Vibration Compensation of Delta 3D Printer with Position-varying Dynamics using Filtered B-Splines

Nosakhare Edoimioya, Cheng-Hao Chou, Chinedum E. Okwudire

***Abstract*—The delta robot is becoming a popular choice for the mechanical design of fused filament fabrication 3D printers because it can reach higher speeds than traditional serial-axis designs. Like serial 3D printers, delta printers suffer from undesirable vibration at high speeds which degrades the quality of fabricated parts. This undesirable vibration has been suppressed in serial printers using linear model-inversion feedforward control methods like the filtered B-splines (FBS) approach. However, techniques like the FBS approach are computationally challenging to implement on delta 3D printers because of their coupled, position-dependent dynamics. In this paper, we propose a methodology to address the computational bottlenecks by (1) parameterizing the position-dependent portions of the dynamics offline to enable efficient online model generation, (2) computing real-time models at sampled points (instead of every point) along the given trajectory, and (3) employing QR factorization to reduce the number of floating-point arithmetic operations associated with matrix inversion. In simulations, we report a computation time reduction of up to 23x using the proposed method, while maintaining high accuracy, when compared to a controller using the computationally expensive exact LPV model. In experiments, we demonstrate significant quality improvements on parts printed at various positions on the delta 3D printer using our proposed controller compared to a baseline alternative, which uses an LTI model from one position. Through acceleration measurements during printing, we also show that the print quality boost of the proposed controller is due to vibration reductions of more than 20% when compared to the baseline controller.**

I. INTRODUCTION

The delta robot is a high-speed, parallel-axis manipulator [1], which makes it a promising candidate for increasing throughput in additive manufacturing. However, delta robots suffer from vibration errors that are a result of structural flexibilities in their kinematic chain [2]; such vibration errors can adversely impact the quality of 3D printed parts. Unfortunately, delta 3D printers have not benefited from the model-based, feedforward control techniques that were recently used to suppress vibration on serial-axis 3D printers [3]–[6] because of the difficulty modeling and controlling the delta's coupled, nonlinear dynamics. These control techniques have resulted in up to 2x productivity increase without sacrificing accuracy on serial-axis printers [6]. This paper aims to address the challenges that prevent application of these control techniques on the delta 3D printer.

Previous work on modeling and controlling delta robots has largely been focused on rotary-joint delta robots, which are actuated with servo motors [7]–[16]. (Most commercial delta 3D printers are prismatic-joint delta robots and typically use stepper motors). For servo motor delta robots, several methods have been studied–most of which rely on state measurements to estimate servo errors for accurate compensation. A PD or PID controller is usually a key element of these compensation methods. However, since standalone PD/PID controllers do not consider the dynamic coupling of delta robots, their performance is affected by the force disturbance inputs from other kinematic chains. To address this issue, Codourey [7] combined a lumped model of the delta manipulator with a PD regulator in a computed torque (CT) control implementation to improve the tracking error performance in pick-and-place tasks when compared with a standalone PD regulator. Similarly, Angel and Viola [8] proposed a fractional PID controller combined with a CT controller. However, CT controllers need to have complete knowledge of the robot's dynamics, which can be challenging to obtain efficiently [17], and are sensitive to uncertainties and disturbance inputs. For example, in [7], workspace accelerations, which are necessary to calculate torques, are computed as second derivatives of the direct-geometric model of the robot (i.e., functions of joint positions). These calculations can be problematic when there is noise or other inaccuracies in the measurements. Perhaps this explains why no experiments that implement the controller on hardware are presented in [8] (only simulations). These challenges have led to the development of other techniques centered on servo error estimation and disturbance rejection [9]–[14]. These include methods like changing the PD gains online as a function of servo error estimates [9], disturbance rejection in the feedback loop using linear disturbance observers [10], [11], injecting inputs learned by a neural network to compensate errors that the PD controller does not reject [12], and using synchronization control strategies to reject coupling disturbances in each actuator from the other actuators [13], [14]. Furthermore, other approaches focus on tuning trajectory-dependent PID controller gains offline to minimize errors along a desired path that is known a priori [15], [16]. These PID gains provide reasonable tracking performance along the trajectory but require a priori knowledge of the entire trajectory.

The central theme of all the methods discussed above is a dependence on feedback regulation to ease the difficulty of efficiently and accurately modeling the delta robot. However, most delta 3D printers cannot benefit from those methods because they do not have sensors for feedback control. Inspired by the feedforward CT controllers in [7] and [8], we recently proposed an efficient framework to obtain linear parameter-varying (LPV) models of prismatic-joint delta robots commonly used in 3D printers [17]. The framework uses receptance coupling to split the full model of the delta robot into sub-models that can be independently identified using empirical measurements and analytical derivations. We demonstrated

that the model accurately captures dynamic variation across different locations in the robot's task space. Accordingly, such a model allows a number of feedforward linear model-based vibration control techniques to be applied to the delta 3D printer. Among those is a class of methods known as model-inversion [18], which compensate vibration errors by using the inverse of the system's dynamics to pre-filter motion commands. Unlike other feedforward control methods such as smooth command generation [19], [20] and input shaping [21], model-inversion does not introduce time delays [22] and can theoretically lead to perfect compensation [18]. In practice, perfect compensation is difficult to achieve due to unmodeled errors [3] and the prevalence of nonminimum phase zeros, which can cause oscillatory or unbounded control commands. Nevertheless, several approximate model-inversion controllers have been employed in the literature [18], [23], [24]. Of the available methods (as reviewed in [18]), the filtered basis functions (FBF) approach has been shown to be versatile, compared to others, regarding its applicability to any linear system dynamics [3], [4], [22], [25]–[27]. The FBF approach expresses motion commands as a linear combination of basis functions, forward filters the basis functions using the system's dynamics, and calculates the coefficients of the basis functions that minimize motion errors. A version of FBF commonly used for controlling manufacturing machines is the filtered B-splines (FBS) method [3]–[6], [25], [27], where B-splines are selected as the basis functions because they are amenable to the lengthy motion trajectories common in manufacturing. FBS is implemented in real-time by sequentially processing small windows of the entire trajectory [4].

FBS has been implemented on serial-axis 3D printers, which are modeled as linear time-invariant (LTI) systems [3]–[6], as well as a parallel-axis LPV 3D printer–the H-frame 3D printer [27]. In the LTI (i.e., standard) implementation of FBS, B-splines are filtered and inverted offline to enable fast computation of their coefficients online. For the LPV system in [27], the authors model motion errors as a linear relationship between the x- and y-axis (and their LTI models). Furthermore, they approximate the dynamics as decoupled, resulting in independent computation of B-spline coefficients for each axis. Using these approximations, the B-splines can also be filtered and inverted offline. However, the delta 3D printer LPV model cannot be decoupled. Hence, its model needs to be recomputed at each new position, rendering real-time control with FBS computationally challenging. One must compute the new model, use it to filter the B-splines, and invert the filtered B-splines at every position along the trajectory.

This paper aims to address the computational challenges that hinder application of FBS on delta 3D printers by making the following contributions:

1) We parameterize expressions of the delta's transfer functions offline, which leads to fast computation of the model online.
2) We select one position per window of motion trajectory points as the position at which a model used to control all points in the window is generated. This choice leads to faster computation and lower memory allocation. We also propose a method to preserve continuity in the controller's prediction of output trajectories when the model switches between windows.
3) We calculate the B-spline coefficients using QR factorization instead of pseudoinversion, leading to faster computations.

The techniques we develop extend the standard FBS controller to the delta 3D printer. Furthermore, the proposed controller is shown to be effective at improving the quality of printed parts on a commercial delta 3D printer.

The rest of the paper is as follows: Section II provides a recap of the dynamic model developed in [17]; Section III gives an overview of the standard FBS approach and describes the extensions we propose to enable real-time control of the delta 3D printer; Section IV validates our proposed approach through simulations and experiments, illustrating the effectiveness of the proposed controller to improve print quality relative to a standard FBS controller; and Section V concludes the paper, summarizing key insights.

II. OVERVIEW OF LPV MODEL OF DELTA 3D PRINTER

A. Description of the delta 3D printer

As depicted in Fig. 1, three pairs of forearms on the delta 3D printer are connected to a carriage on one end and an end-effector on the other end. The carriages translate vertically to position one end of the forearms, which are allowed to rotate freely about universal joints. Each carriage moves on linear guideways and is mounted to a timing belt, which is, in turn, connected to a base-mounted stepper motor via a motor pulley–forming the prismatic joint. The relative position of each carriage (i.e., the joint space) determine the Cartesian position of the 3-DOF end-effector (i.e., the task space), which holds a nozzle that deposits melted filament onto a stationary bed. The parallelogram formed by each pair of forearms guarantees that the end-effector and fixed base remain co-planar.

B. Linear parameter-varying model

The carriage output positions of the delta 3D printer, q_i, are a function of two inputs: (a) the commanded position of each carriage q_{d_i} and (b) the forces F_{q_i} imposed on each carriage due to the dynamics of the forearms and end-effector, where $i \in \{A,B,C\}$ denotes the carriages labeled A, B, and C (see Fig. 2). Hence, the carriage output dynamics are given by

$$q_i(s) = G_{q_d}(s)q_{d_i}(s) + G_{Fq}(s)F_{q_i}(\mathbf{X},s) \tag{1}$$

where s is the Laplace variable, $G_{q_d}(s)$ and $G_{Fq}(s)$ are LTI SISO systems representing the carriage position to position transfer function (TF) and the external force to carriage position TF, respectively, and $\mathbf{X} = [x \quad y \quad z]^T$ is the end-effector's position in the task space coordinates.

