The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-021-07200-5

ORIGINAL ARTICLE

Accurate and computationally efficient approach for simultaneous feedrate optimization and servo error pre-compensation of long toolpaths—with application to a 3D printer

Heejin Kim[1] · **Chinedum E. Okwudire**[1]

Received: 29 November 2020 / Accepted: 3 May 2021

Abstract
Feedrate optimization (FO) and servo error pre-compensation (SEP) are often performed independently to improve the accuracy and speed, respectively, of computer-controlled manufacturing machines. However, this independent approach leads to excessive tradeoff between speed and accuracy. To address this issue, the authors have proposed a new concept of simultaneous FO and SEP (or FOSEP) where SEP is integrated into FO, yielding large reductions in motion time without sacrificing positioning accuracy relative to independent FO and SEP. However, in their prior work, the authors used linear programming to achieve FOSEP resulting in the following: (i) inaccuracy in enforcing nonlinear constraints and (ii) poor computational efficiency for long toolpaths. To address these two problems, this paper proposes a new approach for FOSEP using windowed sequential linear programming (SLP). The use of SLP improves the accuracy of FOSEP in enforcing nonlinear constraints; however, it lowers the computational efficiency of FOSEP. Windowing addresses the problem of low computational efficiency by applying SLP to FOSEP in small overlapping batches. A downside of windowed SLP is that it may lead to infeasibility in the optimization. This problem is resolved by smoothly switching between the optimal solution obtained using windowed SLP and a backup conservative solution in case of impending infeasibility. The proposed windowed SLP with smooth switching approach for FOSEP is validated in simulations where it significantly improves the accuracy and computational efficiency of FOSEP while guaranteeing feasibility. The practical benefits of the proposed approach for FOSEP is demonstrated in experiments on a 3D printer where it achieves up to 25% reduction in cycle time without sacrificing printing quality relative to the conventional approach of independent FO then SEP, both applied to a long toolpath.

Keywords Feedrate optimization · Pre-compensation · Filtered B-splines · Tracking error · CNC · Linear programming · Vibration compensation · 3D printing

1 Introduction

Productivity and quality are two conflicting requirements that must be met by computer controlled manufacturing machines, such as machine tools, 3D printers, and motion stages [1]. This trade-off is handled in practice by maximizing the speed of the machines' feed drives so long as desired accuracy (tolerance) limits are not violated. One major source of inaccuracy in feed drives is servo errors, which could be caused by commanded motion or disturbance forces like friction and manufacturing process forces. Due to the limited bandwidth of servo controllers, errors caused by commanded motion (i.e., motion-induced servo errors) often increase with speed; hence, they play a critical role in the tradeoff between speed and accuracy in manufacturing machines.

Motion-induced servo errors could be reduced without sacrificing motion speed using servo error pre-compensation (SEP). The idea behind SEP (also known as feedforward tracking control) is to modify motion commands to a machine using knowledge of the machine's servo dynamics in order to reduce servo errors. SEP could be performed offline or online, e.g., in the interpolator of a computer numerical controller (CNC). Examples of SEP

✉ Chinedum E. Okwudire
okwudire@umich.edu

[1] Department of Mechanical Engineering, University of Michigan, 2350 Hayward, Ann Arbor, MI, USA

Published online: 20 May 2021

approaches in the literature include zero-phase tracking error controller [2], iterative method [3], path-modification via inverse dynamics [4], input shaper [5], analytical prediction and compensation of contour error [6], model predictive control framework [7], trajectory pre-filter [8], cross-coupled pre-compensation [9], mirror compensation with Taylor's expansion [10], adaptive cross-coupled prediction compensation [11], cross-coupled dynamic friction control [12], contour error pre-compensation using Ferguson curve [13], offline gain adjustment [14], and filtered B splines [15, 16]. However, existing SEP approaches only focus on reducing servo error without trying to maximize feedrate.

Conversely, there are numerous works on feedrate optimization (FO) subject to tolerance constraints. The standard approach to introduce tolerance constraints into FO is by imposing velocity, acceleration and jerk limits [17–20], which indirectly limit the magnitude of servo errors. However, a more accurate approach to maintain a desired tolerance is to explicitly impose tracking or contour accuracy constraints on FO [21–30].

In practice, when FO and SEP are combined, it is done sequentially as illustrated in Fig. 1a. FO is first performed to maximize speed and an optimal motion trajectory is generated. Then, SEP is applied to the generated trajectory to compensate impending servo errors. However, this sequential approach leads to sub-optimality because it provides no systematic way for FO to benefit from the reduction of error provided by SEP in maximizing feedrate. To address this deficiency, the authors have recently proposed a new concept of simultaneous FO and SEP (i.e., FOSEP)—see Fig. 1b [30]. In FOSEP, SEP is integrated into FO as a constraint thus allowing FO to benefit from the error reduction provided by SEP. When applied to a 3D printer and precision motion stage, FOSEP was shown to reduce cycle time up to 47% compared to the conventional approach of independent (or sequential) FO and SEP. However, in the authors' prior work [30], FOSEP was achieved using linear programming (LP). As a result, it had two major shortcomings: (i) inaccuracy in enforcing nonlinear constraints due to linearization errors; and (ii) poor computational efficiency for long trajectories because it processed the full motion trajectory in one batch. To address these shortcomings, this paper makes the following original contributions:

1. It proposes a windowed sequential linear programming (Win-SLP) approach for FOSEP, where LP is iteratively applied to FOSEP in small overlapping batches, thus significantly improving its accuracy and computational efficiency.
2. It proposes smooth switching between the optimal Win-SLP solution and a conservative backup solution to address the potential for infeasibility in achieving FOSEP using Win-SLP, thus guaranteeing the existence of a feasible solution.
3. Using the proposed Win-SLP, it demonstrates the practical benefits of FOSEP on long toolpaths in experiments on a 3D printer, leading to 25% reduction in cycle time without sacrificing motion accuracy compared to independent FO and SEP.

The rest of this paper is organized as follows: Section 2 gives a brief overview of FOSEP using LP, as proposed in the authors' prior work [30]. Section 3 presents the proposed approach for FOSEP using Win-SLP. A series of simulations are carried out in Section 4 to validate the effectiveness of the proposed approach with regard to accuracy, computational efficiency and feasibility. Then, in Section 5, the practical benefit of the proposed Win-SLP approach for FOSEP is demonstrated in experiments on a 3D printer with long tool paths. Conclusions and future work are presented in Section 6.

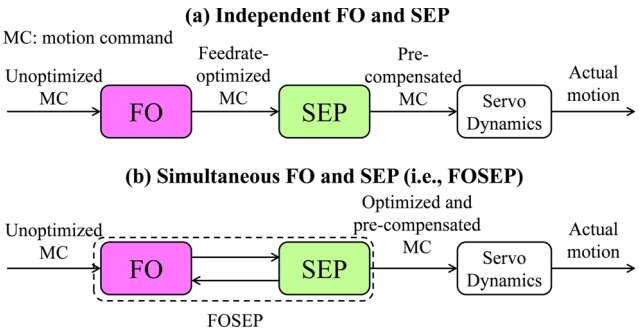

Fig. 1 Illustration of **a** independent FO and SEP and **b** simultaneous FO and SEP (i.e., FOSEP)

2 Overview of FOSEP using linear programming

Figure 2 illustrates an arbitrary, curved toolpath in the x-y plane with path parameter $s \in [0, 1]$. Note that s is a function of time t (i.e., $s = s(t)$). Let $x_d = f(s)$ and $y_d = g(s)$ denote a pair of parametric equations in s, representing the x and y components of desired position,

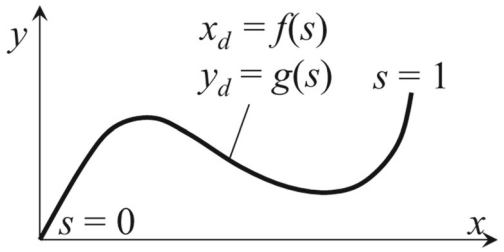

Fig. 2 Parametric planar curve as function of path variable, s

respectively. In our prior work [30], we introduced the concept of FOSEP using LP. To do this, s was discretized as $s(k)$, $k = 0, 1, 2, ...N-1$, with fixed sampling interval, T_s, and expressed as a vector $\boldsymbol{s} = \{s(0), s(1), ..., s(N-1)\}^{\mathrm{T}}$. Then, FO with kinematic constraints F_{max} and A_{max} on feedrate and axis acceleration, respectively, was formulated as:

$$\begin{aligned} &\min_{\boldsymbol{s}} \sum_{k=0}^{N-1} -s(k) \\ &\text{s.t.} \quad s(k-1) \le s(k) \le 1 \quad \forall k = 1, 2, ..., N-1; \\ &\quad L\frac{D[\boldsymbol{s}]}{T_s} \le \boldsymbol{F}_{max}; \\ &\quad \left|\frac{D^2[\hat{\boldsymbol{x}}_d]}{T_s^2}\right|, \left|\frac{D^2[\hat{\boldsymbol{y}}_d]}{T_s^2}\right| \le \boldsymbol{A}_{max} \end{aligned} \tag{1}$$

Here, D denotes a finite difference operator, while $\boldsymbol{F}_{max}$ and $\boldsymbol{A}_{max}$ are vectorized representations of the corresponding kinematic limits, F_{max} and A_{max}; L denotes the total length of path $(x_d(k), y_d(k))$ traversed from $k = 0$ to $N-1$. The variables $\hat{x}_d$ and $\hat{y}_d$ are linearized versions of $x_d = f(s)$ and $y_d = g(s)$, which are generally nonlinear in s. At each time step k, they are linearized with linearization points $s_c(k)$ estimated from an initial un-optimized trajectory as:

$$\hat{x}_d(k) = \frac{\partial f(s)}{\partial s}\Big|_{s=s_c(k)} \cdot (s(k) - s_c(k)) + f(s_c(k)) \tag{2}$$

$\hat{y}_d(k)$ is obtained by linearizing $g(s)$ in the same manner; $\hat{\boldsymbol{x}}_d$ and $\hat{\boldsymbol{y}}_d$ are vectorized versions of $\hat{x}_d(k)$ and $\hat{y}_d(k)$, respectively, similar to $\boldsymbol{s}$. This notation is maintained hereafter. The linearization procedure is based on the assumption that $\boldsymbol{s}_c$ is a feasible trajectory (that satisfies all constraints). In practice, this is often a conservative solution that is known from experience to give satisfactory performance. In FOSEP, it is assumed that $\boldsymbol{s}_c$ is given.

