Modeling and Feedforward Vibration Compensation of Advanced Manipulators for Extrusion-Based Additive Manufacturing

by

Nosakhare Edoimioya

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2023

Nosakhare Edoimioya

nosed@umich.edu

ORCID iD: 0000-0002-6183-3466

This dissertation is dedicated to my parents, who sacrificed selflessly for me to be here.

ACKNOWLEDGEMENTS

Working on a doctoral degree has been the most formative intellectual experience of my life thus far. I will be eternally grateful to all the people who supported me along the way and I would like to take this opportunity to express my sincere gratitude towards them.

First, I would like to thank my advisor, Professor Chinedum Okwudire, for his guidance and support. I met Prof. Okwudire at a professional society conference in 2016 and he offered me the opportunity to work in his lab as an undergraduate researcher. During that summer, I began the work that would eventually become this dissertation. During my time working with him, Prof. Okwudire and I have had numerous conversations about our interest in working on projects with potential for real-world, global impact. He has encouraged me to have big dreams and taught me how to ground them in practical realities. I have learned how to be a better researcher, leader, engineer, and teacher and I want to thank Prof. Okwudire for training me and allowing me to explore other interests along the way. I would like to also thank my committee members Profs. Kira Barton, Albert Shih, and Wesley McGee. Their guidance and feedback has been valuable to me and is greatly appreciated. In particular, learning about Prof. McGee's journey and work was incredibly inspiring and has influenced the next step in my career.

I would like to also thank all my past and present colleagues at the Smart and Sustainable Automation Research Lab for their intellectual support and friendship. I would like to especially thank Dr. Keval Ramani, Heejin Kim, and Cheng-Hao

Chou for their valuable collaboration and feedback on my research. I would like to thank the National Science Foundation (NSF), the National Aeronautics and Space Administration (NASA), and Rackham Graduate School for supporting this research. I would like to take this opportunity to also thank family and friends in Michigan, as well as Texas and California, for their support and encouragement. Last but not the least, I would like to thank my parents and my brother for their support throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

AM Additive Manufacturing

FFF Fused Filament Fabrication

FF Feedforward

SCG Smooth Command Generation

FBF Filtered Basis Function

FBS Filtered B-Splines

LPFBS Limited Preview Filtered B-Splines

IS Input shaping

LTI Linear Time-Invariant

LPV Linear Parameter-Varying

MP Monoprice

SISO Single-Input, Single-Output

MIMO	Multi-Input, Multi-Output
CT	Computed Torque
ANN	Artificial Neural Network
FRF	Frequency Response Function
PD	Proportional-derivative
PID	Proportional-integral-derivative
CNC	Computer Numeric Controlled
RC	Receptance Coupling
RMS	Root-mean-square
LSR	Lifted System Representation

ABSTRACT

There are several challenges impeding widespread adoption of additive manufacturing (AM). In extrusion-based AM, which is the focus of this dissertation, slow production speed and position accuracy in the end-use parts are significant problems. Comparatively, mass-manufacturing processes can make accurate parts with fast production rates. Fortunately, innovative architectures capable of reaching higher speeds, such as the H-frame and delta robot manipulators, are being introduced for use in extrusion-based AM. For these manipulators, maintaining position accuracy at high speeds is challenging because they have complex (i.e., coupled and nonlinear) dynamics: they have nonlinear motion errors that are difficult to model and compensate. The major contributions of this dissertation are to present novel methods to characterize their dynamics with linear parameter-varying (i.e., position-dependent) models and apply feedforward vibration compensation to mitigate their motion-induced errors.

In the vibration compensation literature, researchers have used a model-based vibration control technique known as the filtered B-splines (FBS) approach to increase production rates on traditional printer architectures by 2x without sacrificing accuracy. The FBS approach executes tracking control of a given trajectory by first expressing the control input to the machine as a linear combination of B-spline basis functions. The basis functions are then forward filtered through the machine's dynamics and the coefficients are obtained such that the tracking error is minimized. In this manner, a desired manufacturing trajectory can be accurately tracked at high speeds. However, the FBS approach has only been applied to systems with linear time-invariant dynamics and has not been applied to systems with coupled and position-dependent dynamics. Additionally, coupled dynamics result in larger inversion problems when calculating the FBS controller and position-dependent dynamics require recomputing the models to account for changing dynamics. Both phenomena lead to increased computational effort required to implement FBS. To decrease the computational effort of deploying FBS on advanced manipulators, this dissertation proposes methods to: (1) decouple the coupled dynamics while maintaining high control accuracy and (2) efficiently compute and invert position-dependent models during real-time manufacturing by separating time-invariant and time-varying parts of the model, parameterizing time-varying models, and using efficient matrix methods for computation. We present numerical simulations that demonstrate the effectiveness of these methods to bolster computational efficiency. For example, we implement FBS on the delta robot with a nearly 40% improvement in computational efficiency when compared to a standard FBS approach that does not use the new methods.

Successful vibration compensation also depends on having accurate mathematical models of the controlled system. To obtain the models, we approximate the nonlinear models of the manipulators as linear models using linearization methods and validate that the approximations do not result in loss of accuracy using data from the machines. Additionally, to facilitate adoption of the proposed control approach, this dissertation presents techniques for efficient identification of the model parameters with a few measurements from the H-frame and delta manipulators. We outline the experimental procedures necessary to perform the system identification and present data from commercial 3D printers demonstrating that the prediction model matches the real system at different operating positions—up to 50% better than predicting with a model measured at one central location. More importantly, we show that using these models results in up to 39% reduction of vibration-induced accelerations when compared to a model that does not change based on the printer's position.

CHAPTER I

Introduction and Literature Review

Additive Manufacturing (AM), also known as three-dimensional (3D) printing, is a manufacturing process that creates parts by locally depositing and fusing material together in 3D space. The material is often deposited layer-wise—in 2D cross-sections of the intended 3D geometry. Using AM, it is possible to create an (almost) infinite number of geometries without the need for reconfigured or retooled machines, which is different from most traditional manufacturing processes where retooling the machine is often required when the part geometry changes. This versatility has created the conditions for 3D printing to disrupt traditional manufacturing. Accordingly, many companies view AM as a critical part of their strategy in the coming decade [1]. As an example, in 2020, GE Aviation announced that their GE9X engines, which contain over 300 3D printed parts, had been used on the first flight of the Boeing 777X commercial jetliner and passenger plane [2]. Incorporating 3D printed parts enabled GE engineers to manufacture parts with geometries that cannot be realized with traditional manufacturing methods. Similar examples of 3D printing's flexibility can be found in a wide variety of sectors including fashion, architecture/construction, aerospace/defense, and healthcare [3].

The adoption of 3D printing has been advanced by the fused filament fabrication (FFF) technology, which is used to create desktop 3D printers that are cheap enough for average U.S. consumers to purchase. FFF is an extrusion-based approach, where melted plastic is forced through a nozzle and deposited on a print bed. The easy-to-use technology has begun to democratize access to manufacturing [4]. In 2021, about a million U.S. consumers owned 3D printers and FFF 3D printers made up more than 70% of the total market [1]. Despite their popularity, FFF 3D printers are known to operate with slow production times—on the order of hours to days—to create parts of reasonable quality. Furthermore, attempting to increase production speed can lead to significant degradation in the part quality due to vibration-induced errors and other process errors [5–8]. Compared to traditional manufacturing processes, which are optimized for high production speeds and accurate part dimensions, 3D printing cannot yet meet the speed and quality requirements of industry. Hence, the technology is still confined to specialized parts and industries where the benefits of customization outweigh the costs.

Achieving industrial adoption requires an increase in the throughput of 3D printers without any loss of (and perhaps with an increase in) accuracy. Consequently, this thesis is focused on developing novel techniques to use motion control to increase the productivity and accuracy of extrusion-based FFF systems. Our approach seeks to increase throughput by leveraging advanced manipulators designed to reach high speeds, while maintaining accurate motion using control algorithms in the software.

1.1 Background and Motivation

A number of techniques have been studied to address the slow speed and poor part quality issues in AM. These techniques can be roughly classified into two categories: (1) approaches to improve the motion systems and (2) approaches to improve manufacturing processes and materials. Regarding motion systems, researchers typically consider changes to the actuator design, mechanical design, and controller design to improve the machine's overall speed and accuracy performance. Regarding the manufacturing processes and materials, researchers seek to understand how to model and control the filament material dynamics, how to improve the filament heating technology, and innovative alterations to the chemical compositions of materials used the process to enhance the machine's response to fast extrusion rates. In this thesis, we focus on the design and control of motion systems for extrusion-based 3D printers. However, we are cognizant of the intimate coupling between the motion system design and the manufacturing process and materials used in 3D printing. We believe that approaches from both categories will eventually need to be combined to maximize the speed and quality of 3D printers.

Recently, advanced manipulators with improved dynamic performance have been introduced to increase productivity in 3D printing. In this thesis, we focus on two such architectures that are becoming popular for extrusion-based 3D printing: (1) the H-frame gantry and (2) the delta robot. The H-frame gantry (Fig. 1.1(a)) is a parallel-axis architecture that uses two stationary motors mounted to the printer frame to control the end-effector through a single timing belt (Section II contains more details). Since the motors are stationary, we can leverage the lighter moving mass of the printer's gantry to improve its dynamic performance when compared to traditional serial-axis 3D printers (Fig. 1.1(c)). The delta robot (Fig. 1.1(b)) is similar to the H-frame in its use of a parallel-axis architecture and stationary motors, which are located at the base of its frame. It uses three stationary motors to control three vertical carriages which are arranged in the geometry of an equilateral triangle. Three pairs of forearms are used to connect each carriage to the same end-effector, such that three spherical constraint equations relate the vertical positions of the three carriages to the Cartesian position of the end-effector (as described in Section III). Like the H-frame, this architecture enables a lighter end-effector and improved dynamic performance. Despite the improvements in dynamic performance afforded by these manipulators, they still suffer from accuracy limitations caused

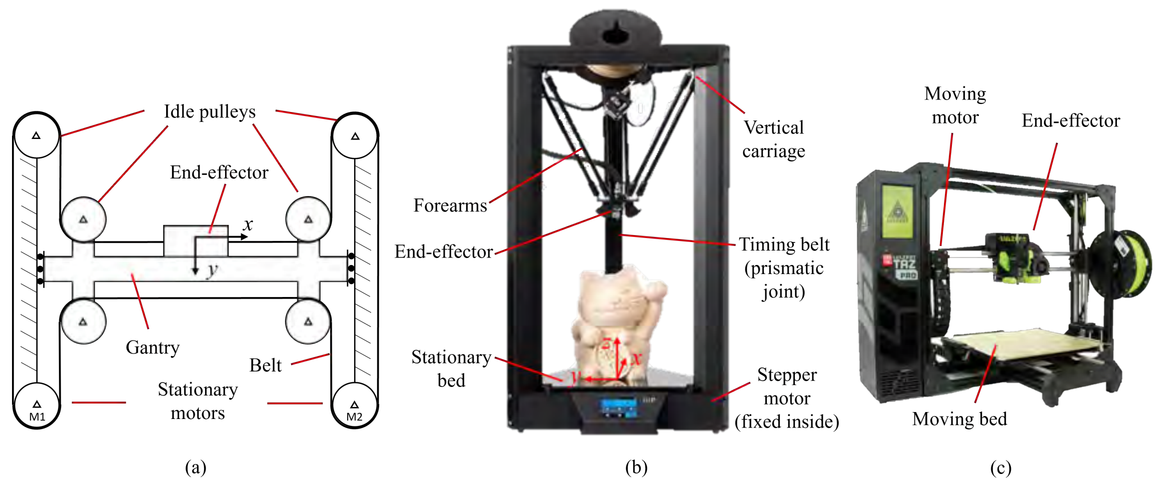


Figure 1.1: Three manipulator architectures for extrusion-based 3D printing: (a) the parallel-axis H-frame gantry with stationary motors (labeled M1 and M2) and the timing belt and pulley configuration that transmits rotational motion to XY translational motion of the gantry and end-effector; (b) the delta robot whereby stationary stepper motors actuate a vertical belt-driven carriage system connected to the end-effector via parallel links (labeled: forearms); and (c) a traditional serial-axis manipulator with an x-axis motor that moves with the z-axis of the machine, a bed that moves to control the y-axis, and a bulky end-effector. Architectures (a) and (b) have better dynamic performance than (c) due to their relatively low moving mass.

by motion-induced errors. These errors are created by vibration—via the inherent compliance in their mechanical structures—and the coupled and (often) nonlinear dynamics between their joints/axes, which create disturbance forces that are difficult to control. Additionally, as we will discuss in Chapters II and III, both manipulators have dynamics that vary as a function of their workspace position: the gantry on the H-frame undergoes torsional rotations whose print errors depend on the end-effector's x-position and the proportion of end-effector's inertia that is moved by each of the delta robot joints also depends on its global workspace position.

Vibration errors can be mitigated with hardware solutions that increase the stiffness or damping of the machine. However, this choice can increase the cost and weight of the machine. Active vibration suppression methods (i.e., feedback controllers) can also be used to mitigate vibration, but optimal performance of feedback controllers require adding sensors to the end-effector (i.e., printing nozzle) which can also increase cost. Given adequate sensing, feedback controllers may also encounter stability issues that are the result of non-collocated actuation and sensing between the motors and the end-effector. Additionally, most commercial 3D printers are controlled in open-loop architectures using stepper motors, which is perhaps the most significant impediment to using feedback control. Fortunately, the challenges of feedback control can be circumvented via model-based feedforward (FF) control, which has been shown to mitigate vibration in a host of manufacturing applications, including 3D printing [5–7, 9]. Model-based FF control is also attractive because it does not require modifying the mechanical architecture of machines which means they can remain low-cost and lightweight. As the name implies, model-based FF control techniques require an accurate model of the system's dynamics, which can be difficult or inefficient to obtain for systems with position-dependent dynamics like the H-frame and delta robot manipulators. Furthermore, even with accurate models of these machines, their position dependence means that the models must be recomputed for control as their position evolves. Additionally, the model-based FF controller may require solving computationally demanding optimization problems. Hence, the combination of updating the models and the controllers can be computationally challenging to achieve on-line during real-time control.

This thesis aims to address the challenges associated with efficiently modeling and applying model-based FF controllers to the H-frame and delta robot manipulators in the context of FFF 3D printing. By doing so, we can mitigate vibration and other motion-induced errors to maintain high accuracy while benefiting from the increased productivity provided by these systems.

1.2 Literature Review

1.2.1 Feedforward Vibration Control of Manufacturing Machines

Feedforward (FF) vibration control has been studied and implemented widely in research and industry to improve tracking and contouring performance in manufacturing machines. One of the most popular approaches to FF control is a class of methods known as smooth command generation (SCG) [10,11]. SCG reduces motion errors by generating commands that have little to no high-frequency content using, for example, low-pass filters [12] and jerk-limited trajectories [13]. However, attenuating high-frequency content implies loss of motion speed which adversely affects productivity [9]. Additionally, SCG methods focus solely on kinematic parameters, neglecting the machine's dynamics [14]. Hence, conservative acceleration and jerk limits are often adopted to minimize the risk of machine damage, leading to a sub-optimal solution. However, if we utilize knowledge of a machine's dynamics, we can optimize the trajectory to achieve a desired performance metric without risk of damaging the machine [15].

Input shaping (IS) is another popular FF control method that eliminates vibration errors [16–18] based on a model of the machine's dynamics. The basic idea for IS is to construct the input command as the convolution of the desired trajectory and a sequence of impulses aimed at mitigating residual vibration in the output motion of the system. The impulses are designed to be equal in magnitude but opposite in phase to the resonance vibrations in the system, which cancels the vibration errors through destructive interference. A major limitation of IS is that it introduces time delays between the desired and actual motion, leading to large tracking/contouring errors and reduction of productivity [19, 20]. Additionally, for parallel and robotic manipulators like the ones considered in this thesis, implementing IS in the joint space can lead to significant Cartesian contour errors [21,22]. Therefore, while IS works well for point-to-point motions, it exhibits poor performance for the tracking/contouring motions prevalent in most manufacturing applications [20].

Another class of FF control methods is known as model-inversion based FF control. Methods in this class compensate motion errors by using the inverse of the motion system's dynamics to pre-filter motion commands. Model-inversion FF control methods do not introduce time delays and can theoretically lead to perfect compensation of motion errors [23]. In practice, perfect compensation is difficult to achieve due to modeling errors [6] and the prevalence of nonminimum phase (or unstable) zeros, which become unstable poles after inversion [24]. Hence, approximate model-inversion FF controllers have been employed in the literature, several of which are discussed extensively in [23, 25, 26]. Of the available methods, the filtered basis function (FBF) approach has been shown to be versatile, compared to others, regarding its applicability to any linear system dynamics [5, 6, 20, 24]. 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 optimal coefficients of the basis functions such that motion errors are minimized. A version of FBF commonly used for controlling manufacturing machines is the FBS method [5], where B-splines are selected as the basis functions because they are amenable to the lengthy motion trajectories common in manufacturing. In prior work, the FBS method was used to reduce the printing time of traditional (serial) 3D printers by up to 54% without sacrificing print quality by compensating vibration-induced errors [9].

The versatility of FBS makes it a good candidate for compensating motion-induced errors on the H-frame and delta robot manipulators. However, the standard implementation of FBS used in prior work was not directly applicable to the compensation of errors in these systems because it assumes that the machine dynamics are LTI and decoupled. As previously discussed, the manipulators considered in this dissertation have nonlinear, treated as linear parameter-varying (LPV), and coupled dynamics.

Hence, the first contribution of this dissertation is the introduction of practical techniques to implement FBS on LPV systems with coupled dynamics.

1.2.2 Modeling and Control of H-frame and Delta Robot Manipulators

The H-frame architecture, as seen in Fig. 1.1(a), has a simple parallel axis design that consists of two motors which are connected to the end-effector through a single timing belt and several pulleys [27, 28]. Translational motion in the x- and y-axis of the end-effector is generated via the rotational motion of the frame-mounted motors. Since the motors are stationary, high-power (and, typically, heavy) motors can be used to achieve high-speed and high-precision motion without increasing the moving mass of the printer. For this reason, the H-frame architecture has been used in several 3D printers, such as the Stratasys Mojo [29], MakerBot Replicator Z18 [30], Creality Ender 4 [31], and MIT's Fast FFF [32]. Using the H-frame as one of several improvements to conventional FFF 3D printers, Go et al. [32] demonstrated a 5–10 times increase in build rate using the Fast FFF printer compared to several commercial printers of the same class. However, parts printed with H-frame 3D printers suffer from quality defects caused by parasitic error motions due to "racking" [28]: when the motors are commanded to rotate in the same direction, a force couple (pure moment) is imposed on the gantry (see Fig. 1.2) which can lead to errors that distort the part shape[1].

As discussed in Section 1.1, motion errors like racking can be mitigated with mechanical solutions such as a rigid linear guideway design [33] or adding counterweights to offset the racking, both of which increase cost and weight of the gantry. A lower-cost strategy that does not add weight to the gantry is to design a modified configuration like the two-belt Core XY architecture [28], which ensures the forces on

[1]An example of this phenomenon can be seen in the video from the YouTube account by K. Kamal entitled "Hbot Mechanism Racking Issue- 3D Printing": https://www.youtube.com/watch?v=2_wWr66bl6Q

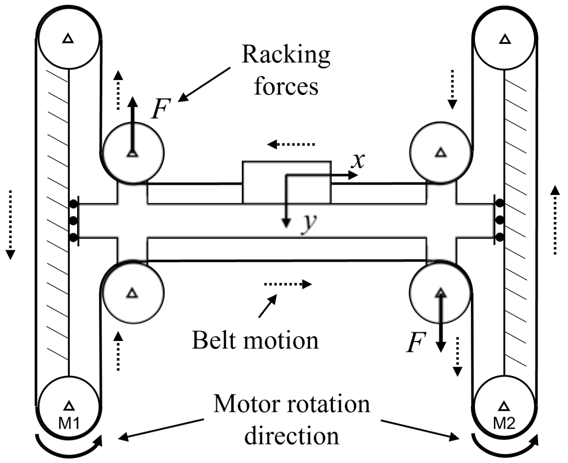


Figure 1.2: Rotations of the H-frame's motors in the same direction create unbalanced forces F that cause racking—parasitic torsional motions–of the gantry.

the gantry do not create a force couple. However, the Core XY and similar designs can be significantly more complex than the H-frame's design, which leads to a longer manufacturing process. Additionally, other sources of error may surface, such as the errors created on the Core XY when the two belts are not equally tensioned [28]. To avoid additional cost, weight, and complexity, we can employ model-based control to the H-frame manipulator to mitigate racking vibration.

In pursuit of a suitable model for control, Sollmann et al. [27] developed an eight-order lumped parameter dynamic model of a servo-motor controlled H-frame gantry using the Lagrange formulation. The model captures the nonlinear friction and the uncertainty of the end-effector position due to the stretching in the belt. However, the model does not capture the racking motion which results in "differences in the amplitude of starting oscillations between [their] simulated and experimental response" [27]. They conclude that although these observations were not captured by the model, they were relatively small (sub-millimeter) compared to other effects (e.g., nonlinear friction) [27]. While this conclusion may hold for some H-frame gantries, it does not generally apply for all, especially when they are expected to travel at high speeds and respond to rapidly changing acceleration inputs. In those cases, racking errors can have significant effects on the output [28]. Accordingly, in a thesis covering more details of the work [34], Sollmann found that proportional-derivative (PD) and proportional-integral-derivative (PID) controllers were inadequate control strategies for high-speed tracking, which underscores the necessity for model-based controllers that can compensate racking errors. In subsequent work, other researchers measured, characterized, and simulated the effects of racking motion on kinematic and dynamic accuracy [28, 35]. However, to the best of our knowledge, no one has explicitly attempted to compensate racking on an H-frame machine through model-based control. Hence, two additional contributions of this dissertation are to (1) propose an H-frame model that captures the racking dynamics and (2) propose a technique to compensate racking on 3D printers with the H-frame design.

The second manipulator of interest in this dissertation is the delta robot, which has been used in a wide variety of research and industrial applications including agriculture, healthcare, haptic devices, and manufacturing [36]. Delta robot 3D printers use three actuators to move three prismatic joints, which are all connected to the end-effector in parallel via forearm links as shown in Fig. 1.3. Vertical motion of the joints results in lateral and vertical motion of the nozzle as it extrudes and deposits material on a heated, stationary bed. As a result of the parallel-axis construction, the delta 3D printer boasts higher speeds and accelerations when compared to conventional 3D printers with serial kinematics [37]. Furthermore, delta 3D printers can command identical speeds in all three Cartesian axes, whereas the axis speed of serial 3D printers vary–with the vertical (z) axis typically having much lower speeds than the lateral (x, y) axes [38]. Accordingly, delta printers have expanded the capabilities of FFF and, for example, have been shown to improve the quality of Curved Layer FFF [39], which varies the z-axis position within layers. Examples of commercial delta 3D printers include the Monoprice (MP) Delta Pro [40], the FLSUN QQ-S Pro [41], the Delta WASP 2040 [42], and the Tractus T3500 [43].

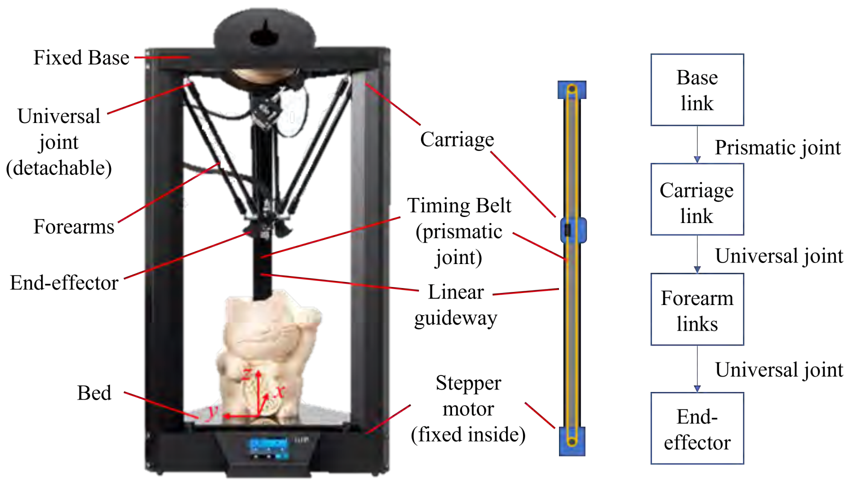


Figure 1.3: From left to right: a commercial delta 3D printer (Monoprice Delta Pro) with labeled components; a schematic of the belt-driven carriage system; and the delta robot configuration showing the connections between joints and links.

Like conventional serial-axis 3D printers, delta 3D printers experience undesirable vibration when they travel at high speeds [38, 44]. However, unlike conventional printers, delta printers have position-dependent and coupled multi-input, multi-output (MIMO) dynamics [45,46]. Thus, identifying their dynamic model can be difficult and time-consuming because one must measure the models at several different positions. Analytical models of delta robots can be derived but they may be: (a) too complex to be suitable for real-time control [46–48] or (b) contain a large number of parameters that are difficult to measure and identify accurately [45]. Hence, simplifications of analytical models have been proposed and used in model-based controllers [49–53]. However, most of the previous work considers delta robots with rotary joints (instead of the prismatic joints used in delta 3D printers), which typically use servo motors with encoders. As a result, these control schemes are usually aided by feedback regulators, most of which rely on state measurements to estimate servo errors for accurate compensation [45, 50–58]. A PD or PID controller is usually a key element of these control methods, but standalone PD/PID controllers do not consider the dynamic coupling of delta robots. Therefore, their performance is adversely affected by the force disturbance inputs from other kinematic chains. To address this issue, Codourey [45] combined a lumped model of the delta robot with a PD regulator in a computed torque (CT) control implementation to improve the tracking error performance in pick-and-place tasks when compared to a standalone PD regulator. Similarly, Angel and Viola [54] proposed a fractional PID controller combined with a CT controller. However, CT controllers also need to have complete knowledge of the robot's dynamics which, as discussed above, are difficult to model efficiently and are sensitive to uncertainties and disturbance inputs. For example, in [45], torque is computed using workspace accelerations, which are in turn obtained by transforming second derivatives of the joint positions of the robot into workspace coordinates. These calculations can be problematic when there is noise, joint flexibility and other unmodeled errors, or inaccuracies in the measurements. Perhaps this explains why no experiments that implement the controller on hardware are presented in [54] (only simulations were used).

In an effort to improve the performance of feedback controllers, recent work has focused on techniques to (a) improve measurement accuracy of real-time servo errors and (b) reject disturbance forces using adaptive control [50,51,55–59]. Some examples include disturbance rejection in the feedback loop using linear disturbance observers [50, 51], changing the PD gains online as a function of servo error estimates [55], injecting inputs learned by an artificial neural network (ANN) to compensate errors that the PD controller does not reject [56, 59], and using synchronization control strategies to reject coupling disturbances in each actuator from the other actuators [57,58]. Other approaches focus on tuning trajectory-dependent PID controller gains offline to minimize errors along a desired path that is known a priori [52, 53]. These PID gains provide reasonable tracking performance along the trajectory but require a priori knowledge of the entire trajectory. Some researchers have also employed sliding mode controllers to improve tracking performance since they are relatively insensitive to the disturbance inputs [60–62]. In summary, the central theme of the prior work is to treat the position-dependent dynamic variations of delta robots as unmodeled disturbances which are suppressed using feedback control. However, most commercial delta 3D printers cannot benefit from such approaches because they utilize stepper motors for actuation, which have no feedback sensors. Hence, they must rely on models that accurately capture their dynamics without the need for real-time state measurements.

Considering the general problem of modeling position-dependent dynamics, researchers have sought to eliminate the need for inefficient and time-consuming measurements by combining sparse offline measurements with techniques like interpolation [63] and machine learning [64]. Voorhoeve et al. [63] measured LTI models of a flexible wafer stage at several frozen positions and then interpolated the mode shapes to obtain a model with continuous position dependence. However, their approach was implemented on a single flexible moving body and it is not clear how to translate the methodology to systems with multiple parallel moving bodies like the delta robot. Machine learning-based methods—mostly using ANNs—have also been employed to model and control delta robots [64–68]. ANNs are promising because of their ability to learn nonlinear behaviors and to save time in computationally challenging problems—characteristics that are useful for controlling delta 3D printers. Currently, the most promising direction is using ANNs to generate a model for parameter varying dynamics. For example, Liu and Altintas [64] trained a transfer learning model using two ANNs: the first using abundant data from the simulated dynamics of a computer numeric controlled (CNC) machine, and the second with significantly less measurements from the machine. The models were combined to fine-tune the simulation model with the “real-world” model. Despite the success of this technique, it requires the laborious process of tuning hyperparameters during training. Additionally, since machine learning models often lack physical interpretation, designing stable controllers for them can be challenging. In general, using ANNs to design low-level control laws can be problematic because it is difficult to guarantee their stability. Therefore, in practical applications, ANNs are typically used to generate adaptive controller gains [65–68] instead of using them in model-based controllers or to directly generate low-level control laws. Therefore, another contribution of this dissertation is to propose a framework that enables identification of accurate and physically-interpretive LPV (i.e., position-dependent) models of the prismatic-joint delta robot using a few measurements from the machine. Additionally, since the delta 3D printer does not have any feedback sensors, this dissertation proposes a methodology for open-loop feedforward control by using the identified LPV model and novel contributions of techniques to improve the computational efficiency of implementing the LPV model with the FBS approach.

In summary, this dissertation aims to address the following gaps identified through the literature survey: (a) the need to develop practical techniques to implement the FBS approach on coupled LPV systems; (b) the need for an accurate model of the racking dynamics of the H-frame manipulator; (c) the need for a vibration control technique that can compensate vibration induced by racking motions on the H-frame; (d) the need for an accurate model of the dynamics of the prismatic-joint delta robot that is not a function of disturbance forces and, thus, does not rely on on-line measurements; and (e) the need for a open-loop vibration compensation technique for delta robot 3D printers.

1.3 Dissertation Contributions and Outline

To address the challenges of modeling and FF vibration compensation of the H-frame and delta robot manipulators to achieve high-quality and high-throughput in extrusion-based additive manufacturing, this dissertation (based on publications [69–72]) makes the following contributions:

1. In Chapter II, we analytically and empirically model the kinematics and dynamics of the H-frame's racking motion. Using the model, we propose an extension of the standard FBS controller to compensate the H-frame's coupled LPV racking dynamics.

2. Also in Chapter II, we develop a simplification of the H-frame's coupled LPV FBS controller and show that the simplification significantly reduces the computational cost of the controller with little to no sacrifice to its compensation accuracy.

3. In Chapter III, we introduce the receptance coupling (RC) framework used to model the delta 3D printer dynamics. With the framework, we propose an efficient methodology for identifying a model with a few measurements at one location and show that the model yields accurate predictions of a delta printer's dynamics in arbitrary configurations.

4. In Chapter IV, we propose a set of techniques to address the computational challenges of controlling the position-dependent delta 3D printer using FBS. Our methodology includes (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 at every point) along the desired trajectory, and (3) employing matrix factorization techniques to reduce the number of floating-point arithmetic operations associated with matrix inversion.

5. Throughout the chapters, we demonstrate the effectiveness and practicality of the developed controllers in compensating the modeled errors through numerical simulations and 3D printing experiments on hardware.

We conclude the dissertation in Chapter V with a discussion of the key insights obtained over the course of this work and a summary of its implications for the future of additive manufacturing. Finally, we close with recommendations for future work.

CHAPTER II

Modeling and Control of the H-Frame Gantry

2.1 Overview

This chapter studies the H-frame 3D printer to apply FF vibration control to improve the shape accuracy of parts manufactured at high speeds. In the course of our study, we discuss its coupled kinematics and dynamics as well as the existence of parasitic racking motion, which must be modeled for optimal control. After modeling the racking motion, we propose an extension to the standard FBS controller (given in [5]) to compensate the coupled LPV H-frame dynamics. Then, we approximate the optimal controller with a simplification to improve computational efficiency without significantly degrading the overall accuracy. Numerical simulations and experiments on an H-frame 3D printer are reported to demonstrate the effectiveness of the proposed techniques.

This chapter is organized as follows: the kinematic and dynamic models of the H-frame racking motion are introduced and validated in Section 2.2. Section 2.3 gives an overview of the standard FBS method and Section 2.4 proposes a coupled LPV FBS method for compensating racking errors. Section 2.4 also demonstrates the increased computational cost of the proposed coupled LPV FBS controller, relative to the standard FBS controller, and proposes a simplification that reduces its computational cost with minimal sacrifice to its performance—hence facilitating its practicality. Section 2.5 presents simulations and experiments on the H-frame 3D printer that demonstrate the effectiveness of the proposed approach, followed by a summary of the chapter in Section 2.6.

2.2 Kinematics and Dynamics

2.2.1 Inverse and Forward Kinematics

The inverse kinematics equations are used to calculate the angular positions of the stationary motors of the H-frame as a function of the x- and y-axis positions. Note (from Figs. 1.1(a) and 1.2) that positive (counterclockwise) rotation of motor 1, denoted by the angle ϕ_1, while keeping the other motor fixed, results in 45° linear motion in the negative x-axis direction and negative y-axis direction (and vice versa for a clockwise rotation). Therefore, we have

$$r\phi_1 = -x - y, \tag{2.1}$$

where r is the radius of the motor pulley, x is the linear position of the end-effector in the x-axis, and y is the linear position of the end-effector in the y-axis. Here, we impose a coordinate axis on the H-frame's workspace so that the reference angular position of the motor is set as the position that corresponds to the end-effector being at the origin of the xy-coordinate plane. Similarly, positive rotation of motor 2, denoted by the angle ϕ_2, results in 45° linear motion in the negative x direction and positive y direction such that

$$r\phi_2 = -x + y. \tag{2.2}$$

Combining Eqs. (2.1) and (2.2) gives the inverse kinematic equations

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{r} & -\frac{1}{r} \\ -\frac{1}{r} & \frac{1}{r} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.3}$$

We can solve the inverse problem to derive the forward kinematics as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}r & -\frac{1}{2}r \\ -\frac{1}{2}r & \frac{1}{2}r \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}, \tag{2.4}$$

which is used to determine the x- and y-axis positions given the angular positions of the motors. Generally, in 3D printing, we are given the Cartesian axis positions from a G-code file and use inverse kinematics equations to determine the motor angles.

2.2.2 Dynamic Model of Parasitic Racking Motion of H-frame

The racking motion of the gantry on the H-frame is caused by a force couple (pure moment) which creates an angular displacement in the rotational axis, θ, on the gantry. Let $\{x,\ y\}$ be the end-effector's output position in the x- and y-axis, respectively. It can be decomposed into two portions: $\{x',\ y'\}$, where x' is the x-axis output position *without* racking errors accounted for, and y' is the y-axis output position of the end-effector *without* the racking errors; and $\{\Delta x,\ \Delta y\}$, which are the errors created by racking angle as shown in Fig. 2.1(a). Therefore, $x = x' - \Delta x$ and $y = y' + \Delta y$. Since the racking angles are small, we can use the small angle approximation (i.e., $\cos\theta \approx 1$, $\sin\theta \approx \theta$) to conclude that $x' = x\cos\theta \approx x$ (i.e., $\Delta x \approx 0$) and the x- and y-axis positions are given by:

$$x = x'\cos\theta \approx x' \tag{2.5}$$

$$y = y' + x\sin\theta \approx y' + x\theta \tag{2.6}$$

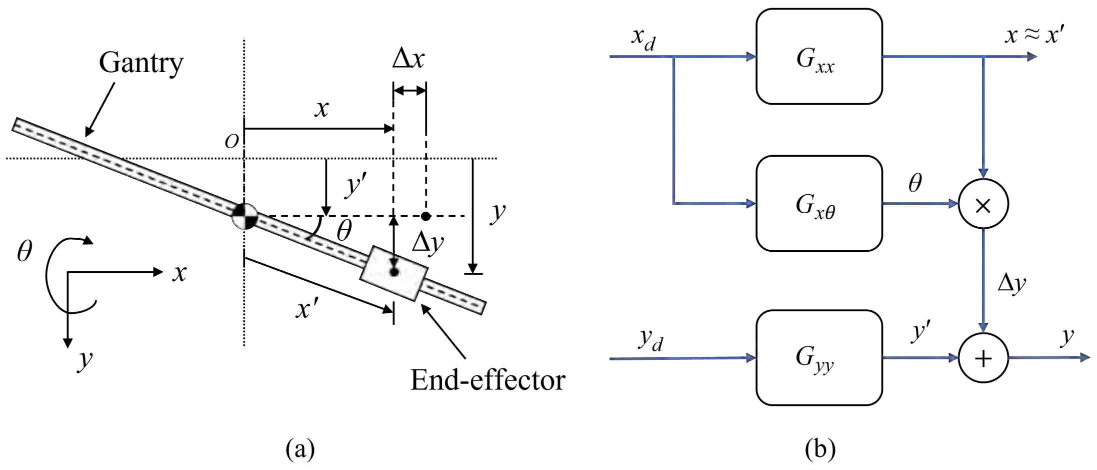


Figure 2.1: (a) Kinematic model of parasitic racking errors of H-frame 3D printer; (b) dynamic model of H-frame 3D printer including effect of y-axis errors (Δy) caused by racking.