The coefficients of G_{q_d} and G_{Fq} can be identified from measurements on the printer and the parameters of the external forces F_{q_i}, which are modeled analytically, are also identified

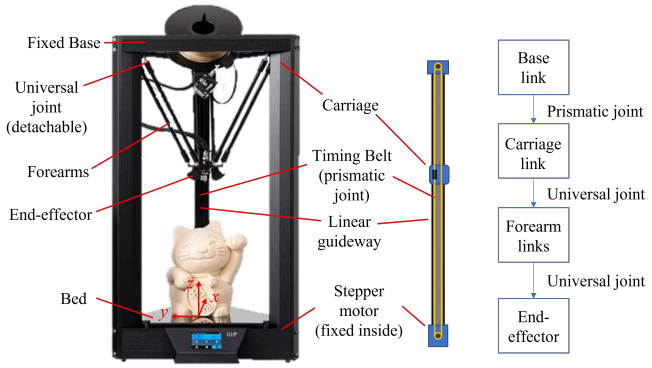

Figure 1. From left to right: A commercial delta 3D Printer (Monoprice Delta Pro) with labeled components, a schematic of the belt-driven carriage system, and the delta manipulator configuration showing the connections between joints and links. The print volume dimensions are 270 × 270 × 300 mm.

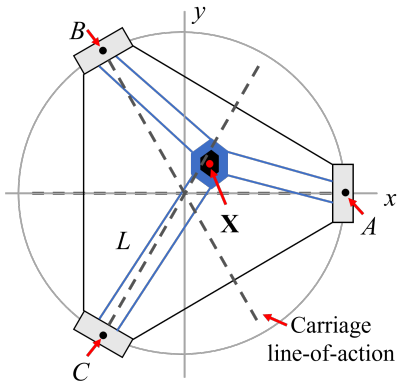

Figure 2. Overhead view of the delta 3D printer showing the (x,y)-coordinate locations of carriages A, B, and C, the end-effector's position in task space $\mathbf{X}$, and the length of the forearms L. End-effector motion along a carriage's line-of-action results in significant change to the carriage dynamics.

from measurements and least squares estimation techniques in [17]. The analytical model of F_{q_i} is obtained by considering the inertial dynamics of the end-effector in the task space, which are transformed into joint dynamics using the Jacobian transpose matrix. A detailed formulation of this process is given in [17]. It transforms Eq. (1) into the following expression:

$$\mathbf{q}(s) = \mathbf{G}_{q_d}(s)\mathbf{q}_d(s) + \mathbf{G}_{Fq}(s)\begin{bmatrix}\bar{\mathbf{J}}_A^T\mathbf{P}_A\\ \bar{\mathbf{J}}_B^T\mathbf{P}_B\\ \bar{\mathbf{J}}_C^T\mathbf{P}_C\end{bmatrix}\mathbf{W}(s)\bar{\mathbf{J}}\mathbf{q}(s) \tag{2}$$

where $\mathbf{G}_{q_d}$ and $\mathbf{G}_{Fq}$ are 3×3 diagonal matrices that contain G_{q_d} and G_{Fq}, respectively, as the diagonal entries, $\mathbf{q} = [q_A \quad q_B \quad q_C]^T$ is the carriage output position vector, $\mathbf{q}_d = [q_{d_A} \quad q_{d_B} \quad q_{d_C}]^T$ is the desired position vector, $\mathbf{P}_i \in \mathbb{R}^{3\times 3}$ is the matrix representing the distribution of task space inertial forces associated with carriage i, $\bar{\mathbf{J}}_i \in \mathbb{R}^{3\times 1}$, the column vector for carriage i extracted from the linearized Jacobian matrix, denoted by $\bar{\mathbf{J}}$ (see [17] for details),

$$\mathbf{W}(s) = \begin{bmatrix} w_x(s) & 0 & 0\\ 0 & w_y(s) & 0\\ 0 & 0 & w_z(s)\end{bmatrix}, \tag{3}$$

and w_x, w_y, and w_z are the flexible inertial dynamics of the end-effector in the x-, y-, and z-axis directions, respectively. The model can be expressed simply as

$$\mathbf{q}(s) = \mathbf{G}(s)\mathbf{q}_d(s) \tag{4}$$

where

$$\mathbf{G}(s) = \left[\mathbf{I} - \mathbf{G}_{Fq}(s)\begin{bmatrix}\bar{\mathbf{J}}_A^T\mathbf{P}_A\\ \bar{\mathbf{J}}_B^T\mathbf{P}_B\\ \bar{\mathbf{J}}_C^T\mathbf{P}_C\end{bmatrix}\mathbf{W}(s)\bar{\mathbf{J}}\right]^{-1}\mathbf{G}_{q_d}(s), \tag{5}$$

yielding a linear parameter-varying (LPV) model of the delta 3D printer that can be used for linear model-inversion feedforward control. Since there are no position sensors, we assume the parameters of $\mathbf{G}(s)$ can be computed using the desired configuration instead of the output configuration, e.g., $\mathbf{X}_d$ instead of $\mathbf{X}$ (see Fig. 3), which was a reasonable assumption in previous work [27].

III. FEEDFORWARD CONTROL WITH FILTERED B-SPLINES

In this section, we give an overview of the standard FBS approach in Section III-A. Then, we discuss the process of extending FBS by: describing the selection of a parameterized model to filter the B-splines in Section III-B, explaining how continuity is preserved when switching models between windows in Section III-C, and describing how the motion command is generated using LU and QR factorization in Section III-D. As a visual aid, the reader can follow a flowchart of the process of generating optimal motion commands in Fig. 3.

A. Overview of the standard FBS approach

The FBS approach (as presented in [4]) controls the lifted system representation (LSR) of $\mathbf{G}(s)$ with a feedforward controller. (See Appendix A for details on the LSR). Let $\mathbf{q}_{i_d} = [q_{i_d}(t_0) \;\; q_{i_d}(t_1) \;\; \cdots \;\; q_{i_d}(t_E)]^T$ represent the entire $E+1$ discrete time steps of the desired trajectory of carriage i, which are processed in sliding windows. Assume that time step k marks the beginning of the current window and that the unknown modified motion command $\mathbf{q}_{i_{dm},\mathrm{C}} = [q_{i_{dm}}(t_k) \;\; q_{i_{dm}}(t_{k+1}) \;\; \cdots \;\; q_{i_{dm}}(t_{k+L_C})]^T$ is parameterized using B-splines such that

$$\begin{bmatrix} q_{i_{dm}}(t_k)\\ q_{i_{dm}}(t_{k+1})\\ \vdots\\ q_{i_{dm}}(t_{k+L_C})\end{bmatrix} = \underbrace{\begin{bmatrix} \phi_{m,m}(t_k) & \cdots & \phi_{m+n,m}(t_k)\\ \phi_{m,m}(t_{k+1}) & \cdots & \phi_{m+n,m}(t_{k+1})\\ \vdots & \ddots & \vdots\\ \phi_{m,m}(t_{k+L_C}) & \cdots & \phi_{m+n,m}(t_{k+L_C})\end{bmatrix}}_{\Phi}\underbrace{\begin{bmatrix} p_{i,m}\\ \vdots\\ p_{i,m+n}\end{bmatrix}}_{\mathbf{p}_{i,\mathrm{C}}} \tag{6}$$

where the (non-italicized) subscript C denotes the current window, L_C is the number of trajectory points considered for each window, Φ is the open-ended B-spline basis functions matrix of degree m [4], $\phi_{j,m}(t)$ are real-valued basis functions [28], $j = m, m+1, \ldots, m+n$, $\mathbf{p}_{i,\mathrm{C}}$ is a vector of $n+1$ unknown coefficients (or control points), $t_k = kT_s$ is the current time, and T_s is the sampling time (see [4] for more

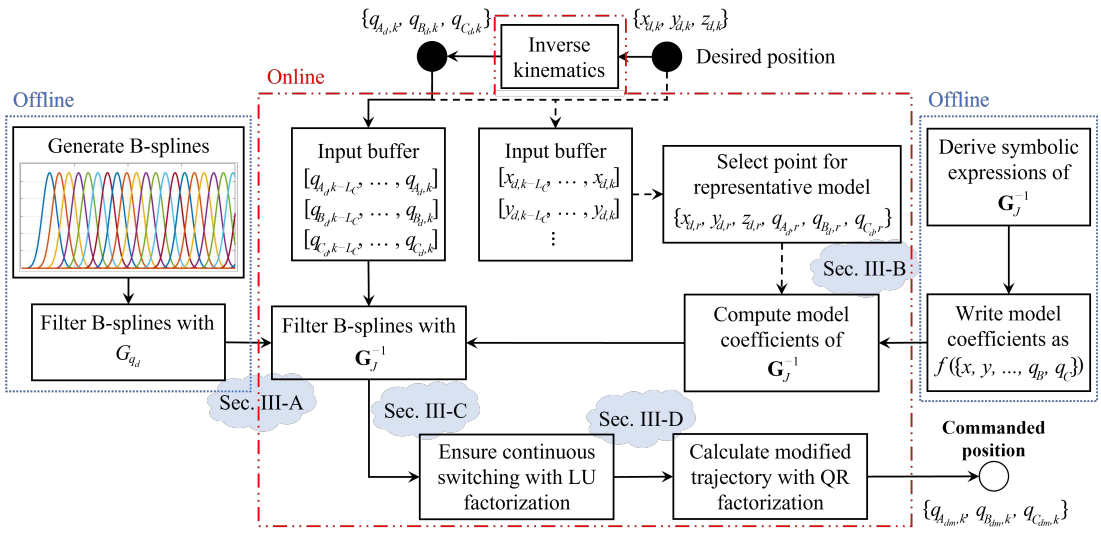

Figure 3. Flowchart of FBS implementation on delta 3D printer. First, the B-splines are generated offline and filtered with the carriage-only position-to-position dynamics as described Sec. III-A (left side). Online, L_C points from the desired Cartesian and joint coordinates are buffered for a new window and a representative configuration is selected from the buffer–the median configuration in this paper. Then, the representative configuration $\{x_{d,r}, ...q_{C_d,r}\}$ is used to compute the transfer function model coefficients of $\mathbf{G}_J^{-1}$ (see Sec. III-B). This transfer function is then used to filter the offline B-splines. Finally, we compute the approximate B-spline coefficients from the previous window to maintain continuity during switching (Sec. III-C) and calculate the modified trajectory (Sec. III-D).