Next, linearized desired x-axis position, $\hat{\boldsymbol{x}}_d$, is used to generate modified position command $\hat{\boldsymbol{x}}_{dm}$ using a SEP process represented by C_x. A linear (and stable) model, $\hat{G}_x$, of the actual servo dynamics, G_x, is used to estimate the x-axis position as $\hat{\boldsymbol{x}}$ and tracking error as $\hat{\boldsymbol{e}}_x = \hat{\boldsymbol{x}}_d - \hat{\boldsymbol{x}}$. A similar process is followed for the y-axis using C_y and $\hat{G}_y$. Then, tracking error limit is imposed as an additional constraint on the LP formulation as:

$$\left|\hat{\boldsymbol{e}}_x\right| = \left|(\mathbf{I} - \hat{\boldsymbol{G}}_x \boldsymbol{C}_x)\hat{\boldsymbol{x}}_d\right| \le \boldsymbol{E}_{max} \tag{3}$$

where $\boldsymbol{E}_{max}$ is the vectorized form of the maximum allowable tracking error E_{max}; $\boldsymbol{C}_x$, $\hat{\boldsymbol{G}}_x$, $\boldsymbol{C}_y$ and $\hat{\boldsymbol{G}}_y$ are matrix (lifted) versions of C_x, $\hat{G}_x$, C_y and $\hat{G}_y$, respectively, based on the finite impulse response (FIR) of the corresponding system dynamics [16]; and $\mathbf{I}$ is the identity matrix. A similar constraint is imposed on $\hat{\boldsymbol{e}}_y$. The implication is that a model of SEP is incorporated into FO, yielding FOSEP. Since the objective and constraints are linear, they can be optimized using LP as in [30]. Figure 3 illustrates a block diagram of FOSEP. The optimized x_d and y_d from FOSEP are applied to the actual servo dynamics, G_x and G_y, pre-compensated using C_x and C_y, respectively.

It is worth pointing out that C_x and C_y can be any SEP (feedforward tracking control) method, e.g., [2, 4–6, 8, 9, 11, 13, 15, 16], as long as it has linear dynamics. Also, note that in lieu of tracking error constraint in Eq. 3, contour error constraint could be imposed (see [30] for example). Observe that axis jerk limits (J_{max}) can readily be introduced into Eq. 1, following a similar approach as axis acceleration limits. However, for the sake of brevity, the imposition of axis jerk limits are not described in this paper, though axis jerk limits are implemented and used in simulations and experiments in Sections 4 and 5.

3 A new approach for FOSEP using windowed sequential linear programming (Win-SLP)

A major problem with the LP-based FOSEP approach presented in Section 2 is that the accuracy of the solution

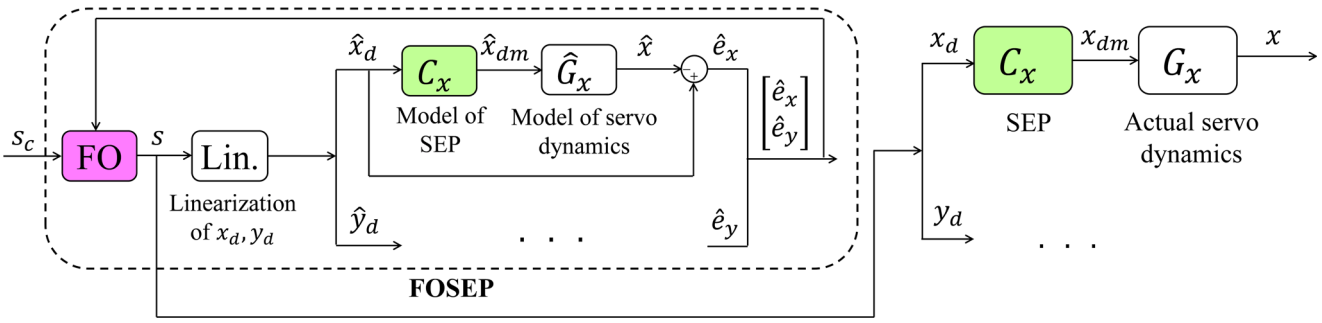

Fig. 3 Block diagram of FOSEP using LP (Note: y-component of SEP and servo dynamics are omitted for simplicity)

highly depends on the trajectory $\boldsymbol{s}_c$ used to initialize the solution and linearize the constraints in Eqs. 1 and 3. As $\boldsymbol{s}$ tends to $\boldsymbol{s}_c$, the linearization error converges to zero; however, when $\boldsymbol{s}$ moves further away from $\boldsymbol{s}_c$, depending on the polynomial order of the nonlinearity in Eqs. 1 and 3, the linearization error diverges. Another issue with the approach in Section 2 is that it processes all N points of the motion trajectory in one shot. This approach is reasonable for short trajectories where N is small [30], but is impractical for long trajectories (which are typical in manufacturing) due to the curse of dimensionality. In this section, we present an approach to address these two issues using windowed sequential linear programming (SLP), together with a scheme to guarantee feasible solutions.

3.1 Formulation of FOSEP using SLP

SLP is an optimization technique for solving nonlinear optimization problems iteratively using LP [33]. Given an estimate of the optimal solution, a sequence of first-order approximations (i.e., linearization) of the problem is executed. In other words, given a constrained nonlinear programming problem with decision variable $\boldsymbol{q}$, cost function $J(\boldsymbol{q})$, and a set of inequality constraints $\boldsymbol{l}(\boldsymbol{q})$:

$$\begin{aligned} &\min_{\boldsymbol{q}} \ J(\boldsymbol{q}) \\ &\text{s.t.} \quad \boldsymbol{l}(\boldsymbol{q}) \leq \boldsymbol{0} \end{aligned} \tag{4}$$

an initial set of linearization points $\boldsymbol{q}_0$ is given to render the problem in Eqs. 4–5.

$$\begin{aligned} &\min_{\boldsymbol{q}} \ J(\boldsymbol{q}_0) + \left.\frac{\partial J}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_0} \cdot (\boldsymbol{q} - \boldsymbol{q}_0) \\ &\text{s.t.} \quad \boldsymbol{l}(\boldsymbol{q}_0) + \nabla \boldsymbol{l}(\boldsymbol{q}_0)^{\mathrm{T}} \cdot (\boldsymbol{q} - \boldsymbol{q}_0) \leq \boldsymbol{0} \end{aligned} \tag{5}$$

Solving (5) using LP gives the optimal solution $\boldsymbol{q}_1$ at the 1st iteration. Then, $\boldsymbol{q}_1$ is used to define a new set of linearization points to compute optimal solution $\boldsymbol{q}_2$ at the 2nd iteration, and this process is repeated i times until an acceptable level of accuracy in the optimal solution $\boldsymbol{q}^* = \boldsymbol{q}_i$ is attained.

In a similar manner, using SLP, first FOSEP uses $\boldsymbol{s}_0 = \boldsymbol{s}_c$ at the 1st iteration to linearize the nonlinear term $\hat{\boldsymbol{x}}_d, \hat{\boldsymbol{y}}_d$ to solve for $\boldsymbol{s}_1$; then $\boldsymbol{s}_1$ is used as linearization points in the 2nd iteration, and so on until the optimal solution $\boldsymbol{s}^* = \boldsymbol{s}_i$ is obtained. Accordingly, Eqs. 1 and 3 can be reformulated as the pseudocode in Eq. 6.

Initialize:
$i = 0$
$\boldsymbol{s}_0 = \boldsymbol{s}_c$

do
$i = i + 1;$
$\min_{\boldsymbol{s}} \ \sum_{k=0}^{N-1} -s_i(k)$
s.t. $\forall k \in [1, N-1],$
$s_i(k-1) \leq s_i(k) \leq 1;$
$L\frac{D[\boldsymbol{s}_i]}{T_s} \leq \boldsymbol{F}_{max};$
$\left|\frac{D^2[\hat{\boldsymbol{x}}_{d,i-1}]}{T_s^2}\right|, \left|\frac{D^2[\hat{\boldsymbol{y}}_{d,i-1}]}{T_s^2}\right| \leq \boldsymbol{A}_{max};$
$\left|(\boldsymbol{I} - \hat{\boldsymbol{G}}_x \boldsymbol{C}_x)\hat{\boldsymbol{x}}_{d,i-1}\right|, \left|(\boldsymbol{I} - \hat{\boldsymbol{G}}_y \boldsymbol{C}_y)\hat{\boldsymbol{y}}_{d,i-1}\right| \leq \boldsymbol{E}_{max}$
while $\left|\sum_{k=0}^{N-1} s_i(k) - \sum_{k=0}^{N-1} s_{i-1}(k)\right| \geq tol$
$\boldsymbol{s}^* = \boldsymbol{s}_i$

(6)

Here, tol represents the maximum allowable difference between previous and current values of the cost function; Furthermore, $\hat{\boldsymbol{x}}_{d,i-1}$ and $\hat{\boldsymbol{y}}_{d,i-1}$ in Eq. 6 respectively represent $\hat{\boldsymbol{x}}_d$ and $\hat{\boldsymbol{y}}_d$ of Eq. 2 evaluated using $\boldsymbol{s}_{i-1}$ instead of $\boldsymbol{s}_c$.