This kinematic model can be used to create a coupled dynamic model including racking, as shown in the block diagram in Fig. 2.1(b), where $\{x_d,\ y_d\}$ represent the desired position of the end-effector. The transfer functions from x_d to x, and y_d to y' are represented by G_{xx} and G_{yy}, respectively, and $G_{x\theta}$ represents the racking contribution (i.e., the transfer function from x_d to θ).

In addition to the small angle approximation, two other assumptions are implied in the model of Fig. 2.1(b). The first is that the H-frame dynamics can be approximated as linear. Therefore, x, y', and θ can be determined from transfer functions G_{xx}, G_{yy}, $G_{x\theta}$, respectively. This assumption has been found to be reasonable in prior work [5–7, 9, 27, 34, 73]. The second assumption is that the transfer function $G_{x\theta}$ does not vary as a function of end-effector position which implies that the location of the center of mass of the gantry does not vary with the position of the end-effector. This assumption will be validated later in this section. Finally, note that the model shown in Fig. 2.1(b) is nonlinear because $\Delta y = x\theta$ is the product of two output states. It can be approximated as LPV by assuming that $x \approx x_d$ when determining Δy. This assumption is reasonable because the tracking errors, $e_x = x_d - x$, caused by G_{xx} are typically much smaller than the magnitude of x. Therefore, they have insignificant contributions to Δy. Accordingly, $\Delta y = x_d\theta$ is assumed in the rest of this section for the sake of simplicity, resulting in a coupled LPV model for H-frame 3D printers.

The 3D printer shown in Fig. 2.2(a) is used to validate the H-frame model of Fig. 2.1. It is fabricated by adapting a Creality Ender 5 3D printer into an H-frame machine. The designed H-frame configuration[1] is actuated by two NEMA 17 stepper motors via a 2-mm pitch, 6-mm wide, rubber timing belt used for motion transmission. The motors are controlled by Pololu DRV8825 high-current stepper motor drivers configured to give a step resolution of 2 milli-radians per step which is transmitted through a pulley with radius $r = 5.15$ mm to give a x- and y-axis resolution of 20.6 μm per step. The motion range of the x and y axes are 280 and 295 mm, respectively. Real-time control of the x, y, z axes, and extrusion motors is performed using dSPACE MicroLabBox (RTI 1202) with stepping frequency of 40 kHz and sampling frequency of 1 kHz. Commands to the printer are generated in MATLAB and sent to the MicroLabBox through a MATLAB Simulink interface.

To validate the model of Fig. 2.1, we commanded sine sweep perturbations across a range of frequencies in the x direction of the printer by applying acceleration commands $\ddot{x}_d$ to the stepper motors and measuring y-axis accelerations at the locations marked P_1 and P_2 in Fig. 2.2(b) using two ADXL335 three-axis accelerometers. The racking angular acceleration is estimated (based on small angle rotations) as

$$\ddot{\theta} = \frac{\ddot{y}_1 - \ddot{y}_2}{L_G}, \tag{2.7}$$

where $\ddot{y}_1$ and $\ddot{y}_2$ are the y-axis accelerations measured at P_1 and P_2 at either end of the bridge and L_G is the perpendicular distance between P_1 and P_2 (Fig. 2.2(b)). Accordingly, $G_{x\theta}$ is computed using $\ddot{x}_d$ as input and $\ddot{\theta}$ as output. Figure 2.3 shows $G_{x\theta}$ measured with the end-effector positioned at $x = 0, \pm 30$, and ± 60 mm. The

[1]Ender 5 Modified to H-Bot (Ender 4), Thingiverse (2020), https://www.thingiverse.com/thing:4425748

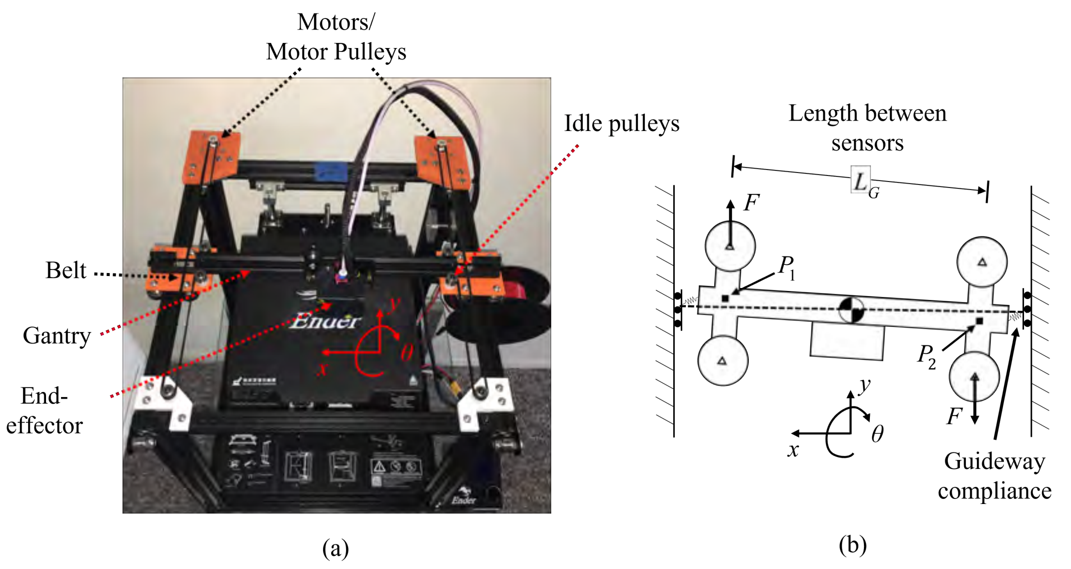


Figure 2.2: (a) Designed H-frame 3D printer (retrofitted from the Creality Ender 5 3D printer) used to validate H-frame dynamic model and conduct experiments in Section 2.5; (b) schematic of dynamic identification of racking motion caused by force couple F and guideway compliance (denoted by spring). Acceleration measurements at P_1 and P_2 are used to generate the racking model.

discrepancy between the FRFs is small, supporting the assumption that the end-effector position does not significantly influence $G_{x\theta}$. Similarly, G_{xx} and G_{yy} (Fig. 2.4) are determined by commanding acceleration perturbations in the x and y directions, respectively, as inputs and and measuring the end-effector's acceleration in the x and y directions as the output—with the gantry positioned at $x = 0$ mm.

By curve-fitting the FRFs for $G_{x\theta}$, G_{xx}, and G_{yy}, discrete transfer functions for each FRF of the form

$$\hat{G}(z) = \frac{b_q z^q + b_{q-1} z^{q-1} + \cdots + b_1 z + b_0}{z^d + a_{d-1} z^{d-1} + \cdots + a_1 z + a_0} \tag{2.8}$$

are obtained, where the $\hat{\cdot}$ accent denotes an estimated model of the actual dynamics, z is the discrete-time forward shift operator, q and d are the degrees of numerator and denominator polynomials, respectively, and the coefficients of each polynomial are given in Tables 2.1 and 2.2. Note that the FRF measured with x at 0 mm is used

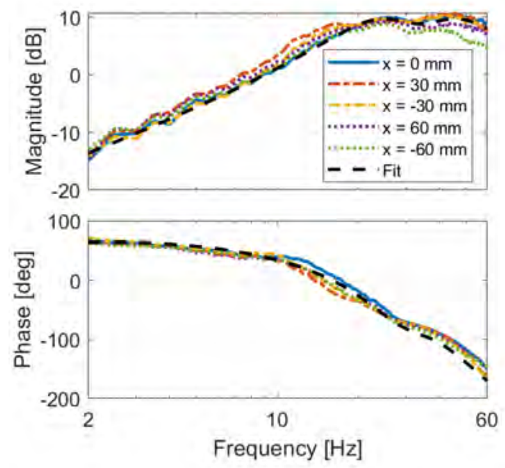


Figure 2.3: Frequency response functions from x-axis command input to θ output ($G_{x\theta}$) measured with the end-effector positioned at $x = 0$, ± 30, and ± 60 mm. The differences between the FRFs are small. Therefore, they are modeled by a single FRF shown with the black dashed lines.

Table 2.1: Numerators of fitted transfer functions in Eq. (2.8).

	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
$\hat{G}_{xx}(z)$	–	–	–	–	0.08	-0.23	0.24	-0.08	-2.19$\times 10^{-13}$
$\hat{G}_{x\theta}(z)$	4.25	-29.3	87.5	-147.0	149.9	-92.8	32.3	-4.87	1.80$\times 10^{-15}$
$\hat{G}_{yy}(z)$	–	–	0.16	-0.89	1.98	-2.2	1.23	-0.28	-9.17$\times 10^{-13}$

to fit $G_{x\theta}$.

2.3 Standard Filtered B-Splines (FBS) Approach

Figure 2.5 shows the block diagram of the standard filtered B-splines (FBS) approach for the x-axis of a decoupled multi-axis system, i.e., without racking compensation, as introduced in [24]. It controls an LTI discrete-time system given by $\mathbf{G}_{xx}$, the lifted system (or matrix) representation of transfer function G_{xx}, through a feedforward controller $\mathbf{C}_x$ (see Appendix A for details on the lifted system representation).

Let $\mathbf{x}_d = [x_d(0)\ x_d(1)\ \cdots\ x_d(E)]^T$ represent $E + 1$ discrete time steps of the

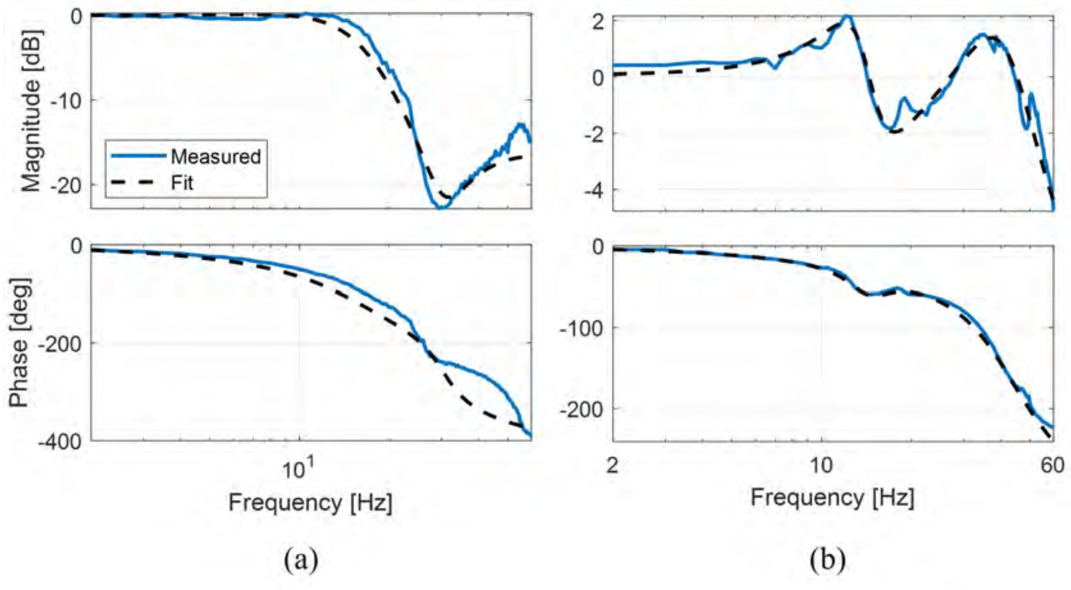
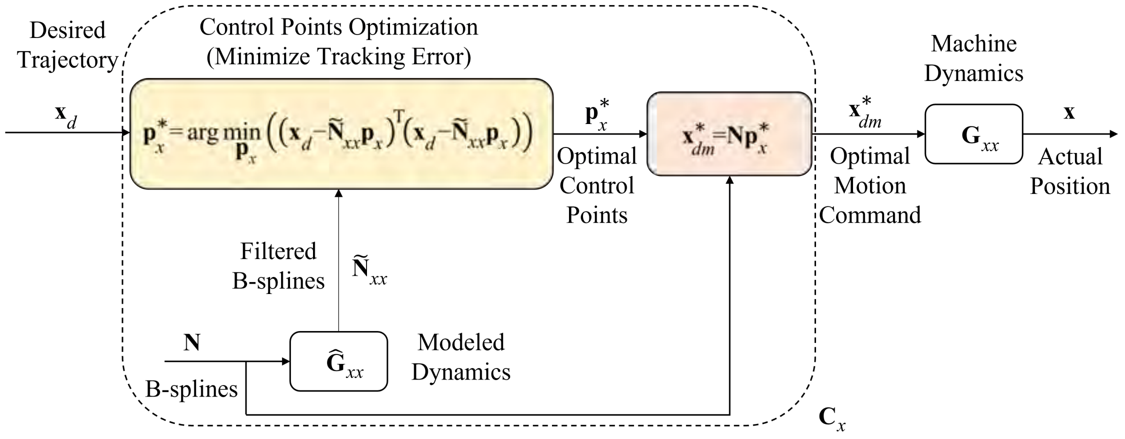


Figure 2.4: Measured and curve fit FRFs for (a) G_{xx} and (b) G_{yy}.

Table 2.2: Denominators of fitted transfer functions in Eq. (2.8).

	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
$\hat{G}_{xx}(z)$	–	–	–	–	-3.58	4.77	-2.82	0.62	1.93×10^{-33}
$\hat{G}_{x\theta}(z)$	-5.87	14.9	-21.0	17.8	-9.06	2.54	-0.30	-2.13×10^{-17}	2.40×10^{-34}
$\hat{G}_{yy}(z)$	–	–	-4.81	9.39	-9.34	4.85	-1.14	0.07	-9.04×10^{-19}



Desired Trajectory
$\mathbf{x}_d$
Control Points Optimization (Minimize Tracking Error)
$\mathbf{p}_x^* = \arg\min_{\mathbf{p}_x}\left((\mathbf{x}_d-\tilde{\mathbf{N}}_{xx}\mathbf{p}_x)^{\mathrm{T}}(\mathbf{x}_d-\tilde{\mathbf{N}}_{xx}\mathbf{p}_x)\right)$
$\mathbf{p}_x^*$
Optimal Control Points
$\mathbf{x}_{dm}^* = \mathbf{N}\mathbf{p}_x^*$
$\mathbf{x}_{dm}^*$
Optimal Motion Command
Machine Dynamics
$\mathbf{G}_{xx}$
$\mathbf{x}$
Actual Position
Filtered B-splines
$\tilde{\mathbf{N}}_{xx}$
$\mathbf{N}$
B-splines
$\hat{\mathbf{G}}_{xx}$
Modeled Dynamics
$\mathbf{C}_x$

Figure 2.5: Block diagram of standard FBS method applied to x-axis of a decoupled multi-axis system.

x-component of the desired trajectory of a multi-axis machine. Assume that the machine has look-ahead capabilities such that the $E+1$ steps of $\mathbf{x}_d$ are known in advance. Furthermore, assume that the modified but un-optimized motion command $\mathbf{x}_{dm} = [x_{dm}(0)\ \ x_{dm}(1)\ \ \cdots\ \ x_{dm}(E)]^T$ is parameterized using B-splines such that

$$\begin{bmatrix} x_{dm}(0) \\ x_{dm}(1) \\ \vdots \\ x_{dm}(E) \end{bmatrix} = \underbrace{\begin{bmatrix} N_{0,m}(\xi_0) & N_{1,m}(\xi_0) & \cdots & N_{n,m}(\xi_0) \\ N_{0,m}(\xi_1) & N_{1,m}(\xi_1) & \cdots & N_{n,m}(\xi_1) \\ \vdots & \vdots & \ddots & \vdots \\ N_{0,m}(\xi_E) & N_{1,m}(\xi_E) & \cdots & N_{n,m}(\xi_E) \end{bmatrix}}_{\mathbf{N}} \underbrace{\begin{bmatrix} p_{x,0} \\ p_{x,1} \\ \vdots \\ p_{x,n} \end{bmatrix}}_{\mathbf{p}_x} \tag{2.9}$$

where $\mathbf{N}$ is the matrix representation of B-spline basis functions of degree m, $\mathbf{p}_x$ is a vector of $n+1$ unknown coefficients (or control points), $j = 0, 1, ..., n$, and $\xi \in [0, 1]$ is the spline parameter, representing normalized time, which is discretized to $E+1$ uniformly spaced points $\xi_0, \xi_1, ..., \xi_E$. The real-valued basis functions, $N_{j,m}(\xi)$, are given by [74]

$$N_{j,m}(\xi) = \frac{\xi - g_j}{g_{j+m} - g_j} N_{j,m-1}(\xi) + \frac{g_{j+m+1} - \xi}{g_{j+m+1} - g_{j+1}} N_{j+1,m-1}(\xi) \tag{2.10}$$

$$N_{j,0} = \begin{cases} 1, & g_j \le \xi \le g_{j+1} \\ 0, & \text{otherwise} \end{cases}$$

where $\mathbf{g} = [g_0\ \ g_1\ \ \cdots\ \ g_{m+n+1}]^T$ is a normalized knot vector defined over $[0, 1]$. For convenience, $\mathbf{g}$ is assumed to be uniformly spaced, i.e.,

$$g_j = \begin{cases} 0, & 0 \le j \le m \\ \frac{j-m}{n-m+1}, & m+1 \le j \le n \\ 1, & n+1 \le j \le m+n+1 \end{cases} \tag{2.11}$$

Let $\mathbf{x}$ represent the $E+1$ discrete steps of x, the output motion of the machine. Accordingly, based on the definition of $\mathbf{x}_{dm}$ in Eq. (2.9), $\mathbf{x}$ can be written as

$$\mathbf{x} = \tilde{\mathbf{N}}_{xx}\mathbf{p}_x \tag{2.12}$$

where $\tilde{\mathbf{N}}_{xx}$ is the filtered B-splines matrix, acquired by filtering the columns of $\mathbf{N}$ through $\hat{G}_{xx}(z)$ (i.e., the matrix product of $\hat{\mathbf{G}}_{xx}$ and $\mathbf{N}$). The tracking error is defined as

$$\mathbf{e}_x = \mathbf{x}_d - \mathbf{x} = \mathbf{x}_d - \tilde{\mathbf{N}}_{xx}\mathbf{p}_x \tag{2.13}$$

The optimal control points $\mathbf{p}_x^*$ are calculated by minimizing the square of the L_2-norm of the tracking error

$$\mathbf{p}_x^* = \arg\min_{\mathbf{p}_x}(\mathbf{e}_x^T\mathbf{e}_x) \tag{2.14}$$

$$= \arg\min_{\mathbf{p}_x}\left((\mathbf{x}_d - \tilde{\mathbf{N}}_{xx}\mathbf{p}_x)^T(\mathbf{x}_d - \tilde{\mathbf{N}}_{xx}\mathbf{p}_x)\right) \tag{2.15}$$

giving the well-known least squares solution

$$\mathbf{p}_x^* = \left(\tilde{\mathbf{N}}_{xx}^T\tilde{\mathbf{N}}_{xx}\right)^{-1}\tilde{\mathbf{N}}_{xx}^T\mathbf{x}_d = \tilde{\mathbf{N}}_{xx}^{\dagger}\mathbf{x}_d \tag{2.16}$$

where the $\dagger$ in the superscript represents the Moore-Penrose inverse (or pseudoinverse) of the matrix. The result can then be used to calculate the optimized motion command $\mathbf{x}_{dm}^* = \mathbf{N}\mathbf{p}_x^*$. The same procedure is followed to find the optimal control input for other axes (e.g., the y-axis).

Remark 2.1: The LPFBS method [5] relaxes the assumption that $\mathbf{x}_d$ is known in advance and instead uses small windows (batches) of $\mathbf{x}_d$ to achieve on-line control. A brief overview of LPFBS is included in Appendix B.

The LTI implementation of FBS discussed above has two issues dealing with the introduction of racking in the H-frame. The first is that the motion of the x-axis affects the y-axis due to racking. Therefore, the y-axis cannot be controlled independent of the x-axis. The second issue is that control of the y-axis depends on the position of the x-axis. Therefore, a coupled LPV FBS approach is needed to include racking dynamics to compensate the motion errors on the H-frame 3D printer.

2.4 FBS with Racking Compensation

Recall that the racking model from Section 2.2 can be used to predict the error Δy from Eqs. (2.5) and (2.6). Hence, we can use the product of $\mathbf{G}_{x\theta}$ and the B-splines matrix $\mathbf{N}$ to obtain $\tilde{\mathbf{N}}_{x\theta}$. Therefore, using Eqs. (2.9) and (2.12), we have

$$\boldsymbol{\Theta} = \tilde{\mathbf{N}}_{x\theta}\mathbf{p}_x \tag{2.17}$$

and

$$\Delta\mathbf{y} = \mathbf{D}_{x_d}\tilde{\mathbf{N}}_{x\theta}\mathbf{p}_x \tag{2.18}$$

where $\mathbf{D}_{x_d} = \text{diag}(\mathbf{x}_d)$. The tracking error for each axis can then be expressed as

$$\mathbf{e}_x = \mathbf{x}_d - \mathbf{x} = \mathbf{x}_d - \tilde{\mathbf{N}}_{xx}\mathbf{p}_x \tag{2.19}$$

$$\mathbf{e}_y = \mathbf{y}_d - \mathbf{y} = \mathbf{y}_d - (\tilde{\mathbf{N}}_{yy}\mathbf{p}_y + \mathbf{D}_{x_d}\tilde{\mathbf{N}}_{x\theta}\mathbf{p}_x) \tag{2.20}$$

and the optimal control points can be calculated to minimize the squared L_2-norm of the tracking errors as

$$\mathbf{p}^* = \arg\min_{\mathbf{p}}(\mathbf{e}^T\mathbf{e}) \tag{2.21}$$

where $\mathbf{e} = [\mathbf{e}_x \;\; \mathbf{e}_y]^T$, which gives

$$\mathbf{p}^* = \begin{bmatrix} \mathbf{p}_x^* \\ \mathbf{p}_y^* \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{N}}_{xx} & \mathbf{0} \\ \mathbf{D}_{x_d}\tilde{\mathbf{N}}_{x\theta} & \tilde{\mathbf{N}}_{yy} \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{x}_d \\ \mathbf{y}_d \end{bmatrix}. \tag{2.22}$$

The formulation of the coupled LPV FBS controller in Eq. (2.22) can be cumbersome to compute during on-line implementation due to the size of the matrix that needs to be inverted. Furthermore, since the H-frame is an LPV system, we cannot pre-invert the matrix when using LPFBS as is done with LTI FBS to reduce the computational load (see Appendix B). Since the x-axis is independent of the y-axis, we can approximate the solution by decoupling the matrices to eliminate the need to compute the pseudoinverse of large matrices during implementation. We can instead compute the control points sequentially: first $\mathbf{p}_x^*$ using (2.16), then $\mathbf{p}_y^*$ considering $\mathbf{p}_x^*$ as a known input

$$\mathbf{p}_y^* = \tilde{\mathbf{N}}_{yy}^{\dagger}\Big(\mathbf{y}_d - \mathbf{D}_{x_d}\tilde{\mathbf{N}}_{x\theta}\mathbf{p}_x^*\Big). \tag{2.23}$$

Note that in the decoupled approximation (using Eqs. (2.16) and (2.23)), we are inverting the same matrices from LTI FBS. Therefore, we can pre-invert the matrices for on-line implementation. Next (and in Section 2.5), we consider the effects of this decoupled approximation on the tracking accuracy and computational complexity of the proposed controller.

Remark 2.2: The LPFBS formulation of the proposed decoupled LPV FBS controller with racking compensation is discussed in Appendix C.

The system with racking can be expressed in lifted system representation as

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{xx} & \mathbf{0} \\ \mathbf{G}_{xy} & \mathbf{G}_{yy} \end{bmatrix} \tag{2.24}$$

where $\mathbf{G}_{xy} = \mathbf{D}_{x_d}\mathbf{G}_{x\theta}$. The inverse of $\mathbf{G}$ is given by

$$\mathbf{G}^{-1} = \begin{bmatrix} \mathbf{G}_{xx}^{-1} & \mathbf{0} \\ -\mathbf{G}_{yy}^{-1}\mathbf{G}_{xy}\mathbf{G}_{xx}^{-1} & \mathbf{G}_{yy}^{-1} \end{bmatrix} \tag{2.25}$$

and therefore the optimal control inputs $\mathbf{x}_{dm}^*$ and $\mathbf{y}_{dm}^*$ are given by

$$\mathbf{x}_{dm}^* = \mathbf{G}_{xx}^{-1}\mathbf{x}_d \tag{2.26}$$

$$\mathbf{y}_{dm}^* = -\mathbf{G}_{yy}^{-1}\mathbf{G}_{xy}\mathbf{G}_{xx}^{-1}\mathbf{x}_d + \mathbf{G}_{yy}^{-1}\mathbf{y}_d \tag{2.27}$$

It can be shown (see [24]) that for $n = E$, $\tilde{\mathbf{N}}_{xx} = \mathbf{G}_{xx}$, $\tilde{\mathbf{N}}_{yy} = \mathbf{G}_{yy}$, and the pseudoinversion of $\tilde{\mathbf{N}}_{xx}$ and $\tilde{\mathbf{N}}_{yy}$ become the matrix inversion of $\mathbf{G}_{xx}$ and $\mathbf{G}_{yy}$ respectively. Therefore, for $n = E$, the motion command for the decoupled approximation is identical to Eqs. (2.26) and (2.27), which shows that that the decoupled LPV FBS approach is exactly the same as the inversion of the coupled LPV system when $n = E$. When $n < E$, the decoupled approach approximates the coupled LPV system. Therefore, the decoupled approach (Eqs. (2.16) and (2.23)) can be considered as another way of approximating the optimal LPV controller when compared to the coupled implementation (Eq. (2.22)).

The computational complexity of the Moore-Penrose inverse, computed using singular value decomposition, is given by $O(lv^2)$ where l and v are the number of rows and columns, respectively, of the matrix to be inverted [75]. We note that the size of the coupled LPV FBS matrix is $2(E+1)\times(n_x+n_y+2)$, where we consider the number of basis functions in the x and y axes independently, and the size of the decoupled matrices are $(E+1) \times (n_x+1)$ and $(E+1) \times (n_y+1)$. Assuming $n = n_x = n_y = E$, the computational complexity of the coupled and decoupled LPV FBS approaches are

$$O_c((2E)(2n)^2) = O_c((2n)^3) = O_c(8n^3) \tag{2.28}$$

$$O_d(En^2 + En^2) = O_d(E(n^2 + n^2) = O_d(En^2) = O_d(n^3) \tag{2.29}$$

where O_c and O_d are the computational orders of the coupled and decoupled approximations, respectively. The expressions in Eqs. (2.28) and (2.29) indicate that the decoupled matrix approximation has lower computational complexity than the coupled one. The implications of using the decoupled or coupled implementations of the LPV dynamics on racking compensation accuracy and computation time are explored further via simulations in the following section.

2.5 Simulations and Experimental Validation

2.5.1 Simulations

Figure 2.6 shows a 120-by-20 mm rectangular path used for our simulations. It was selected to highlight the racking motion which are prevalent during changes in the x-axis acceleration when turning around corners. The trajectory was generated using a jerk-limited motion profile for the two cases presented in Table 2.3. The first is the baseline case which uses the standard (low) speed and acceleration employed on most desktop 3D printers to avoid excessive vibration: 60 mm/s and 1×10^3 mm/s^2, respectively, with a jerk limit of 5×10^7 mm/s^3. The second is a high-speed case, which uses 2.5 times and 10 times the standard speed and acceleration, respectively: 150 mm/s and 1×10^4 mm/s^2, with a jerk limit of 5×10^7 mm/s^3. The trajectory was sampled at $T_s = 1$ ms, leading to $E+1 = 4970$ and 1944 trajectory points for the baseline and high-speed cases, respectively. The dynamic model presented in Section

2.2 was used to simulate the time response of the H-frame in Matlab.

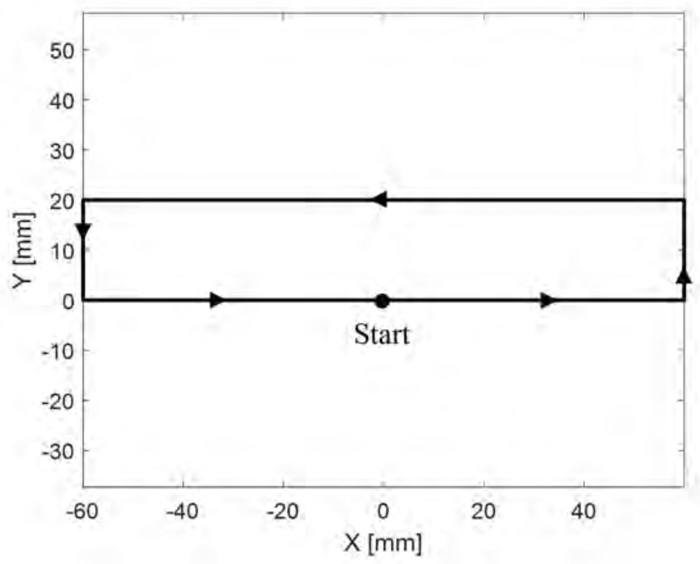


Figure 2.6: Rectangular path (with 120 mm length and 20 mm width) used to simulate the time response of the H-frame 3D Printer. The motion command starts at {0,0} and traverses the rectangle in the counterclockwise direction as indicated by the arrows.

	Baseline	High-speed
Speed	60 mm/s	150 mm/s
Acceleration	1×10^3 mm/s^2	1×10^4 mm/s^2
Jerk	5×10^7 mm/s^3	5×10^7 mm/s^3

Table 2.3: Summary of speed, acceleration, and jerk limits of the baseline and high-speed trajectories.

2.5.1.1 Comparison of decoupled and coupled FBS strategies for racking compensation

We begin our numerical analysis of the proposed racking compensation algorithm by comparing the tracking accuracy and computational complexity of its decoupled and coupled implementation strategies discussed in Section 2.4. The high-speed case is used for the comparison in this subsection since it is more aggressive and likely to induce racking errors. We compare the tracking accuracy for each strategy by

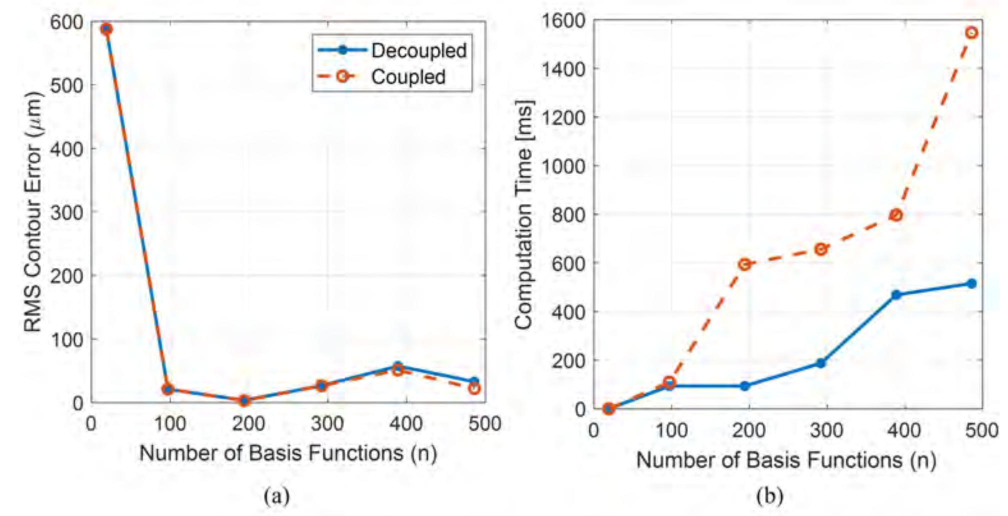


Figure 2.7: (a) RMS contour errors and (b) computation times for the simulated time response of the decoupled (solid line) and coupled (dashed line) LPV FBS H-frame controllers as a function of the number of basis functions used to parameterize the trajectory.

examining the RMS contour errors (i.e., path deviation) of the output trajectory. The number of basis functions was selected to span fractional values of E, namely $n = [0.01,\ 0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25]E$ (rounded to the nearest integer), the B-spline degree was identical for both implementation strategies ($m = 5$), and the knot vector was defined as in Section 2.3. Consistent with the analysis in Section 2.4, note that the RMS contour error between the two methods is similar for all n values (see Fig. 2.7(a)), indicating that using the decoupled approach yields similar accuracy performance to the coupled approach for a reasonable selection of n. We also validated the computational complexity analysis from Section 2.4 by comparing the computation time of the controller between the two approaches in Fig. 2.7(b). Note that the coupled approach requires significantly more computation time as n increases—approximately 3x more time in the worst case. Based on the analysis in Section 2.4 and the results shown in Fig. 2.7, the decoupled implementation of the coupled LPV FBS controller (i.e., FBS with racking compensation) will be used in all simulations and experiments for the remainder of this chapter.

2.5.1.2 Comparison of FBS controller with and without racking compensation

The simulated response of the H-frame machine using the baseline and high-speed cases, controlled with $n = 0.1E$ B-spline basis functions, are shown in Figs. 2.8 and 2.9, respectively. Racking errors can be seen at the corners and edges of the rectangle as the trajectory is traversed using the uncompensated case or FBS without racking compensation. Notice that the severity of the vibration and racking errors are less with the baseline case compared to the high-speed case. Hence, vibration and racking, when not compensated, limit the achievable speed of H-frame 3D printers if loss of print quality is unacceptable. Racking errors are reduced using the proposed FBS controller with racking compensation, which leads to an output that follows the desired path more accurately for both trajectories. Table 2.4 shows the RMS and maximum contour errors for both cases compared across the compensation approaches. Here, the maximum contour error is indicative of whether a part will reach its tolerance specification. Note that, as expected, the baseline has lower maximum contour errors when compared to the high-speed case. For both cases, the uncompensated approach has the highest maximum contour. Comparisons between FBS without racking compensation and FBS with racking compensation show a 99% and 94% reduction in maximum contour error for the baseline and high-speed cases, respectively.

