm,” *IEEE Access*, vol. 6, pp. 66 403–66 417, 2018.
[7] Y. Wang, D. Yang, R. Gai, S. Wang, and S. Sun, “Design of trigonometric velocity scheduling algorithm based on pre-interpolation and look-ahead interpolation,” *Int. J. Mach. Tools Manuf.*, vol. 96, pp. 94–105, 2015.
[8] T. Mercy, N. Jacquod, R. Herzog, and G. Pipeleers, “Spline-based trajectory generation for cnc machines,” *IEEE Trans. Ind. Electron.*, vol. 66, no. 8, pp. 6098–6107, 2018.
[9] T.-C. Lu, S.-L. Chen, and E. C.-Y. Yang, “Near time-optimal s-curve velocity planning for multiple line segments under axis constraints,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 12, pp. 9582–9592, 2018.
[10] K. Erwinski, M. Paprocki, A. Wawrzak, and L. M. Grzesiak, “Pso based feedrate optimization with contour error constraints for nurbs toolpaths,” in *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 2016, pp. 1200–1205.
[11] Y. Zhang, P. Ye, M. Zhao, and H. Zhang, “Dynamic feedrate optimization for parametric toolpath with data-based tracking error prediction,” *Mech. Syst. Signal Process.*, vol. 120, pp. 221–233, 2019.
[12] 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, 2019.
[13] Y. Sun, M. Chen, J. Jia, Y.-S. Lee, and D. Guo, “Jerk-limited feedrate scheduling and optimization for five-axis machining using new piecewise linear programming approach,” *Sci. China Technol. Sci.*, vol. 62, no. 7, pp. 1067–1081, 2019.
[14] 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, pp. 1373–1388, 2013.
[15] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 2012.
[16] H. Ni, C. Zhang, C. Chen, T. Hu, and Y. Liu, “A parametric interpolation method based on prediction and iterative compensation,” *Int. J. Adv. Robot. Syst.*, vol. 16, no. 1, p. 1729881419828188, 2019.
[17] K. Zhao, S. Li, and Z. Kang, “Smooth minimum time trajectory planning with minimal feed fluctuation,” *Int. J. Adv. Manuf. Technol.*, vol. 105, no. 1-4, pp. 1099–1111, 2019.
[18] A. Dumanli and B. Sencer, “Robust trajectory generation for multi-axis vibration avoidance,” *IEEE/ASME Trans. Mechatronics*, 2020.
[19] M. Chen, W.-S. Zhao, and X.-C. Xi, “Augmented taylor’s expansion method for b-spline curve interpolation for cnc machine tools,” *Int. J. Mach. Tools Manuf.*, vol. 94, pp. 109–119, 2015.
[20] T.-Y. Wang, Y.-B. Zhang, J.-C. Dong, R.-J. Ke, and Y.-Y. Ding, “Nurbs interpolator with adaptive smooth feedrate scheduling and minimal feedrate fluctuation,” *Int. J. Precis. Eng. Manuf.*, vol. 21, no. 2, pp. 273–290, 2020.
[21] Z.-Y. Jia, D.-N. Song, J.-W. Ma, G.-Q. Hu, and W.-W. Su, “A nurbs interpolator with constant speed at feedrate-sensitive regions under drive and contour-error constraints,” *Int. J. Mach. Tools Manuf.*, vol. 116, pp. 1–17, 2017.
[22] H. Zhao, L. Zhu, and H. Ding, “A real-time look-ahead interpolation methodology with curvature-continuous b-spline transition scheme for cnc machining of short line segments,” *Int. J. Mach. Tools Manuf.*, vol. 65, pp. 88–98, 2013.
[23] R. Brent, *Algorithms for minimization without derivatives*.
[24] W. V. W. Press, S. Teukolsky and B. Flannery, *Numerical Recipes: The Art of Scientific Computing, 3rd edition*. Cambridge University Press, 2007.
[25] M. Paprocki, A. Wawrzak, K. Erwinski, K. Karwowski, and M. Klosowiak, “Pc-based cnc machine control system with linuxcnc software,” *Measurement Automation Monitoring*, vol. 63, 2017.
[26] T. Tarczewski, M. Skiwski, L. Niewiara, and L. Grzesiak, “High-performance pmsm servo-drive with constrained state feedback position controller,” *Bull. Pol. Ac.: Tech.*, vol. 66, no. 1, 2018.
[27] H. Zhao, L. Zhu, and H. Ding, “A parametric interpolator with minimal feed fluctuation for cnc machine tools using arc-length compensation and feedback correction,” *Int. J. Mach. Tools Manuf.*, vol. 75, pp. 1–8, 2013.
[28] S. Ji, T. Hu, Z. Huang, and C. Zhang, “A nurbs curve interpolator with small feedrate fluctuation based on arc length prediction and correction,” *Int. J. Adv. Manuf. Technol.*, vol. 111, no. 7, pp. 2095–2104, 2020.
[29] D.-N. Song and J.-W. Ma, “Interval partition-based feedrate scheduling with axial drive constraints for five-axis spline toolpaths,” *Int. J. Adv. Manuf. Technol.*, vol. 105, no. 11, pp. 4701–4714, 2019.
[30] Y. Sang, C. Yao, Y. Lv, and G. He, “An improved feedrate scheduling method for nurbs interpolation in five-axis machining,” *Precis. Eng.*, 2020.
[31] K. Erkorkmaz, Q.-G. C. 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, pp. 393–396, 2017.
[32] H. Liu, Q. Liu, and S. Yuan, “Adaptive feedrate planning on parametric tool path with geometric and kinematic constraints for cnc machining,” *Int. J. Adv. Manuf. Technol.*, vol. 90, no. 5, pp. 1889–1896, 2017.

Krystian Erwinski obtained his M.Sc. in Technical Physics at Nicolaus Copernicus University in 2008 and his Ph.D. in Automatics and Robotics at the Warsaw University of Technology in 2014. He is an assistant professor at the Institute of Engineering and Technology, Faculty of Physics Astronomy and Informatics, Nicolaus Copernicus University in Torun, Poland. His research interests include trajectory optimization for multi-axis machines, real-time control systems an industrial communication.

Andrzej Wawrzak obtained his M.Sc. in Electronics in 1986 at the AGH University of Science and Technology in Krakow. He is currently an assistant at the Institute of Engineering and Technology, Faculty of Physics Astronomy and Informatics, Nicolaus Copernicus University in Torun, Poland. His research interests include automation of technological processes, electrical drive control systems and numerical machine control systems.

Marcin Paprocki obtained his M.Sc. in Technical Physics at Nicolaus Copernicus University in 2006 and his Ph.D. in Automatics and Robotics at the Warsaw University of Technology in 2016. He is an assistant professor at the Institute of Engineering and Technology, Faculty of Physics Astronomy and Informatics, Nicolaus Copernicus University in Torun, Poland. His research interests include AI based predictive maintenance, real-time embedded control systems an industrial communication.