3.2 Formulation of FOSEP using Win-SLP

Notice that SLP is more computationally expensive than LP because it involves repeated executions of LP. To address this problem, a windowed SLP (Win-SLP) scheme is implemented as illustrated in Fig. 4a. In Win-SLP, rather than optimizing $s_i(k)$ over $k \in [0, N-1]$ for $i = 1, 2, ...,$ the SLP optimization discussed in Eq. 6 is applied within a window j defined over $k \in [jN_c, jN_c + N_p - 1]$, where $j = 0, 1,$ is the window index, N_p is the length of the preview interval, and $N_c < N_p$ is the length of control interval over $k \in [jN_c, jN_c + N_c - 1]$. Upon completion of the optimization, the window j advances to window $j + 1$ by N_c time steps. This process is repeated until $s_i(jN_c + N_c - 1) = 1$. In mathematical terms, FOSEP using Win-SLP at iteration i and window j can be formulated as Eqs. 7 and 8:

$$\min_{\boldsymbol{s}} \quad \sum_{jN_c}^{jN_c+N_p-1} -s_i(k) \tag{7}$$

$$\begin{aligned} &\text{s.t. } \forall k \in [jN_c + 1, jN_c + N_p - 1], \\ &\quad s_i(k-1) \leq s_i(k) \leq 1; \\ &\quad L\frac{D[\boldsymbol{s}_i]}{T_s} \leq \boldsymbol{F}_{max}; \\ &\quad \left|\frac{D^2[\hat{\boldsymbol{x}}_{d,i-1}]}{T_s^2}\right|, \left|\frac{D^2[\hat{\boldsymbol{y}}_{d,i-1}]}{T_s^2}\right| \leq \boldsymbol{A}_{max} \end{aligned} \tag{8}$$

For $j = 0$, $\boldsymbol{s}_0 = \boldsymbol{s}_c$ is used for determining $\hat{\boldsymbol{x}}_{d0}$ and $\hat{\boldsymbol{y}}_{d0}$. However, starting from $j = 1$, $\boldsymbol{s}_0$ is calculated as

$$s_0(k) = \begin{cases} s^*(k) & k \in [jN_c,\ jN_c + N_p - N_c - 1] \\ \bar{s}(k) & k \in [jN_c + N_p - N_c,\ jN_c + N_p - 1] \end{cases} \tag{9}$$

where $\boldsymbol{s}^*$ as shown in Fig. 4a indicates the optimal solution obtained by applying Win-SLP until window $j-1$; $\bar{\boldsymbol{s}}$ is defined as:

$$\begin{aligned} \bar{s}(k) &= s^*(jN_c + N_p - N_c - 1) \\ &\quad + \Delta_{end}(k - (jN_c + N_p - N_c - 1)) \\ \text{where} \quad & \\ \Delta_{end} &= s^*(jN_c + N_p - N_c - 1) \\ &\quad - s^*(jN_c + N_p - N_c - 2) \end{aligned} \tag{10}$$

The implication of Eq. 9 is that the unused portion of $\boldsymbol{s}^*$ from window $j-1$ is used to partially initialize the optimization in window j. To make up for the missing N_c points used up in window $j-1$, $\bar{\boldsymbol{s}}$ is defined in Eq. 10 such that the speed of the last point of $\boldsymbol{s}^*$ in window $j-1$ is maintained in the last N_c points of $\boldsymbol{s}_0$ in window j, as illustrated in Fig. 4a.

Also, at $k = jN_c$, kinematic and monotonicity constraints must be enforced between the solution from window $j-1$ and the solution being computed by Win-SLP in window j. This is achieved as follows (for $k = jN_c$):

$$\begin{aligned} & s^*(k-1) \le s_i(k) \le s^*(k-1) + \tfrac{F_{max}}{L} T_s \\ & \left| \tfrac{\hat{x}_{d,i-1}(k) - 2\hat{x}^*_d(k-1) + \hat{x}^*_d(k-2)}{T_s^2} \right| \le A_{max} \\ & \left| \tfrac{\hat{y}_{d,i-1}(k) - 2\hat{y}^*_d(k-1) + \hat{y}^*_d(k-2)}{T_s^2} \right| \le A_{max} \\ & \left| \tfrac{\hat{x}_{d,i-1}(k+1) - 2\hat{x}_{d,i-1}(k) + \hat{x}^*_d(k-1)}{T_s^2} \right| \le A_{max} \\ & \left| \tfrac{\hat{y}_{d,i-1}(k+1) - 2\hat{y}_{d,i-1}(k) + \hat{y}^*_d(k-1)}{T_s^2} \right| \le A_{max} \end{aligned} \tag{11}$$

Note that $\hat{x}^*_d(k)$ and $\hat{y}^*_d(k)$ are respectively defined as $\hat{x}_d(k)$ and $\hat{y}_d(k)$ evaluated using $s^*(k)$. The same method in Eq. 11 can be also used to ensure axis jerk continuity. The ten inequalities in Eq. 11 can be concatenated as $\boldsymbol{l}_{cont}$ and $\boldsymbol{m}_{cont}$ in Eq. 12 by using the relationship in Eq. 2 for $\boldsymbol{s}_i$ on $k \in [jN_c,\ jN_c + 1]$:

$$\boldsymbol{l}_{cont,j} \cdot \boldsymbol{s}_i \le \boldsymbol{m}_{cont,j} \tag{12}$$

Furthermore, let $N_{sys,x}$ and $N_{sys,y}$ respectively represent the lengths of the finite impulse response of $\hat{G}_x$ and $\hat{G}_y$. Then, $N_{sys} = \max(N_{sys,x}, N_{sys,y})$ determines the number of time steps needed for perturbations in both x, y axis to decay to negligible levels. Let $\hat{e}_{x,i}(k)$ be defined as $\hat{e}_x(k)$ evaluated using $s_i(k)$. Then, the domain of k that the inequality $\hat{e}_{x,i}(k) \le E_{max}$ is evaluated at should be $k \in [jN_c - N_{sys},\ jN_c + N_p - 1]$. In other words, it also includes the N_{sys} time steps preceding the beginning of the current window j at $k = jN_c$ that contribute to $\hat{e}_{x,i}(k)$. Thus, the constraints on $\hat{\boldsymbol{e}}_{x,i-1}$ are formulated as Eq. 13:

$$\left|\hat{\boldsymbol{e}}_{x,i-1}\right| = \left| \underbrace{(\mathbf{I} - \hat{\boldsymbol{G}}_x \boldsymbol{C}_x)}_{:=\boldsymbol{\Gamma}_x} \begin{bmatrix} \hat{\boldsymbol{x}}^*_d \\ \hat{\boldsymbol{x}}_{d,i-1} \end{bmatrix} \right| \le \boldsymbol{E}_{max} \tag{13}$$

It can be re-written as Eq. 14:

$$\begin{aligned} & \left| \begin{bmatrix} \boldsymbol{\Gamma}_{x,p} & \boldsymbol{\Gamma}_{x,c} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}^*_d \\ \hat{\boldsymbol{x}}_{d,i-1} \end{bmatrix} \right| \le \boldsymbol{E}_{max} \\ & \therefore -\boldsymbol{E}_{max} - \boldsymbol{\Gamma}_{x,p}\hat{\boldsymbol{x}}^*_d \le \boldsymbol{\Gamma}_{x,c}\hat{\boldsymbol{x}}_{d,i-1} \le \boldsymbol{E}_{max} - \boldsymbol{\Gamma}_{x,p}\hat{\boldsymbol{x}}^*_d \end{aligned} \tag{14}$$

where $\boldsymbol{\Gamma}_{x,p}$ and $\boldsymbol{\Gamma}_{x,c}$ represent rows of $\boldsymbol{\Gamma}_x$ that correspond to $\hat{\boldsymbol{x}}^*_d$ (past N_{sys} points from jN_c) and $\hat{\boldsymbol{x}}_{d,i-1}$ (current N_p points from jN_c), respectively; here, $\hat{\boldsymbol{x}}^*_d$ and $\hat{\boldsymbol{x}}_{d,i-1}$ are $\hat{x}^*_d(k)\ \forall k \in [jN_c - N_{sys},\ jN_c - 1]$ and $\hat{x}_{d,i-1}(k)\ \forall k \in [jN_c,\ jN_c + N_p - 1]$, respectively. A similar equation with $\boldsymbol{\Gamma}_{y,p}$, $\boldsymbol{\Gamma}_{y,c}$, $\hat{\boldsymbol{y}}^*_d$ and $\hat{\boldsymbol{y}}_{d,i-1}$ can be written to constrain $\hat{\boldsymbol{e}}_{y,i-1}$.