details). To capture the coupling between carriages, we define $\mathbf{q}_{d,\mathrm{C}} = [\mathbf{q}_{A_d,\mathrm{C}}^T \ \mathbf{q}_{B_d,\mathrm{C}}^T \ \mathbf{q}_{C_d,\mathrm{C}}^T]^T$, such that

$$\mathbf{q}_{dm,\mathrm{C}} = \begin{bmatrix} \mathbf{q}_{A_{dm},\mathrm{C}} \\ \mathbf{q}_{B_{dm},\mathrm{C}} \\ \mathbf{q}_{C_{dm},\mathrm{C}} \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Phi & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Phi \end{bmatrix}}_{\mathbf{N}_\mathrm{C}} \underbrace{\begin{bmatrix} \mathbf{p}_{A,\mathrm{C}} \\ \mathbf{p}_{B,\mathrm{C}} \\ \mathbf{p}_{C,\mathrm{C}} \end{bmatrix}}_{\mathbf{p}_\mathrm{C}}. \tag{7}$$

Our objective is to minimize the tracking error, defined as

$$\bar{\mathbf{e}} = \mathbf{q}_d - \bar{\mathbf{N}}\bar{\mathbf{p}} \Leftrightarrow \begin{bmatrix} \bar{\mathbf{e}}_\mathrm{P} \\ \bar{\mathbf{e}}_\mathrm{C} \\ \bar{\mathbf{e}}_\mathrm{F} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{d,\mathrm{P}} \\ \mathbf{q}_{d,\mathrm{C}} \\ \mathbf{q}_{d,\mathrm{F}} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{N}}_\mathrm{P} & \mathbf{0} & \mathbf{0} \\ \bar{\mathbf{N}}_\mathrm{PC} & \bar{\mathbf{N}}_\mathrm{C} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{N}}_\mathrm{CF} & \bar{\mathbf{N}}_\mathrm{F} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}_\mathrm{P} \\ \bar{\mathbf{p}}_\mathrm{C} \\ \bar{\mathbf{p}}_\mathrm{F} \end{bmatrix} \tag{8}$$

where subscripts P and F denote the past and future windows, respectively,

$$\mathbf{q}_\mathrm{C} = \bar{\mathbf{N}}_\mathrm{C}\bar{\mathbf{p}}_\mathrm{C} + \bar{\mathbf{N}}_\mathrm{PC}\bar{\mathbf{p}}_\mathrm{P} \tag{9}$$

represents the current output carriage motion, and the bar on the matrices and vectors indicates that the impulse response of the transfer function used for filtering the B-splines is truncated [4]. Using local least squares, the optimal coefficients of the current window can be computed as

$$\bar{\mathbf{p}}_\mathrm{C} = (\bar{\mathbf{N}}_\mathrm{C}^T\bar{\mathbf{N}}_\mathrm{C})^{-1}\bar{\mathbf{N}}_\mathrm{C}\left(\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\bar{\mathbf{p}}_\mathrm{P}\right) \tag{10}$$

$$= \bar{\mathbf{N}}_\mathrm{C}^\dagger\left(\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\bar{\mathbf{p}}_\mathrm{P}\right) \tag{11}$$

where $\bar{\mathbf{p}}_\mathrm{P}$ denotes the coefficients calculated in the previous window. Note that although n coefficents are computed, only n_up are updated in each window [4].

For an LTI system, $\mathbf{N}_\mathrm{C}$ is pre-filtered and $\bar{\mathbf{N}}_\mathrm{PC}$ and $\bar{\mathbf{N}}_\mathrm{C}^\dagger$ are computed offline and stored for obtaining the optimal coefficients in every window using Eq. (11). However, for LPV systems, filtering and inverting the (large) B-splines matrix in real-time is computationally challenging for most hardware processors. Constrained by the system's computation and memory capabilities, the rest of this section proposes techniques to optimize the computation and memory resources required to apply FBS to the delta 3D printer without significantly sacrificing the achieved accuracy improvement.

B. Selecting a parameterized model for B-splines filtering

Consider the problem of filtering each column of $\mathbf{N}$ through $\mathbf{G}(s)$, reproduced below:

$$\mathbf{G}(s) = \underbrace{\left[\mathbf{I} - \mathbf{G}_{Fq}(s)\begin{bmatrix} \bar{\mathbf{J}}_A^T\mathbf{P}_A \\ \bar{\mathbf{J}}_B^T\mathbf{P}_B \\ \bar{\mathbf{J}}_C^T\mathbf{P}_C \end{bmatrix}\mathbf{W}(s)\bar{\mathbf{J}}\right]^{-1}}_{\mathbf{G}_J^{-1}(s)}\mathbf{G}_{q_d}(s). \tag{12}$$

Note that $\mathbf{G}_J^{-1} \in \mathbb{R}^{3\times 3}$ depends on the configuration through the Jacobian matrix $\bar{\mathbf{J}}$, while $\mathbf{G}_{q_d}$ is not position dependent. Hence, we can derive symbolic expressions of each transfer function in $\mathbf{G}_J^{-1}$ as functions of position. This derivation leads to symbolic transfer functions of the form

$$G_{J,AA}^{-1} = \frac{b_{AA}(x,y,z,q_A,q_B,q_C)}{a(x,y,z,q_A,q_B,q_C)} \tag{13}$$

where $b_{AA}(\cdot)$ and $a(\cdot)$ are the numerator and denominator of the transfer function, respectively, and the subscript "AA" denotes values pertaining to the A-to-A carriage position dynamics. The other 8 transfer functions ($G_{J,BA}^{-1}$, $G_{J,CA}^{-1}$, $G_{J,AB}^{-1}$, and so on) can be expressed similarly with $b_{BA}(\cdot)$, $b_{CA}(\cdot)$, $b_{AB}(\cdot)$, and so on. Note that all transfer functions share the

same denominator $a(\cdot)$. These parameterized transfer functions enable fast computations of the coefficients of $\mathbf{G}_J^{-1}$ during real-time control by simply substituting the corresponding values of x, y, z, q_A, q_B, and q_C into the symbolic expressions. Furthermore, we can pre-filter $\mathbf{N}$ with $\mathbf{G}_{q_d}$ offline to obtain $\bar{\mathbf{N}}_{q_d}$. Then, for each window of trajectory points processed, we filter $\bar{\mathbf{N}}_{q_d}$ with the transfer functions in Eq. (13) to obtain

$$\begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}} \\ \bar{\mathbf{N}}_{\mathrm{CF}} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{AA}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{AA}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{BA}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{BA}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{CA}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{CA}} \end{bmatrix} \\ \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{AB}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{AB}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{BB}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{BB}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{CB}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{CB}} \end{bmatrix} \\ \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{AC}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{AC}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{BC}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{BC}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{CC}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{CC}} \end{bmatrix} \end{bmatrix} \tag{14}$$

where $[\bar{\mathbf{N}}_{\mathrm{C}_{AA}}^T \quad \bar{\mathbf{N}}_{\mathrm{CF}_{AA}}^T]^T$ is the result of filtering the columns of $\bar{\mathbf{N}}_{q_d}$ through $G_{J,AA}^{-1}$, and so on for the other blocks of the matrix.

In practice, the current and future windows are overlapped for continuity during computation but only L_C points are updated during each sequence [4]. Therefore, each overlapped window has $2L_C$ trajectory points, meaning that the time complexity for computing the transfer function coefficients is $O(2L_C)$ (assuming parallel computation) and the space complexity is $O(2L_c u_a)$, where u_a is the order of the transfer functions. Some computers may not have enough processing power to complete these calculations while maintaining real-time printing–especially the smaller micro-processors commonly used by 3D printer manufacturers. Additionally, allocating the memory resources required to store the coefficients may limit the computer's ability to allocate quick-access memory to other important functions like storing the print trajectory.

To prevent such deleterious effects, we select one point from each window (of the first L_C points) where a transfer function is computed, which reduces the time and space complexity to $O(1)$ and $O(u_a)$, respectively. This trade-off is reasonable because L_C generally represents a small distance where the dynamics do not change significantly. For example, L_C typically ranges from 100-200 points, which represents 100 to 200 ms for a standard sampling interval of 1 ms. For most practical applications, the 3D printer will not cover large enough distances in $\leq$200 ms to create significant dynamic variation. For our implementation in Section IV, we select the median point (i.e., the point in the middle of the window) as the representative point. One can select other points such as the mean point (i.e., the average point in the window for each configuration variable, considered independently) or the point with the minimum total Euclidean distance from all the other points in the same window. In simulations, we found that the tracking accuracy is not significantly different when any reasonable central point is selected. In Section IV, we demonstrate that selecting one point in each window does not significantly degrade the accuracy of the controller through simulations and experiments.