2.5.2 Experiments

The same rectangular profile used in Section 2.5.1 was extruded to a height of 10 mm and printed on the H-frame 3D printer. The CAD model of the rectangular prism can be seen in Fig. 2.10. The G-code for the trajectory was generated using the open-source Ultimaker Cura® software. As with the simulations, two cases were considered in experiments: the baseline and high-speed cases for which the wall speed,

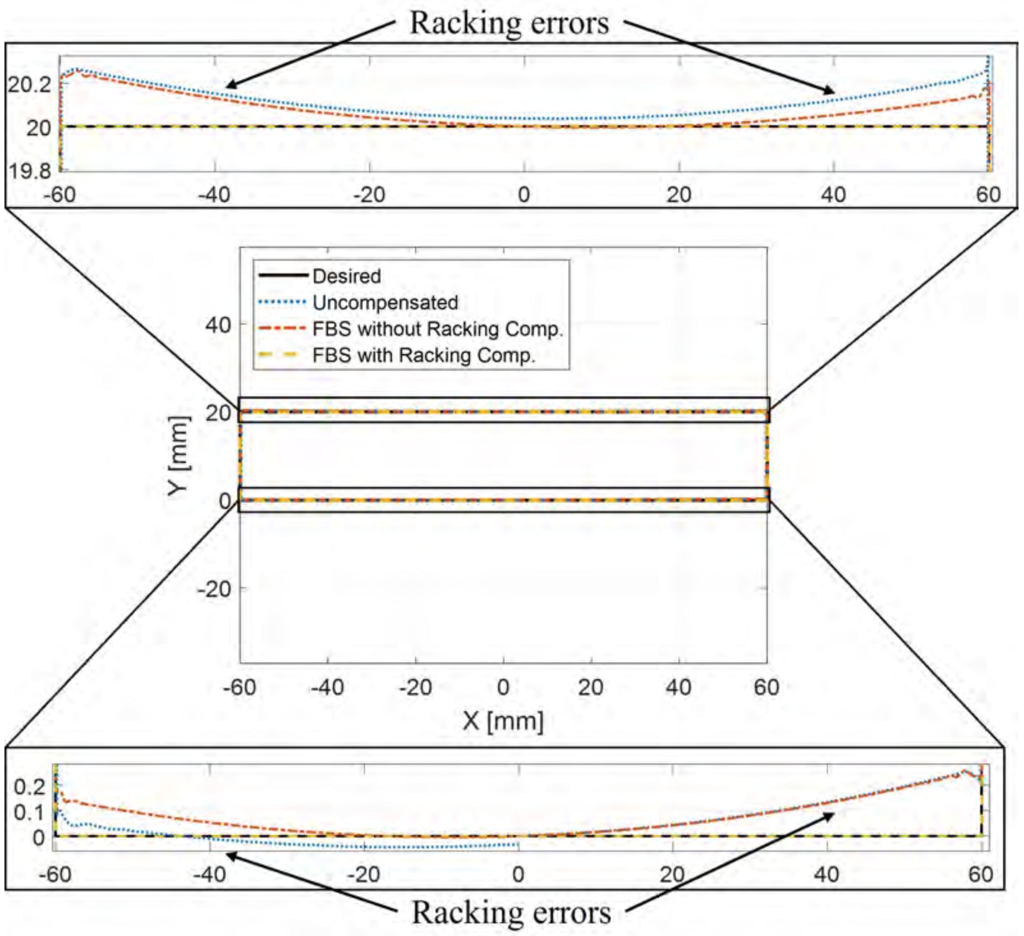


Figure 2.8: Simulated output response of the baseline case on the H-frame 3D printer for the uncompensated trajectory (dotted line) as well as the trajectories generated using FBS controllers without racking compensation (dot-dash line) and with racking compensation (dashed line).

Contour error [μm] (RMS / maximum)	Baseline	High-speed
Uncompensated	127.39 / 341.85	180.01 / 1374.3
FBS without racking compensation	94.02 / 257.84	117.53 / 516.04
FBS with racking compensation	**0.16 / 1.92**	**3.24 / 28.21**

Table 2.4: RMS and maximum contour errors along the baseline (low-speed) and the high-speed cases for all compensation strategies. The FBS with racking compensation controller leads to significant improvements in both the RMS and maximum contour errors for all cases.

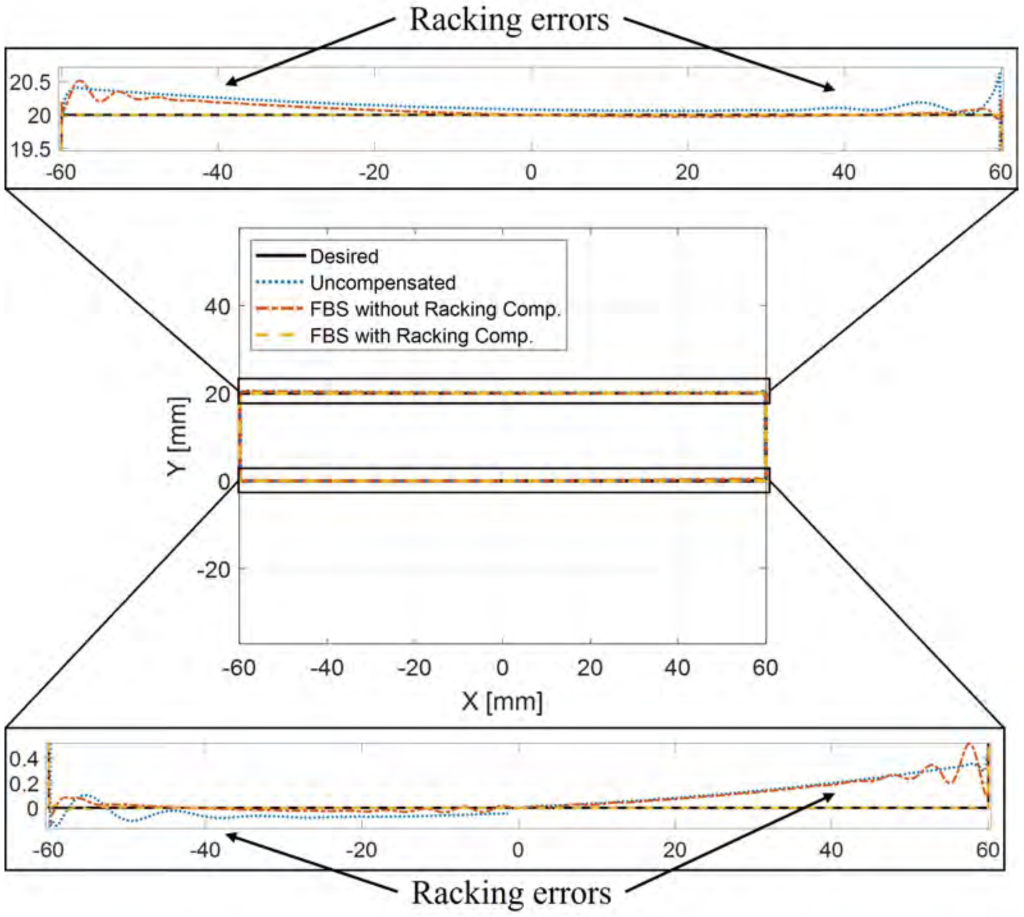


Figure 2.9: Simulated output response of the high-speed case on the H-frame 3D printer for the uncompensated trajectory (dotted line) as well as the trajectories generated using FBS controllers without racking compensation (dot-dash line) and with racking compensation (dashed line).

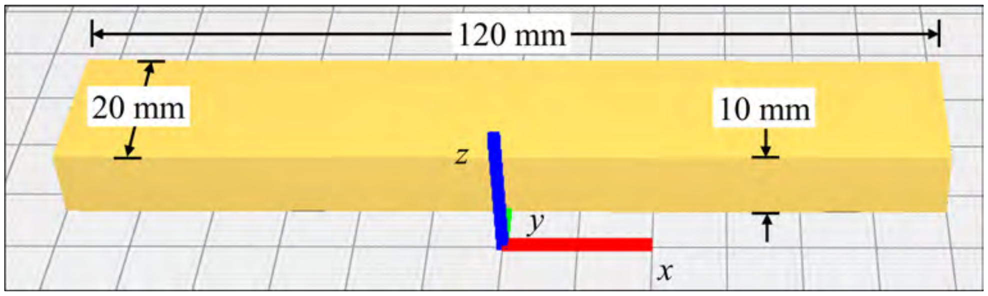


Figure 2.10: CAD model of the part in Ultimaker Cura®.

Print parameters	Baseline	High-speed
Wall speed	60 mm/s	150 mm/s
Maximum acceleration	1×10^3 mm/s^2	1×10^4 mm/s^2
Maximum jerk	5×10^7 mm/s^3	5×10^7 mm/s^3
Print material	Polylactic acid	Polylactic acid
Nozzle diameter	0.4 mm	0.4 mm
Extrusion rate	72 steps/mm	72 steps/mm
Filament volumetric flow rate	1.74×10^{-3} mm/step	1.74×10^{-3} mm/step
Nozzle temperature	205°C	205°C
Bed temperature	60°C	60°C
Layer height	0.1 mm	0.1 mm
Wall thickness	0.8 mm	0.8 mm

Table 2.5: Print parameters for the baseline and high-speed cases of the rectangular prism shown in Fig. 2.10.

maximum acceleration and maximum jerk values are reported in Table 2.5, along with other print parameters. To ensure adhesion to the bed, the first four layers in both cases were printed at a speed of 20 mm/s.

Figures 2.11 and 2.12 compare the rectangular block printed for the baseline and high-speed cases, respectively, using no compensation, FBS without racking compensation and the proposed FBS with racking compensation. LPFBS was used in both FBS cases, with parameters $n_{up} = 11$, $n_C = 22$, $L_C = 220$, $m = 5$, and $L = 20$ (see Appendices B and C for more details). As can be seen from Fig. 2.11(a), the baseline case without compensation yields high-quality prints (in terms of vibration and racking) which are hardly distinguishable from those with compensation (Fig. 2.11(b) and (c)). This observation highlights why printers with vibration and rack-

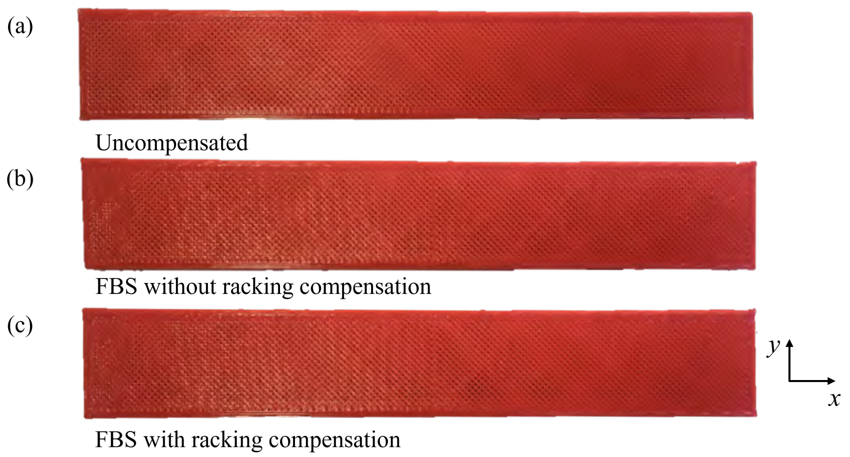

(a)
Uncompensated
(b)
FBS without racking compensation
(c)
FBS with racking compensation
y
x

Figure 2.11: Examples of the baseline parts printed (a) without any compensation, (b) using FBS without racking compensation, and (c) using FBS with racking compensation. At low speed, print quality differences are hardly distinguishable between the uncompensated and compensated control approaches.

ing problems often yield excellent print quality at low speeds (albeit at the cost of productivity). However, as seen from Fig. 2.12(a), the situation is different with the high-speed case. The part printed without compensation suffered from layer shifts during the printing process (as also observed in [5]). The FBS approach without racking compensation (Fig. 2.12(b)) provides sufficient compensation to remove the layer shifts from the printed part. However, the quality of the part is still degraded relative to the baseline case which is evident from the waviness along the edges of the part in Fig. 2.13 (also see in the simulation results of Fig. 2.9). The proposed FBS with racking compensation rectifies both the layer shifting and waviness due to racking leading improved print quality.

To quantify the improvements to the high-speed case due to the proposed approach, the acceleration of the gantry was measured using the ADXL335 accelerometers positioned at P_1 and P_2 in Fig. 2.2(b) during the print motion of the high-speed case. This measurement is used as a proxy for the position errors since the 3D printer is not equipped with position sensors on the gantry or nozzle. Figure 2.14 shows the acceleration measurements for one side of the gantry as a function of the position

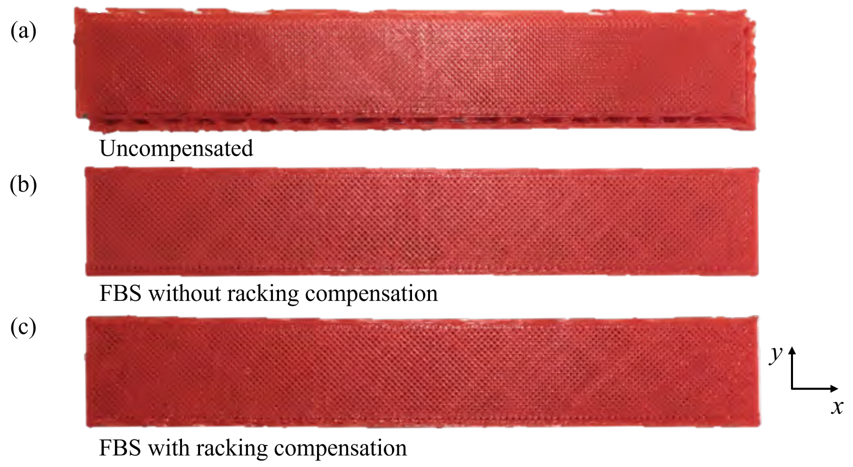

(a)
Uncompensated
(b)
FBS without racking compensation
(c)
FBS with racking compensation
y
x

Figure 2.12: Examples of the high-speed parts printed (a) without any compensation, (b) using FBS without racking compensation, and (c) using FBS with racking compensation. Layer shifts can be seen in the uncompensated part when compared to the FBS compensated parts. Differences between the FBS without racking compensation and FBS with racking compensation parts can be seen in the enlarged corner view comparison in Fig. 2.13

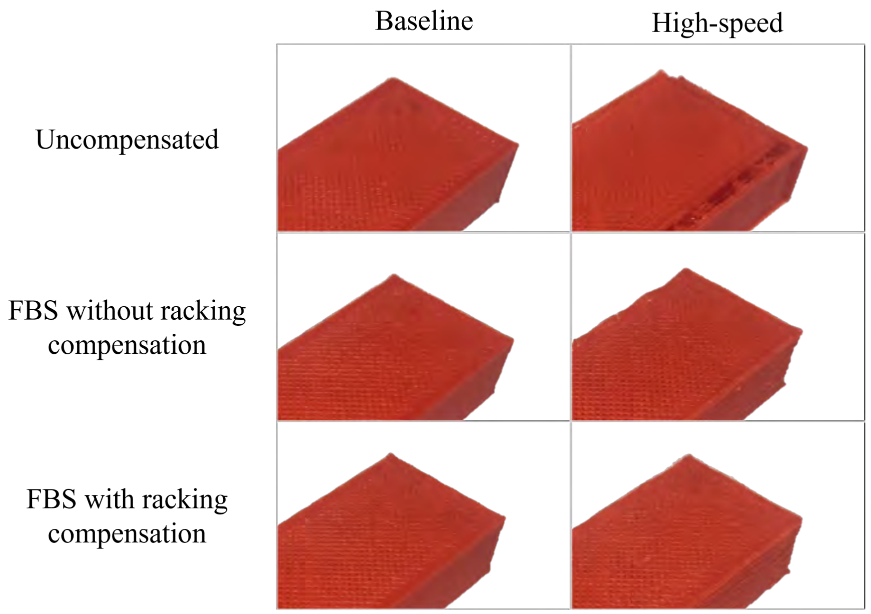

Baseline
High-speed
Uncompensated
FBS without racking compensation
FBS with racking compensation

Figure 2.13: Enlarged corner view comparison of the baseline and high-speed cases using the uncompensated, FBS without racking compensation, and FBS with racking compensation control approaches. The vibration and racking errors cause layer shifting and waviness along the edge, respectively, for the uncompensated and the FBS without racking compensation approaches applied to the high-speed case. The proposed FBS with racking compensation approach eliminates the errors.

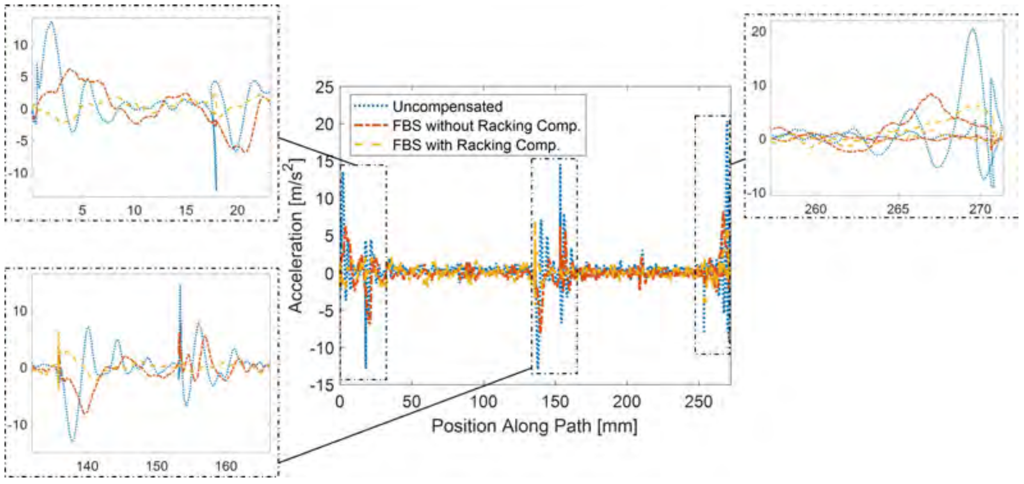


Figure 2.14: Acceleration measurements of the racking motion of the gantry during high-speed printing on the H-frame 3D printer without any compensation (dotted line), using FBS without racking compensation (dot-dash line), and using FBS with racking compensation (dashed line). The positions of high acceleration are enlarged to show the differences.

along the rectangular path of one layer during the print motion for (1) an uncompensated part, (2) a part printed using FBS without racking compensation and (3) a part printed using FBS with racking compensation. (Note that the path of the measurements starts at the $\{x,\ y\}$ position of (60, 0) mm instead of (0, 0) as in Fig. 2.6.) The RMS acceleration of the gantry during the three high-acceleration portions highlighted in Fig. 2.14 are 3.63, 4.25, and 5.61 m/s^2, respectively, for the uncompensated approach, 2.41, 2.62, and 2.63 m/s^2 for FBS without racking compensation, and 1.10, 1.04, and 2.21 m/s^2 for FBS with racking compensation. These data indicate that the additional acceleration created by racking in the uncompensated trajectory and the trajectory using FBS without racking compensation are significantly reduced by the proposed racking compensation approach, leading to greater positional accuracy.

To further quantify how the racking errors affect the final component for the high-speed case, we printed 15 copies of the rectangular prism using the three different compensation approaches (5 copies for each). The width, w, of each of the 15 printed parts was measured at the left, middle, and right side using Husky digital calipers

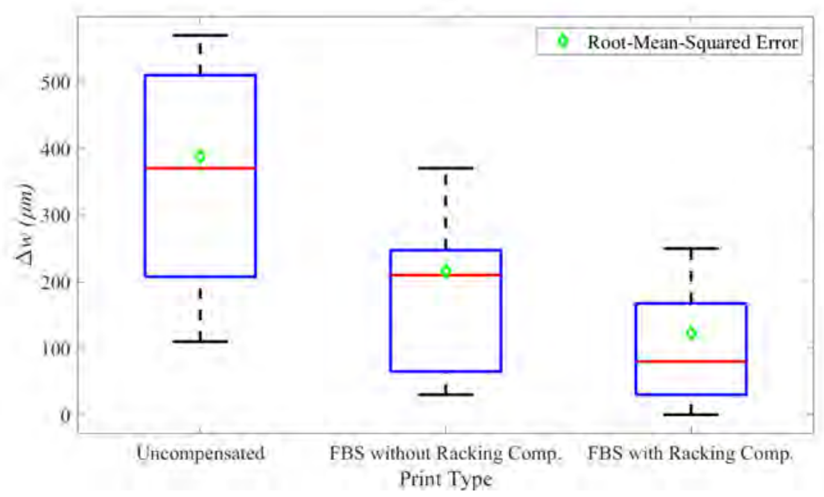


Figure 2.15: Box-and-whisker plot of measured absolute width error, Δw, of printed parts compared to the desired width of 20 mm for each of the compensation strategies applied to the high-speed case: (1) no compensation, (2) FBS without racking compensation, and (3) FBS with racking compensation. The red horizontal lines represent the median of Δw, which is overlaid with root-mean-square width error, Δw_{rms} (diamonds).

(model #1467H, 10 μm resolution) and compared to the desired width of $w_d = 20$ mm. Figure 2.15 shows a box-and-whisker plot of the absolute value of the width error ($\Delta w = |w_d - w|$), in μm, overlaid with the RMS width error, Δw_{rms}. The proposed FBS controller with racking compensation improved the median Δw by 61% when compared to FBS without racking compensation, from 210 μm to 80 μm, and by 78% when compared to no compensation from 370 μm to 80 μm. The proposed controller also improved Δw_{rms} by 43% (216 μm to 122 μm) and 68% (388 μm to 122 μm) when compared to FBS without racking compensation and no compensation, respectively. Note that even though print quality depends on several factors (e.g., material, extrusion rate, extrusion temperature, etc.), in the comparisons discussed above for the baseline and high-speed cases, all other factors are maintained constant except for the compensation approach. Therefore, the differences in print quality are primarily due to the effects of compensation approach.

2.6 Summary

H-frame 3D printer architectures have the potential to achieve higher speeds and improved dynamic performance compared to traditional serial stack 3D printers due to their use of stationary motors. However, these benefits come at the cost of racking errors, caused by parasitic torsional motions, which limit their static and dynamic accuracy. This chapter discusses the proposal of a purely software-based approach for compensating racking errors on H-frame 3D printers using the filtered B-splines (FBS) feedforward approach, which has been used to improve the performance of 3D printers in the literature [5,6,9]. Building on the prior work, the proposed FBS controller is designed to address coupled linear parameter varying dynamics rather than the decoupled linear time invariant dynamics addressed in prior work. Additionally, a decoupled approximation of the coupled FBS controller was developed and validated analytically and numerically. It was shown to significantly increase computational efficiency with little or no sacrifice to error compensation accuracy. The decoupled FBS controller with racking compensation is benchmarked in simulation and experiments on an H-frame 3D printer against the standard FBS controller without racking compensation. We show that racking errors are significantly reduced using the proposed method and a 43% improvement in the shape accuracy of a high-speed 3D printed part is observed in experiments compared to parts printed with the standard FBS controller.

A major practical benefit of the software-based approach for racking error compensation is that it reduces racking errors without requiring mechanical modification of a 3D printer. Hence, it can be applied to existing H-frame 3D printers. It can also be used to augment other mechanical or software-based approaches for addressing racking errors, like the use of stiffer guideways, counterweights, dampers, and feedback controllers. This chapter demonstrates the potential of software-based compensation strategies to improve the dynamic performance of 3D printers with complex architectures. The subsequent chapters aim to further explore how similar ideas can be applied to other complex motion systems like the delta 3D printer.

CHAPTER III

Modeling of the Delta Robot

3.1 Overview

This chapter describes a framework to obtain accurate models of the delta parallel-axis robots used for 3D printing. In the last chapter, the modeling and control of the H-frame 3D printer was presented. Although the H-frame's dynamics are LPV, the relationships between the model's parameters are rather straightforward and geometrically intuitive. Therefore, obtaining the model does not demand significant modeling effort. On the other hand, the dynamics of the delta 3D printer are significantly more complex to model due to its kinematic structure: it uses three vertical joints to move an end-effector vertically and laterally, and the three vertical joints are coupled by a common connection to the end-effector. This structure leads to position-dependent, coupled, and nonlinear dynamics, such that modeling the delta printer accurately—without a thoughtful approach—would require measurements at many positions in the printer's workspace.

In this chapter, we propose a more efficient approach for obtaining an accurate model of the printer. We leverage knowledge about its structure and the idea of receptance coupling (RC) [76–78] to decompose the delta's dynamic model into two sub-models: the first is an experimentally-identified model of the joints without the end-effector connections, which is a decoupled and LTI model; the second is an analytically-derived model that describes the end-effector connections to the joints and, therefore, couples the system. We show that an accurate parameter-varying model can be efficiently obtained with a few measurements at *one* position by identifying the parameters of each subsystem independently before combining them together. Once the model is identified, we use frequency response functions (FRFs), measured at arbitrary locations in the workspace, to validate that our prediction model results in reasonably accurate predictions of the position-dependent dynamics of a commercial delta 3D printer.

The chapter is organized as follows: the kinematic model of the prismatic-joint delta robot is described in Section 3.2, which exemplifies the complexity of the machine. Section 3.3 outlines the framework we propose to determine the delta printer's model with sparse measurements. The exact measurements needed and the process of identifying each parameter in the model is described in Section 3.4. We validate the identified model using empirical measurements from a commercial delta 3D printer in Section 3.5 and discuss the insights gained as a result. Finally, the chapter closes with a summary in Section 3.6.

3.2 Inverse and Forward Kinematics

3.2.1 Description of Delta 3D Printer

Unlike most commercial delta robots that are designed with rotary joints [36], delta 3D printers typically have three prismatic joints (i.e., three vertical columns) actuated by three stationary stepper motors. We will refer to the joints as columns A, B, and C (see Fig. 3.1(a)). The motors each drive a timing belt to control the position of a carriage, which is mounted on a linear guideway located on each column (as shown in Fig. 1.3). The carriages are connected to the end-effector by six rods which we'll refer to as "forearms"—two forearms are connected to each of the three carriages on one end, and the end-effector on the other end. Each forearm is the same length and the connections to the carriage and the end-effector are made through universal (i.e., spherical) joints. This configuration of parallel forearms ensures that the plane of the end-effector platform is always parallel to the bed, meaning that the printer's nozzle is always parallel to the bed. The parallel forearms also force the vertical plane where the end-effector platform and forearms meet to be parallel to the corresponding vertical plane connecting the carriage and the forearms.

3.2.2 Inverse Kinematics

The goal of inverse kinematics is to recover the column positions A_z, B_z, and C_z from the commanded x-, y-, and z-axis workspace position given by the G-code. The column positions are related to the Cartesian coordinates by the following spherical constraint equations (as derived in Appendix D):

$$(x - A_{vx})^2 + (y - A_{vy})^2 + A_{cz}^2 = L^2 \tag{3.1}$$

$$(x - B_{vx})^2 + (y - B_{vy})^2 + B_{cz}^2 = L^2 \tag{3.2}$$

$$(x - C_{vx})^2 + (y - C_{vy})^2 + C_{cz}^2 = L^2 \tag{3.3}$$

where A_{vx} and A_{vy} are the x- and y-axis coordinate position of the virtual column of A, which is defined as the position of column A shifted by an offset to account for the carriage and end-effector width (see Fig. 3.1(b) and Appendix D), B_{vx}, B_{vy}, C_{vx}, and C_{vy} are similarly defined for columns B and C, and

$$A_{cz} = A_z - H_{ez} - z \tag{3.4}$$

$$B_{cz} = B_z - H_{ez} - z \tag{3.5}$$

$$C_{cz} = C_z - H_{ez} - z, \tag{3.6}$$

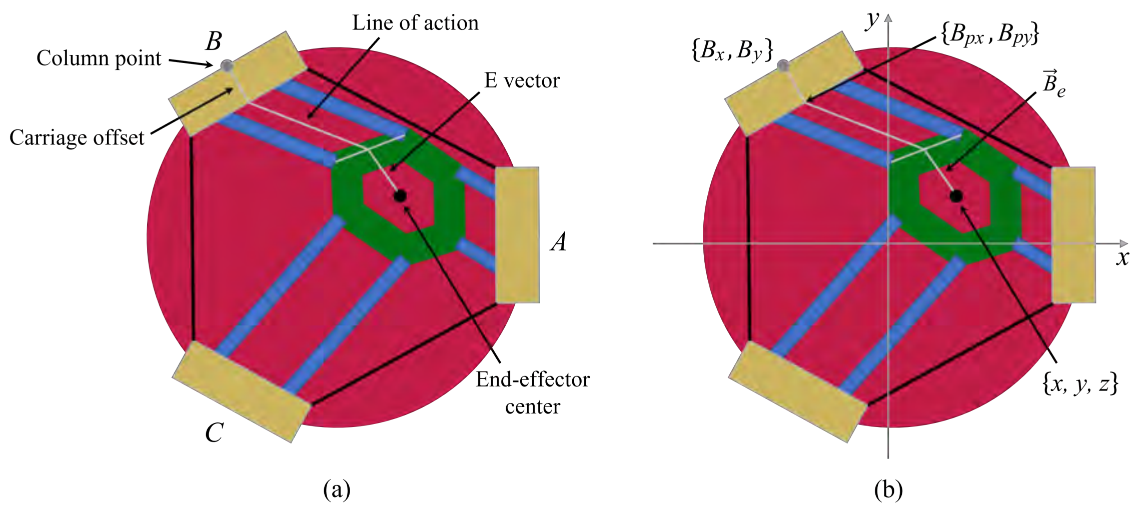


Figure 3.1: (a) Overhead view of the delta printer with labels of reference points; (b) same overhead view as (a), but overlays the xy-coordinate axis and labels of key coordinate positions of the carriage (exemplified for carriage B) and end-effector. (The labels are also helpful for following the derivation in Appendix D).

where A_z is the distance from the print bed to carriage A's position as shown in Fig. 3.2 (same for B_z and C_z), and H_{ez} is distance from the end-effector platform to the tip of the nozzle (also see Fig. 3.2). Therefore, we can find the column positions using the end-effector position and the measured constants as

$$A_z = z + H_{ez} + \sqrt{L^2 - (x - A_{vx})^2 + (y - A_{vy})^2} \tag{3.7}$$

$$B_z = z + H_{ez} + \sqrt{L^2 - (x - B_{vx})^2 + (y - B_{vy})^2} \tag{3.8}$$

$$C_z = z + H_{ez} + \sqrt{L^2 - (x - C_{vx})^2 + (y - C_{vy})^2} \tag{3.9}$$

3.2.3 Forward Kinematics

Forward kinematics is the problem of determining the x-, y-, and z-axis position of the end-effector given the column positions. One approach is to derive the forward kinematics using same spherical constraint equations defined in the previous subsection. With three equations, it is possible to solve for the three unknowns but the

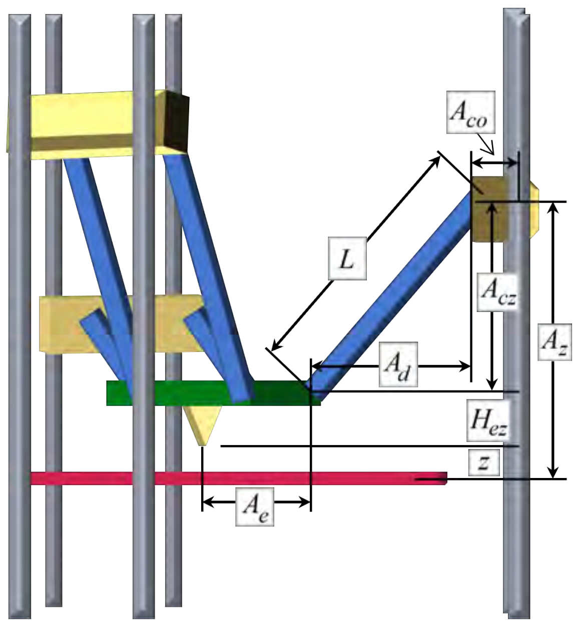


Figure 3.2: Three-dimensional schematic of the delta printer showing the right triangle created by the forearms, the column, and the distance from the end-effector to the column in the xy-plane. The virtual columns used in the derivation are computed using the labeled distances.

squares make it difficult. A simpler approach is to use trilateration to compute the forward kinematics numerically. Trilateration is the process of finding a point in 3D space based on its distance from three known points. (A common application for trilateration is GPS). There are various algorithms for trilateration[1] available online; in Appendix E, we reproduce the algorithm used in this work as a reference (in the Matlab programming language).

3.3 Control-Oriented Dynamic Modeling Framework

To derive the model of the delta printer using RC, we begin by decomposing the model of the full assembly into two sub-models. Sub-model 1 describes the carriage output position q_i as a function of two inputs: (a) the desired position of the carriage q_{d_i} and (b) the forces F_{q_i} imposed on the carriage due to the dynamics of the fore-

[1]see "True-range multilateration" article on Wikipedia: https://en.wikipedia.org/wiki/True-range_multilateration

arms and end-effector, where $i \in \{A, B, C\}$. Sub-model 2 describes the relationship between the end-effector's position $\mathbf{X} = [x \quad y \quad z]^T$ and F_{q_i}.

Sub-model 1 decouples each carriage as though they are disconnected from the end-effector. Accordingly, the model is assumed to be linear since each carriage consists of the carriage mass and timing belt, which can be modeled as a mass-spring-damper system [5–7, 9]. The relationship between the inputs and q_i are given by continuous-time LTI single-input, single-output (SISO) systems $G_{q_{d_i}}(s)$, the carriage position to position FRF, and $G_{Fq_i}(s)$, the external force to carriage position FRF. Both SISO systems are measured from experiments as a summation of vibration modes, such that

$$q_i(s) = G_{q_{d_i}}(s) q_{d_i}(s) + G_{Fq_i}(s) F_{q_i}(\mathbf{X}, s) \tag{3.10}$$

where s is the Laplace variable and F_{q_i} is a function of $\mathbf{X}$. Since each carriage is identical, we assume the SISO FRFs are identical for each carriage, i.e., $G_{q_{d_i}}(s) = G_{q_d}(s)$ and $G_{Fq_i}(s) = G_{Fq}(s)$ for all i.

Sub-model 2 connects the end-effector to the carriages through F_{q_i} in Eq. (3.10). Therefore, it incorporates the flexible dynamics of the forearms and the end-effector (henceforth simply referred to as the end-effector dynamics). The expression of $F_{q_i}(\mathbf{X}, s)$ is characterized by the Jacobian matrix, which relates the joint space and task space velocities [45] as

$$\dot{\mathbf{X}} = \mathbf{J}\dot{\mathbf{q}} \tag{3.11}$$

where $\mathbf{q} = [q_A \quad q_B \quad q_C]^T$ is the joint space coordinate vector (i.e., carriage coordinates) and $\mathbf{J} \in \mathbb{R}^{3\times 3}$ is the Jacobian matrix. Accordingly, we begin by deriving the Jacobian matrix in sub-section 3.3.1. Then, in sub-section 3.3.2, we use the Jacobian to derive the analytical relationship between $\mathbf{X}$ and F_{q_i}. A visual representation of the overall architecture of the delta model, composed of sub-models 1 and 2, is shown

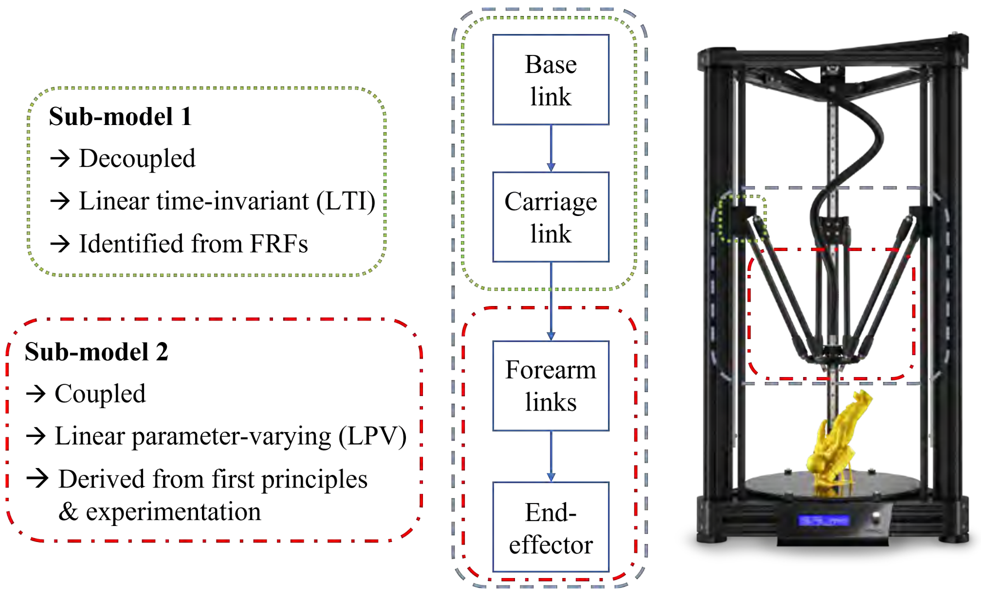


Figure 3.3: Overall architecture of the delta 3D printer model broken up into its two components. Sub-model 1 consists of the base and carriage link dynamics which are modeled as decoupled and LTI. These dynamics can be identified directly from frequency response function (FRF) measurements. Sub-model 2 consists of the forearm links and end-effector dynamics which are modeled as coupled and LPV. These dynamics are derived from first principles and the parameters are identified empirically.

in Fig. 3.3.