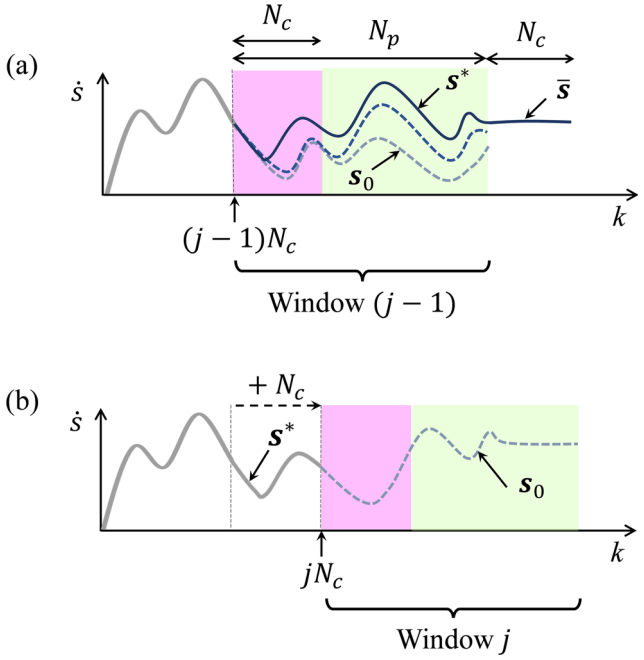

Fig. 4 Illustration of Win-SLP applied on (**a**) the current window $j-1$ with control interval N_c and preview interval N_p, and its augmentation of the final solution $\boldsymbol{s}^*$ by N_c points at the end to serve as a complete initial solution $\boldsymbol{s}_0$ for (**b**) the next window j

Finally, FOSEP using Win-SLP can be represented as the pseudocode in Eq. 15:

Initialize:
  $i = 0$
  $j = 0$
do
  if $j = 0$ **then**
    $\boldsymbol{s}_0 = \boldsymbol{s}_c$
  else
    Eq.(9)
  end if
  do
    $i = i + 1;$
    $\min_{\boldsymbol{s}} \quad \sum_{k=jN_c}^{jN_c+N_p-1} -s_i(k)$
    s.t. Eq.(8),(12),(14)
  while $\left|\sum_{k=jN_c}^{jN_c+N_p-1} s_i(k) - \sum_{k=jN_c}^{jN_c+N_p-1} s_{i-1}(k)\right| \geq tol$
  $s^*(k) = s_i(k)\ \forall k \in [jN_c, (j+1)N_c - 1]$
  $j = j + 1;$
while $s(jN_c + N_c + 1) < 1$ (15)

3.3 Guaranteeing feasibility of Win-SLP

Since (3.2) is defined as a finite horizon optimization with preview length $N_p << N$, the solution determined at window j may lead to infeasibility in future windows, with no recourse to generate a feasible solution [34]. Infeasibility could be catastrophic in online implementation of FOSEP where it is impossible to go back to the past to re-compute a feasible solution. Therefore, to guarantee feasibility using Win-SLP, we develop a scheme whereby at window j, solution $\boldsymbol{s}_i$ must have a complementary backup solution that is feasible $\forall k \in [jN_c, N-1]$.

The proposed approach is illustrated in Fig. 5 and summarized as a flow chart in Fig. 6. In Fig. 5a, an optimal trajectory $s^*(k)\ \forall k \in [(j-1)N_c, (j-1)N_c + N_p - 1]$ (represented as ①), as well as a feasible backup solution $\tilde{s}(k)\ \forall k \in [(j-1)N_c + N_p, (j-1)N_c + N_p + N_{sys} - 1]$ (represented as ②) is created to smoothly patch onto ①. When the window recedes by N_c as in Fig. 5b and k is updated from $k = (j-1)N_c$ to jN_c, the trajectory transitions to the previously-generated backup (represented as ③) if either of the below two attempts encounters infeasibility:

- **Attempt 1**: Optimization of $s_i(k)\ \forall k \in [jN_c, jN_c + N_p - 1]$ (represented as ④) for any SLP step i;
- **Attempt 2**: Generation of a feasible backup solution $\tilde{s}(k)\ \forall k \in [jN_c + N_p, jN_c + N_p + N_{sys} - 1]$ (represented as ⑤)

This iterative process is possible by the definition of backup solution $\boldsymbol{s}_c$ in Eq. 2; at $k = 0$, $\boldsymbol{s}_c$ serves as a kinematically conservative profile that satisfies constraints in Eq. 15.

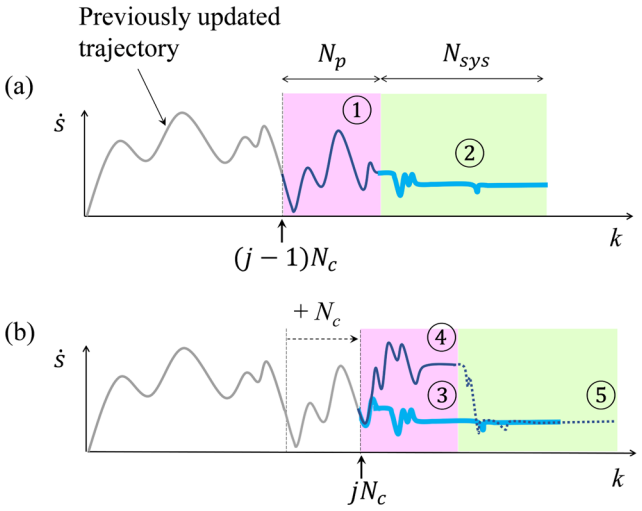

Fig. 5 Overview of scheme to guarantee feasibility of Win-SLP consisting of: **a** generation of backup solution in window $j-1$; and **b** adoption of first N_p points of backup solution (i.e., ③) if no optimal solution ④ or backup solution ⑤ can be found in window j

The reason for generating the backup solution $\tilde{\boldsymbol{s}}$ up to N_{sys} time steps beyond the end of window j is that the transition from ① to ③ can create a transient effect on the actual response of the system dynamics x and y in

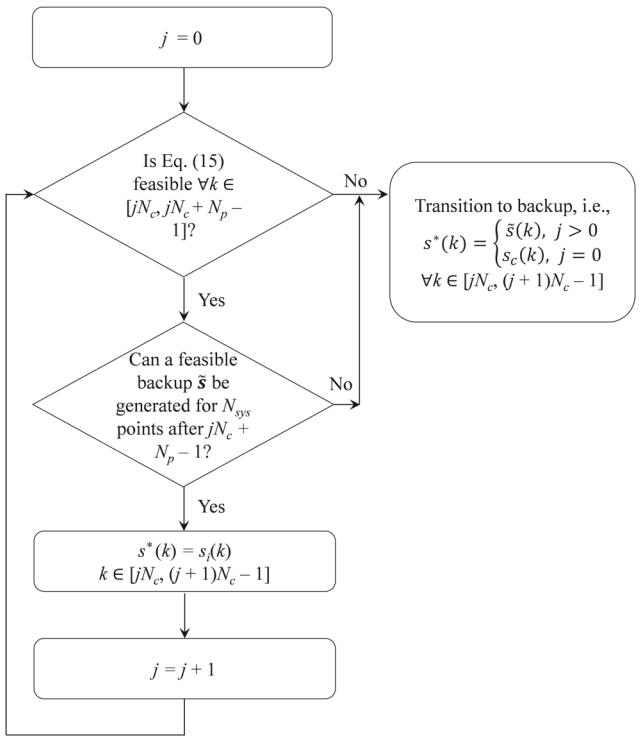

Fig. 6 Flow chart of scheme for guaranteeing feasibility of FOSEP using Win-SLP discussed in Section 3.3

Fig. 3. These transients can cause $\hat{e}_x$ and $\hat{e}_y$ to violate their tolerance limit E_{max}. However, this perturbation will die down in both axes within N_{sys} time steps; thus, if there exists a feasible trajectory $s^*(k)\ \forall k \in [jN_c + N_p, jN_c + N_p + N_{sys} - 1]$, there also will exist a feasible trajectory on the rest of the trajectory, i.e., $\forall k \in [jN_c + N_p + N_{sys}, N-1]$, due to the availability of $\boldsymbol{s}_c$ which is the ultimate backup solution. Therefore, guaranteeing a feasible solution on N_{sys} backup points is enough to guarantee feasibility within the entire trajectory $s^*(k)\ k \in [0, N-1]$.

Furthermore, the smoothness of the transition from ① to ③ can be guaranteed by how $\tilde{\boldsymbol{s}}$, as well as $\boldsymbol{s}_c$, is generated: First, the values of the final displacement, feedrate, and acceleration at $s^*((j-1)N_c + N_p - 1)$ (the last point of ①) are identified and represented as s_{in}, f_{in}, and a_{in}, respectively. Then, a standard trapezoidal acceleration profile (TAP) [8] is created from $s = s^*((j-1)N_c + N_p - 1)$ to $s = 1$. TAP chooses a conservative set of limits on the feedrate and acceleration as $F_{max,c}$ and $A_{max,c}$, as well as initial boundary conditions as s_{in}, f_{in}, and a_{in}, respectively, to create $\tilde{\boldsymbol{s}}$. Also, to ensure smoothness at the axis level, the axis velocity profiles of $\hat{\boldsymbol{x}}_d$ and $\hat{\boldsymbol{y}}_d$ evaluated using $\tilde{\boldsymbol{s}}$ are filtered with a moving average filter $H(z)$ [35] with time constant $\tau = F_{max,c}/A_{max,c}$. By choosing the first N_{sys} points of $\tilde{\boldsymbol{s}}$, $\tilde{s}_k\ \forall k \in [(j-1)N_c + N_p, (j-1)N_c + N_p + N_{sys} - 1]$ is generated such that it connects to ① smoothly. Then, a check is made on whether $\tilde{s}(k)\ \forall k \in [(j-1)N_c + N_p, (j-1)N_c + N_p + N_{sys} - 1]$ respects the limits to determine whether Attempt 2 (described above) is feasible in window $j-1$ as in Eq. 16:

$$-\boldsymbol{E}_{max} - \boldsymbol{\Gamma}_{x,p}\hat{\boldsymbol{x}}_d^* \le \boldsymbol{\Gamma}_{x,c}\tilde{\boldsymbol{x}}_d \le \boldsymbol{E}_{max} - \boldsymbol{\Gamma}_{x,p}\hat{\boldsymbol{x}}_d^* \tag{16}$$

where $\tilde{\boldsymbol{x}}_d$ represents $\hat{\boldsymbol{x}}_d$ defined at $\tilde{s}(k)\ \forall k \in [(j-1)N_c + N_p, (j-1)N_c + N_p + N_{sys} - 1]$ with $H(z)$ applied, and $\hat{\boldsymbol{x}}_d^*$ represents $\hat{x}_d^*(k)\ \forall k \in [(j-1)N_c + N_p - N_{sys}, (j-1)N_c + N_p - 1]$.