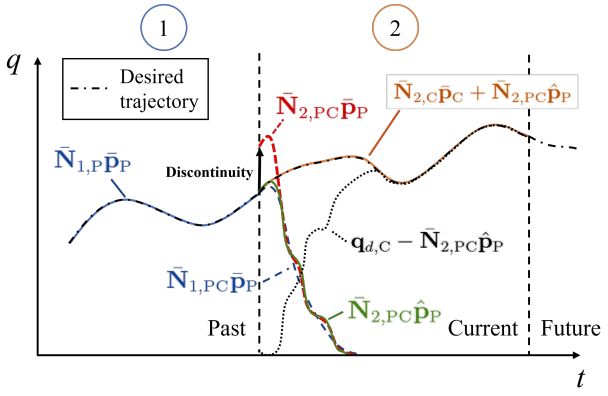

Figure 4. Illustration of the switching compensation technique to maintain continuity described in Sec. III-C. The B-spline coefficients (or control points) from the previous window $\bar{\mathbf{p}}_{\mathrm{P}}$ are approximated as $\hat{\mathbf{p}}_{\mathrm{P}}$ to maintain continuity when the model is switched from model 1 to model 2. Note that $\bar{\mathbf{N}}_{1,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}$ does not have the correct dynamics for the current window and $\bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}$ creates a discontinuity at the window boundary. The difference between the desired trajectory and the approximate residual motion is also shown as $\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_{\mathrm{PC}}\hat{\mathbf{p}}_{\mathrm{P}}$.

C. Smoothly switching models between windows

One drawback to selecting a different model for each window is that switching models can lead to discontinuities in the controller's predicted output trajectories. In the standard FBS approach, continuity is preserved by using the same LTI model to predict the trajectory for every window [4]. To demonstrate what happens in the LPV case, suppose we used model 1 for the past window and update it to model 2 for the current window as shown in Fig. 4. When we switch from model 1 to model 2, note that the prediction of the output trajectory in the current window will be different depending on if we use model 1 (i.e., $\bar{\mathbf{N}}_{1,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}$) or model 2 (i.e., $\bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}$) for the prediction. Since model 2 captures the dynamics in the current window more accurately than model 1, $\bar{\mathbf{N}}_{2,\mathrm{PC}}$ should be used for the prediction. However, using $\bar{\mathbf{N}}_{2,\mathrm{PC}}$ may result in a discontinuity in the prediction of the machine's motion at the point where the window changes because the previous window's control points, $\bar{\mathbf{p}}_{\mathrm{P}}$, were computed using model 1.

To resolve this discrepancy, we approximate the prediction by generating a set of approximate control points that ensure continuity with the output from the past window. The approximate control points are selected to minimize the difference between the new prediction and the (potentially) discontinuous prediction while preserving continuity. We write the optimization problem as

$$\begin{aligned} \hat{\mathbf{p}}_{\mathrm{P}} = \arg\min_{\hat{\mathbf{p}}_{\mathrm{P}}} \quad & \|\bar{\mathbf{N}}_{2,\mathrm{PC}}\hat{\mathbf{p}}_{\mathrm{P}} - \bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}\|_2^2 \\ s.t. \quad & \bar{N}_{2,\mathrm{PC}}^T(t_k)\hat{\mathbf{p}}_{\mathrm{P}} = \bar{N}_{1,\mathrm{PC}}^T(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \\ & \bar{N'}_{2,\mathrm{PC}}^T(t_k)\hat{\mathbf{p}}_{\mathrm{P}} = \bar{N'}_{1,\mathrm{PC}}^T(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \end{aligned} \tag{15}$$

where $\bar{N}_{1,\mathrm{PC}}^T(t_k)$ and $\bar{N}_{2,\mathrm{PC}}^T(t_k)$ are the first rows of $\bar{\mathbf{N}}_{1,\mathrm{PC}}$ and $\bar{\mathbf{N}}_{2,\mathrm{PC}}$ in the window, respectively, $\bar{N'}_{1,\mathrm{PC}}^T(t_k)$ and $\bar{N'}_{2,\mathrm{PC}}^T(t_k)$ are the first rows of $\bar{\mathbf{N}'}_{1,\mathrm{PC}}$ and $\bar{\mathbf{N}'}_{2,\mathrm{PC}}$, respectively (which

are the time derivatives of $\bar{\mathbf{N}}_{1,\mathrm{PC}}$ and $\bar{\mathbf{N}}_{2,\mathrm{PC}}$), and $\hat{\mathbf{p}}_\mathrm{P}$ are the approximate control points. Note that the products

$$\bar{N}_{1,\mathrm{PC}}^T(t_k)\bar{\mathbf{p}}_\mathrm{P} \quad \text{and} \quad \bar{N}_{2,\mathrm{PC}}^T(t_k)\hat{\mathbf{p}}_\mathrm{P} \tag{16}$$

represent positions at the window boundary, and

$$\bar{N'}_{1,\mathrm{PC}}^T(t_k)\bar{\mathbf{p}}_\mathrm{P} \quad \text{and} \quad \bar{N'}_{2,\mathrm{PC}}^T(t_k)\hat{\mathbf{p}}_\mathrm{P} \tag{17}$$

and represent velocities at the boundary. Additional kinematic constraints, such as acceleration and jerk, can be included in the optimization problem from Eq. (15) by taking additional derivatives of the B-splines as described in [28] and [29]. More kinematic constraints leads to smoother transitions when the dynamics change significantly or when the window size is large. In our simulations of the machine used in Section IV, we found that position and velocity constraints led to similar tracking accuracy when compared to optimizing Eq. (15) with acceleration and jerk constraints. Hence, our implementation only uses the position and velocity constraints for Eq. (15).

Using the approximate control points, the coefficients that minimize the tracking error in the current window are obtained by solving

$$\bar{\mathbf{p}}_\mathrm{C} = \arg\min_{\bar{\mathbf{p}}_\mathrm{C}} \Big[\Big((\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\hat{\mathbf{p}}_\mathrm{P}) - \bar{\mathbf{N}}_\mathrm{C}\bar{\mathbf{p}}_\mathrm{C} \Big)^T \Big((\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\hat{\mathbf{p}}_\mathrm{P}) - \bar{\mathbf{N}}_\mathrm{C}\bar{\mathbf{p}}_\mathrm{C} \Big) \Big]. \tag{18}$$

Equation (18) can be computationally expensive to solve in real-time for each window using the pseudoinverse, as done in Eq. (11). Similarly, solving the constrained optimization problem in Eq. (15) in real-time could be challenging. To speed up the computations, we employ the LU and QR factorization methods for solving Eqs. (15) and (18), respectively, as discussed in the following subsection.

D. Command generation with LU and QR factorization

The optimization problem in Eq. (15) can be solved with a number of gradient-based algorithms. For example, Matlab provides functions *fmincon* and *lsqlin* to solve constrained optimization problems. However, such algorithms may require a large number of iterations to converge to a solution, which can stall our controller. To circumvent this problem, we can solve the constrained least squares problem with LU factorization by leveraging properties of the filtered B-splines. To simplify notation, we define the following from Eq. (15):

$$\mathbf{A} = \bar{\mathbf{N}}_{2,\mathrm{PC}}, \quad \mathbf{b} = \bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_\mathrm{P} \tag{19}$$

$$\mathbf{C} = \begin{bmatrix} \bar{N}_{2,\mathrm{PC}}^T(t_k) \\ \bar{N'}_{2,\mathrm{PC}}^T(t_k) \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} \bar{N}_{1,\mathrm{PC}}^T(t_k)\bar{\mathbf{p}}_\mathrm{P} \\ \bar{N'}_{1,\mathrm{PC}}^T(t_k)\bar{\mathbf{p}}_\mathrm{P} \end{bmatrix} \tag{20}$$

Then, the problem can be written as

$$\begin{aligned} \hat{\mathbf{p}}_\mathrm{P} = \arg\min_{\hat{\mathbf{p}}_\mathrm{P}} \quad & \|\mathbf{A}\hat{\mathbf{p}}_\mathrm{P} - \mathbf{b}\|_2^2 \\ s.t. \quad & \mathbf{C}\hat{\mathbf{p}}_\mathrm{P} = \mathbf{d} \end{aligned} \tag{21}$$

We make two assumptions:

1) The stacked matrix
$$\begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix} \tag{22}$$
has linearly independent columns; and
2) $\mathbf{C}$ has linearly independent rows.

As discussed in [22], the filtered B-splines satisfy the above assumptions with high probability and, in the case they do not, the B-splines can be freely selected by the user to satisfy the assumptions. Then, we can construct the Lagrangian,

$$\mathscr{L}(\hat{\mathbf{p}}_\mathrm{P}, \lambda) \triangleq \frac{1}{2}\|\mathbf{A}\hat{\mathbf{p}}_\mathrm{P} - \mathbf{b}\|_2^2 + \lambda^T(\mathbf{C}\hat{\mathbf{p}}_\mathrm{P} - \mathbf{d}), \tag{23}$$

where λ is a set of Lagrange multipliers, and find where its partial derivatives equal zero to obtain the following linear system

$$\begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{p}}_\mathrm{P} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{A}^T\mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{24}$$

Note that the matrix

$$\begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \tag{25}$$

is nonsingular when the above assumptions hold. Therefore, the linear equation given by Eq. (24) can be efficiently solved with LU factorization [30].