3.3.1 Sub-model 2: The Jacobian matrix

The Jacobian matrix is derived based on work from [45]. Without loss of generality, we locate the origin of the task space coordinate system at the center of the bed and align the x-axis with the center of carriage A as discussed in Section 3.2.2. The spherical constraint equations that govern the kinematics can be expanded to include task and joint space coordinates as

$$(x - A_{vx})^2 + (y - A_{vy})^2 + (z + H_{ez} - q_A)^2 = L^2 \tag{3.12}$$

$$(x - B_{vx})^2 + (y - B_{vy})^2 + (z + H_{ez} - q_B)^2 = L^2 \tag{3.13}$$

$$(x - C_{vx})^2 + (y - C_{vy})^2 + (z + H_{ez} - q_C)^2 = L^2 \tag{3.14}$$

In [45], the spherical constraint equations are derived for the rotary-joint delta robot instead of the prismatic-joint delta robot. From there, the procedure is identical and it is reproduced below for the reader's convenience. Equations (3.12)-(3.14) can be written in vector form as

$$\mathbf{s}_i^T \mathbf{s}_i - L^2 = 0 \tag{3.15}$$

where

$$\mathbf{s}_i = \begin{bmatrix} x - i_x \\ y - i_y \\ z + H_{ez} - q_i \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \left(\begin{bmatrix} i_x \\ i_y \\ -H_{ez} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} q_i \right) \tag{3.16}$$

Taking the time derivative of Eq. (3.15) yields

$$\mathbf{s}_i^T \dot{\mathbf{s}}_i + \dot{\mathbf{s}}_i^T \mathbf{s}_i = 0 \tag{3.17}$$

which, from the commutative property of the vector product, can be rewritten as

$$\mathbf{s}_i^T \dot{\mathbf{s}}_i = 0 \tag{3.18}$$

where

$$\dot{\mathbf{s}}_i = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \dot{q}_i = \dot{\mathbf{X}} + \mathbf{b}\dot{q}_i. \tag{3.19}$$

Rearranging Eq. (3.18) with the definition of $\mathbf{b}$, we have

$$\begin{bmatrix} \mathbf{s}_A^T \\ \mathbf{s}_B^T \\ \mathbf{s}_C^T \end{bmatrix} \dot{\mathbf{X}} + \begin{bmatrix} \mathbf{s}_A^T\mathbf{b} & 0 & 0 \\ 0 & \mathbf{s}_B^T\mathbf{b} & 0 \\ 0 & 0 & \mathbf{s}_C^T\mathbf{b} \end{bmatrix} \dot{\mathbf{q}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.20}$$

From Eq. (3.20), we can obtain the relation in Eq. (3.11) where

$$\mathbf{J} = -\begin{bmatrix} \mathbf{s}_A^T \\ \mathbf{s}_B^T \\ \mathbf{s}_C^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{s}_A^T\mathbf{b} & 0 & 0 \\ 0 & \mathbf{s}_B^T\mathbf{b} & 0 \\ 0 & 0 & \mathbf{s}_C^T\mathbf{b} \end{bmatrix}. \tag{3.21}$$

After another time derivative of Eq. (3.20) and some transformations, we find the task space acceleration $\ddot{\mathbf{X}}$ as

$$\ddot{\mathbf{X}} = \begin{bmatrix} \mathbf{s}_A^T \\ \mathbf{s}_B^T \\ \mathbf{s}_C^T \end{bmatrix}^{-1} \left(\begin{bmatrix} \dot{\mathbf{s}}_A^T \\ \dot{\mathbf{s}}_B^T \\ \dot{\mathbf{s}}_C^T \end{bmatrix} \mathbf{J} + \mathbf{T} \right) \dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{3.22}$$

where

$$\mathbf{T} = \begin{bmatrix} \dot{\mathbf{s}}_A^T\mathbf{b} & 0 & 0 \\ 0 & \dot{\mathbf{s}}_B^T\mathbf{b} & 0 \\ 0 & 0 & \dot{\mathbf{s}}_C^T\mathbf{b} \end{bmatrix}.$$

3.3.2 Sub-model 2: End-effector position to carriage forces

To find the exogenous force F_{q_i} imposed on each carriage, we first write the force and moment (torque) balance equations about the end-effector's center of mass in task space coordinates. Then, we transform the resulting reaction forces to joint space coordinates using the Jacobian matrix. From the free body diagram in Fig. 3.4, the forces on the end-effector (in the Laplace domain) are given by

$$F_{Ax}(s) + F_{Bx}(s) + F_{Cx}(s) = w_x(s)x(s) \tag{3.23}$$

$$F_{Ay}(s) + F_{By}(s) + F_{Cy}(s) = w_y(s)y(s) \tag{3.24}$$

$$F_{Az}(s) + F_{Bz}(s) + F_{Cz}(s) = w_z(s)z(s) \tag{3.25}$$

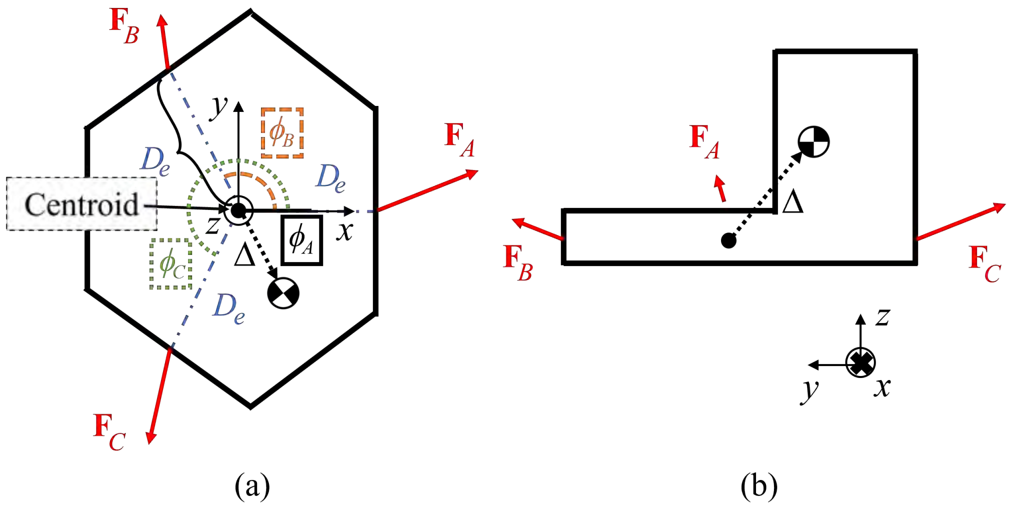


Figure 3.4: (a) Top view and (b) side view free body diagrams of the end-effector with reaction force vectors from each forearm. Note that the center of mass is not located at the centroid of the end-effector, which leads to an uneven distribution of reaction forces. The vector difference between the centroid and the center of mass is given by $\Delta = [\delta_x \quad \delta_y \quad \delta_z]^T$.

where $\mathbf{F}_A = [F_{Ax} \quad F_{Ay} \quad F_{Az}]^T$ are the respective x-, y-, and z-axis components of the force on the end-effector associated with carriage A, similarly for $\mathbf{F}_B = [F_{Bx} \quad F_{By} \quad F_{Cz}]^T$ and $\mathbf{F}_C = [F_{Cx} \quad F_{Cy} \quad F_{Cz}]^T$, 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. (Note that we will, henceforth, omit the Laplace variable, s, in the paragraph text for simplicity when it is understood in context.)

Remark 3.3(a): Here, we assume that the inertial dynamics are decoupled in the task space coordinates since the machine is designed to produce independent motion in each direction. However, the approach can be easily generalized for coupled dynamics in the task space coordinates.

We can compactly express the force equations as

$$\mathbf{F}_A(s) + \mathbf{F}_B(s) + \mathbf{F}_C(s) = \mathbf{W}(s)\mathbf{X}(s) \tag{3.26}$$

where

$$\mathbf{W}(s) = \begin{bmatrix} w_x(s) & 0 & 0 \\ 0 & w_y(s) & 0 \\ 0 & 0 & w_z(s) \end{bmatrix}. \tag{3.27}$$

Since the end-effector does not rotate during motion, the moment equations about the center of mass are given by

$$\mathbf{r}_A \times \mathbf{F}_A(s) + \mathbf{r}_B \times \mathbf{F}_B(s) + \mathbf{r}_C \times \mathbf{F}_C(s) = \mathbf{0}_{3\times 1} \tag{3.28}$$

where

$$\mathbf{r}_i = \begin{bmatrix} D_e \cos(\phi_i) - \delta_x \\ D_e \sin(\phi_i) - \delta_y \\ -\delta_z \end{bmatrix}, \tag{3.29}$$

δ_x, δ_y, and δ_z are the x- y- and z-coordinate distance from the centroid to the center of mass, D_e is the distance from the centroid to the forearm reaction force, and ϕ_i is the angle where carriage i is located with respect to the global x-axis on the horizontal plane (see Fig. 3.4). For simplicity, we neglect rotational effects of forearms, which have been found to be negligible in prior work [45, 79].

We can compute the reaction forces $\mathbf{F}_A$, $\mathbf{F}_B$, $\mathbf{F}_C$ as a function of the end-effector's motion by writing the six equations of motion in matrix form as

$$\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} \\ \mathbf{S}(\mathbf{r}_A) & \mathbf{S}(\mathbf{r}_B) & \mathbf{S}(\mathbf{r}_C) \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} \mathbf{F}_A(s) \\ \mathbf{F}_B(s) \\ \mathbf{F}_C(s) \end{bmatrix}}_{\mathbf{f}(s)} = \underbrace{\begin{bmatrix} \mathbf{W}(s)\mathbf{X}(s) \\ \mathbf{0}_{3\times 1} \end{bmatrix}}_{\mathbf{u}(s)}, \tag{3.30}$$

where $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix and $\mathbf{S}(\cdot) \in \mathbb{R}^{3\times 3}$ is the skew symmetric matrix defined on the input vector. The minimum norm solution for $\mathbf{f}$ is given by

$$\mathbf{f}(s) = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{u}(s) = \mathbf{L}^\dagger\mathbf{u}(s) \tag{3.31}$$

where $\mathbf{L}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{L}$. Note that since the bottom-half rows of $\mathbf{u}$ contain zeros, the last 3 columns of $\mathbf{L}^\dagger \in \mathbb{R}^{9\times 6}$ do not contribute to the reaction forces. Thus, we can use a reduced matrix $\tilde{\mathbf{L}}^\dagger \in \mathbb{R}^{9\times 3}$ and the reaction forces (i.e., each row of $\mathbf{f}$) can be written independently as

$$\mathbf{F}_i(s) = \mathbf{P}_i\mathbf{W}(s)\mathbf{X}(s) \tag{3.32}$$

where $\mathbf{P}_i \in \mathbb{R}^{3\times 3}$ is the matrix of constants representing the distribution of task space forces associated with carriage i, which is extracted from respective portions of $\tilde{\mathbf{L}}^\dagger$. Each force can be transformed to the joint space using $\bar{\mathbf{J}}_i \in \mathbb{R}^{3\times 1}$, the column vector extracted from the linearized Jacobian, denoted by $\bar{\mathbf{J}}$. The transpose of $\bar{\mathbf{J}}_i$ transforms the task space coordinates of the reaction joint associated with carriage i, denoted by (x_i, y_i, z_i), to the joint space coordinate q_i (see Fig. 3.4(a)).

Remark 3.3(b) [70]: The linearized Jacobian is obtained by linearizing Eqs. (3.11) and (3.22) about an equilibrium position denoted by

$$\bar{\mathbf{X}} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} \text{ and } \bar{\mathbf{q}} = \begin{bmatrix} \bar{q}_A \\ \bar{q}_B \\ \bar{q}_C \end{bmatrix}. \tag{3.33}$$

The transformed (and linearized) force is given by

$$F_{q_i} = \bar{\mathbf{J}}_i^T \mathbf{F}_i(s) = \bar{\mathbf{J}}_i^T \mathbf{P}_i \mathbf{W}(s)\mathbf{X}(s). \tag{3.34}$$

Furthermore, we can substitute the Jacobian relationship from Eq. (3.11) such that Eq. (3.34) becomes

$$F_{q_i} = \bar{\mathbf{J}}_i^T \mathbf{P}_i \mathbf{W}(s)\bar{\mathbf{J}}\mathbf{q}(s) \tag{3.35}$$

Finally, we can write the full model in the MIMO form of Eq. (3.10):

$$\mathbf{q}(s) = \mathbf{G}_{q_d}(s)\mathbf{q}_d(s) + \mathbf{G}_{Fq}(s) \begin{bmatrix} \bar{\mathbf{J}}_A^T \mathbf{P}_A \\ \bar{\mathbf{J}}_B^T \mathbf{P}_B \\ \bar{\mathbf{J}}_C^T \mathbf{P}_C \end{bmatrix} \mathbf{W}(s)\bar{\mathbf{J}}\mathbf{q}(s) \tag{3.36}$$

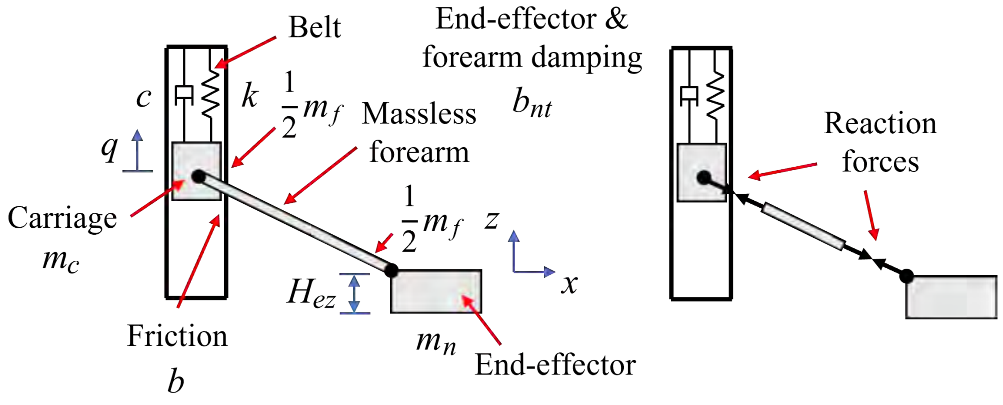
where $\mathbf{G}_{q_d}(s)$ and $\mathbf{G}_{Fq}(s)$ are 3×3 diagonal matrices that contain $G_{q_d}(s)$ and $G_{Fq}(s)$, respectively, as the diagonal entries. The model can be expressed simply as

$$\mathbf{q}(s) = \mathbf{G}(s)\mathbf{q}_d(s) \tag{3.37}$$

where

$$\mathbf{G}(s) = \left[\mathbf{I} - \mathbf{G}_{Fq}(s) \begin{bmatrix} \bar{\mathbf{J}}_A^T \mathbf{P}_A \\ \bar{\mathbf{J}}_B^T \mathbf{P}_B \\ \bar{\mathbf{J}}_C^T \mathbf{P}_C \end{bmatrix} \mathbf{W}(s)\bar{\mathbf{J}} \right]^{-1} \mathbf{G}_{q_d}(s), \tag{3.38}$$

yielding an LPV model of the delta 3D printer that can be used for model-based control. In Section 3.5, we present an example where $\mathbf{G}_{q_d}(s)$, $\mathbf{G}_{Fq}(s)$, $\mathbf{W}(s)$, and other parameters are identified for a delta printer with a flexible, two-mass end-effector. The parameters of the model are efficiently identified using data measured at one position, and used to predict FRFs at other positions.



(a) (b)

Figure 3.5: (a) Modeling schematic for delta 3D printer showing the belt-carriage system modeled as a mass-spring-damper system and, as done in Section 3.4, the forearm modeled as a massless entity with its mass split between the carriage and end-effector; (b) the forearm transmits reaction forces between the end-effector and the carriage.

3.4 Efficient System Identification

To study the framework described above, we identify the model of the MP Delta Pro 3D printer in this section. The printer is sold with a Bowden-style extruder [69] but we augmented it with a direct-drive extruder (shown in Fig. 3.6) to enhance extrusion performance [80]. Source files of the extruder design can be found on Thingiverse[2].

3.4.1 Decoupled carriage model identification (sub-model 1)

The modular nature of commercial delta 3D printers is an advantage in determining the FRFs because we can detach the forearms and the end-effector to measure the carriage position to position FRF, G_{q_d}. As in [45], the mass of the forearms is assumed to be split equally between the carriage and the end-effector (see Fig. 3.5(a)). Then, G_{q_d} can be represented (mechanically) as a mass-spring-damper system with

[2]Extruder design source files can be found at the following Thingiverse directory: https://www.thingiverse.com/ahasib/collections/direct-drive-extruder-mount-for-delta-printer

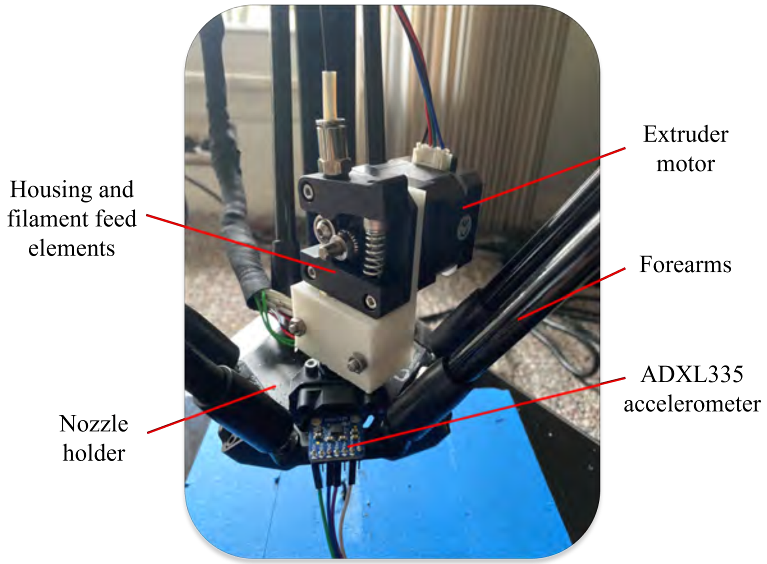


Figure 3.6: Image of a prototype of the direct drive extruder mounted on the MP Delta Pro 3D printer with the nozzle holder, extruder motor and housing labeled. The assembly is designed to fit within the existing end-effector platform without obstructing the forearm motion during printing. The ADXL335 accelerometers (pictured) were used to measure frequency response functions.

stiffness k, belt damping coefficient c, guideway friction b, and mass

$$m = m_c + \frac{1}{2}m_f \tag{3.39}$$

where m_c is the lumped mass of the carriage assembly and m_f is the mass of a pair of forearms (Fig. 3.5).

The carriage FRF is identified from acceleration data measured using ADXL335 accelerometers on the carriage, with the end-effector detached but one of two forearms still attached (as in Eq. (3.39)). We used a dSPACE MicroLabBox and Pololu stepper motor drivers (DRV8825) to command sine sweep perturbations around carriage positions corresponding to the task space positions $(x, y, z) = (0, 0, 30), (0, 0, 50), (0, 0, 70)$ mm. As expected, our measurements indicated that G_{q_d} was similar at the three locations, independent of the z-axis position, so we used the data from $(0, 0, 30)$ mm, shown in Fig. 3.7(a), to fit a 4th-order FRF of the form

$$G_{q_d}(s) = G_{q_{d,m}}(s)G_{q_{d,e}}(s), \tag{3.40}$$

where $G_{q_{d,m}}(s)$ and $G_{q_{d,e}}(s)$ represent the mechanical and electrical dynamics, respectively. (The electrical dynamics are created by the electrical circuitry that generates the stepper motor commands.) Hence, we have

$$\begin{aligned} G_{q_{d,m}}(s) &= \frac{cs+k}{ms^2+(c+b)s+k} && (3.41)\\ &= \frac{\frac{c}{m}s+\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} && (3.42)\end{aligned}$$

and

$$G_{q_{d,e}}(s) = \frac{d_1 s + d_0}{s^2 + d_2 s + d_0} \tag{3.43}$$

where ω_n is the natural frequency, ζ is the damping ratio,

$$2\zeta\omega_n = (c+b)/m, \tag{3.44}$$

$$\omega_n^2 = k/m, \tag{3.45}$$

and d_0, d_1, and d_2 are the coefficients of the electrical FRF. To determine the m, and therefore m_c (since m_f can be measured directly), we conducted the same sine sweep experiment with an additional mass $m_{add} = 200$ g mounted to the carriage. This experiment also generates mechanical FRFs that are identical for each carriage, one of which is shown in Fig. 3.7(b) and given by

$$\begin{aligned} G'_{q_{d,m}}(s) &= \frac{cs+k}{m's^2+(c+b)s+k} && (3.46)\\ &= \frac{\frac{c}{m'}s+\omega_n'^2}{s^2+2\zeta'\omega_n' s+\omega_n'^2} && (3.47)\end{aligned}$$

where $m' = m + m_{add}$,

$$2\zeta'\omega_n' = (c+b)/m', \tag{3.48}$$

and

$$\omega_n'^2 = k/m'. \tag{3.49}$$

From Eqs. (3.45) and (3.49), m can be computed as

$$m = \frac{\omega_n'^2}{(\omega_n^2 - \omega_n'^2)} m_{add} \tag{3.50}$$

and Eqs. (3.44) and (3.48) can be used to determine c and b. Similarly, Eq. (3.45) can be used to determine k. Additionally, the fitted 4th-order FRF of Eq. (3.40) can also be represented as

$$G_{q_d}(s) = \frac{g_2 s^2 + g_1 s + g_0}{s^4 + h_3 s^3 + h_2 s^2 + h_1 s + h_0} \tag{3.51}$$

where $g_{(\cdot)}$ and $h_{(\cdot)}$ are the coefficients of the fit. From the coefficients of Eqs. (3.40) and (3.51), we can write the set of equations

$$\begin{aligned} &g_2 = \frac{c}{m} d_1, \quad g_1 = \frac{c}{m} d_0 + \omega_n^2 d_1, \quad g_0 = \omega_n^2 d_0, \\ &h_3 = d_2 + 2\zeta\omega_n, \quad h_2 = d_0 + \omega_n^2 + 2\zeta\omega_n d_2, \\ &h_1 = \omega_n^2 d_2 + 2\zeta\omega_n d_0, \text{ and } h_0 = \omega_n^2 d_0. \end{aligned} \tag{3.52}$$

Using the fitted coefficients and computed parameters above, we can solve a least-squares problem to find d_0, d_1, and d_2.

Finally, the force to position FRF is given by the (mechanical) characteristic polynomial

$$G_{Fq}(s) = -\frac{1}{ms^2 + (c+b)s + k} \tag{3.53}$$

where the negative sign indicates that the forces involved are disturbance forces. The parameters for G_{q_d} and, therefore, G_{Fq} are reported in Table 3.1.

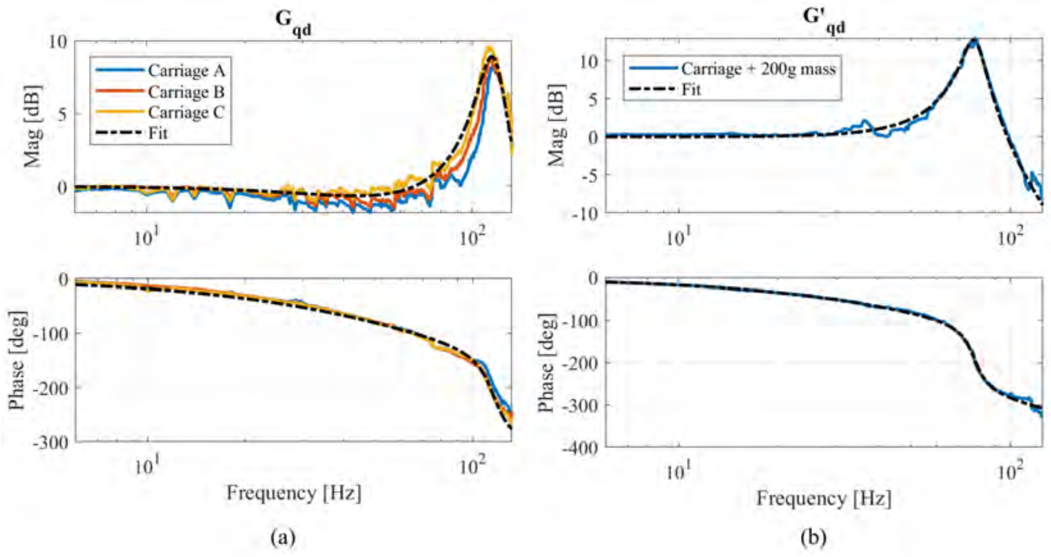


Figure 3.7: Position-to-position frequency response functions of the carriage *without* the end-effector dynamics (a) G_{q_d} and (b) with an additional 200 g mass attached to the carriage, G'_{q_d}. The data was measured for each carriage at the carriage locations (q_A, q_B, q_C) corresponding to $(x, y, z) = (0, 0, 30)$ mm, and a linear fit of one of the of the frequency response functions shown as the black dashed line. Note that only one carriage FRF is shown in (b) since the carriage FRFs are nearly identical as shown in (a).

3.4.2 End-effector model identification (sub-model 2)

To identify the inertial forces from the end-effector motion (Eq. (3.34)), we assume the end-effector can be modeled as a two-mass system—the nozzle holder mass and the extruder motor mass—with a spring and damper between the masses. We validate this assumption by isolating the end-effector and measuring the open loop force to acceleration FRF of the nozzle holder mass, m_1, using a PCB Piezotronics® impact hammer (model #086C03) and tri-axial accelerometer (model #356A44); the impact hammer was used to apply a force to m_1—similar to what is shown in Fig. 3.8. Figure 3.9 shows the measured force to acceleration FRF as well as the computed force to position FRF, which has the characteristic rigid body and flexible modes of a two-mass model. We can derive the force to acceleration FRF of m_1 model and fit the measurement in Fig. 3.9 with a second-order model to identify the masses. The two FRFs are represented by

$$\frac{s^2 X_{act}(s)}{F_h(s)} = \frac{m_2 s^2 + b_{act} s + k_{act}}{m_1 m_2 s^2 + (m_1 + m_2) b_{act} s + (m_1 + m_2) k_{act}} \tag{3.54}$$

$$= \frac{u_2 s^2 + u_1 s + u_0}{s^2 + v_1 s + v_0} \tag{3.55}$$

where m_2 is the extruder motor mass, b_{act} and k_{act} are the damping and stiffness constants in the direction activated by the impact hammer, $u_{(\cdot)}$ and $v_{(\cdot)}$ are the coefficients of the fitted FRF, and X_{act} and F_h are the position of m_1 and impact hammer force on m_1, respectively. Let ω_z and ω_p be the magnitude of the zero and pole location of the fitted FRF in Eq. (3.55), respectively. Then, assuming damping is negligible, it can be shown that

$$\omega_z^2 = \frac{k_{act}}{m_2} \tag{3.56}$$

$$\omega_p^2 = \frac{(m_1 + m_2) k_{act}}{m_1 m_2} \tag{3.57}$$

Solving Eqs. (3.56) and (3.57) simultaneously, we have

$$\frac{\omega_z^2}{\omega_p^2} = \frac{m_1}{m_1 + m_2} \Rightarrow m_1 = \frac{\omega_z^2}{\omega_p^2}(m_1 + m_2) \tag{3.58}$$

The mass of the nozzle holder, extruder motor, and housing elements can be measured with a scale to obtain the total mass, $m_{tot} = m_1 + m_2$. Therefore, m_1 and $m_2 = m_{tot} - m_1$ can be identified. Importantly, note that the values of b_{act} or k_{act} do not need to be known to identify m_1 and m_2. The measurement direction of the accelerometer simply needs to be parallel to the impact hammer force vector.

Using the two-mass model, we can write the set of force equations from Eqs. (3.23)-(3.25) for the nozzle holder and the extruder motor mass, which will be used to determine $\mathbf{W}(s)$ in Eq. (3.34). The equations for the nozzle holder are given by

$$F_{Ax} + F_{Bx} + F_{Cx} = m_1\ddot{x} + b_x\dot{x} + k_x x - b_x\dot{x}_2 - k_x x_2 \tag{3.59}$$

$$F_{Ay} + F_{By} + F_{Cy} = m_1\ddot{y} + b_y\dot{y} + k_y y - b_y\dot{y}_2 - k_y y_2 \tag{3.60}$$

$$F_{Az} + F_{Bz} + F_{Cz} = m_1\ddot{z} + b_z\dot{z} + k_z z - b_z\dot{z}_2 - k_z z_2 \tag{3.61}$$

where the reaction forces are as described in Section 3.3, b_x, b_y, and b_z are the damping coefficients in the x-, y- and z-axis directions, respectively, k_x, k_y, and k_z are the stiffness coefficients in the respective directions, and the subscript “2” denotes the coordinate system for the extruder motor and the parameters pertaining to it. Note that in Eqs. (3.59)-(3.61), we assume that the effect of cross stiffness and damping terms (e.g., x-to-y terms b_{xy}, k_{xy}) are negligible since the motion that each axis induces on the other two axes is negligible. For the extruder motor, the force equations are given by

$$b_x\dot{x} + k_x x = m_2\ddot{x}_2 + b_x\dot{x}_2 + k_x x_2 \tag{3.62}$$

$$b_y\dot{y} + k_y y = m_2\ddot{y}_2 + b_y\dot{y}_2 + k_y y_2 \tag{3.63}$$

$$b_z\dot{z} + k_z z = m_2\ddot{z}_2 + b_z\dot{z}_2 + k_z z_2. \tag{3.64}$$

Let $\mathbf{X}_2 = [x_2 \quad y_2 \quad z_2]^T$ be m_2's position. Then we can write Eqs. (3.62)-(3.64) in Laplace form as

$$\Big[(\mathbf{M}_2 s^2 + \mathbf{B}s + \mathbf{K})^{-1}(\mathbf{B}s + \mathbf{K})\Big]\mathbf{X}(s) = \mathbf{X}_2(s) \tag{3.65}$$

where

$$\mathbf{M}_2 = \begin{bmatrix} m_2 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_2 \end{bmatrix}, \tag{3.66}$$

$$\mathbf{B} = \begin{bmatrix} b_x & 0 & 0 \\ 0 & b_y & 0 \\ 0 & 0 & b_z \end{bmatrix}, \tag{3.67}$$

and

$$\mathbf{K} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}. \tag{3.68}$$

After computing the Laplace transform of Eqs. (3.59)-(3.61) and substituting $\mathbf{X}_2$ into Eqs. (3.59)-(3.61), we obtain a vector equation of the end-effector assembly in the form of Eq. (3.26):

$$\mathbf{F}_A(s) + \mathbf{F}_B(s) + \mathbf{F}_C(s) = \mathbf{W}(s)\mathbf{X}(s) \tag{3.69}$$

where

$$\mathbf{W}(s) = \mathbf{M}_1 s^2 + \mathbf{B}s + \mathbf{K} - (\mathbf{B}s + \mathbf{K})(\mathbf{M}_2 s^2 + \mathbf{B}s + \mathbf{K})^{-1}(\mathbf{B}s + \mathbf{K}) \tag{3.70}$$

and

$$\mathbf{M}_1 = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_1 & 0 \\ 0 & 0 & m_1 \end{bmatrix}. \tag{3.71}$$

Note that once all parameters are identified, we can add the moment equations to write the complete set of equations, as done in Eq. (3.30), and follow the procedure outlined in Section 3.3 to determine the full assembly FRF (Eq. (3.37)).

Finally, we can estimate b_j and k_j for $j \in \{x, y, z\}$ by: (a) measuring the full assembly FRFs during pure x-, y-, and z-axis translational sine sweep perturbations with the end-effector attached (and positioned at $(x, y, z) = (0, 0, 30)$ mm) and (b)

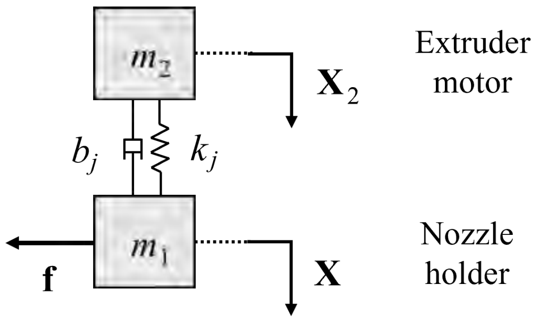


Figure 3.8: Schematic of the two-mass model of the direct drive extruder (end-effector) with the flexible components between the extruder motor and the nozzle holder modeled as a spring-damper system. The damping and spring coefficient, b_j and k_j, respectively, are defined on each axis of the task space (i.e., $j \in \{x, y, z\}$). The relative positions of the nozzle holder and the extruder motor are indicated by $\mathbf{X} = [x \quad y \quad z]^T$ and $\mathbf{X}_2 = [x_2 \quad y_2 \quad z_2]^T$, respectively.