4 Validation of proposed approach via simulation

The goal of this section is to validate the accuracy, computational efficiency and guaranteed feasibility of the proposed Win-SLP approach for FOSEP via a series of simulations. For this purpose, a 2nd order linear system dynamics in the x- and y- axis has been selected, as shown in Eq. 17:

$$G_x = G_y = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{17}$$

Its parameters are chosen as $\omega_n = 2\pi \times 50 = 314.16$ rad/s and $\zeta = 0.1$. Then, G_x and G_y are discretized with sampling time $T_s = 1$ ms and approximated as an FIR filter by stacking its truncated impulse response as Eq. 18:

$$\boldsymbol{G}_x \approx \begin{bmatrix} \boldsymbol{x}_{imp} & \boldsymbol{0} & & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{x}_{imp} & & \vdots \\ \vdots & \vdots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{x}_{imp} \end{bmatrix} \tag{18}$$

where $\boldsymbol{x}_{imp}$ is the truncated impulse response column vector of $\boldsymbol{G}_x$ with length N_{sys}, which is identified as 400 for the system in Eq. 17. For SEP, $\boldsymbol{C}_x = \boldsymbol{C}_y$ is generated via the filtered B spline (FBS) approach [15] because of its effectiveness and versatility in handling any type of linear system dynamics [16].

A 5th degree B spline with uniform knot vector is used to generate $\boldsymbol{C}_x$ using the FBS method. The number of trajectory points in the B spline is selected as the length of $\boldsymbol{s}_c$, and the ratio of number of B spline basis functions to the length of the trajectory is 1:20. In Win-SLP, N_p = 50 and N_c = 15 are used so that adjacent windows are overlapped by 35 points; $tol = 10^{-3}$ is used for SLP termination threshold.

All simulations are conducted using circles of radius R. The simulations are conducted using MATLAB® R2019a on a Windows PC with Intel Core i7-8750H CPU and 16 GB RAM. In each simulation, the initialization trajectory, $\boldsymbol{s}_c$, is generated using trapezoidal acceleration profile (TAP) [8] with conservative kinematic limits as $F_{max,c}$ = 30 mm/s, $A_{max,c}$ = 0.5 m/s^2, $J_{max,c}$ = 5 m/s^3. The following algorithms are used implement FOSEP:

- Case A: LP (performed in one batch);
- Case B: SLP (performed in one batch); and
- Case C: Win-SLP (with smooth switching)

In all cases, F_{max} = 50 mm/s, A_{max} = 10 m/s^2, J_{max} = 5000 m/s^3, which are borrowed from the prior work [30]. For the first set of simulations, a short circular toolpath (R = 5 mm) with tight tracking error constraints of E_{max} = 3 μm are used. Note that, with R = 5 mm, $\boldsymbol{s}_c$ yields maximum tracking error of 1.24 μm with a cycle time of 1.58 s, meaning that it is feasible under all limits of F_{max}, A_{max}, J_{max}, and E_{max}. Figure 7 shows the commanded feedrate, acceleration, and jerk profiles of the three cases. Figure 8 shows the simulated tracking error profiles in both x- and y- axes, $\hat{e}_x$ and $\hat{e}_y$, which are simulated using the discretized version of the dynamics in Eq. 17. LP violates the tracking error tolerance in both x- and y-axes due to linearization errors, because the LP solution is linearized with $\boldsymbol{s}_c$ which is significantly different from the optimal solution. Conversely, SLP and Win-SLP satisfy all the kinematic and tracking error constraints, which highlights their accuracy relative to LP, and why LP is considered unacceptable for FOSEP.

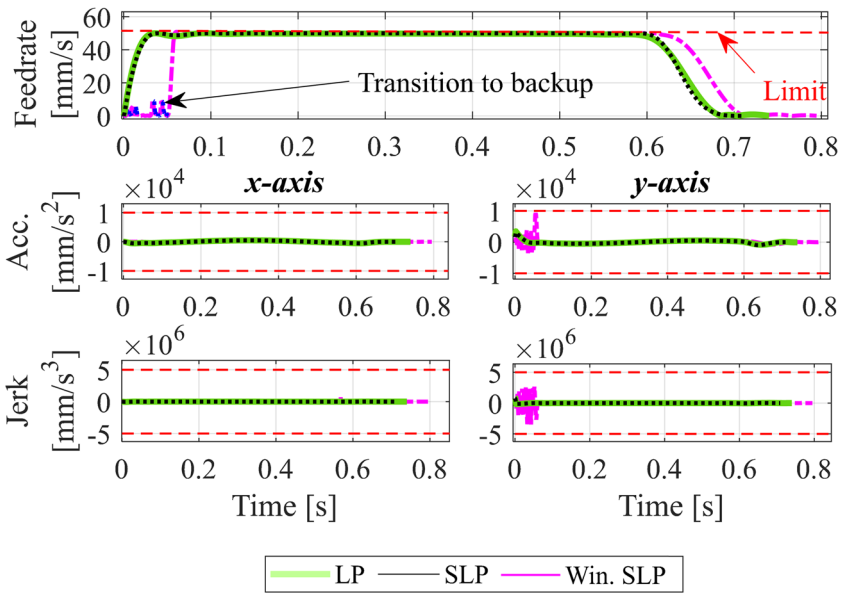

Fig. 7 Feedrate, acceleration and jerk profiles of trajectories generated by FOSEP using LP, SLP and Win-SLP (proposed) for short circular toolpath (R = 5 mm). The switching of the Win-SLP solution to the backup solution at the marked instances enables it to guarantee feasibility at the cost of optimality (as long as the starting solution, s_c, is feasible)

The importance of smooth switching in guaranteeing the feasibility of Win-SLP can also be seen from Figs. 7 and 8. Observe that the Win-SLP solution has to switch several times between the optimal and backup solution in order to maintain feasibility. The implication is that, without the backup solution, Win-SLP would fail to yield a feasible solution. The cycle time of the LP, SLP, and Win-SLP solutions for FOSEP is summarized in Table 1, along with their accuracy in enforcing constraints. Though LP provides the shortest cycle time, it is inaccurate. SLP and Win-SLP are accurate but the cost of switching multiple times to the backup solution is that Win-SLP has slightly (7.9%) longer cycle time than SLP.

The advantage of Win-SLP manifests itself as the length of the toolpath grows, as is typical in practice. The computational efficiency of SLP relative to Win-SLP degrades rapidly with increasing toolpath length. To demonstrate this, a second set of simulations is carried out on circular toolpaths with R ranging from 5 to 100 mm; E_{max} = 3 μm in all cases. As shown in Fig. 9, the computation time for SLP increases exponentially as R grows larger (which in return increases N), whereas Win-SLP achieves nearly linear relationship because of its fixed window size N_p. As R reaches 35 mm, SLP fails due to the computer running out of memory, whereas Win-SLP is able to carry out the optimization up to R = 100 mm (and beyond, not shown). Together, these two sets of simulations demonstrate the accuracy and feasibility guarantees of the proposed Win-SLP approach for FOSEP, together with its computational efficiency for long toolpaths.

5 Experimental validation

5.1 Experimental setup

Long toolpaths are commonplace in manufacturing. Therefore, through experiments carried out in this section, we

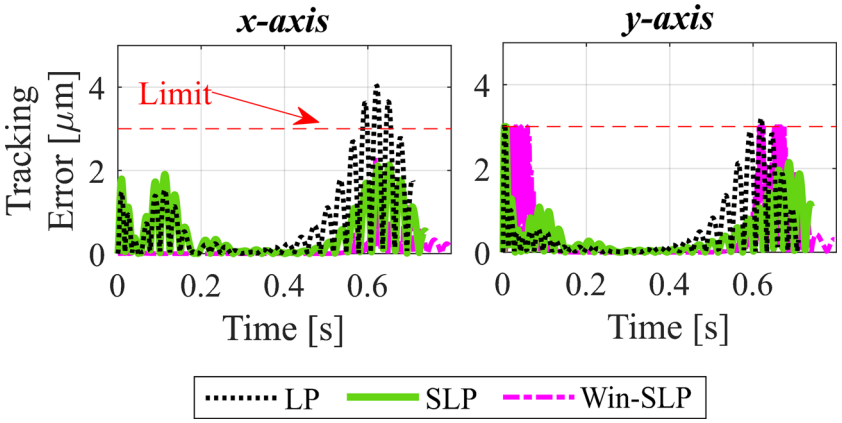

Fig. 8 Simulated axis tracking error profiles of LP, SLP, and Win-SLP (proposed) showing the inaccuracy of LP and the accuracy of SLP and Win-SLP in enforcing constraints

Table 1 Cycle time and constraint satisfaction accuracy of FOSEP using LP, SLP and Win-SLP for short toolpath (R = 5 mm)

	LP	SLP	Win-SLP (proposed)
Cycle time [s]	0.704	0.737	0.795
Satisfies constraints accurately?	No	Yes	Yes

seek to demonstrate the benefit of the proposed Win-SLP approach to FOSEP on improving productivity (compared to FO then SEP) when both are applied to long toolpaths. A Lulzbot Taz 6 3D printer shown in Fig. 10, is used for the experiments. The optimization algorithms are implemented offline on a dSPACE 1007 real-time control board running at 1 kHz sampling rate, connected to DRV8825 stepper motor drivers for x, y, z, and e- (extruder) axes stepper motors.