We also use QR factorization to efficiently compute the control points. Using the pseudoinverse to solve the optimization problem in Eq. (18) requires the following number of floating-point operations (flops) [30]:

$$\bar{\mathbf{N}}_\mathrm{C}^T\bar{\mathbf{N}}_\mathrm{C} : L_C n^2 \ \text{ flops} \tag{26}$$

$$(\bar{\mathbf{N}}_\mathrm{C}^T\bar{\mathbf{N}}_\mathrm{C})^{-1} : n^3 + L_C n^2 \ \text{ flops} \tag{27}$$

$$\bar{\mathbf{N}}_\mathrm{C}\tilde{\mathbf{q}}_{d,\mathrm{C}} : n^3 + L_C n^2 + 2L_C n \ \text{ flops} \tag{28}$$

$$(\bar{\mathbf{N}}_\mathrm{C}^T\bar{\mathbf{N}}_\mathrm{C})^{-1}\Big(\bar{\mathbf{N}}_\mathrm{C}\tilde{\mathbf{q}}_{d,\mathrm{C}}\Big) : n^3 + L_C n^2 + 4L_C n \ \text{ flops}. \tag{29}$$

where $\tilde{\mathbf{q}}_{d,\mathrm{C}} = \mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\hat{\mathbf{p}}_\mathrm{P}$. By factoring

$$\bar{\mathbf{N}}_\mathrm{C} = \mathbf{Q}\mathbf{R} \tag{30}$$

with the modified Gram Schmidt algorithm [31], where $\mathbf{Q} \in \mathbb{R}^{L_C \times L_C}$ is an orthogonal matrix (i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$) and $\mathbf{R} \in \mathbb{R}^{L_C \times n}$ is an upper triangular matrix, the problem in Eq. (18) can be written as

$$\mathbf{R}\bar{\mathbf{p}}_{c,\mathrm{C}} = \mathbf{Q}^T\tilde{\mathbf{q}}_{d,\mathrm{C}} \tag{31}$$

which can be solved using backward substitution. The number of operations required using this method are

$$\bar{\mathbf{N}}_\mathrm{C} = \mathbf{Q}\mathbf{R} : L_C n^2 \ \text{ flops} \tag{32}$$

$$\mathbf{w} = \mathbf{Q}^T\tilde{\mathbf{q}}_{d,\mathrm{C}} : L_C n^2 + 2L_C n \ \text{ flops} \tag{33}$$

$$\mathbf{R}\bar{\mathbf{p}}_{c,\mathrm{C}} = \mathbf{w} : n^2 + L_C n^2 + 2L_C n \ \text{ flops}. \tag{34}$$

Note that the QR factorization solution is more efficient to compute than the pseudoinverse.

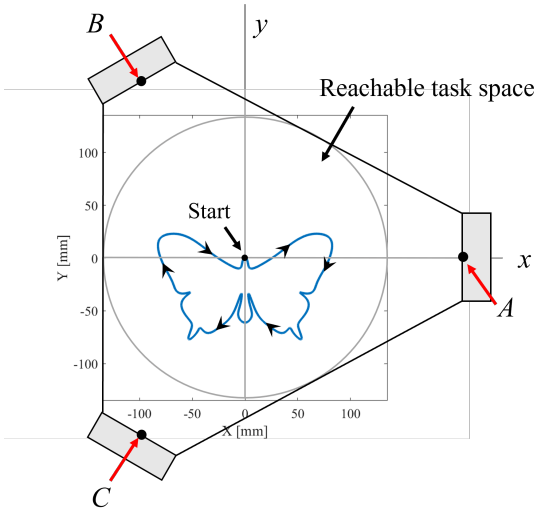

Figure 5. Trajectory of a butterfly used for simulations overlaid on the task space of the delta 3D printer. The butterfly spans $x \in [-82,82]$ mm, $y \in [-77,23]$ mm with a maximum motion speed of 150 mm/s and a maximum acceleration of 20 m/s^2.

IV. SIMULATION AND EXPERIMENTAL VALIDATION

A. Simulation validation

In this subsection, we focus on simulations of the MP Delta Pro 3D printer from Fig. 1. The simulation model is used to validate our assumptions about computation time and preserved accuracy of the controller proposed in Section III. We evaluate the performance of the following controllers:

(a) a controller where the transfer functions are computed in matrix form using Eq. (4) for all points in the window and the coefficients are calculated with the pseudoinverse;
(b) a controller that is the same as controller (a), except the transfer functions are computed using the parameterized model (Sec. III-B);
(c) a controller that is the same as controller (b), except the transfer functions are computed for only one point in the window–*without* the switching compensation discussed in Sec. III-C;
(d) a controller that is the same as controller (c), except *with* the switching compensation; and
(e) a controller that is the same as controller (d), except the coefficients are calculated with QR factorization (our proposed controller).

By adding the proposed modifications separately, we can distinguish the computational efficiency and accuracy effects of each modification. Each controller's performance is compared to a baseline controller–the standard FBS controller using an LTI model for the carriages at $(x,y) = (0,0)$ mm. Without a model for the position-dependent dynamics, the control designer must use one LTI model for FBS and a reasonable choice is the model at the center of the task space.

The simulations are conducted using the trajectory of a butterfly shown in Fig. 5, which spans $x \in [-83,83]$ mm, $y \in [-77,23]$ mm with a maximum speed of 150 mm/s and a maximum acceleration of 20 m/s^2. The trajectory lasts 5 seconds with a sample time of 1 ms (i.e., it contains 5,000 points). The LSR of $\mathbf{G}(s)$ is computed using the known trajectory points and used to simulate the system's response. Parameter-varying compensation for all points (controllers (a) and (b)) is implemented by computing a point-by-point LSR matrix for each window. In other words, we compute the transfer function at each point in the window and compute its impulse response, which becomes the time-shifted columns of the LSR matrix (see Appendix A). For the single point compensation (controllers (c)-(e)), we compute the transfer function and impulse response for the median point in the window, whose time-shifted impulse response is repeated to construct the LSR matrix for each window. All controllers use B-spline basis functions of degree $m = 5$, a window size $L_C = 196$ points, number of B-spline coefficients $n = 44$, and number of updated coefficients $n_{up} = 22$. The window size is determined by the amount of time required for the impulse response of a transfer function (IIR filter) to settle close to zero (see [4]). Since the dynamics vary for the delta 3D printer, we construct a grid of positions in the reachable workspace that are 5 mm apart, compute the impulse response for the transfer function in each position, and use the worst-case settling time to determine the L_C parameter for all windows. The number of B-spline coefficients is computed from the window length as described in [4].

The simulations were run in Matlab (version R2022a) on a 64-bit Microsoft Surface Book with an Intel Core i5-6300U CPU processor and 8 GB of RAM. The computation time of the entire modified trajectory and root-mean-square (RMS) of the contour error across the trajectory points for each controller are reported in Table I. The percent difference of the RMS contour error compared to the baseline controller simulation is also reported and the contour error comparison is shown in more detail in Fig. 6. The RMS contour error of the baseline controller is 11.1 μm and the trajectory is computed in 45 ms because the filtering and inversion is completed offline. The RMS error of the exact LPV model is almost 30 times less at 0.4 μm. However, note that the computation of the matrix model online is much larger (820 s), which is also about 4 times greater than the computation time of the parameterized model (226 s) without any change in the RMS error. When we compute the model for a single point in each window, we can reduce the computation time by about 20 times (to 10 s) but at a cost of about 10x increase in RMS contour error (3.2 μm). The accuracy is improved to only be about 1.3x worse than the exact model (about 0.5 μm) when the switching compensation is implemented.

In Fig. 6, there is a spike in the contour error of controller (c) (no switching compensation) around 2 seconds into the motion. The spike represents a difficult portion of the trajectory–the bottom right of the butterfly wing–between which a switch in models occurs for controllers (c)-(e). Here, the points are on the far side of carriage B's line of action (see Figs. 2 and 5) which, as discussed in [17] and in the following subsection,

Table I
SIMULATION RESULTS COMPARING COMPUTATION TIME AND ACCURACY OF DIFFERENT CONTROLLERS FOR GENERATING MODIFIED BUTTERFLY TRAJECTORY

	Computation Time	RMS Contour Error	% Improvement from Baseline
(*) Baseline LTI, standard FBS controller	0.044 s	11.13 μm	–
(a) Matrix TFs, all points, pseudoinverse	820.06 s	0.38 μm	96.6%
(b) Parameterized TFs, all points, pseudoinverse	226.30 s	0.38 μm	96.6%
(c) Parameterized TFs, single point, pseudoinverse	9.86 s	3.21 μm	71.1%
(d) Same as above with switching compensation	13.46 s	0.53 μm	95.3%
(e) Same as above with QR factorization	9.65 s	0.53 μm	95.3%

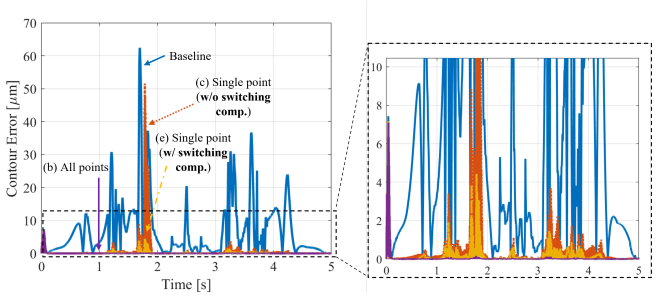

Figure 6. Contour error of the modified trajectories generated by the baseline controller (solid blue line) and controllers (b) using all trajectory points (solid purple line), (c) a single point without switching compensation (dotted red line), (e) and a single point with switching compensation (dash-dotted yellow line).

is prone to larger dynamic variation. (The symmetric points on the bottom left side of the butterfly are not on the far side of carriage *C*'s line of action and, thus, have less dynamic variation). Note that the errors increase for both single point controllers but are exacerbated by the controller without switching compensation. Situations like this one illustrate the utility of using switching compensation for changing models. Finally, note that the computation time reduces by 28% using QR factorization instead of the pseudoinversion when the controller has switching compensation (from about 13.5 to 9.7 s). Overall, the accuracy of the single point approach is worse than using all points, but the overall accuracy improvement is acceptable given the 23x decrease in computation time compared to using the parameterized exact controller (b), which would be challenging to implement on hardware in real-time. In the following subsection, our experiments on the delta 3D printer show that our proposed controller results in significant accuracy improvements compared to the baseline.