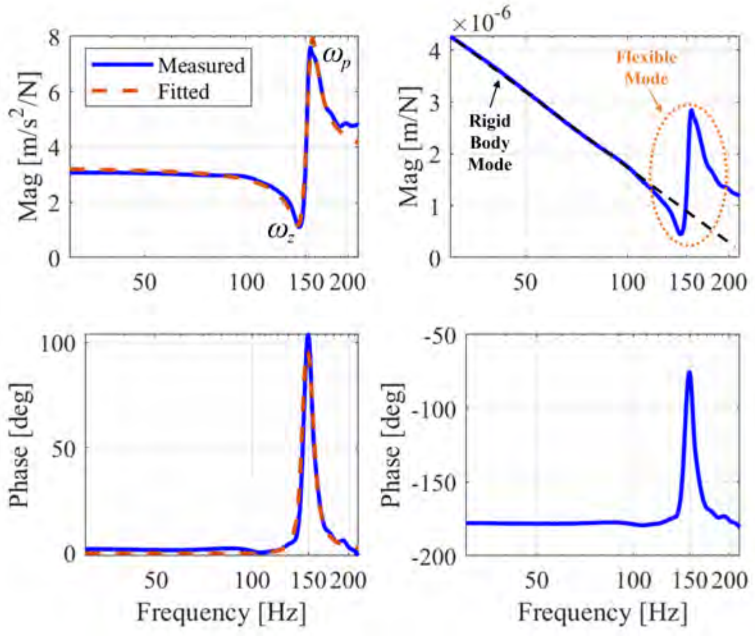


Figure 3.9: Open loop force to acceleration (left) and force to position (right) FRFs of the end-effector from impact hammer experiments. The force to position FRF is estimated at the discrete integral of the measured force to acceleration FRF to show the rigid body and flexible modes indicating a two-mass model of the end-effector. The labels ω_z and ω_p indicate the zero and pole location of the two-mass model, respectively.

using the measurements in the following least squares procedure. First, let

$$\mathbf{P}_i = \begin{bmatrix} p_{i,xx} & p_{i,yx} & p_{i,zx} \\ p_{i,xy} & p_{i,yy} & p_{i,zy} \\ p_{i,xz} & p_{i,yz} & p_{i,zz} \end{bmatrix} \tag{3.72}$$

from Eq. (3.32). Then, from Eq. (3.27),

$$\mathbf{P}_i\mathbf{W} = \begin{bmatrix} p_{i,xx}w_x & p_{i,yx}w_y & p_{i,zx}w_z \\ p_{i,xy}w_x & p_{i,yy}w_y & p_{i,zy}w_z \\ p_{i,xz}w_x & p_{i,yz}w_y & p_{i,zz}w_z \end{bmatrix}. \tag{3.73}$$

Therefore, if we command only one axis (take the z-axis, for example) without moving the other two, we will get three FRF expressions representing the z-to-q_A, z-to-q_B, and z-to-q_C transfer functions. To demonstrate this effect, examine Eq. (3.36) when $\mathbf{X}(s)$ is substituted for $\bar{\mathbf{J}}\mathbf{q}(s)$:

$$\mathbf{q}(s) = \mathbf{G}_{q_d}(s)\mathbf{q}_d(s) + \mathbf{G}_{Fq}(s) \begin{bmatrix} \bar{\mathbf{J}}_A^T\mathbf{P}_A \\ \bar{\mathbf{J}}_B^T\mathbf{P}_B \\ \bar{\mathbf{J}}_C^T\mathbf{P}_C \end{bmatrix} \mathbf{W}(s)\mathbf{X}(s) \tag{3.74}$$

where the input is $\mathbf{X}(s) = [0 \quad 0 \quad z(s)]^T$. Following this procedure, we obtain eight FRFs for the task space coordinates—three FRFs for the x-axis, two FRFs for the y-axis (carriage A does not move on pure y-axis translations), and three FRFs for the z-axis. The measured and fitted FRFs for the x-, y-, and z-axis are shown in Figs. 3.10(a), 3.10(b), and 3.10(c), respectively. We use them to identify the end-effector's stiffness and damping parameters as follows:

First, the full assembly FRFs are fit to a 4th-order mechanical system (6th-order with electrical dynamics)

$$G''_{q_{ji,m}}(s) = \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}. \tag{3.75}$$

It can also be shown that

$$w_j(s) = \frac{s^2(m_1 m_2 s^2 + (m_1 + m_2) b_j s + (m_1 + m_2) k_j)}{m_2 s^2 + b_j s + k_j}, \tag{3.76}$$

which can be substituted into Eq. (3.37) to obtain an expression for the characteristic polynomial of the full assembly carriage mechanical FRFs as

$$\begin{aligned} D_{ji}(s) = (m + \alpha_{ji} m_1) m_2 s^4 \\ + [(m + \alpha_{ji}(m_1 + m_2)) b_j + m_2(c + b)] s^3 \\ + [(m + \alpha_{ji}(m_1 + m_2)) k_j + m_2 k + \alpha_{ji}(c + b) b_j] s^2 \\ + ((c + b) k_j + b_j k) s + k_j k, \end{aligned} \tag{3.77}$$

where α_{ji} is the multiplicative factor that transforms force from the j-axis motion to the force on carriage i, and is computed as

$$\begin{bmatrix} \bar{\mathbf{J}}_A^T \mathbf{P}_A \\ \bar{\mathbf{J}}_B^T \mathbf{P}_B \\ \bar{\mathbf{J}}_C^T \mathbf{P}_C \end{bmatrix} = \begin{bmatrix} \alpha_{xA} & \alpha_{yA} & \alpha_{zA} \\ \alpha_{xB} & \alpha_{yB} & \alpha_{zB} \\ \alpha_{xC} & \alpha_{yC} & \alpha_{zC} \end{bmatrix} \tag{3.78}$$

with $\bar{\mathbf{J}}_i$ computed for the configuration at $(x, y, z) = (0, 0, 30)$ mm. To find $\mathbf{P}_i$, we designed a CAD model of the end-effector in SolidWorks® using the measured mass and dimensions of each component of the end-effector. From the CAD model, the center of mass of the end-effector can be automatically computed by SolidWorks® to obtain δ_x, δ_y, and δ_z and, therefore, $\mathbf{r}_i$ in Eq. (3.29). From Eq. (3.30), $\mathbf{r}_i$ is used to

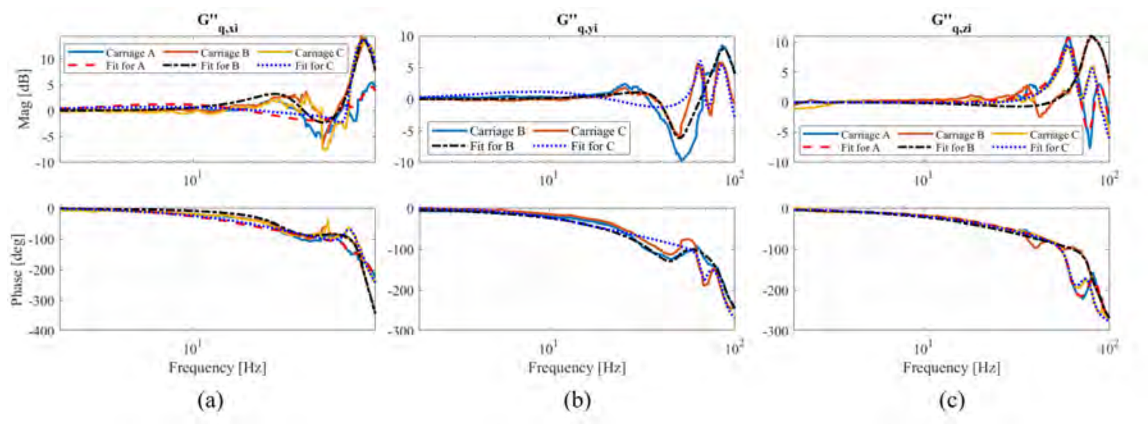


Figure 3.10: Position-to-position frequency response functions of the carriage *with* the end-effector attached, for the x-axis ($G''_{q,xi}$), y-axis ($G''_{q,yi}$) and z-axis ($G''_{q,zi}$). The data was measured for the same carriage location as Fig. 3.7.

compute $\mathbf{L}$ and, therefore, $\tilde{\mathbf{L}}^{\dagger}$ from which $\mathbf{P}_A$, $\mathbf{P}_B$, and $\mathbf{P}_C$ are extracted. Finally, using the fitted coefficients of the characteristic polynomial in Eq. (3.75) (one fit for each of the nine measured FRFs), we approximate b_j and k_j in Eq. (3.77) via least squares using a similar process to the one outlined for Eq. (3.52).

Table 3.1 reports the identified parameters of the delta printer model. In the following section, we validate the model by comparing its predictions to measurements from the machine.

3.5 Experimental Validation

Since we actuate the delta printer using its carriages, predicting how the carriage dynamics vary as a function of position is paramount. By studying the dynamic variation at a few positions, we observed that the dynamics of each carriage varied significantly along the carriage line of action: the line in the xy-plane that extends from the position of the carriage through the origin (see Fig. 3.11 and Section 3.2 for more details). Accordingly, we measured the full assembly carriage FRFs at $(x, y) = (-80, 0)$, $(40, -69)$ and $(40, 69)$ mm, corresponding to a distance of 80 mm from the origin along the line of action for carriages A, B, and C, respectively.

Table 3.1: Identified system parameters of the MP Delta Pro 3D printer

Symbol	Value (units)
m_c	0.179 kg
m_f	0.032 kg
m_1	0.542 kg
m_2	0.109 kg
c	5.31 N-s/m
b	13.4 N-s/m
b_x	20.4 N-s/m
b_y	40.5 N-s/m
b_z	19.6 N-s/m
k	1.21 $\times 10^5$ N/m
k_x	3.31 $\times 10^3$ N/m
k_y	7.04 $\times 10^3$ N/m
k_z	1.29 $\times 10^5$ N/m
ω_n	730 rad/s
ω_n'	490 rad/s
ω_z	1519 rad/s
ω_p	1674 rad/s
ζ	0.092
ζ'	0.077
d_0	1.43 $\times 10^5$ s^{-2}
d_1	-212.1 s^{-1}
d_2	36.2 s^{-1}
δ_x	10.21 mm
δ_y	-16.52 mm
δ_z	19.31 mm
D_e	39.91 mm

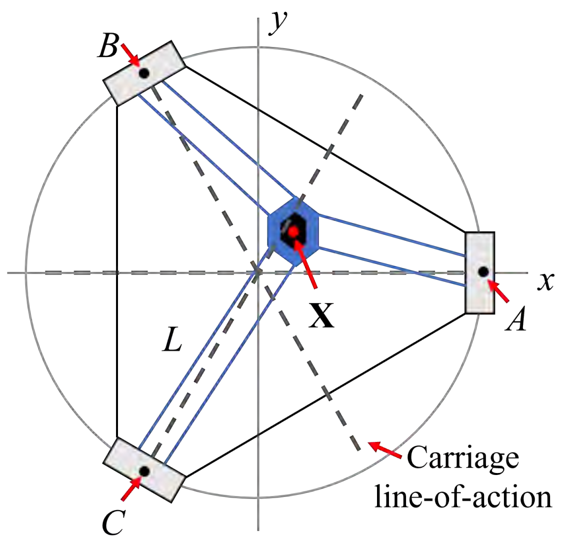


Figure 3.11: Overhead view of the delta 3D printer showing the (x, y)-coordinate locations of carriages A, B, and C, the end-effector's position in task space $\mathbf{X}$, and the length of the forearms L. End-effector motion along a carriage's line-of-action results in significant change to the carriage dynamics.

Then, we used the identified model in Sections 3.4.1 and 3.4.2 to predict the same FRFs. Figures 3.12-3.14 show the predicted and measured FRF comparisons for carriages A, B, and C, respectively. In the plots, we compare the measurements to each other as well as the dynamics at $(x, y) = (0, 0)$. The major trends across the observed frequency range of the predicted and measured FRFs are similar across the sampled positions. Note that the dynamics of each carriage are different because of the asymmetric mass distribution of the end-effector. From Table 3.1 and Figure 3.11, note that δ_y is negative, meaning that the center of mass is positioned closer to carriages A and C (and further from carriage B) causing them to hold a larger proportion of the end-effector's mass than carriage B. This phenomenon is borne out in Figs. 3.12-3.14 as carriages A and C show higher magnitude in the lower frequency mode at $(0, 0)$, while carriage B shows higher magnitude in the higher frequency mode.

Notably, the variation of the FRF at the position furthest along each carriage's line of action is captured by the predictions, highlighting the model's ability to capture position dependence. To quantify the similarity between the predicted and measured data, we use an error based metric. Since our FRFs are not linear with respect to frequency, common methods to quantify the goodness of fit, such as correlation coefficients, may be misleading. Table 3.2 reports the mean absolute percentage accuracy (μ_{acc}) of the predicted model, which can be thought of as how close the predicted model is to the actual measurement at each frequency. It is defined as the complement of the mean absolute percentage error (μ_{err}):

$$\mu_{acc} = 100 - \mu_{err} \tag{3.79}$$

$$\mu_{acc} = 100 - \left\{ \frac{1}{n} \sum_{f=f_1}^{f=f_n} \frac{|M_f - P_f|}{|M_f|} \cdot 100 \right\} \tag{3.80}$$

where f is the frequency, M_f are the measured data, P_f are the data from the predicted model, and $n = 247$ is the number of measured frequency points from $f_1 = 2$ Hz to $f_{247} = 125$ Hz, spaced in 0.5 Hz intervals. Note that when the prediction is perfect (i.e., $P_f = M_f$), $\mu_{err} = 0$ and $\mu_{acc} = 100\%$. As seen in Figs. 3.12-3.14 and Table 3.2, our prediction model often has different magnitudes than the measured data at the same frequency. This indicates that there are disturbances not captured by the model. However, the average mean absolute percentage accuracy (magnitude and phase, respectively) for carriages A (67.7% and 80.0%), B (78.1% and 82.3%), and C (68.7% and 84.4%), indicate reasonable prediction accuracy of the model. Further discussion of unmodeled disturbances and potential sources of error is provided in the following section.

Although our magnitude predictions are inaccurate for some positions, especially positions that are not along the respective carriage's line of action, the frequencies at which the modes occur for each position are predicted with reasonable accuracy, which suggests that our estimates of mass and stiffness parameters are close to the true values. Sources of error in the model include: (1) the CAD model used to

Table 3.2: Goodness of fit between predicted and measured FRFs via mean absolute percentage accuracy (μ_{acc})

$\mu_{acc}[\%]$	$(x, y) = (0, 0)$ mm		$(-80, 0)$ mm		$(40, -69)$ mm		$(40, 69)$ mm		Avg.	
	mag.	phase	mag.	phase	mag.	phase	mag.	phase	mag.	phase
A-to-A	70.1	84.7	72.6	86.0	54.6	81.6	73.6	67.7	67.7	80.0
B-to-B	73.1	88.8	79.4	86.6	81.1	62.2	78.7	91.5	78.1	82.3
C-to-C	82.5	86.5	82.1	89.8	43.7	84.7	66.5	76.7	68.7	84.4

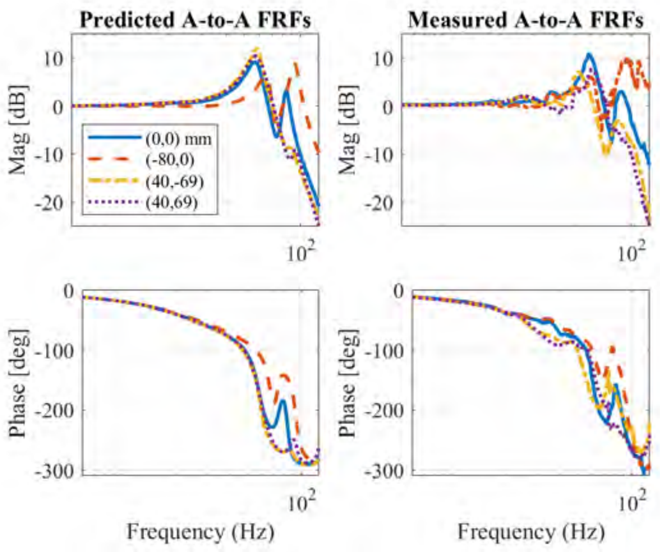


Figure 3.12: Frequency response functions of the A-to-A position at $(x, y) = (0, 0)$, $(-80, 0)$, $(40, -69)$, and $(40, 69)$ ($z = 30$ mm for all) predicted with the linearized joint space FRFs from Eq. (3.37) (left) and measured at carriage A of the Monoprice Delta Pro 3D printer (right).

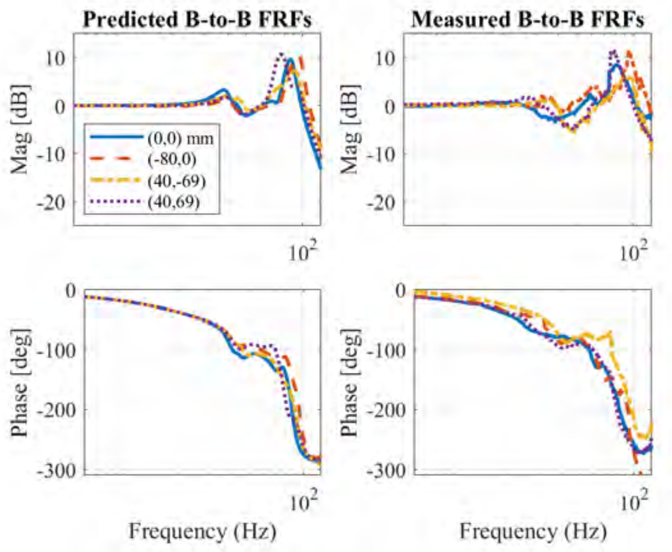


Figure 3.13: Frequency response functions of the B-to-B position at $(x, y) = (0, 0)$, $(-80, 0)$, $(40, -69)$, and $(40, 69)$ ($z = 30$ mm for all) predicted with the linearized joint space FRFs from Eq. (3.37) (left) and measured at carriage B of the Monoprice Delta Pro 3D printer (right).

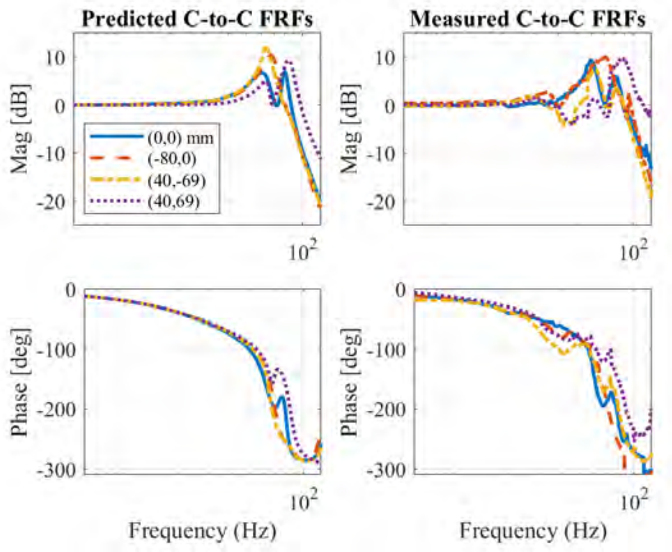


Figure 3.14: Frequency response functions of the C-to-C position at $(x, y) = (0, 0)$, $(-80, 0)$, $(40, -69)$, and $(40, 69)$ ($z = 30$ mm for all) predicted with the linearized joint space FRFs from Eq. (3.37) (left) and measured at carriage C of the Monoprice Delta Pro 3D printer (right).

Table 3.3: Percent error reduction compared to baseline model measured at (0,0)

% error reduction	$(x, y) = (-80, 0)$ mm		$(40, -69)$ mm		$(40, 69)$ mm	
	mag.	phase	mag.	phase	mag.	phase
A-to-A	35.2	60.4	13.1	-19.8	50.6	28.6
B-to-B	26.9	5.8	31.9	14.2	1.1	21.3
C-to-C	19.4	17.6	-12.5	11.8	34.3	23.4

determine the center of mass location, which is difficult to design perfectly accurate, and (2) friction and damping, which is notoriously difficult to model. In the CAD model, we assume that each component of the end-effector has uniform distribution of mass across its volume. However, we know that some components, like the extruder motor, are composed of various metal parts (e.g., aluminum, iron, etc.) with different densities that affect the distribution of mass across their volume. Accounting for such detail is cumbersome because individual components would need to be disassembled and reassembled. We endeavored to be as accurate as possible, while ensuring that our methodology can be replicated without significant difficulty. However, we found that changes of a few millimeters in the end-effector center of mass location (especially in the z-direction) could significantly influence the magnitude and phase of the predicted FRFs. Secondly, the various joint connections between the end-effector, forearms, and carriages create friction that is difficult to capture via our least squares identification methodology.

The proposed model is intended for use in feedforward control of the delta 3D printer. Without this predictive model, an alternative approach for the control designer is to choose one baseline model with which to implement a model-based controller for the printer across the entire workspace. A reasonable choice for the baseline model is the measured FRF when the end-effector is positioned at $(x, y) = (0, 0)$ mm. To compare this alternative to using the predictive model, we study the percentage of reduction in model error attained by using the predictive model. Using μ_{err} defined in Eq. (3.80), the percentage error reduction can be defined as

$$\% \text{ error reduction} = \frac{\mu_{err,B} - \mu_{err,P}}{\mu_{err,B}} \cdot 100 \tag{3.81}$$

where $\mu_{err,B}$ and $\mu_{err,P}$ are the mean absolute percentage errors of the baseline and predicted model, respectively. The expression for $\mu_{err,P}$ is identical to μ_{err} in Eq. (3.80), but for $\mu_{err,B}$, the baseline model is substituted for the prediction model, P_f. Table 3.3 reports the percentage error reduction when the predictive model is used to represent the measured FRFs at $(x, y) = (-80, 0)$, $(40, -69)$ and $(40, 69)$ mm instead of using the baseline model. From Figs. 3.12-3.14 and Table 3.3, note that in all but two instances the predictive model predicts the magnitude and phase of the measured model more accurately than the baseline model. The baseline model has less magnitude or phase error compared to the predicted model at positions where the measured model and the baseline model have small differences (for the respective carriage). One expects there will be a few positions like this across the workspace. Importantly, at the position furthest along each carriage's line of action (diagonal entries in Table 3.3), the predicted FRFs reduce the magnitude error by over 30% compared to the baseline and reduce the phase error by at least 14% and up to about 60%. These results suggest that naively choosing a baseline model to use in model-based feedforward control would result in worse accuracy performance when compared to using the LPV model proposed in this thesis. In Chapter IV, we will see this phenomenon bear out when we compare control of the delta printer using the LPV model and the baseline model.

3.6 Summary

The delta 3D printer offers the potential for higher throughput compared to traditional serial-axis 3D printers. However, it has not benefited from the model-based feedforward vibration compensation methods that have improved the accuracy and speed of serial 3D printers because of the difficulty modeling delta's position-varying nonlinear dynamics. In this chapter, we present an efficient framework that uses RC to identify LPV models for delta 3D printers using a few measurements from only one position. The model is general which enables it to account for different mass distributions and dynamic models of the end-effector, which can have a significant impact on the model, as demonstrated from the measured data. Using RC was particularly convenient for the prismatic-joint delta robot because we can disassemble it easily to measure the dynamics different of sub-assemblies. It's likely that RC has not been studied as an approach for modeling the delta robot because most of the literature has focused on the rotary-joint delta robot, which has a complex assembly such that dividing it into sub-assemblies is not as obvious as it was for the prismatic-joint robot. However, with this new perspective, a similar approach can be used to study the model of the rotary-joint delta robot.

In this chapter, we presented the generalized model, described a procedure to identify its parameters, and demonstrated its efficacy using a commercial delta 3D printer, showing that the resulting model captures the position-dependent dynamic variations with reasonable accuracy. Additionally, at positions where the dynamics of the printer differ from the center (or baseline) model, we showed that the proposed model predicts the true dynamics with greater accuracy than the baseline model. In Chapter IV, we will implement a model-based controller on the delta 3D printer using the LPV model derived above.

CHAPTER IV

Model-based Feedforward Control of the Delta Robot

4.1 Overview

The model of the delta 3D printer identified in Chapter III can be used to design a model-based feedforward controller to suppress vibration and other motion-induced errors. In this chapter, we wish to implement an FBS feedforward controller using the linear parameter-varying (LPV) model to improve the accuracy of the delta robot during high-speed printing. FBS can theoretically improve the accuracy of delta 3D printers, but using its standard form as described in Section 2.3 is impractical due to the computational challenges of optimizing the controller in real-time. Recall that in Chapter II, we were forced to approximate the full LPV model of the H-frame by modeling motion errors as linear relationships between the x and y axes (and their LTI models). Furthermore, we approximated the H-frame's dynamics as decoupled, resulting in independent computation of the B-spline coefficients for each axis. Using these approximations, the B-splines could be filtered and inverted offline to enable fast computation of their coefficients online [5]. However, the delta 3D printer has a coupled kinematic chain and its LPV model cannot be decoupled. Hence, the delta model needs to be updated at each new position and the LPV FBS controller must be re-computed at each position, which can be computationally challenging for a real-time controller.

In this chapter, we propose techniques to mitigate the computational challenges of implementing FBS in real-time on the delta 3D printer. To do so, we address several bottlenecks of its application: first, we parameterize the position-dependent portions of the model offline to enable faster updates of the model online as the position of the delta printer changes; second, we compute the model at sampled points (instead of every point) along the given trajectory; and, third, we employ the QR matrix factorization method to compute the B-spline control points (i.e., coefficients) in FBS, which reduces the number of arithmetic operations necessary when compared to the standard pseudo-inversion.

This chapter is organized as follows: Section 4.2 describes the exact (computationally intensive) LPV implementation of the FBS approach on the delta printer and the simplifying techniques we propose to enable real-time control of delta 3D printers; Section 4.3 demonstrates the expected computational and accuracy benefits of our proposed approach through simulations and experiments; and Section 4.4 concludes the chapter, summarizing key insights.

4.2 Feedforward Control of Delta 3D Printer with Filtered B-Splines

4.2.1 Linear parameter varying FBS controller

The filtered B-splines (FBS) approach can be applied directly to the LPV delta model with a similar implementation to the one described in Section 2.4. The major differences are: (1) the delta robot has three joint axes we must control compared to two on the H-frame; and (2) we cannot decouple the delta robot's dynamics in the same way we did for the H-frame. To elaborate, the H-frame's y-axis position depends on the x-axis position, but not vice versa. Hence, we can compute the controlled x-axis trajectories independently and use the model-predicted x position to compute the controlled y-axis trajectories. On the delta robot, the dynamics are bidirectionally coupled in all three axes (e.g., carriage A's position depends on carriage B's position and vice versa). Hence, we expect the computational burden of the delta's FBS controller to be significantly higher than the H-frame's FBS controller.

To formulate the delta FBS controller, we first define $\mathbf{q}_{i_d} = [q_{i_d}(t_0)\ q_{i_d}(t_1)\ \cdots\ q_{i_d}(t_E)]^T$ as the entire $E+1$ discrete time steps of the desired trajectory of carriage i ($i \in \{A, B, C\}$ as in Section 3.3), which are processed in sliding windows with LPFBS. Assume that time t_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

$$\begin{bmatrix} q_{i_{dm}}(t_k) \\ q_{i_{dm}}(t_{k+1}) \\ \vdots \\ q_{i_{dm}}(t_{k+L_C}) \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_{m,m}(t_k) & \cdots & \phi_{m+n,m}(t_k) \\ \phi_{m,m}(t_{k+1}) & \cdots & \phi_{m+n,m}(t_{k+1}) \\ \vdots & \ddots & \vdots \\ \phi_{m,m}(t_{k+L_C}) & \cdots & \phi_{m+n,m}(t_{k+L_C}) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} p_{i,m} \\ \vdots \\ p_{i,m+n} \end{bmatrix}}_{\mathbf{p}_{i,\mathrm{C}}} \tag{4.1}$$

where the 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, $\phi_{j,m}(t)$ are the real-valued basis functions [74], $j = m, m+1, ..., m+n$, $\mathbf{p}_{i,\mathrm{C}}$ is a vector of $n+1$ unknown coefficients, $t_k = kT_s$ is the current time, and T_s is the sampling time.

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

$$\mathbf{q}_{dm,\mathrm{C}} = \begin{bmatrix} \mathbf{q}_{A_{dm},\mathrm{C}} \\ \mathbf{q}_{B_{dm},\mathrm{C}} \\ \mathbf{q}_{C_{dm},\mathrm{C}} \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Phi & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Phi \end{bmatrix}}_{\mathbf{N}_\mathrm{C}} \underbrace{\begin{bmatrix} \mathbf{p}_{A,\mathrm{C}} \\ \mathbf{p}_{B,\mathrm{C}} \\ \mathbf{p}_{C,\mathrm{C}} \end{bmatrix}}_{\mathbf{p}_\mathrm{C}} \tag{4.2}$$

As in Chapter II, our objective is to minimize the tracking error which is defined as

$$\bar{\mathbf{e}} = \mathbf{q}_d - \mathbf{q} = \mathbf{q}_d - \bar{\mathbf{N}}\bar{\mathbf{p}} \Leftrightarrow \begin{bmatrix} \bar{\mathbf{e}}_\mathrm{P} \\ \bar{\mathbf{e}}_\mathrm{C} \\ \bar{\mathbf{e}}_\mathrm{F} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{d,\mathrm{P}} \\ \mathbf{q}_{d,\mathrm{C}} \\ \mathbf{q}_{d,\mathrm{F}} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{N}}_\mathrm{P} & \mathbf{0} & \mathbf{0} \\ \bar{\mathbf{N}}_\mathrm{PC} & \bar{\mathbf{N}}_\mathrm{C} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{N}}_\mathrm{CF} & \bar{\mathbf{N}}_\mathrm{F} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}_\mathrm{P} \\ \bar{\mathbf{p}}_\mathrm{C} \\ \bar{\mathbf{p}}_\mathrm{F} \end{bmatrix} \tag{4.3}$$

where subscripts P and F denote the past and future windows, respectively, 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 (see [5]). Note, from Eq. (4.3), that the output carriage motion of the current window is given by

$$\mathbf{q}_\mathrm{C} = \bar{\mathbf{N}}_\mathrm{C}\bar{\mathbf{p}}_\mathrm{C} + \bar{\mathbf{N}}_\mathrm{PC}\bar{\mathbf{p}}_\mathrm{P} \tag{4.4}$$

Using local least squares, the optimal coefficients of the current window can be computed as

$$\bar{\mathbf{p}}_\mathrm{C} = (\bar{\mathbf{N}}_\mathrm{C}^T\bar{\mathbf{N}}_\mathrm{C})^{-1}\bar{\mathbf{N}}_\mathrm{C}\Big(\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\bar{\mathbf{p}}_\mathrm{P}\Big) \tag{4.5}$$

$$= \bar{\mathbf{N}}_\mathrm{C}^\dagger\Big(\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\bar{\mathbf{p}}_\mathrm{P}\Big) \tag{4.6}$$

where $\bar{\mathbf{p}}_\mathrm{P}$ denotes the coefficients calculated in the previous window. From Eqs. (4.3) and (4.6), the reader should note that the windows are designed to overlap to ensure continuity. Additionally, n B-spline coefficients are computed in each window but only $n_{up} \leq n$ are updated [5].

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 calculating the optimal coefficients in every window using Eq. (4.6) (see Appendix B). However, for position-dependent systems like the delta 3D printer, filtering and inverting the (large) B-splines matrix must be done online, which is computationally challenging to do at a fast enough rate for real-time requirements on many hardware processors. 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 improved accuracy performance provided by FBS when constrained by the machine's computation and memory capabilities.

4.2.2 Selecting a parameterized model for B-splines filtering

Consider the problem of filtering each column of $\mathbf{N}$ through the delta printer's dynamic model $\mathbf{G}(s)$ from Section 3.3 (reproduced below):

$$\mathbf{G}(s) = \underbrace{\left[\mathbf{I} - \mathbf{G}_{Fq}(s) \begin{bmatrix} \bar{\mathbf{J}}_A^T \mathbf{P}_A \\ \bar{\mathbf{J}}_B^T \mathbf{P}_B \\ \bar{\mathbf{J}}_C^T \mathbf{P}_C \end{bmatrix} \mathbf{W}(s) \bar{\mathbf{J}}\right]^{-1}}_{\mathbf{G}_J^{-1}(s)} \mathbf{G}_{q_d}(s). \tag{4.7}$$

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

$$G_{J,AA}^{-1}(s) = \frac{b_{AA}(x, y, z, q_A, q_B, q_C, s)}{a(x, y, z, q_A, q_B, q_C, s)} \tag{4.8}$$

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 ($G_{J,BA}^{-1}(s)$, $G_{J,CA}^{-1}(s)$, $G_{J,AB}^{-1}(s)$, and so on) can be expressed similarly with $b_{BA}(\cdot)$, $b_{CA}(\cdot)$, $b_{AB}(\cdot)$, and so on, since all transfer functions share the same denominator $a(\cdot)$. These parameterized transfer functions enable fast computations of the coefficients of $\mathbf{G}_J^{-1}(s)$ 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 $\mathbf{G}_{q_d}(s)$ 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. (4.8) to obtain

$$\begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}} \\ \bar{\mathbf{N}}_{\mathrm{CF}} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{AA}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{AA}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{BA}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{BA}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{CA}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{CA}} \end{bmatrix} \\ \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{AB}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{AB}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{BB}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{BB}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{CB}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{CB}} \end{bmatrix} \\ \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{AC}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{AC}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{BC}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{BC}} \end{bmatrix} & \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{C}_{CC}} \\ \bar{\mathbf{N}}_{\mathrm{CF}_{CC}} \end{bmatrix} \end{bmatrix} \tag{4.9}$$

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 $G_{J,AA}^{-1}(s)$, and so on for the other blocks of the matrix.

As described above, the current and future windows are overlapped for continuity during computation, but only L_C points are updated during each sequence [5]. 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 use quick-access memory for other important functions like storing the print trajectory.

To prevent such deleterious effects, we select one of the first L_C points from each window at which a time-invariant transfer function $G^{-1}_{J,(\cdot\cdot)}(s)$ is computed, which reduces the time and space complexity to $O(1)$ and $O(u_a)$, respectively. This trade-off is reasonable because: (a) only the first L_C points will be commanded, and (b) 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 of the single point selection described above, 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 4.3, we demonstrate through simulations and experiments that selecting one point in each window does not significantly degrade the performance of the FBS controller: especially when a technique for smoothly switching between windows is added, as discussed in the following subsection.

4.2.3 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 feedforward controller's predicted output trajectories. In the standard FBS approach, continuity of the predicted trajectories is preserved by using the same LTI model for every window.

To demonstrate what happens in the LPV case, suppose we used model 1 for the

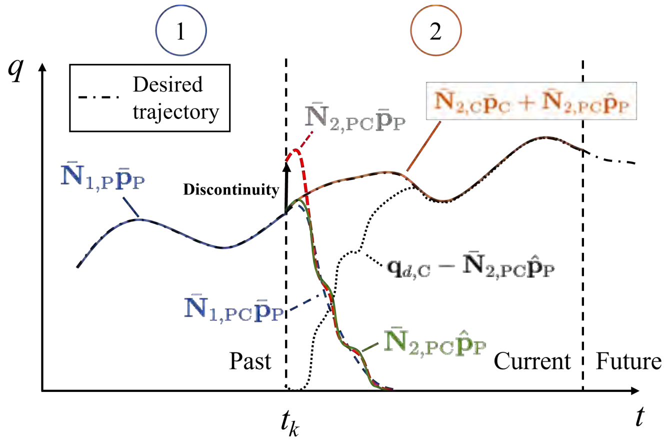

1
2
q
Desired trajectory
$\bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_\mathrm{P}$
$\bar{\mathbf{N}}_{2,\mathrm{C}}\bar{\mathbf{p}}_\mathrm{C} + \bar{\mathbf{N}}_{2,\mathrm{PC}}\hat{\mathbf{p}}_\mathrm{P}$
$\bar{\mathbf{N}}_{1,\mathrm{P}}\bar{\mathbf{p}}_\mathrm{P}$
Discontinuity
$\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_{2,\mathrm{PC}}\hat{\mathbf{p}}_\mathrm{P}$
$\bar{\mathbf{N}}_{1,\mathrm{PC}}\bar{\mathbf{p}}_\mathrm{P}$
$\bar{\mathbf{N}}_{2,\mathrm{PC}}\hat{\mathbf{p}}_\mathrm{P}$
Past
Current
Future
t_k
t

Figure 4.1: Illustration of the switching compensation technique to maintain continuity described in Sec. 4.2.3. 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 $\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_\mathrm{PC}\hat{\mathbf{p}}_\mathrm{P}$.

past window and update it to model 2 for the current window as shown in Fig. 4.1. When we switch from model 1 to model 2, the prediction of the output trajectory in the current window will be different depending on if we use model 1 (i.e., $\bar{\mathbf{N}}_{1,\mathrm{PC}}\bar{\mathbf{p}}_\mathrm{P}$ as the prediction) or model 2 (i.e., $\bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_\mathrm{P}$ as 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 (i.e., $\bar{\mathbf{N}}_{1,\mathrm{P}}$).