To execute both FOSEP and FO then SEP, the x- and y-axes servo dynamics of the printer must be measured in the form of frequency response functions (FRFs) and modeled, via curve fitting, as $\hat{G}_x$ and $\hat{G}_y$. Figure 11 shows the measured and modeled FRFs of the x-and y-axes of the printer. The input of each FRF is swept sine acceleration commands to the stepper motor, and the output is relative acceleration between the build plate and nozzle measured using two ADXL335 tri-axial accelerometers. The discrete-time transfer function representation of $\hat{G}_x$ and $\hat{G}_y$ is shown in Eq. 19.

$$\begin{aligned}\hat{G}_x &= \frac{0.026z^5 - 0.078z^4 + 0.055z^3 + 0.042z^2 - 0.069z + 0.023}{z^6 - 5.652z^5 + 13.4z^4 - 17.07z^3 + 12.31z^2 - 4.768z + 0.775}\\ \hat{G}_y &= \frac{0.199z^3 - 0.349z^2 + 0.174z - 2.14\times10^{-27}}{z^4 - 1.934z^3 + 0.958z^2 - 2.733\times10^{-17}z + 1.947\times10^{-34}}\end{aligned} \tag{19}$$

5.2 Benchmarking to determine E_{max}

The desired path is chosen as a butterfly curve [7] shown in Fig. 12, where $\hat{\boldsymbol{x}}_d$ and $\hat{\boldsymbol{y}}_d$ are parameterized in $\boldsymbol{s}$ using quintic spline interpolation with minimal feedrate fluctuation [31]. Similar to Section 4, $\boldsymbol{s}_c$ is selected as a TAP position trajectory with conservative kinematic limits as $F_{max,c}$ = 30 mm/s, $A_{max,c}$ = 0.5 m/s^2, and $J_{max,c}$ = 5 m/s^3; it is smoothened at the axis level using a digital filter $H(z)$ with time constant τ = 0.06 s. These conservative kinematic limits are known from prior work [30, 32] to give acceptable quality. Therefore, we use the conservative TAP trajectory generated using them to determine E_{max} for FOSEP and FO then SEP. Figure 13 shows the feedrate, axis acceleration, and axis jerk profiles of the conservative TAP command. Figure 14 shows the simulated x- and y-axes tracking errors obtained by applying the conservative TAP command to the transfer functions of the 3D printer given in Eq. 19. The conservative TAP yields maximum tracking error of 127.4 μm and 55.2 μm for the x- and y-axes, respectively. Therefore, E_{max} = 127.4 μm is chosen as the tolerance limit that must be satisfied by FOSEP and FO then SEP.

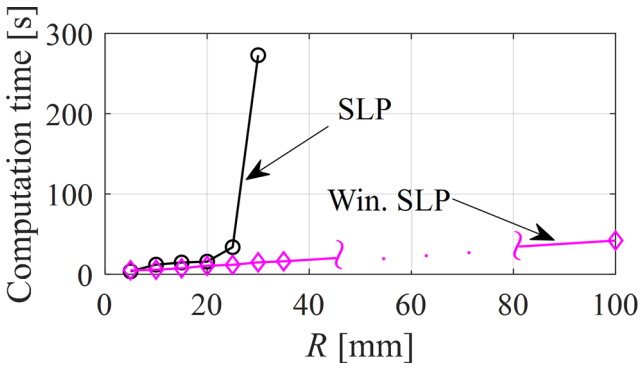

Fig. 9 Computation time vs. radius R of the circular trajectory for FOSEP using SLP and Win-SLP (proposed) showing the superior computational efficiency of Win-SLP relative to SLP for long toolpaths

5.3 Optimization results using FOSEP and FO then SEP using Win-SLP

We compare FO then SEP and the proposed FOSEP using Win-SLP with a goal to achieving E_{max} = 127.4 μm with the shortest cycle time. To do this, aggressive constraints from our prior work [30], namely, F_{max} = 50 mm/s, A_{max} = 10 m/s^2, and J_{max} = 5000 m/s^3 are imposed on both FOSEP and FO then SEP. In FO then SEP, $\boldsymbol{C}_x = \boldsymbol{C}_y = \mathbf{I}$ are selected (i.e., tolerance constraints are imposed without SEP). On the other hand, in FOSEP, $\boldsymbol{C}_x$ and $\boldsymbol{C}_y$ are generated using FBS approach [15, 16], where a 5th degree B-spline with uniform knot vector, and n = 500 control points are used. Both FOSEP and FO then SEP are initialized at $i = 1$ and $j = 0$ by using the conservative TAP in Fig. 13 as $\boldsymbol{s}_c$; N_p = 50, N_c = 20 and $tol = 10^{-3}$ are used for Win-SLP on both FOSEP and FO then SEP, and N_{sys} is identified as 450 for the system dynamics in Eq. 19. Note that in both FOSEP and FO then SEP, servo errors are compensated after the optimization using the C_x and C_y generated using FBS.

Figure 15 shows the commanded feedrate, acceleration and jerk profiles of FOSEP and FO then SEP. Figure 16 shows the resulting tracking error simulated based on the system dynamics in Eq. 19. As a reference, the aggressive kinematic limits are used to generate an aggressive TAP trajectory shown in Fig. 15. The resultant tracking errors after SEP are also shown in Fig. 16. Notice that, because

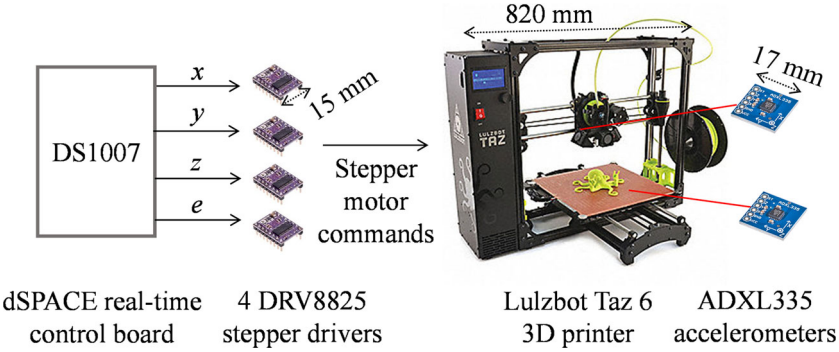

Fig. 10 Experimental setup: Lulzbot Taz 6 3D Printer

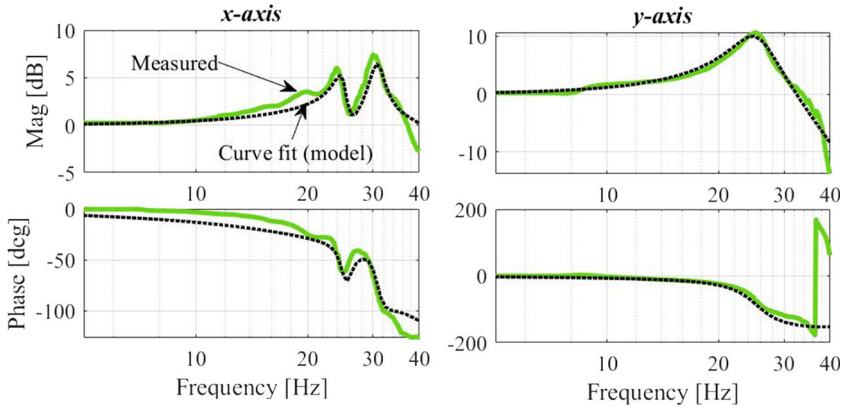

Fig. 11 Measured and curve fitted FRFs of x and y axes of 3D printer

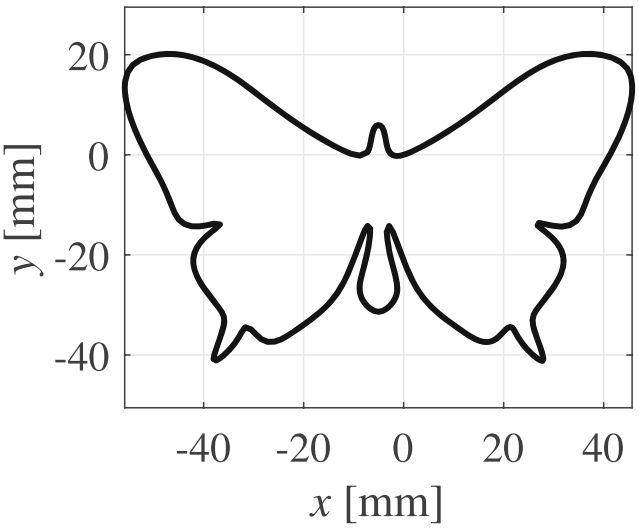

Fig. 12 Desired path of butterfly curve

the aggressive TAP trajectory is not optimized, it results in violations of the tracking error limit. It is this inability to guarantee that tolerance limits will be respected that leads to the use of conservative TAP profiles in practice. Conversely, notice that both FOSEP and FO then SEP enforce the kinematic and tracking error constraints. However, FO then SEP has to slow down by transitioning to the backup solution many times from 0 s to 2 s because the error constraint in Eq. 14 with $\boldsymbol{C}_x = \boldsymbol{C}_y = \mathbf{I}$ has narrower feasible region than FOSEP due to the independent application of FO and SEP. On the other hand, FOSEP is able to stay close to the maximum feedrate throughout the motion with only one transition to the backup at 7.7 s. As a result, FOSEP completes the motion in 8.02 s, which is 21% faster than FO then SEP at 10.15 s, as summarized in Table 2. The computation time for FOSEP and FO then SEP are 37.76 s and 34.91 s, respectively.