B. Experimental validation

In this subsection, we discuss the results of printing a standard "calibration cube"[1] on the delta 3D printer of Fig. 1 using two control strategies: the baseline controller and our proposed controller (case (e) above). Our aim is to demonstrate the utility of our contributions by comparing: (a) the visual quality of parts printed with our controller and the baseline controller at different positions and (b) acceleration amplitudes of the carriages during the execution of each print to understand the effects of our proposed controller.

For the experiments, we locate the center of the calibration cube at the following positions: $(x,y) = (0,0)$, $(-80,0)$, $(40,-69)$, and $(40,69)$ mm. Each position, except the origin, is chosen to target each carriage independently; they are located 80 mm from the origin along the far side of the respective carriage's line-of-action (see Fig. 2). As shown in Figs. 2 and 7, the position $(-80,0)$ mm primarily tests variation in carriage *A*'s dynamics, $(40,-69)$ mm primarily tests variation in carriage *B*'s, and $(40,69)$ mm primarily tests variation in carriage *C*'s [17]. We set the maximum speed and acceleration at 150 mm/s and 20 m/s^2, respectively. Both the baseline and proposed controllers are implemented in Matlab Simulink, which sends motion commands through a dSPACE MicroLabBox to Pololu DRV8825 stepper motor drivers to operate the stepper motors on the delta 3D printer. For the baseline controller, we fit the measured transfer functions at $(0,0)$ mm (see Fig. 7) as the LTI model for the standard FBS approach. The proposed controller uses the LPV model and FBS implementation described in Secs. II and III, respectively. For comparison, we also print the calibration cube at the same positions without vibration compensation.

Figures 8 and 9 show images of the X and Y faces of the calibration cube, respectively, manufactured at the different positions. A visual inspection of the parts reveals the following observations:

1) In the uncompensated parts, there are vibration marks at the edges where there is a change of direction, which are largely eliminated with FBS compensation.
2) The quality of the parts printed at $(0,0)$ mm are similar for both the baseline and proposed controllers.
3) The surface quality of the part printed at $(-80,0)$ mm with the baseline controller is worse than the quality of the part printed with the proposed controller.
4) The quality of the parts printed at $(40,-69)$ mm are similar for both controllers.
5) The part printed at $(40,69)$ mm with the baseline controller drifts from its starting position in the middle of the print, while the part printed with the proposed controller stays aligned.
6) The quality of the parts printed with the proposed controller are always either similar to or better than the parts printed with the baseline controller.

[1]The standard XYZ calibration cube used in this paper can be found at: https://www.thingiverse.com/thing:1278865

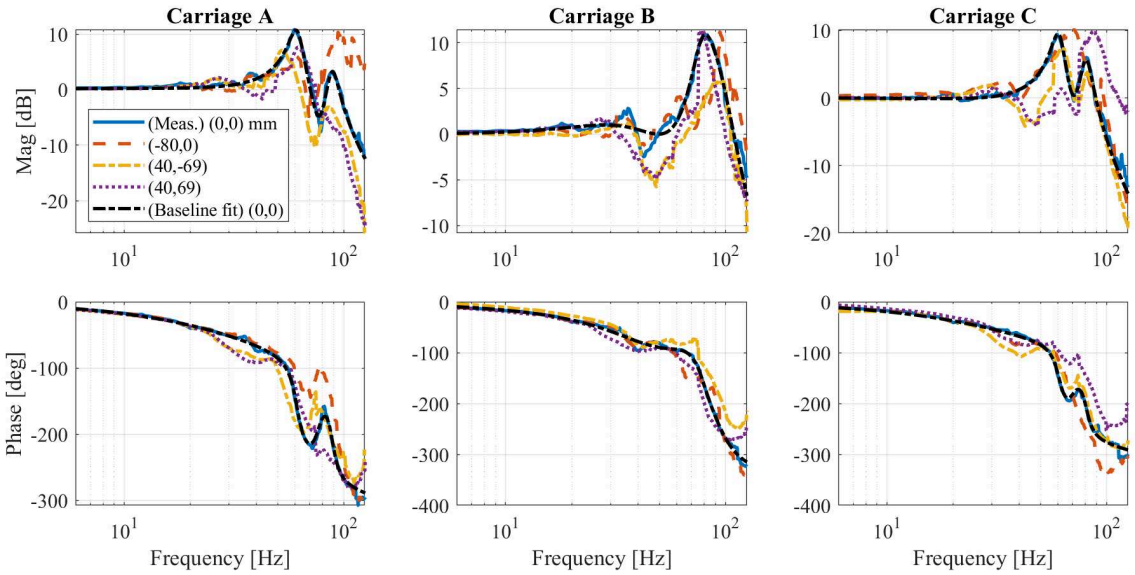

Figure 7. Measured frequency response functions of the carriage position dynamics at $(x,y) = (0,0)$ mm (blue solid lines), $(-80,0)$ mm (red dashed lines), $(40,-69)$ mm (yellow dash-dotted lines), and $(40,69)$ mm (indigo dotted lines) of the Monoprice Delta Pro 3D printer. The black dashed lines indicate the fitted transfer functions of the baseline model (at $(x,y) = (0,0)$ mm) that is used for the baseline controller.

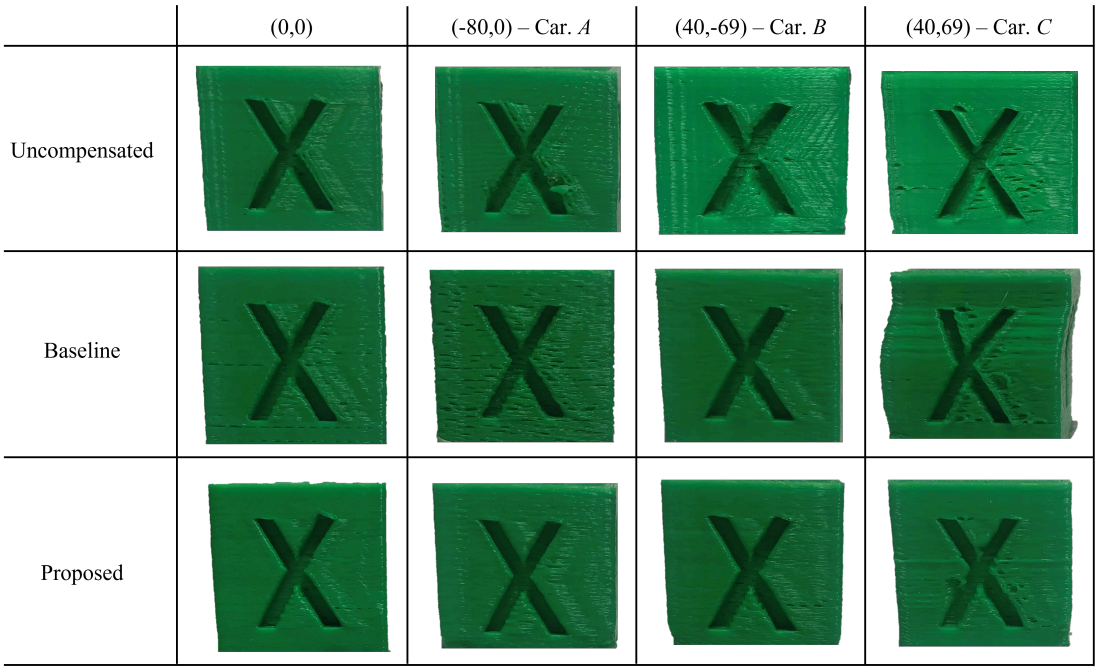

Figure 8. X-axis face of calibration cubes fabricated with the baseline and proposed controllers centered at different positions that target different carriages.