To resolve this discrepancy, we compute an approximate prediction by generating a set of approximate control points that ensure continuity with the output from the past window. The approximate control points, $\hat{\mathbf{p}}_\mathrm{P}$, are selected to minimize the difference between the new prediction and the original prediction while preserving continuity. We write the optimization problem as

$$\begin{aligned} \hat{\mathbf{p}}_{\mathrm{P}} = \arg\min_{\hat{\mathbf{p}}_{\mathrm{P}}} \quad & \|\bar{\mathbf{N}}_{2,\mathrm{PC}}\hat{\mathbf{p}}_{\mathrm{P}} - \bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}\|_2^2 \\ s.t. \quad & \bar{N}^T_{2,\mathrm{PC}}(t_k)\hat{\mathbf{p}}_{\mathrm{P}} = \bar{N}^T_{1,\mathrm{PC}}(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \\ & \bar{N'}^T_{2,\mathrm{PC}}(t_k)\hat{\mathbf{p}}_{\mathrm{P}} = \bar{N'}^T_{1,\mathrm{PC}}(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \end{aligned} \tag{4.10}$$

where $\bar{N}^T_{1,\mathrm{PC}}(t_k)$ and $\bar{N}^T_{2,\mathrm{PC}}(t_k)$ are the first rows of $\bar{\mathbf{N}}_{1,\mathrm{PC}}$ and $\bar{\mathbf{N}}_{2,\mathrm{PC}}$ in the window, respectively, and $\bar{N'}^T_{1,\mathrm{PC}}(t_k)$ and $\bar{N'}^T_{2,\mathrm{PC}}(t_k)$ 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 $\bar{\mathbf{N}}_{2,\mathrm{PC}}$). Note that the products

$$\bar{N}^T_{1,\mathrm{PC}}(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \quad \text{and} \quad \bar{N}^T_{2,\mathrm{PC}}(t_k)\hat{\mathbf{p}}_{\mathrm{P}} \tag{4.11}$$

represent positions at the window boundary, and

$$\bar{N'}^T_{1,\mathrm{PC}}(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \quad \text{and} \quad \bar{N'}^T_{2,\mathrm{PC}}(t_k)\hat{\mathbf{p}}_{\mathrm{P}} \tag{4.12}$$

and represent velocities at the boundary. Additional kinematic constraints, such as acceleration and jerk, can be included in the optimization problem from Eq. (4.10) by taking additional derivatives of the B-splines as described in [74] and [81]. More kinematic constraints leads to smoother transitions when the dynamics change significantly or when the window length is long. In our simulations of the machine used in Section 4.3, we found that position and velocity constraints led to similar tracking accuracy when compared to optimizing Eq. (4.10) with acceleration and jerk constraints. Hence, our implementation only uses the position and velocity constraints for Eq. (4.10).

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

$$\bar{\mathbf{p}}_{\mathrm{C}} = \arg\min_{\bar{\mathbf{p}}_{\mathrm{C}}} \left[\Big((\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_{\mathrm{PC}}\hat{\mathbf{p}}_{\mathrm{P}}) - \bar{\mathbf{N}}_{\mathrm{C}}\bar{\mathbf{p}}_{\mathrm{C}} \Big)^T \Big((\mathbf{q}_{d,\mathrm{C}} - \bar{\mathbf{N}}_{\mathrm{PC}}\hat{\mathbf{p}}_{\mathrm{P}}) - \bar{\mathbf{N}}_{\mathrm{C}}\bar{\mathbf{p}}_{\mathrm{C}} \Big) \right]. \tag{4.13}$$

Solving the constrained optimization problem in Eq. (4.10) in real-time could be challenging. Similarly, we can speed up the computation of Eq. (4.13) by using a least squares optimization method that is faster than the pseudo-inverse. To ensure fast computations, we employ the LU and QR factorization methods for solving Eqs. (4.10) and (4.13), respectively, as discussed in the following subsection.

4.2.4 Command generation with LU and QR Factorization

The optimization problem in Eq. (4.10) 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. (4.10):

$$\mathbf{A} = \bar{\mathbf{N}}_{2,\mathrm{PC}}, \quad \mathbf{b} = \bar{\mathbf{N}}_{2,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}} \tag{4.14}$$

$$\mathbf{C} = \begin{bmatrix} \bar{N}^T_{2,\mathrm{PC}}(t_k) \\ \bar{N}'^T_{2,\mathrm{PC}}(t_k) \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} \bar{N}^T_{1,\mathrm{PC}}(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \\ \bar{N}'^T_{1,\mathrm{PC}}(t_k)\bar{\mathbf{p}}_{\mathrm{P}} \end{bmatrix} \tag{4.15}$$

Then, the problem can be written as

$$\begin{aligned} \hat{\mathbf{p}}_{\mathrm{P}} = \arg\min_{\hat{\mathbf{p}}_{\mathrm{P}}} \quad & \|\mathbf{A}\hat{\mathbf{p}}_{\mathrm{P}} - \mathbf{b}\|_2^2 \\ s.t. \quad & \mathbf{C}\hat{\mathbf{p}}_{\mathrm{P}} = \mathbf{d} \end{aligned} \tag{4.16}$$

To solve the optimization problem, we make two assumptions:

1. The stacked matrix
$$\begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix} \tag{4.17}$$
has linearly independent columns; and

2. $\mathbf{C}$ has linearly independent rows.

As discussed in [24], 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,

$$\mathcal{L}(\hat{\mathbf{p}}_{\mathrm{P}}, \lambda) \triangleq \frac{1}{2}\|\mathbf{A}\hat{\mathbf{p}}_{\mathrm{P}} - \mathbf{b}\|_2^2 + \lambda^T(\mathbf{C}\hat{\mathbf{p}}_{\mathrm{P}} - \mathbf{d}), \tag{4.18}$$

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

$$\begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{p}}_{\mathrm{P}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{A}^T\mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{4.19}$$

Note that the matrix
$$\begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \tag{4.20}$$
is nonsingular when the above assumptions hold. Therefore, the linear equation given by Eq. (4.19) can be efficiently solved with LU factorization [75].

We can also use QR factorization to efficiently compute the control points. Using the pseudo-inverse to solve the optimization problem in Eq. (4.13) requires the following number of floating-point operations (flops) for each step [75] (listed in a cumulative fashion):

$$\bar{\mathbf{N}}_{\mathrm{C}}^T\bar{\mathbf{N}}_{\mathrm{C}} \,:\, L_C n^2 \text{ flops} \tag{4.21}$$

$$(\bar{\mathbf{N}}_{\mathrm{C}}^T\bar{\mathbf{N}}_{\mathrm{C}})^{-1} \,:\, n^3 + L_C n^2 \text{ flops} \tag{4.22}$$

$$\bar{\mathbf{N}}_{\mathrm{C}}\tilde{\mathbf{q}}_{d,\mathrm{C}} \,:\, n^3 + L_C n^2 + 2L_C n \text{ flops} \tag{4.23}$$

$$(\bar{\mathbf{N}}_{\mathrm{C}}^T\bar{\mathbf{N}}_{\mathrm{C}})^{-1}\Big(\bar{\mathbf{N}}_{\mathrm{C}}\tilde{\mathbf{q}}_{d,\mathrm{C}}\Big) \,:\, n^3 + L_C n^2 + 4L_C n \text{ flops.} \tag{4.24}$$

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

$$\bar{\mathbf{N}}_{\mathrm{C}} = \mathbf{QR} \tag{4.25}$$

with the modified Gram Schmidt algorithm [82], 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. (4.13) can be written as

$$\mathbf{R}\bar{\mathbf{p}}_{c,\mathrm{C}} = \mathbf{Q}^T\tilde{\mathbf{q}}_{d,\mathrm{C}} \tag{4.26}$$

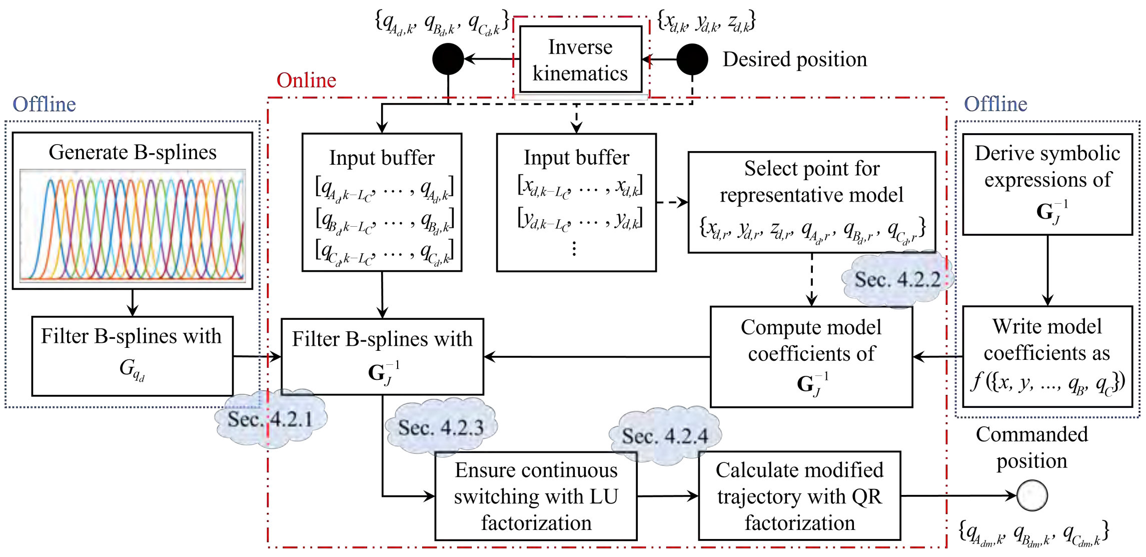
which can be solved using backward substitution. The number of operations required using this method are

$$\bar{\mathbf{N}}_{\mathrm{C}} = \mathbf{QR} \,:\, L_C n^2 \text{ flops} \tag{4.27}$$

$$\mathbf{w} = \mathbf{Q}^T\tilde{\mathbf{q}}_{d,\mathrm{C}} \,:\, L_C n^2 + 2L_C n \text{ flops} \tag{4.28}$$

$$\mathbf{R}\bar{\mathbf{p}}_{c,\mathrm{C}} = \mathbf{w} \,:\, n^2 + L_C n^2 + 2L_C n \text{ flops.} \tag{4.29}$$

Note, from the last step in each method, that the QR factorization solution is more efficient to compute than the pseudo-inverse, which will be validated in the following section.



$\{q_{A_d,k}, q_{B_d,k}, q_{C_d,k}\}$
Inverse kinematics
$\{x_{d,k}, y_{d,k}, z_{d,k}\}$
Desired position
Online
Offline
Generate B-splines
Filter B-splines with G_{q_d}
Input buffer
$[q_{A_d,k-L_C}, \cdots, q_{A_d,k}]$
$[q_{B_d,k-L_C}, \cdots, q_{B_d,k}]$
$[q_{C_d,k-L_C}, \cdots, q_{C_d,k}]$
Input buffer
$[x_{d,k-L_C}, \cdots, x_{d,k}]$
$[y_{d,k-L_C}, \cdots, y_{d,k}]$
Select point for representative model
$\{x_{d,r}, y_{d,r}, z_{d,r}, q_{A_d,r}, q_{B_d,r}, q_{C_d,r}\}$
Offline
Derive symbolic expressions of $\mathbf{G}_J^{-1}$
Sec. 4.2.2
Filter B-splines with $\mathbf{G}_J^{-1}$
Compute model coefficients of $\mathbf{G}_J^{-1}$
Write model coefficients as $f(\{x, y, ..., q_B, q_C\})$
Sec. 4.2.1
Sec. 4.2.3
Sec. 4.2.4
Commanded position
Ensure continuous switching with LU factorization
Calculate modified trajectory with QR factorization
$\{q_{A_{dm},k}, q_{B_{dm},k}, q_{C_{dm},k}\}$

Figure 4.2: 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. G_{q_d} as described Section 4.2.1 (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 is used in this work. 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 Section 4.2.2). 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 (Section 4.2.3) and calculate the modified trajectory (Section 4.2.4).

Remark 4.2(a): The techniques described in Sections 4.2.2, 4.2.3, and 4.2.4 were implemented in Matlab Simulink to control the MP Delta Pro 3D printer, similar to the implementation of the standard FBS approach in [5]. Standard FBS controllers have also been implemented on standalone micro-controllers for 3D printers by a startup company called Ulendo, which indicates that our techniques can be applied to similar micro-controllers. As a visual representation, Fig. 4.2 shows a flowchart of the code implementation. In the following section, simulations and experiments are used to characterize and validate the proposed advantages of efficient computation and improved tracking accuracy.

4.3 Simulations and Experimental Validation

4.3.1 Simulation validation

In this subsection, we simulate the identified dynamics of the MP Delta Pro 3D printer (from Chapter III) with a variety of controllers to compare the computation time and accuracy of the controller proposed in Section 4.2 to that of potential alternatives. We evaluate the performance of following controllers:

(a) an LPV FBS controller where the transfer functions are computed in matrix form using Eq. (3.37) at every point in each window (i.e., the exact LPV controller) and the coefficients are calculated with the pseudo-inverse;

(b) a controller that is the same as controller (a), except the transfer functions are computed using the parameterized model from Section 4.2.2;

(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 Section 4.2.3;

(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 as described in Section 4.2.4.

By adding the modifications one at a time, we can distinguish the effects on computational efficiency and motion accuracy of each modification. Each controller's performance is also compared to a baseline controller—the standard FBS controller that uses an LTI model measured at $(x, y) = (0, 0)$ mm for the carriages. We use $(0, 0)$ because the model at the origin is a reasonable choice for the LTI model for FBS when the control designer does not have a model for the position-dependent dynamics.

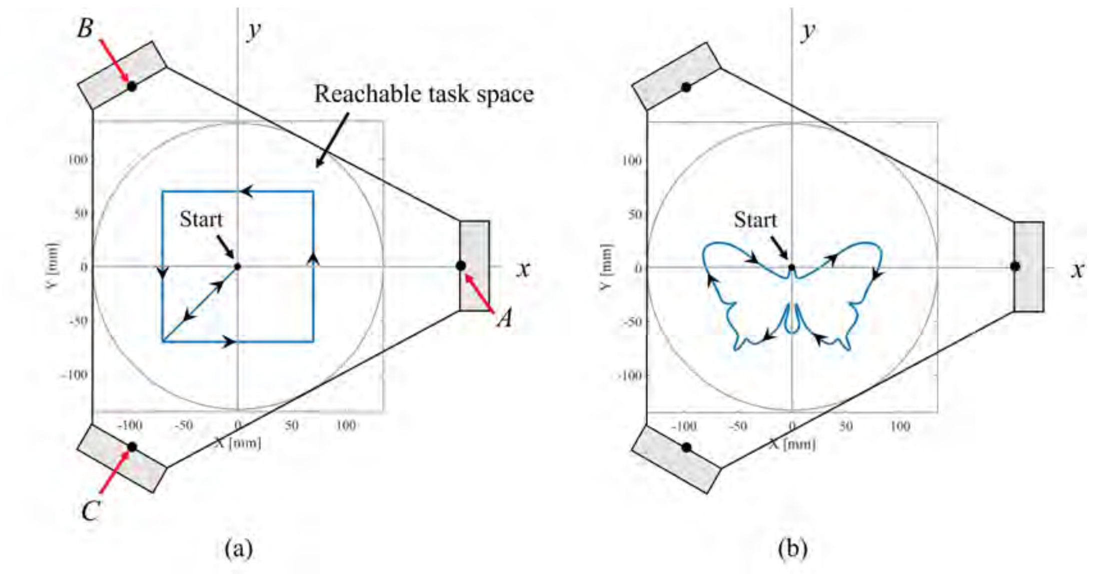


Figure 4.3: Trajectories of (a) a square and (b) a butterfly used for simulations overlaid on the task space of the delta 3D printer. The square has a side length of 140 mm and is centered at the origin. The butterfly spans $x \in [-82, 82]$ mm and $y \in [-77, 23]$ mm. Both trajectories have with a maximum motion speed of 150 mm/s and a maximum acceleration of 20 m/s^2.

The simulations are conducted using two trajectories:

1. the trajectory of a square shown in Fig. 4.3(a) with a side length of 140 mm and its center at the origin; and
2. the trajectory of a butterfly shown in Fig. 4.3(b), which spans $x \in [-83, 83]$ mm and $y \in [-77, 23]$ mm.

Both trajectories have a maximum speed of 150 mm/s and a maximum acceleration of 20 m/s^2. The square is selected to emphasize straight-line motions and sharp 90° turns which are prone to vibration. The butterfly trajectory is selected to emphasize curved motions with tight corners that are prone to contour errors. Each trajectory lasts about 5 seconds with a sample time of 1 ms. To simulate the system's response, the lifted system representation (LSR) of $\mathbf{G}(s)$ is computed using the known trajectory points. 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

Table 4.1: Simulation results comparing the total computation time and accuracy of different controllers for generating modified trajectories of the square and butterfly. Results are listed as [Square / Butterfly].

	Computation Time	RMS Contour Error	Accuracy improvement from baseline
(*) Baseline LTI, standard FBS controller	0.014 / 0.044 s	5.94 / 11.13 μm	–
(a) Matrix TFs, all points, pseudo-inverse	671.55 / 820.06 s	0.32 / 0.38 μm	94.6 / 96.6%
(b) Parameterized TFs, all points, pseudo-inverse	181.16 / 226.30 s	0.32 / 0.38 μm	94.6 / 96.6%
(c) Parameterized TFs, single point, pseudo-inverse	8.17 / 9.86 s	0.64 / 3.21 μm	89.3 / 71.1%
(d) Same as above with switching compensation	12.28 / 13.46 s	0.34 / 0.53 μm	94.3 / 95.3%
(e) Same as above with QR factorization	9.19 / 9.65 s	0.34 / 0.53 μm	94.3 / 95.3%

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 length $L_C = 196$ points, number of B-spline coefficients $n = 44$, and number of updated coefficients $n_{up} = 22$. The window length is determined by the amount of time required for the impulse response of the transfer function used for filtering (i.e., infinite impulse response (IIR) filter) to settle close to zero. Since the transfer functions vary with position, we select the window length for the delta 3D printer using a one-time offline procedure: we construct a grid of positions in the reachable workspace that are 5 mm apart, compute the impulse response for the transfer function at each position, and use the worst-case settling time to determine the window length. Since the time domain of influence of an IIR filter is infinite, the user must select a truncation threshold (e.g., 10^{-3}) such that each window is long enough for the impulse response to *almost* settle to zero. The number of B-spline coefficients are then computed from the window length, after the user selects the knot vector spacing, as described in [5]. The knot vector spacing of the B-splines can be thought of as the frequency bandwidth the machine is expected to track and should be set at a higher frequency than any resonance mode of the measured or predicted FRFs of the machine.

Table 4.2: Mean computation time per window for generating modified trajectories using each controller. Results are listed as [Square / Butterfly].

	Mean computation time per window
Baseline controller	4.57×10^{-4} / 7.04×10^{-4} s
Controller (a)	24.87 / 35.52 s
Controller (b)	6.71 / 12.27 s
Controller (c)	0.30 / 0.41 s
Controller (d)	0.46 / 0.52 s
Controller (e)	0.38 / 0.41 s

The simulations are conducted 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 the root-mean-square (RMS) contour error for each trajectory and each controller is reported in Table 4.1. The percent difference of the RMS contour error of other controllers compared to the RMS contour error of the baseline controller simulation is also reported as "Accuracy improvement from baseline" in Table 4.1. Time-series plots of the contour error comparisons are shown in detail in Figs. 4.4 and 4.5 for the square and butterfly trajectories, respectively. For the butterfly trajectory, the RMS contour error of the baseline controller is 11.13 μm and the trajectory is computed in 44 ms since the filtering and inversion is completed offline. However, note that the RMS error of the exact LPV model is almost 30 times less than the baseline at 0.38 μm, although the computation of the matrix model online is much longer (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 20x (to $\sim$10 s) but at a cost of about 10x increase in RMS contour error (to 3.21 μm). The accuracy is improved to only be about 1.3x worse than the exact LPV model (about 0.5 μm) when the switching compensation is implemented, which indicates that we can significantly reduce the computation time while maintaining relatively high accuracy

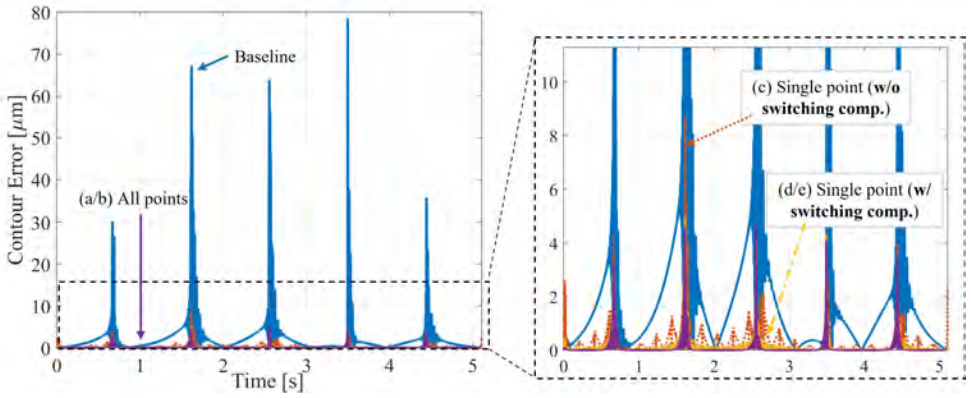


Figure 4.4: Contour error of the modified "square" trajectory generated by the baseline controller (solid blue line) and controllers (a/b) using all trajectory points (solid purple line), (c) a single point without switching compensation (dotted red line), and (d/e) a single point with switching compensation (dash-dotted yellow line).

with controller (e). The results for the square trajectory follow a similar trend to the butterfly trajectory. The baseline controller starts with less absolute contour error compared to the butterfly because the trajectory is mostly composed of straight lines which are less prone to contour errors when compared to the butterfly's curved paths. Hence, we have less relative loss of accuracy using controller (c) for the square (∼5% from 94.6% to 89.3%) when compared to the butterfly (∼25% from 96.6% to 71.1%). However, we see a similar order of magnitude reduction in computation time from controller (a) to controllers (c) and (e) between the two trajectories, which indicates that the computational benefit of the proposed methodology is trajectory-agnostic. Furthermore, Table 4.2 reports the average (mean) computation time per window using each controller for both trajectories. Note that there is a similar order of magnitude reduction in computation time when comparing the parameterized, single-point controller (controller (c)) to the exact LPV controller (controller (a)).

Shifting focus to the accuracy, we note a spike in the contour error of controller (c) around 2 seconds into the butterfly motion in Fig. 4.5. The spike represents a difficult portion of the butterfly trajectory—the bottom right of the butterfly wing—where a switch in models occurs for controllers (c)-(e). Here, the points are close to

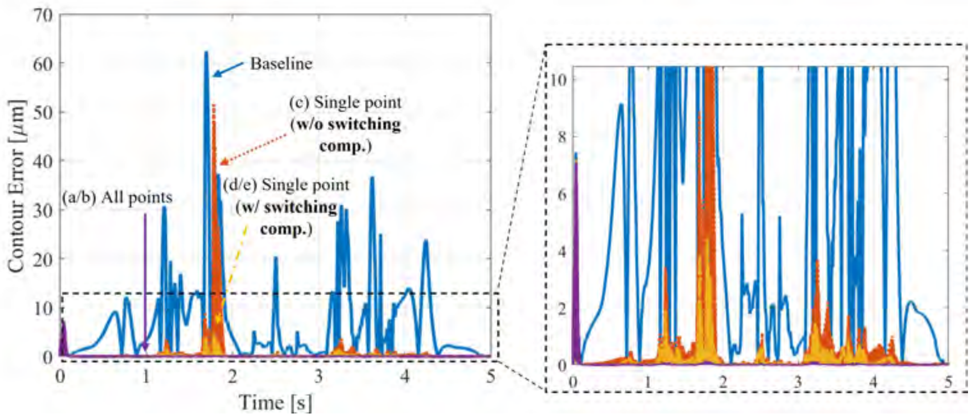


Figure 4.5: Contour error of the modified "butterfly" trajectory generated by the baseline controller (solid blue line) and controllers (a/b) using all trajectory points (solid purple line), (c) a single point without switching compensation (dotted red line), and (d/e) a single point with switching compensation (dash-dotted yellow line).

the far side of carriage B's line of action (see Figs. 3.11 and 4.3) which, as discussed in Section 3.5, 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). Similarly, controller (c) leads to small spikes in error throughout the square trajectory that are not present for controller (d) as shown in Fig. 4.4. The spikes are exacerbated when a switch in windows coincides with a 90° corner on the square, which can be seen about 1.7 seconds into the motion. Generally, the contour error increases for all single point controllers (c), (d), and (e) when compared to the all-point controllers ((a) and (b)), but they are worse for controller (c) which does not include switching compensation. Finally, note that the total computation time is reduced by 28% for the butterfly trajectory and 25% for the square trajectory using QR factorization instead of the pseudo-inversion (controller (e)). Overall, the accuracy of the single point approach is worse than using all the points in a window, but the overall accuracy improvement is acceptable given the (up to) 23x decrease in computation time compared to using the parameterized exact LPV controller (b), which is challenging to implement on hardware in real-time, as discussed in the following subsection on experiments.

4.3.2 Experimental validation

To study the impact of the proposed methodology on fabricated parts, we printed a "calibration cube"[1] and an extruded butterfly on the delta 3D printer using two control strategies: the baseline controller and controller (e) above. We focus most of our attention on the calibration cube since it is a well-known benchmark to characterize vibration in the 3D printing industry. Note that some layers of the cube's trajectory have square paths like those studied in the simulations (Section 4.3.1). The cube also contains letter indentations which create additional sources of accuracy error. The butterfly is printed to evaluate the impact of each controller on curved layer parts. Our aim is to demonstrate the utility of our contributions by comparing: (a) the 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 the proposed techniques.

We position the center of each part at the following locations: $(x, y) = (0, 0)$, $(-80, 0)$, $(40, -69)$, 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. As shown in Fig. 4.6, the position $(-80, 0)$ mm primarily tests variation in carriage A's dynamics, $(40, -69)$ mm primarily tests variation in carriage B's dynamics, and $(40, 69)$ mm primarily tests variation in carriage C's dynamics. The maximum speed and acceleration for both parts are 150 mm/s and 20 m/s^2, respectively. Both the baseline controller and controller (e) are implemented in Matlab Simulink, which sends the motion commands through a dSPACE MicroLabBox to Pololu DRV8825 stepper motor drivers to move the stepper motors on the delta 3D printer. Using the dSPACE hardware and Simulink interface, only controller (e) (of all the controllers in Section 4.3.1) compiles successfully

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

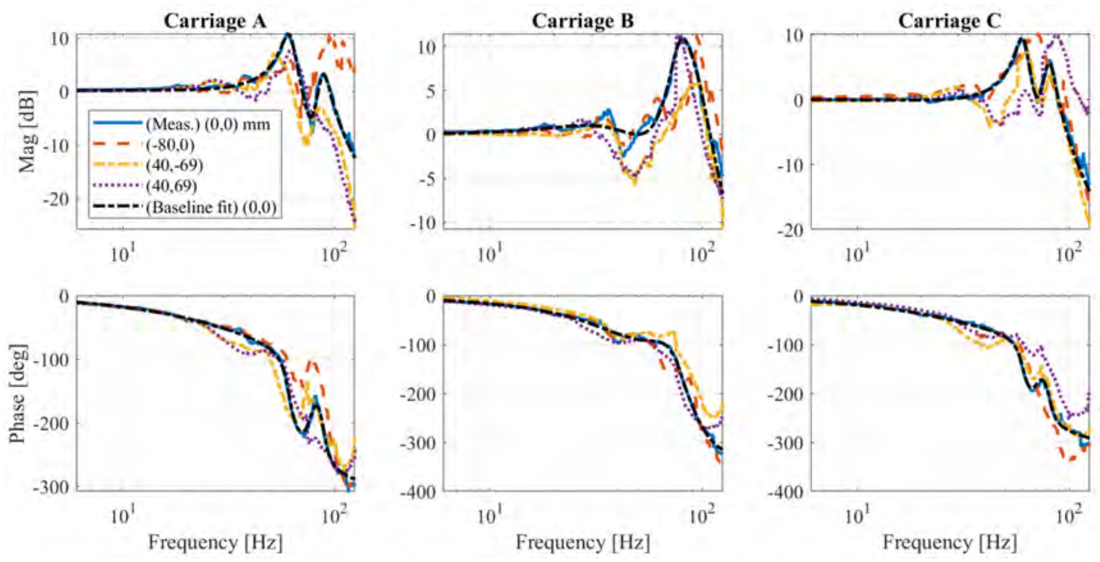


Figure 4.6: Measured frequency response functions of the carriage position dynamics at $(x, y) = (0, 0)$ mm (blue solid lines), $(-80, 0)$ mm (red dashed lines), $(40, -69)$ mm (yellow dash-dotted lines), and $(40, 69)$ mm (indigo dotted lines) of the Monoprice Delta Pro 3D printer. The black dashed lines indicate the fitted transfer functions of the baseline model (at $(x, y) = (0, 0)$ mm) that is used for the baseline controller. Note that the carriage dynamics vary most on the far side of the carriages line of action (see Fig. 3.11): carriage A's dynamics vary significantly at position $(-80, 0)$ mm, carriage B's dynamics vary significantly at position $(40, -69)$ mm, and carriage C's dynamics vary significantly at position $(40, 69)$ mm.

for real-time implementation. Successful compilation of the program means that the commands will be sent to the machine at the MicroLabBox's command frequency of 1 kHz. In other words, despite the increase in each window's computation time for the proposed method compared to the baseline, there is no difference in the production time for the same part using the either method.

To quantify the quality differences between cubes printed at different positions, we use a laser scanner—the Romer Absolute Arm from Hexagon Metrology (model #7525SI)—to scan a face from each part. We chose to scan a flat face (without letter indentations) to isolate the vibration errors. (Unfortunately, some of the fine features of the butterfly cannot be captured accurately by the laser scanner so we only use it for the cube). Figures 4.7 and 4.8 show the color map of the 12 scanned parts printed at the different positions using different compensation techniques. At all locations, the uncompensated part has some clear vibration artifacts on the left side that are largely eliminated when using FBS compensation (baseline or proposed LPV implementation). However, the baseline FBS compensation leads to larger variations of surface position when compared to the proposed controller, as evident by the standard deviation of the point cloud surface for each scanned part, which is reported in Table 4.3. (Note that the proposed controller has lower point cloud surface variance across all the measured parts, which corresponds to smoother surfaces with less vibration errors). Also in Table 4.3, note that there are some locations where the baseline FBS compensation has higher variance in the surface point cloud data when compared to the uncompensated part. This phenomenon occurs because there are some positions where the actual dynamics are sufficiently different from the baseline model, such that imposing the baseline model in the controller is worse than not compensating at all. The baseline-compensated part targeting carriage C at $(40, 69)$ mm is a great example of the poor compensation due to the model's mismatch as we see in Fig. 4.8.

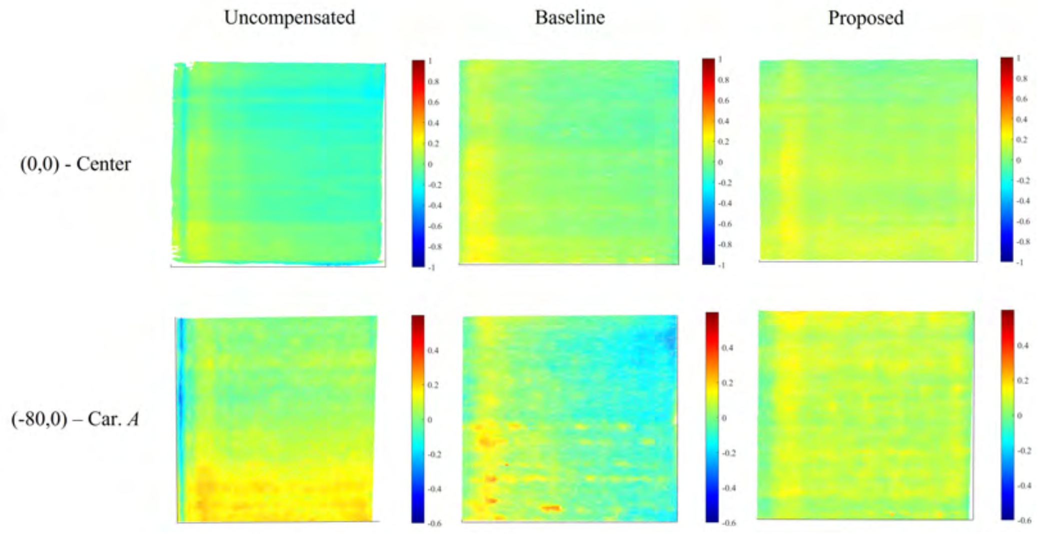


Figure 4.7: Color map of laser scanned point cloud of calibration cubes printed at $(0, 0)$ and $(-80, 0)$ mm. The scans show that the surface quality of the uncompensated and baseline FBS strategies are worse than the surface quality of the proposed strategy, especially for the part at $(-80, 0)$ mm, indicating improved surface accuracy.

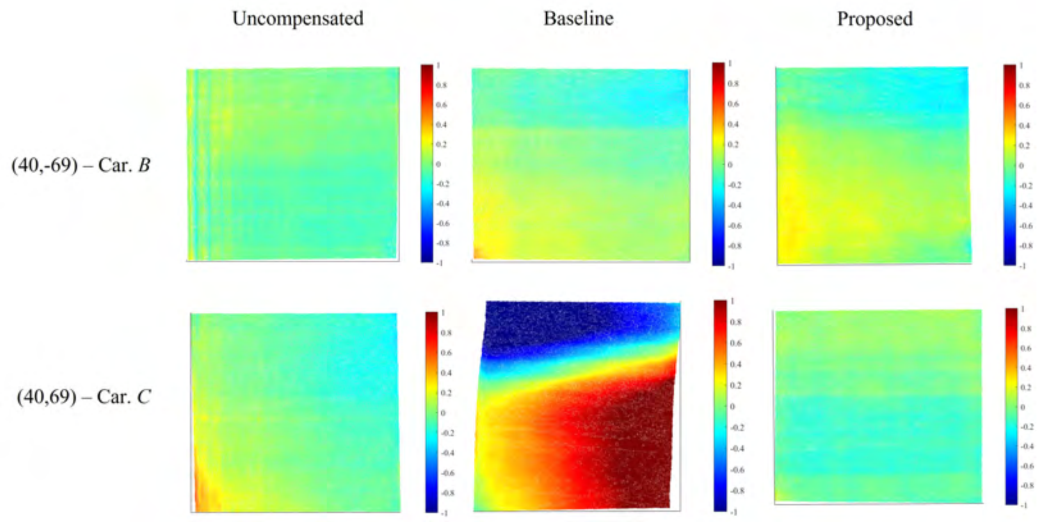


Figure 4.8: Color map of laser scanned point cloud of calibration cubes printed at $(40, -69)$ and $(40, 69)$ mm. Like Fig. 4.7, the scans show that the proposed strategy consistently improves the surface quality of the cube when compared to other compensation methods across different print locations.

Table 4.3: Standard deviation of point cloud data on the surface of scanned calibration cubes printed at various positions using different compensation strategies.

	(0,0)	(-80,0)	(40,-69)	(40,69)
Uncompensated	80 μm	72 μm	160 μm	122 μm
Baseline	74 μm	83 μm	143 μm	772 μm
Proposed	53 μm	36 μm	130 μm	72 μm

The surface variance of the baseline FBS parts can also be observed visually in Figs. 4.9 and 4.10, which show the respective images of the X and Y lettered faces of the calibration cube at all locations. For comparison, we also show the cube printed without vibration compensation. A visual inspection of the all the fabricated 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.