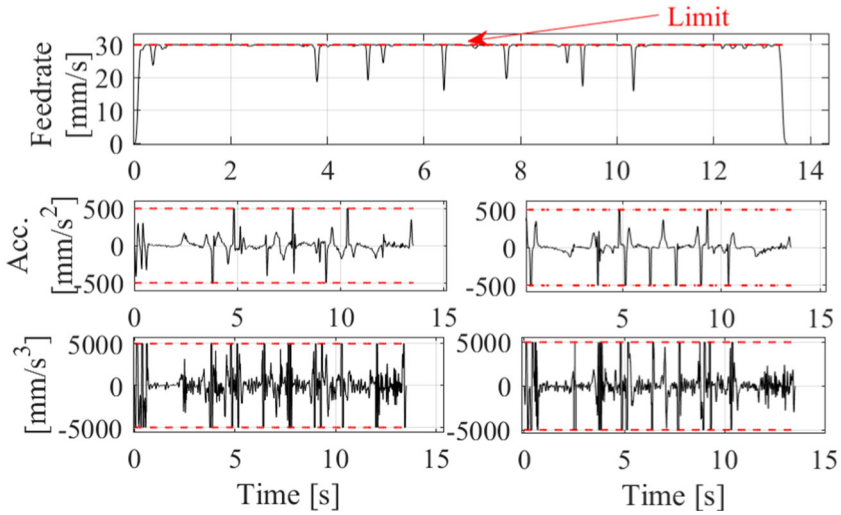

Fig. 13 Commanded feedrate, acceleration and jerk profiles of conservative TAP

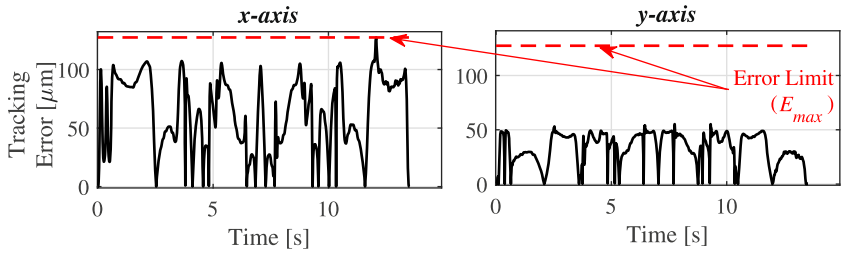

Fig. 14 Simulated tracking error of conservative TAP on x- and y-axes and the approximate error limit E_{max}

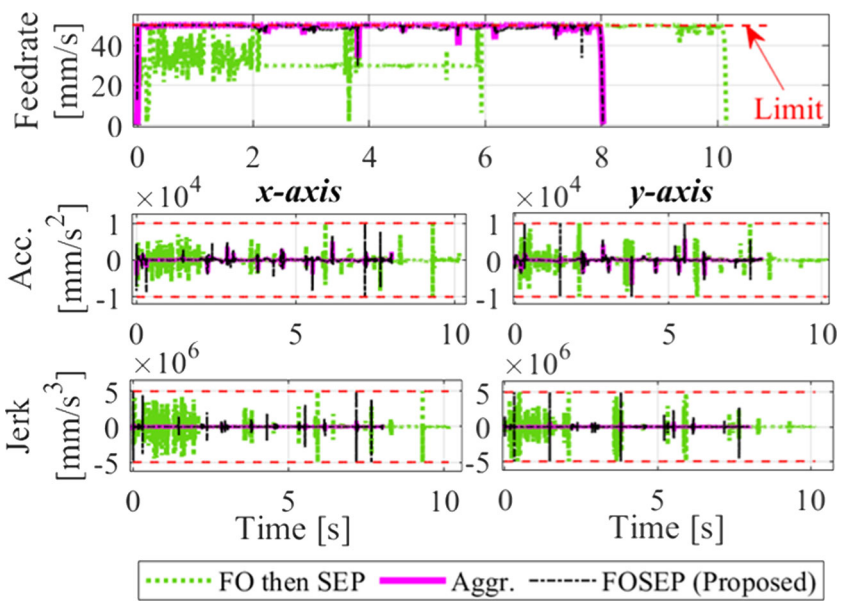

Fig. 15 Commanded feedrate, acceleration and jerk profiles of FO then SEP, FOSEP (proposed), and TAP with aggressive kinematics (Aggr.)

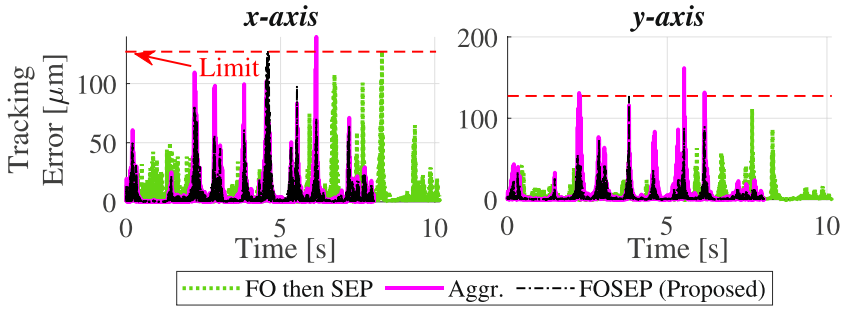

Fig. 16 Simulated tracking error profiles of FO then SEP, FOSEP (proposed), and TAP with aggressive kinematics (Aggr.)

5.4 3D print results

To further validate our findings, a butterfly-shaped 3D part whose CAD model [36] is shown in Fig. 17 is printed using the Taz 6 3D printer of Fig. 10. The butterfly curve shown in Fig. 12 is used to define the outer contour for the part. To parameterize the entire trajectory in s, which consists of curves and lines stacked in z-axis, first the CAD model in Fig. 17 is converted to an STL model using a commercial slicing firmware. Then, each layer in z-axis is further divided into curves and lines, where the curves are parameterized in s using the same method described in Section 5.2 and lines are indexed as $w = 1, 2, ...$ and parameterized individually by identifying the slope $l_m(w)$ and y-intercept $l_y(w)$ using Eq. 20.

$$\begin{aligned} \hat{\boldsymbol{x}}_d &= \frac{1}{\sqrt{l_m(w)^2+1}}\boldsymbol{s} \\ \hat{\boldsymbol{y}}_d &= \frac{l_m(w)}{\sqrt{l_m(w)^2+1}}\boldsymbol{s} + l_y(w) \end{aligned} \tag{20}$$

However, Eq. 20 only maintains axis-level continuity within a given line w; as soon as it shifts to the next line $w+1$, axis acceleration or jerk may violate their limits of A_{max}, J_{max} due to the sharp corner at junction. Therefore, feedrate, or $L\frac{D[s]}{T_s}$, is lowered between two adjacent lines until discontinuity at axis level disappears. Then, all sets of curves and lines are optimized using FOSEP and FO then SEP using Win-SLP, as discussed in Eq. 15, with the same set of limits, F_{max}, A_{max}, J_{max}, E_{max}, and Win-SLP parameters, N_p, N_c, tol, as Section 5.3. As a benchmark, a conservative TAP which uses the same set of slow kinematic limits and moving average filter $H(z)$ as in Section 5.2 is applied for comparison.

Figure 18 shows the printed results of using conservative TAP, as well as FOSEP and FO then SEP both computed using the proposed Win-SLP. The computational time for the three methods are summarized in Table 3. FOSEP saves 25% in cycle time compared to FO then SEP while both maintaining similar quality as the part printed using conservative TAP. The ability to print a very long toolpaths using FOSEP and FO then SEP demonstrates the practicality of the proposed Win-SLP approach. Moreover, the reduction in cycle time highlights the benefits of applying Win-SLP to FOSEP rather than to FO then SEP.

Table 2 Cycle and computation time of FO then SEP and FOSEP using Win-SLP approach

	FO then SEP	FOSEP (proposed)
Cycle time [s]	10.15	8.02
Computation time [s]	34.91	37.76

6 Conclusions and future work

This paper has introduced a method to improve the accuracy in enforcing nonlinear constraints and the

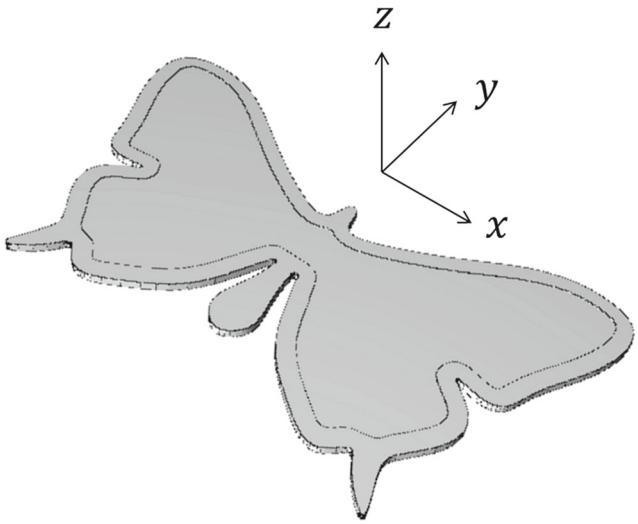

Fig. 17 CAD model of the butterfly plate of height 1.2 mm with outer contour defined by the path in Fig. 12

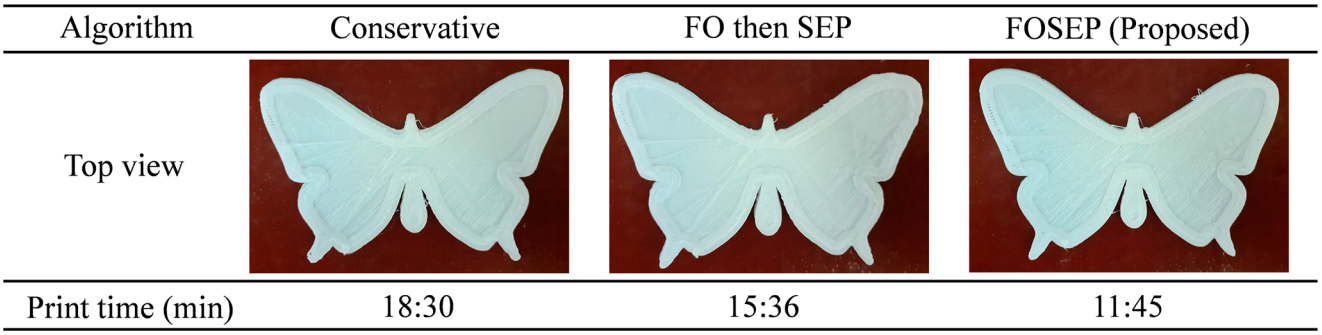

Fig. 18 Top view of printed butterfly plate using conservative TAP, FO then SEP, and FOSEP. Both FO then SEP and FOSEP were computed using the proposed Win-SLP approach

computational efficiency of simultaneous FO and SEP (i.e., FOSEP) applied to long toolpaths. The proposed method, dubbed Win-SLP, achieved windowed sequential linear programming optimization, with feasibility guarantees using smooth transition between the optimal solution and a conservative backup solution.