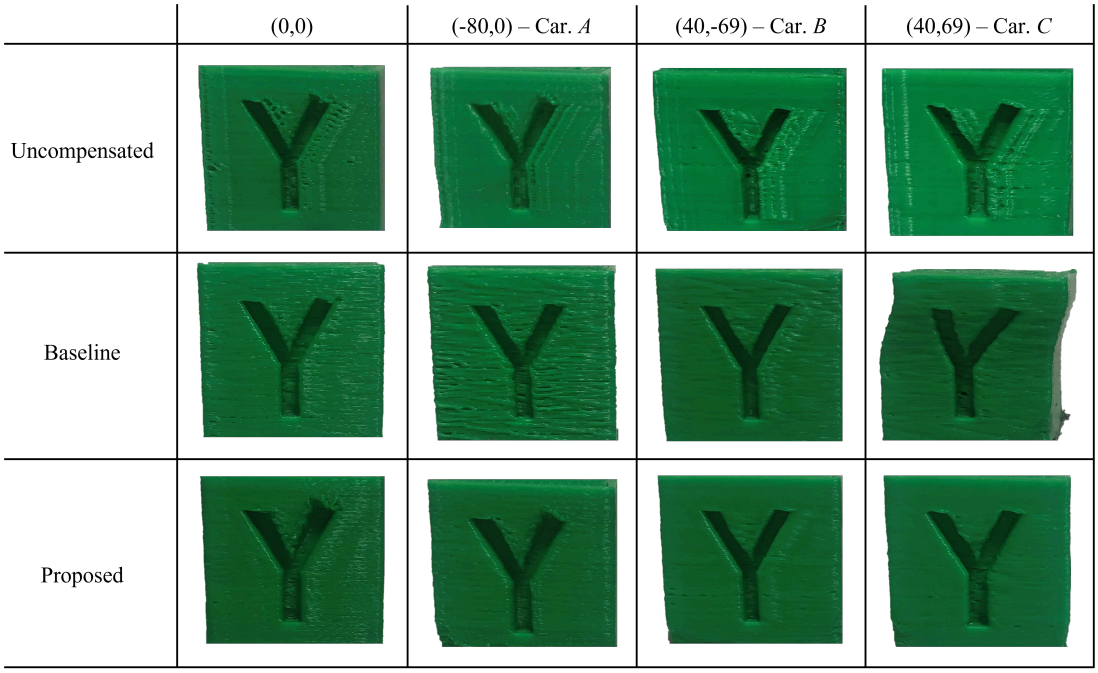

Figure 9. Y-axis face of calibration cubes fabricated with the baseline and proposed controllers centered at different positions that target different carriages.

Observation 2 is expected since the baseline controller performs optimally at $(0,0)$ mm and observation 3 is expected due to the model mismatch. However, observation 4 appears to be an anomaly. A closer look at carriage B's FRFs in Fig. 7 reveals that the measured FRFs from $(0,0)$ and $(40,-69)$ mm have similar resonance frequencies. Also note that carriages A and C have measured FRFs from $(0,0)$ and $(40,-69)$ mm that also have similar resonance frequencies. Hence, the baseline controller is able to adequately compensate vibrations while printing at $(40,-69)$ mm. The drifting signal in observation 5 is due to the baseline controller overcompensating for the fast changes in acceleration on the top half of the Y-face of the cube. Note that the bottom half of the Y-face only has one indentation, while the top half has two indentations in succession, which increases the high frequency content of the acceleration profile. Figure 10 shows the modified motion commands of the baseline and proposed controllers in this region of the print, which shows that the commanded motion of the baseline controller drifts from the desired command while the proposed controller does not. Overcompensation occurs because the baseline model for carriage C (at $(0,0)$ mm in Fig. 7) shows that the amplitude of high frequency content is reduced. Hence, the baseline controller attempts to increase the input of the high frequency commands to achieve the desired motion. However, we know from the measured FRF at $(40,69)$ that the command does not need to be amplified. Thus, the proposed controller, with more accurate dynamics, can compensate correctly.

To quantify the reduction of vibration-induced acceleration, we measure the acceleration of the carriages during each print using the vertical (z-) axis of an ADXL335 3-axis accelerometer from Sparkfun Electronics and compare the acceleration for both controllers to the acceleration of the desired trajectory. Tables II and III give the RMS and maximum values of the carriage acceleration, respectively, during the top half of the calibration cube print. In absolute terms, the proposed controller accelerations are closer to desired acceleration in all cases, illustrating reduction in vibration errors. The maximum difference of deviation reduction of the proposed controller compared to the baseline controller is 8.9% for carriage C at $(40,69)$ in the RMS acceleration and 20.8% for carriage A at $(-80,0)$ in the maximum acceleration. Following the result from observation 4, we note that the least deviation from the desired acceleration occurs for carriage B at $(40,-69)$ for both RMS and maximum acceleration.

Table II
ROOT MEAN SQUARE (RMS) ACCELERATION OF CARRIAGES DURING PRINT OF CALIBRATION CUBE

	Baseline [m/s^2]	Proposed [m/s^2]	Desired [m/s^2]
$(-80,0)$ – Car. A	3.73 (+0.38)	3.24 (-0.11)	3.35
$(40,-69)$ – Car. B	4.04 (+0.08)	3.95 (-0.01)	3.96
$(40,69)$ – Car. C	4.16 (+0.35)	3.82 (+0.01)	3.81

Table III
MAXIMUM ACCELERATION OF CARRIAGES DURING PRINT OF CALIBRATION CUBE

	Baseline [m/s^2]	Proposed [m/s^2]	Desired [m/s^2]
$(-80,0)$ – Car. A	22.46 (+4.57)	17.04 (-0.85)	17.89
$(40,-69)$ – Car. B	24.67 (+0.81)	24.05 (+0.19)	23.86
$(40,69)$ – Car. C	25.53 (+1.67)	23.68 (-0.18)	23.86

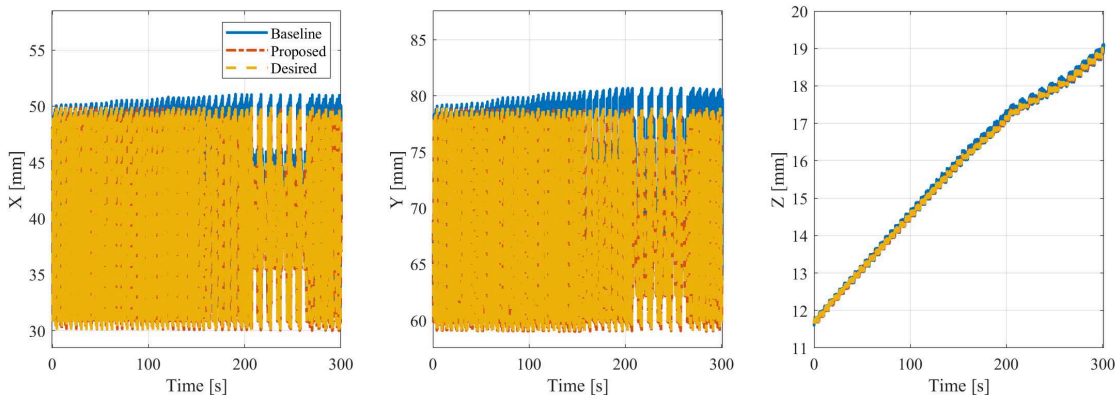

Figure 10. Commanded motion from the baseline (blue solid line) and proposed (red dash-dot line) controller during the drifting motion for the baseline controller while fabricating the part at $(x,y) = (40,69)$ mm (triggering carriage C). Note that the baseline model commands increasing deviations from the nominal position (yellow dashed line), which leads to the drifting part in Figs. 8 and 9. The baseline controller creates the drifting commands because of the differences between the baseline frequency response function and the actual frequency response function at $(40,69)$ mm.

V. CONCLUSIONS

This paper proposes practical techniques to enable real-time, accurate vibration compensation on the prismatic-joint delta 3D printer. Previous work on improving accuracy of delta manipulators has focused on servo motor actuated machines and relies on sensor measurements and feedback control. For most delta 3D printers, feedback sensors are not available so we must employ feedforward control with an accurate model. To achieve this objective, we aimed to use an accurate LPV model of the delta 3D printer we recently proposed in [17] with the model-inversion based FBS approach. However, the need to recompute the model and controller at each new configuration during real-time control is computationally challenging. Therefore, we propose the following to decrease the computational burden: (1) parameterization and pre-filtering of portions of the model for fast online operations, (2) computation of the model at sampled points along the trajectory (while preserving continuity of the controller's predictions when the model changes), and (3) utilization of matrix methods that yield faster matrix inversion.

Simulations are used to assess the trade off between computation time and accuracy. We report that the techniques presented in this paper result in a 23x reduction in computation time from the exact parameter varying controller which recomputes the model/controller at every point. Thus, our approximations save significant computational effort while only increasing contour errors by about 1.3x compared to the exact controller. Images of parts from our experiments also show an overall improvement in the quality of parts printed at different locations using the proposed controller compared to using a baseline controller optimized for the center of the workspace. Furthermore, acceleration measurements during printing show more than 20% reduction of vibration-induced accelerations for the proposed controller when compared to the baseline. This work shows that we can take advantage of the high speed motion of the delta 3D printer (compared to traditional 3D printers) and apply feedforward controllers like FBS to maintain accuracy during vibration-prone motion. This paper's contributions bring us one step closer to the vision of high speed and high quality additive manufacturing.