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. 4.6

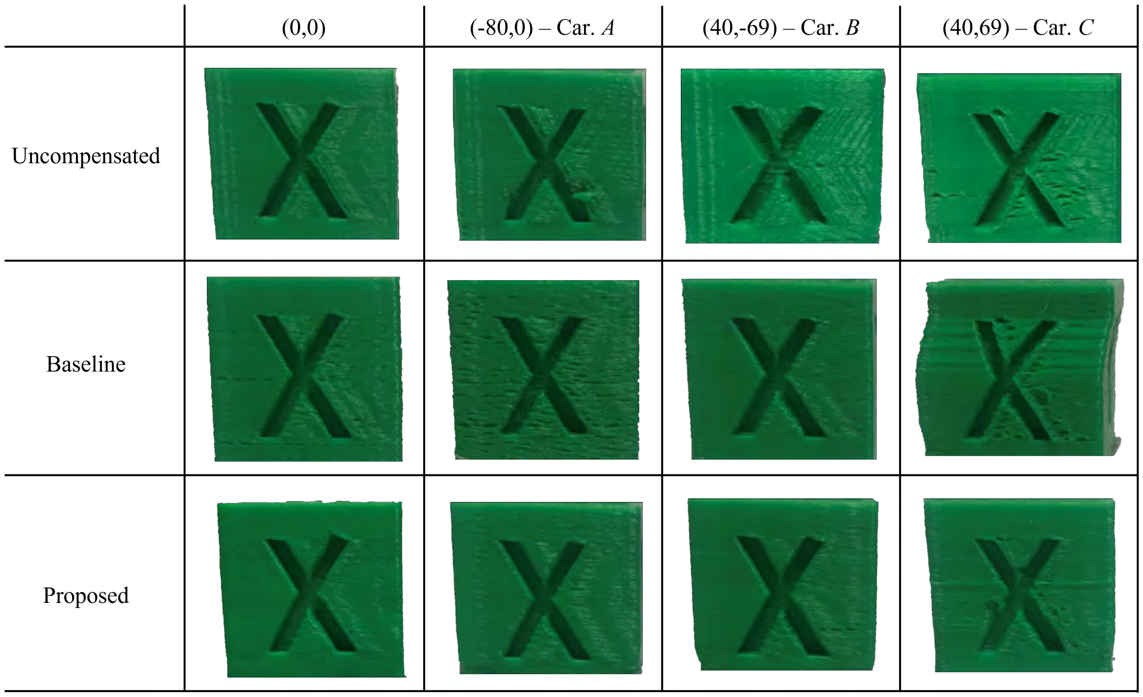


Figure 4.9: X-axis face of calibration cubes fabricated with the baseline and proposed controllers (compared to uncompensated parts) centered at different positions that target different carriages.

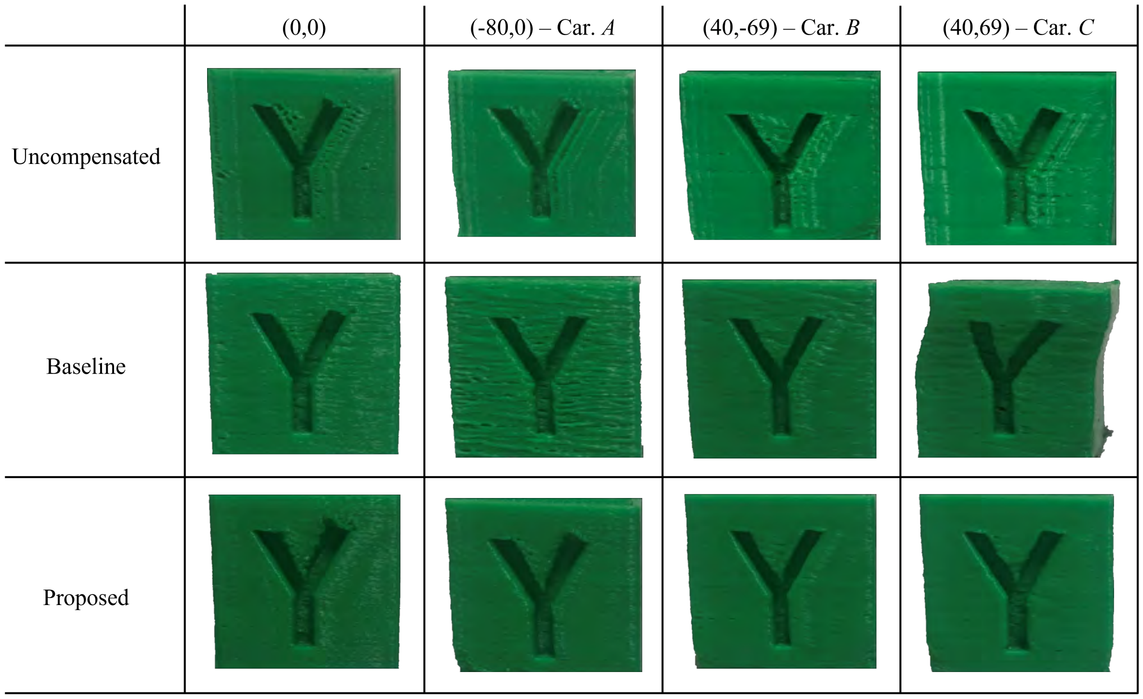


Figure 4.10: Y-axis face of calibration cubes fabricated with the baseline and proposed controllers (compared to uncompensated parts) centered at different positions that target different carriages.

Table 4.4: Root mean square (RMS) acceleration of carriages during print of the 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 4.5: Maximum acceleration of carriages during print of the 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

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 4.11 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. 4.6) 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.

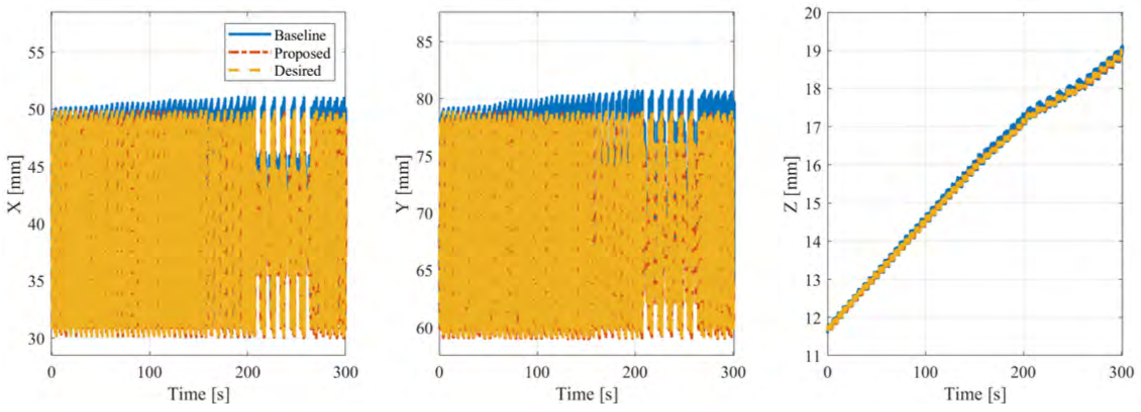


Figure 4.11: Commanded motion from the baseline (blue solid line) and proposed (red dash-dot line) controller during the drifting motion for the baseline controller while fabricating the part at $(x, y) = (40, 69)$ mm (triggering carriage C). Note that the baseline model commands increasing deviations from the nominal position (yellow dashed line), which leads to the drifting part in Figs. 4.9 and 4.10. 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.

Similar behavior is observed on the printed butterfly part. Figure 4.12 shows a comparison of the butterfly printed at $(0, 0)$ and $(40, 69)$ mm using both compensation strategies. The part that uses the baseline FBS compensation at $(40, 69)$ drifts in a similar fashion to the cube. The other butterfly parts printed at different locations do not have such stark visual differences between the baseline FBS compensation and compensation with the proposed controller as seen in Fig. 4.13. Hence, to quantify the reduction of vibration-induced acceleration, we also measure the acceleration of the carriages during each print using the vertical axis of an ADXL335 tri-axial accelerometer from Sparkfun Electronics and compare the acceleration for both controllers to the acceleration of the desired trajectories for both the cube and butterfly. Tables 4.4 and 4.5 give the RMS and maximum values of the carriage acceleration, respectively, measured over the course of a few layers 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 between the proposed controller and the baseline controller is 8.9%

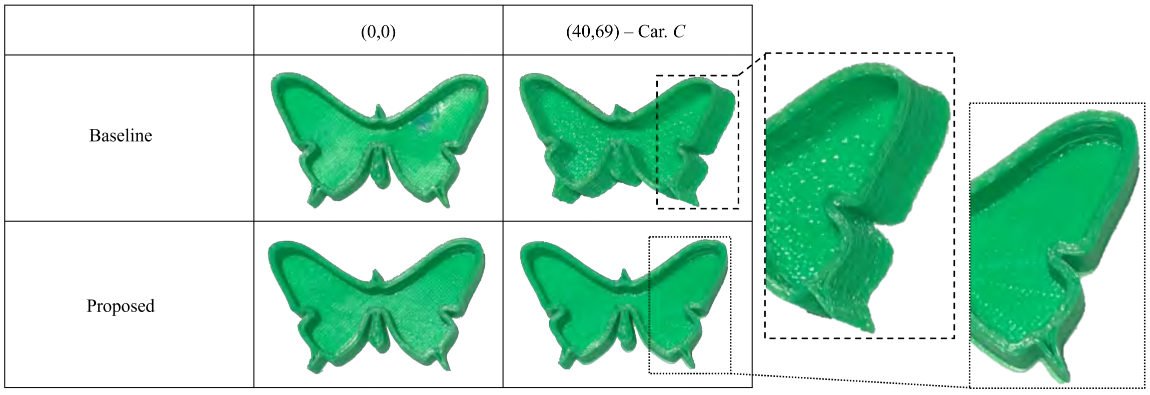


Figure 4.12: Examples of the butterfly part printed at $(0, 0)$ and $(40, 69)$ mm using the baseline FBS and proposed compensation strategies. A drift in trajectory can be seen in the part printed at $(40, 69)$ using the baseline FBS strategy, similar to the drifts seen in the calibration cube in Figs. 4.9 and 4.10.

Table 4.6: Root mean square (RMS) acceleration of carriages during print of the butterfly

	Baseline [m/s^2]	Proposed [m/s^2]	Desired [m/s^2]
$(-80, 0)$ – Car. A	4.31 (+1.38)	3.63 (+0.70)	2.93
$(40, -69)$ – Car. B	2.96 (-0.13)	2.82 (-0.27)	3.09
$(40, 69)$ – Car. C	4.29 (+1.35)	2.74 (-0.20)	2.94

for carriage C at $(40, 69)$ in RMS acceleration and 20.8% for carriage A at $(-80, 0)$ in 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 accelerations. Tables 4.6 and 4.7 give the RMS and maximum values of the carriage acceleration for a few layers of the butterfly print. The proposed method's acceleration measurements are also closer to desired acceleration in all cases for the butterfly part. The butterfly part also has the maximum difference of deviation reduction between the proposed controller and the baseline controller occurring at $(40, 69)$ for RMS acceleration and at $(-80, 0)$ for maximum acceleration. Here, the deviation is reduced by 39.1% and 39.6%, respectively.

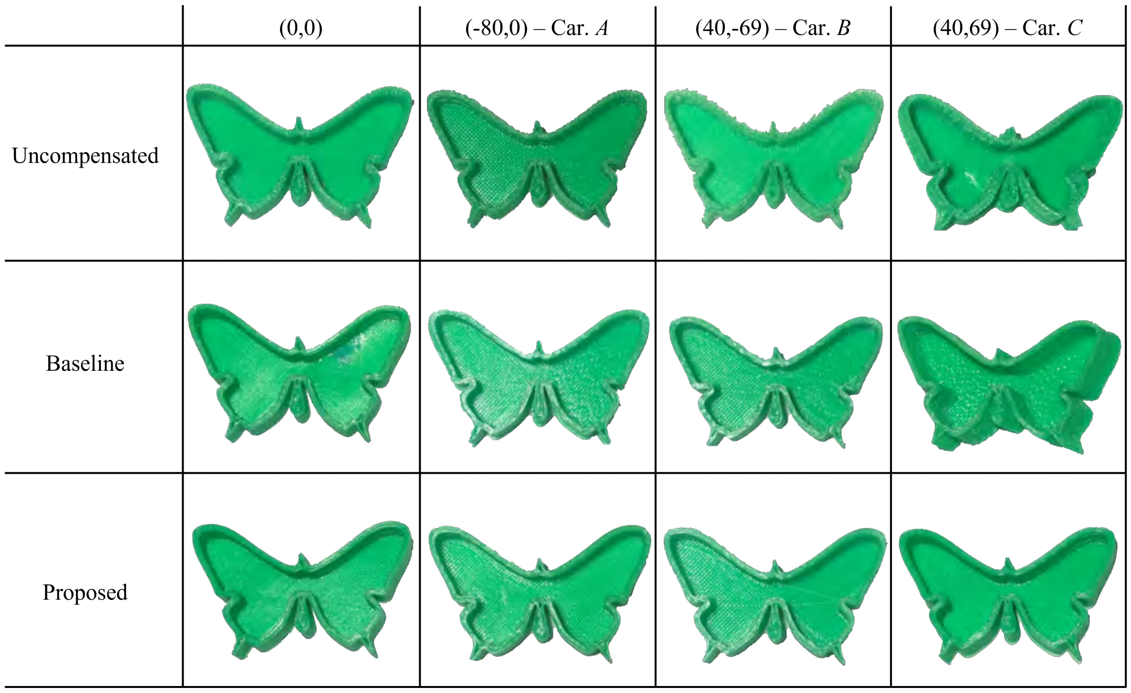


Figure 4.13: Top view of printed butterfly parts at different locations that target different carriages and with the different compensation strategies/controllers.

Table 4.7: Maximum acceleration of carriages during print of the butterfly

	Baseline [m/s^2]	Proposed [m/s^2]	Desired [m/s^2]
$(-80, 0)$ – Car. A	30.20 (+8.64)	21.47 (-0.09)	21.56
$(40, -69)$ – Car. B	19.19 (-2.39)	21.41 (-0.17)	21.58
$(40, 69)$ – Car. C	21.14 (+2.65)	19.38 (+0.89)	18.49

4.4 Summary

This chapter outlines 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 has relied on sensor measurements and feedback control as described in the literature review in Chapter I. For most delta 3D printers, feedback sensors are not available so we must use feedforward control with an accurate dynamic model. In Chapter III, we proposed a framework to efficiently identify an accurate LPV model of the delta 3D printer. In this chapter, we use that model in the FBS feedforward vibration compensation method. FBS can theoretically reduce vibration on delta 3D printers but the need to recompute the model and controller at each new position during real-time control is computationally challenging. Therefore, we propose the following techniques to decrease the computational burden: (1) parameterization and pre-filtering of portions of the model for fast online operations (Section 4.2.2), (2) computation of the model at sampled points along the trajectory while preserving continuity of the controller's predictions when the model changes (Section 4.2.3), and (3) utilization of matrix methods that yield faster matrix inversion (Section 4.2.4).

Simulations are used to assess the trade-off between computation time and accuracy. We report that the techniques presented in this thesis result in a 23x reduction in computation time from the exact parameter varying controller which re-computes the model/controller at every point. Thus, our approximations save significant computational effort while only increasing contour errors by up to about 1.3x compared to the exact controller. Printed parts from our experiments, which are photographed and scanned, also show an overall improvement in the quality of parts printed at different locations using the proposed controller compared to using a baseline controller that uses an LTI model measured at the center of the workspace. Furthermore, acceleration measurements during printing show up to 39% 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 and apply feedforward controllers like FBS to maintain accuracy during vibration-prone motion. The contributions in this chapter (and, more generally, in this thesis) bring us one step closer to the promise of high speed and high quality additive manufacturing.

CHAPTER V

Summary, Conclusions, and Future Work

5.1 Summary and Conclusions

This dissertation proposes techniques for modeling and controlling two parallel-axis manipulators for extrusion-based additive manufacturing: the H-frame gantry and delta robot. The modeling frameworks proposed are used to construct simple and efficient procedures to identify accurate parameter varying machine models. Similarly, for controlling the manipulators, we propose practical techniques to reduce the computational effort necessary for real-time control with model-based feedforward controllers, while retaining the accuracy benefits gained from using them.

Chapter II studies the H-frame gantry and introduces the model-based feedforward control approach used in this dissertation: the filtered B-splines (FBS) method. FBS is selected because of its versatility and consistency: it can be applied to any linear system and its tracking accuracy does not vary significantly based on the plant's dynamics [24]. We model the "racking" motion—parasitic torsional motion—of the H-frame using a geometric approach that relates the angular displacements of the bridge to the end-effector's position in the x and y axes. Using an H-frame 3D printer, we empirically validate the assumptions of the dynamic model, one of which is that the bridge's center of rotation is approximately the same regardless of the end-effector's position along the bridge. To accurately control the H-frame 3D printer, we formulate a coupled LPV FBS controller that captures another assumption of the model: the y-axis position of the end-effector depends on the x-axis position through the racking displacement angle, θ, but not vice versa. This insight leads to a simplification of the controller to reduce the computational effort required to invert the coupled FBS matrices: we can decouple the inversion for each axis by computing the optimal controller for the x-axis first, and use the predicted output of the controlled x-axis to find the optimal controller for the y-axis. Through analysis and simulations, we show that the so-called decoupled LPV FBS controller significantly reduces the computation time of the controller with minimal impact to the tracking and contouring accuracy of the controlled H-frame when compared to the coupled LPV FBS controller. Furthermore, we demonstrate the effectiveness of our modeling and control approach through experiments fabricating a 3D printed rectangular prism on the H-frame at high speeds. Measurements of the rectangle's width reveal a 68% improvement in RMS shape accuracy when we compare the uncompensated part to the part compensated with the decoupled LPV FBS controller. Measurements of the bridge's acceleration during printing also indicate that accelerations from racking are up to 70% larger for the uncompensated motion when compared to the racking controlled motion. The success of FBS to compensate racking on the H-frame indicates its potential to be applied to other LPV systems, including nonlinear systems that can be modeled as LPV like the delta robot.

In Chapters III and IV, we explore the modeling and control of the delta robot 3D printer, respectively. The delta 3D printer is a robotic manipulator with nonlinear, coupled, and position-dependent dynamics. In Chapter III, we focus on techniques to efficiently obtain a dynamic model of the delta printer. Since its dynamics vary with position, one would need to measure the dynamics of the printer at several locations to obtain an accurate model. To circumvent such an arduous process, we propose a modeling framework that uses the idea of receptance coupling (RC) to divide the full assembly model of the printer into models of sub-assemblies. We then identify those sub-models independently, and combine them to form the full assembly's model. Using this approach, we divide the delta model into two sub-models. We identify one of the sub-models empirically and derive an analytical expression for the second sub-model, whose parameters can be obtained with a few measurements at *one* location. Finally, we linearize the delta's nonlinear model to generate a linear parameter varying model that can be used in numerous linear model-based controllers like FBS. We show that the entire position-dependent dynamic model can be identified with five measurements at *one* location—the center of the printer—using our proposed framework. Measurements of FRFs demonstrate that our prediction model results in reasonably accurate predictions of the position-dependent dynamics of a commercial delta 3D printer, augmented with a direct drive extruder, at various positions in its workspace. To quantify the utility of the parameter-varying model, we compare it to an alternative model for model-based control where one model is measured at the center of the printer (also known as the baseline model) and used to control the entire workspace, treating the dynamics as LTI. Importantly, comparisons of the measured FRFs from a commercial delta printer show that using our identified model results less deviation from the true FRF when compared to the baseline model. Hence, we hypothesize that using a model-based controller guided by our model would result in improved accuracy performance compared to using the baseline model.

In Chapter IV, we formulate the LPV FBS controller for the delta printer to test the hypothesis stated above. However, like the computational challenges in Chapter II, the coupled model of the delta 3D printer results in large coupled LPV FBS matrices that are difficult to invert in a real-time controller. Therefore, we propose several novel techniques to reduce the computational burden of the FBS controller, including: (1) parameterizing the position-dependent portions of the dynamics offline to enable efficient computation of the model online; (2) computing 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. Using these techniques, we report a computation time reduction of up to 23x using the proposed method in simulations when compared to using the computationally expensive exact LPV model—all while maintaining high tracking accuracy. Regarding the hypothesis of improved tracking performance, we demonstrate significant quality improvements on parts printed at various positions on a commercial delta 3D printer using our controller with the LPV prediction model compared to the baseline alternative. Acceleration measurements during printing show that the improvement in print quality of the proposed controller is due to vibration reductions of up to 39% when compared to the baseline controller.

The work presented in this dissertation is based on publications [69–72]. The University of Michigan was granted a patent for the filtered B-splines (FBS) method and a company named Ulendo is licensing that patent to work with AM companies to implement controllers on their 3D printers. The company was previously working with 3D printers that can be modeled as LTI systems. Hopefully, the contributions of this dissertation will enable Ulendo to extend its scope to LPV machines.

5.2 Practical Considerations for Implementation on Commercial FFF Systems

Given the practical nature of the research presented in this dissertation, readers may wish implement its methodologies on commercial AM systems. In this section, we discuss considerations for implementing the methods on FFF systems. In the next section, we consider extensions to other AM processes.

The first step required to implement the controller is to understand the dynamic model of the machine the user wishes to control. This understanding can be obtained by an analysis of the physics of the machine or by measuring frequency response functions at several locations in the machine's workspace. Since all machines have some nonlinearity, care must be taken when using the measurement approach because differences in the measured models may be caused by nonlinearities rather than fundamental differences in the model across different positions. Therefore, several measurements should be conducted before arriving at any conclusions. If the user concludes that the machine model is LTI, then regression methods can be used to fit a transfer function model to the measured FRF (see [5]). If the model is LPV, we recommend the user study the machine to derive an analytical form of the transfer function whose parameters can be obtained from measurements, as done in Chapter III. Needless to say, the derivation of analytical models for LPV systems will be a key challenge to commercial implementation.

Implementing the FBS controller requires a reasonable understanding of Non-Uniform Rational B-Splines (or NURBS) [74], digital filtering, and linear algebra [5]. With an LTI transfer function, the user creates the B-splines matrix as described in [5], filters each column of the matrix with the transfer function, and inverts the filtered B-splines matrix offline, which is used to solve the tracking control optimization problem online, as discussed in Chapter III. On a micro-controller, the desired trajectory can be stored and broken up into smaller windows. Each window of trajectory points is (matrix) multiplied by the pseudo-inverted FBS matrix to obtain the optimal control points of the window. The parameters of the FBS matrix (like window length and knot vector spacing) can be selected using the procedure described in Chapter IV. For LPV models, the implementation is similar. One key difference is that the user may need to use a micro-controller with more memory and computation capacity to re-evaluate the model at new positions for each window of trajectory points (see Chapter IV for further discussion).

The most arduous obstacle to implementing these methods commercially is the manual process of measuring the machine models. Since there are many different 3D printer brands and many models of printers within each brand, identifying and updating these models requires a significant amount of labor, which can be expensive. Hence, the industry would greatly benefit from an automated process for identifying machine models of 3D printers. Such a process could be integrated by the 3D printer manufacturer or could be a sold as a standalone device. The components of such a system are: (1) an accelerometer to measure the machine's accelerations, (2) a trajectory command that consists of sine sweep acceleration signals at several frequencies, and (3) an executable script that analyzes the accelerometer data to generate transfer functions from commanded acceleration to measured acceleration. Since the identification process may require measuring accelerations at several locations on the printer, a standalone device with an accelerometer that can transmit data using wireless technology (e.g., bluetooth) would be most suitable. Streamlining this manual process of system identification would increase the viability of implementing the methods from this dissertation on commercial FFF machines.

5.3 Extensions to Other Additive Manufacturing Processes

FBS is a general feedforward control approach for improving the tracking accuracy of CNC machines with vibration errors. As such, FFF 3D printers are one of numerous types of machines that can, in theory, benefit from vibration compensation with FBS [20, 24, 81]. Other AM processes, like streolithography (SLA) and selective laser sintering (SLS), that use a CNC system to position an end-effector can also use FBS. In these cases, the end-effector is a device that emits a light beam—typically an ultraviolet beam for SLA and a laser beam for SLS—instead of the extruder motor and nozzle for FFF. Hence, from a practical perspective, we expect FBS to have less significant impact on SLA and SLS systems when compared to FFF systems because their end-effector inertia is much lower which leads to less vibration errors.

Implementing FBS on commercial SLA and SLS machines is also challenging due to their closed motion systems. These systems are closed to: (1) create a dark environment for the light to react with the resin or powder and (2) ensure user safety by preventing dangerous interactions with the light. In contrast with open architectures which are more common among FFF 3D printers, closed architecture implementation of FBS requires disassembling mechanical elements of the machine to identify the system's dynamics. When the system is reassembled, it is difficult to achieve the same rigidity and control performance as the original system. Hence, we recommend concurrently implementing FBS while fabricating these machines if the reader wishes to study the impact of FBS on non-FFF AM processes.

An additional consideration that will impact the performance of FBS in different AM processes is the temporal constraints of the process. For example, in FFF systems the highest reachable speed is limited by the duration of the filament extrusion and cooling process. The results presented in this dissertation indicate that speeds in the FFF process can be increased further. However, more investigation into other AM processes like SLA and SLS is necessary to reach similar conclusions and to characterize the potential for extending this research to improve their performance.

5.4 Recommendations for Future Research

This dissertation makes the research contribution of extending FBS to tracking control of linear parameter-varying (LPV) systems (and nonlinear systems modeled as LPV), building on several prior research on FBS applied to linear time-invariant (LTI) systems [5, 6, 9, 20, 24]. Our contribution presents an opportunity to use the techniques developed in this dissertation to apply FBS (or, more generally, FBF) to the numerous applications in robotics [83] and aerospace [84] where the dynamics are nonlinear. In the context of additive manufacturing, six degree-of-freedom (DOF) robotic arm manipulators, like the UR5e collaborative robot [85] have the potential to increase productivity in 3D printing. In addition to the increased speed capacity of robotic arms compared to traditional 3D printers [3], the flexibility provided by their additional DOF enable printing conformal (or curved) trajectories, which can be more efficient than layer-by-layer printing. Additionally, robot arms can eliminate or minimize the time-intensive printing of support structures needed to print overhang or sloped features [3, 86]. Despite the potential for increased productivity of robotic arms, they still suffer from accuracy limitations caused by motion-induced errors. These errors are created by vibration—via the inherent compliance in their mechanical structures—and the coupled and nonlinear dynamics between their joints/axes, which create disturbance forces that are difficult to control. Similar to delta robots, FBS can be employed to mitigate motion-induced errors to enable high-speed and high-quality parts manufactured with flexible robotic arms. The process of modeling the robotic arms first entails modeling each joint independently (either analytically or empirically) without an end-effector (e.g., a printer extruder/nozzle) attached. Note that since the robotic arm links are connected in series, one must be careful that each joint model is the sum of its independent dynamics and the dynamic effects of all the joints after it—starting from the base joint and ending at the end-effector joint. Studying the reaction torques between the joints will be an important step to transmitting the joint torques correctly. Once a reasonably accurate configuration-dependent model has been obtained for the joints, an analytical model of the inertial forces and torques of the end-effector can be derived and (its parameters) identified. Then, the Jacobian transpose matrix of the robot can be used to transform the end-effector's inertial forces/torques to torques at each joint. Note that these torques also change with changing robot configurations. Finally, using the obtained joint models, an efficient control approach can be implemented using some of the strategies from Chapter IV—e.g., parameterizing the transmitted torques from the end-effector as functions of the robot configuration, and using a single point instead of all the points each window in LPFBS.

There are a few additional areas where this dissertation can be studied further. In Chapter II, we assume that the center of rotation of the bridge on the H-frame gantry is fixed. This assumption of may not hold for end-effectors with larger mass, such as direct-drive extruders. Therefore, a useful extension of the H-frame model is one that generalizes it to include changes to the center of rotation of the bridge. Additionally, we studied several techniques to increase computational efficiency (mainly in Section 4.2) but there are myriad techniques for improving the computational and memory efficiency of computers. An exploration into methods other than the ones provided in this dissertation may lead to more efficient real-time controllers that can be implemented on (even) lower capacity micro-controllers. Along those lines, the perceptive reader will notice that the models and frameworks introduced in this dissertation are agnostic to the control approach. Hence, a systematic study of other model-based control approaches besides FBS should be undertaken to identify the best approach to leverage the methods described in this dissertation. Finally, discrepancies between the prediction dynamic model in Section 3.5 and the actual dynamics will always exist, which is the case for most systems. A robust version of FBF [6] has been studied to improve tracking control when there is a mismatch between the actual dynamics and the predictive model. Robust FBF has been applied to LTI systems but it could potentially be extended to further improve the tracking accuracy of LPV FBS controllers like the controllers designed in this dissertation.

APPENDICES

APPENDIX A

Lifted System Representation of a Digital Filter

As discussed in [24], consider digital filter p, input signal u, and output signal y defined as:

$$p = \{p_{-2}\ \ p_{-1}\ \ p_0\ \ p_1\ \ p_2\} \tag{A.1}$$

$$u = \{u_0\ \ u_1\ \ u_2\} \tag{A.2}$$

$$y = \{y_0\ \ y_1\ \ y1\} \tag{A.3}$$

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

$$y = u * p \tag{A.4}$$

From Eqs. A.1-A.4,

$$y_0 = p_0 u_0 + p_{-1} u_1 + p_{-2} u_2 \tag{A.5}$$

$$y_1 = p_1 u_0 + p_0 u_1 + p_{-1} u_2 \tag{A.6}$$

$$y_2 = p_2 u_0 + p_1 u_1 + p_0 u_2 \tag{A.7}$$

This can be expressed in matrix form as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} p_0 & p_{-1} & p_{-2} \\ p_1 & p_0 & p_{-1} \\ p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \tag{A.8}$$

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. A.8 is given by

$$p_2 z^{-2} + p_1 z^{-1} + p_0 z^0 + p_{-1} z^1 + p_{-2} z^2 \tag{A.9}$$

which is in accordance with the time-domain definition given in Eqs. A.1-A.3.

APPENDIX B

Standard Implementation of Limited-preview Filtered B-Splines Approach

This appendix presents a brief discussion of the limited preview filtered B-splines (LPFBS) approach. For more details, interested readers can refer to [5]. The LPFBS approach aims to generate optimal feedforward control inputs in sequential windows (batches) of the desired trajectory $\mathbf{x}_d$. Since $\mathbf{x}_d$ is not assumed to be fully known a priori, an un-normalized and open-ended knot vector is used and defined as

$$\bar{g}_j = \begin{cases} 0, & 0 \leq j \leq m \\ (j-m)LT_s, & j \leq m+1 \end{cases} \tag{B.1}$$

where j and m are as defined in Sec. 2.3, and $L \geq 1$ represents the uniform spacing of the knot vector elements as an integer multiple of the sampling time T_s. With the un-normalized knot vector, $N_{j,m}$ is expressed as a function of time t by replacing ξ with t and g_j with $\bar{g}_j$ in Eq. 2.10, and the function is sampled at $t_k = kT_s$ to formulate

$\mathbf{N}$ as in Eq. 2.9. The tracking problem is solved in batches as

$$\bar{\mathbf{e}} = \mathbf{x}_d - \bar{\mathbf{N}}\bar{\mathbf{p}} \Leftrightarrow \begin{bmatrix} \bar{\mathbf{e}}_{\mathrm{P}} \\ \bar{\mathbf{e}}_{\mathrm{C}} \\ \bar{\mathbf{e}}_{\mathrm{F}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{d,\mathrm{P}} \\ \mathbf{x}_{d,\mathrm{C}} \\ \mathbf{x}_{d,\mathrm{F}} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{N}}_{\mathrm{P}} & \mathbf{0} & \mathbf{0} \\ \bar{\mathbf{N}}_{\mathrm{PC}} & \bar{\mathbf{N}}_{\mathrm{C}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{N}}_{\mathrm{CF}} & \bar{\mathbf{N}}_{\mathrm{F}} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}_{\mathrm{P}} \\ \bar{\mathbf{p}}_{\mathrm{C}} \\ \bar{\mathbf{p}}_{\mathrm{F}} \end{bmatrix} \tag{B.2}$$

where subscripts P, C, and F denote the past, current, and future batches, respectively, 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. Using local least squares, the optimal coefficients of the current batch can be computed as

$$\bar{\mathbf{p}}_{\mathrm{C}}^{*} = (\bar{\mathbf{N}}_{\mathrm{C}}^{T}\bar{\mathbf{N}}_{\mathrm{C}})^{-1}\bar{\mathbf{N}}_{\mathrm{C}}\Big(\mathbf{x}_{d,\mathrm{C}} - \bar{\mathbf{N}}_{,\mathrm{PC}}\bar{\mathbf{p}}_{\mathrm{P}}\Big) \tag{B.3}$$

where $\bar{\mathbf{p}}_{\mathrm{P}}$ denotes the coefficients calculated in the last batch. Note that the information from the future batch is not considered while calculating coefficients for the current batch. Also note that the matrix $\bar{\mathbf{N}}_{\mathrm{C}}$ can be pre-inverted once and applied to all batches since the filtering model is assumed to be LTI.

The dimensions of the current window are defined by L_C and n_C, where L_C is the number of trajectory points considered in the current batch and n_C is the number of B-spline coefficients. Note that although n_C coefficents are computed, only n_{up} are updated in each window (see Fig. B.1).

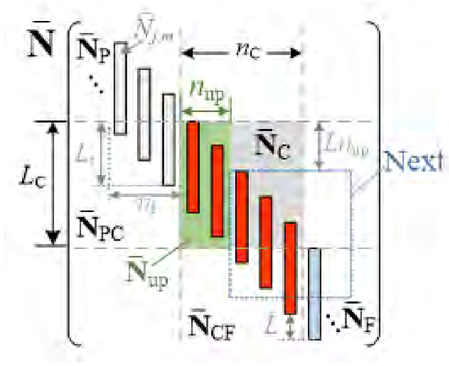

$\bar{\mathbf{N}}$
$\bar{\mathbf{N}}_{\mathrm{P}}$
$\bar{N}_{j,m}$
n_{C}
n_{up}
$\bar{\mathbf{N}}_{\mathrm{C}}$
Ln_{up}
Next
L_{C}
L_j
n_j
$\bar{\mathbf{N}}_{\mathrm{PC}}$
$\bar{\mathbf{N}}_{\mathrm{up}}$
$\bar{\mathbf{N}}_{\mathrm{CF}}$
L
$\bar{\mathbf{N}}_{\mathrm{F}}$

Figure B.1: Illustration of LPFBS.