Win-SLP, which sequentially optimizes the path parameter within a limited horizon length, is formulated and compared with full-preview (i.e., one-shot) LP and SLP. While both SLP and Win-SLP improve accuracy in nonlinear axis-level constraints, Win-SLP shows superiority over SLP in handling longer toolpaths. It is shown in the simulations that Win-SLP achieves both accurate and computationally efficient approach for FOSEP, while guaranteeing feasibility.

Furthermore, compared to the standard practice of sequential FO then SEP, FOSEP relaxes the error tolerance constraints in FO, allowing shorter cycle time without violating tolerance constraints. Experiments carried out on a 3D printer using the proposed Win-SLP approach yielded up to 25% cycle time reduction using FOSEP compared to FO then SEP, subject to the same tolerance and kinematic constraints.

Future work will explore the incorporation of actuator (e.g., torque) limits [22] in FOSEP and develop learning methods to handle nonlinearities and uncertainties in the system dynamics of the machine used for FOSEP. Cloud implementation [32, 37] of Win-SLP applied to FOSEP will also be explored to enable its online implementation.

Table 3 Computation time for FO then SEP and FOSEP for the butterfly plate in Fig. 18

	FO then SEP	FOSEP (proposed)
Computation time [s]	974.98	835.51

Author contributions Not applicable.

Funding This work is funded by the National Science Foundation's award #1825133: Boosting the Speed and Accuracy of Vibration-Prone Manufacturing Machines at Low Cost through Software.

Data availability Not applicable.

Code availability Not applicable.

Declarations

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Altintas Y, Verl A, Brecher C, Uriarte L, Pritschow G (2011) Machine tool feed drives. CIRP Ann 60(2):779–796
2. Tomizuka M (1987) Zero error tracking algorithm for digital control. ASME J Dyn Sys Meas Control 109(1):65–68
3. Zhang D, Chen Y, Chen Y (2016) Iterative Pre-Compensation scheme of tracking error for contouring error reduction. Int J Adv Manuf Technol 87(9-12):3279–3288
4. Ernesto CA, Farouki RT (2010) Solution of inverse dynamics problems for contour error minimization in CNC machines. Int J Adv Manuf Technol 49(5):589–604
5. Altintas Y, Khoshdarregi MR (2012) Contour error control of CNC machine tools with vibration avoidance. CIRP Ann 61(1):335–338
6. Zhang K, Yuen A, Altintas Y (2013) Pre-Compensation Of contour errors in Five-Axis CNC machine tools. Int J Mach Tool Manu 74:1–11
7. Yang S, Ghasemi AH, Lu X, Okwudire CE (2015) Pre-Compensation Of servo contour errors using a model predictive control framework. Int J Mach Tool Manu 98:50–60

8. Gordon DJ, Erkorkmaz K (2013) Accurate control of ball screw drives using Pole-Placement vibration damping and a novel trajectory prefilter. Precis Eng 37(2):308–322
9. Chin JH, Lin TC (1997) Cross-Coupled Precompensation method for the contouring accuracy of computer numerically controlled machine tools. Int J Mach Tool Manu 37(7):947–967
10. Song D, Ma J, Jia Z, Gao Y (2017) Estimation and compensation for Continuous-Path running trajectory error in High-Feed-Speed machining. Int J Adv Manuf Technol 89(5-8):1495–1508
11. Liu W, Sun Y, Yuan X, Chen M (2016) A new approach to the Pre-Compensation of contour errors for Three-Axis machine tools using an adaptive Cross-Coupled controller. Int J Adv Manuf Technol 90(9-12):3711–3725
12. Liu W, Ren F, Sun Y, Jiang S (2018) Contour error Pre-Compensation for Three-Axis machine tools by using Cross-Coupled dynamic friction control. Int J Adv Manuf Technol 98(1):551–563
13. Pi S, Liu Q, Liu Q (2018) A novel dynamic contour error estimation and control in High-Speed CNC. Int J Adv Manuf Technol 96(2):547–560
14. Duong TQ, Rodriguez-Ayerbe P, Lavernhe S, Tournier C, Dumur D (2019) Contour error Pre-Compensation for Five-Axis high speed machining: Offline gain adjustment approach. Int J Adv Manuf Technol 100(1):3113–3125
15. Okwudire C, Ramani K, Duan M (2016) A trajectory optimization method for improved tracking of motion commands using CNC machines that experience unwanted vibration. CIRP Ann 65(1):373–376
16. Duan M, Yoon D, Okwudire CE (2018) A Limited-Preview filtered B-Spline approach to tracking Control–With application to Vibration-Induced error compensation of a 3D printer. Mechatronics 56:287–296
17. Fan W, Gao XS, Lee CH, Zhang K, Zhang Q (2013) Time-Optimal Interpolation for Five-Axis CNC machining along parametric tool path based on linear programming. Int J Adv Manuf Technol 69(5-8):1373–1388
18. Erkorkmaz K, Chen QGC, Zhao MY, Beudaert X, Gao XS (2017) Linear programming and windowing based feedrate optimization for spline toolpaths. CIRP Ann 66(1):393–396
19. Bharathi A, Dong J (2016) Feedrate optimization for smooth Minimum-Time trajectory generation with higher order constraints. Int J Adv Manuf Technol 82(5):1029–1040
20. Zhang Q, Li SR (2013) Efficient computation of smooth minimum time trajectory for CNC machining. Int J Adv Manuf Technol 68:683–692
21. Guo J, Zhang Q, Gao XS, Li H (2015) Time optimal feedrate generation with confined tracking error based on linear programming. J Syst Sci Complex 28(1):90–95
22. Zhang Y, Ye P, Zhao M, Zhang H (2019) Dynamic feedrate optimization for parametric toolpath with Data-Based tracking error prediction. Mech Syst Signal Process 120:221–223
23. Zhang K, Yuan CM, Gao XS (2013) Efficient algorithm for Time-Optimal feedrate planning and smoothing with confined chord error and acceleration. Int J Adv Manuf Technol 66(9-12):1685–1697
24. Sun Y, Zhao Y, Bao Y, Guo D (2015) A smooth curve evolution approach to the feedrate planning on Five-Axis toolpath with geometric and kinematic constraints. Int J Mach Tool Manu 97:86–97
25. Liang F, Zhao J, Ji S (2017) An iterative feed rate scheduling method with confined high order constraints in parametric interpolation. Int J Adv Manuf Technol Manu 92(5):2001–2015
26. Lu L, Zhang L, Ji S, Han Y, Zhao J (2016) An offline predictive feedrate scheduling method for parametric interpolation considering the constraints in trajectory and drive systems. Int J Adv Manuf Technol 83(9-12):2143–2157
27. Lai HY, Lin KY, Tseng SJ, Ueng WD (2008) On the development of a parametric interpolator with confined chord error, feedrate, acceleration and jerk. Int J Adv Manuf Technol 37(1):104–121
28. Yang J, Aslan D, Altintas Y (2018) A feedrate scheduling algorithm to constrain tool tip position and tool orientation errors of Five-Axis CNC machining under cutting load disturbances. CIRP J Manuf Sci Technol 23:78–90
29. Chen M, Sun Y (2019) Contour error–bounded parametric interpolator with minimum feedrate fluctuation for five-axis CNC machine tools. Int J Adv Manuf Technol 103:567–584
30. Kim H, Okwudire CE (2020) Simultaneous servo error pre-compensation and feedrate optimization with tolerance constraints using linear programming. Int J Adv Manuf Technol 109(3):809–821
31. Erkorkmaz K, Altintas Y (2005) Quintic spline interpolation with minimal feed fluctuations. J Manuf Sci Eng 127(2):339–349
32. Okwudire C, Huggi S, Supe S, Huang C, Zeng B (2018) Low-level control of 3D printers from the cloud: a s toward 3D printer control as a service. Inventions 3(3):56
33. Nocedal J, Wright SJ (2006) Numerical optimization. Berlin, New York
34. Camacho EF, Bordons C (2013) Model predictive control. Springer Science & Business Media
35. Suh SH, Kang SK, Chung DH, Stroud IY (2008) Theory and design of CNC systems, Springer-Verlag London Limited
36. Kim H (2020) Butterfly plate. https://www.thingiverse.com/thing:4658078 Accessed 18 Nov 2020
37. Okwudire CE, Lu X, Kumaravelu G, Madhyastha H (2020) A three-tier redundant architecture for safe and reliable cloud-based CNC over public internet networks. Robot Comput-Integr Manuf 1(62):101880