APPENDIX

A. *Lifted system representation of a digital filter*

As discussed in [22], consider digital filter p, input signal u, and output signal y defined as:

$$p = \{p_{-2} \;\; p_{-1} \;\; p_0 \;\; p_1 \;\; p_2\} \tag{35}$$
$$u = \{u_0 \;\; u_1 \;\; u_2\} \tag{36}$$
$$y = \{y_0 \;\; y_1 \;\; y1\} \tag{37}$$

Signals y and u and filter p are related by the convolution operator as follows:

$$y = u * p \tag{38}$$

From Eqs. 35-38,

$$y_0 = p_0u_0 + p_{-1}u_1 + p_{-2}u_2 \tag{39}$$
$$y_1 = p_1u_0 + p_0u_1 + p_{-1}u_2 \tag{40}$$
$$y_2 = p_2u_0 + p_1u_1 + p_0u_2 \tag{41}$$

This can be expressed in matrix form as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} p_0 & p_{-1} & p_{-2} \\ p_1 & p_0 & p_{-1} \\ p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \tag{42}$$

Note that the main diagonal element (p_0) represents the influence of the current input on the current output; the first upper diagonal element (p_{-1}) represents the influence of the succeeding input on the current output and the second upper diagonal element (p_{-2}) represents the influence of the second succeeding input on the current output. Similarly, the first (p_1) and second lower (p_2) elements represent the influence of the first and second preceding inputs on the current output,

respectively. Hence, the discrete time transform of p obtained from Eq. 42 is given by

$$p_2 z^{-2} + p_1 z^{-1} + p_0 z^0 + p_{-1} z^1 + p_{-2} z^2 \tag{43}$$

which is in accordance with the time-domain definition given in Eqs. 35-37.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation [grant numbers 2054715 and DGE 1256260] and a Michigan Space Grant Consortium (MSGC) graduate fellowship from the National Aeronautics and Space Administration (NASA), under award number 80NSSC20M0124. A company founded by C.E. Okwudire holds a commercial license for the filtered B-splines (FBS) algorithm.

REFERENCES

[1] K. Miller, "Experimental verification of modeling of delta robot dynamics by direct application of Hamilton's principle," in *Proc. IEEE Int. Conf. Robot. and Autom.*, pp. 532-37, 1995, doi: 10.1109/ROBOT.1995.525338.

[2] O.V. Zakharov, K.G. Pugin, and T.N. Ivanova, "Modeling and analysis of delta kinematics FDM printer," *J. of Physics: Conf. Series*, vol. 2182, no. 1, 2022, doi:10.1088/1742-6596/2182/1/012069.

[3] K.S. Ramani, N. Edoimioya, C.E. Okwudire, "A robust filtered basis functions approach for feedforward tracking control – with application to a vibration-prone 3D printer," *IEEE/ASME Trans. Mechatronics* vol. 25, no. 5, pp. 2556-2564, 2020, doi: 10.1109/TMECH.2020.2983680.

[4] M. Duan, D. Yoon, and C.E. Okwudire, "A limited-preview filtered B-spline approach to tracking control – with application to vibration-induced error compensation of a 3D printer," *Mechatronics* vol. 56, pp. 287-296, 2018, doi: 10.1016/j.mechatronics.2017.09.002.

[5] H. Kim and C.E. Okwudire, "Simultaneous servo error pre-compensation and feedrate optimization with tolerance constraints using linear programming," *Int. J. of Adv. Manufac. Tech.*, vol. 109, no. 3-4, pp. 809-821, 2020, doi: 10.1007/s00170-020-05651-w.

[6] C.E. Okwudire, S. Huggi, S. Supe, C. Huang, and B. Zeng, "Low-level control of 3D printers from the cloud: a step toward 3D printer control as a service," *Inventions*, vol. 3(3), no. 56, 2018, doi: 10.3390/inventions3030056.

[7] A. Codourey, "Dynamic modeling of parallel robots for computed-torque control implementation," *Int. J. Robot. Res.*, vol. 17, no. 18, pp. 1325-1336, 1998, doi: 10.1177/027836499801701205.

[8] L. Angel, J. Viola, "Fractional order PID for tracking control of a parallel robotic manipulator type delta," *ISA Trans.*, vol. 79, pp. 172-88, 2018, doi: 10.1016/j.isatra.2018.04.010.

[9] C.E. Boudjedir, D. Boukhetala, and M. Bouri, "Nonlinear PD plus sliding mode control with application to a parallel delta robot," *J. Electr. Eng.*, vol. 69, no. 5, pp. 329-336, 2018, doi: 10.2478/jee-2018-0048.

[10] M. Ramirez-Neria, H. Sira-Ramírez, A. Luviano-Juárez, and A. Rodrguez-Ángeles, "Active disturbance rejection control applied to a delta parallel robot in trajectory tracking tasks," *Asian J. Control*, vol. 17, no. 2, pp. 636-647, 2015, doi: 10.1109/ACC.2012.6314934.

[11] L.A. Castañeda, A. Luviano-Juárez, and I. Chairez, "Robust trajectory tracking of a delta robot through adaptive active disturbance rejection control," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1387-98, 2015, doi: 10.1109/TCST.2014.2367313.

[12] J.M. Escorcia-Hernandez, H. Aguilar-Sierra, O. Aguilar-Mejia, A. Chemori, and J.H. Arroyo-Nunez, "An intelligent compensation through B-spline neural network for a delta parallel robot," in *Proc. 6th Int. Conf. Control, Decis. Inf. Technolo. (CoDIT)*, pp. 361-366, 2019, doi: 10.1109/CoDIT.2019.8820472.

[13] Y. Su, D. Sun, L. Ren, and J.K Mills, "Integration of saturated PI synchronous control and PD feedback for control of parallel manipulators," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 202-207, 2006, doi: 10.1109/TRO.2005.858852.

[14] P. Chiacchio, F. Pierrot, L. Sciavicco, and B. Siciliano, "Robust design of independent joint controllers with experimentation on a high-speed parallel robot," *IEEE Trans. Ind. Electron.*, vol. 40, no. 4, pp. 393-403, 1993, doi: 10.1109/41.232228.

[15] Y. Zhiyong and H. Tian, "A new method for tuning PID parameters of a 3 DoF reconfigurable parallel kinematic machine," *Proc. IEEE Int. Conf. Robot. Autom., 2004. Proceedings*, pp. 2249-2254, 2004, doi: 10.1109/ROBOT.2004.1307396.

[16] Q. Zhou, W. Panfeng, and M. Jiangping, "Controller parameter tuning of delta robot based on servo identification," *Chinese J. of Mech. Eng.*, vol. 28, no. 2, pp. 267-275, 2015, doi: 10.3901/CJME.2014.1117.169.

[17] N. Edoimioya and C.E. Okwudire, "A Generalized and Efficient Control-oriented Modeling Approach for Vibration-prone Delta 3D printers using Receptance Coupling," *Trans. Autom. Science Eng. (TASE)*, 2022. doi: 10.1109/TASE.2022.3197057

[18] J. van Zundert and T. Oomen, "On inversion-based approaches for feedforward and ILC," *Mechatronics*, vol. 50, pp. 282–291, 2018, doi: 10.1016/j.mechatronics.2017.09.010.

[19] K. Erkorkmaz and Y. Altintas, "High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation," *Int. J. Mach. Tools Manuf.*, vol. 41, no. 9, pp. 1323–1345, 2001, doi: 10.1016/S0890-6955(01)00002-5.

[20] S. Tajima and B. Sencer, "Online interpolation of 5-axis machining toolpaths with global blending", *Int. J. Mach. Tools Manuf.*,vol. 175, 103862, 2022, doi: 10.1016/j.ijmachtools.2022.103862.

[21] W. Singhose, "Command shaping for flexible systems: a review of the first 50 years," *Int. J. Precis. Eng. Manuf.* vol. 10, no. 4, pp. 153–168, 2009, doi: 10.1007/s12541-009-0084-2.

[22] K. S. Ramani, M. Duan, C. E. Okwudire, and A. G. Ulsoy, "Tracking control of linear time-invariant nonminimum phase systems using filtered basis functions", *J. Dyn. Syst. Meas. Control*, vol. 139, no.1, 011001, 2017, doi: 10.1016/j.cirp.2016.04.100.

[23] B.P. Rigney, L.Y. Pao, and D.A. Lawrence, "Nonminimum phase dynamic inversion for settle time applications," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 989–1005, 2009, doi: 10.1109/TCST.2008.2002035.

[24] G.M. Clayton, S. Tien, K.K. Leang, Q. Zou, and S. Devasia, "A review of feedforward control approaches in nanopositioning for high-speed SPM," *J. Dyn. Syst. Meas. Control*, vol. 131, no. 6, 061101, 2009, doi: 10.1115/1.4000158.

[25] C.E. Okwudire, K.S. Ramani, and M. Duan, "A trajectory optimization method for improved tracking of motion commands using CNC machines that experience unwanted vibration," *CIRP Ann. Manuf. Technol.*, vol. 65, no. 1, pp. 373–376, 2016, doi: 10.1016/j.cirp.2016.04.100.

[26] Y. Kasemsinsup, R. Romagnoli, M. Heertjes, S. Weiland, and H. Butler, "Reference-tracking feedforward control design for linear dynamical systems through signal decomposition," *Am. Control Conf.* pp. 2387–2392, 2017, doi: 10.23919/ ACC.2017.7963310.

[27] N. Edoimioya, K.S. Ramani, and C.E. Okwudire, "Software Compensation of Undesirable Racking Motion of H-frame 3D Printers using Filtered B-Splines," *Additive Manuf.*, vol. 47, 2021, doi: 10.1016/j.addma.2021.102290.

[28] L. Piegl and W. Tiller, *The NURBS book*, Heidelberg: Springer, Berlin, 1995.

[29] M. Duan and C. Okwudire, "Minimum-time cornering for CNC machines using an optimal control method with NURBS parameterization," *Int. J. Adv. Manuf. Technol.*, vol. 85, pp. 1405-1418, 2016. https://doi.org/10.1007/s00170-015-7969-2.

[30] G.H. Golub and C.F. Van Loan, *Matrix Computations*, third edition, The Johns Hopkins University Press, Baltimore, 1996.

[31] H.R. Schwarz, H. Rutishauser, and E. Stiefel, *Numerical analysis of symmetric matrices*, translation of *Numerik symmetrischer Matrizen*, Prentice-Hall, Englewood Cliffs, N.J., 1973.