APPENDIX C

Limited-preview Filtered B-Splines for a Coupled LPV Controller

From Eqs. (2.23) and (B.2), a natural extension of LPFBS can be made for the coupled LPV system. The tracking error is given by

$$\bar{\mathbf{e}} = \mathbf{r}_d - \bar{\mathbf{N}}_r \bar{\mathbf{p}}_r \Leftrightarrow \begin{bmatrix} \bar{\mathbf{e}}_{r,\mathrm{P}} \\ \bar{\mathbf{e}}_{r,\mathrm{C}} \\ \bar{\mathbf{e}}_{r,\mathrm{F}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{d,\mathrm{P}} \\ \mathbf{r}_{d,\mathrm{C}} \\ \mathbf{r}_{d,\mathrm{F}} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{N}}_{r,\mathrm{P}} & \mathbf{0} & \mathbf{0} \\ \bar{\mathbf{N}}_{r,\mathrm{PC}} & \bar{\mathbf{N}}_{r,\mathrm{C}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{N}}_{r,\mathrm{CF}} & \bar{\mathbf{N}}_{r,\mathrm{F}} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}_{\mathrm{P}} \\ \bar{\mathbf{p}}_{\mathrm{C}} \\ \bar{\mathbf{p}}_{\mathrm{F}} \end{bmatrix} \tag{C.1}$$

where $\mathbf{r}_d = [\mathbf{x}_d \quad \mathbf{y}_d]^T$, and the subscript r denotes matrices and vectors related to $\mathbf{r}_d$. Expanding the tracking error of the current batch as

$$\bar{\mathbf{e}}_{r,\mathrm{C}} = \begin{bmatrix} \mathbf{x}_{d,\mathrm{C}} \\ \mathbf{y}_{d,\mathrm{C}} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{N}}_{x,\mathrm{PC}} & \mathbf{0} & \bar{\mathbf{N}}_{x,\mathrm{C}} & \mathbf{0} \\ \mathbf{D}_{\mathbf{x}_{d,\mathrm{PC}}} \bar{\mathbf{N}}_{x\theta,\mathrm{PC}} & \bar{\mathbf{N}}_{y,\mathrm{PC}} & \mathbf{D}_{\mathbf{x}_{d,\mathrm{C}}} \bar{\mathbf{N}}_{x\theta,\mathrm{C}} & \bar{\mathbf{N}}_{y,\mathrm{C}} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}_{x,\mathrm{P}} \\ \bar{\mathbf{p}}_{y,\mathrm{P}} \\ \bar{\mathbf{p}}_{x,\mathrm{C}} \\ \bar{\mathbf{p}}_{y,\mathrm{C}} \end{bmatrix}, \tag{C.2}$$

the coefficients for the current batch are calculated as

$$\begin{bmatrix}\bar{\mathbf{p}}_{x,\mathrm{C}}\\ \bar{\mathbf{p}}_{y,\mathrm{C}}\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{N}}_{x,\mathrm{C}} & \mathbf{0}\\ \mathbf{D}_{\mathbf{x}_{d,\mathrm{C}}}\bar{\mathbf{N}}_{x\theta,\mathrm{C}} & \bar{\mathbf{N}}_{y,\mathrm{C}}\end{bmatrix}^{\dagger}\left(\begin{bmatrix}\mathbf{x}_{d,\mathrm{C}}\\ \mathbf{y}_{d,\mathrm{C}}\end{bmatrix} - \begin{bmatrix}\bar{\mathbf{N}}_{x,\mathrm{PC}} & \mathbf{0}\\ \mathbf{D}_{\mathbf{x}_{d,\mathrm{PC}}}\bar{\mathbf{N}}_{x\theta,\mathrm{PC}} & \bar{\mathbf{N}}_{y,\mathrm{PC}}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{p}}_{x,\mathrm{P}}\\ \bar{\mathbf{p}}_{y,\mathrm{P}}\end{bmatrix}\right), \tag{C.3}$$

The coefficients in the decoupled approximation can be calculated sequentially. First, we calculate $\bar{\mathbf{p}}_{x,\mathrm{C}}$ using $\bar{\mathbf{p}}_{\mathrm{C}}$ in Eq. (B.3) applied to the x-axis, and use the obtained coefficients to obtain $\bar{\mathbf{p}}_{y,\mathrm{C}}$ as

$$\bar{\mathbf{p}}_{y,\mathrm{C}} = \bar{\mathbf{N}}^{\dagger}_{y,\mathrm{C}}\Big(\mathbf{y}_{d,\mathrm{C}} - (\mathbf{D}_{\mathbf{x}_{d,\mathrm{PC}}}\bar{\mathbf{N}}_{x\theta,\mathrm{PC}}\bar{\mathbf{p}}_{x,\mathrm{P}} + \mathbf{D}_{\mathbf{x}_{d,\mathrm{C}}}\bar{\mathbf{N}}_{x\theta,\mathrm{C}}\bar{\mathbf{p}}_{x,\mathrm{C}}) + \bar{\mathbf{N}}_{y,\mathrm{PC}}\bar{\mathbf{p}}_{y,\mathrm{P}}\Big). \tag{C.4}$$

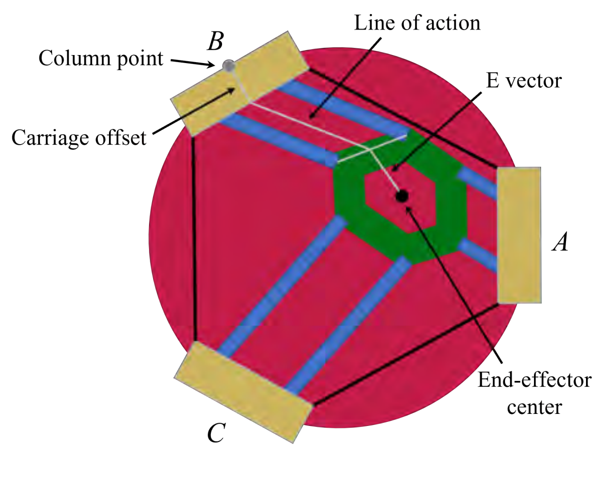
APPENDIX D

Derivation of Inverse Kinematics of Delta 3D Printer

Preliminaries

We assume that the columns form an equilateral triangle. Therefore, we can choose a coordinate system where the columns are all the same distance from the origin in the xy-plane as shown in Fig. 3.1(b) (reproduced in Fig. D.1), i.e., 120° apart on the unit circle. We denote the coordinate locations of the columns on the xy-plane by $\{A_x, A_y\}$, $\{B_x, B_y\}$, and $\{C_x, C_y\}$, corresponding to each column. The origin of the xy-plane is located at the center of the bed and the origin of the z-axis is at the top of the bed. Let the tip of the nozzle be denoted as the point $\{x, y, z\}$. We assume that the x- and y-axis location of the tip of the nozzle is at the center of the end-effector platform in the xy-plane. The z-axis location of the nozzle is at a height H_{ez} below the end-effector platform (see Fig. 3.2—reproduced in Fig. D.2).

Since all three pairs of forearms are the same length, L, we can draw a line of action for each carriage that connects the midpoint of the end-effector edge and the midpoint of the connection point on the corresponding carriage (Figs. 3.1(a) and 3.1(b)). Note that there is an offset from the center of the end-effector to where the spherical joints intersect with the line of action. We denote the end-effector offset vectors by $\vec{A}_e$, $\vec{B}_e$,



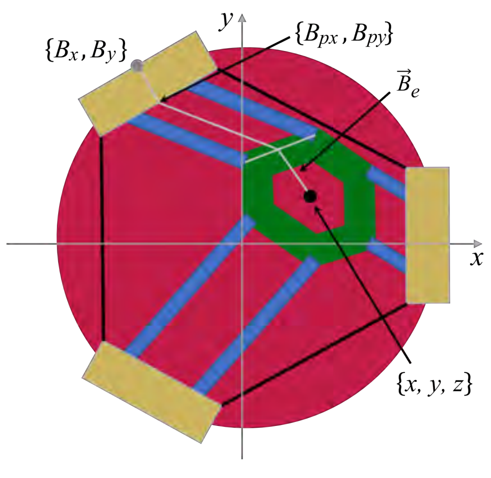


Figure D.1: (a) Overhead view of the delta printer used to define reference points and distances; (b) same overhead view as (a), but overlays the coordinate axis and key coordinates of the carriage (as exemplified for carriage B) and end-effector.

and $\vec{C}_e$ corresponding to their respective column. There is also an offset between the carriage connections and the column points. We distinguish the joint pivot locations with coordinates denoted by $\{A_{px}, A_{py}\}$, $\{B_{px}, B_{py}\}$, and $\{C_{px}, C_{py}\}$. Since the end-effector offset vectors are fixed, we can move them to the location of the carriage connection points, shifting our lines of action towards the center by $\vec{A}_e$, $\vec{B}_e$, and $\vec{C}_e$. By doing this, we create "virtual" columns whose end-effector side of the line of action is located at $\{x, y\}$. Hence, we define virtual column locations $\{A_{vx}, A_{vy}\}$, $\{B_{vx}, B_{vy}\}$, and $\{C_{vx}, C_{vy}\}$ as the original pivot location $\{A_{px}, A_{py}\}$, $\{B_{px}, B_{py}\}$, and $\{C_{px}, C_{py}\}$ shifted by the vectors $\vec{A}_e$, $\vec{B}_e$, and $\vec{C}_e$. For example, the virtual x- and y-axis position of column B is given by

$$B_{vx} = B_{px} - B_{ex} \tag{D.1}$$

$$B_{vy} = B_{py} - B_{ey} \tag{D.2}$$

Furthermore, we can separate the calculation of z-axis location from the x- and y-axis by defining a few other distances: let the height of each carriage above the end-effector platform be denoted by A_{cz}, B_{cz}, and C_{cz} and let the overall height of the carriages above the bed be denoted by A_z, B_z, and C_z as shown in Fig. 3.2. Since the height of the nozzle tip above the bed is z, we can find the z-axis location of the carriages above the bed as

$$A_z = z + A_{cz} + H_{ez}$$

$$B_z = z + B_{cz} + H_{ez}$$

$$C_z = z + C_{cz} + H_{ez}$$

Solving for z gives

$$z = A_z - A_{cz} - H_{ez} \tag{D.3}$$

$$z = B_z - B_{cz} - H_{ez} \tag{D.4}$$

$$z = C_z - C_{cz} - H_{ez} \tag{D.5}$$

Note that if we move all the carriages up or down by an equal amount, then we will change only the z-axis component of the end-effector tip. Thus, the z-axis component is directly related to the carriage heights (A_z, B_z, and C_z). Therefore, Eqs. (D.3)-(D.5) imply that A_{cz}, B_{cz}, and C_{cz} are only dependent on x and y.

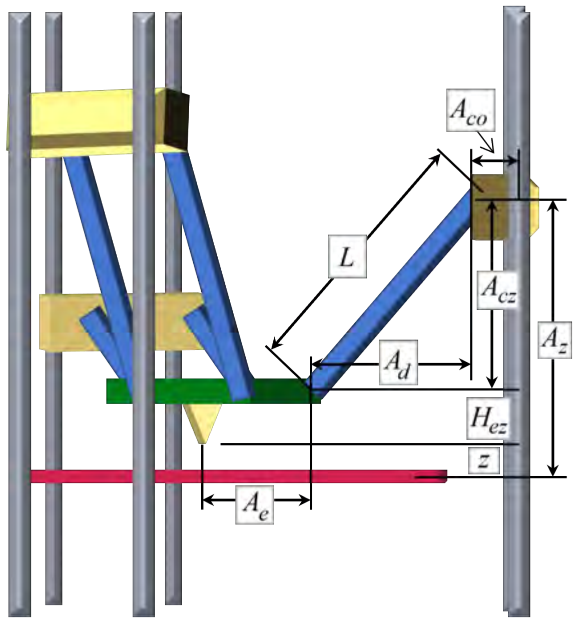


Figure D.2: Three-dimensional schematic of the delta printer showing the right triangle created by the forearms, the column, and the distance from the end-effector to the column in the xy-plane. The virtual columns used in the derivation are computed using the labeled distances.

Geometric Derivation of Inverse Kinematics

The goal of inverse kinematics is to recover the column positions A_z, B_z, and C_z from the commanded $\{x, y, z\}$ position given by G-code. Figure 3.2 shows that there is a right triangle formed by each arm, the virtual column, and the distance from the end-effector to the virtual column in the xy-plane. Let A_d, B_d, and C_d be the distance from the end-effector to the virtual columns. Using the Pythagorean theorem, we can write:

$$A_d^2 + A_{cz}^2 = L^2 \tag{D.6}$$

$$B_d^2 + B_{cz}^2 = L^2 \tag{D.7}$$

$$C_d^2 + C_{cz}^2 = L^2 \tag{D.8}$$

Now, we need to relate the coordinates of our virtual columns to x and y. We will start by determining how the end-effector's position relates to the carriage end of the line of action. Let the coordinates where the lines of action meet the end-effector platform be given by $\{A_{cx}, A_{cy}\}$, $\{B_{cx}, B_{cy}\}$, and $\{C_{cx}, C_{cy}\}$. The vectors $\vec{A}_e$, $\vec{B}_e$, and $\vec{C}_e$ can be broken into their components as $\{A_{ex}, A_{ey}\}$, $\{B_{ex}, B_{ey}\}$, and $\{C_{ex}, C_{ey}\}$. Then, we can write the $\{x, y\}$ position of the end-effector as:

$$x = A_{cx} - A_{ex} = B_{cx} - B_{ex} = C_{cx} - C_{ex} \tag{D.9}$$

$$y = A_{cy} - A_{ey} = B_{cy} - B_{ey} = C_{cy} - C_{ey} \tag{D.10}$$

By observing that the line of action travels in a spherical arc around its spherical joints, we can consider one slice which creates a circular arc. Using the formula for a circle: $(x - c_x)^2 + (y - c_y) = r^2$, where $\{c_x, c_y\}$ is the center of the circle and r is its radius, we can write, for column A, that:

$$(A_{cx} - A_{px})^2 + (A_{cy} - A_{py})^2 = A_d^2 \tag{D.11}$$

Note that A_d is the radius of the circle in the xy-plane while L is the radius of the sphere. Solving for A_{cx} and A_{cy} above, and substituting we get

$$A_{cx} = x + A_{ex} \tag{D.12}$$

$$A_{cy} = y + A_{ey} \tag{D.13}$$

$$\tag{D.14}$$

$$(x + A_{ex} - A_{px})^2 + (y + A_{ey} - A_{py})^2 = A_d^2 \tag{D.15}$$

The expression in Eq. D.15 can be replicated for each of the other columns. Finally, recall that we defined the locations of our virtual columns in terms of the the carriage coordinates and fixed end-effector vectors:

$$A_{vx} = A_{px} - A_{ex} \tag{D.16}$$

$$A_{vy} = A_{py} - A_{ey} \tag{D.17}$$

$$B_{vx} = B_{px} - B_{ex} \tag{D.18}$$

$$B_{vy} = B_{py} - B_{ey} \tag{D.19}$$

$$C_{vx} = C_{px} - C_{ex} \tag{D.20}$$

$$C_{vy} = C_{py} - C_{ey} \tag{D.21}$$

Thus, we can simplify Eq. D.15 using these expressions and write Eq. D.6 as

$$(x - A_{vx})^2 + (y - A_{vy})^2 = A_d^2 = L^2 - A_{cz}^2 \tag{D.22}$$

Solving for A_{cz} gives:

$$A_{cz}^2 = L^2 - (x - A_{vx})^2 + (y - A_{vy})^2 \tag{D.23}$$

$$A_{cz} = \sqrt{L^2 - (x - A_{vx})^2 + (y - A_{vy})^2} \tag{D.24}$$

Similarly, for columns B and C we have

$$B_{cz} = \sqrt{L^2 - (x - B_{vx})^2 + (y - B_{vy})^2} \tag{D.25}$$

$$C_{cz} = \sqrt{L^2 - (x - C_{vx})^2 + (y - C_{vy})^2} \tag{D.26}$$

The variables A_{cz}, B_{cz}, and C_{cz} are the height of each carriage above the end-effector platform. From our relation to z, we can recover the absolute z-axis location of the columns (reproduced below):

$$A_z = z + A_{cz} + H_{ez}$$

$$B_z = z + B_{cz} + H_{ez}$$

$$C_z = z + C_{cz} + H_{ez}$$

APPENDIX E

Matlab Algorithm for Forward Kinematics of Delta 3D Printer

```
function [X,Y,Z] = forward_kinematics(L,dCol,dZ)

%FORWARD_KINEMATICS Returns the current X,Y,Z position of the delta
% robot given the forearm length, x- and y-axis location of the
% columns, and the current z-axis position of the robot carriages.
%   Input(s):   L - forearm length
%               dCol - 2x3 matrix of x- & y-axis location of the
%                      columns
%               dZ - 3x1 vector of z-axis locations of the carriages

    dColP = zeros(3,3);
    for i=1:3
        dColP(1,i) = dCol(1,i);
        dColP(2,i) = dCol(2,i);
        dColP(3,i) = dZ(i);
```

```
end

% dColP has the three points on the columns.

% We establish a new coordinate system in the plane of the three

% carriage points. This system will have the origin at dColP(1,:)

% and dColP(2,:) is on the x-axis. dColP(3,:) is in the xy-plane

% with a z component of zero. We will define unit vectors in this

% coordinate system in our original coordinate system. Then, when

% we calculate the Xnew, Ynew, and Znew values, we can translate

% back to the original system by moving along those unit vectors.


% Create a vector in old coords along x-axis of new coords

p12 = dColP(:,2)-dColP(:,1);

d = norm(p12);

ex = p12/d; % make it a unit vector


% find vector from origin of the new system to the third point

p13 = dColP(:,3)-dColP(:,1);

i = dot(ex,p13); % use dot product to find x-component of p13

iex = ex*i; % create vector along the x-axis


ey = p13-iex; % only y-component of p13

j = norm(ey);

ey = ey/j; % unit vector for y


ez = cross(ex,ey); % use cross product to find z unit vector


% plug into trilateration equations
```

```
    Xnew = d/2;
    Ynew = ((i^2 + j^2)/2 - i*Xnew)/j;
    Znew = sqrt(L^2 - Xnew^2 - Ynew^2);

    % Starting from the origin of the old coords, add vectors that
    % represent the Xnew, Ynew and Znew to find the point in the old
    % system
    cartesian = dColP(:,1) + ex*Xnew;
    cartesian = cartesian + ey*Ynew;
    cartesian = cartesian + ez*(-Znew);

    X = cartesian(1);
    Y = cartesian(2);
    Z = cartesian(3);
end

% --- Calling the function ---
time = 0:0.001:1; % 1 second with sampling time of 1 ms
Az, Bz, Cz = [vector of carriage positions]; % Do NOT run this line
x, y, z = zeros(length(time),1); % Do NOT run this line
dCol = [Avx, Bvx, Cvx;
        Avy, Bvy, Cvy];

for i = 1:length(time)
    dZ = [Az(i);Bz(i);Cz(i)];
    [x(i),y(i),z(i)] = forward_kinematics(L,dCol,dZ);
end
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Sculpteo, Inc., *The state of 3D printing: the data you need to understand the 3D printing world and build your 3D printing strategy.* San Francisco, CA, 2021.

[2] A. Essop, "Boeing 777X: GE9X engines with 300 printed parts powers largest twin-engine jetliner in first flight," 2021. https://3dprintingindustry.com/news/boeing-77x-ge9x-engines-with-300-3d-printed-parts-powers-largest-twin-engine-jetliner-in-first-flight-167793/ (accessed: March 29, 2023).

[3] P. Urhal, A. Weightman, C. Diver, and P. Bartolo, "Robot assisted additive manufacturing: A review," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 335–345, 2019. https://doi.org/10.1016/j.rcim.2019.05.005.

[4] C. Okwudire and H. Madhyastha, "Distributed manufacturing for and by the masses," *Science*, vol. 372, pp. 341–342, 2021.

[5] M. Duan, D. Yoon, and C. 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. https://doi.org/10.1016/j.mechatronics.2017.09.002.

[6] K. Ramani, N. Edoimioya, and C. 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, pp. 2556–2564, 2020. https://doi.org/10.1109/TMECH.2020.2983680.

[7] H. Kim and C. Okwudire, "Simultaneous servo error pre-compensation and feedrate optimization with tolerance constraints using linear programming," *Int. J. of Adv. Manufac. Technol.*, vol. 109, pp. 809–821, 2020. https://doi.org/10.1007/s00170-020-05651-w.

[8] P. Wu, K. Ramani, and C. Okwudire, "Accurate linear and nonlinear model-based feedforward deposition control for material extrusion additive manufacturing," *Additive Manuf.*, vol. 48, p. 102389, 2021. https://doi.org/10.1016/j.addma.2021.102389.

[9] C. 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), pp. 809–821, 2018. https://doi.org/10.3390/inventions3030056.

[10] 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, pp. 1323–1345, 2001. https://doi.org/10.1016/j.ijmachtools.2022.103862.

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

[12] C. Chen and A. Lee, "Design of acceleration/deceleration profiles in motion control based on digital FIR filters," *Int. J. Mach. Tools Manuf.*, vol. 38, pp. 779–825, 1998. https://doi.org/10.1016/S0890-6955(97)00065-5.

[13] J. Chen, S. Yeh, and J. Sun, "An S-curve acceleration/deceleration design for CNC machine tools using quintic feedrate function," *Compt.-Aided Des. Appl.*, vol. 8, pp. 583–592, 2011. https://doi.org/10.3722/cadaps.2011.583-592.

[14] N. Singer, W. Singhose, and W. Seering, "Comparison of filtering methods for reducing residual vibration," *Eur J. Control*, vol. 5, pp. 208–218, 2011. https://doi.org/10.1016/S0947-3580(99)70155-X.

[15] P. Barre, R. Bearee, P. Borne, and E. Dumetz, "Influence of a jerk controlled movement law on the vibratory behavior of high-dynamics systems," *J. Intel. Rob. Syst.*, vol. 42, pp. 275–293, 2005. https://doi.org/10.1007/s10846-004-4002-7.

[16] N. Singer and W. Seering, "Design and comparison of command shaping methods for controlling residual vibration," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 888–893, 1989. https://doi.org/10.1109/ROBOT.1989.100094.

[17] W. Singhose, "Command shaping for flexible systems: a review of the first 50 years," *Int. J. Precis. Eng. Manuf.*, vol. 10, pp. 153–168, 2009. https://doi.org/10.1007/s12541-009-0084-2.

[18] B. Sencer, A. Dumanli, and Y. Yamada, "Spline interpolation with optimal frequency spectrum for vibration avoidance," *CIRP Ann. Manuf. Technol.*, vol. 67, pp. 377–380, 2018. https://doi.org/10.1016/j.cirp.2018.03.002.

[19] Y. Altintas and M. Khoshdarregi, "Contour error control of CNC machine tools with vibration avoidance," *CIRP Ann. Manuf. Technol.*, vol. 61, pp. 335–338, 2012. https://doi.org/10.1016/j.cirp.2012.03.132.

[20] C. Okwudire, K. Ramani, and M. Duan, "A trajectory optimization method for improved tracking of motion commands using CNC machines that experience unwanted vibration," *CIRP Ann. Manuf. Technol.*, vol. 65, pp. 373–376, 2016. https://doi.org/10.1016/j.cirp.2016.04.100.

[21] Y. Liu, Y. Cao, L. Sun, and X. Zheng, "Vibration suppression for wafer trans-

fer robot during trajectory tracking," *Proc. IEEE Int. Conf. Mechatr. Autom.*, pp. 741–756, 2010. https://doi.org/10.1109/ICMA.2010.5589042.

[22] Y. Zhao, W. Chen, T. Tang, and M. Tomizuka, "Zero time delay input shaping for smooth settling of industrial robots," *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, pp. 620–625, 2016. https://doi.org/10.1109/COASE.2016.7743459.

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

[24] K. Ramani, M. Duan, C. Okwudire, and A. Ulsoy, "Tracking control of linear time-invariant nonminimum phase systems using filtered basis functions," *J. Dyn. Syst. Meas. Control*, vol. 139, p. 011001, 2017. https://doi.org/10.1016/j.cirp.2016.04.100.

[25] B. Rigney, L. Pao, and D. Lawrence, "Nonminimum phase dynamic inversion for settle time applications," *IEEE Trans. Control Syst. Technol.*, vol. 17, pp. 989–1005, 2009. https://doi.org/10.1109/TCST.2008.2002035.

[26] G. Clayton, S. Tien, 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, p. 061101, 2009. https://doi.org/10.1115/1.4000158.

[27] K. Sollmann, M. Jouaneh, and D. Lavender, "Dynamic modeling of a two-axis, parallel, H-frame-type XY positioning system," *IEEE/ASME Trans. Mechatronics*, vol. 15, pp. 280–290, 2010. https://doi.org/10.1109/TMECH.2009.2020823.

[28] A. Avdeev, A. Shvets, and I. Torubarov, "Investigation of kinematics of 3D printer print head moving systems," *Proc. 5th Int. Conf. Industrial Eng.*, pp. 461–471, 2020. https://doi.org/10.1007/9783-03022041-9_50.

[29] "Stratasys inc. 3d printers." http://www.stratasys.com/3d-printers (accessed: March 29, 2023).

[30] "Makerbot 3d printers." https://www.makerbot.com/3d-printers/ (accessed: March 29, 2023).

[31] All3DP, "Creality ender 4." https://all3dp.com/1/creality-ender-4-3d-printer-review/ (accessed: March 29, 2023).

[32] J. Go and A. Hart, "Fast desktop-scale extrusion additive manufacturing," *Additive Manuf.*, vol. 18, pp. 276–284, 2017. https://doi.org/10.1016/j.addma.2017.10.016.

[33] Stevens, A.G., *High throughput extrusion additive manufacturing–rate limits and system design*. PhD thesis, Massachusetts Institute of Technology, 2021. https://dspace.mit.edu/handle/1721.1/139957.

[34] K. Sollmann, *Modeling, Simulation and Control of a Belt Driven, Parallel H-Frame Type Two Axes Positioning System*. PhD thesis, University of Rhode Island, 2007. https://digitalcommons.uri.edu/theses/607/.

[35] S. Weikert, R. Ratnaweera, O. Zirn, and K. Wegener, "Modeling and measurement of H-Bot kinematic systems," *Proc. ASPE Annual Meeting 2011*, 2011.

[36] J. Brinker and B. Corves, "A survey on parallel robots with delta-like architecture," *Proc. 14th IFToMM World Congress*, pp. 407–414, 2015. https://doi.org/10.6567/IFTOMM.14TH.WC.PS13.003.

[37] K. Miller, "Experimental verification of modeling of delta robot dynamics by direct application of Hamilton's principle," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 532–537, 1995. https://doi.org/10.1109/ROBOT.1995.525338.

[38] M. Isa and I. Lazoglu, "Five-axis additive manufacturing of freeform models through buildup of transition layers," *J. Manufac. Syst.*, vol. 50, pp. 69–80, 2019. https://doi.org/10.1016/j.jmsy.2018.12.002.

[39] R. Allen and R. Trask, "An experimental demonstration of effective Curved Layer Fused Filament Fabrication utilizing a parallel deposition robot," *Additive Manuf.*, vol. 8, pp. 78–87, 2015. https://doi.org/10.1016/j.addma.2015.09.001.

[40] "Monoprice 3d printers." https://www.monoprice.com/ (accessed: March 29, 2023).

[41] "Flsun 3d printers." https://www.flsun3d.com/ (accessed: March 29, 2023).

[42] "World advanced saving project (wasp)." https://www.3dwasp.com/en/ (accessed: March 29, 2023).

[43] "Tractus 3d." https://www.amtech3d.com/tractus3d (accessed: March 29, 2023).

[44] O. Zakharov, K. Pugin, and T. Ivanova, "Modeling and analysis of delta kinematics FDM printer," *J. Physics: Conf. Series*, vol. 2182, p. 012069, 2022. https://doi.org/10.1088/1742-6596/2182/1/012069.

[45] A. Codourey, "Dynamic modeling of parallel robots for computed-torque control implementation," *Int. J. Robot. Research*, vol. 17, pp. 1325–1336, 1998. https://doi.org/10.1177/027836499801701205.

[46] K. Miller, "Optimal Design and Modeling of Spatial Parallel Manipulators," *Int. J. Robot. Research*, vol. 23, pp. 127–140, 2004. https://doi.org/10.1177/0278364904041322.

[47] G. Lebret, K. Liu, and F. Lewis, "Dynamic analysis and control of a stewart

platform manipulator," *J. Robot. Syst.*, vol. 10, pp. 629–655, 1993. https://doi.org/10.1002/rob.4620100506.

[48] H. Pang and M. Shahinpoor, "Inverse dynamics of a parallel manipulator," *J. Robot. Syst.*, vol. 11, pp. 693–702, 1993. https://doi.org/10.1002/rob.4620110803.

[49] C.-D. Zhang and S.-M. Song, "An efficient method for inverse dynamics of manipulators based on the virtual work principle," *J. Robot. Syst.*, vol. 10, pp. 605–627, 1993. https://doi.org/10.1002/rob.4620100505.

[50] M. Ramirez-Neria, W. Sira-Ramírez, A. Luviano-Juárez, and A. Rodriguez-Ángeles, "Active disturbance rejection control applied to a delta parallel robot in trajectory tracking tasks," *Asian J. Control*, vol. 17, pp. 636–647, 2015. https://doi.org/10.1109/ACC.2012.6314934.

[51] L. 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, pp. 1387–1398, 2015. https://doi.org/10.1109/TCST.2014.2367313.

[52] Q. Zhou, W. Panfeng, and M. Jiangping, "Controller parameter tuning of delta robot based on servo identification," *Chinese J. Mech. Eng.*, vol. 28, pp. 267–275, 2015. https://doi.org/10.3901/CJME.2014.1117.169.

[53] 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.*, pp. 2249–2254, 2004. https://doi.org/10.1109/ROBOT.2004.1307396.

[54] L. Angel and J. Viola, "Fractional order PID for tracking control of a parallel robotic manipulator type delta," *ISA Trans.*, vol. 79, pp. 172–188, 2018. https://doi.org/10.1016/j.isatra.2018.04.010.

[55] C. Boudjedir, D. Boukhetala, and M. Bouri, “Nonlinear PD plus sliding mode control with application to a parallel delta robot,” *J. Electr. Eng.*, vol. 69, pp. 329–336, 2018. https://doi.org/10.2478/jee-2018-0048.

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

[57] Y. Su, D. Sun, L. Ren, and J. Mills, “Integration of saturated PI synchronous control and PD feedback for control of parallel manipulators,” *IEEE Trans. Robot.*, vol. 22, pp. 202–207, 2006. https://doi.org/10.1109/TRO.2005.858852.

[58] P. Chiacchio, F. Pierrot, L. Sciavicco, and B. Siciliano, “An intelligent compensation through B-spline neural network for a delta parallel robot,” *IEEE Trans. Ind. Electron.*, vol. 40, pp. 393–403, 1993. https://doi.org/10.1109/41.232228.

[59] P.-C. Pham and Y.-L. Kuo, “Robust adaptive finite-time synergetic tracking control of delta robot based on radial basis function neural networks,” *Appl. Sci.*, vol. 12, p. 10861, 2022. https://doi.org/10.3390/app122110861.

[60] P. Bègon, F. Pierrot, and P. Dauchez, “Fuzzy sliding mode control of a fast parallel robot,” *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 1178–1183, 1995. https://doi.org/10.1109/ROBOT.1995.525440.

[61] W. Yang and W. Liu, “Delta robot trajectory tracking based on fuzzy adaptive sliding mode controller,” *Proc. IEEE China Autom. Cong. (CAC)*, pp. 7041–7046, 2021. https://doi.org/10.1109/CAC53003.2021.9727620.

[62] F. Azad, S. Rahimi, M. Yazdi, and M. Masouleh, “Design and evaluation of adaptive and sliding mode control for a 3-DOF Delta parallel robot,” *Proc. IEEE*

28th Iranian Conf. Elec. Eng., pp. 1–7, 2020. https://doi.org/10.1109/ICEE50131.2020.9261040.

[63] R. Voorhoeve, R. de Rozario, W. Aangenent, and T. Oomen, “Identifying position-dependent mechanical systems: A modal approach applied to a flexible wafer stage,” *IEEE Trans. Control Syst. Tech.*, vol. 29, pp. 194–206, 2021. https://doi.org/10.1109/TCST.2020.2974140.

[64] Y.-P. Liu and Y. Altintas, “Predicting the position-dependent dynamics of machine tools using progressive network,” *Precision Eng.*, vol. 73, pp. 409–422, 2022. https://doi.org/10.1016/j.precisioneng.2021.10.010.

[65] H. Patiño, R. Carelli, and B. Kuchen, “Neural networks for advanced control of robot manipulators,” *IEEE Trans. Neural Networks*, vol. 13, pp. 343–354, 2002. https://doi.org/10.1109/72.991420.

[66] R. Carelli, E. Camacho, and H. Patiño, “A neural network based feedforward adaptive controller for robots,” *IEEE Trans. Syst. Man. Cybernetics*, vol. 25, pp. 1281–1288, 1995. https://doi.org/10.1109/21.400506.

[67] L. Cheng, Z.-G. Hou, and M. Tan, “Adaptive neural network tracking control for manipulators with uncertain kinematics, dynamics and actuator model,” *Automatica*, vol. 45, pp. 2312–2318, 2009. https://doi.org/10.1016/j.automatica.2009.06.007.

[68] S. Rahimi, H. Jalali, M. Yazdi, A. Kalhor, and M. Masouleh, “Design and practical implementation of a neural network self-tuned inverse dynamic controller for a 3-DoF delta parallel robot based on arc length function for smooth trajectory tracking,” *Mechatonics*, vol. 84, p. 102772, 2022. https://doi.org/10.1016/j.mechatronics.2022.102772.

[69] N. Edoimioya, K. Ramani, and C. Okwudire, “Software compensation of undesirable racking motion of H-frame 3D printers using filtered B-splines,” *Additive Manufac.*, vol. 45, p. 102290, 2021. https://doi.org/10.1016/j.addma.2021.102290.

[70] N. Edoimioya and C. Okwudire, “An efficient control-oriented modeling approach for vibration-prone delta 3D printers using receptance coupling,” *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, pp. 165–170, 2021. https://doi.org/10.1109/CASE49439.2021.9551537.

[71] N. Edoimioya and C. Okwudire, “A generalized and efficient control-oriented modeling approach for vibration-prone delta 3D printers using receptance coupling,” *Trans. Autom. Sci. Eng.*, 2022. https://doi.org/10.1109/TASE.2022.3197057.

[72] N. Edoimioya, C. Chou, and C. Okwudire, “Vibration compensation of delta 3D printer with position-varying dynamics using filtered B-splines,” *Int. J. Adv. Manufac. Technol.*, 2022. https://doi.org/10.1007/s00170-022-10789-w.

[73] G. Phanomchoeng and R. Chancharoen, “Adaptive gain control for a two-axis, H-Frame-type, positioning system,” *Eng. J.*, vol. 21, pp. 223–234, 2017. https://doi.org/10.4186/ej.2017.21.3.223.

[74] L. Piegl and W. Tiller, *The NURBS Book*. Springer, Berlin, Heildelberg, 1995.

[75] G. Golub and C. Van Loan, *Matrix Computations, fourth edition*. The Johns Hopkins University Press, Baltimore, Maryland, 2013.

[76] T. Schmitz and K. Smith, *Mechanical Vibrations*. Springer, Boston, MA, 2012. https://doi.org/10.1007/978-1-4614-0460-6_9.

[77] S. Park and Y. Altintas, “Receptance coupling for end mills,” *Int. J. Mach. Tools Manuf.*, vol. 43, pp. 889–896, 2003. https://doi.org/10.1016/S0890-6955(03)00088-9.

[78] M. Law and S. Ihlenfeldt, “A frequency-based substructuring approach to efficiently model position-dependent dynamics in machine tools,” *Proc. Institution Mech. Eng., Part K: J. Multi-body Dynamics*, vol. 229, pp. 304–317, 2015. https://doi.org/10.1177/1464419314562264.

[79] Z. Ji, “Study of the effect of leg inertia in Stewart platforms,” *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 121–126, 1993. https://doi.org/10.1109/ROBOT.1993.291971.

[80] A. Moetazedian, A. Budisuharto, V. Silberschmidt, and A. Gleadall, “CONVEX (CONtinuously Varied EXtrusion): a new scale of design for additive manufacturing,” *Additive Manufac.*, vol. 37, p. 101576, 2021. https://doi.org/10.1016/j.addma.2020.101576.

[81] 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.

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

[83] A. De Luca and B. Siciliano, “Inversion-Based Nonlinear Control of Robot Arms with Flexible Links,” *J. Guid. Cont. Dyn.*, vol. 16, pp. 1169–1176, 1993. https://doi.org/10.2514/3.21142.

[84] J. Hauser, S. Sastry, and G. Meyer, “Nonlinear Control Design for Slightly Non-Minimum Phase systems: Application to V/STOL Aircraft,” *Automatica*, vol. 28, pp. 665–679, 1992. https://doi.org/10.1016/0005-1098(92)90029-F.

[85] “Universal robots ur5e.” https://www.universal-robots.com/products/ur5-robot/ (accessed: March 29, 2023).

[86] P. Bhatt, R. Malhan, A. Shembekar, Y.-J. Yoon, and S. Gupta, “Expanding capabilities of additive manufacturing through use of robotics technologies: A survey,” *Additive Manuf.*, vol. 31, p. 100933, 2020. https://doi.org/10.1016/j.addma.2019.100933.